Bölüm 3: Dinamik Programlama

Bölüm 3: Dinamik Programlama · Çalışma Soruları

Temel kavramlar | Fibonacci | 0/1 sırt çantası | Sınırsız sırt çantası | Alt küme toplamı | En uzun artan alt dizi | Ortak alt dizgi ve LCS

Sercan KÜLCÜ · Giresun Üniversitesi · Bilgisayar Mühendisliği

© Sercan Külcü. Tüm hakları saklıdır. · Sunumun etkileşimli HTML sürümü:
https://sercankulcu.github.io/files/advanced_algorithms/slides/Bolum_03_Dinamik_Programlama_Calisma_Sorulari.html

ÇOKTAN SEÇMELİ · TEMEL KAVRAMLAR

Soru 1: DP'nin İki Özelliği

Bir problemin dinamik programlamayla verimli çözülebilmesi için hangi iki özelliği taşıması gerekir?

A) Bağımsız alt problemler ve açgözlü seçim özelliği

B) Örtüşen alt problemler ve optimal alt yapı

C) Sıralı girdi ve logaritmik derinlikte özyineleme

D) Örtüşen alt problemler ve açgözlü seçim özelliği

ÇOKTAN SEÇMELİ · TEMEL KAVRAMLAR

Soru 2: Neden Tablo Gerekmez?

Birleştirme sıralaması özyinelemeli olduğu hâlde neden bir DP tablosundan yararlanmaz?

A) Alt problemleri bağımsızdır; aynı alt dizi iki kez sıralanmaz.

B) Optimal alt yapısı olmadığı için alt çözümler birleştirilemez.

C) Özyineleme derinliği log n olduğu için tablo belleğe sığmaz.

D) Sıralama bir optimizasyon problemi olmadığından DP tanımlı değildir.

ÇIKTI SORUSU · FIBONACCI

Soru 3: Çağrı Sayısı

Düz özyinelemeli `fib` metoduna bir çağrı sayacı eklendi. Kod ne yazdırır? Aynı hesap derstekiler gibi bellekleme ile yapılsaydı kaç çağrı olurdu?

```
static int cagri = 0;
static long fib(int n) {
  cagri++;
  if (n <= 1) return n;
  return fib(n - 1) + fib(n - 2);
}
// main:
System.out.println(fib(7) + " " + cagri);
```

▶ **Cevabı göster**

Soru 4: Bellekleme: fib(4)

Bellekleme kullanan `fib(4)` çağrısında `bellek` dizisi nasıl dolar; hangi çağrı tablodan yanıtlanır?

```
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	0	0	0	0	0
	0	1	2	3	4

bellek dizisi 0 ile başlar (0 = "henüz hesaplanmadı"). fib(4) tabloda yok → fib(3) çağrılır.

Adım 1 / 10

DEĞİŞKENLER

n = 4 | a = – | b = –

ÇAĞRI YIĞINI

`fib(4)`

Soru 4: Bellekleme: fib(4)

Bellekleme kullanan `fib(4)` çağrısında `bellek` dizisi nasıl dolar; hangi çağrı tablodan yanıtlanır?

```
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	0	0	0	0	0
	0	1	2	3	4

fib(3) tabloda yok → fib(2) çağrılır.

Adım 2 / 10

DEĞİŞKENLER

`n = 3` `a = –` `b = –`

ÇAĞRI YIĞINI

- **fib(3)**
- fib(4)

Soru 4: Bellekleme: fib(4)

Bellekleme kullanan `fib(4)` çağrısında `bellek` dizisi nasıl dolar; hangi çağrı tablodan yanıtlanır?

```
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	0	0	0	0	0
	0	1	2	3	4

fib(2) tabloda yok → fib(1) çağrılır.

Adım 3 / 10

DEĞİŞKENLER

`n = 2` `a = –` `b = –`

ÇAĞRI YIĞINI

- **fib(2)**
- fib(3)
- fib(4)

Soru 4: Bellekleme: fib(4)

Bellekleme kullanan `fib(4)` çağrısında `bellek` dizisi nasıl dolar; hangi çağrı tablodan yanıtlanır?

```
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 durum, 1 döner.

Adım 4 / 10

DEĞİŞKENLER

`n = 1` `a = –` `b = –`

ÇAĞRI YIĞINI

- `fib(1)`
- `fib(2)`
- `fib(3)`
- `fib(4)`

Soru 4: Bellekleme: fib(4)

Bellekleme kullanan `fib(4)` çağrısında `bellek` dizisi nasıl dolar; hangi çağrı tablodan yanıtlanır?

```
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);
▶    long b = fib(n - 2);
8    bellek[n] = a + b;
9    return bellek[n];
10 }
```

bellek	0	0	0	0	0
	0	1	2	3	4

fib(2)'de a = 1; şimdi fib(0) çağrılır: taban durum, 0 döner.

Adım 5 / 10

DEĞİŞKENLER

n = 0 | a = – | b = –

ÇAĞRI YIĞINI

- **fib(0)**
- fib(2)
- fib(3)
- fib(4)

Soru 4: Bellekleme: fib(4)

Bellekleme kullanan `fib(4)` çağrısında `bellek` dizisi nasıl dolar; hangi çağrı tablodan yanıtlanır?

```
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);
▶    bellek[n] = a + b;
▶    return bellek[n];
10 }
```

bellek	0	0	1	0	0
	0	1	2	3	4

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

Adım 6 / 10

DEĞİŞKENLER

`n = 2` `a = 1` `b = 0`

ÇAĞRI YIĞINI

- **fib(2)**
- fib(3)
- fib(4)

Soru 4: Bellekleme: fib(4)

Bellekleme kullanan `fib(4)` çağrısında `bellek` dizisi nasıl dolar; hangi çağrı tablodan yanıtlanır?

```
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);
▶    long b = fib(n - 2);
8    bellek[n] = a + b;
9    return bellek[n];
10 }
```

bellek	0	0	1	0	0
	0	1	2	3	4

fib(3)'te a = 1; fib(1) çağrılır → 1 (taban).

Adım 7 / 10

DEĞİŞKENLER

`n = 1` `a = –` `b = –`

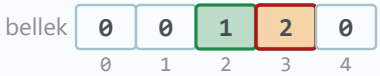
ÇAĞRI YIĞINI

- **fib(1)**
- fib(3)
- fib(4)

Soru 4: Bellekleme: fib(4)

Bellekleme kullanan `fib(4)` çağrısında `bellek` dizisi nasıl dolar; hangi çağrı tablodan yanıtlanır?

```
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);
▶    bellek[n] = a + b;
▶    return bellek[n];
10 }
```



bellek[3] = 1 + 1 = **2**.

Adım 8 / 10

DEĞİŞKENLER

`n = 3` `a = 1` `b = 1`

ÇAĞRI YIĞINI

- **fib(3)**
- fib(4)

Soru 4: Bellekleme: fib(4)

Bellekleme kullanan `fib(4)` çağrısında `bellek` dizisi nasıl dolar; hangi çağrı tablodan yanıtlanır?

```
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)'te a = 2; fib(2) çağrılır: bellek[2] = 1 **hazır**, hiç alt çağrı yapılmadan döner.

Adım 9 / 10

DEĞİŞKENLER

`n = 2` `a = –` `b = –`

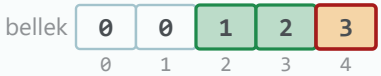
ÇAĞRI YIĞINI

- **fib(2)**
- fib(4)

Soru 4: Bellekleme: fib(4)

Bellekleme kullanan `fib(4)` çağrısında `bellek` dizisi nasıl dolar; hangi çağrı tablodan yanıtlanır?

```
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);
▶    bellek[n] = a + b;
▶    return bellek[n];
10 }
```



bellek[4] = 2 + 1 = **3**. Toplam 7 çağrı (2n − 1); düz özyineleme 9 çağrı yapardı.

Adım 10 / 10

DEĞİŞKENLER

`n = 4` `a = 2` `b = 1`

ÇAĞRI YIĞINI

`fib(4)`

ÇOKTAN SEÇMELİ · FIBONACCI

Soru 5: Bellekleme ve Tablolama

Bellekleme (yukarıdan aşağı) ve tablolama (aşağıdan yukarı) için aşağıdakilerden hangisi **yanlıştır**?

A) Bellekleme yalnızca gerçekten gereken alt problemleri çözer.

B) Tablolamada doldurma sırası önceden doğru belirlenmelidir.

C) Bellekleme her zaman asimptotik olarak tablolamadan hızlıdır.

D) Derin özyinelemede bellekleme yığın taşmasına yol açabilir.

ÇIKTI SORUSU · 0/1 SIRT ÇANTASI

Soru 6: En Büyük Değer

Eşyalar (v, w): (10, 5), (40, 4), (30, 6), (50, 3); kapasite W = 10. 0/1 sırt çantası tablosunun son satırını, en büyük değeri ve seçilen eşyaları bulun.

▶ **Cevabı göster**

ÇOKTAN SEÇMELİ · 0/1 SIRT ÇANTASI

Soru 7: Açgözlü Neden Yanılır?

W = 10; eşyalar (değer, ağırlık): (30 ₺, 6 kg), (20 ₺, 5 kg), (20 ₺, 5 kg). Değer/ağırlık oranına göre açgözlü seçim ve optimal 0/1 çözüm kaç ₺ verir?

A) Açgözlü 30 ₺, optimal 40 ₺

B) Açgözlü 40 ₺, optimal 40 ₺

C) Açgözlü 30 ₺, optimal 50 ₺

D) Açgözlü 50 ₺, optimal 50 ₺

Soru 8: Döngü Yönü

Tek eşya (değer 10, ağırlık 3) ve kapasite 9 için tek boyutlu tablo iki farklı yönde dolduruluyor. Kod ne yazdırır? Hangisi 0/1 sırt çantasıdır?

```
int v = 10, w = 3, W = 9;
int[] dp = new int[W + 1];
int[] dq = new int[W + 1];
for (int c = W; c >= w; c--)   // büyükten küçüğe
  dp[c] = Math.max(dp[c], v + dp[c - w]);
for (int c = w; c <= W; c++)   // küçükten büyüğe
  dq[c] = Math.max(dq[c], v + dq[c - w]);
System.out.println(dp[W] + " " + dq[W]);
```

▶ **Cevabı göster**

Soru 9: Sözde-Polinom

0/1 sırt çantası DP'si $O(n \cdot W)$ sürer. Bu süre neden "sözde-polinom" (pseudo-polynomial) olarak adlandırılır?

A) W kapasitesi her zaman n'den küçük olduğundan süre $O(n^2)$ ile sınırlıdır.

B) n eşyanın 2^n alt kümesi olduğundan süre gizli olarak hep $O(2^n \cdot W)$ kalır.

C) Tablo $(n + 1) \times (W + 1)$ olsa da yalnızca köşegen hücreler hesaplanır.

D) W sayısal değerdir; girdi boyutu ise log W bittir, süre bit sayısına göre üsteldir.

ÇOKTAN SEÇMELİ · SINIRSIZ SIRT ÇANTASI

Soru 10: Kaç Farklı Yol?

Paralar {2, 3, 5} ile 10 tutarı kaç farklı şekilde ödenebilir? (Sıra önemsiz; dış döngü paralar üzerinde: `d[x] += d[x - c]`.)

A) 3

B) 5

C) 6

D) 4

Soru 11: altKume Tablosu

`altKume({3, 4, 5, 2}, 7)` tablosu satır satır nasıl dolar?

```
1  boolean altKume(int[] a, int hedef) {
   int n = a.length; // a[i] ≥ 0, hedef ≥ 0
   boolean[][] dp =
       new boolean[n + 1][hedef + 1];
   for (int i = 0; i <= n; i++)
     dp[i][0] = true;            // boş küme
7  for (int i = 1; i <= n; i++) {
8    int x = a[i - 1];
9    for (int j = 1; j <= hedef; j++)
10     dp[i][j] = dp[i - 1][j]  // x alınmaz
11       || (x <= j && dp[i - 1][j - x]);
12 }
13 return dp[n][hedef];
14 }
```

dp[i][j]

	0	1	2	3	4	5	6	7
0	✓	·	·	·	·	·	·	·
1 (3)								
2 (4)								
3 (5)								
4 (2)								

a = {3, 4, 5, 2}, hedef 7. Sütun 0 hep ✓ (boş küme); satır 0'da diğer toplamlar elde edilemez.

Adım 1 / 6

DEĞİŞKENLER

`i = –` `x = –` `dp[n][hedef] = –`

ÇIKTI

Soru 11: altKume Tablosu

`altKume({3, 4, 5, 2}, 7)` tablosu satır satır nasıl dolar?

```
1  boolean altKume(int[] a, int hedef) {
2    int n = a.length; // a[i] ≥ 0, hedef ≥ 0
3    boolean[][] dp =
4        new boolean[n + 1][hedef + 1];
5    for (int i = 0; i <= n; i++)
6      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]);
12   }
13   return dp[n][hedef];
14 }
```

dp[i][j]

	0	1	2	3	4	5	6	7
0	✓	·	·	·	·	·	·	·
1 (3)	✓	·	·	✓	·	·	·	·
2 (4)								
3 (5)								
4 (2)								

i = 1, x = 3: üst satır kopyalanır; dp[i−1][j−3] ✓ olan j'ler de ✓ olur. Yeni toplamlar (mavi): 3.

Adım 2 / 6

DEĞİŞKENLER

`i = 1` `x = 3` `dp[n][hedef] = –`

ÇIKTI

Soru 11: altKume Tablosu

`altKume({3, 4, 5, 2}, 7)` tablosu satır satır nasıl dolar?

```
1  boolean altKume(int[] a, int hedef) {
2    int n = a.length; // a[i] ≥ 0, hedef ≥ 0
3    boolean[][] dp =
4        new boolean[n + 1][hedef + 1];
5    for (int i = 0; i <= n; i++)
6      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]);
12   }
13   return dp[n][hedef];
14 }
```

dp[i][j]

	0	1	2	3	4	5	6	7
0	✓	·	·	·	·	·	·	·
1 (3)	✓	·	·	✓	·	·	·	·
2 (4)	✓	·	·	✓	✓	·	·	✓
3 (5)								
4 (2)								

i = 2, x = 4: üst satır kopyalanır; dp[i−1][j−4] ✓ olan j'ler de ✓ olur. Yeni toplamlar (mavi): 4, 7.

Adım 3 / 6

DEĞİŞKENLER

`i = 2` `x = 4` `dp[n][hedef] = –`

ÇIKTI

Soru 11: altKume Tablosu

`altKume({3, 4, 5, 2}, 7)` tablosu satır satır nasıl dolar?

```
1  boolean altKume(int[] a, int hedef) {
2    int n = a.length; // a[i] ≥ 0, hedef ≥ 0
3    boolean[][] dp =
4        new boolean[n + 1][hedef + 1];
5    for (int i = 0; i <= n; i++)
6      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]);
12   }
13   return dp[n][hedef];
14 }
```

dp[i][j]

	0	1	2	3	4	5	6	7
0	✓	·	·	·	·	·	·	·
1 (3)	✓	·	·	✓	·	·	·	·
2 (4)	✓	·	·	✓	✓	·	·	✓
3 (5)	✓	·	·	✓	✓	✓	·	✓
4 (2)								

i = 3, x = 5: üst satır kopyalanır; dp[i−1][j−5] ✓ olan j'ler de ✓ olur. Yeni toplamlar (mavi): 5.

Adım 4 / 6

DEĞİŞKENLER

`i = 3` `x = 5` `dp[n][hedef] = –`

ÇIKTI

Soru 11: altKume Tablosu

`altKume({3, 4, 5, 2}, 7)` tablosu satır satır nasıl dolar?

```
1  boolean altKume(int[] a, int hedef) {
2    int n = a.length; // a[i] ≥ 0, hedef ≥ 0
3    boolean[][] dp =
4        new boolean[n + 1][hedef + 1];
5    for (int i = 0; i <= n; i++)
6      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]);
12   }
13   return dp[n][hedef];
14 }
```

dp[i][j]

	0	1	2	3	4	5	6	7
0	✓	·	·	·	·	·	·	·
1 (3)	✓	·	·	✓	·	·	·	·
2 (4)	✓	·	·	✓	✓	·	·	✓
3 (5)	✓	·	·	✓	✓	✓	·	✓
4 (2)	✓	·	✓	✓	✓	✓	✓	✓

i = 4, x = 2: üst satır kopyalanır; dp[i−1][j−2] ✓ olan j'ler de ✓ olur. Yeni toplamlar (mavi): 2, 6.

Adım 5 / 6

DEĞİŞKENLER

`i = 4` `x = 2` `dp[n][hedef] = –`

ÇIKTI

Soru 11: altKume Tablosu

`altKume({3, 4, 5, 2}, 7)` tablosu satır satır nasıl dolar?

```
1  boolean altKume(int[] a, int hedef) {
2    int n = a.length; // a[i] ≥ 0, hedef ≥ 0
3    boolean[][] dp =
4        new boolean[n + 1][hedef + 1];
5    for (int i = 0; i <= n; i++)
6      dp[i][0] = true;            // boş küme
7    for (int i = 1; i <= n; i++) {
8      int x = a[i - 1];
9      for (int j = 1; j <= hedef; j++)
10       dp[i][j] = dp[i - 1][j]  // x alınmaz
11         || (x <= j && dp[i - 1][j - x]);
12   }
▶    return dp[n][hedef];
14 }
```

dp[i][j]

	0	1	2	3	4	5	6	7
0	✓	·	·	·	·	·	·	·
1 (3)	✓	·	·	✓	·	·	·	·
2 (4)	✓	·	·	✓	✓	·	·	✓
3 (5)	✓	·	·	✓	✓	✓	·	✓
4 (2)	✓	·	✓	✓	✓	✓	✓	✓

dp[4][7] = **true**. Geri okuma: dp[4][7] = dp[3][7] = dp[2][7] ✓ (2 ve 5 gerekmez) ama dp[1][7] · → 4 seçildi (j = 3); dp[1][3] ✓ → 3. Alt küme {3, 4}; {5, 2} de geçerlidir.

Adım 6 / 6

DEĞİŞKENLER

`i = –` `x = –` `dp[n][hedef] = true`

ÇIKTI

ÇOKTAN SEÇMELİ · ALT KÜME TOPLAMI

Soru 12: Eşit Bölme

`{1, 5, 11, 5}` dizisinin toplamı eşit iki parçaya bölünüp bölünemeyeceğini alt küme toplamıyla nasıl buluruz ve sonuç nedir?

A) Hedef 22 ile alt küme toplamı; evet

B) Hedef 11 ile alt küme toplamı; hayır

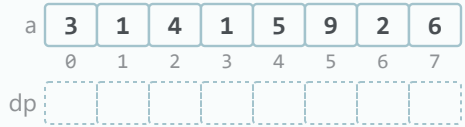
C) Hedef 11 ile alt küme toplamı; evet

D) Hedef 5,5 ile alt küme toplamı; hayır

Soru 13: lis İzleme

`lis({3, 1, 4, 1, 5, 9, 2, 6})` hangi değeri döndürür?

```
1  static int lis(int[] a) {
2    int n = a.length, enUzun = 0;
3    int[] dp = new int[n];
4    for (int i = 0; i < n; i++) {
5      dp[i] = 1;                // tek başına
6      for (int j = 0; j < i; j++)
7        if (a[j] < a[i] && dp[j] + 1 > dp[i])
8          dp[i] = dp[j] + 1;    // uzat
9      enUzun = Math.max(enUzun, dp[i]);
10   }
11   return enUzun;
12 }
```



dp[i]: a[i]'de biten en uzun artan alt dizinin uzunluğu.
Mavi: a[j] < a[i] olan önceki adaylar.

Adım 1 / 10

DEĞİŞKENLER

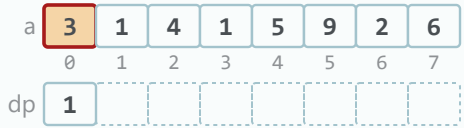
`i = –` `dp = [0, 0, 0, 0, 0, 0, 0, 0]` `enUzun = 0`

ÇIKTI

Soru 13: lis İzleme

`lis({3, 1, 4, 1, 5, 9, 2, 6})` hangi değeri döndürür?

```
1  static int lis(int[] a) {
2    int n = a.length, enUzun = 0;
3    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
▶      enUzan = Math.max(enUzun, dp[i]);
10   }
11   return enUzun;
12 }
```



i = 0 (a = 3): önünde daha küçük öğe yok → dp[0] = 1.

Adım 2 / 10

DEĞİŞKENLER

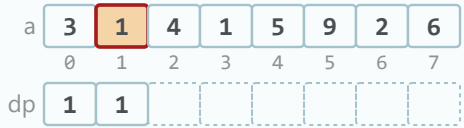
`i = 0` `dp = [1, 0, 0, 0, 0, 0, 0, 0]` `enUzun = 1`

ÇIKTI

Soru 13: lis İzleme

`lis({3, 1, 4, 1, 5, 9, 2, 6})` hangi değeri döndürür?

```
1  static int lis(int[] a) {
2    int n = a.length, enUzun = 0;
3    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
▶      enUzan = Math.max(enUzun, dp[i]);
10   }
11   return enUzun;
12 }
```



i = 1 (a = 1): önünde daha küçük öğe yok → dp[1] = 1.

Adım 3 / 10

DEĞİŞKENLER

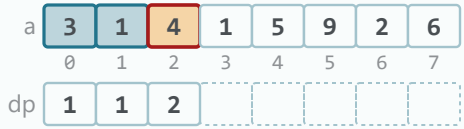
`i = 1` `dp = [1, 1, 0, 0, 0, 0, 0, 0]` `enUzun = 1`

ÇIKTI

Soru 13: lis İzleme

`lis({3, 1, 4, 1, 5, 9, 2, 6})` hangi değeri döndürür?

```
1  static int lis(int[] a) {
2    int n = a.length, enUzun = 0;
3    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]);
10   }
11   return enUzun;
12 }
```



i = 2 (a = 4): adaylar 3, 1; en büyük dp j = 0 (dp = 1) → dp[2] = **2**.

Adım 4 / 10

DEĞİŞKENLER

`i = 2` `dp = [1, 1, 2, 0, 0, 0, 0, 0]` `enUzun = 2`

ÇIKTI

Soru 13: lis İzleme

`lis({3, 1, 4, 1, 5, 9, 2, 6})` hangi değeri döndürür?

```
1  static int lis(int[] a) {
2    int n = a.length, enUzun = 0;
3    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]);
10   }
11   return enUzun;
12 }
```



i = 3 (a = 1): önünde daha küçük öğe yok → dp[3] = 1.

Adım 5 / 10

DEĞİŞKENLER

`i = 3` `dp = [1, 1, 2, 1, 0, 0, 0, 0]` `enUzun = 2`

ÇIKTI

Soru 13: lis İzleme

`lis({3, 1, 4, 1, 5, 9, 2, 6})` hangi değeri döndürür?

```
1  static int lis(int[] a) {
2    int n = a.length, enUzun = 0;
3    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]);
10   }
11   return enUzun;
12 }
```



i = 4 (a = 5): adaylar 3, 1, 4, 1; en büyük dp j = 2 (dp = 2) → dp[4] = **3**.

Adım 6 / 10

DEĞİŞKENLER

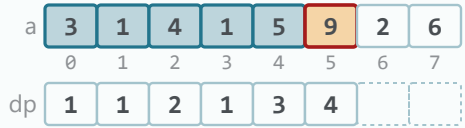
`i = 4` `dp = [1, 1, 2, 1, 3, 0, 0, 0]` `enUzun = 3`

ÇIKTI

Soru 13: lis İzleme

`lis({3, 1, 4, 1, 5, 9, 2, 6})` hangi değeri döndürür?

```
1  static int lis(int[] a) {
2    int n = a.length, enUzun = 0;
3    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]);
10   }
11   return enUzun;
12 }
```



i = 5 (a = 9): adaylar 3, 1, 4, 1, 5; en büyük dp j = 4 (dp = 3) → dp[5] = **4**.

Adım 7 / 10

DEĞİŞKENLER

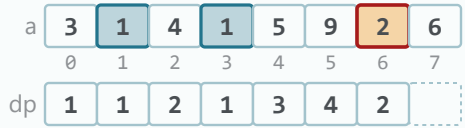
`i = 5` `dp = [1, 1, 2, 1, 3, 4, 0, 0]` `enUzun = 4`

ÇIKTI

Soru 13: lis İzleme

`lis({3, 1, 4, 1, 5, 9, 2, 6})` hangi değeri döndürür?

```
1  static int lis(int[] a) {
2    int n = a.length, enUzun = 0;
3    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]);
10   }
11   return enUzun;
12 }
```



i = 6 (a = 2): adaylar 1, 1; en büyük dp j = 1 (dp = 1) → dp[6] = **2**.

Adım 8 / 10

DEĞİŞKENLER

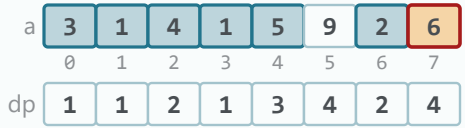
`i = 6` `dp = [1, 1, 2, 1, 3, 4, 2, 0]` `enUzun = 4`

ÇIKTI

Soru 13: lis İzleme

`lis({3, 1, 4, 1, 5, 9, 2, 6})` hangi değeri döndürür?

```
1  static int lis(int[] a) {
2    int n = a.length, enUzun = 0;
3    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]);
10   }
11   return enUzun;
12 }
```



i = 7 (a = 6): adaylar 3, 1, 4, 1, 5, 2; en büyük dp j = 4 (dp = 3) → dp[7] = **4**.

Adım 9 / 10

DEĞİŞKENLER

`i = 7` `dp = [1, 1, 2, 1, 3, 4, 2, 4]` `enUzun = 4`

ÇIKTI

Soru 13: lis İzleme

`lis({3, 1, 4, 1, 5, 9, 2, 6})` hangi değeri döndürür?

```
1  static int lis(int[] a) {
2    int n = a.length, enUzun = 0;
3    int[] dp = new int[n];
4    for (int i = 0; i < n; i++) {
5      dp[i] = 1;                // tek başına
6      for (int j = 0; j < i; j++)
7        if (a[j] < a[i] && dp[j] + 1 > dp[i])
8          dp[i] = dp[j] + 1;    // uzat
9      enUzun = Math.max(enUzun, dp[i]);
10   }
▶    return enUzun;
12 }
```

	0	1	2	3	4	5	6	7
a	3	1	4	1	5	9	2	6
dp	1	1	2	1	3	4	2	4

Sonuç **4** = max dp (ör. 3, 4, 5, 9 ya da 1, 4, 5, 6). dp[n − 1] de 4 ama bu tesadüf; cevap her zaman en büyük dp'dir. O(n^2).

Adım 10 / 10

DEĞİŞKENLER

`i = –` `dp = [1, 1, 2, 1, 3, 4, 2, 4]` `enUzun = 4`

ÇIKTI

ÇOKTAN SEÇMELİ · EN UZUN ARTAN ALT DİZİ

Soru 14: Cevap Hangi Hücrede?

Bir öğrenci `lis` metodunda `enUzun` yerine `dp[n - 1]` döndürüyor. `{5, 1, 6, 2, 7, 0}` için hatalı ve doğru sonuçlar nedir?

A) Hatalı 3, doğru 3

B) Hatalı 1, doğru 3

C) Hatalı 2, doğru 3

D) Hatalı 1, doğru 4

ÇIKTI SORUSU · ORTAK ALT DİZGİ

Soru 15: Alt Dizgi ve Alt Dizi

"programlama" ve "gramer" için en uzun ortak alt dizgi (bitişik) ve en uzun ortak alt dizi (LCS) uzunlukları nedir? Ortak alt dizgiyi yazın; tablolardaki cevap hücreleri nerededir?

▶ **Cevabı göster**

Soru 16: Alt Dizi mi, Alt Dizgi mi?

A = "GXTXAYB" için aşağıdakilerden hangisi **doğrudur**?

A) "GTAB" hem alt dizi hem alt dizgidir.

B) "GTAB" alt dizgidir ama alt dizi değildir.

C) "GTAB" alt dizidir ama alt dizgi değildir.

D) "XAY" alt dizi değildir ama alt dizgidir.

ÇIKTI SORUSU · EN UZUN ORTAK ALT DİZİ

Soru 17: lcs Uzunlukları

Dersteki `lcs` metodu (uzunluk döndürür) ile kod ne yazdırır? Her biri için bir LCS yazın.

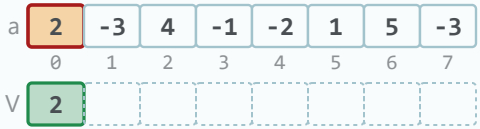
```
System.out.println(lcs("DINAMIK", "DIAMOND"));
System.out.println(lcs("ABCBDAB", "BDCABA"));
System.out.println(lcs("ABC", ""));
```

▶ **Cevabı göster**

Soru 18: Kadane İzleme

En büyük toplamlı alt dizi (Kadane, V(i) = max(a[i], V(i−1) + a[i])): `kadane({2, -3, 4, -1, -2, 1, 5, -3})` ne döndürür?

```
1 static int kadane(int[] a) {
▶   int enIyi = a[0], v = a[0];   // V(0)
3   for (int i = 1; i < a.length; i++) {
4     v = Math.max(a[i], v + a[i]);  // V(i)
5     enIyi = Math.max(enIyi, v);
6   }
7   return enIyi;
8 }
```



V(i): a[i]'de biten en büyük toplamlı alt dizinin toplamı. V(0) = a[0] = 2; enIyi = 2.

Adım 1 / 9

DEĞİŞKENLER

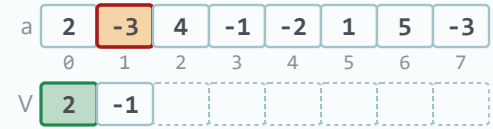
`i = 0` `v = 2` `enIyi = 2`

ÇIKTI

Soru 18: Kadane İzleme

En büyük toplamlı alt dizi (Kadane, V(i) = max(a[i], V(i−1) + a[i])): `kadane({2, −3, 4, −1, −2, 1, 5, −3})` ne döndürür?

```
1 static int kadane(int[] a) {
2   int enIyi = a[0], v = a[0];   // V(0)
▶   for (int i = 1; i < a.length; i++) {
▶     v = Math.max(a[i], v + a[i]);  // V(i)
▶     enIyi = Math.max(enIyi, v);
6   }
7   return enIyi;
8 }
```



i = 1: V = max(-3, 2 + -3) = -1 (uzat); enIyi = 2.

Adım 2 / 9

DEĞİŞKENLER

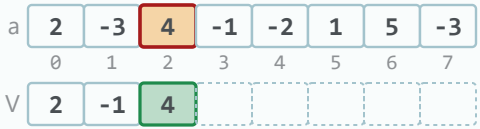
`i = 1` `v = −1` `enIyi = 2`

ÇIKTI

Soru 18: Kadane İzleme

En büyük toplamlı alt dizi (Kadane, V(i) = max(a[i], V(i−1) + a[i])): `kadane({2, −3, 4, −1, −2, 1, 5, −3})` ne döndürür?

```
1 static int kadane(int[] a) {
2   int enIyi = a[0], v = a[0];   // V(0)
    for (int i = 1; i < a.length; i++) {
      v = Math.max(a[i], v + a[i]);  // V(i)
      enIyi = Math.max(enIyi, v);
6   }
7   return enIyi;
8 }
```



i = 2: V = max(4, -1 + 4) = 4 (yeniden başla); enIyi = 4.

Adım 3 / 9

DEĞİŞKENLER

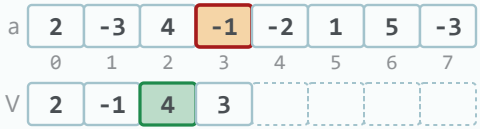
`i = 2` `v = 4` `enIyi = 4`

ÇIKTI

Soru 18: Kadane İzleme

En büyük toplamlı alt dizi (Kadane, V(i) = max(a[i], V(i−1) + a[i])): `kadane({2, −3, 4, −1, −2, 1, 5, −3})` ne döndürür?

```
1 static int kadane(int[] a) {
2   int enIyi = a[0], v = a[0];   // V(0)
▶   for (int i = 1; i < a.length; i++) {
▶     v = Math.max(a[i], v + a[i]);  // V(i)
▶     enIyi = Math.max(enIyi, v);
6   }
7   return enIyi;
8 }
```



i = 3: V = max(-1, 4 + -1) = 3 (uzat); enIyi = 4.

Adım 4 / 9

DEĞİŞKENLER

`i = 3` `v = 3` `enIyi = 4`

ÇIKTI

Soru 18: Kadane İzleme

En büyük toplamlı alt dizi (Kadane, V(i) = max(a[i], V(i−1) + a[i])): `kadane({2, −3, 4, −1, −2, 1, 5, −3})` ne döndürür?

```
1 static int kadane(int[] a) {
2   int enIyi = a[0], v = a[0];   // V(0)
▶   for (int i = 1; i < a.length; i++) {
▶     v = Math.max(a[i], v + a[i]);  // V(i)
▶     enIyi = Math.max(enIyi, v);
6   }
7   return enIyi;
8 }
```



i = 4: V = max(-2, 3 + -2) = 1 (uzat); enIyi = 4.

Adım 5 / 9

DEĞİŞKENLER

`i = 4` `v = 1` `enIyi = 4`

ÇIKTI

Soru 18: Kadane İzleme

En büyük toplamlı alt dizi (Kadane, V(i) = max(a[i], V(i−1) + a[i])): `kadane({2, −3, 4, −1, −2, 1, 5, −3})` ne döndürür?

```
1 static int kadane(int[] a) {
2   int enIyi = a[0], v = a[0];   // V(0)
    for (int i = 1; i < a.length; i++) {
      v = Math.max(a[i], v + a[i]);  // V(i)
      enIyi = Math.max(enIyi, v);
6   }
7   return enIyi;
8 }
```



i = 5: V = max(1, 1 + 1) = 2 (uzat); enIyi = 4.

Adım 6 / 9

DEĞİŞKENLER

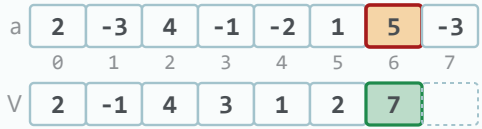
`i = 5` `v = 2` `enIyi = 4`

ÇIKTI

Soru 18: Kadane İzleme

En büyük toplamlı alt dizi (Kadane, V(i) = max(a[i], V(i−1) + a[i])): `kadane({2, −3, 4, −1, −2, 1, 5, −3})` ne döndürür?

```
1 static int kadane(int[] a) {
2   int enIyi = a[0], v = a[0];   // V(0)
▶   for (int i = 1; i < a.length; i++) {
▶     v = Math.max(a[i], v + a[i]);  // V(i)
▶     enIyi = Math.max(enIyi, v);
6   }
7   return enIyi;
8 }
```



i = 6: V = max(5, 2 + 5) = 7 (uzat); enIyi = 7.

Adım 7 / 9

DEĞİŞKENLER

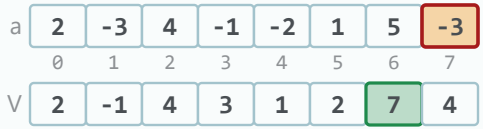
`i = 6` `v = 7` `enIyi = 7`

ÇIKTI

Soru 18: Kadane İzleme

En büyük toplamlı alt dizi (Kadane, V(i) = max(a[i], V(i−1) + a[i])): `kadane({2, −3, 4, −1, −2, 1, 5, −3})` ne döndürür?

```
1 static int kadane(int[] a) {
2   int enIyi = a[0], v = a[0];   // V(0)
▶   for (int i = 1; i < a.length; i++) {
▶     v = Math.max(a[i], v + a[i]);  // V(i)
▶     enIyi = Math.max(enIyi, v);
6   }
7   return enIyi;
8 }
```



i = 7: V = max(-3, 7 + -3) = 4 (uzat); enIyi = 7.

Adım 8 / 9

DEĞİŞKENLER

`i = 7` `v = 4` `enIyi = 7`

ÇIKTI

Soru 18: Kadane İzleme

En büyük toplamlı alt dizi (Kadane, V(i) = max(a[i], V(i−1) + a[i])): `kadane({2, −3, 4, −1, −2, 1, 5, −3})` ne döndürür?

```
1 static int kadane(int[] a) {
2   int enIyi = a[0], v = a[0];   // V(0)
3   for (int i = 1; i < a.length; i++) {
4     v = Math.max(a[i], v + a[i]);  // V(i)
5     enIyi = Math.max(enIyi, v);
6   }
▶   return enIyi;
8 }
```



Sonuç **7**: alt dizi [4, −1, −2, 1, 5] (indis 2–6). Her durum O(1) → O(n) zaman, O(1) bellek.

Adım 9 / 9

DEĞİŞKENLER

`i = –` `v = 4` `enIyi = 7`

ÇIKTI

ÇOKTAN SEÇMELİ · DİĞER PROBLEMLER

Soru 19: Matris Zinciri

Matris zinciri çarpımında `M[i][j] = min { M[i][k] + M[k+1][j] + $p_{i-1} \cdot p_k \cdot p_j$ : i ≤ k < j }`. n matris için zaman karmaşıklığı nedir?

A) $O(n^2)$

B) $O(n^3)$

C) $O(2^n)$

D) $O(n^2 \log n)$

KLASİK SORU · DİĞER PROBLEMLER

Soru 20: DP Tasarımı: Para Üstü

Paralar $c_1, \ldots, c_k$ ve tutar X için "en az kaç para" problemini derste anlatılan adımlarla tasarlayın: durum, bağıntı, taban durum, doldurma sırası, tablo boyutu ve karmaşıklık. Paralar {1, 3, 4} ve X = 6 için tabloyu doldurun.

▶ **Cevabı göster**

Son

Bölüm 3: Dinamik Programlama: Çalışma Soruları

Konu anlatımı: Dinamik Programlama sunumu · Terim sözlüğü