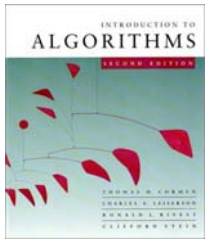
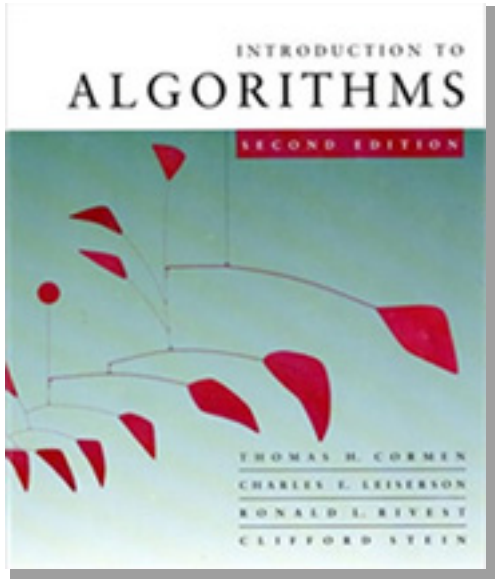


Algoritmalara Giriş

6.046J/18.401J

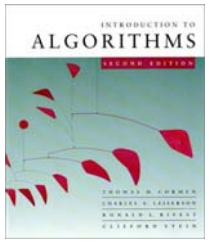


Ders 17

En kısa yollar I

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

Prof. Erik Demaine

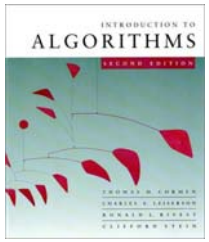


Grafiklerde yollar

$w : E \to \mathbb{R}$ 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.



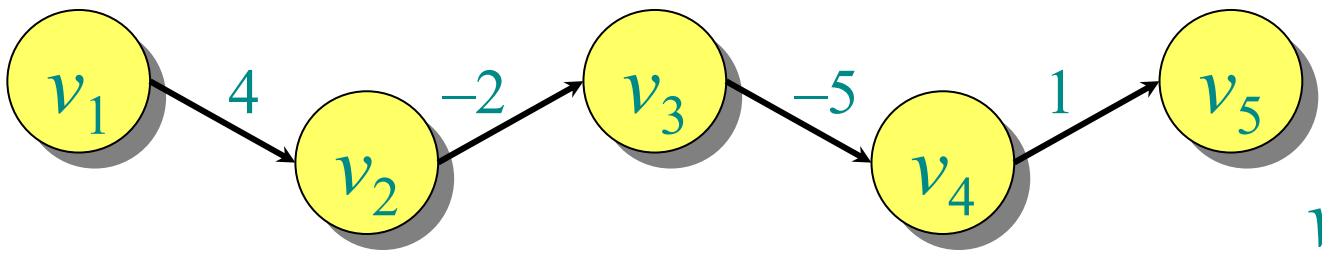
Grafiklerde yollar

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

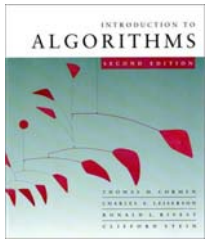
$$p = v_1 \to v_2 \to \cdots \to v_k$$

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

Örnek:



$w(p) = -2$

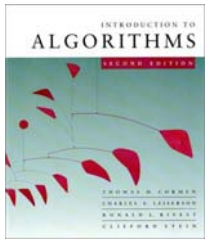


En kısa yollar

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

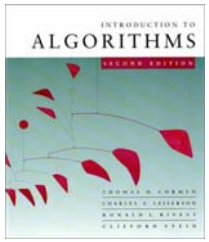
u ' dan v ' ye en kısa yolun ağırlığı $\delta(u, v) = \min\{w(p)$ olarak tanımlanır: p, u dan v ye bir yoldur$\}$.

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



En uygun altyapı

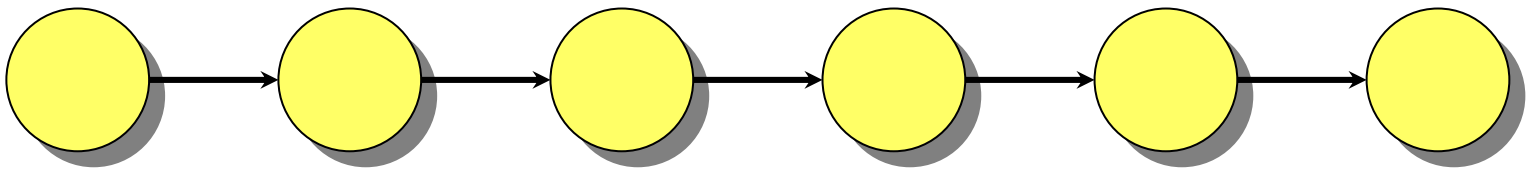
Teorem. En kısa yolun alt yolu, bir en kısa yoldur.

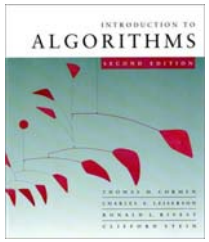


En uygun altyapı

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

Kanıt. Kes ve yapıştır:

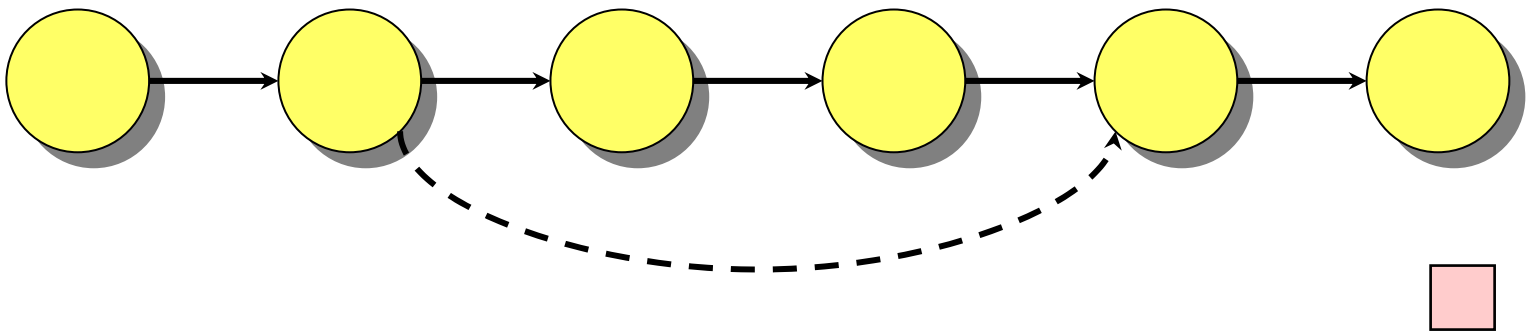


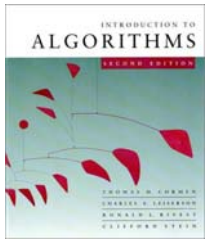


En uygun altyapı

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

Kanıt. Kes ve yapıştır:

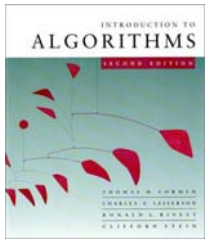




Üçgen eşitsizliği

Teorem. Tüm $u, v, x \in V$ ler için,

$$\delta(u, v) \leq \delta(u, x) + \delta(x, v).$$

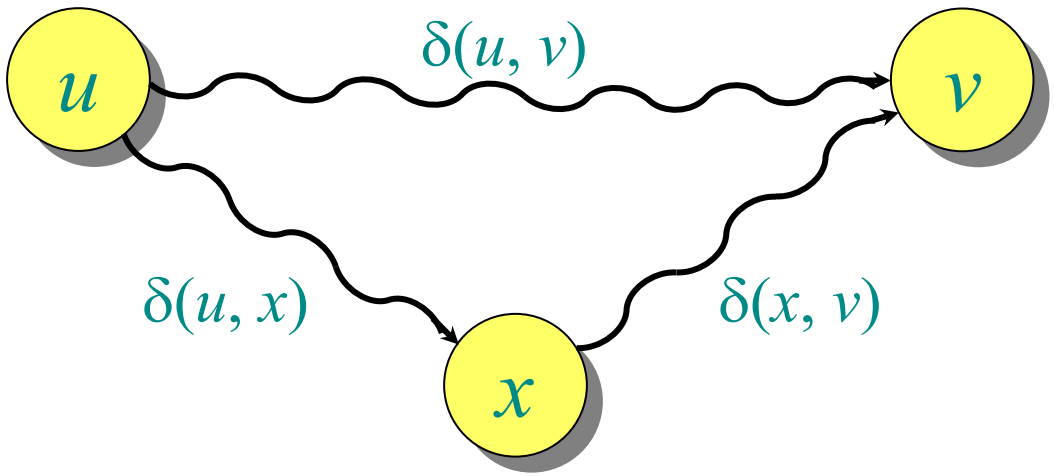


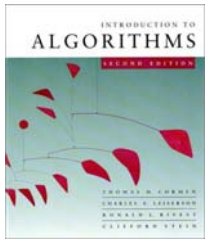
Üçgen eşitsizliği

Teorem. Tüm $u, v, x \in V$ ler için,

$$\delta(u, v) \leq \delta(u, x) + \delta(x, v).$$

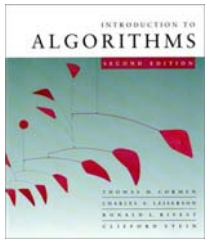
Kanıt.





En kısa yolların iyi tanımlanırlığı

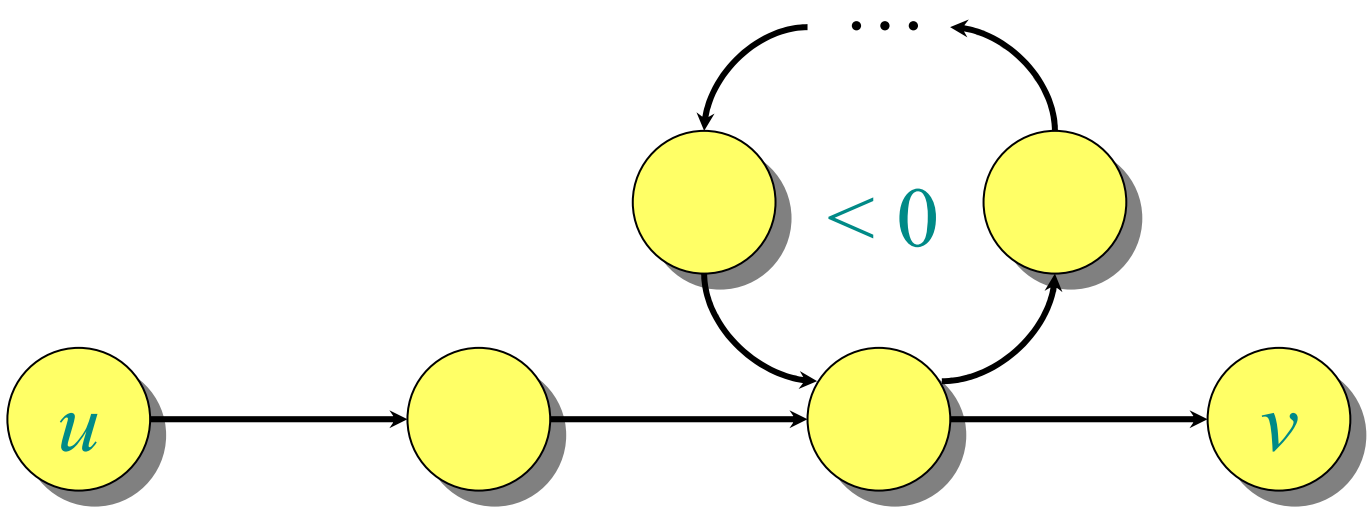
Bir G grafiği negatif ağırlık döngüsü içeriyorsa, bazı en kısa yollar var olmayabilir.

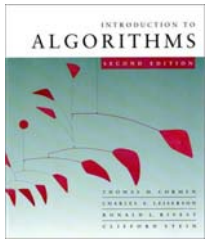


En kısa yolların iyi tanımlanırlığı

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

Örnek:





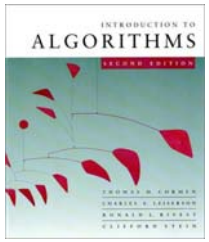
Tek-kaynaklı en kısa yollar

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.



Dijkstra algoritması

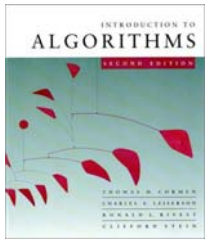
$d[s] \leftarrow 0$

(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.



Dijkstra algoritması

$d[s] \leftarrow 0$

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

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

$S \leftarrow \emptyset$

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

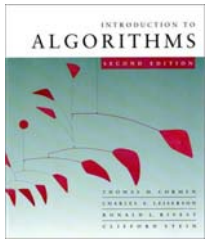
(yap) **do** $u \leftarrow$ EXTRACT-MIN(Q) (en azı çıkar)

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

(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 is a priority queue maintaining $V - S$

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

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

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

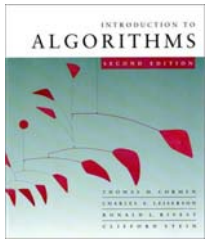
(her) **for** each $v \in Adj[u]$ (için)

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

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

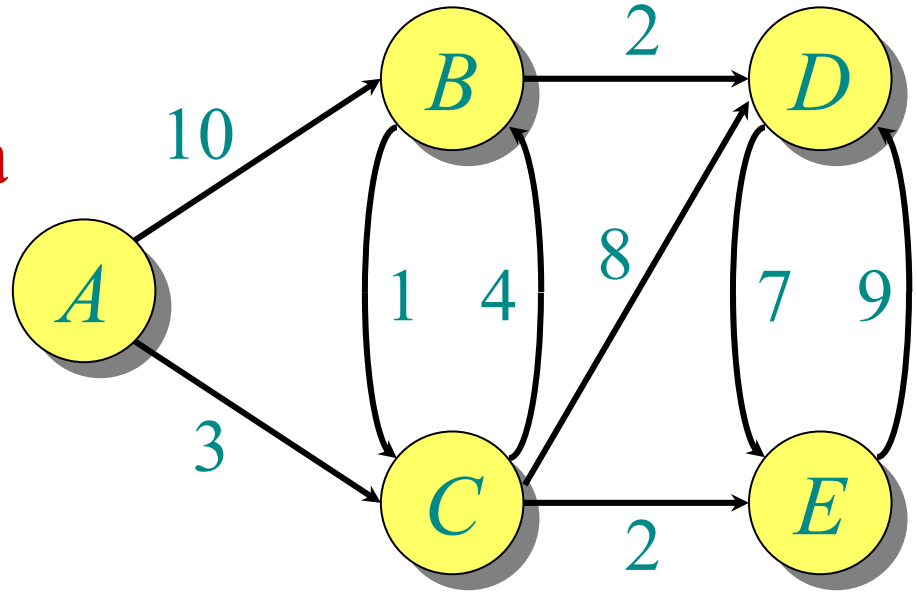
Gevşeme Adımı

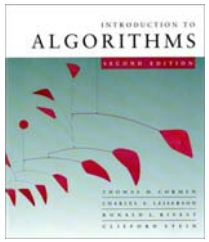
Implicit DECREASE-KEY (azaltılmış anahtar)



Dijkstra algoritmasına örnek

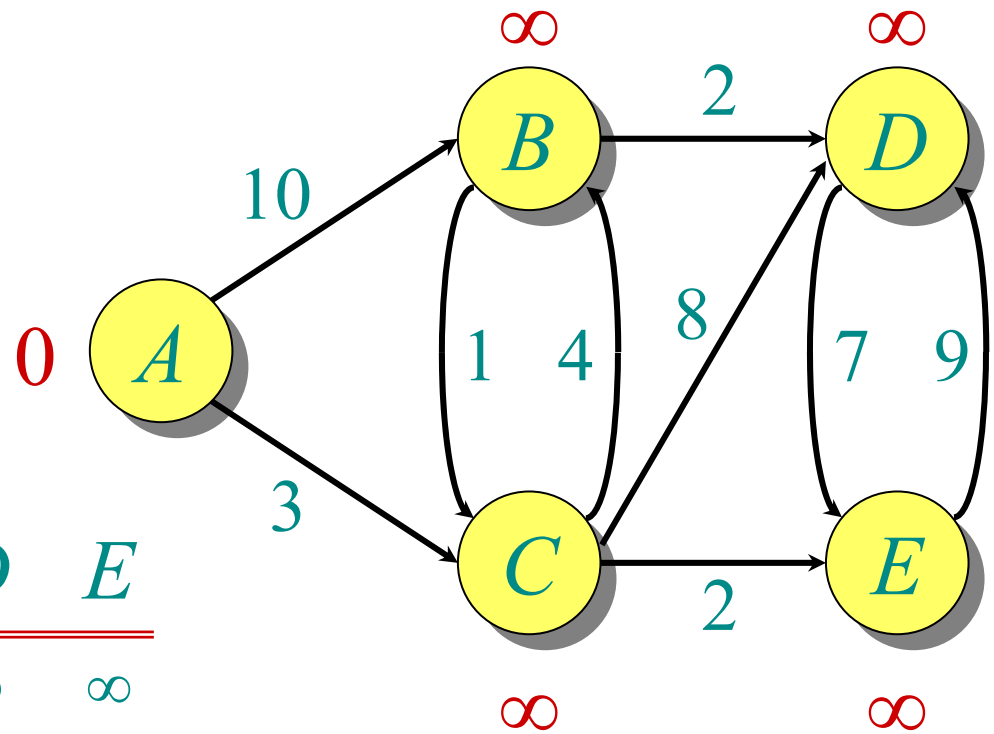
Eksi olmayan kenar ağırlıklarıyla grafik:





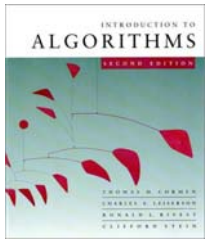
Dijkstra algoritmasına örnek

İlklendirme:



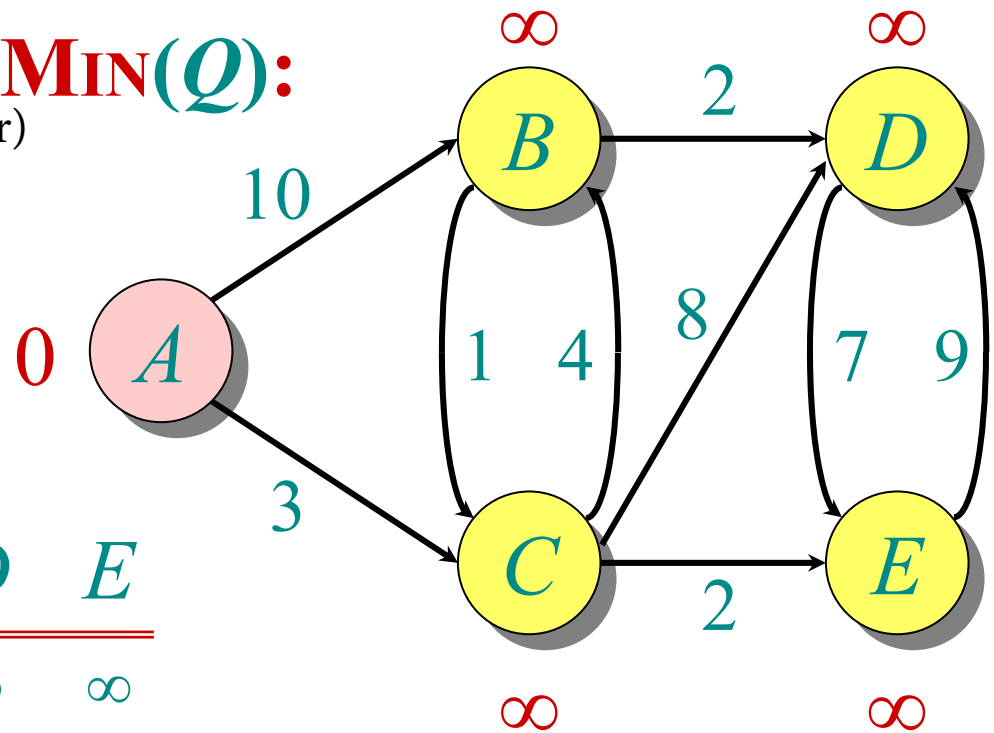
Q:	A	B	C	D	E
	0	∞	∞	∞	∞

S: {}



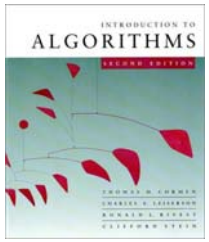
Dijkstra algoritmasına örnek

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



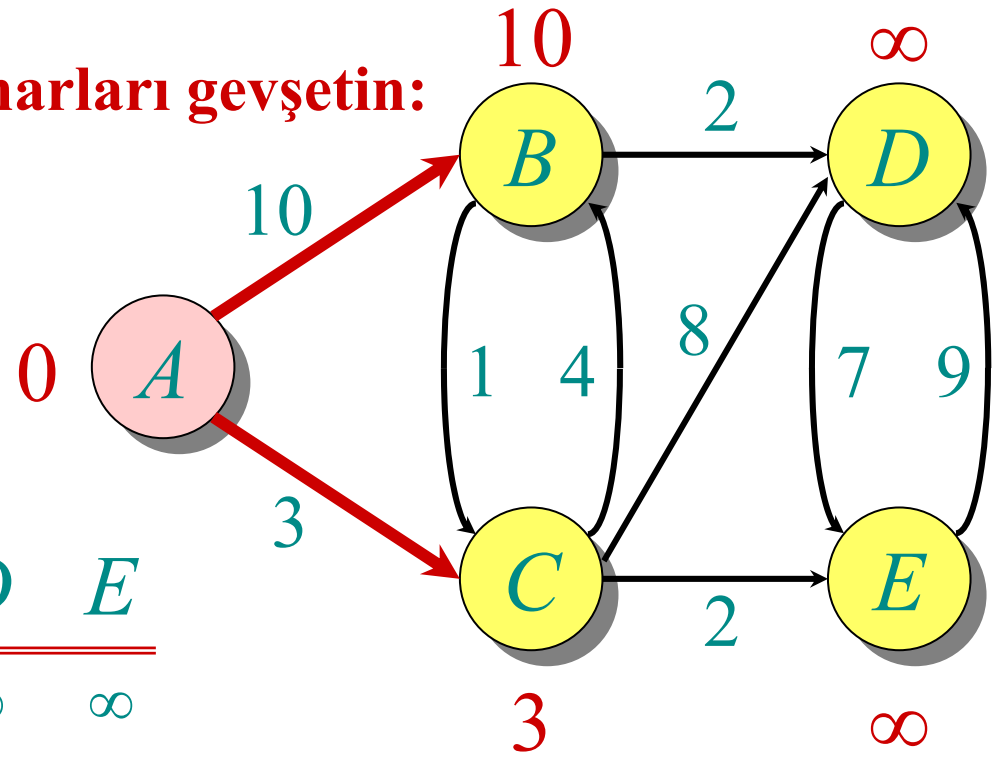
Q:	*A*	*B*	*C*	*D*	*E*
	0	∞	∞	∞	∞

S: { *A* }



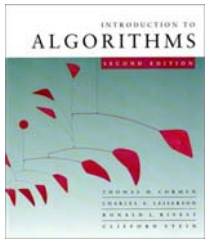
Dijkstra algoritmasına örnek

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



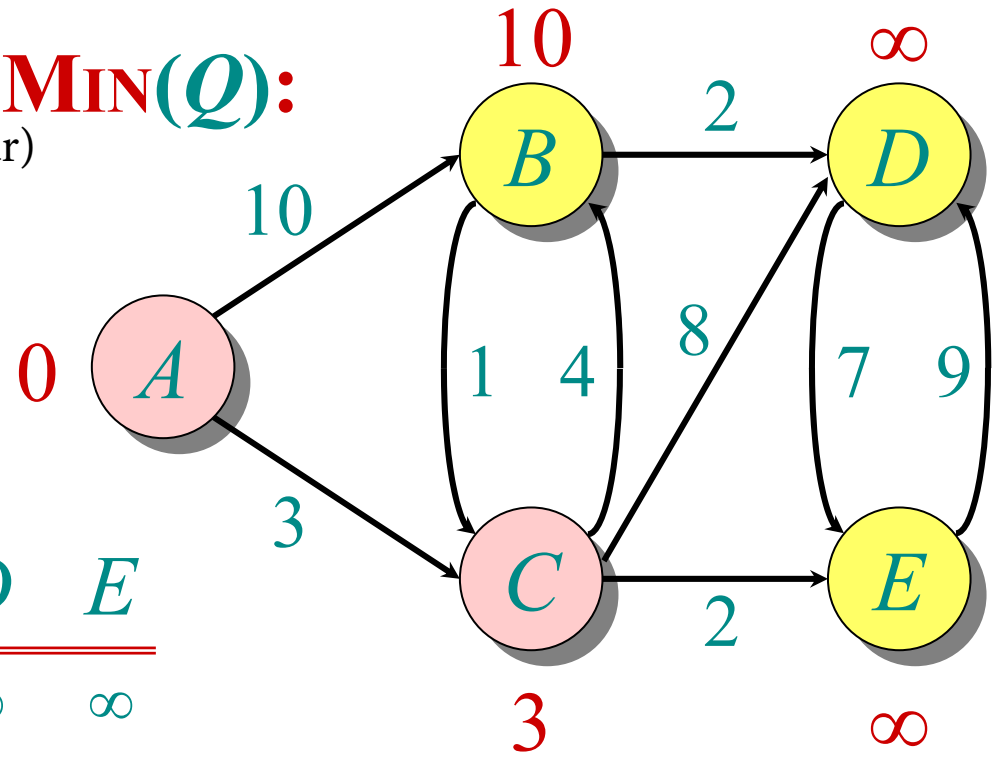
Q:	*A*	*B*	*C*	*D*	*E*
	0	∞	∞	∞	∞
		10	3	∞	∞

S: { *A* }



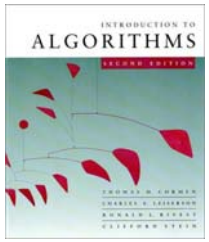
Dijkstra algoritmasına örnek

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



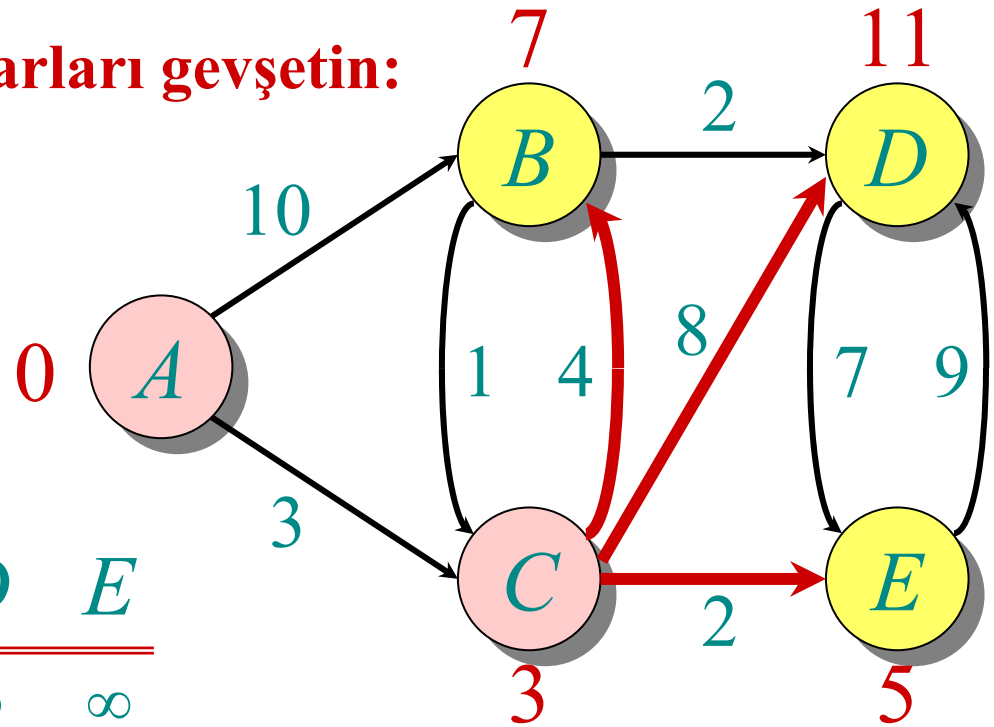
Q:	*A*	*B*	*C*	*D*	*E*
	0	∞	∞	∞	∞
		10	3	∞	∞

S: { *A*, *C* }



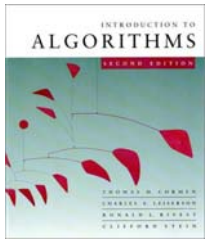
Dijkstra algoritmasına örnek

C'den ayrılan tüm kenarları gevşetin:



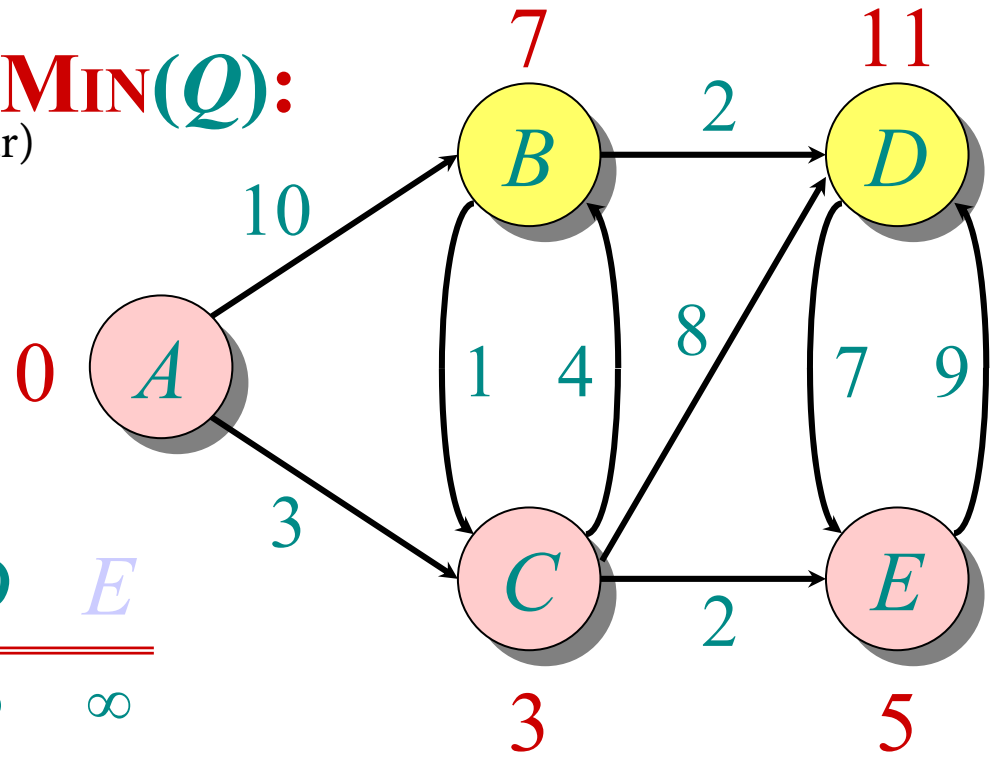
Q:	*A*	*B*	*C*	*D*	*E*
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5

S: { *A*, *C* }



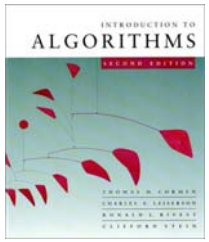
Dijkstra algoritmasına örnek

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



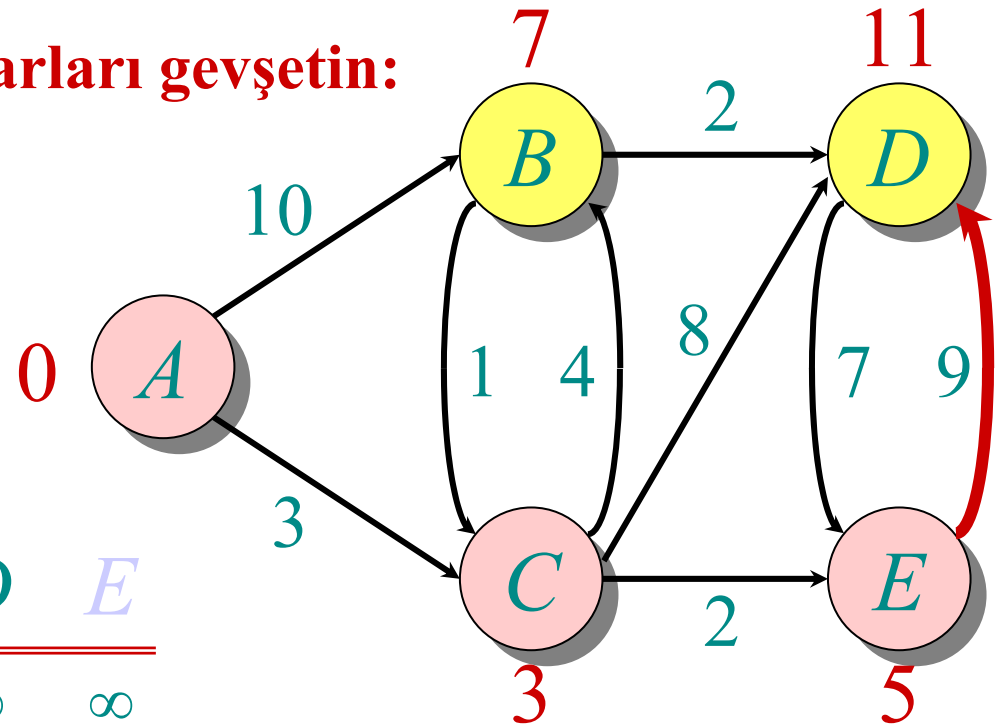
Q:	*A*	*B*	*C*	*D*	*E*
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5

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



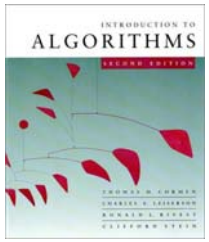
Dijkstra algoritmasına örnek

E'den ayrılan tüm kenarları gevşetin:



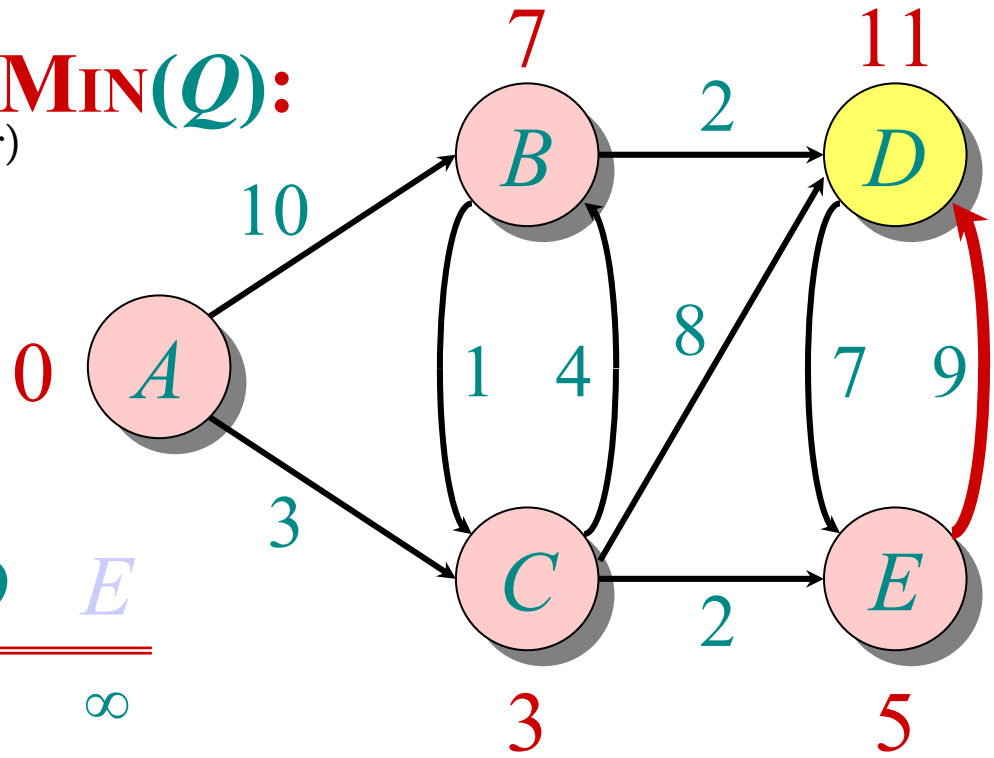
Q:	*A*	*B*	*C*	*D*	*E*
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5
		7		11	

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



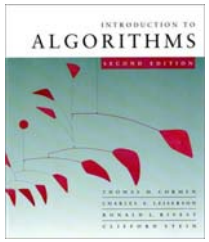
Dijkstra algoritmasına örnek

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



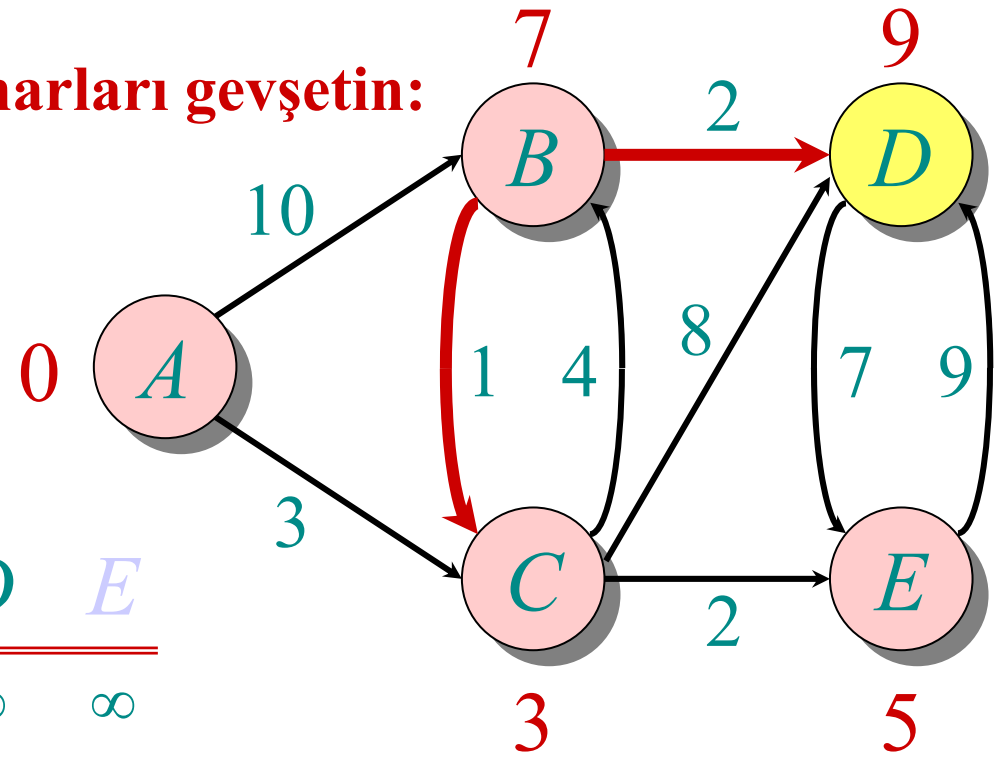
Q:	*A*	*B*	*C*	*D*	*E*
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5
		7		11	

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



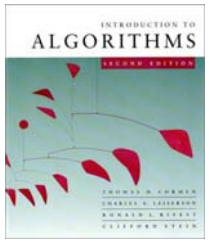
Dijkstra algoritmasına örnek

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



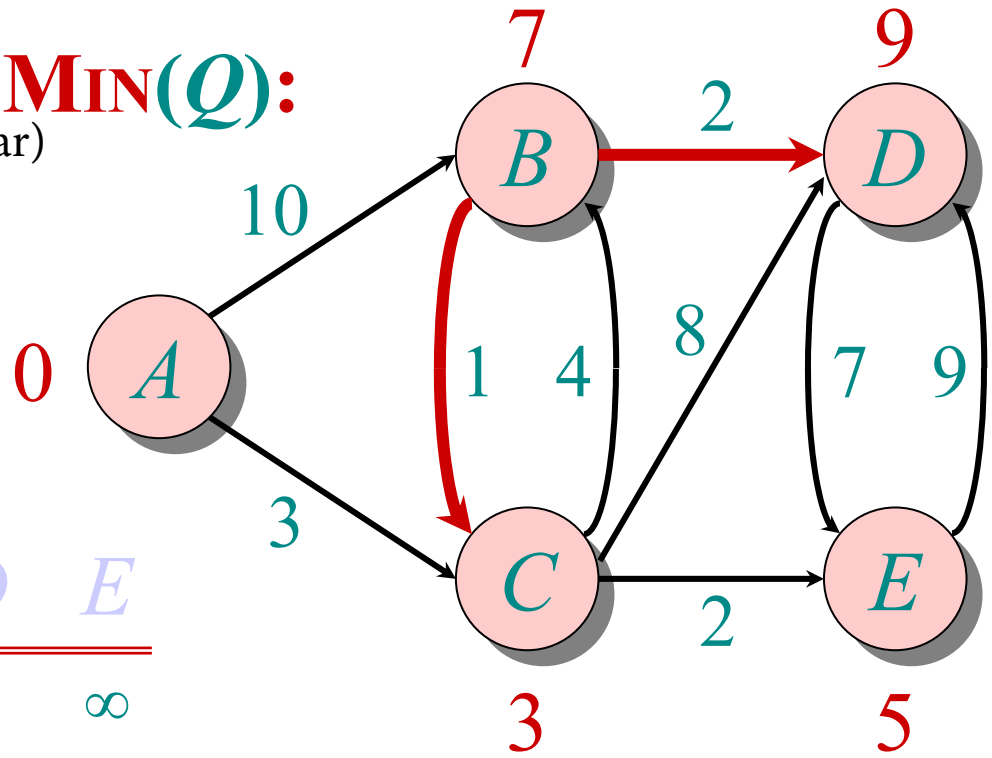
Q:	*A*	*B*	*C*	*D*	*E*
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5
		7		11	
				9	

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



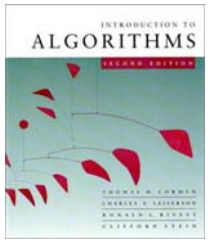
Dijkstra algoritmasına örnek

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



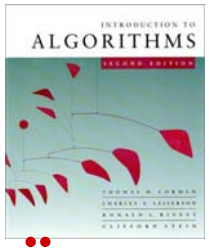
Q:	*A*	*B*	*C*	*D*	*E*
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5
		7		11	
				9	

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



Doğruluk —Bölüm I

Ön kuram. $d[s] \leftarrow 0$ 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 adımlarında korunur.



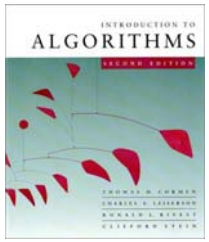
Doğruluk —Bölüm II

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,

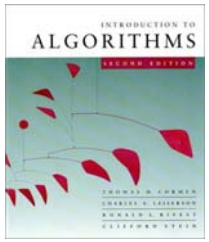
$$
\begin{aligned}
d[v] &< \delta(s, v) && \text{kabul} \\
&\leq \delta(s, u) + \delta(u, v) && \text{üçgen eşitsizliği} \\
&\leq \delta(s,u) + w(u, v) && \text{kısa yol} \leq \text{özel yol} \\
&\leq d[u] + w(u, v) && v \text{ ilk ihlal}
\end{aligned}
$$

Çelişki. □



Doğruluk —Bölüm II

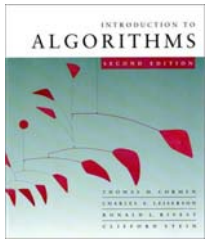
Ön kuram. u, s' den v' ye en kısa yolda v'nin atası olsun. O durumda, eğer $d[u] = \delta(s, u)$ ve kenar (u, v)gevşetilmişse, gevşemeden sonra elimizde $d[v] = \delta(s, v)$ olur.



Doğruluk —Bölüm II

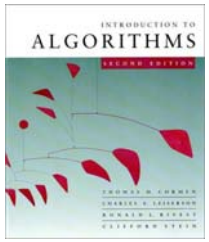
Ön kuram. u, s' den v' ye en kısa yolda v' nin atası olsun. O durumda, eğer $d[u] = \delta(s, u)$ve kenar (u, v) gevşetilmişse, gevşemeden sonra elimizde $d[v] = \delta(s, v)$ olur.

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.



Doğruluk —Bölüm III

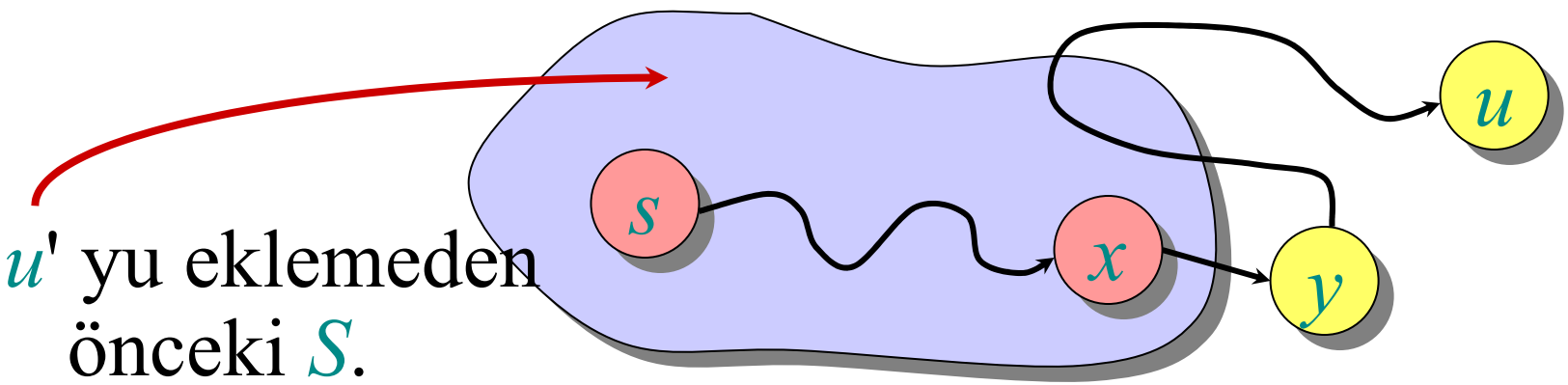
Teorem. Dijkstra algoritması tüm $v \in V$ için $d[v] = \delta(s, v)$ ile sonlanır.



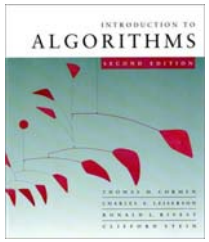
Doğruluk —Bölüm III

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

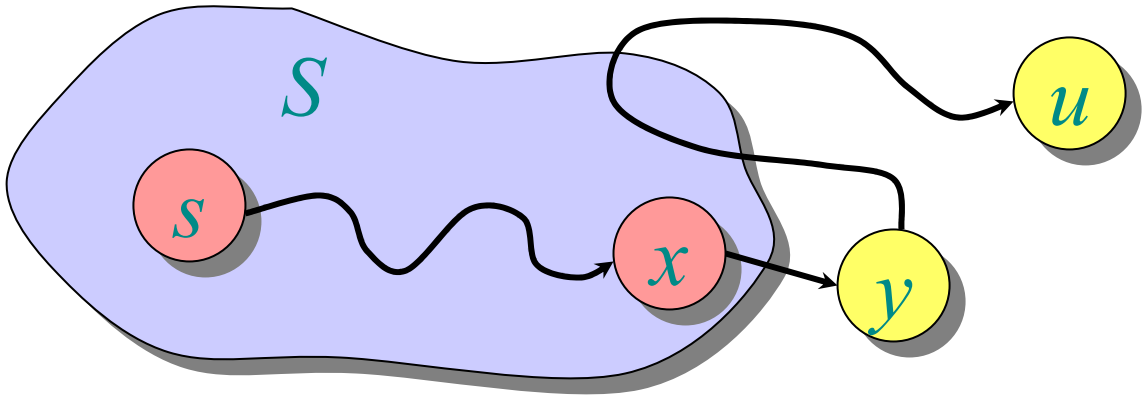
Kanıt. v, S' ye eklenirken, her $v \in V$ için $d[v] = \delta(s, v)$ olduğunu göstermek yeterlidir. Her $d[u] > \delta(s, u)$ için u' nun S' ye eklenen ilk köşe olduğunu düşünün. y, s' den u' ya en kısa yol boyunca $V - S$ de ilk köşe olsun, x de onun atası olsun:



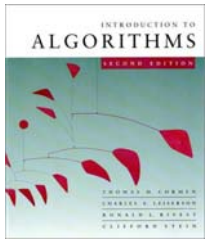
u' yu eklemeden önceki S.



Doğruluk —Bölüm III(devamı)



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. □



Dijkstra' nın çözümlemesi

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

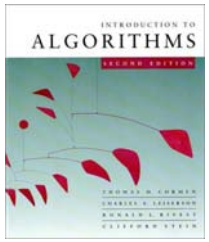
(yap) **do** $u \leftarrow$ EXTRACT-MIN(Q) (en azı çıkar)

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

(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' nın çözümlemesi

$|V|$ kere

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

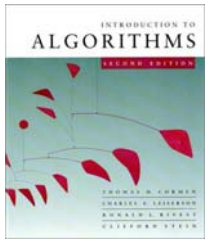
(yap) **do** $u \leftarrow$ EXTRACT-MIN(Q) (en azı çıkar)

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

(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' nın çözümlemesi

$|V|$ kere

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

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

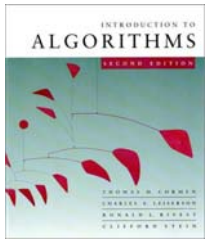
$S \leftarrow S \cup \{u\}$

(u)derecesi kere

(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' nın çözümlemesi

$|V|$ kere

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

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

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

(u)derecesi kere

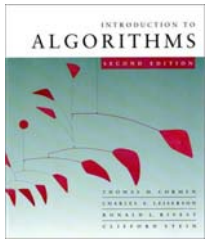
for each $v \in Adj[u]$ (için)

(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)



Dijkstra' nın çözümlemesi

$|V|$ kere

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

(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)

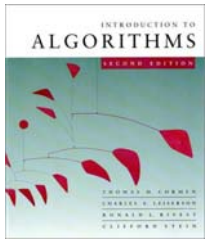
(sonra) **then** $d[v] \leftarrow d[u] + w(u, v)$

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.

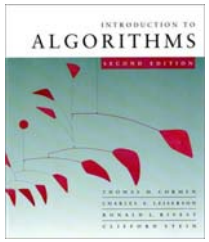


Dijkstra' nın çözümlemesi (devamı)

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

(en azı çıkar) (azaltılmış anahtar)

Q	$T_{\text{EXTRACT-MIN}}$	$T_{\text{DECREASE-KEY}}$	Toplam

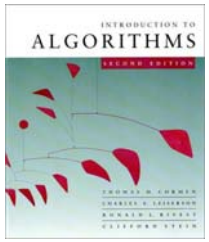


Dijkstra' nın çözümlemesi (devamı)

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

(en azı çıkar) (azaltılmış anahtar)

Q	$T_{\text{EXTRACT-MIN}}$	$T_{\text{DECREASE-KEY}}$	Toplam
dizilim	$O(V)$	$O(1)$	$O(V^2)$

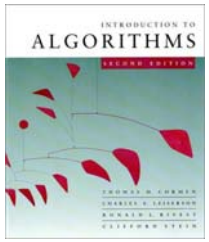


Dijkstra' nın çözümlemesi (devamı)

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

(en azı çıkar) (azaltılmış anahtar)

Q	$T_{\text{EXTRACT-MIN}}$	$T_{\text{DECREASE-KEY}}$	Toplam
dizilim	$O(V)$	$O(1)$	$O(V^2)$
ikili yığın	$O(\lg V)$	$O(\lg V)$	$O(E \lg V)$

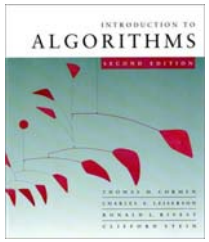


Dijkstra' nın çözümlemesi (devamı)

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

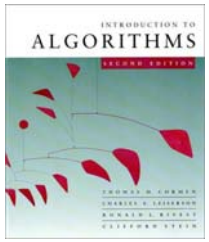
(en azı çıkar) (azaltılmış anahtar)

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



Ağırlıklandırılmamış grafikler

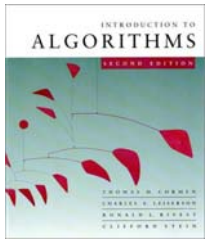
Tüm $(u, v) \in E$'ler için $w(u, v) = 1$ olduğunu farzedin. Dijkstra algoritması geliştirilebilir mi?



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.



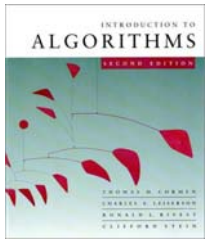
Ağırlıklandırılmamış grafikler

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** $Q \neq \varnothing$
  (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)



Ağırlıklandırılmamış grafikler

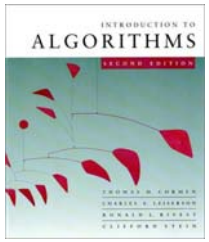
Tüm $(u, v) \in E$'ler için $w(u, v) = 1$ olduğunu farzedin. Dijkstra algoritması geliştirilebilir mi?

- Bir öncelik sırası yerine basit FIFO sırası kullanın.

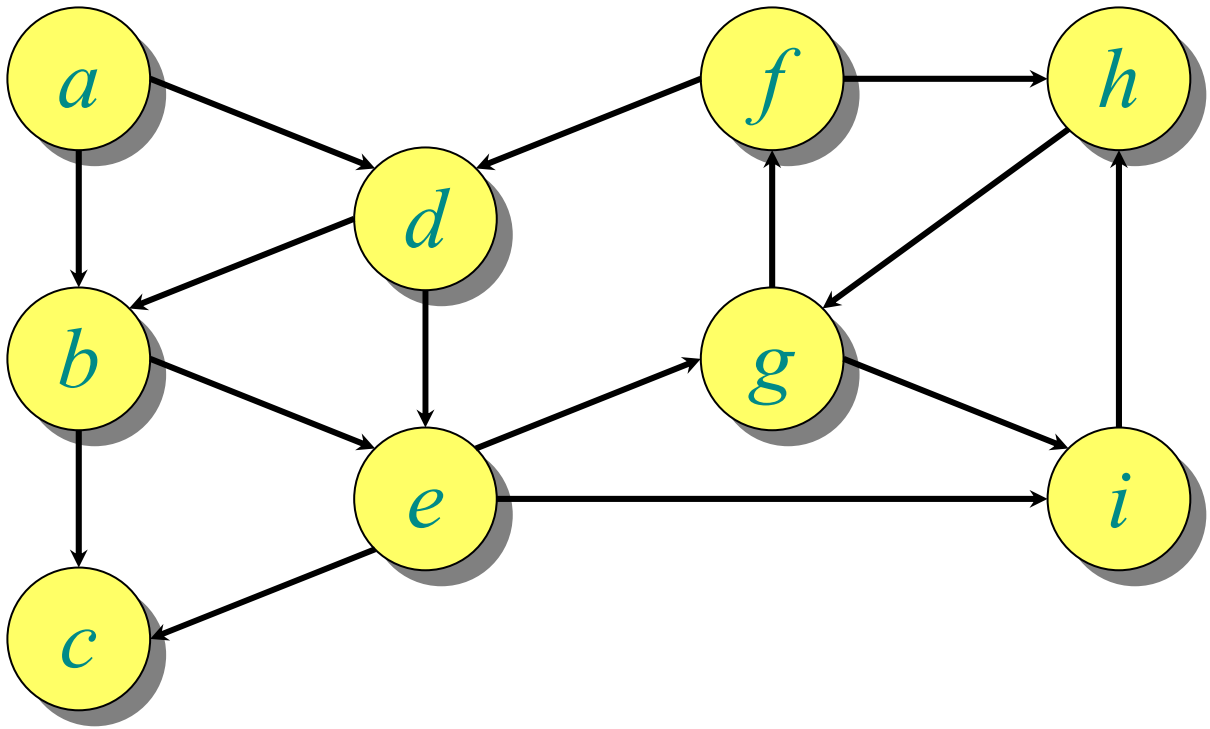
Sığ öncelikli arama

(-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)

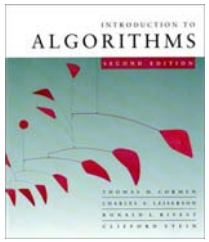
Çözümleme: Time(süre)$= O(V + E)$.



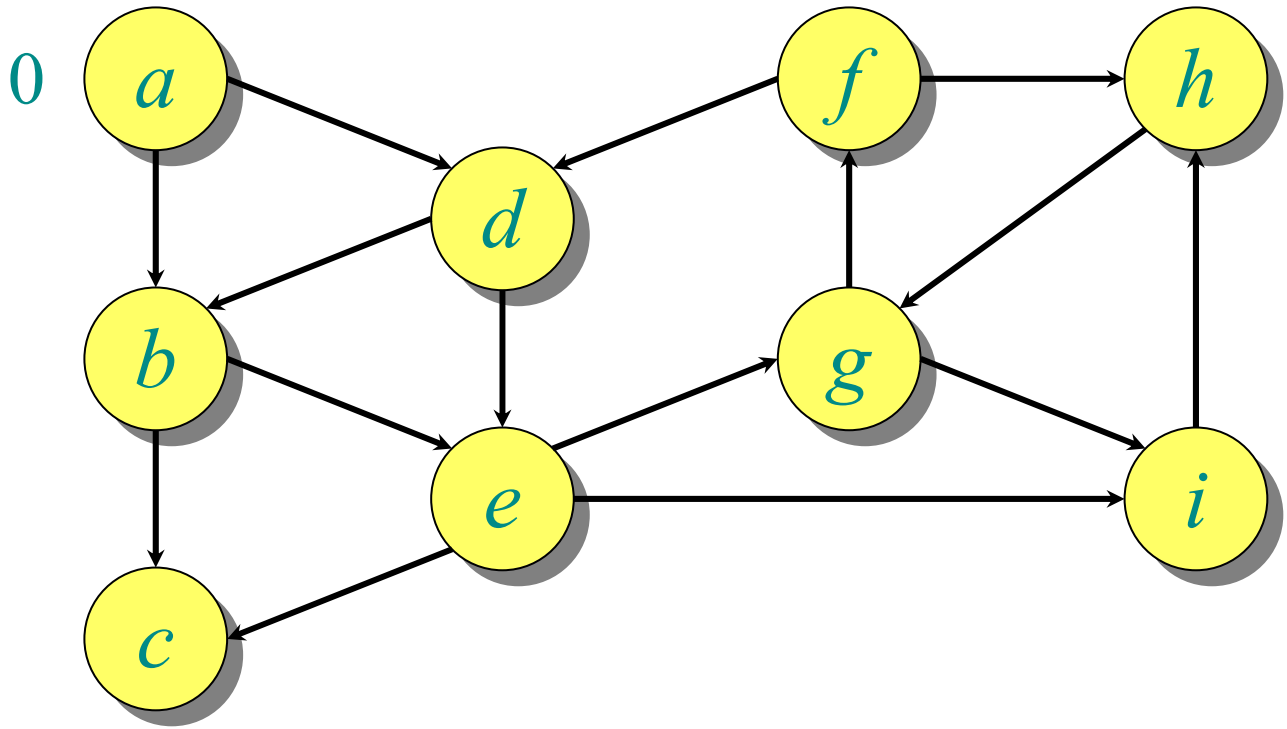
Enine arama için örnek



Q:

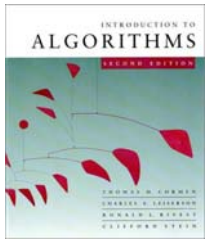


Enine arama için örnek

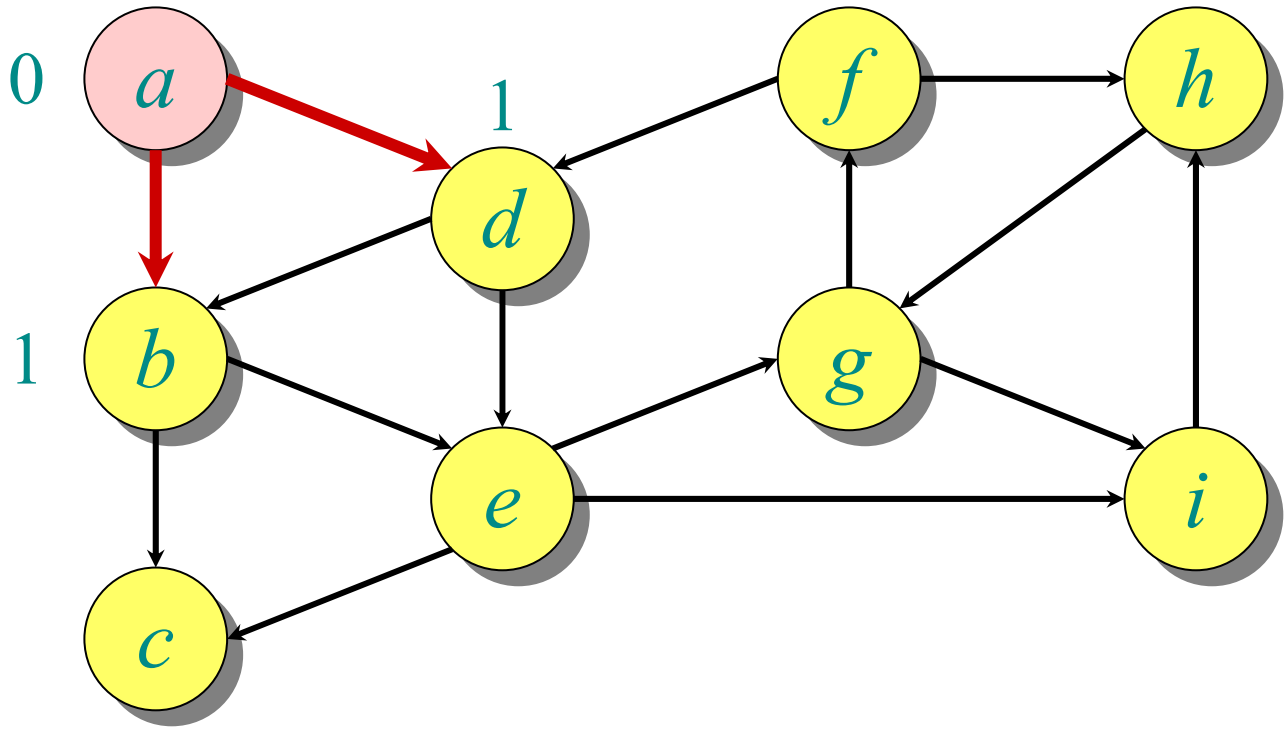


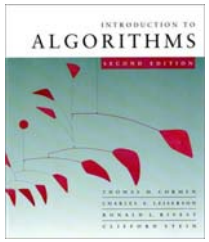
0

Q: a

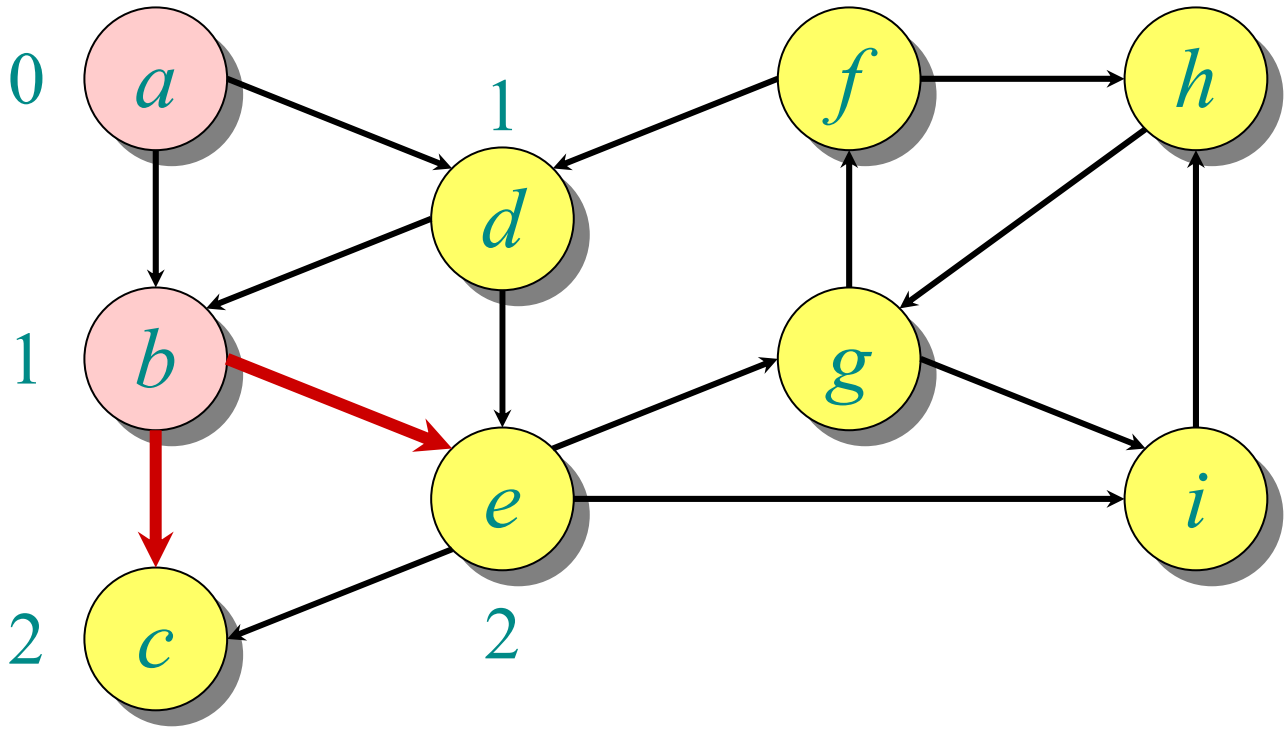


Enine arama için örnek



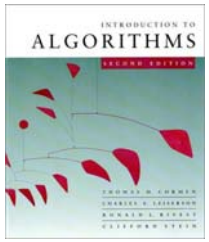


Enine arama için örnek

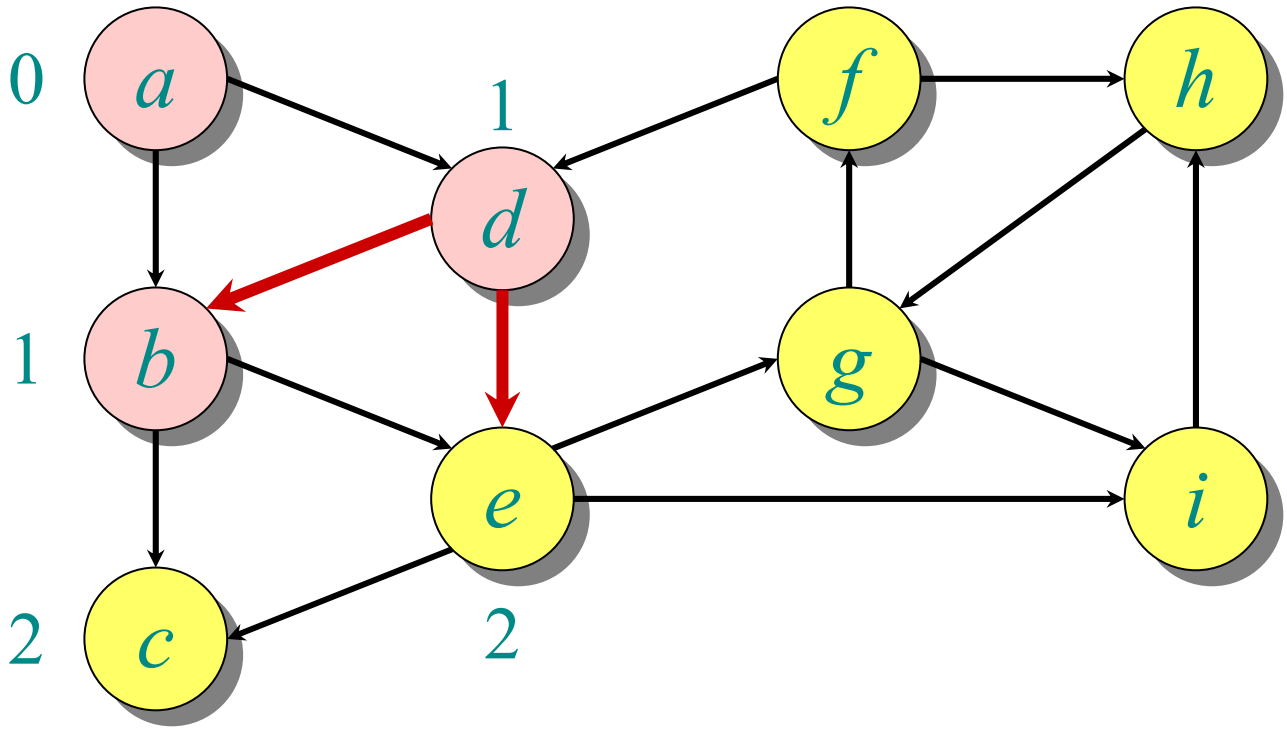


1 2 2

Q: a b d c e

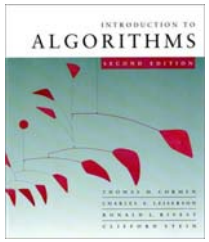


Enine arama için örnek

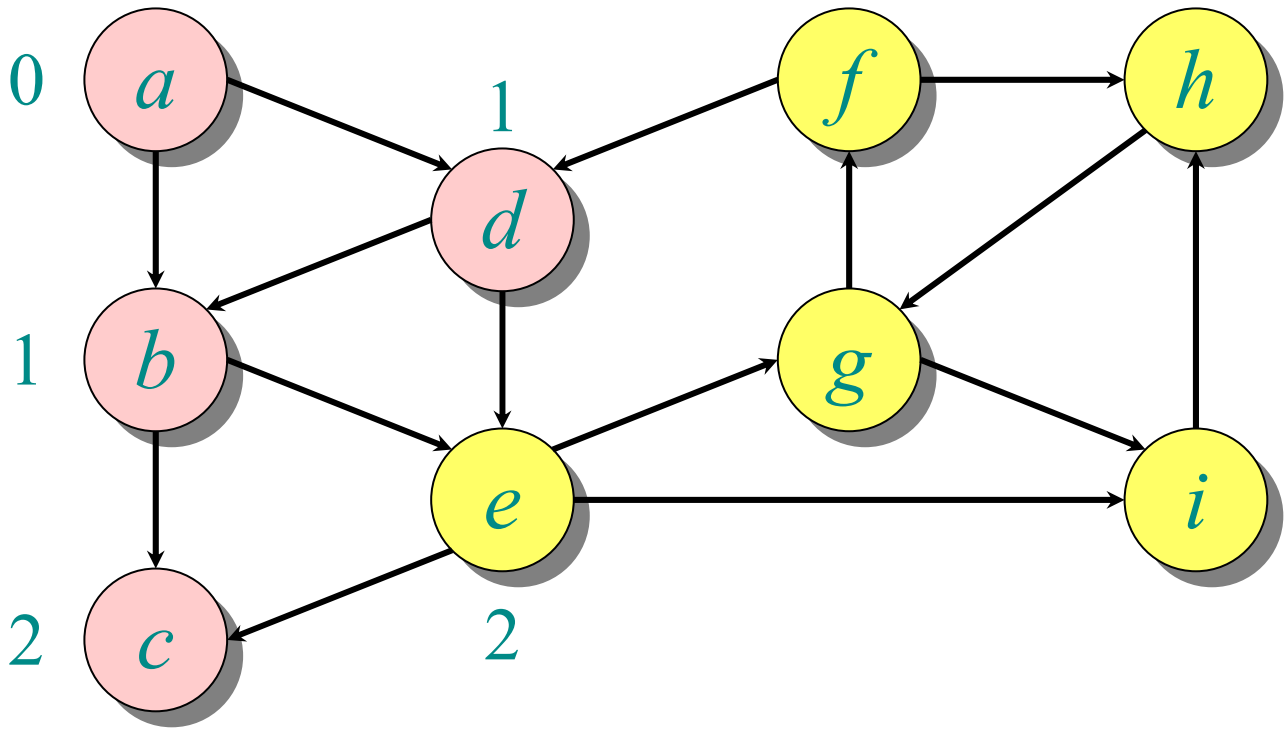


2 2

Q: a b d c e

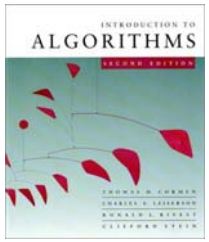


Enine arama için örnek

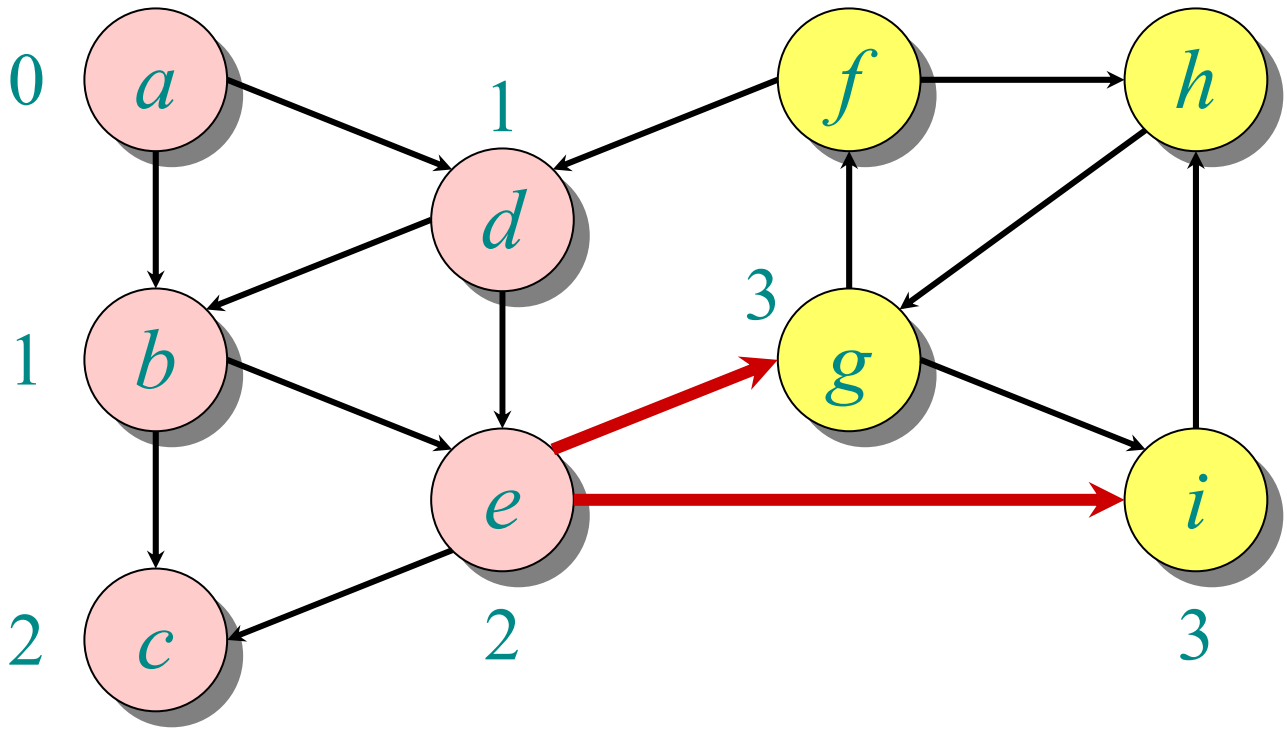


2

Q: a b d c e

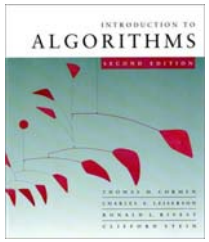


Enine arama için örnek

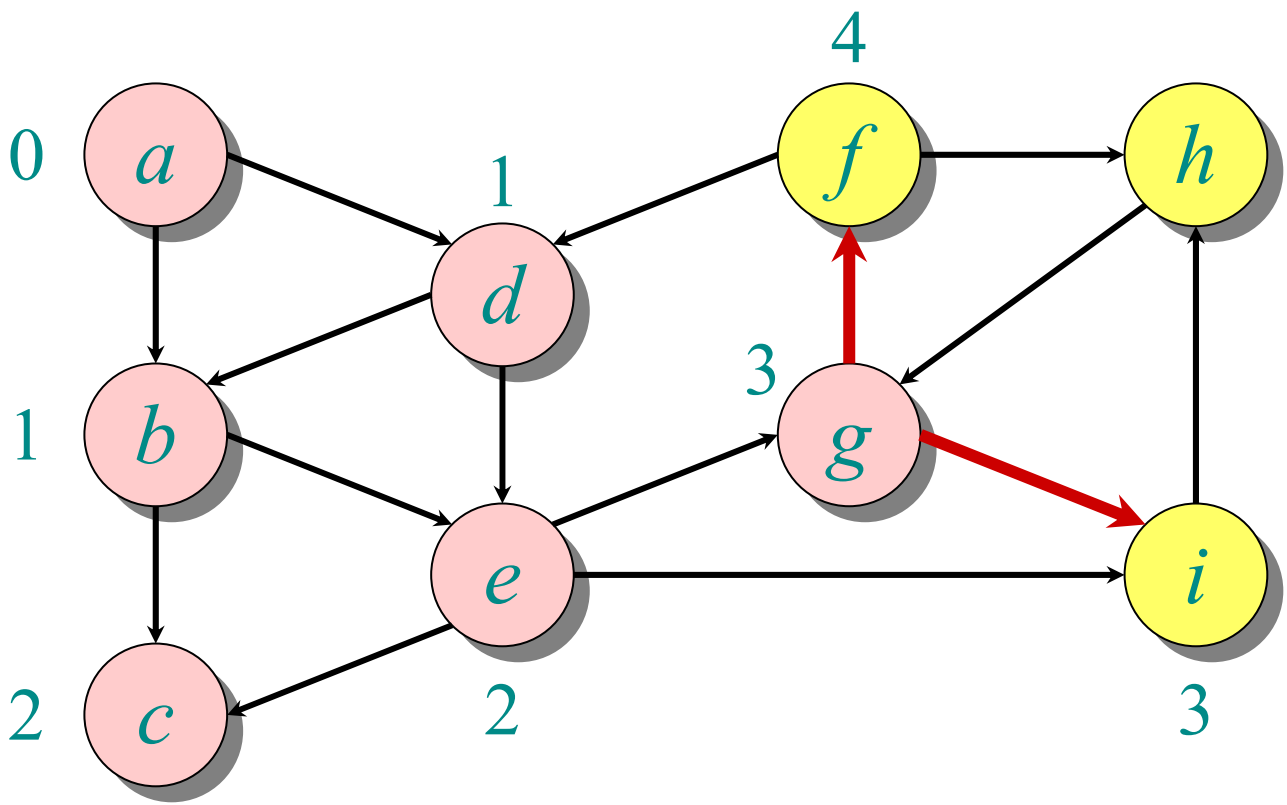


3 3

Q: a b d c e g i

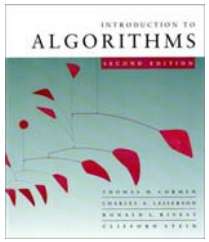


Enine arama için örnek

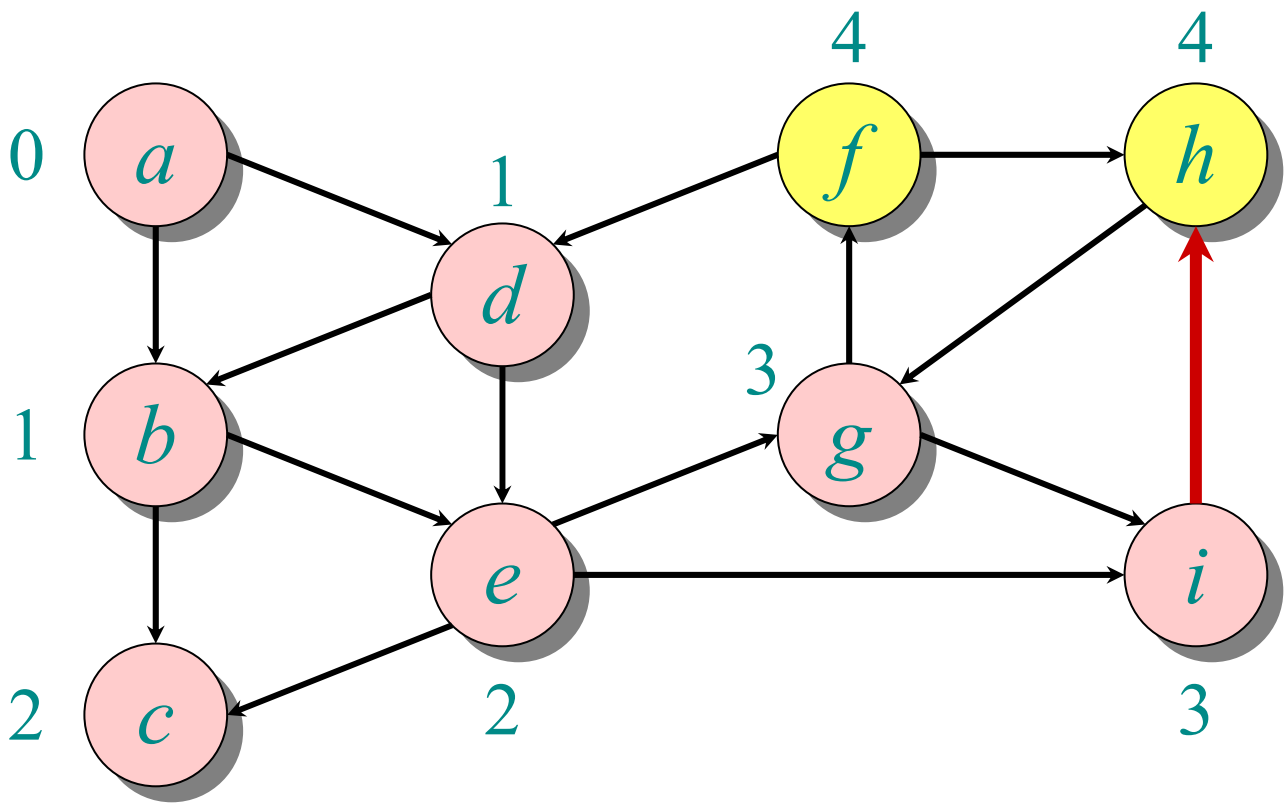


3 4

Q: *a b d c e g i f*

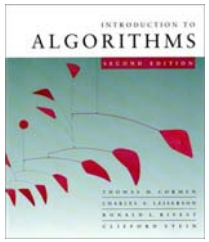


Enine arama için örnek

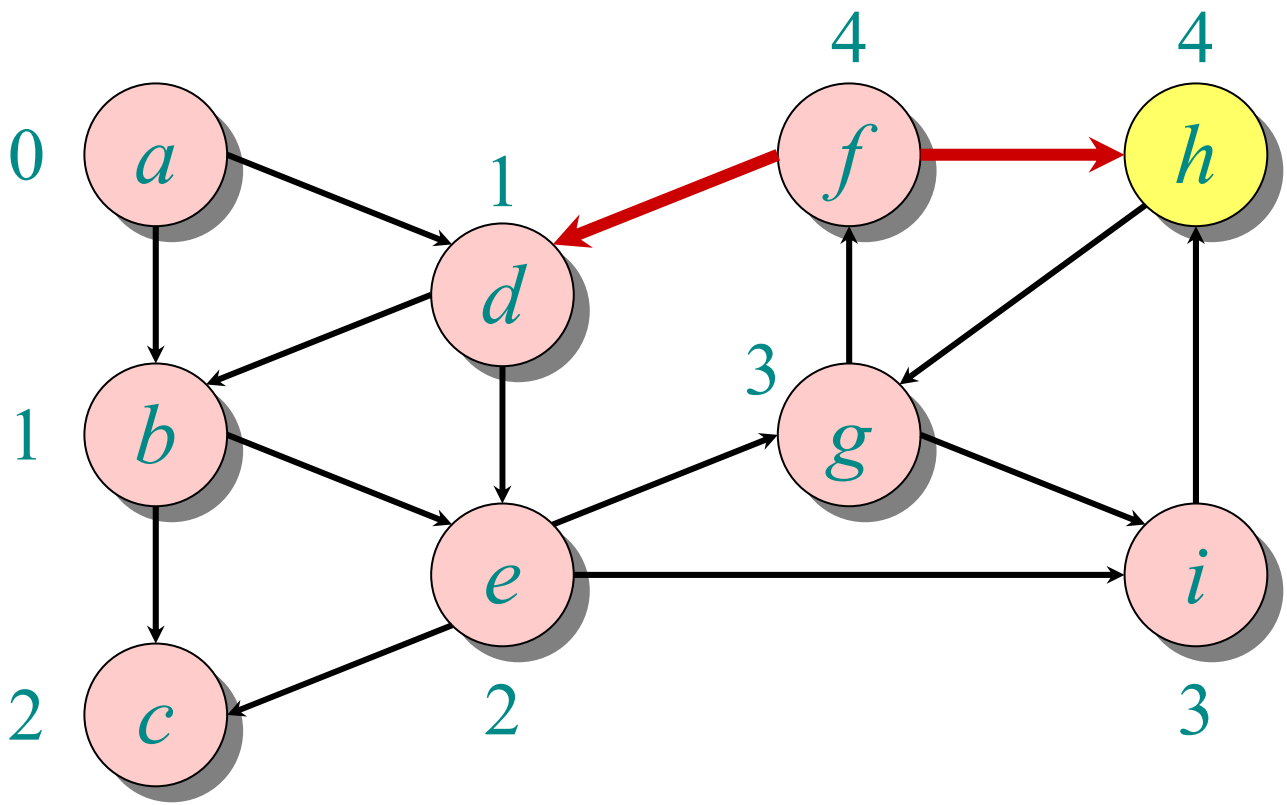


4 4

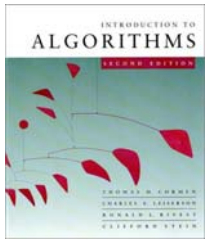
Q: a b d c e g i f h



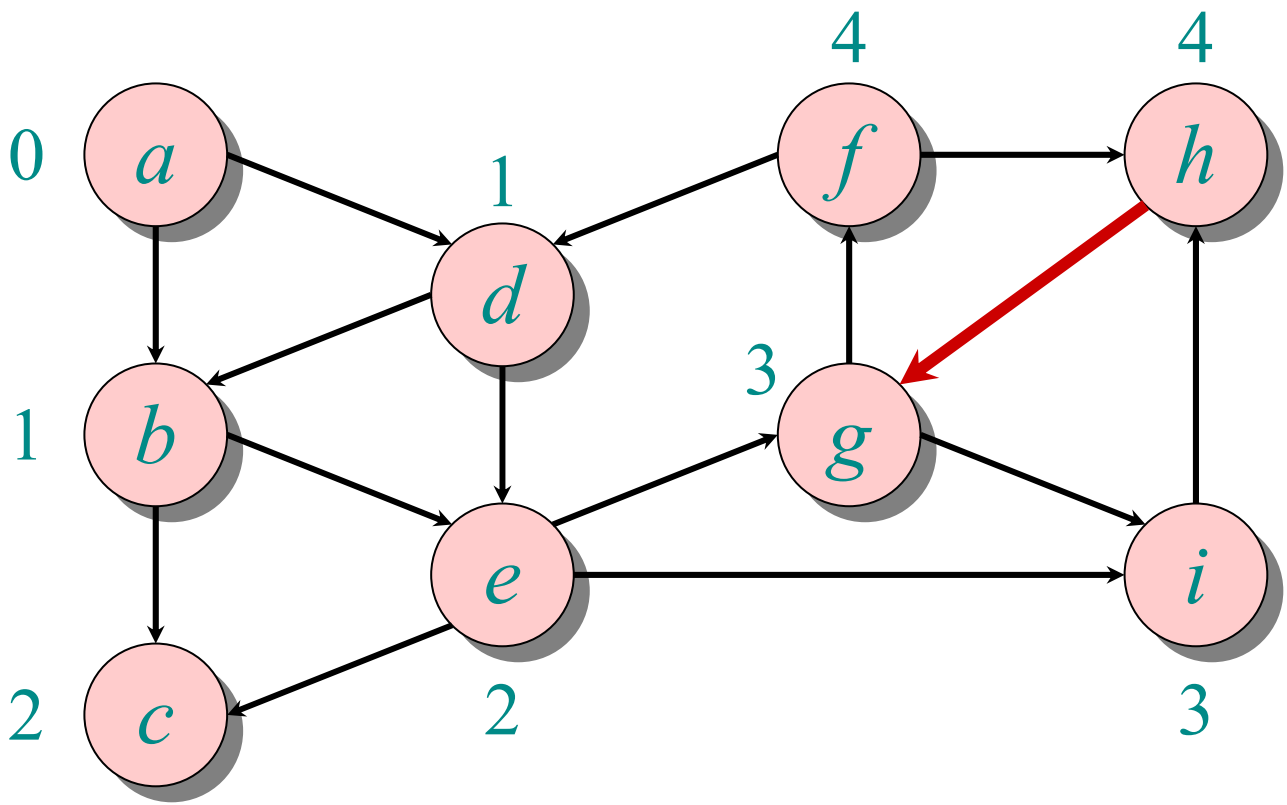
Enine arama için örnek



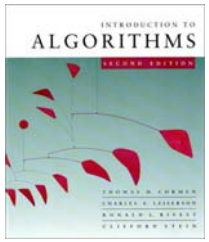
Q: a b d c e g i f h



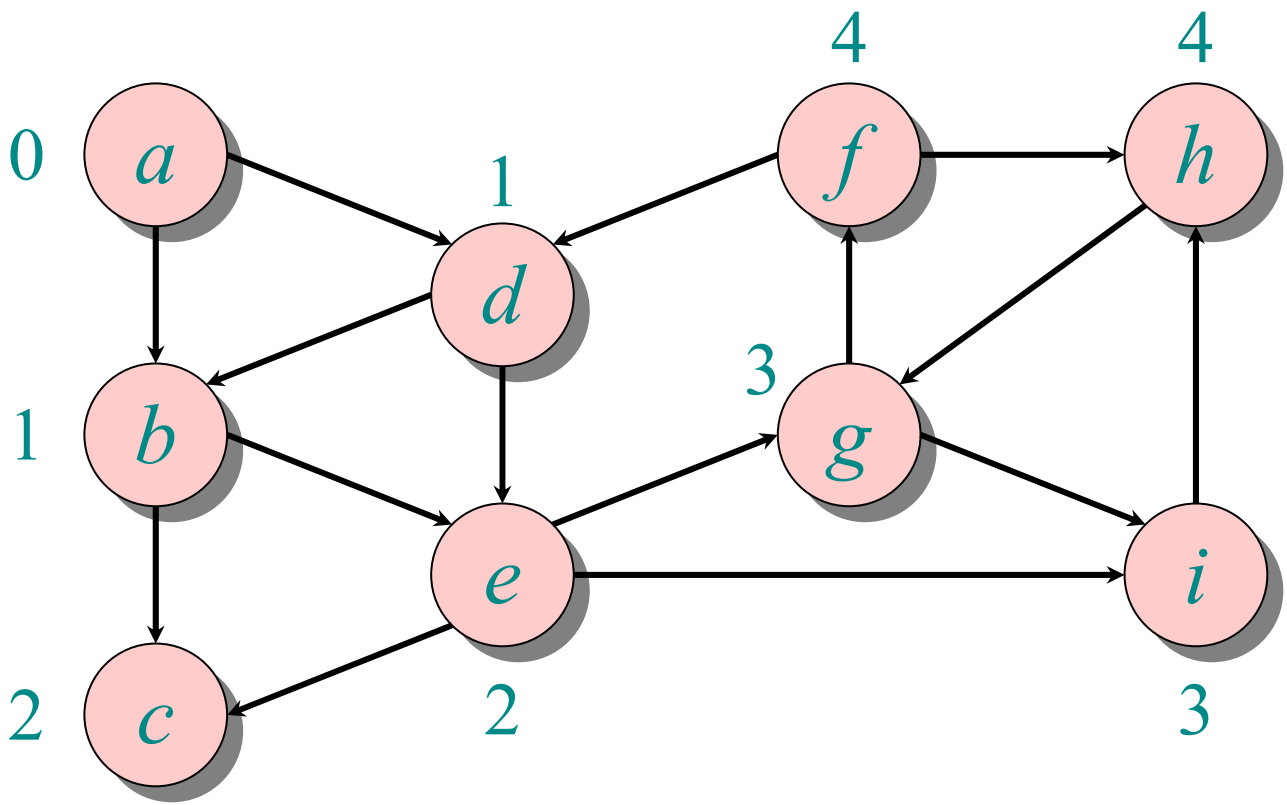
Enine arama için örnek



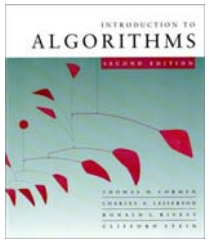
Q: *a b d c e g i f h*



Enine arama için örnek



Q: a b d c e g i f h



(BFS)Enine arama'nın doğruluğu

(-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)

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.

