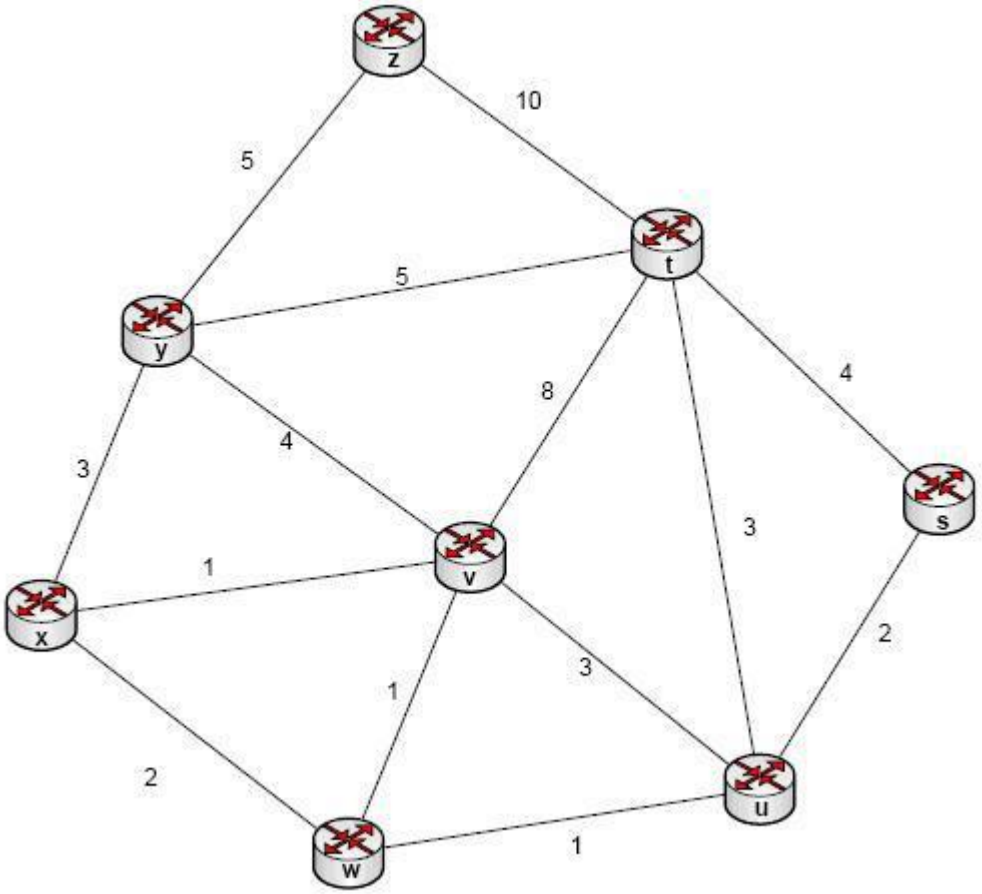
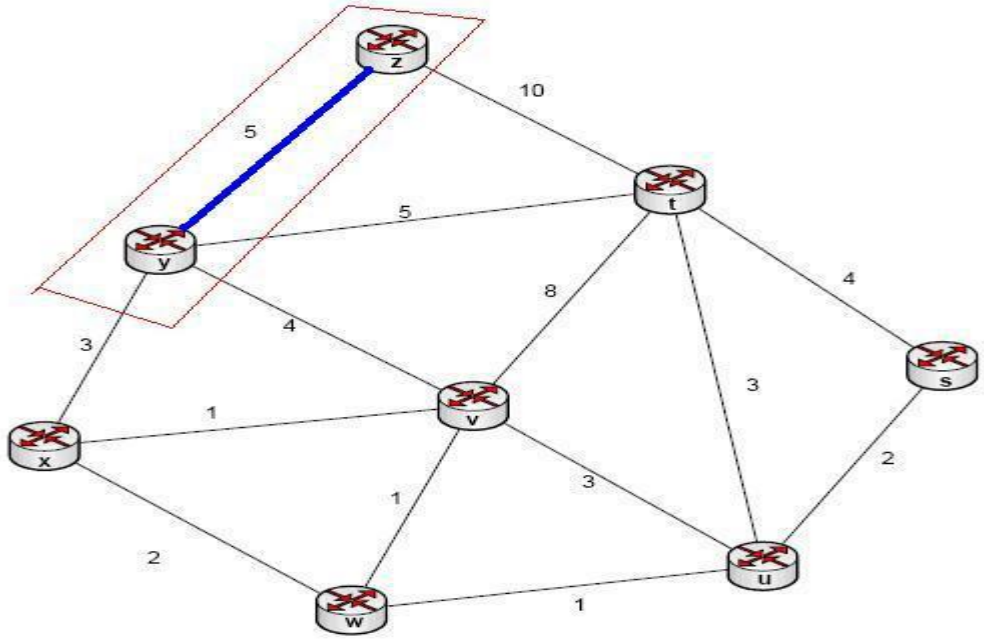
Prim Algoritmasi:

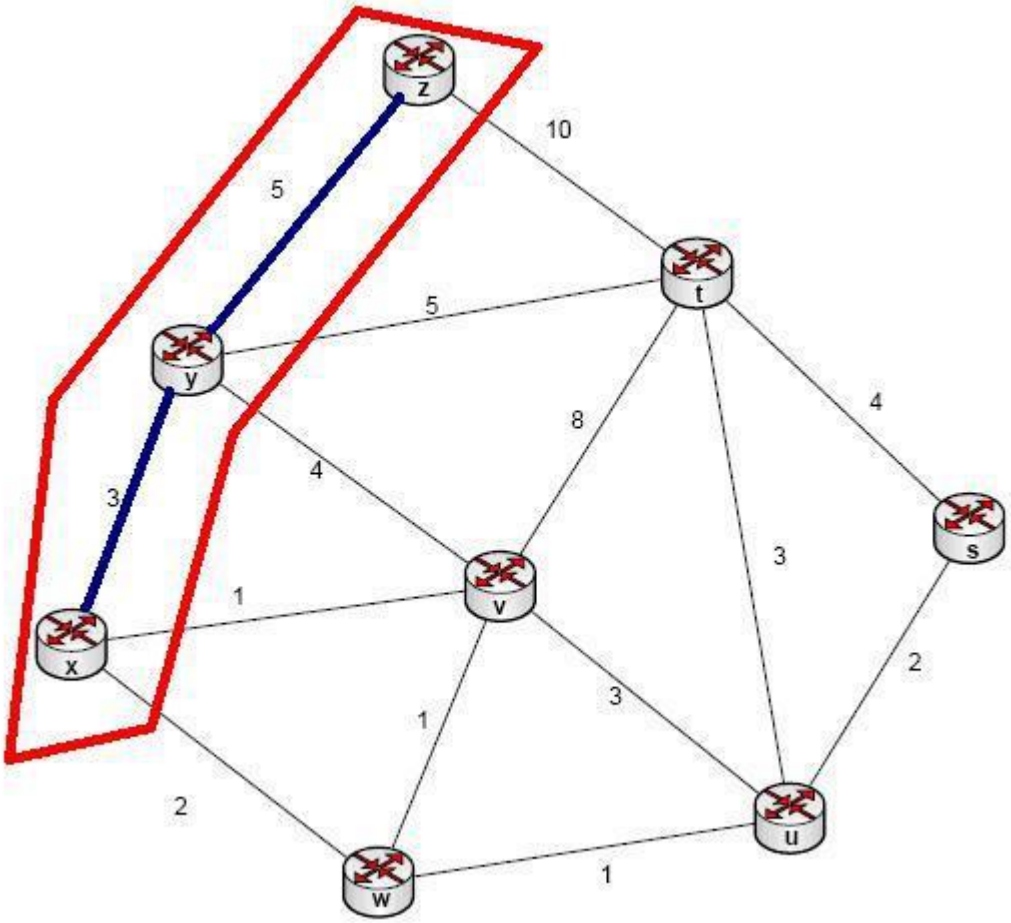
Prim Algoritması, minimum spanning tree algoritmalarından biridir. Kenarların bir alt kümesini, tüm düğümleri kapsayacak ve kenarların toplam ağırlığını minimum yapacak şekilde bulur.

Bir resim üzerinde anlatacak olursak.Resimdeki gibi grafımız olsun.Şimdi Z'den başlayalım.Y'ye 5 ve T'ye 10 birim. En küçük kenar Y olduğu için seçiyoruz.

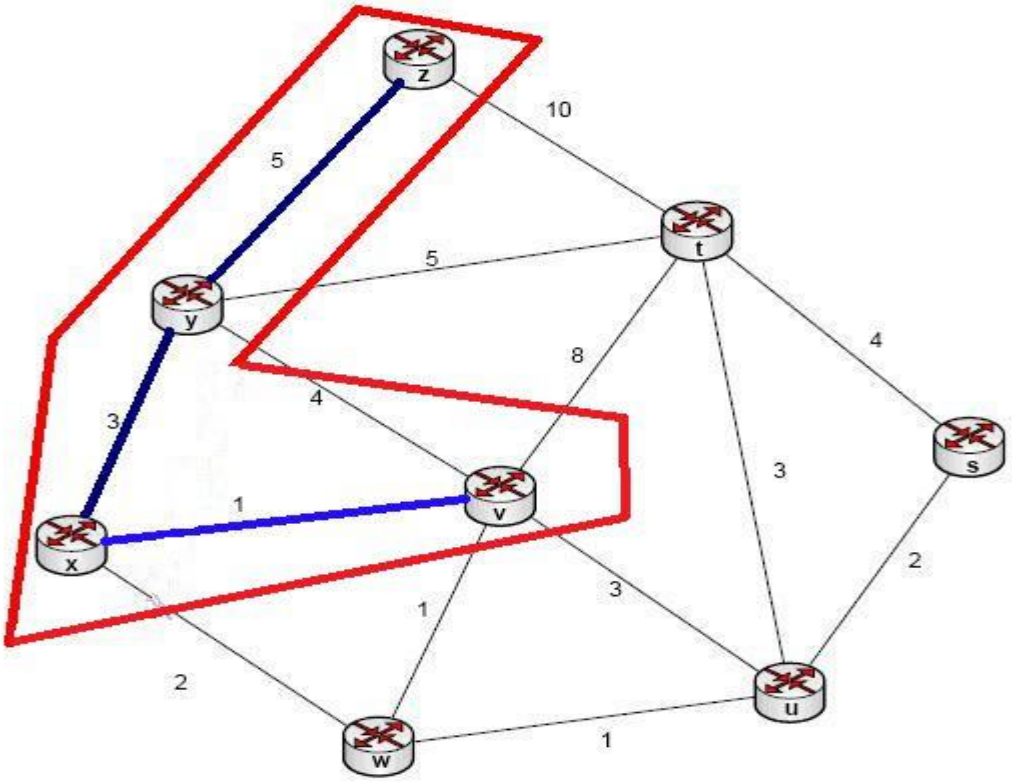


Şimdi Y ve Z deki kenarlardan en küçük olanı bulalım. Yden x'e 3 V'ye 4 ve T'ye 5 birim Z'den t ye 10 en küçük kenar X olduğundan x'i seçiyoruz seçiyoruz.

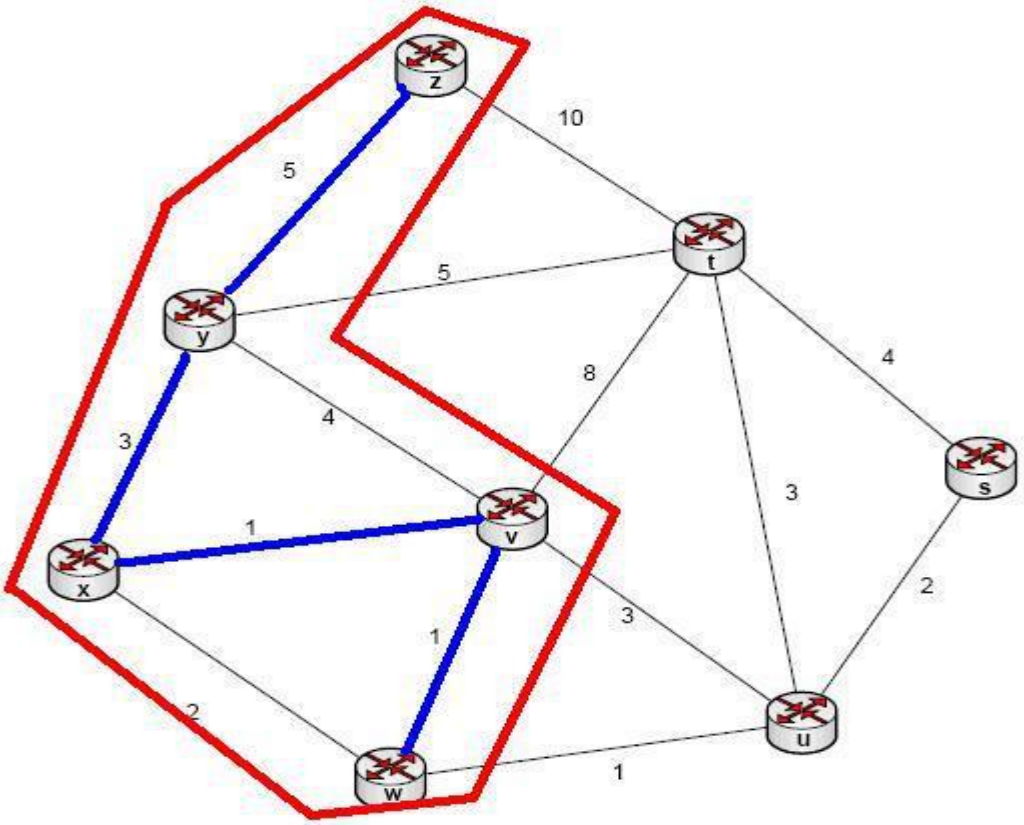

z
t
y
s
v
x
u
w
10
5
5
8
4
4
3
3
1
2
1
3
2
1

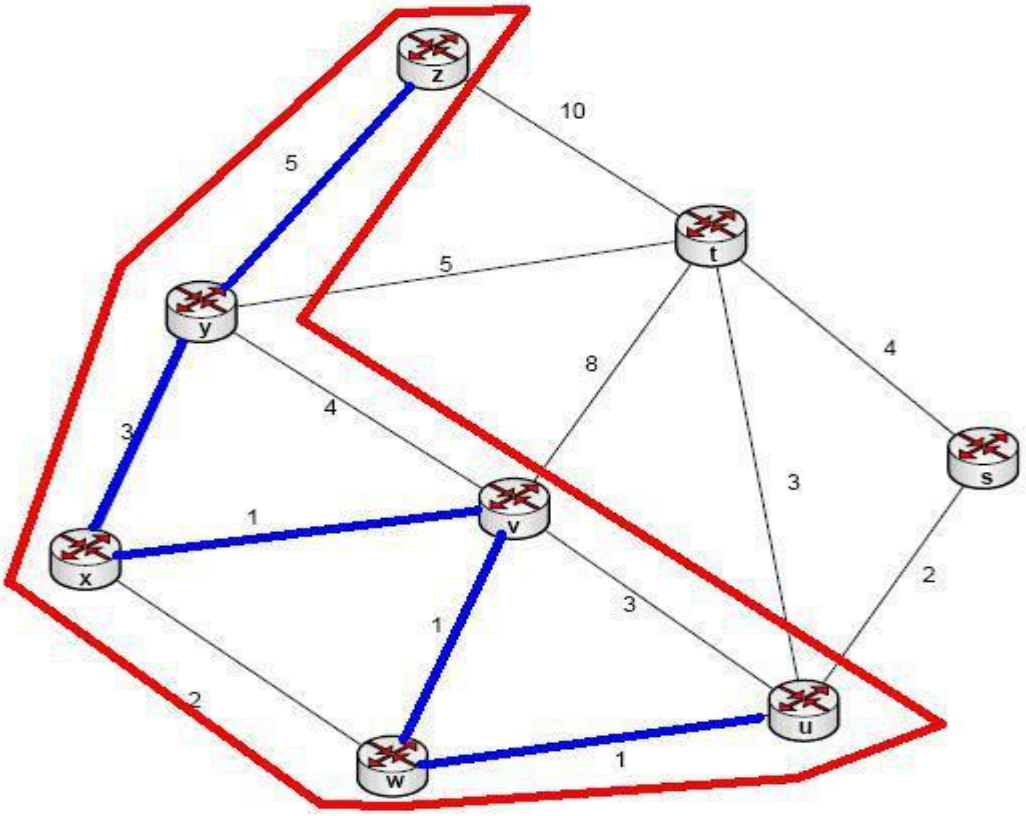

z
t
y
s
v
x
u
w
10
5
5
8
4
4
3
3
1
2
1
3
2
1

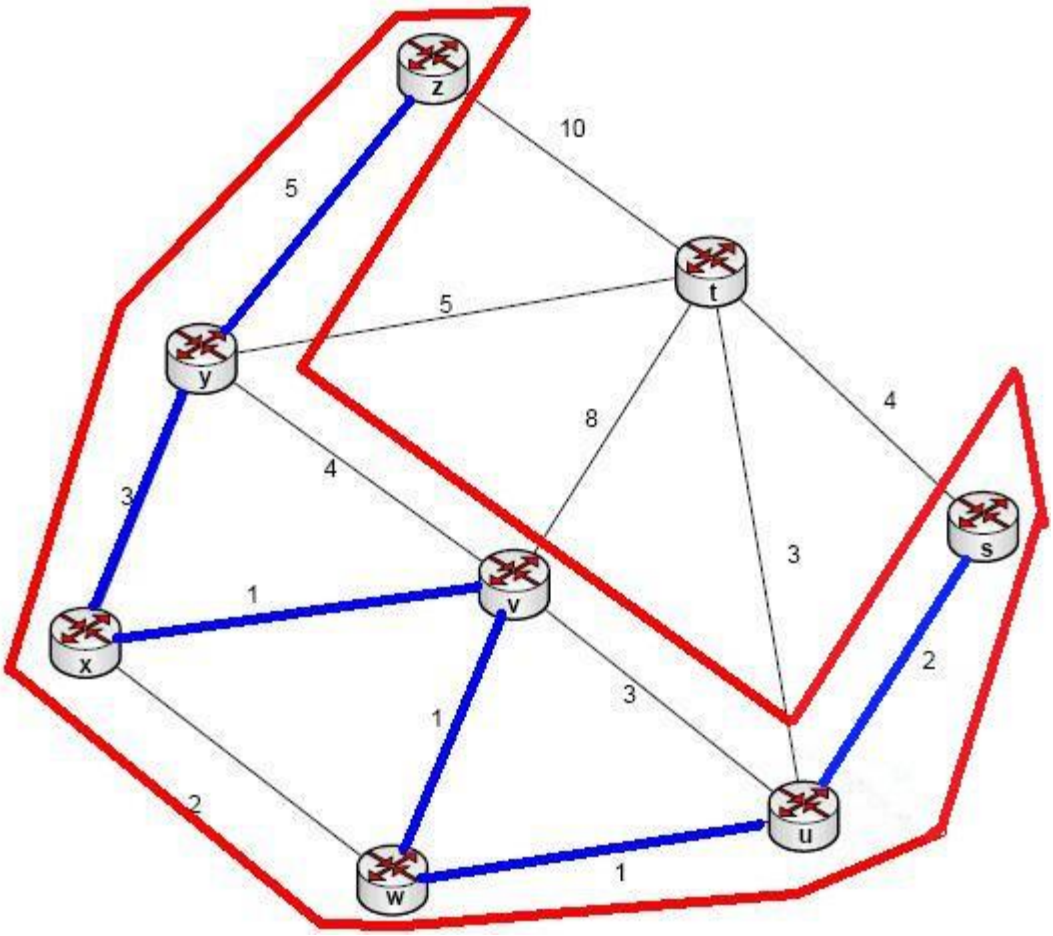
Şimdi ise Z Y ve Xteki Komşu kenarlardan en küçük olanına bakıyoruz.X'ten Vye 1 birim olduğunu görüyoruz. Hep aynı işlem. Amaç seçilen graflarda ki en küçük kenarı bulmak .

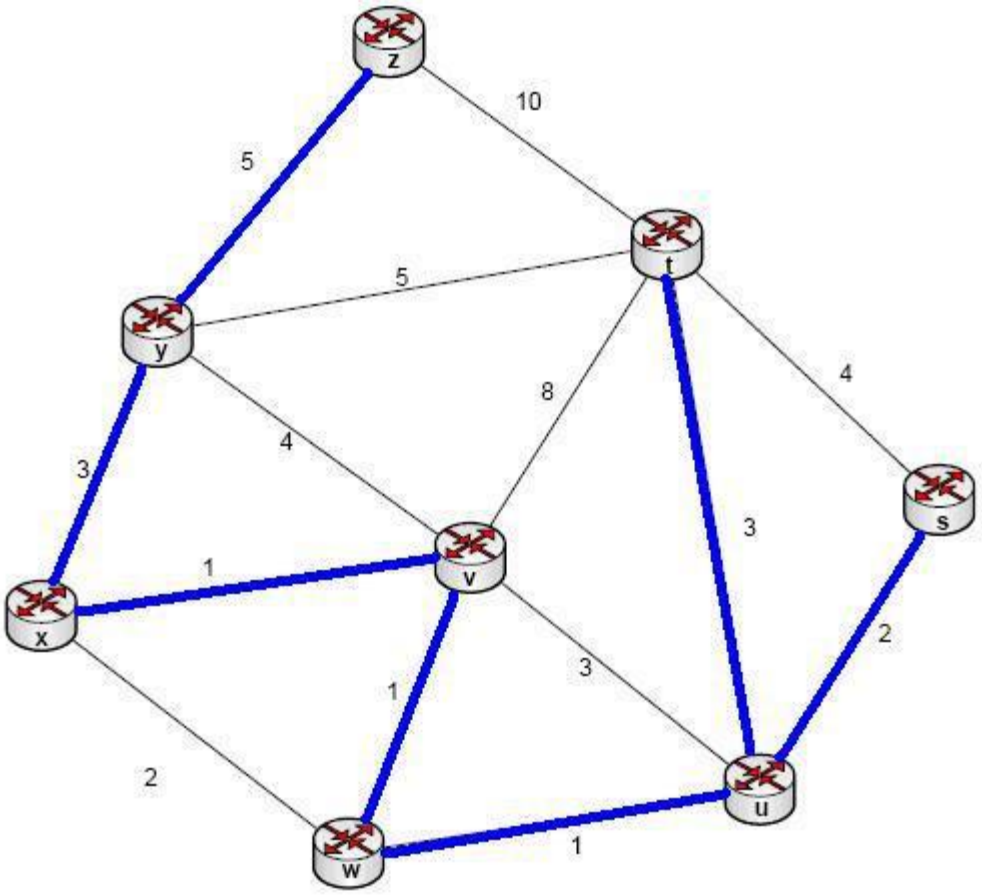


Yine aynı Taktik.V den bu sefer en küçük kenarı 1 olan W'ye


z
10
5
t
5
y
8
4
4
3
s
3
v
1
x
3
2
1
2
u
w
1


z
10
5
t
5
y
8
4
4
3
s
3
v
1
x
3
2
1
2
u
w
1


z
t
y
s
v
x
w
u
10
5
5
8
4
4
3
3
1
3
1
2
2
1



En küçük kenarlar bunlarmış. Arada kalanlardan biri seçilirse bu prim algoritması olmaz. Kruskal alg. ile prim yöntemler farklı olsada sonuçlar her zaman tüm düğümleri kapsayacak ve kenarların toplam ağırlığını minimum yapar.

Algoritma python dilinde:

```python
# prim algoritmasi kullanark minimum agirligi bulayacak algoritma
def prim_algoritma(graph):
    # Prim's Algorithm in Python
    INF = 9999999
    # grafiğin dikeylerin sayısı
    N = len(graph)
    selected_node = [0 for i in range(N)]
    # loop sonlandirabilmek icin bir degisken tanimliyoruz
    no_edge = 0
    sum = 0

    selected_node[0] = True

    while (no_edge < N - 1):

        minimum = INF
        a = 0
        b = 0
        for m in range(N):
            if selected_node[m]:
                for n in range(N):
                    if ((not selected_node[n]) and graph[m][n]):
                        # seçilmemiş ve kenar var
                        if minimum > graph[m][n]:
                            minimum = graph[m][n]
                            a = m
                            b = n
        sum += graph[a][b]
        print(
            f"({a}, {b}) konumndaki sayi: {graph[a][b]}, {graph[a]} kumenin
minimum agirligi")
        selected_node[b] = True
        no_edge += 1
    # agirligin toplami yazdiriyoruz
    print(f"Toplam: {sum}")


graph = [[8, 19, 5, 9, 10],
         [19, 0, 5, 9, 2],
         [5, 5, 0, 1, 6],
         [0, 9, 1, 0, 1],
         [0, 2, 6, 1, 0]]
prim_algoritma(graph=graph)
```

Çalışma zamanı T(N):

$$T(N) = \sum_{i=0}^{N-1} \left[\sum_{m=0}^{N} \left[\sum_{n=0}^{N} 2 \right] \right] = 2N^2(N-1) = 2N^3 - 2N^2$$

Big O:

$$T(N) = 2N^2(N-1) = 2N^3 - 2N^2$$

$$O(N^3) => T(N) \leq C * N3, N \geq N0$$

Big Omega:

$$T(N) = 2N^2(N-1) = 2N^3 - 2N^2$$

$$\Omega(N^3) => T(N) \geq C1 * N3, N \geq N0$$

Big Theta:

$$T(N) = 2N^2(N-1) = 2N^3 - 2N^2$$

$$\Theta(N^3) => \Omega(N^3) \leq \Theta(N^3) \leq O(N^3)$$