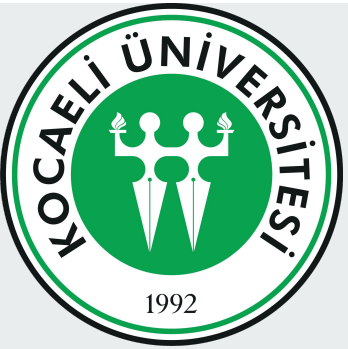


Kesikli Matematik

Ağaçlar

Dr. Öğr. Üyesi Mehmet Ali ALTUNCU
Bilgisayar Mühendisliği

Ağaçlar

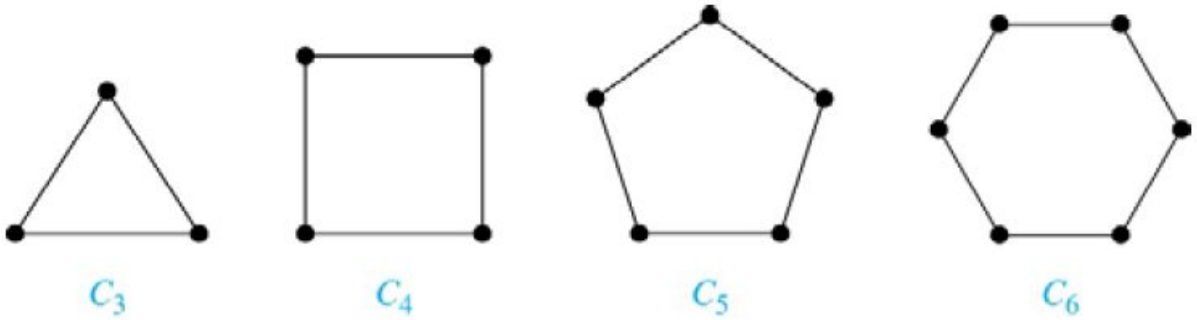
Tanım: Bir ağaç hiç basit devre içermeyen yönsüz graflardır.

Tanım 2: Bir ağaç , herhangi iki köşenin tam olarak bir yolla bağlandığı, döngüsel olmayan yönsüz bir graflardır. Bir ağaçta n düğüm varsa n-1 tane kenar olmalıdır.

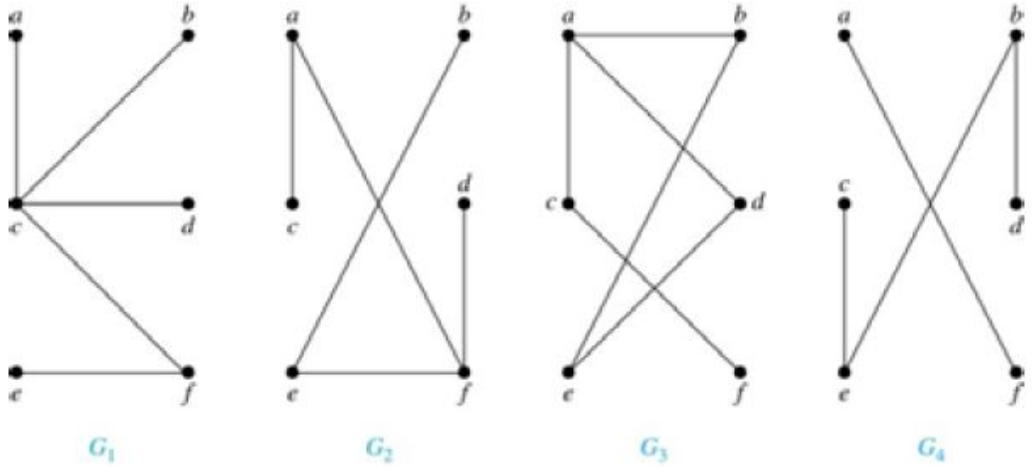
- Bir ağaç basit devre içermediği için, paralel kirişler veya döngü içermez. Dolayısıyla herhangi bir ağaç basit bir graf olmak zorundadır. (Ör: Soyağaçları)
- Ağaçlar bilgisayar bilimlerindeki çeşitli algoritmalarda kullanılırlar.
- Örneğin; ağaçlar bir listedeki nesnelerin yerini bulan verimli algoritmalar tasarlamak için kullanılır.
- Veri iletimi ve depolamada gider azaltma için Huffman kodu gibi verimli kodları tasarlayan algoritmalarda ağaçlar kullanılabilir.
- Ya da dama satranç gibi oyunları incelemede ve bu oyunların kazanma stratejilerini belirlemede kullanılabilir.

Döngü

Tanım: n ≥ 3 için bir C_n döngüsü $v_1, v_2, \cdots, v_n$ n köşeden ve {v1, v2}, {v2, v3} ,$\cdots$, $\{v_{n-1}, v_n\}, \{v_n, v_1\}$ kenarlarından oluşur.



Ağaçlar



Ağaç

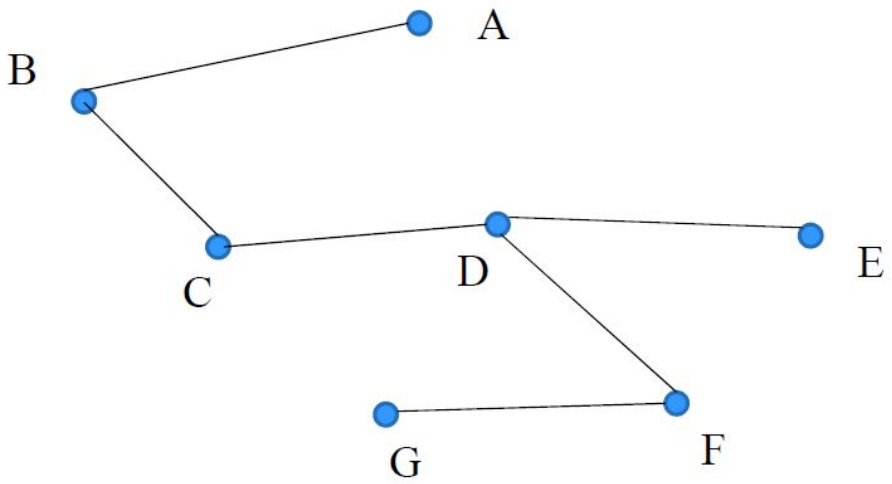
Ağaç

Ağaç
Değil

Ağaç
Değil

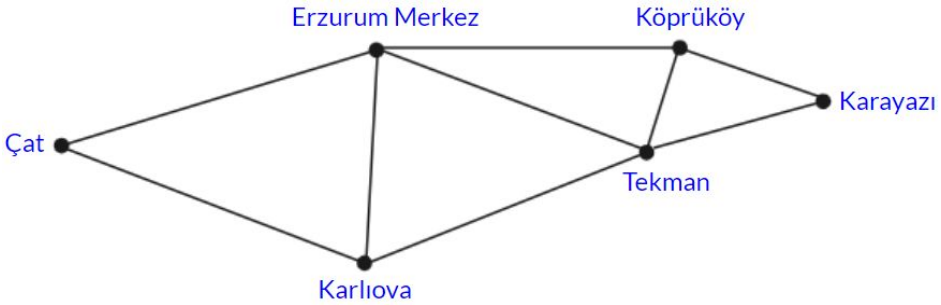
Ağaçlar

Teorem: Yönsüz bir graf, ancak ve ancak her düğüm ikilisi için basit bir tek yol olması durumunda bir ağaçtır.



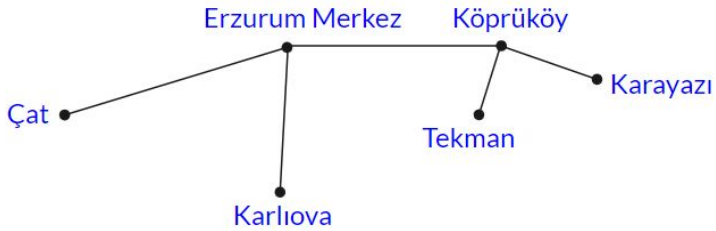
Kapsayan Ağaç

- Şekilde gösterilen basit bir graf ile temsil edilen Erzurum civarındaki yol sistemini ele alalım.
- Kışın yolları açık tutmanın tek yolu yollardaki karı düzenli olarak temizlemektir.
- Karayolları Müdürlüğü, en az sayıdaki yolları temizleyerek herhangi iki ilçenin arasının her zaman açık olmasını istemektedir. Bunu nasıl yapabilir?



Kapsayan Ağaç

- Bu problem, verilen basit bir grafın tüm düğümlerini kapsayan en az kenara sahip bir bağlantılı alt ağaç ile çözülür. Böyle bir graf ağaç olmalıdır.



Tanım: G basit bir graf olsun. G'nin kapsayan bir ağacı, G'nin bir alt grafı olan ve G'nin her düğümünü kapsayan bir ağaçtır. Ya da basit bir graf ancak ve ancak bir kapsayan ağaca sahip olduğunda bağlantılıdır.

Tanım 2: Bir bağlantılı ağırlıklı grafta bir **minimum kapsayan ağaç**, olabilecek en küçük kenarların ağırlıklı toplamına sahip bir kapsayan ağaçtır.

Minimum Kapsayan Ağaçlar İçin Algoritmalar

Prim Algoritması

- Daha az maliyetli kenarları tek tek değerlendirerek yol ağacını bulmaya çalışır.
- Ara işlemler birden çok ağaç olabilir.

Kruskal Algoritması

- En az maliyetli kenardan başlayıp onun uçlarından en az maliyetle genişleyecek kenarın seçilmesine dayanır.
- Sadece bir tane ağaç oluşur.

Prim Algoritması

Prim Algoritmasının uygulanması aşağıdaki adımlardan oluşmaktadır:

Adım-1:

- Rastgele bir tepe noktası seçilir.

Adım-2:

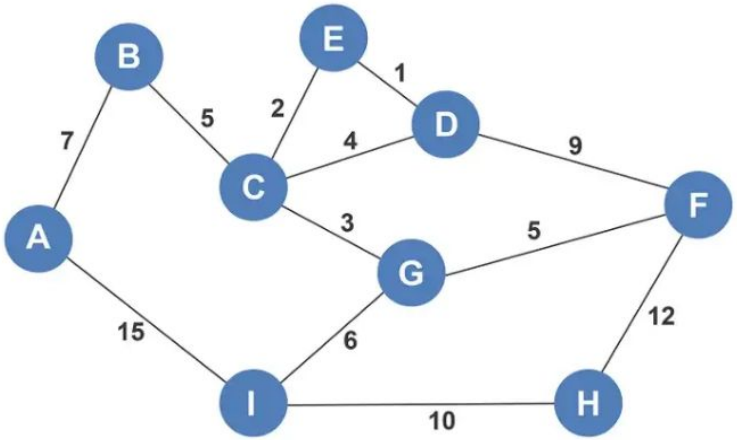
- Ağacı yeni köşelere bağlayan tüm kenarları bulunur.
- Bu kenarlar arasında en az ağırlıklı kenar seçilip, mevcut ağaca dahil edilir.
- Bu kenarı dahil etmek bir döngü oluşturuyorsa, o kenarı seçilmez ve bir sonraki en az ağırlıklı kenarı aranır.

Adım-3:

- Tüm köşeler dahil edilene ve Minimum Yayılan Ağaç elde edilene kadar Adım-2 tekrarlanır.

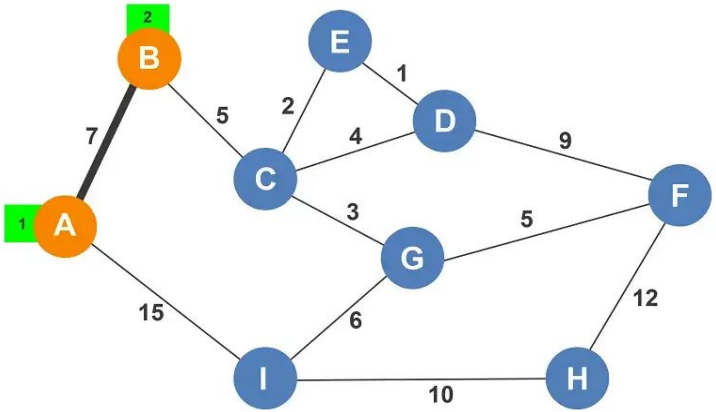
Prim Algoritması

Örnek: Prim Algoritması kullanarak aşağıda verilen graf için minimum kapsayan ağacı ve maliyetini bulalım:



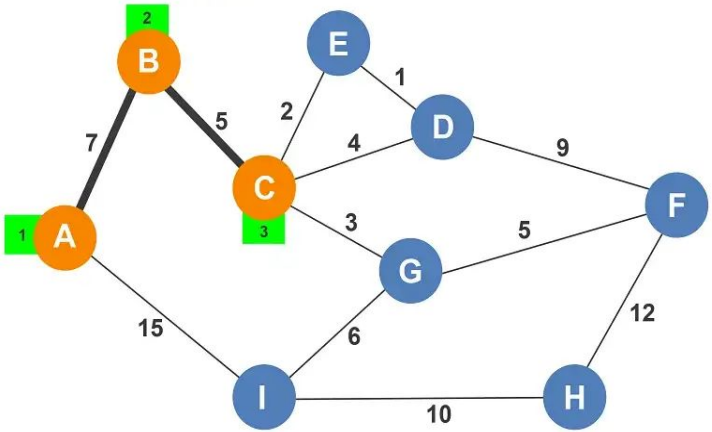
Prim Algoritması

- A tepe noktasını rastgele seçelim. A köşesinin B ve I'ya doğrudan kenarları vardır. B köşesinin maliyeti 7'dir ve I köşesinin maliyeti 15'tir. Prim algoritması açgözlü bir algoritma olduğundan, AB kenarı olan en düşük ağırlığa sahip kenarı seçilir. Yeşil renkli sayılar köşe bulma sırasını temsil eder.



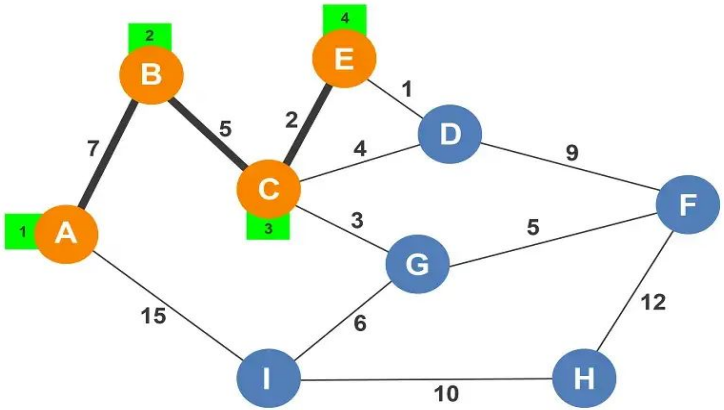
Prim Algoritması

- A ve B köşeleri artık C ve I köşelerini görebilir. B köşesinin C'ye erişimi vardır ve A köşesinin I'ya erişimi vardır. C köşesine bakıldığında, B'den C'ye olan mesafe 5'tir; A ile I arasındaki mesafe 15'tir. Prim algoritması, bu durumda BC'den olan en küçük kenarı seçer.



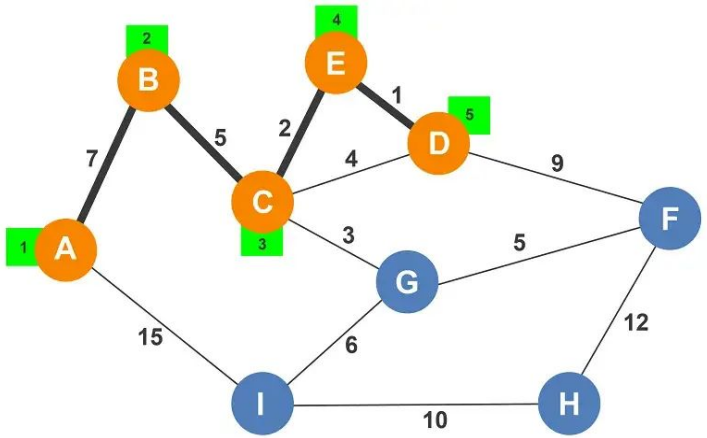
Prim Algoritması

- Algoritma, önceden keşfedilmiş tüm köşelerden görülebilen tüm kenarları dikkate almalıdır. Keşfedilmemiş, görünür kenarlar artık AI, CE, CD ve CG'dir. Prim'in algoritması, her bir kenarın ağırlıklarını karşılaştırarak, en düşük ağırlığın C tepe noktasından E tepe noktasına kadar olduğu sonucuna varır ve bu kenarı seçer.



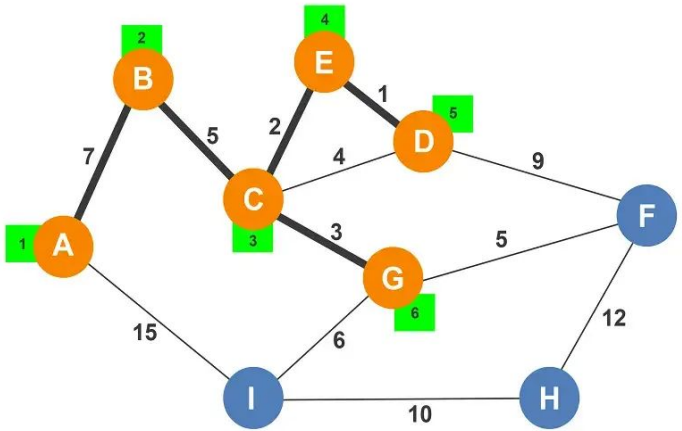
Prim Algoritması

- Algoritmanın artık AI, CD, CG ve ED kenarlarına erişimi var. Edge ED en düşük maliyete sahiptir, bu nedenle algoritma bundan sonra onu seçer.



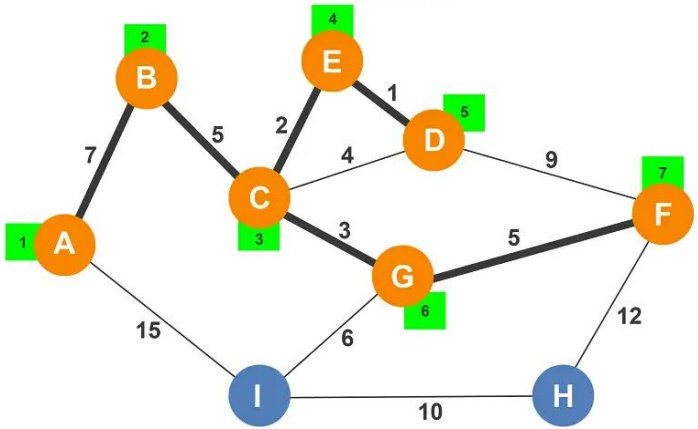
Prim Algoritması

- Prim algoritması artık şu kenarları dikkate almalıdır: AI, CG ve DF. CD kenarı, her iki köşe de bulunduğundan dikkate alınmaz. Bu kenar seçilirse, bir döngü oluşturulur. CG kenarı en küçük ağırlığı içerir, bu nedenle bir sonraki olarak seçilir.



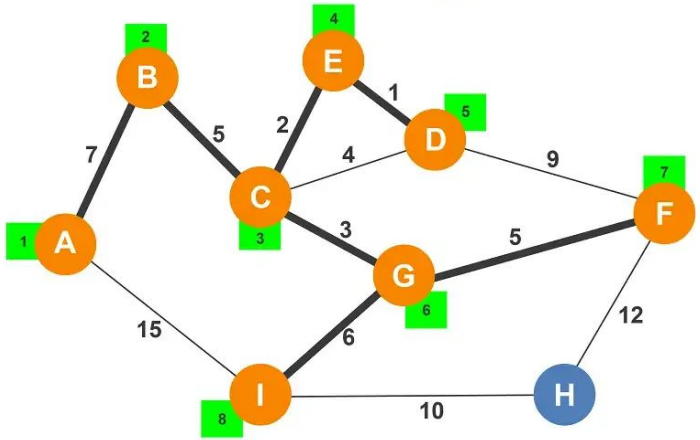
Prim Algoritması

- Keşfedilmemiş üç köşe kaldı. Prim algoritması, AI, DF, GF ve GI kenarlarından en az maliyetli olanı seçecektir. GF en az maliyetli kenarı içerir, bu yüzden bir sonraki olarak seçilir.



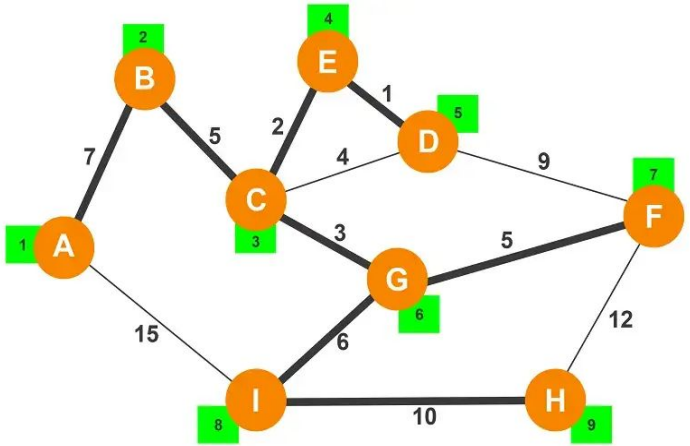
Prim Algoritması

- Prim algoritması AI, FH ve GI kenarlarına bakar. GI en küçük kenar olduğu için, bir sonraki adımda onu seçer.



Prim Algoritması

- Son olarak, H köşesine FH kenarıyla ya da IH kenarı aracılığıyla erişilebilir. IH kenarı daha küçük olduğu için Prim algoritması onu seçer.



- Tüm köşeler bulunduğundan, algoritma sona erer. Minimum kapsayan ağaç bulundu ve ve maliyet;
- **AB + BC + CE + ED + CG + GF + GI + IH : 7 + 5 + 2 + 1 + 3 + 5 + 6 + 10 = 39 olur.**

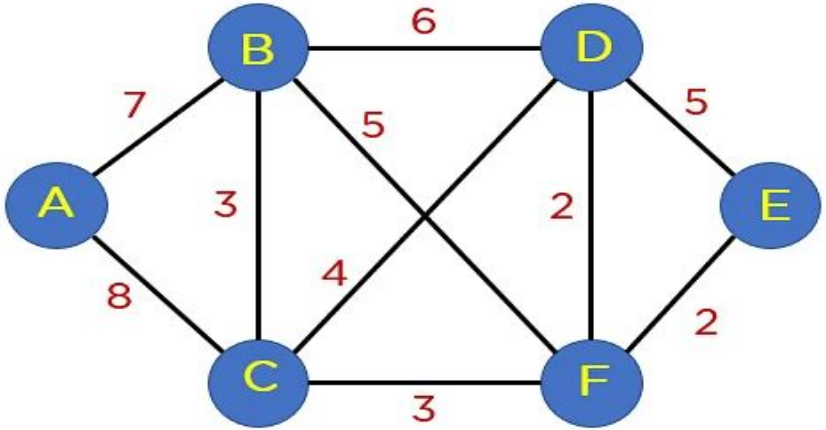
Kruskal Algoritması

Kruskal Algoritmasının uygulanması aşağıdaki adımlardan oluşmaktadır:

- **Adım -1:** Tüm kenarlar, kenar ağırlıklarının artan sırasına göre sıralanır.
- **Adım - 2:** En küçük kenarı seçilir.
- **Adım - 3:** Yeni kenarın, kapsayan ağaçta bir döngü oluşturup oluşturmadığı kontrol edilir.
- **Adım - 4:** Döngü oluşturmuyorsa, bu kenar minimum kapsayan ağaca dahil edilir. Aksi takdirde, dahil edilmez.
- **Adım - 5:** Tüm köşeler dahil edilene kadar Adım - 2 tekrarlanır.

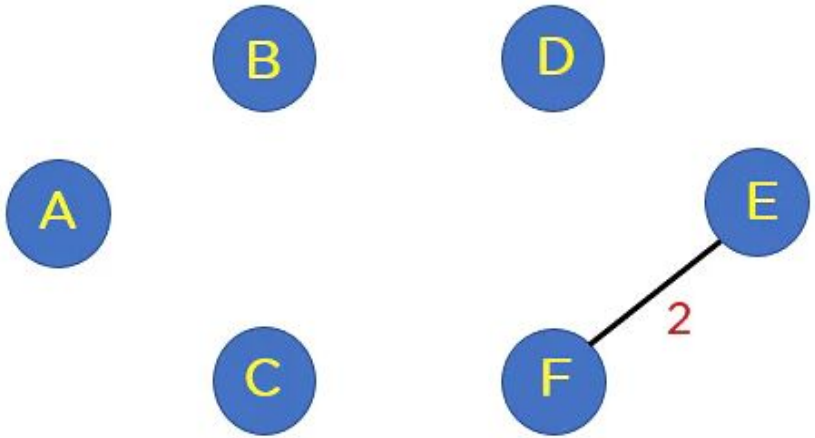
Kruskal Algoritması

Örnek: Kruskal Algoritması kullanarak aşağıda verilen graf için minimum kapsayan ağacı ve maliyetini bulalım:



Kruskal Algoritması

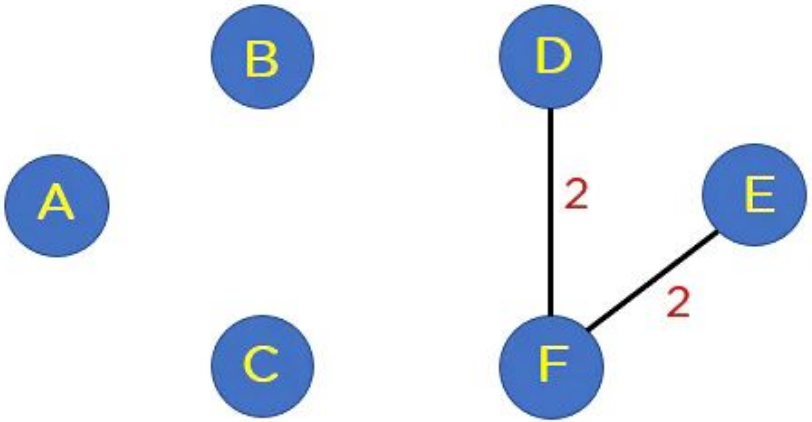
Örnek: Kruskal Algoritması kullanarak aşağıda verilen graf için minimum kapsayan ağacı ve maliyetini bulalım:



Kenar	Maliyet	Kenar	Maliyet
EF	2	DC	4
DF	2	BF	5
BC	3	BD	6
CF	3	AB	7
		AC	8

Kruskal Algoritması

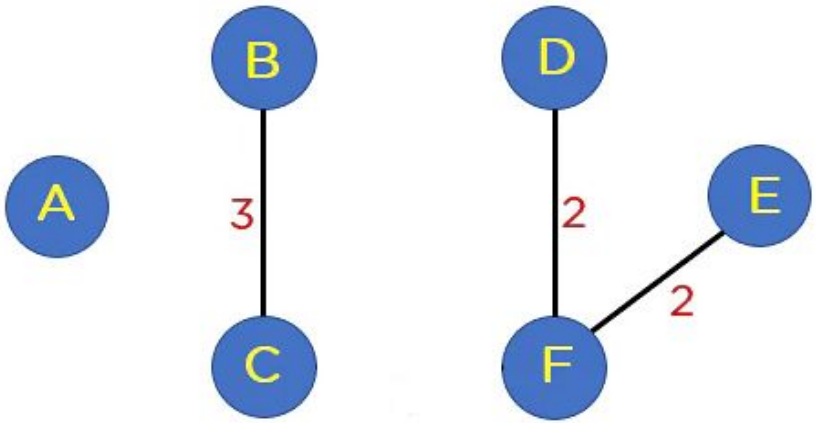
Örnek: Kruskal Algoritması kullanarak aşağıda verilen graf için minimum kapsayan ağacı ve maliyetini bulalım:



Kenar	Maliyet	Kenar	Maliyet
EF	2	DC	4
DF	2	BF	5
BC	3	BD	6
CF	3	AB	7
		AC	8

Kruskal Algoritması

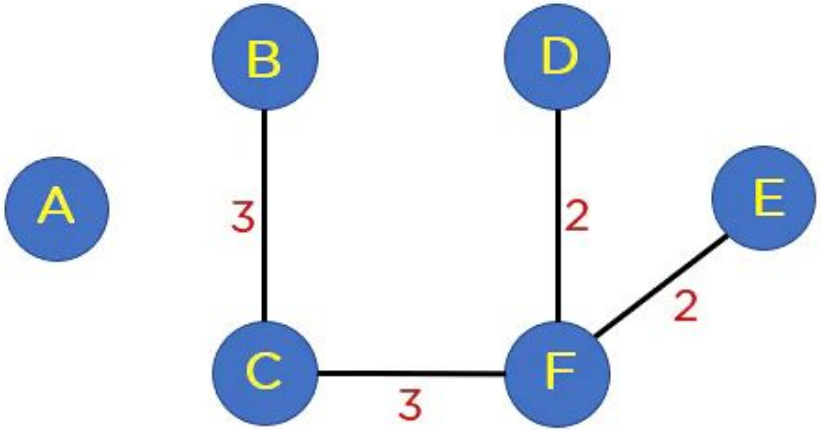
Örnek: Kruskal Algoritması kullanarak aşağıda verilen graf için minimum kapsayan ağacı ve maliyetini bulalım:



Kenar	Maliyet	Kenar	Maliyet
EF	2	DC	4
DF	2	BF	5
BC	3	BD	6
CF	3	AB	7
		AC	8

Kruskal Algoritması

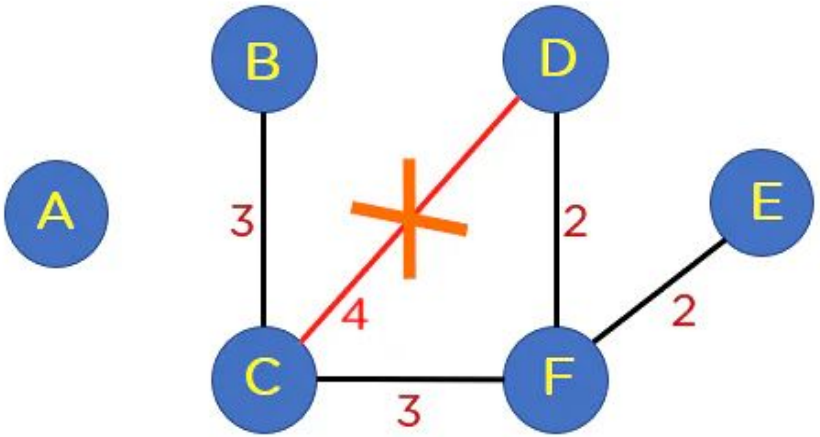
Örnek: Kruskal Algoritması kullanarak aşağıda verilen graf için minimum kapsayan ağacı ve maliyetini bulalım:



Kenar	Maliyet	Kenar	Maliyet
EF	2	DC	4
DF	2	BF	5
BC	3	BD	6
CF	3	AB	7
		AC	8

Kruskal Algoritması

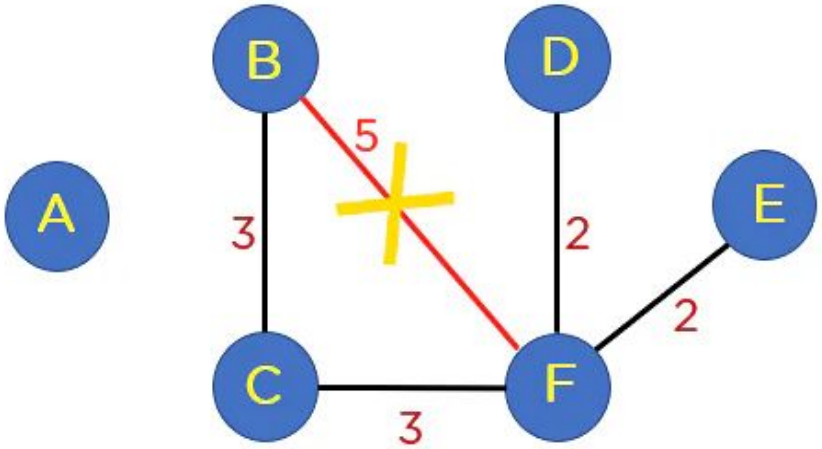
Örnek: Kruskal Algoritması kullanarak aşağıda verilen graf için minimum kapsayan ağacı ve maliyetini bulalım:



Kenar	Maliyet	Kenar	Maliyet
EF	2	DC	4
DF	2	BF	5
BC	3	BD	6
CF	3	AB	7
		AC	8

Kruskal Algoritması

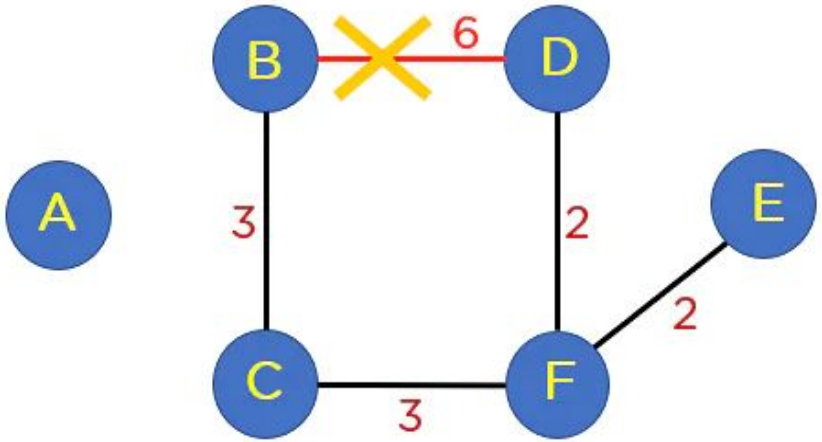
Örnek: Kruskal Algoritması kullanarak aşağıda verilen graf için minimum kapsayan ağacı ve maliyetini bulalım:



Kenar	Maliyet	Kenar	Maliyet
EF	2	DC	4
DF	2	BF	5
BC	3	BD	6
CF	3	AB	7
		AC	8

Kruskal Algoritması

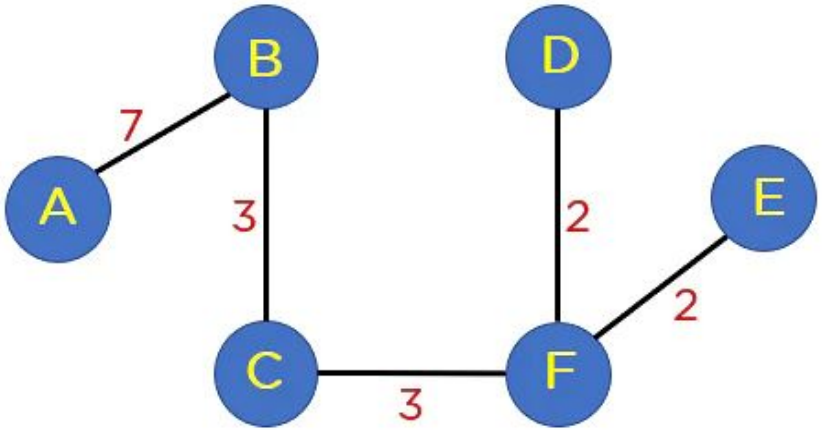
Örnek: Kruskal Algoritması kullanarak aşağıda verilen graf için minimum kapsayan ağacı ve maliyetini bulalım:



Kenar	Maliyet	Kenar	Maliyet
EF	2	DC	4
DF	2	BF	5
BC	3	BD	6
CF	3	AB	7
		AC	8

Kruskal Algoritması

Örnek: Kruskal Algoritması kullanarak aşağıda verilen graf için minimum kapsayan ağacı ve maliyetini bulalım:



Minimum Kapsayan Ağaç

Kenar	Maliyet	Kenar	Maliyet
EF	2	DC	4
DF	2	BF	5
BC	3	BD	6
CF	3	AB	7
		AC	8

Kaynaklar

- **Kenneth Rosen,** “Discrete Mathematics and Its Applications”, 7th Edition , McGraw Hill Publishing Co., 2012.
- **Milos Hauskrecht,** “Discrete Mathematics for Computer Science”, University of Pittsburgh, Ders Notları.
- https://levelup.gitconnected.com/prims-algorithm-visually-explained-fb69a786f352