Bölüm 6: En Kısa Yol Algoritmaları

Bölüm 6: En Kısa Yol Algoritmaları · Çalışma Soruları

Gevşetme | Dijkstra | Bellman-Ford | Negatif döngü | Floyd-Warshall | A* algoritması | Karşılaştırma

Sercan KÜLCÜ · Giresun Üniversitesi · Bilgisayar Mühendisliği

© Sercan Külcü. Tüm hakları saklıdır. · Sunumun etkileşimli HTML sürümü:
https://sercankulcu.github.io/files/algorithms/slides/Bolum_06_En_Kisa_Yol_Algoritmalari_Calisma_Sorulari.html

Soru 1: Gevşetme

d[u] = 6, d[v] = 10 ve w(u, v) = 3 iken `RELAX(u, v, w)` çağrılıyor. Sonra d[v] ve π[v] ne olur?



A) d[v] = 10, π[v] değişmez

B) d[v] = 9, π[v] değişmez

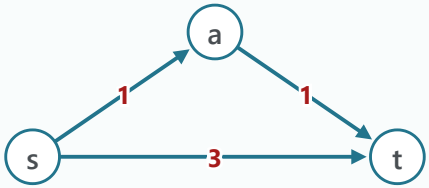
C) d[v] = 3, π[v] = u

D) d[v] = 9, π[v] = u

ÇOKTAN SEÇMELİ · TEMEL KAVRAMLAR

Soru 2: Ağırlıklara Sabit Eklemek

Çizgedeki **her** kenarın ağırlığına 2 eklenirse s'den t'ye en kısa yol ne olur?



A) Değişmez: s → a → t yine en kısa yoldur, uzunluğu 6 olur.

B) Değişir: s → t (5) artık s → a → t'den (6) kısadır.

C) Değişmez: tüm yollar aynı miktarda uzar, sıralama korunur.

D) Değişir: s → a → t'nin uzunluğu 4 olur, s → t 5 kalır.

ÇOKTAN SEÇMELİ · TEMEL KAVRAMLAR

Soru 3: Negatif Döngü

Kaynaktan ulaşılabilen bir negatif döngü varsa aşağıdakilerden hangisi doğrudur?

A) Döngüdeki ve döngüden ulaşılabilen düğümlerin en kısa yol uzunluğu tanımsızdır ($-\infty$).

B) Dijkstra bu döngüyü bulur ve döngüdeki düğümlerin uzaklığını $-\infty$ yapar.

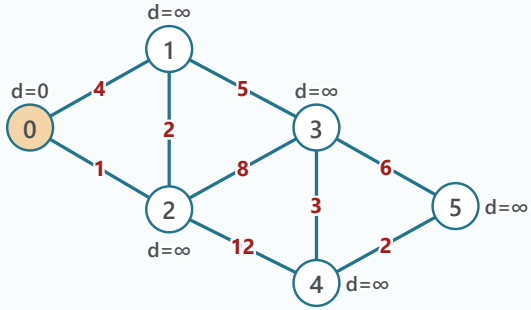
C) En kısa yollar yine en fazla $V - 1$ kenarlıdır; yalnızca değerleri negatiftir.

D) Yalnızca döngüdeki düğümler etkilenir; döngüden ulaşılan düğümler etkilenmez.

Soru 4: Dijkstra (Dizi) İzleme

`dijkstra(0)` düğümleri hangi sırayla kesinleştirir? Son d ve onceki dizileri nedir?

```
1  void dijkstra(int s) {
▶    Arrays.fill(d, SONSUZ); d[s] = 0;
3    for (int i = 0; i < V; i++) {
4      int u = -1;                  // en küçük d
5      for (int v = 0; v < V; v++)
6        if (!S[v] && (u == -1 || d[v] < d[u])) u
   = v;
7      S[u] = true;                 // kesinleşti
8      for (Kenar k : komsu[u])
9        if (!S[k.hedef] && d[u] + k.w <
   d[k.hedef]) {
10         d[k.hedef] = d[u] + k.w;
11         onceki[k.hedef] = u;
12       }
13   }
14 }
```



Yönsüz ağırlıklı çizge, kaynak 0. d[0] = 0, diğerleri ∞; hiçbir düğüm kesinleşmedi.

Adım 1 / 8

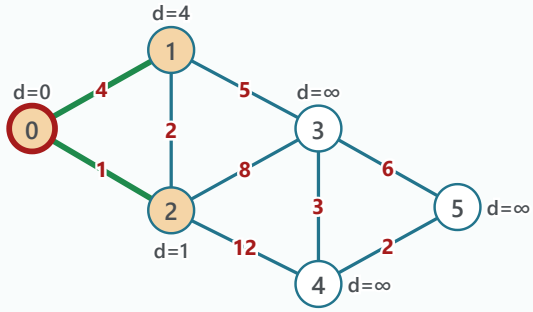
DEĞİŞKENLER

`u = –` `d = [0, ∞, ∞, ∞, ∞, ∞]` `S = {}`

Soru 4: Dijkstra (Dizi) İzleme

`dijkstra(0)` düğümleri hangi sırayla kesinleştirir? Son d ve onceki dizileri nedir?

```
1  void dijkstra(int s) {
2    Arrays.fill(d, SONSUZ); d[s] = 0;
3    for (int i = 0; i < V; i++) {
▶      int u = -1;                  // en küçük d
▶      for (int v = 0; v < V; v++)
▶        if (!S[v] && (u == -1 || d[v] < d[u])) u
   = v;
▶      S[u] = true;                 // kesinleşti
▶      for (Kenar k : komsu[u])
▶        if (!S[k.hedef] && d[u] + k.w <
   d[k.hedef]) {
▶          d[k.hedef] = d[u] + k.w;
▶          onceki[k.hedef] = u;
12       }
13   }
14 }
```



En küçük d'li kesinleşmemiş düğüm **0** (d = 0) seçilir. Gevşetilen komşular: 1: ∞ → 4, 2: ∞ → 1.

Adım 2 / 8

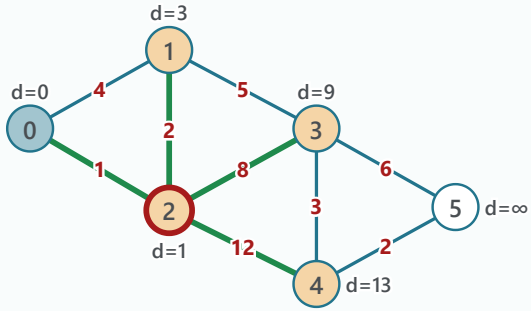
DEĞİŞKENLER

`u = 0` `d = [0, 4, 1, ∞, ∞, ∞]` `S = {0}`

Soru 4: Dijkstra (Dizi) İzleme

`dijkstra(0)` düğümleri hangi sırayla kesinleştirir? Son d ve onceki dizileri nedir?

```
1  void dijkstra(int s) {
2    Arrays.fill(d, SONSUZ); d[s] = 0;
3    for (int i = 0; i < V; i++) {
▶      int u = -1;                  // en küçük d
▶      for (int v = 0; v < V; v++)
▶        if (!S[v] && (u == -1 || d[v] < d[u])) u
   = v;
▶      S[u] = true;                 // kesinleşti
▶      for (Kenar k : komsu[u])
▶        if (!S[k.hedef] && d[u] + k.w <
   d[k.hedef]) {
▶          d[k.hedef] = d[u] + k.w;
▶          onceki[k.hedef] = u;
12       }
13   }
14 }
```



En küçük d'li kesinleşmemiş düğüm **2** (d = 1) seçilir. Gevşetilen komşular: 1: 4 → 3, 3: ∞ → 9, 4: ∞ → 13.

Adım 3 / 8

DEĞİŞKENLER

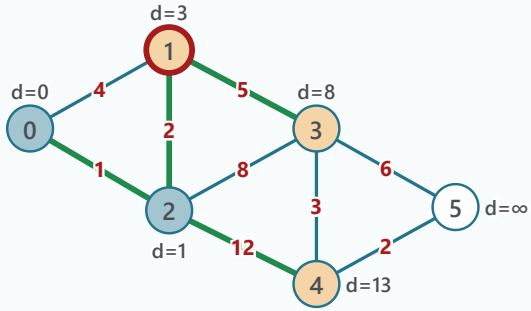
`u = 2` `d = [0, 3, 1, 9, 13, ∞]`

`S = {0, 2}`

Soru 4: Dijkstra (Dizi) İzleme

`dijkstra(0)` düğümleri hangi sırayla kesinleştirir? Son d ve onceki dizileri nedir?

```
1  void dijkstra(int s) {
2    Arrays.fill(d, SONSUZ); d[s] = 0;
3    for (int i = 0; i < V; i++) {
▶      int u = -1;                  // en küçük d
▶      for (int v = 0; v < V; v++)
▶        if (!S[v] && (u == -1 || d[v] < d[u])) u
   = v;
▶      S[u] = true;                 // kesinleşti
▶      for (Kenar k : komsu[u])
▶        if (!S[k.hedef] && d[u] + k.w <
   d[k.hedef]) {
▶          d[k.hedef] = d[u] + k.w;
▶          onceki[k.hedef] = u;
12       }
13   }
14 }
```



En küçük d'li kesinleşmemiş düğüm **1** (d = 3) seçilir. Gevşetilen komşular: 3: 9 → 8.

Adım 4 / 8

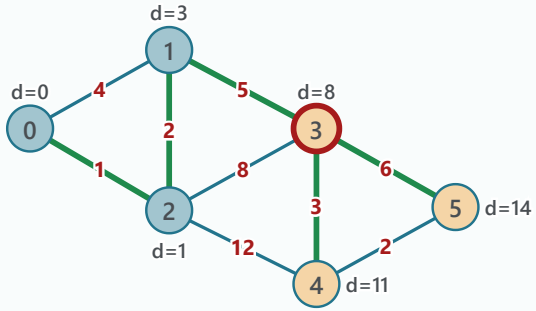
DEĞİŞKENLER

`u = 1` `d = [0, 3, 1, 8, 13, ∞]` `S = {0, 1, 2}`

Soru 4: Dijkstra (Dizi) İzleme

`dijkstra(0)` düğümleri hangi sırayla kesinleştirir? Son d ve onceki dizileri nedir?

```
1  void dijkstra(int s) {
2    Arrays.fill(d, SONSUZ); d[s] = 0;
3    for (int i = 0; i < V; i++) {
▶      int u = -1;                  // en küçük d
▶      for (int v = 0; v < V; v++)
▶        if (!S[v] && (u == -1 || d[v] < d[u])) u
   = v;
▶      S[u] = true;                 // kesinleşti
▶      for (Kenar k : komsu[u])
▶        if (!S[k.hedef] && d[u] + k.w <
   d[k.hedef]) {
▶          d[k.hedef] = d[u] + k.w;
▶          onceki[k.hedef] = u;
12       }
13   }
14 }
```



En küçük d'li kesinleşmemiş düğüm **3** (d = 8) seçilir. Gevşetilen komşular: 4: 13 → 11, 5: ∞ → 14.

Adım 5 / 8

DEĞİŞKENLER

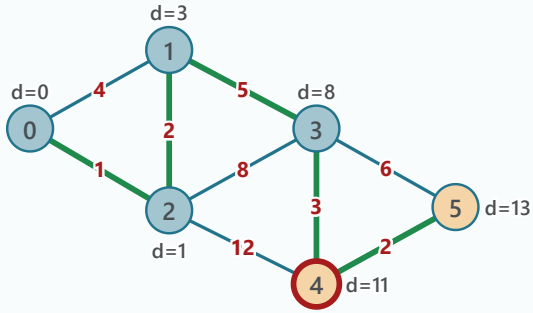
`u = 3` `d = [0, 3, 1, 8, 11, 14]`

`S = {0, 1, 2, 3}`

Soru 4: Dijkstra (Dizi) İzleme

`dijkstra(0)` düğümleri hangi sırayla kesinleştirir? Son d ve onceki dizileri nedir?

```
1  void dijkstra(int s) {
2    Arrays.fill(d, SONSUZ); d[s] = 0;
3    for (int i = 0; i < V; i++) {
▶      int u = -1;                  // en küçük d
▶      for (int v = 0; v < V; v++)
▶        if (!S[v] && (u == -1 || d[v] < d[u])) u
   = v;
▶      S[u] = true;                 // kesinleşti
▶      for (Kenar k : komsu[u])
▶        if (!S[k.hedef] && d[u] + k.w <
   d[k.hedef]) {
▶          d[k.hedef] = d[u] + k.w;
▶          onceki[k.hedef] = u;
12       }
13   }
14 }
```



En küçük d'li kesinleşmemiş düğüm **4** (d = 11) seçilir. Gevşetilen komşular: 5: 14 → 13.

Adım 6 / 8

DEĞİŞKENLER

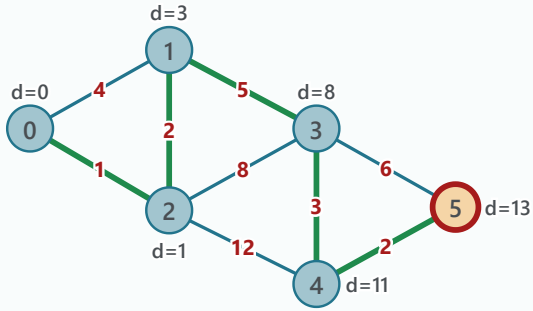
`u = 4` `d = [0, 3, 1, 8, 11, 13]`

`S = {0, 1, 2, 3, 4}`

Soru 4: Dijkstra (Dizi) İzleme

`dijkstra(0)` düğümleri hangi sırayla kesinleştirir? Son d ve onceki dizileri nedir?

```
1  void dijkstra(int s) {
2    Arrays.fill(d, SONSUZ); d[s] = 0;
3    for (int i = 0; i < V; i++) {
▶      int u = -1;                  // en küçük d
▶      for (int v = 0; v < V; v++)
▶        if (!S[v] && (u == -1 || d[v] < d[u])) u
   = v;
▶      S[u] = true;                 // kesinleşti
▶      for (Kenar k : komsu[u])
▶        if (!S[k.hedef] && d[u] + k.w <
   d[k.hedef]) {
10       d[k.hedef] = d[u] + k.w;
11       onceki[k.hedef] = u;
12     }
13   }
14 }
```



En küçük d'li kesinleşmemiş düğüm **5** (d = 13) seçilir. Hiçbir komşu iyileşmez.

Adım 7 / 8

DEĞİŞKENLER

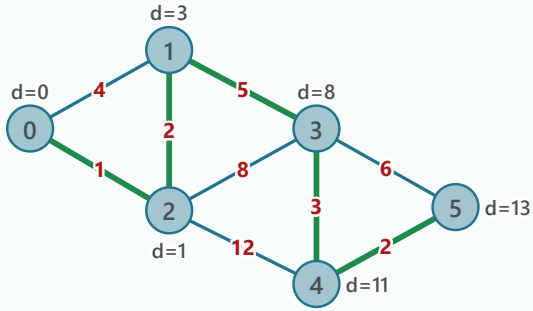
`u = 5` `d = [0, 3, 1, 8, 11, 13]`

`S = {0, 1, 2, 3, 4, 5}`

Soru 4: Dijkstra (Dizi) İzleme

`dijkstra(0)` düğümleri hangi sırayla kesinleştirir? Son d ve onceki dizileri nedir?

```
1  void dijkstra(int s) {
2    Arrays.fill(d, SONSUZ); d[s] = 0;
▶    for (int i = 0; i < V; i++) {
4      int u = -1;                  // en küçük d
5      for (int v = 0; v < V; v++)
6        if (!S[v] && (u == -1 || d[v] < d[u])) u
   = v;
7      S[u] = true;                 // kesinleşti
8      for (Kenar k : komsu[u])
9        if (!S[k.hedef] && d[u] + k.w <
   d[k.hedef]) {
10         d[k.hedef] = d[u] + k.w;
11         onceki[k.hedef] = u;
12       }
13   }
14 }
```



Bitti. Seçilme sırası 0, 2, 1, 3, 4, 5. Yeşil kenarlar en kısa yol ağacıdır; örneğin 5'e yol 0 → 2 → 1 → 3 → 4 → 5 (13).

Adım 8 / 8

DEĞİŞKENLER

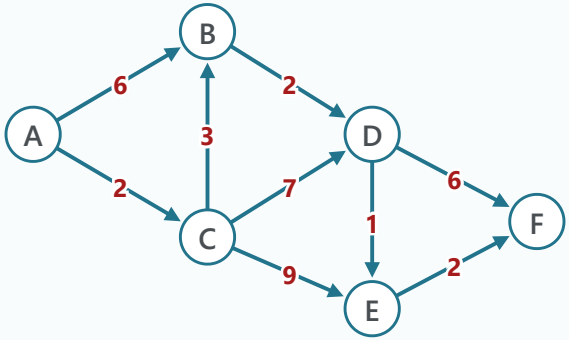
`u = -` `d = [0, 3, 1, 8, 11, 13]`

`onceki = [-, 2, 0, 1, 3, 4]`

ÇIKTI SORUSU · DIJKSTRA

Soru 5: Dijkstra Tablosu

Kaynak A. Dijkstra düğümleri hangi sırayla kesinleştirir? Son d ve π değerleri nedir? A'dan F'ye en kısa yol hangisidir?



▶ **Cevabı göster**

ÇIKTI SORUSU · DIJKSTRA

Soru 6: Yolu Geri Kurmak

Kaynak 4 için Dijkstra `onceki = {3, 4, 0, 4, −1, 2}` dizisini üretti (indis 0–5). `yolYaz(onceki, uzaklik, 4, 1)` ne yazdırır (bütün düğümler ulaşılabilir)? Hedef 5 olsaydı ne yazdırırdı?

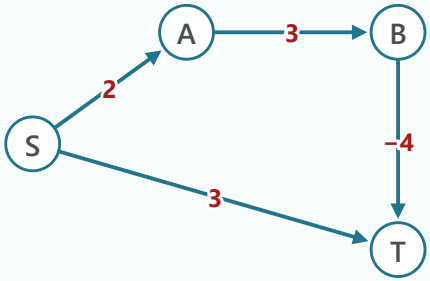
```
void yolYaz(int[] onceki, int[] uzaklik,
            int kaynak, int hedef) {
  if (uzaklik[hedef] == SONSUZ) {   // ulaşılamaz
    System.out.print("yol yok"); return;
  }
  Deque<Integer> yigin = new ArrayDeque<>();
  for (int v = hedef; v != kaynak; v = onceki[v])
    yigin.push(v);
  yigin.push(kaynak);
  while (!yigin.isEmpty())
    System.out.print(yigin.pop() + " ");
}
```

▶ **Cevabı göster**

ÇOKTAN SEÇMELİ · DIJKSTRA

Soru 7: Negatif Kenarda Dijkstra

Dijkstra (kesinleşmiş düğüme bir daha dokunmayan sürüm) S'den çalıştırılıyor. Bulduğu d[T] ve gerçek en kısa uzaklık nedir?



A) d[T] = 1, gerçek 1

B) d[T] = 3, gerçek 3

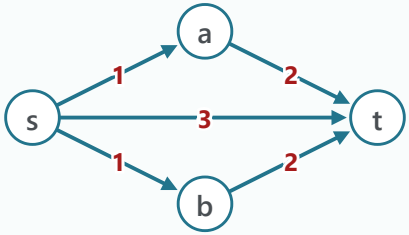
C) d[T] = 3, gerçek 1

D) d[T] = 5, gerçek 1

KLASİK SORU · DIJKSTRA

Soru 8: En Kısa Yol Sayısı

Dijkstra'yı, her düğüme giden **farklı en kısa yolların sayısını** (sayi[v]) da hesaplayacak biçimde nasıl değiştirirsiniz? Kenarlar s → a (1), s → b (1), a → t (2), b → t (2), s → t (3) olan çizgede sayi[t] kaçtır?

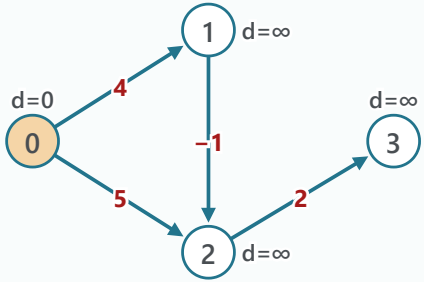


▶ **Cevabı göster**

Soru 9: Bellman-Ford İzleme

Kaynak 0 için Bellman-Ford hangi turda hangi kenarları gevşetir? Son d dizisi nedir?

```
for (int i = 1; i < V; i++) {
  boolean degisti = false;
  for (Kenar k : kenarlar)
    if (d[k.u] != SONSUZ && d[k.u] + k.w < d[k.v]) {
      d[k.v] = d[k.u] + k.w;
      degisti = true;
    }
  if (!degisti) break;              // erken durma
}
for (Kenar k : kenarlar)             // denetim
  if (d[k.u] != SONSUZ && d[k.u] + k.w < d[k.v])
    throw new IllegalStateException("Negatif döngü");
```



Kaynak 0, V = 4. Kenar listesi bu sırayla incelenir: (2,3,2), (1,2,−1), (0,1,4), (0,2,5).

Adım 1 / 10

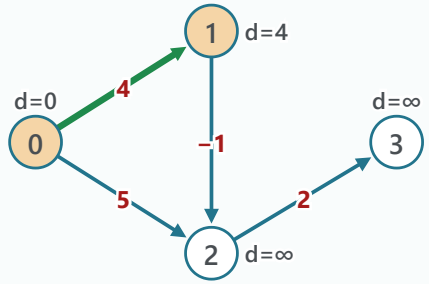
DEĞİŞKENLER

`i = –` `d = [0, ∞, ∞, ∞]` `degisti = –`

Soru 9: Bellman-Ford İzleme

Kaynak 0 için Bellman-Ford hangi turda hangi kenarları gevşetir? Son d dizisi nedir?

```
1  for (int i = 1; i < V; i++) {
2    boolean degisti = false;
▶    for (Kenar k : kenarlar)
▶      if (d[k.u] != SONSUZ && d[k.u] + k.w <
   d[k.v]) {
▶        d[k.v] = d[k.u] + k.w;
▶        degisti = true;
7      }
8    if (!degisti) break;              // erken
   durma
9  }
10 for (Kenar k : kenarlar)             // denetim
11   if (d[k.u] != SONSUZ && d[k.u] + k.w <
   d[k.v])
12     throw new IllegalStateException("Negatif
   döngü");
```



Tur 1, kenar (0,1): 0 + (4) = 4 < ∞ → d[1] = **4**.

Adım 2 / 10

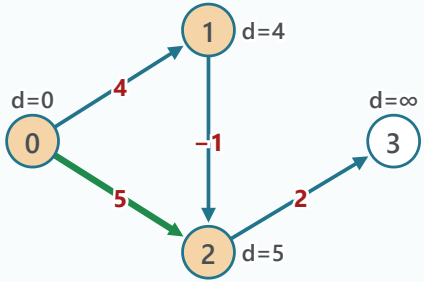
DEĞİŞKENLER

`i = 1` `d = [0, 4, ∞, ∞]` `degisti = true`

Soru 9: Bellman-Ford İzleme

Kaynak 0 için Bellman-Ford hangi turda hangi kenarları gevşetir? Son d dizisi nedir?

```
1  for (int i = 1; i < V; i++) {
2    boolean degisti = false;
▶    for (Kenar k : kenarlar)
▶      if (d[k.u] != SONSUZ && d[k.u] + k.w <
   d[k.v]) {
▶        d[k.v] = d[k.u] + k.w;
▶        degisti = true;
7      }
8    if (!degisti) break;              // erken
   durma
9  }
10 for (Kenar k : kenarlar)             // denetim
11   if (d[k.u] != SONSUZ && d[k.u] + k.w <
   d[k.v])
12     throw new IllegalStateException("Negatif
   döngü");
```



Tur 1, kenar (0,2): 0 + (5) = 5 < ∞ → d[2] = **5**.

Adım 3 / 10

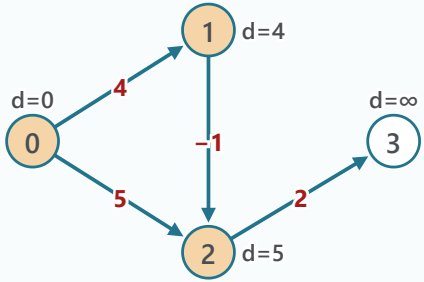
DEĞİŞKENLER

`i = 1` `d = [0, 4, 5, ∞]` `degisti = true`

Soru 9: Bellman-Ford İzleme

Kaynak 0 için Bellman-Ford hangi turda hangi kenarları gevşetir? Son d dizisi nedir?

```
1  for (int i = 1; i < V; i++) {
2    boolean degisti = false;
3    for (Kenar k : kenarlar)
4      if (d[k.u] != SONSUZ && d[k.u] + k.w <
   d[k.v]) {
5        d[k.v] = d[k.u] + k.w;
6        degisti = true;
7      }
▶    if (!degisti) break;              // erken
   durma
9  }
10 for (Kenar k : kenarlar)              // denetim
11   if (d[k.u] != SONSUZ && d[k.u] + k.w <
   d[k.v])
12     throw new IllegalStateException("Negatif
   döngü");
```



Tur 1 bitti; değişmeyen kenarlar: (2,3), (1,2). Bir şey değişti, sonraki tura geçilir.

Adım 4 / 10

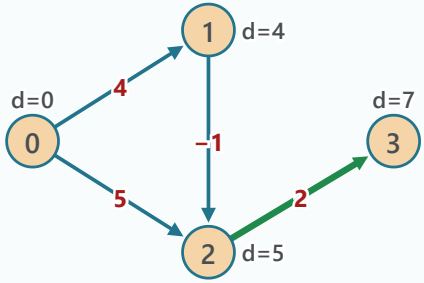
DEĞİŞKENLER

`i = 1` `d = [0, 4, 5, ∞]` `degisti = true`

Soru 9: Bellman-Ford İzleme

Kaynak 0 için Bellman-Ford hangi turda hangi kenarları gevşetir? Son d dizisi nedir?

```
1  for (int i = 1; i < V; i++) {
2    boolean degisti = false;
▶    for (Kenar k : kenarlar)
▶      if (d[k.u] != SONSUZ && d[k.u] + k.w <
   d[k.v]) {
▶        d[k.v] = d[k.u] + k.w;
▶        degisti = true;
7      }
8    if (!degisti) break;              // erken
   durma
9  }
10 for (Kenar k : kenarlar)             // denetim
11   if (d[k.u] != SONSUZ && d[k.u] + k.w <
   d[k.v])
12     throw new IllegalStateException("Negatif
   döngü");
```



Tur 2, kenar (2,3): 5 + (2) = 7 < ∞ → d[3] = **7**.

Adım 5 / 10

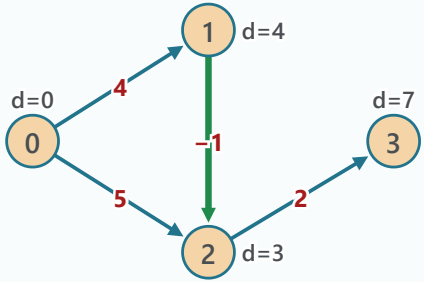
DEĞİŞKENLER

`i = 2` `d = [0, 4, 5, 7]` `degisti = true`

Soru 9: Bellman-Ford İzleme

Kaynak 0 için Bellman-Ford hangi turda hangi kenarları gevşetir? Son d dizisi nedir?

```
1  for (int i = 1; i < V; i++) {
2    boolean degisti = false;
▶    for (Kenar k : kenarlar)
▶      if (d[k.u] != SONSUZ && d[k.u] + k.w <
   d[k.v]) {
▶        d[k.v] = d[k.u] + k.w;
▶        degisti = true;
7      }
8    if (!degisti) break;              // erken
   durma
9  }
10 for (Kenar k : kenarlar)             // denetim
11   if (d[k.u] != SONSUZ && d[k.u] + k.w <
   d[k.v])
12     throw new IllegalStateException("Negatif
   döngü");
```



Tur 2, kenar (1,2): 4 + (−1) = 3 < 5 → d[2] = **3**.

Adım 6 / 10

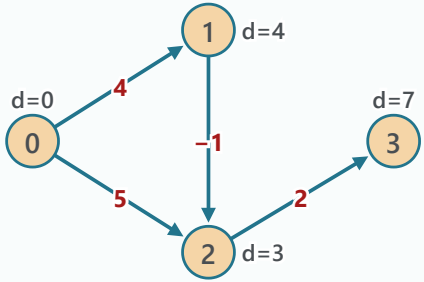
DEĞİŞKENLER

`i = 2` `d = [0, 4, 3, 7]` `degisti = true`

Soru 9: Bellman-Ford İzleme

Kaynak 0 için Bellman-Ford hangi turda hangi kenarları gevşetir? Son d dizisi nedir?

```
1  for (int i = 1; i < V; i++) {
2    boolean degisti = false;
3    for (Kenar k : kenarlar)
4      if (d[k.u] != SONSUZ && d[k.u] + k.w <
   d[k.v]) {
5        d[k.v] = d[k.u] + k.w;
6        degisti = true;
7      }
▶    if (!degisti) break;              // erken
   durma
9  }
10 for (Kenar k : kenarlar)              // denetim
11   if (d[k.u] != SONSUZ && d[k.u] + k.w <
   d[k.v])
12     throw new IllegalStateException("Negatif
   döngü");
```



Tur 2 bitti; değişmeyen kenarlar: (0,1), (0,2). Bir şey değişti, sonraki tura geçilir.

Adım 7 / 10

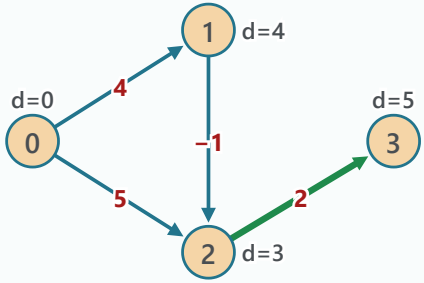
DEĞİŞKENLER

`i = 2` `d = [0, 4, 3, 7]` `degisti = true`

Soru 9: Bellman-Ford İzleme

Kaynak 0 için Bellman-Ford hangi turda hangi kenarları gevşetir? Son d dizisi nedir?

```
1  for (int i = 1; i < V; i++) {
2    boolean degisti = false;
▶    for (Kenar k : kenarlar)
▶      if (d[k.u] != SONSUZ && d[k.u] + k.w <
   d[k.v]) {
▶        d[k.v] = d[k.u] + k.w;
▶        degisti = true;
7      }
8    if (!degisti) break;              // erken
   durma
9  }
10 for (Kenar k : kenarlar)             // denetim
11   if (d[k.u] != SONSUZ && d[k.u] + k.w <
   d[k.v])
12     throw new IllegalStateException("Negatif
   döngü");
```



Tur 3, kenar (2,3): 3 + (2) = 5 < 7 → d[3] = **5**.

Adım 8 / 10

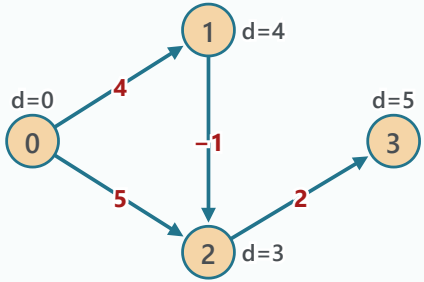
DEĞİŞKENLER

`i = 3` `d = [0, 4, 3, 5]` `degisti = true`

Soru 9: Bellman-Ford İzleme

Kaynak 0 için Bellman-Ford hangi turda hangi kenarları gevşetir? Son d dizisi nedir?

```
1  for (int i = 1; i < V; i++) {
2    boolean degisti = false;
3    for (Kenar k : kenarlar)
4      if (d[k.u] != SONSUZ && d[k.u] + k.w <
   d[k.v]) {
5        d[k.v] = d[k.u] + k.w;
6        degisti = true;
7      }
▶    if (!degisti) break;              // erken
   durma
9  }
10 for (Kenar k : kenarlar)              // denetim
11   if (d[k.u] != SONSUZ && d[k.u] + k.w <
   d[k.v])
12     throw new IllegalStateException("Negatif
   döngü");
```



Tur 3 bitti; değişmeyen kenarlar: (1,2), (0,1), (0,2). i = V − 1 = 3 son turdu.

Adım 9 / 10

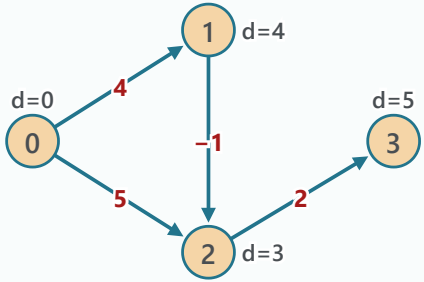
DEĞİŞKENLER

`i = 3` `d = [0, 4, 3, 5]` `degisti = true`

Soru 9: Bellman-Ford İzleme

Kaynak 0 için Bellman-Ford hangi turda hangi kenarları gevşetir? Son d dizisi nedir?

```
1  for (int i = 1; i < V; i++) {
2    boolean degisti = false;
3    for (Kenar k : kenarlar)
4      if (d[k.u] != SONSUZ && d[k.u] + k.w <
   d[k.v]) {
5        d[k.v] = d[k.u] + k.w;
6        degisti = true;
7      }
8    if (!degisti) break;              // erken
   durma
9  }
▶  for (Kenar k : kenarlar)             // denetim
▶    if (d[k.u] != SONSUZ && d[k.u] + k.w <
   d[k.v])
12     throw new IllegalStateException("Negatif
   döngü");
```



Denetim turu: hiçbir kenar gevşemez, istisna yok. Sonuç d = [0, 4, 3, 5]. (2,3) kenarı listenin başında olduğu için her turda bir öncekinin değerini kullandı: V − 1 turun hepsi gerekti.

Adım 10 / 10

DEĞİŞKENLER

`i = –` `d = [0, 4, 3, 5]` `degisti = –`

ÇOKTAN SEÇMELİ · BELLMAN-FORD

Soru 10: Döngü Ağırlığı

Bir çizgede A → B (3), B → C (−5) ve C → A (w) kenarlarından oluşan bir döngü var ve A kaynaktan ulaşılabilir. w tam sayısının hangi değerleri için Bellman-Ford'un denetim turu negatif döngü bildirir?

A) $w \le 1$

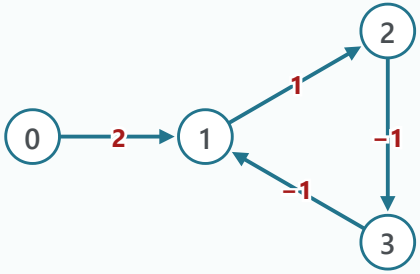
B) $w \ge 2$

C) $w \le 2$

D) $w < 5$

Soru 11: Negatif Döngü Denetimi

Kaynak 0, kenar sırası (0,1), (1,2), (2,3), (3,1). Bellman-Ford V − 1 = 3 turdan sonra hangi d dizisini bulur? Denetim turunda ilk hangi kenar gevşer?



▶ **Cevabı göster**

Soru 12: Floyd-Warshall İzleme

Her k turunda hangi hücreler değişir? Son D matrisi nedir? (Vurgulu hücre: bu turda kısalan.)

```
1 for (int k = 0; k < V; k++)          // ara düğüm
2   for (int i = 0; i < V; i++)
3     for (int j = 0; j < V; j++)
4       if (d[i][k] != SONSUZ && d[k][j] !=
  SONSUZ
5           && d[i][k] + d[k][j] < d[i][j])
6         d[i][j] = d[i][k] + d[k][j];
7 for (int i = 0; i < V; i++)
8   if (d[i][i] < 0)
9     throw new IllegalStateException("Negatif
  döngü");
```

	0	1	2	3
0	0	3	∞	7
1	8	0	2	∞
2	5	∞	0	1
3	2	∞	∞	0

Kenarlar: 0→1 (3), 0→3 (7), 1→0 (8), 1→2 (2), 2→0 (5), 2→3 (1), 3→0 (2). D başlangıçta kenar ağırlıkları, köşegen 0, kenar yoksa ∞.

Adım 1 / 6

DEĞİŞKENLER

`k = –` `değişen = –`

Soru 12: Floyd-Warshall İzleme

Her k turunda hangi hücreler değişir? Son D matrisi nedir? (Vurgulu hücre: bu turda kısalan.)

```
for (int k = 0; k < V; k++)        // ara düğüm
  for (int i = 0; i < V; i++)
    for (int j = 0; j < V; j++)
      if (d[i][k] != SONSUZ && d[k][j] !=
SONSUZ
          && d[i][k] + d[k][j] < d[i][j])
        d[i][j] = d[i][k] + d[k][j];
for (int i = 0; i < V; i++)
  if (d[i][i] < 0)
    throw new IllegalStateException("Negatif
döngü");
```

	0	1	2	3
0	0	3	∞	7
1	8	0	2	15
2	5	8	0	1
3	2	5	∞	0

k = 0 (ara düğüm 0; mavi çerçeveli satır ve sütun değişmez): D[1][3]: ∞ → 15 (8 + 7), D[2][1]: ∞ → 8 (5 + 3), D[3][1]: ∞ → 5 (2 + 3).

Adım 2 / 6

DEĞİŞKENLER

`k = 0` `değişen = 3 hücre`

Soru 12: Floyd-Warshall İzleme

Her k turunda hangi hücreler değişir? Son D matrisi nedir? (Vurgulu hücre: bu turda kısalan.)

```
for (int k = 0; k < V; k++)        // ara düğüm
  for (int i = 0; i < V; i++)
    for (int j = 0; j < V; j++)
      if (d[i][k] != SONSUZ && d[k][j] !=
SONSUZ
          && d[i][k] + d[k][j] < d[i][j])
        d[i][j] = d[i][k] + d[k][j];
for (int i = 0; i < V; i++)
  if (d[i][i] < 0)
    throw new IllegalStateException("Negatif
döngü");
```

	0	1	2	3
0	0	3	5	7
1	8	0	2	15
2	5	8	0	1
3	2	5	7	0

k = 1 (ara düğüm 1; mavi çerçeveli satır ve sütun değişmez): D[0][2]: ∞ → 5 (3 + 2), D[3][2]: ∞ → 7 (5 + 2).

Adım 3 / 6

DEĞİŞKENLER

`k = 1` `değişen = 2 hücre`

Soru 12: Floyd-Warshall İzleme

Her k turunda hangi hücreler değişir? Son D matrisi nedir? (Vurgulu hücre: bu turda kısalan.)

```
for (int k = 0; k < V; k++)        // ara düğüm
  for (int i = 0; i < V; i++)
    for (int j = 0; j < V; j++)
      if (d[i][k] != SONSUZ && d[k][j] !=
SONSUZ
          && d[i][k] + d[k][j] < d[i][j])
        d[i][j] = d[i][k] + d[k][j];
for (int i = 0; i < V; i++)
  if (d[i][i] < 0)
    throw new IllegalStateException("Negatif
döngü");
```

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

k = 2 (ara düğüm 2; mavi çerçeveli satır ve sütun değişmez): D[0][3]: 7 → 6 (5 + 1), D[1][0]: 8 → 7 (2 + 5), D[1][3]: 15 → 3 (2 + 1).

Adım 4 / 6

DEĞİŞKENLER

`k = 2` `değişen = 3 hücre`

Soru 12: Floyd-Warshall İzleme

Her k turunda hangi hücreler değişir? Son D matrisi nedir? (Vurgulu hücre: bu turda kısalan.)

```
for (int k = 0; k < V; k++)        // ara düğüm
  for (int i = 0; i < V; i++)
    for (int j = 0; j < V; j++)
      if (d[i][k] != SONSUZ && d[k][j] !=
SONSUZ
          && d[i][k] + d[k][j] < d[i][j])
        d[i][j] = d[i][k] + d[k][j];
for (int i = 0; i < V; i++)
  if (d[i][i] < 0)
    throw new IllegalStateException("Negatif
döngü");
```

	0	1	2	3
0	0	3	5	6
1	5	0	2	3
2	3	6	0	1
3	2	5	7	0

k = 3 (ara düğüm 3; mavi çerçeveli satır ve sütun değişmez): D[1][0]: 7 → 5 (3 + 2), D[2][0]: 5 → 3 (1 + 2), D[2][1]: 8 → 6 (1 + 5).

Adım 5 / 6

DEĞİŞKENLER

k = 3 | değişen = 3 hücre

Soru 12: Floyd-Warshall İzleme

Her k turunda hangi hücreler değişir? Son D matrisi nedir? (Vurgulu hücre: bu turda kısalan.)

```
1 for (int k = 0; k < V; k++)        // ara düğüm
2   for (int i = 0; i < V; i++)
3     for (int j = 0; j < V; j++)
4       if (d[i][k] != SONSUZ && d[k][j] !=
  SONSUZ
5           && d[i][k] + d[k][j] < d[i][j])
6         d[i][j] = d[i][k] + d[k][j];
▶ for (int i = 0; i < V; i++)
▶   if (d[i][i] < 0)
9     throw new IllegalStateException("Negatif
  döngü");
```

	0	1	2	3
0	0	3	5	6
1	5	0	2	3
2	3	6	0	1
3	2	5	7	0

Köşegende negatif değer yok → negatif döngü yok. Son D tüm çiftler arası en kısa uzaklıklardır; örneğin D[1][0] = 5 (1 → 2 → 3 → 0).

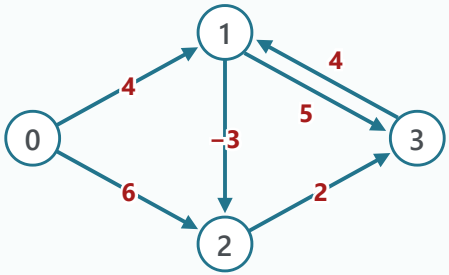
Adım 6 / 6

DEĞİŞKENLER

`k = –` `köşegen = 0 0 0 0`

Soru 13: Negatif Kenarla Floyd-Warshall

Floyd-Warshall bu çizgede çalıştırılıyor. Son D matrisi nedir? Negatif döngü var mı? 0'dan 3'e en kısa yol hangisidir?



▶ **Cevabı göster**

ÇOKTAN SEÇMELİ · FLOYD-WARSHALL

Soru 14: Döngü Sırası

Floyd-Warshall kodunda döngüler `for i … for j … for k …` sırasına (k en içte) getirilirse ne olur?

A) Sonuç yine doğrudur; yalnızca bellek erişim düzeni ve önbellek davranışı değişir.

B) Bazı D[i][j] değerleri gereğinden büyük (yanlış) kalabilir.

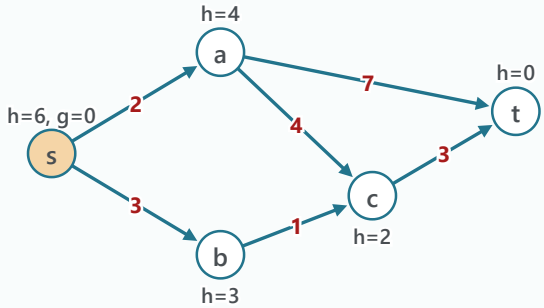
C) Zaman karmaşıklığı $\Theta(V^3)$'ten $\Theta(V^4)$'e çıkar, sonuç doğru kalır.

D) Negatif döngü varken sonsuz döngüye girer ve hiç bitmez.

Soru 15: A* İzleme

A* hangi düğümleri hangi sırayla genişletir? Bulunan yol ve maliyeti nedir?

```
▶ g[s] = 0; acik.add(s);
2 while (!acik.isEmpty()) {
3   int u = enKucukF(acik);            // f = g + h
4   acik.remove(u); kapali.add(u);
5   if (u == hedef) break;
6   for (Kenar k : komsu[u]) {
7     int yeni = g[u] + k.w;
8     if (!kapali.contains(k.hedef)
9         && yeni < g[k.hedef]) {
10      g[k.hedef] = yeni; onceki[k.hedef] = u;
11      acik.add(k.hedef);
12    }
13  }
14 }
```



s'den t'ye; düğüm yanında h (hedefe tahmin) ve g (bulunan maliyet). h kabul edilebilir ve tutarlıdır; f eşitse alfabetik ilk seçilir.

Adım 1 / 6

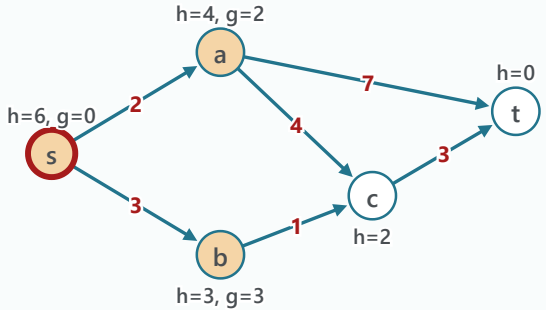
DEĞİŞKENLER

`u = –` `açık = {s:6}` `kapalı = {}`

Soru 15: A* İzleme

A* hangi düğümleri hangi sırayla genişletir? Bulunan yol ve maliyeti nedir?

```
1  g[s] = 0; acik.add(s);
2  while (!acik.isEmpty()) {
▶    int u = enKucukF(acik);          // f = g + h
▶    acik.remove(u); kapali.add(u);
▶    if (u == hedef) break;
▶    for (Kenar k : komsu[u]) {
▶      int yeni = g[u] + k.w;
▶      if (!kapali.contains(k.hedef)
▶          && yeni < g[k.hedef]) {
▶        g[k.hedef] = yeni; onceki[k.hedef] = u;
▶        acik.add(k.hedef);
12     }
13   }
14 }
```



En küçük f = 6 olan **s** genişletilir. Açılan/güncellenen: a: g = 2, f = 6; b: g = 3, f = 6.

Adım 2 / 6

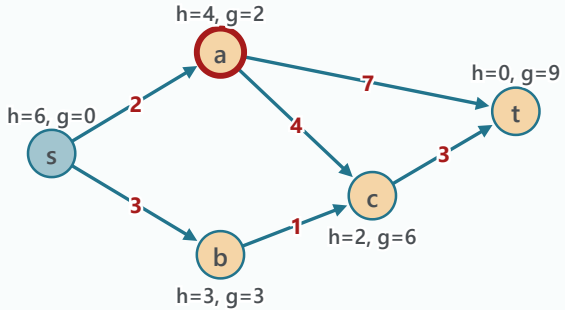
DEĞİŞKENLER

`u = s` `açık = {a:6, b:6}` `kapalı = {s}`

Soru 15: A* İzleme

A* hangi düğümleri hangi sırayla genişletir? Bulunan yol ve maliyeti nedir?

```
1  g[s] = 0; acik.add(s);
2  while (!acik.isEmpty()) {
▶    int u = enKucukF(acik);          // f = g + h
▶    acik.remove(u); kapali.add(u);
▶    if (u == hedef) break;
▶    for (Kenar k : komsu[u]) {
▶      int yeni = g[u] + k.w;
▶      if (!kapali.contains(k.hedef)
▶          && yeni < g[k.hedef]) {
▶        g[k.hedef] = yeni; onceki[k.hedef] = u;
▶        acik.add(k.hedef);
12     }
13   }
14 }
```



En küçük f = 6 olan **a** genişletilir. Açılan/güncellenen: c: g = 6, f = 8; t: g = 9, f = 9.

Adım 3 / 6

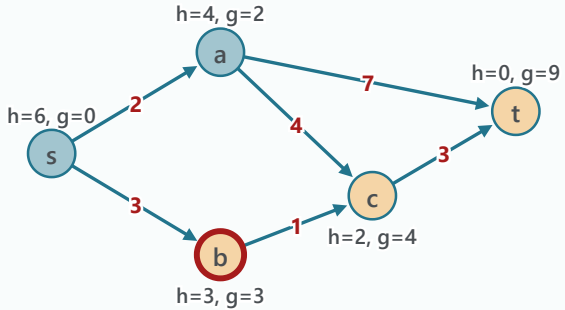
DEĞİŞKENLER

`u = a` `açık = {b:6, c:8, t:9}` `kapalı = {s, a}`

Soru 15: A* İzleme

A* hangi düğümleri hangi sırayla genişletir? Bulunan yol ve maliyeti nedir?

```
1  g[s] = 0; acik.add(s);
2  while (!acik.isEmpty()) {
▶    int u = enKucukF(acik);          // f = g + h
▶    acik.remove(u); kapali.add(u);
▶    if (u == hedef) break;
▶    for (Kenar k : komsu[u]) {
▶      int yeni = g[u] + k.w;
▶      if (!kapali.contains(k.hedef)
▶          && yeni < g[k.hedef]) {
▶        g[k.hedef] = yeni; onceki[k.hedef] = u;
▶        acik.add(k.hedef);
12     }
13   }
14 }
```



En küçük f = 6 olan **b** genişletilir. Açılan/güncellenen: c: g = 4, f = 6.

Adım 4 / 6

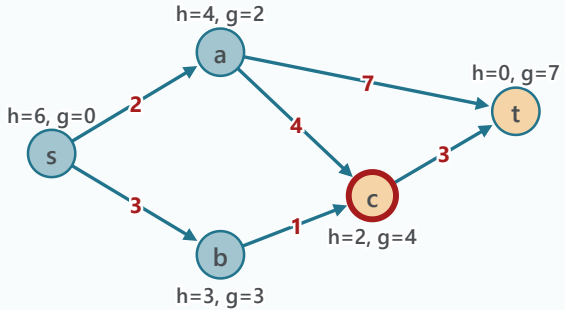
DEĞİŞKENLER

`u = b` `açık = {c:6, t:9}` `kapalı = {s, a, b}`

Soru 15: A* İzleme

A* hangi düğümleri hangi sırayla genişletir? Bulunan yol ve maliyeti nedir?

```
1  g[s] = 0; acik.add(s);
2  while (!acik.isEmpty()) {
▶    int u = enKucukF(acik);          // f = g + h
▶    acik.remove(u); kapali.add(u);
▶    if (u == hedef) break;
▶    for (Kenar k : komsu[u]) {
▶      int yeni = g[u] + k.w;
▶      if (!kapali.contains(k.hedef)
▶          && yeni < g[k.hedef]) {
▶        g[k.hedef] = yeni; onceki[k.hedef] = u;
▶        acik.add(k.hedef);
12     }
13   }
14 }
```



En küçük f = 6 olan **c** genişletilir. Açılan/güncellenen: t: g = 7, f = 7.

Adım 5 / 6

DEĞİŞKENLER

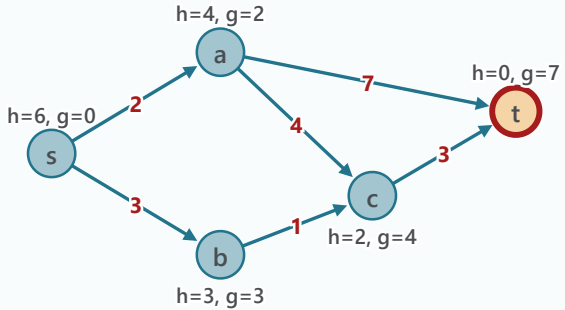
`u = c` `açık = {t:7}`

`kapalı = {s, a, b, c}`

Soru 15: A* İzleme

A* hangi düğümleri hangi sırayla genişletir? Bulunan yol ve maliyeti nedir?

```
1  g[s] = 0; acik.add(s);
2  while (!acik.isEmpty()) {
▶    int u = enKucukF(acik);          // f = g + h
▶    acik.remove(u); kapali.add(u);
▶    if (u == hedef) break;
6    for (Kenar k : komsu[u]) {
7      int yeni = g[u] + k.w;
8      if (!kapali.contains(k.hedef)
9          && yeni < g[k.hedef]) {
10       g[k.hedef] = yeni; onceki[k.hedef] = u;
11       acik.add(k.hedef);
12     }
13   }
14 }
```



En küçük f: t (7). Hedef açık listeden **çıkarıldı**: dur. Yol (onceki ile) s → b → c → t, maliyet 7. a üzerinden açılan pahalı yollar (c: 6, t: 9) b ve c genişletilince düzeltildi.

Adım 6 / 6

DEĞİŞKENLER

`u = t` `açık = {}`

`kapalı = {s, a, b, c, t}`

ÇOKTAN SEÇMELİ · A* ALGORİTMASI

Soru 16: Sıradaki Genişletme

A*'ın açık listesinde şu düğümler var (g, h): p (4, 7), q (6, 3), r (2, 10), u (8, 2). Hangisi genişletilir?

A) r, çünkü g değeri en küçüktür

B) u, çünkü h değeri en küçüktür

C) q, çünkü f = g + h en küçüktür

D) p, çünkü g ile h dengelidir

ÇOKTAN SEÇMELİ · A* ALGORİTMASI

Soru 17: Kabul Edilebilir Sezgisel

Izgarada yol aranıyor; her adımın maliyeti 1'dir. Hangisi doğrudur?

A) Manhattan uzaklığı 8 yönlü harekette de kabul edilebilir, çünkü Öklid uzaklığından büyük değildir.

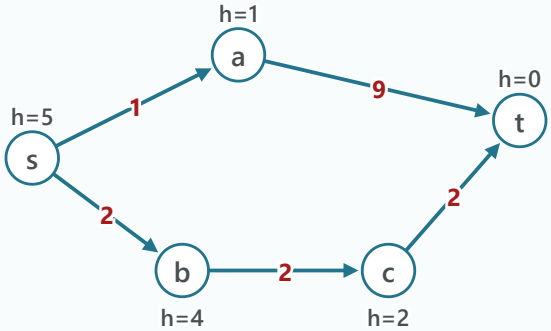
B) h = 0 kabul edilebilir değildir; bu yüzden A* en kısa yolu bulamaz.

C) Gerçek kalanı aşan h, daha az düğüm açtığı için en kısa yolu da garanti eder.

D) Manhattan uzaklığı 4 yönlü harekette kabul edilebilir, 8 yönlü harekette değildir.

Soru 18: Açgözlü Arama ve A*

s'den t'ye; düğüm yanında h değeri. (a) Açık listeden yalnızca h'si en küçük olanı seçen açgözlü en iyi ilk arama hangi yolu bulur, maliyeti nedir? (b) A* (f = g + h) hangi yolu bulur?



▶ **Cevabı göster**

ÇOKTAN SEÇMELİ · KARŞILAŞTIRMA

Soru 19: Algoritma Seçimi

Aşağıdaki eşleştirmelerden hangisi **yanlıştır**?

A) Ağırlıksız çizgede tek kaynaktan en az kenarlı yollar → BFS, $O(V + E)$ zamanda

B) Negatif kenarlı çizgede tek kaynaklı yollar → Dijkstra, $O((V + E) \log V)$

C) $V = 200$ düğümde tüm çiftler arası en kısa yollar → Floyd-Warshall, $\Theta(V^3)$

D) Haritada tek başlangıç–hedef çifti, iyi bir tahmin var → A*

Son

Bölüm 6: En Kısa Yol Algoritmaları: Çalışma Soruları

Konu anlatımı: En kısa yol sunumu · Terim sözlüğü