T.C.
KIRIKKALE
ÜNİVERSİTESİ
BİLGİSAYAR
MÜHENDİSLİĞİ

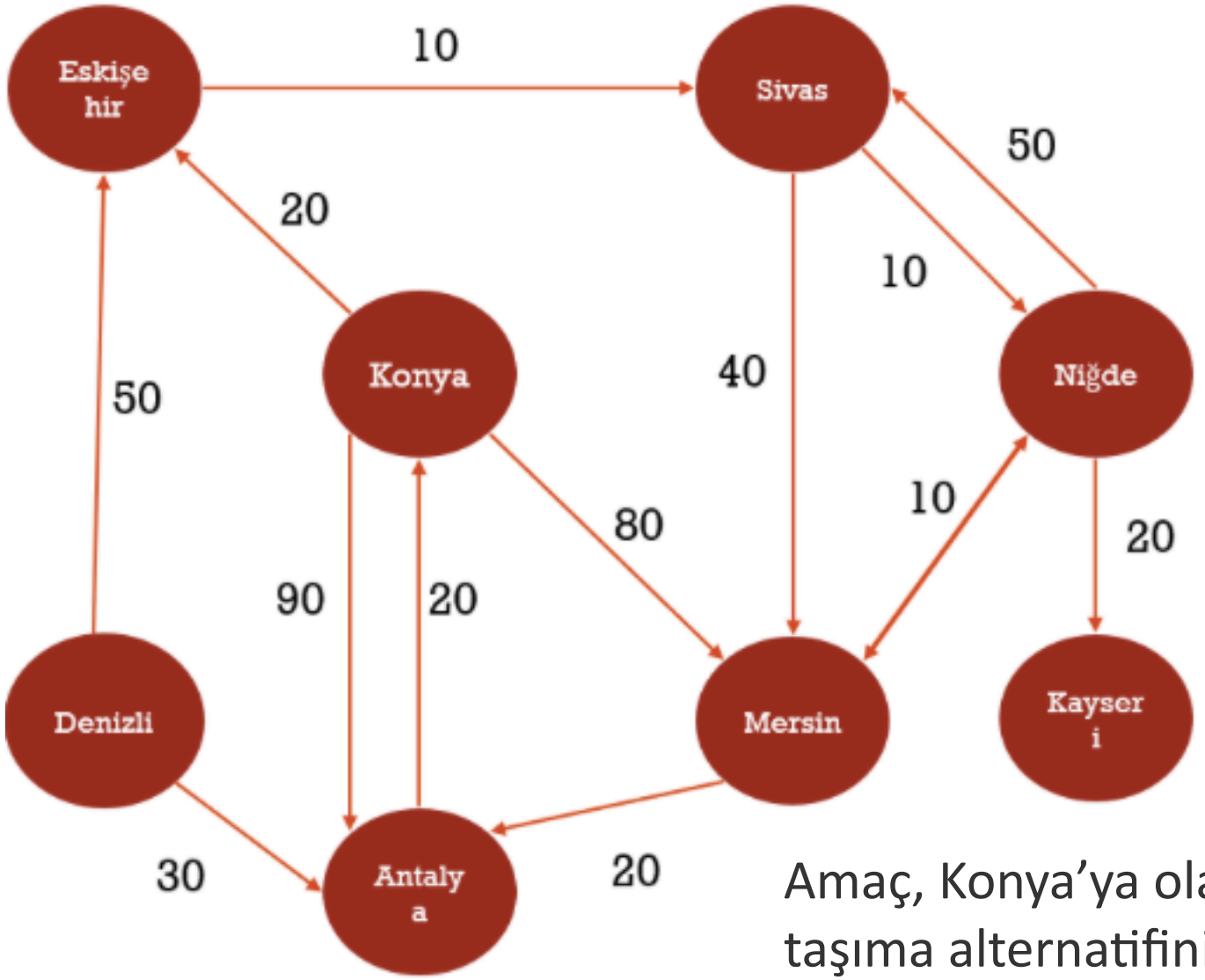
AĞ OPTİMİZASYONU

DR. EVRENCAN ÖZCAN

DERS İÇERİĞİ

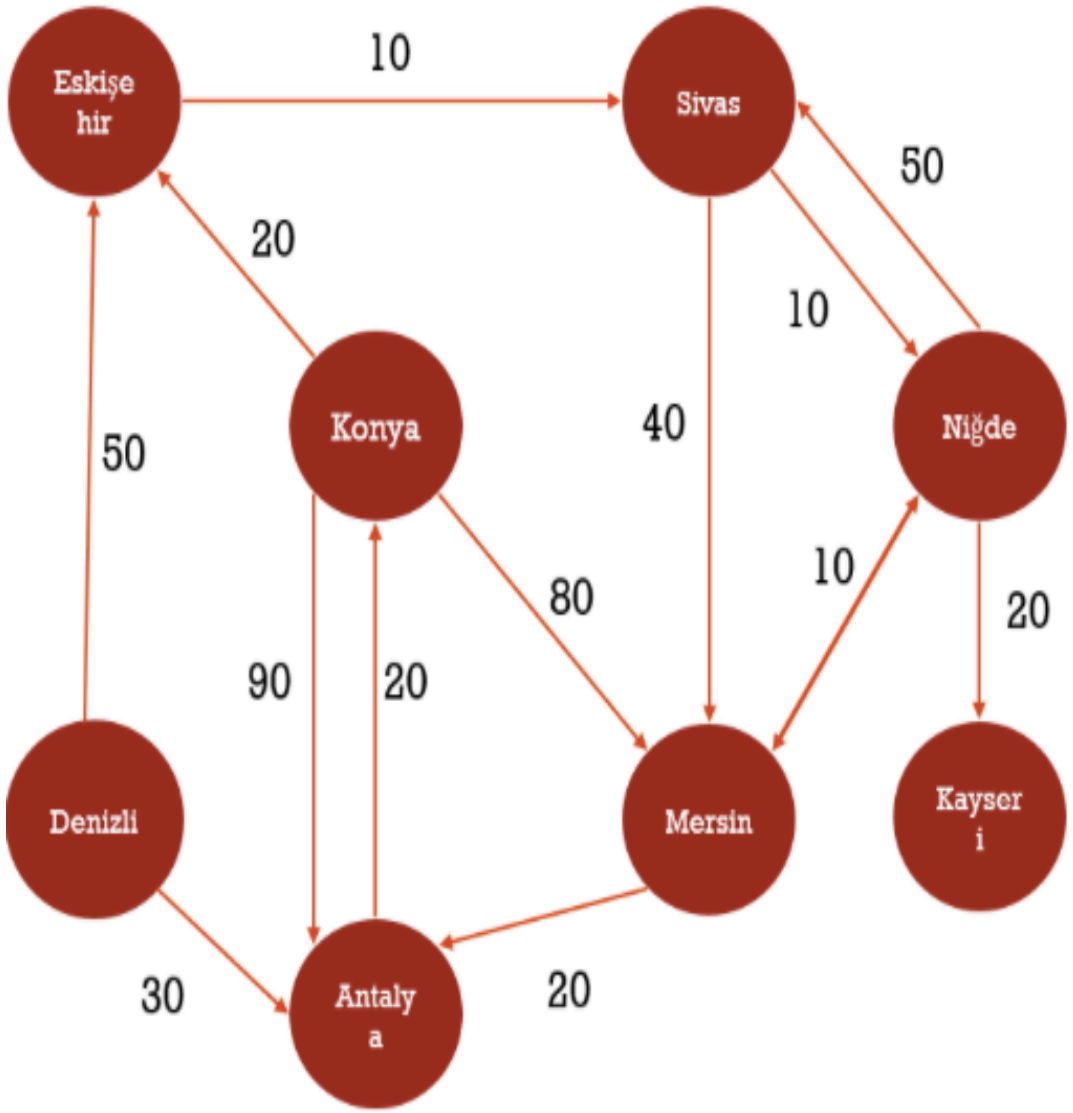
- AĞ OPTİMİZASYONUNA GİRİŞ
  - OPTİMİZASYON KAVRAMI
  - TEMEL ŞEBEKE KAVRAMLARI
  - ŞEBEKE OPTİMİZASYONUNUN UYGULAMA ALANLARI
- MİNİMUM YAYILAN AĞAÇ PROBLEMİ
- EN KISA YOL PROBLEMİ
- MAKSİMUM AKIŞ PROBLEMİ
- PROJE YÖNETİMİ
  - KRİTİK YOL METODU (CPM)
  - PROJE DEĞERLENDİRME VE GÖZDEN GEÇİRME TEKNİĞİ (PERT)
  - PROJE PLANLAMASINDA ZAMAN-MALİYET İLİŞKİSİ

EN KISA YOL PROBLEMİ – DIJKSTRA ALGORİTMASI



Başlangıç Konya

Amaç, Konya'ya olan en düşük maliyetli taşıma alternatifini şehir bazlı bulmak



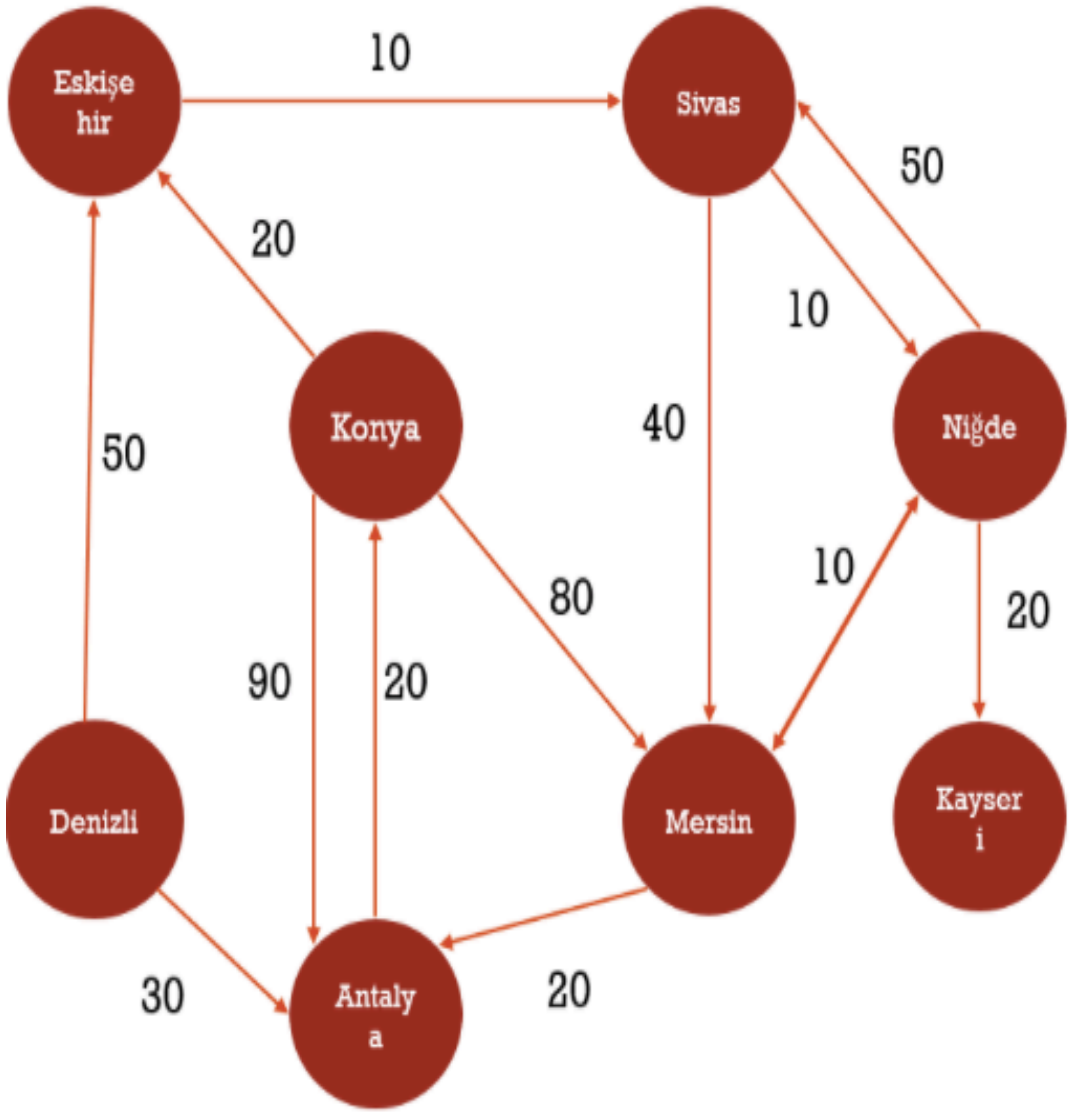
1. Adım:

Başlangıç düğümü olarak Konya'yı seçtiğimize göre Konya'nın komşularının maliyetlerini güncelleriz.

Düğüm	Konya	Eskişehir	Niğde	Mersin	Denizli	Sivas	Antalya	Kayseri
Konya	0	20 Konya	∞	80 Konya	∞	∞	90 Konya	∞

Konya'nın komşuları Eskişehir, Mersin ve Antalya'dır. (okların yönüne dikkat edin) Bu düğümlere olan maliyetler güncellenmiştir. Diğer tüm düğümler hala sonsuz maliyettedir. Konya düğümü başlangıç düğümü olduğu için sıfır'dır, hep sıfır olarak kalacaktır.

Dikkat! Maliyetlerin altında yazan düğüm isimleri hangi düğümden geldiği bilgisini tutmak içindir. Kodlarken parent şeklinde kullanacağız.

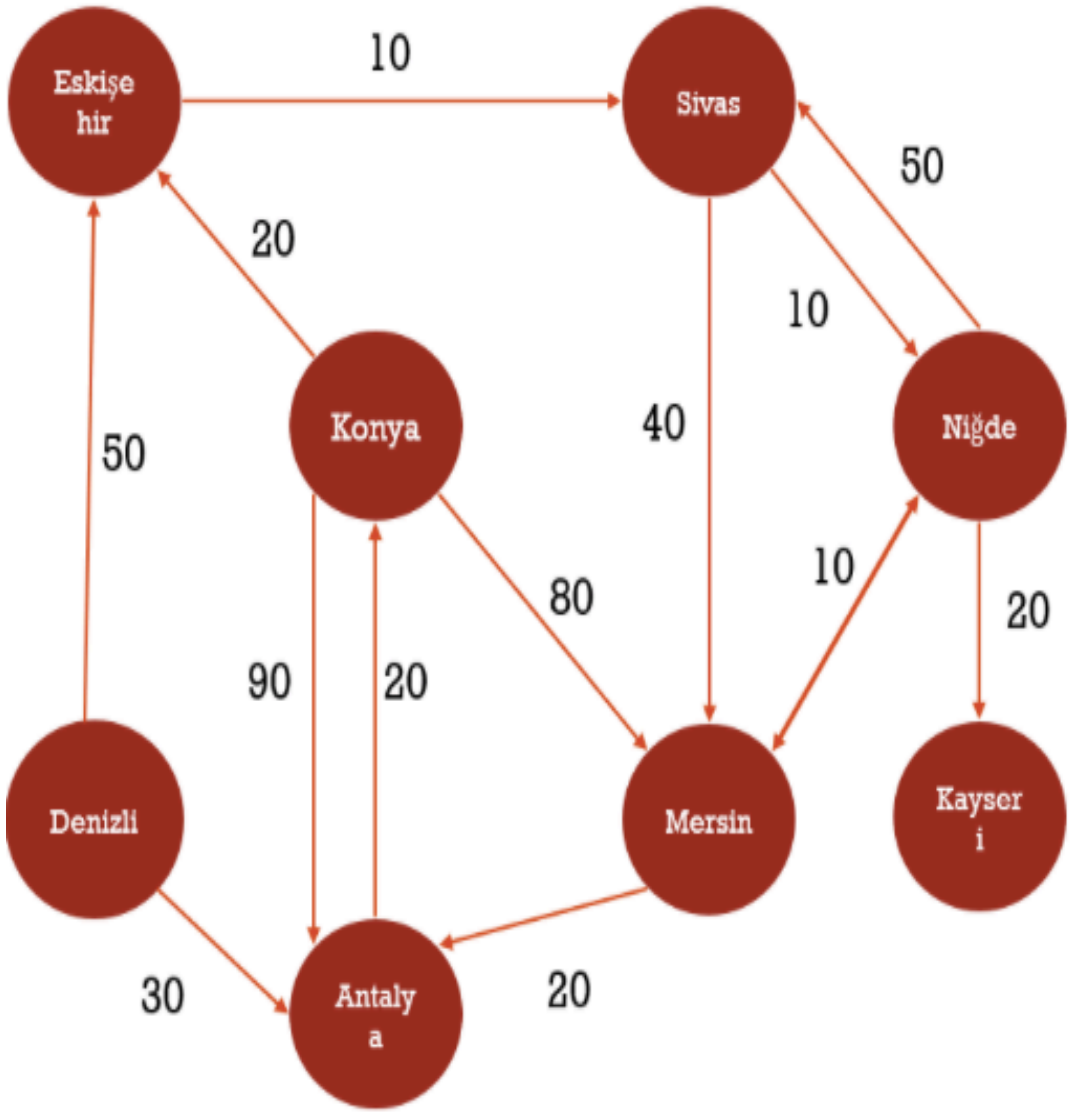


2. Adım:

Bu adımda 1. satırdaki en az maliyetli düğümü seçeriz. Ancak şuna her zaman dikkat edeceğiz, seçeceğimiz düğüm asla ziyaret edilmiş düğüm OLMAMALI. En az maliyetli düğüm Eskişehir olduğu için Eskişehir'e gidiyoruz.

Düğüm	Konya	Eskişehir	Niğde	Mersin	Denizli	Sivas	Antalya	Kayseri
Konya	0	20 Konya	∞	80 Konya	∞	∞	90 Konya	∞
Eskişehir 20	0	20 Konya	∞		∞	30 Eskişe	90 Konya	∞

Eskişehir'den gidilebilecek tek bir düğüm var, o da Sivas (okların yönüne dikkat). Ancak Sivas'a maliyet 10'du neden 30 yazdık? Çünkü zaten Eskişehir'e gidişin maliyeti 20, Sivas'a gidiş de 10, toplam 30 olarak güncelleriz. (başlangıç noktamızın Konya olduğunu unutmayın, başlangıç noktası çok önemli)

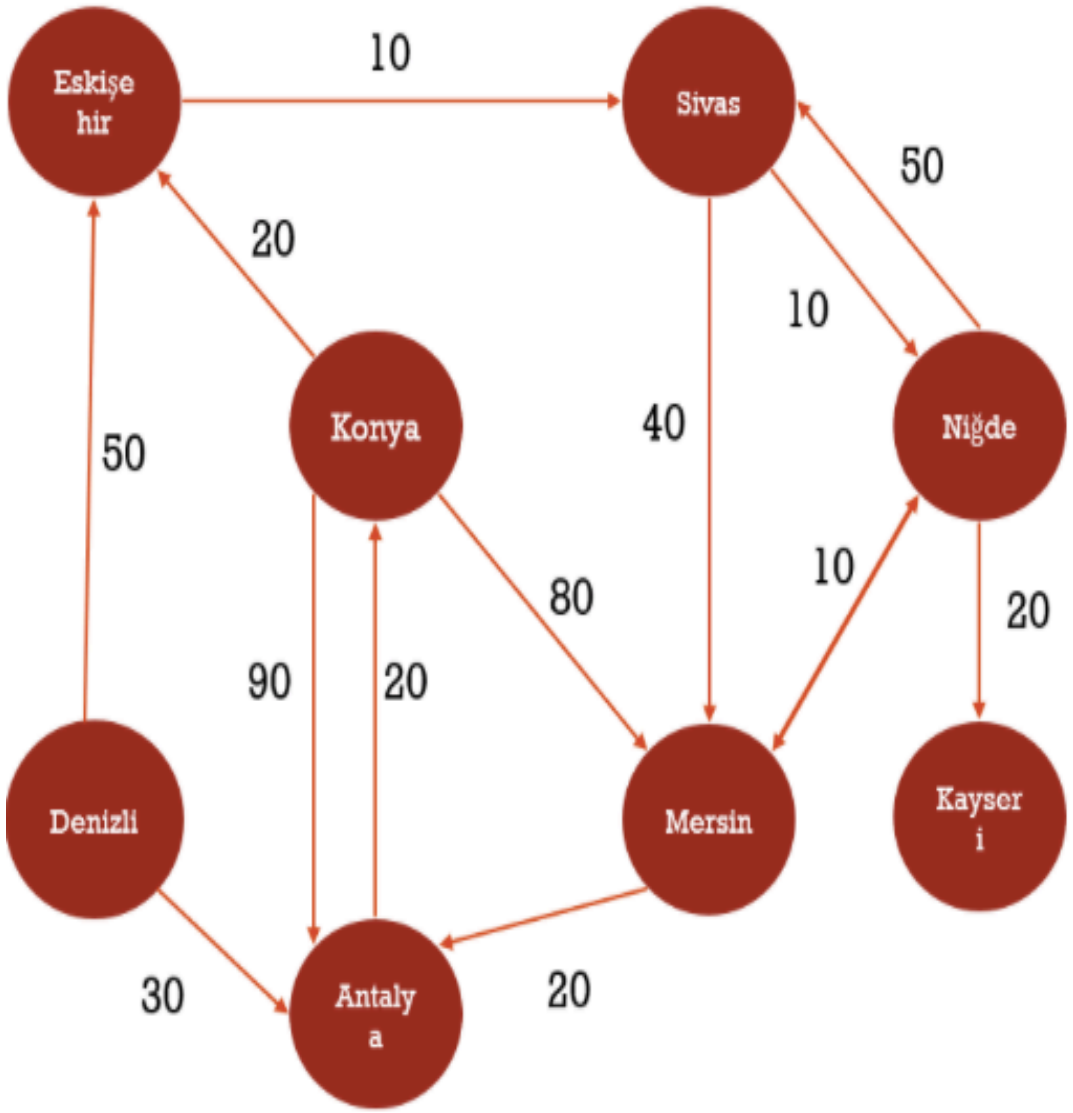


3. Adım:

Yine en düşük maliyetli düğümü seçeceğiz, en az maliyetli düğümler Konya ve Eskişehir aslında ama biz bu iki düğümü de ziyaret ettik. (düğüm sütununda ismi varsa ziyaret etmişiz demektir) bu yüzden bu düğümlerin haricindeki en düşük maliyetli düğüme gideriz. O da Sivas.

Düğüm	Konya	Eskişehir	Niğde	Mersin	Denizli	Sivas	Antalya	Kayseri
Konya	0	20 Konya	∞	80 Konya	∞	∞	90 Konya	∞
Eskişehir 20	0	20 Konya	∞	80 Konya	∞	30 Eskişe	90 Konya	∞
Sivas 30	0	20 Konya	40 Sivas	70 Sivas	∞	30 Eskişe	90 Konya	∞

Sivas'tan çıkan iki ok var, Bunlar Niğde ve Mersin. Niğde'ye Sivas'tan maliyet 10, ama Sivas'ın da 30 maliyeti var. Bu yüzden maliyeti 40 olarak güncelleriz. Mersin'e olan maliyet ise 40, Sivas'ın maliyetini de ekleyince 70 oluyor. Önceki maliyet ise 80'di, 70 sayısı 80'den küçük olduğu için Mersin'in maliyetini güncelleriz. İşte dijkstra algoritmasının esas olayını burada net olarak görüyoruz. Daha kısa bir yol bulmuş olduk.



4. Adım:

Bu adımda yine en düşük maliyetli, ziyaret edilmemiş düğümü seçiyoruz (3. satırdaki), o da Niğde (40).

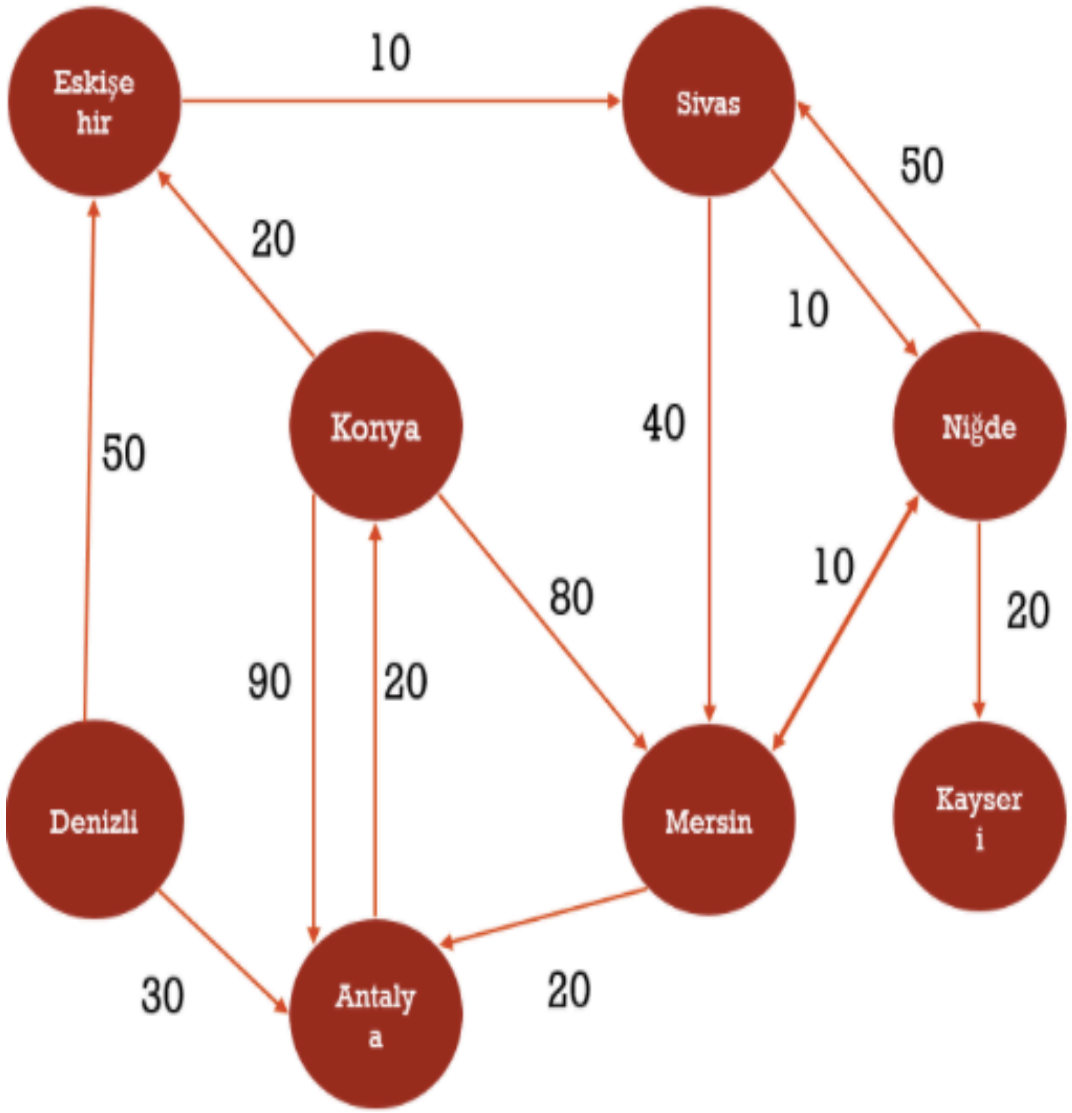
Düğüm	Konya	Eskişehir	Niğde	Mersin	Denizli	Sivas	Antalya	Kayseri
Konya	0	20 Konya	∞	80 Konya	∞	∞	90 Konya	∞
Eskişehir 20	0	20 Konya	∞	80 Konya	∞	30 Eskişe	90 Konya	∞
Sivas 30	0	20 Konya	40 Sivas	70 Sivas	∞	30 Eskişe	90 Konya	∞
Niğde 40	0	20 Konya	40 Sivas	50 Niğde	∞	30 Eskişe	90 Konya	60 Niğde

Niğde düğümünün Sivas, Mersin ve Kayseri düğümlerine komşuluğu var. Sivas düğümüne gidiş, Sivas'ın maliyetini artırıyor (zaten oraya geldik) bu yüzden Sivas'ta güncelleme yapmıyoruz.

Eğer Negatif maliyetli kenarlar olsaydı iş burada değişirdi, ancak dijkstra algoritması negatif kenarları hesaplamaz. (Hata verir, loop'a girer)

Geriye Kayseri ve Mersin kalıyor. Niğde Kayseri arası 20, Niğde'nin maliyeti de 40, bu yüzden 60 olacak şekilde güncelliyoruz.

Mersin'e maliyet ise 10, Niğde'nin maliyeti ise 40, yani Mersin'in Maliyeti 50 oldu, önceki maliyet 70 olduğu için 50 şeklinde güncelliyoruz.

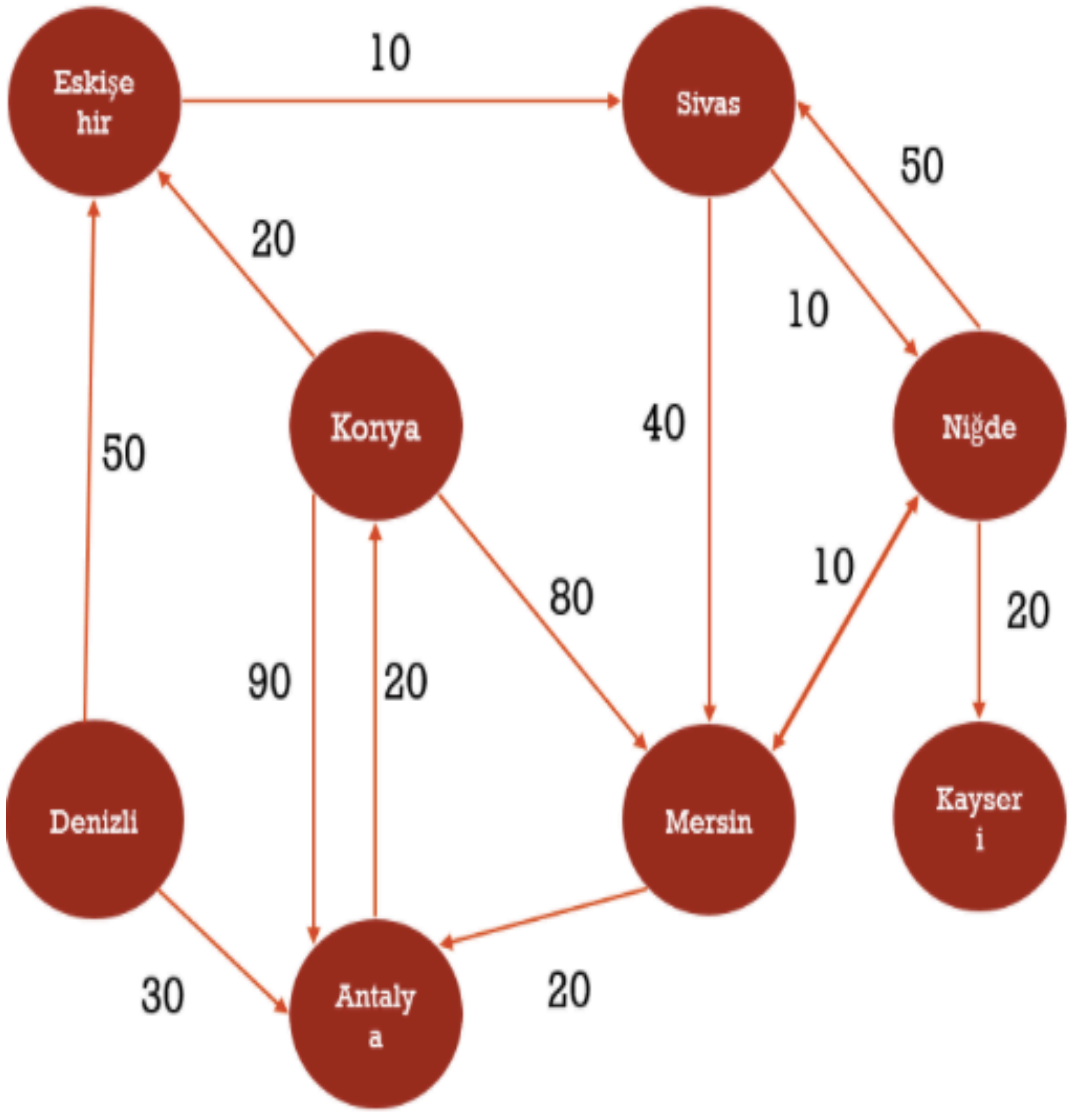


5. Adım:

Bu adımda en düşük maliyetli düğümü seçeceğiz, en düşük maliyetli düğüm Mersin. (Mersin ziyaret edilmedi çünkü)

Düğüm	Konya	Eskişehir	Niğde	Mersin	Denizli	Sivas	Antalya	Kayseri
Konya	0	20 Konya	∞	80 Konya	∞	∞	90 Konya	∞
Eskişehir 20	0	20 Konya	∞	80 Konya	∞	30 Eskişe	90 Konya	∞
Sivas 30	0	20 Konya	40 Sivas	70 Sivas	∞	30 Eskişe	90 Konya	∞
Niğde 40	0	20 Konya	40 Sivas	50 Niğde	∞	30 Eskişe	90 Konya	60 Niğde
Mersin 50	0	20 Konya	40 Sivas	50 Niğde	∞	30 Eskişe	70 Antalya	60 Niğde

Mersin'den bir tek Antalya'ya gidebiliyoruz. Mersin'e toplam maliyet 50, Mersin Antalya arası ise 20 maliyete sahip. Toplam 70 oluyor. Antalya'nın önceki maliyeti 90'dı bu yüzden 70 olarak güncelliyoruz.



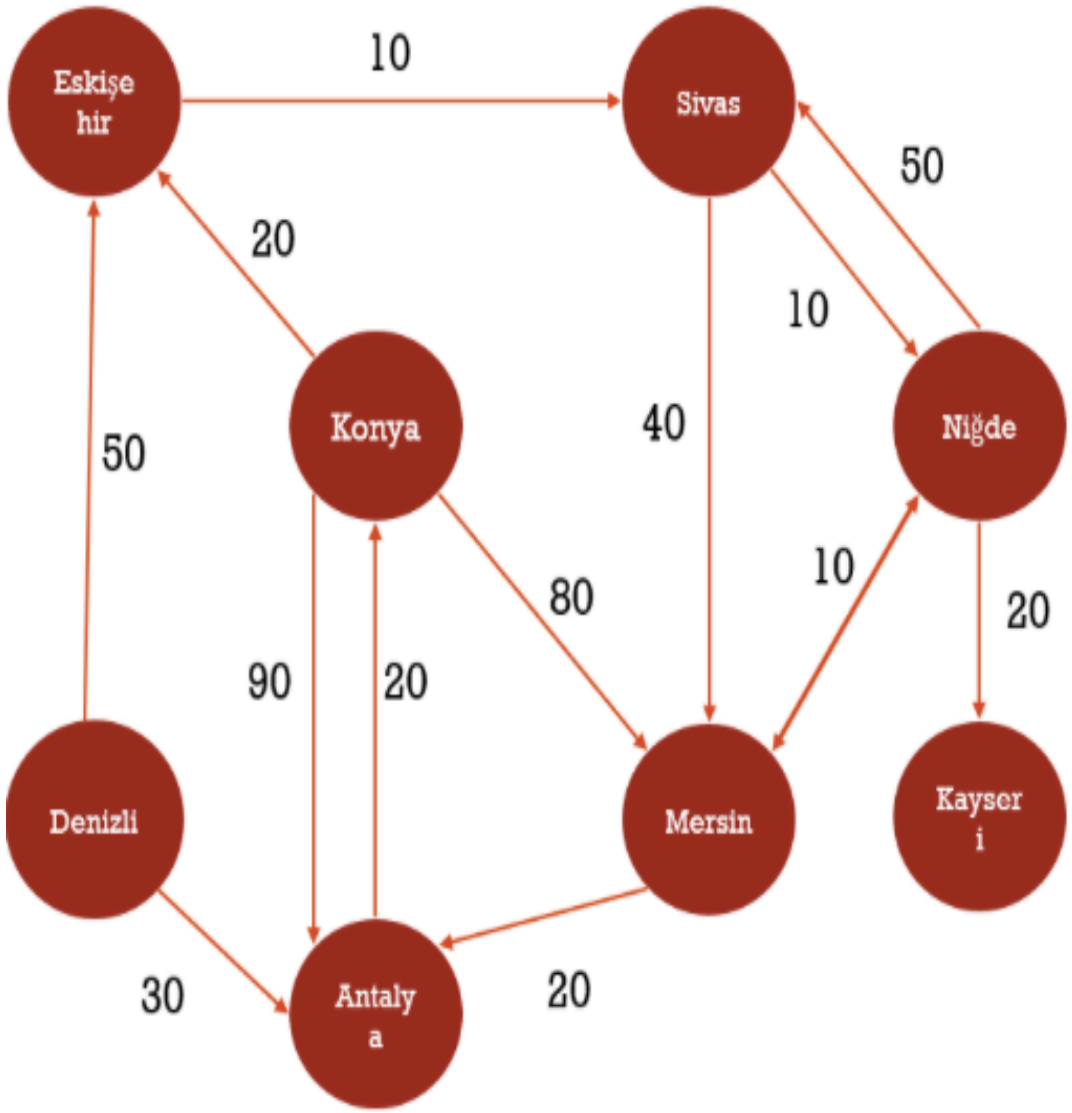
6. Adım:

Yine bu adımda en düşük maliyetli, ziyaret edilmemiş düğümü seçeceğiz. 5. satıra (Mersinli satır) bakıp en düşük maliyetlinin Kayseri olduğunu görüyoruz.

Düğüm	Konya	Eskişehir	Niğde	Mersin	Denizli	Sivas	Antalya	Kayseri
Konya	0	20 Konya	∞	80 Konya	∞	∞	90 Konya	∞
Eskişehir 20	0	20 Konya	∞	80 Konya	∞	30 Eskişe	90 Konya	∞
Sivas 30	0	20 Konya	40 Sivas	70 Sivas	∞	30 Eskişe	90 Konya	∞
Niğde 40	0	20 Konya	40 Sivas	50 Niğde	∞	30 Eskişe	90 Konya	60 Niğde
Mersin 50	0	20 Konya	40 Sivas	50 Niğde	∞	30 Eskişe	70 Antalya	60 Niğde
Kayseri 60	0	20 Konya	40 Sivas	50 Niğde	∞	30 Eskişe	70 Antalya	60 Niğde

En kolay adım bu oldu, zira Kayseri'nin hiç gidilebilecek komşusu yok, tüm maliyetleri aynı şekilde alta geçiriyoruz.

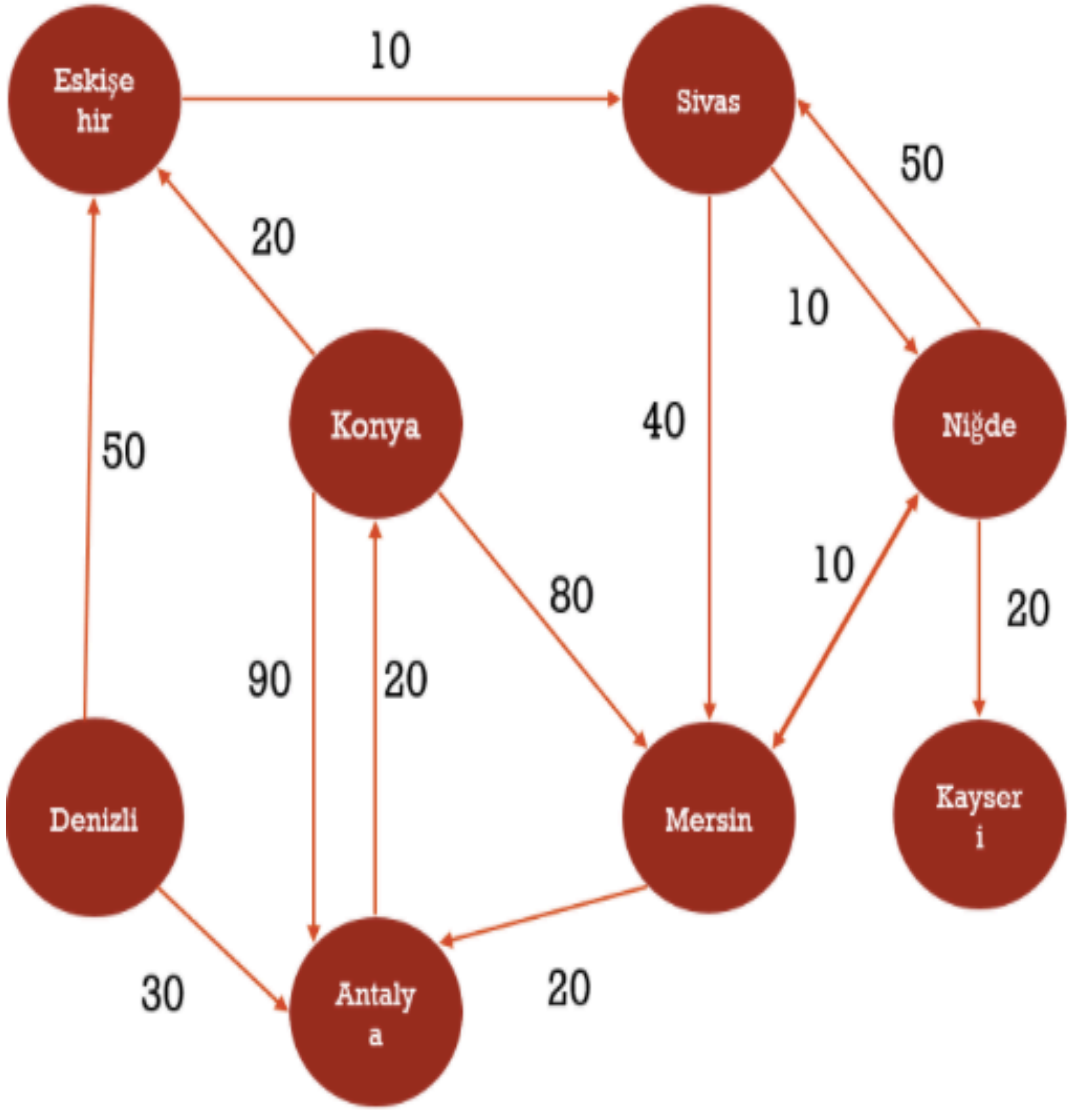
7. Adım:



Geriye kalan en küçük maliyetli düğüm Antalya oldu (Denizli'nin maliyeti sonsuz, diğer düğümler de ziyaret edilmiş durumda)

Düğüm	Konya	Eskişehir	Niğde	Mersin	Denizli	Sivas	Antalya	Kayseri
Konya	0	20 Konya	∞	80 Konya	∞	∞	90 Konya	∞
Eskişehir 20	0	20 Konya	∞	80 Konya	∞	30 Eskişe	90 Konya	∞
Sivas 30	0	20 Konya	40 Sivas	70 Sivas	∞	30 Eskişe	90 Konya	∞
Niğde 40	0	20 Konya	40 Sivas	50 Niğde	∞	30 Eskişe	90 Konya	60 Niğde
Mersin 50	0	20 Konya	40 Sivas	50 Niğde	∞	30 Eskişe	70 Antalya	60 Niğde
Kayseri 60	0	20 Konya	40 Sivas	50 Niğde	∞	30 Eskişe	70 Antalya	60 Niğde
Antalya 70	0	20 Konya	40 Sivas	50 Niğde	∞	30 Eskişe	70 Antalya	60 Niğde

Antalya'dan bir tek Konya'ya yol var, ancak Konya'nın maliyeti zaten sıfır. (Çünkü oradan başladık) Bu yüzden güncelleme yapmıyoruz.



8. Adım:

Bu adımda en küçük maliyetli yol sonsuz değere sahip Denizli. Çünkü ziyaret edilmiş düğümleri baştan eliyoruz. Ama Denizli'nin maliyeti sonsuz olduğu için herhangi bir güncelleme yapmanın olanağı yok. Graf yapısını tekrar incelediğimizde Denizli'ye yol yok. Ulaşamadığımız düğümün değerini de güncelleyemeyiz. Yolu olmayan yere gidemeyiz. Bu yüzden Denizli sonsuz olarak kalır. Yani biz Konya'dan diğer düğümlere olan en kısa yolları bulmuş durumdayız.

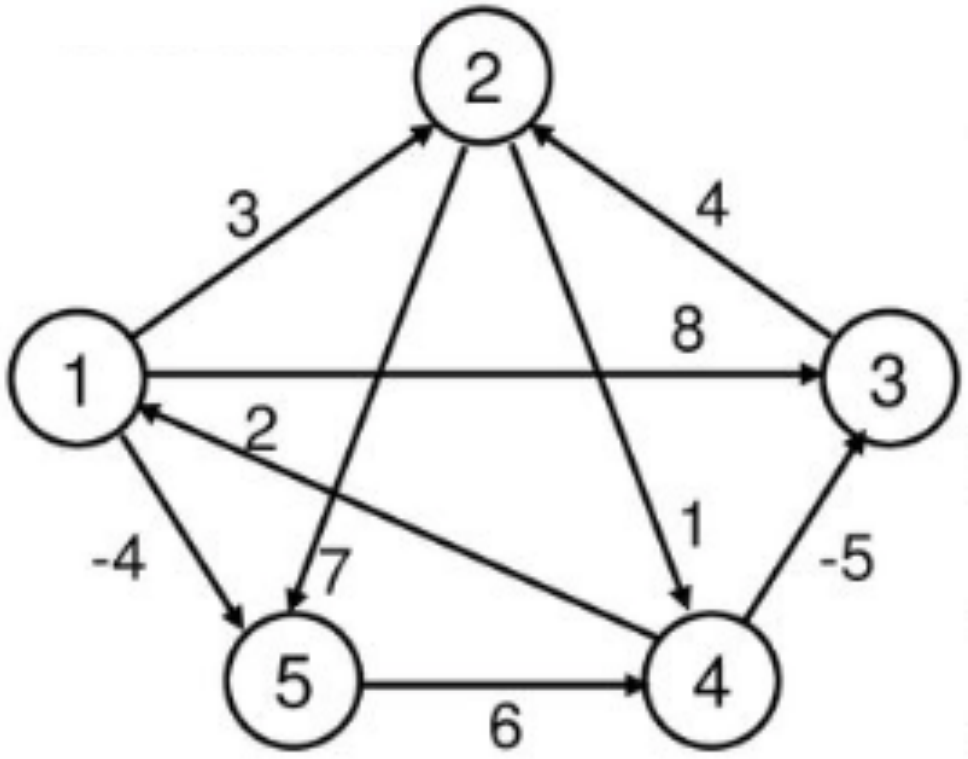
Dijkstra Algoritması C Kodu

```c
//Kodlar Geeks for Geeks'ten alınmıştır. Türkçe yorum satı

#include <stdio.h>
#include <limits.h>

// Graftaki düğüm sayısını V global değişkeni ile ifade ed
#define V 9

//En kısa yolu bulan yardımcı fonksiyon.
//Ziyaret edilmemiş düğümler içerisindeki en kısa maliyetl
int minDistance(int dist[], bool sptSet[])
{
   //İlk başta Sonsuz değer atıyoruz.
   int min = INT_MAX, min_index;
   //Aşağıdaki döngü ziyaret edilmemiş düğümler içerisinde
   //düğümleri tespit ediyor. En küçüğünü buluyor.
   for (int v = 0; v < V; v++)
     if (sptSet[v] == false && dist[v] <= min)
         min = dist[v], min_index = v;

   //En küçük düğümün indexini return ediyoruz.
   return min_index;
}

// Başlangıç noktasından diğer düğümlere olan maliyeti yaz
int printSolution(int dist[], int n)
{
   printf("Dugumlerin ana kaynağa olan uzakligi");
   for (int i = 0; i < V; i++)
      printf("%d tt %dn", i, dist[i]);
}

// Dijkstra algoritmasının temel fonksiyonu
// Ağırlıklı graf yapımızı graph[V][V] dizisinde tutuyoruz
// int src ise başlangıç noktamız.
void dijkstra(int graph[V][V], int src)
{
     int dist[V];     //Başlangıç düğümünden bir başka düğ

     bool sptSet[V];  //Düğüm ziyaret edildiğinde bu diziy


     //İlk başta Tüm düğümleri ziyaret edilmemiş olarak iş
     for (int i = 0; i < V; i++)
        dist[i] = INT_MAX, sptSet[i] = false;

     //Başlangıç noktasının maliyeti her zaman sıfır olur.
     dist[src] = 0;

     // Tüm düğümlere olan uzaklığın (kaynaktan) bulunacağ
     for (int count = 0; count < V-1; count++)
     {
       // Henüz ziyaret edilmemiş, en az maliyetli düğümü
       int u = minDistance(dist, sptSet);

       // Bu düğümü (aslında indisini) ziyaret edildi olar
       sptSet[u] = true;

       // Mevcut düğümden diğer komşu düğümleri tarayan dö
       for (int v = 0; v < V; v++)

         // !sptSet[v] => Eğer düğüm ziyaret edilmediyse V
         // graph[u][v] => Graf üzerinde bu nokta bulunuyo
         // dist[u] != INT_MAX ==> Maliyet Sonsuz değilse
         // ist[u]+graph[u][v] < dist[v] => Mevcut uzaklık
         if (!sptSet[v] && graph[u][v] && dist[u] != INT_M
                                       && dist[u]+graph[u]

            //Uzaklığı güncelliyoruz.
            dist[v] = dist[u] + graph[u][v];
     }

     // Kaynağa olan uzaklığı ekrana bastırıyoruz.
     printSolution(dist, V);
}

// driver program to test above function
int main()
{
   /* Ağırlıklı Graf Yapımızı  aşağıdaki gibi oluşturduk*/
   int graph[V][V] = {{0, 4, 0, 0, 0, 0, 0, 8, 0},
                      {4, 0, 8, 0, 0, 0, 0, 11, 0},
                      {0, 8, 0, 7, 0, 4, 0, 0, 2},
                      {0, 0, 7, 0, 9, 14, 0, 0, 0},
                      {0, 0, 0, 9, 0, 10, 0, 0, 0},
                      {0, 0, 4, 14, 10, 0, 2, 0, 0},
                      {0, 0, 0, 0, 0, 2, 0, 1, 6},
                      {8, 11, 0, 0, 0, 0, 1, 0, 7},
                      {0, 0, 2, 0, 0, 0, 6, 7, 0}
                     };
```

```
92      //0 düğümünü başlangıç noktası olarak belirttik.
93      dijkstra(graph, 0);
94
95      return 0;
96  }
```

FLOYD - ÖRNEK

Aşağıdaki şebekede her iki düğüm arasındaki en kısa yolları veren matris ile sıralama matrisini bulunuz.



D0

	1	2	3	4	5
1	-	3	8	∞	-4
2	∞	-	∞	1	7
3	∞	4	-	∞	∞
4	2	∞	-5	-	∞
5	∞	∞	∞	6	-

S0

	1	2	3	4	5
1	-	2	3	4	5
2	1	-	3	4	5
3	1	2	-	4	5
4	1	2	3	-	5
5	1	2	3	4	-

D1

	1	2	3	4	5
1	-	3	8	∞	-4
2	∞	-	∞	1	7
3	∞	4	-	∞	∞
4	2	5	-5	-	-2
5	∞	∞	∞	6	-

S1

	1	2	3	4	5
1	-	2	3	4	5
2	1	-	3	4	5
3	1	2	-	4	5
4	1	1	3	-	1
5	1	2	3	4	-

D2

	1	2	3	4	5
1	-	3	8	4	-4
2	∞	-	∞	1	7
3	∞	4	-	5	11
4	2	5	-5	-	-2
5	∞	∞	∞	6	-

S2

	1	2	3	4	5
1	-	2	3	2	5
2	1	-	3	4	5
3	1	2	-	2	2
4	1	1	3	-	1
5	1	2	3	4	-

D0

	1	2	3	4	5
1	-	3	8	∞	-4
2	∞	-	∞	1	7
3	∞	4	-	∞	∞
4	2	∞	-5	-	∞
5	∞	∞	∞	6	-

S0

	1	2	3	4	5
1	-	2	3	4	5
2	1	-	3	4	5
3	1	2	-	4	5
4	1	2	3	-	5
5	1	2	3	4	-

D1

	1	2	3	4	5
1	-	3	8	∞	-4
2	∞	-	∞	1	7
3	∞	4	-	∞	∞
4	2	5	-5	-	-2
5	∞	∞	∞	6	-

S1

	1	2	3	4	5
1	-	2	3	4	5
2	1	-	3	4	5
3	1	2	-	4	5
4	1	1	3	-	1
5	1	2	3	4	-

D2

	1	2	3	4	5
1	-	3	8	4	-4
2	∞	-	∞	1	7
3	∞	4	-	5	11
4	2	5	-5	-	-2
5	∞	∞	∞	6	-

S2

	1	2	3	4	5
1	-	2	3	2	5
2	1	-	3	4	5
3	1	2	-	2	2
4	1	1	3	-	1
5	1	2	3	4	-

D3

	1	2	3	4	5
1	-	3	8	4	-4
2	∞	-	∞	1	7
3	∞	4	-	5	11
4	2	-1	-5	-	-2
5	∞	∞	∞	6	-

S3

	1	2	3	4	5
1	-	2	3	2	5
2	1	-	3	4	5
3	1	2	-	2	2
4	1	3	3	-	1
5	1	2	3	4	-

D4

	1	2	3	4	5
1	-	3	-1	4	-4
2	3	-	-4	1	-1
3	7	4	-	5	3
4	2	-1	-5	-	-2
5	8	5	1	6	-

S4

	1	2	3	4	5
1	-	2	4	2	5
2	4	-	4	4	4
3	4	2	-	2	4
4	1	3	3	-	1
5	4	4	4	4	-


TEŞEKKÜRLER