Bölüm 6: Yaklaşım Algoritmaları

Bölüm 6: Yaklaşım Algoritmaları · Çalışma Soruları

Yaklaşım oranı | Gezgin satıcı | 2-opt yerel arama | İş çizelgeleme | Düğüm kapsama | Sırt çantası | Power Iteration

Sercan KÜLCÜ · Giresun Üniversitesi · Bilgisayar Mühendisliği

© Sercan Külcü. Tüm hakları saklıdır. · Sunumun etkileşimli HTML sürümü:
https://sercankulcu.github.io/files/advanced_algorithms/slides/Bolum_06_Yaklasim_Algoritmalari_Calisma_Sorulari.html

ÇOKTAN SEÇMELİ · YAKLAŞIM ORANI

Soru 1: En Büyüklemede Garanti

Bir en büyükleme problemi (ör. sırt çantası) için 2-yaklaşım algoritması kullanılıyor. Bir girdinin optimal değeri $C^* = 90$ ise algoritmanın bulduğu C değeri için **kesin** olarak ne söylenebilir?

A) $C \leq 45$

B) $C \geq 180$

C) $C \geq 45$

D) $C = 45$

ÇOKTAN SEÇMELİ · YAKLAŞIM ORANI

Soru 2: Yaklaşım Türleri

Yaklaşım algoritmaları ve sezgiseller hakkında hangisi **doğrudur**?

A) En yakın komşu sezgiseli metrik TSP'de 2-yaklaşım garantisi verir.

B) FPTAS'ta süre hem n'de hem de 1/ε'da polinomdur, oran 1 + ε olur.

C) PTAS'ta süre 1/ε'da da polinom olmak zorundadır; aksi FPTAS olur.

D) 2-opt yerel araması turu her zaman genel optimuma götürür.

KLASİK SORU · YAKLAŞIM ORANI

Soru 3: Genel TSP'de Sabit Oran Yok

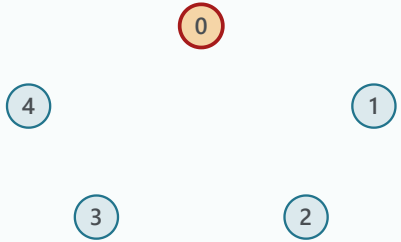
$P \neq NP$ ise genel (üçgen eşitsizliğini sağlamayan) TSP için hiçbir sabit $\rho \geq 1$ ile polinom zamanlı ρ-yaklaşım algoritması olmadığını kanıt fikriyle açıklayın. Hamilton çevrimi problemini kullanın.

▶ **Cevabı göster**

Soru 4: En Yakın Komşu İzleme

5 şehrin uzaklık matrisinin satırları: 0: (0 3 4 2 7) · 1: (3 0 4 6 3) · 2: (4 4 0 5 8) · 3: (2 6 5 0 6) · 4: (7 3 8 6 0). `enYakinKomsu(d, 0)` hangi turu ve uzunluğu bulur?

```
1  int[] enYakinKomsu(int[][] d, int bas) {
▶    int n = d.length, toplam = 0, simdiki = bas;
▶    boolean[] ziyaret = new boolean[n];
▶    int[] tur = new int[n + 1];
▶    tur[0] = bas; ziyaret[bas] = true;
6    for (int k = 1; k < n; k++) {
7      int enYakin = -1, enAz = Integer.MAX_VALUE;
8      for (int j = 0; j < n; j++)
9        if (!ziyaret[j] && d[simdiki][j] < enAz) {
10         enAz = d[simdiki][j]; enYakin = j; }
11     tur[k] = enYakin; ziyaret[enYakin] = true;
12     toplam += enAz; simdiki = enYakin;
13   }
14   tur[n] = bas; toplam += d[simdiki][bas];
15   return tur;
16 }
```



Başlangıç şehri 0 ziyaret edildi; `tur = [0]`, `toplam = 0`. Matris simetrik ama metrik değil.

Adım 1 / 10

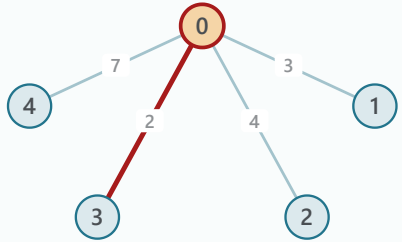
DEĞİŞKENLER

`k = –` `simdiki = 0` `enYakin = –` `enAz = –` `toplam = 0` `tur = [0]`

Soru 4: En Yakın Komşu İzleme

5 şehrin uzaklık matrisinin satırları: 0: (0 3 4 2 7) · 1: (3 0 4 6 3) · 2: (4 4 0 5 8) · 3: (2 6 5 0 6) · 4: (7 3 8 6 0). `enYakinKomsu(d, 0)` hangi turu ve uzunluğu bulur?

```
1  int[] enYakinKomsu(int[][] d, int bas) {
2    int n = d.length, toplam = 0, simdiki = bas;
3    boolean[] ziyaret = new boolean[n];
4    int[] tur = new int[n + 1];
5    tur[0] = bas; ziyaret[bas] = true;
▶    for (int k = 1; k < n; k++) {
▶      int enYakin = -1, enAz = Integer.MAX_VALUE;
▶      for (int j = 0; j < n; j++)
▶        if (!ziyaret[j] && d[simdiki][j] < enAz) {
▶          enAz = d[simdiki][j]; enYakin = j; }
11     tur[k] = enYakin; ziyaret[enYakin] = true;
12     toplam += enAz; simdiki = enYakin;
13   }
14   tur[n] = bas; toplam += d[simdiki][bas];
15   return tur;
16 }
```



k = 1: 0. satırda ziyaret edilmemişler: d[0][1] = 3, d[0][2] = 4, d[0][3] = 2, d[0][4] = 7 → en küçüğü **3** (2).

Adım 2 / 10

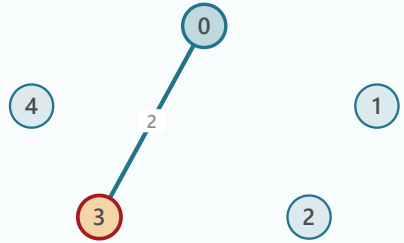
DEĞİŞKENLER

`k = 1` `simdiki = 0` `enYakin = 3` `enAz = 2` `toplam = 0` `tur = [0]`

Soru 4: En Yakın Komşu İzleme

5 şehrin uzaklık matrisinin satırları: 0: (0 3 4 2 7) · 1: (3 0 4 6 3) · 2: (4 4 0 5 8) · 3: (2 6 5 0 6) · 4: (7 3 8 6 0). `enYakinKomsu(d, 0)` hangi turu ve uzunluğu bulur?

```
1  int[] enYakinKomsu(int[][] d, int bas) {
2    int n = d.length, toplam = 0, simdiki = bas;
3    boolean[] ziyaret = new boolean[n];
4    int[] tur = new int[n + 1];
5    tur[0] = bas; ziyaret[bas] = true;
6    for (int k = 1; k < n; k++) {
7      int enYakin = -1, enAz = Integer.MAX_VALUE;
8      for (int j = 0; j < n; j++)
9        if (!ziyaret[j] && d[simdiki][j] < enAz) {
10         enAz = d[simdiki][j]; enYakin = j; }
▶      tur[k] = enYakin; ziyaret[enYakin] = true;
▶      toplam += enAz; simdiki = enYakin;
13   }
14   tur[n] = bas; toplam += d[simdiki][bas];
15   return tur;
16 }
```



`tur[1] = 3`, toplam += 2 → **2**; artık `simdiki = 3`.

Adım 3 / 10

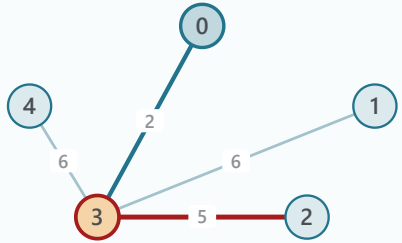
DEĞİŞKENLER

`k = 1` `simdiki = 3` `enYakin = 3`
`enAz = 2` `toplam = 2` `tur = [0, 3]`

Soru 4: En Yakın Komşu İzleme

5 şehrin uzaklık matrisinin satırları: 0: (0 3 4 2 7) · 1: (3 0 4 6 3) · 2: (4 4 0 5 8) · 3: (2 6 5 0 6) · 4: (7 3 8 6 0). `enYakinKomsu(d, 0)` hangi turu ve uzunluğu bulur?

```
1  int[] enYakinKomsu(int[][] d, int bas) {
2    int n = d.length, toplam = 0, simdiki = bas;
3    boolean[] ziyaret = new boolean[n];
4    int[] tur = new int[n + 1];
5    tur[0] = bas; ziyaret[bas] = true;
▶    for (int k = 1; k < n; k++) {
▶      int enYakin = -1, enAz = Integer.MAX_VALUE;
▶      for (int j = 0; j < n; j++)
▶        if (!ziyaret[j] && d[simdiki][j] < enAz) {
▶          enAz = d[simdiki][j]; enYakin = j; }
11     tur[k] = enYakin; ziyaret[enYakin] = true;
12     toplam += enAz; simdiki = enYakin;
13   }
14   tur[n] = bas; toplam += d[simdiki][bas];
15   return tur;
16 }
```



k = 2: 3. satırda ziyaret edilmemişler: d[3][1] = 6, d[3][2] = 5, d[3][4] = 6 → en küçüğü **2** (5).

Adım 4 / 10

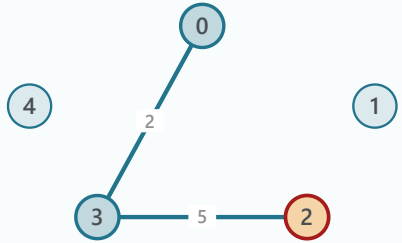
DEĞİŞKENLER

`k = 2` `simdiki = 3` `enYakin = 2` `enAz = 5` `toplam = 2` `tur = [0, 3]`

Soru 4: En Yakın Komşu İzleme

5 şehrin uzaklık matrisinin satırları: 0: (0 3 4 2 7) · 1: (3 0 4 6 3) · 2: (4 4 0 5 8) · 3: (2 6 5 0 6) · 4: (7 3 8 6 0). `enYakinKomsu(d, 0)` hangi turu ve uzunluğu bulur?

```
1  int[] enYakinKomsu(int[][] d, int bas) {
2    int n = d.length, toplam = 0, simdiki = bas;
3    boolean[] ziyaret = new boolean[n];
4    int[] tur = new int[n + 1];
5    tur[0] = bas; ziyaret[bas] = true;
6    for (int k = 1; k < n; k++) {
7      int enYakin = -1, enAz = Integer.MAX_VALUE;
8      for (int j = 0; j < n; j++)
9        if (!ziyaret[j] && d[simdiki][j] < enAz) {
10         enAz = d[simdiki][j]; enYakin = j; }
▶      tur[k] = enYakin; ziyaret[enYakin] = true;
▶      toplam += enAz; simdiki = enYakin;
13   }
14   tur[n] = bas; toplam += d[simdiki][bas];
15   return tur;
16 }
```



`tur[2] = 2`, toplam += 5 → **7**; artık `simdiki = 2`.

Adım 5 / 10

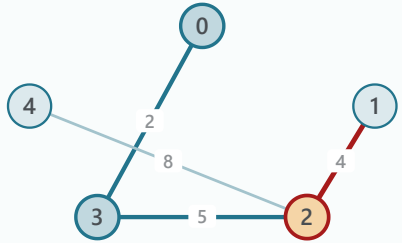
DEĞİŞKENLER

`k = 2` `simdiki = 2` `enYakin = 2` `enAz = 5` `toplam = 7` `tur = [0, 3, 2]`

Soru 4: En Yakın Komşu İzleme

5 şehrin uzaklık matrisinin satırları: 0: (0 3 4 2 7) · 1: (3 0 4 6 3) · 2: (4 4 0 5 8) · 3: (2 6 5 0 6) · 4: (7 3 8 6 0). `enYakinKomsu(d, 0)` hangi turu ve uzunluğu bulur?

```
1  int[] enYakinKomsu(int[][] d, int bas) {
2    int n = d.length, toplam = 0, simdiki = bas;
3    boolean[] ziyaret = new boolean[n];
4    int[] tur = new int[n + 1];
5    tur[0] = bas; ziyaret[bas] = true;
▶    for (int k = 1; k < n; k++) {
▶      int enYakin = -1, enAz = Integer.MAX_VALUE;
▶      for (int j = 0; j < n; j++)
▶        if (!ziyaret[j] && d[simdiki][j] < enAz) {
▶          enAz = d[simdiki][j]; enYakin = j; }
11     tur[k] = enYakin; ziyaret[enYakin] = true;
12     toplam += enAz; simdiki = enYakin;
13   }
14   tur[n] = bas; toplam += d[simdiki][bas];
15   return tur;
16 }
```



k = 3: 2. satırda ziyaret edilmemişler: d[2][1] = 4, d[2][4] = 8 → en küçüğü **1** (4).

Adım 6 / 10

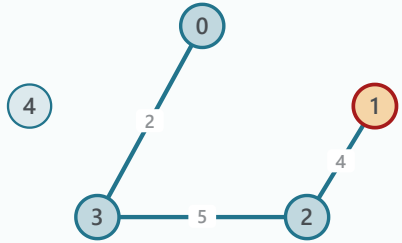
DEĞİŞKENLER

`k = 3` `simdiki = 2` `enYakin = 1` `enAz = 4` `toplam = 7` `tur = [0, 3, 2]`

Soru 4: En Yakın Komşu İzleme

5 şehrin uzaklık matrisinin satırları: 0: (0 3 4 2 7) · 1: (3 0 4 6 3) · 2: (4 4 0 5 8) · 3: (2 6 5 0 6) · 4: (7 3 8 6 0). `enYakinKomsu(d, 0)` hangi turu ve uzunluğu bulur?

```
1  int[] enYakinKomsu(int[][] d, int bas) {
2    int n = d.length, toplam = 0, simdiki = bas;
3    boolean[] ziyaret = new boolean[n];
4    int[] tur = new int[n + 1];
5    tur[0] = bas; ziyaret[bas] = true;
6    for (int k = 1; k < n; k++) {
7      int enYakin = -1, enAz = Integer.MAX_VALUE;
8      for (int j = 0; j < n; j++)
9        if (!ziyaret[j] && d[simdiki][j] < enAz) {
10         enAz = d[simdiki][j]; enYakin = j; }
▶      tur[k] = enYakin; ziyaret[enYakin] = true;
▶      toplam += enAz; simdiki = enYakin;
13   }
14   tur[n] = bas; toplam += d[simdiki][bas];
15   return tur;
16 }
```



`tur[3] = 1`, toplam += 4 → **11**; artık `simdiki = 1`.

Adım 7 / 10

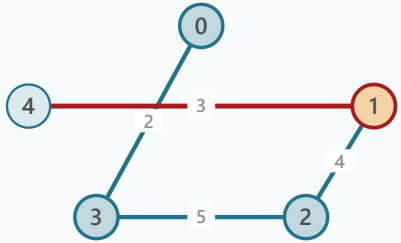
DEĞİŞKENLER

`k = 3` `simdiki = 1` `enYakin = 1` `enAz = 4` `toplam = 11` `tur = [0, 3, 2, 1]`

Soru 4: En Yakın Komşu İzleme

5 şehrin uzaklık matrisinin satırları: 0: (0 3 4 2 7) · 1: (3 0 4 6 3) · 2: (4 4 0 5 8) · 3: (2 6 5 0 6) · 4: (7 3 8 6 0). `enYakinKomsu(d, 0)` hangi turu ve uzunluğu bulur?

```
1  int[] enYakinKomsu(int[][] d, int bas) {
2    int n = d.length, toplam = 0, simdiki = bas;
3    boolean[] ziyaret = new boolean[n];
4    int[] tur = new int[n + 1];
5    tur[0] = bas; ziyaret[bas] = true;
▶    for (int k = 1; k < n; k++) {
▶      int enYakin = -1, enAz = Integer.MAX_VALUE;
▶      for (int j = 0; j < n; j++)
▶        if (!ziyaret[j] && d[simdiki][j] < enAz) {
▶          enAz = d[simdiki][j]; enYakin = j; }
11     tur[k] = enYakin; ziyaret[enYakin] = true;
12     toplam += enAz; simdiki = enYakin;
13   }
14   tur[n] = bas; toplam += d[simdiki][bas];
15   return tur;
16 }
```



k = 4: 1. satırda ziyaret edilmemişler: d[1][4] = 3 → en küçüğü **4** (3).

Adım 8 / 10

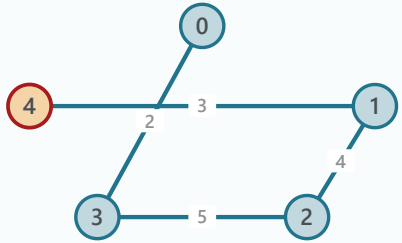
DEĞİŞKENLER

`k = 4` `simdiki = 1` `enYakin = 4` `enAz = 3` `toplam = 11` `tur = [0, 3, 2, 1]`

Soru 4: En Yakın Komşu İzleme

5 şehrin uzaklık matrisinin satırları: 0: (0 3 4 2 7) · 1: (3 0 4 6 3) · 2: (4 4 0 5 8) · 3: (2 6 5 0 6) · 4: (7 3 8 6 0). `enYakinKomsu(d, 0)` hangi turu ve uzunluğu bulur?

```
1  int[] enYakinKomsu(int[][] d, int bas) {
2    int n = d.length, toplam = 0, simdiki = bas;
3    boolean[] ziyaret = new boolean[n];
4    int[] tur = new int[n + 1];
5    tur[0] = bas; ziyaret[bas] = true;
6    for (int k = 1; k < n; k++) {
7      int enYakin = -1, enAz = Integer.MAX_VALUE;
8      for (int j = 0; j < n; j++)
9        if (!ziyaret[j] && d[simdiki][j] < enAz) {
10         enAz = d[simdiki][j]; enYakin = j; }
▶      tur[k] = enYakin; ziyaret[enYakin] = true;
▶      toplam += enAz; simdiki = enYakin;
13   }
14   tur[n] = bas; toplam += d[simdiki][bas];
15   return tur;
16 }
```



`tur[4] = 4`, toplam += 3 → **14**; artık `simdiki = 4`.

Adım 9 / 10

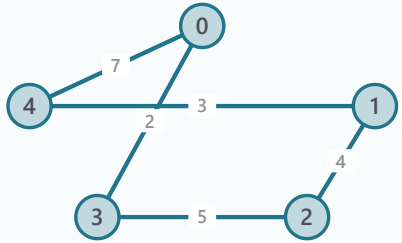
DEĞİŞKENLER

`k = 4` `simdiki = 4` `enYakin = 4` `enAz = 3` `toplam = 14` `tur = [0, 3, 2, 1, 4]`

Soru 4: En Yakın Komşu İzleme

5 şehrin uzaklık matrisinin satırları: 0: (0 3 4 2 7) · 1: (3 0 4 6 3) · 2: (4 4 0 5 8) · 3: (2 6 5 0 6) · 4: (7 3 8 6 0). `enYakinKomsu(d, 0)` hangi turu ve uzunluğu bulur?

```
1  int[] enYakinKomsu(int[][] d, int bas) {
2    int n = d.length, toplam = 0, simdiki = bas;
3    boolean[] ziyaret = new boolean[n];
4    int[] tur = new int[n + 1];
5    tur[0] = bas; ziyaret[bas] = true;
6    for (int k = 1; k < n; k++) {
7      int enYakin = -1, enAz = Integer.MAX_VALUE;
8      for (int j = 0; j < n; j++)
9        if (!ziyaret[j] && d[simdiki][j] < enAz) {
10         enAz = d[simdiki][j]; enYakin = j; }
11     tur[k] = enYakin; ziyaret[enYakin] = true;
12     toplam += enAz; simdiki = enYakin;
13   }
▶  tur[n] = bas; toplam += d[simdiki][bas];
▶  return tur;
16 }
```



Başlangıca dönülür: toplam += d[4][0] = 7 → **21**. Tur 0 → 3 → 2 → 1 → 4 → 0. Kaba kuvvetle optimum **19** (0 → 2 → 1 → 4 → 3 → 0): son dönüş kenarı (7) pahalı kaldı.

Adım 10 / 10

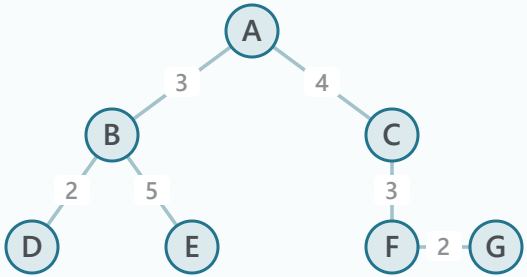
DEĞİŞKENLER

`k = 5` `simdiki = 4` `enYakin = -` `enAz = -` `toplam = 21` `tur = [0, 3, 2, 1, 4, 0]`

ÇIKTI SORUSU · GEZGİN SATICI

Soru 5: MST'den Tura

Metrik bir TSP örneğinde Prim ile bulunan minimum kapsayan ağaç şekildeki gibidir (w(T) = 19). Ağaç A'dan önce-kök (preorder) sırayla, çocuklar alfabetik sırada gezilip tekrar eden şehirler atlanıyor. Bulunan tur nedir? Tur uzunluğu ve optimum OPT için hangi sınırlar kesindir?



▶ **Cevabı göster**

ÇOKTAN SEÇMELİ · GEZGİN SATICI

Soru 6: Metrik TSP Sonuçları

TSP yaklaşım sonuçlarıyla ilgili hangisi **yanlıştır**?

A) Christofides algoritması metrik TSP'de 1,5 oranını garanti eder.

B) MST tabanlı algoritma metrik TSP'de 2 oranını garanti eder.

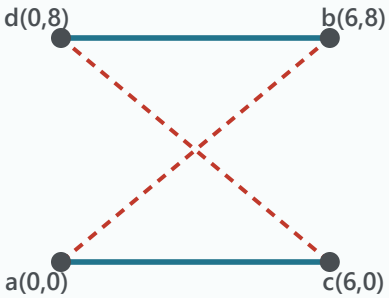
C) Öklid TSP için her $\varepsilon > 0$'da $(1 + \varepsilon)$ oranlı bir PTAS vardır.

D) En yakın komşu metrik TSP'de 1,5 oranını garanti eder.

ÇOKTAN SEÇMELİ · YEREL ARAMA

Soru 7: 2-opt Kazancı

Turda (a, b) ve (c, d) kenarları var; noktalar a(0, 0), b(6, 8), c(6, 0), d(0, 8) ve uzaklık Öklid uzaklığıdır. 2-opt hamlesinin kazancı Δ = d(a,c) + d(b,d) − d(a,b) − d(c,d) kaçtır, hamle yapılır mı?



A) Δ = −8, hamle yapılır

B) Δ = +8, hamle yapılmaz

C) Δ = −4, hamle yapılır

D) Δ = +2, hamle yapılmaz

ÇIKTI SORUSU · YEREL ARAMA

Soru 8: 2-opt Hamlesi Sonrası Tur

8 şehirli bir turda `ikiOpt` hamlesi i = 1, j = 5 için kabul ediliyor (`tersCevir(tur, i + 1, j)` , iki sınır dahil ters çevirir). Hamleden önce `tur = {0, 1, 2, 3, 4, 5, 6, 7}` ise sonra `tur` nedir; hangi iki kenar çıkar, hangileri girer?

```
for (int i = 0; i < n - 1; i++)
  for (int j = i + 2; j < n; j++) {
    int a = tur[i], b = tur[i + 1];
    int c = tur[j], e = tur[(j + 1) % n];
    double fark = d[a][c] + d[b][e]
                - d[a][b] - d[c][e];
    if (fark < -1e-9) {  // kısalıyor
      tersCevir(tur, i + 1, j);
      iyilesti = true;
    }
  }
```

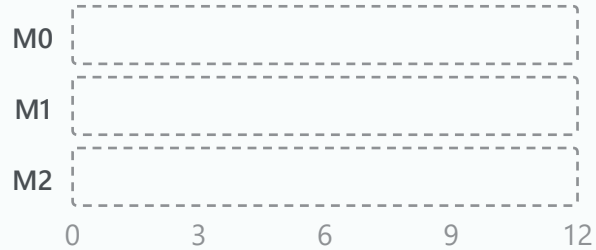
▶ **Cevabı göster**

ADIM ADIM ÇALIŞTIRMA · İŞ ÇİZELGELEME

Soru 9: Liste Çizelgeleme İzleme

Graham'ın liste çizelgelemesi her işi o an en az yüklü makineye atar. `listeCizelge(new int[] {5, 3, 4, 6, 2, 7}, 3)` çağrısında yükler nasıl değişir, C_{max} kaçtır?

```
1  int listeCizelge(int[] p, int m) {
▶    int[] yuk = new int[m];  // makine yükleri
3    for (int j = 0; j < p.length; j++) {
4      int enAz = 0;  // ilk boşalan makine
5      for (int i = 1; i < m; i++)
6        if (yuk[i] < yuk[enAz]) enAz = i;
7      yuk[enAz] += p[j];
8    }
9    int cmax = 0;
10   for (int y : yuk) cmax = Math.max(cmax,
   y);
11   return cmax;
12 }
```



Üç özdeş makine boş: `yuk = [0, 0, 0]`. İşler sırayla p = [5, 3, 4, 6, 2, 7].

Adım 1 / 8

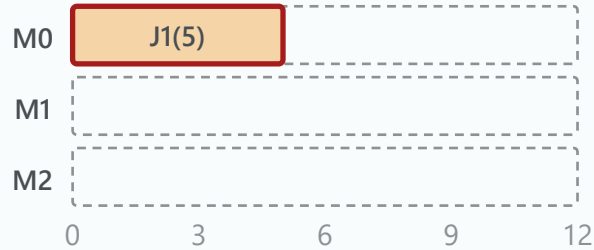
DEĞİŞKENLER

`j = –` `enAz = –` `yuk = [0, 0, 0]` `cmax = –`

Soru 9: Liste Çizelgeleme İzleme

Graham'ın liste çizelgelemesi her işi o an en az yüklü makineye atar. `listeCizelge(new int[] {5, 3, 4, 6, 2, 7}, 3)` çağrısında yükler nasıl değişir, C_{max} kaçtır?

```
1  int listeCizelge(int[] p, int m) {
2    int[] yuk = new int[m];  // makine yükleri
3    for (int j = 0; j < p.length; j++) {
▶      int enAz = 0;  // ilk boşalan makine
▶      for (int i = 1; i < m; i++)
▶        if (yuk[i] < yuk[enAz]) enAz = i;
▶      yuk[enAz] += p[j];
8    }
9    int cmax = 0;
10   for (int y : yuk) cmax = Math.max(cmax,
   y);
11   return cmax;
12 }
```



J1 (süre 5): eşit yüklerde ilk makine kalır → `yuk[0] = 0 + 5 = 5`.

Adım 2 / 8

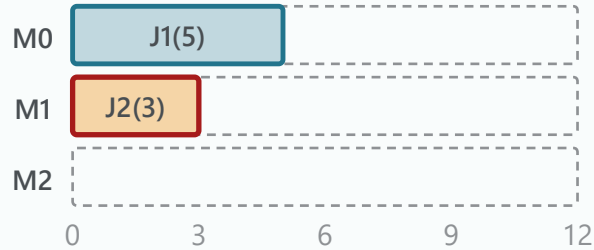
DEĞİŞKENLER

`j = 0` `enAz = 0` `yuk = [5, 0, 0]` `cmax = –`

Soru 9: Liste Çizelgeleme İzleme

Graham'ın liste çizelgelemesi her işi o an en az yüklü makineye atar. `listeCizelge(new int[] {5, 3, 4, 6, 2, 7}, 3)` çağrısında yükler nasıl değişir, C_{max} kaçtır?

```
1  int listeCizelge(int[] p, int m) {
2    int[] yuk = new int[m];  // makine yükleri
3    for (int j = 0; j < p.length; j++) {
▶      int enAz = 0;  // ilk boşalan makine
▶      for (int i = 1; i < m; i++)
▶        if (yuk[i] < yuk[enAz]) enAz = i;
▶      yuk[enAz] += p[j];
8    }
9    int cmax = 0;
10   for (int y : yuk) cmax = Math.max(cmax,
   y);
11   return cmax;
12 }
```



J2 (süre 3): en az yüklü makine M1 (0) → `yuk[1] = 0 + 3 = 3`.

Adım 3 / 8

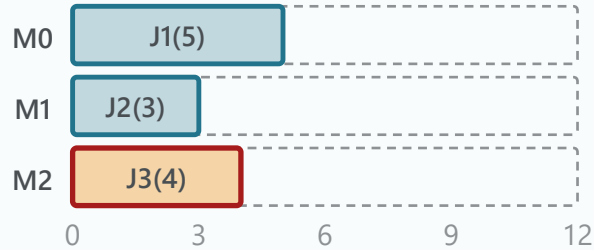
DEĞİŞKENLER

`j = 1` `enAz = 1` `yuk = [5, 3, 0]` `cmax = –`

Soru 9: Liste Çizelgeleme İzleme

Graham'ın liste çizelgelemesi her işi o an en az yüklü makineye atar. `listeCizelge(new int[] {5, 3, 4, 6, 2, 7}, 3)` çağrısında yükler nasıl değişir, C_{max} kaçtır?

```
1  int listeCizelge(int[] p, int m) {
2    int[] yuk = new int[m];  // makine yükleri
3    for (int j = 0; j < p.length; j++) {
▶      int enAz = 0;  // ilk boşalan makine
▶      for (int i = 1; i < m; i++)
▶        if (yuk[i] < yuk[enAz]) enAz = i;
▶      yuk[enAz] += p[j];
8    }
9    int cmax = 0;
10   for (int y : yuk) cmax = Math.max(cmax,
   y);
11   return cmax;
12 }
```



J3 (süre 4): en az yüklü makine M2 (0) → `yuk[2] = 0 + 4 = 4`.

Adım 4 / 8

DEĞİŞKENLER

`j = 2` `enAz = 2` `yuk = [5, 3, 4]` `cmax = –`

Soru 9: Liste Çizelgeleme İzleme

Graham'ın liste çizelgelemesi her işi o an en az yüklü makineye atar. `listeCizelge(new int[] {5, 3, 4, 6, 2, 7}, 3)` çağrısında yükler nasıl değişir, C_{max} kaçtır?

```
1  int listeCizelge(int[] p, int m) {
2    int[] yuk = new int[m];  // makine yükleri
3    for (int j = 0; j < p.length; j++) {
▶      int enAz = 0;  // ilk boşalan makine
▶      for (int i = 1; i < m; i++)
▶        if (yuk[i] < yuk[enAz]) enAz = i;
▶      yuk[enAz] += p[j];
8    }
9    int cmax = 0;
10   for (int y : yuk) cmax = Math.max(cmax,
   y);
11   return cmax;
12 }
```



J4 (süre 6): en az yüklü makine M1 (3) → `yuk[1] = 3 + 6 = 9`.

Adım 5 / 8

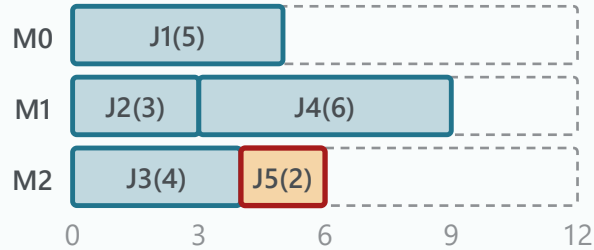
DEĞİŞKENLER

`j = 3` `enAz = 1` `yuk = [5, 9, 4]` `cmax = –`

Soru 9: Liste Çizelgeleme İzleme

Graham'ın liste çizelgelemesi her işi o an en az yüklü makineye atar. `listeCizelge(new int[] {5, 3, 4, 6, 2, 7}, 3)` çağrısında yükler nasıl değişir, C_{max} kaçtır?

```
1  int listeCizelge(int[] p, int m) {
2    int[] yuk = new int[m];  // makine yükleri
3    for (int j = 0; j < p.length; j++) {
▶      int enAz = 0;  // ilk boşalan makine
▶      for (int i = 1; i < m; i++)
▶        if (yuk[i] < yuk[enAz]) enAz = i;
▶      yuk[enAz] += p[j];
8    }
9    int cmax = 0;
10   for (int y : yuk) cmax = Math.max(cmax,
   y);
11   return cmax;
12 }
```



J5 (süre 2): en az yüklü makine M2 (4) → `yuk[2] = 4 + 2 = 6`.

Adım 6 / 8

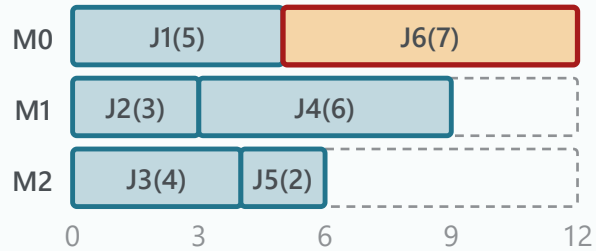
DEĞİŞKENLER

`j = 4` `enAz = 2` `yuk = [5, 9, 6]` `cmax = –`

Soru 9: Liste Çizelgeleme İzleme

Graham'ın liste çizelgelemesi her işi o an en az yüklü makineye atar. `listeCizelge(new int[] {5, 3, 4, 6, 2, 7}, 3)` çağrısında yükler nasıl değişir, C_{max} kaçtır?

```
1  int listeCizelge(int[] p, int m) {
2    int[] yuk = new int[m];  // makine yükleri
3    for (int j = 0; j < p.length; j++) {
▶      int enAz = 0;  // ilk boşalan makine
▶      for (int i = 1; i < m; i++)
▶        if (yuk[i] < yuk[enAz]) enAz = i;
▶      yuk[enAz] += p[j];
8    }
9    int cmax = 0;
10   for (int y : yuk) cmax = Math.max(cmax,
   y);
11   return cmax;
12 }
```



J6 (süre 7): en az yüklü makine M0 (5) → `yuk[0] = 5 + 7 = 12`.

Adım 7 / 8

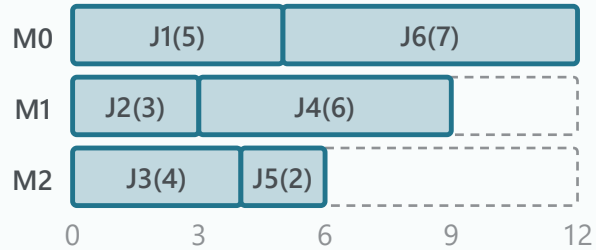
DEĞİŞKENLER

`j = 5` `enAz = 0` `yuk = [12, 9, 6]` `cmax = –`

Soru 9: Liste Çizelgeleme İzleme

Graham'ın liste çizelgelemesi her işi o an en az yüklü makineye atar. `listeCizelge(new int[] {5, 3, 4, 6, 2, 7}, 3)` çağrısında yükler nasıl değişir, C_{max} kaçtır?

```
1  int listeCizelge(int[] p, int m) {
2    int[] yuk = new int[m];  // makine yükleri
3    for (int j = 0; j < p.length; j++) {
4      int enAz = 0;  // ilk boşalan makine
5      for (int i = 1; i < m; i++)
6        if (yuk[i] < yuk[enAz]) enAz = i;
7      yuk[enAz] += p[j];
8    }
▶    int cmax = 0;
▶    for (int y : yuk) cmax = Math.max(cmax,
     y);
▶    return cmax;
12 }
```



C_{max} = **12**. Alt sınır: OPT ≥ Σp / m = 27 / 3 = 9; gerçekten {7, 2}, {5, 4}, {6, 3} ile OPT = 9. Oran 12/9 ≈ 1,33 ≤ 2 − 1/3.

Adım 8 / 8

DEĞİŞKENLER

`j = –` `enAz = –` `yuk = [12, 9, 6]` `cmax = 12`

ÇIKTI SORUSU · İŞ ÇİZELGELEME

Soru 10: LPT Sırası

İşler (5, 3, 4, 6, 2, 7) ve m = 3 için önce süreye göre **azalan** sıralanıp sonra `listeCizelge` çağrılıyor (LPT). Makine yükleri ve C_{max} nedir? Liste çizelgelemeyle karşılaştırın.

▶ **Cevabı göster**

ÇOKTAN SEÇMELİ · İŞ ÇİZELGELEME

Soru 11: Liste Çizelgelemenin Oranı

m = 4 özdeş makinede **liste çizelgeleme** (işler verildiği sırayla) hangi yaklaşım oranını garanti eder?

A) 1,75

B) 1,25

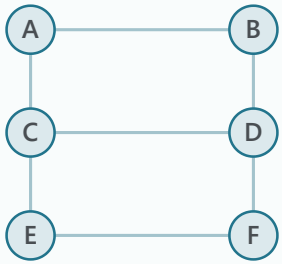
C) 2,00

D) 1,33

Soru 12: Düğüm Kapsama 2-Yaklaşımı

`dugumKapsama` şekildeki çizgenin kenarlarını verilen sırayla işliyor. Kapsama kümesi nasıl büyür; sonuç optimumla karşılaştırıldığında nasıldır?

```
1 Set<Character> dugumKapsama(char[][]
  kenarlar) {
▶   Set<Character> kapsama = new HashSet<>();
3   for (char[] k : kenarlar) {
4     char u = k[0], v = k[1];
5     if (!kapsama.contains(u) &&
  !kapsama.contains(v)) {
6       kapsama.add(u); kapsama.add(v); }  //
  iki uç
7   }
8   return kapsama;
9 }
```



Kapsama boş. Kenarlar sırayla: (A,B), (A,C), (B,D), (C,D), (C,E), (D,F), (E,F).

Adım 1 / 9

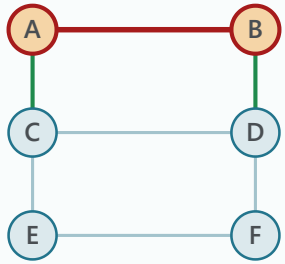
DEĞİŞKENLER

`k = –` `kapsama = {}`

Soru 12: Düğüm Kapsama 2-Yaklaşımı

`dugumKapsama` şekildeki çizgenin kenarlarını verilen sırayla işliyor. Kapsama kümesi nasıl büyür; sonuç optimumla karşılaştırıldığında nasıldır?

```
1 Set<Character> dugumKapsama(char[][]
  kenarlar) {
2   Set<Character> kapsama = new HashSet<>();
3   for (char[] k : kenarlar) {
      char u = k[0], v = k[1];
      if (!kapsama.contains(u) &&
  !kapsama.contains(v)) {
        kapsama.add(u); kapsama.add(v); }  //
  iki uç
7   }
8   return kapsama;
9 }
```



(A, B): iki uç da kapsamada değil → **A ve B eklenir**. Kenar eşleştirmeye girer.

Adım 2 / 9

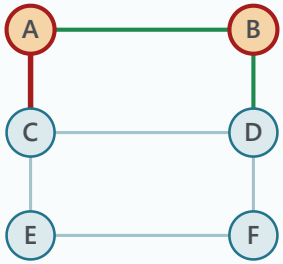
DEĞİŞKENLER

`k = (A,B)` `kapsama = {A, B}`

Soru 12: Düğüm Kapsama 2-Yaklaşımı

`dugumKapsama` şekildeki çizgenin kenarlarını verilen sırayla işliyor. Kapsama kümesi nasıl büyür; sonuç optimumla karşılaştırıldığında nasıldır?

```
1 Set<Character> dugumKapsama(char[][]
  kenarlar) {
2   Set<Character> kapsama = new HashSet<>();
3   for (char[] k : kenarlar) {
▶     char u = k[0], v = k[1];
▶     if (!kapsama.contains(u) &&
  !kapsama.contains(v)) {
6       kapsama.add(u); kapsama.add(v); }  //
  iki uç
7   }
8   return kapsama;
9 }
```



(A, C): A zaten kapsamada → kenar kapsanmış, atlanır.

Adım 3 / 9

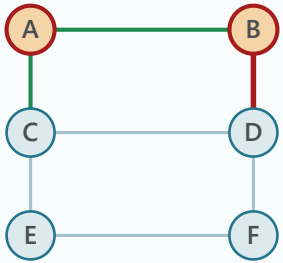
DEĞİŞKENLER

`k = (A,C)` `kapsama = {A, B}`

Soru 12: Düğüm Kapsama 2-Yaklaşımı

`dugumKapsama` şekildeki çizgenin kenarlarını verilen sırayla işliyor. Kapsama kümesi nasıl büyür; sonuç optimumla karşılaştırıldığında nasıldır?

```
1 Set<Character> dugumKapsama(char[][]
  kenarlar) {
2   Set<Character> kapsama = new HashSet<>();
3   for (char[] k : kenarlar) {
▶     char u = k[0], v = k[1];
▶     if (!kapsama.contains(u) &&
  !kapsama.contains(v)) {
6       kapsama.add(u); kapsama.add(v); }  //
  iki uç
7   }
8   return kapsama;
9 }
```



(B, D): B zaten kapsamada → kenar kapsanmış, atlanır.

Adım 4 / 9

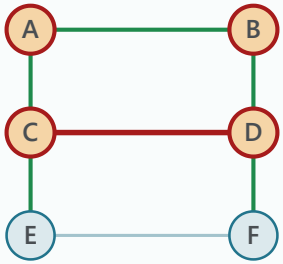
DEĞİŞKENLER

`k = (B,D)` `kapsama = {A, B}`

Soru 12: Düğüm Kapsama 2-Yaklaşımı

`dugumKapsama` şekildeki çizgenin kenarlarını verilen sırayla işliyor. Kapsama kümesi nasıl büyür; sonuç optimumla karşılaştırıldığında nasıldır?

```
1 Set<Character> dugumKapsama(char[][]
  kenarlar) {
2   Set<Character> kapsama = new HashSet<>();
3   for (char[] k : kenarlar) {
      char u = k[0], v = k[1];
      if (!kapsama.contains(u) &&
  !kapsama.contains(v)) {
        kapsama.add(u); kapsama.add(v); }  //
  iki uç
7   }
8   return kapsama;
9 }
```



(C, D): iki uç da kapsamada değil → **C ve D eklenir**. Kenar eşleştirmeye girer.

Adım 5 / 9

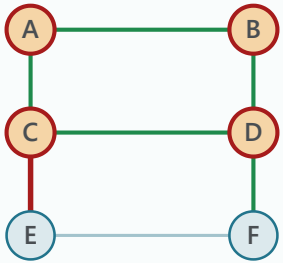
DEĞİŞKENLER

`k = (C,D)` `kapsama = {A, B, C, D}`

Soru 12: Düğüm Kapsama 2-Yaklaşımı

`dugumKapsama` şekildeki çizgenin kenarlarını verilen sırayla işliyor. Kapsama kümesi nasıl büyür; sonuç optimumla karşılaştırıldığında nasıldır?

```
1 Set<Character> dugumKapsama(char[][]
  kenarlar) {
2   Set<Character> kapsama = new HashSet<>();
3   for (char[] k : kenarlar) {
      char u = k[0], v = k[1];
      if (!kapsama.contains(u) &&
  !kapsama.contains(v)) {
6       kapsama.add(u); kapsama.add(v); }  //
  iki uç
7   }
8   return kapsama;
9 }
```



(C, E): C zaten kapsamada → kenar kapsanmış, atlanır.

Adım 6 / 9

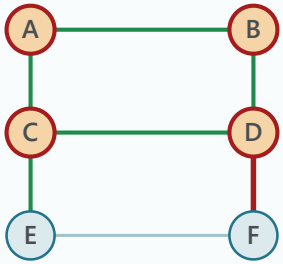
DEĞİŞKENLER

`k = (C,E)` `kapsama = {A, B, C, D}`

Soru 12: Düğüm Kapsama 2-Yaklaşımı

`dugumKapsama` şekildeki çizgenin kenarlarını verilen sırayla işliyor. Kapsama kümesi nasıl büyür; sonuç optimumla karşılaştırıldığında nasıldır?

```
1 Set<Character> dugumKapsama(char[][]
  kenarlar) {
2   Set<Character> kapsama = new HashSet<>();
3   for (char[] k : kenarlar) {
▶     char u = k[0], v = k[1];
▶     if (!kapsama.contains(u) &&
  !kapsama.contains(v)) {
6       kapsama.add(u); kapsama.add(v); }  //
  iki uç
7   }
8   return kapsama;
9 }
```



(D, F): D zaten kapsamada → kenar kapsanmış, atlanır.

Adım 7 / 9

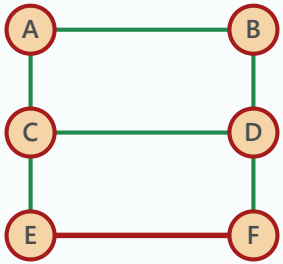
DEĞİŞKENLER

`k = (D,F)` `kapsama = {A, B, C, D}`

Soru 12: Düğüm Kapsama 2-Yaklaşımı

`dugumKapsama` şekildeki çizgenin kenarlarını verilen sırayla işliyor. Kapsama kümesi nasıl büyür; sonuç optimumla karşılaştırıldığında nasıldır?

```
1 Set<Character> dugumKapsama(char[][]
  kenarlar) {
2   Set<Character> kapsama = new HashSet<>();
3   for (char[] k : kenarlar) {
      char u = k[0], v = k[1];
      if (!kapsama.contains(u) &&
  !kapsama.contains(v)) {
        kapsama.add(u); kapsama.add(v); }  //
  iki uç
7   }
8   return kapsama;
9 }
```



(E, F): iki uç da kapsamada değil → **E ve F eklenir**. Kenar eşleştirmeye girer.

Adım 8 / 9

DEĞİŞKENLER

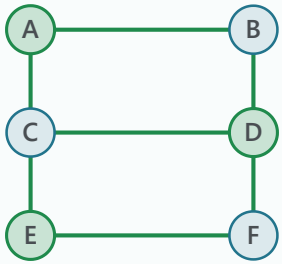
`k = (E,F)`

`kapsama = {A, B, C, D, E, F}`

Soru 12: Düğüm Kapsama 2-Yaklaşımı

`dugumKapsama` şekildeki çizgenin kenarlarını verilen sırayla işliyor. Kapsama kümesi nasıl büyür; sonuç optimumla karşılaştırıldığında nasıldır?

```
1 Set<Character> dugumKapsama(char[][]
  kenarlar) {
2   Set<Character> kapsama = new HashSet<>();
3   for (char[] k : kenarlar) {
4     char u = k[0], v = k[1];
5     if (!kapsama.contains(u) &&
  !kapsama.contains(v)) {
6       kapsama.add(u); kapsama.add(v); }  //
  iki uç
7   }
▶   return kapsama;
9 }
```



Sonuç 6 düğüm (tümü). Optimum **3**: {A, D, E} (yeşil). Oran 6/3 = **2**: sınır bu girdide sıkı. Seçilen kenarlar (A,B), (C,D), (E,F) ortak ucu olmayan bir eşleştirmedir.

Adım 9 / 9

DEĞİŞKENLER

`k = –` `kapsama = {A, B, C, D, E, F}`

ÇOKTAN SEÇMELİ · DÜĞÜM KAPSAMA

Soru 13: Yolda Kenar Sırası

1–2–3–4–5–6–7–8 yolunun kenarları `dugumKapsama`'ya şu sırayla veriliyor: (2,3), (4,5), (6,7), (1,2), (3,4), (5,6), (7,8). Algoritma kaç düğüm seçer ve optimum kaçtır?



A) 8 düğüm; optimum 4

B) 6 düğüm; optimum 3

C) 6 düğüm; optimum 4

D) 4 düğüm; optimum 4

ÇOKTAN SEÇMELİ · DÜĞÜM KAPSAMA

Soru 14: 3-SAT'tan İndirgeme

3-SAT → düğüm kapsama indirgemesinde (her değişkene bir kenar, her tümceye bir üçgen) formülde **l = 3** değişken ve **m = 4** tümce varsa aranan kapsama boyutu k kaçtır?

A) 7

B) 14

C) 11

D) 15

ÇIKTI SORUSU · SIRT ÇANTASI

Soru 15: Oran Sıralı Açgözlü

Kapasite W = 15. Eşyalar (değer, ağırlık): E1 (30, 10), E2 (20, 8), E3 (18, 7), E4 (6, 4). Değer/ağırlık oranına göre azalan sırayla *sığanı al* kuralı hangi eşyaları alır ve toplam değer kaçtır? Optimum ve iyileştirilmiş kural max(açgözlü, en değerli tek eşya) ne verir?

▶ **Cevabı göster**

ÇOKTAN SEÇMELİ · SIRT ÇANTASI

Soru 16: İyileştirilmiş Kural

W = 20; eşyalar (değer, ağırlık): (3, 1) ve (20, 20). Oran sıralı açgözlü ile *max(açgözlü değer, sığan en değerli tek eşya)* kuralı sırasıyla hangi değerleri bulur?

A) 3 ve 3

B) 20 ve 20

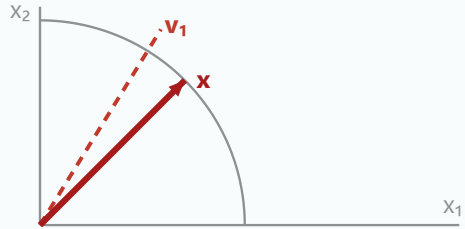
C) 23 ve 23

D) 3 ve 20

Soru 17: Kuvvet Yinelemesi İzleme

`kuvvet(new double[][]{{2, 1}, {1, 3}}, 3)` çağrısında her adımda y, ‖y‖ ve x nasıl değişir?

```
1  double[] kuvvet(double[][] a, int adim) {
▶    int n = a.length;
▶    double[] x = new double[n];
▶    Arrays.fill(x, 1.0); // x₀ ≠ 0 ve v₁'e dik değil
5    for (int k = 0; k < adim; k++) {
6      double[] y = new double[n];
7      for (int i = 0; i < n; i++)  // y = A·x
8        for (int j = 0; j < n; j++)
9          y[i] += a[i][j] * x[j];
10     double norm = 0;
11     for (double yi : y) norm += yi * yi;
12     norm = Math.sqrt(norm);
13     if (norm == 0) break; // A·x = 0
14     for (int i = 0; i < n; i++)  // ‖x‖ = 1
15       x[i] = y[i] / norm;
16   }
17   return x;
18 }
```



A = [[2, 1], [1, 3]], `x = [1, 1]` (şekilde yön olarak). Baskın özvektör $v_1 \approx (0{,}526;\ 0{,}851)$, $\lambda_1 = (5 + \sqrt{5})/2 \approx 3{,}618$.

Adım 1 / 8

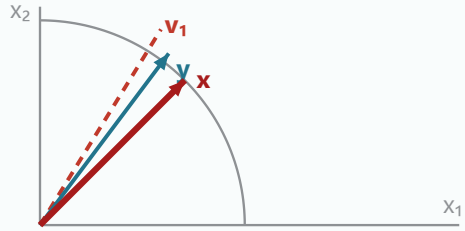
DEĞİŞKENLER

`k = –` `x = [1.0, 1.0]` `y = –` `norm = –`

Soru 17: Kuvvet Yinelemesi İzleme

`kuvvet(new double[][]{{2, 1}, {1, 3}}, 3)` çağrısında her adımda y, ||y|| ve x nasıl değişir?

```
1  double[] kuvvet(double[][] a, int adim) {
2    int n = a.length;
3    double[] x = new double[n];
4    Arrays.fill(x, 1.0); // x₀ ≠ 0 ve v₁'e dik değil
5    for (int k = 0; k < adim; k++) {
▶      double[] y = new double[n];
▶      for (int i = 0; i < n; i++)  // y = A·x
▶        for (int j = 0; j < n; j++)
▶          y[i] += a[i][j] * x[j];
▶      double norm = 0;
▶      for (double yi : y) norm += yi * yi;
▶      norm = Math.sqrt(norm);
13     if (norm == 0) break; // A·x = 0
14     for (int i = 0; i < n; i++)  // ‖x‖ = 1
15       x[i] = y[i] / norm;
16   }
17   return x;
18 }
```



k = 0: y = A·x = (3,0000; 4,0000), ||y|| = **5,0000**.

Adım 2 / 8

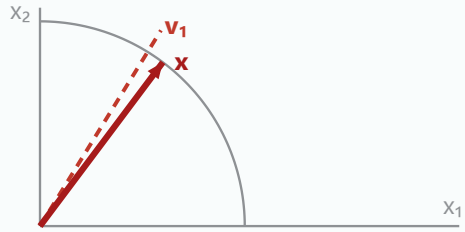
DEĞİŞKENLER

`k = 0` `x = [1.0000, 1.0000]` `y = [3.0000, 4.0000]` `norm = 5.0000`

Soru 17: Kuvvet Yinelemesi İzleme

`kuvvet(new double[][]{{2, 1}, {1, 3}}, 3)` çağrısında her adımda y, ||y|| ve x nasıl değişir?

```
1  double[] kuvvet(double[][] a, int adim) {
2    int n = a.length;
3    double[] x = new double[n];
4    Arrays.fill(x, 1.0); // x₀ ≠ 0 ve v₁'e dik değil
5    for (int k = 0; k < adim; k++) {
6      double[] y = new double[n];
7      for (int i = 0; i < n; i++)  // y = A·x
8        for (int j = 0; j < n; j++)
9          y[i] += a[i][j] * x[j];
10     double norm = 0;
11     for (double yi : y) norm += yi * yi;
12     norm = Math.sqrt(norm);
▶      if (norm == 0) break; // A·x = 0
▶      for (int i = 0; i < n; i++)  // ‖x‖ = 1
▶        x[i] = y[i] / norm;
16   }
17   return x;
18 }
```



Normalleştirme: x = y / ||y|| = (0,6000; 0,8000). Uzunluk 1 olur.

Adım 3 / 8

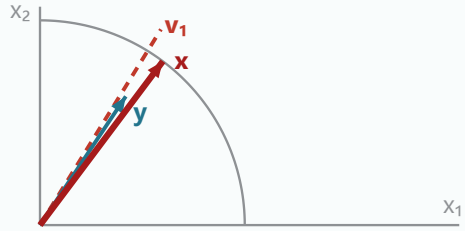
DEĞİŞKENLER

`k = 0` `x = [0.6000, 0.8000]` `y = [3.0000, 4.0000]` `norm = 5.0000`

Soru 17: Kuvvet Yinelemesi İzleme

`kuvvet(new double[][]{{2, 1}, {1, 3}}, 3)` çağrısında her adımda y, ||y|| ve x nasıl değişir?

```
1  double[] kuvvet(double[][] a, int adim) {
2    int n = a.length;
3    double[] x = new double[n];
4    Arrays.fill(x, 1.0); // x₀ ≠ 0 ve v₁'e dik değil
5    for (int k = 0; k < adim; k++) {
▶      double[] y = new double[n];
▶      for (int i = 0; i < n; i++)  // y = A·x
▶        for (int j = 0; j < n; j++)
▶          y[i] += a[i][j] * x[j];
▶      double norm = 0;
▶      for (double yi : y) norm += yi * yi;
▶      norm = Math.sqrt(norm);
13     if (norm == 0) break; // A·x = 0
14     for (int i = 0; i < n; i++)  // ‖x‖ = 1
15       x[i] = y[i] / norm;
16   }
17   return x;
18 }
```



k = 1: y = A·x = (2,0000; 3,0000), ||y|| = **3,6056** (= √13).

Adım 4 / 8

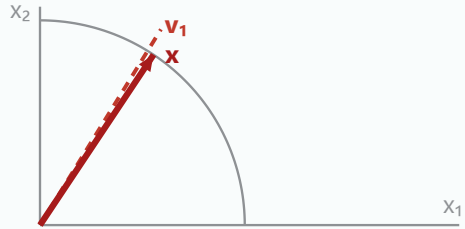
DEĞİŞKENLER

`k = 1` `x = [0.6000, 0.8000]` `y = [2.0000, 3.0000]` `norm = 3.6056`

Soru 17: Kuvvet Yinelemesi İzleme

`kuvvet(new double[][]{{2, 1}, {1, 3}}, 3)` çağrısında her adımda y, ||y|| ve x nasıl değişir?

```
1  double[] kuvvet(double[][] a, int adim) {
2    int n = a.length;
3    double[] x = new double[n];
4    Arrays.fill(x, 1.0); // x₀ ≠ 0 ve v₁'e dik değil
5    for (int k = 0; k < adim; k++) {
6      double[] y = new double[n];
7      for (int i = 0; i < n; i++)  // y = A·x
8        for (int j = 0; j < n; j++)
9          y[i] += a[i][j] * x[j];
10     double norm = 0;
11     for (double yi : y) norm += yi * yi;
12     norm = Math.sqrt(norm);
▶      if (norm == 0) break; // A·x = 0
▶      for (int i = 0; i < n; i++)  // ||x|| = 1
▶        x[i] = y[i] / norm;
16   }
17   return x;
18 }
```



Normalleştirme: x = y / ||y|| = (0,5547; 0,8321). x, v_1'e yaklaşıyor.

Adım 5 / 8

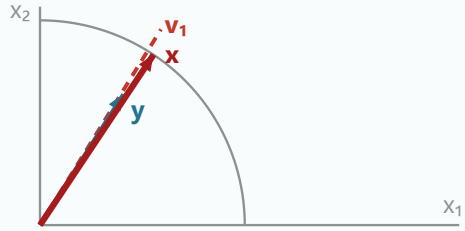
DEĞİŞKENLER

`k = 1` `x = [0.5547, 0.8321]` `y = [2.0000, 3.0000]` `norm = 3.6056`

Soru 17: Kuvvet Yinelemesi İzleme

`kuvvet(new double[][]{{2, 1}, {1, 3}}, 3)` çağrısında her adımda y, ‖y‖ ve x nasıl değişir?

```
1  double[] kuvvet(double[][] a, int adim) {
2    int n = a.length;
3    double[] x = new double[n];
4    Arrays.fill(x, 1.0); // x₀ ≠ 0 ve v₁'e dik değil
5    for (int k = 0; k < adim; k++) {
▶      double[] y = new double[n];
▶      for (int i = 0; i < n; i++)  // y = A·x
▶        for (int j = 0; j < n; j++)
▶          y[i] += a[i][j] * x[j];
▶      double norm = 0;
▶      for (double yi : y) norm += yi * yi;
▶      norm = Math.sqrt(norm);
13     if (norm == 0) break; // A·x = 0
14     for (int i = 0; i < n; i++)  // ‖x‖ = 1
15       x[i] = y[i] / norm;
16   }
17   return x;
18 }
```



k = 2: y = A·x = (1,9415; 3,0509), ‖y‖ = **3,6162**.

Adım 6 / 8

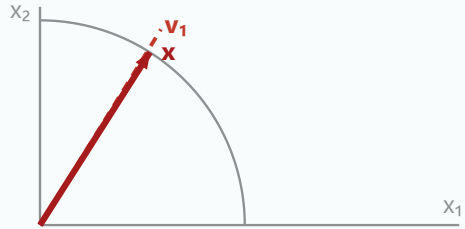
DEĞİŞKENLER

`k = 2` `x = [0.5547, 0.8321]` `y = [1.9415, 3.0509]` `norm = 3.6162`

Soru 17: Kuvvet Yinelemesi İzleme

`kuvvet(new double[][]{{2, 1}, {1, 3}}, 3)` çağrısında her adımda y, ||y|| ve x nasıl değişir?

```
1  double[] kuvvet(double[][] a, int adim) {
2    int n = a.length;
3    double[] x = new double[n];
4    Arrays.fill(x, 1.0); // x₀ ≠ 0 ve v₁'e dik değil
5    for (int k = 0; k < adim; k++) {
6      double[] y = new double[n];
7      for (int i = 0; i < n; i++)  // y = A·x
8        for (int j = 0; j < n; j++)
9          y[i] += a[i][j] * x[j];
10     double norm = 0;
11     for (double yi : y) norm += yi * yi;
12     norm = Math.sqrt(norm);
▶      if (norm == 0) break; // A·x = 0
▶      for (int i = 0; i < n; i++)  // ||x|| = 1
▶        x[i] = y[i] / norm;
16   }
17   return x;
18 }
```



Normalleştirme: x = y / ||y|| = (0,5369; 0,8437). x, v_1'e yaklaşıyor.

Adım 7 / 8

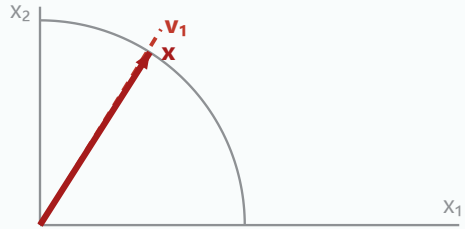
DEĞİŞKENLER

`k = 2` `x = [0.5369, 0.8437]` `y = [1.9415, 3.0509]` `norm = 3.6162`

Soru 17: Kuvvet Yinelemesi İzleme

`kuvvet(new double[][]{{2, 1}, {1, 3}}, 3)` çağrısında her adımda y, ||y|| ve x nasıl değişir?

```
1  double[] kuvvet(double[][] a, int adim) {
2    int n = a.length;
3    double[] x = new double[n];
4    Arrays.fill(x, 1.0); // x₀ ≠ 0 ve v₁'e dik değil
5    for (int k = 0; k < adim; k++) {
6      double[] y = new double[n];
7      for (int i = 0; i < n; i++)  // y = A·x
8        for (int j = 0; j < n; j++)
9          y[i] += a[i][j] * x[j];
10     double norm = 0;
11     for (double yi : y) norm += yi * yi;
12     norm = Math.sqrt(norm);
13     if (norm == 0) break; // A·x = 0
14     for (int i = 0; i < n; i++)  // ‖x‖ = 1
15       x[i] = y[i] / norm;
16   }
▶  return x;
18 }
```



3 adımda x ≈ (0,5369; 0,8437) ≈ v_1; ||A·x|| tahmini 5 → 3,606 → 3,616 → λ_1 = 3,618'e yaklaşır. Hata her adımda ≈ λ_2/λ_1 = 1,382/3,618 ≈ 0,38 katına iner.

Adım 8 / 8

DEĞİŞKENLER

`k = 3` `x = [0.5369, 0.8437]` `y = –` `norm = –`

ÇOKTAN SEÇMELİ · POWER ITERATION

Soru 18: En Hızlı Yakınsama

Dört simetrik matrisin mutlak değerce en büyük iki özdeğeri (λ_1, λ_2) verilmiştir. Power Iteration hangisinde en **hızlı** yakınsar?

A) $\lambda_1 = 10, \lambda_2 = 9$

B) $\lambda_1 = 4, \lambda_2 = 1$

C) $\lambda_1 = 3, \lambda_2 = -2{,}9$

D) $\lambda_1 = 100, \lambda_2 = 60$

Son

Bölüm 6: Yaklaşım Algoritmaları: Çalışma Soruları

Konu anlatımı: Yaklaşım Algoritmaları sunumu · Terim sözlüğü