YZM 3207- ALGORİTMA ANALİZİ VE TASARIM

DERS#9: AÇGÖZLÜ ALGORİTMALAR

Aç Gözlü Algoritmalar (Greedy)

- Bozuk para verme problemi
  - Bir kasiyer 48 kuruş para üstünü nasıl verir?
    - 25 kuruş, 10 kuruş, 5 kuruş, 1 kuruş
  - Hırslı teknik ilk olarak 25 kuruş verir
    - Geri kalan miktarı en çok o azaltır
  - İkinci de 25 kuruş veremez
    - 10 kuruş verir
  - Üçüncüde kalan miktarı en aza indirmek için 10 kuruş verir
  - Kalan kısım için 3 adet 1 kuruş verebilir

Aç Gözlü Algoritmalar (Greedy)

- Sadece optimizasyon problemlerinde kullanılabilmelerine rağmen genel bir tasarım tekniği olarak kabul edilir.
- Açgözlü algoritmalar ardışık adımlarla bir çözüm oluşturma yaklaşımı izler
  - O ana kadar oluşturulan kısmi çözüm problemin tam çözümüne ulaşana kadar her adımda genişletilir
  - Atılan her adımda yapılacak seçim:
    - Problemin kısıtlarına göre uygulanması mümkün olmalıdır
    - O adım için tüm mümkün seçenekler arasında en optimal olanı olmalıdır
    - Gerçekleştirildikten sonra ki adımlarda değiştirilemez olmalıdır

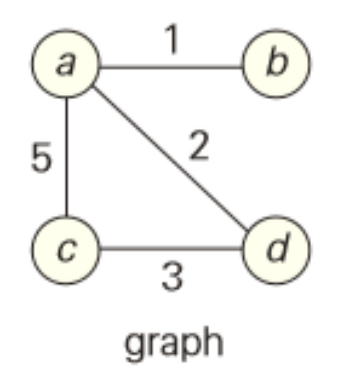
Minimum Kapsama Ağacı Problemi

- Tüm network türlerinde karşılaşılan bir problem
  - N adet noktayı her nokta çifti arasında bir yol olacak şekilde en ucuz maliyetle birleştirmek
    - Elektrik
    - Haberleşme
    - Ulaşım
  - Veri setleri içerisindeki noktaları kümeleme için kullanılabilirler
    - Sınıflandırma problemlerinin çözümünde kullanılabilirler
  - Bir graf yapısı ile gösterilebilirler
    - Notalar düğüm
    - Ayrıtlar yollar
    - Maliyetler ise ayrıt ağırlıkları ile ifade edilir
  - Bu durumda problem «minimum kapsama ağacı»nın (Minimum Spanning Tree-MST) bulunması olarak nitelenebilir

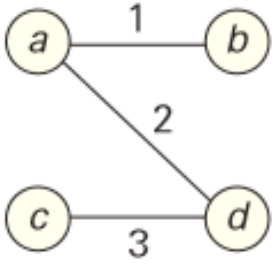
Minimum Kapsama Ağacı Problemi

- Kapsama Ağacı (Spanning Tree)
  - Yönsüz, temaslı bir grafın tüm düğümlerini kapsayan, çevrimsiz, temaslı bir alt grafıdır (Ağaç)
  - Grafın ayrıtlarının ağırlıkları varsa minimum tarama ağacı en düşük ağırlıklı kapsama ağacıdır
    - Ağacın ağırlığı = Tüm ayrıtlarının ağırlık toplamı
  - Minimum Kapsama Ağacı (Minimum Spanning Tree – MST) problemi verilen ağırlıklı bir grafın MST'sinin bulunmasıdır

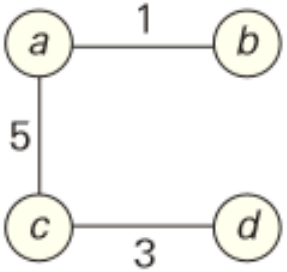
Minimum Kapsama Ağacı Problemi



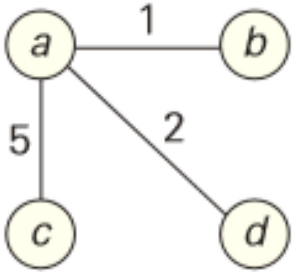
graph



$w(T_1) = 6$



$w(T_2) = 9$



$w(T_3) = 8$

Graph and its spanning trees, with T_1 being the minimum spanning tree.

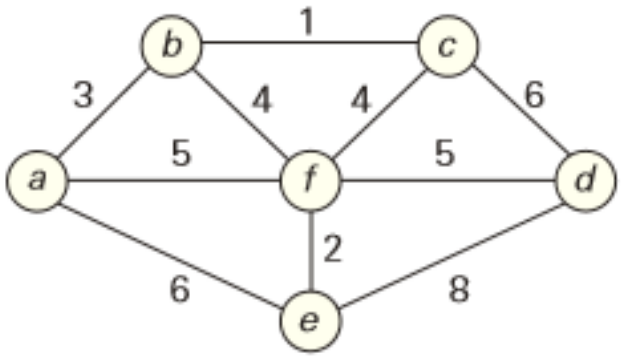
Prim’in MST Algoritması

- Prim’in alt ağacı ardışık adımlarla genişleterek MST oluşturur
  - Başlangıç olarak herhangi bir düğüm seçilir
  - Her adımda bulunulan düğüme en yakın (ağaca önceden dahil olmamış) düğüm ağaca dahil edilir
  - Eşit mesafede iki düğüm varsa belirlenen kurala göre biri tercih edilir
  - n düğüm için n-1 adet iterasyon gerçekleşir

Prim'in MST Algoritması

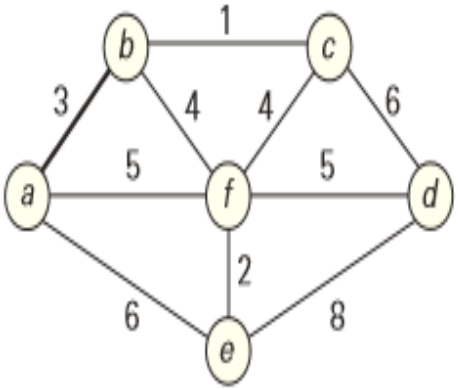
ALGORITHM *Prim(G)*

//Prim's algorithm for constructing a minimum spanning tree
//Input: A weighted connected graph $G = \langle V, E \rangle$
//Output: E_T, the set of edges composing a minimum spanning tree of G
$V_T \leftarrow \{v_0\}$ //the set of tree vertices can be initialized with any vertex
$E_T \leftarrow \varnothing$
for $i \leftarrow 1$ **to** $|V| - 1$ **do**
    find a minimum-weight edge $e^* = (v^*, u^*)$ among all the edges (v, u)
    such that v is in V_T and u is in $V - V_T$
    $V_T \leftarrow V_T \cup \{u^*\}$
    $E_T \leftarrow E_T \cup \{e^*\}$
return E_T



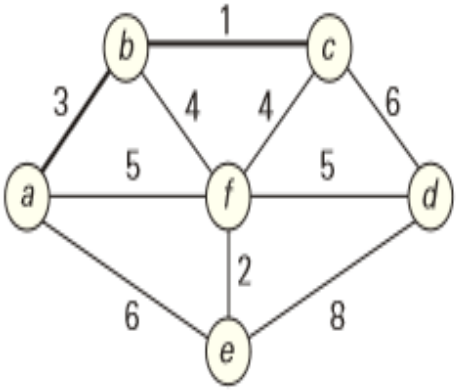
a(−, −)

b(a, 3) c(−, ∞) d(−, ∞)
e(a, 6) f(a, 5)



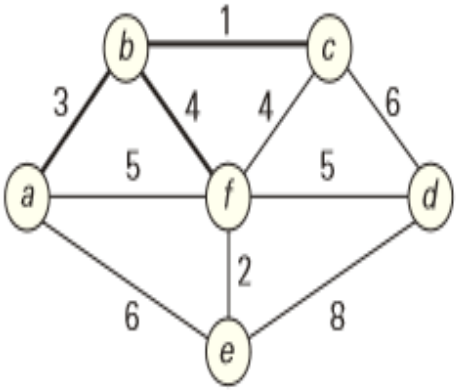
b(a, 3)

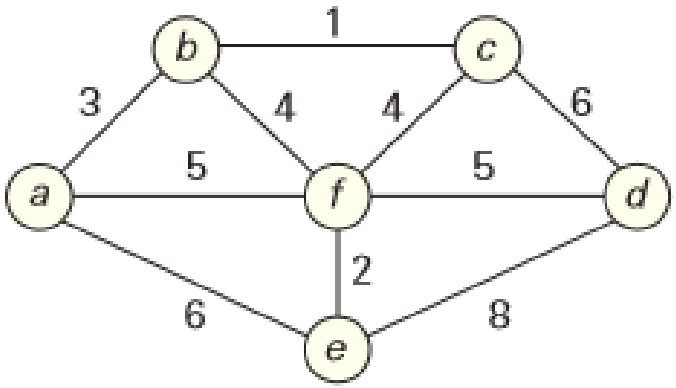
c(b, 1) d(−, ∞) e(a, 6)
f(b, 4)



c(b, 1)

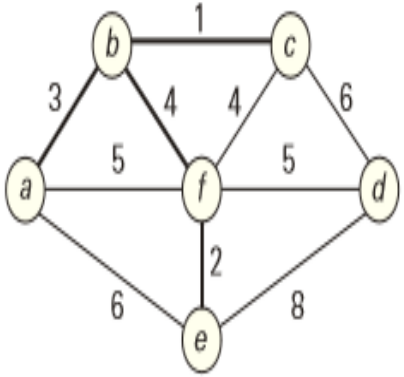
d(c, 6) e(a, 6) **f(b, 4)**





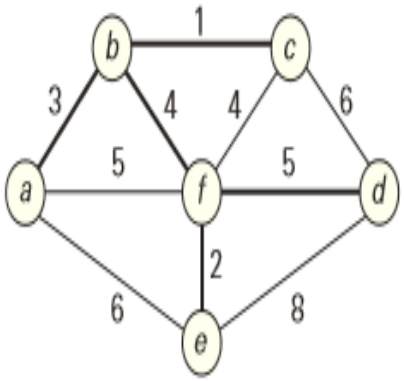
f(b, 4)

d(f, 5) **e(f, 2)**



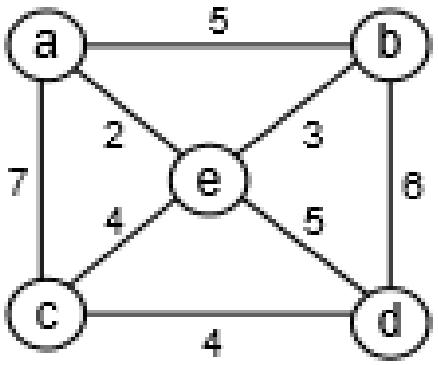
e(f, 2)

d(f, 5)



d(f, 5)

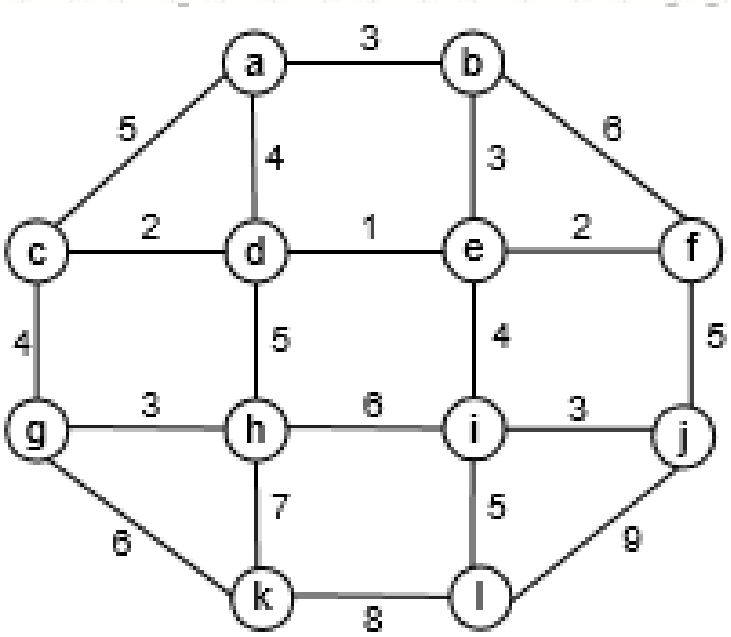
Örnek



Tree vertices	Priority queue of remaining vertices
a(-,-)	b(a,5) c(a,7) d(a,∞) **e(a,2)**
e(a,2)	**b(e,3)** c(e,4) d(e,5)
b(e,3)	**c(e,4)** d(e,5)
c(e,4)	**d(c,4)**
d(c,4)	

The minimum spanning tree found by the algorithm comprises the edges *ae*, *eb*, *ec*, and *cd*.

Örnek

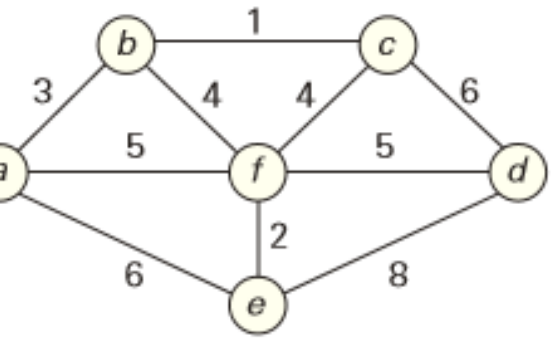


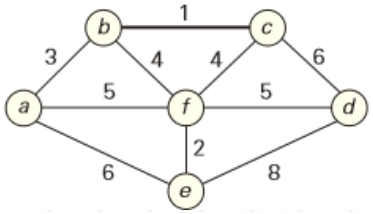
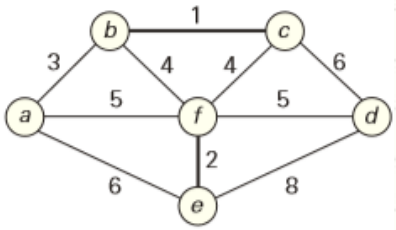
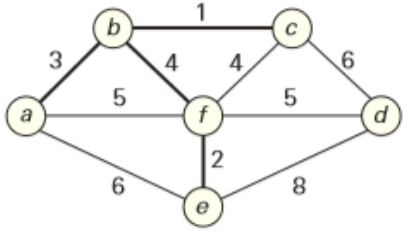
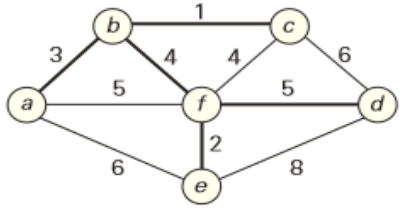
Tree vertices	Priority queue of fringe vertices
a(-,-)	**b(a,3)** c(a,5) d(a,4)
b(a,3)	c(a,5) d(a,4) **e(b,3)** f(b,6)
e(b,3)	*c(a,5)* **d(e,1)** f(e,2) i(e,4)
d(e,1)	**c(d,2)** f(e,2) i(e,4) h(d,5)
c(d,2)	**f(e,2)** i(e,4) h(d,5) g(c,4)
f(e,2)	**i(e,4)** h(d,5) g(c,4) j(f,5)
i(e,4)	h(d,5) g(c,4) **j(i,3)** l(i,5)
j(i,3)	h(d,5) **g(c,4)** l(i,5)
g(c,4)	**h(g,3)** l(i,5) k(g,6)
h(g,3)	**l(i,5)** k(g,6)
l(i,5)	**k(g,6)**
k(g,6)	

The minimum spanning tree found by the algorithm comprises the edges *ab*, *be*, *ed*, *dc*, *ef*, *ei*, *ij*, *cg*, *gh*, *il*, *gk*.

Kruskal'ın Algoritması

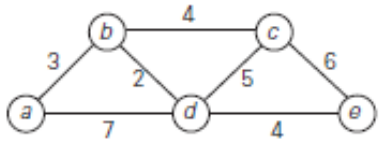
- MST probleminin çözümü için geliştirilmiş açgözlü başka bir algoritma
  - Bu algoritma ağırlıklı, temaslı bir G=(V, E) grafının MST'sine ayrıt ağırlıkları toplamı en düşük, çevrimsel olmayan |V|-1 adet ayrıtı olan bir alt graf olarak oluşturur
  - Bu alt graf aynı zamanda bir ağaçtır
  - Algoritma ardışık adımlarla ilerler
    - Her adımda altgrafı çevrimsel olmayacak şekilde genişletir
    - Altgrafa eklenen yeni düğümün temaslı olması gerekmez

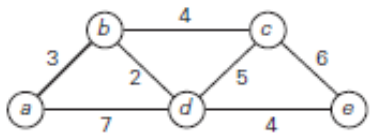
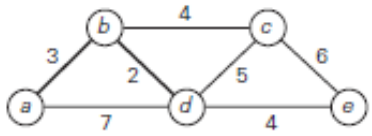
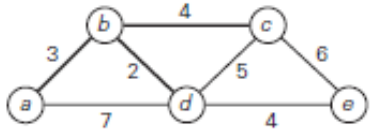
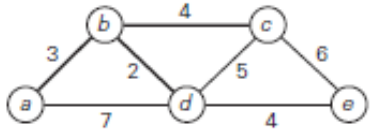


Tree edges	Sorted list of edges	Illustration
	bc ef ab bf cf af df ae cd de 1 2 3 4 4 5 5 6 6 8	
bc 1	bc **ef** ab bf cf af df ae cd de 1 2 3 4 4 5 5 6 6 8	
ab 3	bc ef ab **bf** cf af df ae cd de 1 2 3 4 4 5 5 6 6 8	
bf 4	bc ef ab bf cf af **df** ae cd de 1 2 3 4 4 5 5 6 6 8	
df 5		

Dijkstra Algoritması

- Dijkstra Algoritması
  - En kısa yol algoritması
  - Komşu düğümlerden en yakın olan seçilerek hedefe ulaşılmaya çalışılır.



Tree vertices	Remaining vertices	Illustration
a(−, 0)	**b(a, 3)** c(−, ∞) d(a, 7) e(−, ∞)	
b(a, 3)	c(b, 3 + 4) **d(b, 3 + 2)** e(−, ∞)	
d(b, 5)	**c(b, 7)** e(d, 5 + 4)	
c(b, 7)	**e(d, 9)**	
e(d, 9)		

from a to b : $a - b$ of length 3
from a to d : $a - b - d$ of length 5
from a to c : $a - b - c$ of length 7
from a to e : $a - b - d - e$ of length 9