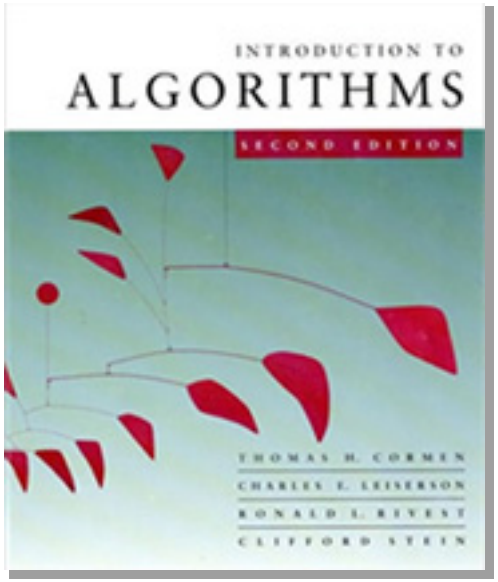
Algoritmalara Giriş
6.046J/18.401J

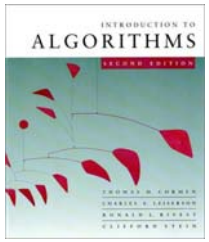


Ders 16

Açgözlü Algoritmalar (ve Grafikler)

- Grafik gösterim
- Minimum kapsayan ağaç
- Optimal altyapı
- Açgözlü seçim
- Prim'in açgözlü MST algoritması

Prof. Charles E. Leiserson



Grafikler (tekrar)

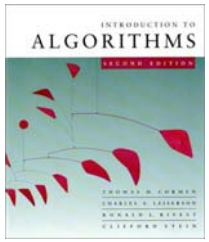
Tanım. ***Yönlendirilmiş grafik (digraf)*** $G = (V, E)$,

- V köşeler kümesi ve
- $E \subseteq V \times V$ kenarlar kümelerinden oluşur.

Yönlendirilmemiş grafik $G = (V, E)$'de, E kenar kümesi, sıralı olmayan köşe çiftlerinden oluşur.

Her durumda, elimizde $|E| = O(V^2)$ vardır.
Ayrıca, eğer G bağlantılıysa, $|E| \geq |V| - 1$, ki bu da $\lg|E| = \Theta(\lg V)$ anlamına gelir.

(Bakınız CLRS, Appendix B.)

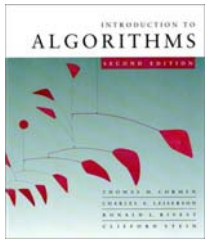


Komşuluk matrisi gösterimi

Bir $G = (V, E)$ grafiğinin *komşuluk matrisi* $V = \{1, 2, \ldots, n\}$ olduğunda,

$$A[i, j] = \begin{cases} 1 \text{ eğer } (i, j) \in \text{E ise,} \\ 0 \text{ eğer} (i, j) \notin \text{E ise,} \end{cases}$$

$A[1 \ldots n, 1 \ldots n]$ matrisidir.

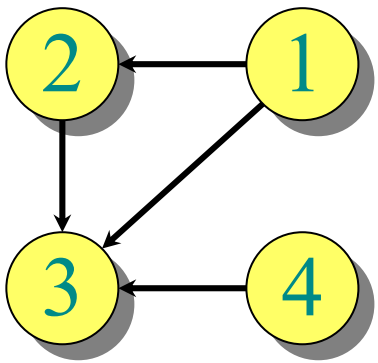


Komşuluk matrisi gösterimi

Bir $G = (V, E)$ grafiğinin *komşuluk matrisi* $V = \{1, 2, \ldots, n\}$ iken,

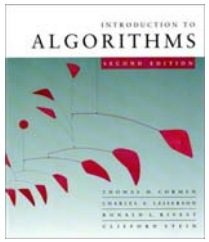
$$A[i,j] = \begin{cases} 1 \text{ eğer } (i,j) \in E \text{ ise,} \\ 0 \text{ eğer } (i,j) \notin E \text{ ise,} \end{cases}$$

$A[1 \ldots n, 1 \ldots n]$ matrisidir.



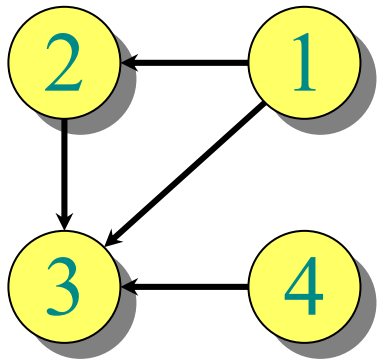
A	1	2	3	4
1	0	1	1	0
2	0	0	1	0
3	0	0	0	0
4	0	0	1	0

$\Theta(V^2)$ depolama:
$\Rightarrow$***yoğun*** gösterimi.

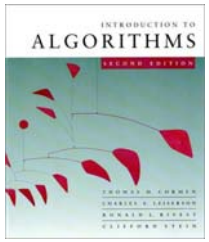


Komşuluk listesi gösterimi

Bir $v \in V$ köşesinin *komşuluk listesi*, v' ye komşu olan köşelerin $Adj[v]$ listesidir.

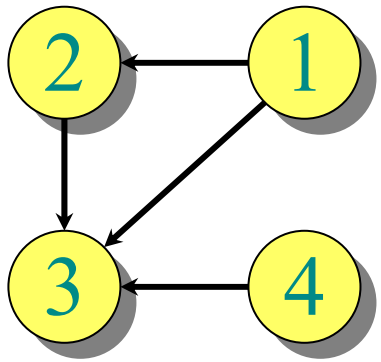


$Adj[1] = \{2, 3\}$
$Adj[2] = \{3\}$
$Adj[3] = \{\}$
$Adj[4] = \{3\}$



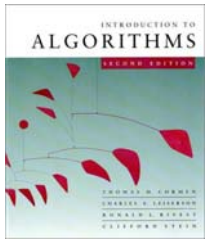
Komşuluk listesi gösterimi

Bir $v \in V$ köşesinin *komşuluk listesi,* v 'ye komşu olan köşelerin $Adj[v]$ listesidir.



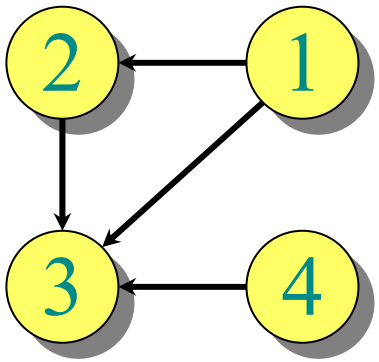
$Adj[1] = \{2, 3\}$
$Adj[2] = \{3\}$
$Adj[3] = \{\}$
$Adj[4] = \{3\}$

Yönlendirilmemiş grafikler için, $|Adj[v]| =$ *degree* (derece)(v).
Yönlendirilmiş grafikler için, $|Adj[v]| =$ *out-degree (dış-derece)*(v).



Komşuluk listesi gösterimi

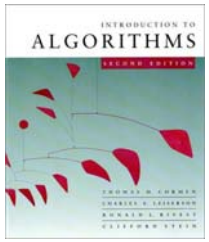
Bir $v \in V$ köşesinin *komşuluk listesi,* v′ ye komşu olan köşelerin $Adj[v]$ listesidir.



$Adj[1] = \{2, 3\}$
$Adj[2] = \{3\}$
$Adj[3] = \{\}$
$Adj[4] = \{3\}$

Yönlendirilmemiş grafikler için, $|Adj[v]| = degree$ (derece)(v).
Yönlendirilmiş grafikler için, $|Adj[v]| = out\text{-}degree$ *(dış-derece)*(v).

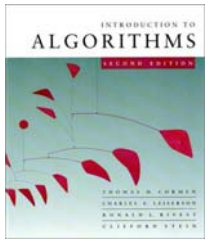
Tokalaşma önkuramı: Yönlendirilmemiş grafikler için $\sum_{v \in V} = 2|E \Rightarrow$ komşuluk listeleri, $\Theta(V + E)$ depolama alanını kullanır. Bu seyrek gösterimdir. (her tür grafik için.)



Minimum kapsayan ağaçlar

Girdi: $w : E \to \mathbb{R}$ ağırlık fonksiyonlu, $G = (V, E)$ bağlantılı, yönlendirilmemiş grafik.

- Basitlik adına, farzedin ki tüm kenar ağırlıkları farklı olsun. (CLRS genel durumu kapsar.)



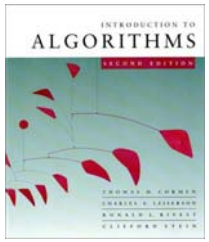
Minimum kapsayan ağaçlar

Girdi: $w : E \to \mathbb{R}$ ağırlık fonksiyonlu, $G = (V, E)$ bağlantılı, yönlendirilmemiş grafik.

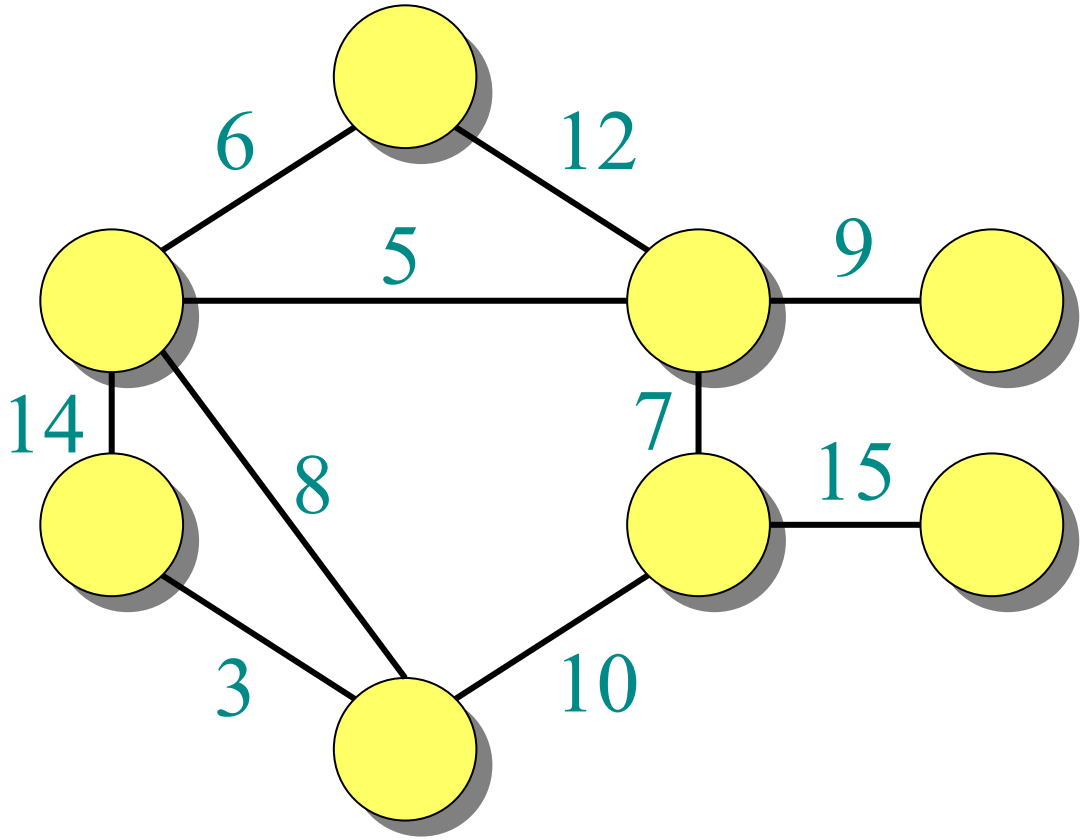
- Basitlik adına, farzedin ki tüm kenar ağırlıkları farklı olsun. (CLRS genel durumu kapsar.)

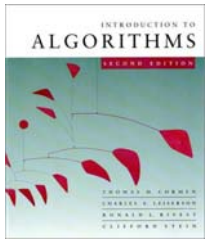
Çıktı: T kapsayan ağacı--en az ağırlıkla tüm köşeleri birleştiren bir ağaç:

$$w(T) = \sum_{(u,v)\in T} w(u,v) .$$

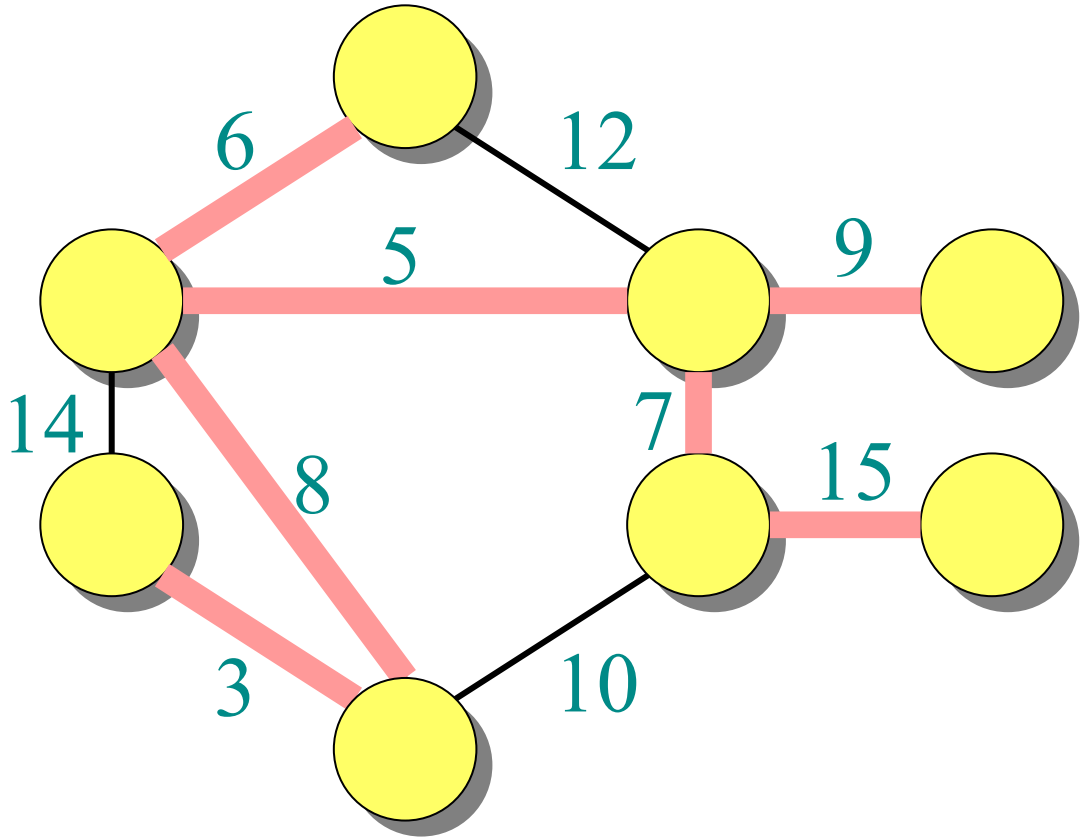


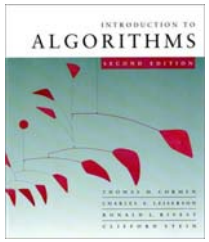
MST (minimum kapsayan ağaç) örneği





MST (minimum kapsayan ağaç) örneği

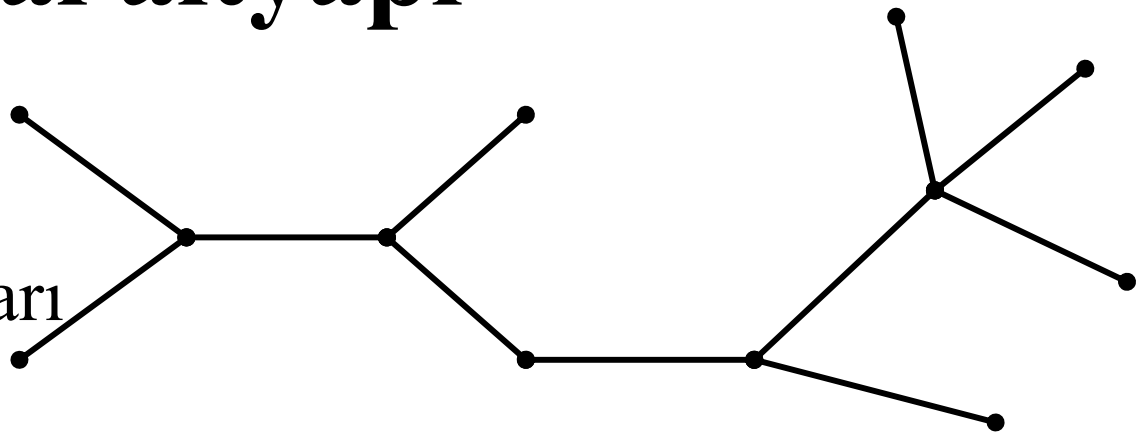


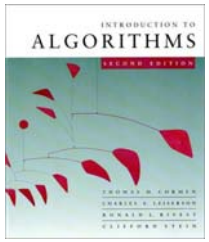


Optimal altyapı

MST T:

(G' nin diğer kenarları gösterilmemiştir.)

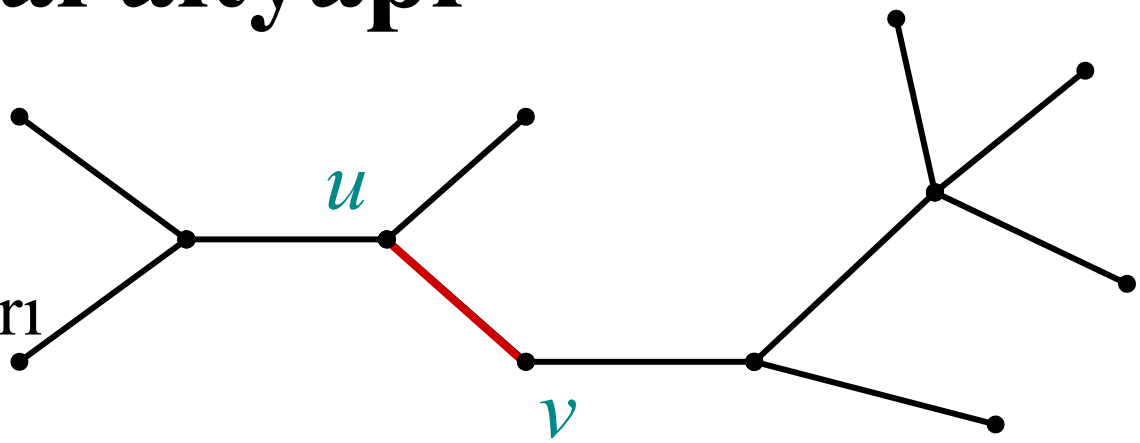




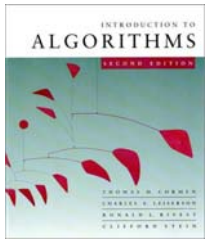
Optimal altyapı

MST T:

(G' nin diğer kenarları gösterilmemiştir.)



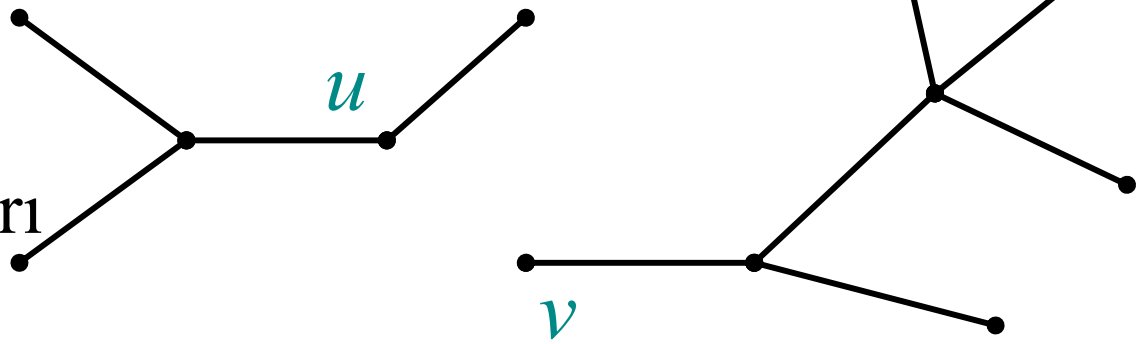
Herhangi bir $(u, v) \in T$ kenarını kaldır.



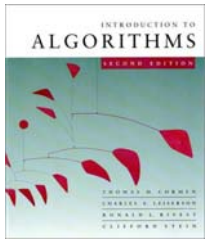
Optimal altyapı

MST T:

(G' nin diğer kenarları gösterilmemiştir.)



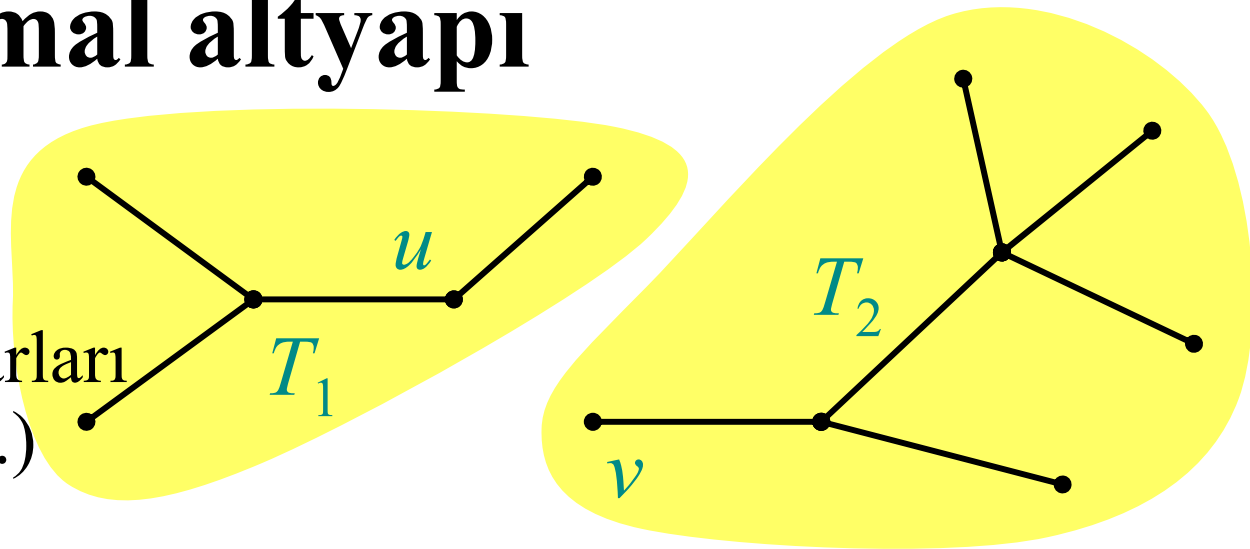
Herhangi bir $(u, v) \in T$ kenarını kaldır.



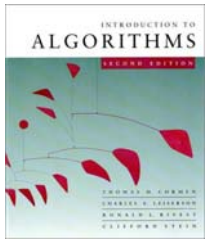
Optimal altyapı

MST T:

(G' nin diğer kenarları gösterilmemiştir.)



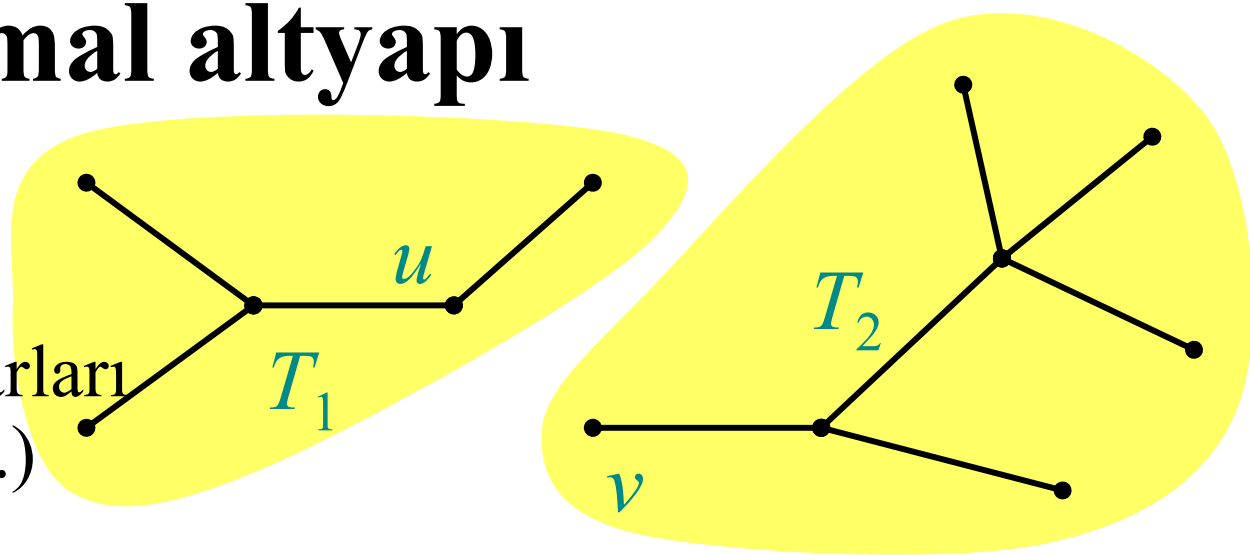
Herhangi bir $(u, v) \in T$ kenarını kaldır. Bu durumda T, T_1 ve T_2 olarak iki altağaca bölüntülenir.



Optimal altyapı

MST T:

(G' nin diğer kenarları gösterilmemiştir.)

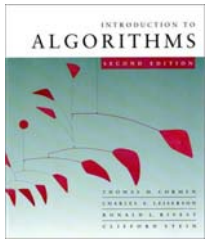


Herhangi bir $(u, v) \in T$ kenarını kaldırın.Bu durumda T, T_1 and T_2 olarak iki altağaca bölüntülenir.

Teorem. Altağaç T_1, $G_1 = (V_1, E_1)$ 'in bir MST'sidir ve T_1'in köşeleri tarafından oluşturulan G' nin altgrafiğidir:

$$V_1 = T_1\text{' in köşeleri}$$
$$E_1 = \{(x, y) \in E : x, y \in V_1\}.$$

T_2 için de bu böyledir.

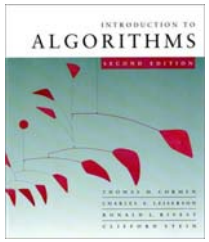


Optimal altyapının kanıtı

Kanıt. Kes ve yapıştır:

$$w(T) = w(u, v) + w(T_1) + w(T_2).$$

Eğer T_1' G_1 için T_1 den daha az ağırlıklı bir kapsayan ağaçsa, bu durumda $T' = \{(u, v)\} \cup T_1' \cup T_2$ G için T 'den daha az ağırlıklı bir kapsayan ağaç olur. □



Optimal altyapının kanıtı

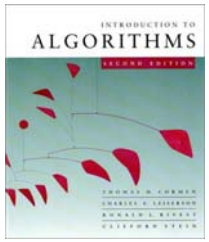
Kanıt. Kes ve yapıştır:

$$w(T) = w(u, v) + w(T_1) + w(T_2).$$

Eğer T_1' G_1 için T_1 den daha az ağırlıklı bir kapsayan ağaçsa, bu durumda $T' = \{(u, v)\} \cup T_1' \cup T_2$ G için T 'den daha az ağırlıklı bir kapsayan ağaç olur. □

Aynı zamanda çakışan altproblemlerimiz de var mı?

- Evet var.



Optimal altyapının kanıtı

Kanıt. Kes ve yapıştır

$$w(T) = w(u, v) + w(T_1) + w(T_2).$$

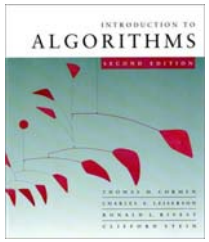
Eğer T_1' G_1 için T_1 den daha az ağırlıklı bir kapsayan ağaçsa, bu durumda $T' = \{(u, v)\} \cup T_1' \cup T_2$ G için T 'den daha az ağırlıklı bir kapsayan ağaç olur.

Aynı zamanda çakışan altproblemlerimiz de var mı?

- Evet var.

Harika, o zaman dinamik programlama çalışabilir!

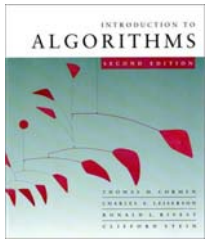
- Evet, fakat MST daha da verimli bir algoritmaya yol açan bir başka kuvvetli özelliğini sergiler.



Açgözlü algoritmalar için Kalite işareti

Açgözlü-seçim özelliği

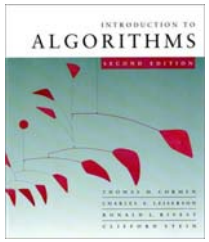
Yerel olarak en uygun seçim genel olarak da en uygundur.



Açgözlü algoritmalar için Kalite İşareti

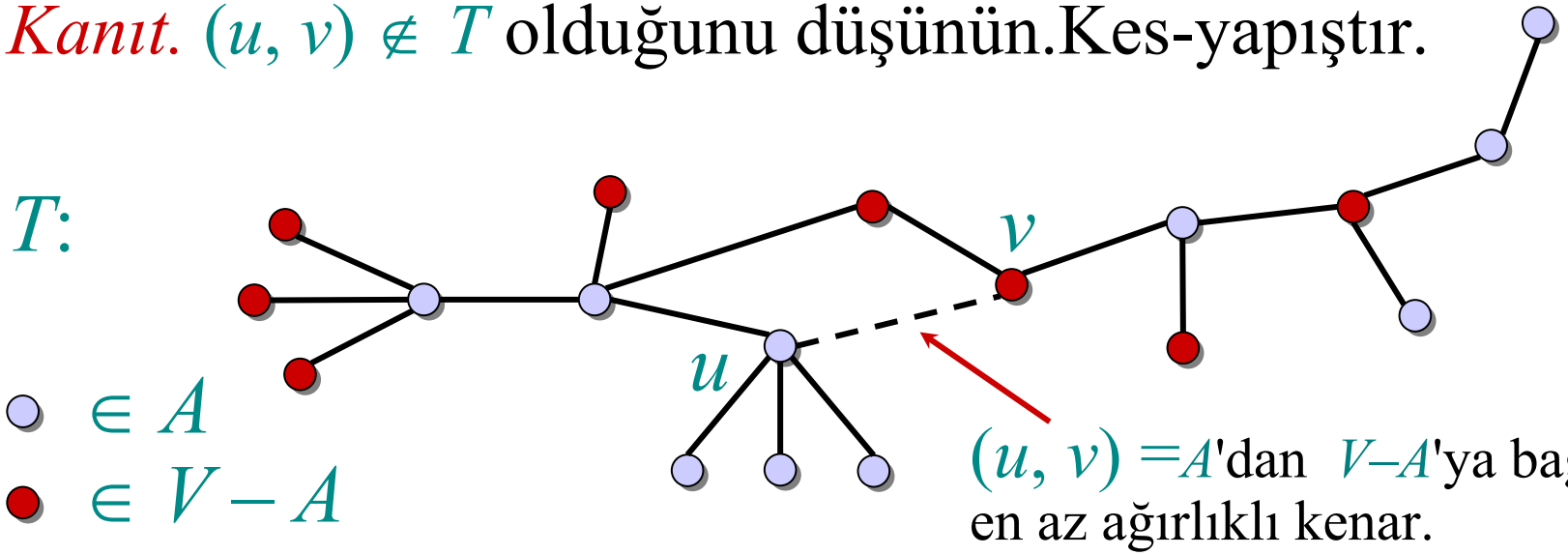
Açgözlü-seçim özelliği
Yerel olarak en uygun seçim genel olarak da en uygundur.

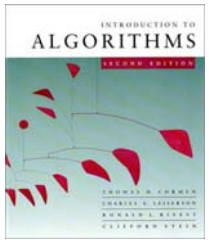
Teorem. T, $G = (V, E)$'nin MST'si olsun ve $A \subseteq V$ olsun. $(u, v) \in E$ 'nin A yı $V - A$'ya bağlayan en az ağırlıklı kenar olduğunu farzedin. O zaman $(u, v) \in T$ olur.



Teorem' in kanıtı

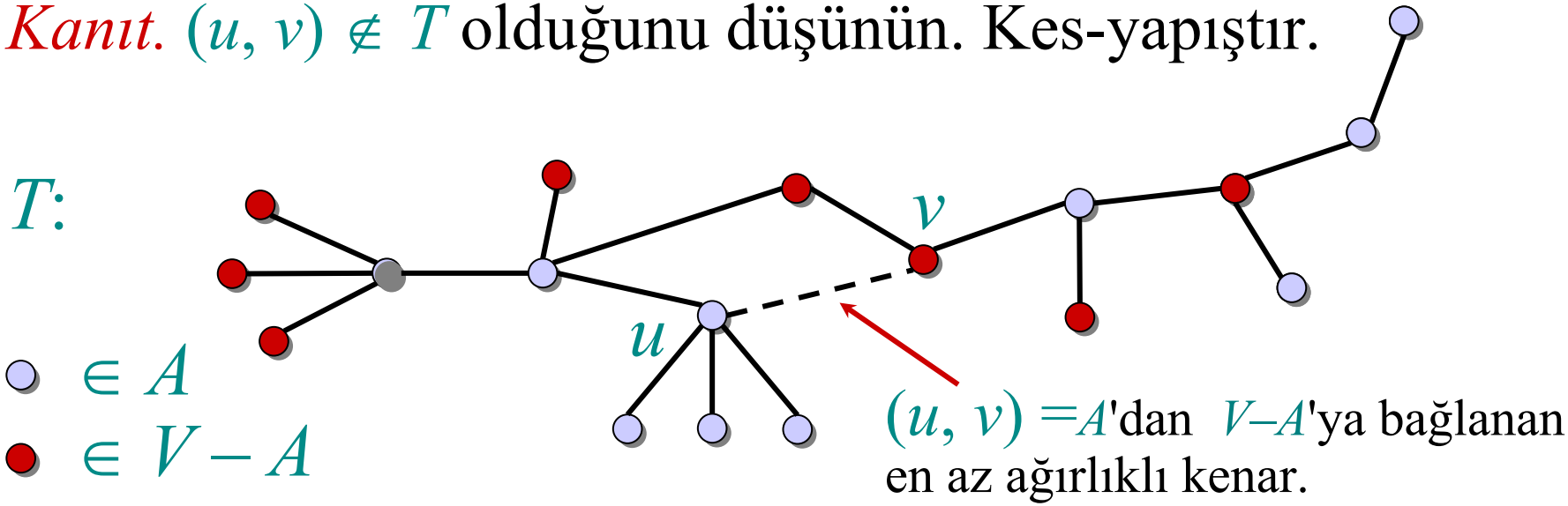
Kanıt. $(u, v) \notin T$ olduğunu düşünün.Kes-yapıştır.



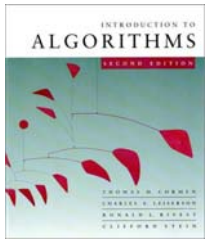


Teorem' in kanıtı

Kanıt. $(u, v) \notin T$ olduğunu düşünün. Kes-yapıştır.

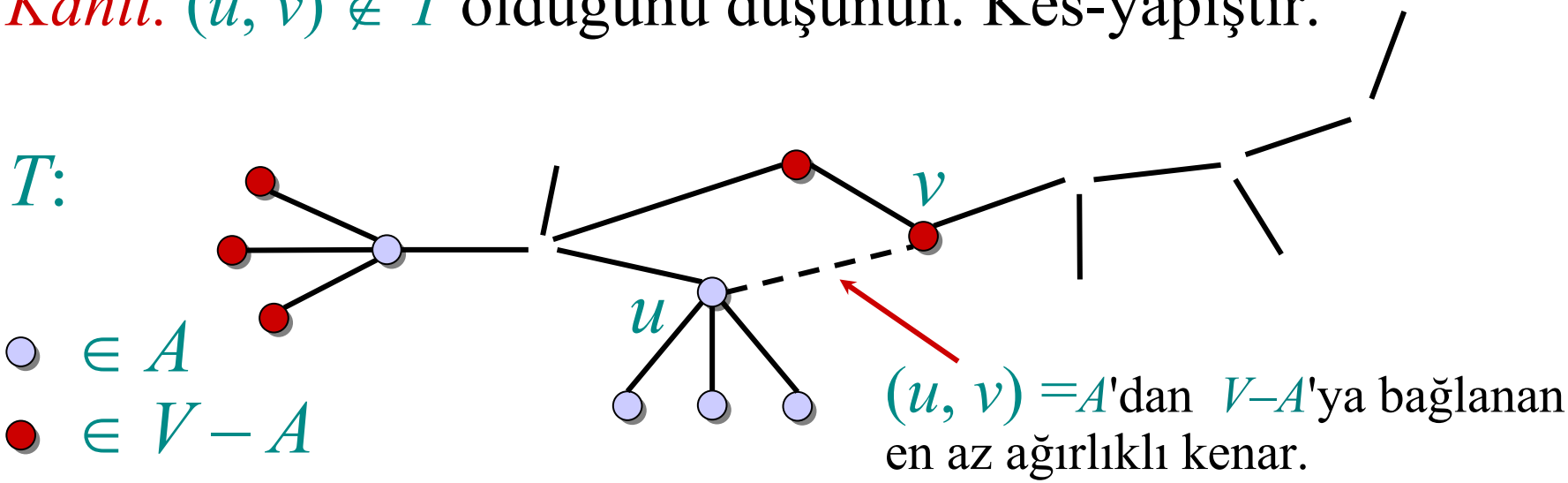


T ' de u'dan v'ye benzeri olmayan basit yolu düşünün.

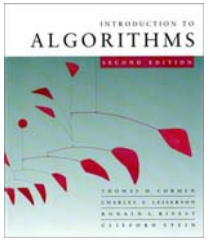


Teorem' in kanıtı

Kanıt. $(u, v) \notin T$ olduğunu düşünün. Kes-yapıştır.

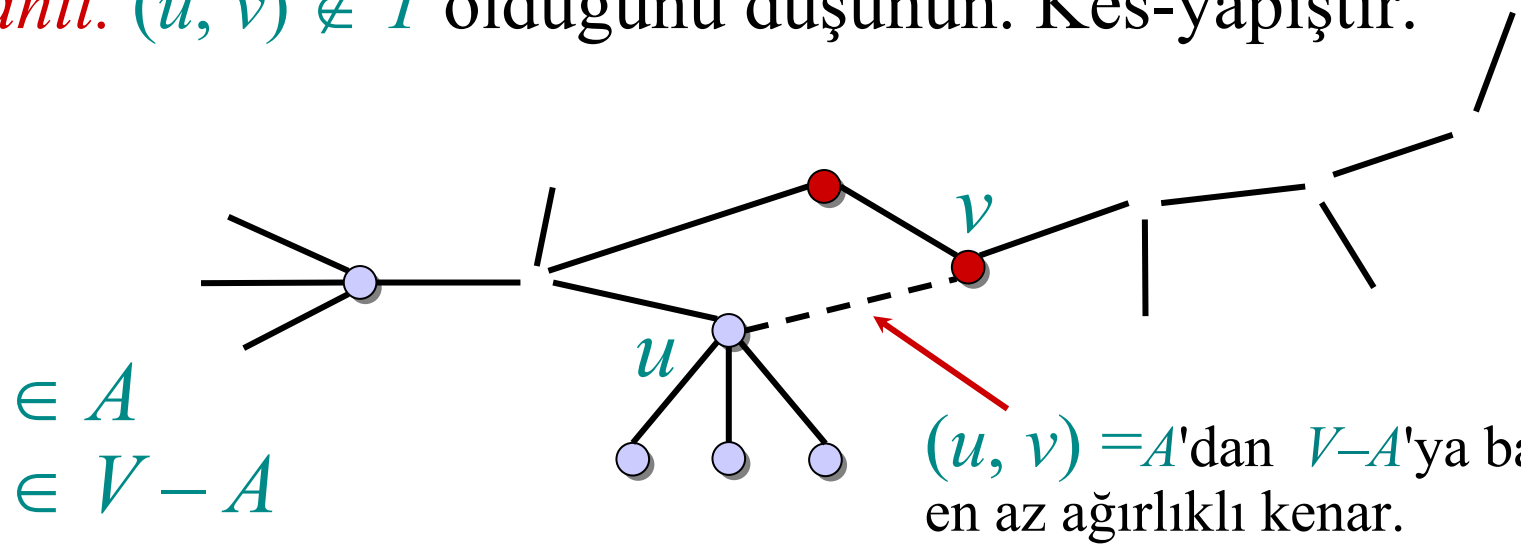


T ' de u'dan v'ye benzeri olmayan basit yolu düşünün.
(u, v)' yi, bu yolda A' daki bir köşeyi
$V - A$' daki bir köşeye bağlayan ilk kenarla değiştirin.

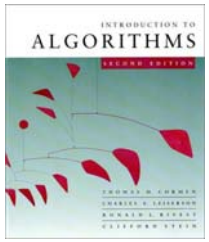


Teorem' in kanıtı

Kanıt. $(u, v) \notin T$ olduğunu düşünün. Kes-yapıştır.



T ' de u'dan v'ye benzeri olmayan basit yolu düşünün.
(u, v)' yi, bu yolda A' daki bir köşeyi
$V - A$' daki bir köşeye bağlayan ilk kenarla değiştirin.
T 'den daha az ağırlıklı bir kapsayan ağaç oluşur. □



Prim'in Algoritması

Fikir: $V - A$'yı bir Q öncelikli sırası olarak koruyun. Q' daki her köşeyi, A' daki bir köşeye bağlayan en az ağırlıklı kenarın ağırlığıyla KEYleyin. (anahtarlayın)

$Q \leftarrow V$

$key[v] \leftarrow \infty$ tüm $v \in V$ 'ler için

$key[s] \leftarrow 0$ rastgele $s \in V$ 'için

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

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

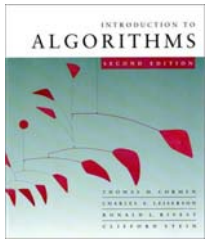
her $v \in Adj[u]$ için

(yap-eğer) **do if** $v \in Q$ ve $w(u, v) < key[v]$

(sonra) **then** $key[v] \leftarrow w(u, v)$ ▷ DECREASE-KEY

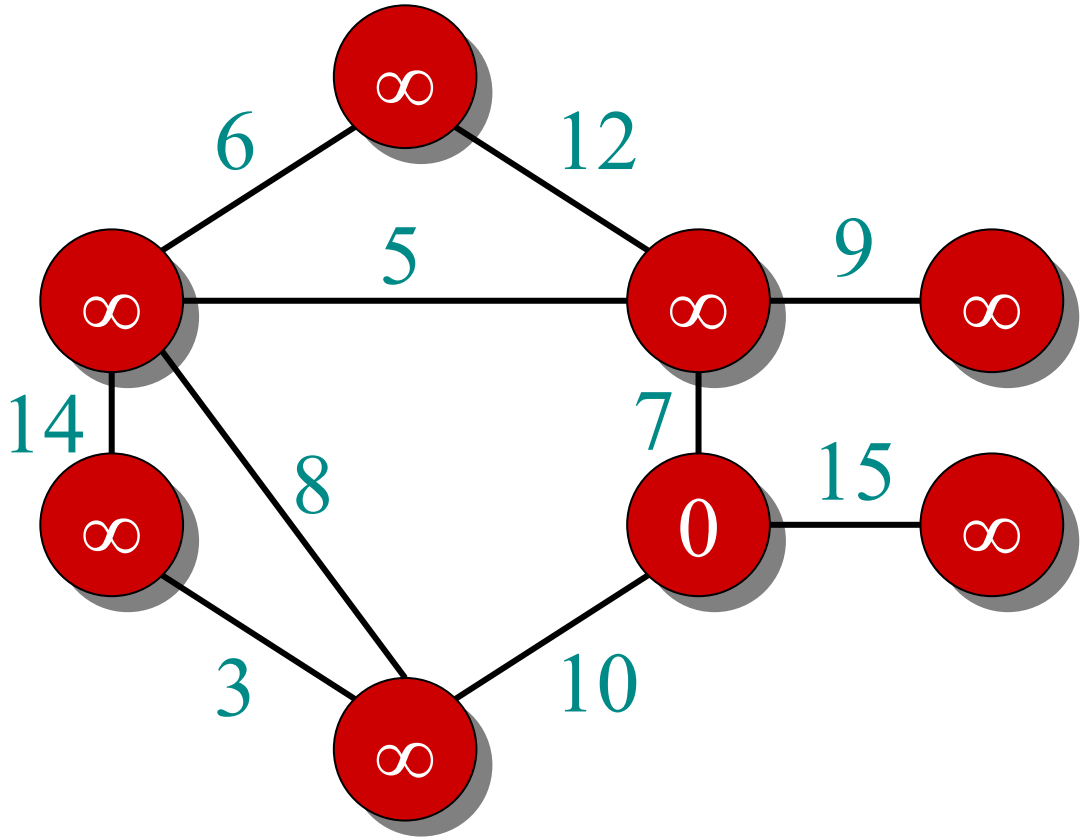
$\pi[v] \leftarrow u$ (anahtarı küçült)

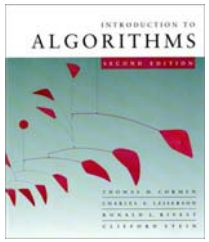
Sonunda, $\{(v, \pi[v])\}$, MST' yi biçimlendirir.



Prim’in algoritmasına örnek

- $\in A$
- $\in V - A$

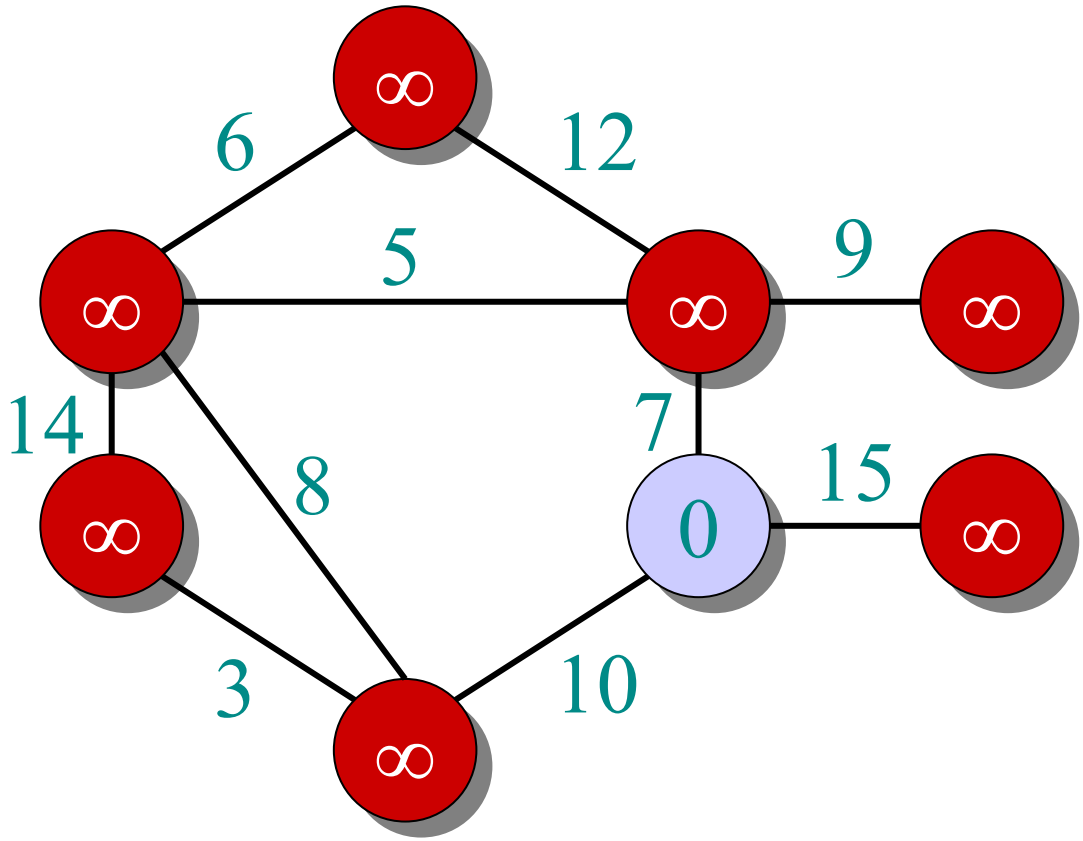


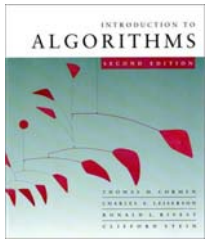


Prim’in algoritmasına örnek

$\in A$

$\in V - A$

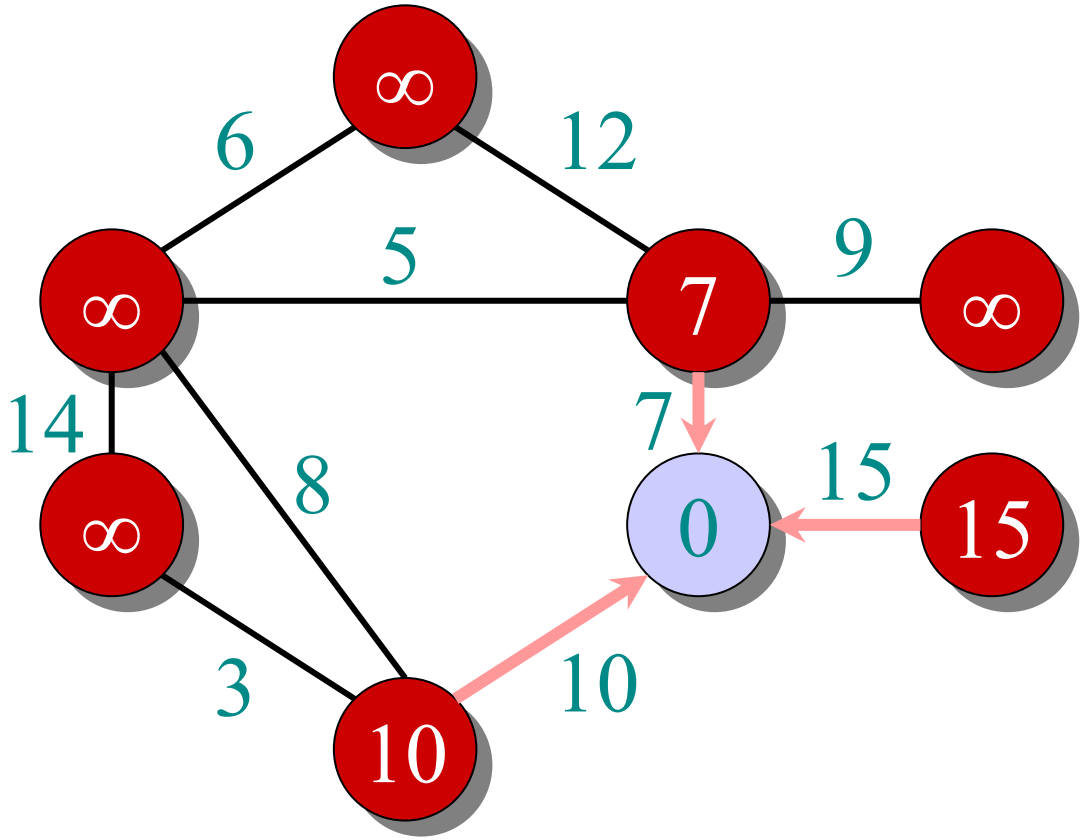


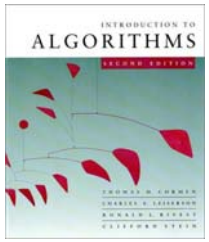


Prim’in algoritmasına örnek

$\in A$

$\in V - A$

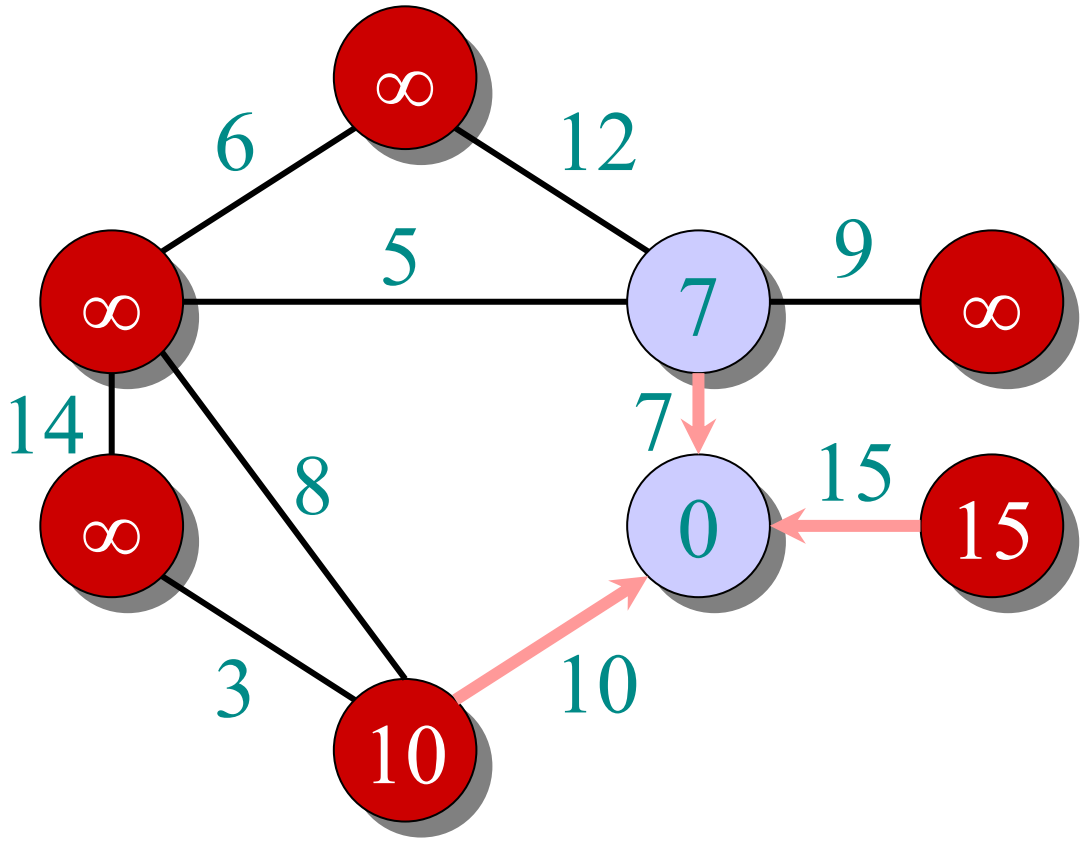


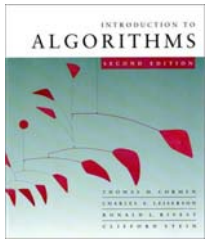


Prim’in algoritmasına örnek

$\in A$

$\in V - A$

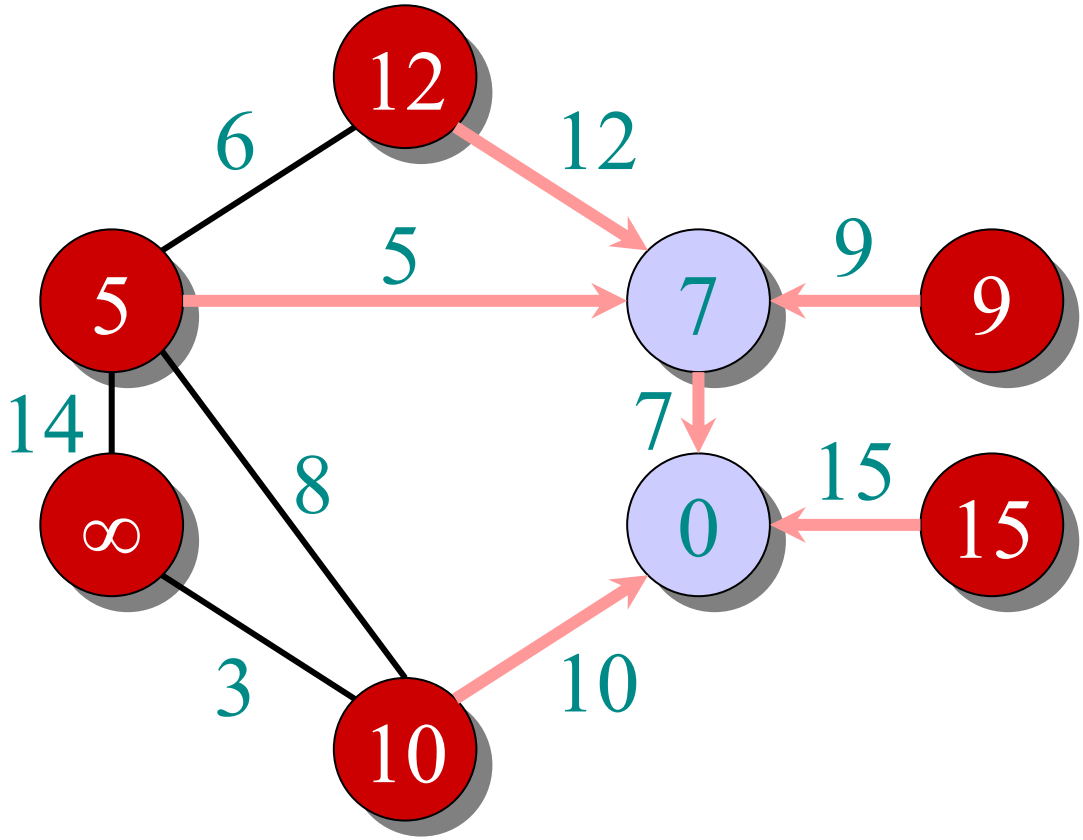


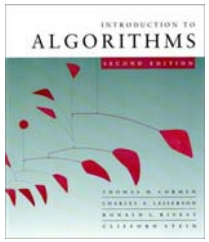


Prim’in algoritmasına örnek

$\circ \in A$

$\circ \in V - A$

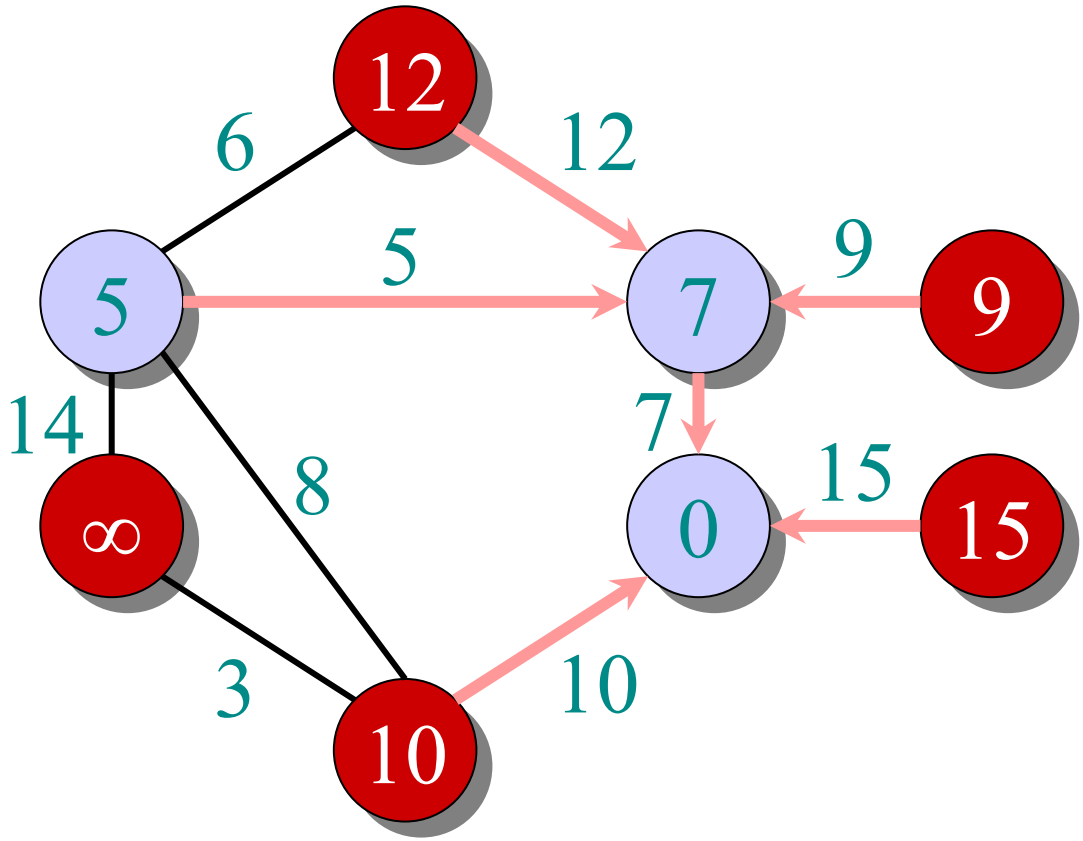


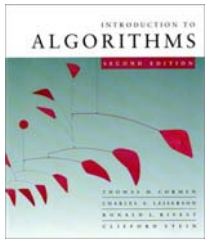


Prim’in algoritmasına örnek

$\in A$

$\in V - A$

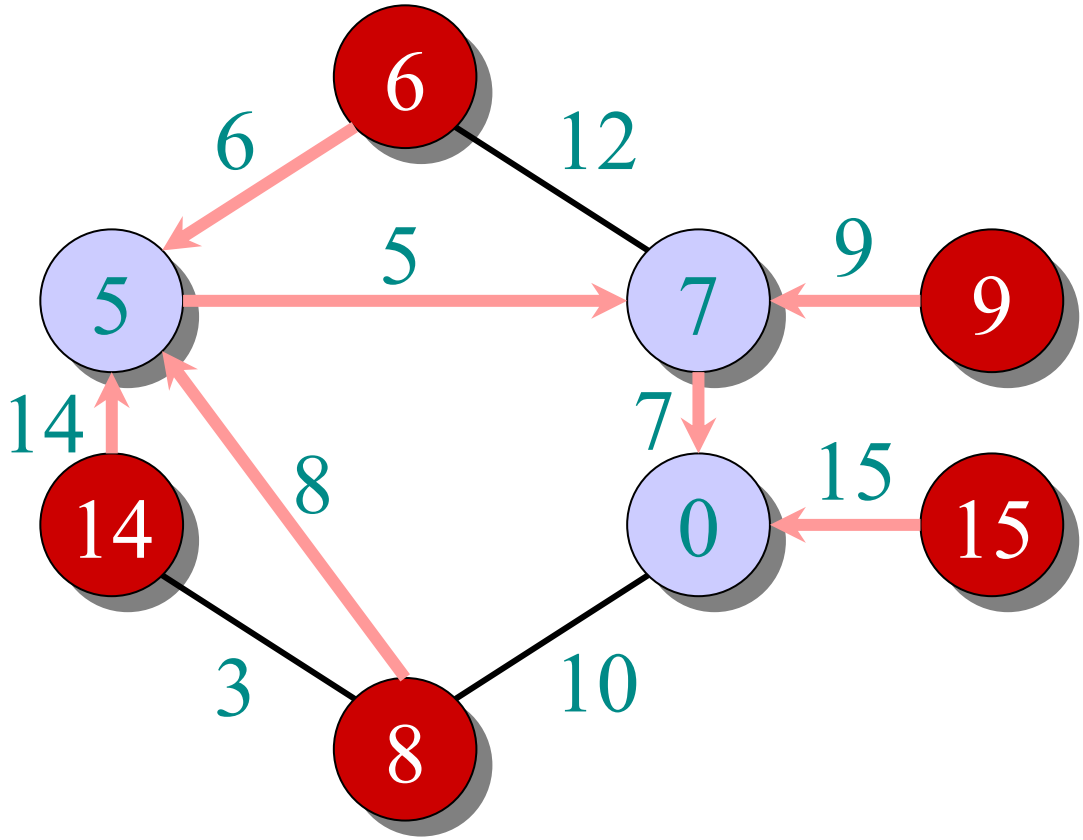


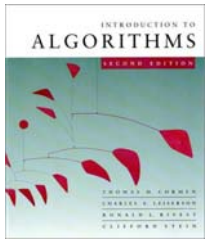


Prim’in algoritmasına örnek

$\in A$

$\in V - A$

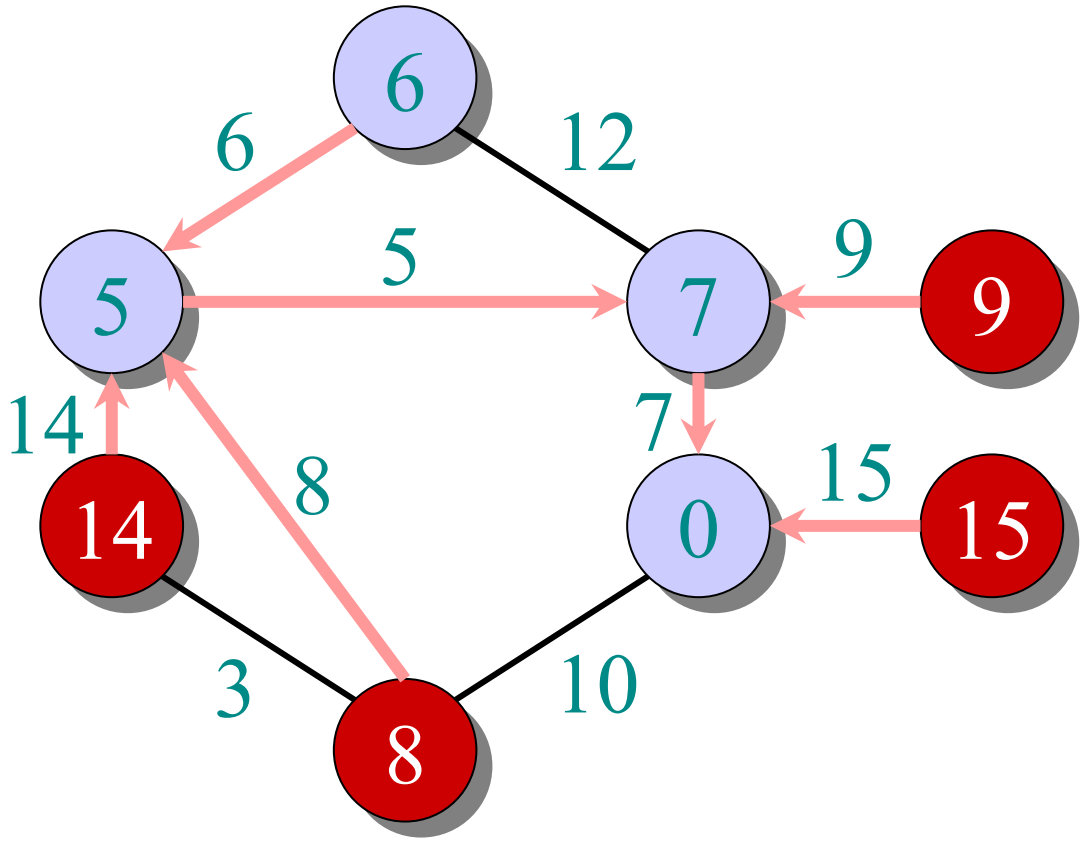


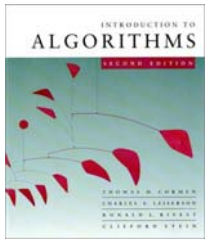


Prim’in algoritmasına örnek

$\in A$

$\in V - A$

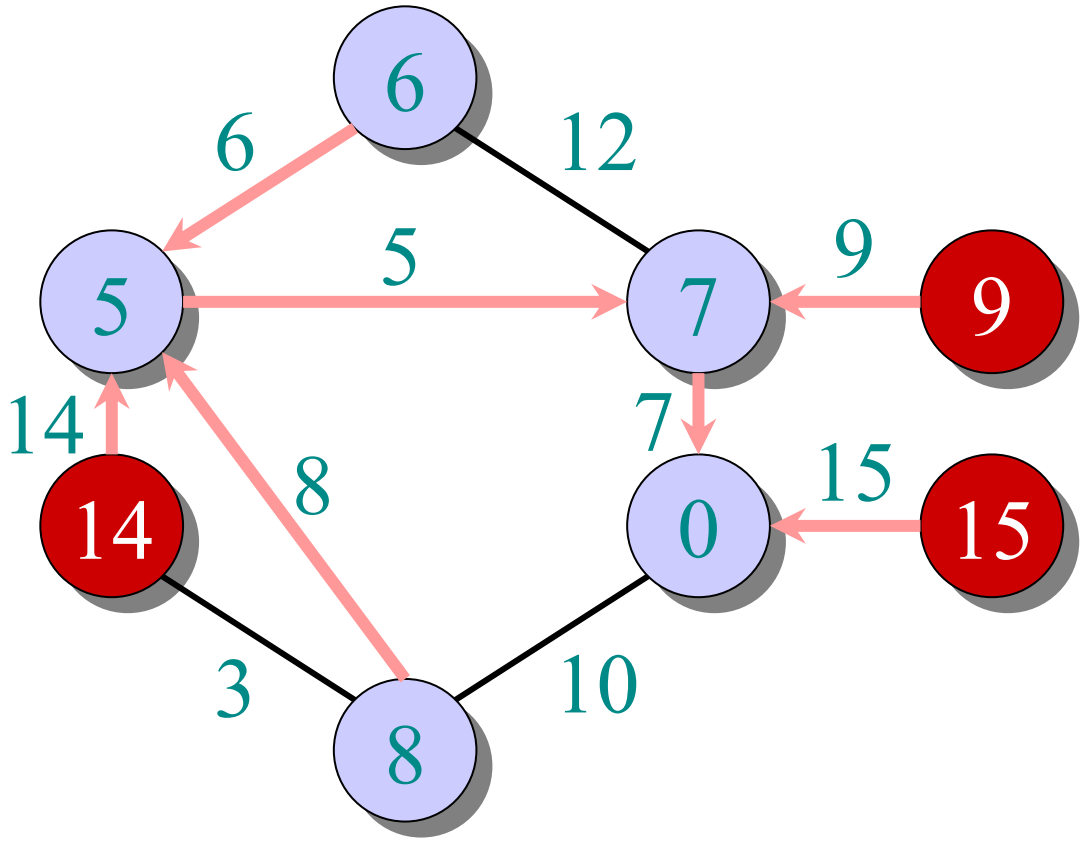


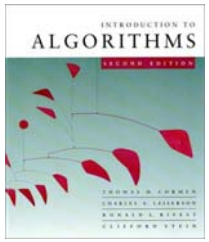


Prim’in algoritmasına örnek

$\in A$

$\in V - A$

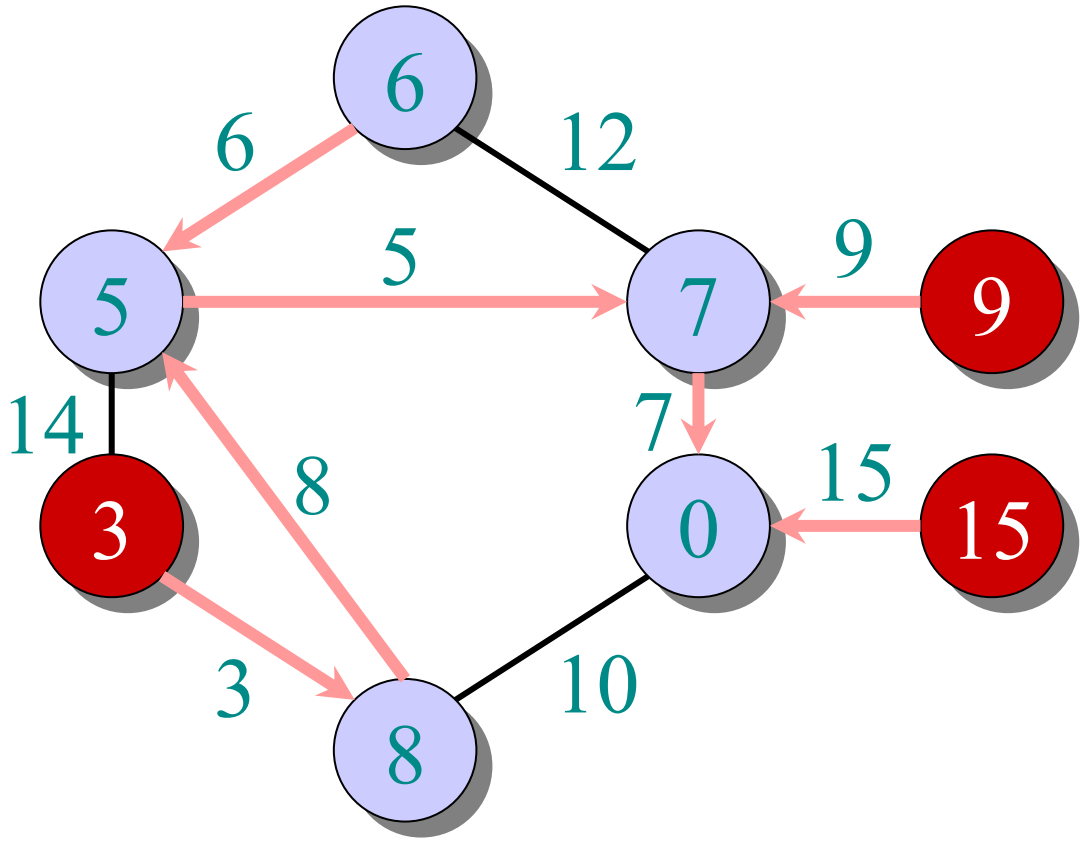


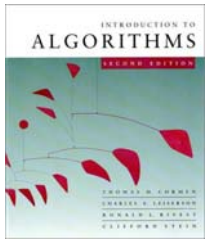


Prim’in algoritmasına örnek

$\in A$

$\in V - A$

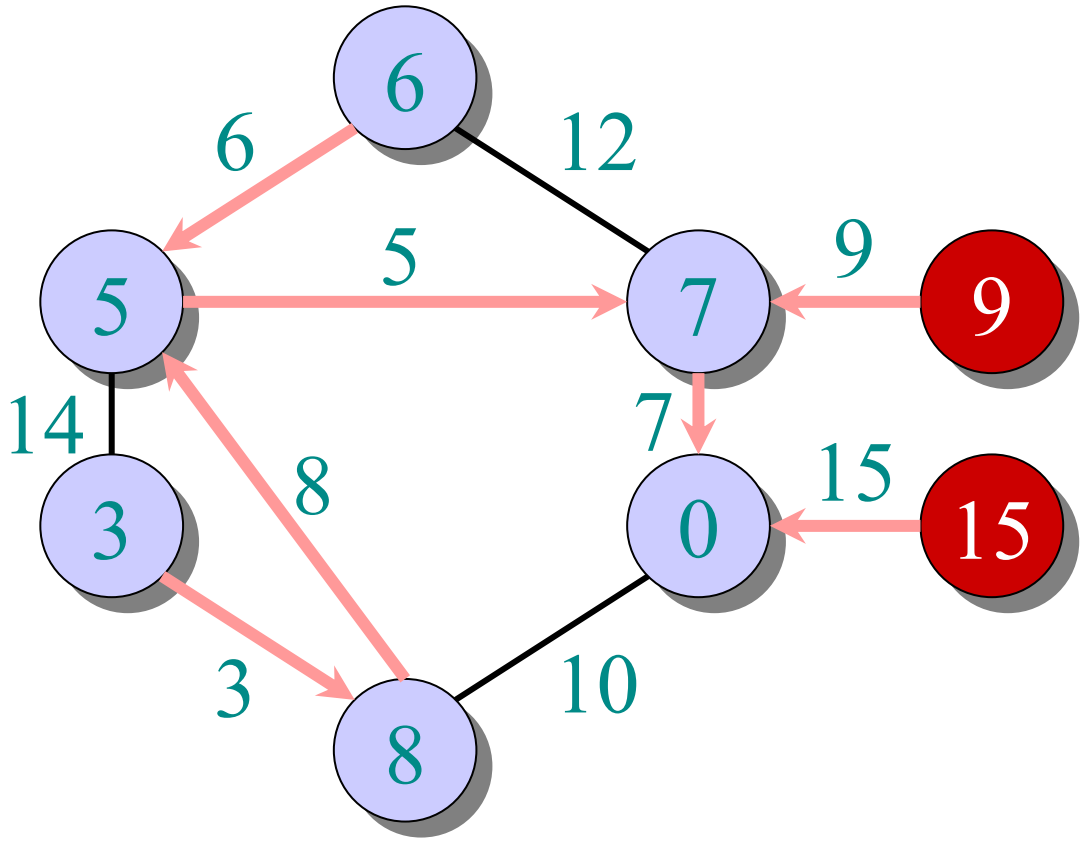


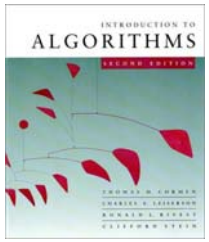


Prim’in algoritmasına örnek

$\in A$

$\in V - A$

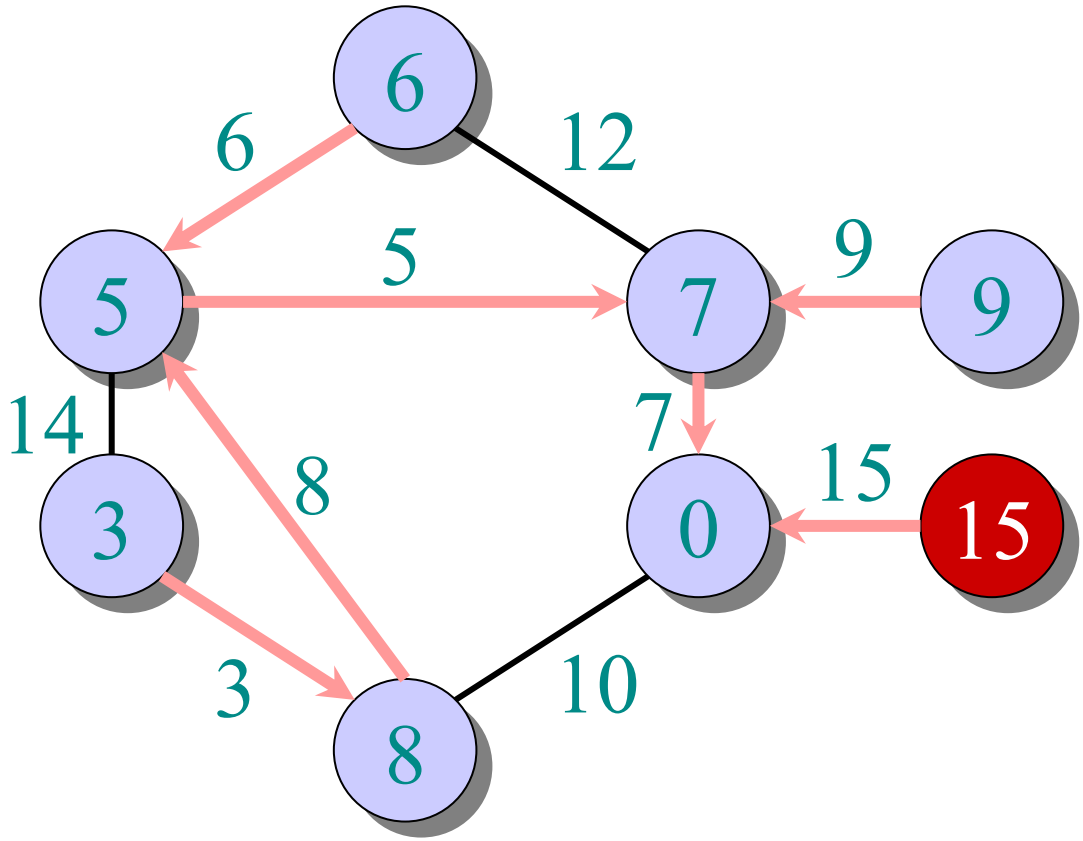


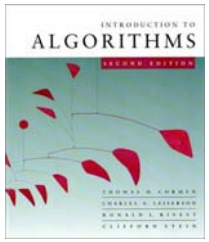


Prim’in algoritmasına örnek

$\in A$

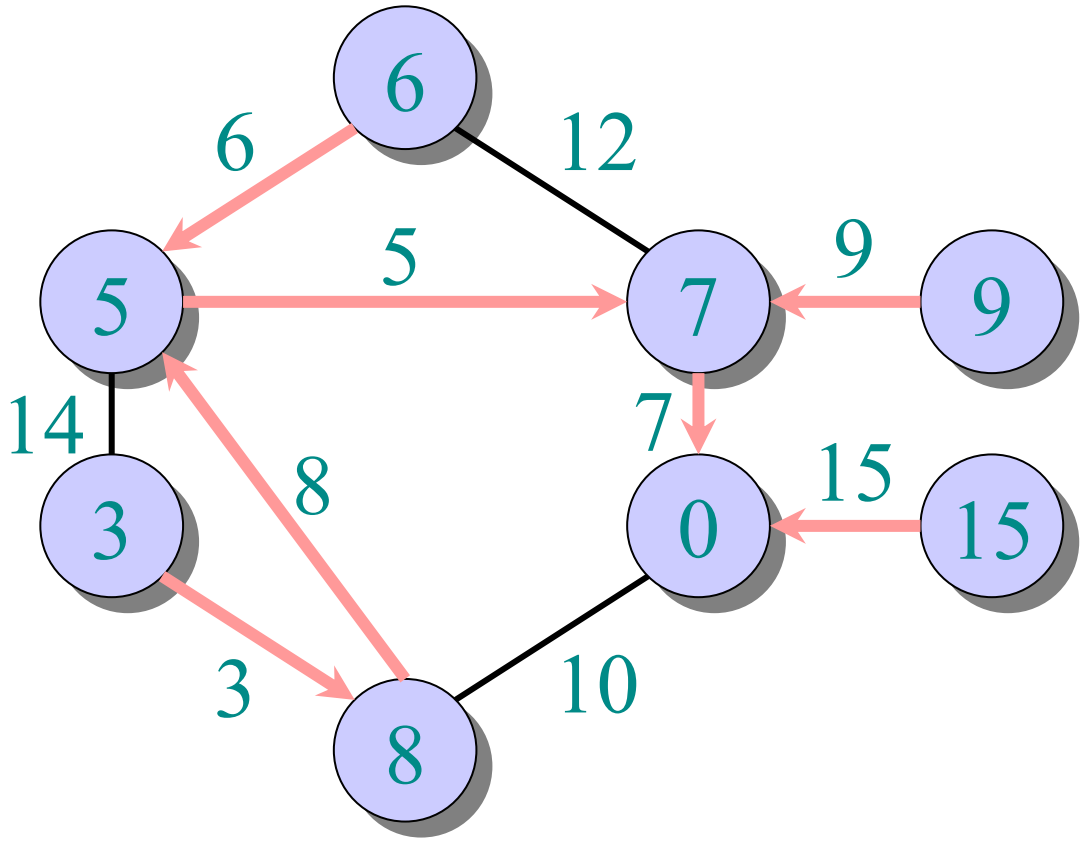
$\in V - A$

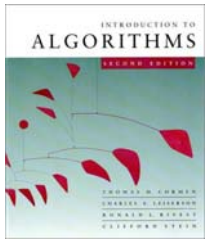




Prim’in algoritmasına örnek

- $\in A$
- $\in V - A$





Prim' in çözümlemesi

$Q \leftarrow V$

$key[v] \leftarrow \infty$ tüm $v \in V$'ler için.

$key[s] \leftarrow 0$ rastgele $s \in V$'ler için.

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

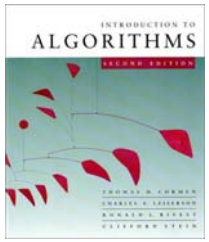
(yap) **do** $u \leftarrow$ En az (Q)' yu çıkar.

her $v \in Adj[u]$ için

do if (yap-eğer) $v \in Q$ ve $w(u, v) < key[v]$

then (sonra) $key[v] \leftarrow w(u, v)$

$\pi[v] \leftarrow u$



Prim' in çözümlemesi

$\Theta(V)$ toplam

$Q \leftarrow V$

$key[v] \leftarrow \infty$ tüm $v \in V$'ler için.

$key[s] \leftarrow 0$ rastgele $s \in V$'ler için.

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

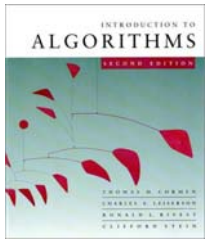
(yap)**do** $u \leftarrow$ En az (Q)' yu çıkar.

her $v \in Adj[u]$ **için**

do if (yap-eğer) $v \in Q$ ve $w(u, v) < key[v]$

then (sonra) $key[v] \leftarrow w(u, v)$

$\pi[v] \leftarrow u$



Prim' in çözümlemesi

$\Theta(V)$ toplam

$Q \leftarrow V$

$key[v] \leftarrow \infty$ tüm $v \in V$'ler için.

$key[s] \leftarrow 0$ rastgele $s \in V$'ler için.

$|V|$ kere

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

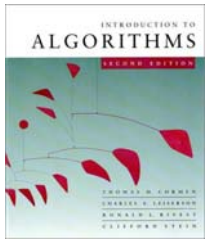
(yap) **do** $u \leftarrow$ En az(Q)'yu çıkar.

her $v \in Adj[u]$ **için**

do if $v \in Q$ ve $w(u, v) < key[v]$

(yap-eğer) **then** $key[v] \leftarrow w(u, v)$

(sonra) $\pi[v] \leftarrow u$



Prim' in çözümlemesi

$\Theta(V)$ toplam

$Q \leftarrow V$

$key[v] \leftarrow \infty$ tüm $v \in V$' ler için.

$key[s] \leftarrow 0$ rastgele $s \in V$' ler için.

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

(yap) **do** $u \leftarrow$ En az(Q)' yu çıkar.

$|V|$ kere

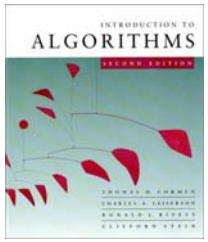
(derece) $degree(u)$ kere

her $v \in Adj[u]$ **için**

(yap-eğer) **do if** $v \in Q$ ve $w(u, v) < key[v]$

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

$\pi[v] \leftarrow u$



Prim' in çözümlemesi

$\Theta(V)$ toplam

$Q \leftarrow V$

$key[v] \leftarrow \infty$ tüm $v \in V$' ler için.

$key[s] \leftarrow 0$ rastgele $s \in V$' ler için.

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

(yap) **do** $u \leftarrow$ En az(Q)' yu çıkar.

$|V|$ kere

(derece) $degree(u)$ kere

her $v \in Adj[u]$ **için**

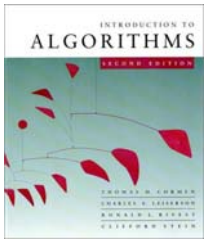
do if $v \in Q$ ve $w(u, v) < key[v]$

(yap-eğer) **then** $key[v] \leftarrow w(u, v)$

(sonra) $\pi[v] \leftarrow u$

Tokalaşma önkuramı $\Rightarrow$ $\Theta(E)$ varsayılan DECREASE-KEY'ler.

(Anahtarı küçült)



Prim' in çözümlemesi

$\Theta(V)$ toplam {

$Q \leftarrow V$

$key[v] \leftarrow \infty$ tüm $v \in V$' ler için.

$key[s] \leftarrow 0$ rastgele $s \in V$' ler için.

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

(yap)**do** $u \leftarrow$ En az (Q)' yu çıkar.

$|V|$ kere {

(derece) $degree(u)$ kere {

her $v \in Adj[u]$ için

do if $v \in Q$ ve $w(u, v) < key[v]$

(yap-eğer) **then** $key[v] \leftarrow w(u, v)$

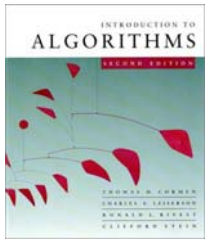
(sonra) $\pi[v] \leftarrow u$

Tokalaşma önkuramı $\Rightarrow \Theta(E)$ varsayılan DECREASE-KEY'ler.

(Anahtarı küçült)

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

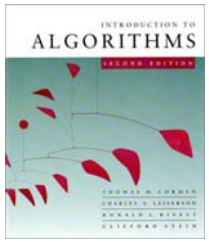
(en küçüğü çıkar)



Prim' in çözümlemesi (devamı)

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

(en küçüğü çıkar) (Anahtarı küçült)

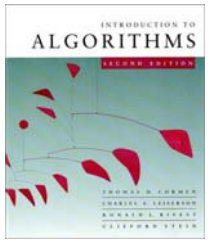


Prim' in çözümlemesi (devamı)

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

(en küçüğü çıkar) (Anahtarı küçült)

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

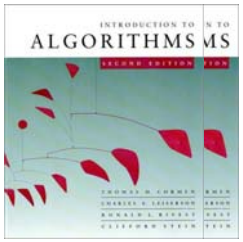


Prim' in çözümlemesi(devamı)

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	$O(V)$	$O(1)$	$O(V^2)$

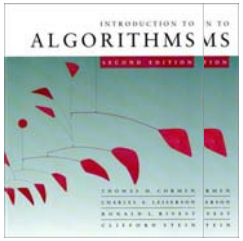


Prim' in çözümlemesi(devamı)

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	$O(V)$	$O(1)$	$O(V^2)$
ikili yığın	$O(\lg V)$	$O(\lg V)$	$O(E \lg V)$

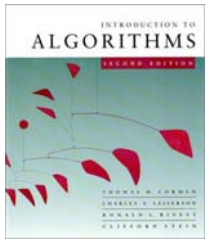


Prim' in çözümlemesi (devamı)

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

(en azı çıkar) (Azaltılmış anahtar)

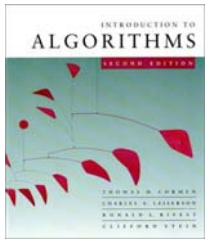
Q	$T_{\text{EXTRACT-MIN}}$	$T_{\text{DECREASE-KEY}}$	Toplam
dizilim	$O(V)$	$O(1)$	$O(V^2)$
ikili yığın	$O(\lg V)$	$O(\lg V)$	$O(E \lg V)$
Fibonacci yığını	$O(\lg V)$ amortize edilmiş	$O(1)$ amortize edilmiş	$O(E + V \lg V)$ worst case en kötü durum



MST algoritmaları

Kruskal algoritması (CLRS ye bakınız):

- ***Ayrışık-küme veri yapısı*** 'nı kullanır.(Ders 10)
- Koşma süresi = $O(E \lg V)$.



MST algoritmaları

Kruskal algoritması (CLRS ye bakınız):

- ***Kopuk-küme veri yapısı*** 'nı kullanır.(Ders 10)
- Koşma süresi = $O(E \lg V)$.

Bugüne kadar en iyisi:

- Karger, Klein, and Tarjan [1993].
- Rastgele algoritma.
- $O(V + E)$ beklenen süre.