CENG 213 Veri Yapıları 10: Çizge Algoritmaları(Graph Algorithms)

Öğr.Gör. Şevket Umut ÇAKIR

Pamukkale Üniversitesi

Hafta 10

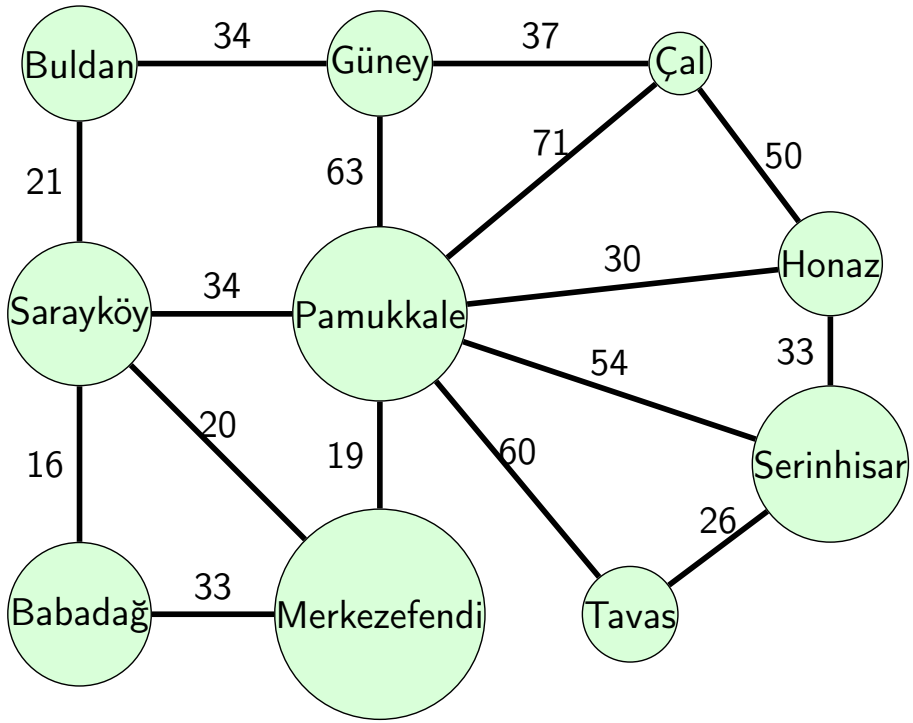
Anahat

1. Minimum Açılım Ağaçları(Minimum Spanning Trees)
   - Kruskal'ın MST Algoritması
   - Prim'in MST Algoritması
2. En Kısa Yol
   - Dijkstra'nın Algoritması

Ağırlıklı Çizgeler/Weighted Graphs

Tanım

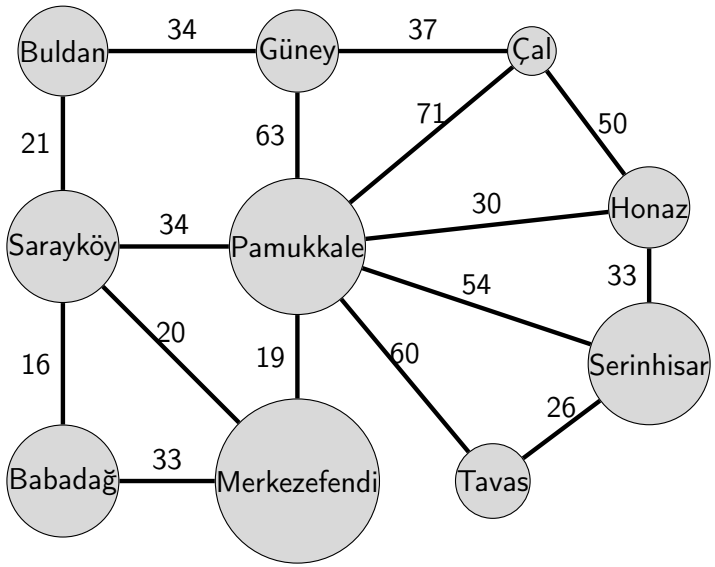
Kenarları üzerinde ağırlık değerleri bulunan çizgelere **ağırlıklı çizge(weighted graphs)** adı verilir.



Minimum Açılım Ağaçları/Minimum Spanning Trees

Tanım

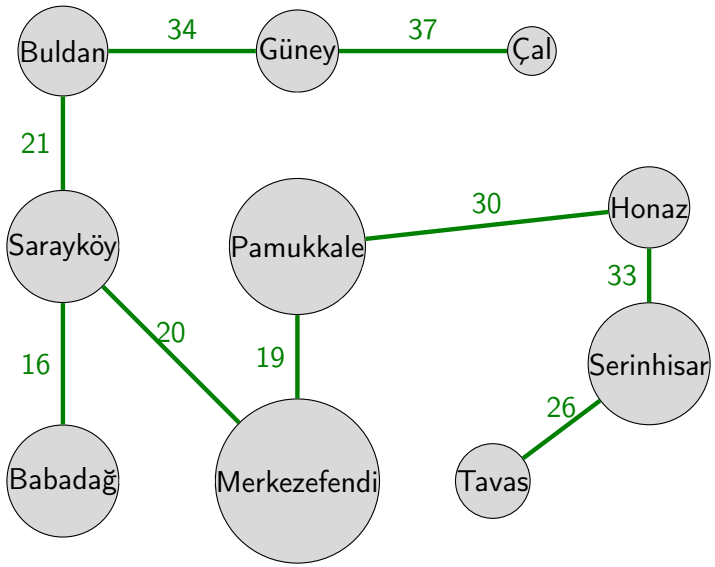
Açılım ağaçları bağlı ve yönsüz bir çizgede bütün düğümleri birbirine bağlayan ağaç yapısındaki bir alt çizgedir. **Minimum açılım ağacı(minimum spanning tree, MST)**, açılım ağaçları içinde toplam ağırlığı en az olan ağaçtır.



Minimum Açılım Ağaçları/Minimum Spanning Trees

Tanım

Açılım ağaçları bağlı ve yönsüz bir çizgede bütün düğümleri birbirine bağlayan ağaç yapısındaki bir alt çizgedir. **Minimum açılım ağacı(minimum spanning tree, MST)**, açılım ağaçları içinde toplam ağırlığı en az olan ağaçtır.



Minimum Açılım Ağaçları/Minimum Spanning Trees

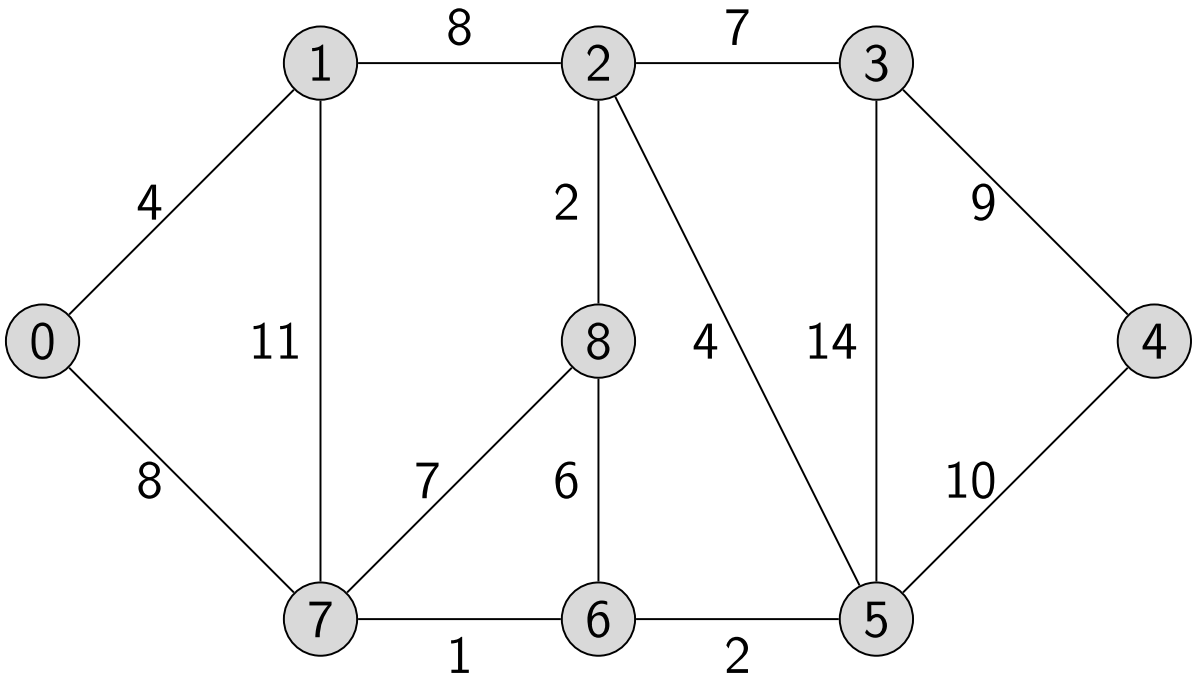
- Minimum açılım ağaçlarını bulmak için çeşitli yöntemler mevcuttur
- En bilinenleri Prim'in ve Kruskal'ın algoritmalarıdır.
- Minimum açılım ağacı $V - 1$ tane kenar içerir

Kruskal'ın MST Algoritması

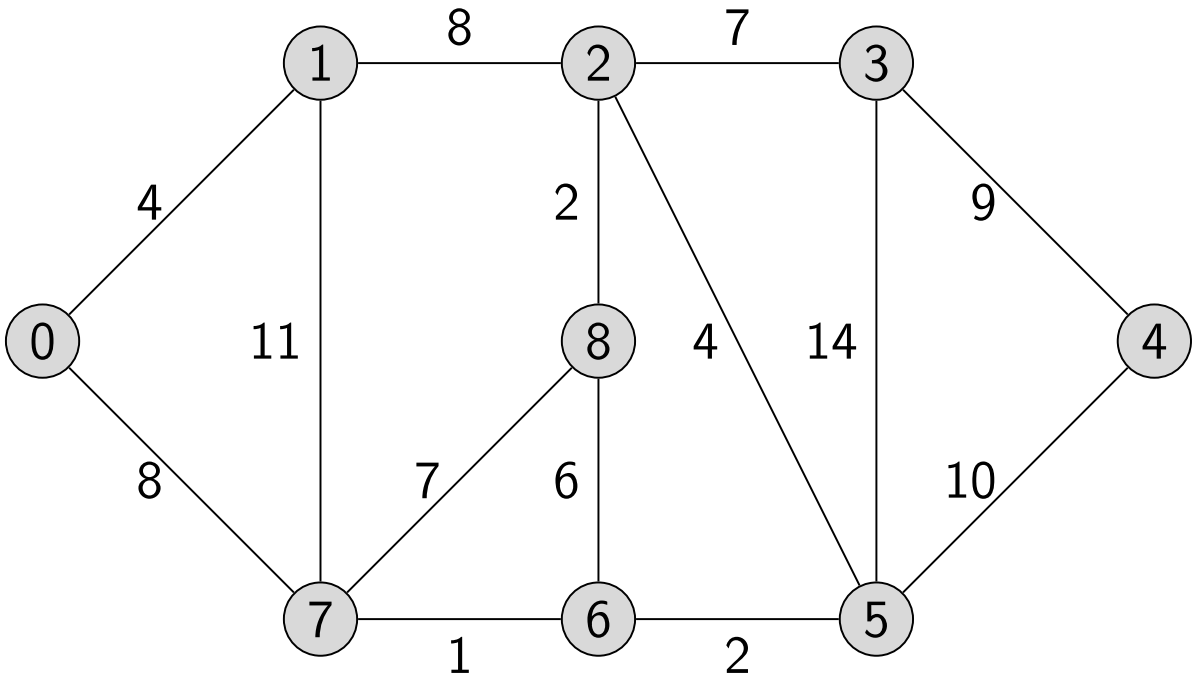
1. Kenarları ağırlıklarına göre artan sırada sırala
2. En küçük kenarı ele al. Eğer mevcut seçilen kenarlarla bir döngü/çevrim(cycle) içermiyrsa kenarı seç, aksi takdirde kenarı bırak
3. Kapsayan ağaçta $V - 1$ tane düğüm olana kadar Adım 2'yi tekrarla

Şekil: Kruskal'ın MST Algoritması

Kruskal Örnek

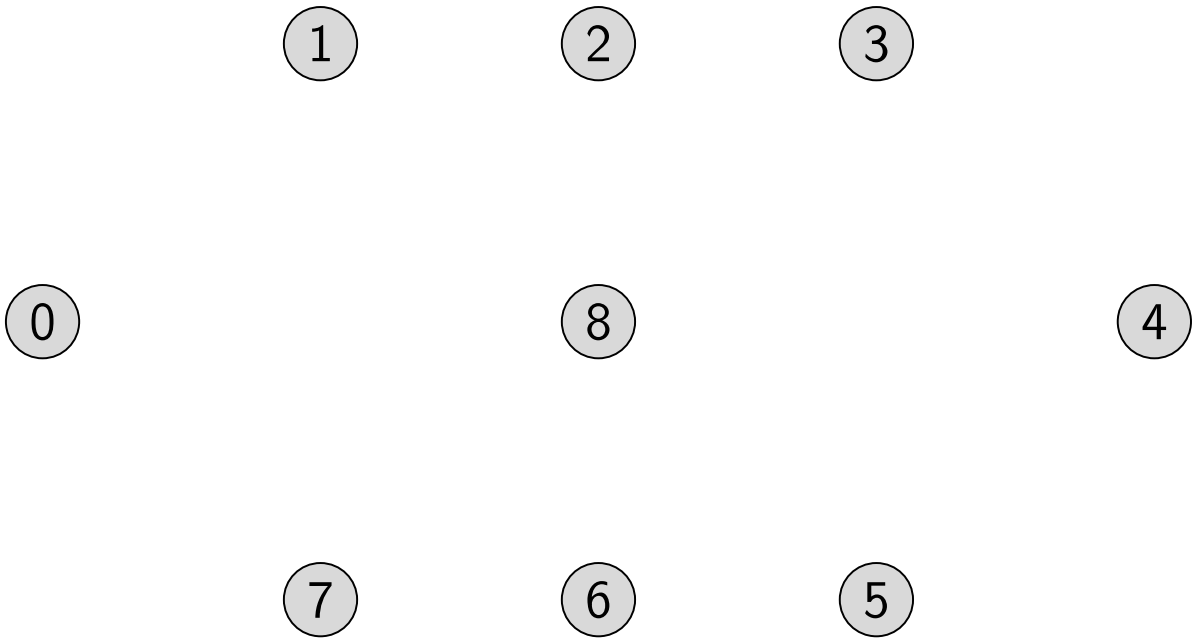


Kruskal Örnek



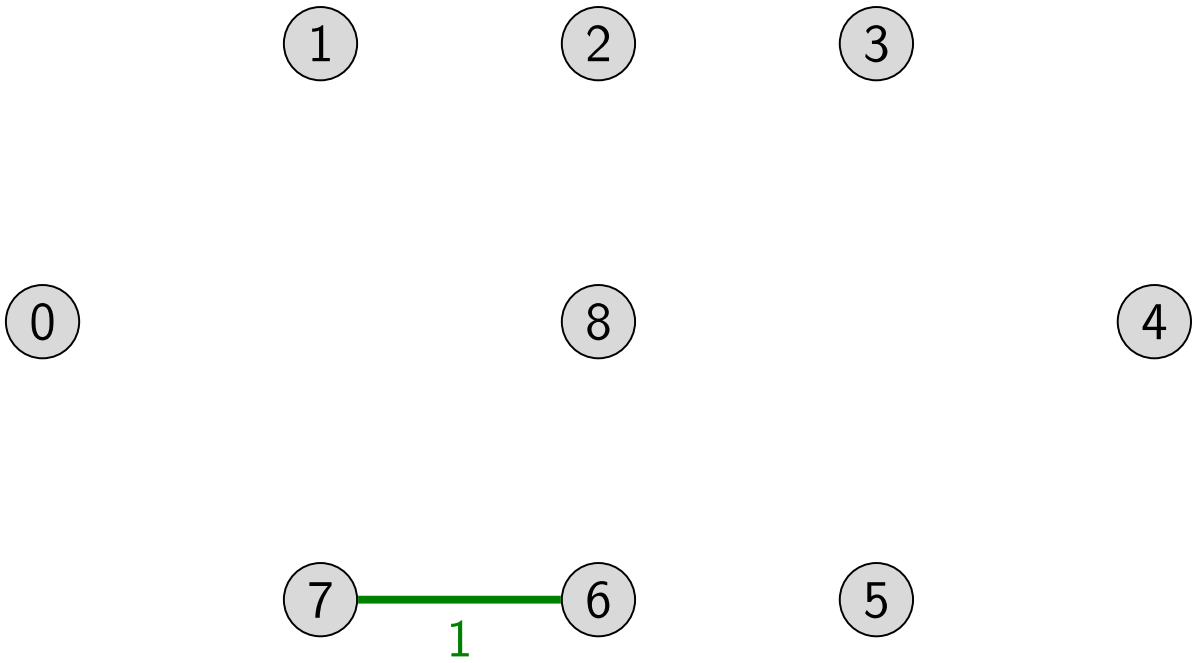
Ağırlık	Hedef	Kaynak
1	7	6
2	8	2
2	6	5
4	0	1
4	2	5
6	8	6
7	2	3
7	7	8
8	0	7
8	1	2
9	3	4
10	5	4
11	1	7
14	3	5

Kruskal Örnek



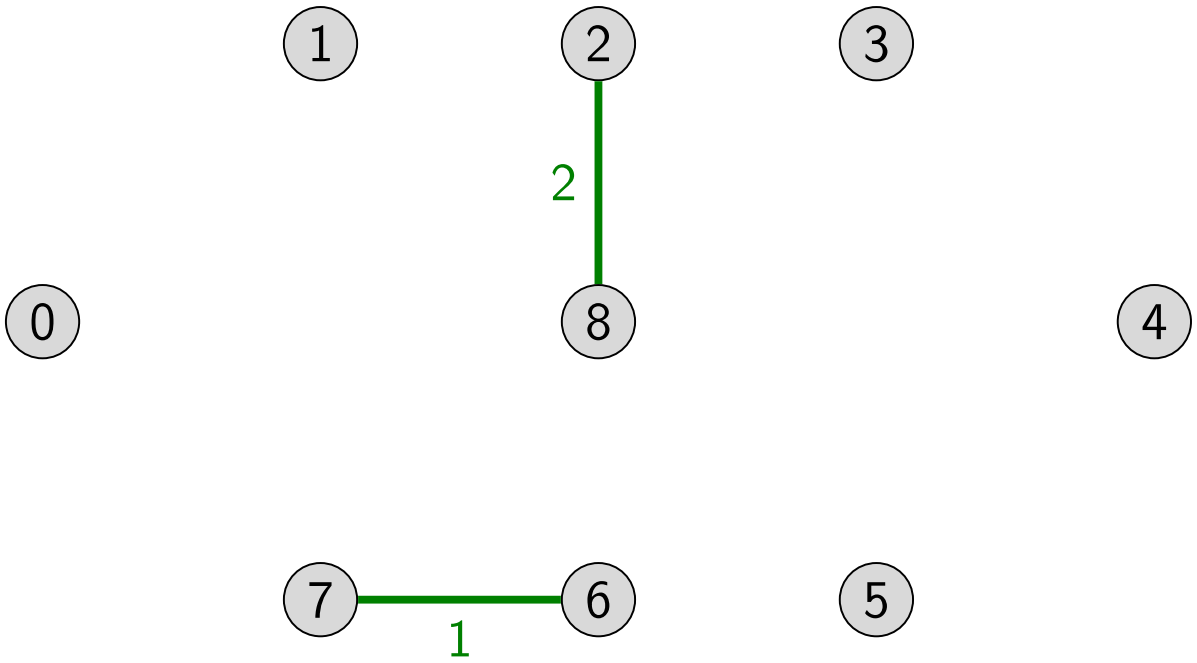
Ağırlık	Hedef	Kaynak
1	7	6
2	8	2
2	6	5
4	0	1
4	2	5
6	8	6
7	2	3
7	7	8
8	0	7
8	1	2
9	3	4
10	5	4
11	1	7
14	3	5

Kruskal Örnek



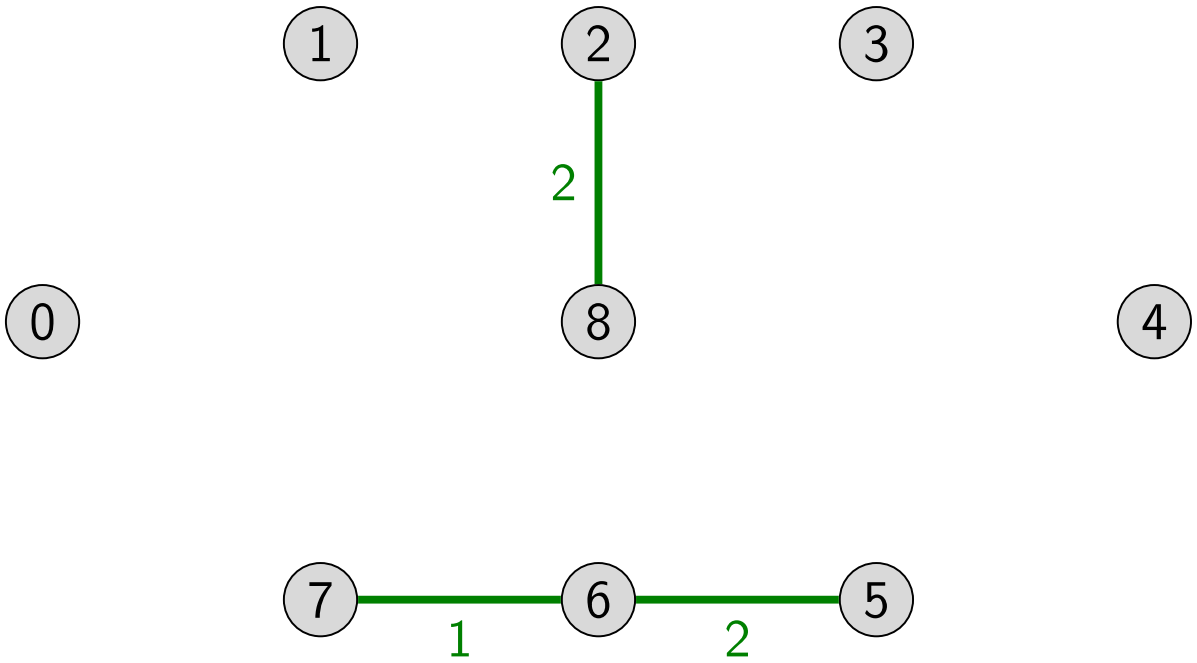
Ağırlık	Hedef	Kaynak
1	7	6
2	8	2
2	6	5
4	0	1
4	2	5
6	8	6
7	2	3
7	7	8
8	0	7
8	1	2
9	3	4
10	5	4
11	1	7
14	3	5

Kruskal Örnek



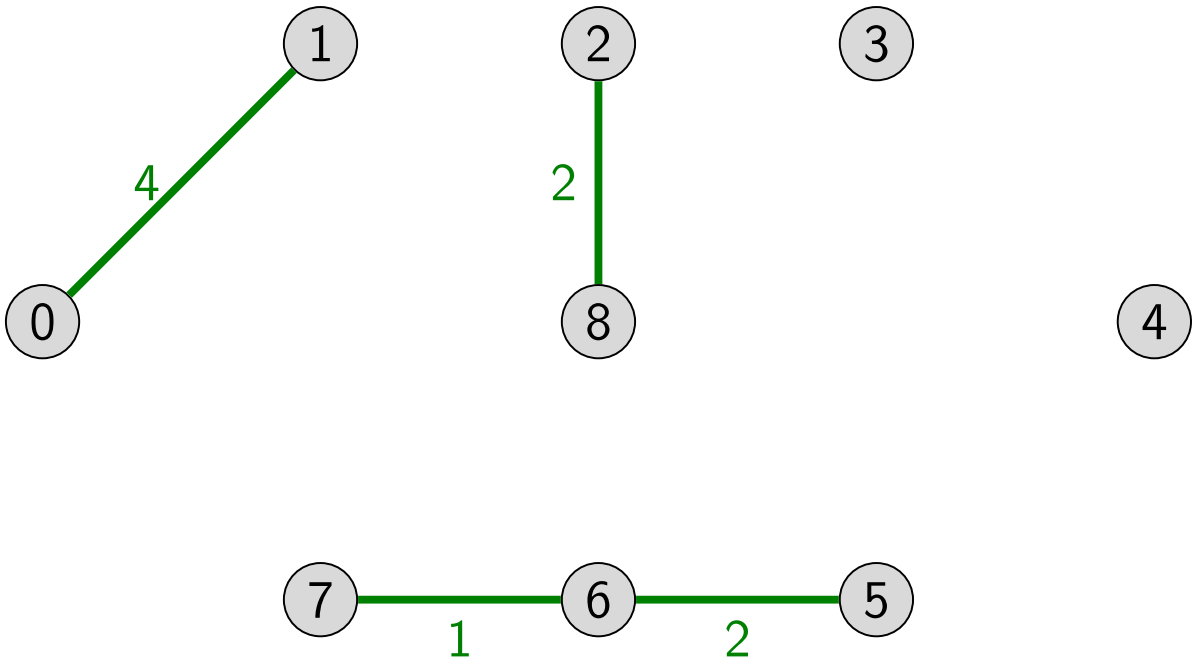
Ağırlık	Hedef	Kaynak
1	7	6
2	8	2
2	6	5
4	0	1
4	2	5
6	8	6
7	2	3
7	7	8
8	0	7
8	1	2
9	3	4
10	5	4
11	1	7
14	3	5

Kruskal Örnek



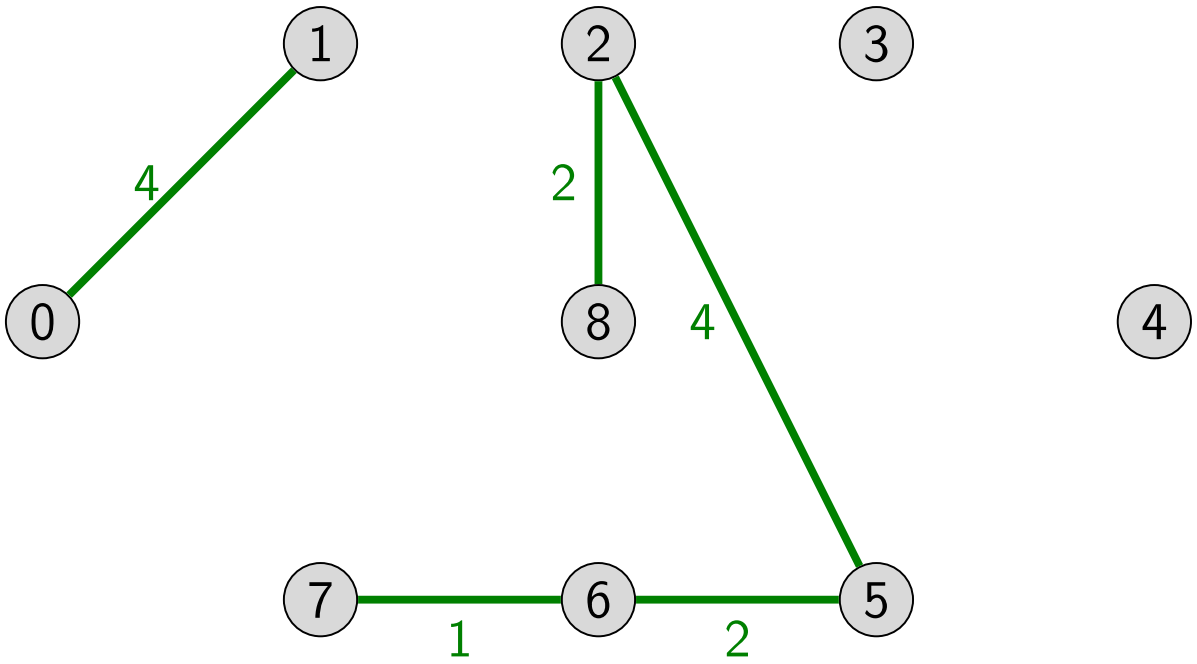
Ağırlık	Hedef	Kaynak
1	7	6
2	8	2
2	6	5
4	0	1
4	2	5
6	8	6
7	2	3
7	7	8
8	0	7
8	1	2
9	3	4
10	5	4
11	1	7
14	3	5

Kruskal Örnek



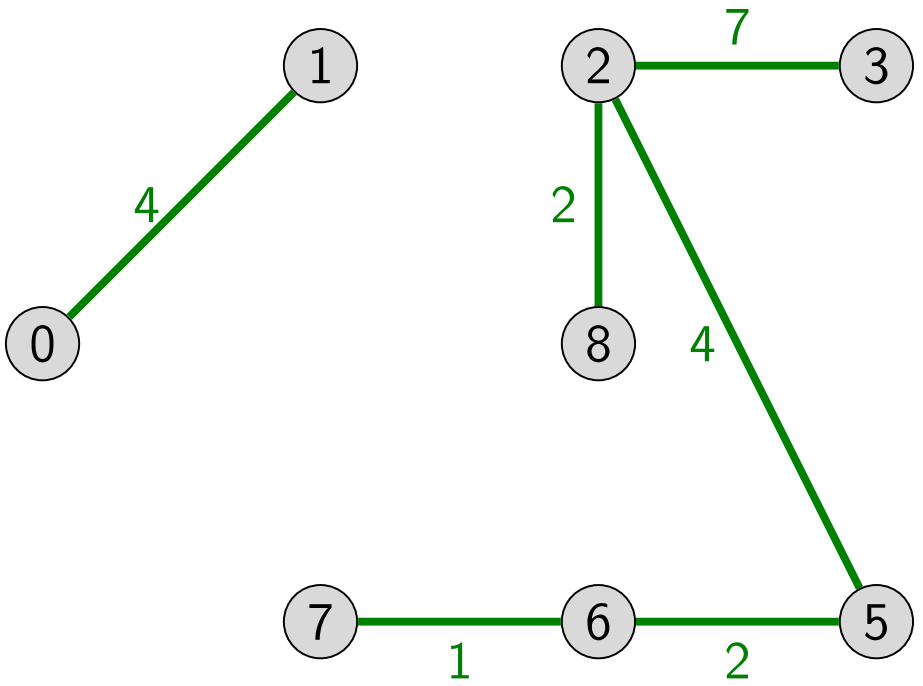
Ağırlık	Hedef	Kaynak
1	7	6
2	8	2
2	6	5
4	0	1
4	2	5
6	8	6
7	2	3
7	7	8
8	0	7
8	1	2
9	3	4
10	5	4
11	1	7
14	3	5

Kruskal Örnek



Ağırlık	Hedef	Kaynak
1	7	6
2	8	2
2	6	5
4	0	1
4	2	5
6	8	6
7	2	3
7	7	8
8	0	7
8	1	2
9	3	4
10	5	4
11	1	7
14	3	5

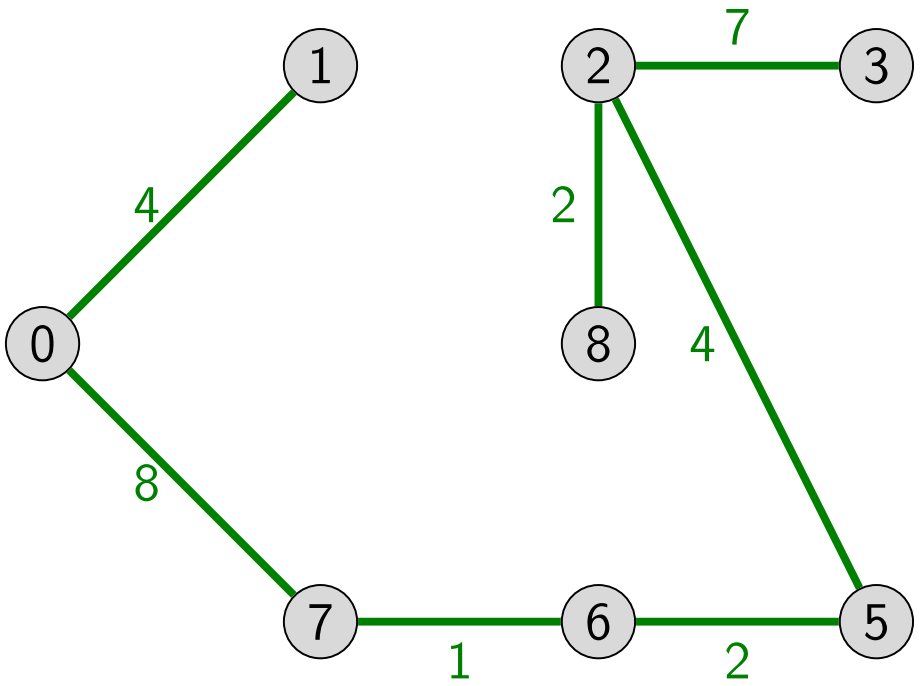
Kruskal Örnek



4

Ağırlık	Hedef	Kaynak
1	7	6
2	8	2
2	6	5
4	0	1
4	2	5
6	8	6
7	2	3
7	7	8
8	0	7
8	1	2
9	3	4
10	5	4
11	1	7
14	3	5

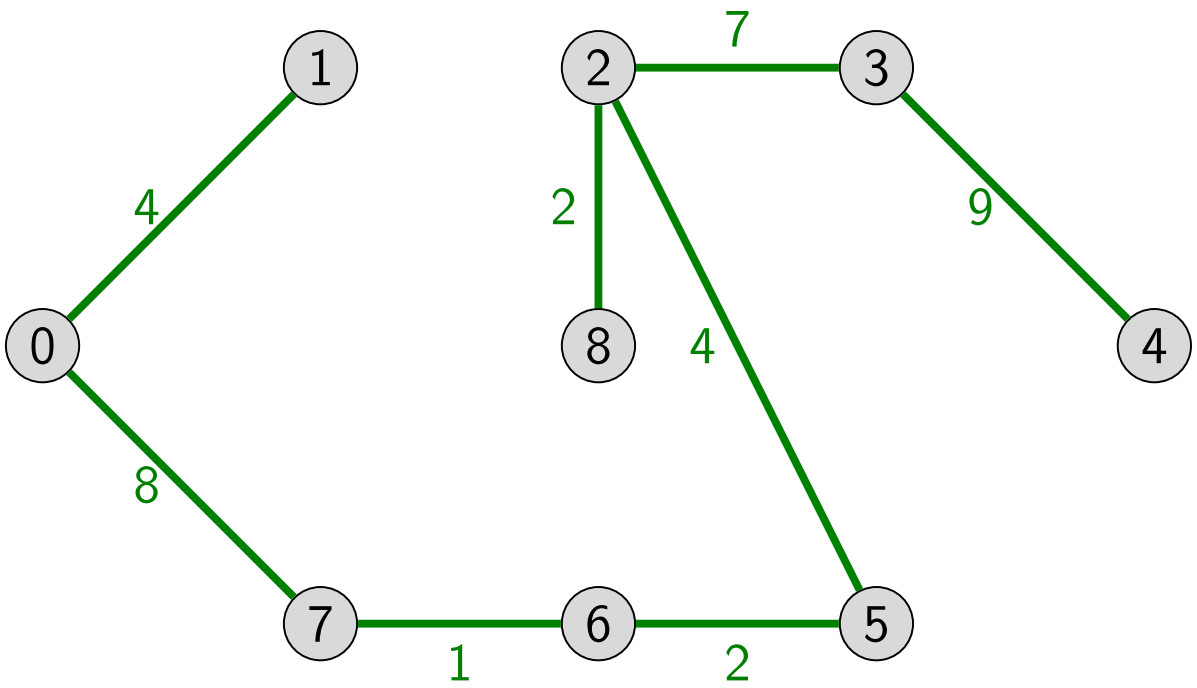
Kruskal Örnek



4

Ağırlık	Hedef	Kaynak
1	7	6
2	8	2
2	6	5
4	0	1
4	2	5
6	8	6
7	2	3
7	7	8
8	0	7
8	1	2
9	3	4
10	5	4
11	1	7
14	3	5

Kruskal Örnek

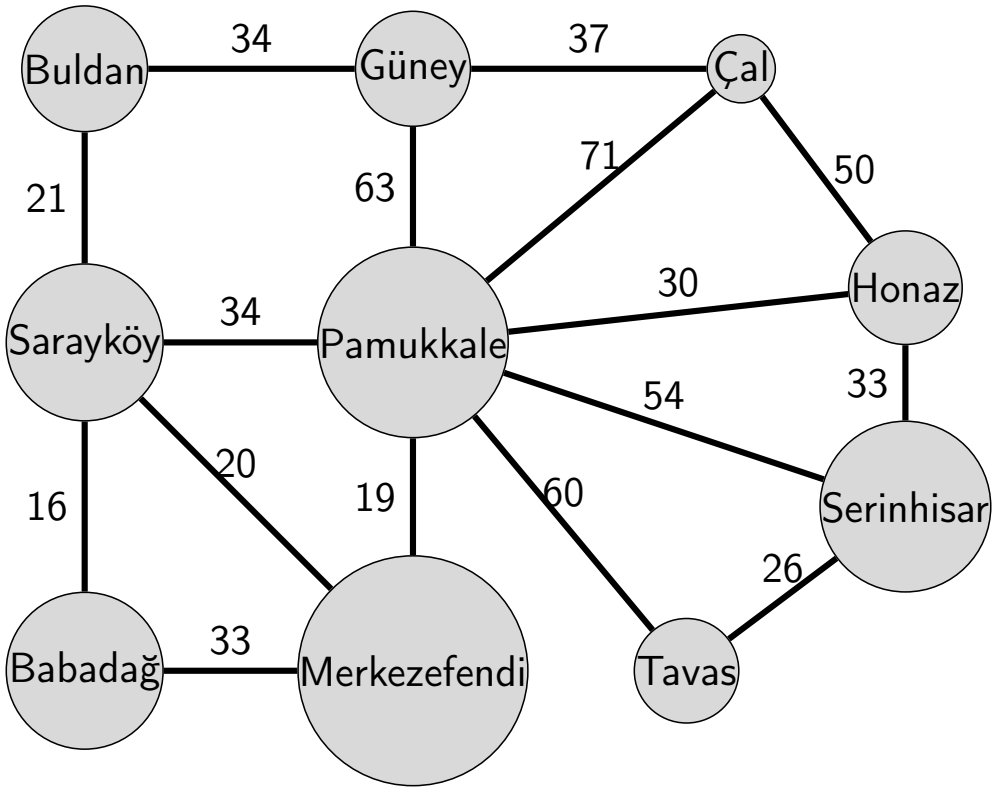


Ağırlık	Hedef	Kaynak
1	7	6
2	8	2
2	6	5
4	0	1
4	2	5
6	8	6
7	2	3
7	7	8
8	0	7
8	1	2
9	3	4
10	5	4
11	1	7
14	3	5

Toplam ağırlık: **37**

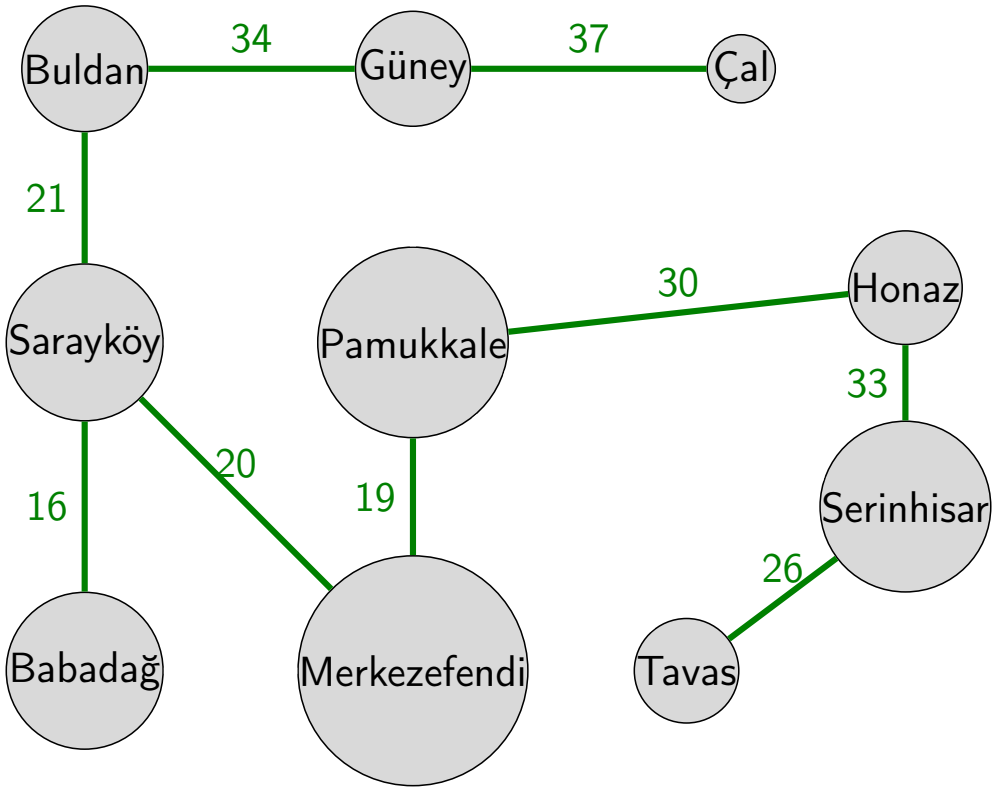
Kruskal Soru

Aşağıdaki çizgenin Kruskal algoritmasına göre minimum kapsayan ağacını bulunuz.



Kruskal Soru

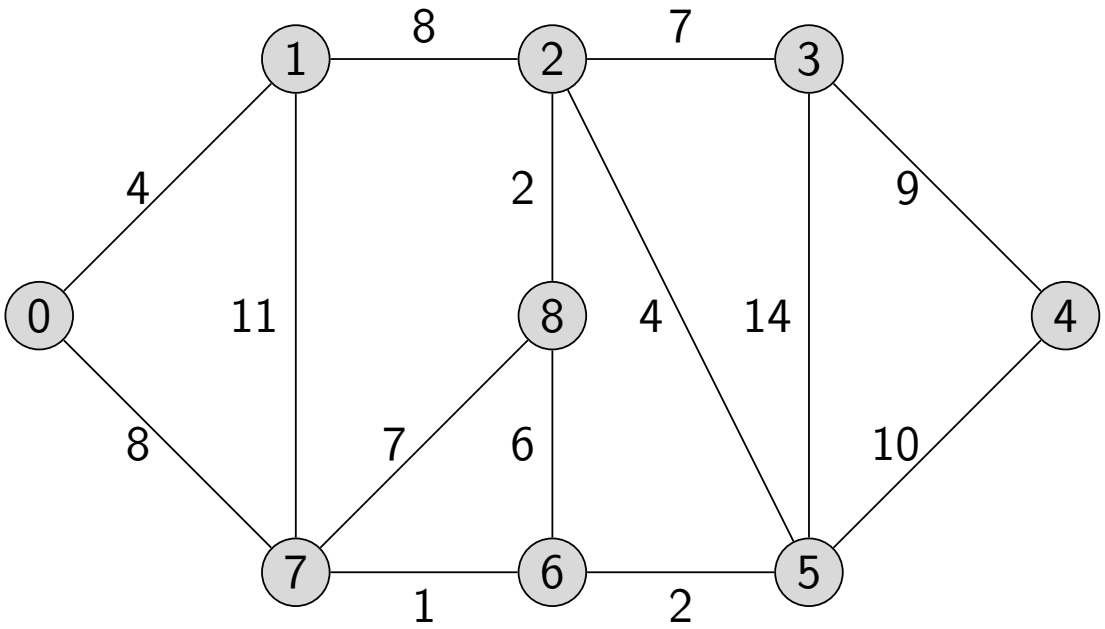
Aşağıdaki çizgenin Kruskal algoritmasına göre minimum kapsayan ağacını bulunuz.



Prim'in MST Algoritması

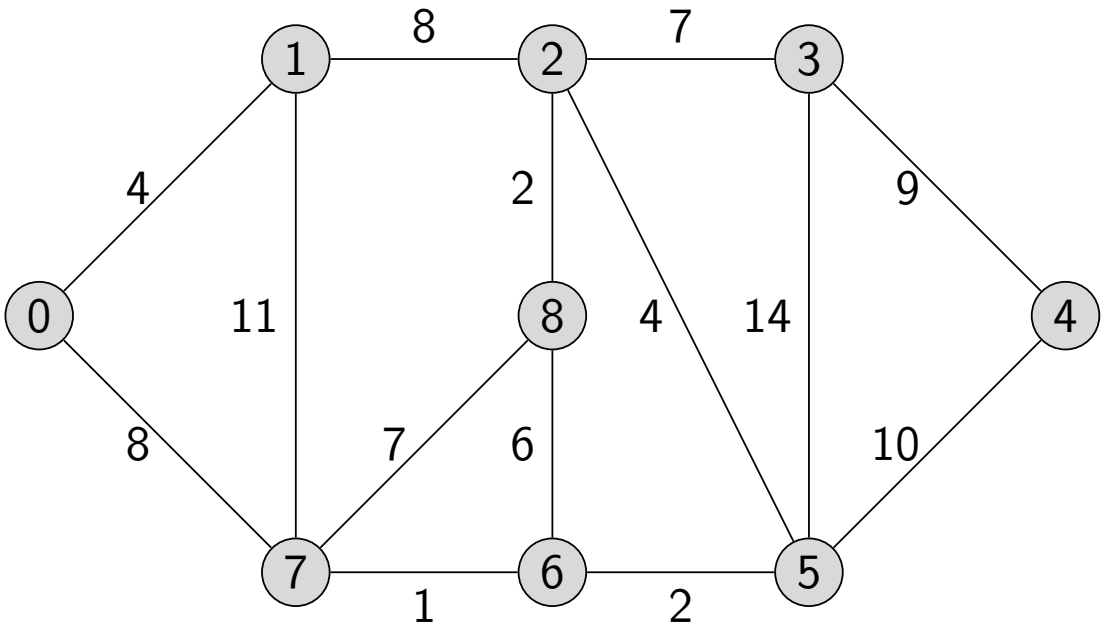
1. MST içindeki düğümleri tutacak S kümesini oluştur
2. Başlangıç düğümüne 0, diğer düğümlere ∞ anahtar değeri verilir
3. Bütün düğümler S kümesinde olmadığı sürece
   a) S kümesinden minimum anahtar değerine sahip u düğümünü al
   b) u'yu S kümesine ekle
   c) u'nun komşu düğümlerinin anahtar değerini güncelle: u'nun komşusu v için eğer u-v kenarının ağırlığı v'nin anahtar değerinden küçükse u-v kenarının değeri ile güncelle

Prim Örnek



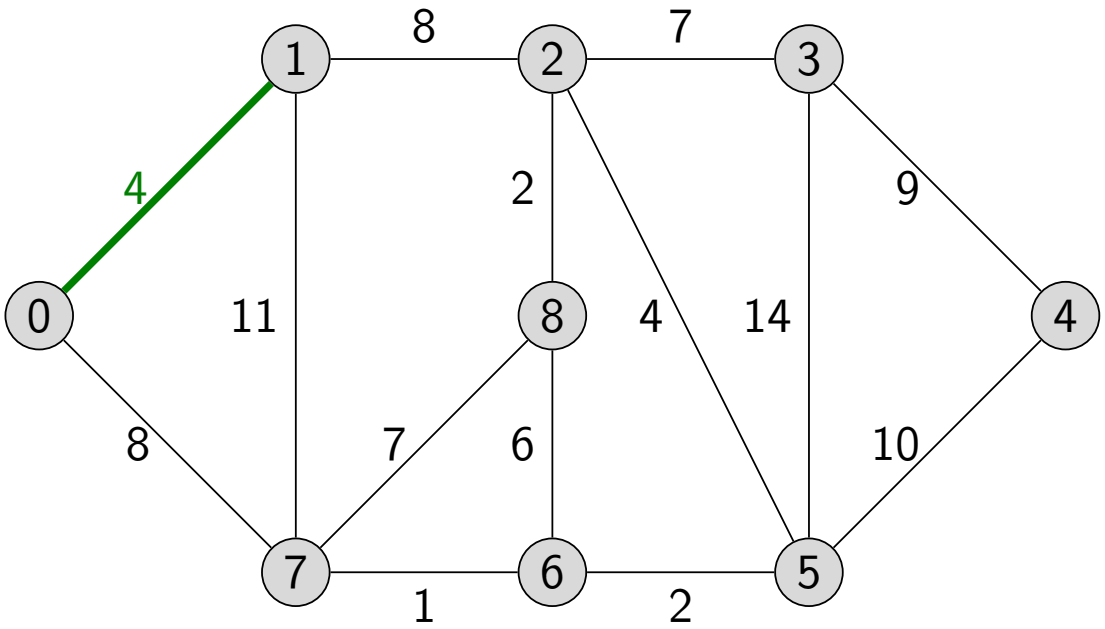
V	**Uzaklık**	$\in$
0	0	✓
1	∞	
2	∞	
3	∞	
4	∞	
5	∞	
6	∞	
7	∞	
8	∞	

Prim Örnek



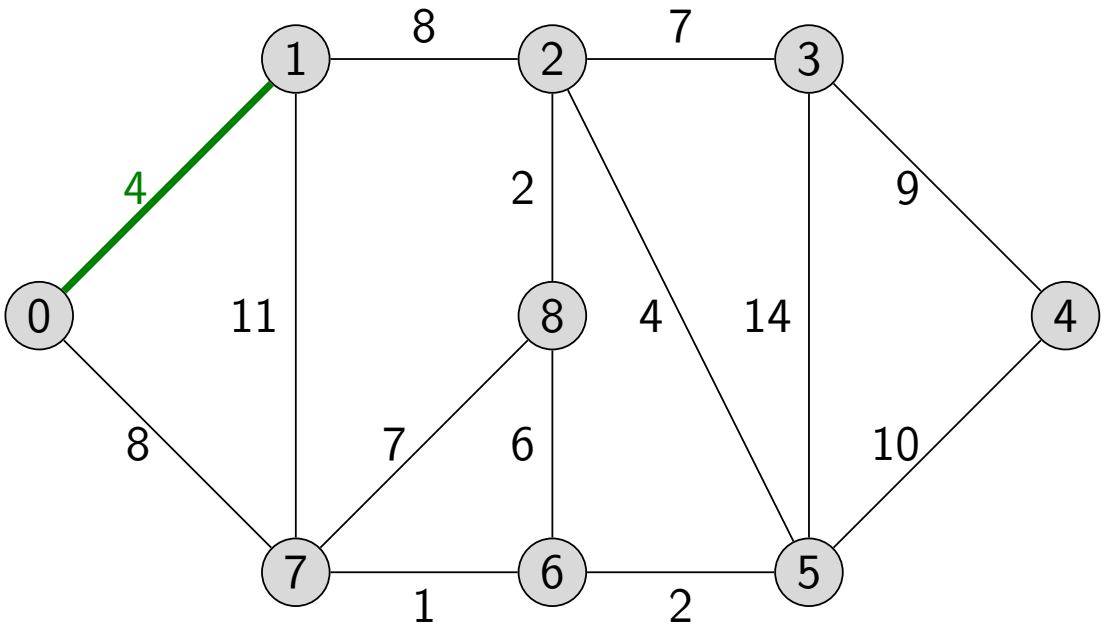
V	**Uzaklık**	$\in$
0	0	✓
1	4	
2	∞	
3	∞	
4	∞	
5	∞	
6	∞	
7	8	
8	∞	

Prim Örnek



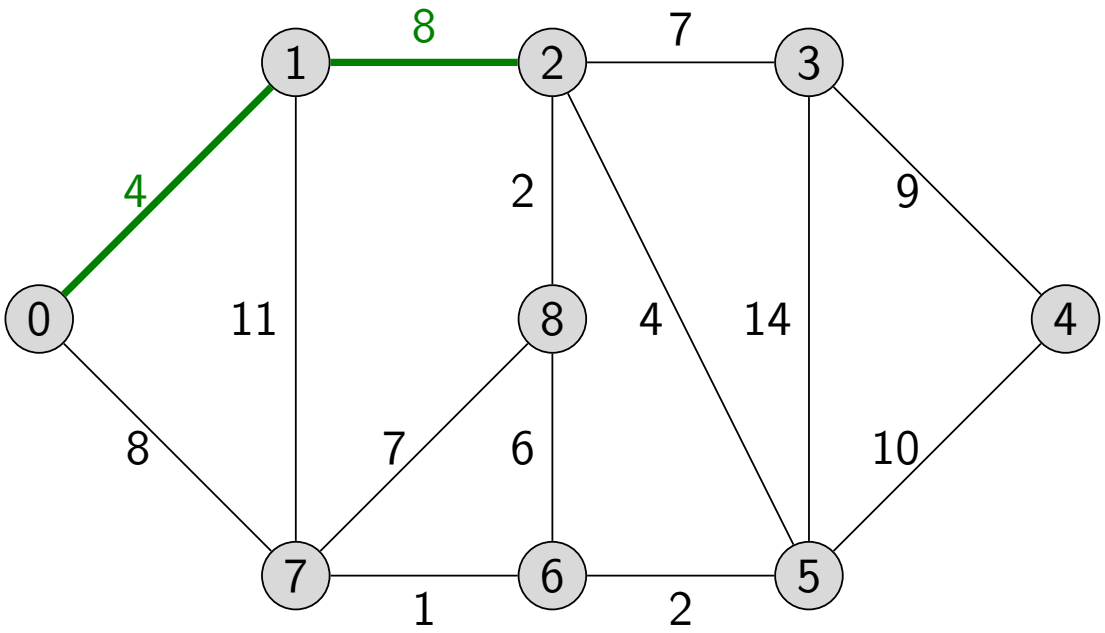
V	**Uzaklık**	$\in$
0	0	✓
1	4	✓
2	∞	
3	∞	
4	∞	
5	∞	
6	∞	
7	8	
8	∞	

Prim Örnek



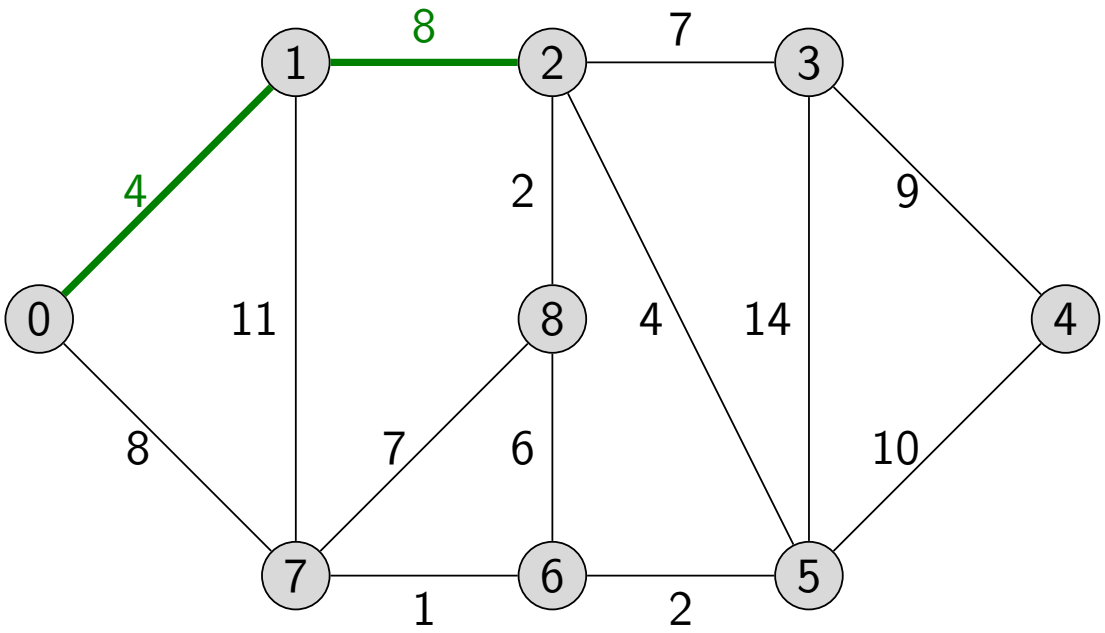
V	**Uzaklık**	$\in$
0	0	✓
1	4	✓
2	8	
3	∞	
4	∞	
5	∞	
6	∞	
7	8	
8	∞	

Prim Örnek



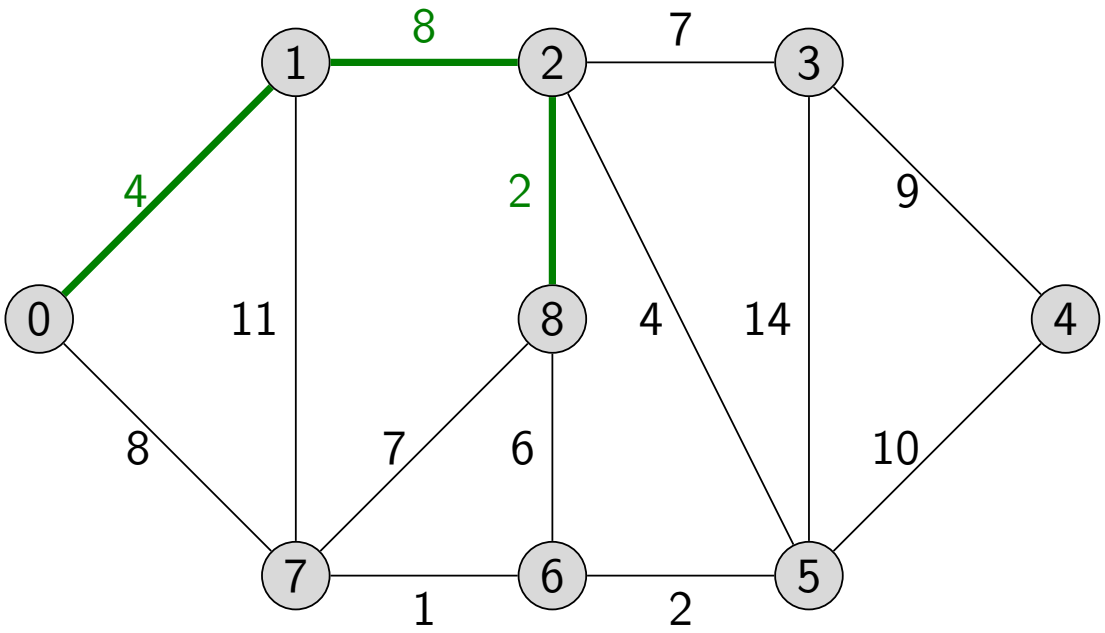
V	**Uzaklık**	$\in$
0	0	✓
1	4	✓
2	8	✓
3	∞	
4	∞	
5	∞	
6	∞	
7	8	
8	∞	

Prim Örnek



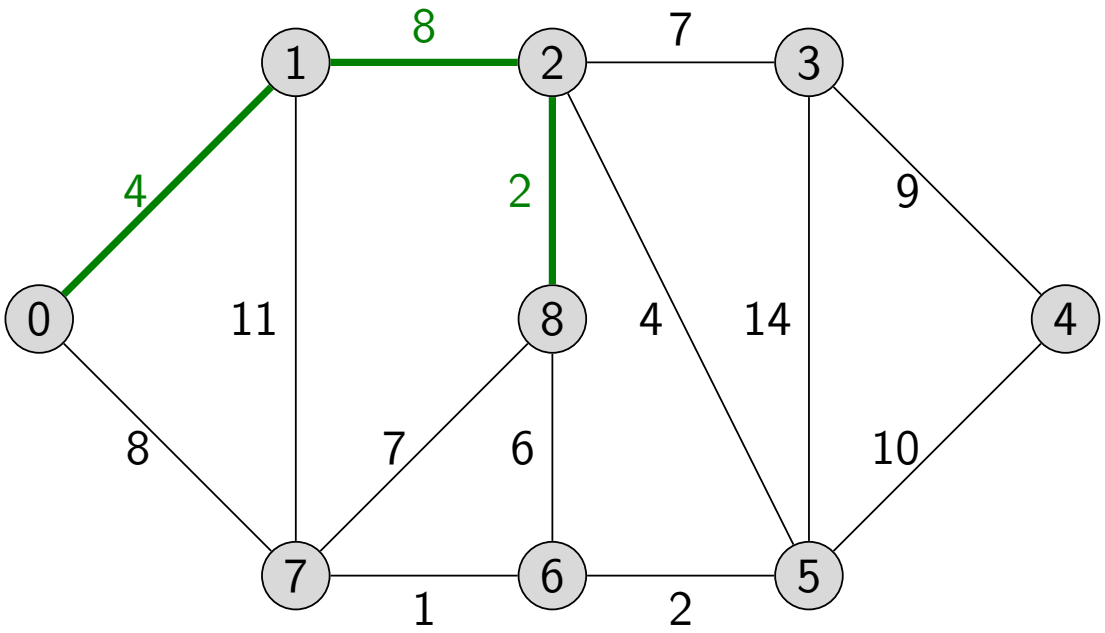
V	**Uzaklık**	$\in$
0	0	✓
1	4	✓
2	8	✓
3	7	
4	∞	
5	4	
6	∞	
7	8	
8	2	

Prim Örnek



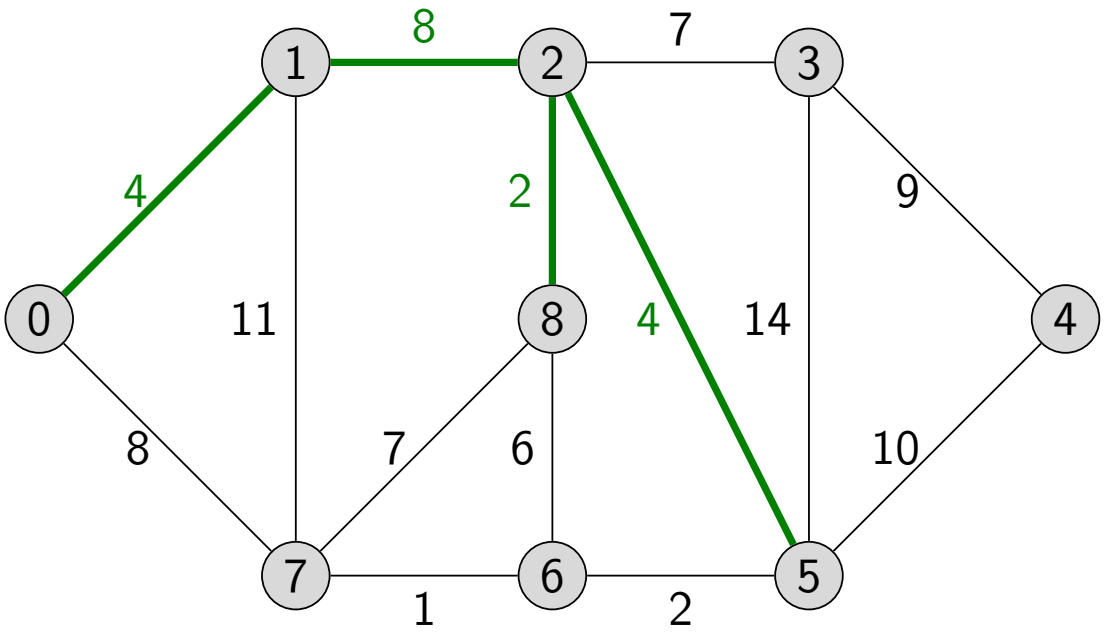
V	**Uzaklık**	$\in$
0	0	✓
1	4	✓
2	8	✓
3	7	
4	∞	
5	4	
6	∞	
7	8	
8	2	✓

Prim Örnek



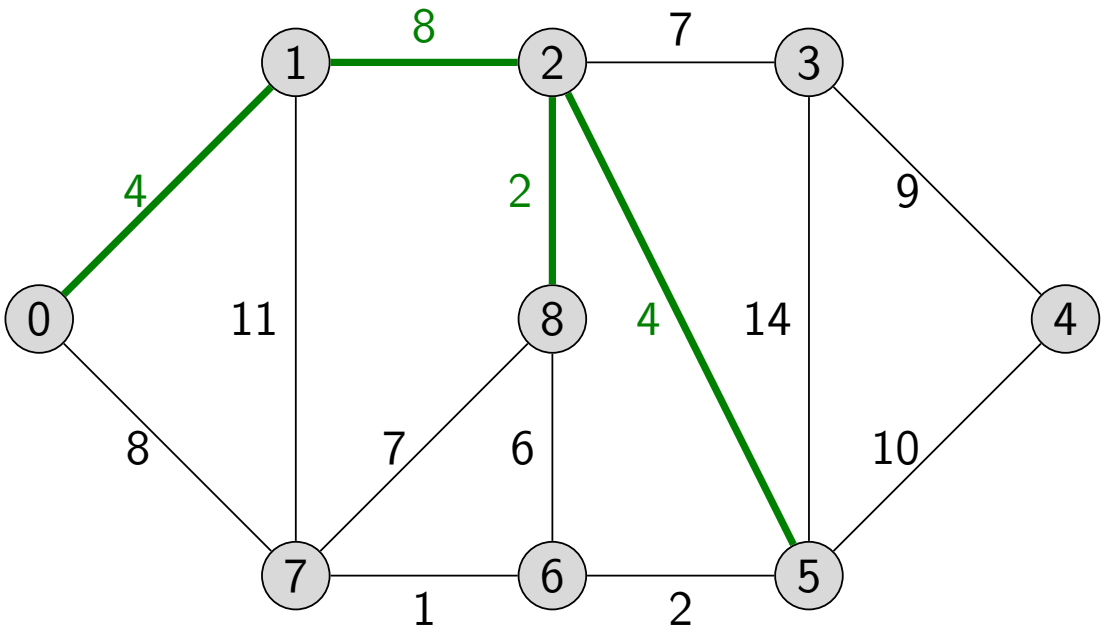
V	**Uzaklık**	$\in$
0	0	✓
1	4	✓
2	8	✓
3	7	
4	∞	
5	4	
6	6	
7	7	
8	2	✓

Prim Örnek



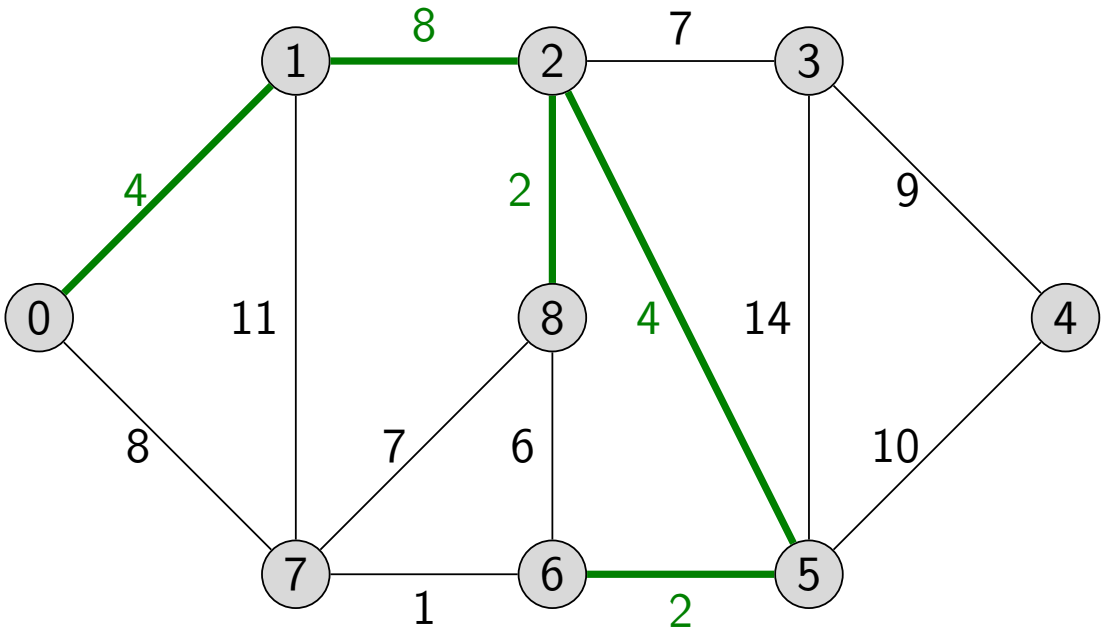
V	**Uzaklık**	$\in$
0	0	✓
1	4	✓
2	8	✓
3	7	
4	∞	
5	4	✓
6	6	
7	7	
8	2	✓

Prim Örnek



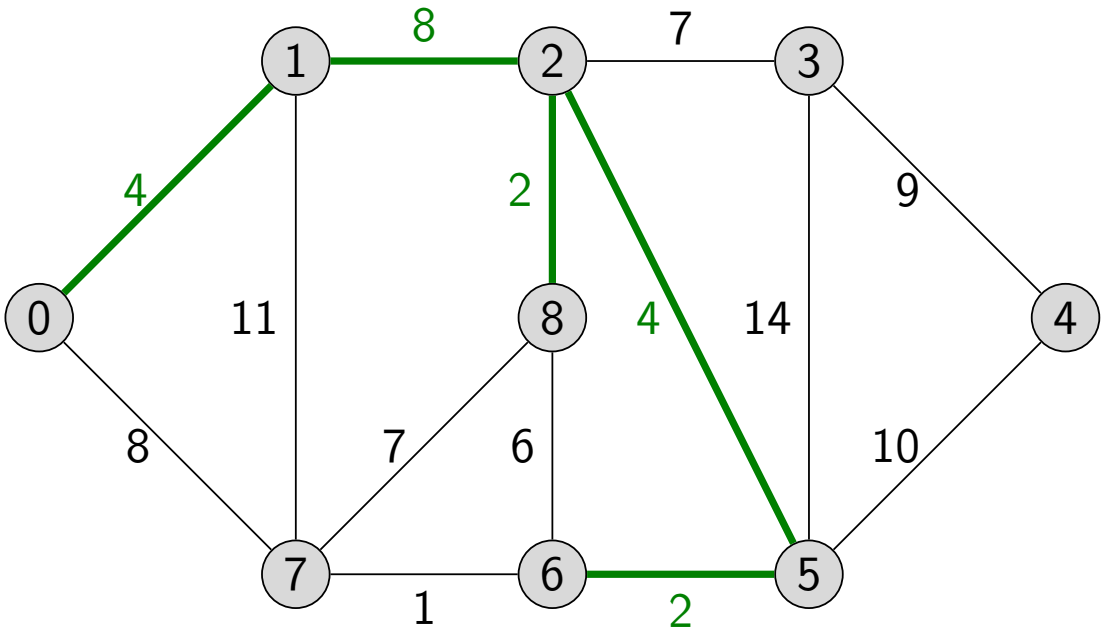
V	**Uzaklık**	$\in$
0	0	✓
1	4	✓
2	8	✓
3	7	
4	10	
5	4	✓
6	2	
7	7	
8	2	✓

Prim Örnek



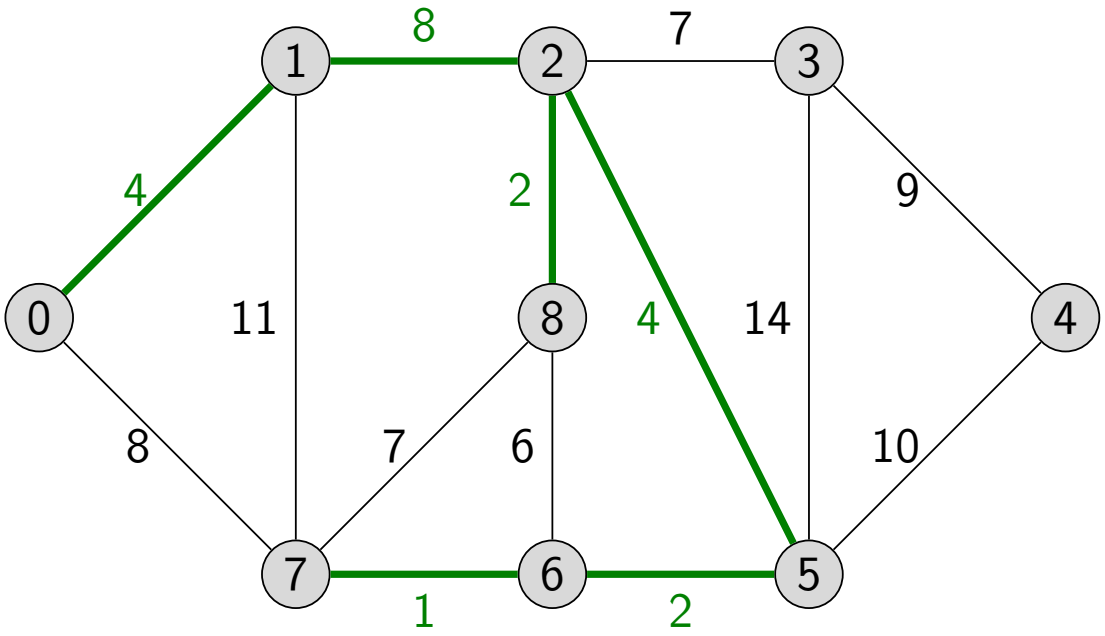
V	**Uzaklık**	$\in$
0	0	✓
1	4	✓
2	8	✓
3	7	
4	10	
5	4	✓
6	2	✓
7	7	
8	2	✓

Prim Örnek



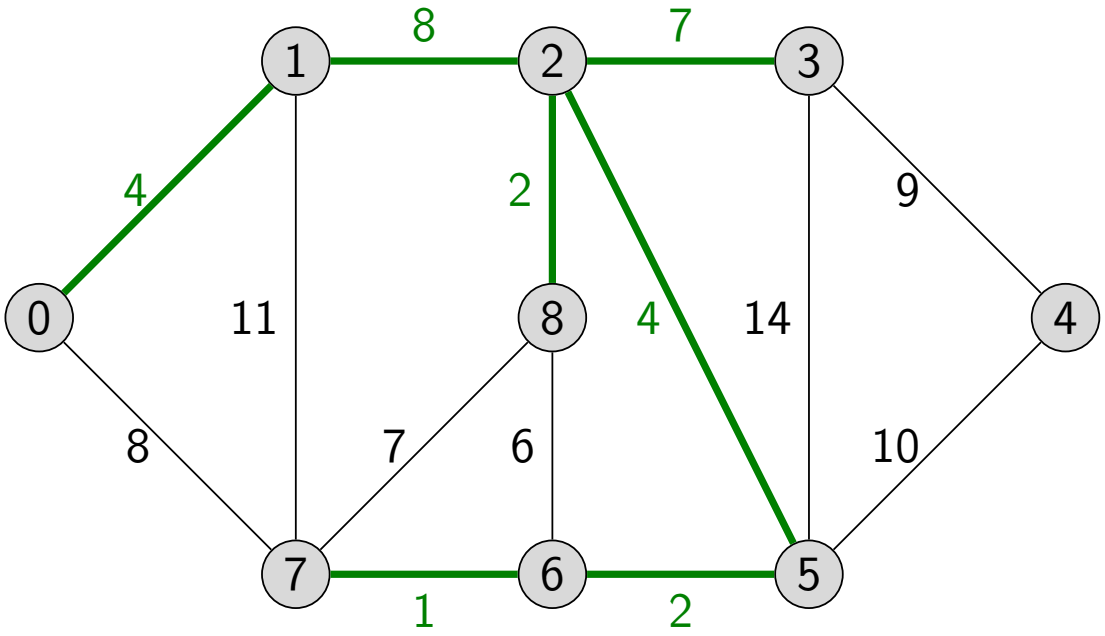
V	**Uzaklık**	$\in$
0	0	✓
1	4	✓
2	8	✓
3	7	
4	10	
5	4	✓
6	2	✓
7	1	
8	2	✓

Prim Örnek



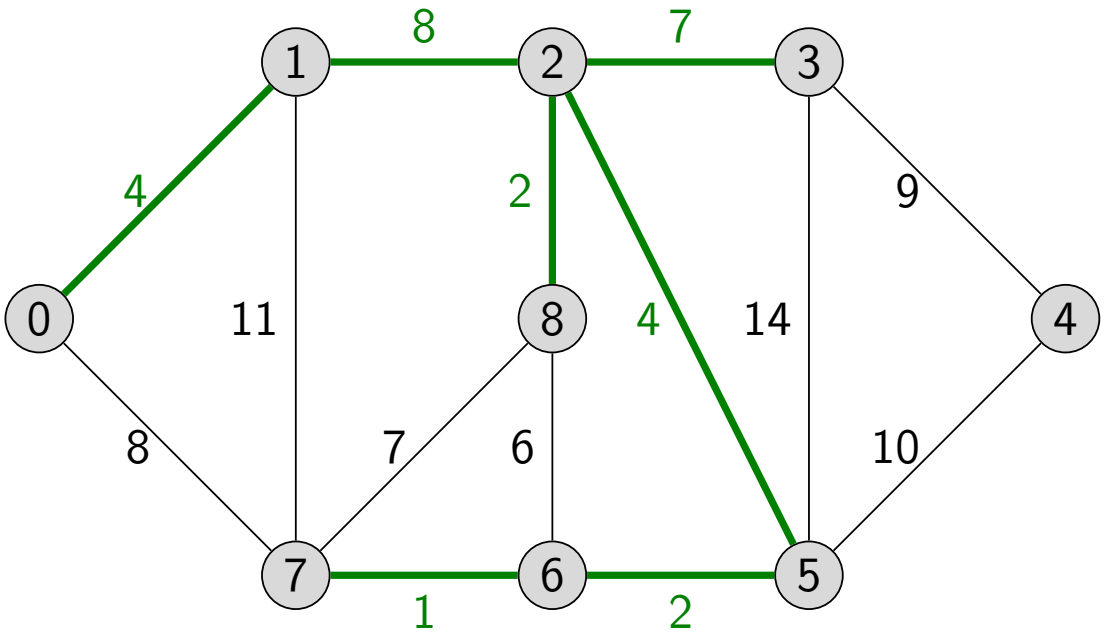
V	**Uzaklık**	$\in$
0	0	✓
1	4	✓
2	8	✓
3	7	
4	10	
5	4	✓
6	2	✓
7	1	✓
8	2	✓

Prim Örnek



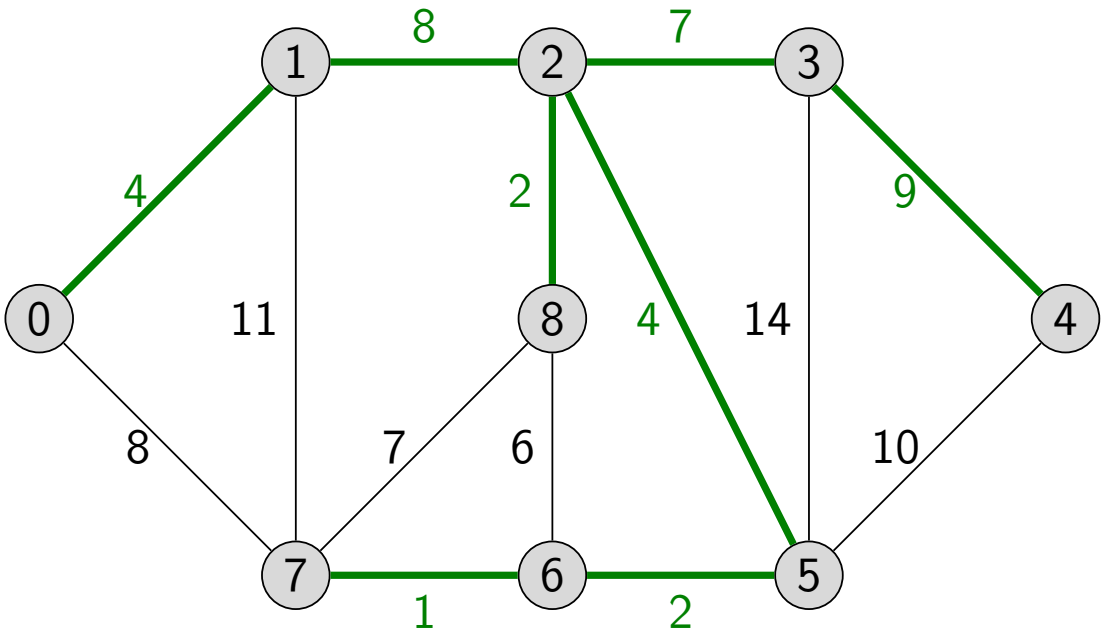
V	**Uzaklık**	$\in$
0	0	✓
1	4	✓
2	8	✓
3	7	✓
4	10	
5	4	✓
6	2	✓
7	1	✓
8	2	✓

Prim Örnek



V	**Uzaklık**	$\in$
0	0	✓
1	4	✓
2	8	✓
3	7	✓
4	9	
5	4	✓
6	2	✓
7	1	✓
8	2	✓

Prim Örnek

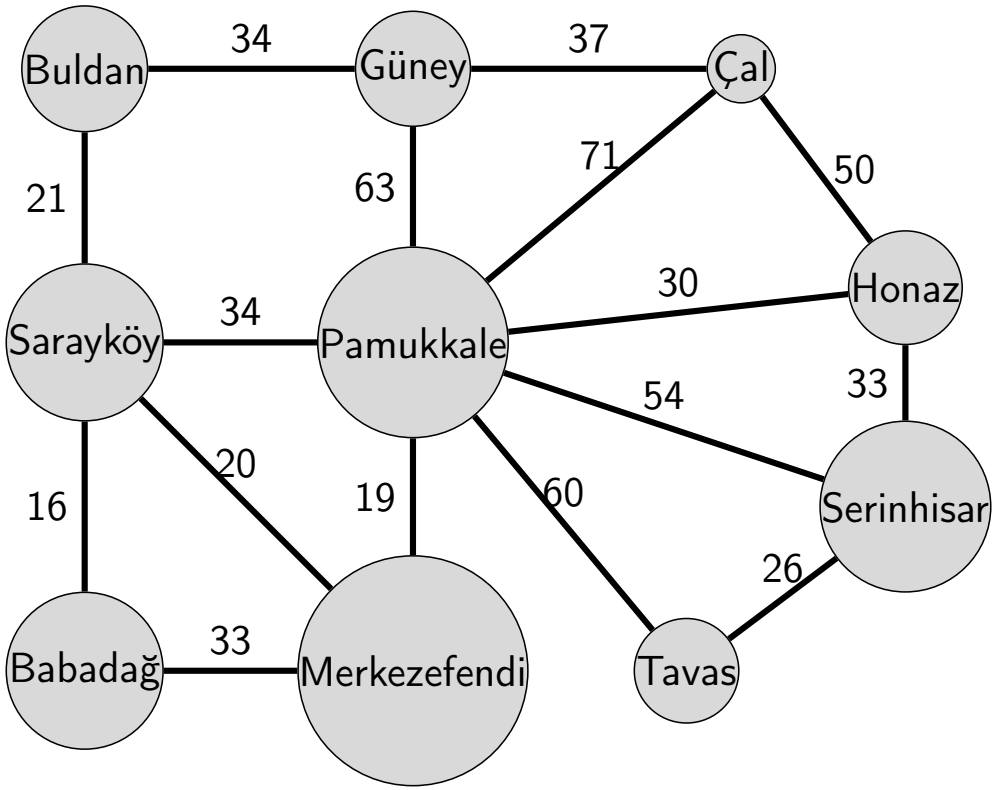


V	**Uzaklık**	$\in$
0	0	✓
1	4	✓
2	8	✓
3	7	✓
4	9	✓
5	4	✓
6	2	✓
7	1	✓
8	2	✓

Toplam ağırlık: **37**

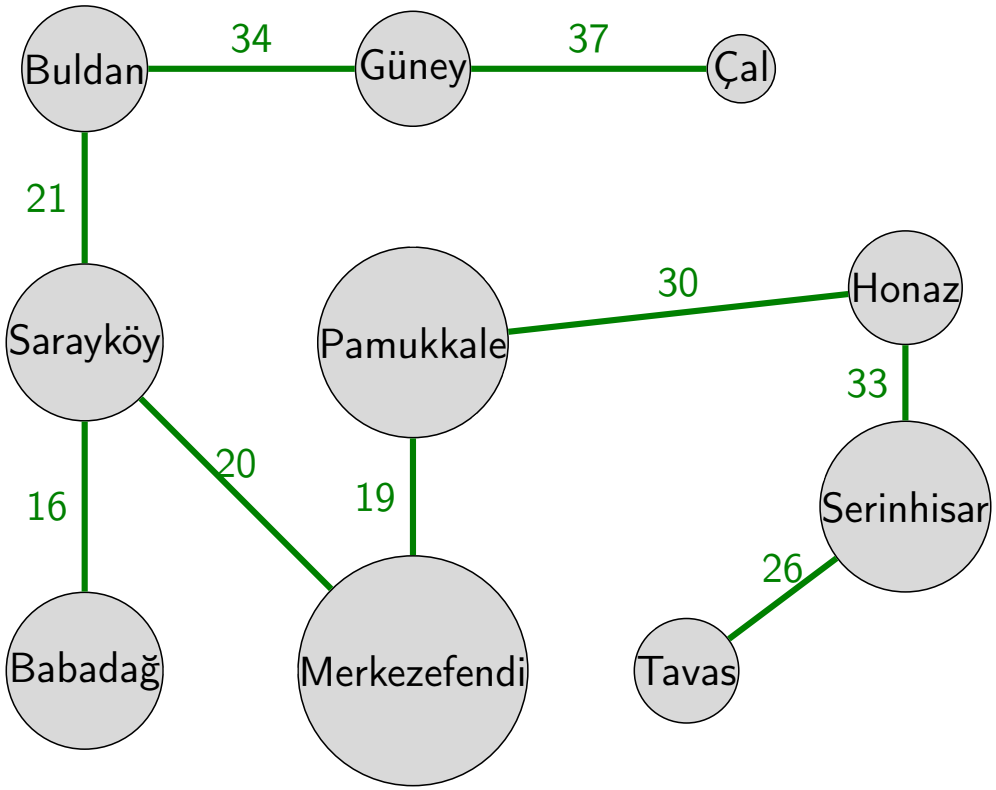
Prim Soru

Aşağıdaki çizgenin Prim algoritmasına göre minimum açılım ağacını bulunuz.



Prim Soru

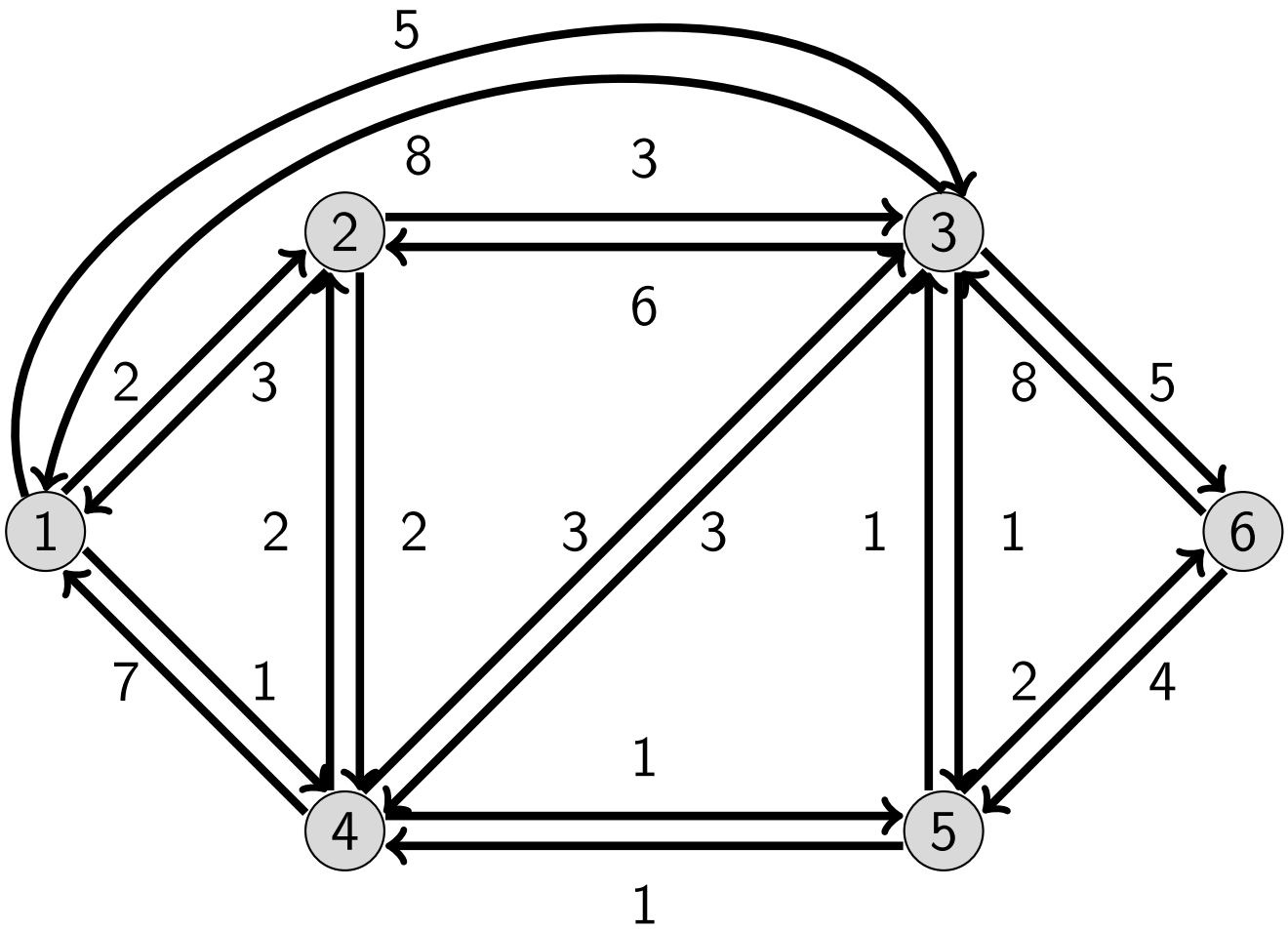
Aşağıdaki çizgenin Prim algoritmasına göre minimum açılım ağacını bulunuz.



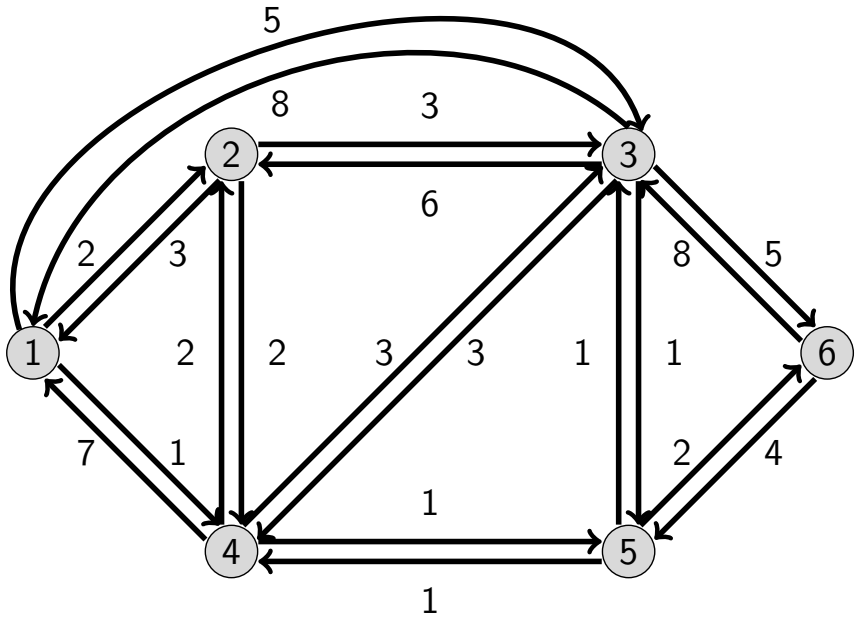
Dijkstra'nın Algoritması

- Seçilen bir düğümden diğer bütün düğümlere olan en kısa yolu bulur.
- Prim'in algoritmasına benzer şekilde çalışır.
- Başlangıçta boş bir küme il başlanır ve her seferinde kümeye komşu düğümlerden ağırlığı en küçük olan seçilir ve güncellemeler yapılır.
- Eğer seçilen elemanın komşularında daha kısa bir yol varsa uzunluk ve yol güncellenir.

Dijkstra'nın Algoritması

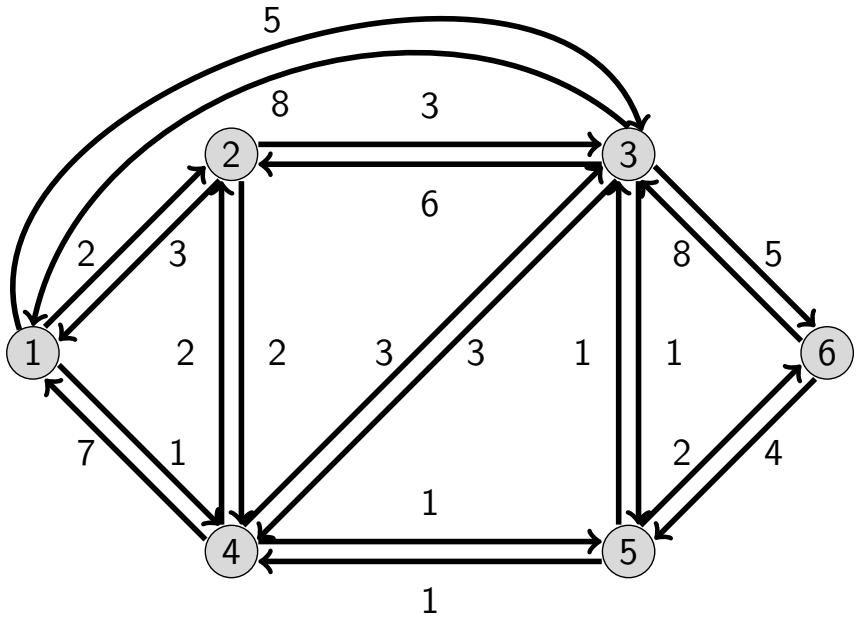


Dijkstra Örnek



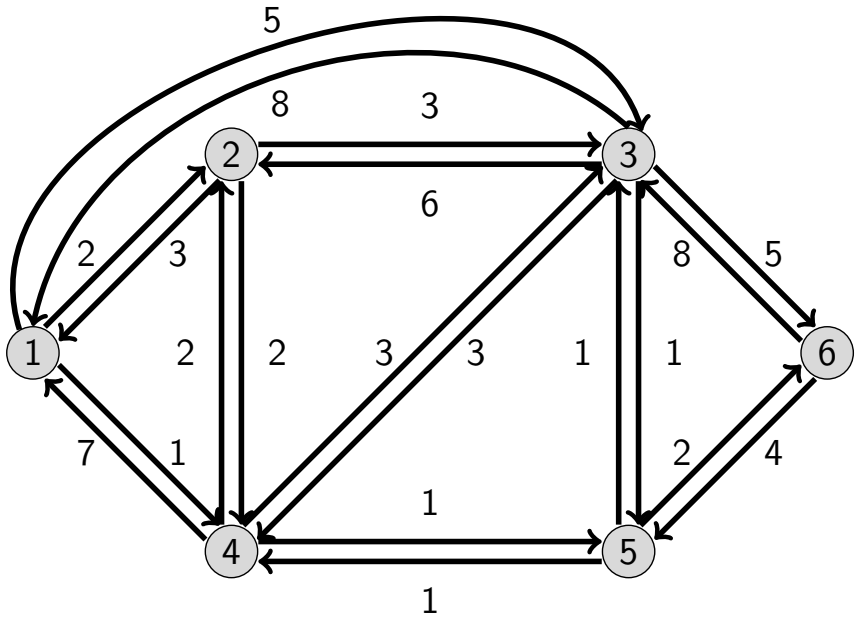
It	T	U.2	Y.2	U.3	Y.3	U.4	Y.4	U.5	Y.5	U.6	Y.6

Dijkstra Örnek



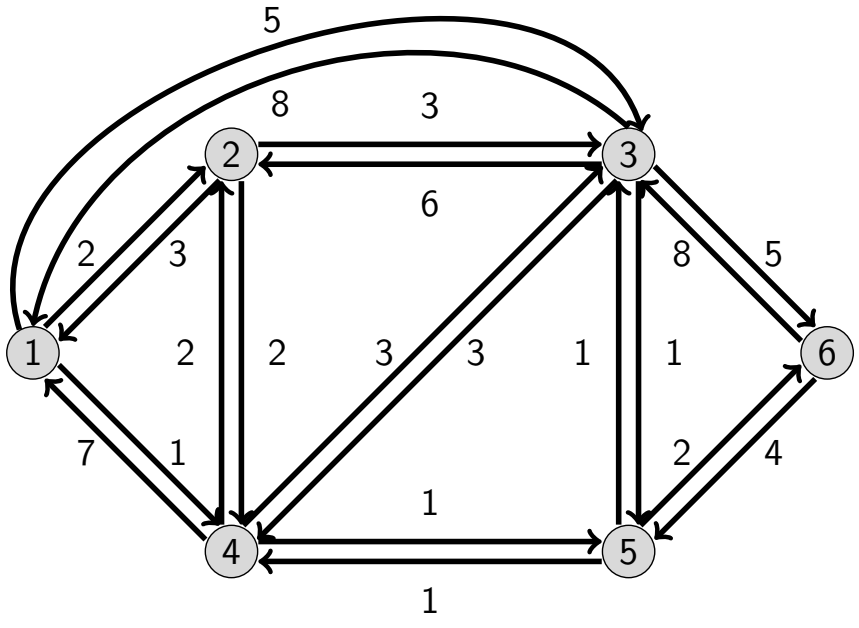
It	T	U.2	Y.2	U.3	Y.3	U.4	Y.4	U.5	Y.5	U.6	Y.6
1	1	2	1-2	5	1-3	1	1-4	∞	-	∞	-

Dijkstra Örnek



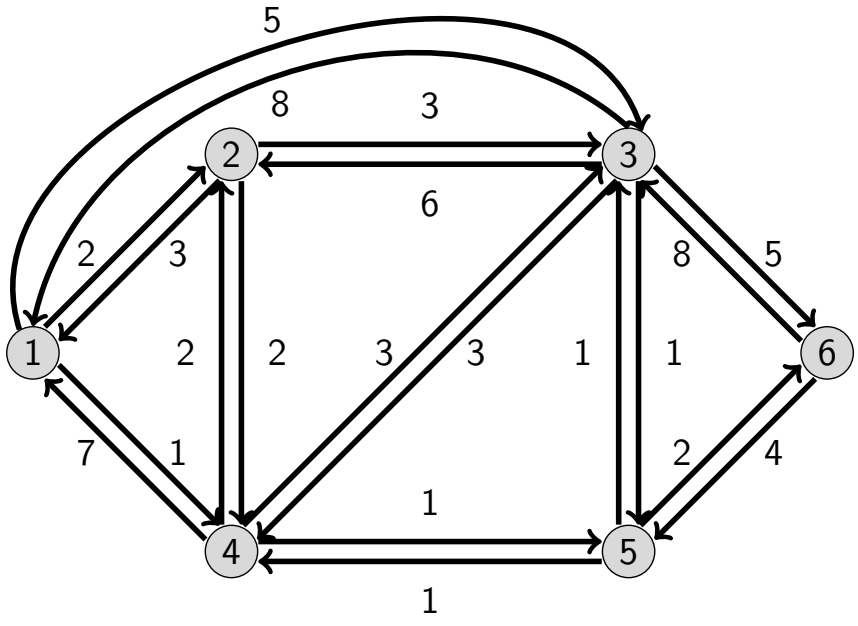
It	T	U.2	Y.2	U.3	Y.3	U.4	Y.4	U.5	Y.5	U.6	Y.6
1	1	2	1-2	5	1-3	**1**	**1-4**	∞	-	∞	-
2	1,4	**2**	**1-2**	4	1-4-3			2	1-4-5	∞	-

Dijkstra Örnek



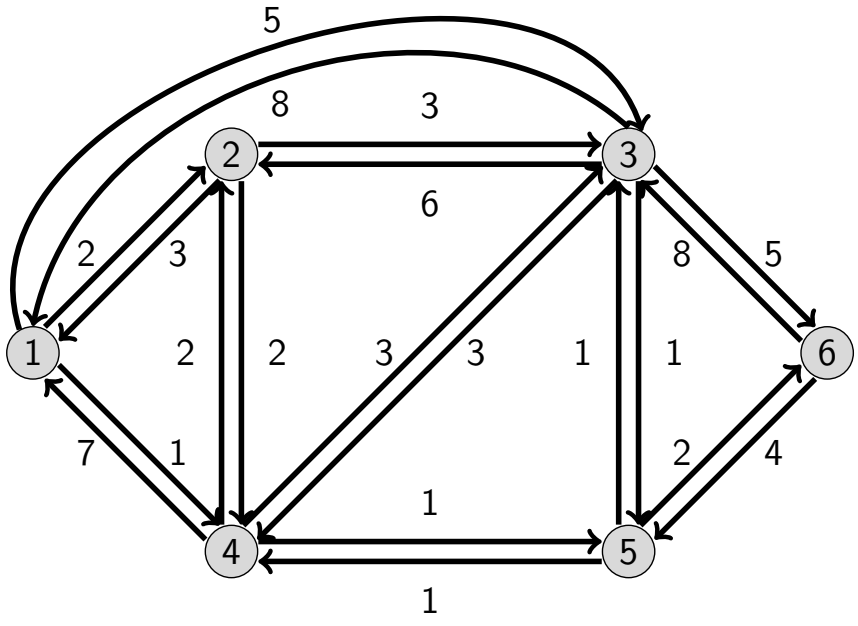
It	T	U.2	Y.2	U.3	Y.3	U.4	Y.4	U.5	Y.5	U.6	Y.6
1	1	2	1-2	5	1-3	**1**	**1-4**	∞	-	∞	-
2	1,4	**2**	**1-2**	4	1-4-3			2	1-4-5	∞	-
3	1,4,2			4	1-4-3			**2**	**1-4-5**	∞	-

Dijkstra Örnek



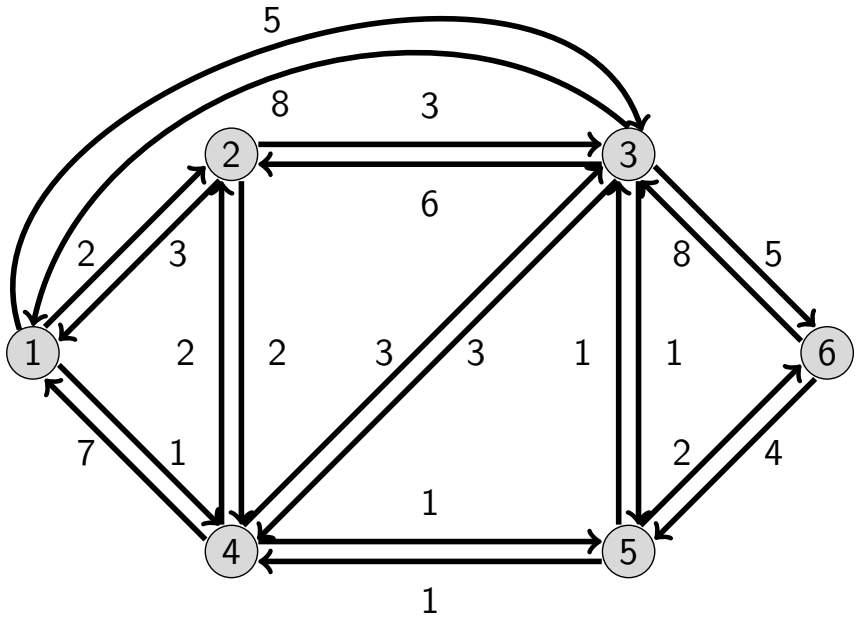
It	T	U.2	Y.2	U.3	Y.3	U.4	Y.4	U.5	Y.5	U.6	Y.6
1	1	2	1-2	5	1-3	**1**	**1-4**	∞	-	∞	-
2	1,4	**2**	**1-2**	4	1-4-3			2	1-4-5	∞	-
3	1,4,2			4	1-4-3			**2**	**1-4-5**	∞	-
4	1,4,2,5			**3**	**1-4-5-3**					4	1-4-5-6

Dijkstra Örnek

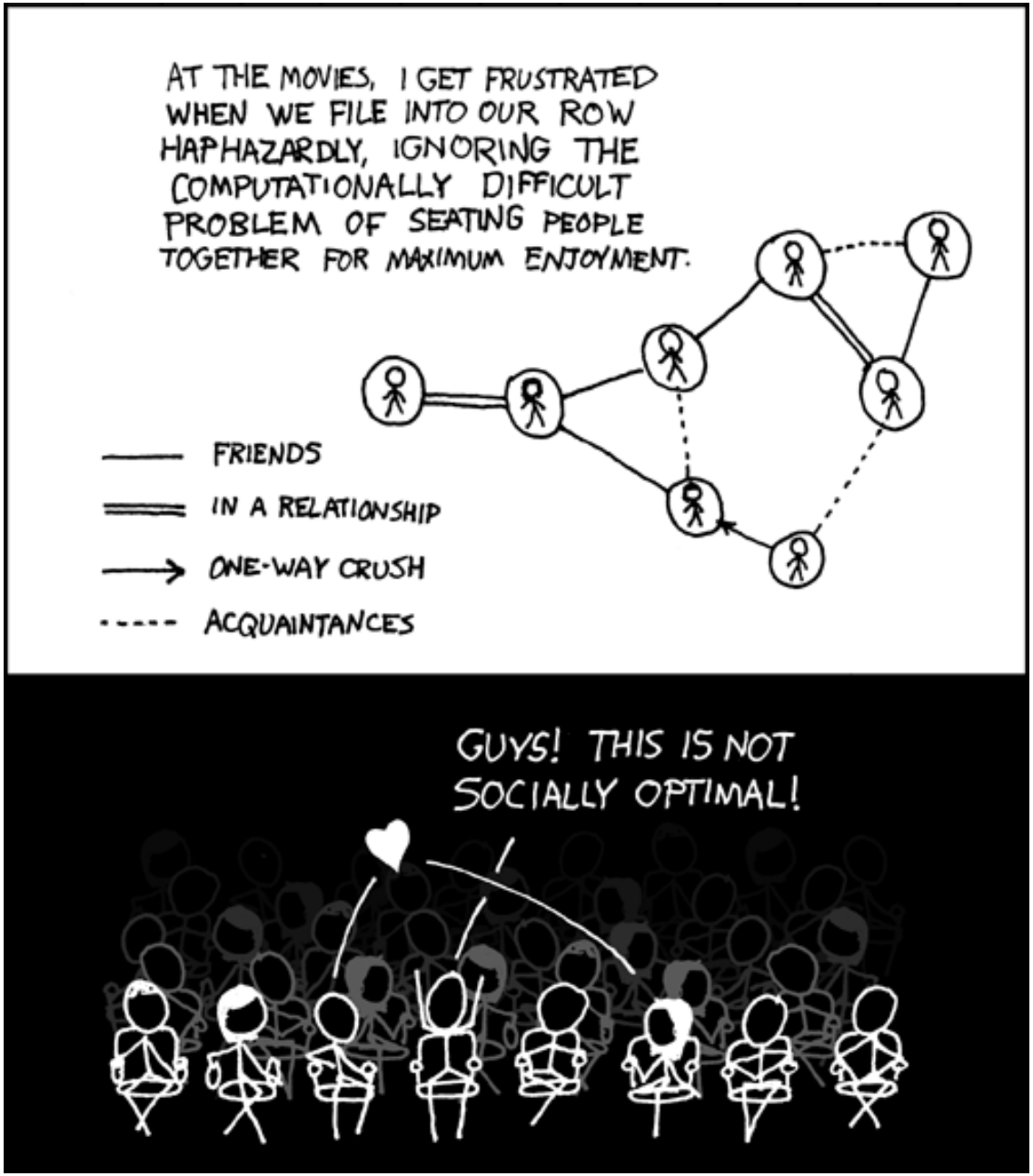


It	T	U.2	Y.2	U.3	Y.3	U.4	Y.4	U.5	Y.5	U.6	Y.6
1	1	2	1-2	5	1-3	**1**	**1-4**	∞	-	∞	-
2	1,4	**2**	**1-2**	4	1-4-3			2	1-4-5	∞	-
3	1,4,2			4	1-4-3			**2**	**1-4-5**	∞	-
4	1,4,2,5			**3**	**1-4-5-3**					4	1-4-5-6
5	1,4,2,5,3									**4**	**1-4-5-6**

Dijkstra Örnek



It	T	U.2	Y.2	U.3	Y.3	U.4	Y.4	U.5	Y.5	U.6	Y.6
1	1	2	1-2	5	1-3	**1**	**1-4**	∞	-	∞	-
2	1,4	**2**	**1-2**	4	1-4-3			2	1-4-5	∞	-
3	1,4,2			4	1-4-3			**2**	**1-4-5**	∞	-
4	1,4,2,5			**3**	**1-4-5-3**					4	1-4-5-6
5	1,4,2,5,3									**4**	**1-4-5-6**
Çözüm	1,4,2,5,3,6	2	1-2	3	1-4-5-3	1	1-4	2	1-4-5	4	1-4-5-6


AT THE MOVIES, I GET FRUSTRATED WHEN WE FILE INTO OUR ROW HAPHAZARDLY, IGNORING THE COMPUTATIONALLY DIFFICULT PROBLEM OF SEATING PEOPLE TOGETHER FOR MAXIMUM ENJOYMENT.
FRIENDS
IN A RELATIONSHIP
ONE-WAY CRUSH
ACQUAINTANCES
GUYS! THIS IS NOT SOCIALLY OPTIMAL!

[1]

[1] xkcd