Bölüm 7: Minimum Kapsayan Ağaç Algoritmaları

Bölüm 7: Minimum Kapsayan Ağaç Algoritmaları · Çalışma Soruları

MST kavramı | Kesim ve döngü özelliği | Kruskal | Union-find | Prim | Borůvka | Ters silme

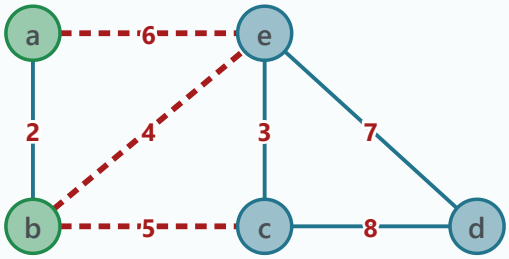
Sercan KÜLCÜ · Giresun Üniversitesi · Bilgisayar Mühendisliği

© Sercan Külcü. Tüm hakları saklıdır. · Sunumun etkileşimli HTML sürümü:
https://sercankulcu.github.io/files/algorithms/slides/Bolum_07_Minimum_Kapsayan_Agac_Algoritmalari_Calisma_Sorulari.html

ÇOKTAN SEÇMELİ · MST KAVRAMI

Soru 1: Kesim Özelliği

Düğümler {a, b} (yeşil) ve {c, d, e} (mavi) olarak ikiye ayrılıyor; kesimi geçen kenarlar kesikli çizilmiştir. Kesim özelliğine göre hangi kenar **kesinlikle** bir MST'ye aittir?



A) a–b (2): çizgedeki en hafif kenar

B) e–c (3): mavi tarafın en hafif kenarı

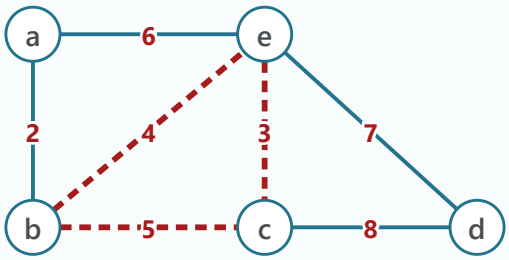
C) b–e (4): kesimi geçen en hafif kenar

D) a–e (6): a'nın kesimi geçen tek kenarı

ÇOKTAN SEÇMELİ · MST KAVRAMI

Soru 2: Döngü Özelliği

Şekildeki çizgede b–e–c döngüsü kesikli çizilmiştir. Döngü özelliği yalnızca bu döngüye uygulanırsa hangi kenarın **hiçbir** MST'de yer almadığı söylenebilir?



A) b–c (5), döngünün en ağır kenarı

B) e–c (3), döngünün en hafif kenarı

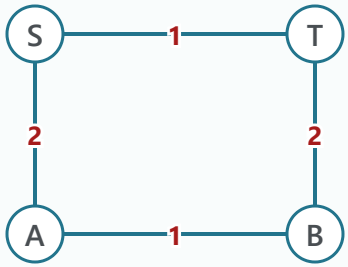
C) b–e (4), döngünün orta kenarı

D) a–b (2), döngüye komşu en hafif kenar

KLASİK SORU · MST KAVRAMI

Soru 3: Birden Çok MST

Bir çizgenin birden fazla minimum kapsayan ağacı olabilir mi? Yandaki çizge üzerinde açıklayın. Hangi koşulda MST tektir?



▶ **Cevabı göster**

ÇOKTAN SEÇMELİ · MST KAVRAMI

Soru 4: MST ve En Kısa Yol

Minimum kapsayan ağaç için aşağıdakilerden hangisi doğrudur?

A) MST'de iki düğüm arasındaki yol, çizgedeki en kısa yoldur.

B) MST algoritmaları negatif ağırlıklı kenarlarla da doğru çalışır.

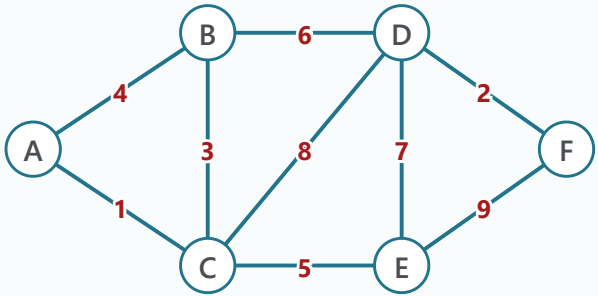
C) MST, kaynağa göre değişir; farklı kaynak farklı toplam verir.

D) Yönlü çizgede Kruskal doğrudan minimum kapsayan dallanmayı verir.

ÇIKTI SORUSU · KRUSKAL

Soru 5: Kruskal Kenar Sırası

Kruskal algoritması bu çizgede hangi kenarları hangi sırayla inceler? Hangileri eklenir, hangileri reddedilir? MST'nin toplam ağırlığı nedir?

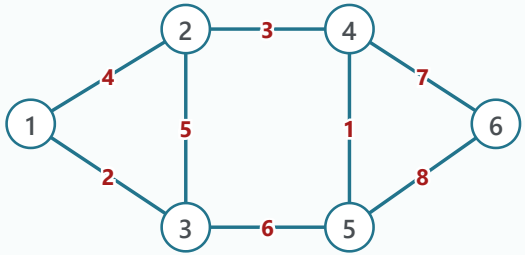


▶ **Cevabı göster**

Soru 6: Kruskal Kodu İzleme

`kruskal(6, ke)` hangi kenarları ekler? Ebeveyn dizisi (`ata`) ve w(T) nasıl değişir?

```
1  void kruskal(int V, List<Kenar> ke) {
▶    Collections.sort(ke);          // artan
3    int[] ata = new int[V + 1];
4    for (int i = 1; i <= V; i++) ata[i] = i;
5    List<Kenar> agac = new ArrayList<>();
6    int k = 0;
7    while (agac.size() < V - 1) {
8      Kenar e = ke.get(k++);
9      int x = bul(ata, e.u);
10     int y = bul(ata, e.v);
11     if (x != y) {                // ayrı ağaç
12       agac.add(e);
13       ata[x] = y;                // birleştir
14     }
15   }
16 }
```



Kenarlar artan sıralandı: (4,5)=1, (1,3)=2, (2,4)=3, (1,2)=4, (2,3)=5, (3,5)=6, (4,6)=7, (5,6)=8.

Adım 1 / 10

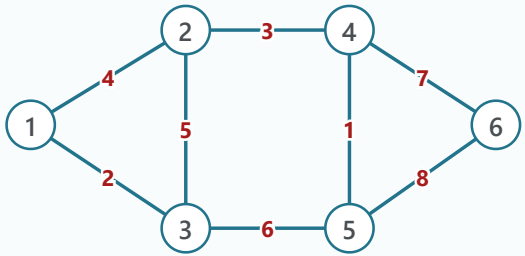
DEĞİŞKENLER

`k = 0` `ata = –` `ağaç = []` `w(T) = 0`

Soru 6: Kruskal Kodu İzleme

`kruskal(6, ke)` hangi kenarları ekler? Ebeveyn dizisi (`ata`) ve w(T) nasıl değişir?

```
1  void kruskal(int V, List<Kenar> ke) {
2    Collections.sort(ke);          // artan
▶    int[] ata = new int[V + 1];
▶    for (int i = 1; i <= V; i++) ata[i] = i;
5    List<Kenar> agac = new ArrayList<>();
6    int k = 0;
7    while (agac.size() < V - 1) {
8      Kenar e = ke.get(k++);
9      int x = bul(ata, e.u);
10     int y = bul(ata, e.v);
11     if (x != y) {                // ayrı ağaç
12       agac.add(e);
13       ata[x] = y;                // birleştir
14     }
15   }
16 }
```



ata[i] = i: her düğüm kendi ağacının kökü (6 ağaçlı orman). Ebeveyn dizisi 1..6 için gösteriliyor.

Adım 2 / 10

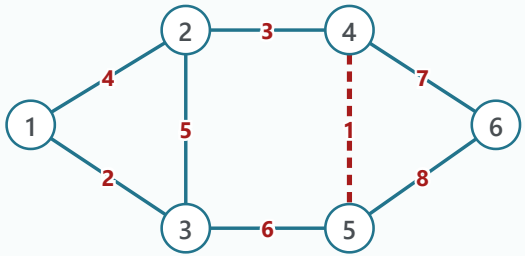
DEĞİŞKENLER

`k = 0` `ata = [1, 2, 3, 4, 5, 6]` `ağaç = []` `w(T) = 0`

Soru 6: Kruskal Kodu İzleme

`kruskal(6, ke)` hangi kenarları ekler? Ebeveyn dizisi (`ata`) ve w(T) nasıl değişir?

```
1  void kruskal(int V, List<Kenar> ke) {
2    Collections.sort(ke);          // artan
3    int[] ata = new int[V + 1];
4    for (int i = 1; i <= V; i++) ata[i] = i;
5    List<Kenar> agac = new ArrayList<>();
6    int k = 0;
7    while (agac.size() < V - 1) {
▶      Kenar e = ke.get(k++);
▶      int x = bul(ata, e.u);
▶      int y = bul(ata, e.v);
▶      if (x != y) {                // ayrı ağaç
▶        agac.add(e);
▶        ata[x] = y;                // birleştir
14     }
15   }
16 }
```



(4,5) = 1: bul(4) = 4, bul(5) = 5 → farklı ağaçlar, kenar eklenir; ata[4] = 5.

Adım 3 / 10

DEĞİŞKENLER

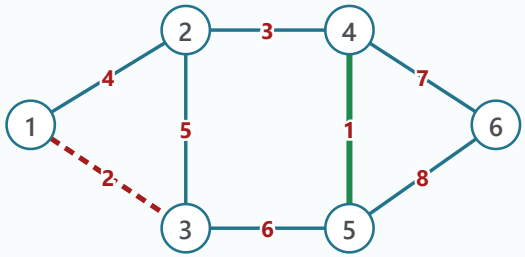
`k = 1` `e = (4,5,1)` `x = 4` `y = 5`

`ata = [1, 2, 3, 5, 5, 6]` `w(T) = 1`

Soru 6: Kruskal Kodu İzleme

`kruskal(6, ke)` hangi kenarları ekler? Ebeveyn dizisi (`ata`) ve w(T) nasıl değişir?

```
1  void kruskal(int V, List<Kenar> ke) {
2    Collections.sort(ke);          // artan
3    int[] ata = new int[V + 1];
4    for (int i = 1; i <= V; i++) ata[i] = i;
5    List<Kenar> agac = new ArrayList<>();
6    int k = 0;
7    while (agac.size() < V - 1) {
▶      Kenar e = ke.get(k++);
▶      int x = bul(ata, e.u);
▶      int y = bul(ata, e.v);
▶      if (x != y) {                // ayrı ağaç
▶        agac.add(e);
▶        ata[x] = y;                // birleştir
14     }
15   }
16 }
```



(1,3) = 2: bul(1) = 1, bul(3) = 3 → farklı ağaçlar, kenar eklenir; ata[1] = 3.

Adım 4 / 10

DEĞİŞKENLER

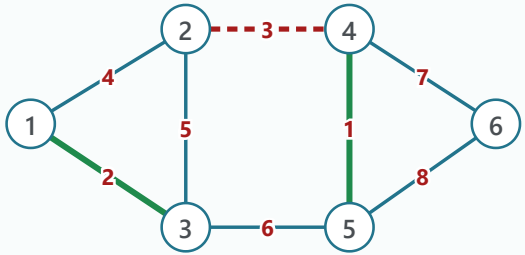
`k = 2` `e = (1,3,2)` `x = 1` `y = 3`

`ata = [3, 2, 3, 5, 5, 6]` `w(T) = 3`

Soru 6: Kruskal Kodu İzleme

`kruskal(6, ke)` hangi kenarları ekler? Ebeveyn dizisi (`ata`) ve w(T) nasıl değişir?

```
1  void kruskal(int V, List<Kenar> ke) {
2    Collections.sort(ke);          // artan
3    int[] ata = new int[V + 1];
4    for (int i = 1; i <= V; i++) ata[i] = i;
5    List<Kenar> agac = new ArrayList<>();
6    int k = 0;
7    while (agac.size() < V - 1) {
▶      Kenar e = ke.get(k++);
▶      int x = bul(ata, e.u);
▶      int y = bul(ata, e.v);
▶      if (x != y) {                // ayrı ağaç
▶        agac.add(e);
▶        ata[x] = y;                // birleştir
14     }
15   }
16 }
```



(2,4) = 3: bul(2) = 2, bul(4) = 5 → farklı ağaçlar, kenar eklenir; ata[2] = 5.

Adım 5 / 10

DEĞİŞKENLER

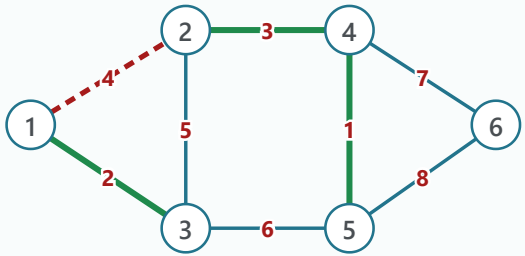
`k = 3` `e = (2,4,3)` `x = 2` `y = 5`

`ata = [3, 5, 3, 5, 5, 6]` `w(T) = 6`

Soru 6: Kruskal Kodu İzleme

`kruskal(6, ke)` hangi kenarları ekler? Ebeveyn dizisi (`ata`) ve w(T) nasıl değişir?

```
1  void kruskal(int V, List<Kenar> ke) {
2    Collections.sort(ke);          // artan
3    int[] ata = new int[V + 1];
4    for (int i = 1; i <= V; i++) ata[i] = i;
5    List<Kenar> agac = new ArrayList<>();
6    int k = 0;
7    while (agac.size() < V - 1) {
▶      Kenar e = ke.get(k++);
▶      int x = bul(ata, e.u);
▶      int y = bul(ata, e.v);
▶      if (x != y) {                // ayrı ağaç
▶        agac.add(e);
▶        ata[x] = y;                // birleştir
14     }
15   }
16 }
```



(1,2) = 4: bul(1) = 3, bul(2) = 5 → farklı ağaçlar, kenar eklenir; ata[3] = 5.

Adım 6 / 10

DEĞİŞKENLER

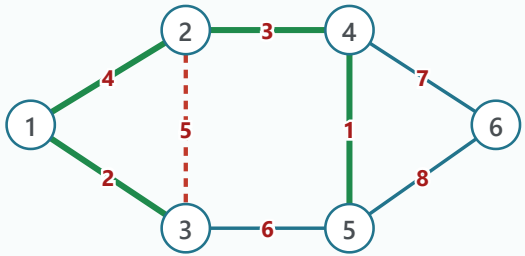
`k = 4` `e = (1,2,4)` `x = 3` `y = 5`

`ata = [3, 5, 5, 5, 5, 6]` `w(T) = 10`

Soru 6: Kruskal Kodu İzleme

`kruskal(6, ke)` hangi kenarları ekler? Ebeveyn dizisi (`ata`) ve w(T) nasıl değişir?

```
1  void kruskal(int V, List<Kenar> ke) {
2    Collections.sort(ke);          // artan
3    int[] ata = new int[V + 1];
4    for (int i = 1; i <= V; i++) ata[i] = i;
5    List<Kenar> agac = new ArrayList<>();
6    int k = 0;
7    while (agac.size() < V - 1) {
▶      Kenar e = ke.get(k++);
▶      int x = bul(ata, e.u);
▶      int y = bul(ata, e.v);
▶      if (x != y) {                // ayrı ağaç
12       agac.add(e);
13       ata[x] = y;                // birleştir
14     }
15   }
16 }
```



(2,3) = 5: bul(2) = bul(3) = 5 → aynı ağaç, eklenirse **döngü** oluşur; atlanır.

Adım 7 / 10

DEĞİŞKENLER

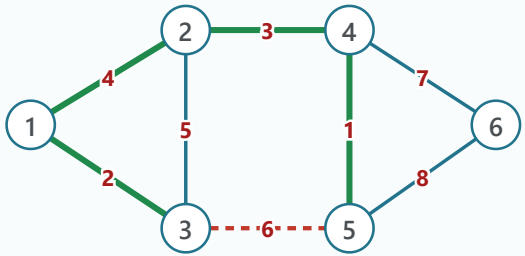
`k = 5` `e = (2,3,5)` `x = 5` `y = 5`

`ata = [3, 5, 5, 5, 5, 6]` `w(T) = 10`

Soru 6: Kruskal Kodu İzleme

`kruskal(6, ke)` hangi kenarları ekler? Ebeveyn dizisi (`ata`) ve w(T) nasıl değişir?

```
1  void kruskal(int V, List<Kenar> ke) {
2    Collections.sort(ke);          // artan
3    int[] ata = new int[V + 1];
4    for (int i = 1; i <= V; i++) ata[i] = i;
5    List<Kenar> agac = new ArrayList<>();
6    int k = 0;
7    while (agac.size() < V - 1) {
▶      Kenar e = ke.get(k++);
▶      int x = bul(ata, e.u);
▶      int y = bul(ata, e.v);
▶      if (x != y) {                // ayrı ağaç
12       agac.add(e);
13       ata[x] = y;                // birleştir
14     }
15   }
16 }
```



(3,5) = 6: bul(3) = bul(5) = 5 → aynı ağaç, eklenirse **döngü** oluşur; atlanır.

Adım 8 / 10

DEĞİŞKENLER

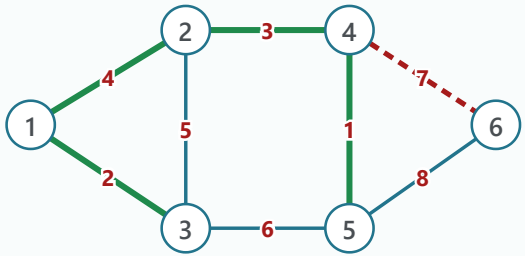
`k = 6` `e = (3,5,6)` `x = 5` `y = 5`

`ata = [3, 5, 5, 5, 5, 6]` `w(T) = 10`

Soru 6: Kruskal Kodu İzleme

`kruskal(6, ke)` hangi kenarları ekler? Ebeveyn dizisi (`ata`) ve w(T) nasıl değişir?

```
1  void kruskal(int V, List<Kenar> ke) {
2    Collections.sort(ke);          // artan
3    int[] ata = new int[V + 1];
4    for (int i = 1; i <= V; i++) ata[i] = i;
5    List<Kenar> agac = new ArrayList<>();
6    int k = 0;
7    while (agac.size() < V - 1) {
▶      Kenar e = ke.get(k++);
▶      int x = bul(ata, e.u);
▶      int y = bul(ata, e.v);
▶      if (x != y) {                // ayrı ağaç
▶        agac.add(e);
▶        ata[x] = y;                // birleştir
14     }
15   }
16 }
```



(4,6) = 7: bul(4) = 5, bul(6) = 6 → farklı ağaçlar, kenar eklenir; ata[5] = 6.

Adım 9 / 10

DEĞİŞKENLER

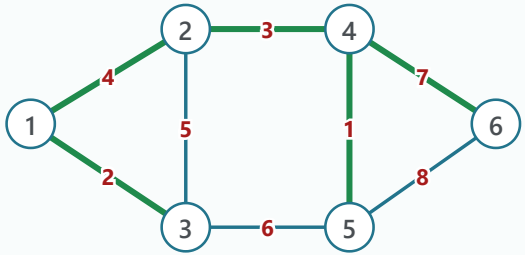
`k = 7` `e = (4,6,7)` `x = 5` `y = 6`

`ata = [3, 5, 5, 5, 6, 6]` `w(T) = 17`

Soru 6: Kruskal Kodu İzleme

`kruskal(6, ke)` hangi kenarları ekler? Ebeveyn dizisi (`ata`) ve w(T) nasıl değişir?

```
1  void kruskal(int V, List<Kenar> ke) {
2    Collections.sort(ke);          // artan
3    int[] ata = new int[V + 1];
4    for (int i = 1; i <= V; i++) ata[i] = i;
5    List<Kenar> agac = new ArrayList<>();
6    int k = 0;
▶    while (agac.size() < V - 1) {
8      Kenar e = ke.get(k++);
9      int x = bul(ata, e.u);
10     int y = bul(ata, e.v);
11     if (x != y) {                // ayrı ağaç
12       agac.add(e);
13       ata[x] = y;                // birleştir
14     }
15   }
16 }
```



agac.size() = 5 = V − 1: döngü biter; (5,6) = 8 hiç incelenmez. w(T) = **17**.

Adım 10 / 10

DEĞİŞKENLER

`k = 7` `ata = [3, 5, 5, 5, 6, 6]` `w(T) = 17`

ÇOKTAN SEÇMELİ · KRUSKAL

Soru 7: Döngü Denetimi

Bir öğrenci Kruskal'da döngü denetimini "u ile v arasında MST'de doğrudan kenar var mı?" sorusuyla yapıyor. Bu neden yanlıştır?

A) Doğrudan kenar denetimi O(1) değildir, her seferinde O(E) sürer.

B) u ve v başka kenarlar üzerinden zaten bağlı olabilir.

C) Yönsüz çizgede u–v ve v–u ayrı kenarlar sayıldığı için iki kez eklenir.

D) Bu denetim yalnızca Prim için yeterlidir, Kruskal için yeterli değildir.

Soru 8: Union-Find İzleme

Program ne yazdırır? Birleştirmeler ve yol kısaltma ormanı nasıl değiştirir?

```
1  int bul(int[] ata, int i) {
2    if (ata[i] != i)
3      ata[i] = bul(ata, ata[i]); // yol kısaltma
4    return ata[i];
5  }
6  void birlestir(int[] ata, int x, int y) {
7    ata[bul(ata, x)] = bul(ata, y);
8  }
9  // main
▶ for (int i = 0; i < 6; i++) ata[i] = i;
11 birlestir(ata, 0, 1); birlestir(ata, 1, 2);
12 birlestir(ata, 3, 4); birlestir(ata, 2, 4);
13 birlestir(ata, 5, 0);
14 System.out.println(Arrays.toString(ata));
```



Her düğüm kendi kümesinin kökü (yeşil). Oklar ata[i]'yi gösterir; kök kendini gösterir.

Adım 1 / 7

DEĞİŞKENLER

`ata = [0, 1, 2, 3, 4, 5]`

ÇIKTI

Soru 8: Union-Find İzleme

Program ne yazdırır? Birleştirmeler ve yol kısaltma ormanı nasıl değiştirir?

```
1  int bul(int[] ata, int i) {
2    if (ata[i] != i)
3      ata[i] = bul(ata, ata[i]); // yol kısaltma
4    return ata[i];
5  }
▶  void birlestir(int[] ata, int x, int y) {
▶    ata[bul(ata, x)] = bul(ata, y);
8  }
9  // main
10 for (int i = 0; i < 6; i++) ata[i] = i;
▶  birlestir(ata, 0, 1); birlestir(ata, 1, 2);
12 birlestir(ata, 3, 4); birlestir(ata, 2, 4);
13 birlestir(ata, 5, 0);
14 System.out.println(Arrays.toString(ata));
```



birlestir(0, 1): bul(0) = 0, bul(1) = 1 → ata[0] = 1.

Adım 2 / 7

DEĞİŞKENLER

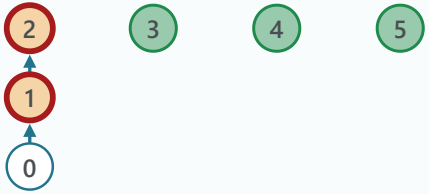
`ata = [1, 1, 2, 3, 4, 5]`

ÇIKTI

Soru 8: Union-Find İzleme

Program ne yazdırır? Birleştirmeler ve yol kısaltma ormanı nasıl değiştirir?

```
1  int bul(int[] ata, int i) {
2    if (ata[i] != i)
3      ata[i] = bul(ata, ata[i]); // yol kısaltma
4    return ata[i];
5  }
▶ void birlestir(int[] ata, int x, int y) {
▶    ata[bul(ata, x)] = bul(ata, y);
8  }
9  // main
10 for (int i = 0; i < 6; i++) ata[i] = i;
▶ birlestir(ata, 0, 1); birlestir(ata, 1, 2);
12 birlestir(ata, 3, 4); birlestir(ata, 2, 4);
13 birlestir(ata, 5, 0);
14 System.out.println(Arrays.toString(ata));
```



birlestir(1, 2): bul(1) = 1, bul(2) = 2 → ata[1] = 2.

Adım 3 / 7

DEĞİŞKENLER

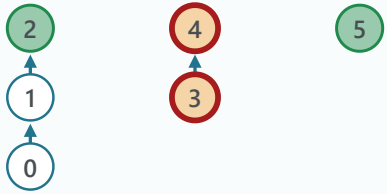
`ata = [1, 2, 2, 3, 4, 5]`

ÇIKTI

Soru 8: Union-Find İzleme

Program ne yazdırır? Birleştirmeler ve yol kısaltma ormanı nasıl değiştirir?

```
1  int bul(int[] ata, int i) {
2    if (ata[i] != i)
3      ata[i] = bul(ata, ata[i]); // yol kısaltma
4    return ata[i];
5  }
▶ void birlestir(int[] ata, int x, int y) {
▶   ata[bul(ata, x)] = bul(ata, y);
8  }
9  // main
10 for (int i = 0; i < 6; i++) ata[i] = i;
11 birlestir(ata, 0, 1); birlestir(ata, 1, 2);
▶ birlestir(ata, 3, 4); birlestir(ata, 2, 4);
13 birlestir(ata, 5, 0);
14 System.out.println(Arrays.toString(ata));
```



birlestir(3, 4): bul(3) = 3, bul(4) = 4 → ata[3] = 4.

Adım 4 / 7

DEĞİŞKENLER

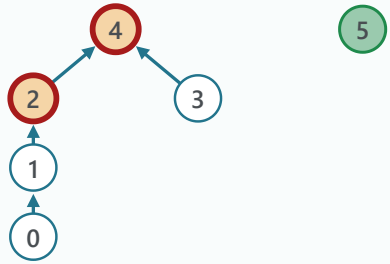
`ata = [1, 2, 2, 4, 4, 5]`

ÇIKTI

Soru 8: Union-Find İzleme

Program ne yazdırır? Birleştirmeler ve yol kısaltma ormanı nasıl değiştirir?

```
1  int bul(int[] ata, int i) {
2    if (ata[i] != i)
3      ata[i] = bul(ata, ata[i]); // yol kısaltma
4    return ata[i];
5  }
▶ void birlestir(int[] ata, int x, int y) {
▶   ata[bul(ata, x)] = bul(ata, y);
8  }
9  // main
10 for (int i = 0; i < 6; i++) ata[i] = i;
11 birlestir(ata, 0, 1); birlestir(ata, 1, 2);
▶ birlestir(ata, 3, 4); birlestir(ata, 2, 4);
13 birlestir(ata, 5, 0);
14 System.out.println(Arrays.toString(ata));
```



birlestir(2, 4): bul(2) = 2 (0 → 1 → 2 zincirinin kökü), bul(4) = 4 → ata[2] = 4. Zincir uzadı: bul(0) artık 3 adım sürer.

Adım 5 / 7

DEĞİŞKENLER

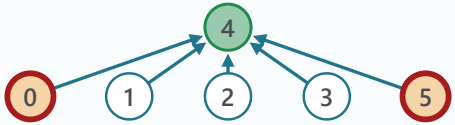
`ata = [1, 2, 4, 4, 4, 5]`

ÇIKTI

Soru 8: Union-Find İzleme

Program ne yazdırır? Birleştirmeler ve yol kısaltma ormanı nasıl değiştirir?

```
1  int bul(int[] ata, int i) {
2    if (ata[i] != i)
3      ata[i] = bul(ata, ata[i]); // yol kısaltma
4    return ata[i];
5  }
▶ void birlestir(int[] ata, int x, int y) {
▶    ata[bul(ata, x)] = bul(ata, y);
8  }
9  // main
10 for (int i = 0; i < 6; i++) ata[i] = i;
11 birlestir(ata, 0, 1); birlestir(ata, 1, 2);
12 birlestir(ata, 3, 4); birlestir(ata, 2, 4);
▶ birlestir(ata, 5, 0);
14 System.out.println(Arrays.toString(ata));
```



birlestir(5, 0): bul(5) = 5; bul(0) 0 → 1 → 2 → 4 yolunu izler ve dönerken **yol kısaltma** ile 0, 1, 2'yi doğrudan 4'e bağlar. Sonra ata[5] = 4.

Adım 6 / 7

DEĞİŞKENLER

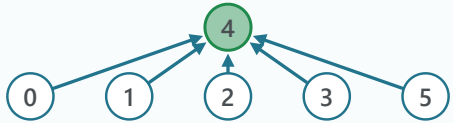
`ata = [4, 4, 4, 4, 4, 4]`

ÇIKTI

Soru 8: Union-Find İzleme

Program ne yazdırır? Birleştirmeler ve yol kısaltma ormanı nasıl değiştirir?

```
1  int bul(int[] ata, int i) {
2    if (ata[i] != i)
3      ata[i] = bul(ata, ata[i]); // yol kısaltma
4    return ata[i];
5  }
6  void birlestir(int[] ata, int x, int y) {
7    ata[bul(ata, x)] = bul(ata, y);
8  }
9  // main
10 for (int i = 0; i < 6; i++) ata[i] = i;
11 birlestir(ata, 0, 1); birlestir(ata, 1, 2);
12 birlestir(ata, 3, 4); birlestir(ata, 2, 4);
13 birlestir(ata, 5, 0);
▶  System.out.println(Arrays.toString(ata));
```



Tek küme kaldı, kökü 4. Çıktı: **[4, 4, 4, 4, 4, 4]**. Sonraki tüm bul çağrıları en çok 1 adım sürer.

Adım 7 / 7

DEĞİŞKENLER

`ata = [4, 4, 4, 4, 4, 4]`

ÇIKTI

```
[4, 4, 4, 4, 4, 4]
```

ÇIKTI SORUSU · KRUSKAL

Soru 9: Bağlı Olmayan Çizge

Kruskal kodu V = 5 ve kenarlar (1,2,3), (2,3,1), (1,3,2), (4,5,4) ile çağrılıyor. Ne olur? Kodu minimum kapsayan orman döndürecek biçimde nasıl düzeltirsiniz?

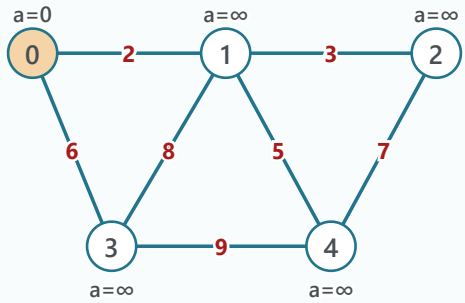
```
while (agac.size() < V - 1) {
  Kenar e = ke.get(k++);
  ...
}
```

▶ **Cevabı göster**

Soru 10: Prim Kodu İzleme

`prim(G, 0)` düğümleri hangi sırayla ağaca alır? anahtar ve ebeveyn dizileri nasıl değişir?

```
1  void prim(int[][] G, int kaynak) {
2    int V = G.length;
▶    Arrays.fill(anahtar, Integer.MAX_VALUE);
▶    anahtar[kaynak] = 0; ata[kaynak] = -1;
5    for (int s = 0; s < V - 1; s++) {
6      int u = enKucukAnahtar(anahtar, agacta);
7      agacta[u] = true;
8      for (int v = 0; v < V; v++)
9        if (G[u][v] != 0 && !agacta[v]
10           && G[u][v] < anahtar[v]) {
11         anahtar[v] = G[u][v];
12         ata[v] = u;
13       }
14   }
15 }
```



Komşuluk matrisli Prim, kaynak 0. anahtar[0] = 0, diğerleri ∞ (etiket a = anahtar).

Adım 1 / 8

DEĞİŞKENLER

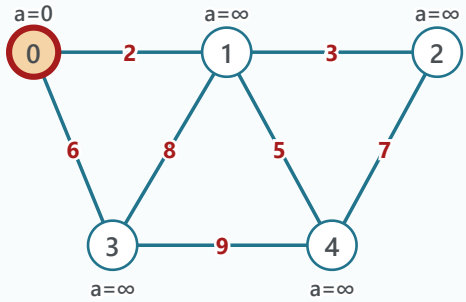
`u = –` `anahtar = [0, ∞, ∞, ∞, ∞]`

`ata = [–, –, –, –, –]`

Soru 10: Prim Kodu İzleme

`prim(G, 0)` düğümleri hangi sırayla ağaca alır? anahtar ve ebeveyn dizileri nasıl değişir?

```
1  void prim(int[][] G, int kaynak) {
2    int V = G.length;
3    Arrays.fill(anahtar, Integer.MAX_VALUE);
4    anahtar[kaynak] = 0; ata[kaynak] = -1;
▶    for (int s = 0; s < V - 1; s++) {
▶      int u = enKucukAnahtar(anahtar, agacta);
▶      agacta[u] = true;
8      for (int v = 0; v < V; v++)
9        if (G[u][v] != 0 && !agacta[v]
10           && G[u][v] < anahtar[v]) {
11         anahtar[v] = G[u][v];
12         ata[v] = u;
13       }
14   }
15 }
```



En küçük anahtarlı u = 0 ağaca girer.

Adım 2 / 8

DEĞİŞKENLER

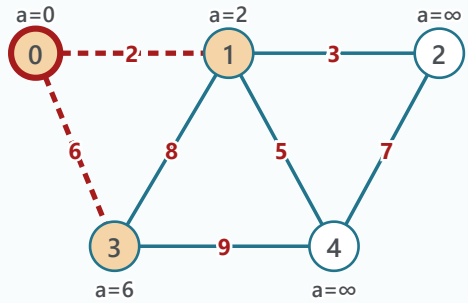
`s = 0` `u = 0` `anahtar = [0, ∞, ∞, ∞, ∞]`

`ata = [-, -, -, -, -]`

Soru 10: Prim Kodu İzleme

`prim(G, 0)` düğümleri hangi sırayla ağaca alır? anahtar ve ebeveyn dizileri nasıl değişir?

```
1  void prim(int[][] G, int kaynak) {
2    int V = G.length;
3    Arrays.fill(anahtar, Integer.MAX_VALUE);
4    anahtar[kaynak] = 0; ata[kaynak] = -1;
5    for (int s = 0; s < V - 1; s++) {
6      int u = enKucukAnahtar(anahtar, agacta);
7      agacta[u] = true;
▶      for (int v = 0; v < V; v++)
▶        if (G[u][v] != 0 && !agacta[v]
▶            && G[u][v] < anahtar[v]) {
▶          anahtar[v] = G[u][v];
▶          ata[v] = u;
13       }
14   }
15 }
```



Ağaçta olmayan komşular güncellenir: anahtar[1]: ∞ → 2, anahtar[3]: ∞ → 6; ebeveynleri 0.

Adım 3 / 8

DEĞİŞKENLER

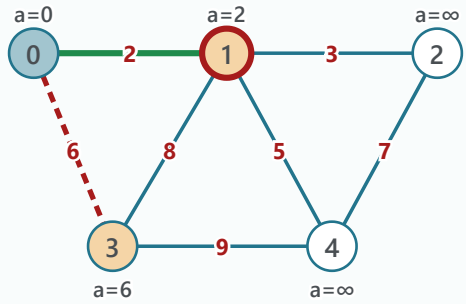
`s = 0` `u = 0` `anahtar = [0, 2, ∞, 6, ∞]`

`ata = [-, 0, -, 0, -]`

Soru 10: Prim Kodu İzleme

`prim(G, 0)` düğümleri hangi sırayla ağaca alır? anahtar ve ebeveyn dizileri nasıl değişir?

```
1  void prim(int[][] G, int kaynak) {
2    int V = G.length;
3    Arrays.fill(anahtar, Integer.MAX_VALUE);
4    anahtar[kaynak] = 0; ata[kaynak] = -1;
▶    for (int s = 0; s < V - 1; s++) {
▶      int u = enKucukAnahtar(anahtar, agacta);
▶      agacta[u] = true;
8      for (int v = 0; v < V; v++)
9        if (G[u][v] != 0 && !agacta[v]
10           && G[u][v] < anahtar[v]) {
11         anahtar[v] = G[u][v];
12         ata[v] = u;
13       }
14   }
15 }
```



En küçük anahtarlı u = 1 ağaca girer ve (0,1) = 2 kenarı MST'ye eklenir.

Adım 4 / 8

DEĞİŞKENLER

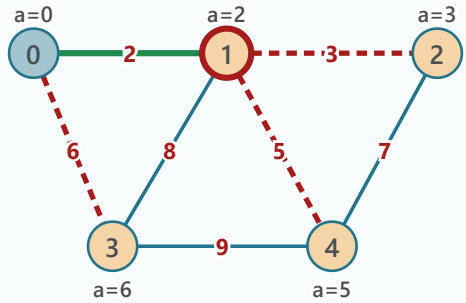
`s = 1` `u = 1` `anahtar = [0, 2, ∞, 6, ∞]`

`ata = [-, 0, -, 0, -]`

Soru 10: Prim Kodu İzleme

`prim(G, 0)` düğümleri hangi sırayla ağaca alır? anahtar ve ebeveyn dizileri nasıl değişir?

```
1  void prim(int[][] G, int kaynak) {
2    int V = G.length;
3    Arrays.fill(anahtar, Integer.MAX_VALUE);
4    anahtar[kaynak] = 0; ata[kaynak] = -1;
5    for (int s = 0; s < V - 1; s++) {
6      int u = enKucukAnahtar(anahtar, agacta);
7      agacta[u] = true;
▶      for (int v = 0; v < V; v++)
▶        if (G[u][v] != 0 && !agacta[v]
▶            && G[u][v] < anahtar[v]) {
▶          anahtar[v] = G[u][v];
▶          ata[v] = u;
13       }
14   }
15 }
```



Ağaçta olmayan komşular güncellenir: anahtar[2]: ∞ → 3, anahtar[4]: ∞ → 5; ebeveynleri 1.

Adım 5 / 8

DEĞİŞKENLER

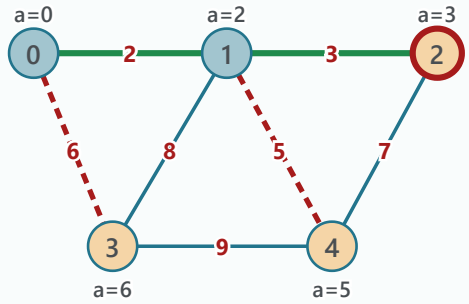
`s = 1` `u = 1` `anahtar = [0, 2, 3, 6, 5]`

`ata = [-, 0, 1, 0, 1]`

Soru 10: Prim Kodu İzleme

`prim(G, 0)` düğümleri hangi sırayla ağaca alır? anahtar ve ebeveyn dizileri nasıl değişir?

```
1  void prim(int[][] G, int kaynak) {
2    int V = G.length;
3    Arrays.fill(anahtar, Integer.MAX_VALUE);
4    anahtar[kaynak] = 0; ata[kaynak] = -1;
▶    for (int s = 0; s < V - 1; s++) {
▶      int u = enKucukAnahtar(anahtar, agacta);
▶      agacta[u] = true;
8      for (int v = 0; v < V; v++)
9        if (G[u][v] != 0 && !agacta[v]
10           && G[u][v] < anahtar[v]) {
11         anahtar[v] = G[u][v];
12         ata[v] = u;
13       }
14   }
15 }
```



En küçük anahtarlı u = 2 ağaca girer ve (1,2) = 3 kenarı MST'ye eklenir.

Adım 6 / 8

DEĞİŞKENLER

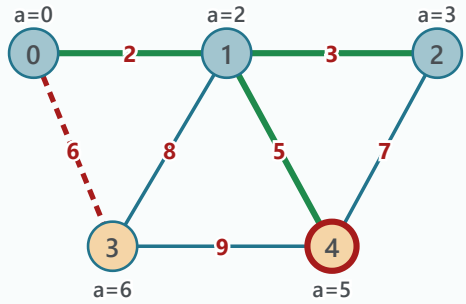
`s = 2` `u = 2` `anahtar = [0, 2, 3, 6, 5]`

`ata = [-, 0, 1, 0, 1]`

Soru 10: Prim Kodu İzleme

`prim(G, 0)` düğümleri hangi sırayla ağaca alır? anahtar ve ebeveyn dizileri nasıl değişir?

```
1  void prim(int[][] G, int kaynak) {
2    int V = G.length;
3    Arrays.fill(anahtar, Integer.MAX_VALUE);
4    anahtar[kaynak] = 0; ata[kaynak] = -1;
▶    for (int s = 0; s < V - 1; s++) {
▶      int u = enKucukAnahtar(anahtar, agacta);
▶      agacta[u] = true;
8      for (int v = 0; v < V; v++)
9        if (G[u][v] != 0 && !agacta[v]
10           && G[u][v] < anahtar[v]) {
11         anahtar[v] = G[u][v];
12         ata[v] = u;
13       }
14   }
15 }
```



En küçük anahtarlı u = 4 ağaca girer ve (1,4) = 5 kenarı MST'ye eklenir.

Adım 7 / 8

DEĞİŞKENLER

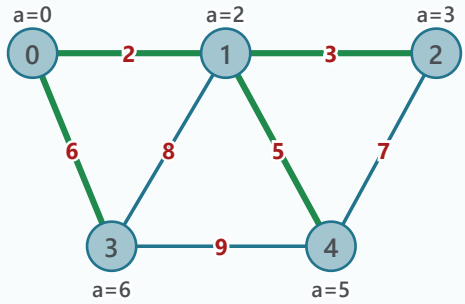
`s = 3` `u = 4` `anahtar = [0, 2, 3, 6, 5]`

`ata = [-, 0, 1, 0, 1]`

Soru 10: Prim Kodu İzleme

`prim(G, 0)` düğümleri hangi sırayla ağaca alır? anahtar ve ebeveyn dizileri nasıl değişir?

```
1  void prim(int[][] G, int kaynak) {
2    int V = G.length;
3    Arrays.fill(anahtar, Integer.MAX_VALUE);
4    anahtar[kaynak] = 0; ata[kaynak] = -1;
▶   for (int s = 0; s < V - 1; s++) {
6      int u = enKucukAnahtar(anahtar, agacta);
7      agacta[u] = true;
8      for (int v = 0; v < V; v++)
9        if (G[u][v] != 0 && !agacta[v]
10           && G[u][v] < anahtar[v]) {
11         anahtar[v] = G[u][v];
12         ata[v] = u;
13       }
14   }
15 }
```



Döngü V − 1 = 4 kez döndü; son kalan düğüm 3'ün kenarı (0,3) = 6 da hazırdır. MST = (ata[v], v) kenarları, w(T) = **16**.

Adım 8 / 8

DEĞİŞKENLER

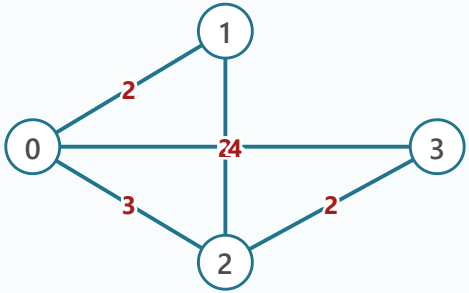
`anahtar = [0, 2, 3, 6, 5]`

`ata = [-, 0, 1, 0, 1]`

ÇIKTI SORUSU · PRIM

Soru 11: Prim'de Toplam Yazmak

Prim kodundaki koşul ve atama `anahtar[u] + G[u][v] < anahtar[v]` ve `anahtar[v] = anahtar[u] + G[u][v]` olarak yanlış yazılıyor. Kaynak 0 için (ata[v], v) kenarlarından oluşan ağaç ve toplam ağırlığı nedir? Doğru Prim ne verir?



▶ **Cevabı göster**

ÇOKTAN SEÇMELİ · PRIM

Soru 12: Prim: Matris mi, Heap mi?

V = 1000 düğümlü bir çizgede Prim'in matris sürümü ($O(V^2)$) ile ikili heap sürümü ($O(E \log V)$) karşılaştırılıyor. E = 5.000 ve E = 400.000 için hangisi daha az iş yapar?

A) E = 5.000'de heap; E = 400.000'de matris

B) Her iki durumda da heap, çünkü log V küçüktür

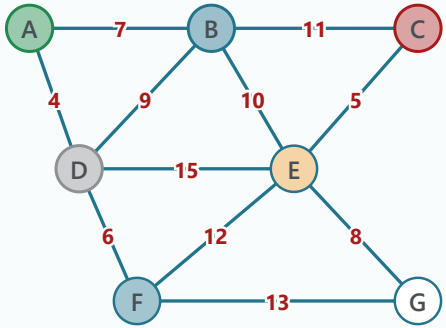
C) E = 5.000'de matris; E = 400.000'de heap

D) Her iki durumda da matris, çünkü heap ek bellek ister

Soru 13: Borůvka İzleme

Borůvka her turda hangi kenarları seçer? Kaç tur sürer, MST ağırlığı nedir?

```
▶ while (bilesenSayisi > 1) {
2   Kenar[] enUcuz = new Kenar[V];
3   for (Kenar e : kenarlar) {        // 1) en ucuzlar
4     int k1 = bul(ata, e.u), k2 = bul(ata, e.v);
5     if (k1 == k2) continue;
6     if (enUcuz[k1] == null || e.w < enUcuz[k1].w)
7       enUcuz[k1] = e;
8     if (enUcuz[k2] == null || e.w < enUcuz[k2].w)
9       enUcuz[k2] = e;
10  }
11  boolean eklendi = false;
12  for (Kenar e : enUcuz) {          // 2) hepsini
    ekle
13    if (e == null) continue;
14    int k1 = bul(ata, e.u), k2 = bul(ata, e.v);
15    if (k1 == k2) continue;
16    agac.add(e); ata[k1] = k2; bilesenSayisi--;
    eklendi = true;
17  }
18  if (!eklendi) break;  // orman
19 }
```



Başlangıçta her düğüm ayrı bir bileşen (7 bileşen, farklı renkler).

Adım 1 / 6

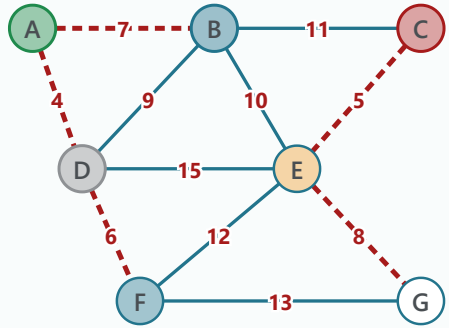
DEĞİŞKENLER

`bileşen = 7` `ağaç = []`

Soru 13: Borůvka İzleme

Borůvka her turda hangi kenarları seçer? Kaç tur sürer, MST ağırlığı nedir?

```
1  while (bilesenSayisi > 1) {
2    Kenar[] enUcuz = new Kenar[V];
▶    for (Kenar e : kenarlar) {        // 1) en ucuzlar
▶      int k1 = bul(ata, e.u), k2 = bul(ata, e.v);
▶      if (k1 == k2) continue;
▶      if (enUcuz[k1] == null || e.w < enUcuz[k1].w)
▶        enUcuz[k1] = e;
▶      if (enUcuz[k2] == null || e.w < enUcuz[k2].w)
▶        enUcuz[k2] = e;
10   }
11   boolean eklendi = false;
12   for (Kenar e : enUcuz) {          // 2) hepsini
     ekle
13     if (e == null) continue;
14     int k1 = bul(ata, e.u), k2 = bul(ata, e.v);
15     if (k1 == k2) continue;
16     agac.add(e); ata[k1] = k2; bilesenSayisi--;
     eklendi = true;
17   }
18   if (!eklendi) break;  // orman
19 }
```



Tur 1, en ucuz dış kenarlar (kesikli): {A}: AD=4, {B}: AB=7, {C}: CE=5, {D}: AD=4, {E}: CE=5, {F}: DF=6, {G}: EG=8.

Adım 2 / 6

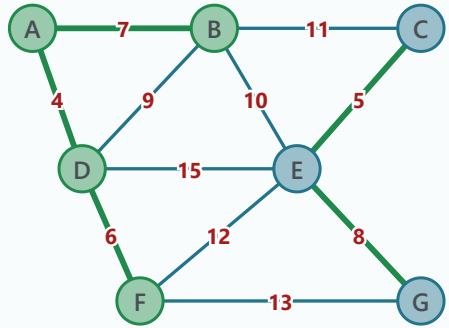
DEĞİŞKENLER

`tur = 1` `bileşen = 7` `ağaç = []`

Soru 13: Borůvka İzleme

Borůvka her turda hangi kenarları seçer? Kaç tur sürer, MST ağırlığı nedir?

```
1  while (bilesenSayisi > 1) {
2    Kenar[] enUcuz = new Kenar[V];
3    for (Kenar e : kenarlar) {        // 1) en ucuzlar
4      int k1 = bul(ata, e.u), k2 = bul(ata, e.v);
5      if (k1 == k2) continue;
6      if (enUcuz[k1] == null || e.w < enUcuz[k1].w)
7        enUcuz[k1] = e;
8      if (enUcuz[k2] == null || e.w < enUcuz[k2].w)
9        enUcuz[k2] = e;
10   }
11   boolean eklendi = false;
▶    for (Kenar e : enUcuz) {          // 2) hepsini
     ekle
▶      if (e == null) continue;
▶      int k1 = bul(ata, e.u), k2 = bul(ata, e.v);
▶      if (k1 == k2) continue;
▶      agac.add(e); ata[k1] = k2; bilesenSayisi--;
     eklendi = true;
17   }
18   if (!eklendi) break;  // orman
19 }
```



Eklenen: AD=4, AB=7, CE=5, DF=6, EG=8 (iki kez seçilen kenar ikinci kez atlanır). Kalan bileşen: 2; toplam 30.

Adım 3 / 6

DEĞİŞKENLER

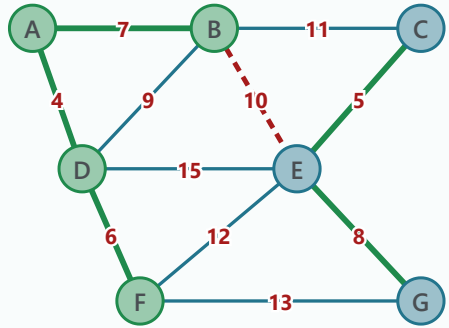
`tur = 1` `bileşen = 2`

`ağaç = [AD=4 AB=7 CE=5 DF=6 EG=8]`

Soru 13: Borůvka İzleme

Borůvka her turda hangi kenarları seçer? Kaç tur sürer, MST ağırlığı nedir?

```
1  while (bilesenSayisi > 1) {
2    Kenar[] enUcuz = new Kenar[V];
▶    for (Kenar e : kenarlar) {        // 1) en ucuzlar
▶      int k1 = bul(ata, e.u), k2 = bul(ata, e.v);
▶      if (k1 == k2) continue;
▶      if (enUcuz[k1] == null || e.w < enUcuz[k1].w)
▶        enUcuz[k1] = e;
▶      if (enUcuz[k2] == null || e.w < enUcuz[k2].w)
▶        enUcuz[k2] = e;
10   }
11   boolean eklendi = false;
12   for (Kenar e : enUcuz) {          // 2) hepsini
     ekle
13     if (e == null) continue;
14     int k1 = bul(ata, e.u), k2 = bul(ata, e.v);
15     if (k1 == k2) continue;
16     agac.add(e); ata[k1] = k2; bilesenSayisi--;
     eklendi = true;
17   }
18   if (!eklendi) break;  // orman
19 }
```



Tur 2, en ucuz dış kenarlar (kesikli): {ABDF}: BE=10, {CEG}: BE=10.

Adım 4 / 6

DEĞİŞKENLER

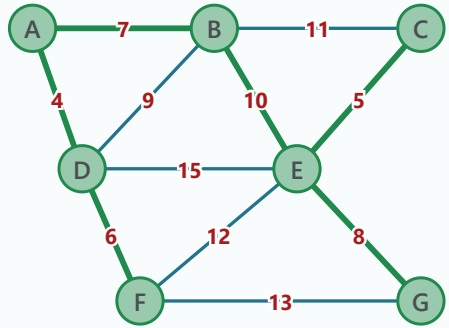
`tur = 2` `bileşen = 2`

`ağaç = [AD=4 AB=7 CE=5 DF=6 EG=8]`

Soru 13: Borůvka İzleme

Borůvka her turda hangi kenarları seçer? Kaç tur sürer, MST ağırlığı nedir?

```
1  while (bilesenSayisi > 1) {
2    Kenar[] enUcuz = new Kenar[V];
3    for (Kenar e : kenarlar) {        // 1) en ucuzlar
4      int k1 = bul(ata, e.u), k2 = bul(ata, e.v);
5      if (k1 == k2) continue;
6      if (enUcuz[k1] == null || e.w < enUcuz[k1].w)
7        enUcuz[k1] = e;
8      if (enUcuz[k2] == null || e.w < enUcuz[k2].w)
9        enUcuz[k2] = e;
10   }
11   boolean eklendi = false;
▶    for (Kenar e : enUcuz) {          // 2) hepsini
     ekle
▶      if (e == null) continue;
▶      int k1 = bul(ata, e.u), k2 = bul(ata, e.v);
▶      if (k1 == k2) continue;
▶      agac.add(e); ata[k1] = k2; bilesenSayisi--;
     eklendi = true;
17   }
18   if (!eklendi) break;  // orman
19 }
```



Eklenen: BE=10 (iki kez seçilen kenar ikinci kez atlanır). Kalan bileşen: 1; toplam 40.

Adım 5 / 6

DEĞİŞKENLER

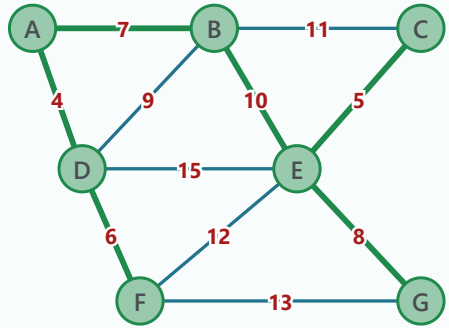
`tur = 2` `bileşen = 1`

`ağaç = [AD=4 AB=7 CE=5 DF=6 EG=8 BE=10]`

Soru 13: Borůvka İzleme

Borůvka her turda hangi kenarları seçer? Kaç tur sürer, MST ağırlığı nedir?

```
▶ while (bilesenSayisi > 1) {
2   Kenar[] enUcuz = new Kenar[V];
3   for (Kenar e : kenarlar) {        // 1) en ucuzlar
4     int k1 = bul(ata, e.u), k2 = bul(ata, e.v);
5     if (k1 == k2) continue;
6     if (enUcuz[k1] == null || e.w < enUcuz[k1].w)
7       enUcuz[k1] = e;
8     if (enUcuz[k2] == null || e.w < enUcuz[k2].w)
9       enUcuz[k2] = e;
10  }
11  boolean eklendi = false;
12  for (Kenar e : enUcuz) {          // 2) hepsini
    ekle
13    if (e == null) continue;
14    int k1 = bul(ata, e.u), k2 = bul(ata, e.v);
15    if (k1 == k2) continue;
16    agac.add(e); ata[k1] = k2; bilesenSayisi--;
    eklendi = true;
17  }
18  if (!eklendi) break;  // orman
19 }
```



Tek bileşen kaldı: 2 turda MST = **40** (6 kenar). Kenarlar sıralanmadı; her tur tüm kenarları bir kez taradı.

Adım 6 / 6

DEĞİŞKENLER

`bileşen = 1` `w(T) = 40`

ÇOKTAN SEÇMELİ · BORŮVKA

Soru 14: Borůvka Tur Sayısı

V = 1000 düğümlü bağlı bir çizgede Borůvka algoritması en fazla kaç tur çalışır ve toplam zaman nedir?

A) En çok 999 tur; O(V · E)

B) En çok 500 tur; O(E log E)

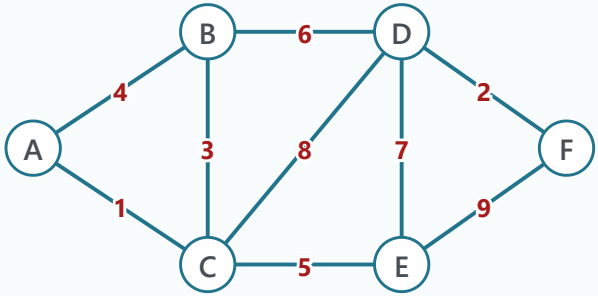
C) En çok 10 tur; O(E log V)

D) En çok 32 tur; O(E √V)

ÇIKTI SORUSU · TERS SİLME

Soru 15: Ters Silme Sonucu

Şekildeki çizgeye ters silme uygulanıyor. Hangi kenarlar silinir, hangileri kalır? Sonucu Kruskal'ınkiyle karşılaştırın.



▶ **Cevabı göster**

ÇOKTAN SEÇMELİ · TERS SİLME

Soru 16: Ters Silmenin Maliyeti

Ters silme algoritması her kenar için bir BFS/DFS ile bağlılık testi yapıyor. Toplam zaman karmaşıklığı nedir?

A) O(E log E), çünkü sıralama baskındır

B) O(V + E), çünkü her kenar bir kez silinir

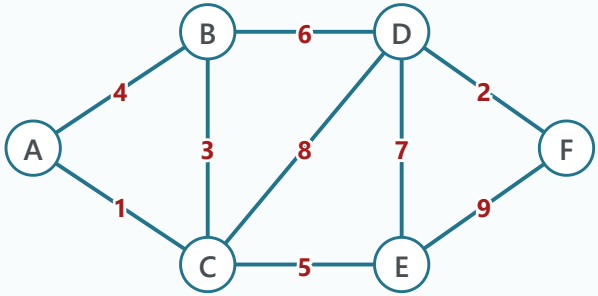
C) O(V · E), çünkü her test O(V) sürer

D) O(E · (V + E)), çünkü E test O(V + E) sürer

ÇIKTI SORUSU · KARŞILAŞTIRMA

Soru 17: Darboğaz Yol

Şekildeki çizgede E'den F'ye giden yollar arasında **en ağır kenarı en hafif** olan yol (darboğaz yol) hangisidir? MST'yi kullanarak bulun. Bu yol en kısa yol mudur?



▶ **Cevabı göster**

ÇOKTAN SEÇMELİ · KARŞILAŞTIRMA

Soru 18: Hangi Algoritma?

Aşağıdaki eşleştirmelerden hangisi **yanlıştır**?

A) Seyrek çizge, kenar listesi hazır → Kruskal, $O(E \log E)$

B) Yoğun çizge, komşuluk matrisi → Prim (dizi taramalı), $O(V^2)$

C) Her bileşenin seçimini paralel işlemcilere dağıtmak → Borůvka algoritması

D) Bağlı olmayan çizgede minimum kapsayan orman → tek başlangıçlı Prim

Son

Bölüm 7: Minimum Kapsayan Ağaç Algoritmaları: Çalışma Soruları

Konu anlatımı: Minimum kapsayan ağaç sunumu · Terim sözlüğü