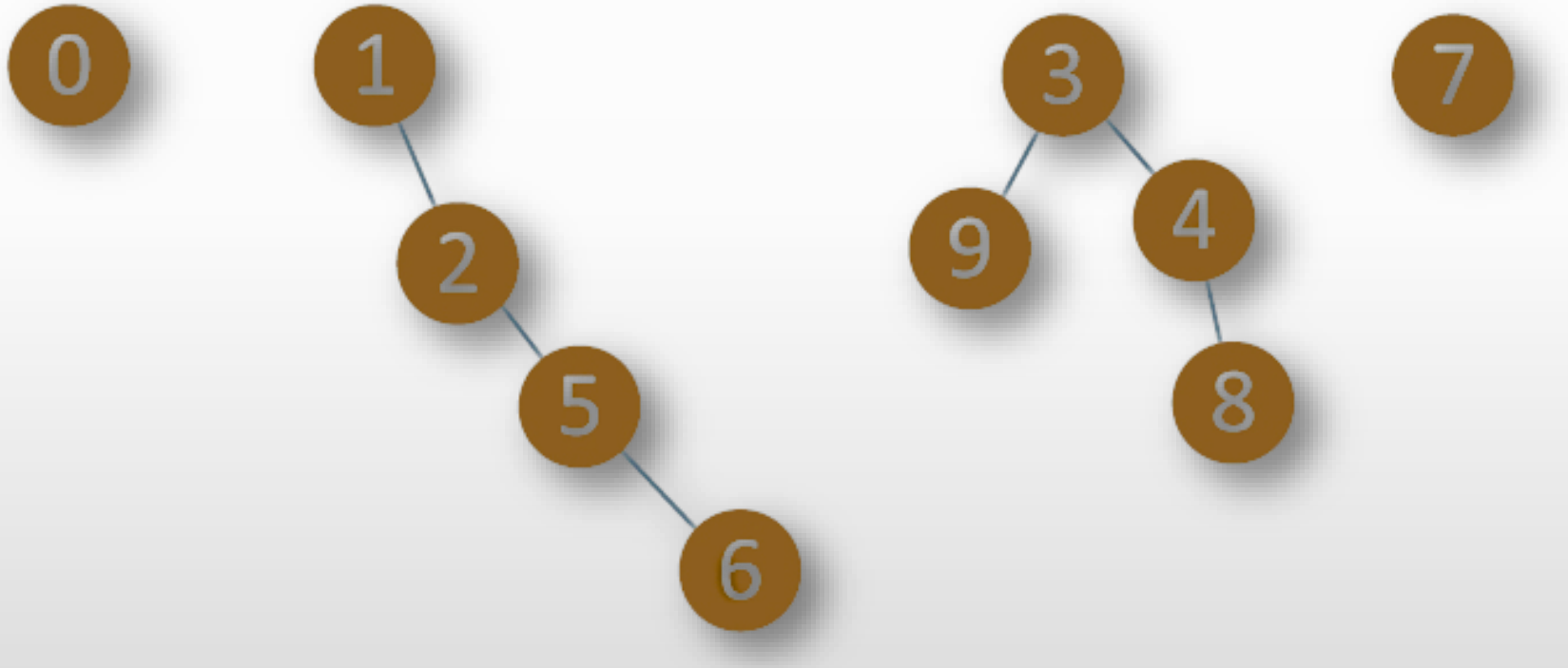
Bölüm 12: Kümeler

Veri Yapıları

Kümeler



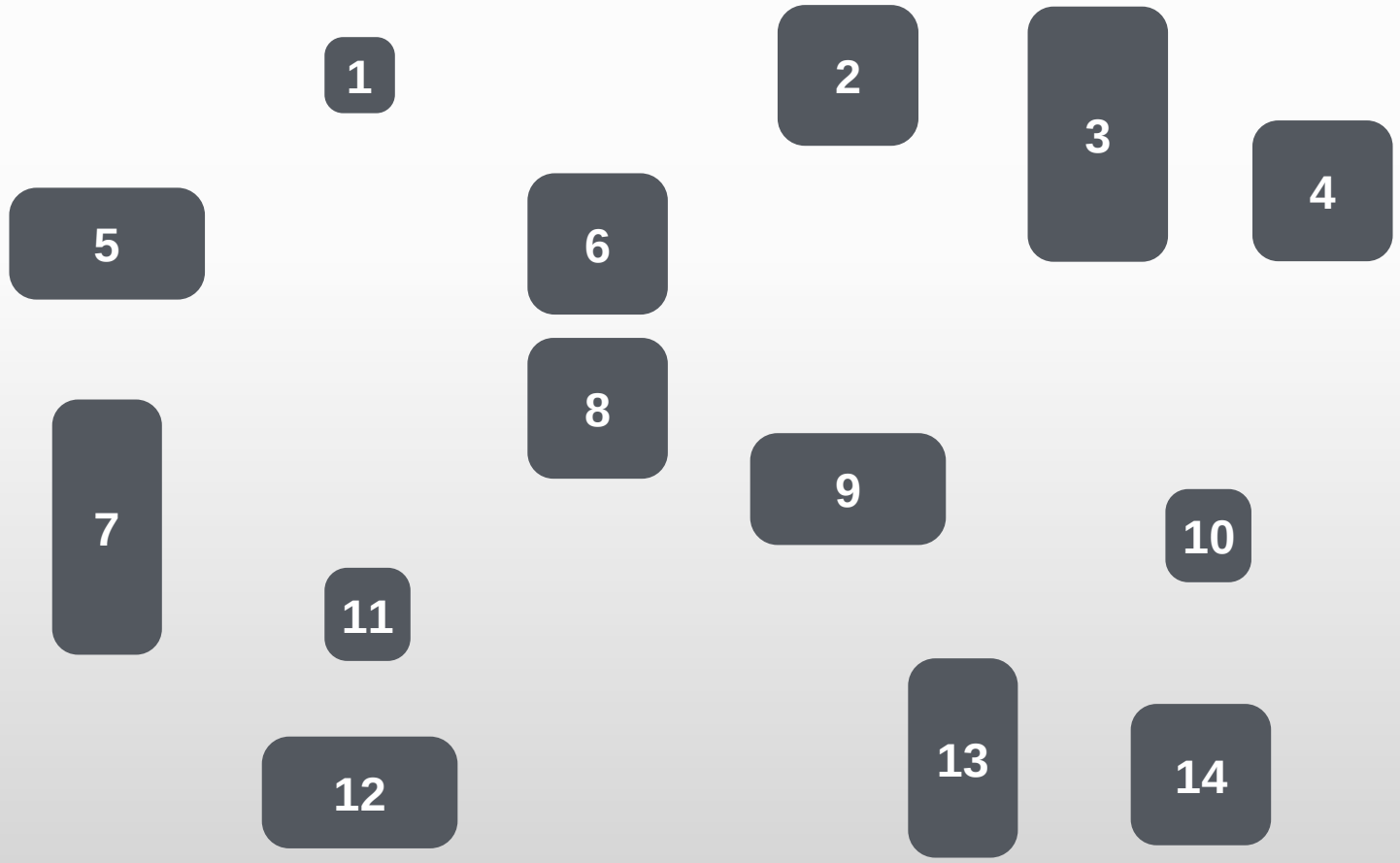
Kümeler

- Benzersiz (tekil) öğelerden oluşur.
- Çizge teorisinde, ağ algoritmalarında ve benzeri alanlarda kullanılır.
- Union-Find, bir veya daha fazla ayrık kümeden oluşan veri yapısıdır.
- Temel olarak iki ana işlemi vardır:
  - find (bul): Bir elemanın ait olduğu kümeyi bulma
  - union (birleştir): İki kümeyi birleştirme.

Mıknatıs Örneği

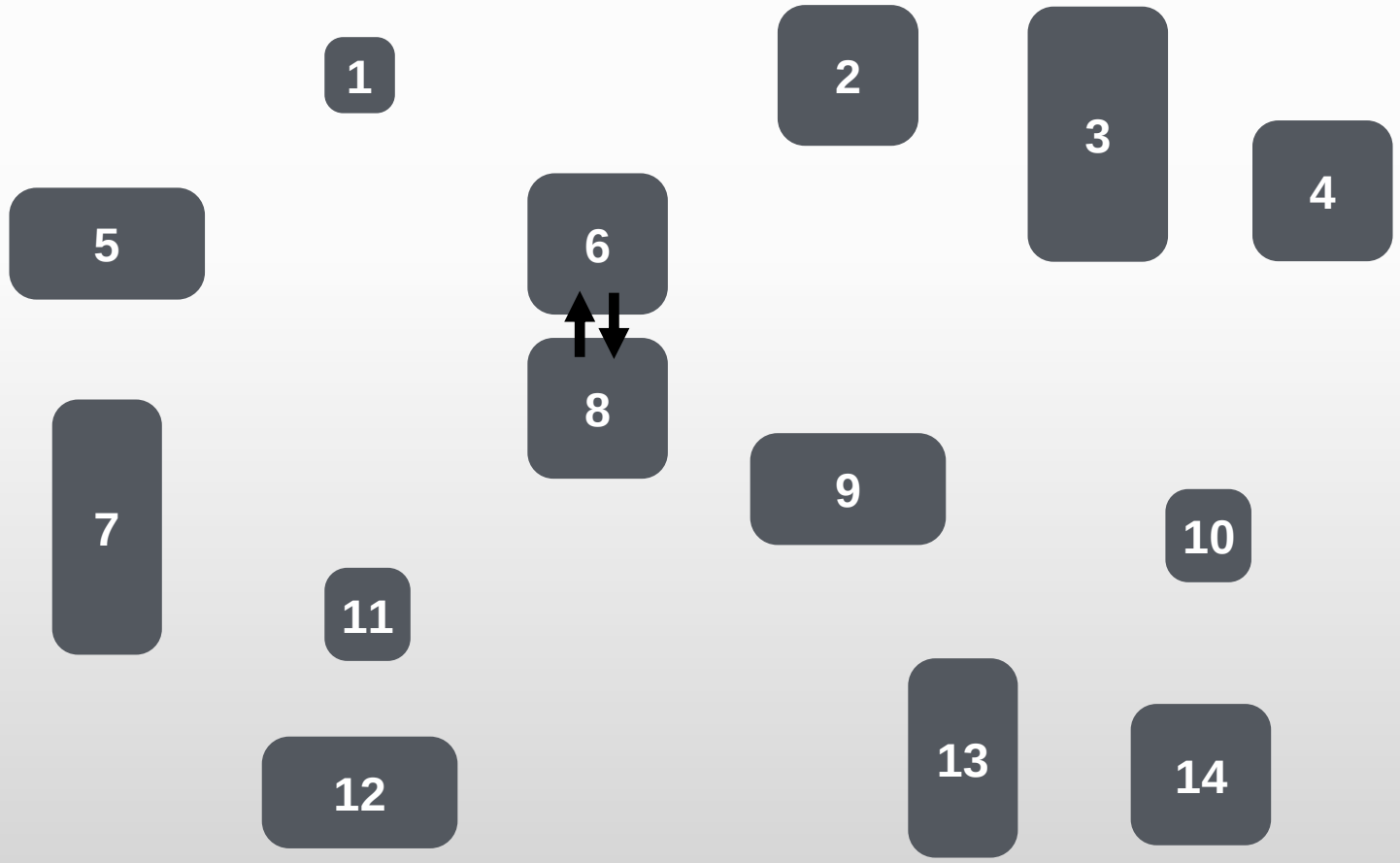
Mıknatıs Örneği

Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14



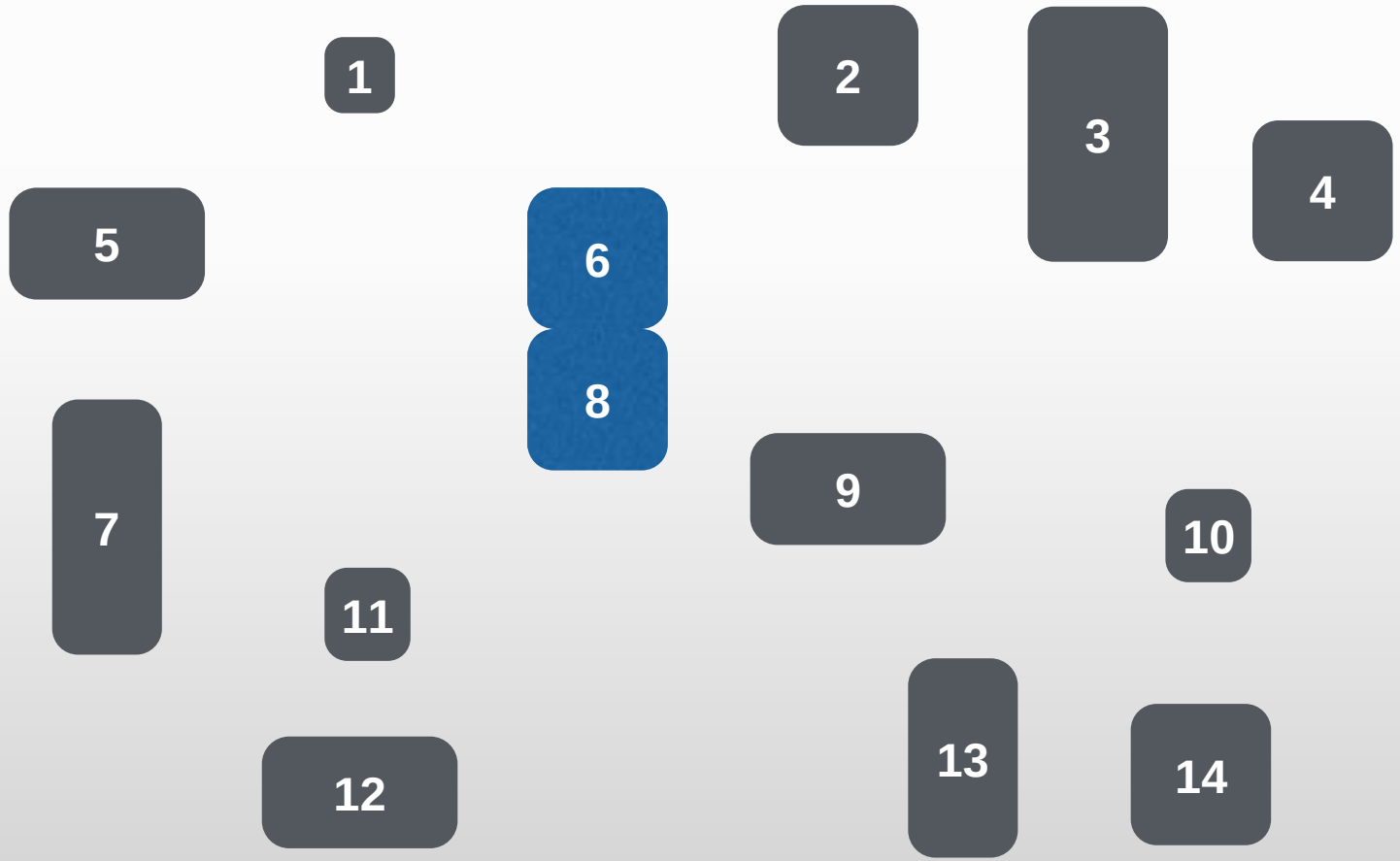
Mıknatıs Örneği

Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14



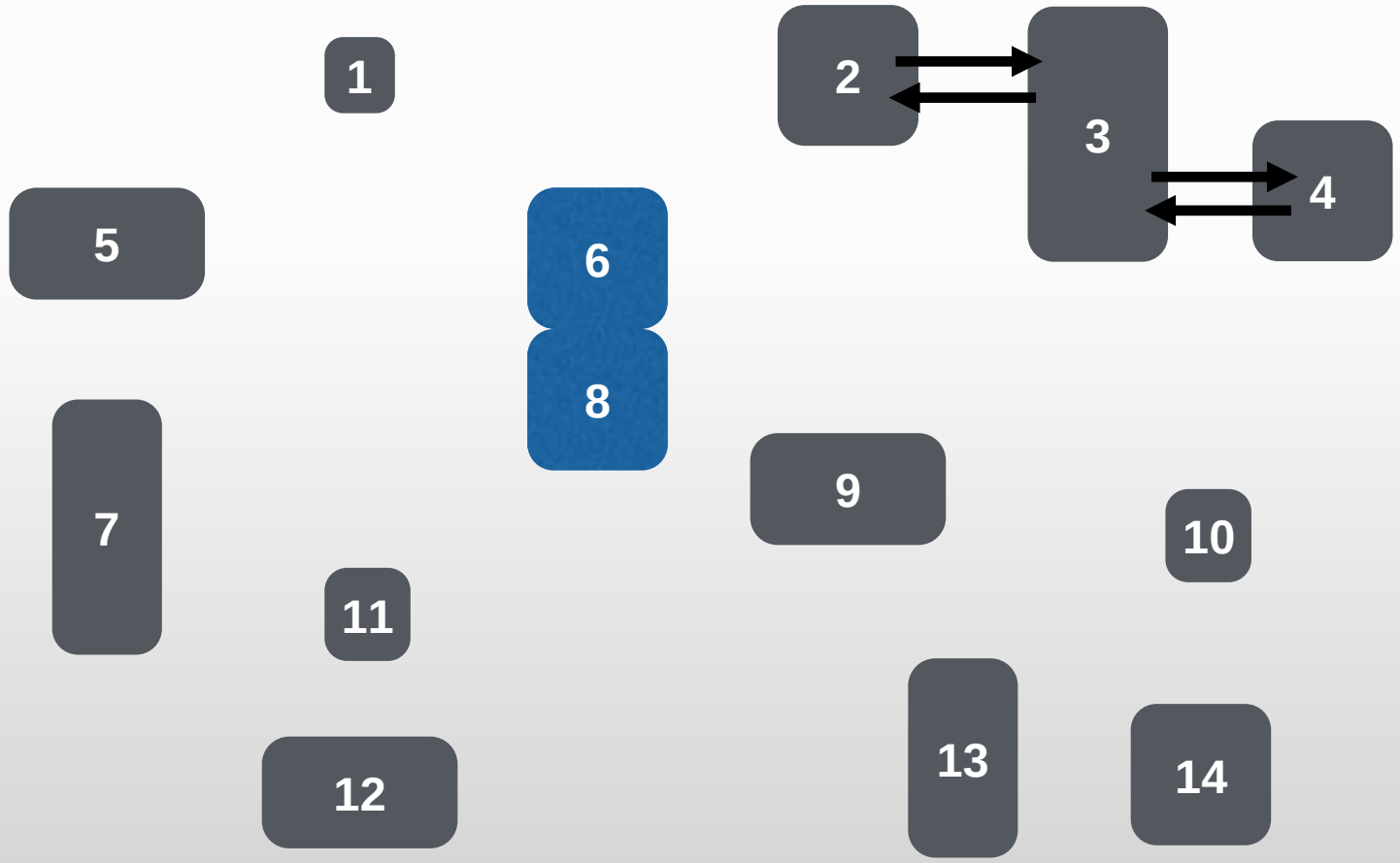
Mıknatıs Örneği

Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14



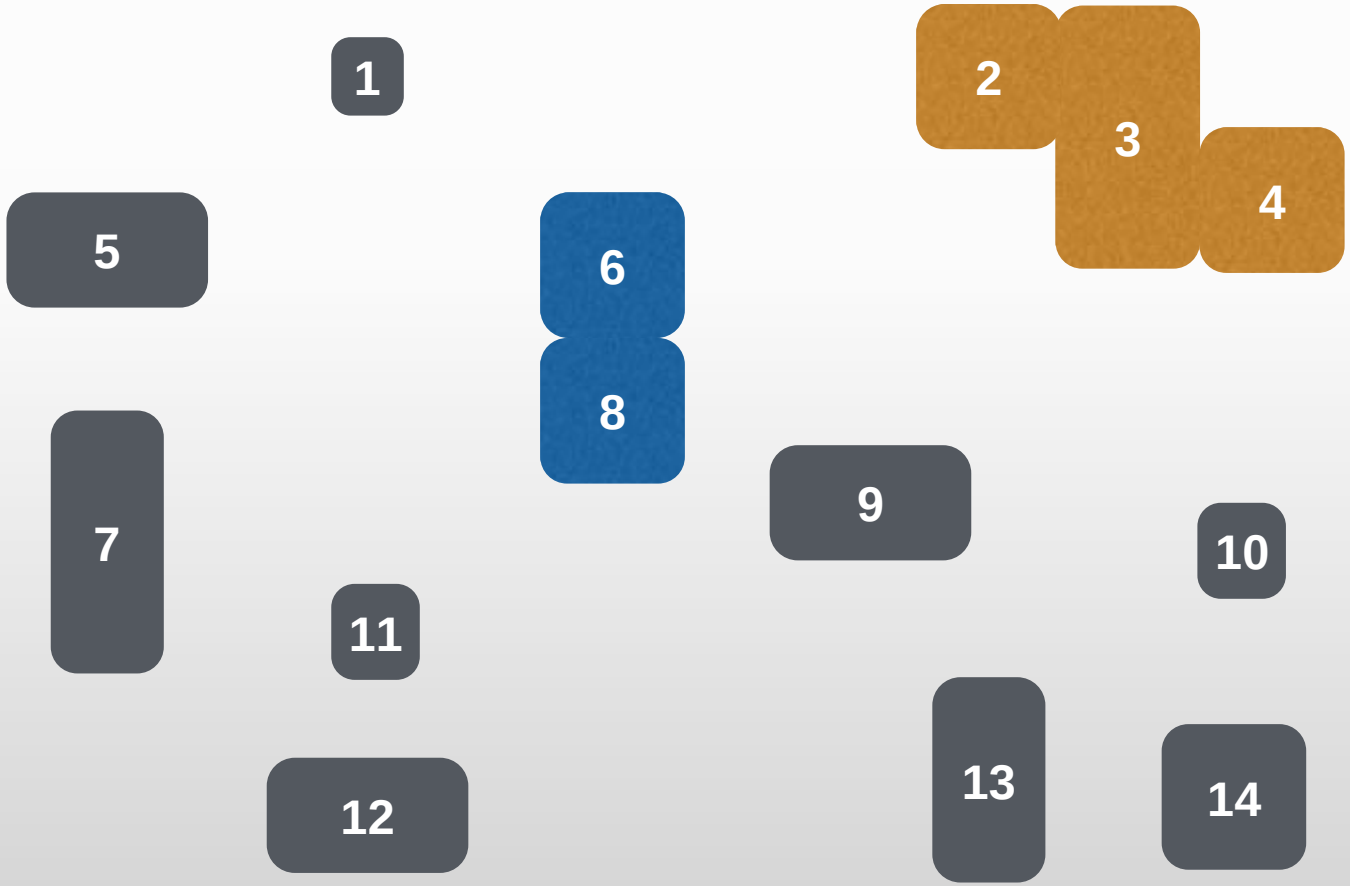
Mıknatıs Örneği

Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14



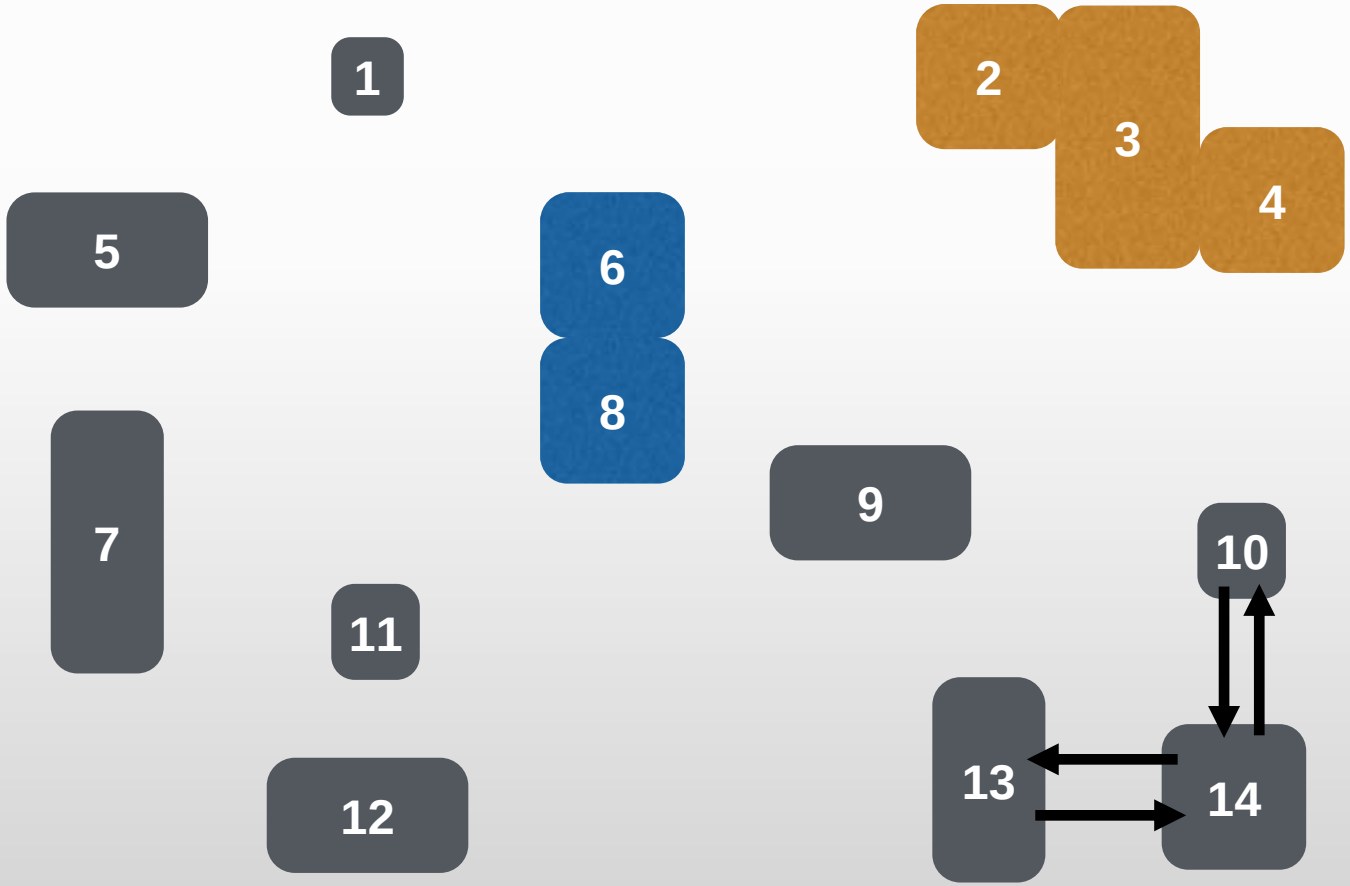
Mıknatıs Örneği

Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14



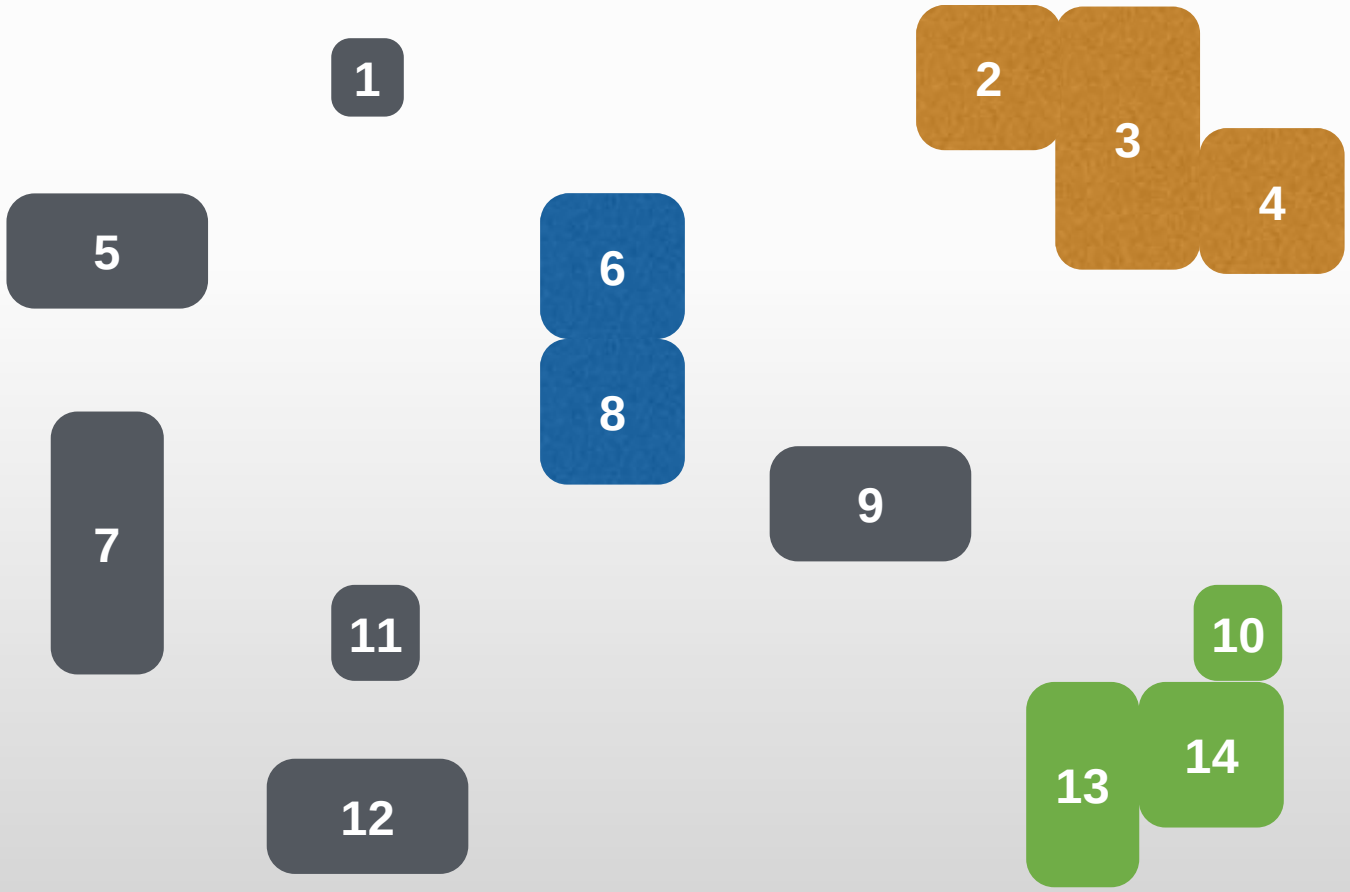
Mıknatıs Örneği

Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14



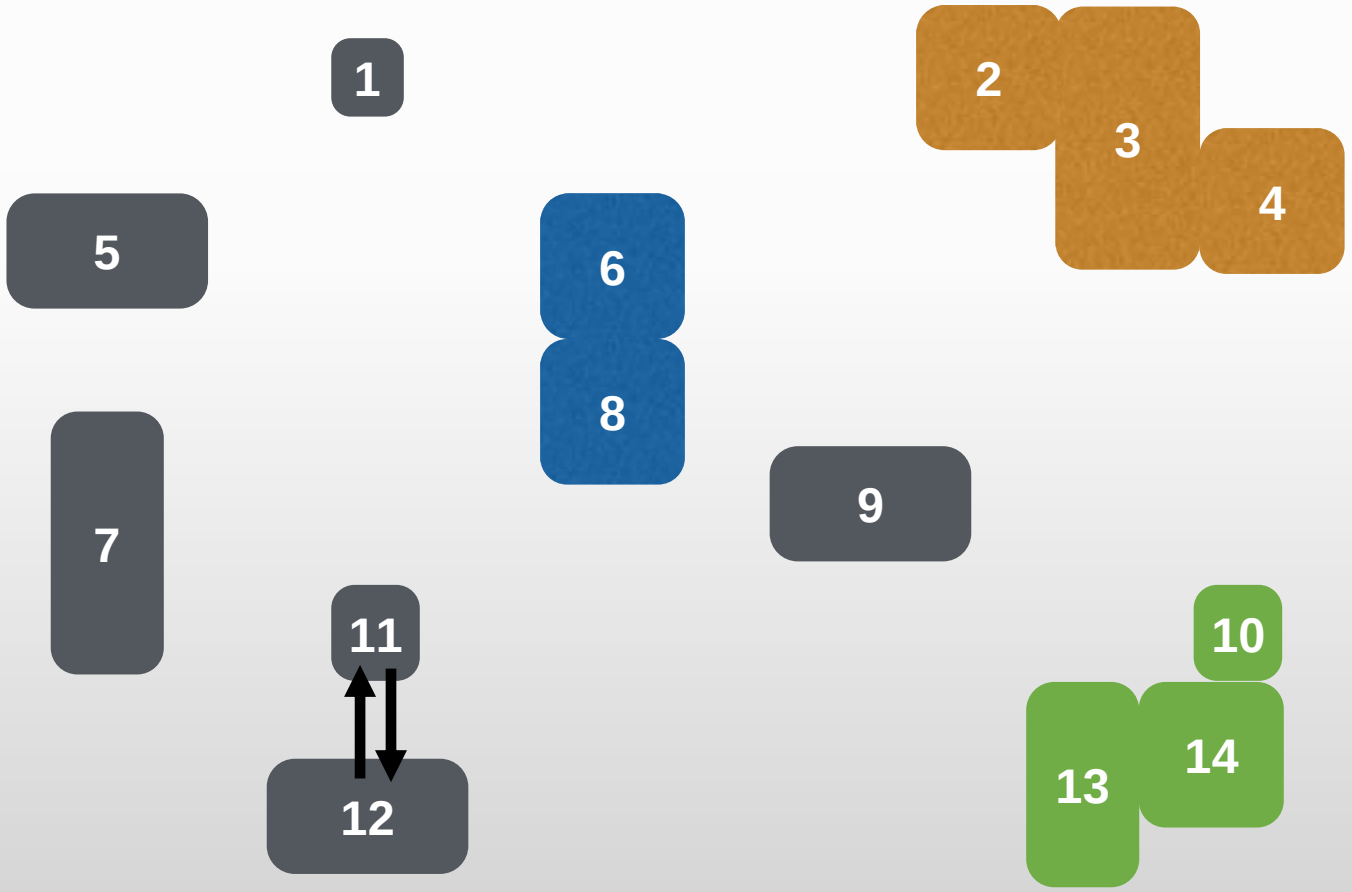
Mıknatıs Örneği

Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14



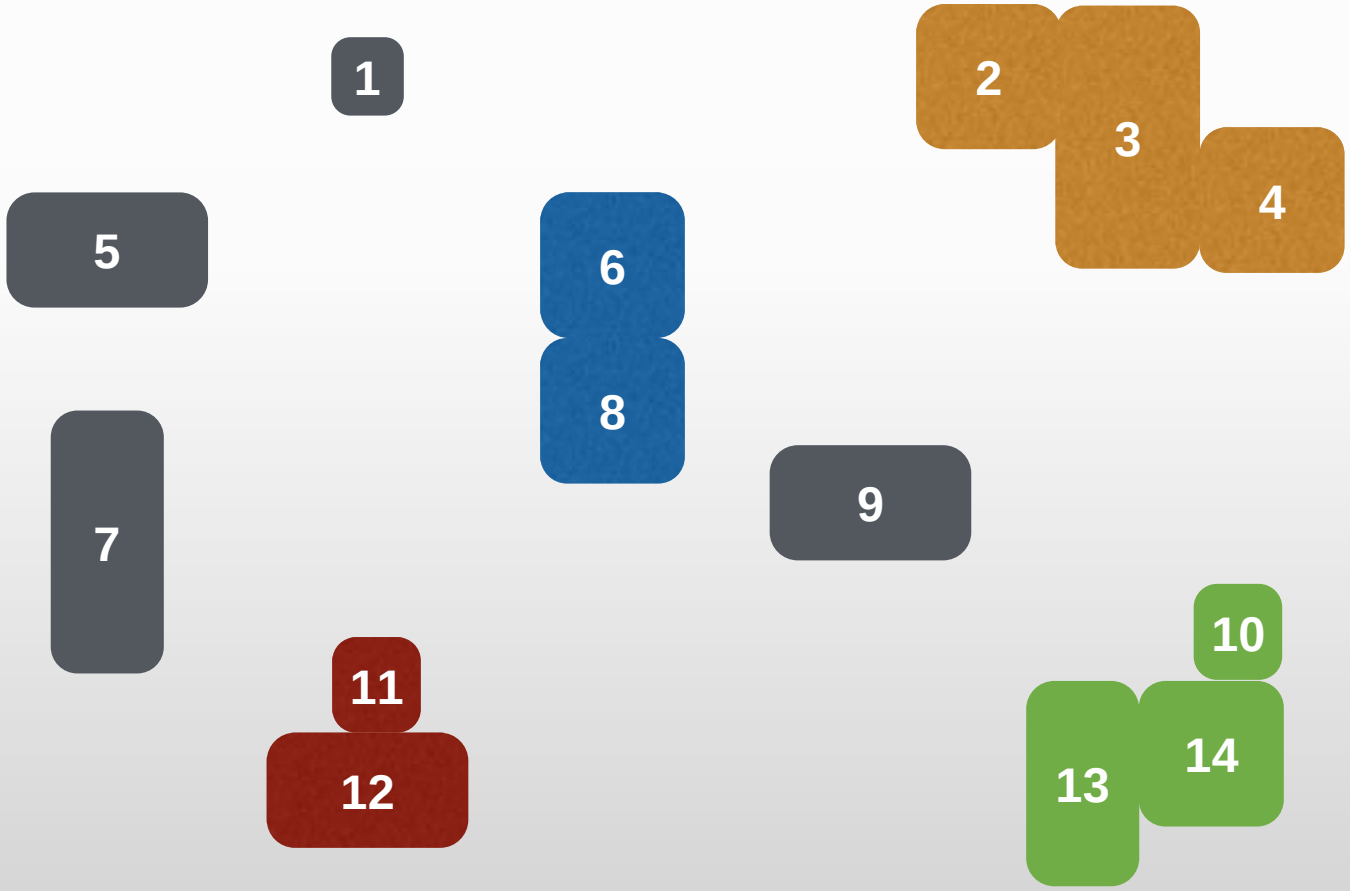
Mıknatıs Örneği

Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14



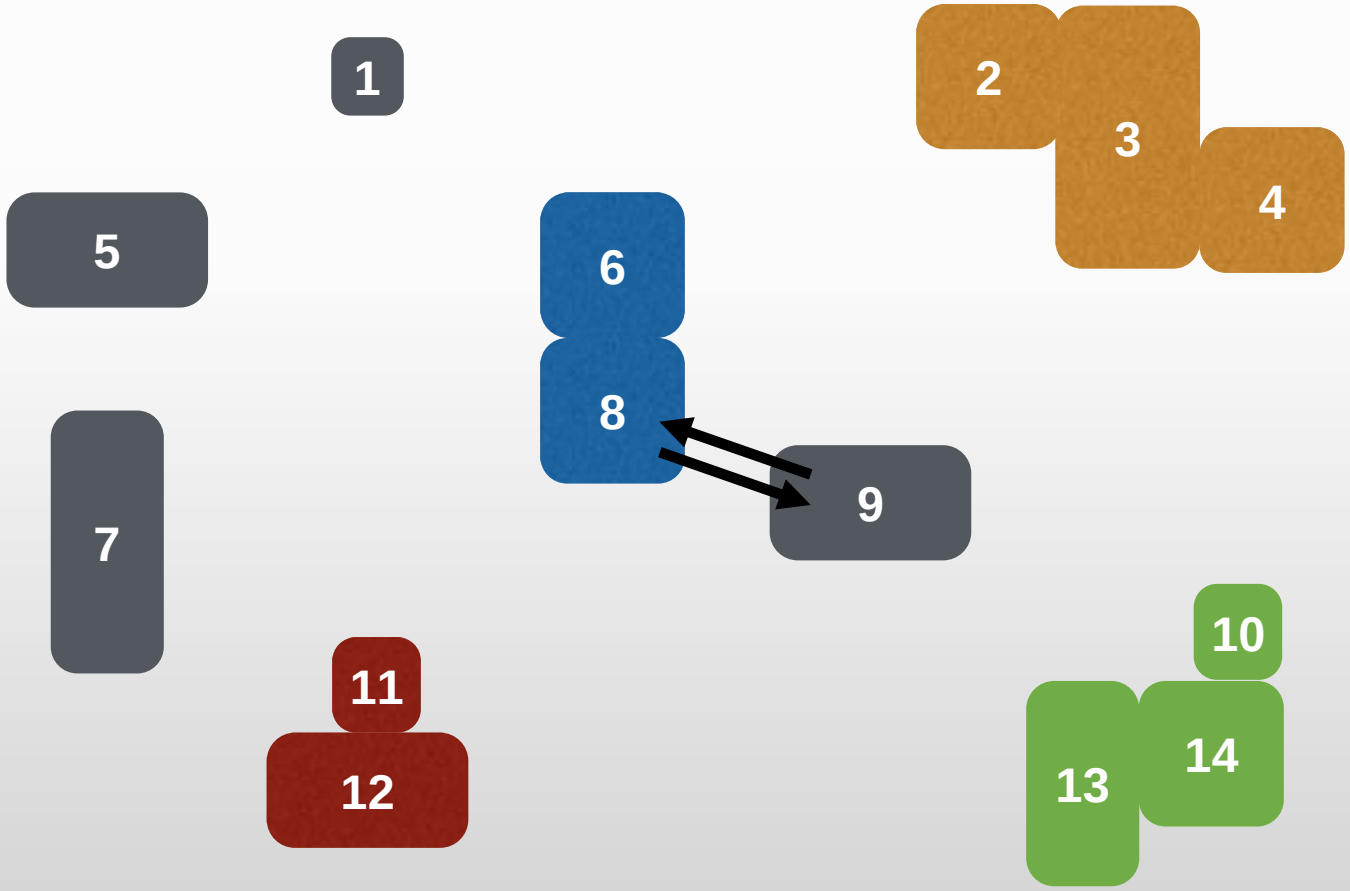
Mıknatıs Örneği

Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14



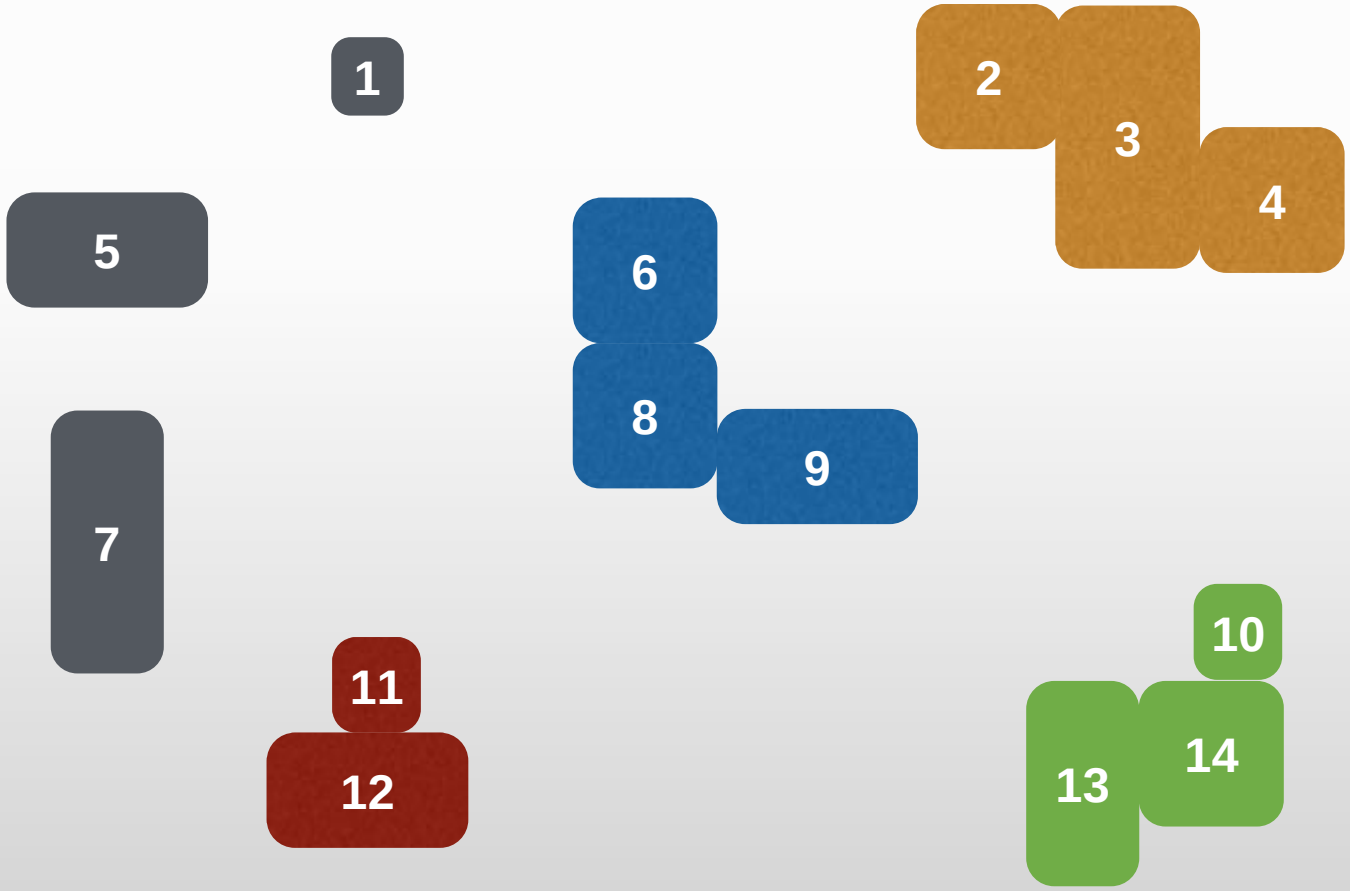
Mıknatıs Örneği

Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14



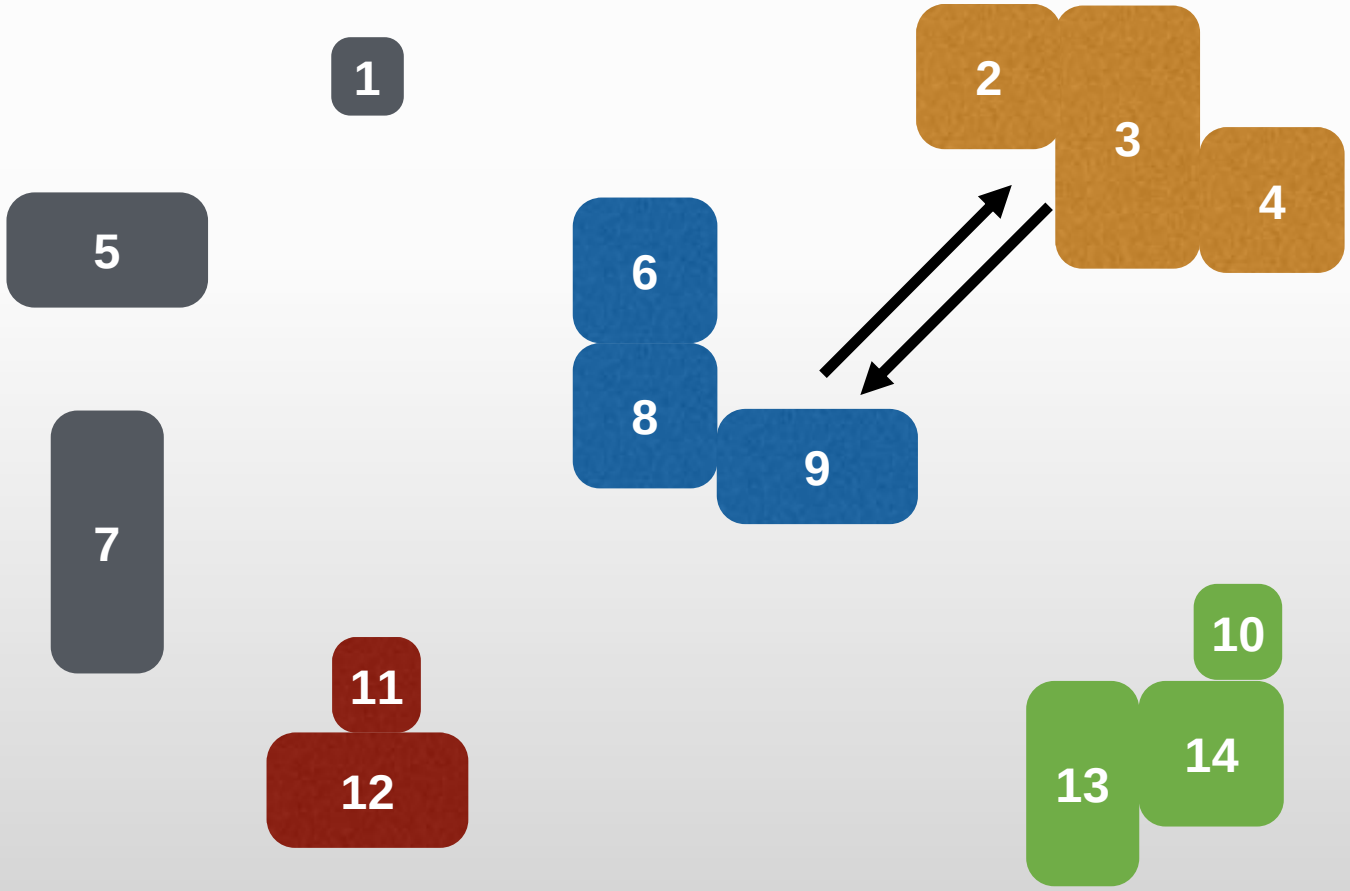
Mıknatıs Örneği

Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14



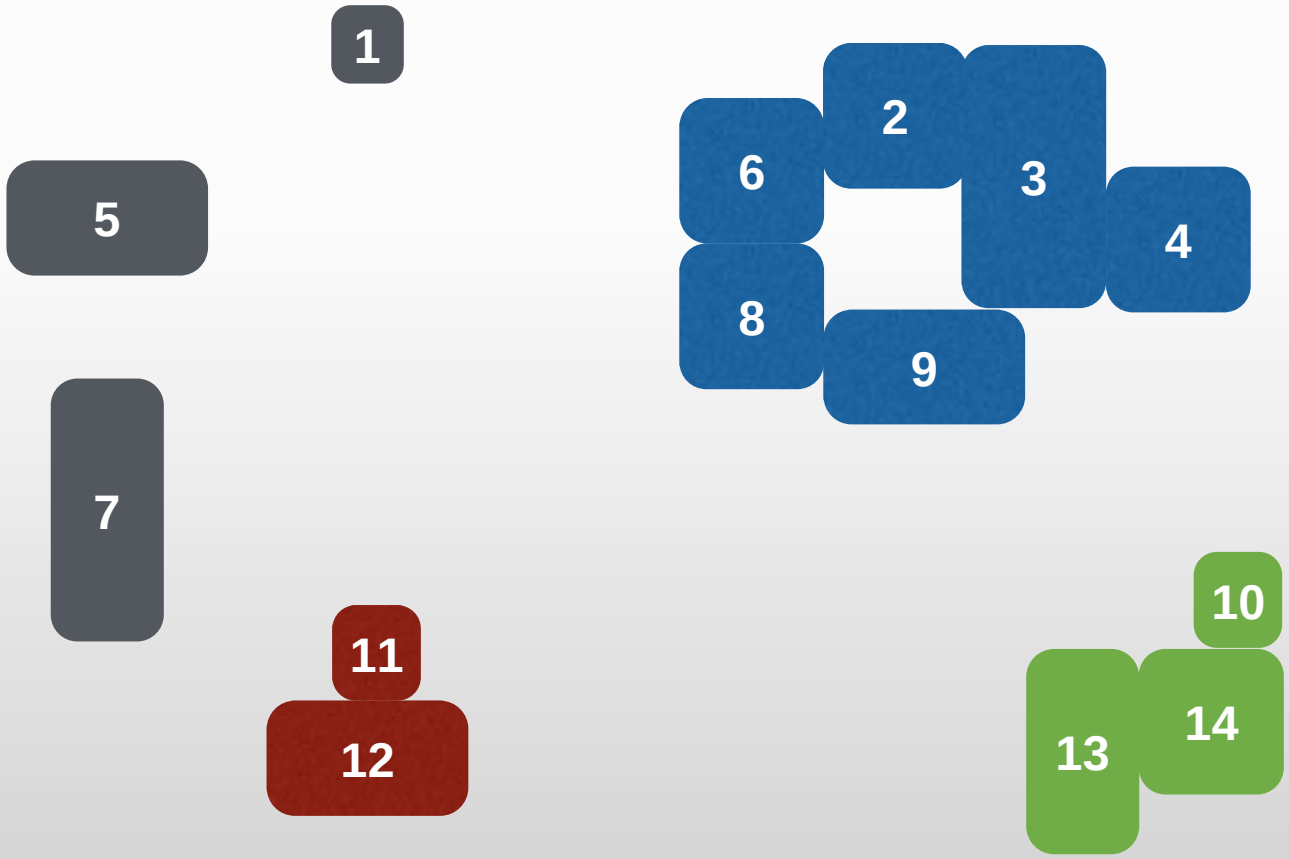
Mıknatıs Örneği

Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14



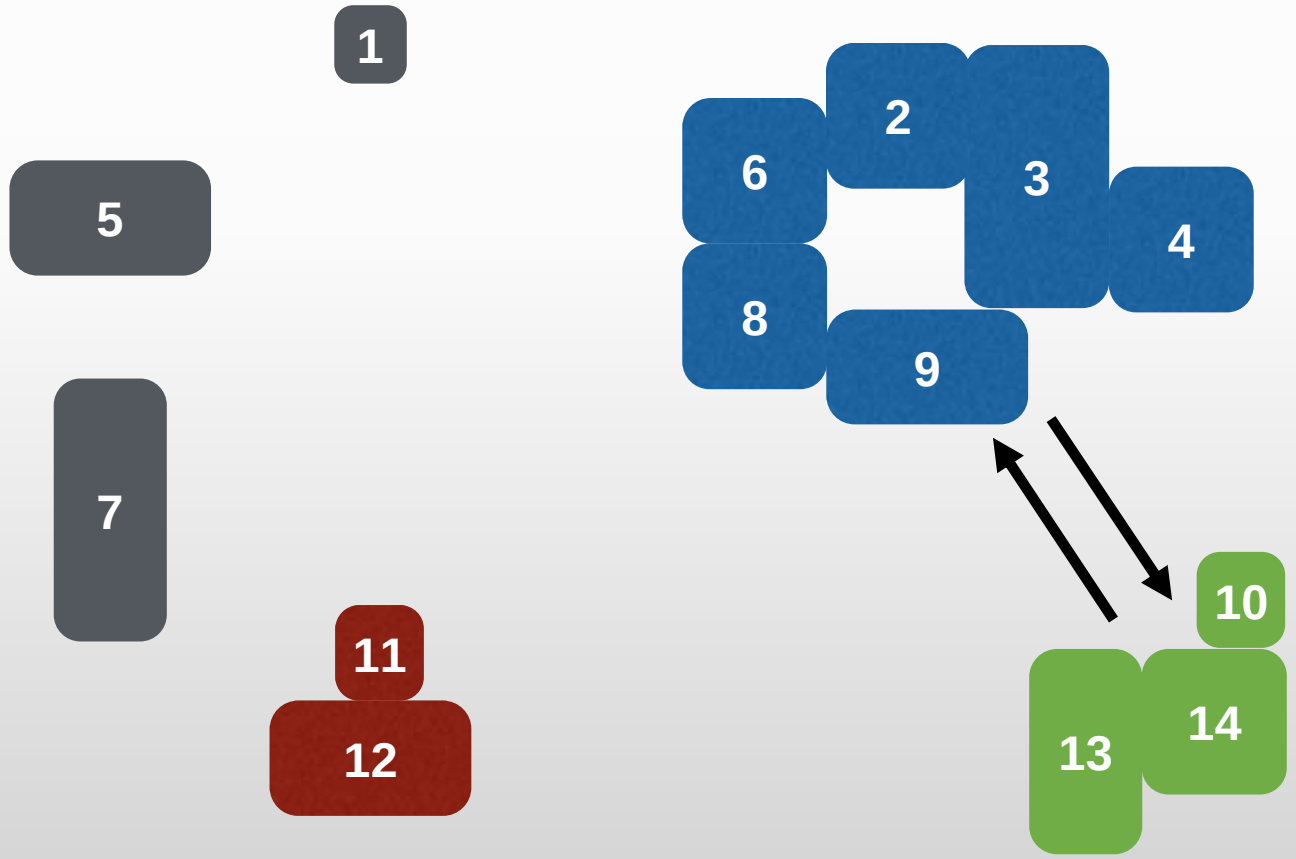
Mıknatıs Örneği

Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14



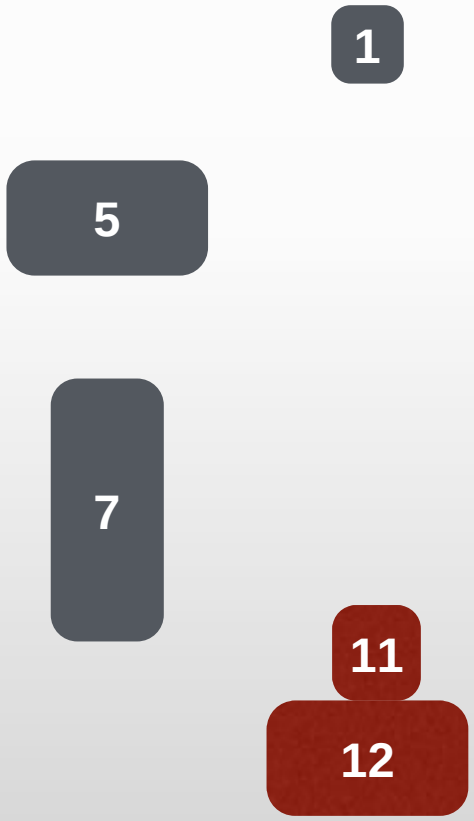
Mıknatıs Örneği

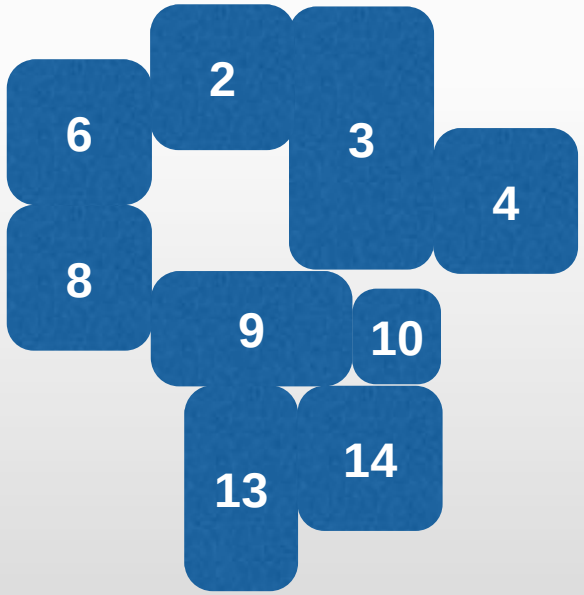
Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14



Mıknatıs Örneği

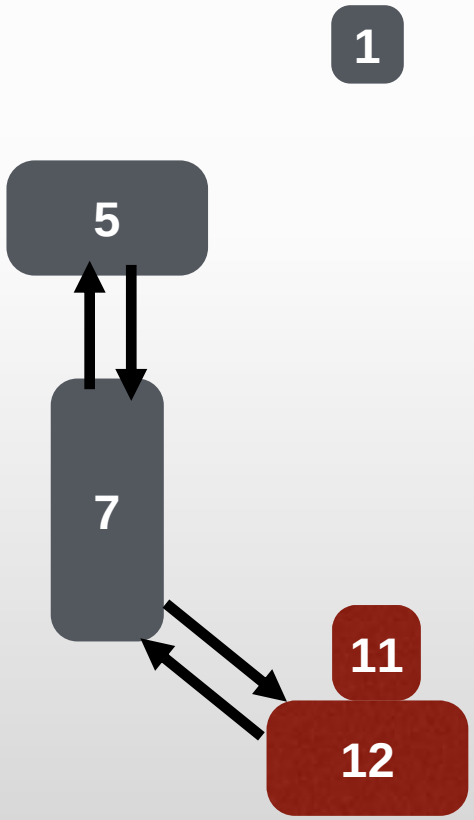
Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14

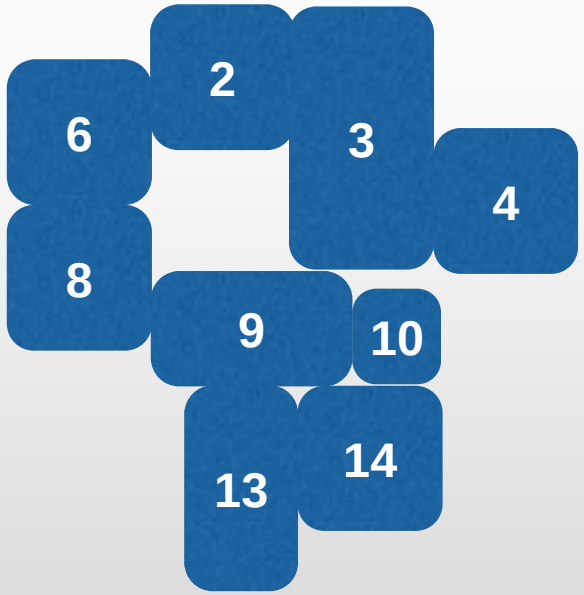




Mıknatıs Örneği

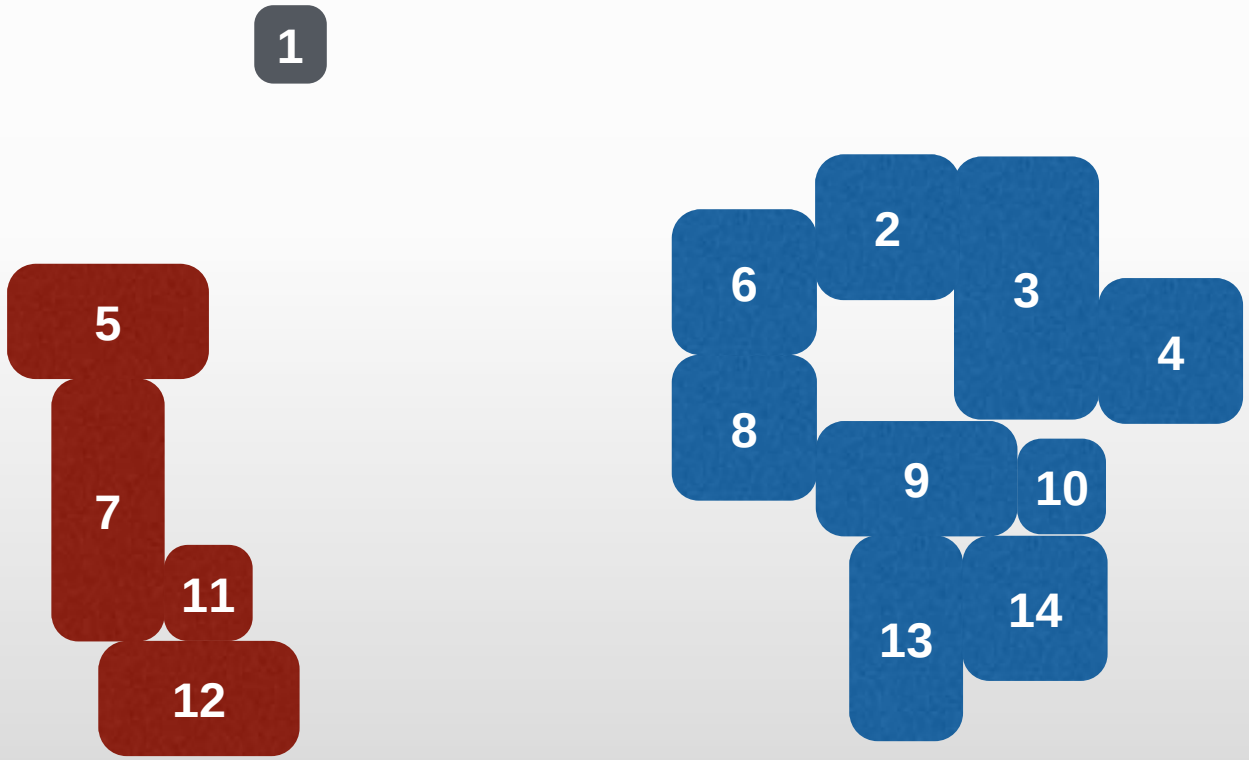
Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14





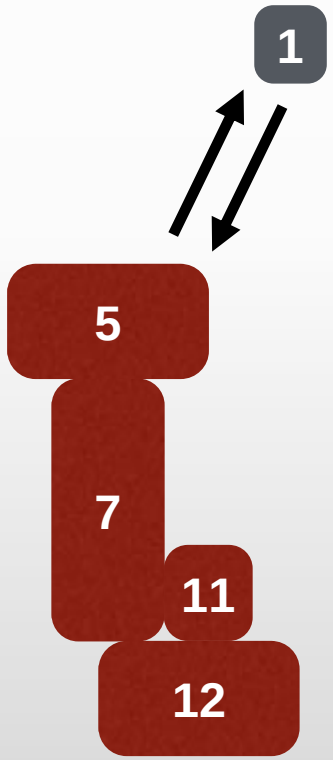
Mıknatıs Örneği

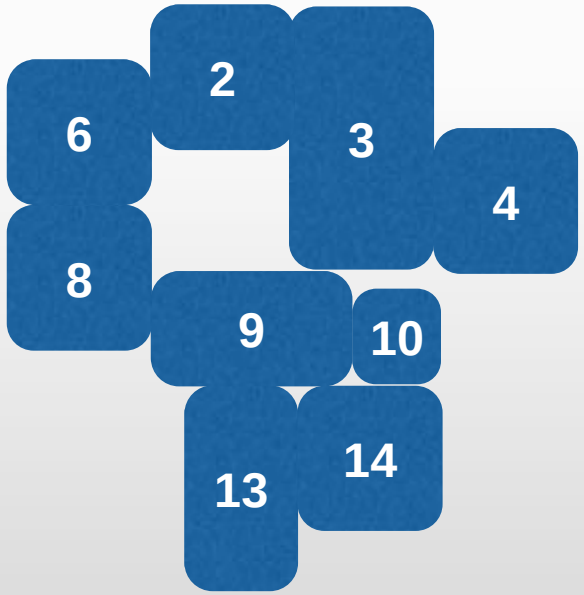
Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14



Mıknatıs Örneği

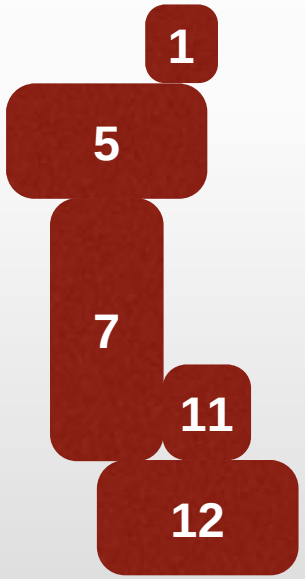
Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14

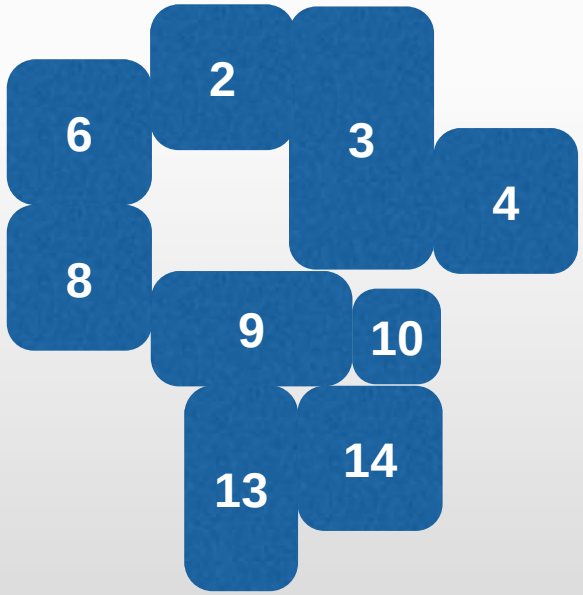




Mıknatıs Örneği

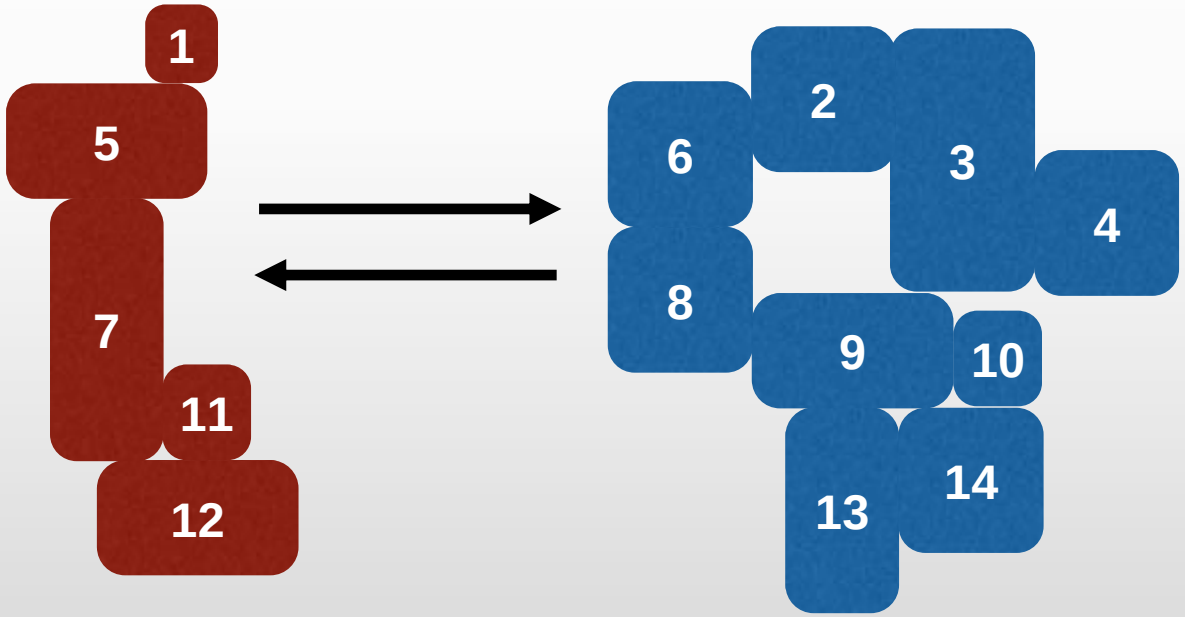
Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14





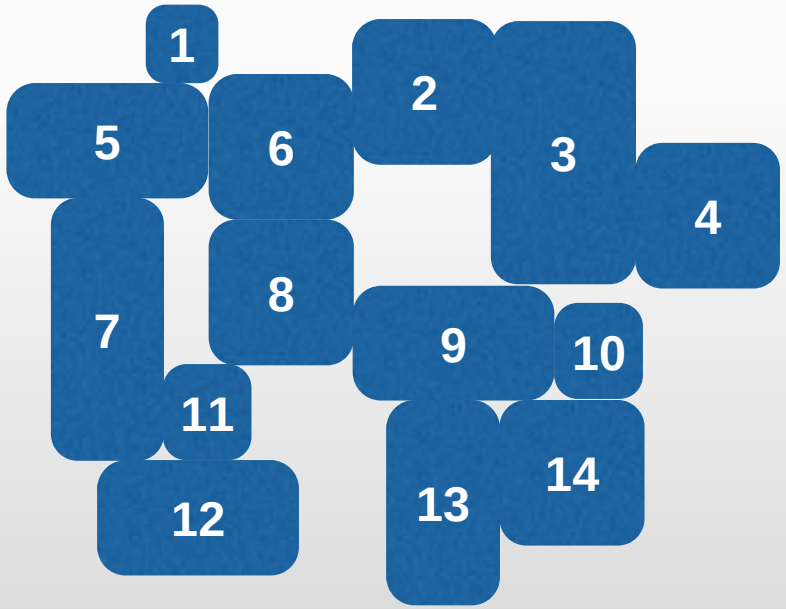
Mıknatıs Örneği

Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14



Mıknatıs Örneği

Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14



Find ve Union Fonksiyonları

- find(eleman)
  - Elemanı içeren kümenin adını döndürür.
  - Arama ağaçlarındaki arama işlemi ile aynı değildir.
- union(küme1, küme2)
  - İki kümenin yerine yeni bir küme yaratır.

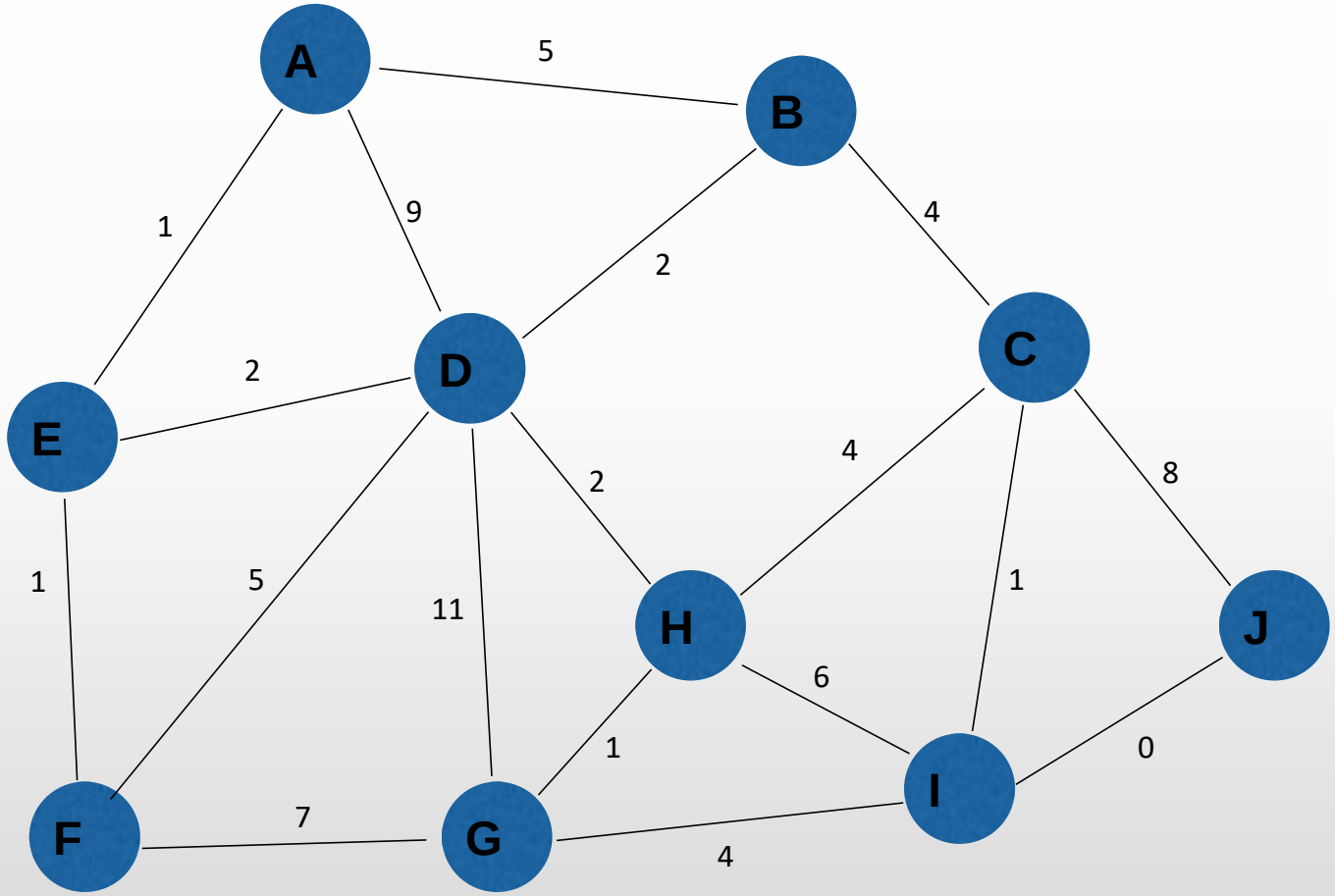
Find ve Union Fonksiyonları

```
find(x)
    if (x.ata != x)
        x.ata = find(x.ata)
    return x.ata

union(küme1, küme2)
    kök1 = find(küme1)
    kök2 = find(küme2)

    if (kök1 != kök2)
        // Bir kümenin kökünü diğer kümenin köküne ata
        kök1.ata = kök2
```

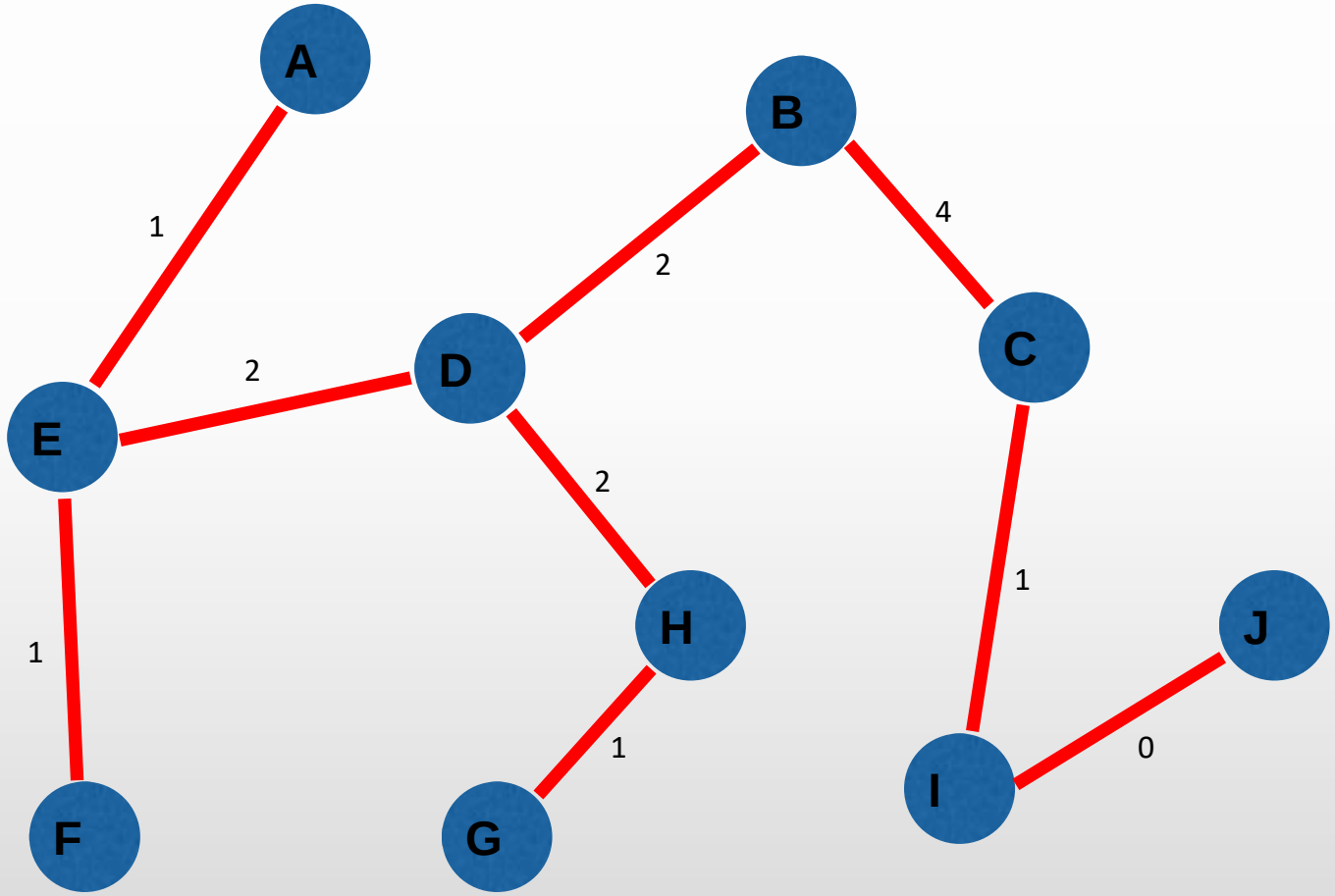
Kruskal Minimum Kapsayan Ağaç

- Çizgedeki tüm düğümleri minimum maliyetle kapsayan ağaçtır.
- Toplam kenar maliyeti en düşük olan kenarlar alt kümesidir.

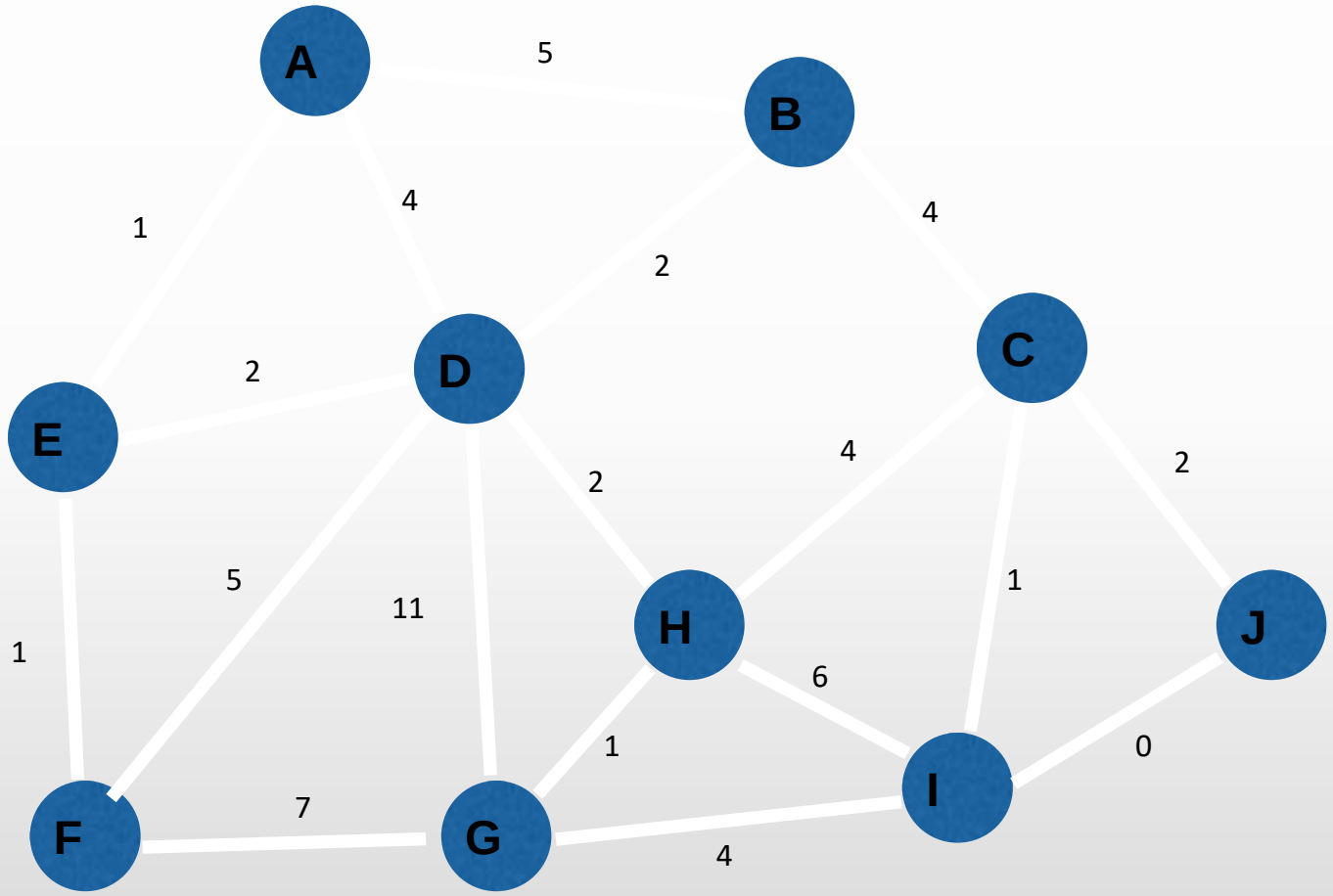
Kruskal Minimum Kapsayan Ağaç



Kruskal Minimum Kapsayan Ağaç

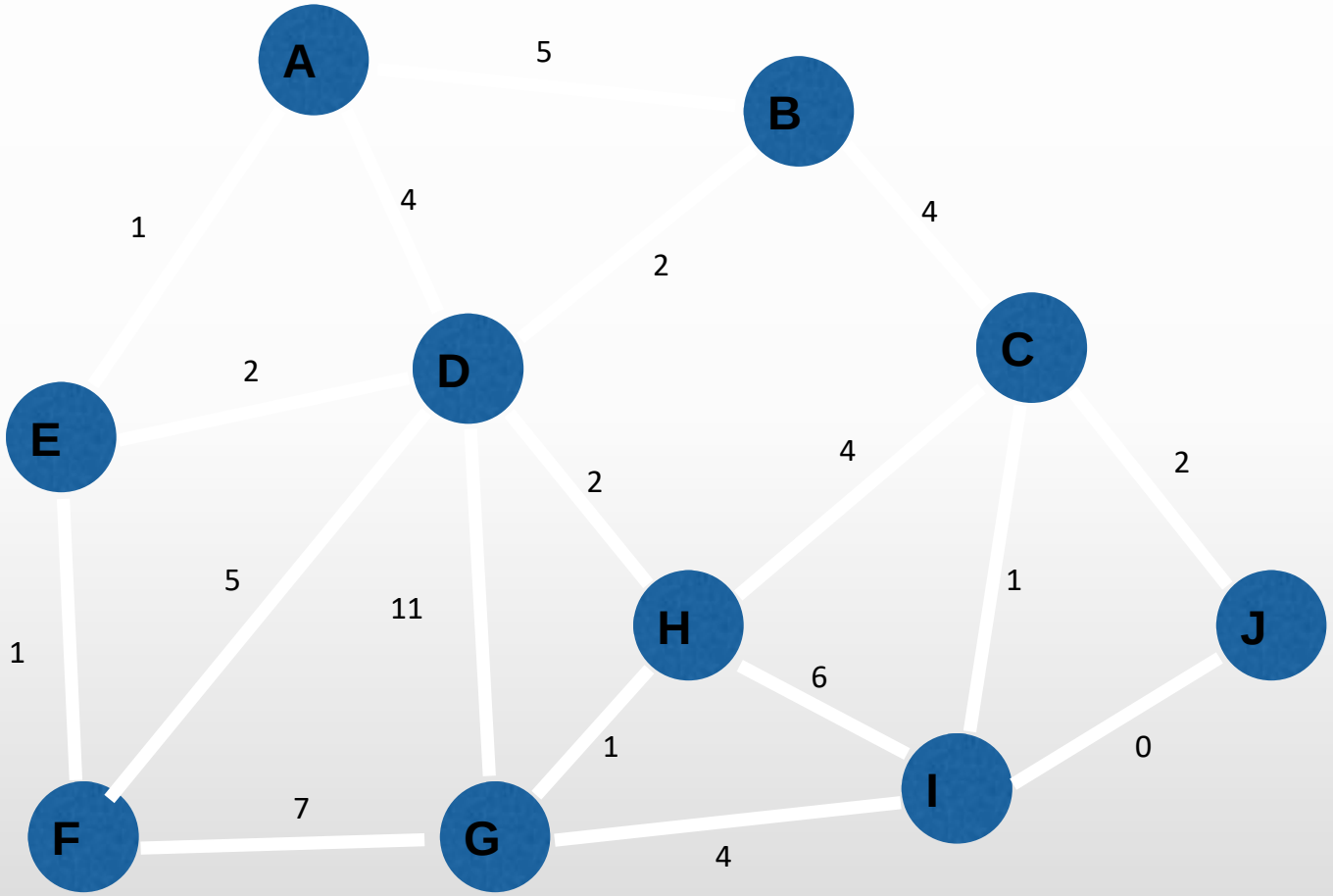


Kruskal Algoritması Uygulama



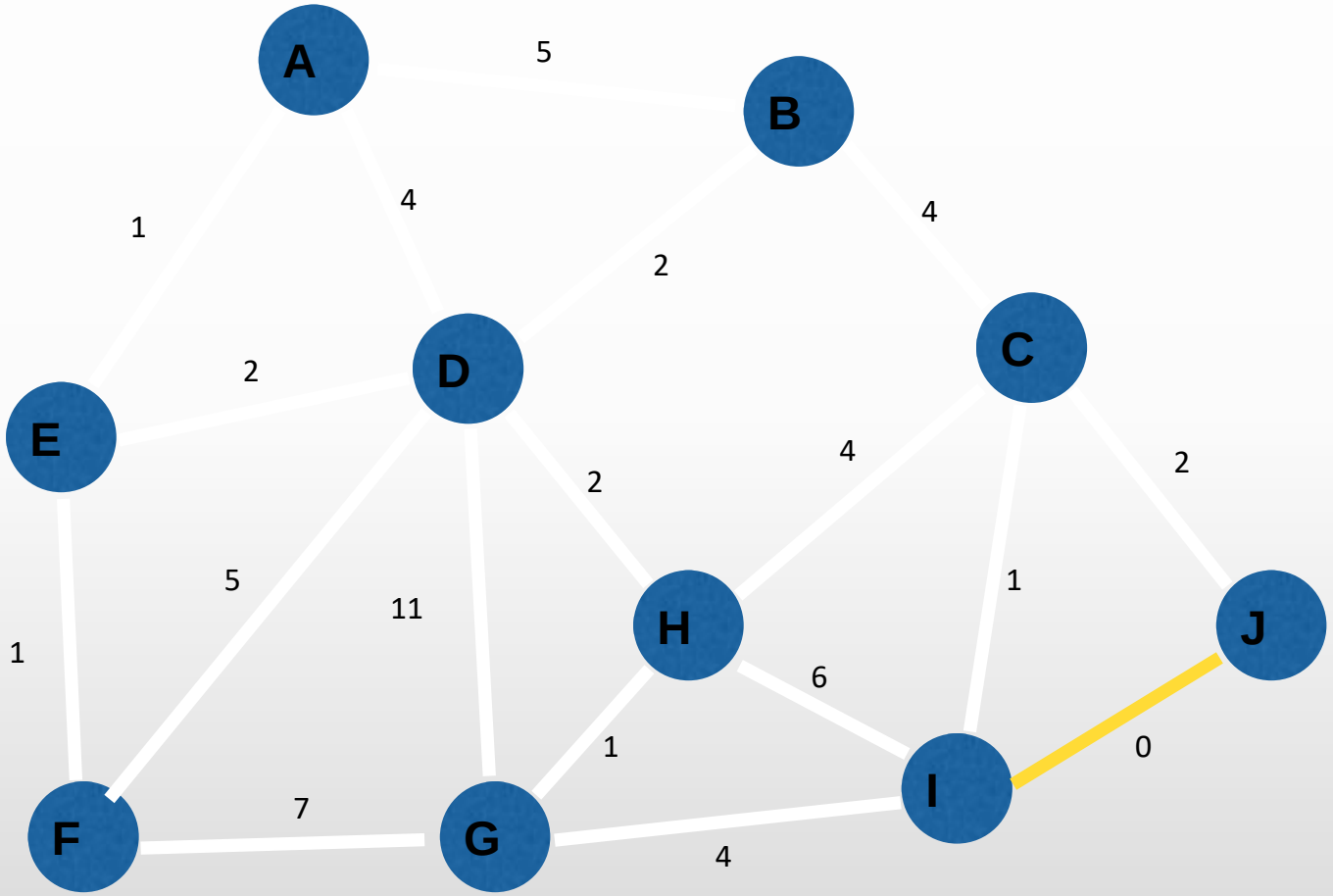
Kruskal Algoritması Uygulama

```
I -> J = 0
A -> E = 1
C -> I = 1
E -> F = 1
G -> H = 1
B -> D = 2
C -> J = 2
D -> E = 2
D -> H = 2
A -> D = 4
B -> C = 4
C -> H = 4
G -> I = 4
A -> B = 5
D -> F = 5
H -> I = 6
F -> G = 7
D -> G = 11
```



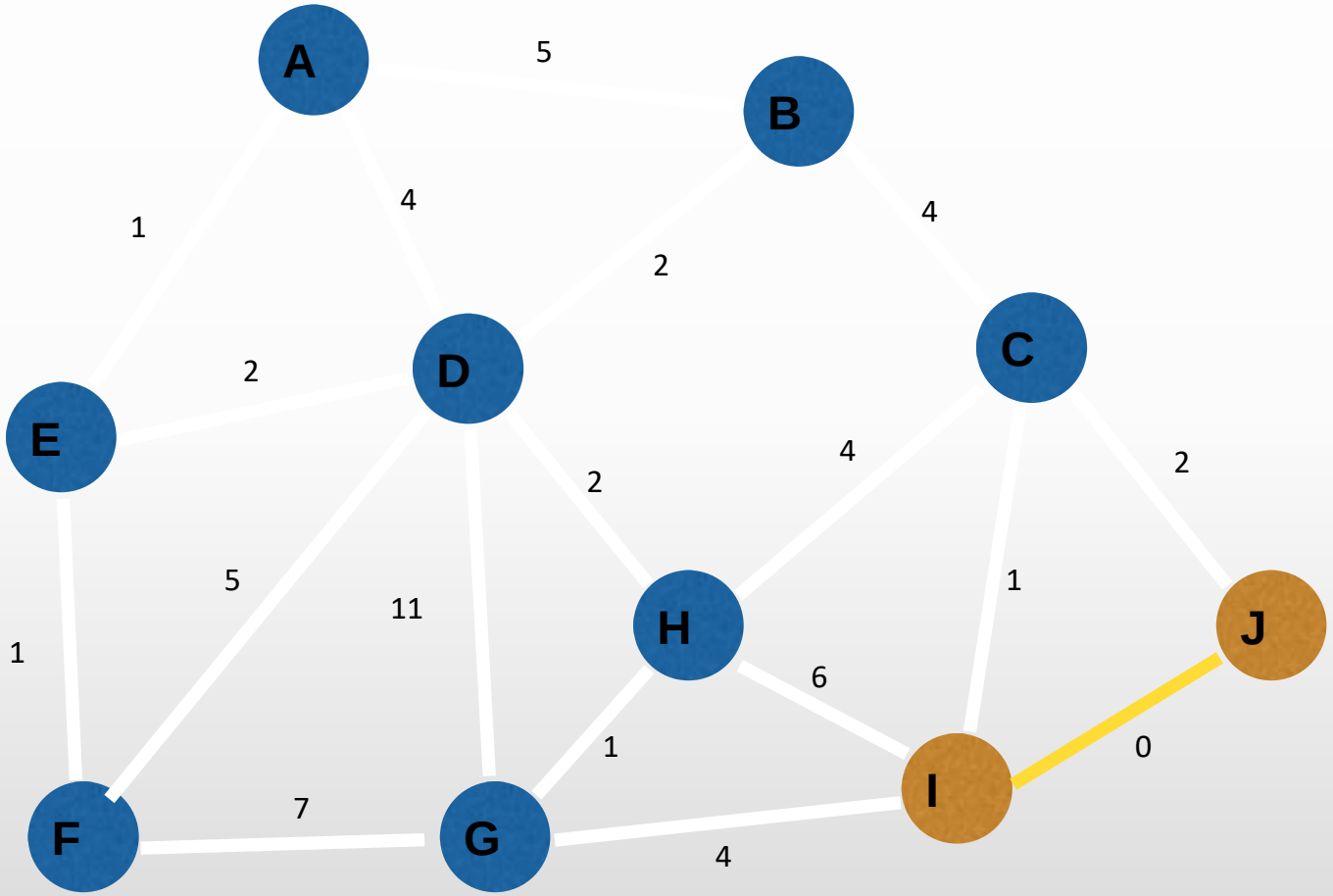
Kruskal Algoritması Uygulama

```
I -> J = 0
A -> E = 1
C -> I = 1
E -> F = 1
G -> H = 1
B -> D = 2
C -> J = 2
D -> E = 2
D -> H = 2
A -> D = 4
B -> C = 4
C -> H = 4
G -> I = 4
A -> B = 5
D -> F = 5
H -> I = 6
F -> G = 7
D -> G = 11
```



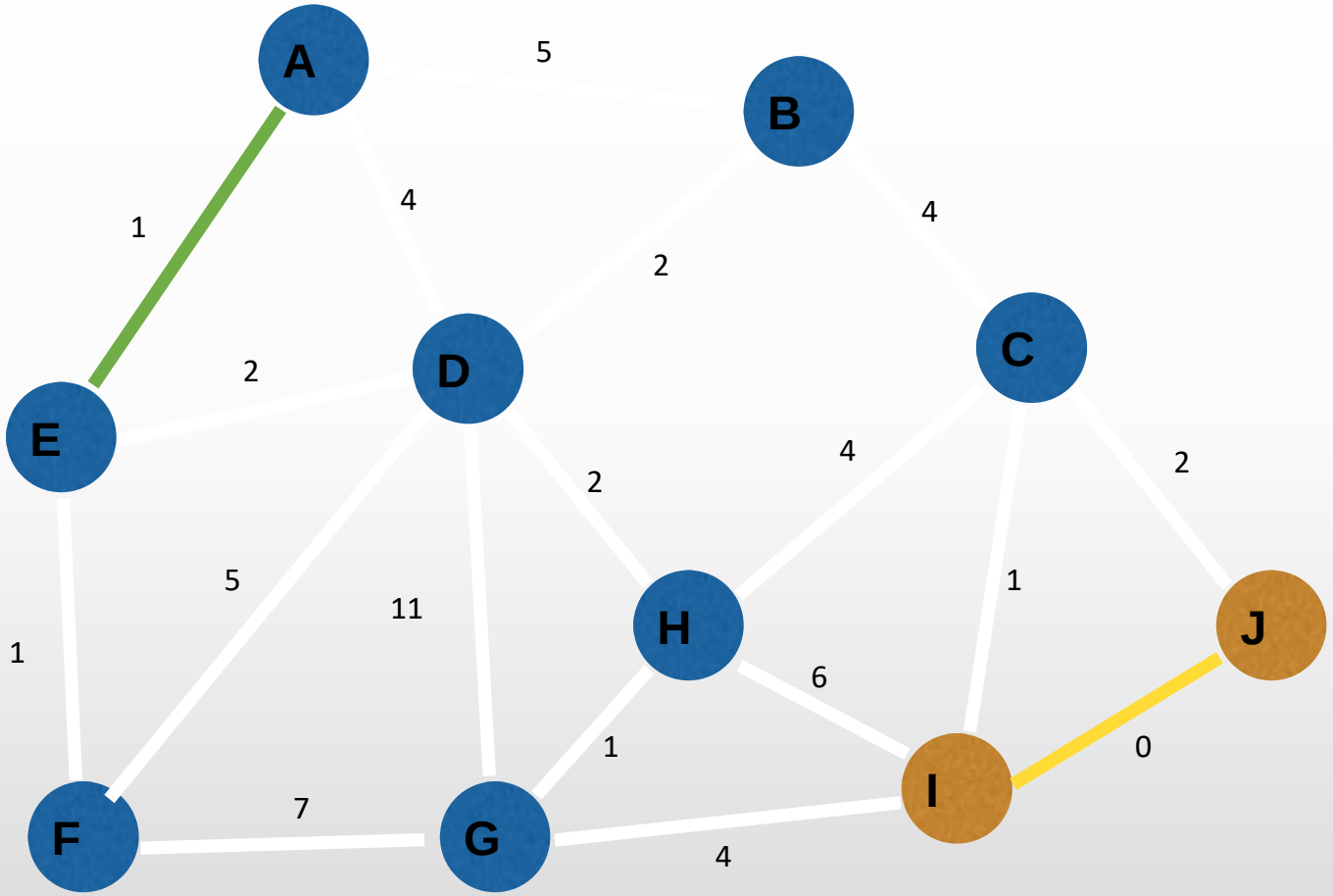
Kruskal Algoritması Uygulama

```
I -> J = 0
A -> E = 1
C -> I = 1
E -> F = 1
G -> H = 1
B -> D = 2
C -> J = 2
D -> E = 2
D -> H = 2
A -> D = 4
B -> C = 4
C -> H = 4
G -> I = 4
A -> B = 5
D -> F = 5
H -> I = 6
F -> G = 7
D -> G = 11
```



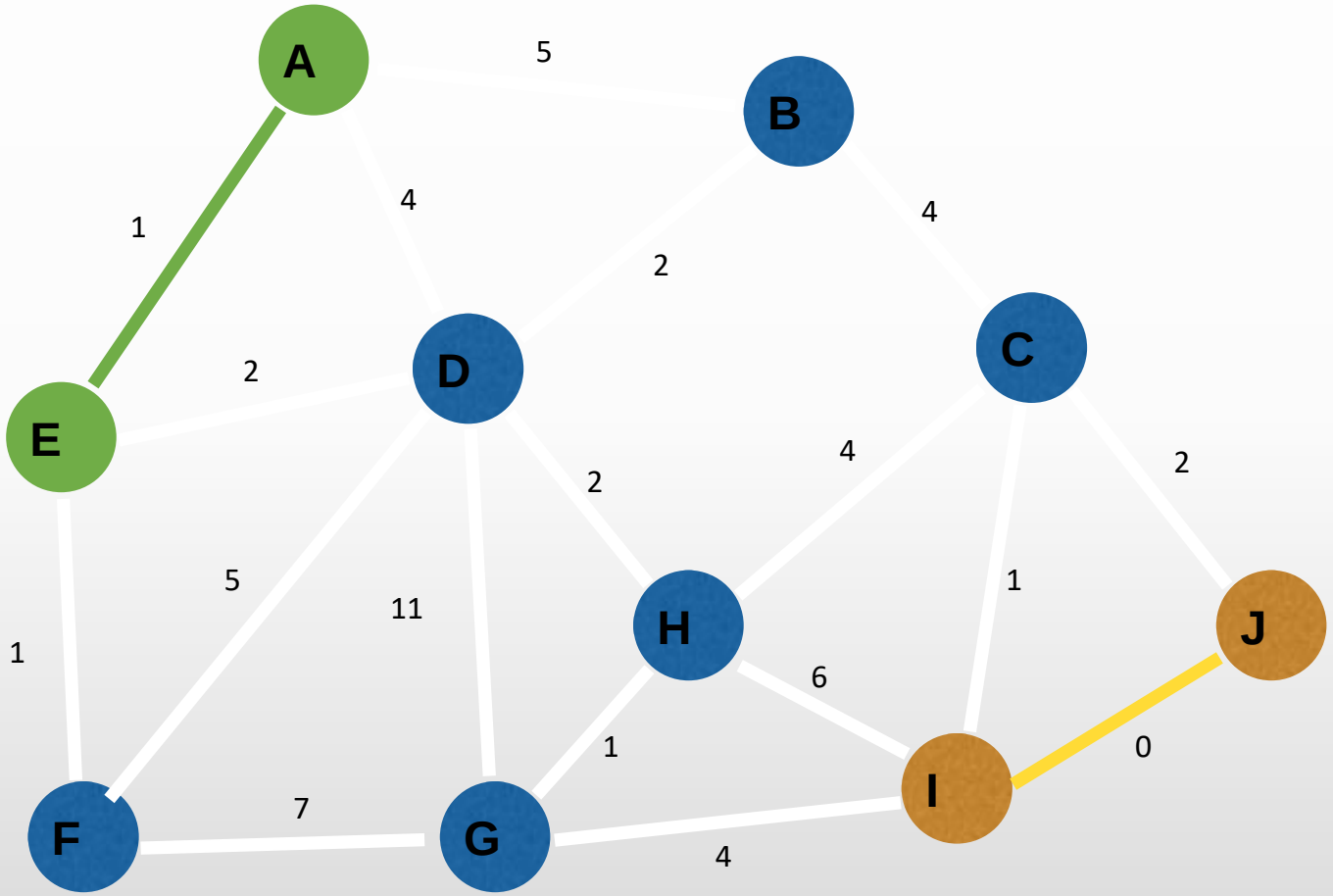
Kruskal Algoritması Uygulama

```
I -> J = 0
A -> E = 1
C -> I = 1
E -> F = 1
G -> H = 1
B -> D = 2
C -> J = 2
D -> E = 2
D -> H = 2
A -> D = 4
B -> C = 4
C -> H = 4
G -> I = 4
A -> B = 5
D -> F = 5
H -> I = 6
F -> G = 7
D -> G = 11
```



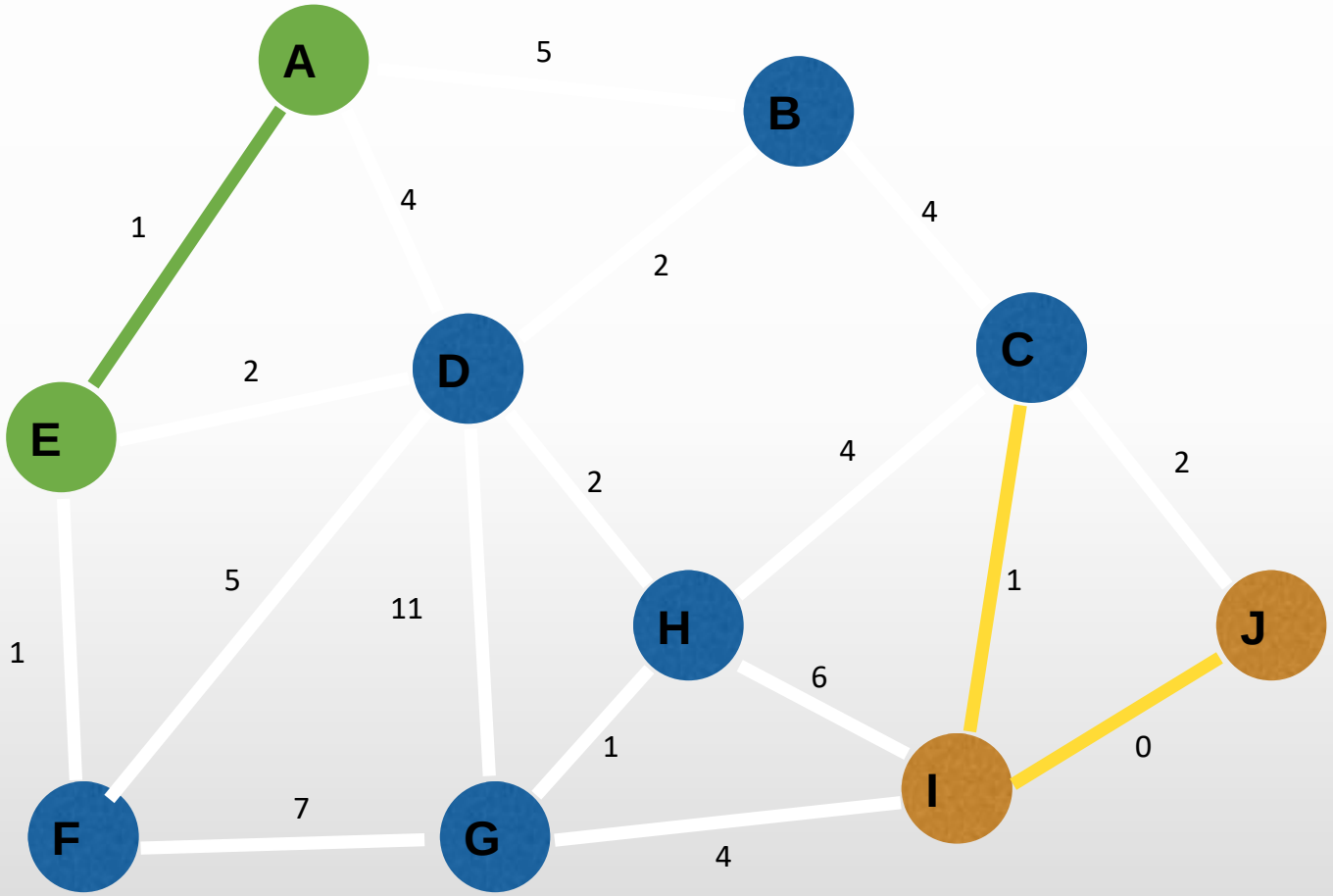
Kruskal Algoritması Uygulama

```
I -> J = 0
A -> E = 1
C -> I = 1
E -> F = 1
G -> H = 1
B -> D = 2
C -> J = 2
D -> E = 2
D -> H = 2
A -> D = 4
B -> C = 4
C -> H = 4
G -> I = 4
A -> B = 5
D -> F = 5
H -> I = 6
F -> G = 7
D -> G = 11
```



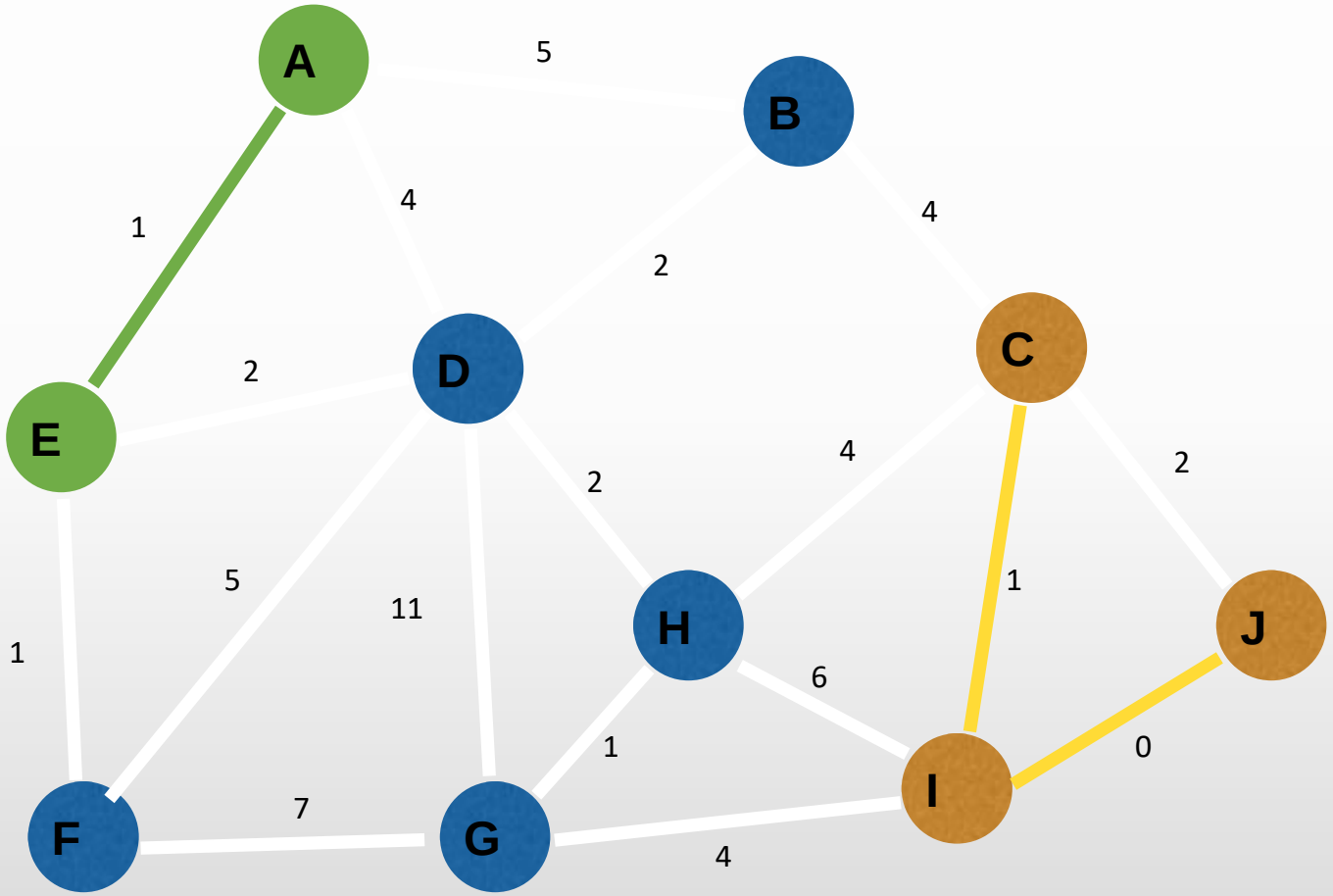
Kruskal Algoritması Uygulama

```
I -> J = 0
A -> E = 1
C -> I = 1
E -> F = 1
G -> H = 1
B -> D = 2
C -> J = 2
D -> E = 2
D -> H = 2
A -> D = 4
B -> C = 4
C -> H = 4
G -> I = 4
A -> B = 5
D -> F = 5
H -> I = 6
F -> G = 7
D -> G = 11
```



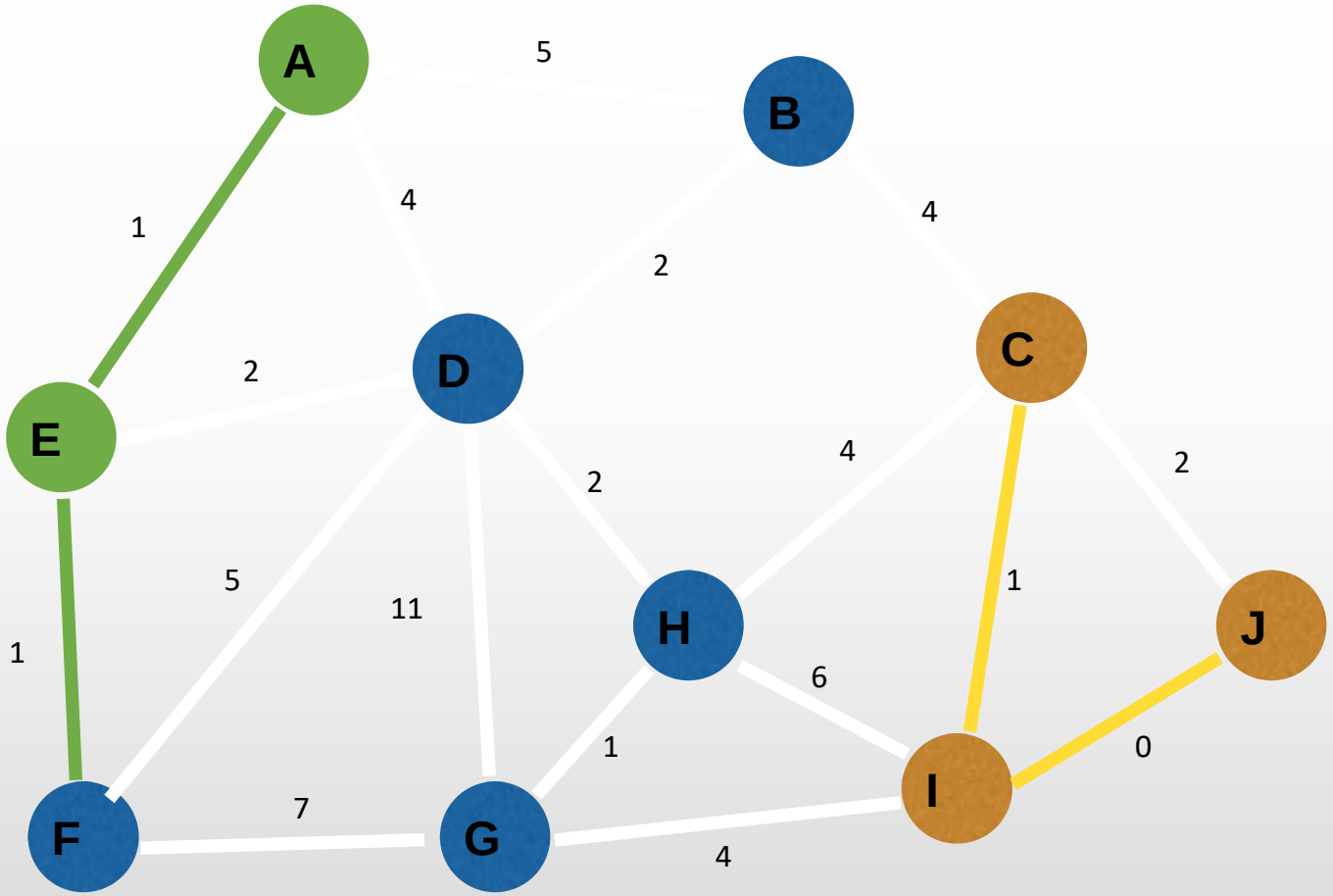
Kruskal Algoritması Uygulama

```
I -> J = 0
A -> E = 1
C -> I = 1
E -> F = 1
G -> H = 1
B -> D = 2
C -> J = 2
D -> E = 2
D -> H = 2
A -> D = 4
B -> C = 4
C -> H = 4
G -> I = 4
A -> B = 5
D -> F = 5
H -> I = 6
F -> G = 7
D -> G = 11
```



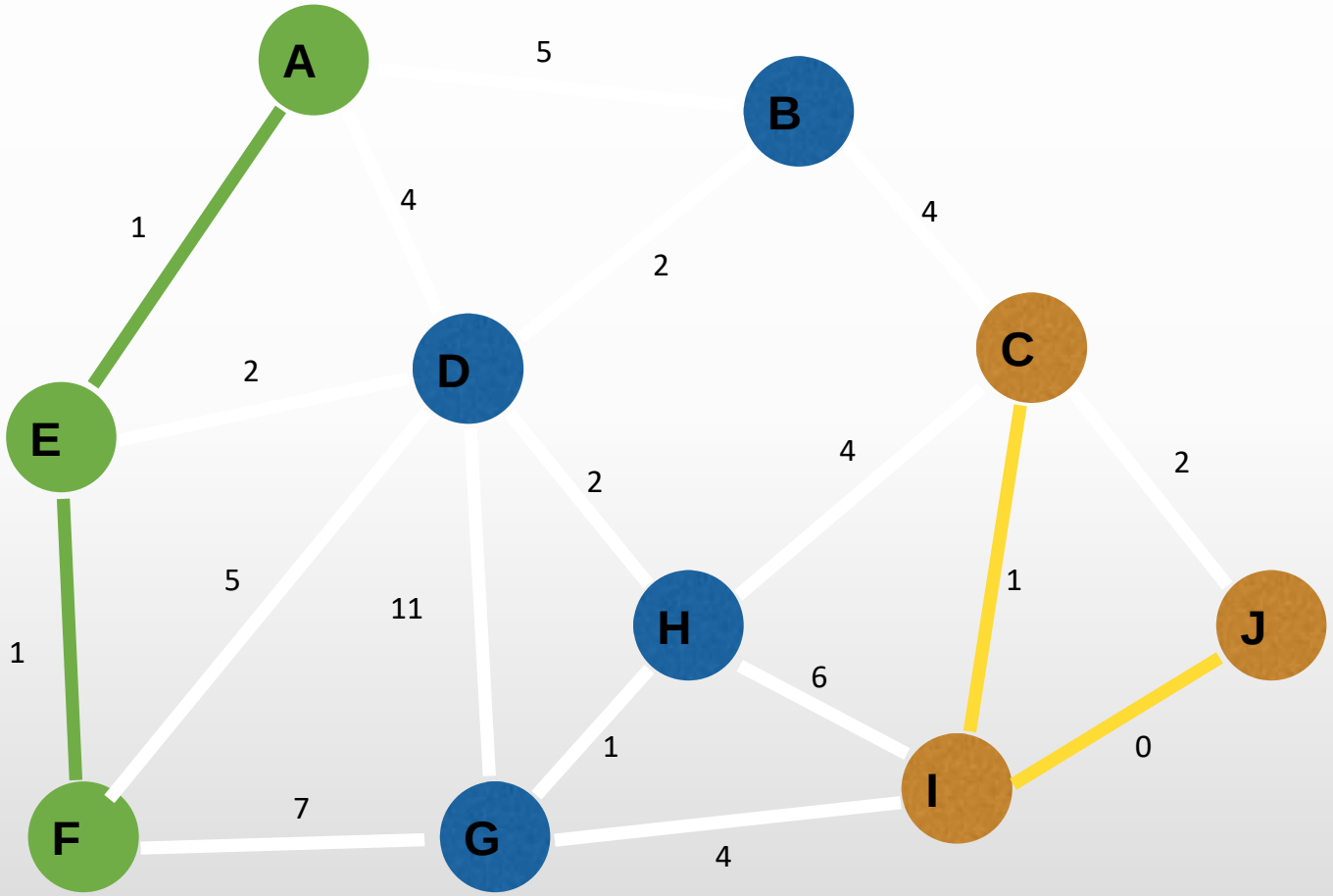
Kruskal Algoritması Uygulama

```
I -> J = 0
A -> E = 1
C -> I = 1
E -> F = 1
G -> H = 1
B -> D = 2
C -> J = 2
D -> E = 2
D -> H = 2
A -> D = 4
B -> C = 4
C -> H = 4
G -> I = 4
A -> B = 5
D -> F = 5
H -> I = 6
F -> G = 7
D -> G = 11
```



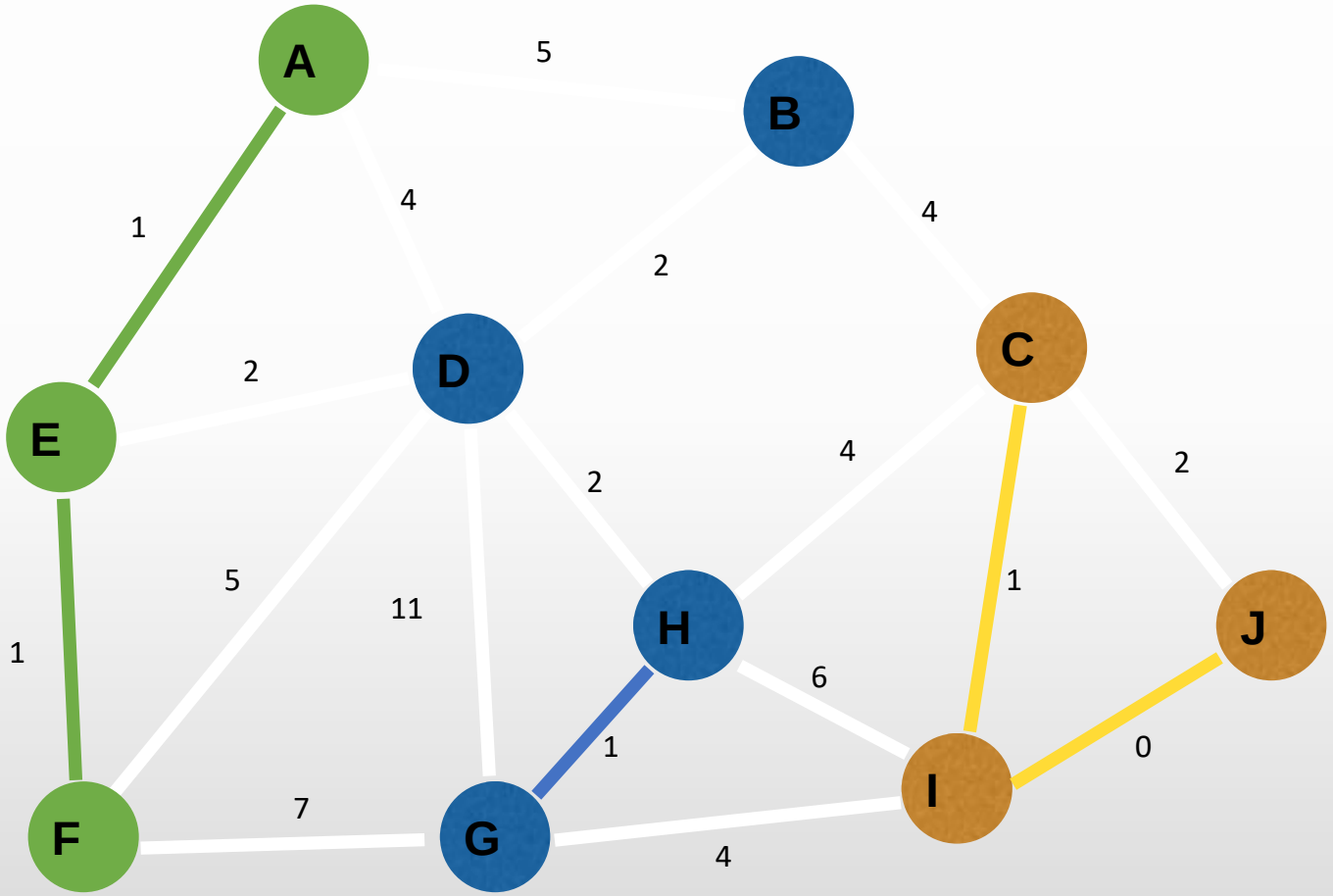
Kruskal Algoritması Uygulama

```
I -> J = 0
A -> E = 1
C -> I = 1
E -> F = 1
G -> H = 1
B -> D = 2
C -> J = 2
D -> E = 2
D -> H = 2
A -> D = 4
B -> C = 4
C -> H = 4
G -> I = 4
A -> B = 5
D -> F = 5
H -> I = 6
F -> G = 7
D -> G = 11
```



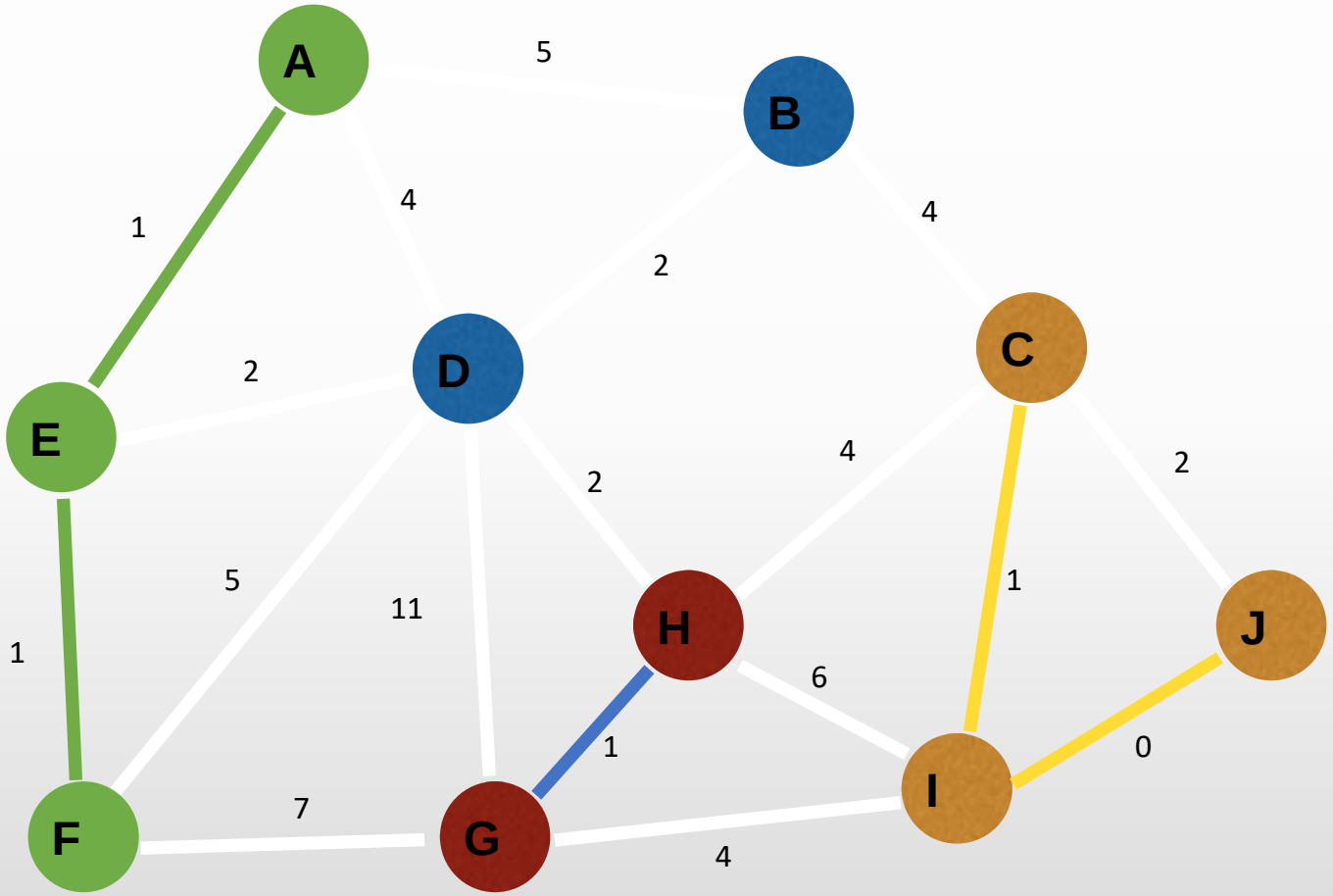
Kruskal Algoritması Uygulama

```
I -> J = 0
A -> E = 1
C -> I = 1
E -> F = 1
G -> H = 1
B -> D = 2
C -> J = 2
D -> E = 2
D -> H = 2
A -> D = 4
B -> C = 4
C -> H = 4
G -> I = 4
A -> B = 5
D -> F = 5
H -> I = 6
F -> G = 7
D -> G = 11
```



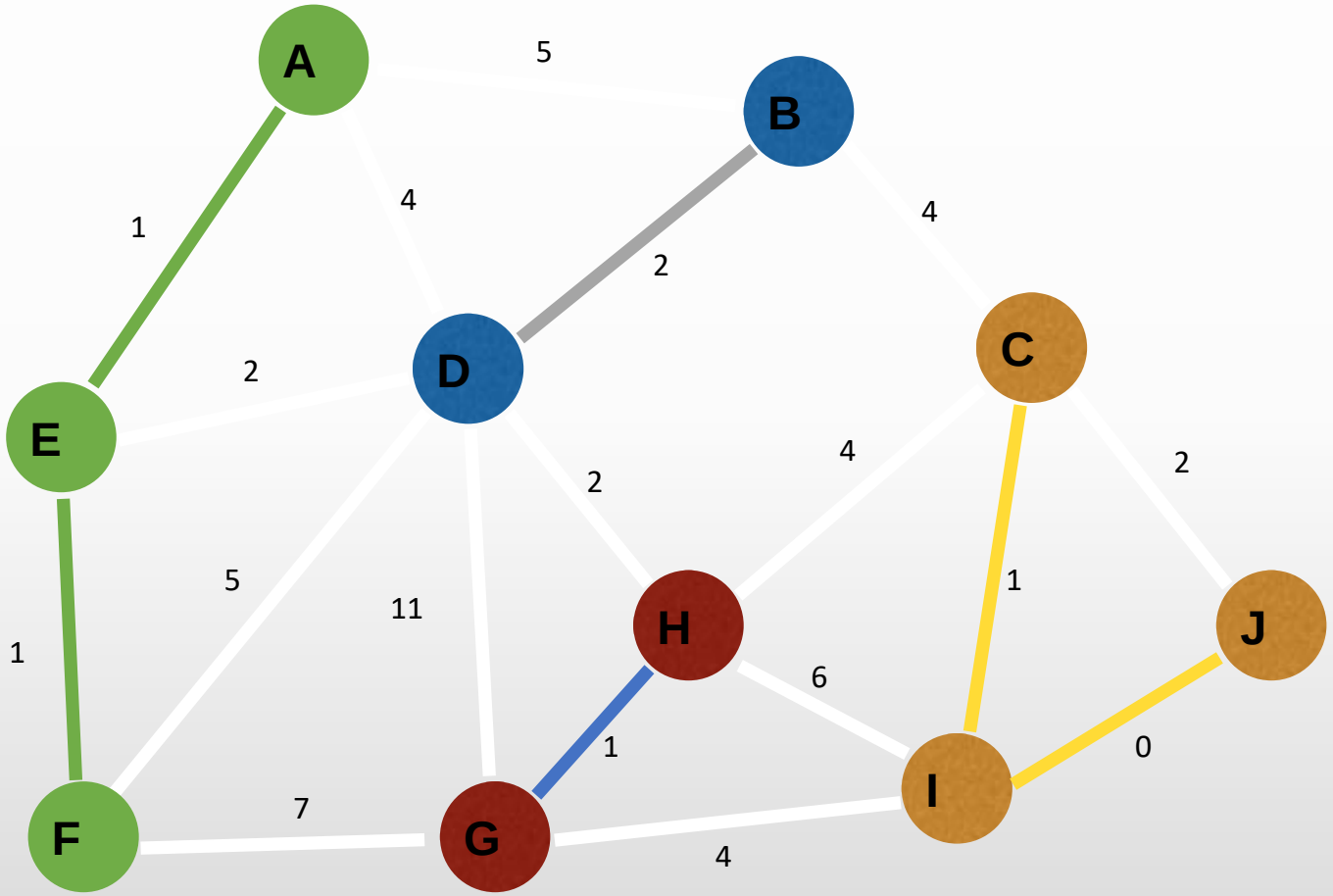
Kruskal Algoritması Uygulama

```
I -> J = 0
A -> E = 1
C -> I = 1
E -> F = 1
G -> H = 1
B -> D = 2
C -> J = 2
D -> E = 2
D -> H = 2
A -> D = 4
B -> C = 4
C -> H = 4
G -> I = 4
A -> B = 5
D -> F = 5
H -> I = 6
F -> G = 7
D -> G = 11
```



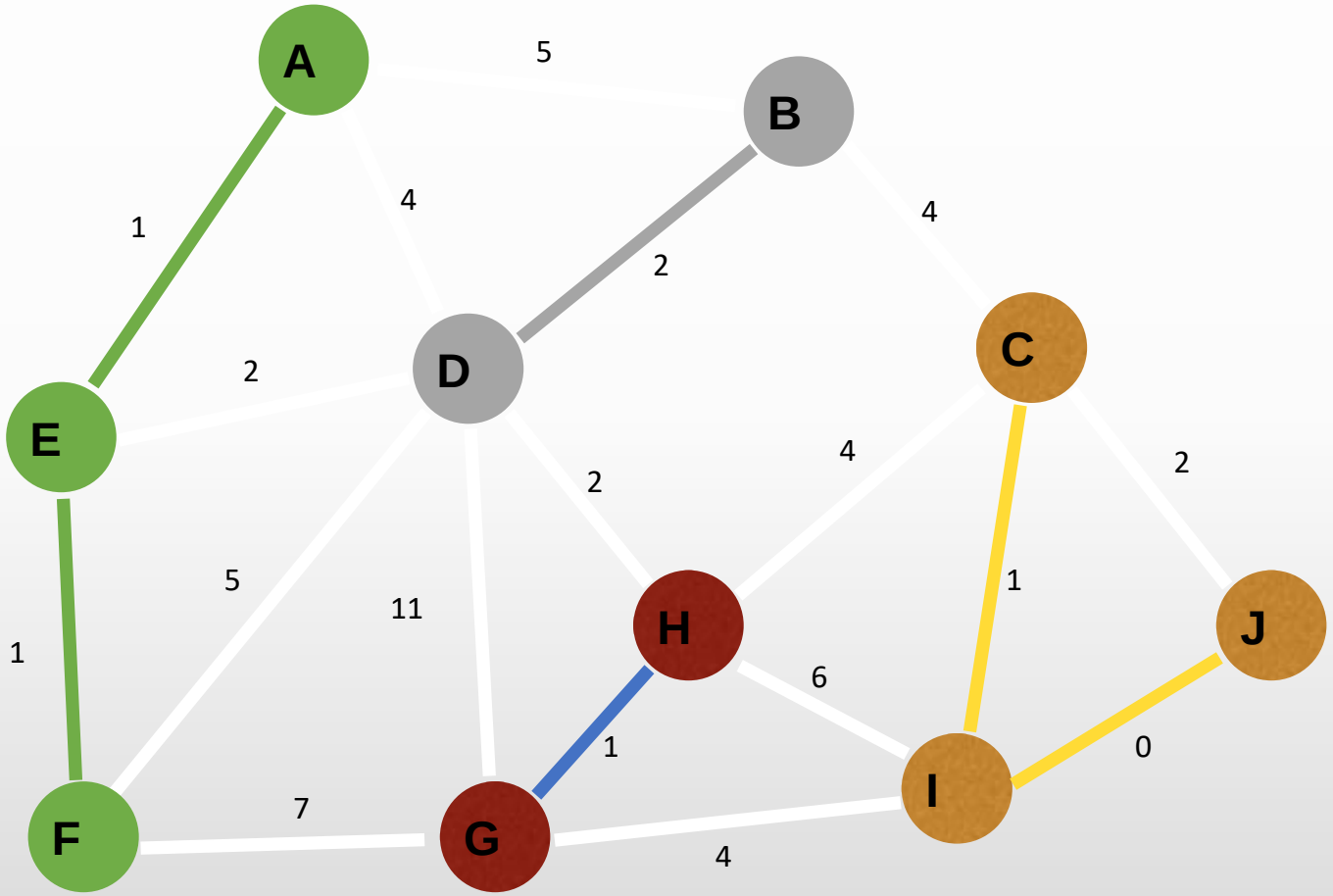
Kruskal Algoritması Uygulama

```
I -> J = 0
A -> E = 1
C -> I = 1
E -> F = 1
G -> H = 1
B -> D = 2
C -> J = 2
D -> E = 2
D -> H = 2
A -> D = 4
B -> C = 4
C -> H = 4
G -> I = 4
A -> B = 5
D -> F = 5
H -> I = 6
F -> G = 7
D -> G = 11
```



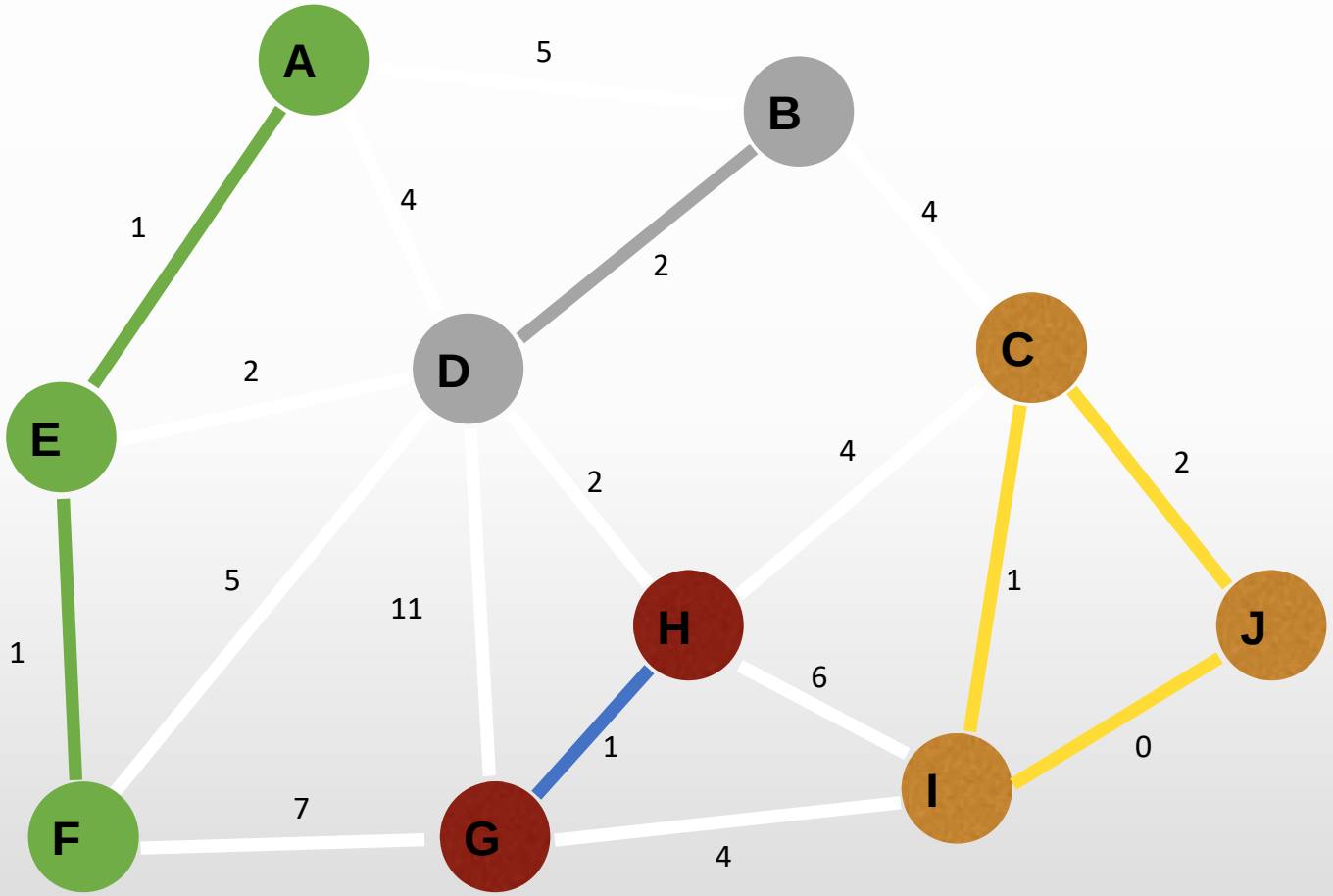
Kruskal Algoritması Uygulama

```
I -> J = 0
A -> E = 1
C -> I = 1
E -> F = 1
G -> H = 1
B -> D = 2
C -> J = 2
D -> E = 2
D -> H = 2
A -> D = 4
B -> C = 4
C -> H = 4
G -> I = 4
A -> B = 5
D -> F = 5
H -> I = 6
F -> G = 7
D -> G = 11
```



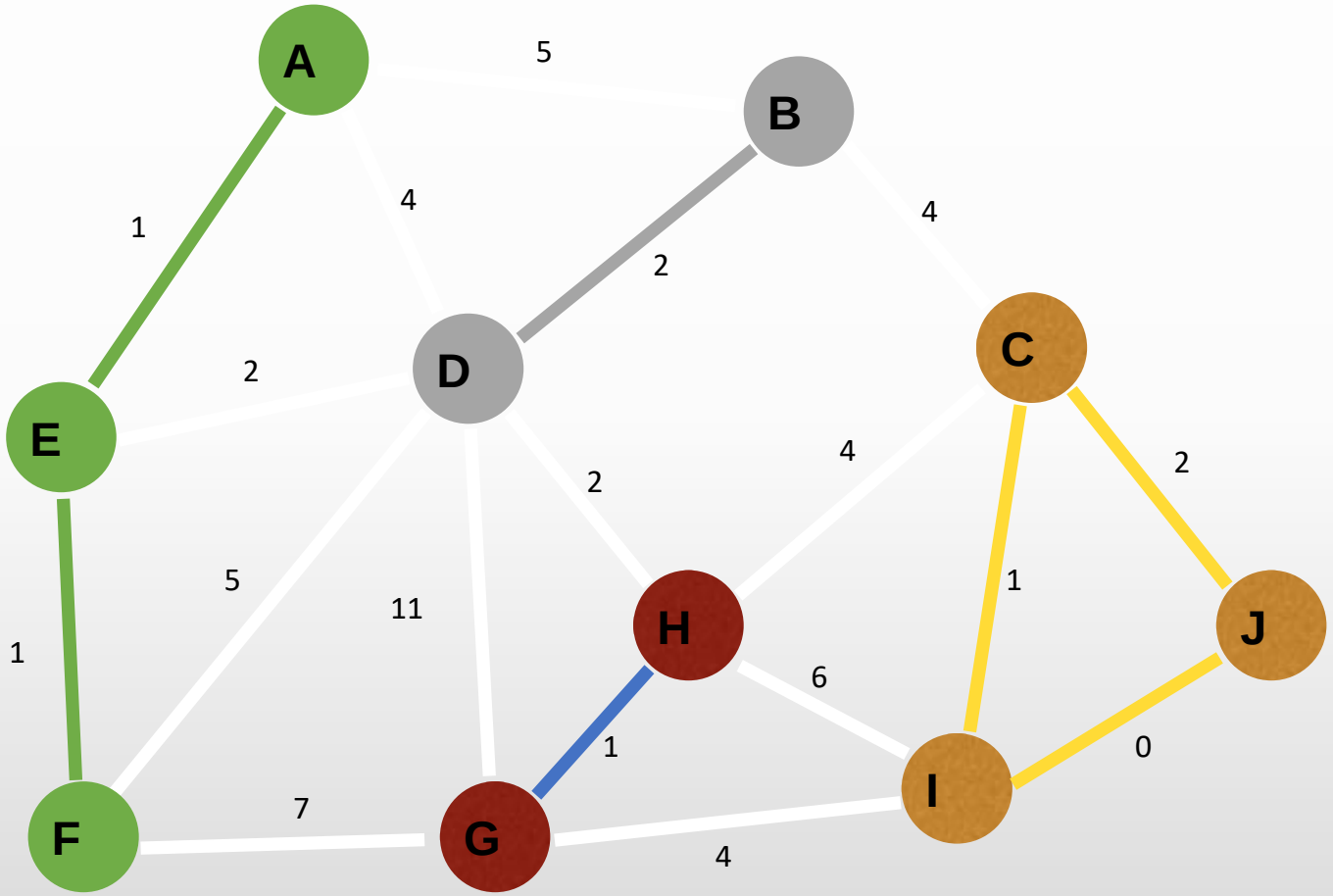
Kruskal Algoritması Uygulama

```
I -> J = 0
A -> E = 1
C -> I = 1
E -> F = 1
G -> H = 1
B -> D = 2
C -> J = 2
D -> E = 2
D -> H = 2
A -> D = 4
B -> C = 4
C -> H = 4
G -> I = 4
A -> B = 5
D -> F = 5
H -> I = 6
F -> G = 7
D -> G = 11
```



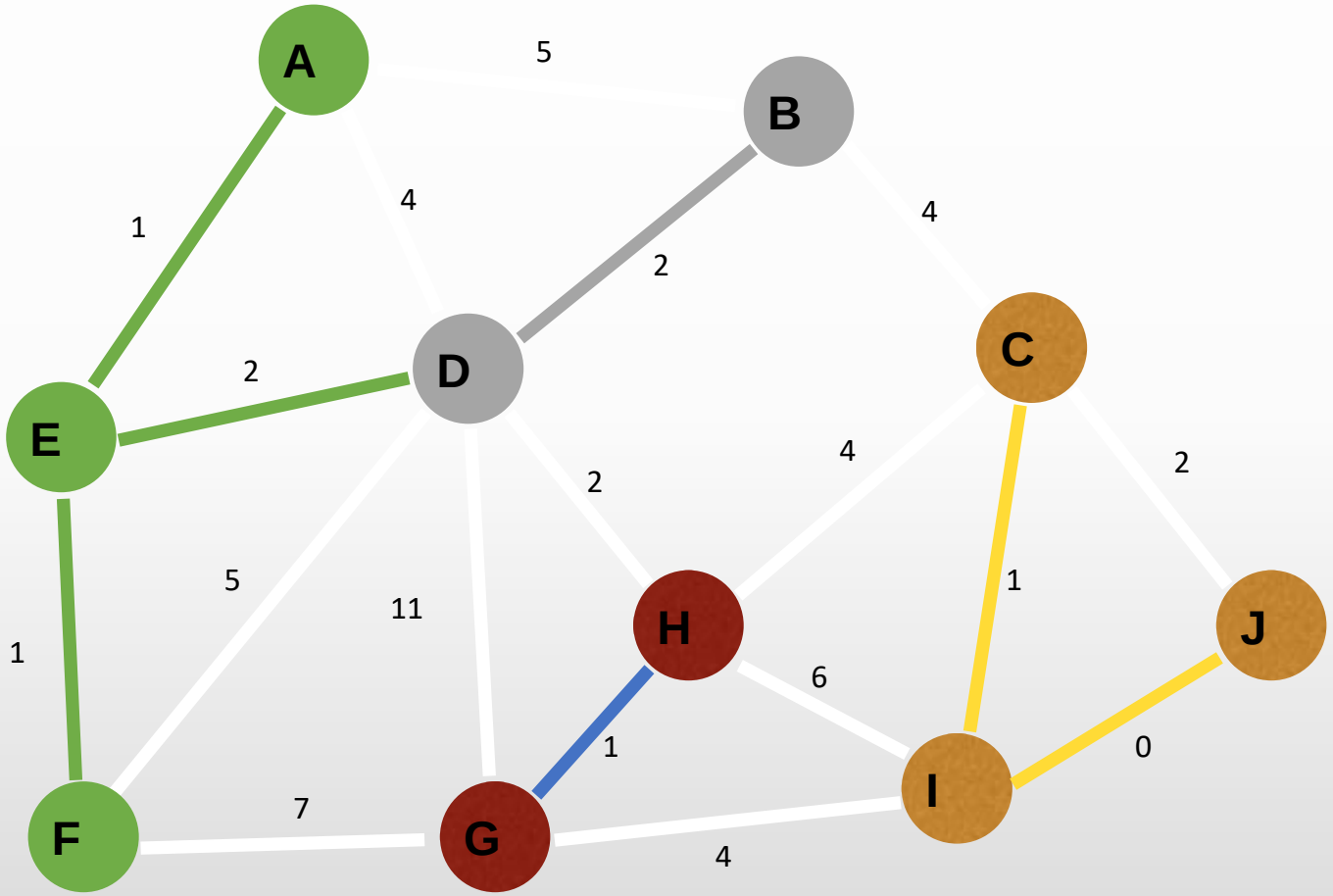
Kruskal Algoritması Uygulama

```
I -> J = 0
A -> E = 1
C -> I = 1
E -> F = 1
G -> H = 1
B -> D = 2
C -> J = 2
D -> E = 2
D -> H = 2
A -> D = 4
B -> C = 4
C -> H = 4
G -> I = 4
A -> B = 5
D -> F = 5
H -> I = 6
F -> G = 7
D -> G = 11
```



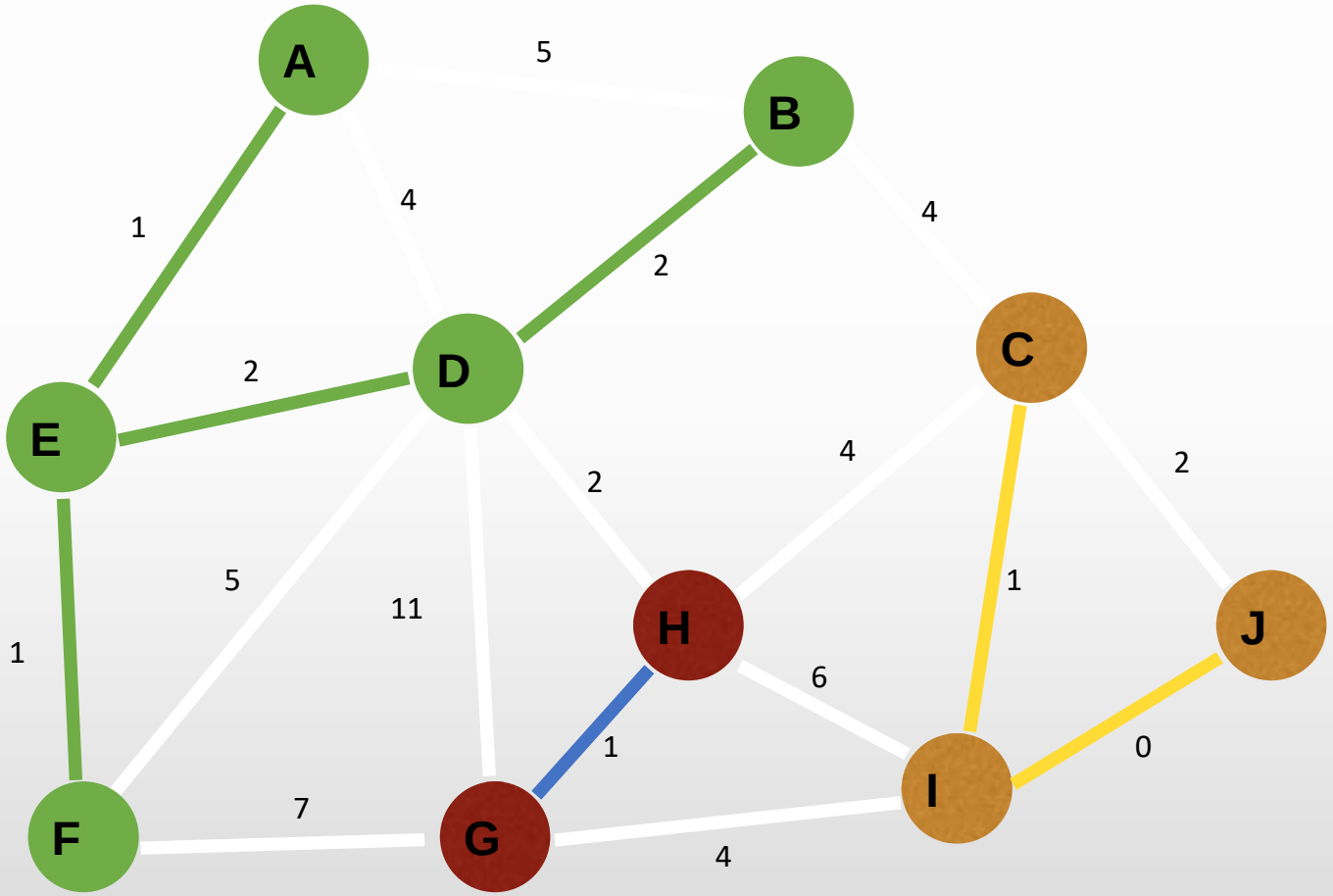
Kruskal Algoritması Uygulama

```
I -> J = 0
A -> E = 1
C -> I = 1
E -> F = 1
G -> H = 1
B -> D = 2
C -> J = 2
D -> E = 2
D -> H = 2
A -> D = 4
B -> C = 4
C -> H = 4
G -> I = 4
A -> B = 5
D -> F = 5
H -> I = 6
F -> G = 7
D -> G = 11
```



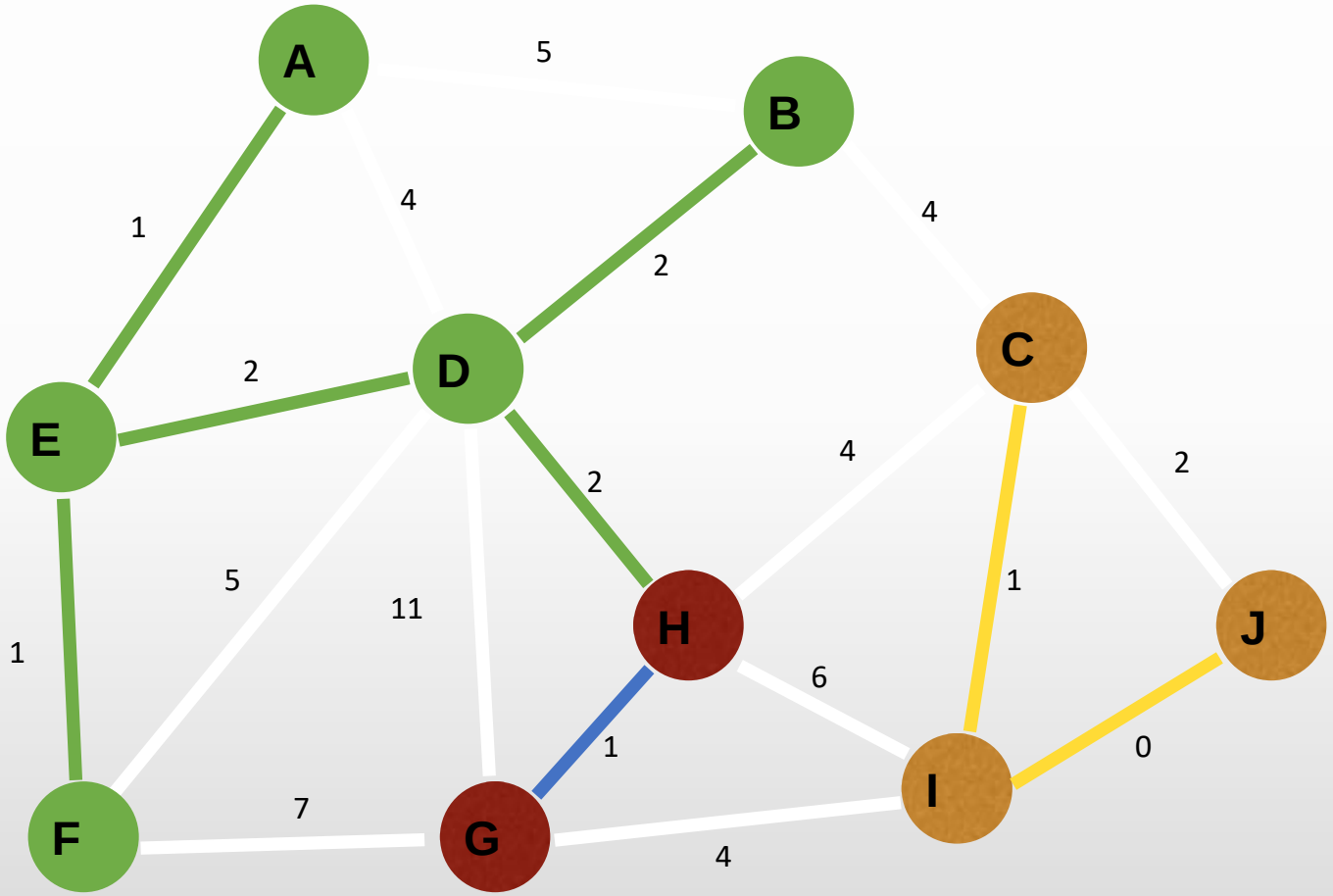
Kruskal Algoritması Uygulama

```
I -> J = 0
A -> E = 1
C -> I = 1
E -> F = 1
G -> H = 1
B -> D = 2
C -> J = 2
D -> E = 2
D -> H = 2
A -> D = 4
B -> C = 4
C -> H = 4
G -> I = 4
A -> B = 5
D -> F = 5
H -> I = 6
F -> G = 7
D -> G = 11
```



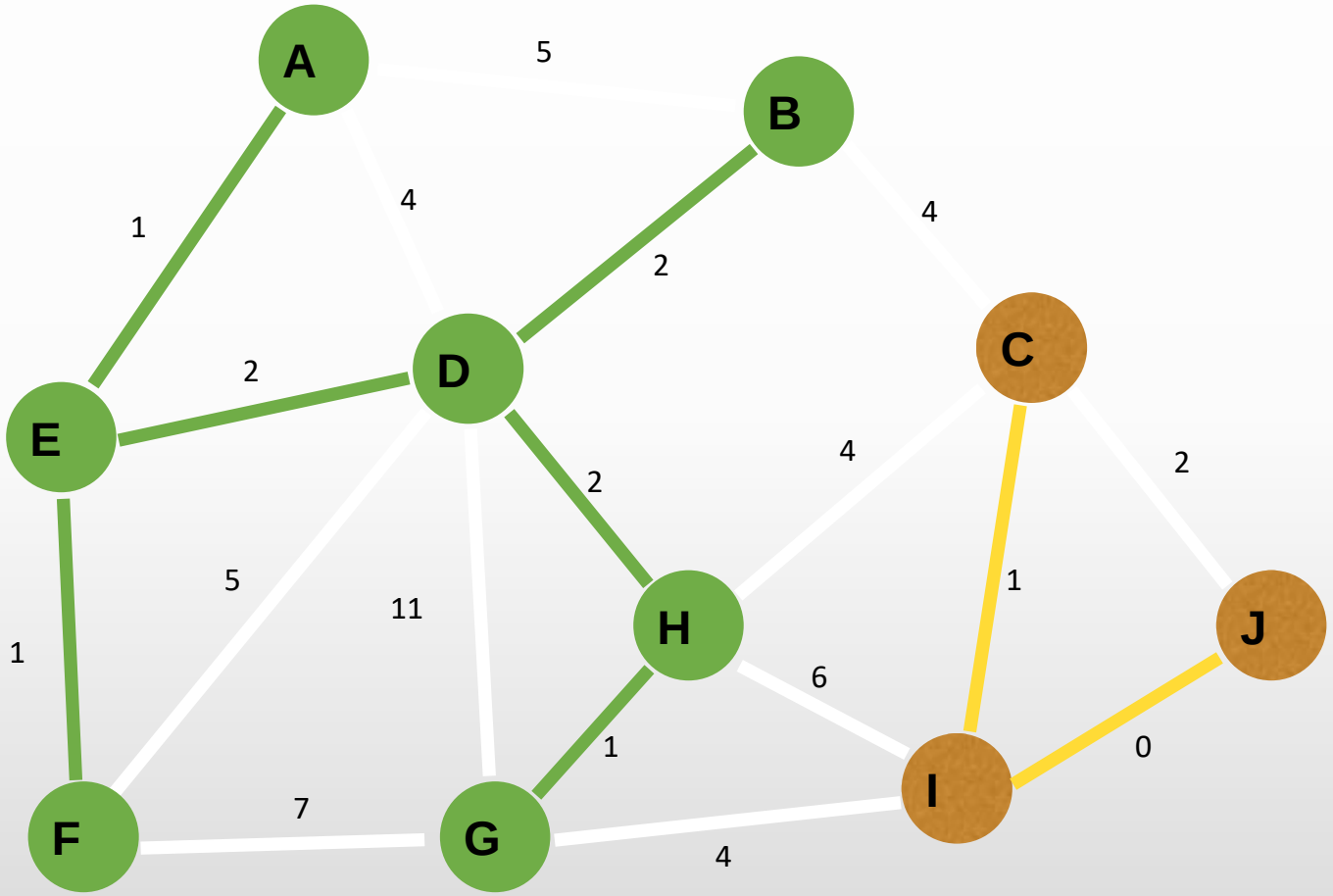
Kruskal Algoritması Uygulama

```
I -> J = 0
A -> E = 1
C -> I = 1
E -> F = 1
G -> H = 1
B -> D = 2
C -> J = 2
D -> E = 2
D -> H = 2
A -> D = 4
B -> C = 4
C -> H = 4
G -> I = 4
A -> B = 5
D -> F = 5
H -> I = 6
F -> G = 7
D -> G = 11
```



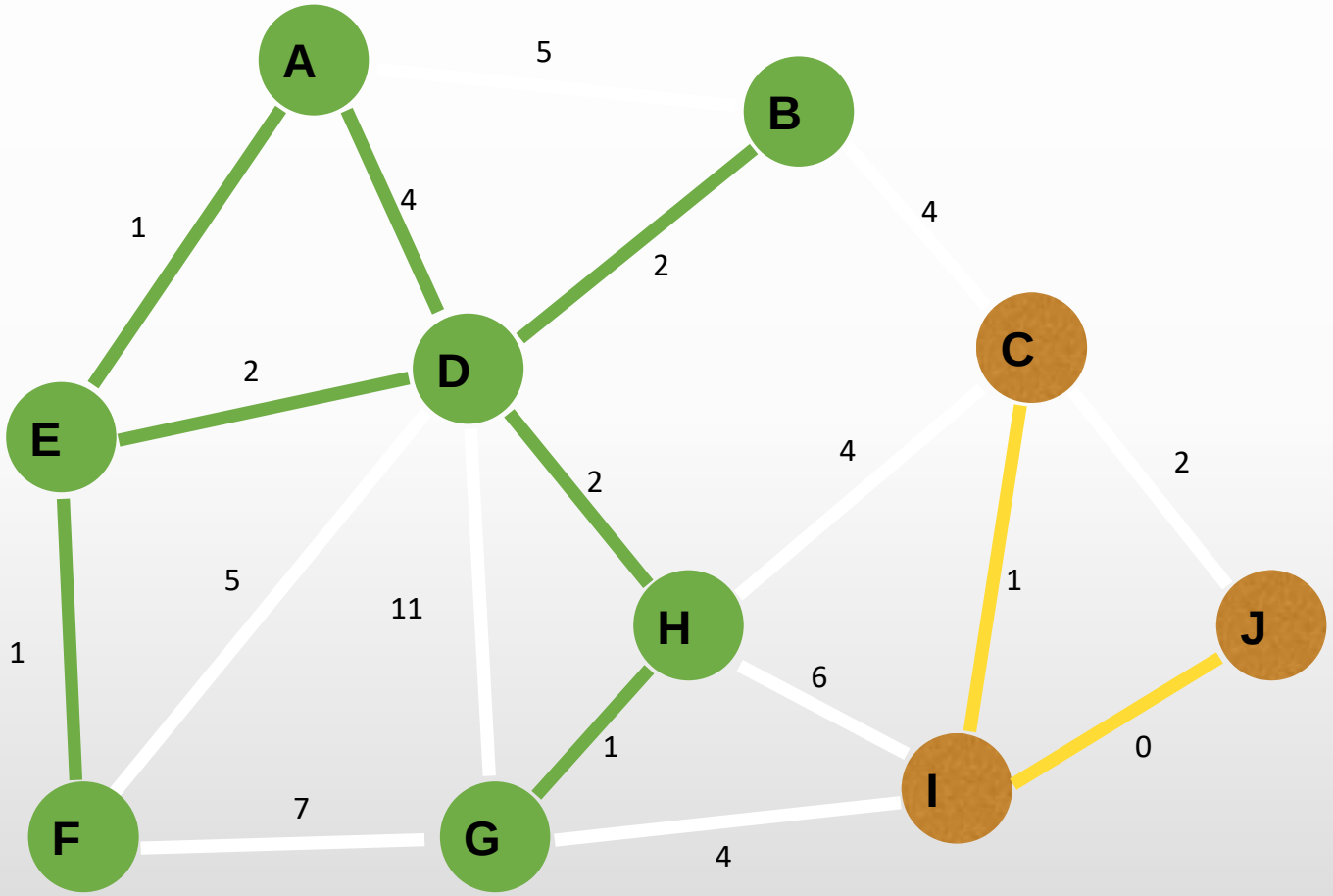
Kruskal Algoritması Uygulama

```
I -> J = 0
A -> E = 1
C -> I = 1
E -> F = 1
G -> H = 1
B -> D = 2
C -> J = 2
D -> E = 2
D -> H = 2
A -> D = 4
B -> C = 4
C -> H = 4
G -> I = 4
A -> B = 5
D -> F = 5
H -> I = 6
F -> G = 7
D -> G = 11
```



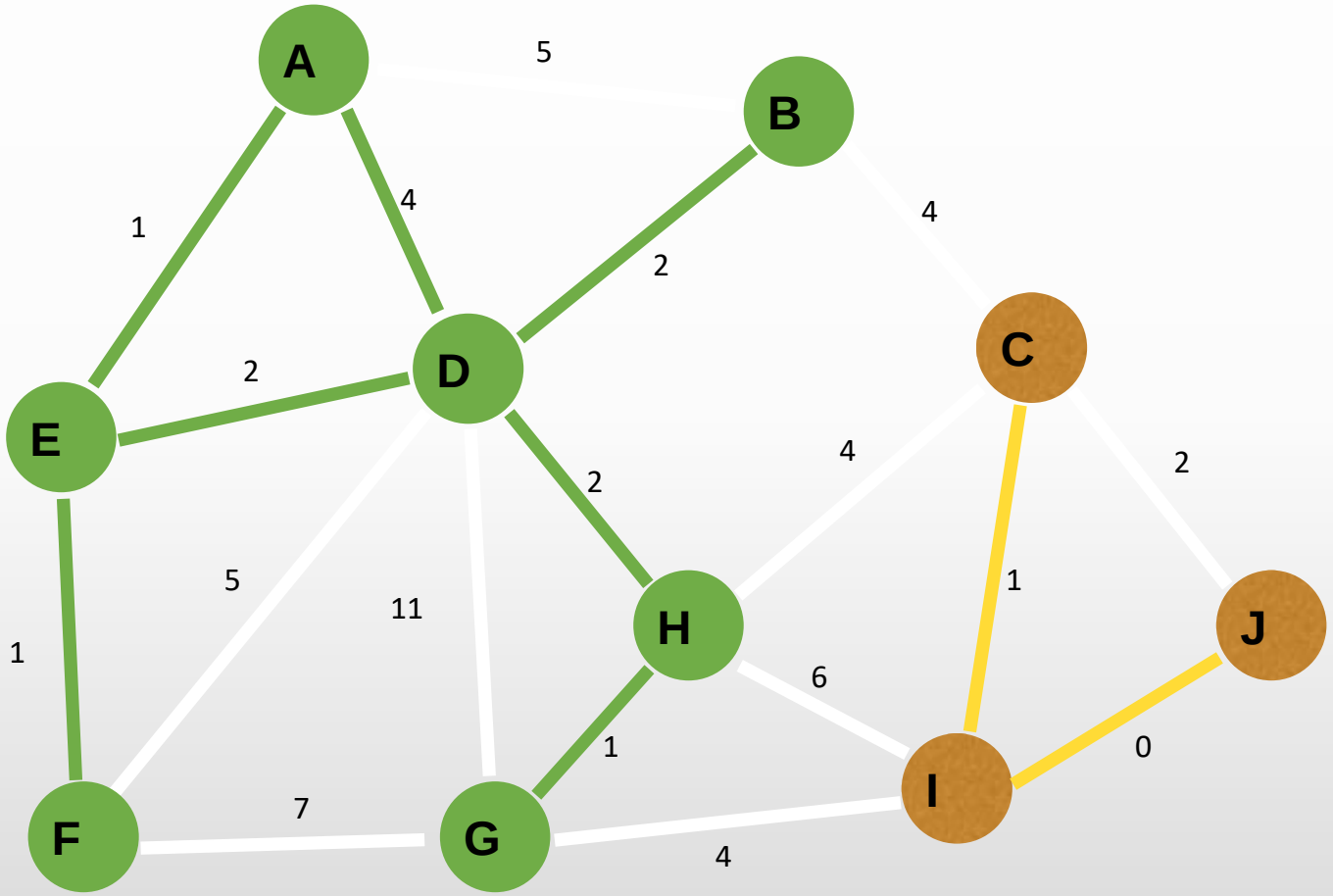
Kruskal Algoritması Uygulama

```
I -> J = 0
A -> E = 1
C -> I = 1
E -> F = 1
G -> H = 1
B -> D = 2
C -> J = 2
D -> E = 2
D -> H = 2
A -> D = 4
B -> C = 4
C -> H = 4
G -> I = 4
A -> B = 5
D -> F = 5
H -> I = 6
F -> G = 7
D -> G = 11
```



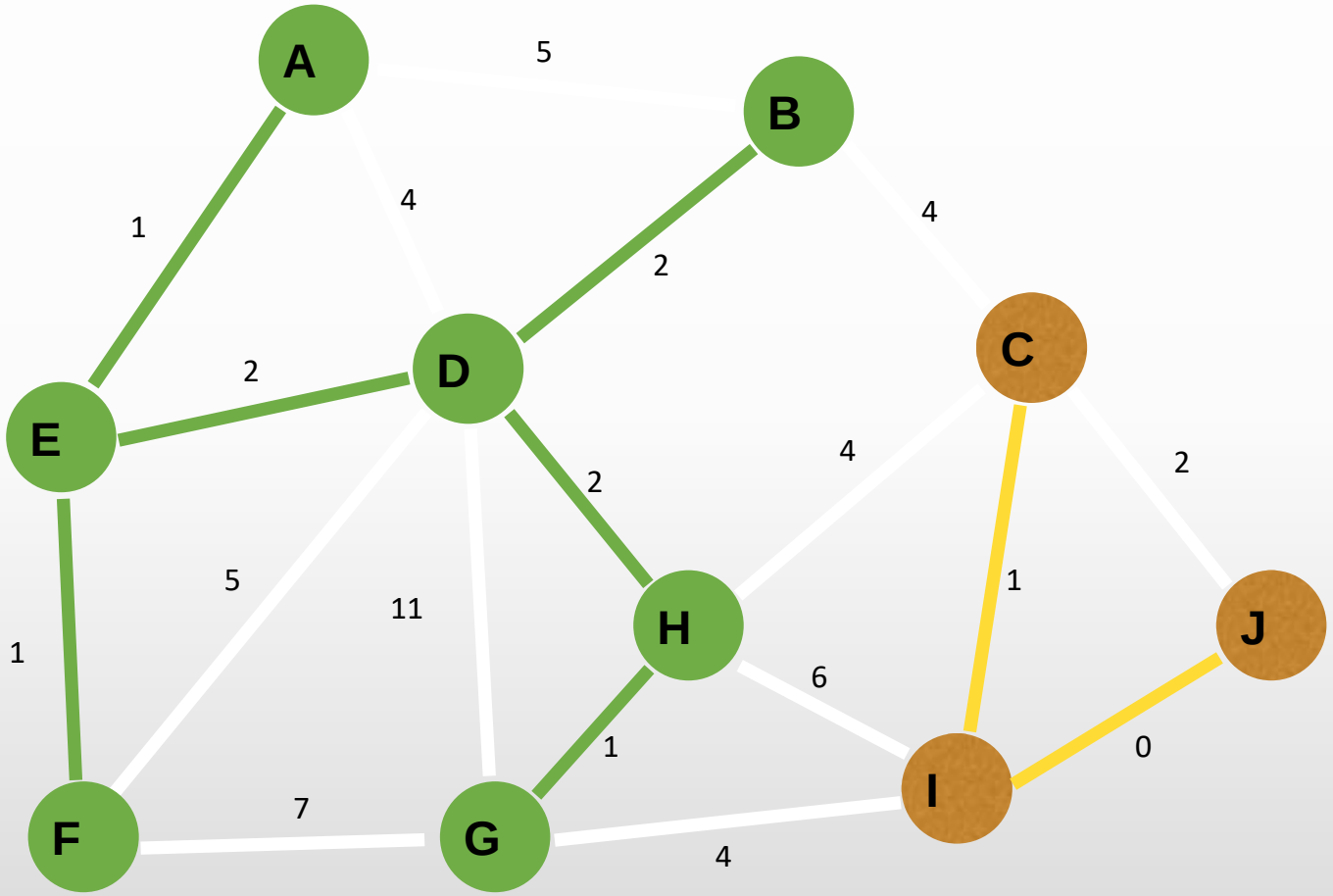
Kruskal Algoritması Uygulama

```
I -> J = 0
A -> E = 1
C -> I = 1
E -> F = 1
G -> H = 1
B -> D = 2
C -> J = 2
D -> E = 2
D -> H = 2
A -> D = 4
B -> C = 4
C -> H = 4
G -> I = 4
A -> B = 5
D -> F = 5
H -> I = 6
F -> G = 7
D -> G = 11
```



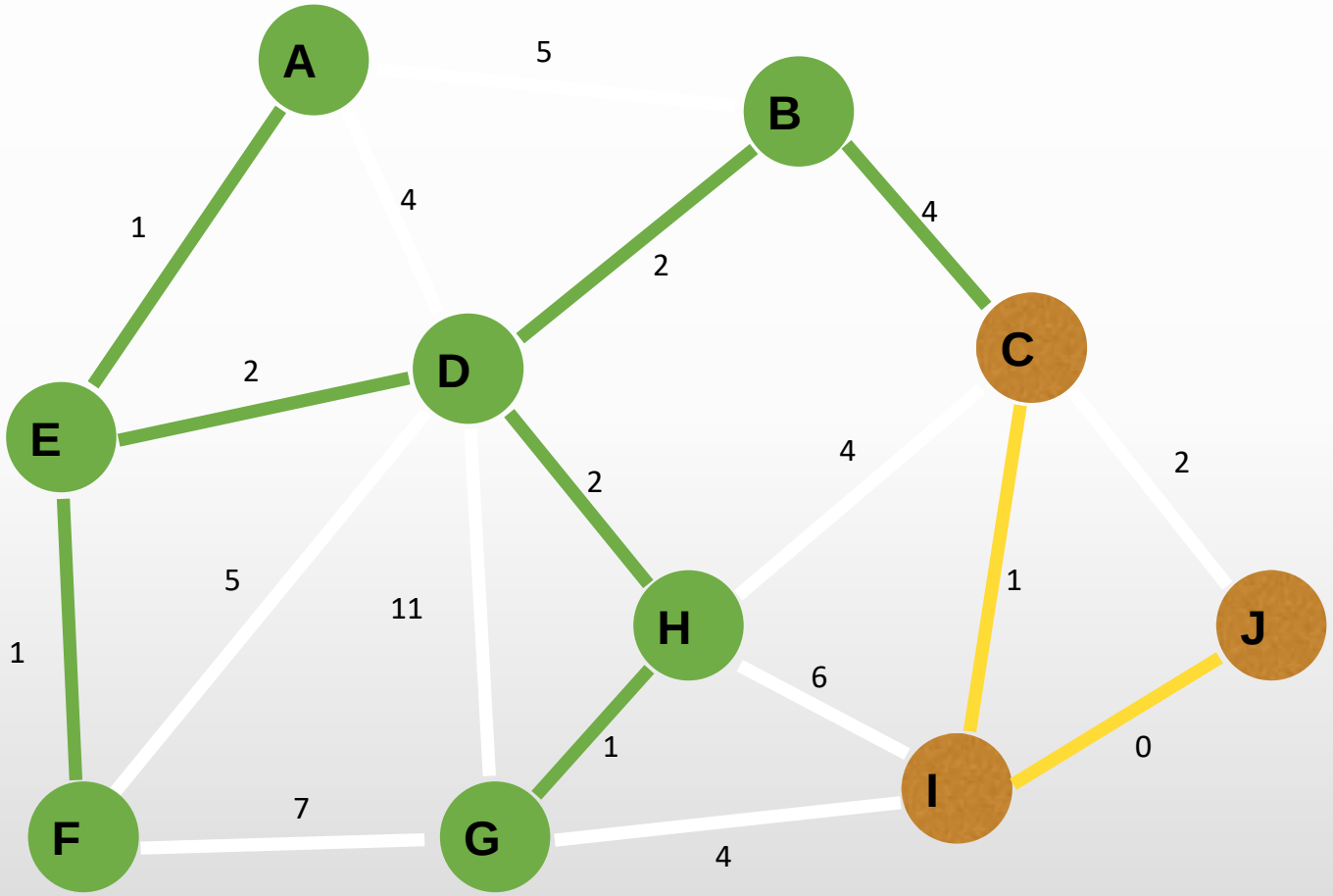
Kruskal Algoritması Uygulama

```
I -> J = 0
A -> E = 1
C -> I = 1
E -> F = 1
G -> H = 1
B -> D = 2
C -> J = 2
D -> E = 2
D -> H = 2
A -> D = 4
B -> C = 4
C -> H = 4
G -> I = 4
A -> B = 5
D -> F = 5
H -> I = 6
F -> G = 7
D -> G = 11
```



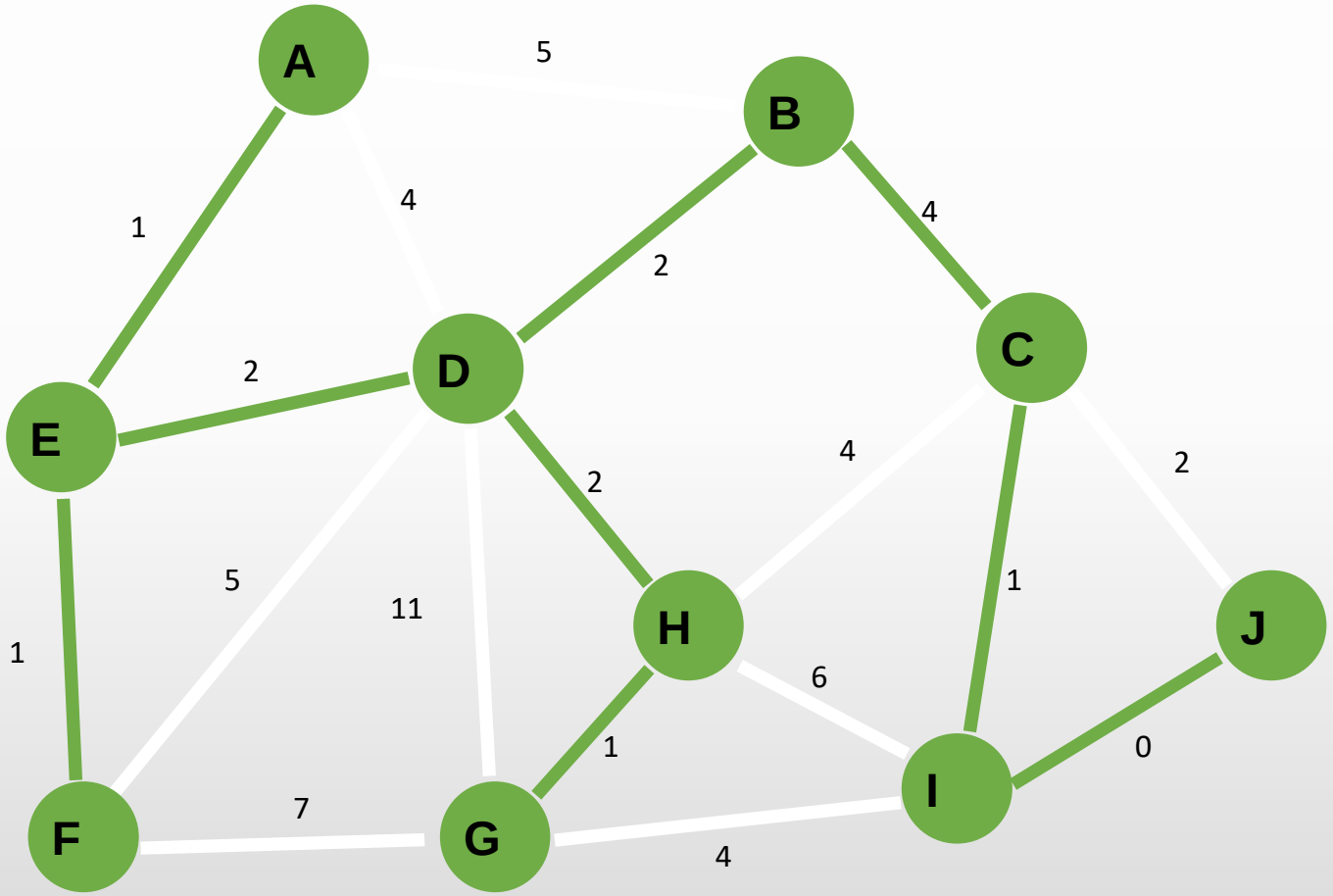
Kruskal Algoritması Uygulama

```
I -> J = 0
A -> E = 1
C -> I = 1
E -> F = 1
G -> H = 1
B -> D = 2
C -> J = 2
D -> E = 2
D -> H = 2
A -> D = 4
B -> C = 4
C -> H = 4
G -> I = 4
A -> B = 5
D -> F = 5
H -> I = 6
F -> G = 7
D -> G = 11
```



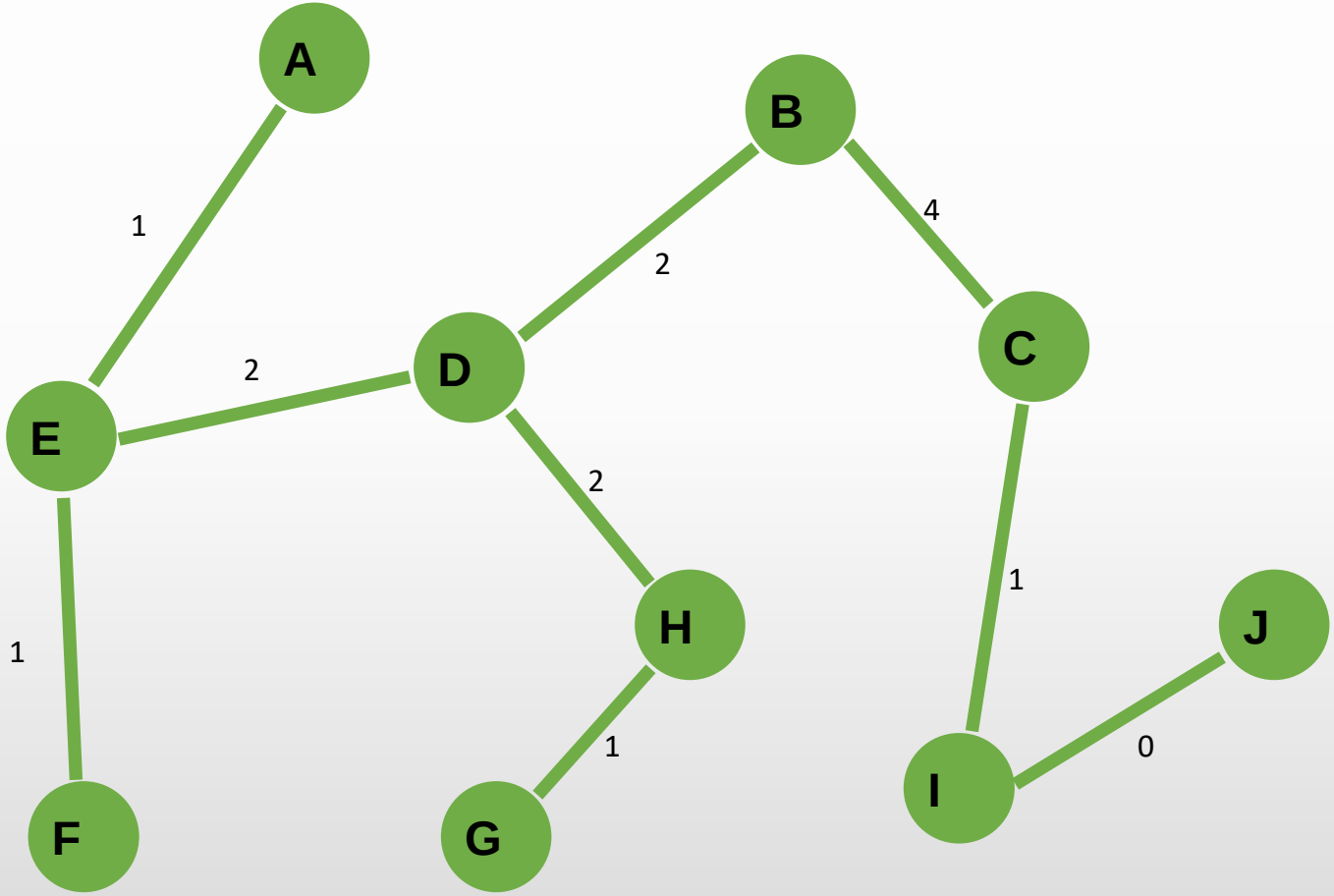
Kruskal Algoritması Uygulama

```
I -> J = 0
A -> E = 1
C -> I = 1
E -> F = 1
G -> H = 1
B -> D = 2
C -> J = 2
D -> E = 2
D -> H = 2
A -> D = 4
B -> C = 4
C -> H = 4
G -> I = 4
A -> B = 5
D -> F = 5
H -> I = 6
F -> G = 7
D -> G = 11
```

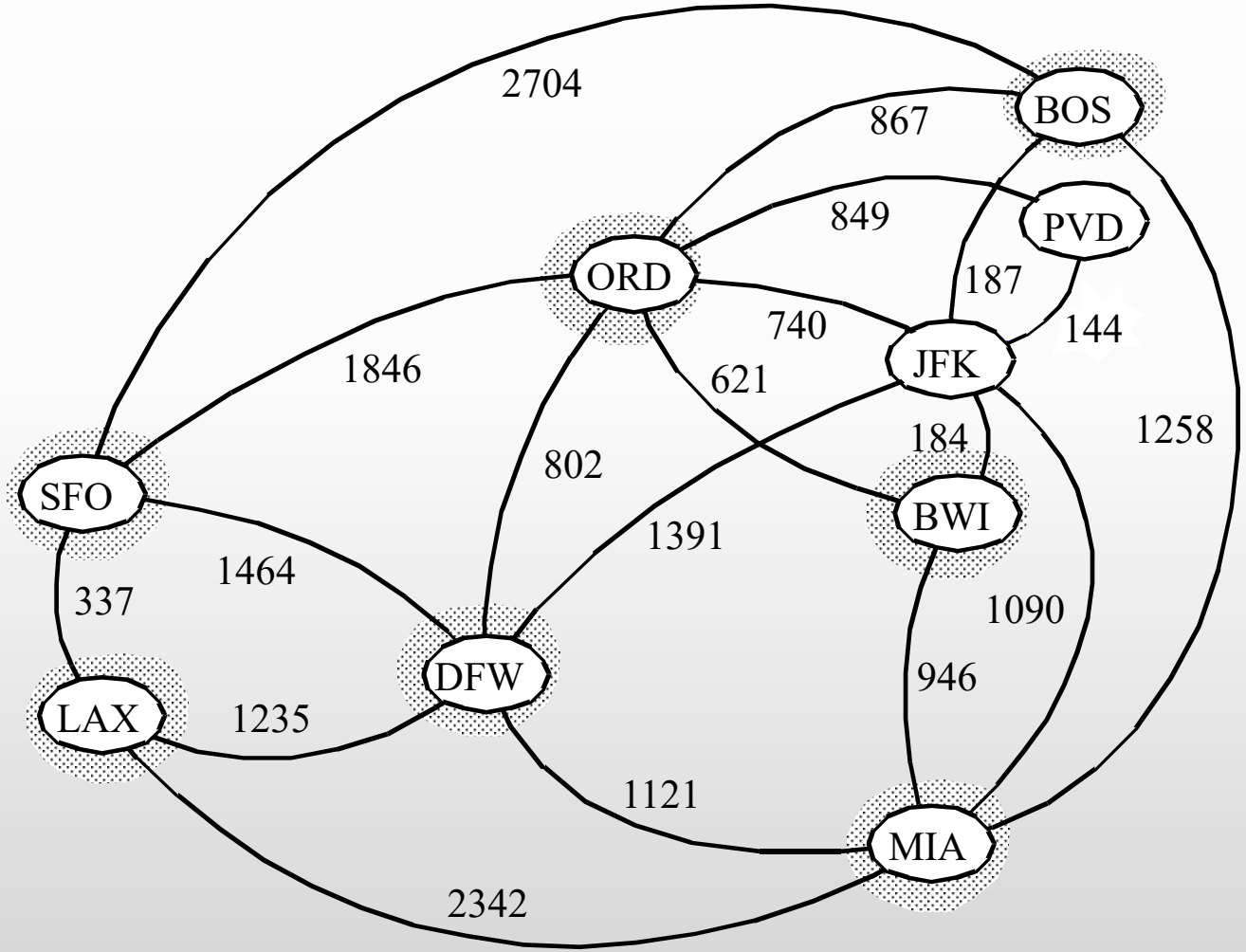


Kruskal Algoritması Uygulama

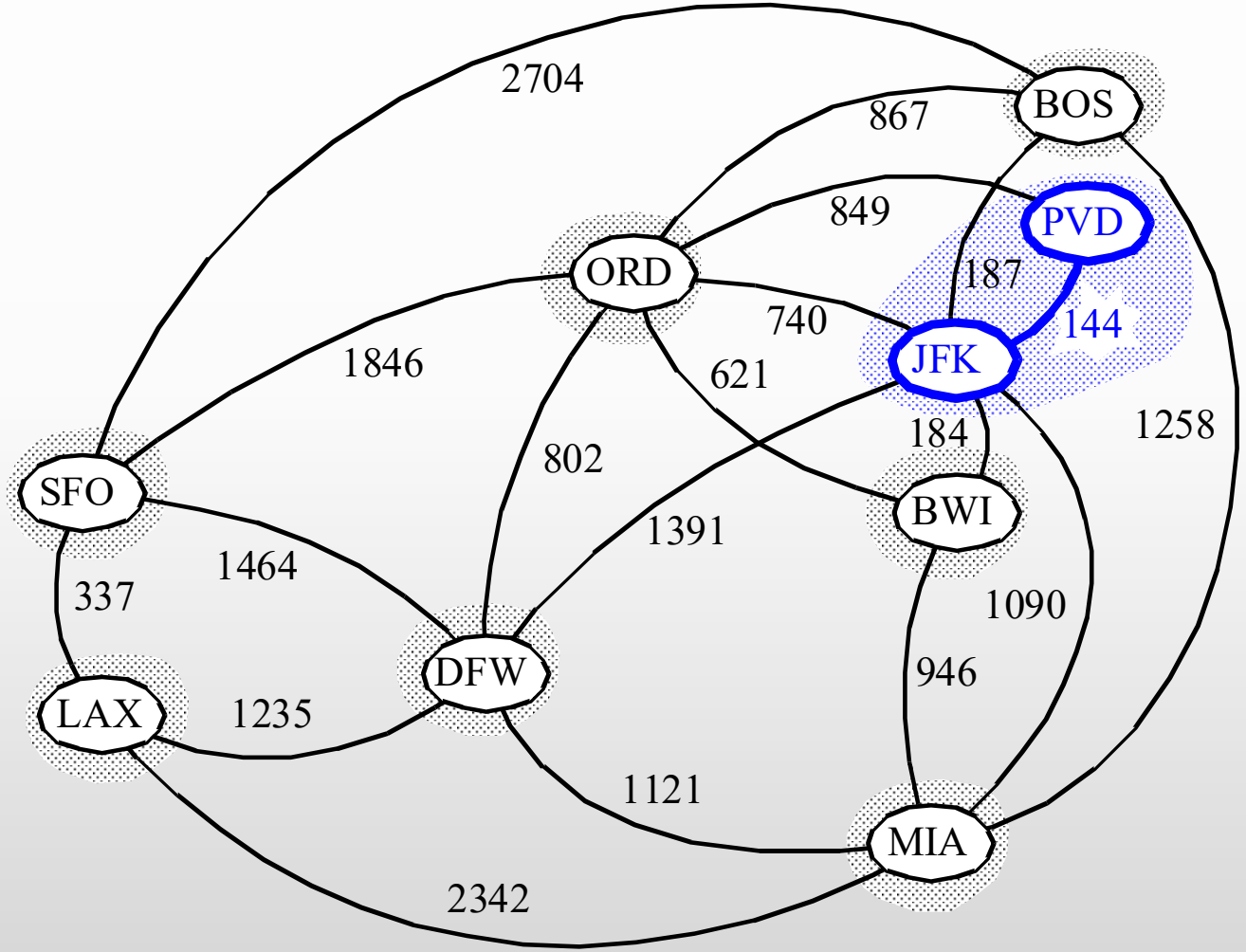
```
I -> J = 0
A -> E = 1
C -> I = 1
E -> F = 1
G -> H = 1
B -> D = 2
C -> J = 2
D -> E = 2
D -> H = 2
A -> D = 4
B -> C = 4
C -> H = 4
G -> I = 4
A -> B = 5
D -> F = 5
H -> I = 6
F -> G = 7
D -> G = 11
```



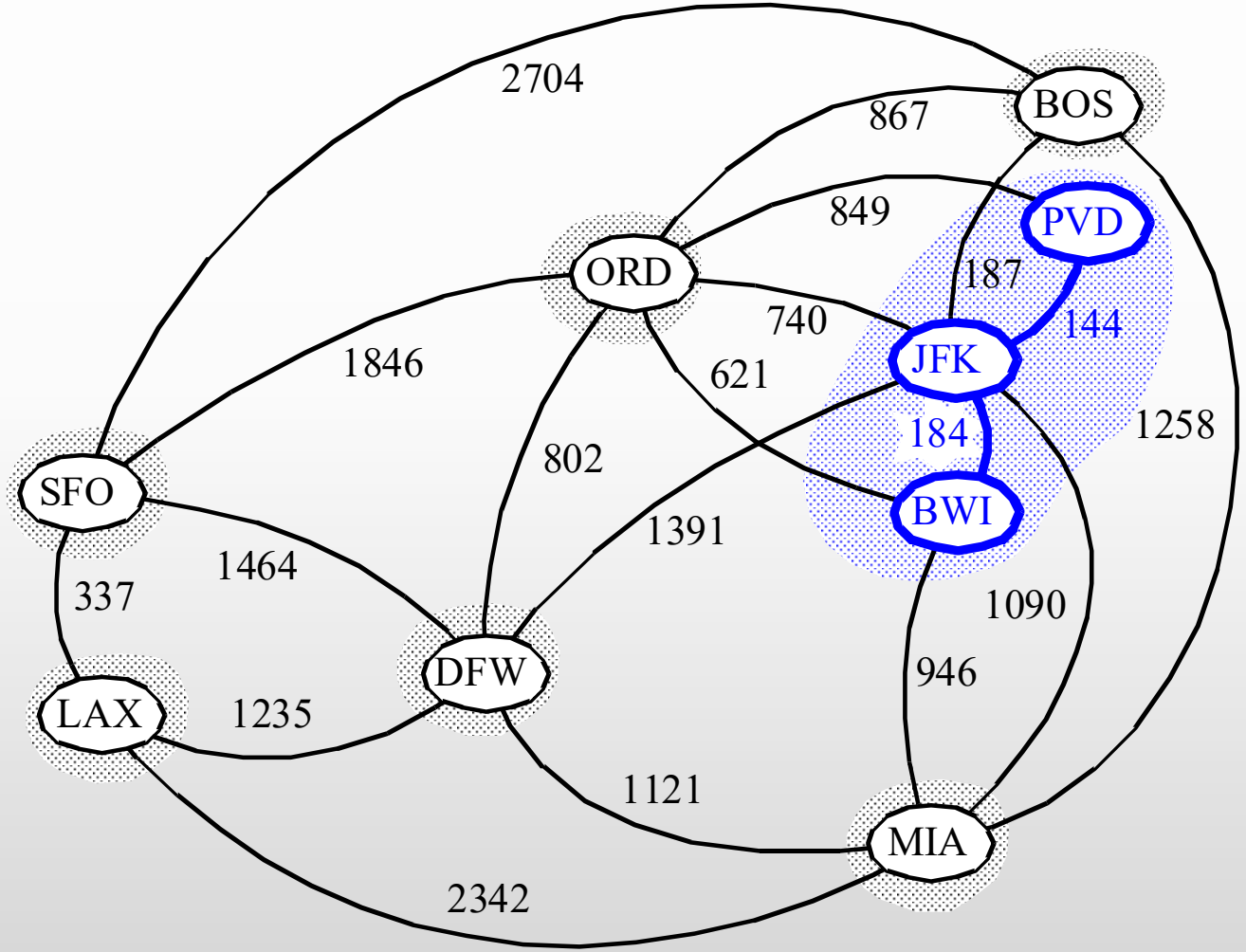
Kruskal Algoritması Uygulama



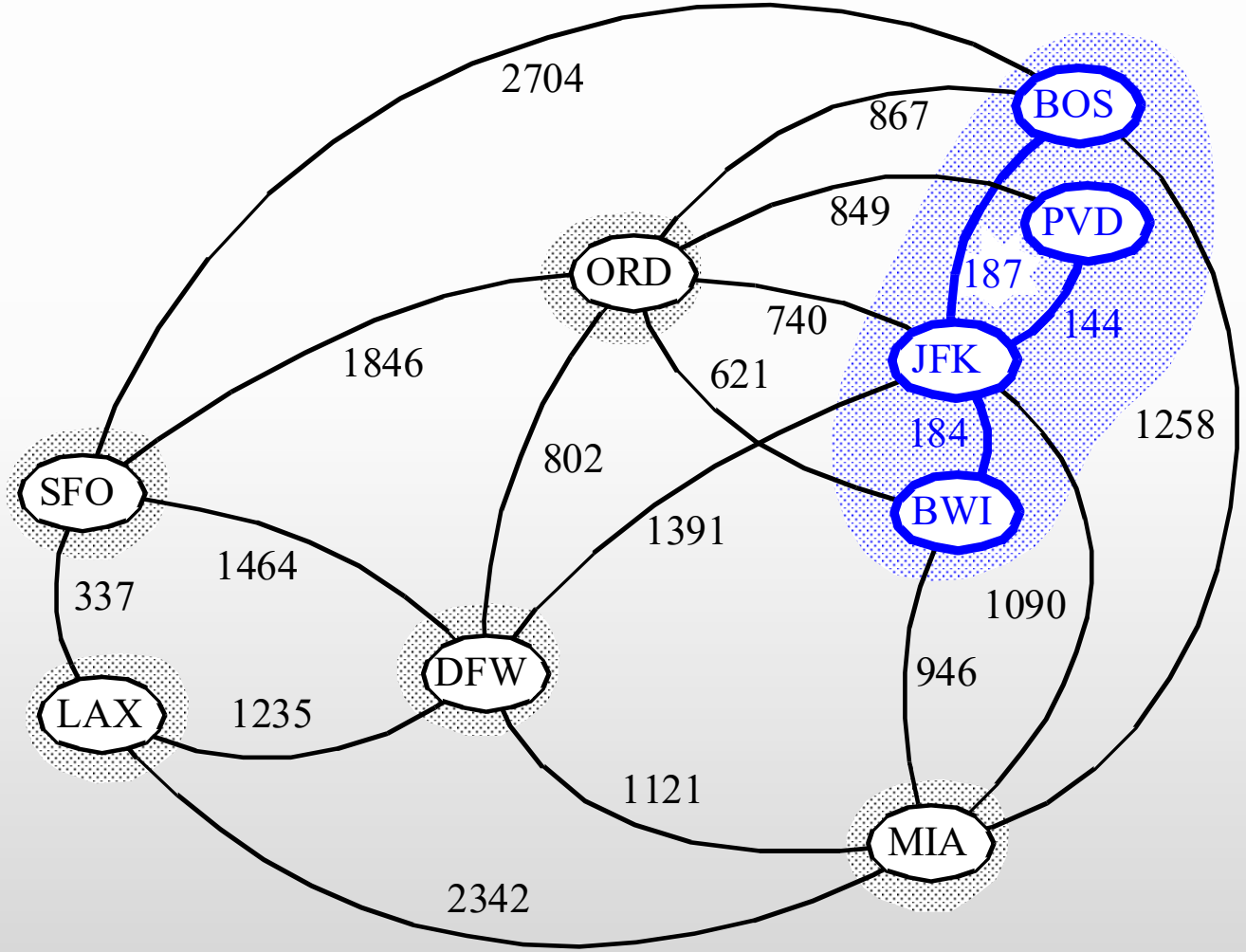
Kruskal Algoritması Uygulama



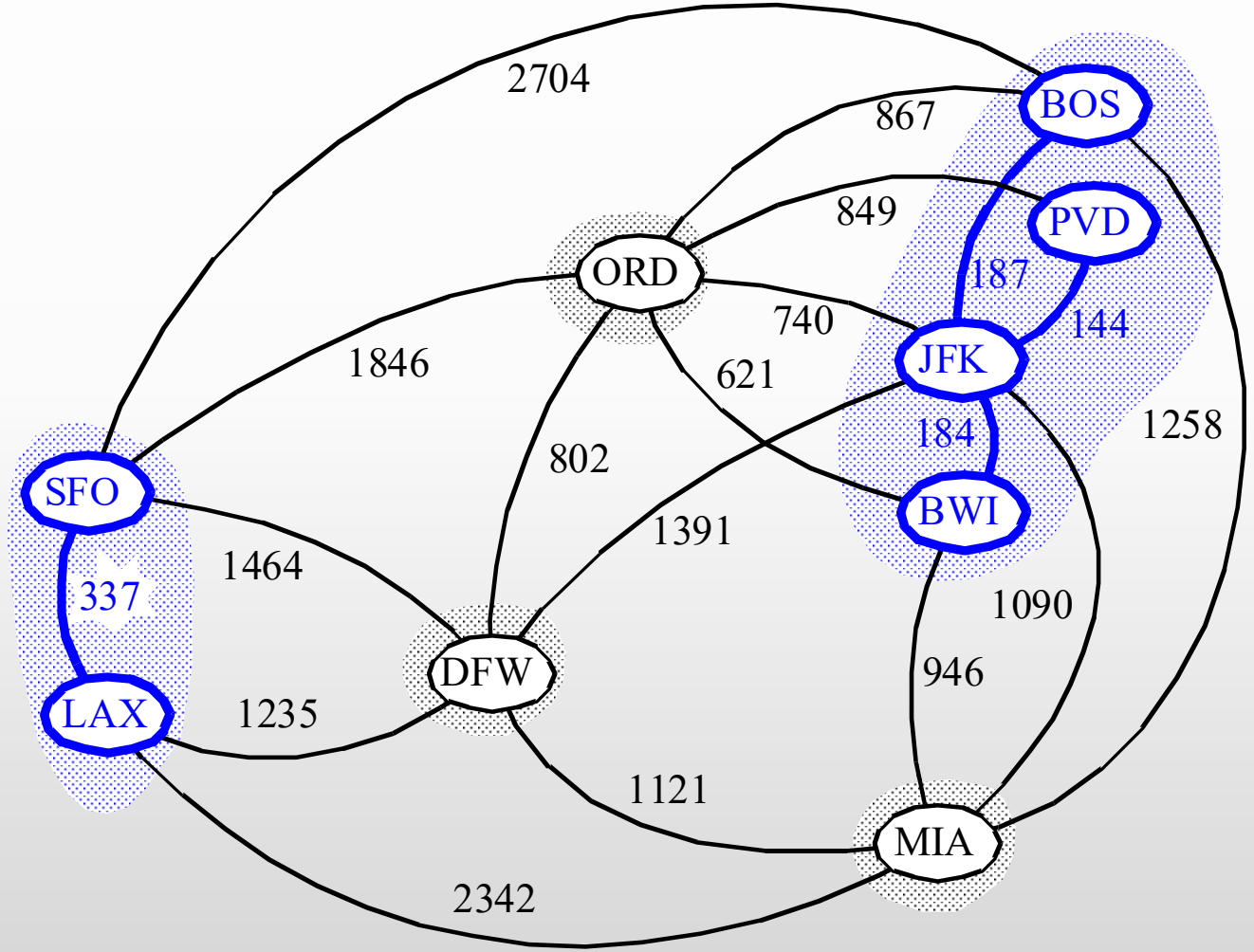
Kruskal Algoritması Uygulama



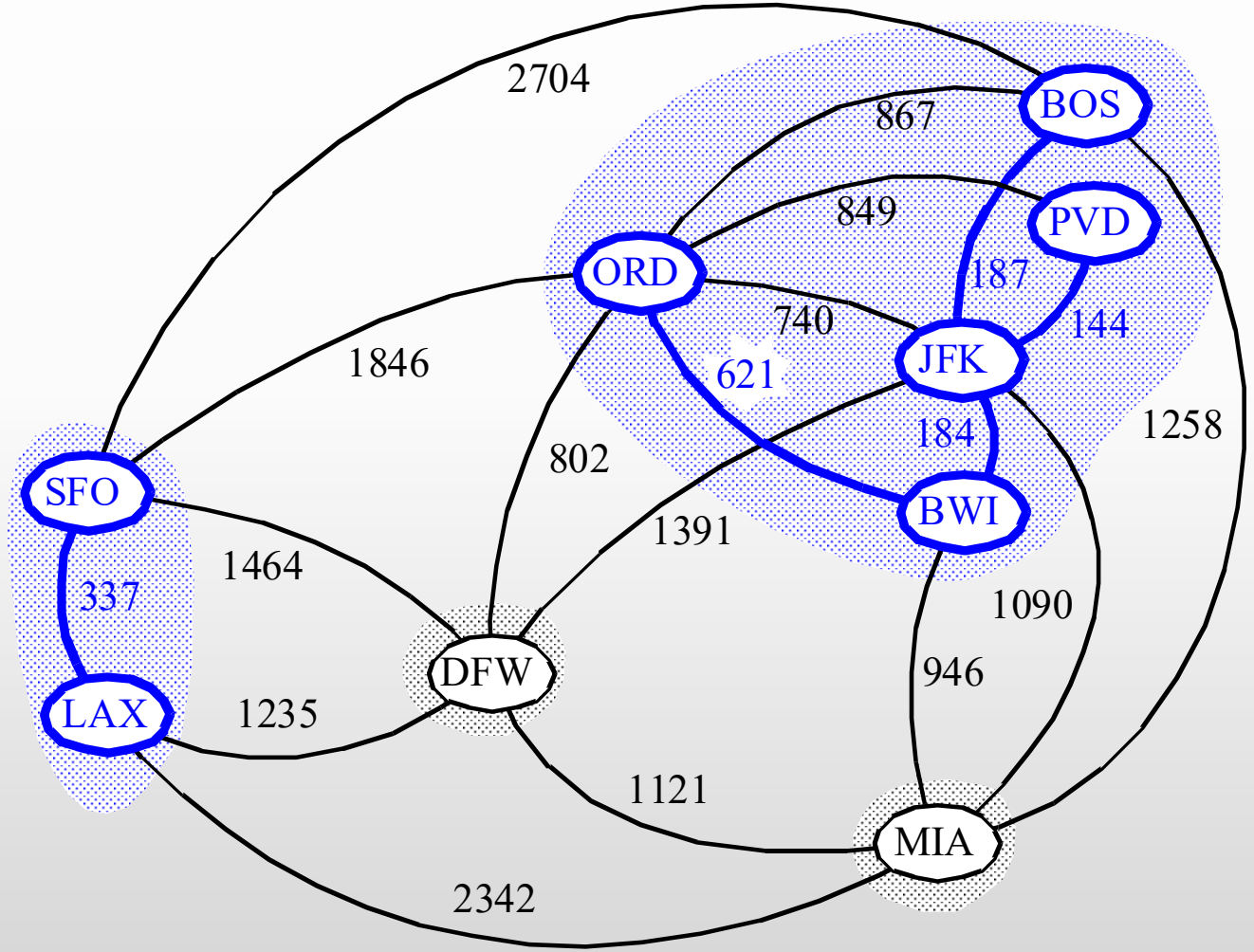
Kruskal Algoritması Uygulama



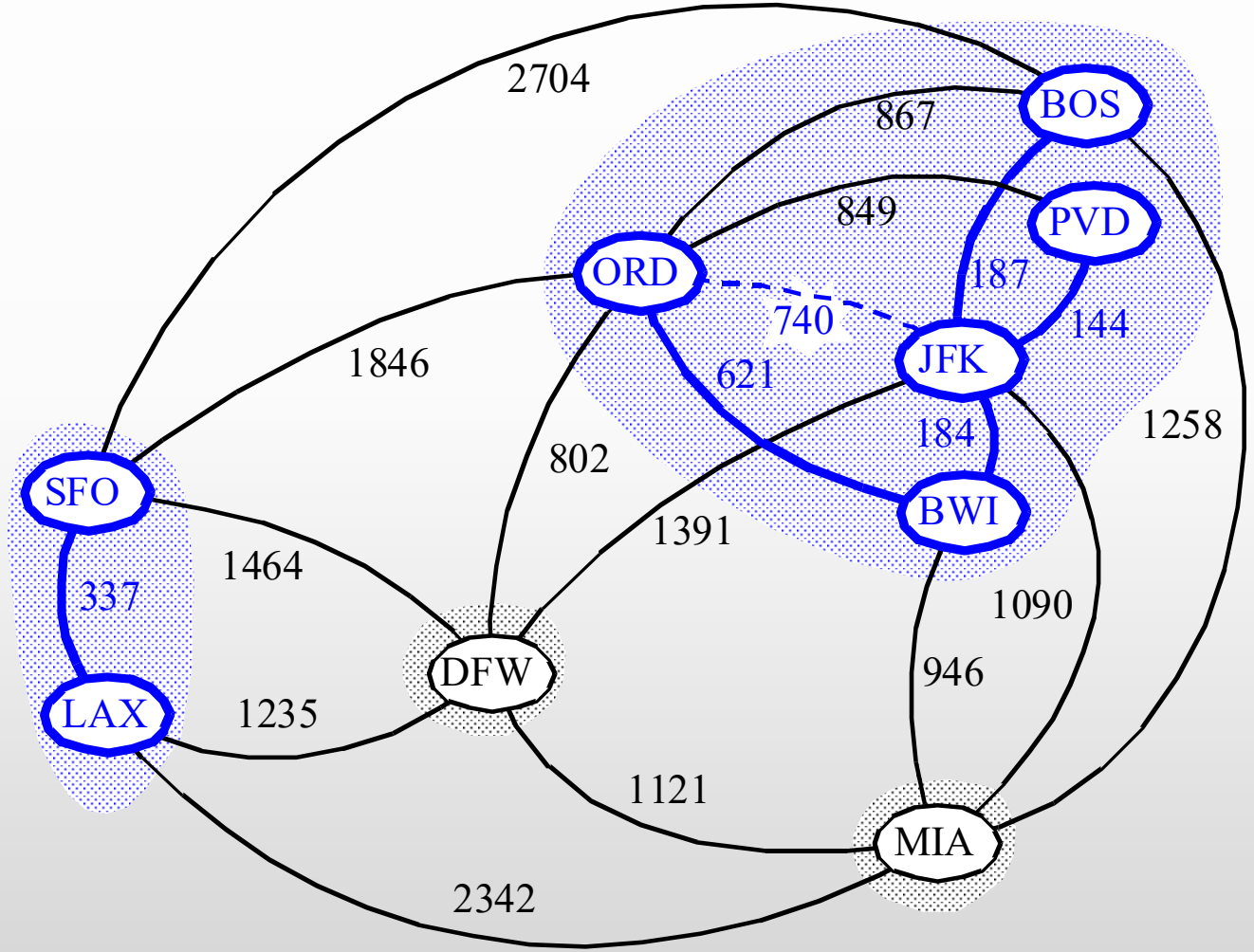
Kruskal Algoritması Uygulama



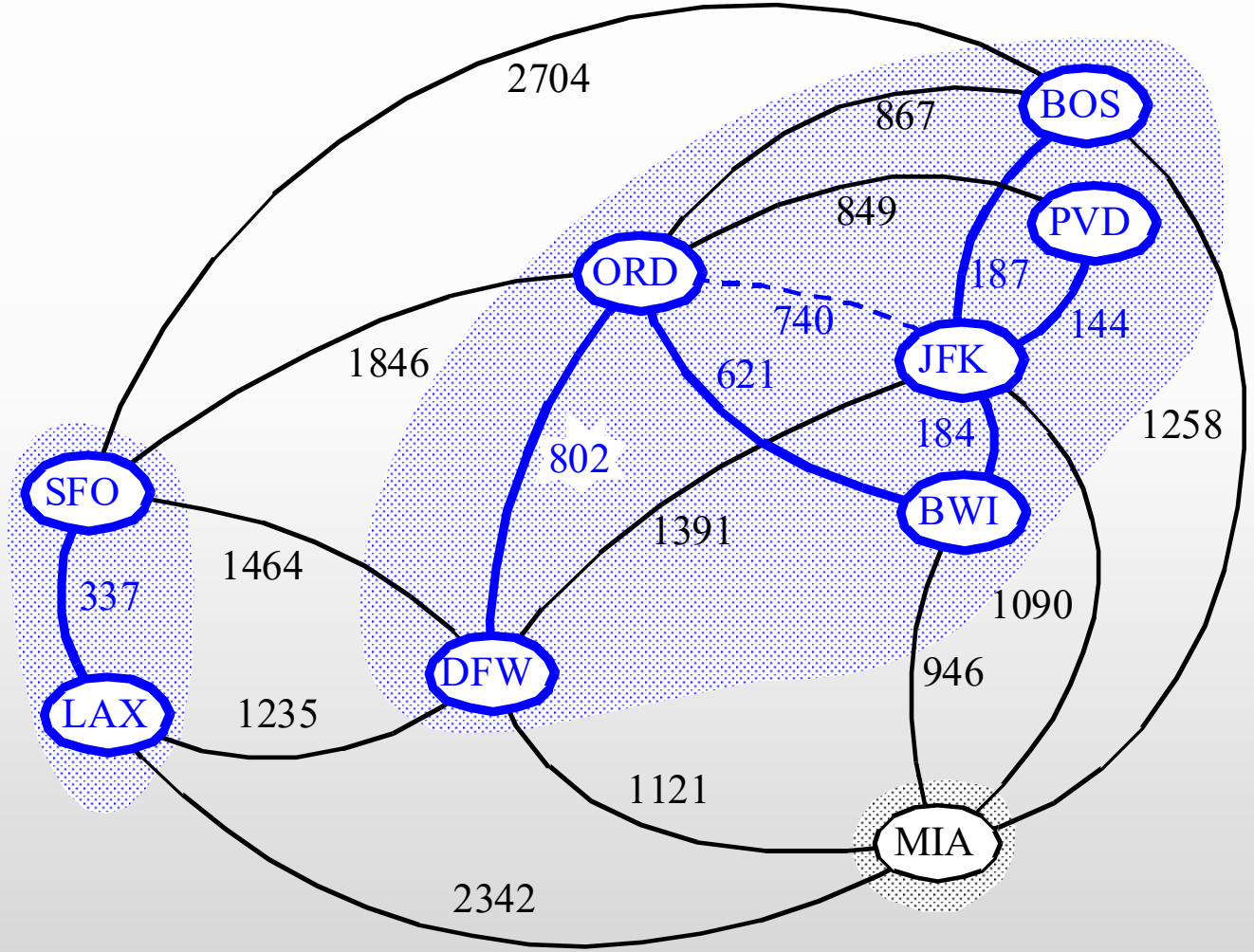
Kruskal Algoritması Uygulama



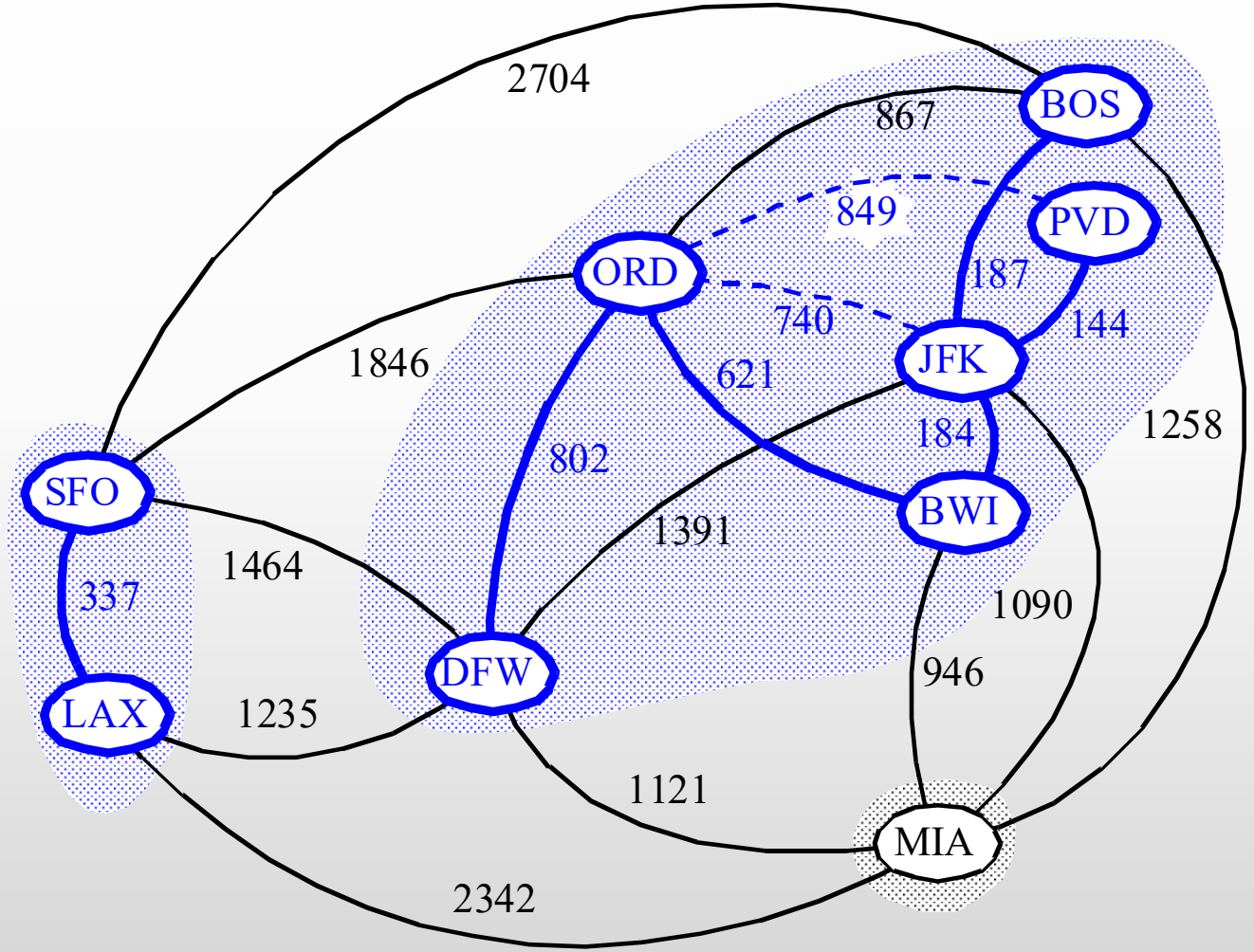
Kruskal Algoritması Uygulama



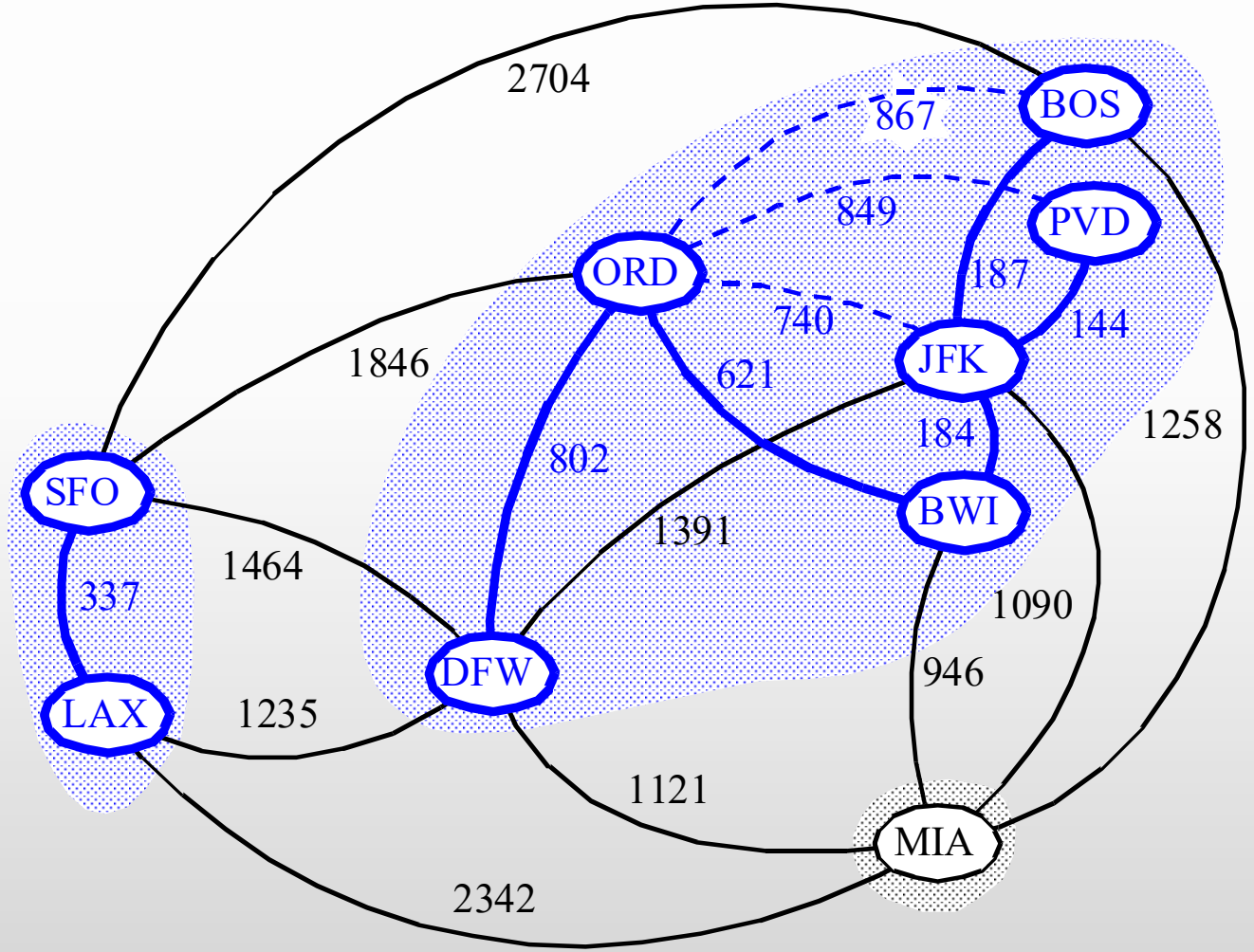
Kruskal Algoritması Uygulama



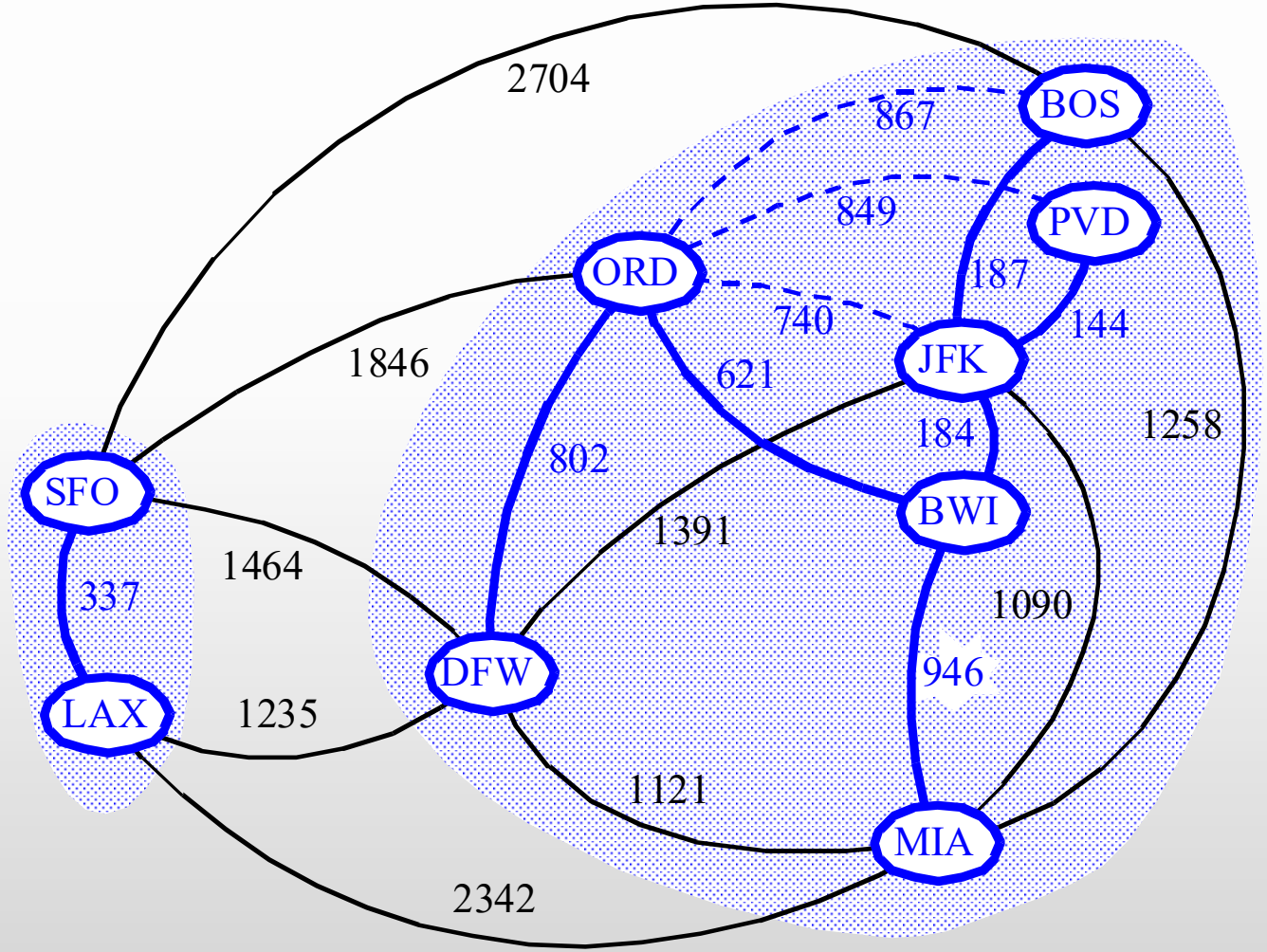
Kruskal Algoritması Uygulama



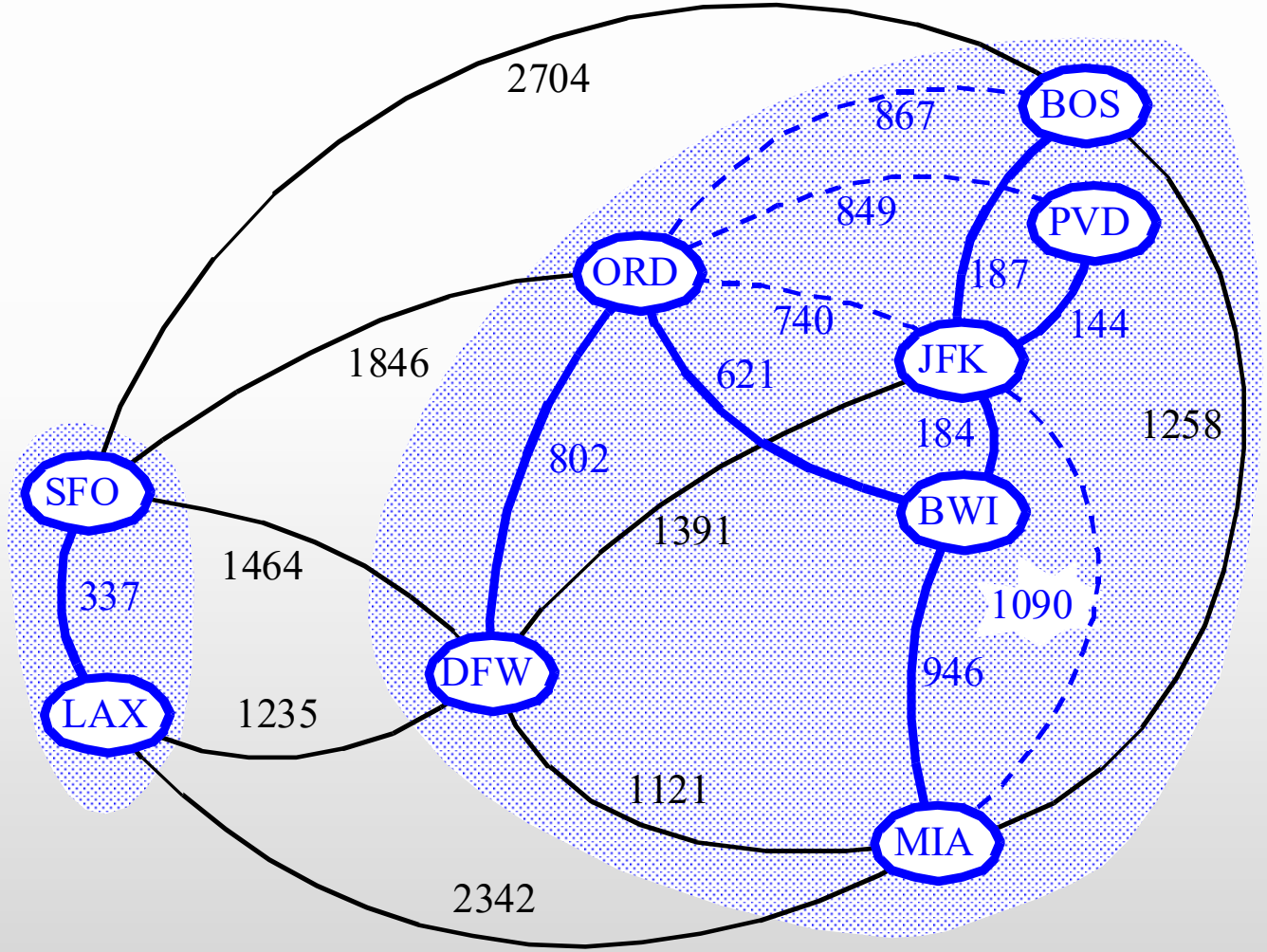
Kruskal Algoritması Uygulama



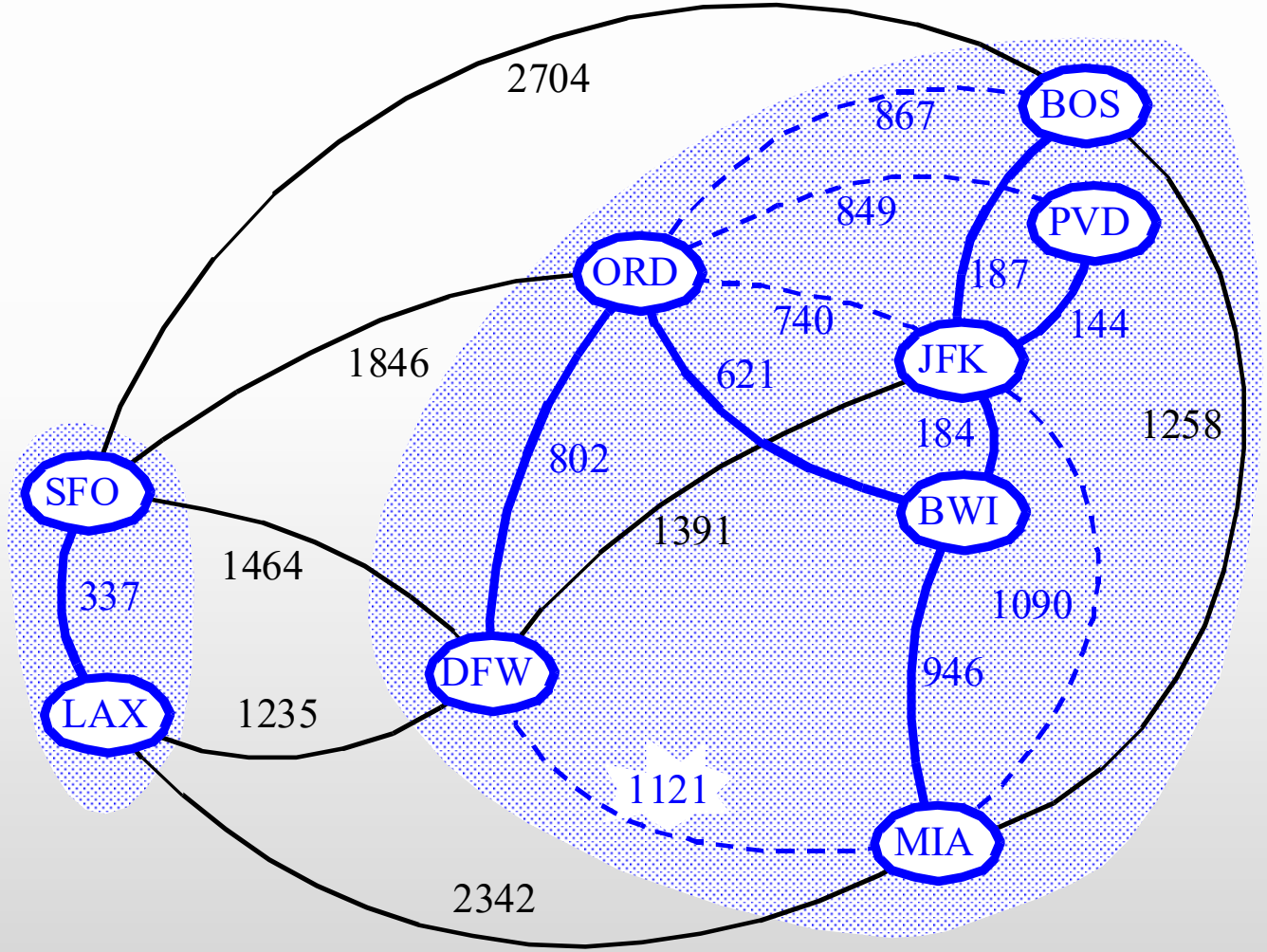
Kruskal Algoritması Uygulama



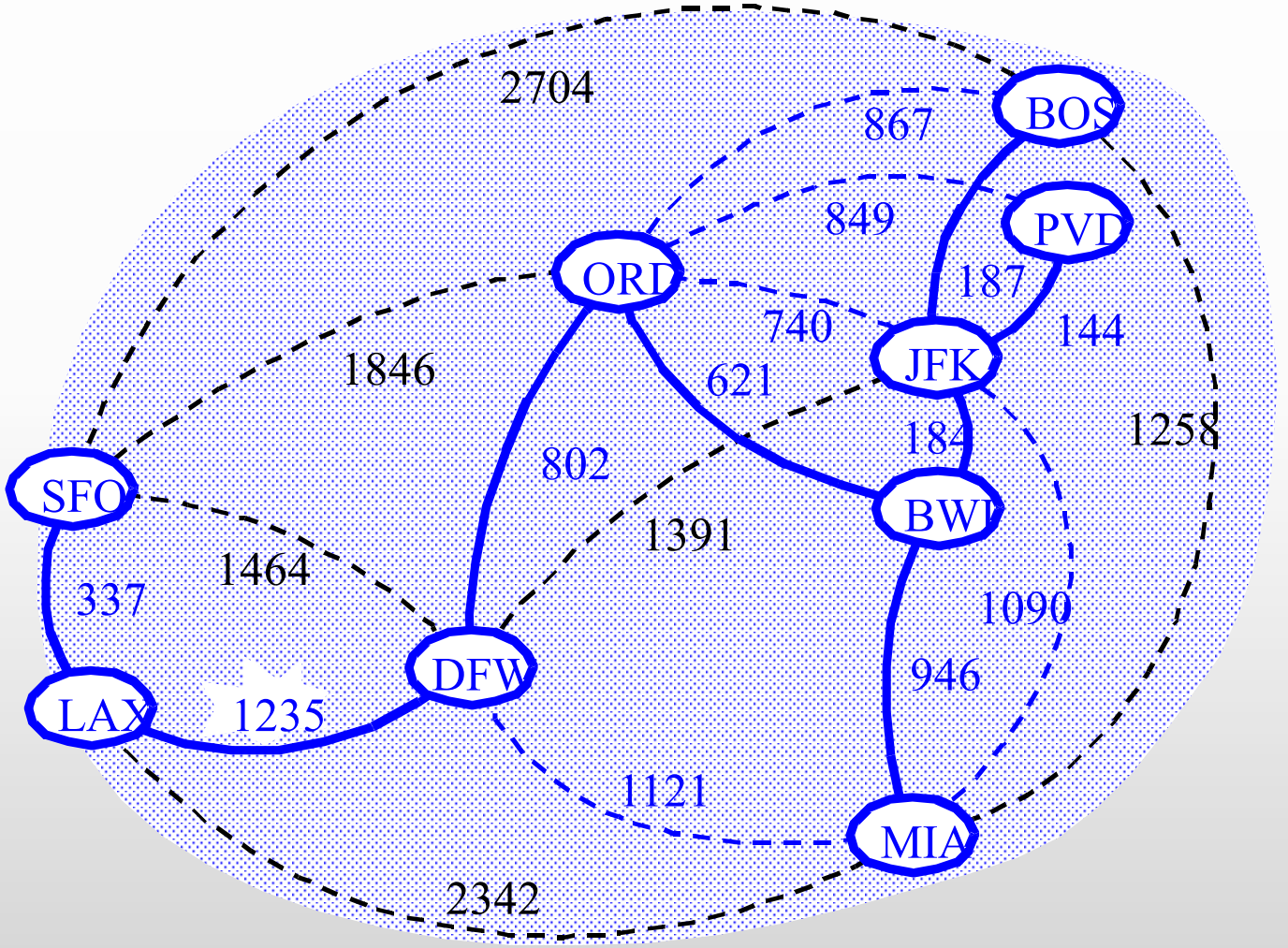
Kruskal Algoritması Uygulama



Kruskal Algoritması Uygulama



Kruskal Algoritması Uygulama



Küme İşleçleri

- **Birleştirme (Union):** İki kümenin elemanlarından bir küme oluşturur.
- **Kesişim (Intersection):** İki kümenin ortak elemanlarından bir küme oluşturur.
- **Fark (Difference):** Bir kümenin diğer kümede olmayan elemanlarından bir küme oluşturur.
- **Alt küme (Subset):** Bir kümenin diğer bir kümenin alt kümesi olup olmadığını söyler.

Kümelerin Gösterimi

- **A** = {1, 2, 3, 4, 5}
- **B** = {3, 4, 5, 6, 7}
- **Birleştirme**: A ∪ B = {1, 2, 3, 4, 5, 6, 7}
- **Kesişim**: A ∩ B = {3, 4, 5}
- **Fark**: A - B = {1, 2}
- **Alt küme**: A ⊆ B (A, B'nin alt kümesi değildir)

java.util.Set Arayüzü Metodları

- **add(E eleman):** Belirtilen elemanı kümeye ekler.
- **remove(Object eleman):** Belirtilen elemanı kümeden çıkarır.
- **contains(Object eleman):** Elemanın kümede olup olmadığını döndürür.
- **size():** Kümenin eleman sayısını döndürür.
- **isEmpty():** Kümenin boş olup olmadığını söyler.
- **clear():** Kümeden tüm elemanları çıkarır.

java.util.Set Arayüzü Uygulamaları

- **HashSet**: Elemanları hash fonksiyonu çıktısına göre bir sırada saklar.
- **LinkedHashSet**: Elemanları kümeye eklenme sırasına göre saklar.
- **TreeSet**: Elemanları belirli sırada saklar. (alfabetik gibi)

Hash Tabanlı Küme

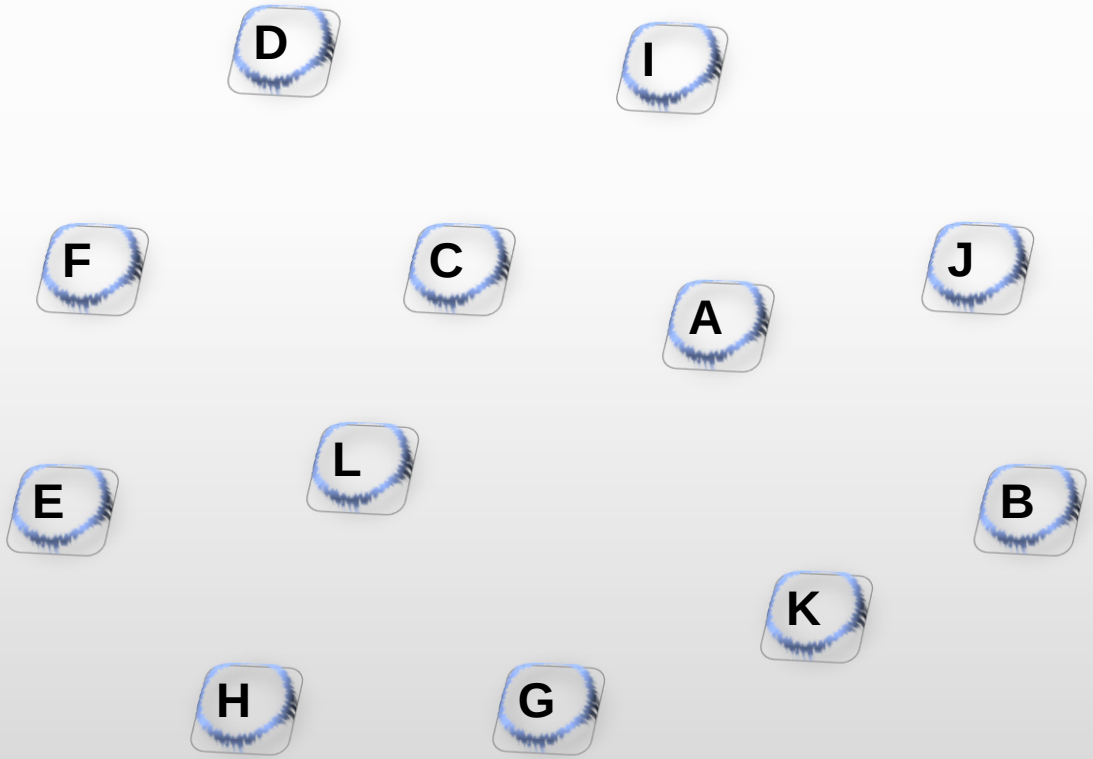
- Ekleme ve arama işlemleri hızlıdır.
- Küme, bir hash tablosu olarak temsil edilir.
- Her bir eleman, hash koduna dayalı olarak saklanır.
- Elemanların konumu hızlı bir şekilde hesaplanabilir.

Ağaç Tabanlı Küme

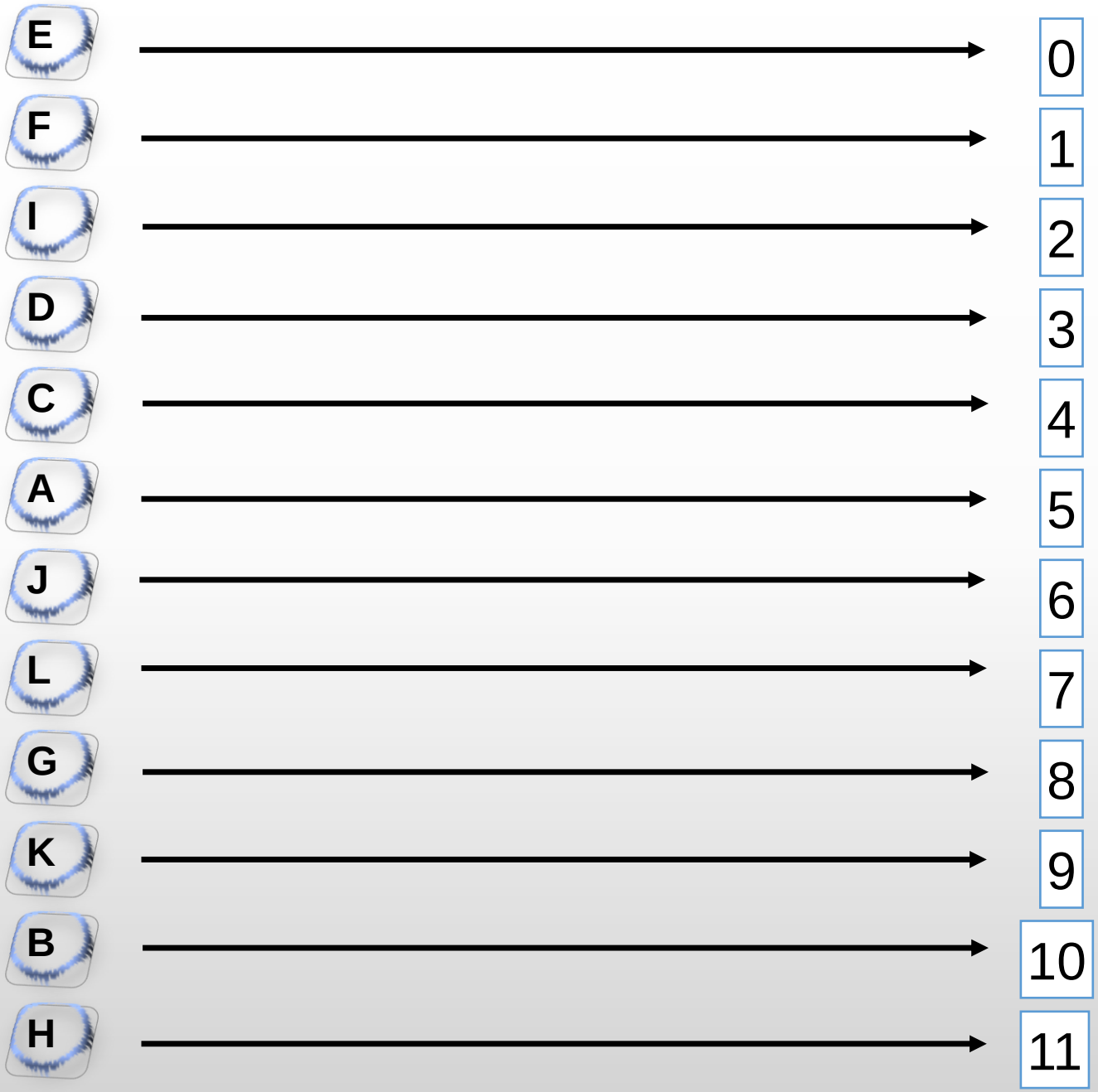
- Elemanları sıralı bir şekilde saklar.
- Küme, bir ikili arama ağacı olarak temsil edilir.
- Elemanlar sıralı bir şekilde saklanır.
- Arama işlemi O(log n) zaman karmaşıklığına sahiptir.

Union İşlemi

- Nesne ve sayıları rastgele eşleştir.



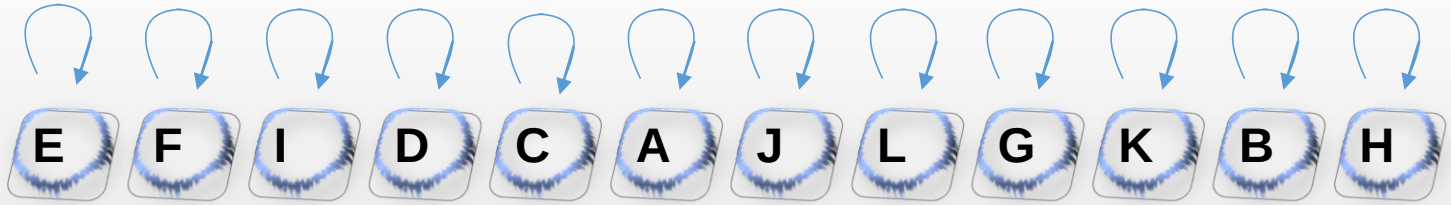
0
1
2
3
4
5
6
7
8
9
10
11


E
F
I
D
C
A
J
L
G
K
B
H
0
1
2
3
4
5
6
7
8
9
10
11

Union İşlemi

```
Union(C,K)
Union(F,E)
Union(A,J)
Union(A,B)
Union(C,D)
Union(D,I)
Union(L,F)
Union(C,A)
Union(A,B)
Union(H,G)
Union(H,F)
Union(H,B)
```

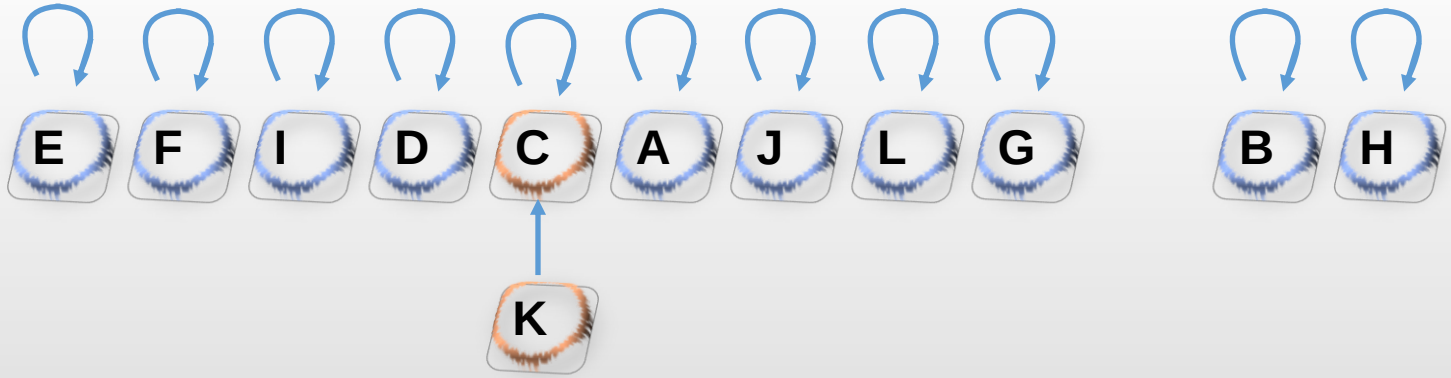
E	F	I	D	C	A	J	L	G	K	B	H
0	1	2	3	4	5	6	7	8	9	10	11
0	1	2	3	4	5	6	7	8	9	10	11



Union İşlemi

```
Union(C,K)
Union(F,E)
Union(A,J)
Union(A,B)
Union(C,D)
Union(D,I)
Union(L,F)
Union(C,A)
Union(A,B)
Union(H,G)
Union(H,F)
Union(H,B)
```

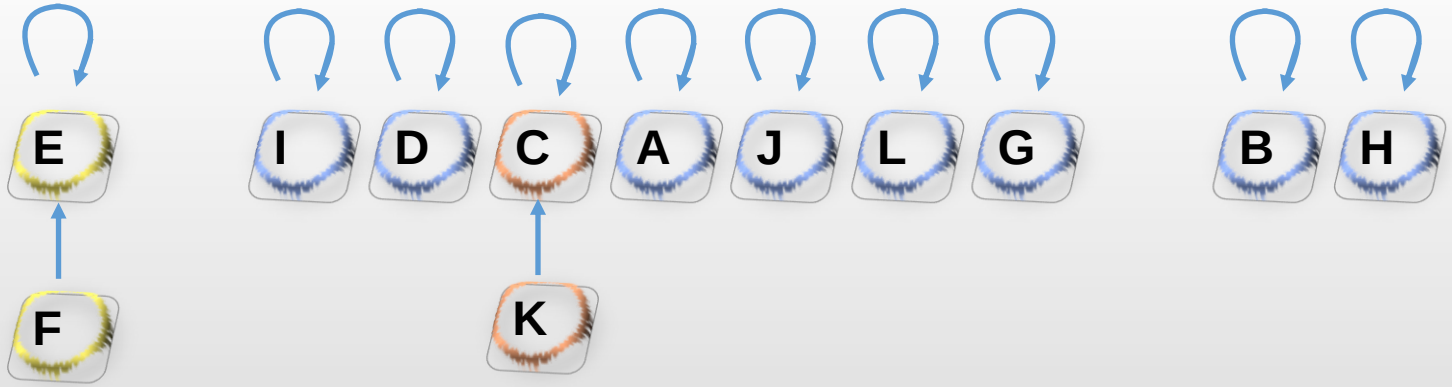
E	F	I	D	C	A	J	L	G	K	B	H
0	1	2	3	4	5	6	7	8	4	10	11
0	1	2	3	4	5	6	7	8	9	10	11



Union İşlemi

```
Union(C,K)
Union(F,E)
Union(A,J)
Union(A,B)
Union(C,D)
Union(D,I)
Union(L,F)
Union(C,A)
Union(A,B)
Union(H,G)
Union(H,F)
Union(H,B)
```

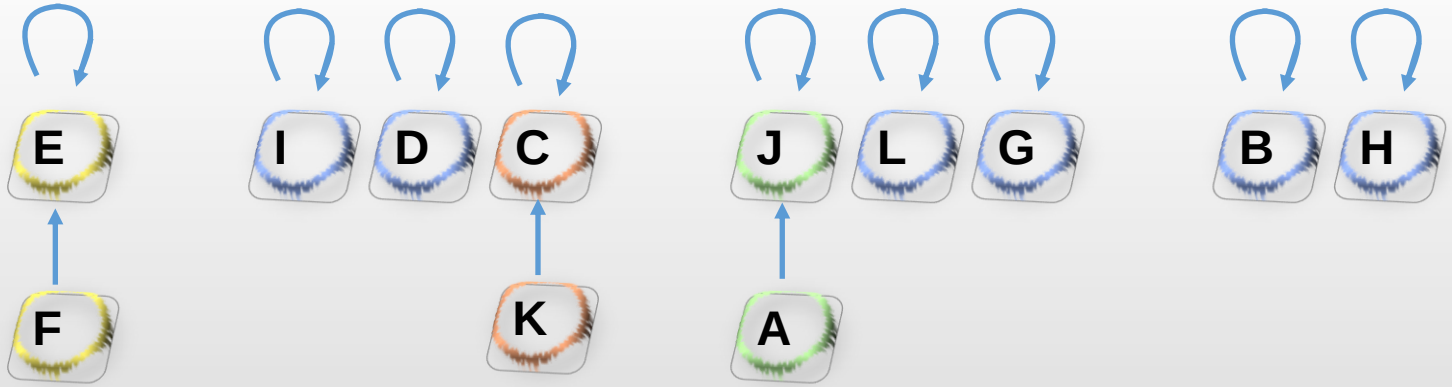
E	F	I	D	C	A	J	L	G	K	B	H
0	0	2	3	4	5	6	7	8	4	10	11
0	1	2	3	4	5	6	7	8	9	10	11



Union İşlemi

```
Union(C,K)
Union(F,E)
Union(A,J)
Union(A,B)
Union(C,D)
Union(D,I)
Union(L,F)
Union(C,A)
Union(A,B)
Union(H,G)
Union(H,F)
Union(H,B)
```

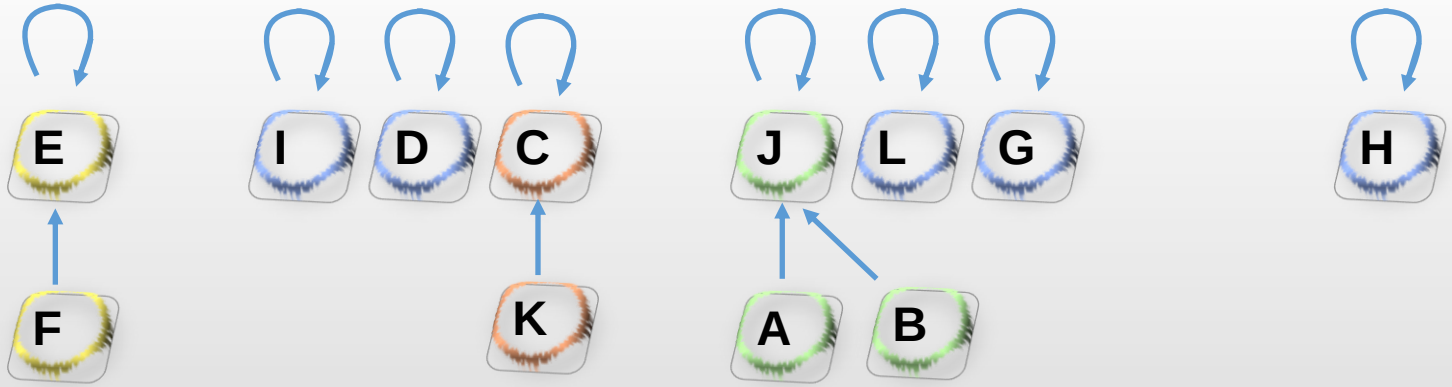
E	F	I	D	C	A	J	L	G	K	B	H
0	0	2	3	4	6	6	7	8	4	10	11
0	1	2	3	4	5	6	7	8	9	10	11



Union İşlemi

```
Union(C,K)
Union(F,E)
Union(A,J)
Union(A,B)
Union(C,D)
Union(D,I)
Union(L,F)
Union(C,A)
Union(A,B)
Union(H,G)
Union(H,F)
Union(H,B)
```

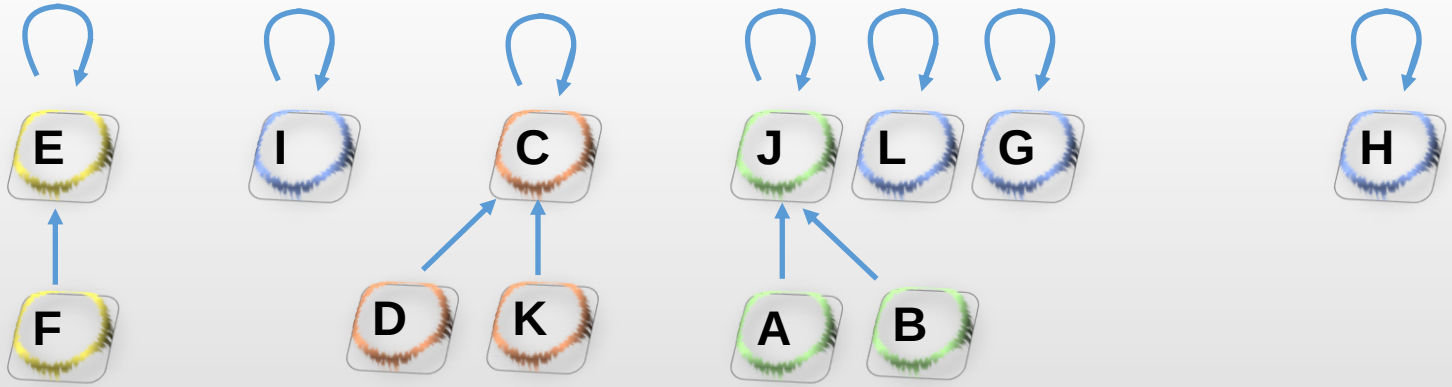
E	F	I	D	C	A	J	L	G	K	B	H
0	0	2	3	4	6	6	7	8	4	6	11
0	1	2	3	4	5	6	7	8	9	10	11



Union İşlemi

```
Union(C,K)
Union(F,E)
Union(A,J)
Union(A,B)
Union(C,D)
Union(D,I)
Union(L,F)
Union(C,A)
Union(A,B)
Union(H,G)
Union(H,F)
Union(H,B)
```

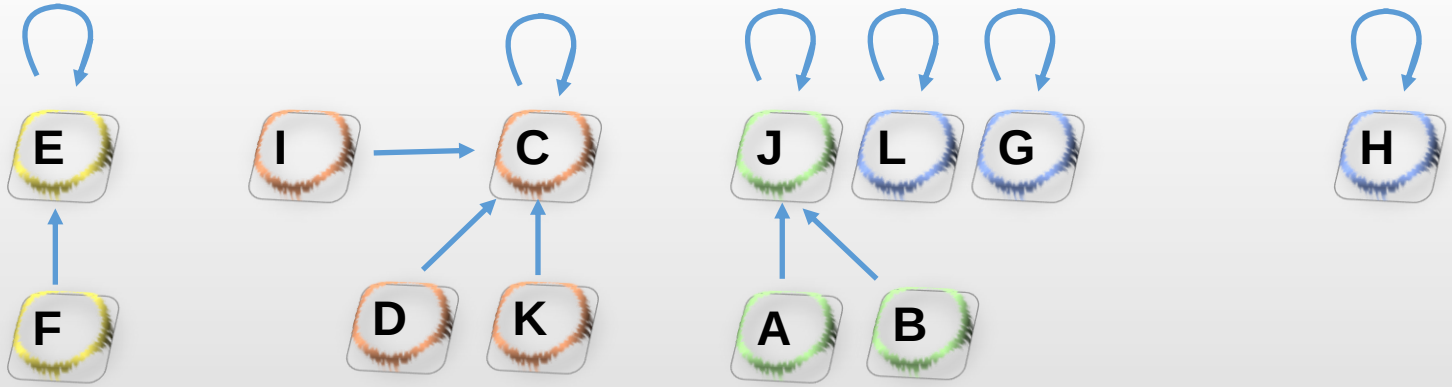
E	F	I	D	C	A	J	L	G	K	B	H
0	0	2	4	4	6	6	7	8	4	6	11
0	1	2	3	4	5	6	7	8	9	10	11



Union İşlemi

```
Union(C,K)
Union(F,E)
Union(A,J)
Union(A,B)
Union(C,D)
Union(D,I)
Union(L,F)
Union(C,A)
Union(A,B)
Union(H,G)
Union(H,F)
Union(H,B)
```

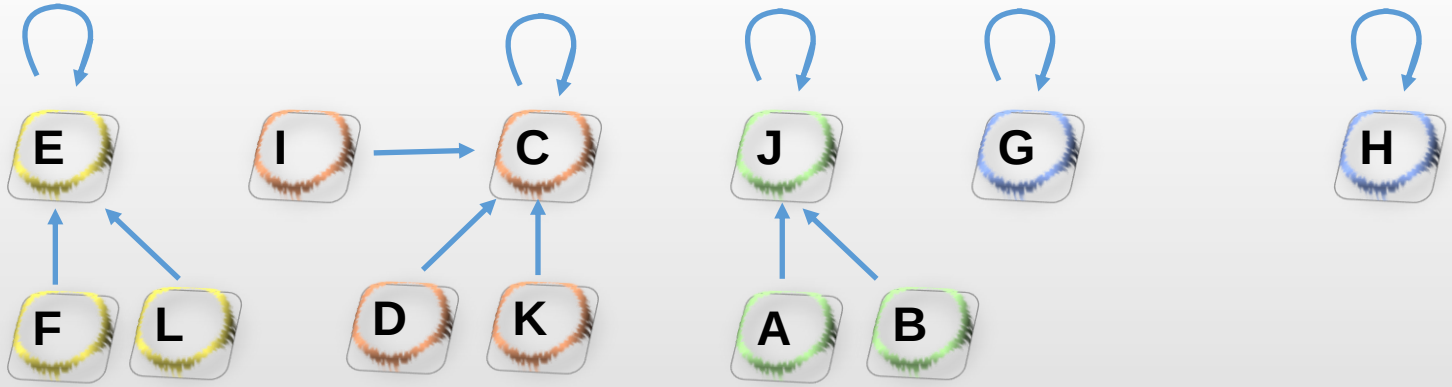
E	F	I	D	C	A	J	L	G	K	B	H
0	0	4	4	4	6	6	7	8	4	6	11
0	1	2	3	4	5	6	7	8	9	10	11



Union İşlemi

```
Union(C,K)
Union(F,E)
Union(A,J)
Union(A,B)
Union(C,D)
Union(D,I)
Union(L,F)
Union(C,A)
Union(A,B)
Union(H,G)
Union(H,F)
Union(H,B)
```

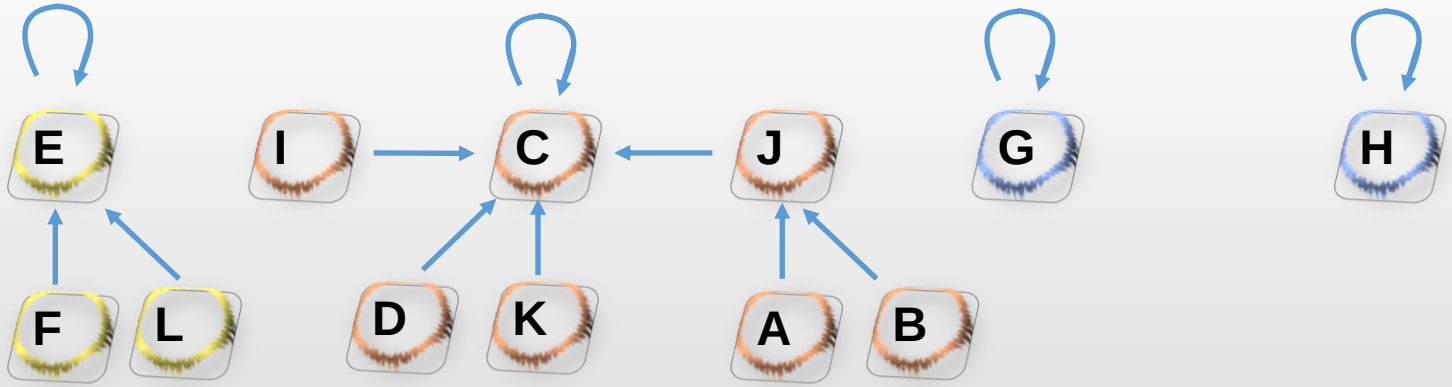
E	F	I	D	C	A	J	L	G	K	B	H
0	0	4	4	4	6	6	0	8	4	6	11
0	1	2	3	4	5	6	7	8	9	10	11



Union İşlemi

```
Union(C,K)
Union(F,E)
Union(A,J)
Union(A,B)
Union(C,D)
Union(D,I)
Union(L,F)
Union(C,A)
Union(A,B)
Union(H,G)
Union(H,F)
Union(H,B)
```

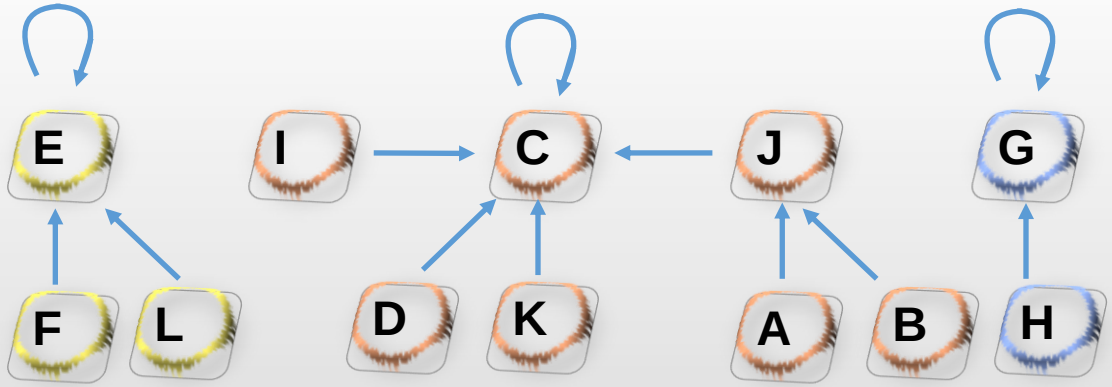
E	F	I	D	C	A	J	L	G	K	B	H
0	0	4	4	4	6	4	0	8	4	6	11
0	1	2	3	4	5	6	7	8	9	10	11



Union İşlemi

```
Union(C,K)
Union(F,E)
Union(A,J)
Union(A,B)
Union(C,D)
Union(D,I)
Union(L,F)
Union(C,A)
Union(A,B)
Union(H,G)
Union(H,F)
Union(H,B)
```

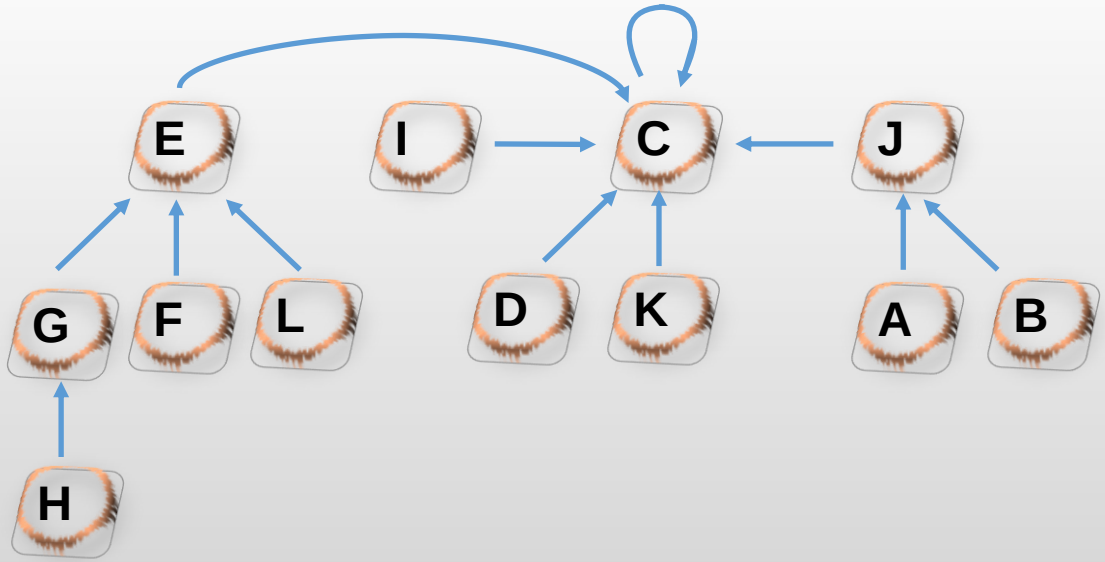
E	F	I	D	C	A	J	L	G	K	B	H
0	0	4	4	4	6	4	0	8	4	6	8
0	1	2	3	4	5	6	7	8	9	10	11



Union İşlemi

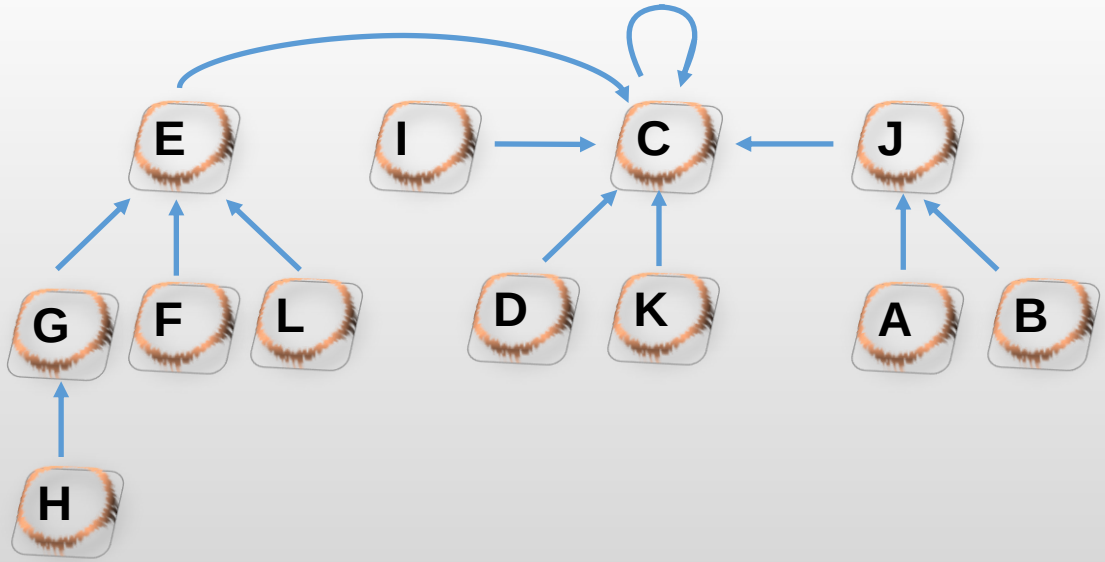
```
Union(C,K)
Union(F,E)
Union(A,J)
Union(A,B)
Union(C,D)
Union(D,I)
Union(L,F)
Union(C,A)
Union(A,B)
Union(H,G)
Union(H,F)
Union(H,B)
```

E	F	I	D	C	A	J	L	G	K	B	H
4	0	4	4	4	6	4	0	0	4	6	8
0	1	2	3	4	5	6	7	8	9	10	11



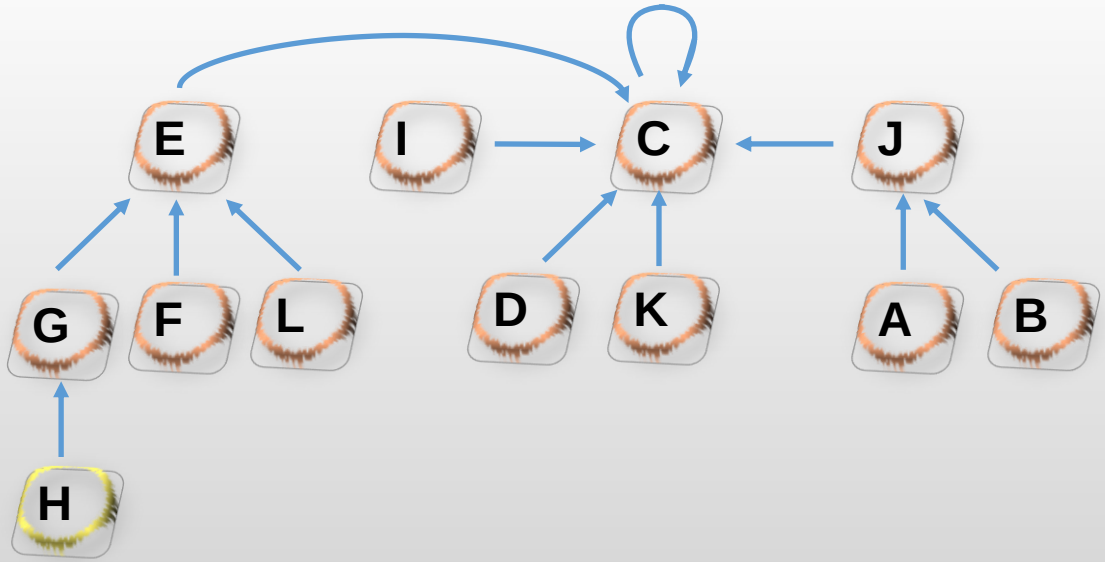
Union İşlemi

- H ve B'nin aynı gruba ait olup olmadığını kontrol etmek beş adım alır.
- En kötü durumda, bu süreç çok daha uzun sürebilir.
- α(n) zaman karmaşıklığı için iyileştirme yapılmalı.



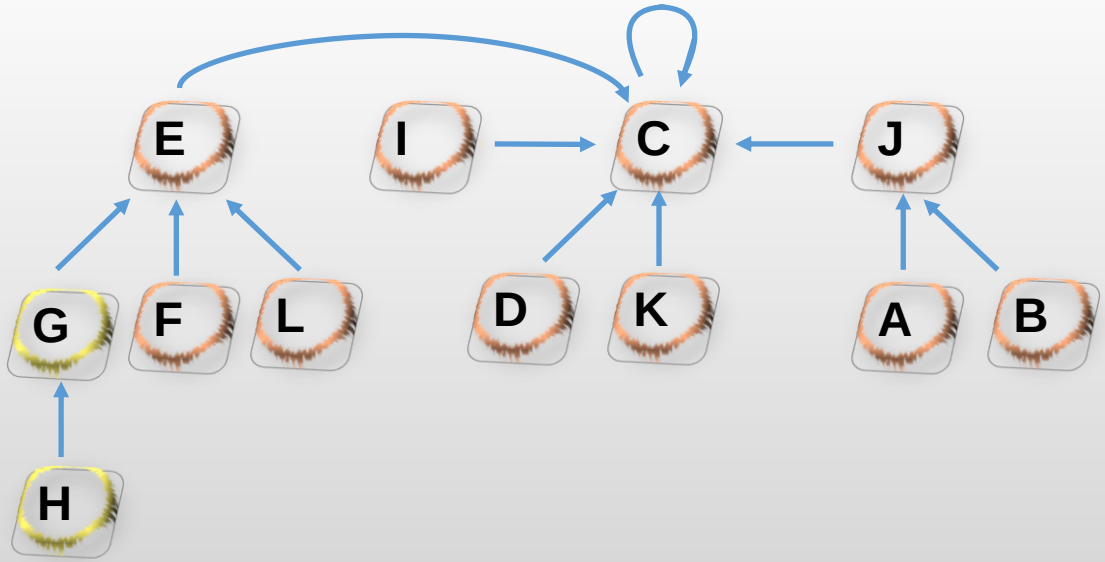
Union İşlemi

- H ve B'nin aynı gruba ait olup olmadığını kontrol etmek beş adım alır.
- En kötü durumda, bu süreç çok daha uzun sürebilir.
- α(n) zaman karmaşıklığı için iyileştirme yapılmalı.



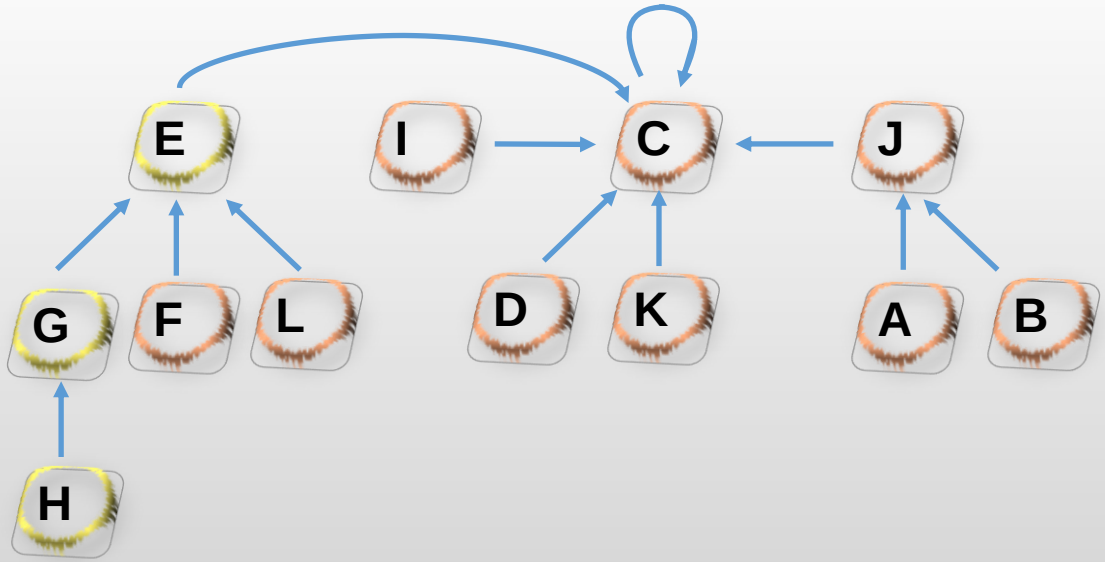
Union İşlemi

- H ve B'nin aynı gruba ait olup olmadığını kontrol etmek beş adım alır.
- En kötü durumda, bu süreç çok daha uzun sürebilir.
- α(n) zaman karmaşıklığı için iyileştirme yapılmalı.



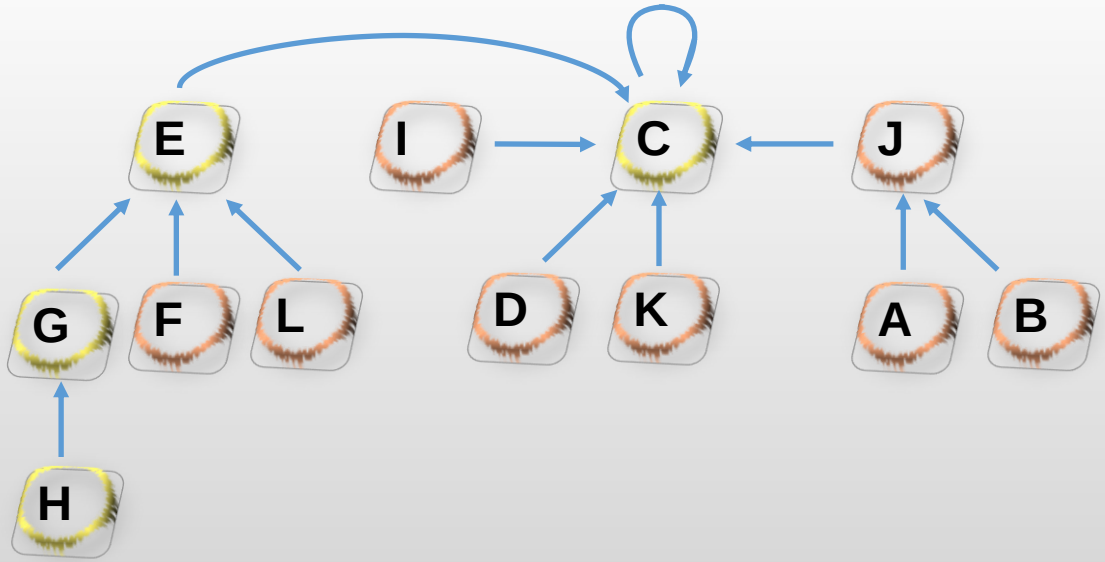
Union İşlemi

- H ve B'nin aynı gruba ait olup olmadığını kontrol etmek beş adım alır.
- En kötü durumda, bu süreç çok daha uzun sürebilir.
- α(n) zaman karmaşıklığı için iyileştirme yapılmalı.



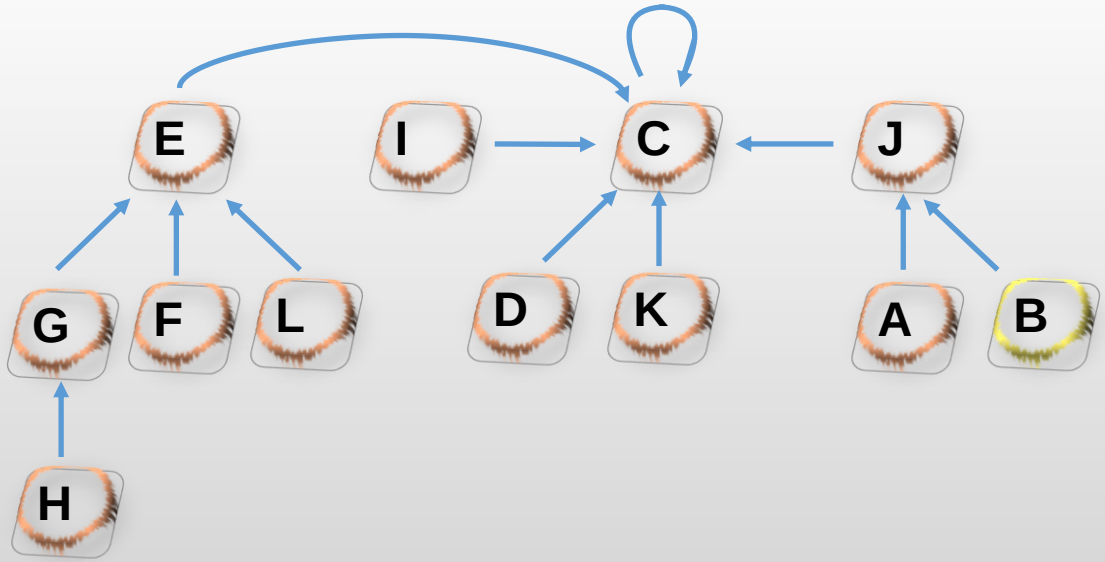
Union İşlemi

- H ve B'nin aynı gruba ait olup olmadığını kontrol etmek beş adım alır.
- En kötü durumda, bu süreç çok daha uzun sürebilir.
- α(n) zaman karmaşıklığı için iyileştirme yapılmalı.



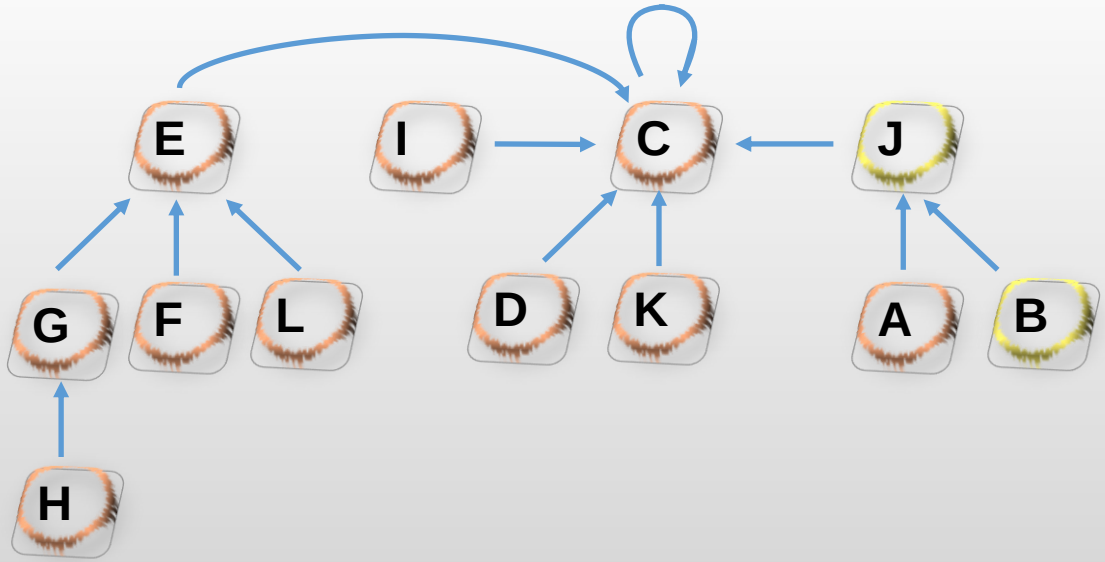
Union İşlemi

- H ve B'nin aynı gruba ait olup olmadığını kontrol etmek beş adım alır.
- En kötü durumda, bu süreç çok daha uzun sürebilir.
- α(n) zaman karmaşıklığı için iyileştirme yapılmalı.



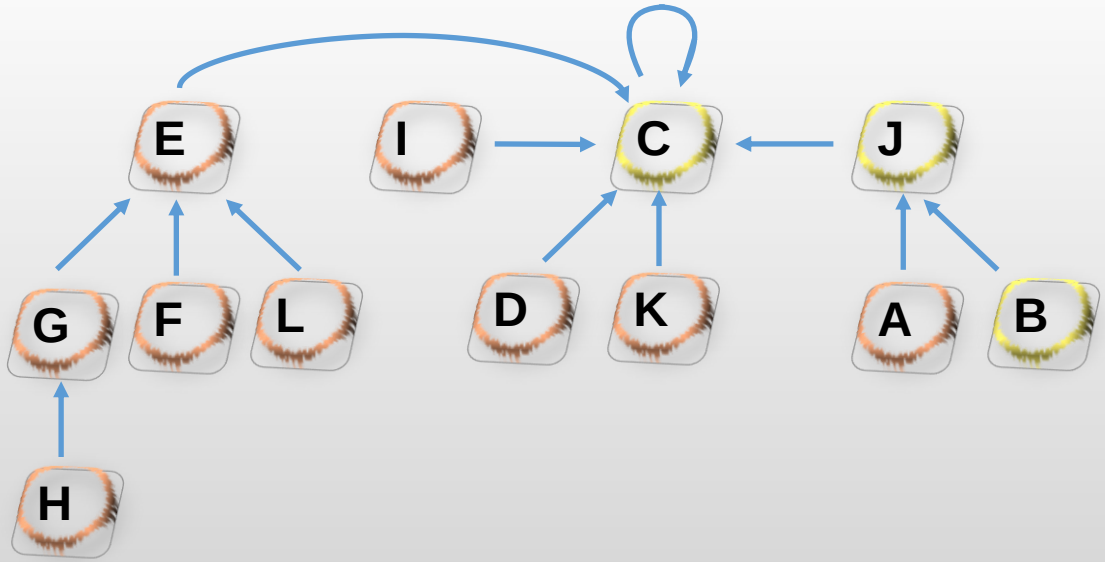
Union İşlemi

- H ve B'nin aynı gruba ait olup olmadığını kontrol etmek beş adım alır.
- En kötü durumda, bu süreç çok daha uzun sürebilir.
- α(n) zaman karmaşıklığı için iyileştirme yapılmalı.



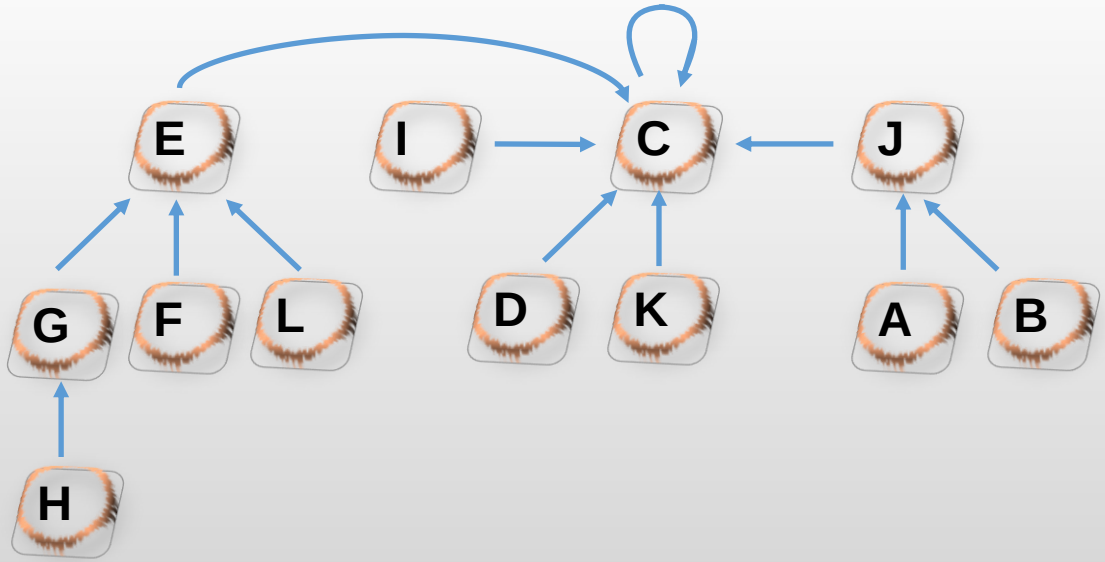
Union İşlemi

- H ve B'nin aynı gruba ait olup olmadığını kontrol etmek beş adım alır.
- En kötü durumda, bu süreç çok daha uzun sürebilir.
- α(n) zaman karmaşıklığı için iyileştirme yapılmalı.

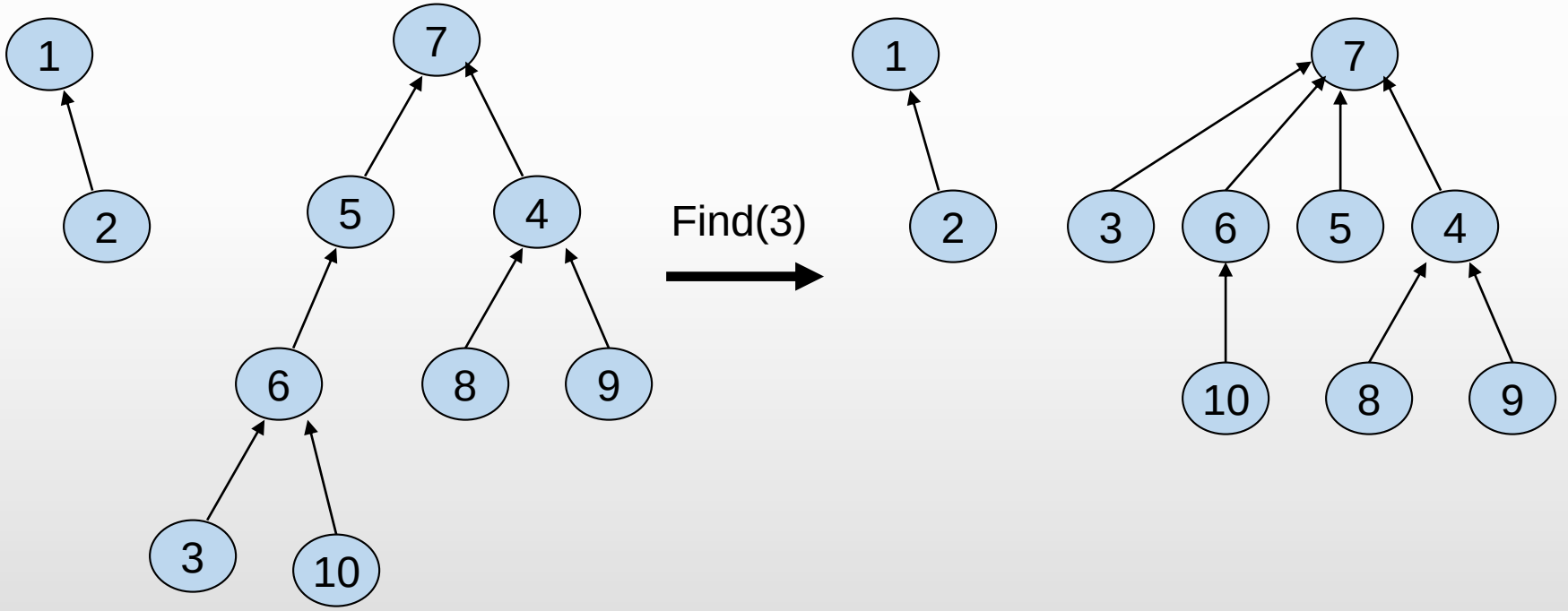


Union İşlemi

- H ve B'nin aynı gruba ait olup olmadığını kontrol etmek beş adım alır.
- En kötü durumda, bu süreç çok daha uzun sürebilir.
- α(n) zaman karmaşıklığı için iyileştirme yapılmalı.



Yol Kısaltma (path compression)

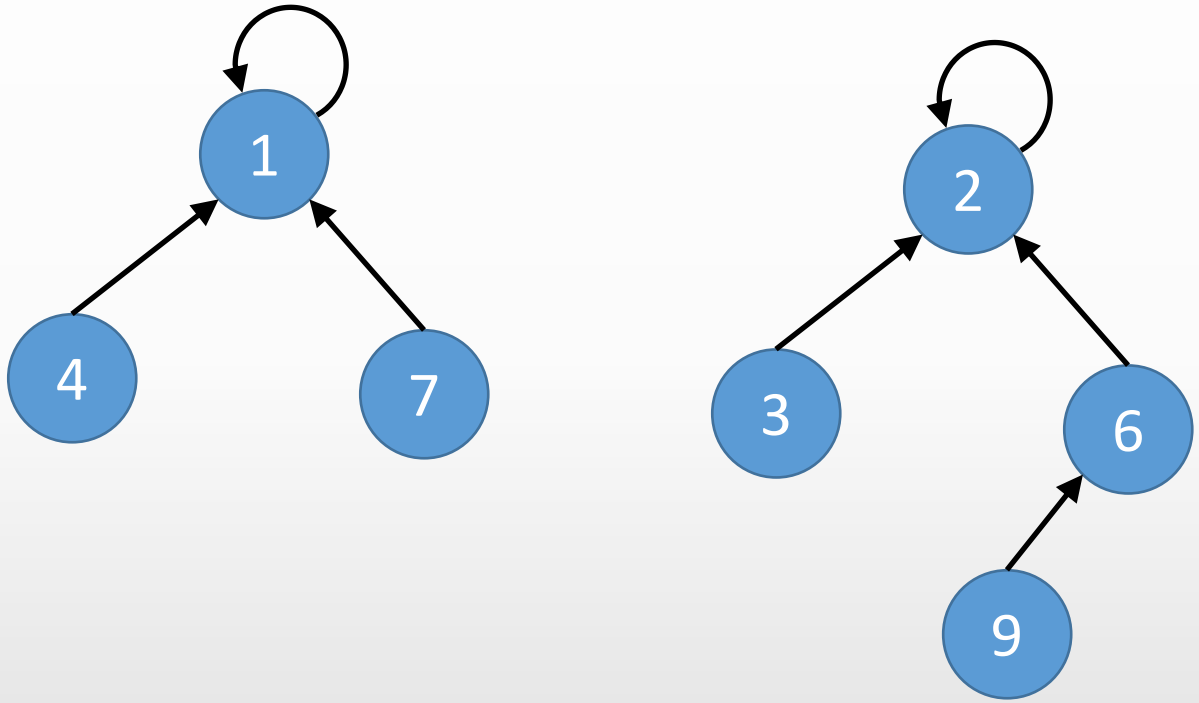


Algoritmalar

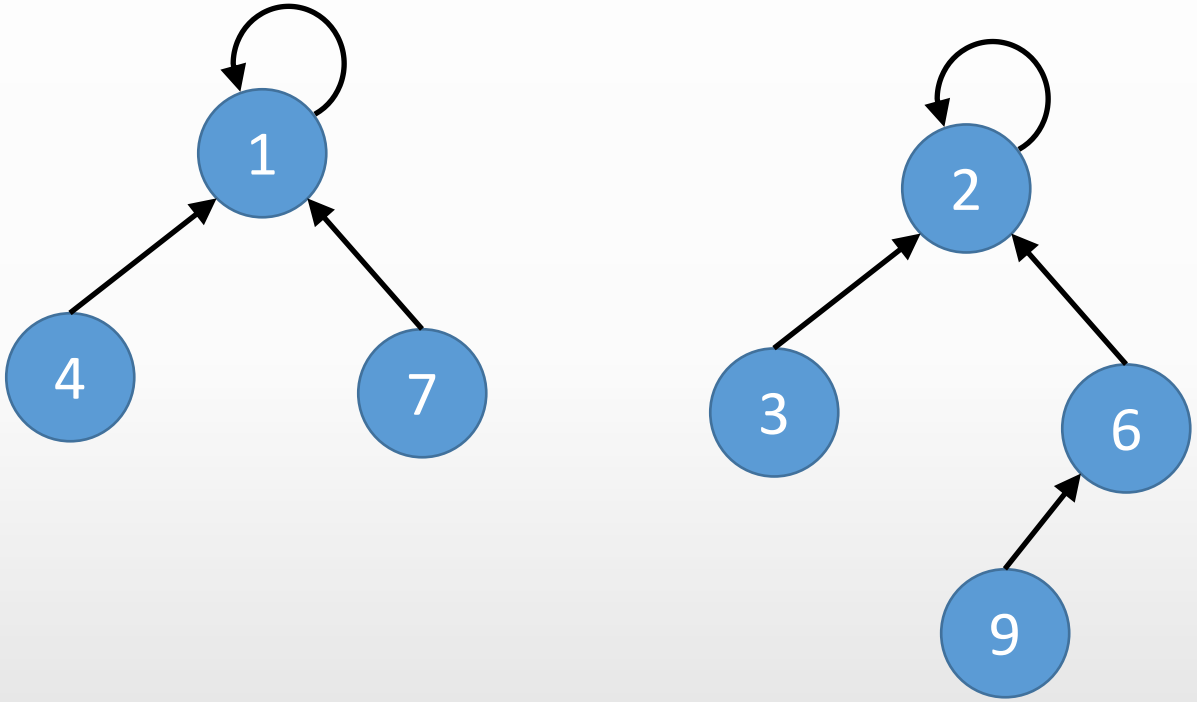
```
MakeSet(x)
    p[x] = x
    rank[x] = 0

Link(x,y)
  if (rank[x] > rank[y])
    p[y] = x
  else
    p[x] = y
    if (rank[x] == rank[y])
      rank[y]++
```

```
FindSet(x)
  if (x == p[x])
    p[x] = FindSet(p[x])
  return p[x]

Union(x,y)
  Link(FindSet(x), FindSet(y))
```

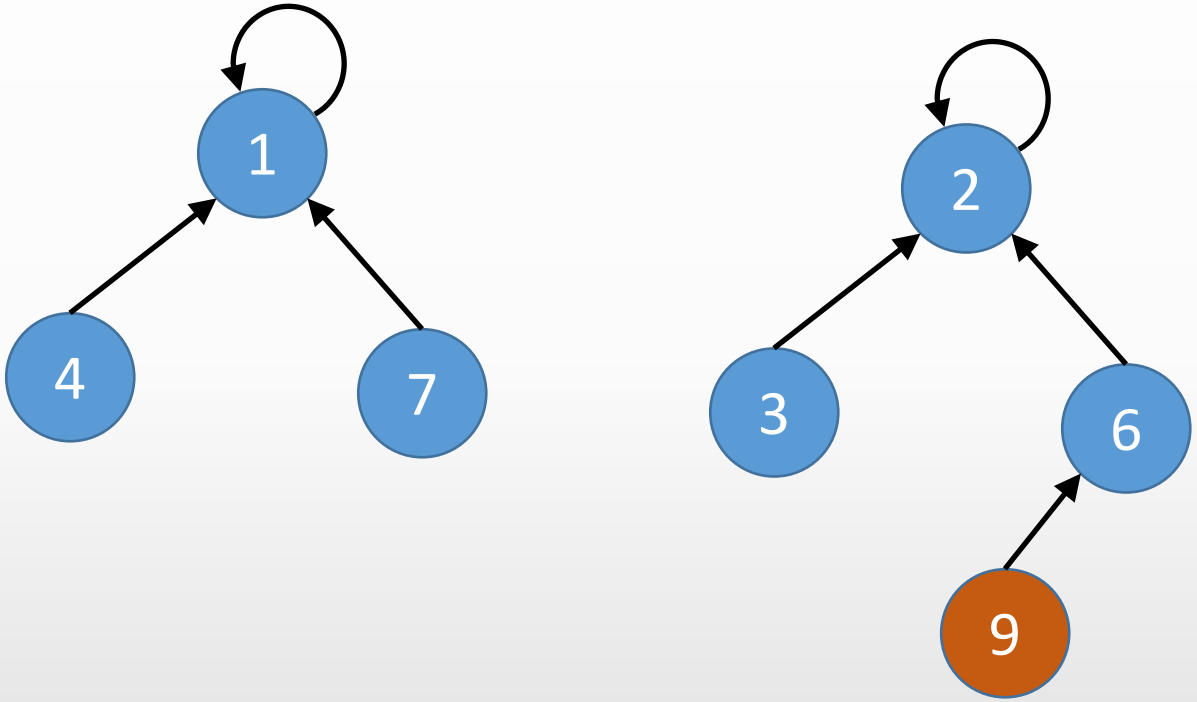
Küme Yapısının Ağaç Temsili



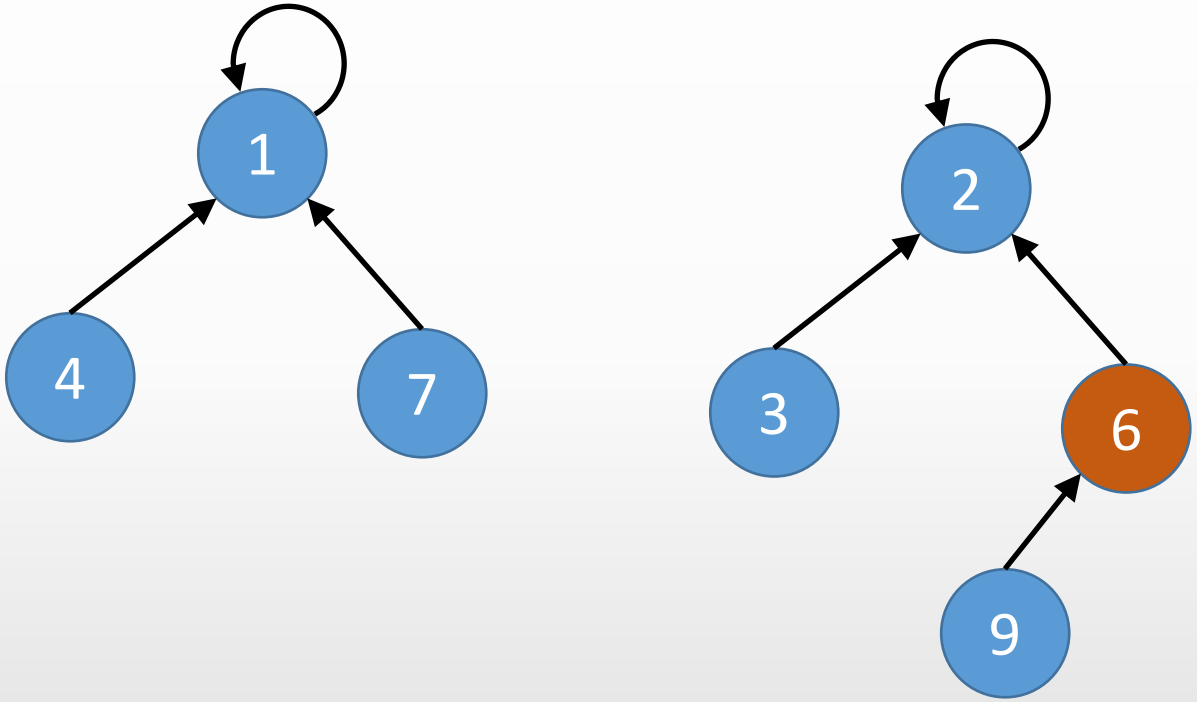
Find İşlemi



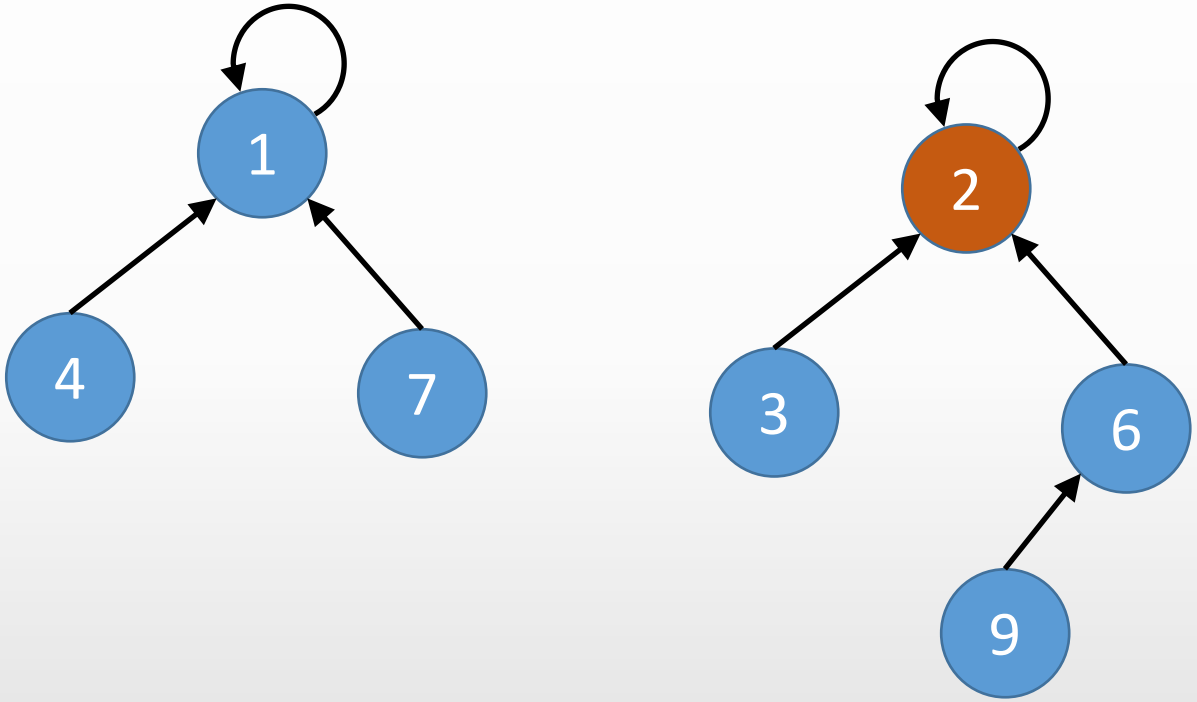
Find İşlemi



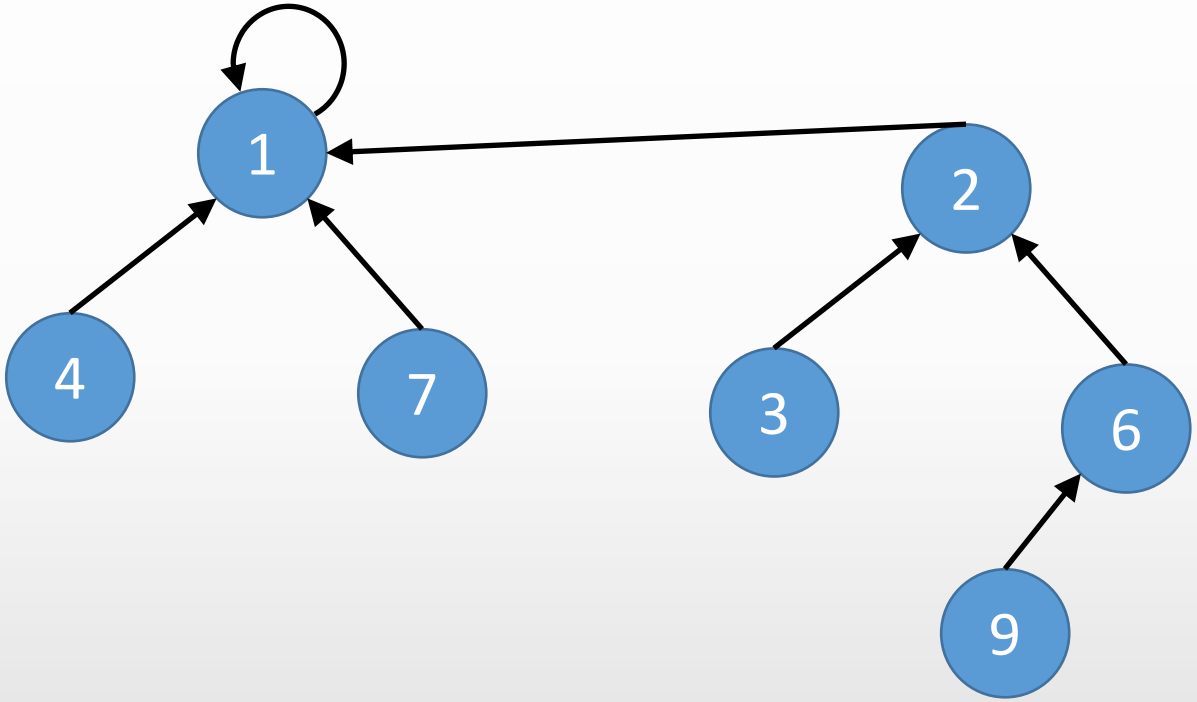
Find İşlemi



Find İşlemi

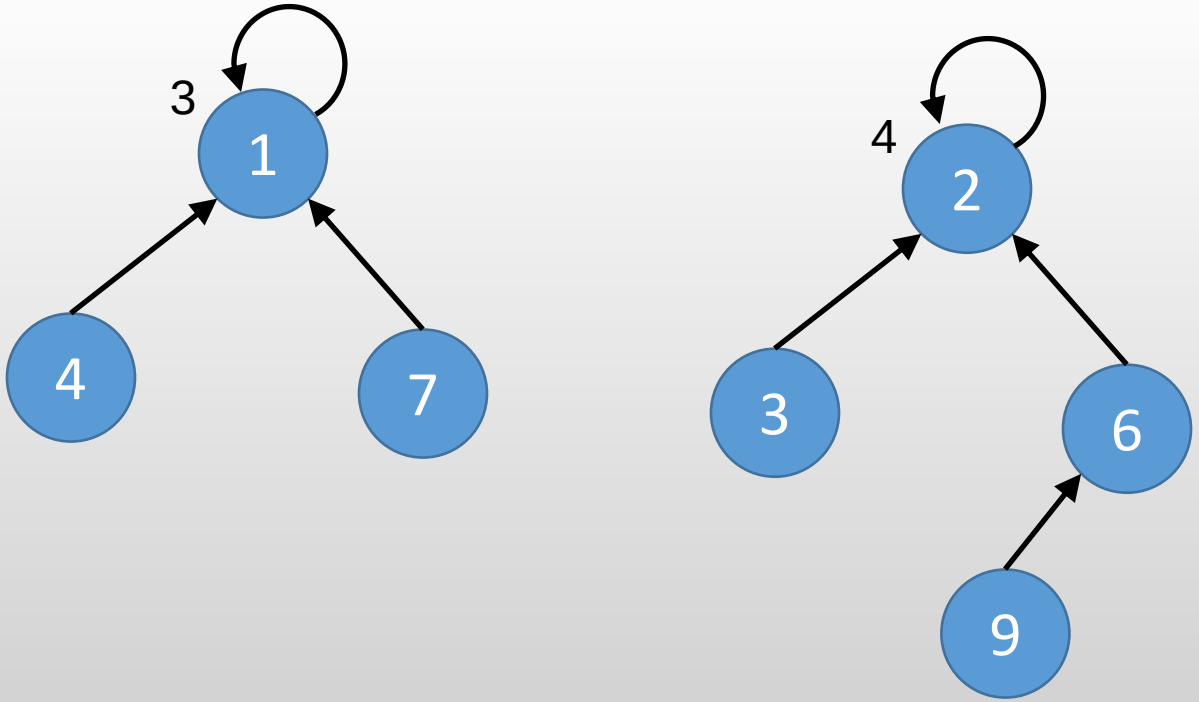


Union İşlemi



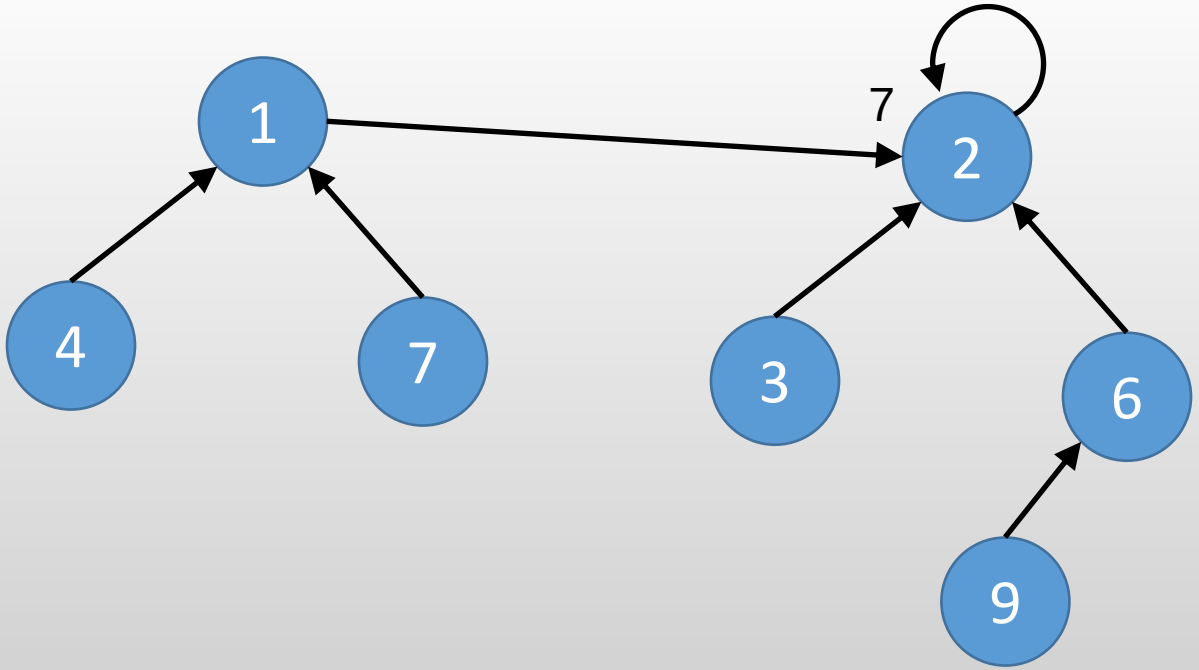
Boyuta Bağlı Union İşlemi

- Ağacın boyutunu sakla
- Büçük ağacın kökünü diğer ağacın da kökü yap.
- Ağacın boyutunu güncelle



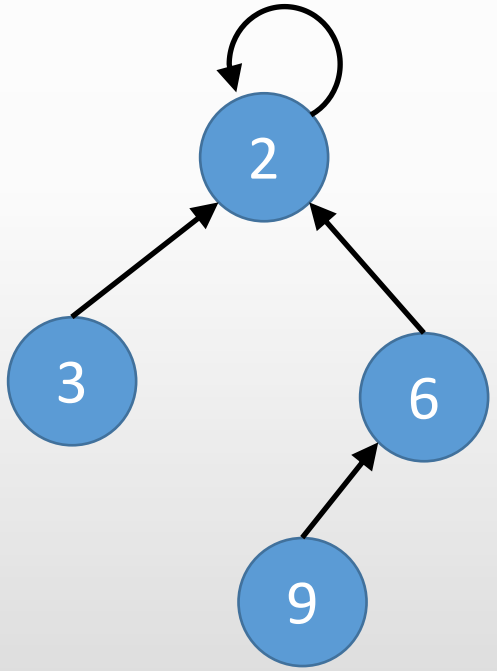
Boyuta Bağlı Union İşlemi

- Ağacın boyutunu sakla
- Büçük ağacın kökünü diğer ağacın da kökü yap.
- Ağacın boyutunu güncelle

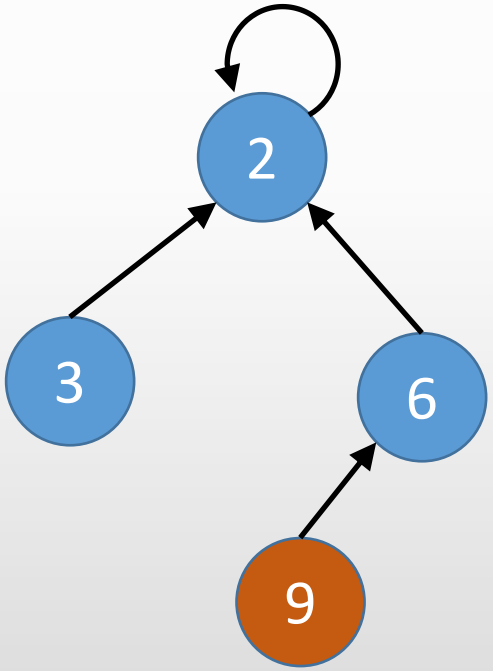


Yol Kısaltma

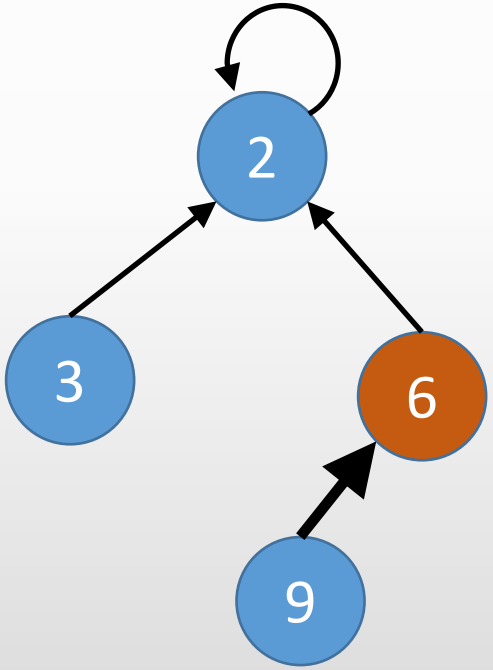
- Find işleminde rastlanan tüm düğümleri kök'e ata



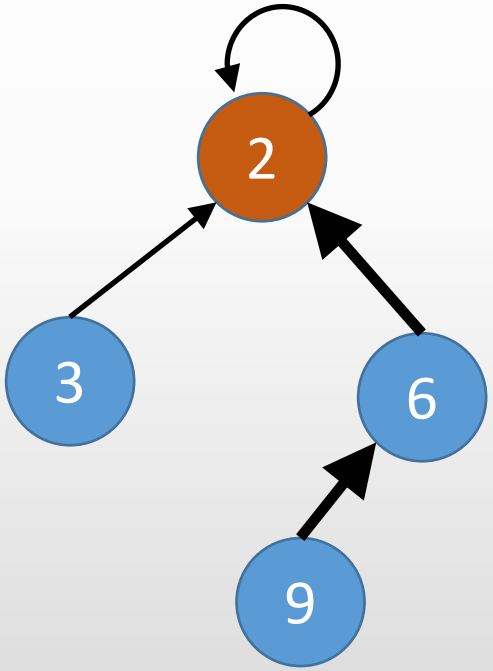
Yol Kısaltma



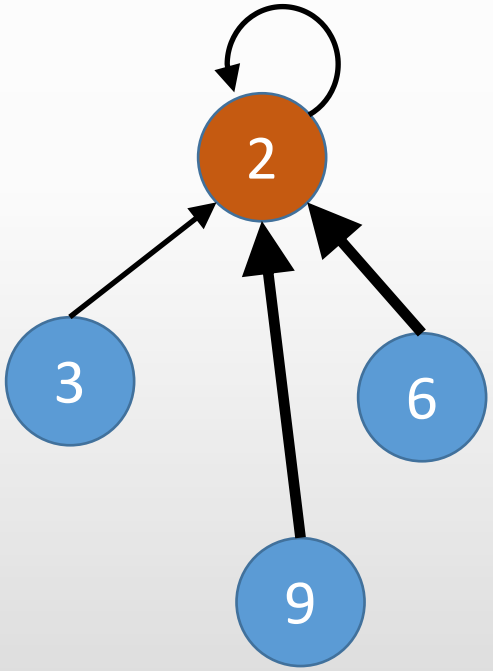
Yol Kısaltma



Yol Kısaltma



Yol Kısaltma



SetDugum Sınıfı

```
class SetDugum {

  int veri;
  int sira;
  SetDugum ata;

  public SetDugum(int veri) {
    this.veri = veri;
    this.ata = this;
    this.sira = 0;
  }
}
```

Find İşlemi

```
SetDugum find(SetDugum node) {

  if (node != node.ata) {
  // yol kısaltma
    node.ata = find(node.ata);
  }
  return node.ata;
}
```

Union İşlemi

```
void union(SetDugum x, SetDugum y) {
  SetDugum kokX = find(x);
  SetDugum kokY = find(y);
  if (kokX != kokY) {// ağaç büyüklüğüne göre birleştirme
    if (kokX.sira < kokY.sira) {
      kokX.ata = kokY;
    } else if (kokX.sira > kokY.sira) {
      kokY.ata = kokX;
    } else { // siralar eşitse birini seç
      kokY.ata = kokX;
      kokX.sira++;
    }
  }
}
```

Find İşlemi

```
int find(int x) {

  if (ata[x] != x) {
  // yol kısaltma
    ata[x] = find(ata[x]);
  }
  return ata[x];
}
```

Union İşlemi

```
void union(int x, int y) {
  int kokX = find(x);
  int kokY = find(y);
  if (kokX != kokY) { // ağaç büyüklüğüne göre birleştirme
    if (sira[kokX] < sira[kokY]) {
      ata[kokX] = kokY;
    } else if (sira[kokX] > sira[kokY]) {
      ata[kokY] = kokX;
    } else { // siralar eşitse birini seç
      ata[kokX] = kokY;
      sira[kokY]++;
    }
  }
}
```

SON