

Bölüm 6: Dinamik Programlama

Algoritmalar

Örtüşen alt problemler

Bellekleme ve tablolama

Fibonacci

0/1 ve sınırsız sırt çantası

Alt küme toplamı

En uzun artan alt dizi

En uzun ortak alt dizgi

En uzun ortak alt dizi

Sercan KÜLCÜ · Giresun Üniversitesi · Bilgisayar Mühendisliği

GİRİŞ

Öğrenme Çıktıları

Tanıma

Bir problemde **örtüşen alt problemleri** ve **optimal alt yapıyı** fark etmek; dinamik programlamanın ne zaman uygun olduğuna karar vermek.

Formülleme

Alt problemi (durumu) tanımlamak, özyineleme bağıntısını ve taban durumlarını yazmak, tablo doldurma sırasını belirlemek.

Uygulama

Fibonacci, 0/1 ve sınırsız sırt çantası, alt küme toplamı, LIS, en uzun ortak alt dizgi ve LCS tablolarını elle doldurmak ve Java ile yazmak.

Analiz

Bellekleme ile tablolamayı karşılaştırmak; $O(n \cdot W)$, $O(m \cdot n)$, $O(n^2)$, $O(n \log n)$ karmaşıklıklarını gerekçelendirmek; tablodan çözümü geri okumak.

Dinamik Programlama (Dynamic Programming) Nedir?

- Bazen karmaşık problemleri çözmek için **akıllı yöntemlere** ihtiyaç olur; dinamik programlama bunlardan biridir.
- Büyük problemi **küçük, tekrarlayan alt problemlere** böler ve adım adım çözer.
- Her alt problemin çözümü bir **tabloya** yazılır; aynı alt problemle yeniden karşılaşıldığında çözüm **tablodan okunur**, böylece zaman kazanılır.
- Özellikle **optimizasyon problemlerinde** (en büyük değer, en kısa yol, en az işlem) sıkça kullanılır.

Kısaca

DP = özyineleme + **hatırlama**. Her dinamik programlama problemi düz özyineleme ile de çözülebilir; DP, aynı işi **iki kez yapmamayı** sağlar.

Ad, Richard Bellman'dan (1950'ler) gelir; "programlama" burada kod yazmak değil, **tablo ile planlama** anlamındadır.

DP İçin İki Temel Özellik

1. Örtüşen alt problemler (overlapping subproblems)

Özyinelemeli çözüm **aynı alt problemi tekrar tekrar** çağırır. Örnek: fib(5) hesaplanırken fib(3) iki kez, fib(2) üç kez hesaplanır.

Alt problemler birbirinden bağımsızsa (birleştirme sıralaması gibi) tablo işe yaramaz; bu **böl ve fethettir**.

2. Optimal alt yapı (optimal substructure)

Problemin optimal çözümü, alt problemlerin **optimal çözümlerinden** kurulabilir.

Bellman'ın optimallik ilkesi: optimal bir çözümün içerdiği her alt çözüm de kendi alt problemi için optimaldir.

Bellman denklemi örneği (en büyük toplamlı alt dizi): $V(i) = i$ 'de biten en büyük toplam ise $V(i) = \max(a[i], V(i-1) + a[i])$. Değer, bir önceki alt problemin **optimal** değerinden hesaplanır.

Bir DP Çözümü Tasarlamanın Adımları

1. **Optimal çözümün yapısını** anla: hangi alt problemler çözülmeli?
2. **Durumu ve bağıntıyı** yaz: çözümü daha küçük alt problemler cinsinden ifade et (özyineleme bağıntısı + taban durumları).
3. **Hesaplama sırasını** belirle: bir hücre hesaplanırken ihtiyaç duyduğu hücreler hazır olmalı.
4. **Tablonun boyutunu** belirle: kaç boyut, her boyut kaç değer? (ör. $(n+1) \times (W+1)$)
5. **Tabloyu doldur**: aşağıdan yukarı (tablolama) ya da yukarıdan aşağı (bellekleme).
6. **Çözümü geri oku**: yalnızca değeri değil, seçilen öğeleri de tablodan geriye doğru okuyarak bul.

Toplam süre $\approx$ (durum sayısı) $\times$ (bir durumun hesaplanma süresi). Bu kural bölümdeki tüm karmaşıklıkları açıklar.

Bellekleme (Bellekleme) ve Tablolama (Tabulation)

Yukarıdan aşağı: Bellekleme

- Özyinelemeli çözüm aynen yazılır; sonuçlar bir **önbellekte** (dizi / HashMap) saklanır.
- Çağrıdan önce tabloya bakılır: varsa döndürülür, yoksa hesaplanıp yazılır.
- Yalnızca **gerçekten ihtiyaç duyulan** alt problemler çözülür.
- Derin özyinelemede yığın taşması (StackOverflowError) riski.

Aşağıdan yukarı: Tablolama

- En küçük alt problemten başlanır, tablo **döngülerle** sırayla doldurulur.
- Özyineleme yok, çağrı maliyeti yok; genellikle daha hızlıdır.
- Doldurma sırası önceden doğru belirlenmelidir.
- Yalnızca önceki satır gerekiyorsa **bellek azaltılabilir** (ör. $O(W)$).

İkisi de her alt problemi **bir kez** çözer; asimptotik süre aynıdır. Fark, sıra ve uygulama biçimindedir.

DP, Böl ve Fethet ve Açgözlü Yaklaşım

	Böl ve Fethet	Dinamik Programlama	Açgözlü (Greedy)
Alt problemler	bağımsız	örtüşen	her adımda tek seçim
Seçim	yok, böl ve birleştir	tüm seçenekler denenir, en iyisi tabloya	o anki en iyi seçilir, geri dönülmez
Optimallik	—	optimal alt yapı varsa garanti	yalnızca açgözlü seçim özelliği varsa
Örnek	birleştirme sıralaması, ikili arama	0/1 sırt çantası, LCS, Bellman-Ford	kesirli sırt çantası, Dijkstra, Kruskal
Bellek	özyineleme yığını	tablo (çoğu zaman $O(n \cdot W)$ ya da $O(m \cdot n)$)	genellikle $O(1)$ ek bellek

Dinamik Programlama Uygulamaları

En kısa yol

GPS cihazları, haritalar:
Bellman-Ford, Floyd-Warshall.

Dizgi karşılaştırma

diff, yazım denetimi: LCS, düzenleme uzaklığı (edit distance).

Biyoinformatik

DNA/protein hizalama (Needleman-Wunsch, Smith-Waterman), RNA yapısının belirlenmesi.

Kaynak dağıtımı

Sırt çantası, bütçe ve yatırım planlama, üretim hattı çizelgeleme.

Derleyiciler ve sayısal hesap

Matris zinciri çarpımı, optimal ikili arama ağacı, ayrıştırma (CYK).

Yapay zekâ

Satranç oynayan bilgisayarlar, makine çevirisi, pekiştirmeli öğrenme (Bellman denklemi).

Fibonacci: Düz Özyineleme

Fibonacci dizisi: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34,

...

$$F(0) = 0, F(1) = 1$$

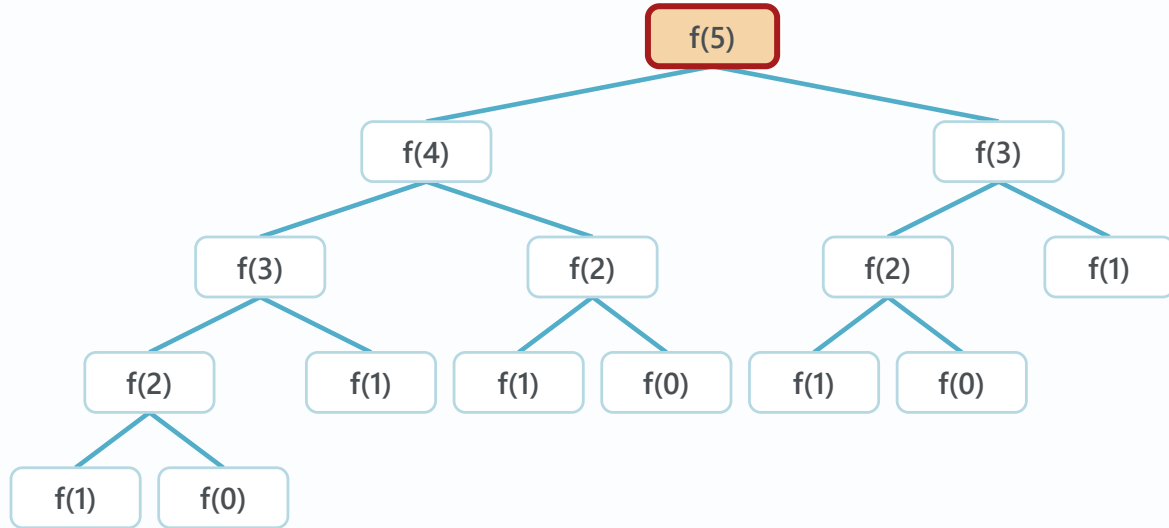
$$F(n) = F(n-1) + F(n-2), n \geq 2$$

```
static long fib(int n) {  
    if (n ≤ 1)  
        return n;           // taban durumları  
    return fib(n - 1) + fib(n - 2);  
}
```

- Tanımın birebir kopyası; doğru ama **çok yavaş**.
- Çağrı sayısı: $T(n) = T(n-1) + T(n-2) + 1$
→ $\Theta(\varphi^n)$, $\varphi \approx 1,618$ (altın oran).
- fib(40) yaklaşık **331 milyon** çağrı yapar; fib(50) ise dakikalar sürer.
- Neden? Aynı alt problemler (fib(3), fib(2), ...) **tekrar tekrar** hesaplanır: örtüşen alt problemler.

fib(5) Özyineleme Ağacı: Çağrı Sırası

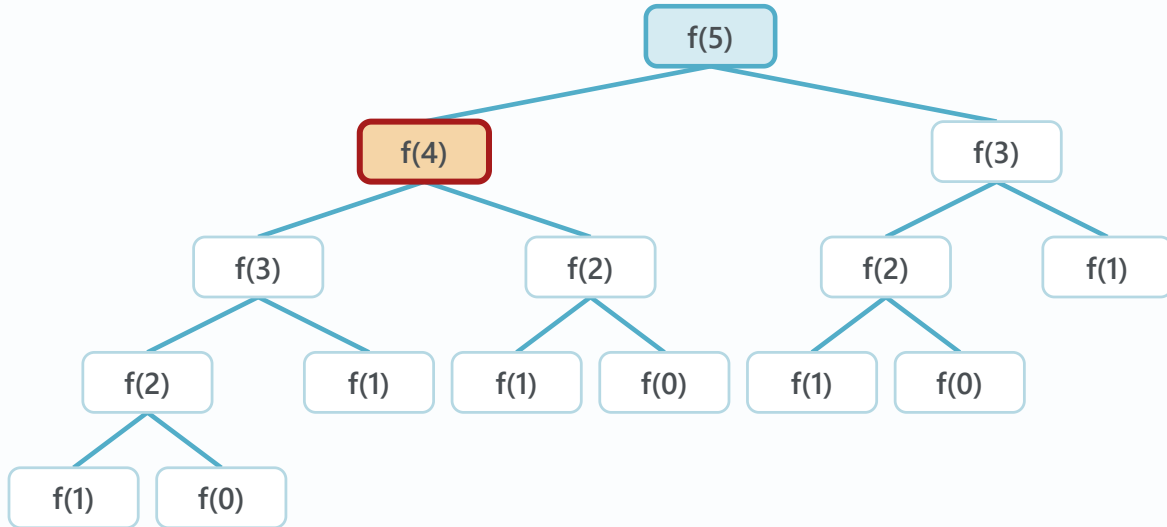
Turuncu: çalışan çağrı · mavi: yığında bekleyen çağrılar · yeşil: bitmiş çağrı (değeriyle).



`fib(5)` çağrıldı Yığın: fib(5)

fib(5) Özyineleme Ağacı: Çağrı Sırası

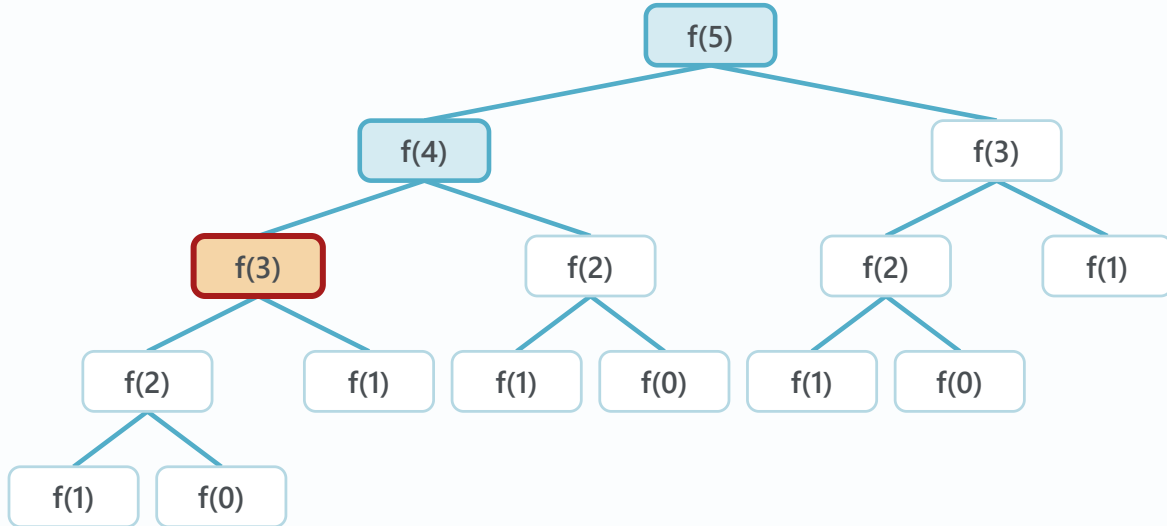
Turuncu: çalışan çağrı · mavi: yığında bekleyen çağrılar · yeşil: bitmiş çağrı (değeriyle).



`fib(4)` çağrıldı Yığın: `fib(5) → fib(4)`

fib(5) Özyineleme Ağacı: Çağrı Sırası

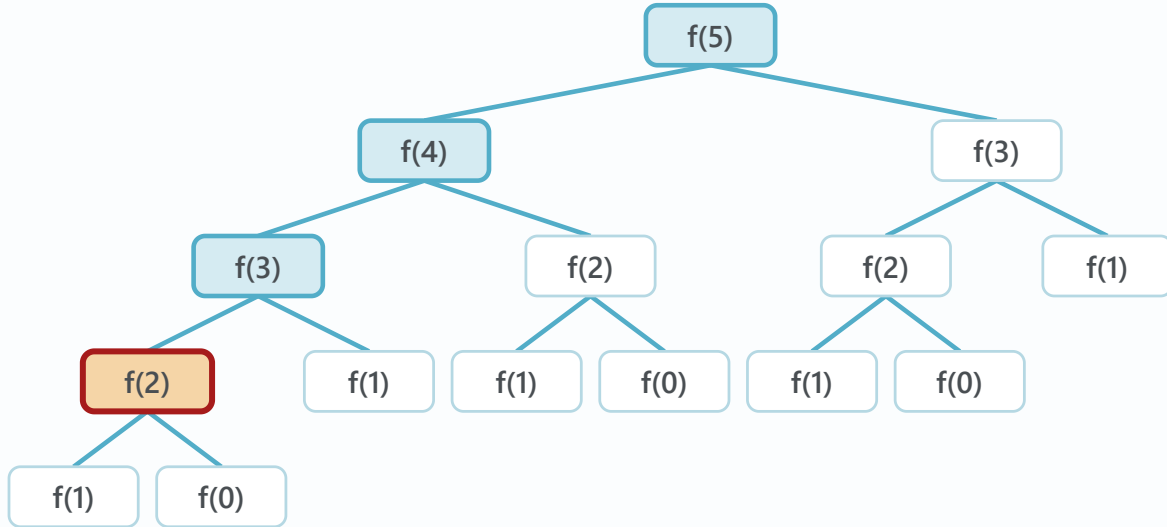
Turuncu: çalışan çağrı · mavi: yığında bekleyen çağrılar · yeşil: bitmiş çağrı (değeriyle).



`fib(3)` çağrıldı Yığın: `fib(5) → fib(4) → fib(3)`

fib(5) Özyineleme Ağacı: Çağrı Sırası

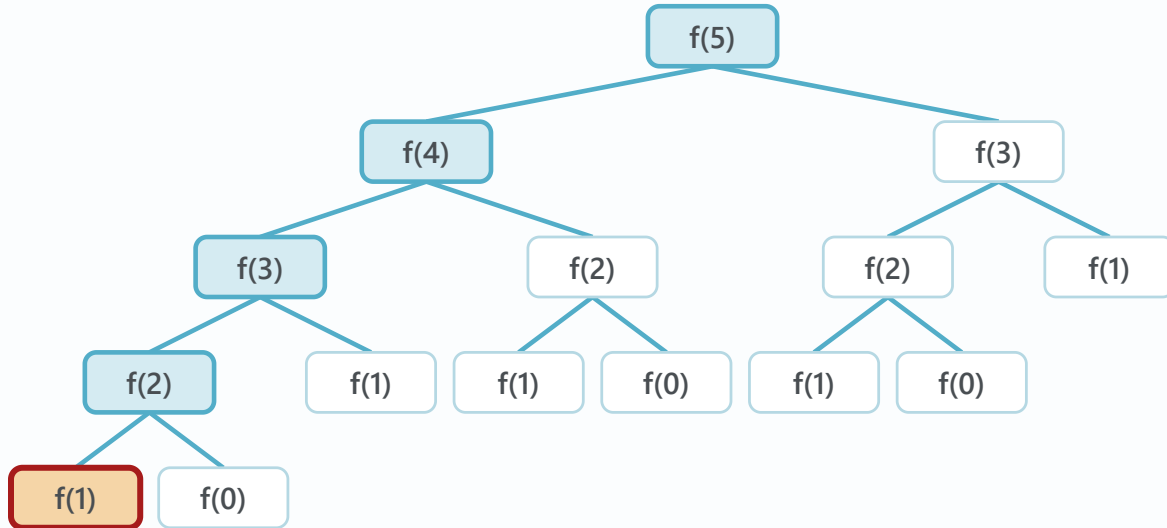
Turuncu: çalışan çağrı · mavi: yığında bekleyen çağrılar · yeşil: bitmiş çağrı (değeriyle).



fib(2) çağrıldı Yığın: fib(5) → fib(4) → fib(3) → fib(2)

fib(5) Özyineleme Ağacı: Çağrı Sırası

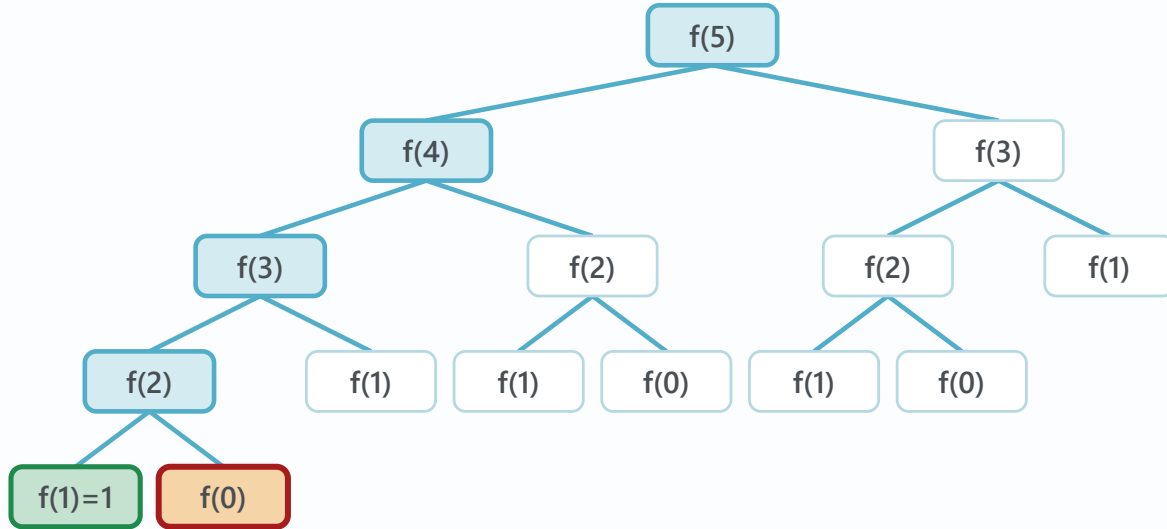
Turuncu: çalışan çağrı · mavi: yığında bekleyen çağrılar · yeşil: bitmiş çağrı (değeriyle).



fib(1) taban durumu: hemen **1** döner. Yığın: fib(5) → fib(4) → fib(3) → fib(2) → fib(1)

fib(5) Özyineleme Ağacı: Çağrı Sırası

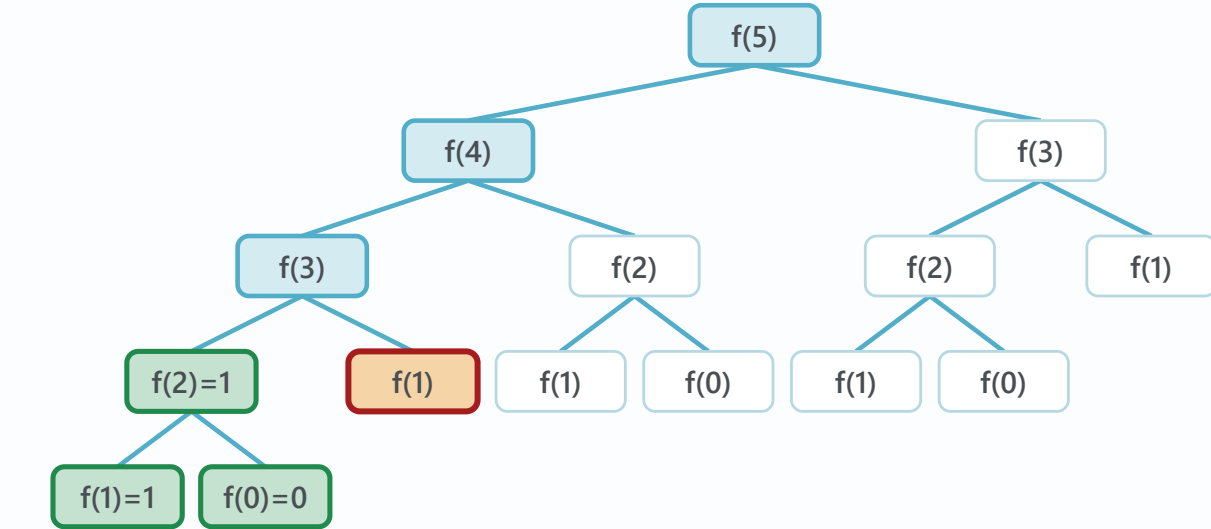
Turuncu: çalışan çağrı · mavi: yığında bekleyen çağrılar · yeşil: bitmiş çağrı (değeriyle).



`fib(0)` taban durumu: hemen **0** döner. Yığın: `fib(5) → fib(4) → fib(3) → fib(2) → fib(0)`

fib(5) Özyineleme Ağacı: Çağrı Sırası

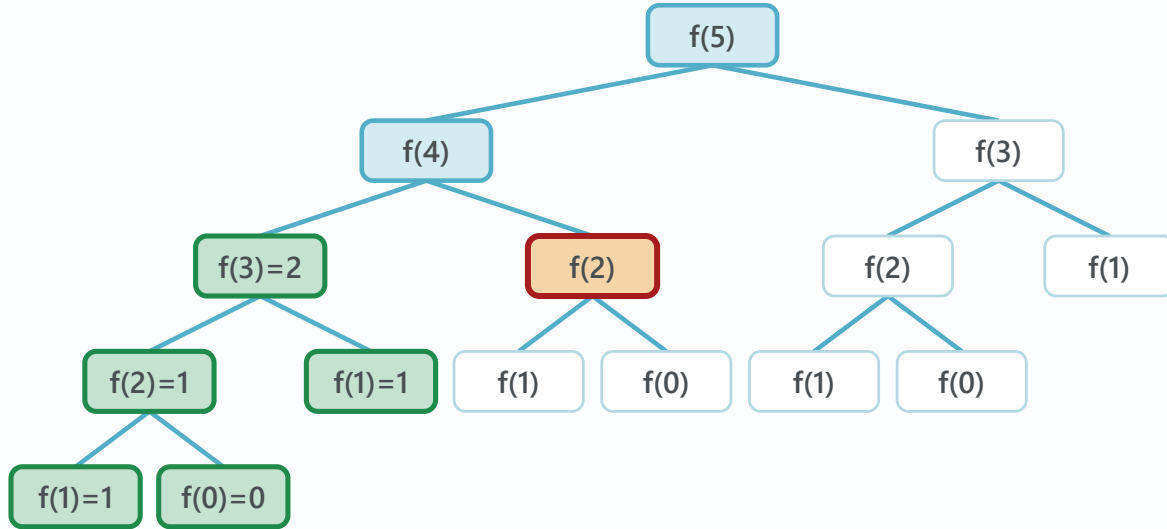
Turuncu: çalışan çağrı · mavi: yığında bekleyen çağrılar · yeşil: bitmiş çağrı (değeriyle).



fib(1) taban durumu: hemen **1** döner. Yığın: fib(5) → fib(4) → fib(3) → fib(1)

fib(5) Özyineleme Ağacı: Çağrı Sırası

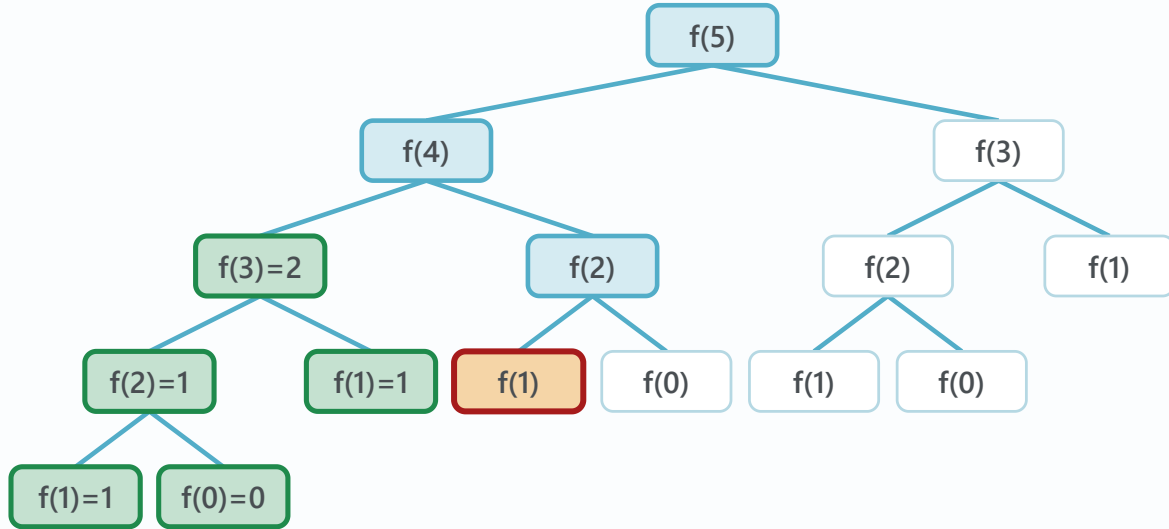
Turuncu: çalışan çağrı · mavi: yığında bekleyen çağrılar · yeşil: bitmiş çağrı (değeriyle).



fib(2) çağrıldı — **2. kez** hesaplanıyor! Yığın: fib(5) → fib(4) → fib(2)

fib(5) Özyineleme Ağacı: Çağrı Sırası

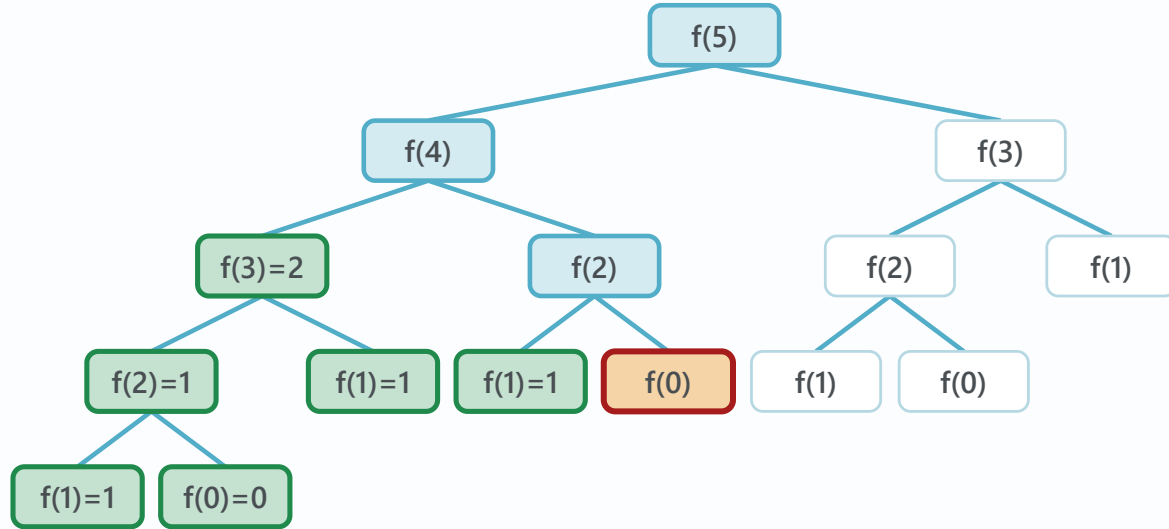
Turuncu: çalışan çağrı · mavi: yığında bekleyen çağrılar · yeşil: bitmiş çağrı (değeriyle).



`fib(1)` taban durumu: hemen **1** döner. Yığın: `fib(5) → fib(4) → fib(2) → fib(1)`

fib(5) Özyineleme Ağacı: Çağrı Sırası

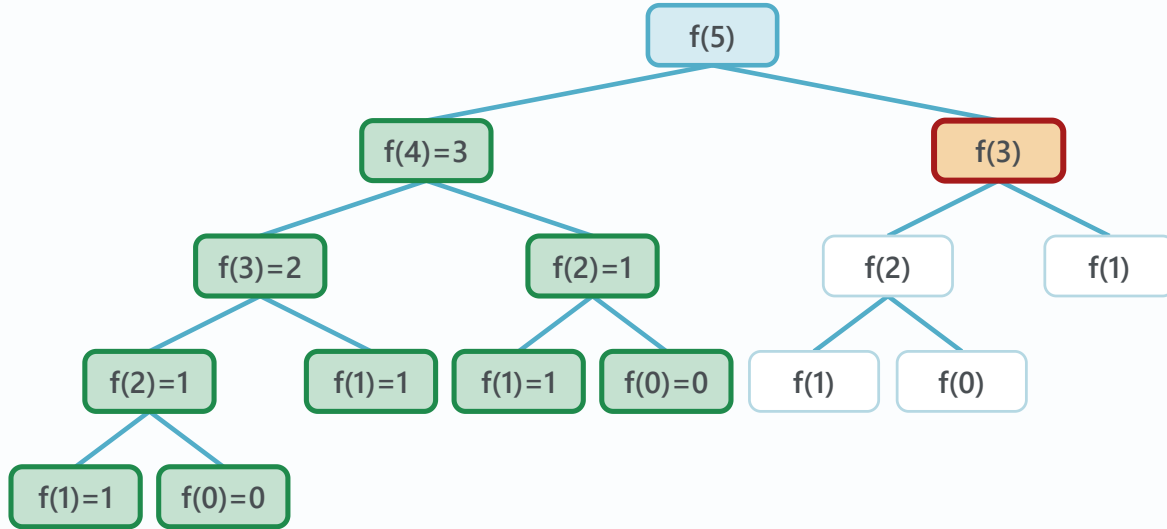
Turuncu: çalışan çağrı · mavi: yığında bekleyen çağrılar · yeşil: bitmiş çağrı (değeriyle).



fib(0) taban durumu: hemen **0** döner. Yığın: fib(5) → fib(4) → fib(2) → fib(0)

fib(5) Özyineleme Ağacı: Çağrı Sırası

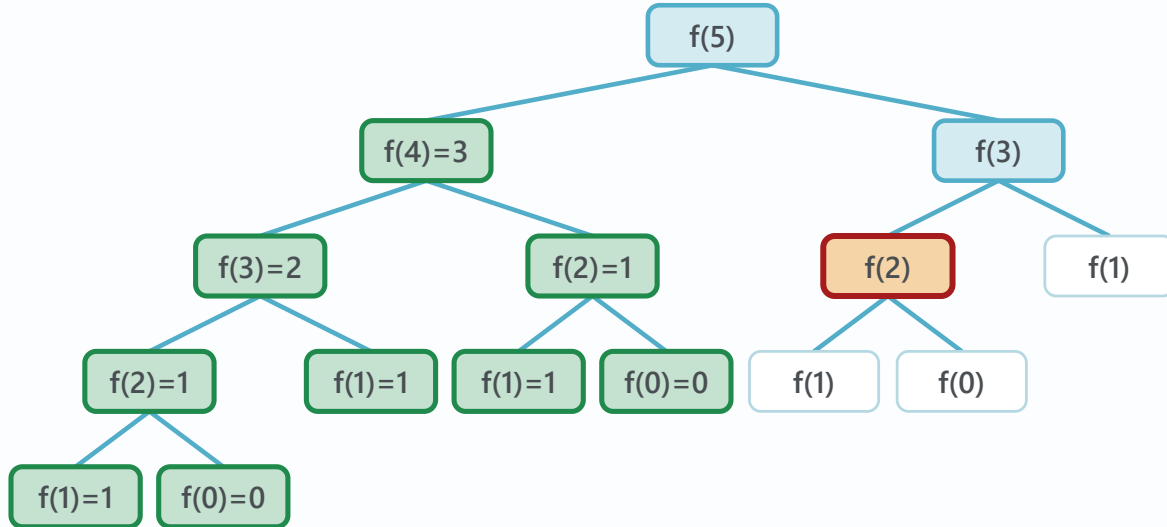
Turuncu: çalışan çağrı · mavi: yığında bekleyen çağrılar · yeşil: bitmiş çağrı (değeriyle).



fib(3) çağrıldı — **2. kez** hesaplanıyor! Yığın: fib(5) → fib(3)

fib(5) Özyineleme Ağacı: Çağrı Sırası

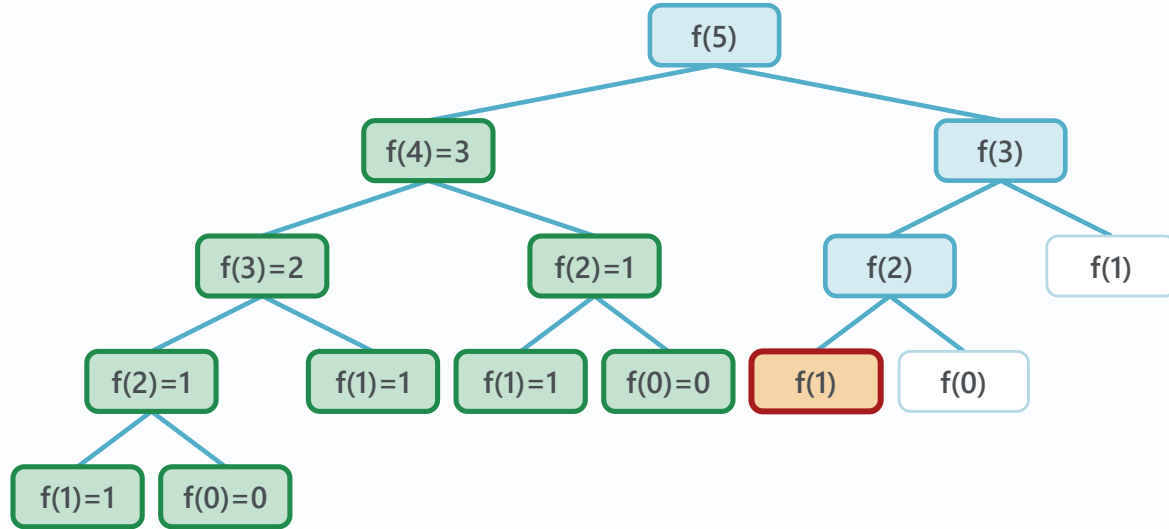
Turuncu: çalışan çağrı · mavi: yığında bekleyen çağrılar · yeşil: bitmiş çağrı (değeriyle).



fib(2) çağrıldı — **3. kez** hesaplanıyor! Yığın: fib(5) → fib(3) → fib(2)

fib(5) Özyineleme Ağacı: Çağrı Sırası

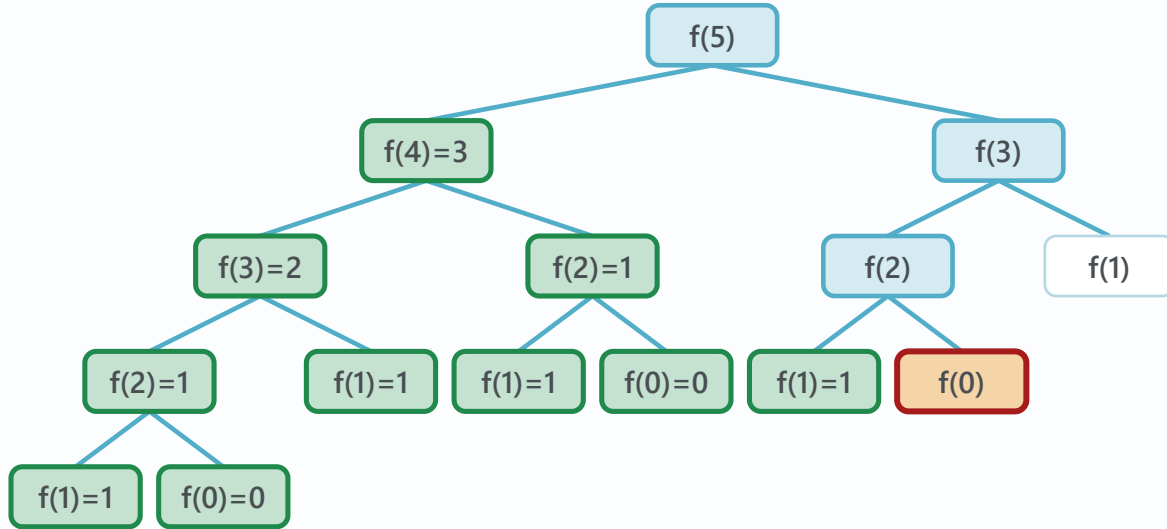
Turuncu: çalışan çağrı · mavi: yığında bekleyen çağrılar · yeşil: bitmiş çağrı (değeriyle).



fib(1) taban durumu: hemen **1** döner. Yığın: fib(5) → fib(3) → fib(2) → fib(1)

fib(5) Özyineleme Ağacı: Çağrı Sırası

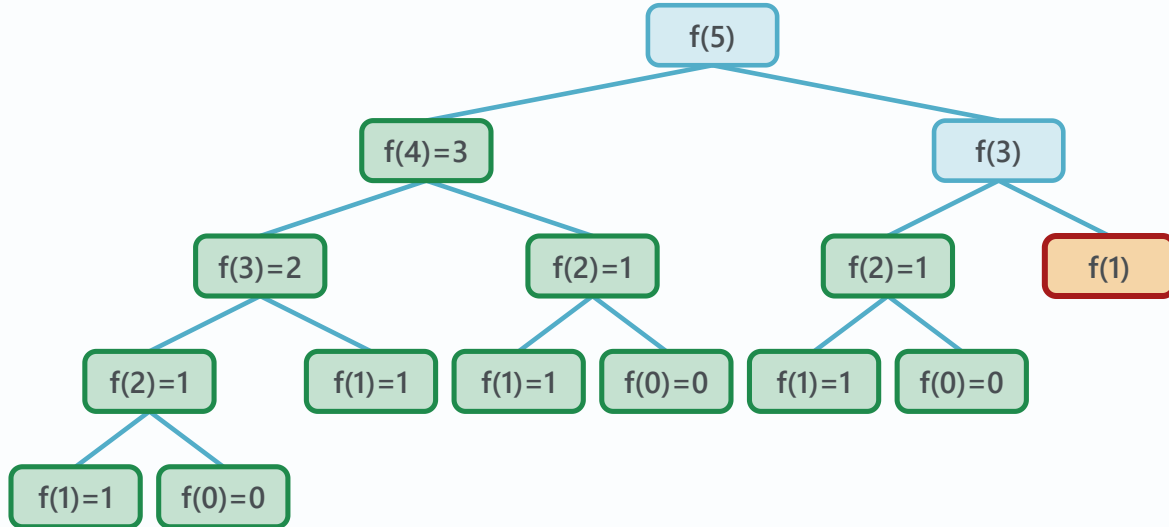
Turuncu: çalışan çağrı · mavi: yığında bekleyen çağrılar · yeşil: bitmiş çağrı (değeriyle).



`fib(0)` taban durumu: hemen **0** döner. Yığın: `fib(5) → fib(3) → fib(2) → fib(0)`

fib(5) Özyineleme Ağacı: Çağrı Sırası

Turuncu: çalışan çağrı · mavi: yığında bekleyen çağrılar · yeşil: bitmiş çağrı (değeriyle).



`fib(1)` taban durumu: hemen **1** döner. Yığın: fib(5) → fib(3) → fib(1)

```

graph TD
    n5["f(5)=5"] --> n4["f(4)=3"]
    n5 --> n3r["f(3)=2"]
    n4 --> n3g["f(3)=2"]
    n4 --> n2r["f(2)=1"]
    n3r --> n2rg["f(2)=1"]
    n3r --> n1g["f(1)=1"]
    n3g --> n2gg["f(2)=1"]
    n3g --> n1gg["f(1)=1"]
    n2r --> n1rg["f(1)=1"]
    n2r --> n0r["f(0)=0"]
    n2rg --> n1rg2["f(1)=1"]
    n2rg --> n0rg["f(0)=0"]
    n2gg --> n1gg2["f(1)=1"]
    n2gg --> n0gg["f(0)=0"]
    n1gg2 --> n0gg2["f(0)=0"]

```

Adım 16 / 16

Tekrarlanan Hesapların Bedeli

n	F(n)	düz özyineleme çağrısı	bellekleme çağrısı
5	5	15	9
10	55	177	19
20	6.765	21.891	39
30	832.040	2.692.537	59
40	102.334.155	331.160.281	79

- Düz özyinelemede çağrı sayısı $2 \cdot F(n+1) - 1$: Fibonacci sayılarının kendisi kadar hızlı (üstel) büyür.
- Oysa yalnızca **n+1 farklı** alt problem vardır: fib(0), fib(1), ..., fib(n).
- Her birini **bir kez** çözüp saklarsak çağrı sayısı **2n - 1**'e iner: üstelden doğrusala.

DP'nin özü: **farklı alt problem sayısı** küçük, ama özyinelemenin onları **çağırma sayısı** büyükse, sonuçları saklamak büyük kazanç sağlar.

FIBONACCI

Bellekleme: fib(5) Adım Adım

```
1 static long[] bellek = new long[50];
2
3 static long fib(int n) {
4     if (n <= 1) return n;          // taban
5     if (bellek[n] != 0) return bellek[n];
6     ▶ long a = fib(n - 1);
7     long b = fib(n - 2);
8     bellek[n] = a + b;
9     return bellek[n];
10 }
```

f(5)

bellek

—	—	0	0	0	0
0	1	2	3	4	5

fib(5) : tabloda yok; önce fib(4) çağrılır.

Adım 1 / 17

DEĞİŞKENLER

n = 5

a = ?

b = -

ÇAĞRI YIĞINI

fib(5)

FIBONACCI

Bellekleme: fib(5) Adım Adım

```
1 static long[] bellek = new long[50];
2
3 static long fib(int n) {
4     if (n <= 1) return n;          // taban
5     if (bellek[n] != 0) return bellek[n];
6     ▶ long a = fib(n - 1);
7     long b = fib(n - 2);
8     bellek[n] = a + b;
9     return bellek[n];
10 }
```



fib(4) : tabloda yok; önce fib(3) çağrılır.

Adım 2 / 17

DEĞİŞKENLER

n = 4

a = ?

b = -

ÇAĞRI YIĞINI

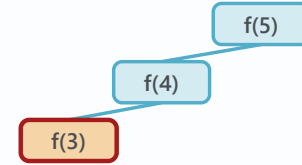
fib(4)

fib(5)

FIBONACCI

Bellekleme: fib(5) Adım Adım

```
1 static long[] bellek = new long[50];
2
3 static long fib(int n) {
4     if (n <= 1) return n;          // taban
5     if (bellek[n] != 0) return bellek[n];
6     ▶ long a = fib(n - 1);
7     long b = fib(n - 2);
8     bellek[n] = a + b;
9     return bellek[n];
10 }
```



fib(3) : tabloda yok; önce fib(2) çağrılır.

Adım 3 / 17

DEĞİŞKENLER

n = 3

a = ?

b = -

ÇAĞRI YIĞINI

fib(3)

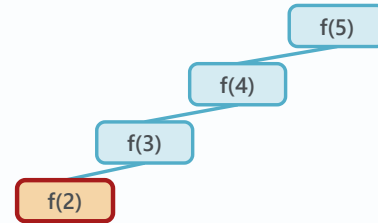
fib(4)

fib(5)

FIBONACCI

Bellekleme: fib(5) Adım Adım

```
1 static long[] bellek = new long[50];
2
3 static long fib(int n) {
4     if (n <= 1) return n;          // taban
5     if (bellek[n] != 0) return bellek[n];
6     ▶ long a = fib(n - 1);
7     long b = fib(n - 2);
8     bellek[n] = a + b;
9     return bellek[n];
10 }
```



fib(2) : tabloda yok; önce fib(1) çağrılır.

Adım 4 / 17

DEĞİŞKENLER

n = 2

a = ?

b = -

ÇAĞRI YIĞINI

fib(2)

fib(3)

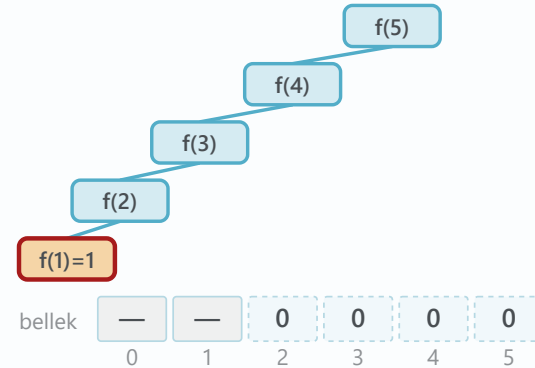
fib(4)

fib(5)

FIBONACCI

Bellekleme: fib(5) Adım Adım

```
1 static long[] bellek = new long[50];
2
3 static long fib(int n) {
4     if (n <= 1) return n; // taban
5     if (bellek[n] != 0) return bellek[n];
6     long a = fib(n - 1);
7     long b = fib(n - 2);
8     bellek[n] = a + b;
9     return bellek[n];
10 }
```



fib(1) : taban durumu, 1 döner.

Adım 5 / 17

DEĞİŞKENLER

n = 1

a = -

b = -

ÇAĞRI YIĞINI

fib(1)

fib(2)

fib(3)

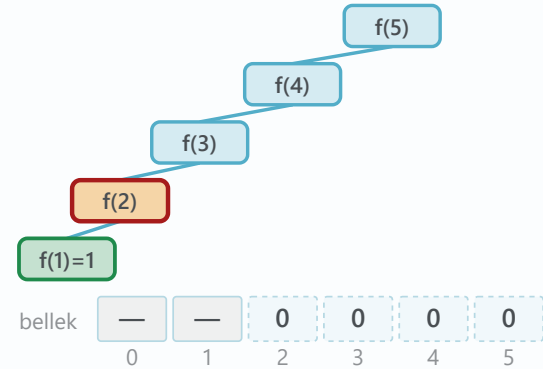
fib(4)

fib(5)

FIBONACCI

Bellekleme: fib(5) Adım Adım

```
1 static long[] bellek = new long[50];
2
3 static long fib(int n) {
4     if (n <= 1) return n;          // taban
5     if (bellek[n] != 0) return bellek[n];
6     long a = fib(n - 1);
7     ► long b = fib(n - 2);
8     bellek[n] = a + b;
9     return bellek[n];
10 }
```



fib(1) döndü (a = 1); şimdi fib(0) çağrılır.

Adım 6 / 17

DEĞİŞKENLER

n = 2

a = 1

b = ?

ÇAĞRI YIĞINI

fib(2)

fib(3)

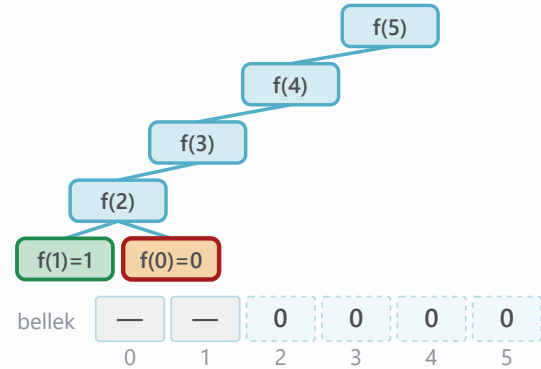
fib(4)

fib(5)

FIBONACCI

Bellekleme: fib(5) Adım Adım

```
1 static long[] bellek = new long[50];
2
3 static long fib(int n) {
4     if (n <= 1) return n; // taban
5     if (bellek[n] != 0) return bellek[n];
6     long a = fib(n - 1);
7     long b = fib(n - 2);
8     bellek[n] = a + b;
9     return bellek[n];
10 }
```



fib(0) : taban durumu, 0 döner.

Adım 7 / 17

DEĞİŞKENLER

n = 0

a = -

b = -

ÇAĞRI YIĞINI

fib(0)

fib(2)

fib(3)

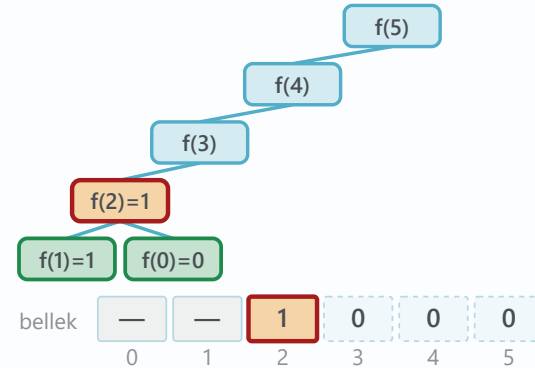
fib(4)

fib(5)

FIBONACCI

Bellekleme: fib(5) Adım Adım

```
1 static long[] bellek = new long[50];
2
3 static long fib(int n) {
4     if (n <= 1) return n;          // taban
5     if (bellek[n] != 0) return bellek[n];
6     long a = fib(n - 1);
7     long b = fib(n - 2);
8     ▶ bellek[n] = a + b;
9     ▶ return bellek[n];
10 }
```



bellek[2] = 1 + 0 = 1 tabloya yazılır ve döndürülür.

Adım 8 / 17

DEĞİŞKENLER

n = 2

a = 1

b = 0

ÇAĞRI YIĞINI

fib(2)

fib(3)

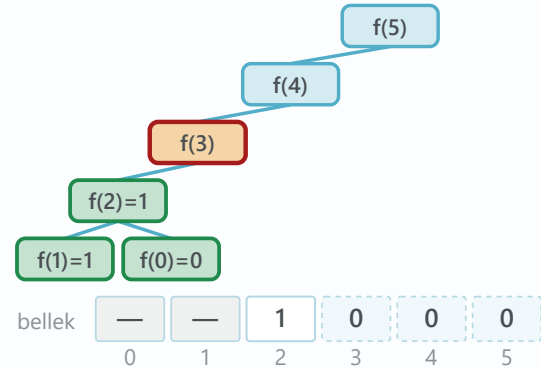
fib(4)

fib(5)

FIBONACCI

Bellekleme: fib(5) Adım Adım

```
1 static long[] bellek = new long[50];
2
3 static long fib(int n) {
4     if (n <= 1) return n;          // taban
5     if (bellek[n] != 0) return bellek[n];
6     long a = fib(n - 1);
7     ► long b = fib(n - 2);
8     bellek[n] = a + b;
9     return bellek[n];
10 }
```



fib(2) döndü (a = 1); şimdi fib(1) çağrılır.

Adım 9 / 17

DEĞİŞKENLER

n = 3

a = 1

b = ?

ÇAĞRI YIĞINI

fib(3)

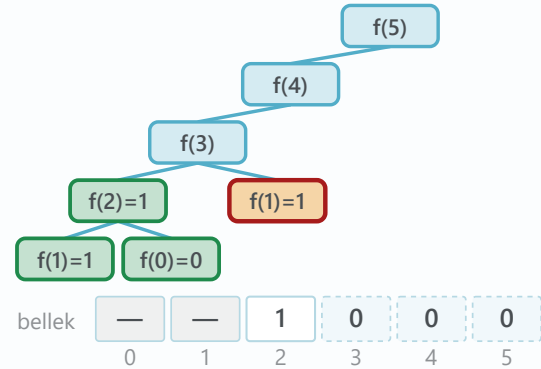
fib(4)

fib(5)

FIBONACCI

Bellekleme: fib(5) Adım Adım

```
1 static long[] bellek = new long[50];
2
3 static long fib(int n) {
4     ► if (n <= 1) return n; // taban
5     if (bellek[n] != 0) return bellek[n];
6     long a = fib(n - 1);
7     long b = fib(n - 2);
8     bellek[n] = a + b;
9     return bellek[n];
10 }
```



fib(1) : taban durumu, 1 döner.

Adım 10 / 17

DEĞİŞKENLER

n = 1

a = —

b = —

ÇAĞRI YIĞINI

fib(1)

fib(3)

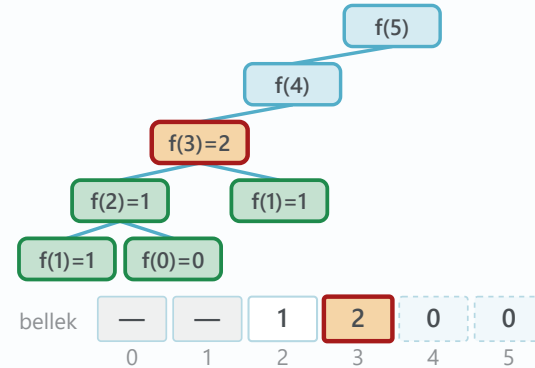
fib(4)

fib(5)

FIBONACCI

Bellekleme: fib(5) Adım Adım

```
1 static long[] bellek = new long[50];  
2  
3 static long fib(int n) {  
4     if (n <= 1) return n; // taban  
5     if (bellek[n] != 0) return bellek[n];  
6     long a = fib(n - 1);  
7     long b = fib(n - 2);  
8     ▶ bellek[n] = a + b;  
9     ▶ return bellek[n];  
10 }
```



bellek[3] = 1 + 1 = 2 tabloya yazılır ve döndürülür.

Adım 11 / 17

DEĞİŞKENLER

n = 3

a = 1

b = 1

ÇAĞRI YIĞINI

fib(3)

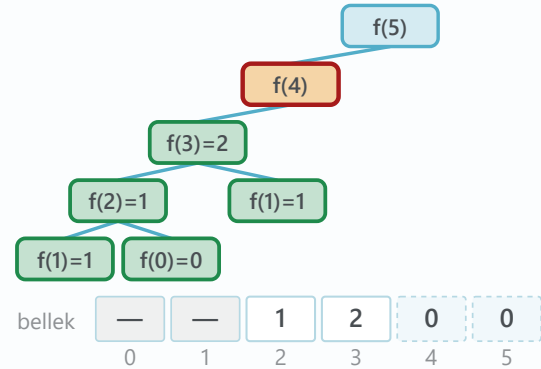
fib(4)

fib(5)

FIBONACCI

Bellekleme: fib(5) Adım Adım

```
1 static long[] bellek = new long[50];
2
3 static long fib(int n) {
4     if (n <= 1) return n;          // taban
5     if (bellek[n] != 0) return bellek[n];
6     long a = fib(n - 1);
7     ► long b = fib(n - 2);
8     bellek[n] = a + b;
9     return bellek[n];
10 }
```



fib(3) döndü (a = 2); şimdi fib(2) çağrılır.

Adım 12 / 17

DEĞİŞKENLER

n = 4

a = 2

b = ?

ÇAĞRI YIĞINI

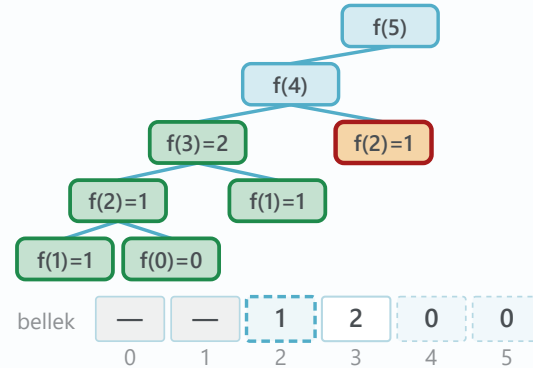
fib(4)

fib(5)

FIBONACCI

Bellekleme: fib(5) Adım Adım

```
1 static long[] bellek = new long[50];
2
3 static long fib(int n) {
4     if (n <= 1) return n;          // taban
5     ▶ if (bellek[n] != 0) return bellek[n];
6     long a = fib(n - 1);
7     long b = fib(n - 2);
8     bellek[n] = a + b;
9     return bellek[n];
10 }
```



fib(2) : bellek[2] = 1 tabloda **hazır**; yeniden hesaplanmaz.

Adım 13 / 17

DEĞİŞKENLER

n = 2

a = —

b = —

ÇAĞRI YIĞINI

fib(2)

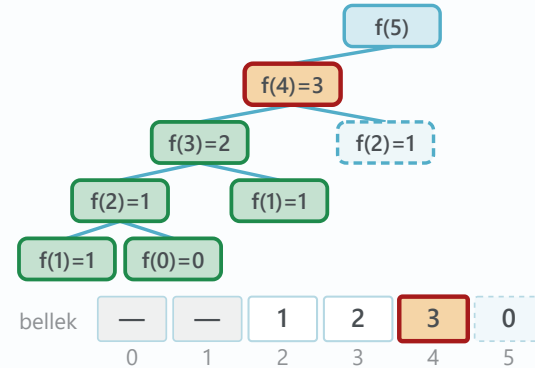
fib(4)

fib(5)

FIBONACCI

Bellekleme: fib(5) Adım Adım

```
1 static long[] bellek = new long[50];
2
3 static long fib(int n) {
4     if (n <= 1) return n;          // taban
5     if (bellek[n] != 0) return bellek[n];
6     long a = fib(n - 1);
7     long b = fib(n - 2);
8     ▶ bellek[n] = a + b;
9     ▶ return bellek[n];
10 }
```



bellek[4] = 2 + 1 = 3 tabloya yazılır ve döndürülür.

Adım 14 / 17

DEĞİŞKENLER

n = 4

a = 2

b = 1

ÇAĞRI YIĞINI

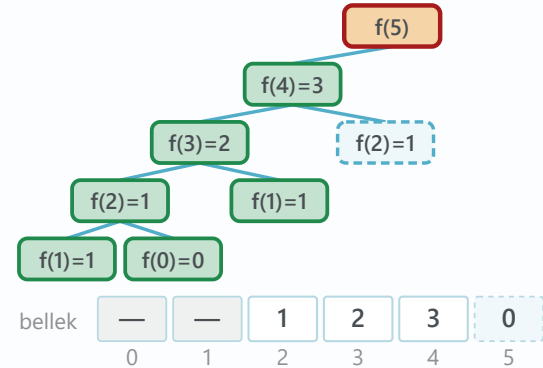
fib(4)

fib(5)

FIBONACCI

Bellekleme: fib(5) Adım Adım

```
1 static long[] bellek = new long[50];
2
3 static long fib(int n) {
4     if (n <= 1) return n;          // taban
5     if (bellek[n] != 0) return bellek[n];
6     long a = fib(n - 1);
7     ► long b = fib(n - 2);
8     bellek[n] = a + b;
9     return bellek[n];
10 }
```



fib(4) döndü (a = 3); şimdi fib(3) çağrılır.

Adım 15 / 17

DEĞİŞKENLER

n = 5

a = 3

b = ?

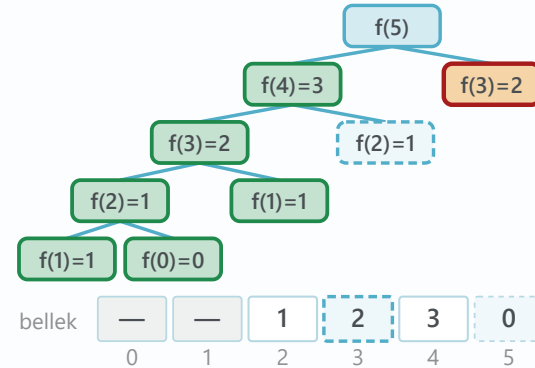
ÇAĞRI YIĞINI

fib(5)

FIBONACCI

Bellekleme: fib(5) Adım Adım

```
1 static long[] bellek = new long[50];
2
3 static long fib(int n) {
4     if (n <= 1) return n;          // taban
5     if (bellek[n] != 0) return bellek[n];
6     long a = fib(n - 1);
7     long b = fib(n - 2);
8     bellek[n] = a + b;
9     return bellek[n];
10 }
```



fib(3) : bellek[3] = 2 tabloda **hazır**; yeniden hesaplanmaz.

Adım 16 / 17

DEĞİŞKENLER

n = 3

a = —

b = —

ÇAĞRI YIĞINI

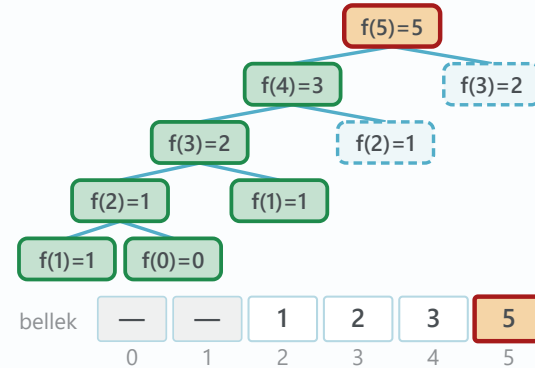
fib(3)

fib(5)

FIBONACCI

Bellekleme: fib(5) Adım Adım

```
1 static long[] bellek = new long[50];
2
3 static long fib(int n) {
4     if (n <= 1) return n;           // taban
5     if (bellek[n] != 0) return bellek[n];
6     long a = fib(n - 1);
7     long b = fib(n - 2);
8     ▶ bellek[n] = a + b;
9     ▶ return bellek[n];
10 }
```



bellek[5] = 3 + 2 = 5 tabloya yazılır ve döndürülür.

Adım 17 / 17

DEĞİŞKENLER

n = 5

a = 3

b = 2

ÇAĞRI YIĞINI

fib(5)

FIBONACCI

Tablolama: fibonacci(9) Adım Adım

```
1 static int fibonacci(int n) {  
2     int[] fib = new int[n + 1];  
3     fib[0] = 0;    // taban durumları  
4     fib[1] = 1;  
5     for (int i = 2; i <= n; i++)  
6         fib[i] = fib[i - 1] + fib[i - 2];  
7     return fib[n];  
8 }
```

0	0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9

fib dizisi $n + 1 = 10$ öğeyle oluşturulur (Java'da başlangıçta hepsi 0).

Adım 1 / 11

DEĞİŞKENLER

n = 9

i = -

FIBONACCI

Tablolama: fibonacci(9) Adım Adım

```
1 static int fibonacci(int n) {  
2   int[] fib = new int[n + 1];  
▶   fib[0] = 0;    // taban durumları  
▶   fib[1] = 1;  
5   for (int i = 2; i <= n; i++)  
6     fib[i] = fib[i - 1] + fib[i - 2];  
7   return fib[n];  
8 }
```

0	1								
0	1	2	3	4	5	6	7	8	9

Taban durumları yazılır: `fib[0] = 0` , `fib[1] = 1` . Boş hücreler henüz hesaplanmadı.

Adım 2 / 11

DEĞİŞKENLER

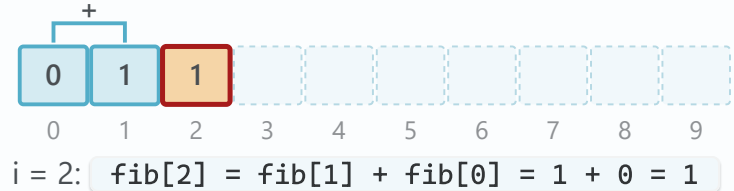
`n = 9`

`i = -`

FIBONACCI

Tablolama: fibonacci(9) Adım Adım

```
1 static int fibonacci(int n) {  
2     int[] fib = new int[n + 1];  
3     fib[0] = 0;    // taban durumları  
4     fib[1] = 1;  
5     for (int i = 2; i <= n; i++)  
6         fib[i] = fib[i - 1] + fib[i - 2];  
7     return fib[n];  
8 }
```



Adım 3 / 11

DEĞİŞKENLER

`n = 9`

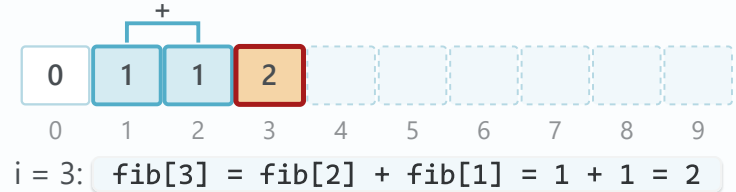
`i = 2`

`fib[i] = 1`

FIBONACCI

Tablolama: fibonacci(9) Adım Adım

```
1 static int fibonacci(int n) {  
2     int[] fib = new int[n + 1];  
3     fib[0] = 0;    // taban durumları  
4     fib[1] = 1;  
5     for (int i = 2; i <= n; i++)  
6         fib[i] = fib[i - 1] + fib[i - 2];  
7     return fib[n];  
8 }
```



Adım 4 / 11

DEĞİŞKENLER

n = 9

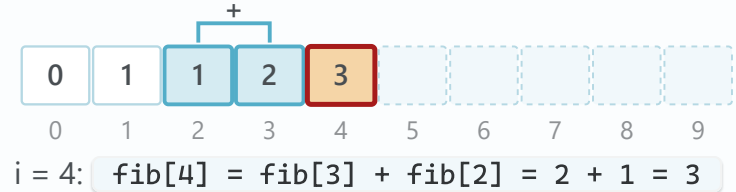
i = 3

fib[i] = 2

FIBONACCI

Tablolama: fibonacci(9) Adım Adım

```
1 static int fibonacci(int n) {  
2     int[] fib = new int[n + 1];  
3     fib[0] = 0;    // taban durumları  
4     fib[1] = 1;  
5     for (int i = 2; i <= n; i++)  
6         fib[i] = fib[i - 1] + fib[i - 2];  
7     return fib[n];  
8 }
```



Adım 5 / 11

DEĞİŞKENLER

n = 9

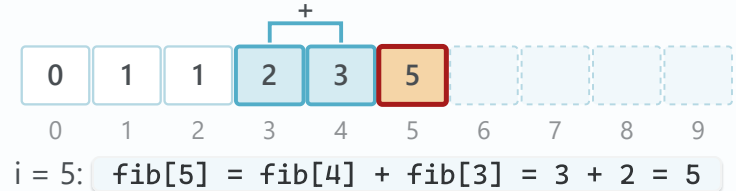
i = 4

fib[i] = 3

FIBONACCI

Tablolama: fibonacci(9) Adım Adım

```
1 static int fibonacci(int n) {  
2     int[] fib = new int[n + 1];  
3     fib[0] = 0;    // taban durumları  
4     fib[1] = 1;  
5     for (int i = 2; i <= n; i++)  
6         fib[i] = fib[i - 1] + fib[i - 2];  
7     return fib[n];  
8 }
```



Adım 6 / 11

DEĞİŞKENLER

n = 9

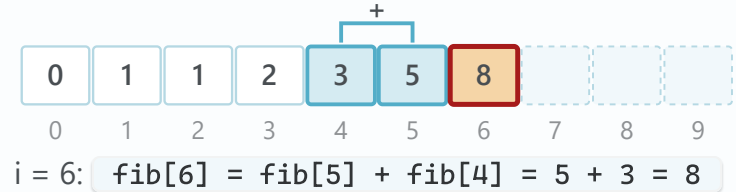
i = 5

fib[i] = 5

FIBONACCI

Tablolama: fibonacci(9) Adım Adım

```
1 static int fibonacci(int n) {  
2     int[] fib = new int[n + 1];  
3     fib[0] = 0;    // taban durumları  
4     fib[1] = 1;  
5     for (int i = 2; i <= n; i++)  
6         fib[i] = fib[i - 1] + fib[i - 2];  
7     return fib[n];  
8 }
```



Adım 7 / 11

DEĞİŞKENLER

n = 9

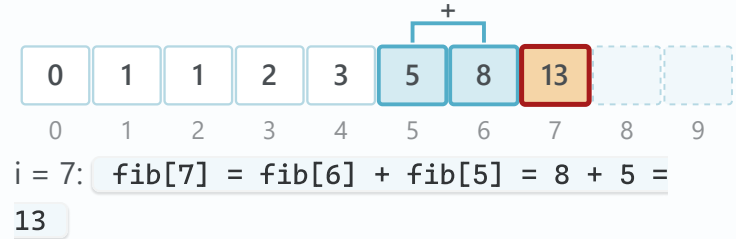
i = 6

fib[i] = 8

FIBONACCI

Tablolama: fibonacci(9) Adım Adım

```
1 static int fibonacci(int n) {  
2     int[] fib = new int[n + 1];  
3     fib[0] = 0;    // taban durumları  
4     fib[1] = 1;  
5     for (int i = 2; i <= n; i++)  
6         fib[i] = fib[i - 1] + fib[i - 2];  
7     return fib[n];  
8 }
```



Adım 8 / 11

DEĞİŞKENLER

n = 9

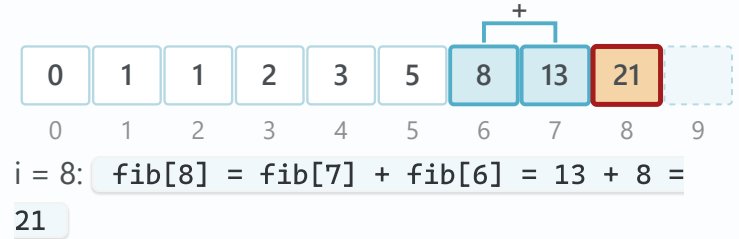
i = 7

fib[i] = 13

FIBONACCI

Tablolama: fibonacci(9) Adım Adım

```
1 static int fibonacci(int n) {  
2     int[] fib = new int[n + 1];  
3     fib[0] = 0;    // taban durumları  
4     fib[1] = 1;  
5     for (int i = 2; i <= n; i++)  
6         fib[i] = fib[i - 1] + fib[i - 2];  
7     return fib[n];  
8 }
```



Adım 9 / 11

DEĞİŞKENLER

n = 9

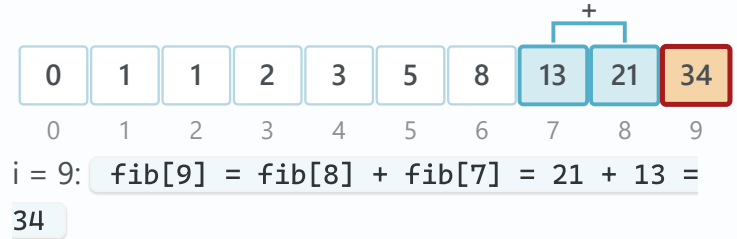
i = 8

fib[i] = 21

FIBONACCI

Tablolama: fibonacci(9) Adım Adım

```
1 static int fibonacci(int n) {  
2     int[] fib = new int[n + 1];  
3     fib[0] = 0;    // taban durumları  
4     fib[1] = 1;  
5     for (int i = 2; i <= n; i++)  
6         fib[i] = fib[i - 1] + fib[i - 2];  
7     return fib[n];  
8 }
```



Adım 10 / 11

DEĞİŞKENLER

n = 9

i = 9

fib[i] = 34

FIBONACCI

Tablolama: fibonacci(9) Adım Adım

```
1 static int fibonacci(int n) {  
2     int[] fib = new int[n + 1];  
3     fib[0] = 0;    // taban durumları  
4     fib[1] = 1;  
5     for (int i = 2; i <= n; i++)  
6         fib[i] = fib[i - 1] + fib[i - 2];  
7     return fib[n];  
8 }
```

0	1	1	2	3	5	8	13	21	34
0	1	2	3	4	5	6	7	8	9

Döngü biter; `fib[9] = 34` döndürülür. Her hücre **bir kez** hesaplandı: $O(n)$ zaman, $O(n)$ bellek.

Adım 11 / 11

DEĞİŞKENLER

`n = 9`

`i = 10`

`fib[i] = -`

FIBONACCI

Fibonacci: O(1) Bellek ve Karşılaştırma

```
static long fib(int n) {  
    if (n ≤ 1) return n;  
    long once = 0, simdi = 1;  
    for (int i = 2; i ≤ n; i++) {  
        long yeni = once + simdi;  
        once = simdi;  
        simdi = yeni;  
    }  
    return simdi;  
}
```

Yöntem	Zaman	Ek bellek
Düz özyineleme	$\Theta(\varphi^n)$	O(n) yığın
Bellekleme	O(n)	O(n) tablo + yığın
Tablolama	O(n)	O(n) tablo
İki değişken	O(n)	O(1)

Bellek azaltma: fib[i] yalnızca son **iki** değere bağlıdır; tüm diziyi saklamaya gerek yoktur. Aynı fikir sırt çantasında "yalnızca önceki satır" olarak karşımıza çıkacak.

0/1 SIRT ÇANTASI

0/1 Sırt Çantası (0/1 Knapsack) Problemi

- n eşya: her eşyanın bir **değeri** v_i ve bir **ağırlığı** (hacmi) w_i var.
- Çantanın **kapasitesi** W .
- Her eşya ya **tamamen alınır (1)** ya da **hiç alınmaz (0)**; bölünemez.
- Amaç: toplam ağırlık $\leq W$ olacak şekilde **toplam değeri en büyük** yapmak.



Soru: Resimdeki çantaya hangi eşyalar konmalı?

► **Cevap**

Neden Kaba Kuvvet ya da Açgözlü Değil?

Kaba kuvvet

Her eşya için 2 seçenek $\rightarrow 2^n$ alt küme. $n = 50$ için 10^{15} 'ten fazla küme: pratikte imkânsız.

Açgözlü (değer/ağırlık oranı)

$W = 50$; eşyalar (60 ₺, 10 kg), (100 ₺, 20 kg), (120 ₺, 30 kg). Oranlar 6, 5, 4 $\rightarrow$ açgözlü ilk ikisini alır: **160 ₺**. Optimal: ikinci + üçüncü = **220 ₺**.

Açgözlü seçim yalnızca eşyalar **bölünebiliyorsa** (kesirli sırt çantası) optimaldir. 0/1 durumunda bir eşyayı almak, sonraki seçenekleri değiştirir: her iki seçeneği de denemek gerekir $\rightarrow$ **dinamik programlama**.

0/1 Sırt Çantası: Durum ve Bağını

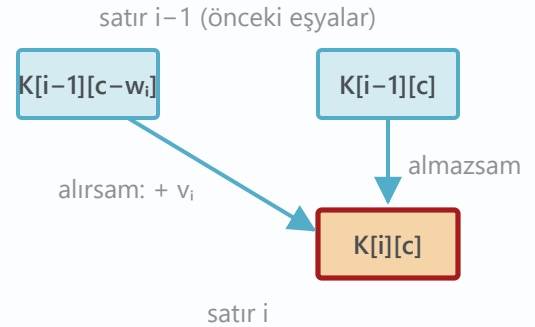
Durum: $K[i][c]$ = ilk i eşya ve c kapasite ile elde edilebilecek en büyük değer.

$$K[0][c] = 0, K[i][0] = 0$$

$$w_i > c \text{ ise: } K[i][c] = K[i-1][c]$$

$$\text{aksi hâlde: } K[i][c] = \max(K[i-1][c], v_i + K[i-1][c - w_i])$$

- **Almazsam:** kapasite aynı kalır, önceki eşyalarla en iyisi.
- **Alırsam:** v_i kazanırım, kalan $c - w_i$ kapasiteyi önceki eşyalarla en iyi şekilde doldururum.
- Cevap: **$K[n][W]$** . Tablo $(n+1) \times (W+1)$, satır satır doldurulur.



0/1 SIRT ÇANTASI

0/1 Sırt Çantası: Tabloyu Adım Adım Doldur ($W = 6$)

```
1 int sirtCantasi(int[] v, int[] w, int W) {  
▶   int n = v.length;  
▶   int[][] K = new int[n + 1][W + 1];  
4   for (int i = 1; i <= n; i++)  
5       for (int c = 1; c <= W; c++) {  
6           int vi = v[i - 1], wi = w[i - 1];  
7           if (wi > c)                // sığmaz  
8               K[i][c] = K[i - 1][c];  
9           else                        // alma / al  
10              K[i][c] = Math.max(K[i - 1][c],  
11                               vi + K[i - 1][c - wi]);  
12     }  
13     return K[n][W];  
14 }
```

			c (kapasite) →						
i	v	w	0	1	2	3	4	5	6
0			0	0	0	0	0	0	0
1	5	4	0						
2	4	3	0						
3	3	2	0						
4	2	1	0						

Tablo $(n+1) \times (W+1) = 5 \times 7$. Satır 0 (eşya yok) ve sütun 0 (kapasite yok) **0**'dir.

Adım 1 / 26

DEĞİŞKENLER

n = 4

W = 6

i = -

c = -

0/1 SIRT ÇANTASI

0/1 Sırt Çantası: Tabloyu Adım Adım Doldur (W = 6)

```
1 int sirtCantasi(int[] v, int[] w, int W) {
2     int n = v.length;
3     int[][] K = new int[n + 1][W + 1];
4     for (int i = 1; i <= n; i++)
5         for (int c = 1; c <= W; c++) {
6             int vi = v[i - 1], wi = w[i - 1];
7             ▶ if (wi > c) // sığmaz
8                 ▶ K[i][c] = K[i - 1][c];
9             else // alma / al
10                 K[i][c] = Math.max(K[i - 1][c],
11                                     vi + K[i - 1][c - wi]);
12         }
13     return K[n][W];
14 }
```

			c (kapasite) →						
i	v	w	0	1	2	3	4	5	6
0			0	0	0	0	0	0	0
1	5	4	0	0					
2	4	3	0						
3	3	2	0						
4	2	1	0						

i=1, c=1: w=4 > 1, eşya **sığmaz** → üstteki değer kopyalanır: K[1][1] = 0

Adım 2 / 26

DEĞİŞKENLER

i = 1

c = 1

vi = 5

wi = 4

alma = 0

al = -

0/1 SIRT ÇANTASI

0/1 Sırt Çantası: Tabloyu Adım Adım Doldur (W = 6)

```
1 int sirtCantasi(int[] v, int[] w, int W) {
2     int n = v.length;
3     int[][] K = new int[n + 1][W + 1];
4     for (int i = 1; i <= n; i++)
5         for (int c = 1; c <= W; c++) {
6             int vi = v[i - 1], wi = w[i - 1];
7             ▶ if (wi > c) // sığmaz
8                 ▶ K[i][c] = K[i - 1][c];
9             else // alma / al
10                 K[i][c] = Math.max(K[i - 1][c],
11                                     vi + K[i - 1][c - wi]);
12        }
13    return K[n][W];
14 }
```

			c (kapasite) →						
i	v	w	0	1	2	3	4	5	6
0			0	0	0	0	0	0	0
1	5	4	0	0	0				
2	4	3	0						
3	3	2	0						
4	2	1	0						

i=1, c=2: w=4 > 2, eşya **sığmaz** → üstteki değer kopyalanır: K[1][2] = 0

Adım 3 / 26

DEĞİŞKENLER

i = 1

c = 2

vi = 5

wi = 4

alma = 0

al = -

0/1 SIRT ÇANTASI

0/1 Sırt Çantası: Tabloyu Adım Adım Doldur (W = 6)

```
1 int sirtCantasi(int[] v, int[] w, int W) {  
2     int n = v.length;  
3     int[][] K = new int[n + 1][W + 1];  
4     for (int i = 1; i <= n; i++)  
5         for (int c = 1; c <= W; c++) {  
6             int vi = v[i - 1], wi = w[i - 1];  
7             ▶ if (wi > c) // sığmaz  
8                 ▶ K[i][c] = K[i - 1][c];  
9             else // alma / al  
10                 K[i][c] = Math.max(K[i - 1][c],  
11                                     vi + K[i - 1][c - wi]);  
12         }  
13     return K[n][W];  
14 }
```

			c (kapasite) →						
i	v	w	0	1	2	3	4	5	6
0			0	0	0	0	0	0	0
1	5	4	0	0	0	0			
2	4	3	0						
3	3	2	0						
4	2	1	0						

i=1, c=3: w=4 > 3, eşya **sığmaz** → üstteki değer kopyalanır: K[1][3] = 0

Adım 4 / 26

DEĞİŞKENLER

i = 1

c = 3

vi = 5

wi = 4

alma = 0

al = -

0/1 SIRT ÇANTASI

0/1 Sırt Çantası: Tabloyu Adım Adım Doldur (W = 6)

```
1 int sirtCantasi(int[] v, int[] w, int W) {  
2     int n = v.length;  
3     int[][] K = new int[n + 1][W + 1];  
4     for (int i = 1; i <= n; i++)  
5         for (int c = 1; c <= W; c++) {  
6             int vi = v[i - 1], wi = w[i - 1];  
7             if (wi > c) // sığmaz  
8                 K[i][c] = K[i - 1][c];  
9             else // alma / al  
10                K[i][c] = Math.max(K[i - 1][c],  
11                                   vi + K[i - 1][c - wi]);  
12        }  
13    return K[n][W];  
14 }
```

			c (kapasite) →						
i	v	w	0	1	2	3	4	5	6
0			0	0	0	0	0	0	0
1	5	4	0	0	0	0	5		
2	4	3	0						
3	3	2	0						
4	2	1	0						

i=1, c=4: max(alma = 0, al = 5 + K[0][0] = 5) = 5 (al)

Adım 5 / 26

DEĞİŞKENLER

i = 1

c = 4

vi = 5

wi = 4

alma = 0

al = 5

0/1 SIRT ÇANTASI

0/1 Sırt Çantası: Tabloyu Adım Adım Doldur (W = 6)

```
1 int sirtCantasi(int[] v, int[] w, int W) {
2     int n = v.length;
3     int[][] K = new int[n + 1][W + 1];
4     for (int i = 1; i <= n; i++)
5         for (int c = 1; c <= W; c++) {
6             int vi = v[i - 1], wi = w[i - 1];
7             if (wi > c) // sığmaz
8                 K[i][c] = K[i - 1][c];
9             else // alma / al
10                 K[i][c] = Math.max(K[i - 1][c],
11                                     vi + K[i - 1][c - wi]);
12         }
13     return K[n][W];
14 }
```

			c (kapasite) →						
i	v	w	0	1	2	3	4	5	6
0			0	0	0	0	0	0	0
1	5	4	0	0	0	0	5	5	
2	4	3	0						
3	3	2	0						
4	2	1	0						

$i=1, c=5: \max(\text{alma} = 0, \text{al} = 5 + K[0][1] = 5) = 5$
(al)

Adım 6 / 26

DEĞİŞKENLER

i = 1

c = 5

vi = 5

wi = 4

alma = 0

al = 5

0/1 SIRT ÇANTASI

0/1 Sırt Çantası: Tabloyu Adım Adım Doldur (W = 6)

```
1 int sirtCantasi(int[] v, int[] w, int W) {  
2     int n = v.length;  
3     int[][] K = new int[n + 1][W + 1];  
4     for (int i = 1; i <= n; i++)  
5         for (int c = 1; c <= W; c++) {  
6             int vi = v[i - 1], wi = w[i - 1];  
7             if (wi > c)                // sığmaz  
8                 K[i][c] = K[i - 1][c];  
9             else                        // alma / al  
10                K[i][c] = Math.max(K[i - 1][c],  
11                                   vi + K[i - 1][c - wi]);  
12     }  
13     return K[n][W];  
14 }
```

			c (kapasite) →						
i	v	w	0	1	2	3	4	5	6
0			0	0	0	0	0	0	0
1	5	4	0	0	0	0	5	5	5
2	4	3	0						
3	3	2	0						
4	2	1	0						

i=1, c=6: max(alma = 0, al = 5 + K[0][2] = 5) = 5 (al)

Adım 7 / 26

DEĞİŞKENLER

i = 1

c = 6

vi = 5

wi = 4

alma = 0

al = 5

0/1 SIRT ÇANTASI

0/1 Sırt Çantası: Tabloyu Adım Adım Doldur (W = 6)

```
1 int sirtCantasi(int[] v, int[] w, int W) {
2     int n = v.length;
3     int[][] K = new int[n + 1][W + 1];
4     for (int i = 1; i <= n; i++)
5         for (int c = 1; c <= W; c++) {
6             int vi = v[i - 1], wi = w[i - 1];
7             ▶ if (wi > c) // sığmaz
8                 ▶ K[i][c] = K[i - 1][c];
9             else // alma / al
10                 K[i][c] = Math.max(K[i - 1][c],
11                                     vi + K[i - 1][c - wi]);
12         }
13     return K[n][W];
14 }
```

			c (kapasite) →						
i	v	w	0	1	2	3	4	5	6
0			0	0	0	0	0	0	0
1	5	4	0	0	0	0	5	5	5
2	4	3	0	0					
3	3	2	0						
4	2	1	0						

i=2, c=1: w=3 > 1, eşya **sığmaz** → üstteki değer kopyalanır: K[2][1] = 0

Adım 8 / 26

DEĞİŞKENLER

i = 2

c = 1

vi = 4

wi = 3

alma = 0

al = -

0/1 SIRT ÇANTASI

0/1 Sırt Çantası: Tabloyu Adım Adım Doldur (W = 6)

```
1 int sirtCantasi(int[] v, int[] w, int W) {
2     int n = v.length;
3     int[][] K = new int[n + 1][W + 1];
4     for (int i = 1; i <= n; i++)
5         for (int c = 1; c <= W; c++) {
6             int vi = v[i - 1], wi = w[i - 1];
7             ▶ if (wi > c) // sığmaz
8                 K[i][c] = K[i - 1][c];
9             ▶ else // alma / al
10                K[i][c] = Math.max(K[i - 1][c],
11                                   vi + K[i - 1][c - wi]);
12        }
13    return K[n][W];
14 }
```

			c (kapasite) →						
i	v	w	0	1	2	3	4	5	6
0			0	0	0	0	0	0	0
1	5	4	0	0	0	0	5	5	5
2	4	3	0	0	0				
3	3	2	0						
4	2	1	0						

i=2, c=2: w=3 > 2, eşya **sığmaz** → üstteki değer kopyalanır: K[2][2] = 0

Adım 9 / 26

DEĞİŞKENLER

i = 2

c = 2

vi = 4

wi = 3

alma = 0

al = -

0/1 SIRT ÇANTASI

0/1 Sırt Çantası: Tabloyu Adım Adım Doldur (W = 6)

```
1 int sirtCantasi(int[] v, int[] w, int W) {
2     int n = v.length;
3     int[][] K = new int[n + 1][W + 1];
4     for (int i = 1; i <= n; i++)
5         for (int c = 1; c <= W; c++) {
6             int vi = v[i - 1], wi = w[i - 1];
7             if (wi > c) // sığmaz
8                 K[i][c] = K[i - 1][c];
9             else // alma / al
10                 K[i][c] = Math.max(K[i - 1][c],
11                                     vi + K[i - 1][c - wi]);
12         }
13     return K[n][W];
14 }
```

			c (kapasite) →						
i	v	w	0	1	2	3	4	5	6
0			0	0	0	0	0	0	0
1	5	4	0	0	0	0	5	5	5
2	4	3	0	0	0	4			
3	3	2	0						
4	2	1	0						

i=2, c=3: max(alma = 0, al = 4 + K[1][0] = 4) = 4 (al)

Adım 10 / 26

DEĞİŞKENLER

i = 2

c = 3

vi = 4

wi = 3

alma = 0

al = 4

0/1 SIRT ÇANTASI

0/1 Sırt Çantası: Tabloyu Adım Adım Doldur (W = 6)

```
1 int sirtCantasi(int[] v, int[] w, int W) {
2     int n = v.length;
3     int[][] K = new int[n + 1][W + 1];
4     for (int i = 1; i <= n; i++)
5         for (int c = 1; c <= W; c++) {
6             int vi = v[i - 1], wi = w[i - 1];
7             if (wi > c) // sığmaz
8                 K[i][c] = K[i - 1][c];
9             else // alma / al
10                 K[i][c] = Math.max(K[i - 1][c],
11                                     vi + K[i - 1][c - wi]);
12         }
13     return K[n][W];
14 }
```

			c (kapasite) →							
i	v	w	0	1	2	3	4	5	6	
0			0	0	0	0	0	0	0	
1	5	4	0	0	0	0	5	5	5	
2	4	3	0	0	0	4	5			
3	3	2	0							
4	2	1	0							

i=2, c=4: $\max(\text{alma} = 5, \text{al} = 4 + K[1][1] = 4) = 5$
(alma)

Adım 11 / 26

DEĞİŞKENLER

i = 2

c = 4

vi = 4

wi = 3

alma = 5

al = 4

0/1 SIRT ÇANTASI

0/1 Sırt Çantası: Tabloyu Adım Adım Doldur (W = 6)

```
1 int sirtCantasi(int[] v, int[] w, int W) {
2     int n = v.length;
3     int[][] K = new int[n + 1][W + 1];
4     for (int i = 1; i <= n; i++)
5         for (int c = 1; c <= W; c++) {
6             int vi = v[i - 1], wi = w[i - 1];
7             if (wi > c) // sığmaz
8                 K[i][c] = K[i - 1][c];
9             else // alma / al
10                 K[i][c] = Math.max(K[i - 1][c],
11                                     vi + K[i - 1][c - wi]);
12         }
13     return K[n][W];
14 }
```

				c (kapasite) →						
i	v	w		0	1	2	3	4	5	6
0				0	0	0	0	0	0	0
1	5	4		0	0	0	0	5	5	5
2	4	3		0	0	0	4	5	5	
3	3	2		0						
4	2	1		0						

i=2, c=5: $\max(\text{alma} = 5, \text{al} = 4 + K[1][2] = 4) = 5$
(alma)

Adım 12 / 26

DEĞİŞKENLER

i = 2

c = 5

vi = 4

wi = 3

alma = 5

al = 4

0/1 SIRT ÇANTASI

0/1 Sırt Çantası: Tabloyu Adım Adım Doldur (W = 6)

```
1 int sirtCantasi(int[] v, int[] w, int W) {
2     int n = v.length;
3     int[][] K = new int[n + 1][W + 1];
4     for (int i = 1; i <= n; i++)
5         for (int c = 1; c <= W; c++) {
6             int vi = v[i - 1], wi = w[i - 1];
7             if (wi > c) // sığmaz
8                 K[i][c] = K[i - 1][c];
9             else // alma / al
10                 K[i][c] = Math.max(K[i - 1][c],
11                                     vi + K[i - 1][c - wi]);
12         }
13     return K[n][W];
14 }
```

			c (kapasite) →						
i	v	w	0	1	2	3	4	5	6
0			0	0	0	0	0	0	0
1	5	4	0	0	0	0	5	5	5
2	4	3	0	0	0	4	5	5	5
3	3	2	0						
4	2	1	0						

i=2, c=6: $\max(\text{alma} = 5, \text{al} = 4 + K[1][3] = 4) = 5$
(alma)

Adım 13 / 26

DEĞİŞKENLER

i = 2

c = 6

vi = 4

wi = 3

alma = 5

al = 4

0/1 SIRT ÇANTASI

0/1 Sırt Çantası: Tabloyu Adım Adım Doldur (W = 6)

```
1 int sirtCantasi(int[] v, int[] w, int W) {
2     int n = v.length;
3     int[][] K = new int[n + 1][W + 1];
4     for (int i = 1; i <= n; i++)
5         for (int c = 1; c <= W; c++) {
6             int vi = v[i - 1], wi = w[i - 1];
7             ▶ if (wi > c) // sığmaz
8                 K[i][c] = K[i - 1][c];
9             ▶ else // alma / al
10                K[i][c] = Math.max(K[i - 1][c],
11                                   vi + K[i - 1][c - wi]);
12        }
13    return K[n][W];
14 }
```

			c (kapasite) →						
i	v	w	0	1	2	3	4	5	6
0			0	0	0	0	0	0	0
1	5	4	0	0	0	0	5	5	5
2	4	3	0	0	0	4	5	5	5
3	3	2	0	0					
4	2	1	0						

i=3, c=1: w=2 > 1, eşya **sığmaz** → üstteki değer kopyalanır: K[3][1] = 0

Adım 14 / 26

DEĞİŞKENLER

i = 3

c = 1

vi = 3

wi = 2

alma = 0

al = -

0/1 SIRT ÇANTASI

0/1 Sırt Çantası: Tabloyu Adım Adım Doldur (W = 6)

```
1 int sirtCantasi(int[] v, int[] w, int W) {
2     int n = v.length;
3     int[][] K = new int[n + 1][W + 1];
4     for (int i = 1; i <= n; i++)
5         for (int c = 1; c <= W; c++) {
6             int vi = v[i - 1], wi = w[i - 1];
7             if (wi > c) // sığmaz
8                 K[i][c] = K[i - 1][c];
9             else // alma / al
10                K[i][c] = Math.max(K[i - 1][c],
11                                   vi + K[i - 1][c - wi]);
12        }
13     return K[n][W];
14 }
```

			c (kapasite) →						
i	v	w	0	1	2	3	4	5	6
0			0	0	0	0	0	0	0
1	5	4	0	0	0	0	5	5	5
2	4	3	0	0	0	4	5	5	5
3	3	2	0	0	3				
4	2	1	0						

i=3, c=2: $\max(\text{alma} = 0, \text{al} = 3 + K[2][0] = 3) = 3$ (al)

Adım 15 / 26

DEĞİŞKENLER

i = 3

c = 2

vi = 3

wi = 2

alma = 0

al = 3

0/1 SIRT ÇANTASI

0/1 Sırt Çantası: Tabloyu Adım Adım Doldur (W = 6)

```
1 int sirtCantasi(int[] v, int[] w, int W) {
2     int n = v.length;
3     int[][] K = new int[n + 1][W + 1];
4     for (int i = 1; i <= n; i++)
5         for (int c = 1; c <= W; c++) {
6             int vi = v[i - 1], wi = w[i - 1];
7             if (wi > c) // sığmaz
8                 K[i][c] = K[i - 1][c];
9             else // alma / al
10                 K[i][c] = Math.max(K[i - 1][c],
11                                     vi + K[i - 1][c - wi]);
12         }
13     return K[n][W];
14 }
```

			c (kapasite) →						
i	v	w	0	1	2	3	4	5	6
0			0	0	0	0	0	0	0
1	5	4	0	0	0	0	5	5	5
2	4	3	0	0	0	4	5	5	5
3	3	2	0	0	3	4			
4	2	1	0						

i=3, c=3: $\max(\text{alma} = 4, \text{al} = 3 + K[2][1] = 3) = 4$
(alma)

Adım 16 / 26

DEĞİŞKENLER

i = 3

c = 3

vi = 3

wi = 2

alma = 4

al = 3

0/1 SIRT ÇANTASI

0/1 Sırt Çantası: Tabloyu Adım Adım Doldur (W = 6)

```
1 int sirtCantasi(int[] v, int[] w, int W) {
2     int n = v.length;
3     int[][] K = new int[n + 1][W + 1];
4     for (int i = 1; i <= n; i++)
5         for (int c = 1; c <= W; c++) {
6             int vi = v[i - 1], wi = w[i - 1];
7             if (wi > c) // sığmaz
8                 K[i][c] = K[i - 1][c];
9             else // alma / al
10                 K[i][c] = Math.max(K[i - 1][c],
11                                     vi + K[i - 1][c - wi]);
12         }
13     return K[n][W];
14 }
```

				c (kapasite) →						
i	v	w		0	1	2	3	4	5	6
0				0	0	0	0	0	0	0
1	5	4		0	0	0	0	5	5	5
2	4	3		0	0	0	4	5	5	5
3	3	2		0	0	3	4	5		
4	2	1		0						

i=3, c=4: $\max(\text{alma} = 5, \text{al} = 3 + K[2][2] = 3) = 5$
(alma)

Adım 17 / 26

DEĞİŞKENLER

i = 3

c = 4

vi = 3

wi = 2

alma = 5

al = 3

0/1 SIRT ÇANTASI

0/1 Sırt Çantası: Tabloyu Adım Adım Doldur (W = 6)

```
1 int sirtCantasi(int[] v, int[] w, int W) {  
2     int n = v.length;  
3     int[][] K = new int[n + 1][W + 1];  
4     for (int i = 1; i <= n; i++)  
5         for (int c = 1; c <= W; c++) {  
6             int vi = v[i - 1], wi = w[i - 1];  
7             if (wi > c) // sığmaz  
8                 K[i][c] = K[i - 1][c];  
9             else // alma / al  
10                K[i][c] = Math.max(K[i - 1][c],  
11                                   vi + K[i - 1][c - wi]);  
12     }  
13     return K[n][W];  
14 }
```

			c (kapasite) →						
i	v	w	0	1	2	3	4	5	6
0			0	0	0	0	0	0	0
1	5	4	0	0	0	0	5	5	5
2	4	3	0	0	0	4	5	5	5
3	3	2	0	0	3	4	5	7	
4	2	1	0						

i=3, c=5: $\max(\text{alma} = 5, \text{al} = 3 + K[2][3] = 7) = 7$ (al)

Adım 18 / 26

DEĞİŞKENLER

i = 3

c = 5

vi = 3

wi = 2

alma = 5

al = 7

0/1 SIRT ÇANTASI

0/1 Sırt Çantası: Tabloyu Adım Adım Doldur (W = 6)

```
1 int sirtCantasi(int[] v, int[] w, int W) {
2     int n = v.length;
3     int[][] K = new int[n + 1][W + 1];
4     for (int i = 1; i <= n; i++)
5         for (int c = 1; c <= W; c++) {
6             int vi = v[i - 1], wi = w[i - 1];
7             if (wi > c) // sığmaz
8                 K[i][c] = K[i - 1][c];
9             else // alma / al
10                 K[i][c] = Math.max(K[i - 1][c],
11                                     vi + K[i - 1][c - wi]);
12         }
13     return K[n][W];
14 }
```

			c (kapasite) →						
i	v	w	0	1	2	3	4	5	6
0			0	0	0	0	0	0	0
1	5	4	0	0	0	0	5	5	5
2	4	3	0	0	0	4	5	5	5
3	3	2	0	0	3	4	5	7	8
4	2	1	0						

i=3, c=6: $\max(\text{alma} = 5, \text{al} = 3 + K[2][4] = 8) = 8$
(al)

Adım 19 / 26

DEĞİŞKENLER

i = 3

c = 6

vi = 3

wi = 2

alma = 5

al = 8

0/1 SIRT ÇANTASI

0/1 Sırt Çantası: Tabloyu Adım Adım Doldur (W = 6)

```
1 int sirtCantasi(int[] v, int[] w, int W) {
2     int n = v.length;
3     int[][] K = new int[n + 1][W + 1];
4     for (int i = 1; i <= n; i++)
5         for (int c = 1; c <= W; c++) {
6             int vi = v[i - 1], wi = w[i - 1];
7             if (wi > c) // sığmaz
8                 K[i][c] = K[i - 1][c];
9             else // alma / al
10                 K[i][c] = Math.max(K[i - 1][c],
11                                     vi + K[i - 1][c - wi]);
12         }
13     return K[n][W];
14 }
```

			c (kapasite) →						
i	v	w	0	1	2	3	4	5	6
0			0	0	0	0	0	0	0
1	5	4	0	0	0	0	5	5	5
2	4	3	0	0	0	4	5	5	5
3	3	2	0	0	3	4	5	7	8
4	2	1	0	2					

i=4, c=1: $\max(\text{alma} = 0, \text{al} = 2 + K[3][0] = 2) = 2$ (al)

Adım 20 / 26

DEĞİŞKENLER

i = 4

c = 1

vi = 2

wi = 1

alma = 0

al = 2

0/1 SIRT ÇANTASI

0/1 Sırt Çantası: Tabloyu Adım Adım Doldur (W = 6)

```
1 int sirtCantasi(int[] v, int[] w, int W) {  
2     int n = v.length;  
3     int[][] K = new int[n + 1][W + 1];  
4     for (int i = 1; i <= n; i++)  
5         for (int c = 1; c <= W; c++) {  
6             int vi = v[i - 1], wi = w[i - 1];  
7             if (wi > c) // sığmaz  
8                 K[i][c] = K[i - 1][c];  
9             else // alma / al  
10                K[i][c] = Math.max(K[i - 1][c],  
11                                   vi + K[i - 1][c - wi]);  
12     }  
13     return K[n][W];  
14 }
```

			c (kapasite) →						
i	v	w	0	1	2	3	4	5	6
0			0	0	0	0	0	0	0
1	5	4	0	0	0	0	5	5	5
2	4	3	0	0	0	4	5	5	5
3	3	2	0	0	3	4	5	7	8
4	2	1	0	2	3				

$i=4, c=2: \max(\text{alma} = 3, \text{al} = 2 + K[3][1] = 2) = 3$
(alma)

Adım 21 / 26

DEĞİŞKENLER

$i = 4$

$c = 2$

$vi = 2$

$wi = 1$

$\text{alma} = 3$

$\text{al} = 2$

0/1 SIRT ÇANTASI

0/1 Sırt Çantası: Tabloyu Adım Adım Doldur (W = 6)

```
1 int sirtCantasi(int[] v, int[] w, int W) {
2     int n = v.length;
3     int[][] K = new int[n + 1][W + 1];
4     for (int i = 1; i <= n; i++)
5         for (int c = 1; c <= W; c++) {
6             int vi = v[i - 1], wi = w[i - 1];
7             if (wi > c) // sığmaz
8                 K[i][c] = K[i - 1][c];
9             else // alma / al
10                 K[i][c] = Math.max(K[i - 1][c],
11                                     vi + K[i - 1][c - wi]);
12         }
13     return K[n][W];
14 }
```

			c (kapasite) →						
i	v	w	0	1	2	3	4	5	6
0			0	0	0	0	0	0	0
1	5	4	0	0	0	0	5	5	5
2	4	3	0	0	0	4	5	5	5
3	3	2	0	0	3	4	5	7	8
4	2	1	0	2	3	5			

i=4, c=3: max(alma = 4, al = 2 + K[3][2] = 5) = 5 (al)

Adım 22 / 26

DEĞİŞKENLER

i = 4

c = 3

vi = 2

wi = 1

alma = 4

al = 5

0/1 SIRT ÇANTASI

0/1 Sırt Çantası: Tabloyu Adım Adım Doldur (W = 6)

```
1 int sirtCantasi(int[] v, int[] w, int W) {
2     int n = v.length;
3     int[][] K = new int[n + 1][W + 1];
4     for (int i = 1; i <= n; i++)
5         for (int c = 1; c <= W; c++) {
6             int vi = v[i - 1], wi = w[i - 1];
7             if (wi > c) // sığmaz
8                 K[i][c] = K[i - 1][c];
9             else // alma / al
10                 K[i][c] = Math.max(K[i - 1][c],
11                                     vi + K[i - 1][c - wi]);
12         }
13     return K[n][W];
14 }
```

			c (kapasite) →						
i	v	w	0	1	2	3	4	5	6
0			0	0	0	0	0	0	0
1	5	4	0	0	0	0	5	5	5
2	4	3	0	0	0	4	5	5	5
3	3	2	0	0	3	4	5	7	8
4	2	1	0	2	3	5	6		

i=4, c=4: max(alma = 5, al = 2 + K[3][3] = 6) = **6**
(al)

Adım 23 / 26

DEĞİŞKENLER

i = 4

c = 4

vi = 2

wi = 1

alma = 5

al = 6

0/1 SIRT ÇANTASI

0/1 Sırt Çantası: Tabloyu Adım Adım Doldur (W = 6)

```
1 int sirtCantasi(int[] v, int[] w, int W) {
2     int n = v.length;
3     int[][] K = new int[n + 1][W + 1];
4     for (int i = 1; i <= n; i++)
5         for (int c = 1; c <= W; c++) {
6             int vi = v[i - 1], wi = w[i - 1];
7             if (wi > c) // sığmaz
8                 K[i][c] = K[i - 1][c];
9             else // alma / al
10                 K[i][c] = Math.max(K[i - 1][c],
11                                     vi + K[i - 1][c - wi]);
12         }
13     return K[n][W];
14 }
```

			c (kapasite) →						
i	v	w	0	1	2	3	4	5	6
0			0	0	0	0	0	0	0
1	5	4	0	0	0	0	5	5	5
2	4	3	0	0	0	4	5	5	5
3	3	2	0	0	3	4	5	7	8
4	2	1	0	2	3	5	6	7	

i=4, c=5: $\max(\text{alma} = 7, \text{al} = 2 + K[3][4] = 7) = 7$
(alma)

Adım 24 / 26

DEĞİŞKENLER

i = 4

c = 5

vi = 2

wi = 1

alma = 7

al = 7

0/1 SIRT ÇANTASI

0/1 Sırt Çantası: Tabloyu Adım Adım Doldur (W = 6)

```
1 int sirtCantasi(int[] v, int[] w, int W) {
2     int n = v.length;
3     int[][] K = new int[n + 1][W + 1];
4     for (int i = 1; i <= n; i++)
5         for (int c = 1; c <= W; c++) {
6             int vi = v[i - 1], wi = w[i - 1];
7             if (wi > c) // sığmaz
8                 K[i][c] = K[i - 1][c];
9             else // alma / al
10                 K[i][c] = Math.max(K[i - 1][c],
11                                     vi + K[i - 1][c - wi]);
12         }
13     return K[n][W];
14 }
```

			c (kapasite) →						
i	v	w	0	1	2	3	4	5	6
0			0	0	0	0	0	0	0
1	5	4	0	0	0	0	5	5	5
2	4	3	0	0	0	4	5	5	5
3	3	2	0	0	3	4	5	7	8
4	2	1	0	2	3	5	6	7	9

i=4, c=6: max(alma = 8, al = 2 + K[3][5] = 9) = 9
(al)

Adım 25 / 26

DEĞİŞKENLER

i = 4

c = 6

vi = 2

wi = 1

alma = 8

al = 9

0/1 SIRT ÇANTASI

0/1 Sırt Çantası: Tabloyu Adım Adım Doldur (W = 6)

```
1 int sirtCantasi(int[] v, int[] w, int W) {  
2     int n = v.length;  
3     int[][] K = new int[n + 1][W + 1];  
4     for (int i = 1; i <= n; i++)  
5         for (int c = 1; c <= W; c++) {  
6             int vi = v[i - 1], wi = w[i - 1];  
7             if (wi > c) // sığmaz  
8                 K[i][c] = K[i - 1][c];  
9             else // alma / al  
10                K[i][c] = Math.max(K[i - 1][c],  
11                                   vi + K[i - 1][c - wi]);  
12        }  
13     return K[n][W];  
14 }
```

i	v	w	c (kapasite) →						
			0	1	2	3	4	5	6
0			0	0	0	0	0	0	0
1	5	4	0	0	0	0	5	5	5
2	4	3	0	0	0	4	5	5	5
3	3	2	0	0	3	4	5	7	8
4	2	1	0	2	3	5	6	7	9

Tablo doldu. Cevap $K[4][6] = 9$. Toplam $4 \times 6 = 24$ hücre, her biri $O(1)$.

Adım 26 / 26

DEĞİŞKENLER

i = 5

c = -

sonuc = 9

Geri Okuma: Hangi Eşyalar Alındı?

Kural: $K[i][c] \neq K[i-1][c]$ ise eşya i alınmıştır (değer ancak onu alarak arttı) $\rightarrow c \leftarrow c - w_i$; eşitse alınmamıştır.

			c (kapasite) $\rightarrow$						
i	v	w	0	1	2	3	4	5	6
0			0	0	0	0	0	0	0
1	5	4	0	0	0	0	5	5	5
2	4	3	0	0	0	4	5	5	5
3	3	2	0	0	3	4	5	7	8
4	2	1	0	2	3	5	6	7	9

Geri okuma $K[4][6] = 9$ hücresinden başlar. Hangi eşyalar alındı?

Geri Okuma: Hangi Eşyalar Alındı?

Kural: $K[i][c] \neq K[i-1][c]$ ise eşya i alınmıştır (değer ancak onu alarak arttı) $\rightarrow c \leftarrow c - w_i$; eşitse alınmamıştır.

			c (kapasite) $\rightarrow$						
i	v	w	0	1	2	3	4	5	6
0			0	0	0	0	0	0	0
1	5	4	0	0	0	0	5	5	5
2	4	3	0	0	0	4	5	5	5
3	3	2	0	0	3	4	5	7	8
4	2	1	0	2	3	5	6	7	9

$K[4][6] = 9 \neq K[3][6] = 8 \rightarrow$ eşya 4 **alındı** ($v=2, w=1$); $c = 6 - 1 = 5$

Geri Okuma: Hangi Eşyalar Alındı?

Kural: $K[i][c] \neq K[i-1][c]$ ise eşya i alınmıştır (değer ancak onu alarak arttı) $\rightarrow c \leftarrow c - w_i$; eşitse alınmamıştır.

			c (kapasite) $\rightarrow$						
i	v	w	0	1	2	3	4	5	6
0			0	0	0	0	0	0	0
1	5	4	0	0	0	0	5	5	5
2	4	3	0	0	0	4	5	5	5
3	3	2	0	0	3	4	5	7	8
4	2	1	0	2	3	5	6	7	9

$K[3][5] = 7 \neq K[2][5] = 5 \rightarrow$ eşya 3 **alındı** ($v=3, w=2$); $c = 5 - 2 = 3$

Gerçek Okuma: Hangi Eşyalar Alındı?

Kural: $K[i][c] \neq K[i-1][c]$ ise eşya i alınmıştır (değer ancak onu alarak arttı) $\rightarrow c \leftarrow c - w_i$; eşitse alınmamıştır.

			c (kapasite) $\rightarrow$						
i	v	w	0	1	2	3	4	5	6
0			0	0	0	0	0	0	0
1	5	4	0	0	0	0	5	5	5
2	4	3	0	0	0	4	5	5	5
3	3	2	0	0	3	4	5	7	8
4	2	1	0	2	3	5	6	7	9

$K[2][3] = 4 \neq K[1][3] = 0 \rightarrow$ eşya 2 **alındı** ($v=4, w=3$); $c = 3 - 3 = 0$

Geri Okuma: Hangi Eşyalar Alındı?

Kural: $K[i][c] \neq K[i-1][c]$ ise eşya i alınmıştır (değer ancak onu alarak arttı) $\rightarrow c \leftarrow c - w_i$; eşitse alınmamıştır.

			c (kapasite) $\rightarrow$						
i	v	w	0	1	2	3	4	5	6
0			0	0	0	0	0	0	0
1	5	4	0	0	0	0	5	5	5
2	4	3	0	0	0	4	5	5	5
3	3	2	0	0	3	4	5	7	8
4	2	1	0	2	3	5	6	7	9

$K[1][0] = K[0][0] = 0 \rightarrow$ eşya 1 **alınmadı**; c değişmez.

Geri Okuma: Hangi Eşyalar Alındı?

Kural: $K[i][c] \neq K[i-1][c]$ ise eşya i alınmıştır (değer ancak onu alarak arttı) $\rightarrow c \leftarrow c - w_i$; eşitse alınmamıştır.

			c (kapasite) $\rightarrow$						
i	v	w	0	1	2	3	4	5	6
0			0	0	0	0	0	0	0
1	5	4	0	0	0	0	5	5	5
2	4	3	0	0	0	4	5	5	5
3	3	2	0	0	3	4	5	7	8
4	2	1	0	2	3	5	6	7	9

Seçilen eşyalar: 2, 3, 4 $\rightarrow$ değer $4 + 3 + 2 = 9$, ağırlık $3 + 2 + 1 = 6 = W$.

- ```
int[] dp = new int[W + 1];
for (int i = 0; i < n; i++)
 for (int c = W; c ≥ w[i]; c--)
 dp[c] = Math.max(dp[c],
 v[i] + dp[c - w[i]]);
return dp[W];
```

# Sınırsız Sirt Çantası (Unbounded Knapsack)

- Her eşyadan **istenildiği kadar** (sınırsız sayıda) alınabilir; gerisi 0/1 ile aynıdır.
- Durum:**  $dp[i][c] = 0..i$  eşya türleri ve  $c$  kapasite ile en büyük kâr.

$$dp[i][c] = \max( dp[i-1][c] , kâr_i + dp[i][c - ağırlık_i] ) \text{ (ağırlık}_i \leq c \text{ ise)}$$

### 0/1 ile tek fark

"Alırsam" durumunda **aynı satıra** ( $i$ ) bakılır, üst satıra ( $i-1$ ) değil: eşyayı aldıktan sonra yine alabilirim.

### Örnek

Kâr = [15, 50, 60, 90], ağırlık = [1, 3, 4, 5], kapasite = 8. Tablo  $4 \times 9$ ; ilk satırda yalnızca eşya 0 (15 kâr, 1 kg) kullanılabilir.

Satır indisi 0'dan başlar ( $dp[-1][c] = 0$  kabul edilir).

## Sınırsız Sirt Çantası: Tabloyu Adım Adım Doldur

| kâr | ağır. | i | kapasite → |   |   |   |   |   |   |   |   |
|-----|-------|---|------------|---|---|---|---|---|---|---|---|
|     |       |   | 0          | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 15  | 1     | 0 | 0          |   |   |   |   |   |   |   |   |
| 50  | 3     | 1 | 0          |   |   |   |   |   |   |   |   |
| 60  | 4     | 2 | 0          |   |   |   |   |   |   |   |   |
| 90  | 5     | 3 | 0          |   |   |   |   |   |   |   |   |

Kapasite 0 sütunu 0'dır: hiçbir eşya sığmaz.

# Sınırsız Sirt Çantası: Tabloyu Adım Adım Doldur

| kâr | ağır. | i | kapasite → |    |   |   |   |   |   |   |   |
|-----|-------|---|------------|----|---|---|---|---|---|---|---|
|     |       |   | 0          | 1  | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 15  | 1     | 0 | 0          | 15 |   |   |   |   |   |   |   |
| 50  | 3     | 1 | 0          |    |   |   |   |   |   |   |   |
| 60  | 4     | 2 | 0          |    |   |   |   |   |   |   |   |
| 90  | 5     | 3 | 0          |    |   |   |   |   |   |   |   |

$i=0, c=1$ : yalnızca eşya 0 var, ağırlığı 1 sığar  $\rightarrow 15 + dp[0][0] = \mathbf{15}$ .

# Sınırsız Sirt Çantası: Tabloyu Adım Adım Doldur

|     |       |   | kapasite → |    |    |    |    |    |    |     |     |
|-----|-------|---|------------|----|----|----|----|----|----|-----|-----|
| kâr | ağır. | i | 0          | 1  | 2  | 3  | 4  | 5  | 6  | 7   | 8   |
| 15  | 1     | 0 | 0          | 15 | 30 | 45 | 60 | 75 | 90 | 105 | 120 |
| 50  | 3     | 1 | 0          |    |    |    |    |    |    |     |     |
| 60  | 4     | 2 | 0          |    |    |    |    |    |    |     |     |
| 90  | 5     | 3 | 0          |    |    |    |    |    |    |     |     |

$i=0, c=2..8$ :  $dp[0][c] = 15 + dp[0][c-1] \rightarrow$  eşya 0'dan  $c$  tane: 30, 45, ..., **120**. Aynı satır tekrar kullanıldı!

# Sınırsız Sirt Çantası: Tabloyu Adım Adım Doldur

|     |       |   | kapasite → |    |    |    |    |    |    |     |     |
|-----|-------|---|------------|----|----|----|----|----|----|-----|-----|
| kâr | ağır. | i | 0          | 1  | 2  | 3  | 4  | 5  | 6  | 7   | 8   |
| 15  | 1     | 0 | 0          | 15 | 30 | 45 | 60 | 75 | 90 | 105 | 120 |
| 50  | 3     | 1 | 0          | 15 |    |    |    |    |    |     |     |
| 60  | 4     | 2 | 0          |    |    |    |    |    |    |     |     |
| 90  | 5     | 3 | 0          |    |    |    |    |    |    |     |     |

$i=1, c=1$ : ağırlık  $3 > 1$ , sığmaz  $\rightarrow dp[1][1] = dp[0][1] = \mathbf{15}$

# Sınırsız Sirt Çantası: Tabloyu Adım Adım Doldur

|     |       |   | kapasite → |    |    |    |    |    |    |     |     |  |
|-----|-------|---|------------|----|----|----|----|----|----|-----|-----|--|
| kâr | ağır. | i | 0          | 1  | 2  | 3  | 4  | 5  | 6  | 7   | 8   |  |
| 15  | 1     | 0 | 0          | 15 | 30 | 45 | 60 | 75 | 90 | 105 | 120 |  |
| 50  | 3     | 1 | 0          | 15 | 30 |    |    |    |    |     |     |  |
| 60  | 4     | 2 | 0          |    |    |    |    |    |    |     |     |  |
| 90  | 5     | 3 | 0          |    |    |    |    |    |    |     |     |  |

$i=1, c=2$ : ağırlık  $3 > 2$ , sığmaz  $\rightarrow dp[1][2] = dp[0][2] = \mathbf{30}$

# Sınırsız Sirt Çantası: Tabloyu Adım Adım Doldur

| kâr | ağır. | i | kapasite → |    |    |    |    |    |    |     |     |
|-----|-------|---|------------|----|----|----|----|----|----|-----|-----|
|     |       |   | 0          | 1  | 2  | 3  | 4  | 5  | 6  | 7   | 8   |
| 15  | 1     | 0 | 0          | 15 | 30 | 45 | 60 | 75 | 90 | 105 | 120 |
| 50  | 3     | 1 | 0          | 15 | 30 | 50 |    |    |    |     |     |
| 60  | 4     | 2 | 0          |    |    |    |    |    |    |     |     |
| 90  | 5     | 3 | 0          |    |    |    |    |    |    |     |     |

$i=1, c=3: \max(dp[0][3] = 45, 50 + dp[1][0] = 50) = \mathbf{50}$

# Sınırsız Sirt Çantası: Tabloyu Adım Adım Doldur

|     |       |   | kapasite → |    |    |    |    |    |    |     |     |
|-----|-------|---|------------|----|----|----|----|----|----|-----|-----|
| kâr | ağır. | i | 0          | 1  | 2  | 3  | 4  | 5  | 6  | 7   | 8   |
| 15  | 1     | 0 | 0          | 15 | 30 | 45 | 60 | 75 | 90 | 105 | 120 |
| 50  | 3     | 1 | 0          | 15 | 30 | 50 | 65 |    |    |     |     |
| 60  | 4     | 2 | 0          |    |    |    |    |    |    |     |     |
| 90  | 5     | 3 | 0          |    |    |    |    |    |    |     |     |

$i=1, c=4: \max(dp[0][4] = 60, 50 + dp[1][1] = 65) = \mathbf{65}$

# Sınırsız Sirt Çantası: Tabloyu Adım Adım Doldur

|     |       |   | kapasite → |    |    |    |    |    |    |     |     |
|-----|-------|---|------------|----|----|----|----|----|----|-----|-----|
| kâr | ağır. | i | 0          | 1  | 2  | 3  | 4  | 5  | 6  | 7   | 8   |
| 15  | 1     | 0 | 0          | 15 | 30 | 45 | 60 | 75 | 90 | 105 | 120 |
| 50  | 3     | 1 | 0          | 15 | 30 | 50 | 65 | 80 |    |     |     |
| 60  | 4     | 2 | 0          |    |    |    |    |    |    |     |     |
| 90  | 5     | 3 | 0          |    |    |    |    |    |    |     |     |

$i=1, c=5: \max(dp[0][5] = 75, 50 + dp[1][2] = 80) = \mathbf{80}$

# Sınırsız Sirt Çantası: Tabloyu Adım Adım Doldur

|     |       |   | kapasite → |    |    |    |    |    |     |     |     |
|-----|-------|---|------------|----|----|----|----|----|-----|-----|-----|
| kâr | ağır. | i | 0          | 1  | 2  | 3  | 4  | 5  | 6   | 7   | 8   |
| 15  | 1     | 0 | 0          | 15 | 30 | 45 | 60 | 75 | 90  | 105 | 120 |
| 50  | 3     | 1 | 0          | 15 | 30 | 50 | 65 | 80 | 100 |     |     |
| 60  | 4     | 2 | 0          |    |    |    |    |    |     |     |     |
| 90  | 5     | 3 | 0          |    |    |    |    |    |     |     |     |

$i=1, c=6: \max(dp[0][6] = 90, 50 + dp[1][3] = 100) = \mathbf{100}$

# Sınırsız Sirt Çantası: Tabloyu Adım Adım Doldur

|     |       |   | kapasite → |    |    |    |    |    |     |     |     |
|-----|-------|---|------------|----|----|----|----|----|-----|-----|-----|
| kâr | ağır. | i | 0          | 1  | 2  | 3  | 4  | 5  | 6   | 7   | 8   |
| 15  | 1     | 0 | 0          | 15 | 30 | 45 | 60 | 75 | 90  | 105 | 120 |
| 50  | 3     | 1 | 0          | 15 | 30 | 50 | 65 | 80 | 100 | 115 |     |
| 60  | 4     | 2 | 0          |    |    |    |    |    |     |     |     |
| 90  | 5     | 3 | 0          |    |    |    |    |    |     |     |     |

$i=1, c=7: \max(dp[0][7] = 105, 50 + dp[1][4] = 115) = \mathbf{115}$

# Sınırsız Sirt Çantası: Tabloyu Adım Adım Doldur

|     |       |   | kapasite → |    |    |    |    |    |     |     |     |
|-----|-------|---|------------|----|----|----|----|----|-----|-----|-----|
| kâr | ağır. | i | 0          | 1  | 2  | 3  | 4  | 5  | 6   | 7   | 8   |
| 15  | 1     | 0 | 0          | 15 | 30 | 45 | 60 | 75 | 90  | 105 | 120 |
| 50  | 3     | 1 | 0          | 15 | 30 | 50 | 65 | 80 | 100 | 115 | 130 |
| 60  | 4     | 2 | 0          |    |    |    |    |    |     |     |     |
| 90  | 5     | 3 | 0          |    |    |    |    |    |     |     |     |

$i=1, c=8: \max(dp[0][8] = 120, 50 + dp[1][5] = 130) = \mathbf{130}$

# Sınırsız Sirt Çantası: Tabloyu Adım Adım Doldur

| kâr | ağır. | i | kapasite → |    |    |    |    |    |     |     |     |
|-----|-------|---|------------|----|----|----|----|----|-----|-----|-----|
|     |       |   | 0          | 1  | 2  | 3  | 4  | 5  | 6   | 7   | 8   |
| 15  | 1     | 0 | 0          | 15 | 30 | 45 | 60 | 75 | 90  | 105 | 120 |
| 50  | 3     | 1 | 0          | 15 | 30 | 50 | 65 | 80 | 100 | 115 | 130 |
| 60  | 4     | 2 | 0          | 15 |    |    |    |    |     |     |     |
| 90  | 5     | 3 | 0          |    |    |    |    |    |     |     |     |

$i=2, c=1$ : ağırlık 4 > 1, sığmaz →  $dp[2][1] = dp[1][1] = \mathbf{15}$

# Sınırsız Sirt Çantası: Tabloyu Adım Adım Doldur

| kâr | ağır. | i | kapasite → |    |    |    |    |    |     |     |     |
|-----|-------|---|------------|----|----|----|----|----|-----|-----|-----|
|     |       |   | 0          | 1  | 2  | 3  | 4  | 5  | 6   | 7   | 8   |
| 15  | 1     | 0 | 0          | 15 | 30 | 45 | 60 | 75 | 90  | 105 | 120 |
| 50  | 3     | 1 | 0          | 15 | 30 | 50 | 65 | 80 | 100 | 115 | 130 |
| 60  | 4     | 2 | 0          | 15 | 30 |    |    |    |     |     |     |
| 90  | 5     | 3 | 0          |    |    |    |    |    |     |     |     |

$i=2, c=2$ : ağırlık 4 > 2, sığmaz →  $dp[2][2] = dp[1][2] = \mathbf{30}$

# Sınırsız Sirt Çantası: Tabloyu Adım Adım Doldur

| kâr | ağır. | i | kapasite → |    |    |    |    |    |     |     |     |
|-----|-------|---|------------|----|----|----|----|----|-----|-----|-----|
|     |       |   | 0          | 1  | 2  | 3  | 4  | 5  | 6   | 7   | 8   |
| 15  | 1     | 0 | 0          | 15 | 30 | 45 | 60 | 75 | 90  | 105 | 120 |
| 50  | 3     | 1 | 0          | 15 | 30 | 50 | 65 | 80 | 100 | 115 | 130 |
| 60  | 4     | 2 | 0          | 15 | 30 | 50 |    |    |     |     |     |
| 90  | 5     | 3 | 0          |    |    |    |    |    |     |     |     |

$i=2, c=3$ : ağırlık 4 > 3, sığmaz →  $dp[2][3] = dp[1][3] = 50$

# Sınırsız Sirt Çantası: Tabloyu Adım Adım Doldur

| kâr | ağır. | i | kapasite → |    |    |    |    |    |     |     |     |
|-----|-------|---|------------|----|----|----|----|----|-----|-----|-----|
|     |       |   | 0          | 1  | 2  | 3  | 4  | 5  | 6   | 7   | 8   |
| 15  | 1     | 0 | 0          | 15 | 30 | 45 | 60 | 75 | 90  | 105 | 120 |
| 50  | 3     | 1 | 0          | 15 | 30 | 50 | 65 | 80 | 100 | 115 | 130 |
| 60  | 4     | 2 | 0          | 15 | 30 | 50 | 65 |    |     |     |     |
| 90  | 5     | 3 | 0          |    |    |    |    |    |     |     |     |

$i=2, c=4: \max(dp[1][4] = 65, 60 + dp[2][0] = 60) = \mathbf{65}$

# Sınırsız Sirt Çantası: Tabloyu Adım Adım Doldur

|     |       |   | kapasite → |    |    |    |    |    |     |     |     |
|-----|-------|---|------------|----|----|----|----|----|-----|-----|-----|
| kâr | ağır. | i | 0          | 1  | 2  | 3  | 4  | 5  | 6   | 7   | 8   |
| 15  | 1     | 0 | 0          | 15 | 30 | 45 | 60 | 75 | 90  | 105 | 120 |
| 50  | 3     | 1 | 0          | 15 | 30 | 50 | 65 | 80 | 100 | 115 | 130 |
| 60  | 4     | 2 | 0          | 15 | 30 | 50 | 65 | 80 |     |     |     |
| 90  | 5     | 3 | 0          |    |    |    |    |    |     |     |     |

$i=2, c=5: \max(dp[1][5] = 80, 60 + dp[2][1] = 75) = \mathbf{80}$

# Sınırsız Sirt Çantası: Tabloyu Adım Adım Doldur

| kâr | ağır. | i | kapasite → |    |    |    |    |    |     |     |     |
|-----|-------|---|------------|----|----|----|----|----|-----|-----|-----|
|     |       |   | 0          | 1  | 2  | 3  | 4  | 5  | 6   | 7   | 8   |
| 15  | 1     | 0 | 0          | 15 | 30 | 45 | 60 | 75 | 90  | 105 | 120 |
| 50  | 3     | 1 | 0          | 15 | 30 | 50 | 65 | 80 | 100 | 115 | 130 |
| 60  | 4     | 2 | 0          | 15 | 30 | 50 | 65 | 80 | 100 |     |     |
| 90  | 5     | 3 | 0          |    |    |    |    |    |     |     |     |

$i=2, c=6: \max(dp[1][6] = 100, 60 + dp[2][2] = 90) = \mathbf{100}$

# Sınırsız Sirt Çantası: Tabloyu Adım Adım Doldur

| kâr | ağır. | i | kapasite → |    |    |    |    |    |     |     |     |
|-----|-------|---|------------|----|----|----|----|----|-----|-----|-----|
|     |       |   | 0          | 1  | 2  | 3  | 4  | 5  | 6   | 7   | 8   |
| 15  | 1     | 0 | 0          | 15 | 30 | 45 | 60 | 75 | 90  | 105 | 120 |
| 50  | 3     | 1 | 0          | 15 | 30 | 50 | 65 | 80 | 100 | 115 | 130 |
| 60  | 4     | 2 | 0          | 15 | 30 | 50 | 65 | 80 | 100 | 115 |     |
| 90  | 5     | 3 | 0          |    |    |    |    |    |     |     |     |

$i=2, c=7: \max(dp[1][7] = 115, 60 + dp[2][3] = 110) = \mathbf{115}$

# Sınırsız Sirt Çantası: Tabloyu Adım Adım Doldur

| kâr | ağır. | i | kapasite → |    |    |    |    |    |     |     |     |
|-----|-------|---|------------|----|----|----|----|----|-----|-----|-----|
|     |       |   | 0          | 1  | 2  | 3  | 4  | 5  | 6   | 7   | 8   |
| 15  | 1     | 0 | 0          | 15 | 30 | 45 | 60 | 75 | 90  | 105 | 120 |
| 50  | 3     | 1 | 0          | 15 | 30 | 50 | 65 | 80 | 100 | 115 | 130 |
| 60  | 4     | 2 | 0          | 15 | 30 | 50 | 65 | 80 | 100 | 115 | 130 |
| 90  | 5     | 3 | 0          |    |    |    |    |    |     |     |     |

$i=2, c=8: \max(dp[1][8] = 130, 60 + dp[2][4] = 125) = \mathbf{130}$

# Sınırsız Sirt Çantası: Tabloyu Adım Adım Doldur

| kâr | ağır. | i | kapasite → |    |    |    |    |    |     |     |     |
|-----|-------|---|------------|----|----|----|----|----|-----|-----|-----|
|     |       |   | 0          | 1  | 2  | 3  | 4  | 5  | 6   | 7   | 8   |
| 15  | 1     | 0 | 0          | 15 | 30 | 45 | 60 | 75 | 90  | 105 | 120 |
| 50  | 3     | 1 | 0          | 15 | 30 | 50 | 65 | 80 | 100 | 115 | 130 |
| 60  | 4     | 2 | 0          | 15 | 30 | 50 | 65 | 80 | 100 | 115 | 130 |
| 90  | 5     | 3 | 0          | 15 |    |    |    |    |     |     |     |

$i=3, c=1$ : ağırlık 5 > 1, sığmaz →  $dp[3][1] = dp[2][1] = \mathbf{15}$

# Sınırsız Sirt Çantası: Tabloyu Adım Adım Doldur

| kâr | ağır. | i | kapasite → |    |    |    |    |    |     |     |     |
|-----|-------|---|------------|----|----|----|----|----|-----|-----|-----|
|     |       |   | 0          | 1  | 2  | 3  | 4  | 5  | 6   | 7   | 8   |
| 15  | 1     | 0 | 0          | 15 | 30 | 45 | 60 | 75 | 90  | 105 | 120 |
| 50  | 3     | 1 | 0          | 15 | 30 | 50 | 65 | 80 | 100 | 115 | 130 |
| 60  | 4     | 2 | 0          | 15 | 30 | 50 | 65 | 80 | 100 | 115 | 130 |
| 90  | 5     | 3 | 0          | 15 | 30 |    |    |    |     |     |     |

$i=3, c=2$ : ağırlık 5 > 2, sığmaz →  $dp[3][2] = dp[2][2] = \mathbf{30}$

# Sınırsız Sirt Çantası: Tabloyu Adım Adım Doldur

| kâr | ağır. | i | kapasite → |    |    |    |    |    |     |     |     |
|-----|-------|---|------------|----|----|----|----|----|-----|-----|-----|
|     |       |   | 0          | 1  | 2  | 3  | 4  | 5  | 6   | 7   | 8   |
| 15  | 1     | 0 | 0          | 15 | 30 | 45 | 60 | 75 | 90  | 105 | 120 |
| 50  | 3     | 1 | 0          | 15 | 30 | 50 | 65 | 80 | 100 | 115 | 130 |
| 60  | 4     | 2 | 0          | 15 | 30 | 50 | 65 | 80 | 100 | 115 | 130 |
| 90  | 5     | 3 | 0          | 15 | 30 | 50 |    |    |     |     |     |

$i=3, c=3$ : ağırlık 5 > 3, sığmaz →  $dp[3][3] = dp[2][3] = \mathbf{50}$

# Sınırsız Sirt Çantası: Tabloyu Adım Adım Doldur

| kâr | ağır. | i | kapasite → |    |    |    |    |    |     |     |     |
|-----|-------|---|------------|----|----|----|----|----|-----|-----|-----|
|     |       |   | 0          | 1  | 2  | 3  | 4  | 5  | 6   | 7   | 8   |
| 15  | 1     | 0 | 0          | 15 | 30 | 45 | 60 | 75 | 90  | 105 | 120 |
| 50  | 3     | 1 | 0          | 15 | 30 | 50 | 65 | 80 | 100 | 115 | 130 |
| 60  | 4     | 2 | 0          | 15 | 30 | 50 | 65 | 80 | 100 | 115 | 130 |
| 90  | 5     | 3 | 0          | 15 | 30 | 50 | 65 |    |     |     |     |

$i=3, c=4$ : ağırlık 5 > 4, sığmaz →  $dp[3][4] = dp[2][4] = \mathbf{65}$

# Sınırsız Sirt Çantası: Tabloyu Adım Adım Doldur

| kâr | ağır. | i | kapasite → |    |    |    |    |    |     |     |     |
|-----|-------|---|------------|----|----|----|----|----|-----|-----|-----|
|     |       |   | 0          | 1  | 2  | 3  | 4  | 5  | 6   | 7   | 8   |
| 15  | 1     | 0 | 0          | 15 | 30 | 45 | 60 | 75 | 90  | 105 | 120 |
| 50  | 3     | 1 | 0          | 15 | 30 | 50 | 65 | 80 | 100 | 115 | 130 |
| 60  | 4     | 2 | 0          | 15 | 30 | 50 | 65 | 80 | 100 | 115 | 130 |
| 90  | 5     | 3 | 0          | 15 | 30 | 50 | 65 | 90 |     |     |     |

$i=3, c=5: \max(dp[2][5] = 80, 90 + dp[3][0] = 90) = \mathbf{90}$

# Sınırsız Sirt Çantası: Tabloyu Adım Adım Doldur

| kâr | ağır. | i | kapasite → |    |    |    |    |    |     |     |     |
|-----|-------|---|------------|----|----|----|----|----|-----|-----|-----|
|     |       |   | 0          | 1  | 2  | 3  | 4  | 5  | 6   | 7   | 8   |
| 15  | 1     | 0 | 0          | 15 | 30 | 45 | 60 | 75 | 90  | 105 | 120 |
| 50  | 3     | 1 | 0          | 15 | 30 | 50 | 65 | 80 | 100 | 115 | 130 |
| 60  | 4     | 2 | 0          | 15 | 30 | 50 | 65 | 80 | 100 | 115 | 130 |
| 90  | 5     | 3 | 0          | 15 | 30 | 50 | 65 | 90 | 105 |     |     |

$i=3, c=6: \max(dp[2][6] = 100, 90 + dp[3][1] = 105) = \mathbf{105}$

# Sınırsız Sirt Çantası: Tabloyu Adım Adım Doldur

| kâr | ağır. | i | kapasite → |    |    |    |    |    |     |     |     |
|-----|-------|---|------------|----|----|----|----|----|-----|-----|-----|
|     |       |   | 0          | 1  | 2  | 3  | 4  | 5  | 6   | 7   | 8   |
| 15  | 1     | 0 | 0          | 15 | 30 | 45 | 60 | 75 | 90  | 105 | 120 |
| 50  | 3     | 1 | 0          | 15 | 30 | 50 | 65 | 80 | 100 | 115 | 130 |
| 60  | 4     | 2 | 0          | 15 | 30 | 50 | 65 | 80 | 100 | 115 | 130 |
| 90  | 5     | 3 | 0          | 15 | 30 | 50 | 65 | 90 | 105 | 120 |     |

$i=3, c=7: \max(dp[2][7] = 115, 90 + dp[3][2] = 120) = \mathbf{120}$

# Sınırsız Sirt Çantası: Tabloyu Adım Adım Doldur

| kâr | ağır. | i | kapasite → |    |    |    |    |    |     |     |     |
|-----|-------|---|------------|----|----|----|----|----|-----|-----|-----|
|     |       |   | 0          | 1  | 2  | 3  | 4  | 5  | 6   | 7   | 8   |
| 15  | 1     | 0 | 0          | 15 | 30 | 45 | 60 | 75 | 90  | 105 | 120 |
| 50  | 3     | 1 | 0          | 15 | 30 | 50 | 65 | 80 | 100 | 115 | 130 |
| 60  | 4     | 2 | 0          | 15 | 30 | 50 | 65 | 80 | 100 | 115 | 130 |
| 90  | 5     | 3 | 0          | 15 | 30 | 50 | 65 | 90 | 105 | 120 | 140 |

$i=3, c=8: \max(dp[2][8] = 130, 90 + dp[3][3] = 140) = \mathbf{140}$

# Sınırsız Sirt Çantası: Tabloyu Adım Adım Doldur

| kâr | ağır. | i | kapasite → |    |    |    |    |    |     |     |     |
|-----|-------|---|------------|----|----|----|----|----|-----|-----|-----|
|     |       |   | 0          | 1  | 2  | 3  | 4  | 5  | 6   | 7   | 8   |
| 15  | 1     | 0 | 0          | 15 | 30 | 45 | 60 | 75 | 90  | 105 | 120 |
| 50  | 3     | 1 | 0          | 15 | 30 | 50 | 65 | 80 | 100 | 115 | 130 |
| 60  | 4     | 2 | 0          | 15 | 30 | 50 | 65 | 80 | 100 | 115 | 130 |
| 90  | 5     | 3 | 0          | 15 | 30 | 50 | 65 | 90 | 105 | 120 | 140 |

Cevap  $dp[3][8] = \mathbf{140}$ :  $90 + dp[3][3] = 90 + 50 \rightarrow$  bir tane eşya 3 (90, 5 kg) + bir tane eşya 1 (50, 3 kg).

# Sınırsız Sirt Çantası: Tek Boyutlu Çözüm

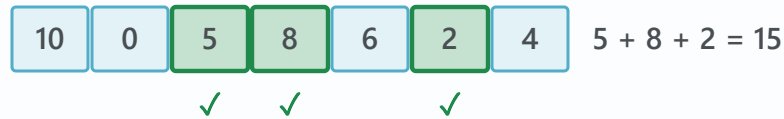
```
static int sinirsizSirtCantasi(int[] w, int[] v, int W) {
 int[] dp = new int[W + 1]; // dp[c]: c kapasiteyle en iyi
 for (int c = 1; c ≤ W; c++)
 for (int j = 0; j < w.length; j++)
 if (w[j] ≤ c)
 dp[c] = Math.max(dp[c], dp[c - w[j]] + v[j]);
 return dp[W];
}
```

- Satırlar birleştirilebilir:  $dp[c] = \text{tüm eşyalar için } \max(dp[c - w_j] + v_j)$ .
- Örnek için  $dp = [0, 15, 30, 50, 65, 90, 105, 120, 140]$  — tablonun son satırı.
- Zaman  $O(n \cdot W)$ , bellek  $O(W)$ .

### Döngü yönü her şeyi belirler

Tek boyutlu dizide kapasite **küçükten büyüğe** gezilirse  $dp[c - w]$  bu tur güncellenmiş olur → eşya tekrar alınabilir (**sınırsız**). **Büyükten küçüğe** gezilirse eski değer okunur → **0/1**.

# Alt Küme Toplamı (Subset Sum)



- Verilen tam sayılardan, toplamı tam olarak **hedef** değere eşit olan bir alt küme var mı? (Karar problemi: evet / hayır.)
- Durum:**  $dp[i][j]$  = ilk  $i$  öğeyle toplam  $j$  elde edilebilir mi? (boolean)

$dp[i][0] = \text{doğru}$  ·  $dp[0][j] = \text{yanlış}$  ( $j > 0$ )

$dp[i][j] = dp[i-1][j]$  **veya** ( $a_i \leq j$  **ve**  $dp[i-1][j - a_i]$ )

Sırt çantasının "değer = ağırlık" özel durumudur: max yerine mantıksal **veya** kullanılır.

## ALT KÜME TOPLAMI

**Alt Küme Toplamı: Tablo ( $a = 10, 0, 5, 8, 6, 2, 4$ ; hedef 15)**

|   |                | j (toplam) → |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |
|---|----------------|--------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|
| i | a <sub>i</sub> | 0            | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 0 | —              | ✓            | . | . | . | . | . | . | . | . | . | .  | .  | .  | .  | .  | .  |
| 1 | 10             | ✓            |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |
| 2 | 0              | ✓            |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |
| 3 | 5              | ✓            |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |
| 4 | 8              | ✓            |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |
| 5 | 6              | ✓            |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |
| 6 | 2              | ✓            |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |
| 7 | 4              | ✓            |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |

Sütun 0 hep ✓ (boş küme toplamı 0). Satır 0'da (öğe yok) diğer toplamlar elde edilemez (·).

## ALT KÜME TOPLAMI

### Alt Küme Toplamı: Tablo ( $a = 10, 0, 5, 8, 6, 2, 4$ ; hedef 15)

|   |       | j (toplam) → |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |
|---|-------|--------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|
| i | $a_i$ | 0            | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 0 | —     | ✓            | . | . | . | . | . | . | . | . | . | .  | .  | .  | .  | .  | .  |
| 1 | 10    | ✓            | . | . | . | . | . | . | . | . | . | ✓  | .  | .  | .  | .  | .  |
| 2 | 0     | ✓            |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |
| 3 | 5     | ✓            |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |
| 4 | 8     | ✓            |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |
| 5 | 6     | ✓            |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |
| 6 | 2     | ✓            |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |
| 7 | 4     | ✓            |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |

$i=1, a=10$ : satır  $i-1$  kopyalanır, ayrıca  $dp[i-1][j-10]$  ✓ olan her  $j$  için ✓ → yeni ulaşılan toplamalar: 10.

## ALT KÜME TOPLAMI

### Alt Küme Toplamı: Tablo ( $a = 10, 0, 5, 8, 6, 2, 4$ ; hedef 15)

| i | $a_i$ | j (toplam) → |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |
|---|-------|--------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|
|   |       | 0            | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 0 | —     | ✓            | . | . | . | . | . | . | . | . | . | .  | .  | .  | .  | .  | .  |
| 1 | 10    | ✓            | . | . | . | . | . | . | . | . | . | ✓  | .  | .  | .  | .  | .  |
| 2 | 0     | ✓            | . | . | . | . | . | . | . | . | . | ✓  | .  | .  | .  | .  | .  |
| 3 | 5     | ✓            |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |
| 4 | 8     | ✓            |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |
| 5 | 6     | ✓            |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |
| 6 | 2     | ✓            |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |
| 7 | 4     | ✓            |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |

$i=2, a=0$ : satır  $i-1$  kopyalanır, ayrıca  $dp[i-1][j-0]$  ✓ olan her  $j$  için ✓ → yeni toplam yok (0 eklemek hiçbir toplamı değiştirmez).

## ALT KÜME TOPLAMI

### Alt Küme Toplamı: Tablo ( $a = 10, 0, 5, 8, 6, 2, 4$ ; hedef 15)

| i | $a_i$ | j (toplam) → |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |
|---|-------|--------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|
|   |       | 0            | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 0 | —     | ✓            | . | . | . | . | . | . | . | . | . | .  | .  | .  | .  | .  | .  |
| 1 | 10    | ✓            | . | . | . | . | . | . | . | . | . | ✓  | .  | .  | .  | .  | .  |
| 2 | 0     | ✓            | . | . | . | . | . | . | . | . | . | ✓  | .  | .  | .  | .  | .  |
| 3 | 5     | ✓            | . | . | . | . | ✓ | . | . | . | . | ✓  | .  | .  | .  | .  | ✓  |
| 4 | 8     | ✓            |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |
| 5 | 6     | ✓            |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |
| 6 | 2     | ✓            |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |
| 7 | 4     | ✓            |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |

$i=3, a=5$ : satır  $i-1$  kopyalanır, ayrıca  $dp[i-1][j-5]$  ✓ olan her  $j$  için ✓ → yeni ulaşılan toplamalar: 5, 15.

## ALT KÜME TOPLAMI

### Alt Küme Toplamı: Tablo ( $a = 10, 0, 5, 8, 6, 2, 4$ ; hedef 15)

|   |       | j (toplam) → |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |
|---|-------|--------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|
| i | $a_i$ | 0            | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 0 | —     | ✓            | . | . | . | . | . | . | . | . | . | .  | .  | .  | .  | .  | .  |
| 1 | 10    | ✓            | . | . | . | . | . | . | . | . | . | ✓  | .  | .  | .  | .  | .  |
| 2 | 0     | ✓            | . | . | . | . | . | . | . | . | . | ✓  | .  | .  | .  | .  | .  |
| 3 | 5     | ✓            | . | . | . | . | ✓ | . | . | . | . | ✓  | .  | .  | .  | .  | ✓  |
| 4 | 8     | ✓            | . | . | . | . | ✓ | . | . | ✓ | . | ✓  | .  | .  | ✓  | .  | ✓  |
| 5 | 6     | ✓            |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |
| 6 | 2     | ✓            |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |
| 7 | 4     | ✓            |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |

$i=4, a=8$ : satır  $i-1$  kopyalanır, ayrıca  $dp[i-1][j-8]$  ✓ olan her  $j$  için ✓ → yeni ulaşılan toplamalar: 8, 13.

## ALT KÜME TOPLAMI

### Alt Küme Toplamı: Tablo ( $a = 10, 0, 5, 8, 6, 2, 4$ ; hedef 15)

|   |       | j (toplam) → |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |
|---|-------|--------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|
| i | $a_i$ | 0            | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 0 | —     | ✓            | . | . | . | . | . | . | . | . | . | .  | .  | .  | .  | .  | .  |
| 1 | 10    | ✓            | . | . | . | . | . | . | . | . | . | ✓  | .  | .  | .  | .  | .  |
| 2 | 0     | ✓            | . | . | . | . | . | . | . | . | . | ✓  | .  | .  | .  | .  | .  |
| 3 | 5     | ✓            | . | . | . | . | ✓ | . | . | . | . | ✓  | .  | .  | .  | .  | ✓  |
| 4 | 8     | ✓            | . | . | . | . | ✓ | . | . | ✓ | . | ✓  | .  | .  | ✓  | .  | ✓  |
| 5 | 6     | ✓            | . | . | . | . | ✓ | ✓ | . | ✓ | . | ✓  | ✓  | .  | ✓  | ✓  | ✓  |
| 6 | 2     | ✓            |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |
| 7 | 4     | ✓            |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |

$i=5, a=6$ : satır  $i-1$  kopyalanır, ayrıca  $dp[i-1][j-6]$  ✓ olan her  $j$  için ✓ → yeni ulaşılan toplamlar: 6, 11, 14.

## ALT KÜME TOPLAMI

### Alt Küme Toplamı: Tablo ( $a = 10, 0, 5, 8, 6, 2, 4$ ; hedef 15)

|   |       | j (toplam) → |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |
|---|-------|--------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|
| i | $a_i$ | 0            | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 0 | —     | ✓            | . | . | . | . | . | . | . | . | . | .  | .  | .  | .  | .  | .  |
| 1 | 10    | ✓            | . | . | . | . | . | . | . | . | . | ✓  | .  | .  | .  | .  | .  |
| 2 | 0     | ✓            | . | . | . | . | . | . | . | . | . | ✓  | .  | .  | .  | .  | .  |
| 3 | 5     | ✓            | . | . | . | . | ✓ | . | . | . | . | ✓  | .  | .  | .  | .  | ✓  |
| 4 | 8     | ✓            | . | . | . | . | ✓ | . | . | ✓ | . | ✓  | .  | .  | ✓  | .  | ✓  |
| 5 | 6     | ✓            | . | . | . | . | ✓ | ✓ | . | ✓ | . | ✓  | ✓  | .  | ✓  | ✓  | ✓  |
| 6 | 2     | ✓            | . | ✓ | . | . | ✓ | ✓ | ✓ | ✓ | . | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  |
| 7 | 4     | ✓            |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |

$i=6, a=2$ : satır  $i-1$  kopyalanır, ayrıca  $dp[i-1][j-2]$  ✓ olan her  $j$  için ✓ → yeni ulaşılan toplamalar: 2, 7, 12.

## ALT KÜME TOPLAMI

### Alt Küme Toplamı: Tablo ( $a = 10, 0, 5, 8, 6, 2, 4$ ; hedef 15)

|   |       | j (toplam) → |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |
|---|-------|--------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|
| i | $a_i$ | 0            | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 0 | —     | ✓            | . | . | . | . | . | . | . | . | . | .  | .  | .  | .  | .  | .  |
| 1 | 10    | ✓            | . | . | . | . | . | . | . | . | . | ✓  | .  | .  | .  | .  | .  |
| 2 | 0     | ✓            | . | . | . | . | . | . | . | . | . | ✓  | .  | .  | .  | .  | .  |
| 3 | 5     | ✓            | . | . | . | . | ✓ | . | . | . | . | ✓  | .  | .  | .  | .  | ✓  |
| 4 | 8     | ✓            | . | . | . | . | ✓ | . | . | ✓ | . | ✓  | .  | .  | ✓  | .  | ✓  |
| 5 | 6     | ✓            | . | . | . | . | ✓ | ✓ | . | ✓ | . | ✓  | ✓  | .  | ✓  | ✓  | ✓  |
| 6 | 2     | ✓            | . | ✓ | . | . | ✓ | ✓ | ✓ | ✓ | . | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  |
| 7 | 4     | ✓            | . | ✓ | . | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  |

$i=7, a=4$ : satır  $i-1$  kopyalanır, ayrıca  $dp[i-1][j-4]$  ✓ olan her  $j$  için ✓ → yeni ulaşılan toplamalar: 4, 9.

## ALT KÜME TOPLAMI

### Alt Küme Toplamı: Tablo (a = 10, 0, 5, 8, 6, 2, 4; hedef 15)

|   |                | j (toplam) → |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |
|---|----------------|--------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|
| i | a <sub>i</sub> | 0            | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 0 | —              | ✓            | . | . | . | . | . | . | . | . | . | .  | .  | .  | .  | .  | .  |
| 1 | 10             | ✓            | . | . | . | . | . | . | . | . | . | ✓  | .  | .  | .  | .  | .  |
| 2 | 0              | ✓            | . | . | . | . | . | . | . | . | . | ✓  | .  | .  | .  | .  | .  |
| 3 | 5              | ✓            | . | . | . | . | ✓ | . | . | . | . | ✓  | .  | .  | .  | .  | ✓  |
| 4 | 8              | ✓            | . | . | . | . | ✓ | . | . | ✓ | . | ✓  | .  | .  | ✓  | .  | ✓  |
| 5 | 6              | ✓            | . | . | . | . | ✓ | ✓ | . | ✓ | . | ✓  | ✓  | .  | ✓  | ✓  | ✓  |
| 6 | 2              | ✓            | . | ✓ | . | . | ✓ | ✓ | ✓ | ✓ | . | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  |
| 7 | 4              | ✓            | . | ✓ | . | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  |

$dp[7][15] = dp[6][15] = \checkmark \rightarrow 4$  olmadan da toplam 15 mümkün; seçilmez, yukarı çık.

## ALT KÜME TOPLAMI

### Alt Küme Toplamı: Tablo ( $a = 10, 0, 5, 8, 6, 2, 4$ ; hedef 15)

|   |       | j (toplam) → |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |
|---|-------|--------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|
| i | $a_i$ | 0            | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 0 | —     | ✓            | . | . | . | . | . | . | . | . | . | .  | .  | .  | .  | .  | .  |
| 1 | 10    | ✓            | . | . | . | . | . | . | . | . | . | ✓  | .  | .  | .  | .  | .  |
| 2 | 0     | ✓            | . | . | . | . | . | . | . | . | . | ✓  | .  | .  | .  | .  | .  |
| 3 | 5     | ✓            | . | . | . | . | ✓ | . | . | . | . | ✓  | .  | .  | .  | .  | ✓  |
| 4 | 8     | ✓            | . | . | . | . | ✓ | . | . | ✓ | . | ✓  | .  | .  | ✓  | .  | ✓  |
| 5 | 6     | ✓            | . | . | . | . | ✓ | ✓ | . | ✓ | . | ✓  | ✓  | .  | ✓  | ✓  | ✓  |
| 6 | 2     | ✓            | . | ✓ | . | . | ✓ | ✓ | ✓ | ✓ | . | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  |
| 7 | 4     | ✓            | . | ✓ | . | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  |

$dp[6][15] = dp[5][15] = \checkmark \rightarrow 2$  olmadan da toplam 15 mümkün; seçilmez, yukarı çık.

## ALT KÜME TOPLAMI

### Alt Küme Toplamı: Tablo (a = 10, 0, 5, 8, 6, 2, 4; hedef 15)

|   |                | j (toplam) → |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |
|---|----------------|--------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|
| i | a <sub>i</sub> | 0            | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 0 | —              | ✓            | . | . | . | . | . | . | . | . | . | .  | .  | .  | .  | .  | .  |
| 1 | 10             | ✓            | . | . | . | . | . | . | . | . | . | ✓  | .  | .  | .  | .  | .  |
| 2 | 0              | ✓            | . | . | . | . | . | . | . | . | . | ✓  | .  | .  | .  | .  | .  |
| 3 | 5              | ✓            | . | . | . | . | ✓ | . | . | . | . | ✓  | .  | .  | .  | .  | ✓  |
| 4 | 8              | ✓            | . | . | . | . | ✓ | . | . | ✓ | . | ✓  | .  | .  | ✓  | .  | ✓  |
| 5 | 6              | ✓            | . | . | . | . | ✓ | ✓ | . | ✓ | . | ✓  | ✓  | .  | ✓  | ✓  | ✓  |
| 6 | 2              | ✓            | . | ✓ | . | . | ✓ | ✓ | ✓ | ✓ | . | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  |
| 7 | 4              | ✓            | . | ✓ | . | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  |

$dp[5][15] = dp[4][15] = \checkmark \rightarrow 6$  olmadan da toplam 15 mümkün; seçilmez, yukarı çık.

## ALT KÜME TOPLAMI

### Alt Küme Toplamı: Tablo (a = 10, 0, 5, 8, 6, 2, 4; hedef 15)

|   |                | j (toplam) → |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |
|---|----------------|--------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|
| i | a <sub>i</sub> | 0            | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 0 | —              | ✓            | . | . | . | . | . | . | . | . | . | .  | .  | .  | .  | .  | .  |
| 1 | 10             | ✓            | . | . | . | . | . | . | . | . | . | ✓  | .  | .  | .  | .  | .  |
| 2 | 0              | ✓            | . | . | . | . | . | . | . | . | . | ✓  | .  | .  | .  | .  | .  |
| 3 | 5              | ✓            | . | . | . | . | ✓ | . | . | . | . | ✓  | .  | .  | .  | .  | ✓  |
| 4 | 8              | ✓            | . | . | . | . | ✓ | . | . | ✓ | . | ✓  | .  | .  | ✓  | .  | ✓  |
| 5 | 6              | ✓            | . | . | . | . | ✓ | ✓ | . | ✓ | . | ✓  | ✓  | .  | ✓  | ✓  | ✓  |
| 6 | 2              | ✓            | . | ✓ | . | . | ✓ | ✓ | ✓ | ✓ | . | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  |
| 7 | 4              | ✓            | . | ✓ | . | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  |

$dp[4][15] = dp[3][15] = \checkmark \rightarrow 8$  olmadan da toplam 15 mümkün; seçilmez, yukarı çık.

## ALT KÜME TOPLAMI

### Alt Küme Toplamı: Tablo (a = 10, 0, 5, 8, 6, 2, 4; hedef 15)

|   |                | j (toplam) → |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |
|---|----------------|--------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|
| i | a <sub>i</sub> | 0            | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 0 | —              | ✓            | . | . | . | . | . | . | . | . | . | .  | .  | .  | .  | .  | .  |
| 1 | 10             | ✓            | . | . | . | . | . | . | . | . | . | ✓  | .  | .  | .  | .  | .  |
| 2 | 0              | ✓            | . | . | . | . | . | . | . | . | . | ✓  | .  | .  | .  | .  | .  |
| 3 | 5              | ✓            | . | . | . | . | ✓ | . | . | . | . | ✓  | .  | .  | .  | .  | ✓  |
| 4 | 8              | ✓            | . | . | . | . | ✓ | . | . | ✓ | . | ✓  | .  | .  | ✓  | .  | ✓  |
| 5 | 6              | ✓            | . | . | . | . | ✓ | ✓ | . | ✓ | . | ✓  | ✓  | .  | ✓  | ✓  | ✓  |
| 6 | 2              | ✓            | . | ✓ | . | . | ✓ | ✓ | ✓ | ✓ | . | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  |
| 7 | 4              | ✓            | . | ✓ | . | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  |

dp[3][15] ✓ ama dp[2][15] . → 5 **seçildi**; j = 15 - 5 = 10

## ALT KÜME TOPLAMI

### Alt Küme Toplamı: Tablo (a = 10, 0, 5, 8, 6, 2, 4; hedef 15)

|   |                | j (toplam) → |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |
|---|----------------|--------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|
| i | a <sub>i</sub> | 0            | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 0 | —              | ✓            | . | . | . | . | . | . | . | . | . | .  | .  | .  | .  | .  | .  |
| 1 | 10             | ✓            | . | . | . | . | . | . | . | . | . | ✓  | .  | .  | .  | .  | .  |
| 2 | 0              | ✓            | . | . | . | . | . | . | . | . | . | ✓  | .  | .  | .  | .  | .  |
| 3 | 5              | ✓            | . | . | . | . | ✓ | . | . | . | . | ✓  | .  | .  | .  | .  | ✓  |
| 4 | 8              | ✓            | . | . | . | . | ✓ | . | . | ✓ | . | ✓  | .  | .  | ✓  | .  | ✓  |
| 5 | 6              | ✓            | . | . | . | . | ✓ | ✓ | . | ✓ | . | ✓  | ✓  | .  | ✓  | ✓  | ✓  |
| 6 | 2              | ✓            | . | ✓ | . | . | ✓ | ✓ | ✓ | ✓ | . | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  |
| 7 | 4              | ✓            | . | ✓ | . | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  |

$dp[2][10] = dp[1][10] = \checkmark \rightarrow 0$  olmadan da toplam 10 mümkün; seçilmez, yukarı çık.

## ALT KÜME TOPLAMI

### Alt Küme Toplamı: Tablo (a = 10, 0, 5, 8, 6, 2, 4; hedef 15)

|   |                | j (toplam) → |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |
|---|----------------|--------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|
| i | a <sub>i</sub> | 0            | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 0 | —              | ✓            | · | · | · | · | · | · | · | · | · | ·  | ·  | ·  | ·  | ·  | ·  |
| 1 | 10             | ✓            | · | · | · | · | · | · | · | · | · | ✓  | ·  | ·  | ·  | ·  | ·  |
| 2 | 0              | ✓            | · | · | · | · | · | · | · | · | · | ✓  | ·  | ·  | ·  | ·  | ·  |
| 3 | 5              | ✓            | · | · | · | · | ✓ | · | · | · | · | ✓  | ·  | ·  | ·  | ·  | ✓  |
| 4 | 8              | ✓            | · | · | · | · | ✓ | · | · | ✓ | · | ✓  | ·  | ·  | ✓  | ·  | ✓  |
| 5 | 6              | ✓            | · | · | · | · | ✓ | ✓ | · | ✓ | · | ✓  | ✓  | ·  | ✓  | ✓  | ✓  |
| 6 | 2              | ✓            | · | ✓ | · | · | ✓ | ✓ | ✓ | ✓ | · | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  |
| 7 | 4              | ✓            | · | ✓ | · | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  |

dp[1][10] ✓ ama dp[0][10] · → 10 **seçildi**; j = 10 - 10 = 0

## ALT KÜME TOPLAMI

### Alt Küme Toplamı: Tablo ( $a = 10, 0, 5, 8, 6, 2, 4$ ; hedef 15)

|   |       | j (toplam) → |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |
|---|-------|--------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|
| i | $a_i$ | 0            | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 0 | —     | ✓            | . | . | . | . | . | . | . | . | . | .  | .  | .  | .  | .  | .  |
| 1 | 10    | ✓            | . | . | . | . | . | . | . | . | . | ✓  | .  | .  | .  | .  | .  |
| 2 | 0     | ✓            | . | . | . | . | . | . | . | . | . | ✓  | .  | .  | .  | .  | .  |
| 3 | 5     | ✓            | . | . | . | . | ✓ | . | . | . | . | ✓  | .  | .  | .  | .  | ✓  |
| 4 | 8     | ✓            | . | . | . | . | ✓ | . | . | ✓ | . | ✓  | .  | .  | ✓  | .  | ✓  |
| 5 | 6     | ✓            | . | . | . | . | ✓ | ✓ | . | ✓ | . | ✓  | ✓  | .  | ✓  | ✓  | ✓  |
| 6 | 2     | ✓            | . | ✓ | . | . | ✓ | ✓ | ✓ | ✓ | . | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  |
| 7 | 4     | ✓            | . | ✓ | . | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  |

Geri okuma  $\{5, 10\}$  alt kümesini bulur:  $10 + 5 = 15$ .  $\{5, 8, 2\}$  ve  $\{5, 6, 4\}$  de geçerlidir; tablo yalnızca **birini** verir.

# Alt Küme Toplamı: Kod ve Analiz

```
boolean altKume(int[] a, int hedef) {
 int n = a.length;
 boolean[][] dp = new boolean[n + 1][hedef + 1];
 for (int i = 0; i ≤ n; i++)
 dp[i][0] = true; // boş küme
 for (int i = 1; i ≤ n; i++) {
 int x = a[i - 1];
 for (int j = 1; j ≤ hedef; j++)
 dp[i][j] = dp[i - 1][j] // x alınmaz
 || (x ≤ j && dp[i - 1][j - x]);
 }
 return dp[n][hedef];
}
```

- Zaman ve bellek:  **$O(n \cdot \text{hedef})$**  — yine sözde-polinom.
- Problem **NP-tam**dır; hedef büyükse tablo kurulamaz.
- Bu örnekte 6 farklı çözüm var: {10, 5}, {5, 8, 2}, {5, 6, 4} ve bunların 0 eklenmiş hâlleri.
- Tek boyutlu sürüm: j **büyükten küçüğe** gezilir (0/1 gibi).
- Uygulama: **eşit bölme** problemi — toplamı S olan diziyi iki eşit parçaya bölmek = hedef  $S/2$  ile alt küme toplamı.

# En Uzun Artan Alt Dizi (Longest Increasing Subsequence)

- **Alt dizi** (subsequence): öğeler sırası korunarak, **atlanarak** seçilebilir. **Alt dizgi / alt dizi parçası** (substring, subarray): öğeler **bitişik** olmalıdır.
- LIS: öğeleri **kesin artan** en uzun alt dizi.

0, 8, 4, 12, **2**, 10, **6**, 14, 1, **9**, 5, 13, 3, **11**, 7, **15** → LIS = **0, 2, 6, 9, 11, 15** (uzunluk 6)

### $O(n^2)$ DP

$dp[i] = a[i]$ 'de **biten** en uzun artan alt dizinin uzunluğu.  $dp[i] = 1 + \max\{ dp[j] : j < i, a[j] < a[i] \}$ .

### $O(n \log n)$ : aktif listeler

Her uzunluk için en küçük sonlu listeyi tut; yeni öğe ikili aramayla yerine konur.

Bu dizinin birden çok LIS'i vardır (ör. 0, 4, 6, 9, 13, 15); uzunluk her zaman tektir.

## EN UZUN ARTAN ALT DİZİ

# LIS: $O(n^2)$ Dinamik Programlama

|        | 0 | 1 | 2 | 3  | 4 | 5  | 6 | 7  | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|--------|---|---|---|----|---|----|---|----|---|---|----|----|----|----|----|----|
| a      | 0 | 8 | 4 | 12 | 2 | 10 | 6 | 14 | 1 | 9 | 5  | 13 | 3  | 11 | 7  | 15 |
| dp     |   |   |   |    |   |    |   |    |   |   |    |    |    |    |    |    |
| önceki |   |   |   |    |   |    |   |    |   |   |    |    |    |    |    |    |

Başlangıç: her öge tek başına uzunluk 1'lik artan alt dizidir.

Adım 1 / 18

Turuncu: işlenen öge · mavi:  $a[j] < a[i]$  olan adaylar · yeşil: en iyi önceki öge. Toplam  $O(n^2)$  karşılaştırma.

## LIS: $O(n^2)$ Dinamik Programlama

|        | 0 | 1 | 2 | 3  | 4 | 5  | 6 | 7  | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|--------|---|---|---|----|---|----|---|----|---|---|----|----|----|----|----|----|
| a      | 0 | 8 | 4 | 12 | 2 | 10 | 6 | 14 | 1 | 9 | 5  | 13 | 3  | 11 | 7  | 15 |
| dp     | 1 |   |   |    |   |    |   |    |   |   |    |    |    |    |    |    |
| önceki | — |   |   |    |   |    |   |    |   |   |    |    |    |    |    |    |

$i=0$ :  $a[0]=0$ , öncesinde öge yok  $\rightarrow dp[0] = 1$ .

Adım 2 / 18

Turuncu: işlenen öge · mavi:  $a[j] < a[i]$  olan adaylar · yeşil: en iyi önceki öge. Toplam  $O(n^2)$  karşılaştırma.

# LIS: $O(n^2)$ Dinamik Programlama

|        | 0 | 1 | 2 | 3  | 4 | 5  | 6 | 7  | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|--------|---|---|---|----|---|----|---|----|---|---|----|----|----|----|----|----|
| a      | 0 | 8 | 4 | 12 | 2 | 10 | 6 | 14 | 1 | 9 | 5  | 13 | 3  | 11 | 7  | 15 |
| dp     | 1 | 2 |   |    |   |    |   |    |   |   |    |    |    |    |    |    |
| önceki | — | 0 |   |    |   |    |   |    |   |   |    |    |    |    |    |    |

$i=1$ :  $a[1]=8$ ; küçük öncekiler (mavi) arasında en büyük dp,  $j=0$  ( $dp=1$ )  $\rightarrow dp[1] = 2$

Adım 3 / 18

Turuncu: işlenen öge · mavi:  $a[j] < a[i]$  olan adaylar · yeşil: en iyi önceki öge. Toplam  $O(n^2)$  karşılaştırma.

# LIS: $O(n^2)$ Dinamik Programlama

|        | 0 | 1 | 2 | 3  | 4 | 5  | 6 | 7  | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|--------|---|---|---|----|---|----|---|----|---|---|----|----|----|----|----|----|
| a      | 0 | 8 | 4 | 12 | 2 | 10 | 6 | 14 | 1 | 9 | 5  | 13 | 3  | 11 | 7  | 15 |
| dp     | 1 | 2 | 2 |    |   |    |   |    |   |   |    |    |    |    |    |    |
| önceki | — | 0 | 0 |    |   |    |   |    |   |   |    |    |    |    |    |    |

$i=2$ :  $a[2]=4$ ; küçük öncekiler (mavi) arasında en büyük dp,  $j=0$  ( $dp=1$ )  $\rightarrow dp[2] = 2$

Adım 4 / 18

Turuncu: işlenen öge · mavi:  $a[j] < a[i]$  olan adaylar · yeşil: en iyi önceki öge. Toplam  $O(n^2)$  karşılaştırma.

# LIS: $O(n^2)$ Dinamik Programlama

|        | 0 | 1 | 2 | 3  | 4 | 5  | 6 | 7  | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|--------|---|---|---|----|---|----|---|----|---|---|----|----|----|----|----|----|
| a      | 0 | 8 | 4 | 12 | 2 | 10 | 6 | 14 | 1 | 9 | 5  | 13 | 3  | 11 | 7  | 15 |
| dp     | 1 | 2 | 2 | 3  |   |    |   |    |   |   |    |    |    |    |    |    |
| önceki | — | 0 | 0 | 1  |   |    |   |    |   |   |    |    |    |    |    |    |

$i=3$ :  $a[3]=12$ ; küçük öncekiler (mavi) arasında en büyük dp,  $j=1$  ( $dp=2$ )  $\rightarrow dp[3] = 3$

Adım 5 / 18

Turuncu: işlenen öge · mavi:  $a[j] < a[i]$  olan adaylar · yeşil: en iyi önceki öge. Toplam  $O(n^2)$  karşılaştırma.

## EN UZUN ARTAN ALT DİZİ

# LIS: $O(n^2)$ Dinamik Programlama

|        | 0 | 1 | 2 | 3  | 4 | 5  | 6 | 7  | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|--------|---|---|---|----|---|----|---|----|---|---|----|----|----|----|----|----|
| a      | 0 | 8 | 4 | 12 | 2 | 10 | 6 | 14 | 1 | 9 | 5  | 13 | 3  | 11 | 7  | 15 |
| dp     | 1 | 2 | 2 | 3  | 2 |    |   |    |   |   |    |    |    |    |    |    |
| önceki | — | 0 | 0 | 1  | 0 |    |   |    |   |   |    |    |    |    |    |    |

$i=4$ :  $a[4]=2$ ; küçük öncekiler (mavi) arasında en büyük dp,  $j=0$  ( $dp=1$ )  $\rightarrow dp[4] = 2$

Adım 6 / 18

Turuncu: işlenen öge · mavi:  $a[j] < a[i]$  olan adaylar · yeşil: en iyi önceki öge. Toplam  $O(n^2)$  karşılaştırma.

# LIS: $O(n^2)$ Dinamik Programlama

|        | 0 | 1 | 2 | 3  | 4 | 5  | 6 | 7  | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|--------|---|---|---|----|---|----|---|----|---|---|----|----|----|----|----|----|
| a      | 0 | 8 | 4 | 12 | 2 | 10 | 6 | 14 | 1 | 9 | 5  | 13 | 3  | 11 | 7  | 15 |
| dp     | 1 | 2 | 2 | 3  | 2 | 3  |   |    |   |   |    |    |    |    |    |    |
| önceki | — | 0 | 0 | 1  | 0 | 1  |   |    |   |   |    |    |    |    |    |    |

$i=5$ :  $a[5]=10$ ; küçük öncekiler (mavi) arasında en büyük dp,  $j=1$  ( $dp=2$ )  $\rightarrow dp[5] = 3$

Adım 7 / 18

Turuncu: işlenen öge · mavi:  $a[j] < a[i]$  olan adaylar · yeşil: en iyi önceki öge. Toplam  $O(n^2)$  karşılaştırma.

## EN UZUN ARTAN ALT DİZİ

# LIS: $O(n^2)$ Dinamik Programlama

|        | 0 | 1 | 2 | 3  | 4 | 5  | 6 | 7  | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|--------|---|---|---|----|---|----|---|----|---|---|----|----|----|----|----|----|
| a      | 0 | 8 | 4 | 12 | 2 | 10 | 6 | 14 | 1 | 9 | 5  | 13 | 3  | 11 | 7  | 15 |
| dp     | 1 | 2 | 2 | 3  | 2 | 3  | 3 |    |   |   |    |    |    |    |    |    |
| önceki | — | 0 | 0 | 1  | 0 | 1  | 2 |    |   |   |    |    |    |    |    |    |

$i=6$ :  $a[6]=6$ ; küçük öncekiler (mavi) arasında en büyük dp,  $j=2$  ( $dp=2$ )  $\rightarrow dp[6] = 3$

Adım 8 / 18

Turuncu: işlenen öge · mavi:  $a[j] < a[i]$  olan adaylar · yeşil: en iyi önceki öge. Toplam  $O(n^2)$  karşılaştırma.

# LIS: $O(n^2)$ Dinamik Programlama

|        | 0 | 1 | 2 | 3  | 4 | 5  | 6 | 7  | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|--------|---|---|---|----|---|----|---|----|---|---|----|----|----|----|----|----|
| a      | 0 | 8 | 4 | 12 | 2 | 10 | 6 | 14 | 1 | 9 | 5  | 13 | 3  | 11 | 7  | 15 |
| dp     | 1 | 2 | 2 | 3  | 2 | 3  | 3 | 4  |   |   |    |    |    |    |    |    |
| önceki | — | 0 | 0 | 1  | 0 | 1  | 2 | 3  |   |   |    |    |    |    |    |    |

$i=7$ :  $a[7]=14$ ; küçük öncekiler (mavi) arasında en büyük dp,  $j=3$  ( $dp=3$ )  $\rightarrow dp[7] = 4$

Adım 9 / 18

Turuncu: işlenen öge · mavi:  $a[j] < a[i]$  olan adaylar · yeşil: en iyi önceki öge. Toplam  $O(n^2)$  karşılaştırma.

# LIS: $O(n^2)$ Dinamik Programlama

|        | 0 | 1 | 2 | 3  | 4 | 5  | 6 | 7  | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|--------|---|---|---|----|---|----|---|----|---|---|----|----|----|----|----|----|
| a      | 0 | 8 | 4 | 12 | 2 | 10 | 6 | 14 | 1 | 9 | 5  | 13 | 3  | 11 | 7  | 15 |
| dp     | 1 | 2 | 2 | 3  | 2 | 3  | 3 | 4  | 2 |   |    |    |    |    |    |    |
| önceki | — | 0 | 0 | 1  | 0 | 1  | 2 | 3  | 0 |   |    |    |    |    |    |    |

$i=8$ :  $a[8]=1$ ; küçük öncekiler (mavi) arasında en büyük dp,  $j=0$  ( $dp=1$ )  $\rightarrow dp[8] = 2$

Adım 10 / 18

Turuncu: işlenen öge · mavi:  $a[j] < a[i]$  olan adaylar · yeşil: en iyi önceki öge. Toplam  $O(n^2)$  karşılaştırma.

# LIS: $O(n^2)$ Dinamik Programlama

|        | 0 | 1 | 2 | 3  | 4 | 5  | 6 | 7  | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|--------|---|---|---|----|---|----|---|----|---|---|----|----|----|----|----|----|
| a      | 0 | 8 | 4 | 12 | 2 | 10 | 6 | 14 | 1 | 9 | 5  | 13 | 3  | 11 | 7  | 15 |
| dp     | 1 | 2 | 2 | 3  | 2 | 3  | 3 | 4  | 2 | 4 |    |    |    |    |    |    |
| önceki | — | 0 | 0 | 1  | 0 | 1  | 2 | 3  | 0 | 6 |    |    |    |    |    |    |

$i=9$ :  $a[9]=9$ ; küçük öncekiler (mavi) arasında en büyük dp,  $j=6$  ( $dp=3$ )  $\rightarrow dp[9] = 4$

Adım 11 / 18

Turuncu: işlenen öge · mavi:  $a[j] < a[i]$  olan adaylar · yeşil: en iyi önceki öge. Toplam  $O(n^2)$  karşılaştırma.

## EN UZUN ARTAN ALT DİZİ

# LIS: $O(n^2)$ Dinamik Programlama

|        | 0 | 1 | 2 | 3  | 4 | 5  | 6 | 7  | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|--------|---|---|---|----|---|----|---|----|---|---|----|----|----|----|----|----|
| a      | 0 | 8 | 4 | 12 | 2 | 10 | 6 | 14 | 1 | 9 | 5  | 13 | 3  | 11 | 7  | 15 |
| dp     | 1 | 2 | 2 | 3  | 2 | 3  | 3 | 4  | 2 | 4 | 3  |    |    |    |    |    |
| önceki | — | 0 | 0 | 1  | 0 | 1  | 2 | 3  | 0 | 6 | 2  |    |    |    |    |    |

$i=10$ :  $a[10]=5$ ; küçük öncekiler (mavi) arasında en büyük dp,  $j=2$  ( $dp=2$ )  $\rightarrow dp[10] = 3$

Adım 12 / 18

Turuncu: işlenen öge · mavi:  $a[j] < a[i]$  olan adaylar · yeşil: en iyi önceki öge. Toplam  $O(n^2)$  karşılaştırma.

# LIS: $O(n^2)$ Dinamik Programlama

|        | 0 | 1 | 2 | 3  | 4 | 5  | 6 | 7  | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|--------|---|---|---|----|---|----|---|----|---|---|----|----|----|----|----|----|
| a      | 0 | 8 | 4 | 12 | 2 | 10 | 6 | 14 | 1 | 9 | 5  | 13 | 3  | 11 | 7  | 15 |
| dp     | 1 | 2 | 2 | 3  | 2 | 3  | 3 | 4  | 2 | 4 | 3  | 5  |    |    |    |    |
| önceki | — | 0 | 0 | 1  | 0 | 1  | 2 | 3  | 0 | 6 | 2  | 9  |    |    |    |    |

$i=11$ :  $a[11]=13$ ; küçük öncekiler (mavi) arasında en büyük dp,  $j=9$  ( $dp=4$ )  $\rightarrow dp[11] = 5$

Adım 13 / 18

Turuncu: işlenen öge · mavi:  $a[j] < a[i]$  olan adaylar · yeşil: en iyi önceki öge. Toplam  $O(n^2)$  karşılaştırma.

# LIS: $O(n^2)$ Dinamik Programlama

|        | 0 | 1 | 2 | 3  | 4 | 5  | 6 | 7  | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|--------|---|---|---|----|---|----|---|----|---|---|----|----|----|----|----|----|
| a      | 0 | 8 | 4 | 12 | 2 | 10 | 6 | 14 | 1 | 9 | 5  | 13 | 3  | 11 | 7  | 15 |
| dp     | 1 | 2 | 2 | 3  | 2 | 3  | 3 | 4  | 2 | 4 | 3  | 5  | 3  |    |    |    |
| önceki | — | 0 | 0 | 1  | 0 | 1  | 2 | 3  | 0 | 6 | 2  | 9  | 4  |    |    |    |

$i=12$ :  $a[12]=3$ ; küçük öncekiler (mavi) arasında en büyük dp,  $j=4$  ( $dp=2$ )  $\rightarrow dp[12] = 3$

Adım 14 / 18

Turuncu: işlenen öge · mavi:  $a[j] < a[i]$  olan adaylar · yeşil: en iyi önceki öge. Toplam  $O(n^2)$  karşılaştırma.

## EN UZUN ARTAN ALT DİZİ

# LIS: $O(n^2)$ Dinamik Programlama

|        | 0 | 1 | 2 | 3  | 4 | 5  | 6 | 7  | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|--------|---|---|---|----|---|----|---|----|---|---|----|----|----|----|----|----|
| a      | 0 | 8 | 4 | 12 | 2 | 10 | 6 | 14 | 1 | 9 | 5  | 13 | 3  | 11 | 7  | 15 |
| dp     | 1 | 2 | 2 | 3  | 2 | 3  | 3 | 4  | 2 | 4 | 3  | 5  | 3  | 5  |    |    |
| önceki | — | 0 | 0 | 1  | 0 | 1  | 2 | 3  | 0 | 6 | 2  | 9  | 4  | 9  |    |    |

$i=13$ :  $a[13]=11$ ; küçük öncekiler (mavi) arasında en büyük dp,  $j=9$  ( $dp=4$ )  $\rightarrow dp[13] = 5$

Adım 15 / 18

Turuncu: işlenen öge · mavi:  $a[j] < a[i]$  olan adaylar · yeşil: en iyi önceki öge. Toplam  $O(n^2)$  karşılaştırma.

# LIS: $O(n^2)$ Dinamik Programlama

|        | 0 | 1 | 2 | 3  | 4 | 5  | 6 | 7  | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|--------|---|---|---|----|---|----|---|----|---|---|----|----|----|----|----|----|
| a      | 0 | 8 | 4 | 12 | 2 | 10 | 6 | 14 | 1 | 9 | 5  | 13 | 3  | 11 | 7  | 15 |
| dp     | 1 | 2 | 2 | 3  | 2 | 3  | 3 | 4  | 2 | 4 | 3  | 5  | 3  | 5  | 4  |    |
| önceki | — | 0 | 0 | 1  | 0 | 1  | 2 | 3  | 0 | 6 | 2  | 9  | 4  | 9  | 6  |    |

$i=14$ :  $a[14]=7$ ; küçük öncekiler (mavi) arasında en büyük dp,  $j=6$  ( $dp=3$ )  $\rightarrow dp[14] = 4$

Adım 16 / 18

Turuncu: işlenen öge · mavi:  $a[j] < a[i]$  olan adaylar · yeşil: en iyi önceki öge. Toplam  $O(n^2)$  karşılaştırma.

# LIS: $O(n^2)$ Dinamik Programlama

|        | 0 | 1 | 2 | 3  | 4 | 5  | 6 | 7  | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|--------|---|---|---|----|---|----|---|----|---|---|----|----|----|----|----|----|
| a      | 0 | 8 | 4 | 12 | 2 | 10 | 6 | 14 | 1 | 9 | 5  | 13 | 3  | 11 | 7  | 15 |
| dp     | 1 | 2 | 2 | 3  | 2 | 3  | 3 | 4  | 2 | 4 | 3  | 5  | 3  | 5  | 4  | 6  |
| önceki | — | 0 | 0 | 1  | 0 | 1  | 2 | 3  | 0 | 6 | 2  | 9  | 4  | 9  | 6  | 11 |

$i=15$ :  $a[15]=15$ ; küçük öncekiler (mavi) arasında en büyük dp,  $j=11$  ( $dp=5$ )  $\rightarrow dp[15] = 6$

Adım 17 / 18

Turuncu: işlenen öge · mavi:  $a[j] < a[i]$  olan adaylar · yeşil: en iyi önceki öge. Toplam  $O(n^2)$  karşılaştırma.

# LIS: $O(n^2)$ Dinamik Programlama

|        | 0 | 1 | 2 | 3  | 4 | 5  | 6 | 7  | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|--------|---|---|---|----|---|----|---|----|---|---|----|----|----|----|----|----|
| a      | 0 | 8 | 4 | 12 | 2 | 10 | 6 | 14 | 1 | 9 | 5  | 13 | 3  | 11 | 7  | 15 |
| dp     | 1 | 2 | 2 | 3  | 2 | 3  | 3 | 4  | 2 | 4 | 3  | 5  | 3  | 5  | 4  | 6  |
| önceki | — | 0 | 0 | 1  | 0 | 1  | 2 | 3  | 0 | 6 | 2  | 9  | 4  | 9  | 6  | 11 |

En büyük  $dp = 6$  ( $i=15$ ). **önceki** indisleri izlenir:  $15 \rightarrow 11 \rightarrow 9 \rightarrow 6 \rightarrow 2 \rightarrow 0 \Rightarrow \mathbf{0, 4, 6, 9, 13, 15}$  (eşitlikte ilk  $j$  seçilir; 0, 2, 6, 9, 11, 15 de 6 uzunlukta bir LIS'tir).

Adım 18 / 18

Turuncu: işlenen öge · mavi:  $a[j] < a[i]$  olan adaylar · yeşil: en iyi önceki öge. Toplam  $O(n^2)$  karşılaştırma.

# LIS: Aktif Listeler Yöntemi

Her uzunluk için **sonu en küçük** olan artan liste tutulur; küçük son, gelecekte uzamaya daha açıktır.

**0**, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15

**0**

Liste yok: [0] ile ilk aktif liste başlatılır.

# LIS: Aktif Listeler Yöntemi

Her uzunluk için **sonu en küçük** olan artan liste tutulur; küçük son, gelecekte uzamaya daha açıktır.

0, **8**, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15

0

**0, 8**

8 tüm sonlardan büyük: en uzun liste kopyalanıp **8** eklenir (uzunluk 2).

# LIS: Aktif Listeler Yöntemi

Her uzunluk için **sonu en küçük** olan artan liste tutulur; küçük son, gelecekte uzamaya daha açıktır.

0, 8, **4**, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15

0

**0, 4**

~~0, 8~~ pasif (atıldı)

4, sonu 0 olan listeyi uzatır  $\rightarrow [0, 4]$ ; aynı uzunluktaki  $[0, 8]$  artık gereksiz (sonu daha büyük).

# LIS: Aktif Listeler Yöntemi

Her uzunluk için **sonu en küçük** olan artan liste tutulur; küçük son, gelecekte uzamaya daha açıktır.

0, 8, 4, **12**, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15

0

0, 4

**0, 4, 12**

12 tüm sonlardan büyük: en uzun liste kopyalanıp **12** eklenir (uzunluk 3).

# LIS: Aktif Listeler Yöntemi

Her uzunluk için **sonu en küçük** olan artan liste tutulur; küçük son, gelecekte uzamaya daha açıktır.

0, 8, 4, 12, **2**, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15

0

**0, 2**

~~0, 4~~ pasif (atıldı)

0, 4, 12

2, sonu 0 olan listeyi uzatır  $\rightarrow [0, 2]$ ; aynı uzunluktaki  $[0, 4]$  artık gereksiz (sonu daha büyük).

# LIS: Aktif Listeler Yöntemi

Her uzunluk için **sonu en küçük** olan artan liste tutulur; küçük son, gelecekte uzamaya daha açıktır.

0, 8, 4, 12, 2, **10**, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15

0

0, 2

**0, 2, 10**

~~0, 4, 12~~ pasif (atıldı)

10, sonu 2 olan listeyi uzatır  $\rightarrow$  [0, 2, 10]; aynı uzunluktaki [0, 4, 12] artık gereksiz (sonu daha büyük).

# LIS: Aktif Listeler Yöntemi

Her uzunluk için **sonu en küçük** olan artan liste tutulur; küçük son, gelecekte uzamaya daha açıktır.

0, 8, 4, 12, 2, 10, **6**, 14, 1, 9, 5, 13, 3, 11, 7, 15

0

0, 2

**0, 2, 6**

~~0, 2, 10~~ pasif (atıldı)

6, sonu 2 olan listeyi uzatır  $\rightarrow [0, 2, 6]$ ; aynı uzunluktaki  $[0, 2, 10]$  artık gereksiz (sonu daha büyük).

# LIS: Aktif Listeler Yöntemi

Her uzunluk için **sonu en küçük** olan artan liste tutulur; küçük son, gelecekte uzamaya daha açıktır.

0, 8, 4, 12, 2, 10, 6, **14**, 1, 9, 5, 13, 3, 11, 7, 15

0

0, 2

0, 2, 6

**0, 2, 6, 14**

14 tüm sonlardan büyük: en uzun liste kopyalanıp **14** eklenir (uzunluk 4).

# LIS: Aktif Listeler Yöntemi

Her uzunluk için **sonu en küçük** olan artan liste tutulur; küçük son, gelecekte uzamaya daha açıktır.

0, 8, 4, 12, 2, 10, 6, 14, **1**, 9, 5, 13, 3, 11, 7, 15

0

**0, 1**

~~0, 2~~ pasif (atıldı)

0, 2, 6

0, 2, 6, 14

1, sonu 0 olan listeyi uzatır  $\rightarrow [0, 1]$ ; aynı uzunluktaki  $[0, 2]$  artık gereksiz (sonu daha büyük).

# LIS: Aktif Listeler Yöntemi

Her uzunluk için **sonu en küçük** olan artan liste tutulur; küçük son, gelecekte uzamaya daha açıktır.

0, 8, 4, 12, 2, 10, 6, 14, 1, **9**, 5, 13, 3, 11, 7, 15

0

0, 1

0, 2, 6

**0, 2, 6, 9**

~~0, 2, 6, 14~~ pasif (atıldı)

9, sonu 6 olan listeyi uzatır  $\rightarrow$  [0, 2, 6, 9]; aynı uzunluktaki [0, 2, 6, 14] artık gereksiz (sonu daha büyük).

# LIS: Aktif Listeler Yöntemi

Her uzunluk için **sonu en küçük** olan artan liste tutulur; küçük son, gelecekte uzamaya daha açıktır.

0, 8, 4, 12, 2, 10, 6, 14, 1, 9, **5**, 13, 3, 11, 7, 15

0

0, 1

**0, 1, 5**

~~0, 2, 6~~ pasif (atıldı)

0, 2, 6, 9

5, sonu 1 olan listeyi uzatır  $\rightarrow$  [0, 1, 5]; aynı uzunluktaki [0, 2, 6] artık gereksiz (sonu daha büyük).

# LIS: Aktif Listeler Yöntemi

Her uzunluk için **sonu en küçük** olan artan liste tutulur; küçük son, gelecekte uzamaya daha açıktır.

0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, **13**, 3, 11, 7, 15

0

0, 1

0, 1, 5

0, 2, 6, 9

**0, 2, 6, 9, 13**

13 tüm sonlardan büyük: en uzun liste kopyalanıp **13** eklenir (uzunluk 5).

# LIS: Aktif Listeler Yöntemi

Her uzunluk için **sonu en küçük** olan artan liste tutulur; küçük son, gelecekte uzamaya daha açıktır.

0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, **3**, 11, 7, 15

0

0, 1

**0, 1, 3**

~~0, 1, 5~~ pasif (atıldı)

0, 2, 6, 9

0, 2, 6, 9, 13

3, sonu 1 olan listeyi uzatır  $\rightarrow$  [0, 1, 3]; aynı uzunluktaki [0, 1, 5] artık gereksiz (sonu daha büyük).

# LIS: Aktif Listeler Yöntemi

Her uzunluk için **sonu en küçük** olan artan liste tutulur; küçük son, gelecekte uzamaya daha açıktır.

0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, **11**, 7, 15

0

0, 1

0, 1, 3

0, 2, 6, 9

**0, 2, 6, 9, 11**

~~0, 2, 6, 9, 13~~ pasif (atıldı)

11, sonu 9 olan listeyi uzatır → [0, 2, 6, 9, 11]; aynı uzunluktaki [0, 2, 6, 9, 13] artık gereksiz (sonu daha büyük).

# LIS: Aktif Listeler Yöntemi

Her uzunluk için **sonu en küçük** olan artan liste tutulur; küçük son, gelecekte uzamaya daha açıktır.

0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, **7**, 15

0

0, 1

0, 1, 3

**0, 1, 3, 7**

~~0, 2, 6, 9~~ pasif (atıldı)

0, 2, 6, 9, 11

7, sonu 3 olan listeyi uzatır → [0, 1, 3, 7]; aynı uzunluktaki [0, 2, 6, 9] artık gereksiz (sonu daha büyük).

# LIS: Aktif Listeler Yöntemi

Her uzunluk için **sonu en küçük** olan artan liste tutulur; küçük son, gelecekte uzamaya daha açıktır.

0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, **15**

0

0, 1

0, 1, 3

0, 1, 3, 7

0, 2, 6, 9, 11

**0, 2, 6, 9, 11, 15**

15 tüm sonlardan büyük: en uzun liste kopyalanıp **15** eklenir (uzunluk 6).

# LIS: Aktif Listeler Yöntemi

Her uzunluk için **sonu en küçük** olan artan liste tutulur; küçük son, gelecekte uzamaya daha açıktır.

0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15

0

0, 1

0, 1, 3

0, 1, 3, 7

0, 2, 6, 9, 11

**0, 2, 6, 9, 11, 15**

← en uzun artan alt dizi

Dizi bitti: en uzun aktif liste **0, 2, 6, 9, 11, 15** → LIS uzunluğu 6.

# LIS: $O(n^2)$ Kod

```
static int lis(int[] a) {
 int n = a.length, enUzun = 0;
 int[] dp = new int[n];
 for (int i = 0; i < n; i++) {
 dp[i] = 1; // tek başına
 for (int j = 0; j < i; j++)
 if (a[j] < a[i] && dp[j] + 1 > dp[i])
 dp[i] = dp[j] + 1; // uzat
 enUzun = Math.max(enUzun, dp[i]);
 }
 return enUzun;
}
```

- İç döngü,  $a[i]$ 'den küçük her önceki öğenin dizisini uzatmayı dener.
- Zaman  $O(n^2)$ , bellek  $O(n)$ .
- Diziyi kurmak için `onceki[i] = j` saklanır ya da sondan başa `dp` değerleri azalarak taranır.
- Cevap `dp[n-1]` değil, **max dp[i]**'dir: LIS herhangi bir yerde bitebilir.

# LIS: $O(n \log n)$ — Sonlar Dizisi ve İkili Arama

```
static int lisHizli(int[] a) {
 int[] son = new int[a.length];
 int uz = 0; // aktif liste sayısı
 for (int x : a) {
 int alt = 0, ust = uz; // son[alt] ≥ x ilk yer
 while (alt < ust) {
 int orta = (alt + ust) / 2;
 if (son[orta] < x) alt = orta + 1;
 else ust = orta;
 }
 son[alt] = x; // yerine koy / uzat
 if (alt == uz) uz++;
 }
 return uz;
}
```

- Aktif listelerin yalnızca **sonları** saklanır; son[] her zaman artan sıradadır, yer **ikili aramayla** bulunur →  $O(n \log n)$ .
- Örnek dizide son = [0, 1, 3, 7, 11, 15]: uzunluk doğru (6) ama bu dizi bir LIS **değildir** (7, 11'den sonra gelir).
- Diziyi kurmak için her öğenin önceki indisi ayrıca tutulur.

# En Uzun Ortak Alt Dizgi (Longest Common Substring)

- İki dizginin **bitişik** karakterlerden oluşan en uzun ortak parçası.
- Örnek: "abdca" ve "cbda" → ortak alt dizgiler "a", "b", "d", "c", "**bd**" → cevap "**bd**", uzunluk 2.
- Durum:**  $L[i][j]$  = Y'nin ilk i ve X'in ilk j karakterinin **tam sonunda biten** en uzun ortak parçanın uzunluğu.

$Y[i-1] = X[j-1]$  ise  $L[i][j] = L[i-1][j-1] + 1$

aksi hâlde  $L[i][j] = 0$  (bitişiklik bozulur, sayaç sıfırlanır)

Cevap = tablodaki **en büyük** değer (son hücre değil!)

Kaba kuvvet: X'in  $O(m^2)$  alt dizgisini Y içinde aramak →  $O(m^2 \cdot n)$ . DP:  $O(m \cdot n)$  zaman.

## Ortak Alt Dizgi: Tabloyu Adım Adım Doldur

Sütunlar X = "abdca", satırlar Y = "cbda". Turuncu: hesaplanan hücre · mavi: kullanılan çapraz hücre.

|   |   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
|   |   |   | a | b | d | c | a |
| 0 |   | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 |   |   |   |   |   |
| b | 2 | 0 |   |   |   |   |   |
| d | 3 | 0 |   |   |   |   |   |
| a | 4 | 0 |   |   |   |   |   |

Satır 0 ve sütun 0 sıfırdır: boş dizgiyle ortak parça yoktur.

## Ortak Alt Dizgi: Tabloyu Adım Adım Doldur

Sütunlar X = "abdca", satırlar Y = "cbda". Turuncu: hesaplanan hücre · mavi: kullanılan çapraz hücre.

|   |   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
|   |   |   | a | b | d | c | a |
| 0 |   | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 | 0 |   |   |   |   |
| b | 2 | 0 |   |   |   |   |   |
| d | 3 | 0 |   |   |   |   |   |
| a | 4 | 0 |   |   |   |   |   |

'c' ≠ 'a' → L[1][1] = 0 · en iyi: 0

## Ortak Alt Dizgi: Tabloyu Adım Adım Doldur

Sütunlar  $X = \text{"abdca"}$ , satırlar  $Y = \text{"cbda"}$ . Turuncu: hesaplanan hücre · mavi: kullanılan çapraz hücre.

|   |   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
|   |   |   | a | b | d | c | a |
| c | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| b | 1 | 0 | 0 | 0 |   |   |   |
| d | 2 | 0 |   |   |   |   |   |
| a | 3 | 0 |   |   |   |   |   |
|   | 4 | 0 |   |   |   |   |   |

'c'  $\neq$  'b'  $\rightarrow L[1][2] = 0$  · en iyi: 0

## Ortak Alt Dizgi: Tabloyu Adım Adım Doldur

Sütunlar  $X = \text{"abdca"}$ , satırlar  $Y = \text{"cbda"}$ . Turuncu: hesaplanan hücre · mavi: kullanılan çapraz hücre.

|   |   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
|   |   |   | a | b | d | c | a |
| 0 |   | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 | 0 | 0 | 0 |   |   |
| b | 2 | 0 |   |   |   |   |   |
| d | 3 | 0 |   |   |   |   |   |
| a | 4 | 0 |   |   |   |   |   |

'c'  $\neq$  'd'  $\rightarrow L[1][3] = 0$  · en iyi: 0

## Ortak Alt Dizgi: Tabloyu Adım Adım Doldur

Sütunlar  $X = \text{"abdca"}$ , satırlar  $Y = \text{"cbda"}$ . Turuncu: hesaplanan hücre · mavi: kullanılan çapraz hücre.

|   |   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
|   |   |   | a | b | d | c | a |
| 0 |   | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 | 0 | 0 | 0 | 1 |   |
| b | 2 | 0 |   |   |   |   |   |
| d | 3 | 0 |   |   |   |   |   |
| a | 4 | 0 |   |   |   |   |   |

$$Y[0]='c' = X[3]='c' \rightarrow L[1][4] = L[0][3] + 1 = 1 \cdot \text{en iyi: } 1$$

## Ortak Alt Dizgi: Tabloyu Adım Adım Doldur

Sütunlar  $X = \text{"abdca"}$ , satırlar  $Y = \text{"cbda"}$ . Turuncu: hesaplanan hücre · mavi: kullanılan çapraz hücre.

|   |   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
|   |   |   | a | b | d | c | a |
| 0 |   | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| b | 2 | 0 |   |   |   |   |   |
| d | 3 | 0 |   |   |   |   |   |
| a | 4 | 0 |   |   |   |   |   |

'c'  $\neq$  'a'  $\rightarrow L[1][5] = 0 \cdot \text{en iyi: } 1$

## Ortak Alt Dizgi: Tabloyu Adım Adım Doldur

Sütunlar  $X = \text{"abdca"}$ , satırlar  $Y = \text{"cbda"}$ . Turuncu: hesaplanan hücre · mavi: kullanılan çapraz hücre.

|   |   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
|   |   |   | a | b | d | c | a |
| 0 |   | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| b | 2 | 0 | 0 |   |   |   |   |
| d | 3 | 0 |   |   |   |   |   |
| a | 4 | 0 |   |   |   |   |   |

'b'  $\neq$  'a'  $\rightarrow L[2][1] = 0$  · en iyi: 1

## Ortak Alt Dizgi: Tabloyu Adım Adım Doldur

Sütunlar  $X = \text{"abdca"}$ , satırlar  $Y = \text{"cbda"}$ . Turuncu: hesaplanan hücre · mavi: kullanılan çapraz hücre.

|   |   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
|   |   |   | a | b | d | c | a |
| 0 |   | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| b | 2 | 0 | 0 | 1 |   |   |   |
| d | 3 | 0 |   |   |   |   |   |
| a | 4 | 0 |   |   |   |   |   |

$Y[1] = \text{'b'} = X[1] = \text{'b'} \rightarrow L[2][2] = L[1][1] + 1 = \mathbf{1}$  · en iyi: 1

# Ortak Alt Dizgi: Tabloyu Adım Adım Doldur

Sütunlar  $X = \text{"abdca"}$ , satırlar  $Y = \text{"cbda"}$ . Turuncu: hesaplanan hücre · mavi: kullanılan çapraz hücre.

|   |   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
|   |   |   | a | b | d | c | a |
| 0 |   | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| b | 2 | 0 | 0 | 1 | 0 |   |   |
| d | 3 | 0 |   |   |   |   |   |
| a | 4 | 0 |   |   |   |   |   |

'b'  $\neq$  'd'  $\rightarrow L[2][3] = 0$  · en iyi: 1

## Ortak Alt Dizgi: Tabloyu Adım Adım Doldur

Sütunlar  $X = \text{"abdca"}$ , satırlar  $Y = \text{"cbda"}$ . Turuncu: hesaplanan hücre · mavi: kullanılan çapraz hücre.

|   |   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
|   |   |   | a | b | d | c | a |
| 0 |   | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| b | 2 | 0 | 0 | 1 | 0 | 0 |   |
| d | 3 | 0 |   |   |   |   |   |
| a | 4 | 0 |   |   |   |   |   |

'b'  $\neq$  'c'  $\rightarrow L[2][4] = 0 \cdot \text{en iyi: } 1$

## Ortak Alt Dizgi: Tabloyu Adım Adım Doldur

Sütunlar  $X = \text{"abdca"}$ , satırlar  $Y = \text{"cbda"}$ . Turuncu: hesaplanan hücre · mavi: kullanılan çapraz hücre.

|   |   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
|   |   |   | a | b | d | c | a |
| 0 |   | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| b | 2 | 0 | 0 | 1 | 0 | 0 | 0 |
| d | 3 | 0 |   |   |   |   |   |
| a | 4 | 0 |   |   |   |   |   |

'b'  $\neq$  'a'  $\rightarrow L[2][5] = 0 \cdot \text{en iyi: } 1$

## Ortak Alt Dizgi: Tabloyu Adım Adım Doldur

Sütunlar  $X = \text{"abdca"}$ , satırlar  $Y = \text{"cbda"}$ . Turuncu: hesaplanan hücre · mavi: kullanılan çapraz hücre.

|   |   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
|   |   |   | a | b | d | c | a |
| 0 |   | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| b | 2 | 0 | 0 | 1 | 0 | 0 | 0 |
| d | 3 | 0 | 0 |   |   |   |   |
| a | 4 | 0 |   |   |   |   |   |

'd'  $\neq$  'a'  $\rightarrow L[3][1] = 0 \cdot \text{en iyi: } 1$

## Ortak Alt Dizgi: Tabloyu Adım Adım Doldur

Sütunlar  $X = \text{"abdca"}$ , satırlar  $Y = \text{"cbda"}$ . Turuncu: hesaplanan hücre · mavi: kullanılan çapraz hücre.

|   |   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
|   |   |   | a | b | d | c | a |
| 0 |   | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| b | 2 | 0 | 0 | 1 | 0 | 0 | 0 |
| d | 3 | 0 | 0 | 0 |   |   |   |
| a | 4 | 0 |   |   |   |   |   |

'd'  $\neq$  'b'  $\rightarrow L[3][2] = 0$  · en iyi: 1

## Ortak Alt Dizgi: Tabloyu Adım Adım Doldur

Sütunlar  $X = \text{"abdca"}$ , satırlar  $Y = \text{"cbda"}$ . Turuncu: hesaplanan hücre · mavi: kullanılan çapraz hücre.

|   |   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
|   |   |   | a | b | d | c | a |
| 0 |   | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| b | 2 | 0 | 0 | 1 | 0 | 0 | 0 |
| d | 3 | 0 | 0 | 0 | 2 |   |   |
| a | 4 | 0 |   |   |   |   |   |

$Y[2] = \text{'d'} = X[2] = \text{'d'} \rightarrow L[3][3] = L[2][2] + 1 = 2 \cdot \text{en iyi: } 2$

## Ortak Alt Dizgi: Tabloyu Adım Adım Doldur

Sütunlar  $X = \text{"abdca"}$ , satırlar  $Y = \text{"cbda"}$ . Turuncu: hesaplanan hücre · mavi: kullanılan çapraz hücre.

|   |   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
|   |   |   | a | b | d | c | a |
| 0 |   | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| b | 2 | 0 | 0 | 1 | 0 | 0 | 0 |
| d | 3 | 0 | 0 | 0 | 2 | 0 |   |
| a | 4 | 0 |   |   |   |   |   |

'd'  $\neq$  'c'  $\rightarrow L[3][4] = 0$  · en iyi: 2

## Ortak Alt Dizgi: Tabloyu Adım Adım Doldur

Sütunlar  $X = \text{"abdca"}$ , satırlar  $Y = \text{"cbda"}$ . Turuncu: hesaplanan hücre · mavi: kullanılan çapraz hücre.

|   |   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
|   |   |   | a | b | d | c | a |
| 0 |   | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| b | 2 | 0 | 0 | 1 | 0 | 0 | 0 |
| d | 3 | 0 | 0 | 0 | 2 | 0 | 0 |
| a | 4 | 0 |   |   |   |   |   |

'd'  $\neq$  'a'  $\rightarrow L[3][5] = 0 \cdot \text{en iyi: } 2$

## Ortak Alt Dizgi: Tabloyu Adım Adım Doldur

Sütunlar  $X = \text{"abdca"}$ , satırlar  $Y = \text{"cbda"}$ . Turuncu: hesaplanan hücre · mavi: kullanılan çapraz hücre.

|   |   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
|   |   |   | a | b | d | c | a |
| 0 |   | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| b | 2 | 0 | 0 | 1 | 0 | 0 | 0 |
| d | 3 | 0 | 0 | 0 | 2 | 0 | 0 |
| a | 4 | 0 | 1 |   |   |   |   |

$Y[3] = \text{'a'} = X[0] = \text{'a'} \rightarrow L[4][1] = L[3][0] + 1 = 1 \cdot \text{en iyi: } 2$

## Ortak Alt Dizgi: Tabloyu Adım Adım Doldur

Sütunlar  $X = \text{"abdca"}$ , satırlar  $Y = \text{"cbda"}$ . Turuncu: hesaplanan hücre · mavi: kullanılan çapraz hücre.

|   |   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
|   |   |   | a | b | d | c | a |
| 0 |   | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| b | 2 | 0 | 0 | 1 | 0 | 0 | 0 |
| d | 3 | 0 | 0 | 0 | 2 | 0 | 0 |
| a | 4 | 0 | 1 | 0 |   |   |   |

'a'  $\neq$  'b'  $\rightarrow L[4][2] = 0$  · en iyi: 2

## Ortak Alt Dizgi: Tabloyu Adım Adım Doldur

Sütunlar  $X = \text{"abdca"}$ , satırlar  $Y = \text{"cbda"}$ . Turuncu: hesaplanan hücre · mavi: kullanılan çapraz hücre.

|   |   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
|   |   |   | a | b | d | c | a |
| 0 |   | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| b | 2 | 0 | 0 | 1 | 0 | 0 | 0 |
| d | 3 | 0 | 0 | 0 | 2 | 0 | 0 |
| a | 4 | 0 | 1 | 0 | 0 |   |   |

'a'  $\neq$  'd'  $\rightarrow L[4][3] = 0$  · en iyi: 2

## Ortak Alt Dizgi: Tabloyu Adım Adım Doldur

Sütunlar  $X = \text{"abdca"}$ , satırlar  $Y = \text{"cbda"}$ . Turuncu: hesaplanan hücre · mavi: kullanılan çapraz hücre.

|   |   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
|   |   |   | a | b | d | c | a |
| 0 |   | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| b | 2 | 0 | 0 | 1 | 0 | 0 | 0 |
| d | 3 | 0 | 0 | 0 | 2 | 0 | 0 |
| a | 4 | 0 | 1 | 0 | 0 | 0 |   |

'a'  $\neq$  'c'  $\rightarrow L[4][4] = 0 \cdot \text{en iyi: } 2$

## Ortak Alt Dizgi: Tabloyu Adım Adım Doldur

Sütunlar  $X = \text{"abdca"}$ , satırlar  $Y = \text{"cbda"}$ . Turuncu: hesaplanan hücre · mavi: kullanılan çapraz hücre.

|   |   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
|   |   |   | a | b | d | c | a |
| 0 |   | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| b | 2 | 0 | 0 | 1 | 0 | 0 | 0 |
| d | 3 | 0 | 0 | 0 | 2 | 0 | 0 |
| a | 4 | 0 | 1 | 0 | 0 | 0 | 1 |

$Y[3] = 'a' = X[4] = 'a' \rightarrow L[4][5] = L[3][4] + 1 = 1 \cdot \text{en iyi: } 2$

## Ortak Alt Dizgi: Tabloyu Adım Adım Doldur

Sütunlar  $X = \text{"abdca"}$ , satırlar  $Y = \text{"cbda"}$ . Turuncu: hesaplanan hücre · mavi: kullanılan çapraz hücre.

|   |   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
|   |   |   | a | b | d | c | a |
| 0 |   | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| b | 2 | 0 | 0 | 1 | 0 | 0 | 0 |
| d | 3 | 0 | 0 | 0 | 2 | 0 | 0 |
| a | 4 | 0 | 1 | 0 | 0 | 0 | 1 |

En büyük değer  $L[3][3] = 2$ ; çapraz geriye 2 adım:  $X[1..2] = \text{"bd"}$ . Son hücre  $L[4][5] = 1$  cevap değildir.

# En Uzun Ortak Alt Dizi (Longest Common Subsequence)

- İki dizinin, sırası korunarak (atlamalı) seçilebilen en uzun ortak alt dizisi.
- Örnek: "abdca" ve "cbda" → "bda", uzunluk 3 (ortak alt dizgi yalnızca 2 idi).
- Kullanım: `diff` / git karşılaştırması, DNA dizilerinin benzerliği, intihal tespiti.

$Y[i-1] = X[j-1]$  ise  $L[i][j] = L[i-1][j-1] + 1$  (ortak karakteri al)  
aksi hâlde  $L[i][j] = \max( L[i-1][j] , L[i][j-1] )$  (birini atla)  
Cevap = **L[n][m]** (son hücre)

|                             | Eşleşme    | Eşleşmeme            | Cevap              |
|-----------------------------|------------|----------------------|--------------------|
| <b>Ortak alt dizgi</b>      | çapraz + 1 | <b>0</b>             | tablonun maksimumu |
| <b>Ortak alt dizi (LCS)</b> | çapraz + 1 | <b>max(üst, sol)</b> | son hücre          |

# LCS: Kodu Adım Adım İzle

```
1 static int lcs(String x, String y) {
2 int m = x.length(), n = y.length();
3 int[][] L = new int[n + 1][m + 1];
4 for (int i = 1; i <= n; i++)
5 for (int j = 1; j <= m; j++)
6 if (y.charAt(i - 1) == x.charAt(j - 1))
7 L[i][j] = L[i - 1][j - 1] + 1;
8 else
9 L[i][j] = Math.max(L[i - 1][j],
10 L[i][j - 1]);
11 return L[n][m];
12 }
```

|   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
|   |   | a | b | d | c | a |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 |   |   |   |   |
| b | 2 | 0 |   |   |   |   |
| d | 3 | 0 |   |   |   |   |
| a | 4 | 0 |   |   |   |   |

L dizisi  $(n+1) \times (m+1) = 5 \times 6$ ; Java'da başlangıçta tüm hücreler 0.

Adım 1 / 22

## DEĞİŞKENLER

x = "abdca"

y = "cbda"

i = -

j = -

# LCS: Kodu Adım Adım İzle

```
1 static int lcs(String x, String y) {
2 int m = x.length(), n = y.length();
3 int[][] L = new int[n + 1][m + 1];
4 for (int i = 1; i <= n; i++)
5 for (int j = 1; j <= m; j++)
6 if (y.charAt(i - 1) == x.charAt(j - 1))
7 L[i][j] = L[i - 1][j - 1] + 1;
8 else
9 L[i][j] = Math.max(L[i - 1][j],
10 L[i][j - 1]);
11 return L[n][m];
12 }
```

|   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
|   |   | a | b | d | c | a |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 | 0 |   |   |   |
| b | 2 | 0 |   |   |   |   |
| d | 3 | 0 |   |   |   |   |
| a | 4 | 0 |   |   |   |   |

'c' ≠ 'a' → max(üst 0, sol 0) = 0

Adım 2 / 22

## DEĞİŞKENLER

i = 1

j = 1

y[i-1] = 'c'

x[j-1] = 'a'

L[i][j] = 0

# LCS: Kodu Adım Adım İzle

```
1 static int lcs(String x, String y) {
2 int m = x.length(), n = y.length();
3 int[][] L = new int[n + 1][m + 1];
4 for (int i = 1; i <= n; i++)
5 for (int j = 1; j <= m; j++)
6 if (y.charAt(i - 1) == x.charAt(j - 1))
7 L[i][j] = L[i - 1][j - 1] + 1;
8 else
9 L[i][j] = Math.max(L[i - 1][j],
10 L[i][j - 1]);
11 return L[n][m];
12 }
```

|   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
|   |   | a | b | d | c | a |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 | 0 |   |   |   |
| b | 2 | 0 |   |   |   |   |
| d | 3 | 0 |   |   |   |   |
| a | 4 | 0 |   |   |   |   |

'c' ≠ 'b' → max(üst 0, sol 0) = 0

Adım 3 / 22

## DEĞİŞKENLER

i = 1

j = 2

y[i-1] = 'c'

x[j-1] = 'b'

L[i][j] = 0

## LCS: Kodu Adım Adım İzle

```

1 static int lcs(String x, String y) {
2 int m = x.length(), n = y.length();
3 int[][] L = new int[n + 1][m + 1];
4 for (int i = 1; i <= n; i++)
5 for (int j = 1; j <= m; j++)
6 if (y.charAt(i - 1) == x.charAt(j - 1))
7 L[i][j] = L[i - 1][j - 1] + 1;
8 else
9 L[i][j] = Math.max(L[i - 1][j],
10 L[i][j - 1]);
11 return L[n][m];
12 }

```

|   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
|   |   | a | b | d | c | a |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 | 0 | 0 |   |   |
| b | 2 | 0 |   |   |   |   |
| d | 3 | 0 |   |   |   |   |
| a | 4 | 0 |   |   |   |   |

'c' ≠ 'd' → max(üst 0, sol 0) = 0

Adım 4 / 22

## DEĞİŞKENLER

i = 1

j = 3

y[i-1] = 'c'

x[j-1] = 'd'

L[i][j] = 0

## LCS: Kodu Adım Adım İzle

```

1 static int lcs(String x, String y) {
2 int m = x.length(), n = y.length();
3 int[][] L = new int[n + 1][m + 1];
4 for (int i = 1; i <= n; i++)
5 for (int j = 1; j <= m; j++)
6 ▶ if (y.charAt(i - 1) == x.charAt(j - 1))
7 ▶ L[i][j] = L[i - 1][j - 1] + 1;
8 else
9 L[i][j] = Math.max(L[i - 1][j],
10 L[i][j - 1]);
11 return L[n][m];
12 }

```

|   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
|   |   | a | b | d | c | a |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 | 0 | 0 | 1 |   |
| b | 2 | 0 |   |   |   |   |
| d | 3 | 0 |   |   |   |   |
| a | 4 | 0 |   |   |   |   |

'c' = 'c' → çapraz + 1:  $L[1][4] = 0 + 1 = 1$

Adım 5 / 22

## DEĞİŞKENLER

i = 1

j = 4

y[i-1] = 'c'

x[j-1] = 'c'

L[i][j] = 1

## LCS: Kodu Adım Adım İzle

```

1 static int lcs(String x, String y) {
2 int m = x.length(), n = y.length();
3 int[][] L = new int[n + 1][m + 1];
4 for (int i = 1; i <= n; i++)
5 for (int j = 1; j <= m; j++)
6 if (y.charAt(i - 1) == x.charAt(j - 1))
7 L[i][j] = L[i - 1][j - 1] + 1;
8 else
9 L[i][j] = Math.max(L[i - 1][j],
10 L[i][j - 1]);
11 return L[n][m];
12 }

```

|   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
|   |   | a | b | d | c | a |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 | 0 | 0 | 1 | 1 |
| b | 2 | 0 |   |   |   |   |
| d | 3 | 0 |   |   |   |   |
| a | 4 | 0 |   |   |   |   |

'c' ≠ 'a' → max(üst 0, sol 1) = 1

Adım 6 / 22

## DEĞİŞKENLER

i = 1

j = 5

y[i-1] = 'c'

x[j-1] = 'a'

L[i][j] = 1

## LCS: Kodu Adım Adım İzle

```

1 static int lcs(String x, String y) {
2 int m = x.length(), n = y.length();
3 int[][] L = new int[n + 1][m + 1];
4 for (int i = 1; i <= n; i++)
5 for (int j = 1; j <= m; j++)
6 if (y.charAt(i - 1) == x.charAt(j - 1))
7 L[i][j] = L[i - 1][j - 1] + 1;
8 else
9 L[i][j] = Math.max(L[i - 1][j],
10 L[i][j - 1]);
11 return L[n][m];
12 }

```

|   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
|   |   | a | b | d | c | a |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 | 0 | 0 | 1 | 1 |
| b | 2 | 0 | 0 |   |   |   |
| d | 3 | 0 |   |   |   |   |
| a | 4 | 0 |   |   |   |   |

'b' ≠ 'a' → max(üst 0, sol 0) = 0

Adım 7 / 22

## DEĞİŞKENLER

i = 2

j = 1

y[i-1] = 'b'

x[j-1] = 'a'

L[i][j] = 0

# LCS: Kodu Adım Adım İzle

```
1 static int lcs(String x, String y) {
2 int m = x.length(), n = y.length();
3 int[][] L = new int[n + 1][m + 1];
4 for (int i = 1; i <= n; i++)
5 for (int j = 1; j <= m; j++)
6 ▶ if (y.charAt(i - 1) == x.charAt(j - 1))
7 ▶ L[i][j] = L[i - 1][j - 1] + 1;
8 else
9 L[i][j] = Math.max(L[i - 1][j],
10 L[i][j - 1]);
11 return L[n][m];
12 }
```

|   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
|   |   | a | b | d | c | a |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 | 0 | 0 | 1 | 1 |
| b | 2 | 0 | 0 | 1 |   |   |
| d | 3 | 0 |   |   |   |   |
| a | 4 | 0 |   |   |   |   |

'b' = 'b' → çapraz + 1:  $L[2][2] = 0 + 1 = 1$

Adım 8 / 22

## DEĞİŞKENLER

i = 2

j = 2

y[i-1] = 'b'

x[j-1] = 'b'

L[i][j] = 1

## LCS: Kodu Adım Adım İzle

```

1 static int lcs(String x, String y) {
2 int m = x.length(), n = y.length();
3 int[][] L = new int[n + 1][m + 1];
4 for (int i = 1; i <= n; i++)
5 for (int j = 1; j <= m; j++)
6 if (y.charAt(i - 1) == x.charAt(j - 1))
7 L[i][j] = L[i - 1][j - 1] + 1;
8 else
9 L[i][j] = Math.max(L[i - 1][j],
10 L[i][j - 1]);
11 return L[n][m];
12 }

```

|   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
|   |   | a | b | d | c | a |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 | 0 | 0 | 1 | 1 |
| b | 2 | 0 | 0 | 1 | 1 |   |
| d | 3 | 0 |   |   |   |   |
| a | 4 | 0 |   |   |   |   |

'b' ≠ 'd' → max(üst 0, sol 1) = 1

Adım 9 / 22

## DEĞİŞKENLER

i = 2

j = 3

y[i-1] = 'b'

x[j-1] = 'd'

L[i][j] = 1

## LCS: Kodu Adım Adım İzle

```

1 static int lcs(String x, String y) {
2 int m = x.length(), n = y.length();
3 int[][] L = new int[n + 1][m + 1];
4 for (int i = 1; i <= n; i++)
5 for (int j = 1; j <= m; j++)
6 if (y.charAt(i - 1) == x.charAt(j - 1))
7 L[i][j] = L[i - 1][j - 1] + 1;
8 else
9 L[i][j] = Math.max(L[i - 1][j],
10 L[i][j - 1]);
11 return L[n][m];
12 }

```

|   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
|   |   | a | b | d | c | a |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 | 0 | 0 | 1 | 1 |
| b | 2 | 0 | 0 | 1 | 1 |   |
| d | 3 | 0 |   |   |   |   |
| a | 4 | 0 |   |   |   |   |

'b' ≠ 'c' → max(üst 1, sol 1) = 1

Adım 10 / 22

## DEĞİŞKENLER

i = 2

j = 4

y[i-1] = 'b'

x[j-1] = 'c'

L[i][j] = 1

## LCS: Kodu Adım Adım İzle

```

1 static int lcs(String x, String y) {
2 int m = x.length(), n = y.length();
3 int[][] L = new int[n + 1][m + 1];
4 for (int i = 1; i <= n; i++)
5 for (int j = 1; j <= m; j++)
6 if (y.charAt(i - 1) == x.charAt(j - 1))
7 L[i][j] = L[i - 1][j - 1] + 1;
8 else
9 L[i][j] = Math.max(L[i - 1][j],
10 L[i][j - 1]);
11 return L[n][m];
12 }

```

|   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
|   |   | a | b | d | c | a |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 | 0 | 0 | 1 | 1 |
| b | 2 | 0 | 0 | 1 | 1 | 1 |
| d | 3 |   |   |   |   |   |
| a | 4 |   |   |   |   |   |

'b' ≠ 'a' → max(üst 1, sol 1) = 1

Adım 11 / 22

## DEĞİŞKENLER

i = 2

j = 5

y[i-1] = 'b'

x[j-1] = 'a'

L[i][j] = 1

## LCS: Kodu Adım Adım İzle

```

1 static int lcs(String x, String y) {
2 int m = x.length(), n = y.length();
3 int[][] L = new int[n + 1][m + 1];
4 for (int i = 1; i <= n; i++)
5 for (int j = 1; j <= m; j++)
6 if (y.charAt(i - 1) == x.charAt(j - 1))
7 L[i][j] = L[i - 1][j - 1] + 1;
8 else
9 L[i][j] = Math.max(L[i - 1][j],
10 L[i][j - 1]);
11 return L[n][m];
12 }

```

|   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
|   |   | a | b | d | c | a |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 | 0 | 0 | 1 | 1 |
| b | 2 | 0 | 0 | 1 | 1 | 1 |
| d | 3 | 0 | 0 |   |   |   |
| a | 4 | 0 |   |   |   |   |

'd' ≠ 'a' → max(üst 0, sol 0) = 0

Adım 12 / 22

## DEĞİŞKENLER

i = 3

j = 1

y[i-1] = 'd'

x[j-1] = 'a'

L[i][j] = 0

## LCS: Kodu Adım Adım İzle

```

1 static int lcs(String x, String y) {
2 int m = x.length(), n = y.length();
3 int[][] L = new int[n + 1][m + 1];
4 for (int i = 1; i <= n; i++)
5 for (int j = 1; j <= m; j++)
6 if (y.charAt(i - 1) == x.charAt(j - 1))
7 L[i][j] = L[i - 1][j - 1] + 1;
8 else
9 L[i][j] = Math.max(L[i - 1][j],
10 L[i][j - 1]);
11 return L[n][m];
12 }

```

|   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
|   |   | a | b | d | c | a |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 | 0 | 0 | 1 | 1 |
| b | 2 | 0 | 0 | 1 | 1 | 1 |
| d | 3 | 0 | 0 | 1 |   |   |
| a | 4 | 0 |   |   |   |   |

'd' ≠ 'b' → max(üst 1, sol 0) = 1

Adım 13 / 22

## DEĞİŞKENLER

i = 3

j = 2

y[i-1] = 'd'

x[j-1] = 'b'

L[i][j] = 1

# LCS: Kodu Adım Adım İzle

```
1 static int lcs(String x, String y) {
2 int m = x.length(), n = y.length();
3 int[][] L = new int[n + 1][m + 1];
4 for (int i = 1; i <= n; i++)
5 for (int j = 1; j <= m; j++)
6 ▶ if (y.charAt(i - 1) == x.charAt(j - 1))
7 ▶ L[i][j] = L[i - 1][j - 1] + 1;
8 else
9 L[i][j] = Math.max(L[i - 1][j],
10 L[i][j - 1]);
11 return L[n][m];
12 }
```

|   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
|   |   | a | b | d | c | a |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 | 0 | 0 | 1 | 1 |
| b | 2 | 0 | 0 | 1 | 1 | 1 |
| d | 3 | 0 | 0 | 1 | 2 |   |
| a | 4 | 0 |   |   |   |   |

'd' = 'd' → çapraz + 1:  $L[3][3] = 1 + 1 = 2$

Adım 14 / 22

## DEĞİŞKENLER

i = 3

j = 3

y[i-1] = 'd'

x[j-1] = 'd'

L[i][j] = 2

## LCS: Kodu Adım Adım İzle

```

1 static int lcs(String x, String y) {
2 int m = x.length(), n = y.length();
3 int[][] L = new int[n + 1][m + 1];
4 for (int i = 1; i <= n; i++)
5 for (int j = 1; j <= m; j++)
6 if (y.charAt(i - 1) == x.charAt(j - 1))
7 L[i][j] = L[i - 1][j - 1] + 1;
8 else
9 L[i][j] = Math.max(L[i - 1][j],
10 L[i][j - 1]);
11 return L[n][m];
12 }

```

|   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
|   |   | a | b | d | c | a |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 | 0 | 0 | 1 | 1 |
| b | 2 | 0 | 0 | 1 | 1 | 1 |
| d | 3 | 0 | 0 | 1 | 2 | 2 |
| a | 4 | 0 |   |   |   |   |

'd' ≠ 'c' → max(üst 1, sol 2) = 2

Adım 15 / 22

## DEĞİŞKENLER

i = 3

j = 4

y[i-1] = 'd'

x[j-1] = 'c'

L[i][j] = 2

## LCS: Kodu Adım Adım İzle

```

1 static int lcs(String x, String y) {
2 int m = x.length(), n = y.length();
3 int[][] L = new int[n + 1][m + 1];
4 for (int i = 1; i <= n; i++)
5 for (int j = 1; j <= m; j++)
6 if (y.charAt(i - 1) == x.charAt(j - 1))
7 L[i][j] = L[i - 1][j - 1] + 1;
8 else
9 L[i][j] = Math.max(L[i - 1][j],
10 L[i][j - 1]);
11 return L[n][m];
12 }

```

|     | 0 | 1 | 2 | 3 | 4 | 5 |
|-----|---|---|---|---|---|---|
|     |   | a | b | d | c | a |
| 0   | 0 | 0 | 0 | 0 | 0 | 0 |
| c 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| b 2 | 0 | 0 | 1 | 1 | 1 | 1 |
| d 3 | 0 | 0 | 1 | 2 | 2 | 2 |
| a 4 | 0 |   |   |   |   |   |

'd' ≠ 'a' → max(üst 1, sol 2) = 2

Adım 16 / 22

## DEĞİŞKENLER

i = 3

j = 5

y[i-1] = 'd'

x[j-1] = 'a'

L[i][j] = 2

## LCS: Kodu Adım Adım İzle

```

1 static int lcs(String x, String y) {
2 int m = x.length(), n = y.length();
3 int[][] L = new int[n + 1][m + 1];
4 for (int i = 1; i <= n; i++)
5 for (int j = 1; j <= m; j++)
6 ▶ if (y.charAt(i - 1) == x.charAt(j - 1))
7 ▶ L[i][j] = L[i - 1][j - 1] + 1;
8 else
9 L[i][j] = Math.max(L[i - 1][j],
10 L[i][j - 1]);
11 return L[n][m];
12 }

```

|     | 0 | 1 | 2 | 3 | 4 | 5 |
|-----|---|---|---|---|---|---|
|     |   | a | b | d | c | a |
| 0   | 0 | 0 | 0 | 0 | 0 | 0 |
| c 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| b 2 | 0 | 0 | 1 | 1 | 1 | 1 |
| d 3 | 0 | 0 | 1 | 2 | 2 | 2 |
| a 4 | 0 | 1 |   |   |   |   |

'a' = 'a' → çapraz + 1:  $L[4][1] = 0 + 1 = 1$

Adım 17 / 22

## DEĞİŞKENLER

$i = 4$

$j = 1$

$y[i-1] = 'a'$

$x[j-1] = 'a'$

$L[i][j] = 1$

## LCS: Kodu Adım Adım İzle

```

1 static int lcs(String x, String y) {
2 int m = x.length(), n = y.length();
3 int[][] L = new int[n + 1][m + 1];
4 for (int i = 1; i <= n; i++)
5 for (int j = 1; j <= m; j++)
6 if (y.charAt(i - 1) == x.charAt(j - 1))
7 L[i][j] = L[i - 1][j - 1] + 1;
8 else
9 L[i][j] = Math.max(L[i - 1][j],
10 L[i][j - 1]);
11 return L[n][m];
12 }

```

|     | 0 | 1 | 2 | 3 | 4 | 5 |
|-----|---|---|---|---|---|---|
|     |   | a | b | d | c | a |
| 0   | 0 | 0 | 0 | 0 | 0 | 0 |
| c 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| b 2 | 0 | 0 | 1 | 1 | 1 | 1 |
| d 3 | 0 | 0 | 1 | 2 | 2 | 2 |
| a 4 | 0 | 1 | 1 |   |   |   |

'a' ≠ 'b' → max(üst 1, sol 1) = 1

Adım 18 / 22

## DEĞİŞKENLER

i = 4

j = 2

y[i-1] = 'a'

x[j-1] = 'b'

L[i][j] = 1

## LCS: Kodu Adım Adım İzle

```

1 static int lcs(String x, String y) {
2 int m = x.length(), n = y.length();
3 int[][] L = new int[n + 1][m + 1];
4 for (int i = 1; i <= n; i++)
5 for (int j = 1; j <= m; j++)
6 if (y.charAt(i - 1) == x.charAt(j - 1))
7 L[i][j] = L[i - 1][j - 1] + 1;
8 else
9 L[i][j] = Math.max(L[i - 1][j],
10 L[i][j - 1]);
11 return L[n][m];
12 }

```

|     | 0 | 1 | 2 | 3 | 4 | 5 |
|-----|---|---|---|---|---|---|
|     |   | a | b | d | c | a |
| 0   | 0 | 0 | 0 | 0 | 0 | 0 |
| c 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| b 2 | 0 | 0 | 1 | 1 | 1 | 1 |
| d 3 | 0 | 0 | 1 | 2 | 2 | 2 |
| a 4 | 0 | 1 | 1 | 2 |   |   |

'a' ≠ 'd' → max(üst 2, sol 1) = 2

Adım 19 / 22

## DEĞİŞKENLER

i = 4

j = 3

y[i-1] = 'a'

x[j-1] = 'd'

L[i][j] = 2

## LCS: Kodu Adım Adım İzle

```

1 static int lcs(String x, String y) {
2 int m = x.length(), n = y.length();
3 int[][] L = new int[n + 1][m + 1];
4 for (int i = 1; i <= n; i++)
5 for (int j = 1; j <= m; j++)
6 if (y.charAt(i - 1) == x.charAt(j - 1))
7 L[i][j] = L[i - 1][j - 1] + 1;
8 else
9 L[i][j] = Math.max(L[i - 1][j],
10 L[i][j - 1]);
11 return L[n][m];
12 }

```

|   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
|   |   | a | b | d | c | a |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 | 0 | 0 | 1 | 1 |
| b | 2 | 0 | 0 | 1 | 1 | 1 |
| d | 3 | 0 | 0 | 1 | 2 | 2 |
| a | 4 | 0 | 1 | 1 | 2 | 2 |

'a' ≠ 'c' → max(üst 2, sol 2) = 2

Adım 20 / 22

## DEĞİŞKENLER

i = 4

j = 4

y[i-1] = 'a'

x[j-1] = 'c'

L[i][j] = 2

# LCS: Kodu Adım Adım İzle

```
1 static int lcs(String x, String y) {
2 int m = x.length(), n = y.length();
3 int[][] L = new int[n + 1][m + 1];
4 for (int i = 1; i <= n; i++)
5 for (int j = 1; j <= m; j++)
6 ▶ if (y.charAt(i - 1) == x.charAt(j - 1))
7 ▶ L[i][j] = L[i - 1][j - 1] + 1;
8 else
9 L[i][j] = Math.max(L[i - 1][j],
10 L[i][j - 1]);
11 return L[n][m];
12 }
```

|   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
|   |   | a | b | d | c | a |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 | 0 | 0 | 1 | 1 |
| b | 2 | 0 | 0 | 1 | 1 | 1 |
| d | 3 | 0 | 0 | 1 | 2 | 2 |
| a | 4 | 0 | 1 | 1 | 2 | 3 |

'a' = 'a' → çapraz + 1:  $L[4][5] = 2 + 1 = 3$

Adım 21 / 22

## DEĞİŞKENLER

i = 4

j = 5

y[i-1] = 'a'

x[j-1] = 'a'

L[i][j] = 3

## LCS: Kodu Adım Adım İzle

```

1 static int lcs(String x, String y) {
2 int m = x.length(), n = y.length();
3 int[][] L = new int[n + 1][m + 1];
4 for (int i = 1; i <= n; i++)
5 for (int j = 1; j <= m; j++)
6 if (y.charAt(i - 1) == x.charAt(j - 1))
7 L[i][j] = L[i - 1][j - 1] + 1;
8 else
9 L[i][j] = Math.max(L[i - 1][j],
10 L[i][j - 1]);
11 return L[n][m];
12 }

```

|     | 0 | 1 | 2 | 3 | 4 | 5 |
|-----|---|---|---|---|---|---|
|     |   | a | b | d | c | a |
| 0   | 0 | 0 | 0 | 0 | 0 | 0 |
| c 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| b 2 | 0 | 0 | 1 | 1 | 1 | 1 |
| d 3 | 0 | 0 | 1 | 2 | 2 | 2 |
| a 4 | 0 | 1 | 1 | 2 | 2 | 3 |

Tablo doldu:  $L[4][5] = 3$ . 20 hücre, her biri  $O(1) \rightarrow O(m \cdot n)$ .

Adım 22 / 22

## DEĞİŞKENLER

i = 5

j = -

sonuc = 3

## LCS: Çözümü Geri Okuma

|   |   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
|   |   |   | a | b | d | c | a |
| 0 |   | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| b | 2 | 0 | 0 | 1 | 1 | 1 | 1 |
| d | 3 | 0 | 0 | 1 | 2 | 2 | 2 |
| a | 4 | 0 | 1 | 1 | 2 | 2 | 3 |

Geri okuma son hücreden,  $L[4][5] = 3$ 'ten başlar; bulunan karakterler sondan başa toplanır.

## LCS: Çözümü Geri Okuma

|   |   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
|   |   |   | a | b | d | c | a |
| 0 |   | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| b | 2 | 0 | 0 | 1 | 1 | 1 | 1 |
| d | 3 | 0 | 0 | 1 | 2 | 2 | 2 |
| a | 4 | 0 | 1 | 1 | 2 | 2 | 3 |

'a' = 'a' → karakter LCS'ye eklenir ("a"), çapraz geçilir.

# LCS: Çözümü Geri Okuma

|                  |   | 0 | 1 | 2 | 3 | 4 | 5 |
|------------------|---|---|---|---|---|---|---|
|                  |   |   | a | b | d | c | a |
| c<br>b<br>d<br>a | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|                  | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
|                  | 2 | 0 | 0 | 1 | 1 | 1 | 1 |
|                  | 3 | 0 | 0 | 1 | 2 | 2 | 2 |
|                  | 4 | 0 | 1 | 1 | 2 | 2 | 3 |

'd'  $\neq$  'c'; sol (2) > üst (1)  $\rightarrow$  sola gidilir.

## LCS: Çözümü Geri Okuma

|   |   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
|   |   |   | a | b | d | c | a |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| b | 2 | 0 | 0 | 1 | 1 | 1 | 1 |
| d | 3 | 0 | 0 | 1 | 2 | 2 | 2 |
| a | 4 | 0 | 1 | 1 | 2 | 2 | 3 |

'd' = 'd' → karakter LCS'ye eklenir ("da"), çapraz geçilir.

## LCS: Çözümü Geri Okuma

|   |   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
|   |   |   | a | b | d | c | a |
| 0 |   | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| b | 2 | 0 | 0 | 1 | 1 | 1 | 1 |
| d | 3 | 0 | 0 | 1 | 2 | 2 | 2 |
| a | 4 | 0 | 1 | 1 | 2 | 2 | 3 |

'b' = 'b' → karakter LCS'ye eklenir ("bda"), çapraz geçilir.

# LCS: Çözümü Geri Okuma

|   |   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
|   |   |   | a | b | d | c | a |
| 0 |   | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| b | 2 | 0 | 0 | 1 | 1 | 1 | 1 |
| d | 3 | 0 | 0 | 1 | 2 | 2 | 2 |
| a | 4 | 0 | 1 | 1 | 2 | 2 | 3 |

'c' ≠ 'a'; üst (0) ≥ sol (0) → yukarı çıkılır.

## LCS: Çözümü Geri Okuma

|   |   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
|   |   |   | a | b | d | c | a |
| 0 |   | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| b | 2 | 0 | 0 | 1 | 1 | 1 | 1 |
| d | 3 | 0 | 0 | 1 | 2 | 2 | 2 |
| a | 4 | 0 | 1 | 1 | 2 | 2 | 3 |

Sınıra ulaşıldı. Yeşil hücreler eşleşmeler: LCS = "**bda**", uzunluk 3. Geri okuma  $O(m + n)$ .

## Deneyin: LCS ve Ortak Alt Dizgi Tablosu

X  Y  LCS ortak alt dizgi düzenleme uzaklığı

|   |   | a | b | d | c | a |
|---|---|---|---|---|---|---|
|   | 0 | 0 | 0 | 0 | 0 | 0 |
| c | 0 | 0 | 0 | 0 | 1 | 1 |
| b | 0 | 0 | 1 | 1 | 1 | 1 |
| d | 0 | 0 | 1 | 2 | 2 | 2 |
| a | 0 | 1 | 1 | 2 | 2 | 3 |

LCS uzunluğu **3**: "bda" (son hücre).

X en fazla 9, Y en fazla 7 karakter. Yeşil hücreler geri okuma yolunu gösterir. Düzenleme uzaklığı: ekleme, silme ve değiştirme 1 birim.

## DİĞER PROBLEMLER

# Diğer Klasik DP Problemleri

| Problem                                   | Durum ve bağıntı                                                                                       | Süre               |
|-------------------------------------------|--------------------------------------------------------------------------------------------------------|--------------------|
| <b>Düzenleme uzaklığı</b> (edit distance) | $D[i][j] = \min(D[i-1][j] + 1, D[i][j-1] + 1, D[i-1][j-1] + [a_i \neq b_j]); D[i][0] = i, D[0][j] = j$ | $O(m \cdot n)$     |
| <b>Para üstü</b> (coin change)            | en az para: $P[x] = 1 + \min P[x - c] \ (c \leq x); P[0] = 0$                                          | $O(k \cdot X)$     |
| <b>En büyük toplamli alt dizi</b>         | $V(i) = \max(a[i], V(i-1) + a[i])$ — Kadane; cevap $\max V(i)$                                         | $O(n)$             |
| <b>Matris zinciri çarpımı</b>             | $M[i][j] = \min \{ M[i][k] + M[k+1][j] + p_{i-1} \cdot p_k \cdot p_j : i \leq k < j \}$                | $O(n^3)$           |
| <b>Gezgin satıcı</b> (Held-Karp)          | $dp[S][i] = S$ kümesini gezip $i$ 'de biten en kısa yol; $dp[S][i] = \min dp[S - i][j] + d(j, i)$      | $O(n^2 \cdot 2^n)$ |
| <b>Bellman-Ford</b>                       | $d_k(v) =$ en fazla $k$ kenarlı en kısa yol; $d_k(v) = \min(d_{k-1}(v), d_{k-1}(u) + w(u, v))$         | $O(V \cdot E)$     |
| <b>Optimal ikili arama ağacı</b>          | $e[i][j] = \min \{ e[i][r-1] + e[r+1][j] : i \leq r \leq j \} + w(i, j)$                               | $O(n^3)$           |

## ÖZET

# Özet

| Problem               | Durum / bağıntı                                                             | Zaman · bellek            |
|-----------------------|-----------------------------------------------------------------------------|---------------------------|
| Fibonacci             | $F(n) = F(n-1) + F(n-2)$ ; bellekleme ya da tablo                           | $O(n) \cdot O(1)$         |
| 0/1 sırt çantası      | $K[i][c] = \max(K[i-1][c], v_i + K[i-1][c-w_i])$                            | $O(nW) \cdot O(W)$        |
| Sınırsız sırt çantası | $dp[i][c] = \max(dp[i-1][c], p_i + dp[i][c-w_i])$                           | $O(nW) \cdot O(W)$        |
| Alt küme toplamı      | $dp[i][j] = dp[i-1][j] \vee dp[i-1][j-a_i]$                                 | $O(n \cdot \text{hedef})$ |
| LIS                   | $dp[i] = 1 + \max dp[j] \ (j < i, a[j] < a[i])$ ; ya da aktif listeler      | $O(n^2) / O(n \log n)$    |
| Ortak alt dizgi       | eşitse çapraz + 1, değilse 0; cevap tablonun maksimumu                      | $O(mn)$                   |
| LCS                   | eşitse çapraz + 1, değilse $\max(\text{üst}, \text{sol})$ ; cevap son hücre | $O(mn)$                   |

DP = **örtüşen alt problemler** + **optimal alt yapı**. Süre  $\approx$  durum sayısı  $\times$  durum başına iş.  
Çözümün kendisi tablodan **geri okunarak** bulunur.

## ÖZET

# Sık Yapılan Hatalar

### Taban durumlarını unutmak

$K[0][c]$ ,  $dp[i][0]$  ya da  $fib(0)/fib(1)$  yanlış başlatılırsa bütün tablo kayar.

### İndis kayması

Tablo  $(n+1)$  boyutlu ama dizi  $n$ :  $i$ . satır  $v[i - 1]$  'i kullanır.  $v[i]$  yazmak son eşyada taşma verir.

### Yanlış döngü yönü

Tek boyutlu 0/1 sırt çantasında kapasiteyi küçükten büyüğe gezmek eşyayı birden çok kez alır (sınırsız olur).

### Alt dizgi $\neq$ alt dizi

Ortak alt dizgide eşleşmeme 0 yazar ve cevap tablonun maksimumudur; LCS'de  $\max(\text{üst}, \text{sol})$  yazılır ve cevap son hücredir.

### Açgözlü sanmak

0/1 sırt çantasında ve para üstünde ( $\{1, 3, 4\}$  ile 6) açgözlü seçim optimal değildir.

### $O(nW)$ 'yi polinom sanmak

$W$  sayısal değerdir; girdi boyutu  $\log W$  bittir. Büyük  $W$ 'de tablo kurulamaz.

## ÖZET

# Örnek Problemler

1. 0/1 sırt çantası:  $W = 5$ ; eşyalar  $(v, w) = (3, 2), (4, 3), (5, 4), (6, 5)$ . En büyük değer?
2.  $\text{LCS}(\text{"ABCBADAB"}, \text{"BDCABA"})$  uzunluğu kaçtır? Bir LCS yazın.
3. Paralar  $\{1, 3, 4\}$ , tutar 6: en az kaç para? Açgözlü kaç para kullanır?
4.  $[10, 22, 9, 33, 21, 50, 41, 60]$  dizisinin LIS uzunluğu ve bir LIS?
5. "kitten"  $\rightarrow$  "sitting" düzenleme uzaklığı?
6. Düz özyinelemeli  $\text{fib}(6)$  kaç çağrı yapar? Bellekleme ile kaç?

► Cevapları göster

## ÖZET

# Alıştırmalar

1. 0/1 sırt çantasını **bellekleme** (yukarıdan aşağı) ile yazın.
2. Paralar  $\{1, 2, 5\}$  ile 5 tutarı **kaç farklı şekilde** ödenebilir?
3. "ababc" ve "babca" için en uzun ortak alt dizgiyi bulun.
4.  $\{3, 34, 4, 12, 5, 2\}$  kümesinde toplamı 9 ve 30 olan alt küme var mı?
5.  $[-2, 1, -3, 4, -1, 2, 1, -5, 4]$  için en büyük toplamalı alt dizi (Kadane)?
6. Ağırlık  $\{2, 3, \dots, 8\}$ , değer  $\{3, 5, 8, 6, 10, 11, 15\}$ ,  $W = 10$ : 0/1 ve sınırsız sonuçlar?

► Cevaplar / ipuçları

# Kaynaklar

---

- Cormen, Leiserson, Rivest ve Stein, *Introduction to Algorithms*, 4. baskı, MIT Press, 2022 (Bölüm 14: Dynamic Programming).
- Kleinberg ve Tardos, *Algorithm Design*, Pearson, 2005 (Bölüm 6).
- Dasgupta, Papadimitriou ve Vazirani, *Algorithms*, McGraw-Hill, 2006 (Bölüm 6).
- R. Bellman, *Dynamic Programming*, Princeton University Press, 1957.

# Son

## Bölüm 6: Dinamik Programlama

Sonraki bölüm: Bölüm 7: Yaklaşım Algoritmaları · [Terim sözlüğü](#)