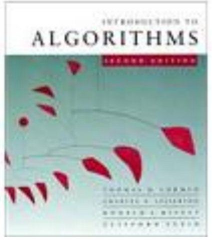
13.Hafta
En kısa yollar

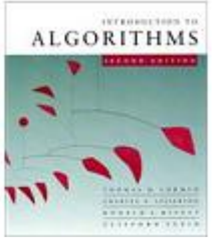
En kısa yolların özellikleri
- Dijkstra algoritması
- Doğruluk
- Çözümleme
- Enine arama



Konular

- Ağırlıklandırılmış graflarda (weighted graphs) tek kaynaklı (single-source) en kısa yollar(shortest paths)
  - En kısa yol problemleri
  - En Kısa yol özellikleri ve gevşeme(relexation)
  - Dijkstra algoritması
  - Bellman Ford algoritması

Yönlü Grafiklerde yollar-
En kısa yol

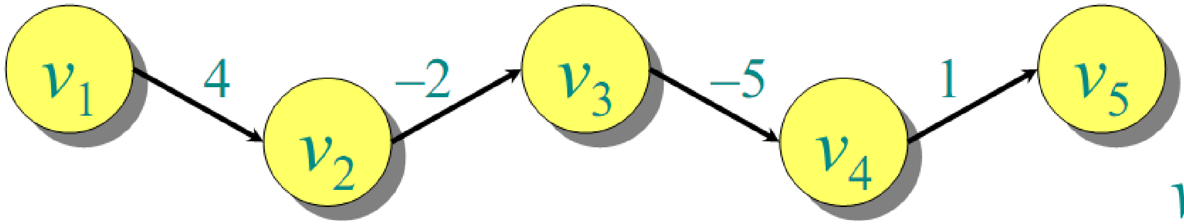


$w : E \to \mathbb{R}$ kenar-ağırlık fonksiyonu olan bir $G = (V, E)$ yönlendirilmiş grafiği olduğunu düşünün. Yolun ağırlığı olan

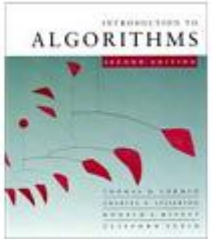
$$p = v_1 \to v_2 \to \cdots \to v_k$$

$$w(p) = \sum_{i=1}^{k-1} w(v_i, v_{i+1})$$ olarak tanımlanır.

Örnek:



$$w(p) = -2$$



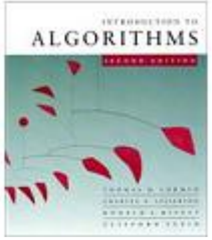
En kısa yollar

u' dan v ' ye en kısa yol, u ' dan v ' ye en az ağırlıklı yoldur.

u ' dan v ' ye en kısa yolun ağırlığı

$\delta(u, v) = \min\{w(p)$ olarak tanımlanır: p, u dan v ye bir yoldur$\}$.

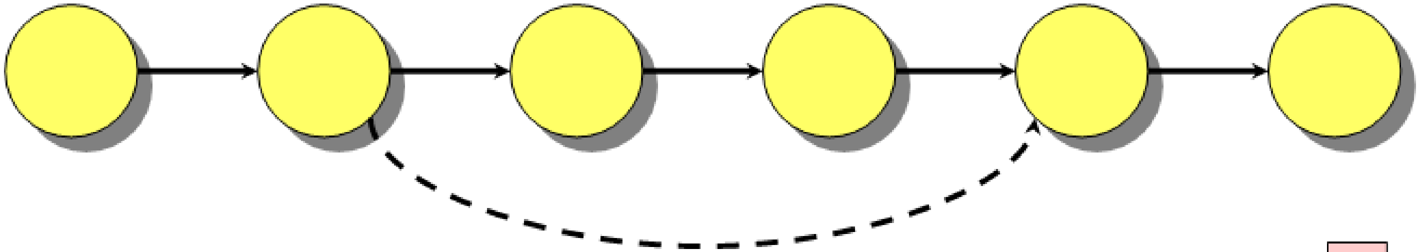
Not: u ' dan v ' bir yol yoksa $\delta(u, v) = \infty$



En uygun altyapı

Teorem. En kısa yolun alt yolu, bir en kısa yoldur.

Kanıt. Kes ve yapıştır:

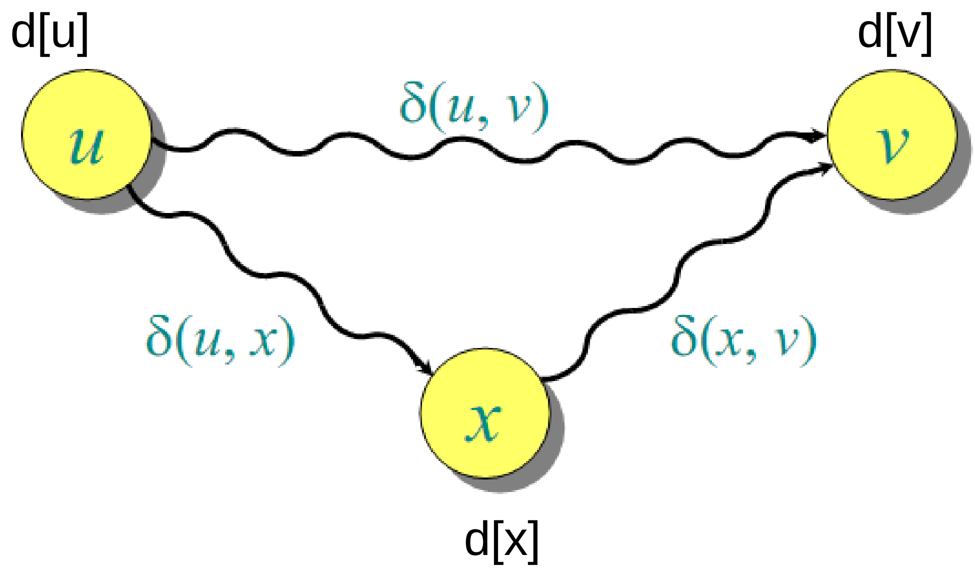


Üçgen eşitsizliği

Teorem. Tüm $u, v, x \in V$' ler için,

d[v]= $\delta(u, v) \leq \delta(u, x) + \delta(x, v)$.

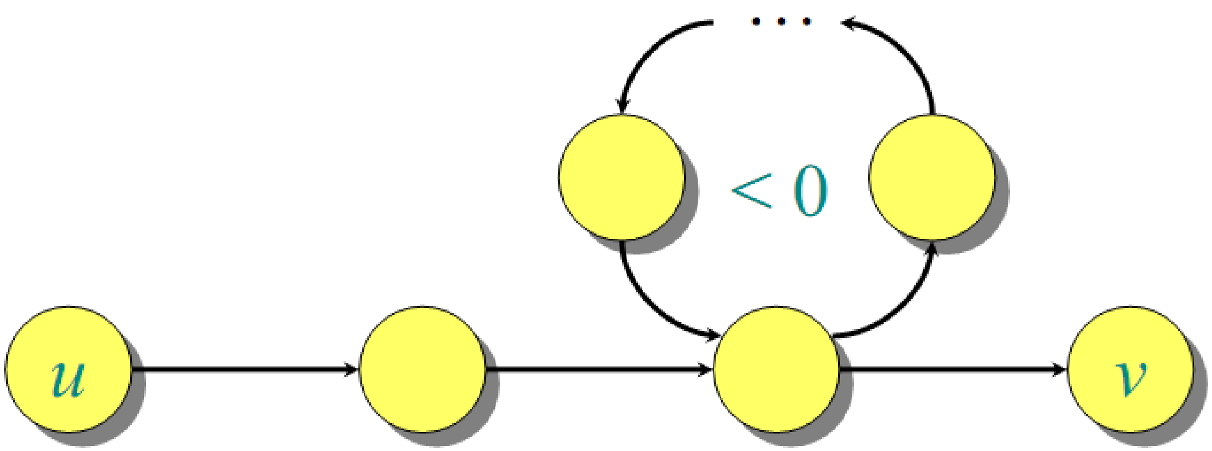
Kanıt.

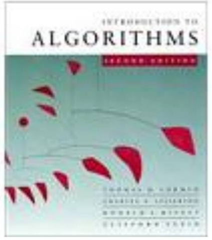


En kısa yolların iyi tanımlanırlığı

Bir G grafiği negatif ağırlık döngüsü içeriyorsa, bazı en kısa yollar var olmayabilir.

Örnek:

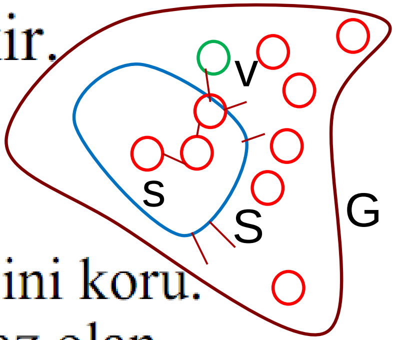




Tek-kaynaklı (single-source) En kısa yollar

Problem. $s \in V$ ' deki verilen bir kaynak köşeden, tüm $v \in V$' ler için, $\delta(s, v)$ en kısa yol ağırlıklarını bulun.

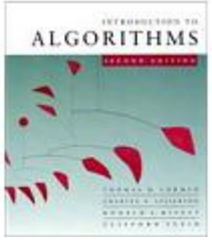
Tüm $w(u, v)$ kenar ağırlıkları *eksi* değilse bütün en kısa yol ağırlıklarının olması gerekir.



Fikir: Açgözlü.

1. s ' den başlayan ve S içindeki tüm köşelere olan en kısa yol uzunlukları bilinen köşelerin kümesini koru.
2. Her adımda S' ye, s' ye olan uzaklık tahmini en az olan $v \in V - S$ köşesine ekle.
3. v' ye bitişik köşelerin uzaklık tahminlerini güncelle.

Ağırlıklı en kısa yol algoritmaları



- **Dijkstra Algoritması:**
  - Ağırlıklı ve yönlü graflar için geliştirilmiştir.
  - Graf üzerindeki kenarların ağırlıkları 0 veya sıfırdan büyük sayılar olmalıdır.
  - Negatif ağırlıklar için çalışmaz.
- **Bellman ve Ford Algoritması:**
  - Negatif ağırlıklı graflar için geliştirilmiştir.
- **Floyd-Warshall Algoritması**
  - Negatif ağırlıklı graflar için geliştirilmiştir.
- **Johnson Algoritması**
  - Negatif ağırlıklı graflar için geliştirilmiştir.

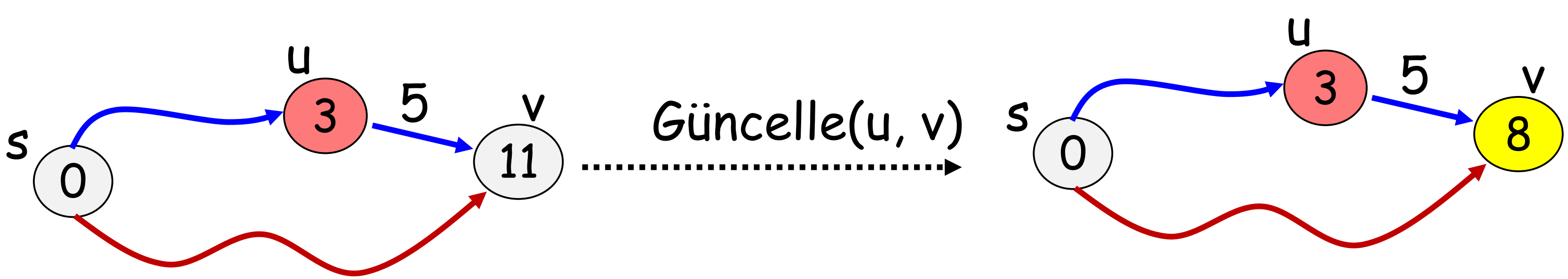
Dijkstra Algoritması

- Başlangıç olarak sadece başlangıç düğümünün en kısa yolu bilinir. (0 dır.)

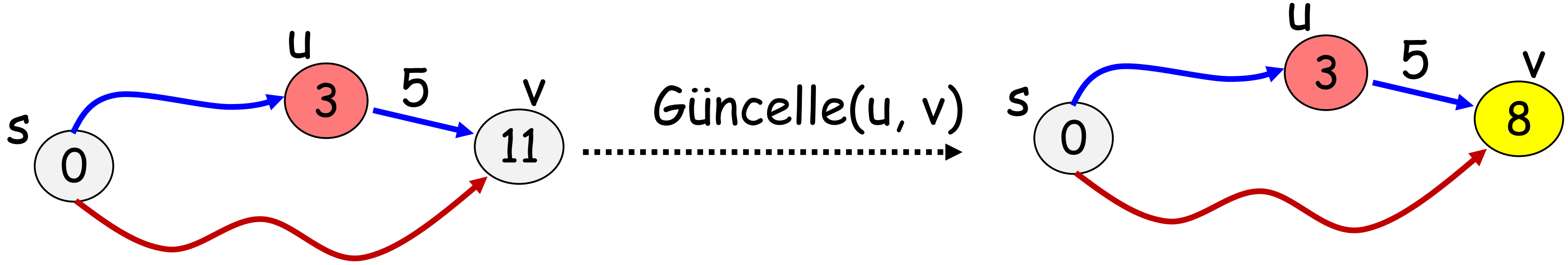
- Tüm düğümlerin maliyeti bilinene kadar devam et.
  1. O anki bilinen düğümler içerisinden en iyi düğümü şeç. (en az maliyetli düğümü seç, daha sonra bu düğümü bilinen düğümler kümesine ekle)
  2. Seçilen düğümün komşularının maliyetlerini güncelle.

Güncelleme

- Adım-1 de seçilen düğüm u olsun.
- u düğümünün komşularının maliyetini güncelleme işlemi aşağıdaki şekilde yapılır.
  - s’den v’ye gitmek için iki yol vardır.
  - Kırmızı yol izlenebilir. Maliyet 11.
  - veya mavi yol izlenebilir. Önce s’den u’ya 3 maliyeti ile gidilir. Daha sonra (u, v) kenarı üzerinden 8 maliyetle v’ye ulaşılır.

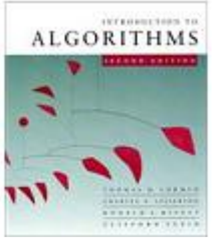


Güncelleme - Kaba Kod




```
Guncelle(u, v){
        if (maliyet[u] + w(u, v) < maliyet[v]){        // U üzerinden yol daha kısa ise
                maliyet[v] = maliyet[u] + w(u, v);        // Evet! Güncelle
                pred[v] = u;                              // u'dan geldiğimizi kaydet.
        }
}
```

Dijkstra algoritması



$d[s] \leftarrow 0$

(her) **for each** $v \in V - \{s\}$ (için)

(yap) **do** $d[v] \leftarrow \infty$

$S \leftarrow \varnothing$

$Q \leftarrow V$ ▷ Q, $V - S$'yi koruyan bir öncelikli sıradır.

while $Q \neq \varnothing$ (-iken)

(yap) **do** $u \leftarrow$ EXTRACT-MIN(Q) (en azı çıkar)

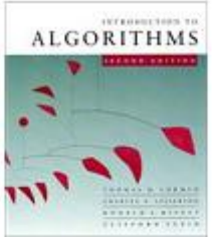
$S \leftarrow S \cup \{u\}$

(her) **for each** $v \in Adj[u]$ (için)

(yap eğer) **do if** $d[v] > d[u] + w(u, v)$

(sonra) **then** $d[v] \leftarrow d[u] + w(u, v)$

Dijkstra algoritması



$d[s] \leftarrow 0$

(her) **for each** $v \in V - \{s\}$ (için)

(yap) **do** $d[v] \leftarrow \infty$

$S \leftarrow \varnothing$

$Q \leftarrow V$ ⊳ Q, $V - S$'yi koruyan bir öncelikli sıradır.

while $Q \neq \varnothing$ (-iken)

(yap) **do** $u \leftarrow$ EXTRACT-MIN(Q) (en azı çıkar)

$S \leftarrow S \cup \{u\}$

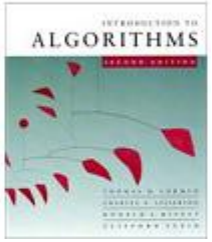
(her) **for each** $v \in Adj[u]$ (için)

(yap eğer) **do if** $d[v] > d[u] + w(u, v)$

(sonra) **then** $d[v] \leftarrow d[u] + w(u, v)$

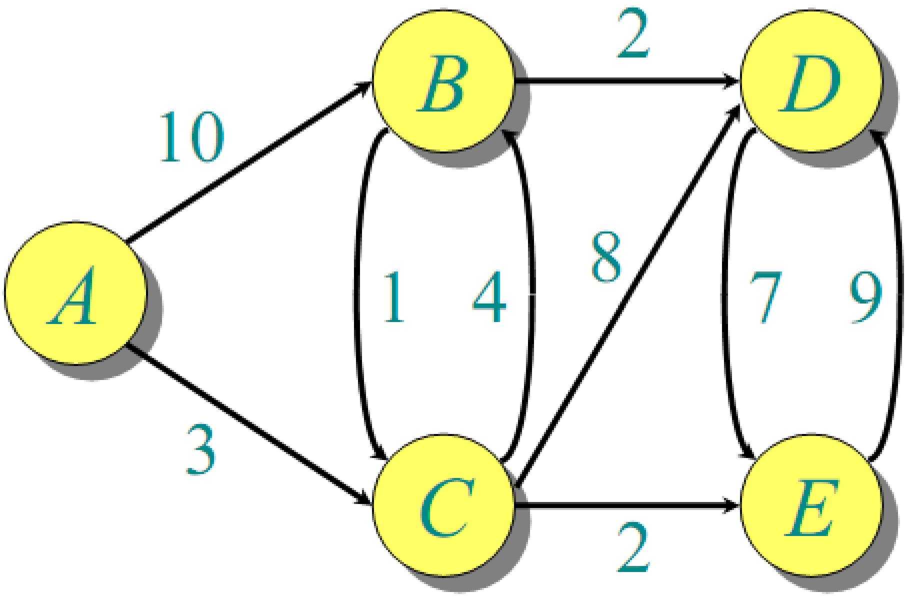
Gevşeme Adımı

Implicit DECREASE-KEY (azaltılmış anahtar)



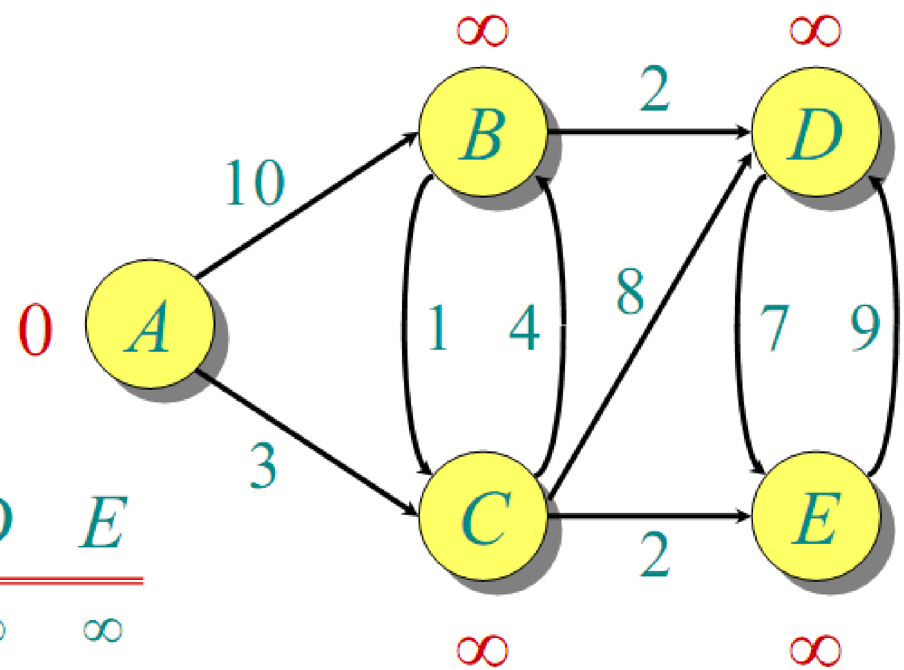
Dijkstra algoritmasına örnek

Eksi olmayan kenar ağırlıklarıyla grafik:



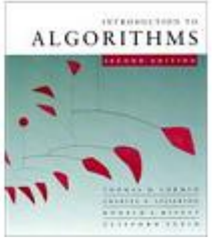
Dijkstra algoritmasına örnek

İlklendirme:



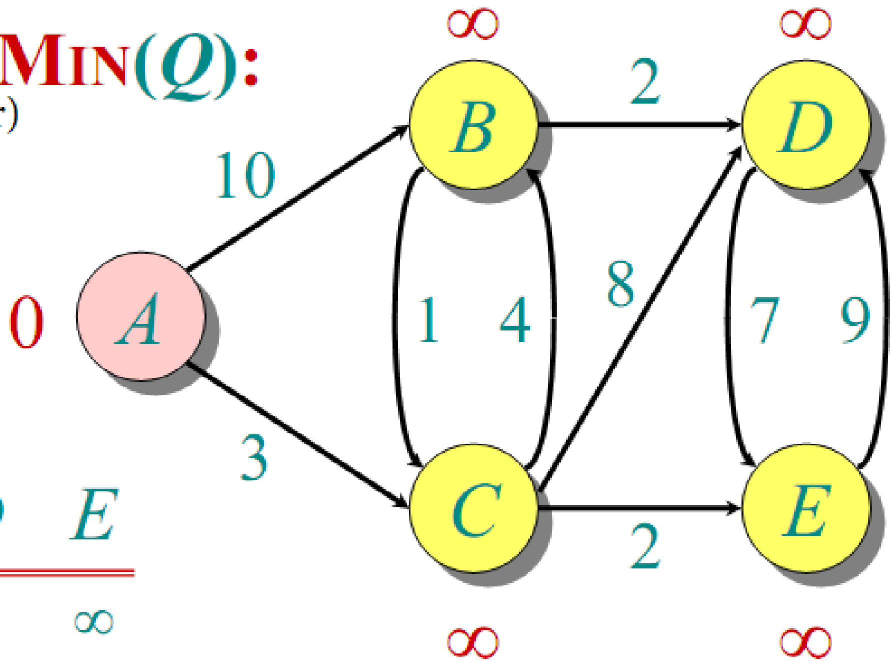
Q:	A	B	C	D	E
	0	∞	∞	∞	∞

S: {}



Dijkstra algoritmasına örnek

"A" $\leftarrow$ EXTRACT-MIN(Q):
(en azı çıkar)

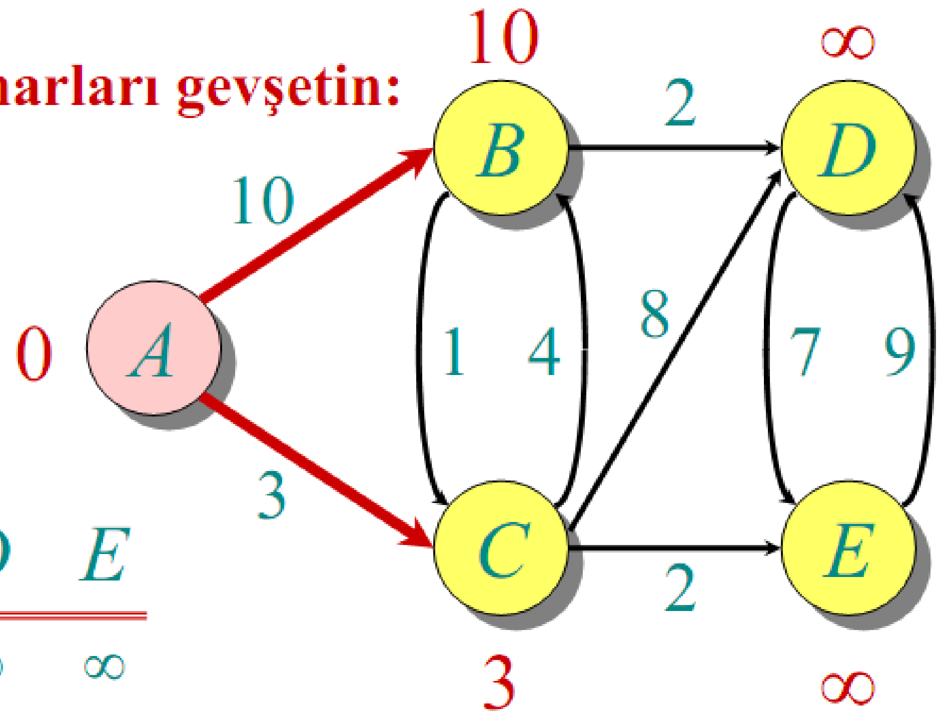


Q:	A	B	C	D	E
	0	∞	∞	∞	∞

S: { A }

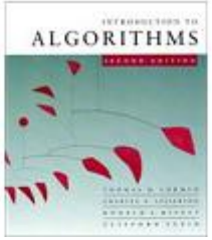
Dijkstra algoritmasına örnek

***A*'dan ayrılan tüm kenarları gevşetin:**



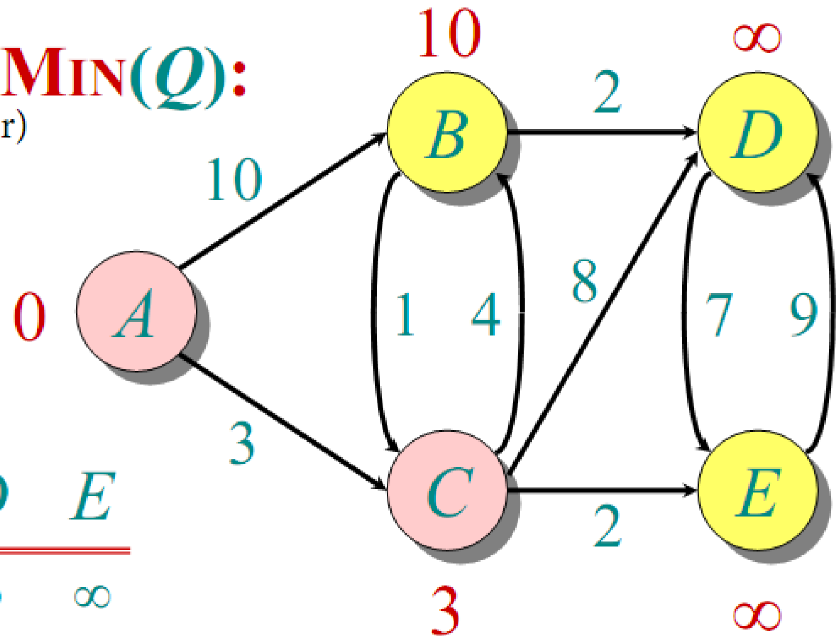
Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞

S: { A }



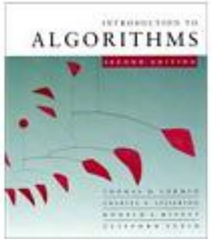
Dijkstra algoritmasına örnek

"C" ← EXTRACT-MIN(Q):
(en azı çıkar)



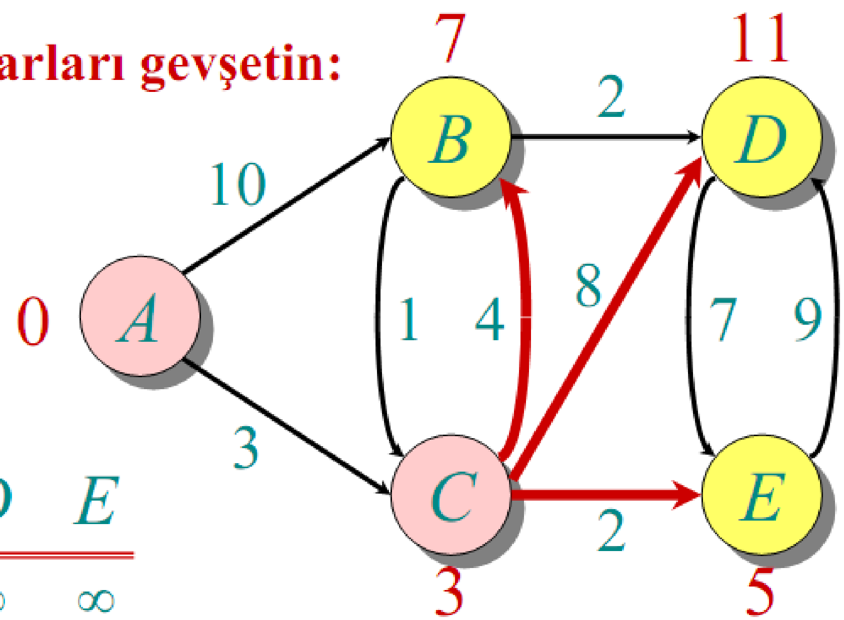
Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞

S: { A, C }



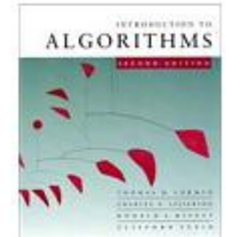
Dijkstra algoritmasına örnek

***C*'den ayrılan tüm kenarları gevşetin:**



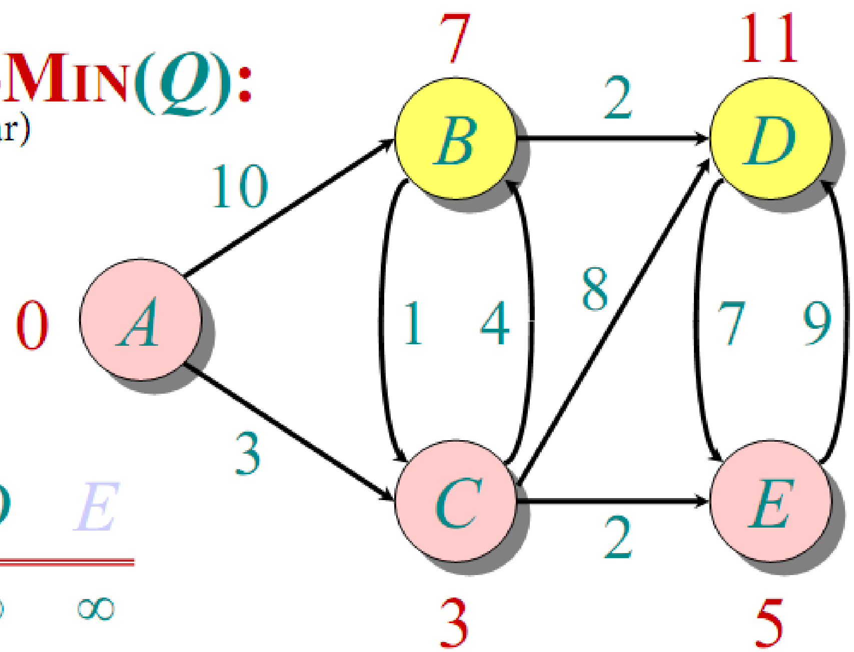
Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5

S: { A, C }



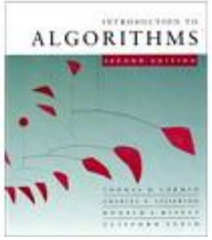
Dijkstra algoritmasına örnek

"E" $\leftarrow$ EXTRACT-MIN(Q):
(en azı çıkar)



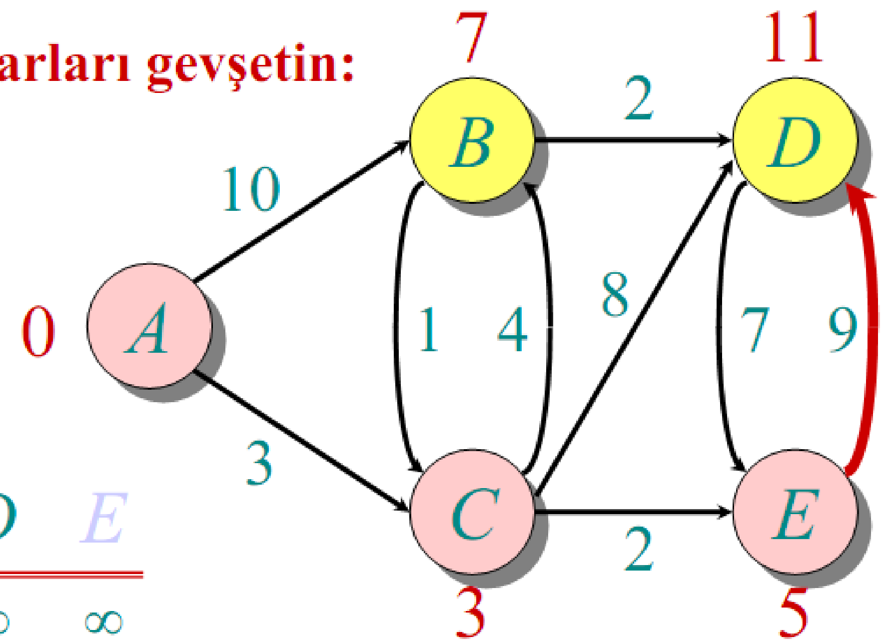
Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5

S: { A, C, E }



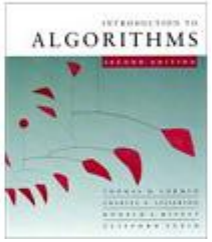
Dijkstra algoritmasına örnek

E'den ayrılan tüm kenarları gevşetin:



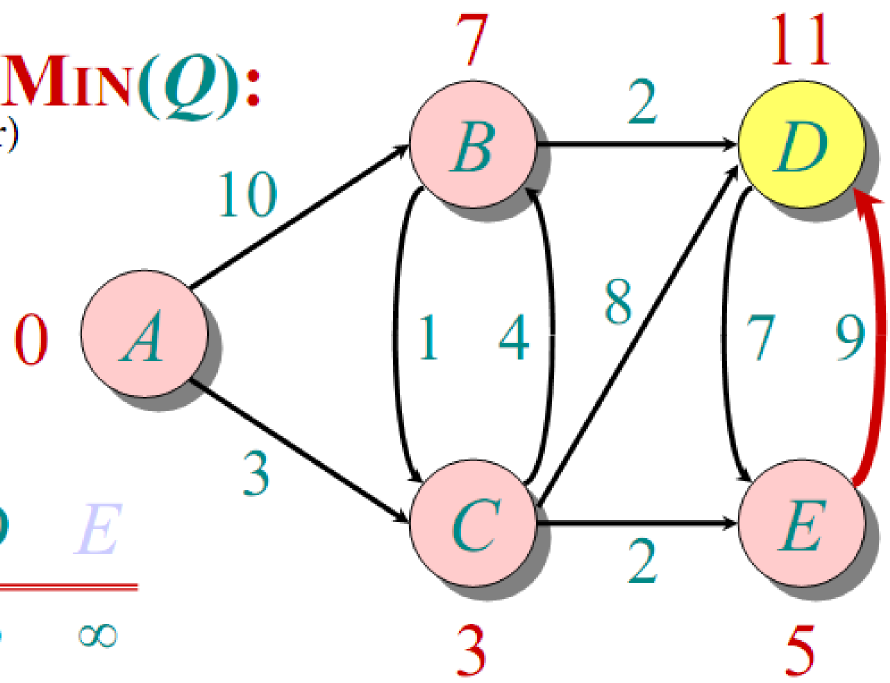
Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5
		7		11	

S: { *A*, *C*, *E* }



Dijkstra algoritmasına örnek

"B" ← EXTRACT-MIN(Q):
(en azı çıkar)

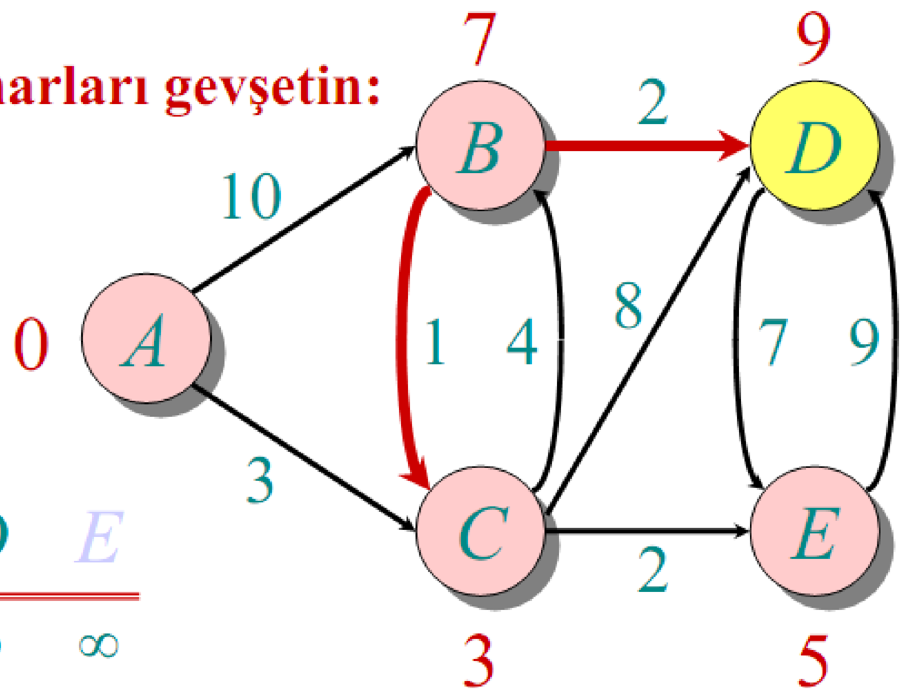


Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5
		7		11	

S: { A, C, E, B }

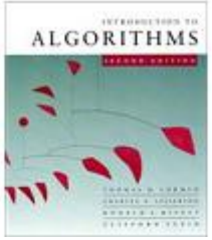
Dijkstra algoritmasına örnek

***B*'den ayrılan tüm kenarları gevşetin:**



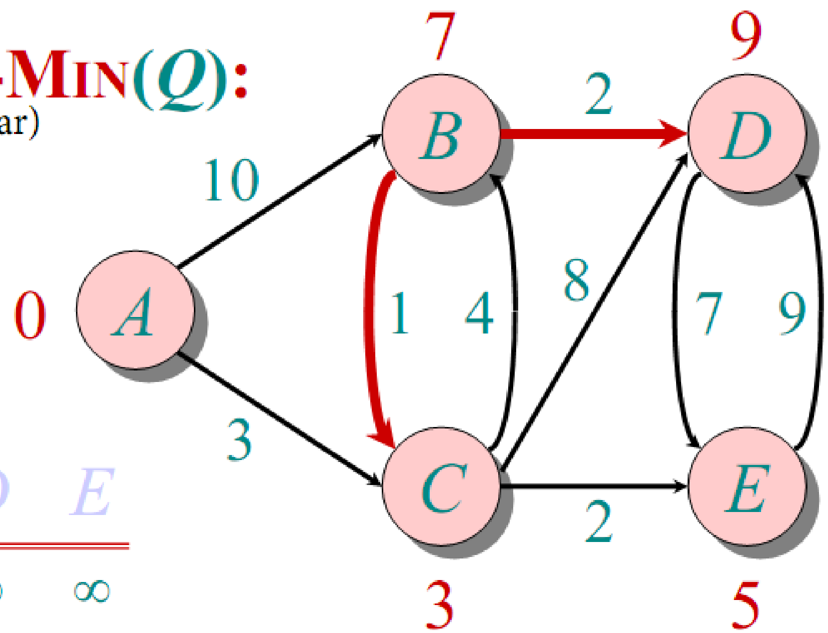
Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5
		7		11	
				9	

S: { A, C, E, B }



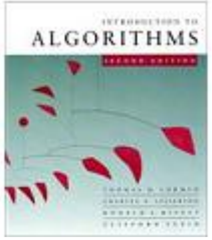
Dijkstra algoritmasına örnek

"*D*" ← EXTRACT-MIN(*Q*):
(en azı çıkar)



Q:	*A*	*B*	*C*	*D*	*E*
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5
		7		11	
				9	

S: { *A*, *C*, *E*, *B*, *D* }



Dijkstra' nın çözümlemesi (analizi)

$|V|$ kere

(-iken) **while** $Q \neq \emptyset$
  (yap) **do** $u \leftarrow$ EXTRACT-MIN(Q) (en azı çıkar)
    $S \leftarrow S \cup \{u\}$

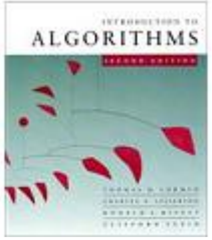
(u)derecesi kere

    for each $v \in Adj[u]$ (için)
    (her) **do if** $d[v] > d[u] + w(u, v)$
      (yap eğer)
      (sonra) **then** $d[v] \leftarrow d[u] + w(u, v)$

Tokalaşma önkuramı $\Rightarrow \Theta(E)$ implicit (gizli) DECREASE-KEY's.
(azaltılmış anahtar)

Time(süre) $= \Theta(V \cdot T_{\text{EXTRACT-MIN}} + E \cdot T_{\text{DECREASE-KEY}})$

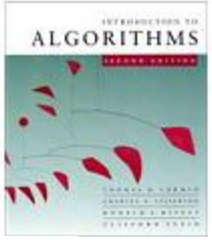
Not: Prim'in en az kapsayan ağaç algoritmasının çözümlemesinde de aynı formül.



Dijkstra' nın çözümlemesi (analizi)

$$\text{Süre} = \Theta(V)\cdot T_{\underset{\text{(en azı çıkar)}}{\text{EXTRACT-MIN}}} + \Theta(E)\cdot T_{\underset{\text{(azaltılmış anahtar)}}{\text{DECREASE-KEY}}}$$

Q	$T_{\text{EXTRACT-MIN}}$	$T_{\text{DECREASE-KEY}}$	Toplam
dizilim	$O(V)$	$O(1)$	$O(V^2)$
ikili yığın	$O(\lg V)$	$O(\lg V)$	$O(E \lg V)$ O((V+E)logV)
Fibonacci yığını	$O(\lg V)$ amortize edilmiş	$O(1)$ amortize edilmiş	$O(E + V \lg V)$ en kötü durum



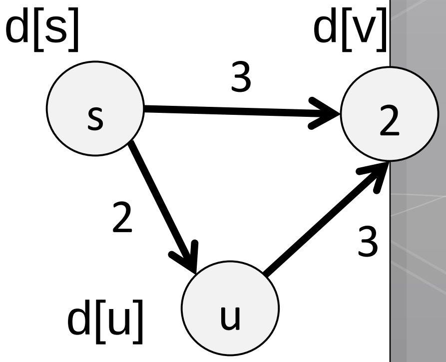
Doğruluk —Bölüm I

Önkuram. $d[s] \leftarrow 0$ ve $d[v] \leftarrow \infty$ ' yi tüm $v \in V - \{s\}$' ler için ilklendirme $d[v] \geq \delta(s, v)$' yi sağlar- tüm $v \in V$ ' ler için: Ve bu değişmez dizideki tüm gevşetme adımlarımda korunur.

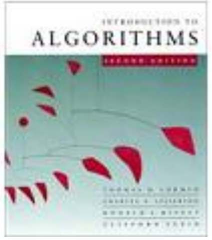
Kanıt. Şunun olmadığını düşünün. v, $d[v] < \delta(s, v)$' deki ilk köşe olsun ve u' da $d[v]$' yi değiştiren ilk köşe olsun: $d[v] = d[u] + w(u, v)$. O zaman,

$d[v] < \delta(s, v)$	kabul
$\leq \delta(s, u) + \delta(u, v)$	üçgen eşitsizliği
$\leq \delta(s,u) + w(u, v)$	kısa yol $\leq$ özel yol
$\leq d[u] + w(u, v)$	v ilk ihlal

Çelişki. $\square$



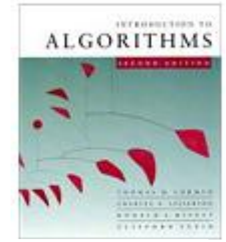
Genişletmede bir hata olduğunu gösterir



Doğruluk —Bölüm II

Ön kuram. u, s' den v' ye en kısa yolda v' nin atası olsun. O durumda, eğer $d[u] = \delta(s, u)$ve kenar (u, v) gevşetilmişse, gevşemeden sonra elimizde $d[v] = \delta(s, v)$ olur.

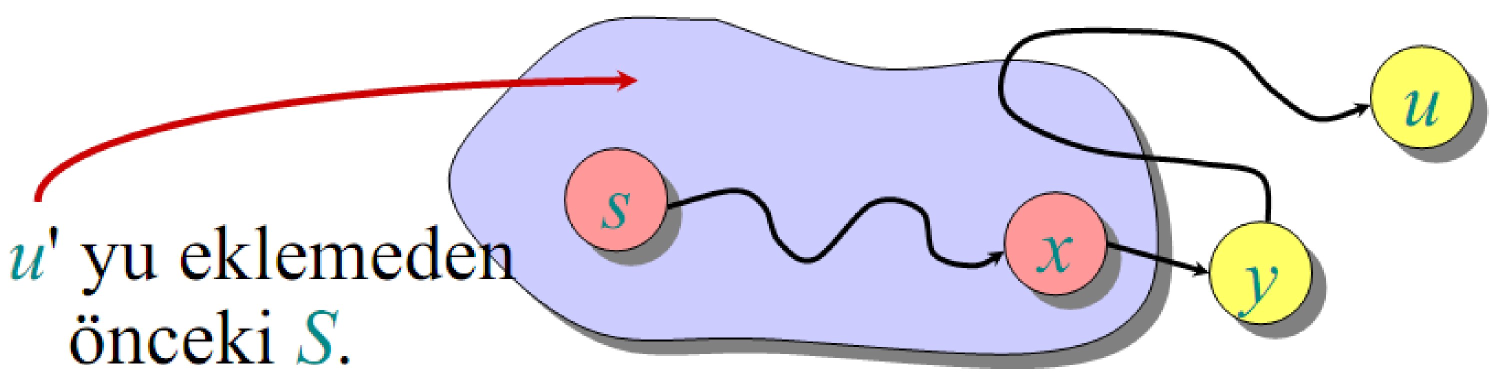
Kanıt. $\delta(s, v) = \delta(s, u) + w(u, v)$ olduğuna dikkat edin. Gevşetmeden önce $d[v] > \delta(s, v)$ olduğunu farzedin. (Diğer türlü, bitirmiştik.) Sonra, $d[v] > d[u] + w(u, v)$ testi başarılı, çünkü $d[v] > \delta(s, v) = \delta(s, u) + w(u, v) = d[u] + w(u, v)$ ve algoritma $d[v] = d[u] + w(u, v) = \delta(s, v)$' yi ayarlar. $\square$

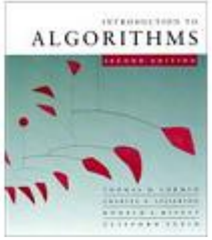


Doğruluk —Bölüm III

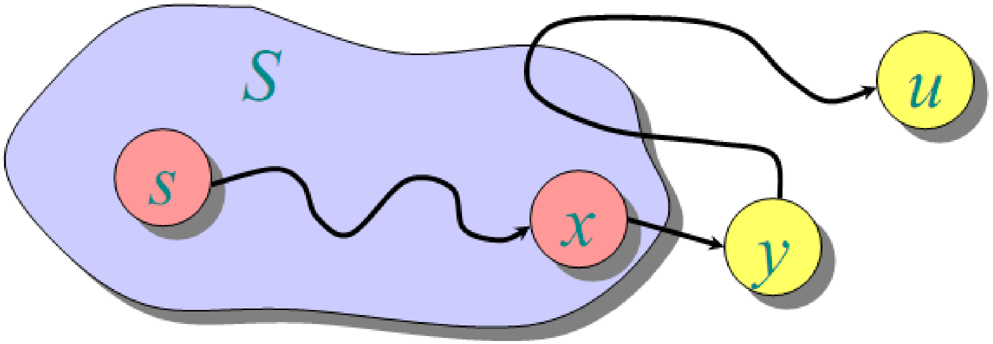
Teorem. Dijkstra algoritması tüm $v \in V$ için $d[v] = \delta(s, v)$ ile sonlanır.

Kanıt. v, S' ye eklenirken, her $v \in V$ için $d[v] = \delta(s, v)$ olduğunu göstermek yeterlidir. Her $d[u] > \delta(s, u)$ için u' nun S' ye eklenen ilk köşe olduğunu düşünün. y, s' den u' ya en kısa yol boyunca $V - S$ de ilk köşe olsun, x de onun atası olsun:





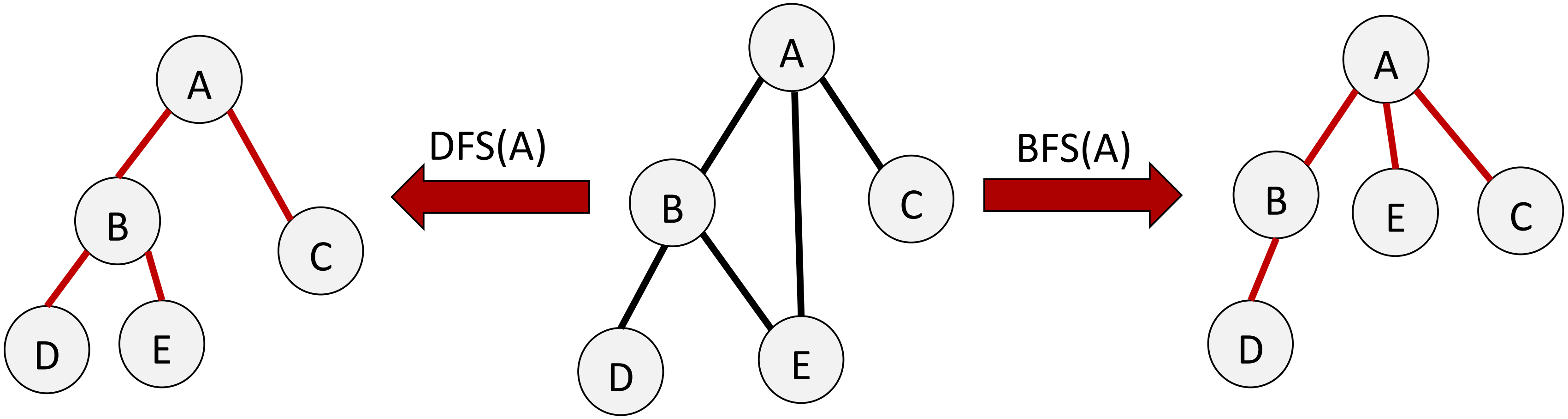
Doğruluk —Bölüm III

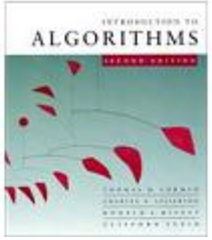


Eğer u, belirlenen değişmezi ihlal eden ilk köşe ise $d[x] = \delta(s, x)$ elde ederiz . x, S' ye eklendiğinde kenar (x, y) gevşetildi ki bu $d[y] = \delta(s, y) \leq \delta(s, u) < d[u]$ anlamına gelir. Fakat, bizim u seçimimizle $d[u] \leq d[y]$ olur. Çelişki. $\square$

MST Hesaplama – Ağırlıksız Graf

- Graf ağırlıksızsa veya tüm kenarların ağırlıkları eşit ise MST nasıl bulunur?
  - BSF veya DSF çalıştırın oluşan ağaç MST'dir





Ağırlıklandırılmamış grafikler

Tüm $(u, v) \in E$'ler için $w(u, v) = 1$ olduğunu farzedin Dijkstra algoritması geliştirilebilir mi?

- Bir öncelik sırası yerine basit FIFO sırası kullanın.

Enine arama

(-iken)**while** $Q \neq \emptyset$

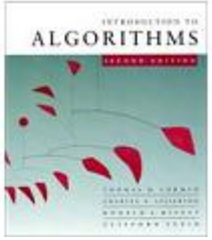
(yap)**do** $u \leftarrow \text{DEQUEUE}(Q)$ (sıradan çıkar)

(her) **for each** $v \in Adj[u]$(için)

(yap eğer) **do if** $d[v] = \infty$

(sonra) **then** $d[v] \leftarrow d[u] + 1$

$\text{ENQUEUE}(Q, v)$(sıraya ekle)



Ağırlıklandırılmamış grafikler

Tüm $(u, v) \in E$'ler için $w(u, v) = 1$ olduğunu farzedin. Dijkstra algoritması geliştirilebilir mi?

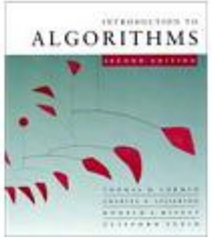
- Bir öncelik sırası yerine basit FIFO sırası kullanın.

Sığ öncelikli arama

(-iken)**while** $Q \neq \emptyset$
    (yap)**do** $u \leftarrow$ DEQUEUE(Q) (sıradan çıkar)
        (her)**for each** $v \in Adj[u]$(için)
        (yap eğer)**do if** $d[v] = \infty$
            (sonra)**then** $d[v] \leftarrow d[u] + 1$
                ENQUEUE(Q, v)(sıraya ekle)

Çözümleme: Time(süre)= $O(V + E)$.

(BFS)Enine arama'nın doğruluğu



(-iken)**while** $Q \neq \varnothing$

(yap)**do** $u \leftarrow \text{DEQUEUE}(Q)$ (sıradan çıkar)

(her) **for each** $v \in Adj[u]$(için)

(yap eğer) **do if** $d[v] = \infty$

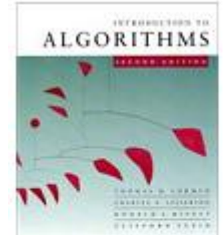
(sonra) **then** $d[v] \leftarrow d[u] + 1$

$\text{ENQUEUE}(Q, v)$(sıraya ekle)

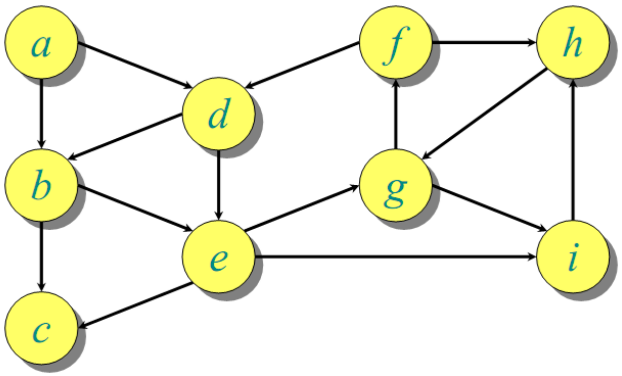
Anahtar fikir:

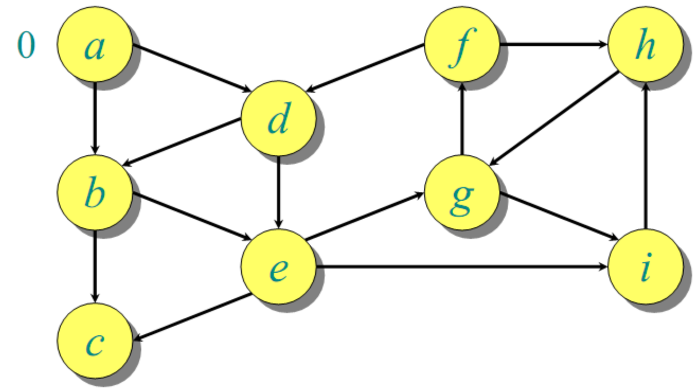
Enine aramadaki FIFO Q, Dijkstra' nın öncelikli sıralamasındaki kuyruk Q' yu taklit eder.

•**Değişmez:** Q'da v, u' dan sonra gelirse bu $d[v] = d[u]$ ya da $d[v] = d[u] + 1$ anlamına gelir.



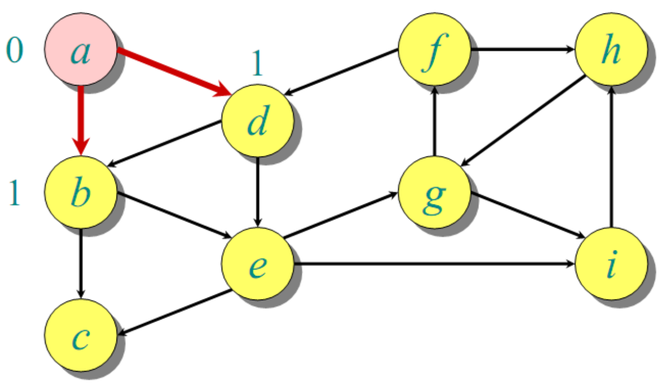
Enine arama için örnek



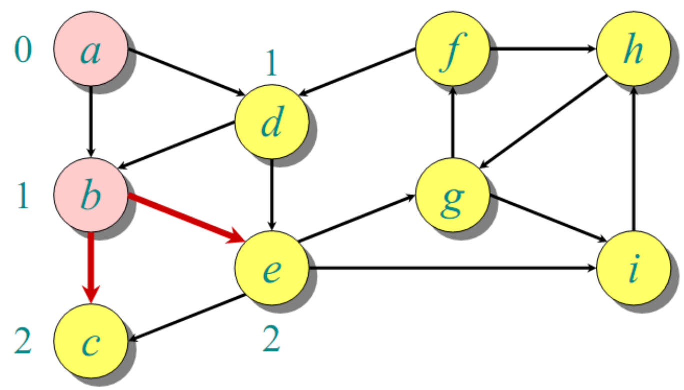


Q:

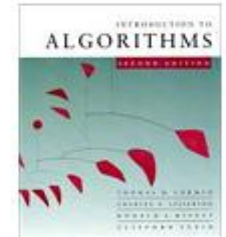
0
Q: a



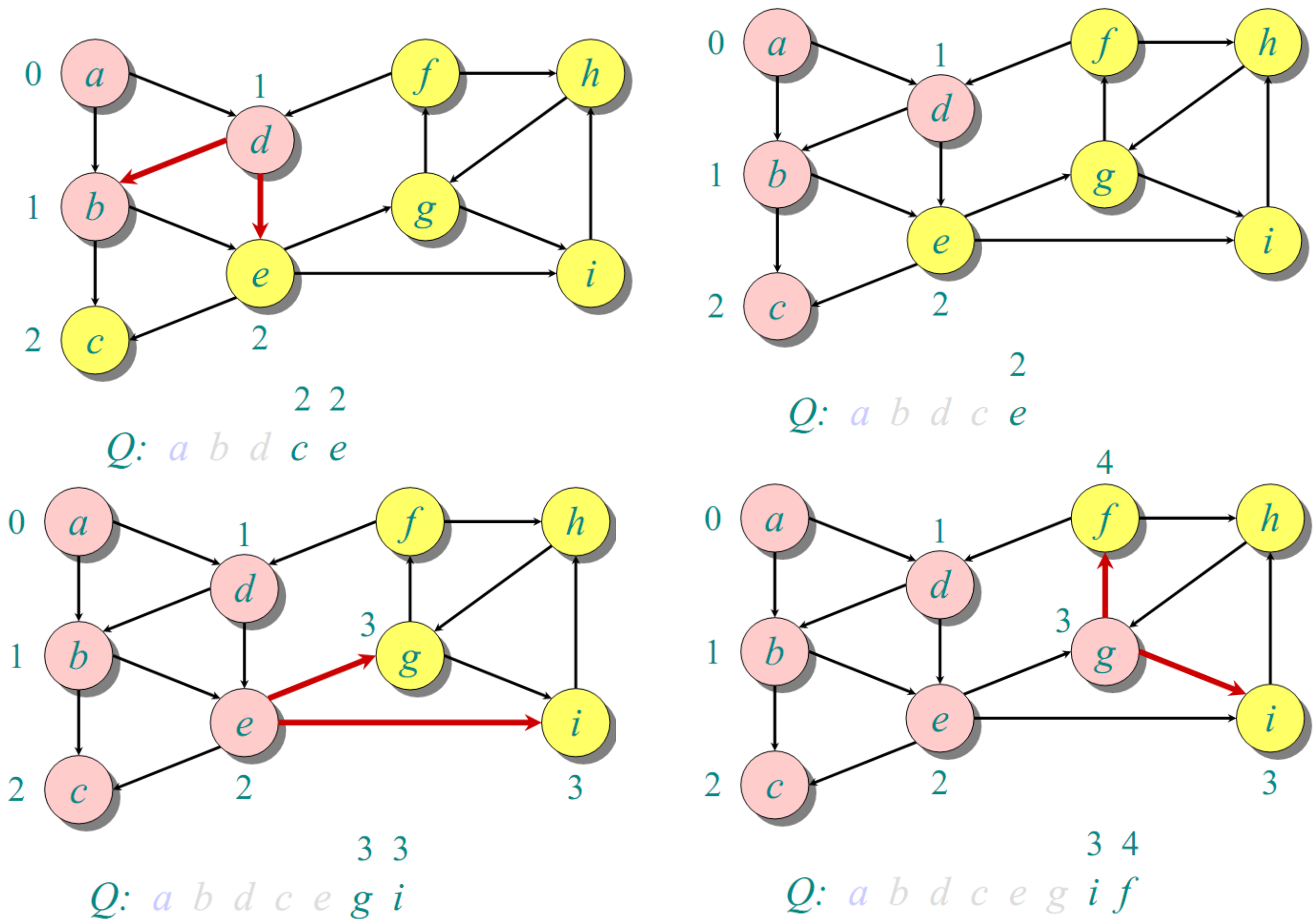
1 1
Q: a b d

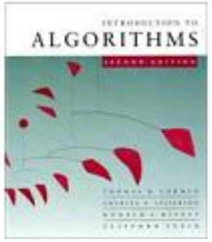


1 2 2
Q: a b d c e

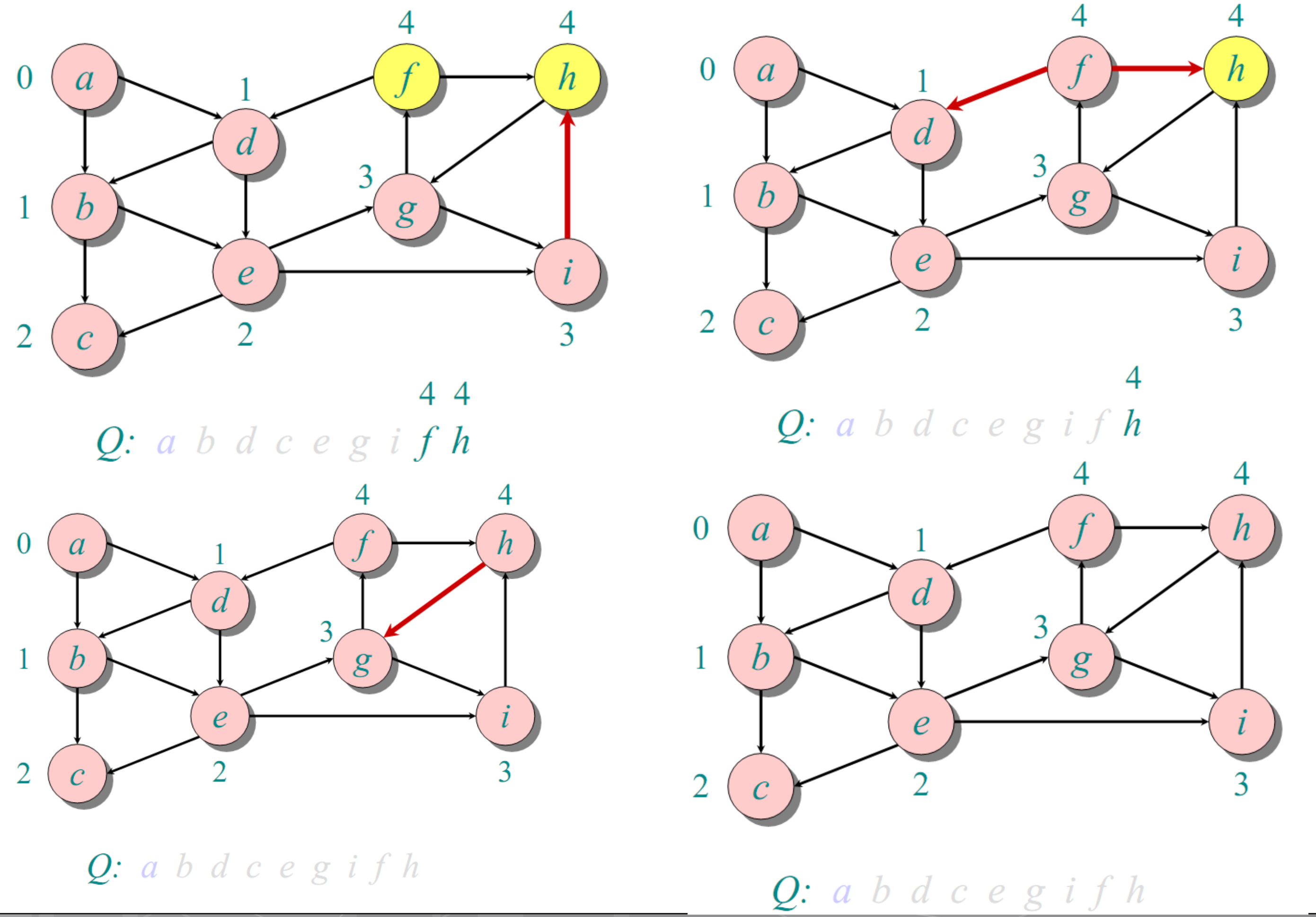


Enine arama için örnek





Enine arama için örnek

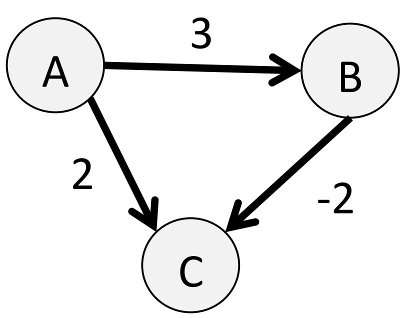


Negatif Ağırlıklı En Kısa Yollar

- Doğruluk
- Çözümleme

Negatif Ağırlıklı Dijkstra

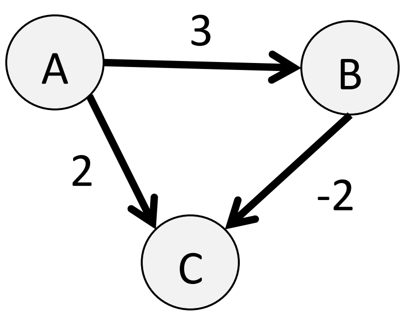
- Eğer kenarların ağırlıkları negatif ise Dijkstra algoritması en az maliyetli yolu bulmada başarısız oluyor.
- Bunun sebebi eksi (-) değerdeki kenarın sürekli olarak mevcut durumdan daha iyi bir sonuç üretmesi ve algoritmanın hiçbir zaman için kararlı hale gelememesidir.



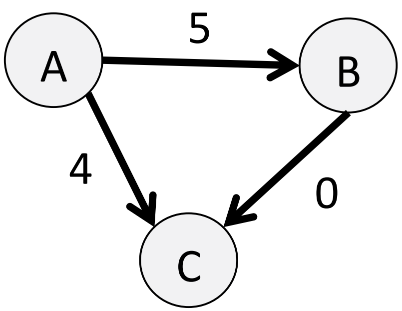
- A’dan C’ye en az maliyetli yol
  - Dijkstra : A->C, maliyet: 2
  - Gerçek yol A->B->C, maliyet: 1
- Her kenara pozitif sabit eklersek ne olur?

Negatif Ağırlıklı Dijkstra

- Her kenara pozitif sabit eklersek ne olur?



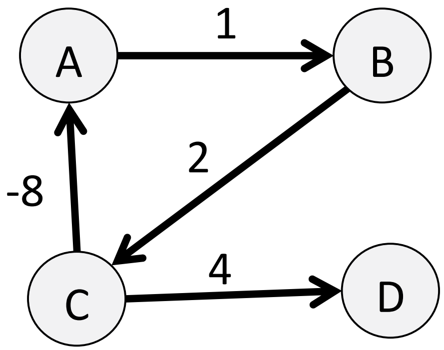
Her kenara
2 ekle



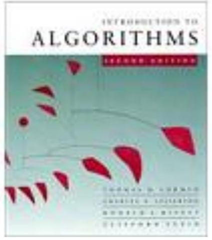
- A'dan C'ye en az maliyetli yol
  - Dijkstra: A->C
  - Gerçek Yol: A->B->C

Negatif Maliyetli Çember

- Eğer graf negatif maliyetli çember içeriyorsa, en az maliyetli yol tanımlanamaz.



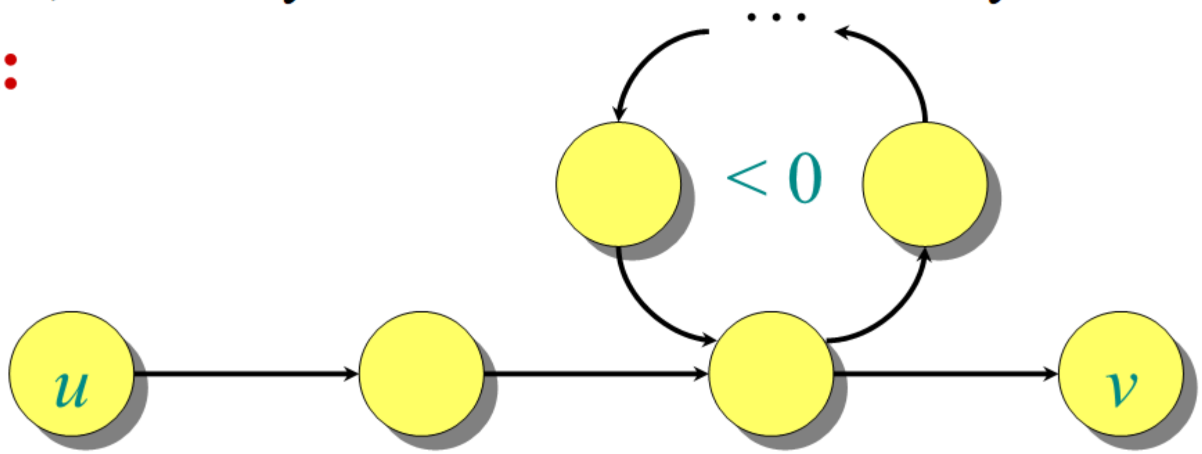
- A'dan D'ye en az maliyetli yol nedir?
  - veya B'den C'ye?



Negatif-ağırlık çevrimleri

Hatırlatma: Eğer grafik $G = (V, E)$ negatif ağırlık çevrimi içeriyorsa, en kısa yollardan bazıları bulunmayabilir.

Örnek:

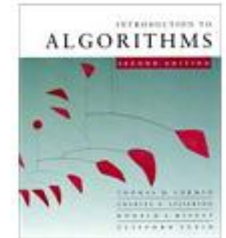


Bellman-Ford algoritması: Bir $s \in V$ kaynağından tüm $v \in V'$ lere bütün kısa yol uzunluklarını bulur ya da bir negatif ağırlık çevrimi olduğunu saptar.

Bellman-Ford Algoritması

- Ana mantık:
  - Her düğüm için bir uzaklık tahmini oluşturulur.
  - Başlangıç olarak maliyet(s)=0 diğer düğümler için maliyet(u)= ∞ olarak atanır.
  - En az maliyetli yol hesaplanana kadar tüm kenarlar üzerinden güncelleme yapılır.
- Algoritma ayrıca grafın negatif maliyetli kenarının olup olmadığını da bulur.
- Dijkstra’nın algoritmasına göre daha yavaş çalışır.
- (Dijkstra algoritması negatif kenarlarda kullanılmaz.)

Bellman-Ford algoritması



$d[s] \leftarrow 0$
for each(her bir) $v \in V - \{s\}$(için)
    do(yap) $d[v] \leftarrow \infty$

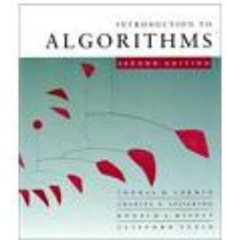
ilklendirme

for(için) $i \leftarrow 1$ **to** $|V| - 1$ ' (e)
    do for each edge(her kenar için yap) $(u, v) \in E$
        do(yap) **if**(eğer) $d[v] > d[u] + w(u, v)$
            then(sonra) $d[v] \leftarrow d[u] + w(u, v)$

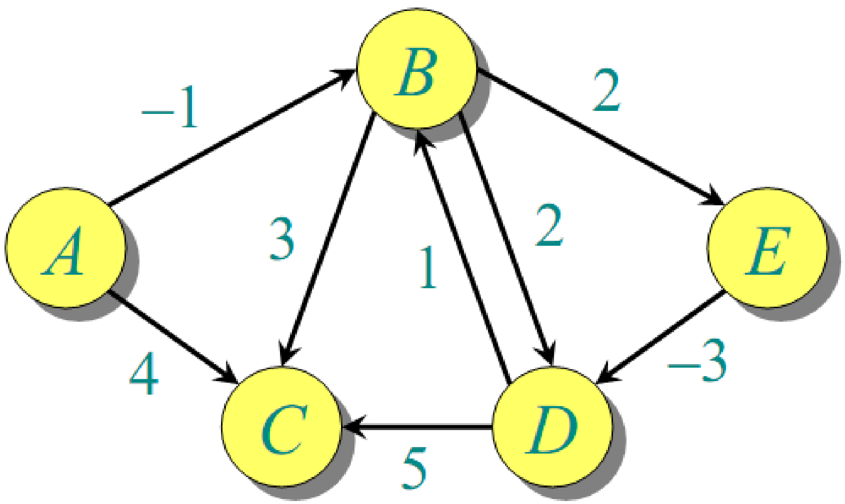
Gevşetme adımı

(her)**for** each edge $(u, v) \in E$ (kenarı için)
(yap eğer)**do if** $d[v] > d[u] + w(u, v)$
    sonra bunu negatif ağırlık çevrimi var diyerek raporla

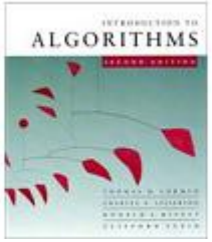
Sonunda, $d[v] = \delta(s, v)$, negatif ağırlık çevrimi yoksa.
Süre = $O(VE)$.



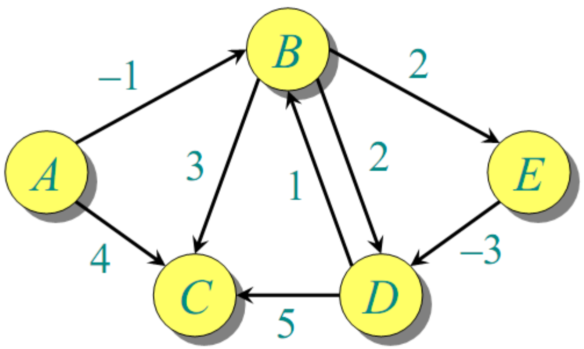
Bellman-Ford örneği

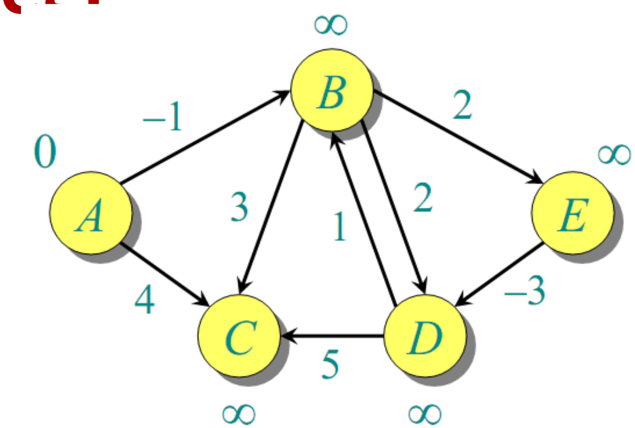


A	B	C	D	E
0	∞	∞	∞	∞

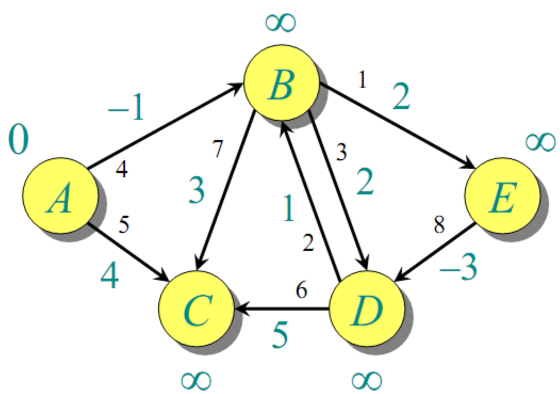


Bellman-Ford örneği

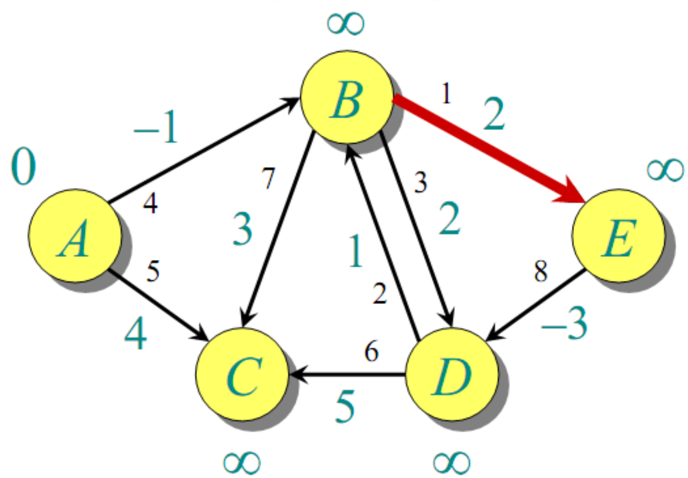


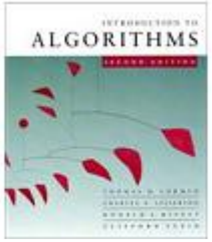


İlklendirme.

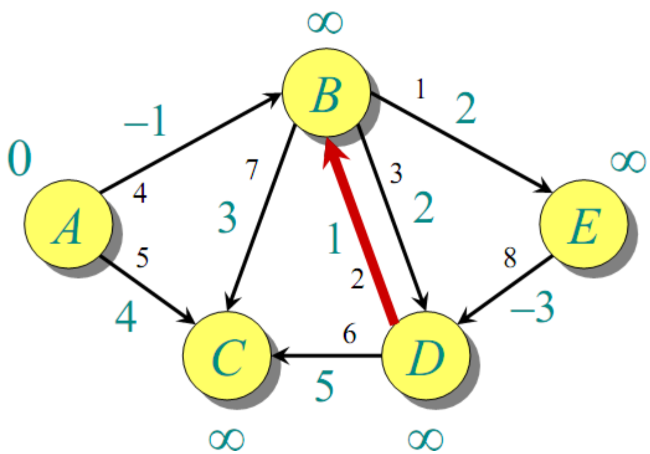


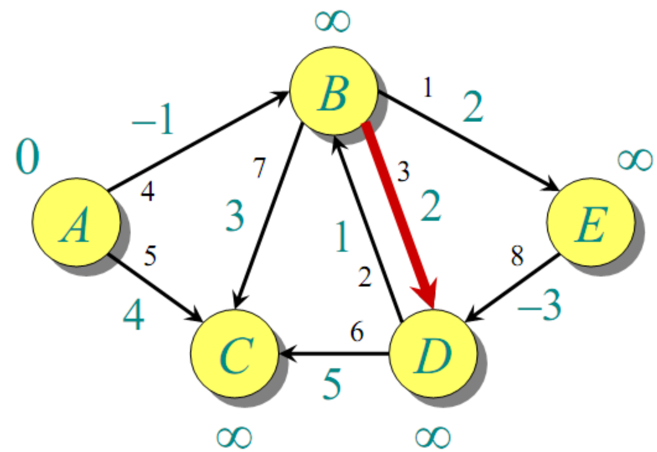
Köşe gevşetme düzeni

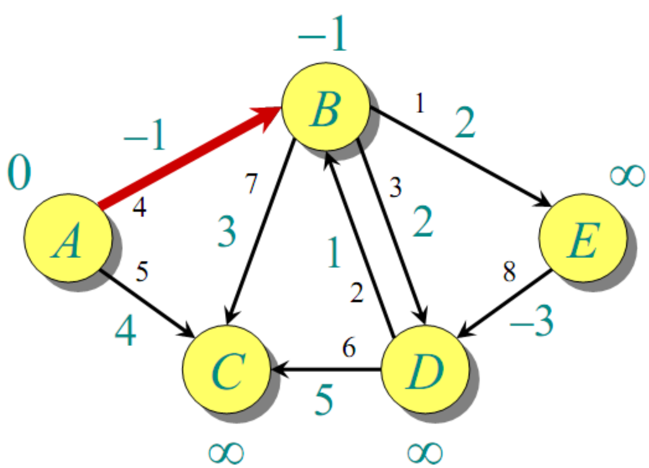




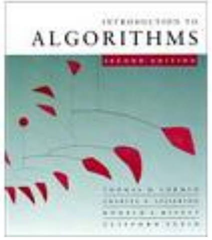
Bellman-Ford örneği



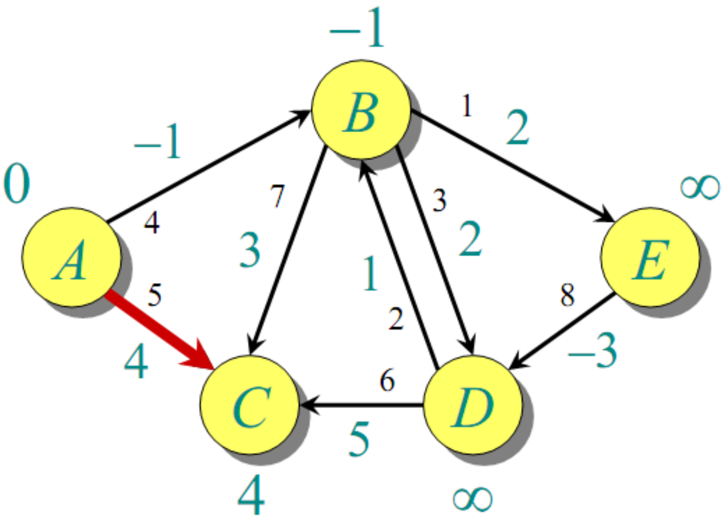




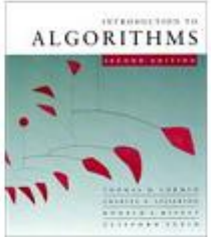
A	B	C	D	E
0	∞	∞	∞	∞
0	-1	∞	∞	∞



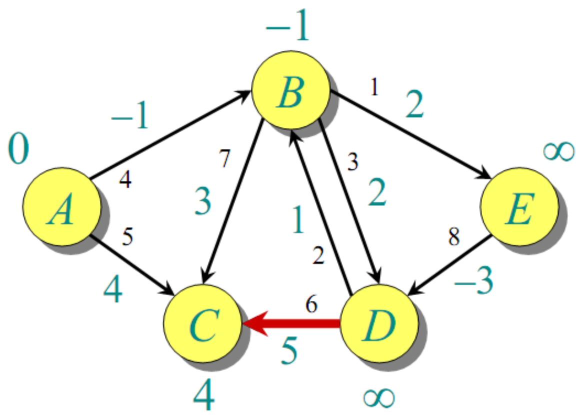
Bellman-Ford örneği

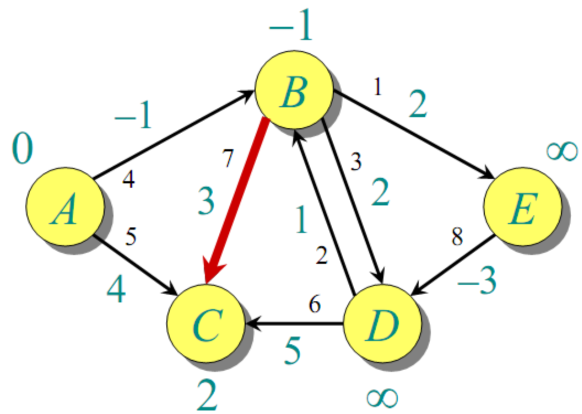


A	B	C	D	E
0	∞	∞	∞	∞
0	–1	∞	∞	∞
0	–1	4	∞	∞

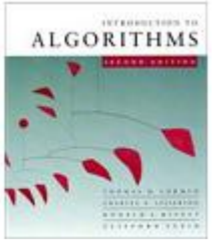


Bellman-Ford örneği

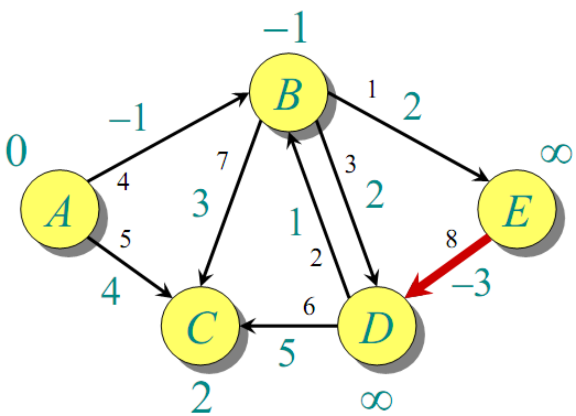




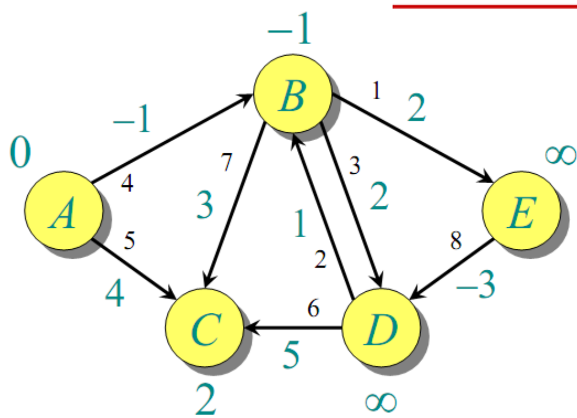
A	B	C	D	E
0	∞	∞	∞	∞
0	-1	∞	∞	∞
0	-1	4	∞	∞
0	-1	2	∞	∞



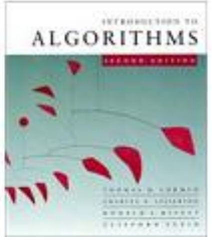
Bellman-Ford örneği



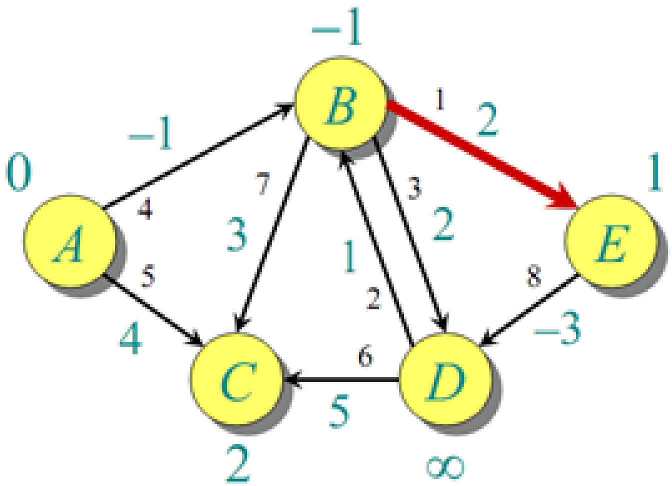
A	B	C	D	E
0	∞	∞	∞	∞
0	–1	∞	∞	∞
0	–1	4	∞	∞
0	–1	2	∞	∞



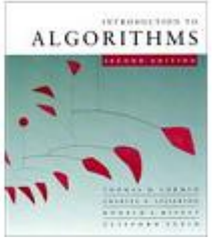
1. geçişin sonunda



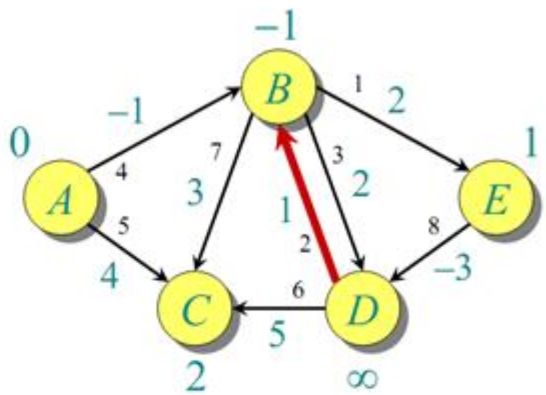
Bellman-Ford örneği

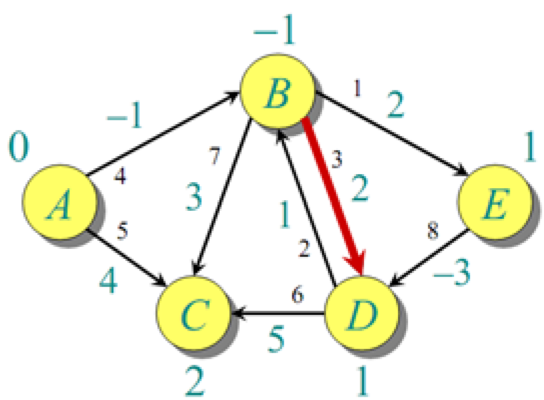


A	B	C	D	E
0	∞	∞	∞	∞
0	−1	∞	∞	∞
0	−1	4	∞	∞
0	−1	2	∞	∞
0	−1	2	∞	1

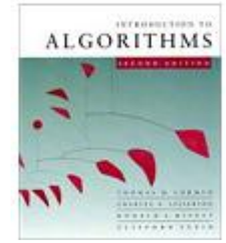


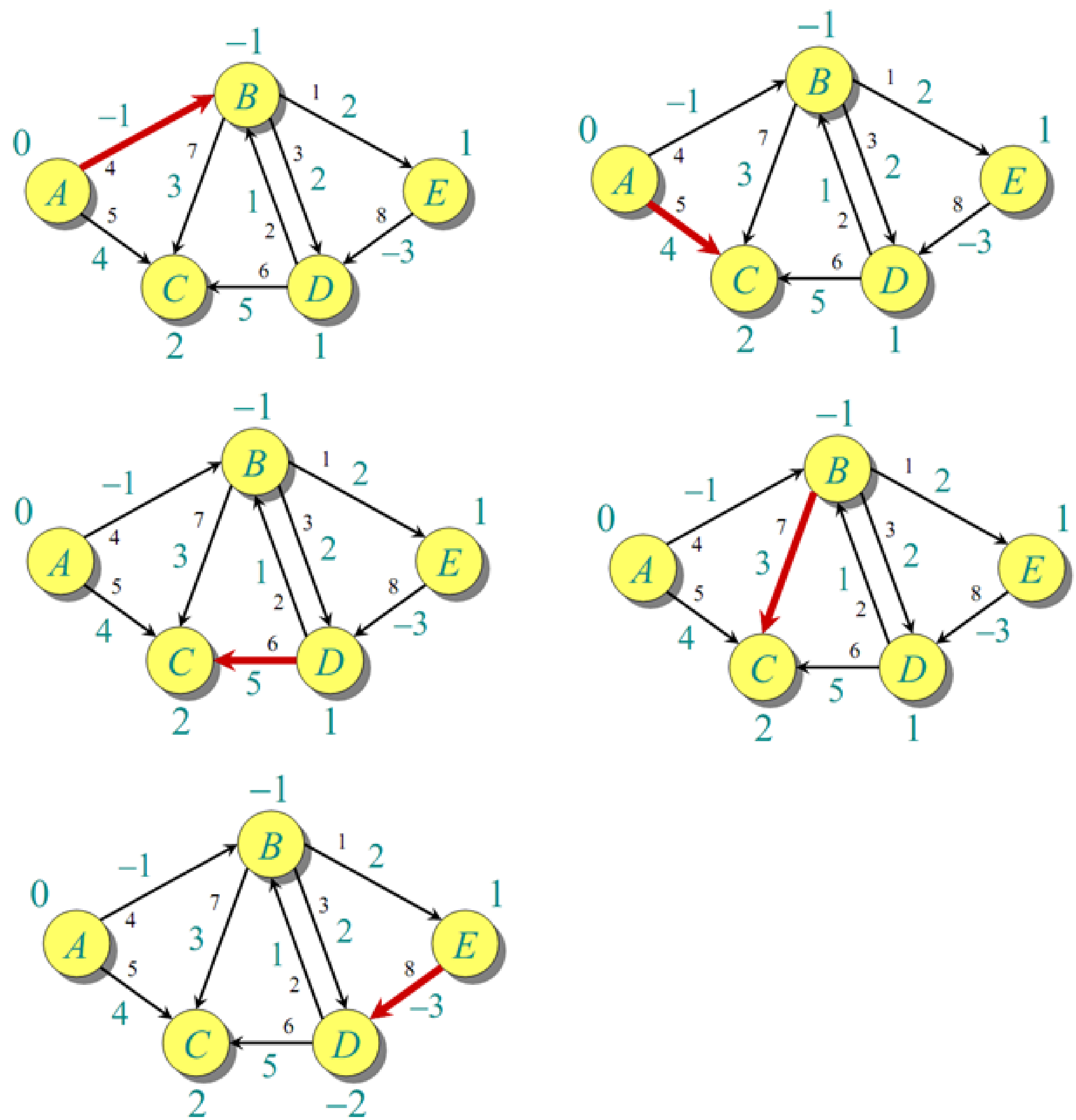
Bellman-Ford örneği





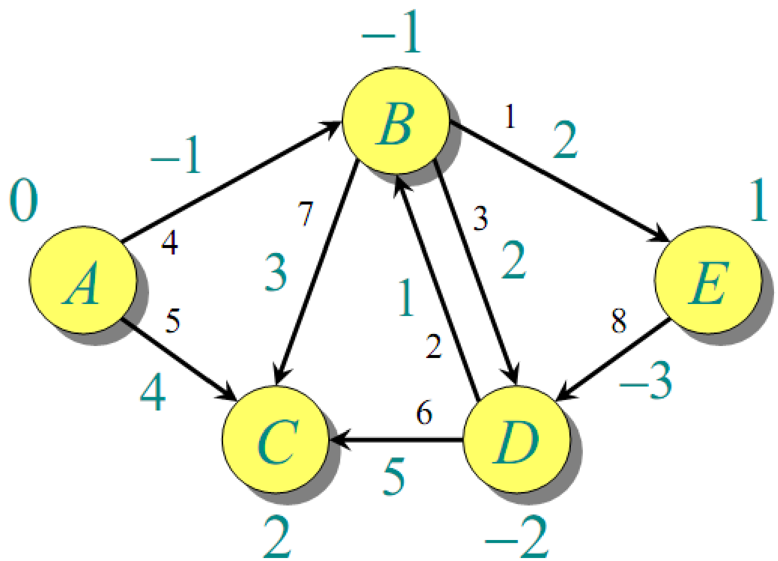
A	B	C	D	E
0	∞	∞	∞	∞
0	-1	∞	∞	∞
0	-1	4	∞	∞
0	-1	2	∞	∞
0	-1	2	∞	1
0	-1	2	1	1



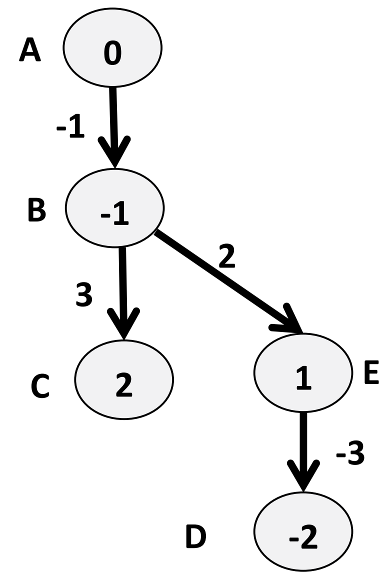


A	B	C	D	E
0	∞	∞	∞	∞
0	–1	∞	∞	∞
0	–1	4	∞	∞
0	–1	2	∞	∞
0	–1	2	∞	1
0	–1	2	1	1
0	–1	2	–2	1

Bellman-Ford örneği



A	B	C	D	E
0	∞	∞	∞	∞
0	–1	∞	∞	∞
0	–1	4	∞	∞
0	–1	2	∞	∞
0	–1	2	∞	1
0	–1	2	1	1
0	–1	2	–2	1

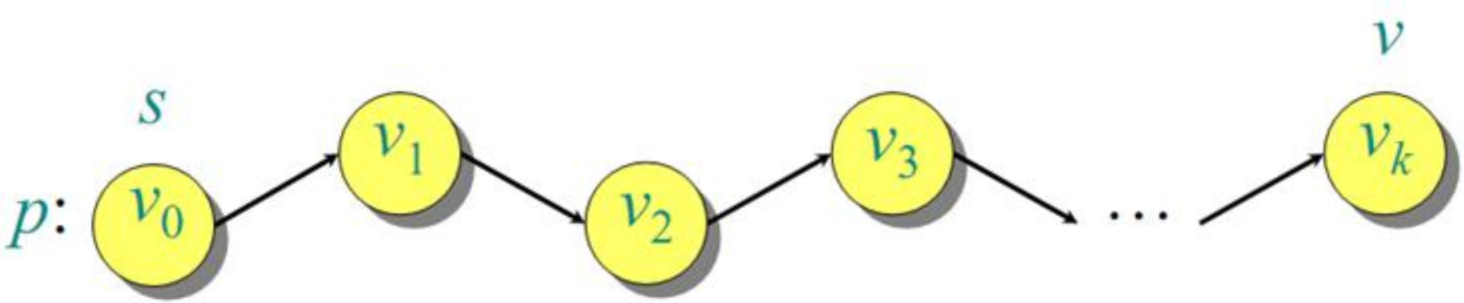


2. geçişin sonu (ve 3 ve 4).

Doğruluk

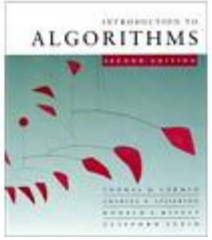
Teorem. Eğer $G = (V, E)$ hiç negatif ağırlık çevrimi içermiyorsa, sonrasında Bellman-Ford algoritması bütün $v \in V$ ler için $d[v] = \delta(s, v)$ yi çalıştırır.

Kanıt. $v \in V$ herhangi bir köşe olsun ve s' den v' ye, üzerinde en az sayıda köşe olan en kısa yolun p olduğunu farzedin.

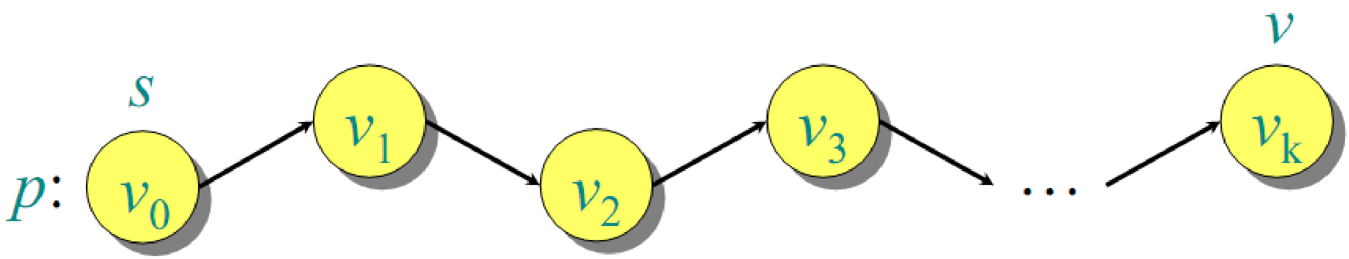


p en kısa yol ise,

$$\delta(s, v_i) = \delta(s, v_{i-1}) + w(v_{i-1}, v_i).$$



Doğruluk



İlk olarak, $d[v_0] = 0 = \delta(s, v_0)$ ve $d[v_0]$sonraki gevşetmeler tarafından değiştirilmemiş.
(Ders 14' teki $d[v] \geq \delta(s, v)$ kuramı sebebiyle).

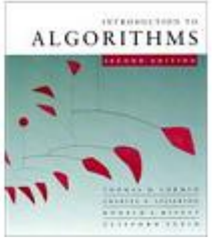
- E' den 1 geçiş sonra, $d[v_1] = \delta(s, v_1)$.
- E' den 2 geçiş sonra, $d[v_2] = \delta(s, v_2)$.

  ⋮

- E' den k geçiş sonra, $d[v_k] = \delta(s, v_k)$.

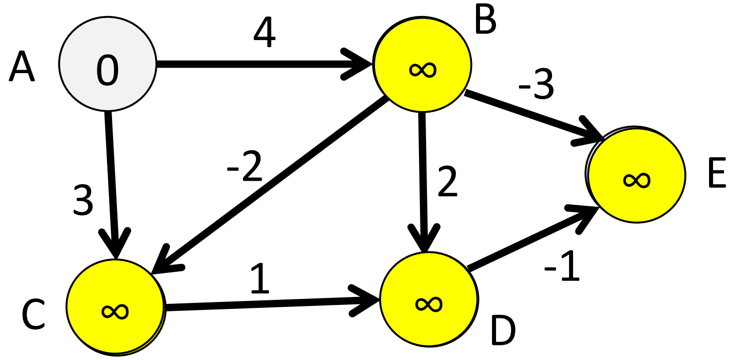
Eğer G negatif ağırlık çevrimi içermiyorsa, p basittir.
En uzun basit yolun $\leq |V| - 1$ kadar kenarı vardır. □

Negatif ağırlık çevrimlerini bulma



Doğal Sonuç. $|V|-1$ geçiş sonra $d[v]$ değeri birleşmede başarısız olursa, G' de s' den erişilebilir bir negatif ağırlık çevrimi vardır.

Bellman-Ford Alg: Örnek

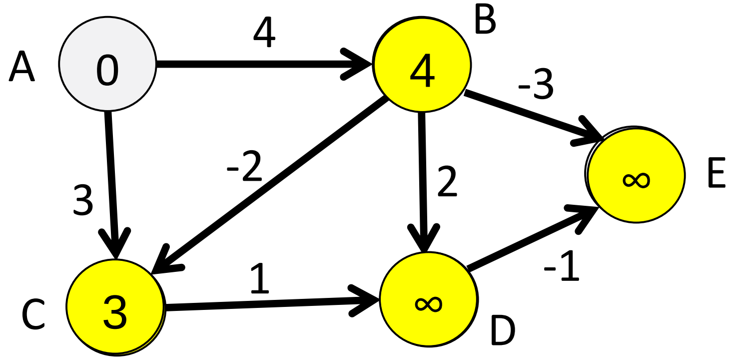


Düğüm	M	Ata
A	0	-
B	∞	
C	∞	
D	∞	
E	∞	

1	2	3	4	5	6	7
(C, D)	(A, B)	(A, C)	(B, C)	(B, D)	(B, E)	(D, E)

İlk Yineleme
(İlklendirme)

Bellman-Ford Alg: Örnek

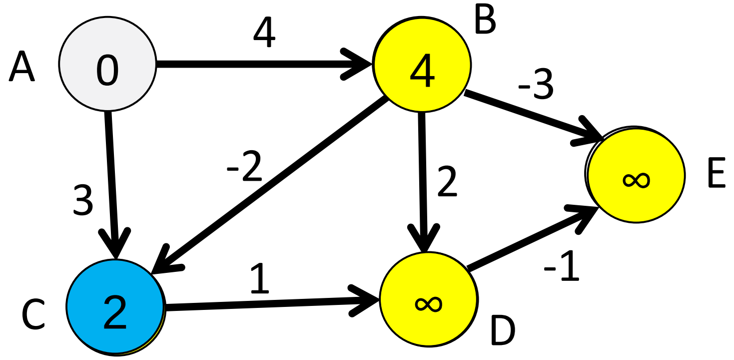


Düğüm	M	Ata
A	0	-
B	4	A
C	3	A
D	∞	
E	∞	

1	2	3	4	5	6	7
(C, D)	(A, B)	(A, C)	(B, C)	(B, D)	(B, E)	(D, E)

Köşe gevşetme düzeni

Bellman-Ford Alg: Örnek

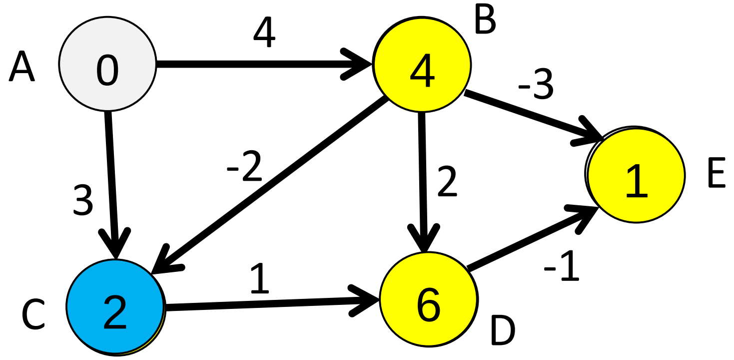


Düğüm	M	Ata
A	0	-
B	4	A
C	2	B
D	∞	
E	∞	

1	2	3	4	5	6	7
(C, D)	(A, B)	(A, C)	(B, C)	(B, D)	(B, E)	(D, E)

Köşe gevşetme düzeni

Bellman-Ford Alg: Örnek



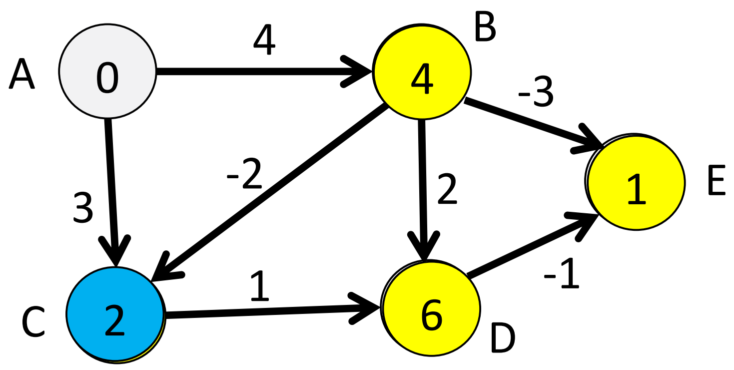
Düğüm	M	Ata
A	0	-
B	4	A
C	2	B
D	6	B
E	1	B

1	2	3	4	5	6	7
(C, D)	(A, B)	(A, C)	(B, C)	(B, D)	(B, E)	(D, E)

1.Tur tamamlandı

Bellman-Ford Alg: Örnek

- B=AB 4, min(A,B)= 0-> 0+4 = 4, C=AC 3, min(A,C)=0->0+3 =3
- C= BC -2, min(B,C)= 4-> 4-2 = 2, D=BD 2, min(B,D)=4->4+2 =6
- E= BE -3, min(B,C)= 4-> 4-3 = 1,



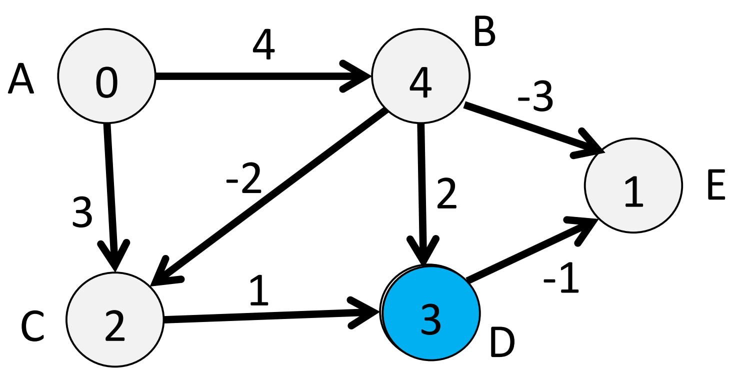
Düğüm	Ma.	Pred
A	0	-
B	4	A
C	2	B
D	6	B
E	1	B

(C, D) (A, B) (A, C) (B, C) (B, D) (B, E) (D, E)

1.Tur sonu

Bellman-Ford Alg: Örnek

- D=CD 1, min(C,D)= 2-> 2+1 =3

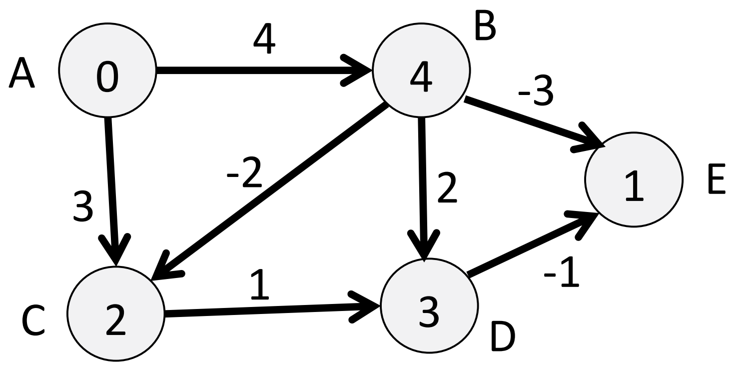


Düğüm	Ma	Pred
A	0	-
B	4	A
C	2	B
D	3	C
E	1	B

1	2	3	4	5	6	7
(C, D)	(A, B)	(A, C)	(B, C)	(B, D)	(B, E)	(D, E)

2. tur yineleme

Bellman-Ford Alg: Örnek

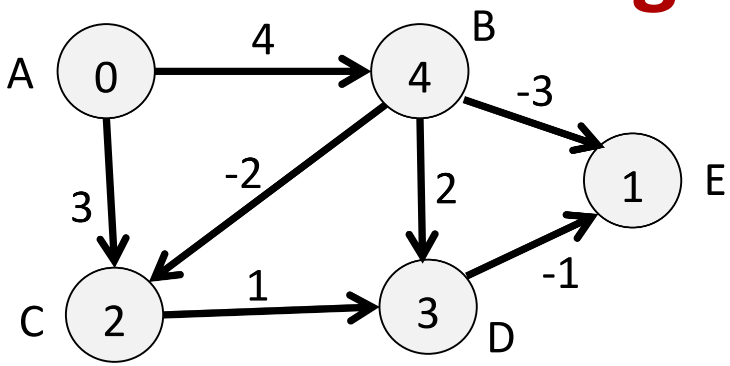


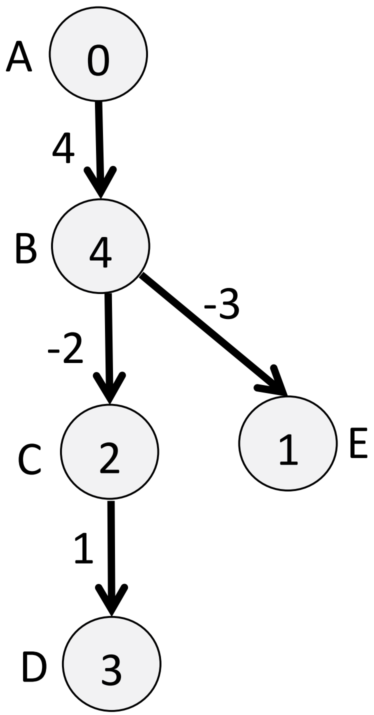
Düğüm	Ma	Pred
A	0	-
B	4	A
C	2	B
D	3	C
E	1	B

(C, D) (A, B) (A, C) (B, C) (B, D) (B, E) (D, E)

Üçüncü & Dördüncü Yineleme

Bellman-Ford Alg: Sonuç





Düğüm	Ma	Pred
A	0	-
B	4	A
C	2	B
D	3	C
E	1	B