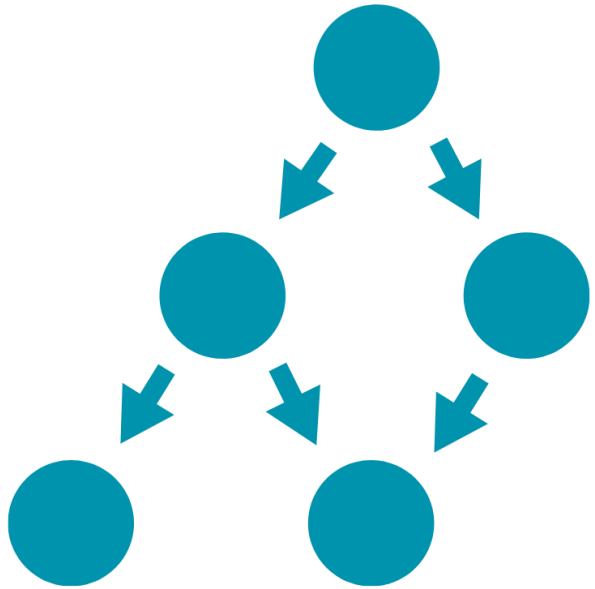
BTK AKADEMİ

ZAFER CÖMERT
Öğretim Üyesi



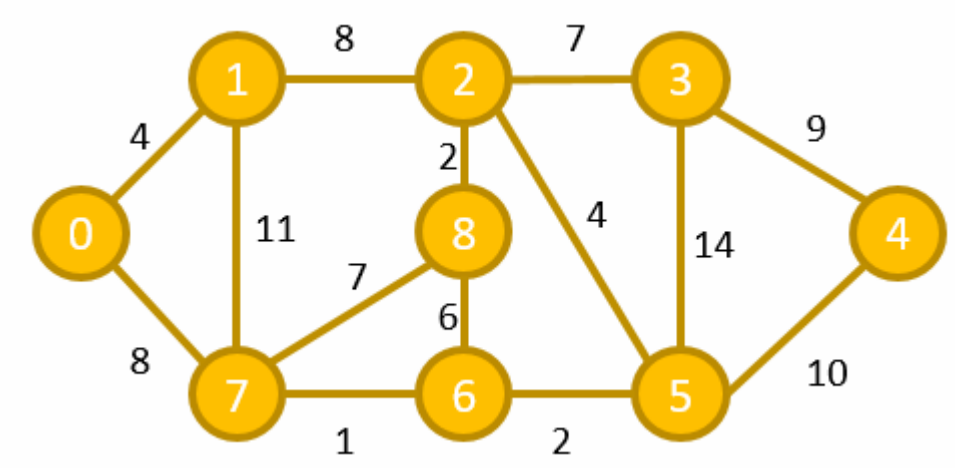
VERİ YAPILARILARI VE ALGORİTMALAR

Graph
Prim’s Algorithm

Minimum Kapsama Ağaçları (Minimum Spanning Tree)

- **Kapsama/Yayılım Ağacı (Spanning Tree)**
  Çizgenin tüm düğümlerini içeren (bağlı, çevrimsiz) bir ağaçtır.

- **Minimum Kapsama Ağacı (Minimum Spanning Tree)**
  Ağırlıkların toplamının en küçük olduğu kapsama ağacıdır.

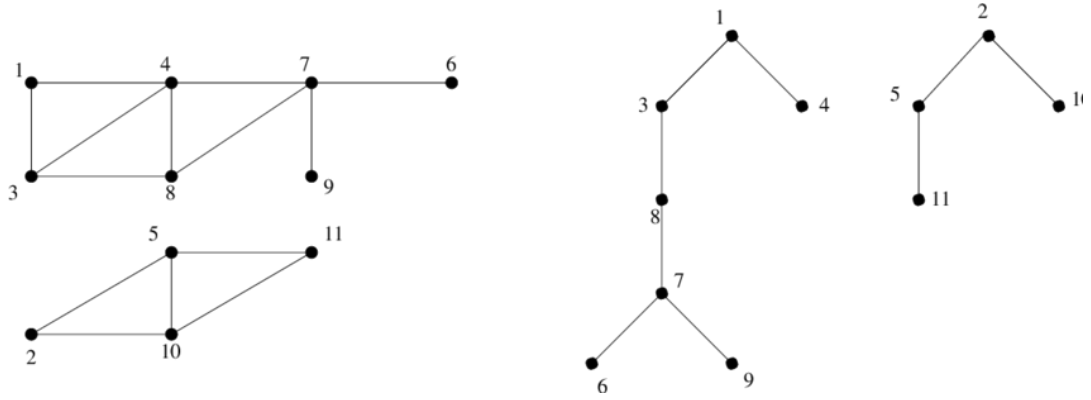


BTK AKADEMİ

Minimum Kapsama Ağaçları (MST, Minimum Spanning Tree)

- **Kapsama ormanı (Yayılan orman / Spanning forest)**
- Bir çizge bağlantılı (connected) değilse; orada çizgenin her bileşeni için bir kapsama ağacı vardır.

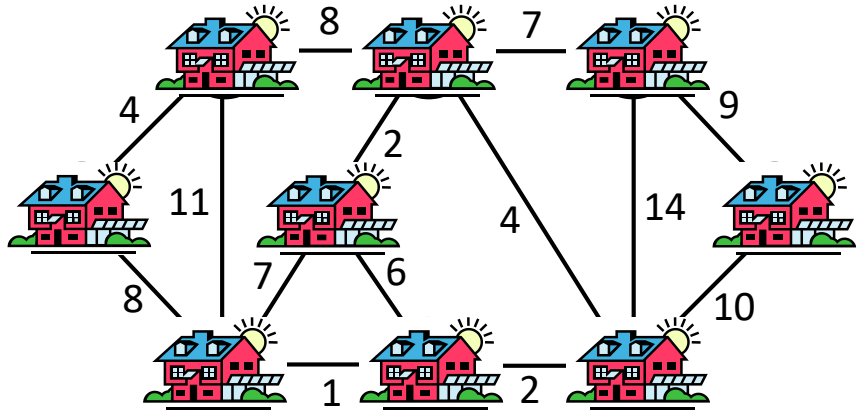
- Bir dizi şehir, terminaller ya da bilgisayar arası en ucuz maliyetli yolu bulmak üzere minimum kapsama ağaçları kullanılabilir.



Minimum Kapsama Ağaçları

- Bir ağırlıklı, bağlı, yönsüz çizgede: düğümler (vertex) ev ve kenarlar (edges) yol olarak modellendiğinde;
- Ağırlık : Her kenar üzerinde $weight(u, v)$ $edge(u, v) \in \mathrm{E}$
- Bul $T \subseteq \mathrm{E}$ ise;

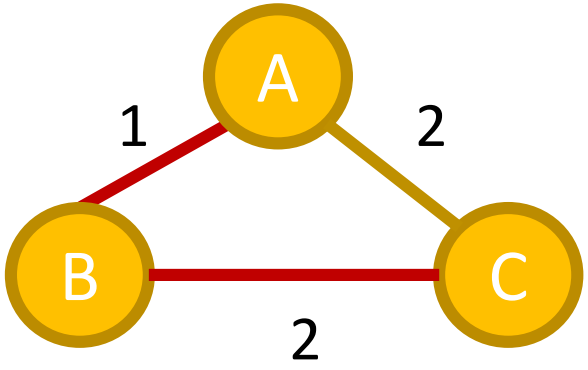
1. T tüm düğümleri bağlar
2. $w(T) = \sum_{(u,v)\in T} w(u, v)$ minimize edilir.

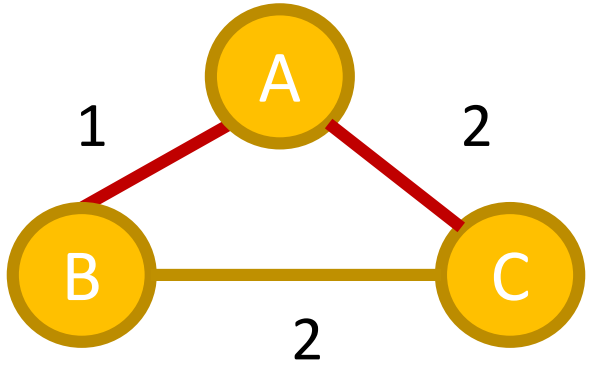


BTK AKADEMİ

Minimum Kapsama Ağaçları

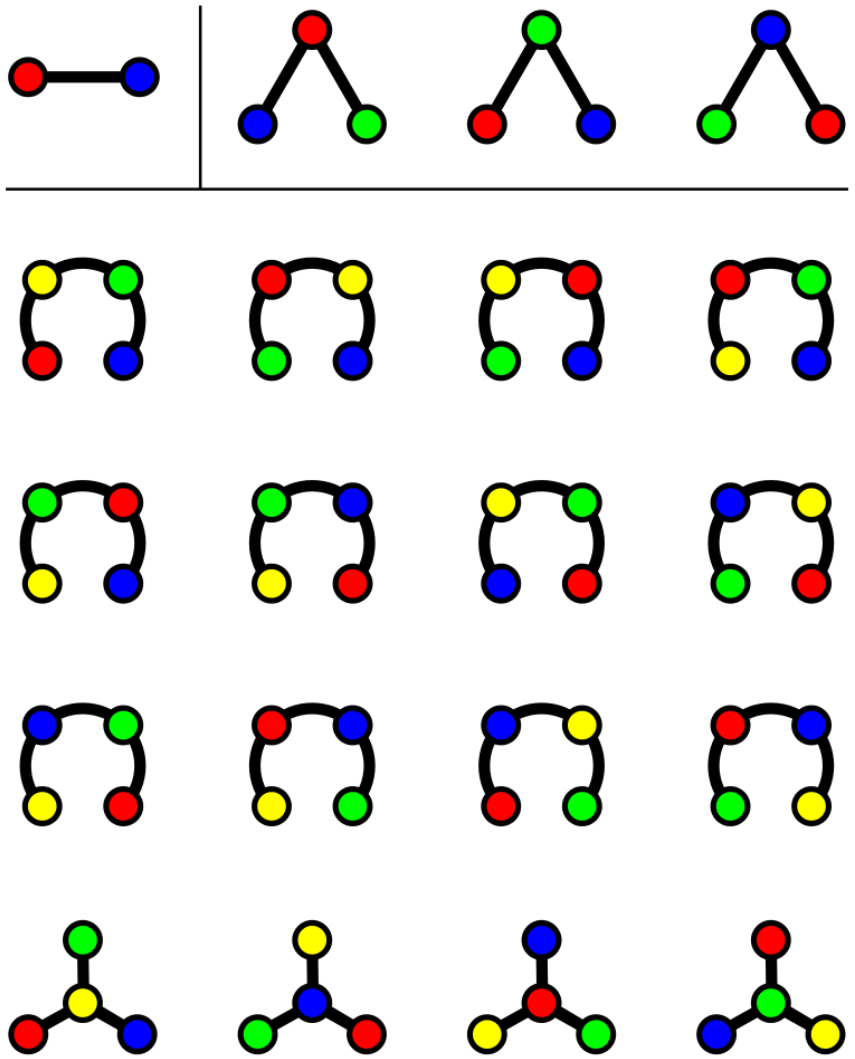
- Kapsama ağaçları benzersiz değildir.





Minimum Kapsama Ağaçları

- Kapsama ağaçlarda çevrim (cycle) yoktur.
- Bir döngünün/çevrimin kenarını kaldırabiliriz ve yine de maliyeti düşürürken ilgili düğüm kapsama ağacına bağlı kalabilir.
- MST kenar sayısı: $|V| - 1$



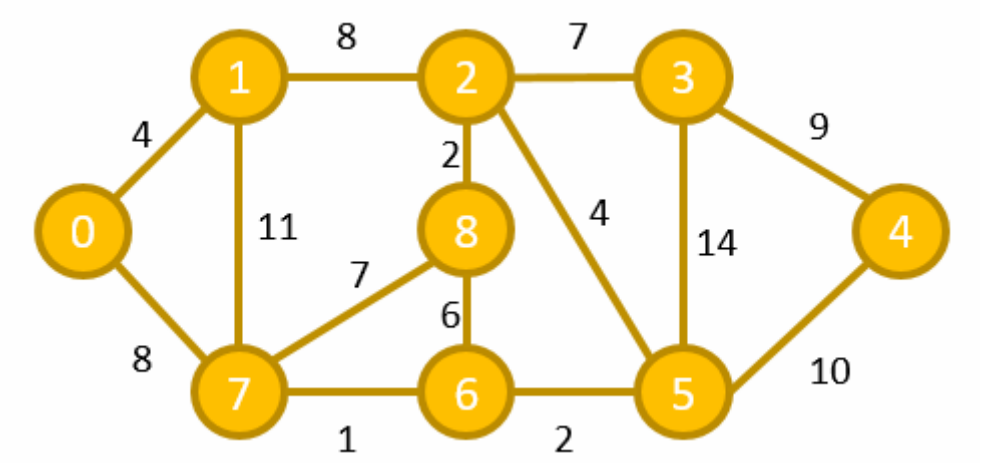
BTK AKADEMİ

Minimum Kapsama Ağaçları

```
A ← ∅

while A is not a spanning tree

    do

        find an edge (u, v) that is safe for A,

        A ← A ∪ {(u, v)}

return A
```

Prim’s Algoritması

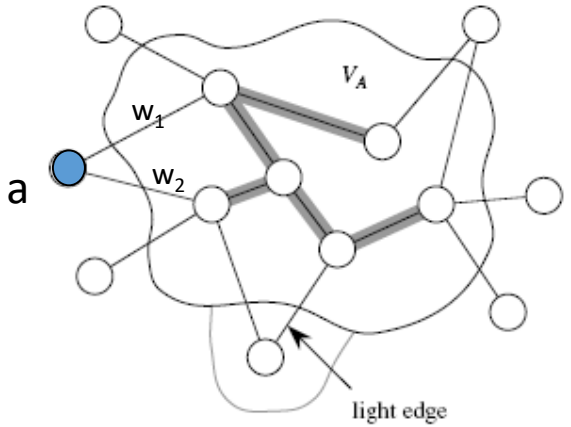
- A içerisindeki kenarlar tek bir ağaç oluşturur.
- Her hangi bir kökten (root) başlayarak; $V_A = \{a\}$
- Her bir adımda:
  - Düşük maliyetli kenar bulunur: $(V_A, V - V_A)$
  - Bu kenar A’ya eklenir.
  - Ağaç tüm düğümleri kapsayıncaya kadar bu işleme devam edilir.



Düşük maliyetli kenarı bulma

- Öncelikli kuyruk yapısı (Q) kullanın:
  - Henüz ağaca dahil edilmemiş düğümleri içerir. Yani: $(V - V_A)$
  - V_A = {a}, Q = {b, c, d, e, f, g, h, i}

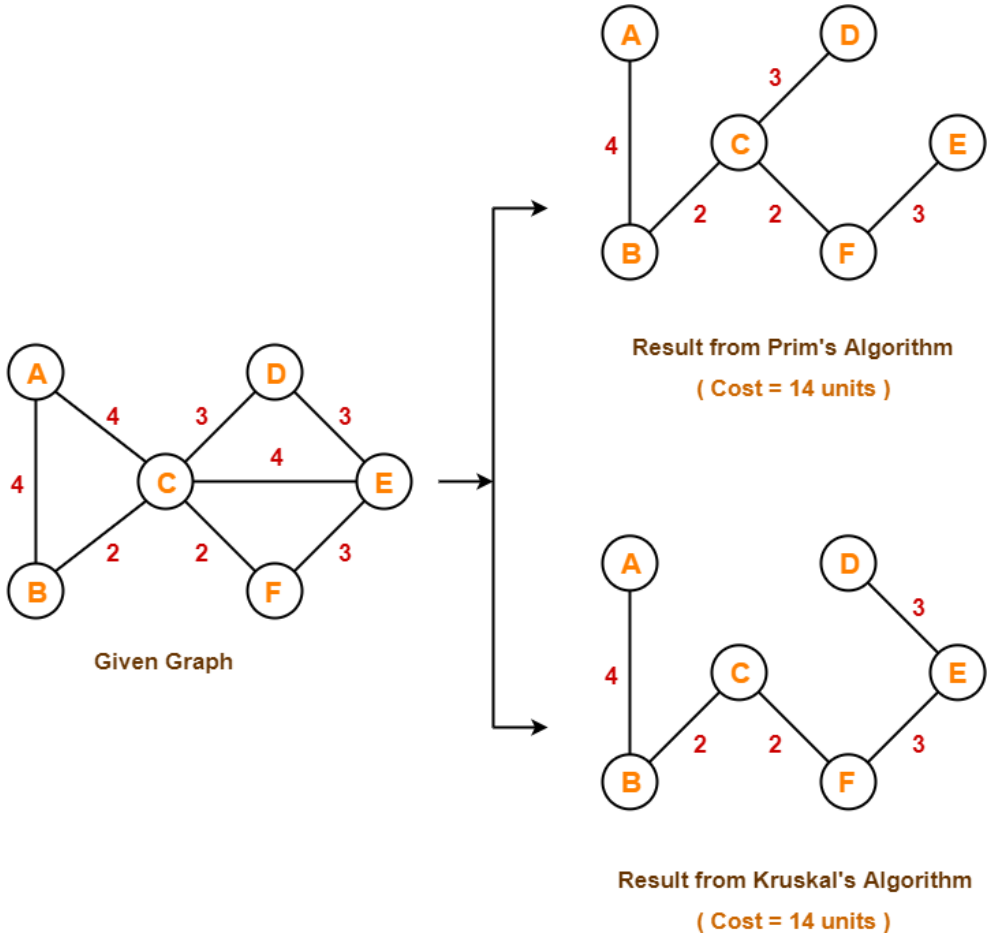
- Birleştirme işlemi her düğümün (v) anahtarına (değer) göre yapılır:
  - key[v]: Herhangi bir kenarın (u,v) minimum ağırlığı ile V_A bağlanması.
  - **Key[a]=min(w_1,w_2)**



Prim(V,E,w,r)

```
1.   Q ← ∅
2.   for each u ∈ V
3.       do key[u] ← ∞
4.          π[u] ← NIL
5.          INSERT(Q, u)
6.   DECREASE-KEY(Q, r, 0)        ► key[r] ← 0
7.   while Q ≠ ∅
8.          do u ← EXTRACT-MIN(Q)
9.             for each v ∈ Adj[u]
10.               do if v ∈ Q and w(u, v) < key[v]
11.                  then π[v] ← u
12.                       DECREASE-KEY(Q, v, w(u, v))
```

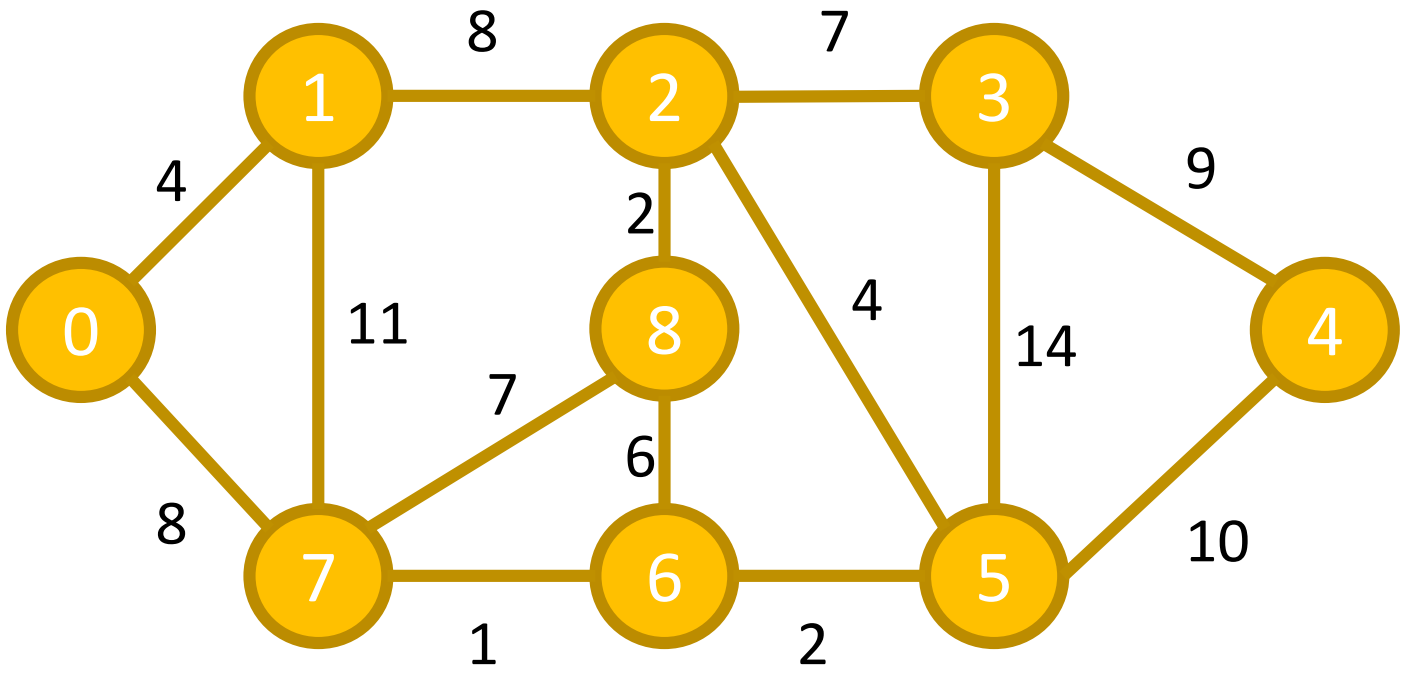
Prim’s Algoritması



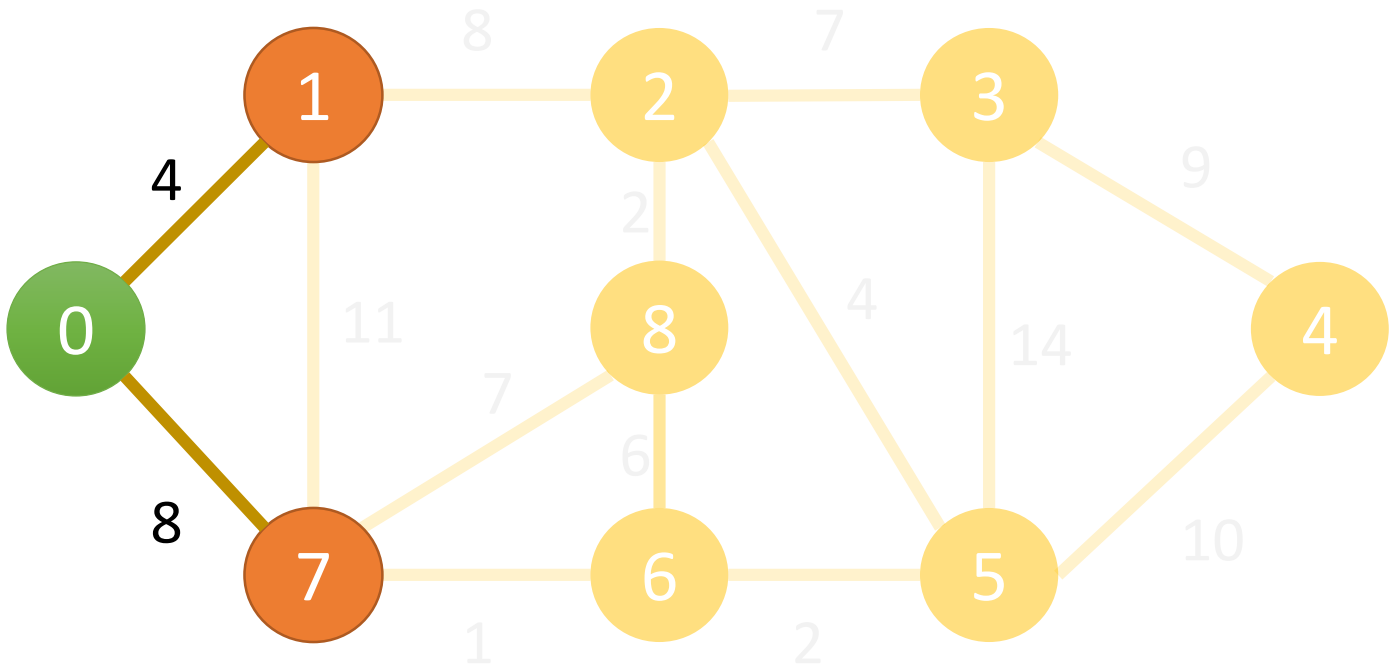
- Prim’s algoritması açgözlü (greedy) algoritmadır.
  - Açgözlü algoritmalar, her adımda «yerel olarak» optimal olan bir dizi seçime dayalı çözümler bulur.
- Yinede, Prim’s algoritmasının stratejisi global optimum çözümü sağlar.

BTK AKADEMİ

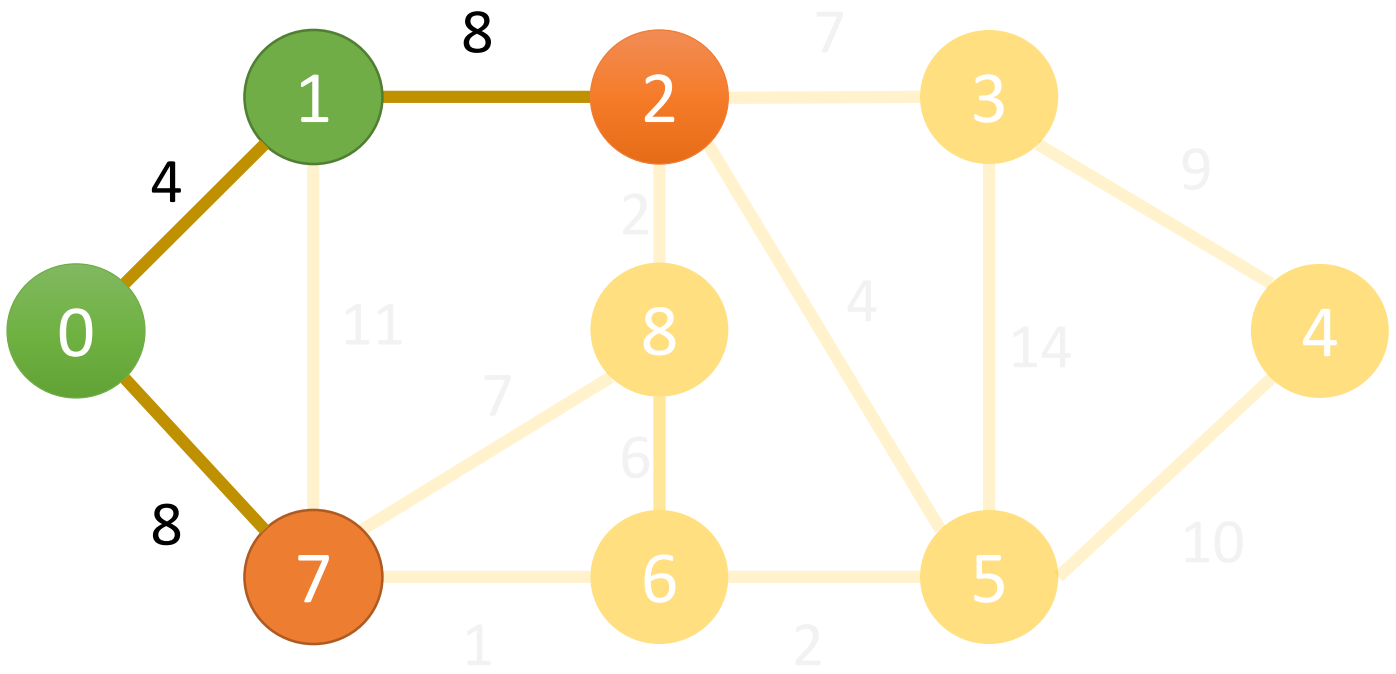
Prim’s Algoritması



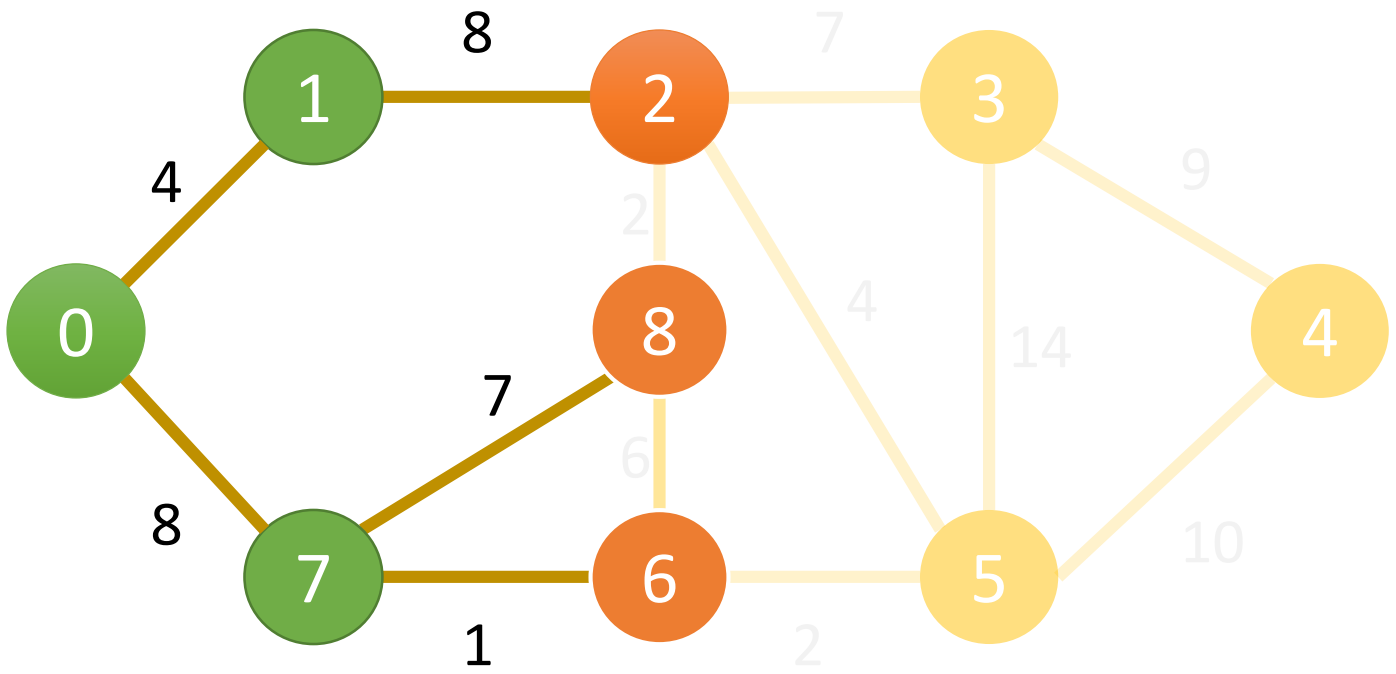
Prim’s Algoritması



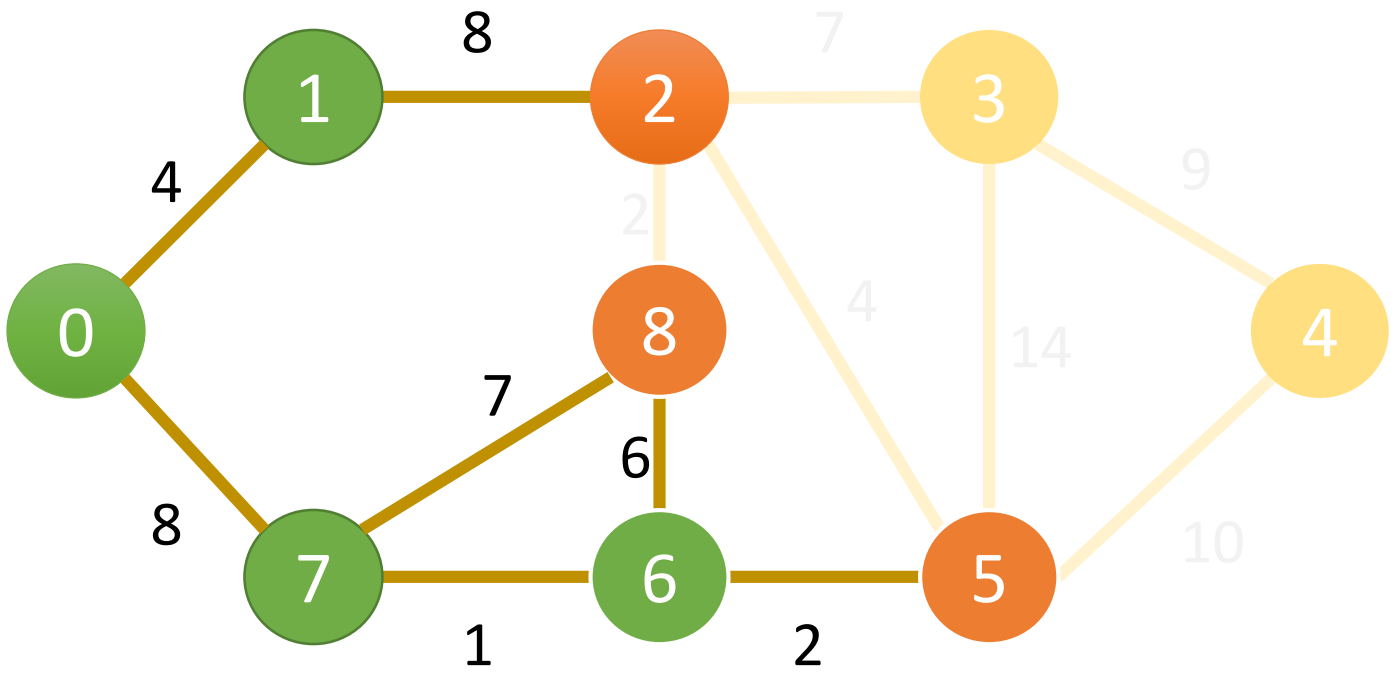
Prim’s Algoritması



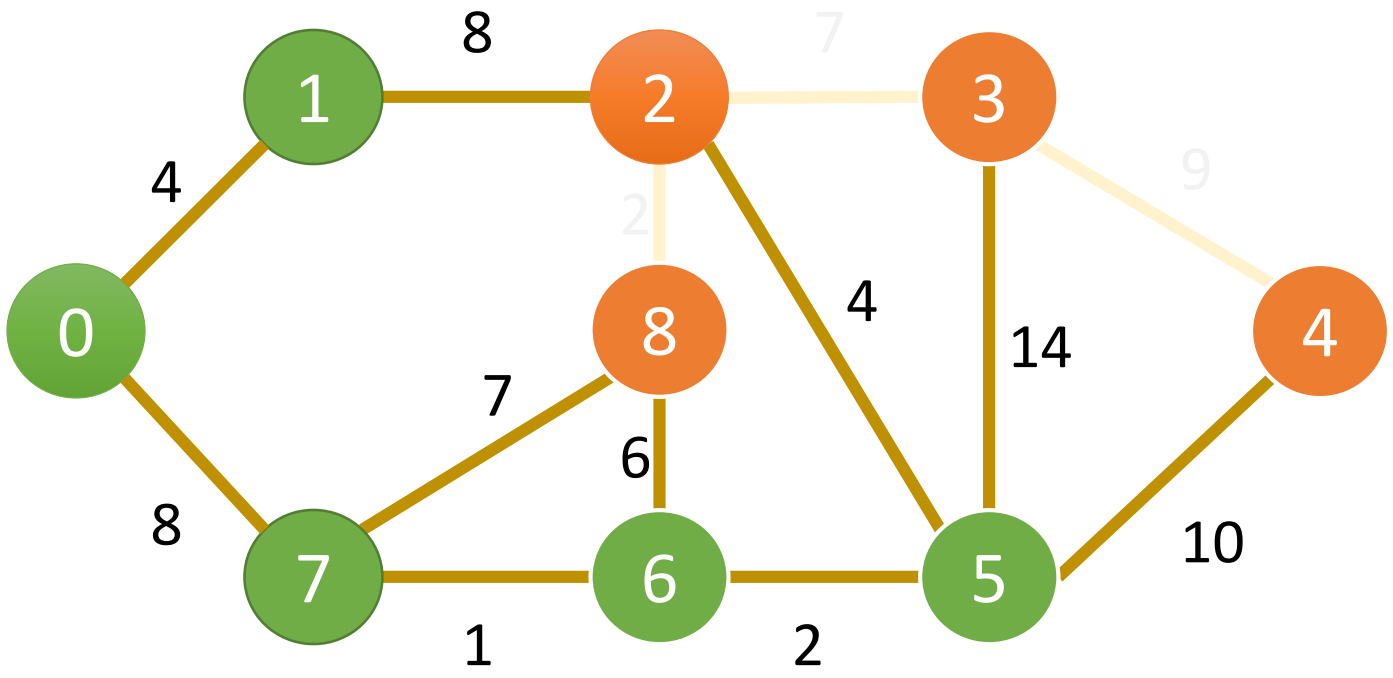
Prim’s Algoritması



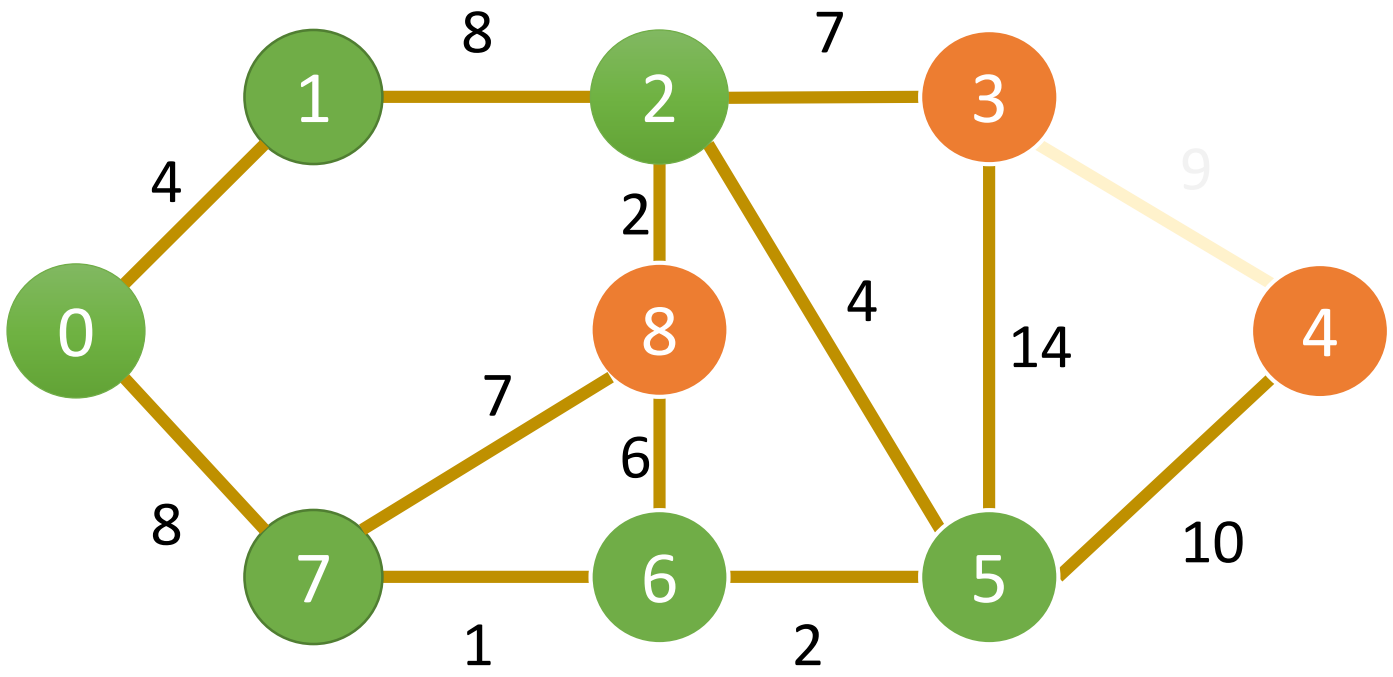
Prim’s Algoritması



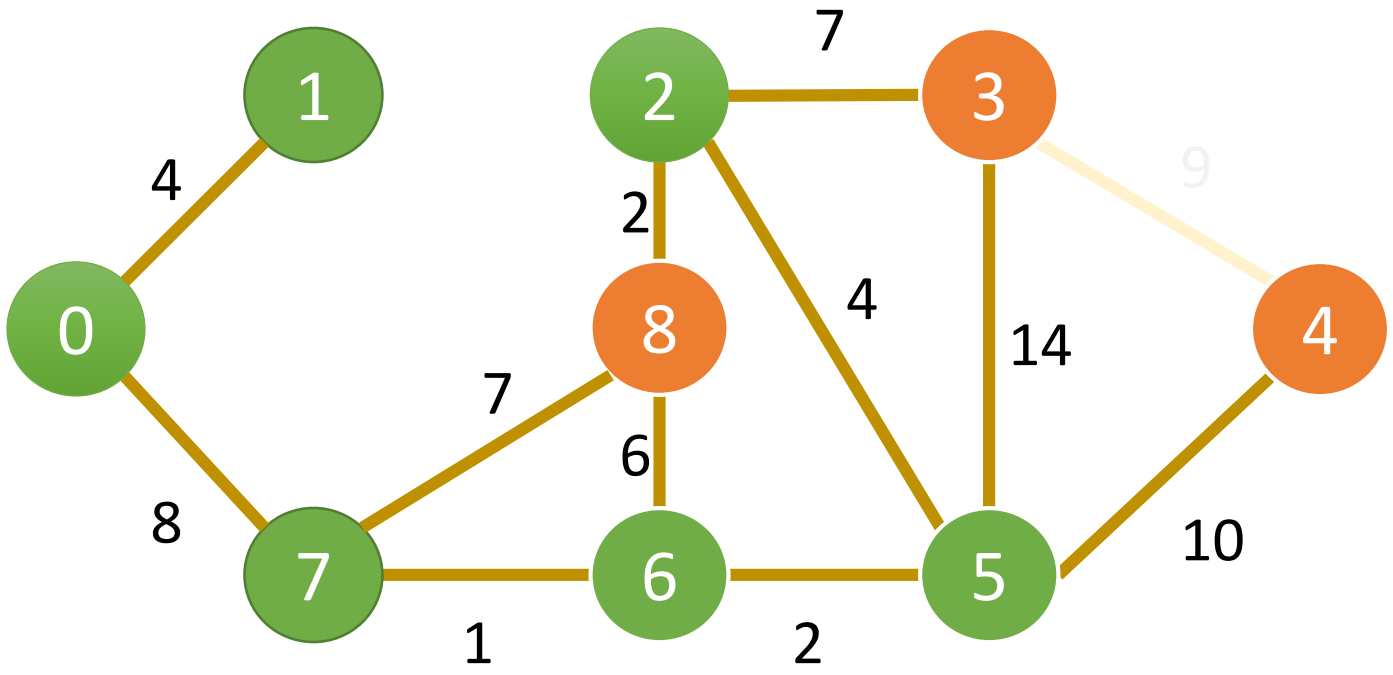
Prim’s Algoritması



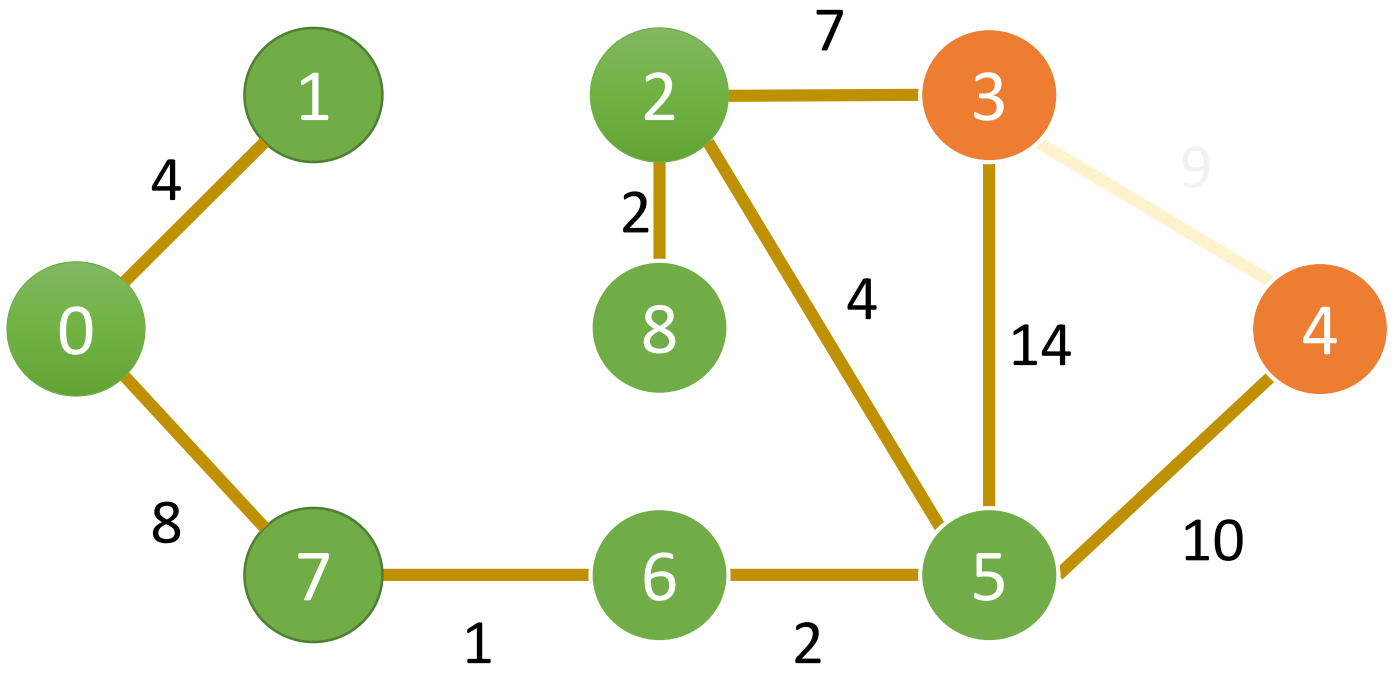
Prim’s Algoritması



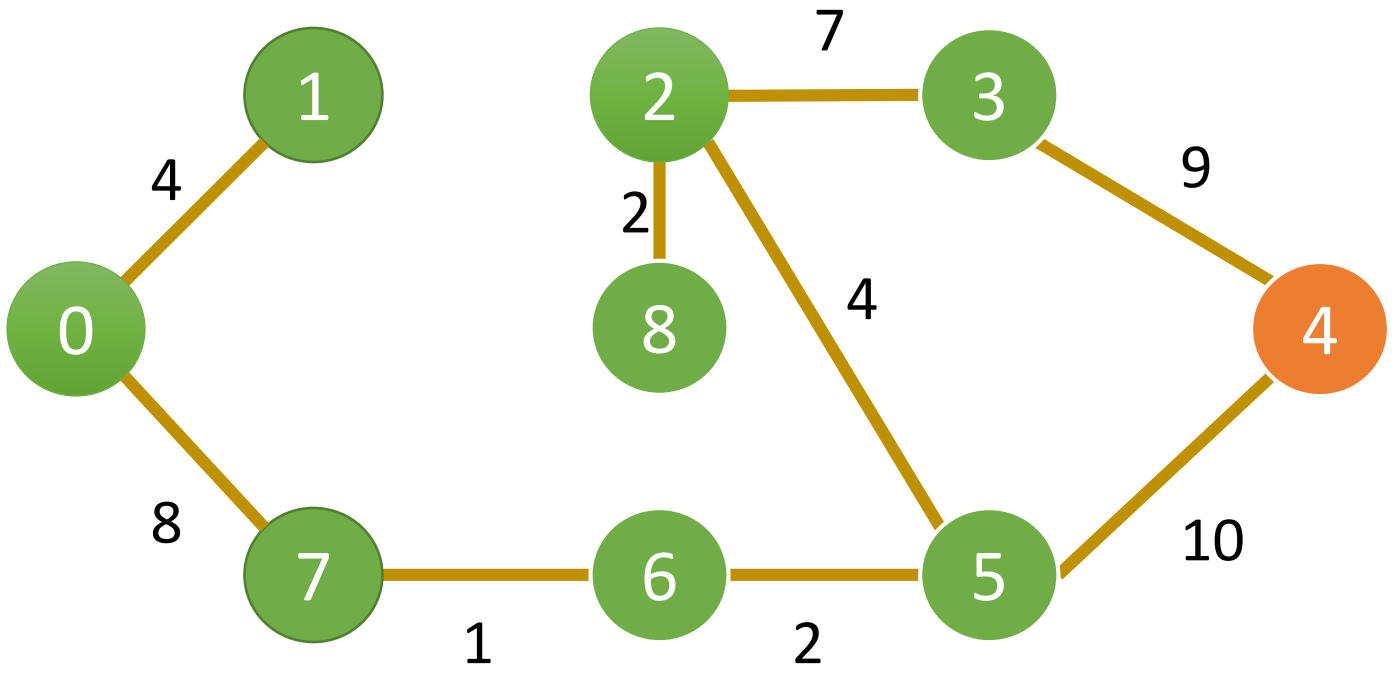
Prim’s Algoritması



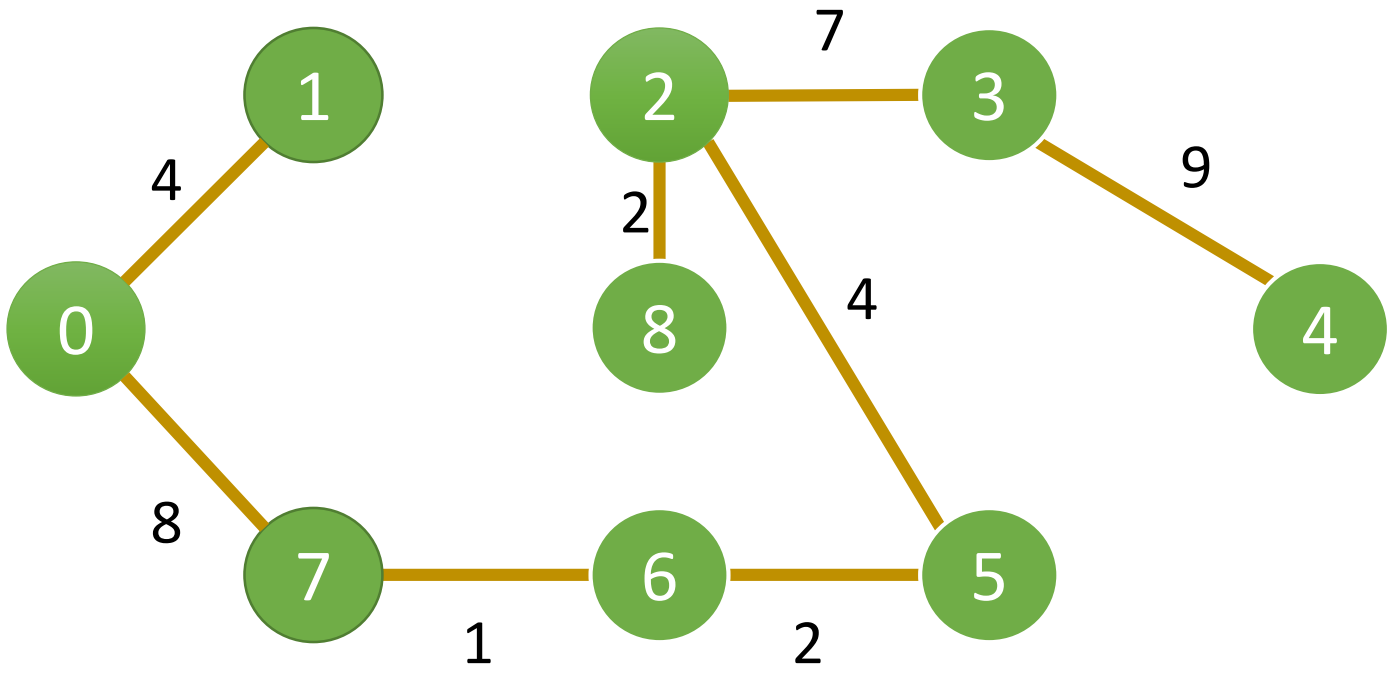
Prim’s Algoritması



Prim’s Algoritması

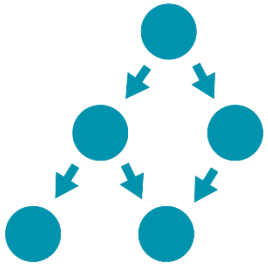


Prim’s Algoritması



Prim’s Algoritması Her Zaman Çalışır mı?

- Prim’s algoritması açgözlü bir algoritmadır.
- MST’ye bir kenar eklemeye karar verdiğinde; asla kararını sorgulamaz.
- Açgözlü algoritmalar nadiren çalışır.
- MST’lerin açgözlü algoritmalar ile çözülmesine imkan veren özellikleri bulunmaktadır. Hatta bu özellikleri nedeniyle farklı açgözlü algoritmalar MST üzerinde çözüm sunabilmektedir.



Veri Yapıları ve Algoritmalar

ZAFER CÖMERT
Öğretim Üyesi

BTK AKADEMİ