Türkiye Cumhuriyeti

Yıldız Teknik Üniversitesi

Bilgisayar Mühendisliği Bölümü

Öğrenci No: 19011085

Öğrenci Adı Soyadı: Osman Yiğit Sökel

Öğrenci Mail Adresi: l1119085@std.yildiz.edu.tr

DERS/GRUP: YAPISAL PROGRAMLAMAYA GİRİŞ / GR-1
FİNAL PROJESİ

Ders Yürütücüsü
Doç. Dr. Fatih AMASYALI

İçindekiler

Algoritma Tanımı

Dijkstra Algoritması, en kısa yol (shortest path) algoritmalarından biridir. Yani Dijkstra Algoritması, herhangi bir düğümden diğer düğümlere olan yolları analiz edip en kısa yolu bulan bir algoritmadır. Bu algoritmanın temeli çizge teoremine (Graf Teorisi) dayanır.

Program girdi olarak kaç düğümden oluştuğunu, düğümün birbirlerine olan uzaklıklarını ve başlangıç noktasını almaktadır.

Algoritmanın Tarifi

1. Düğüm sayısı girilir.
2. Düğümlerin birbirlerine olan uzaklıkları girilir.
3. Bu değerler alındıktan sonra başlangıç noktası seçilir ve bu düğümden diğer düğümlere olan en kısa yol hesaplanmaya başlanır.
4. Seçtiğimiz düğümle bağlantısı olmayan düğümlere sonsuz değer atanır.
5. Seçtiğimiz düğümlerin diğer düğümlere olan uzaklığı bir diziye atanır. Hangi yol üzerinden ilerlediğimizi tutmak için de önceki düğümleri tutan bir diziye ve ziyaret edilip edilmediğini kontrol eden bir diziye ihtiyacımız var. Başlangıç düğümümüzü önceki düğüm dizimize atıyoruz ve başlangıç değerimizin ziyaret edildiğini işliyoruz.
6. Bir sonraki aşamamızda bulunduğumuz düğümünün bağlantılı olduğu diğer düğümlere uzaklığını, daha önce ölçülen minimum uzaklık ile karşılaştırıyoruz. Eğer minimum uzaklığımızdan daha küçük bir değer ise ve daha önce ziyaret edilmemişse yeni minimum uzaklığımıza bu değer atanır ve izlediğimiz yolun değerlerine eklenir.
7. Bu işlem tüm düğümler için tekrarlanır.
8. Ziyaret edilen düğümü atadığımız için aynı yoldan geçme ihtimali yoktur ve izlediğimiz yolu önceki düğüm dizisi ile tuttuğumuz için izlenen yolu işleyebiliriz.
9. Son olarak en kısa mesafeyi ve izlenen yolu ekrana yazdırıyoruz.

Rakipleri

- Bellman-Ford Algoritması
- A* arama algoritması
- Floyd-Warshall Algoritması
- Johnson Algoritması
- Viterbi Algoritması

	Dijkstra	Bellman-Ford	A* arama	Floyd-Warshall	Johnson	Viterbi
Çözüm şekli	Eksi değerlere izin vermez	Eksi değerlere izin verir	Sezgisel yöntemler	Eksi değerlere izin verir	Grafik algoritması	Olasılıksal ağırlıklar
Bulduğu düğüm sayısı	Tek düğüm	Tek düğüm	İki düğüm arası	Bütün düğümler	Bütün düğümler	Bütün düğümler
	Aç gözlü (Greedy) bir algoritmadır.		Yöntemi sayesinde aramayı hızlandırır		Seyrek çizgilerde daha hızlıdır	

Avantajları

- Sonuçlanınca kalıcı olarak etiketlenmiş tüm düğümlere giden en az ağırlıklı yolu bulabilirsiniz.
- Her geçiş için yeni bir şemaya gerek yoktur.
- $N^{2\prime}$lik bir düzene sahiptir bu yüzden büyük problemlerde kullanmak için verimlidir.

Dezavantajları

- Gerekli kaynakları çok fazla zaman harcayarak kör arama yapması.
- Eksi değer bulunması halinde başarılı çalışmaz. Sürekli mevcut durumdan iyi bir durum oluşturduğu için kararlı hale gelemez.
- Bütün düğümler için değil tek bir düğüm için en kısa yolu bulması.

Karmaşıklığı

Dijkstra Algoritmasının bazı yöntemlere göre karmaşıklığı O(MlogN) fakat benim yazdığım yöntem ile karmaşıklığı ile $O(N^2)$'dır.

Çalışma Prensibi

Kaynak noktasına ihtiyaç duyar ve etiketleme yöntemi ile çalışır. İki çeşit etiketlemesi mevuttur. Bunlar, geçici etiket ve kalıcı etikettir. Geçici etiketleme yapılırken iki düğüm arası mesafe hesaplanılır ve yazılır. Sonraki süreçte yapılan hesaplamalarda daha kısa bir yola rastlanmaz ise geçici etiket, kalıcı etikete dönüşür. Daha kısa bir yol hesaplanır ise kalıcı etiket yeni yol değerini alır.

Uygulama Alanları

- **Navigasyon:** Bulunduğumuz konumdan gitmek istediğimiz konuma en az maliyetli, en yakın, en kolay yolu bulmak için kullanılır.
- **Sosyal Ağ Uygulamaları:** Önerilen arkadaşlar, önerilen sayfalar gibi bize tavsiye edilenleri bulmak için profildeki bağlantıları Dijkstra Algoritması kullanarak bize sunuyor.
- **Telefon Ağları:** Bantlar ve istasyonlarda bu algoritma kullanılır.
- **En Kısa Yolu Bulmak için IP Yönlendirmesi:** Yönlendirme tablolarını güncellemek için bu algoritmayı kullanır.
- **Ulaşım Ağları:** İnternet üzerinden bilet alırken ulaşım merkezlerini, uçuş numaralarını, uçuş saatlerini düzenlemek için bu algoritma kullanılır.
- **Dosya Sunucusu Belirlemede:** LAN'da dosya sunucusu belirlemek için bu algoritma kullanılır. Minimum sayıda atlama yaparak daha hızlı gerçekleşmesini sağlar.
- **Robotik Yol:** Drone ve robotların verilen talimatları daha hızlı gerçekleştirmesi için bu algoritma kullanılır.

Kod

```
1.  #include<stdio.h>
2.  #include<stdlib.h>
3.  #include <time.h>
4.  #define MAX 100
5.  #define SONSUZ 999999
6.
7.
8.  int main()
9.  {
10.   clock_t surebas, surebit;
11.   long double sure[MAX], ortsure, toplamsure, l;
12.   int i, j, k, selection, count, n, baslangic, sayac ,sonrakidugum, dongu;
13.   long maliyet[MAX][MAX], uzaklik[MAX], minuzaklik, a[MAX][MAX];
14.   int ziyaret[MAX], oncekidugum[MAX];
15.   sayac = 0;
16.   k=0;
17.
18.   //While döngüsü menü oluşumunu sağlıyor
19.
20.   while(selection != 3) {
21.             printf("\nMenu:\n(1)Progami Baslat\n(2)Programi Calistirdiktan Sonra
    Analizler\n(3)Cikis\n");
22.             scanf("%d", &selection);
23.             if(selection ==1){
24.
25.                       //En kısa yolu hesaplamak için nokta sayısını ve uzaklıklarını
    kullanıcıdan alıyor
26.
27.                       printf("Dugum sayisini giriniz = ");
28.                       scanf("%d", &n);
29.
30.                       for(i=1; i<=n; i++){
31.                                 for(j=1; j<=n; j++){
32.                                           if(i != j){
33.                                           printf("%d.dugumun %d.dugume uzakligini
    giriniz(Baglanti yoksa 0 giriniz) =", i, j);
34.                                           scanf("%ld", &a[i][j]);
35.                                           }
36.                                           else
37.                                                     a[i][j]=0;
38.                                 }
39.                       }
40.
41.                       /*Dijkstra Algoritması tek kaynaklı bir shortest path algoritması
    olduğu için
42.                       hangi noktanın ölçümünü yapcağını giriyor. */
43.
44.
45.                       printf("Baslangic dugumunu giriniz = ");
46.                       scanf("%d", &baslangic);
47.
48.                       surebas = clock();
49.
50.                                 /*Düğümler arasındaki uzaklıkları matrise eşitliyor ve
51.                                 0 olan değerlere sonsuz değerini atıyor. */
52.   for(dongu=0;dongu<1000; dongu++){
53.                                 for(i=1; i<=n; i++){
54.                                           for(j=1; j<=n; j++){
55.                                                     printf(" *");
              //Yıldız burdaki karmaşıklığı ölçüyor.
56.                                                     if(a[i][j] == 0)
57.                                                               maliyet[i][j] = SONSUZ;
58.                                                     else
59.                                                               maliyet[i][j] = a[i][j];
60.                                           }
61.                                 }
```

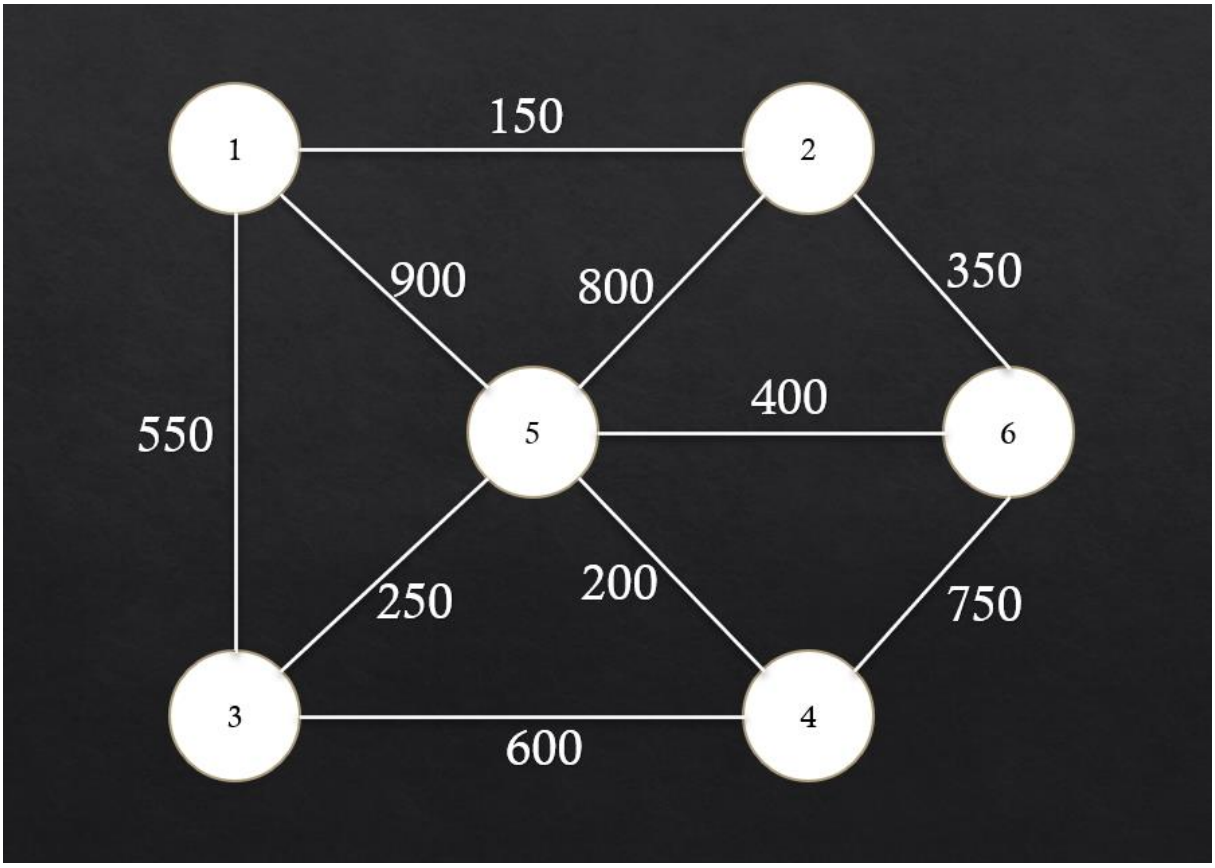
```
62.                                 printf("\n");
63.
64.                                 //Uzaklık dizisine maliyet matrisindeki başlangıç
    düğümünün diğer düğümlere olan uzaklığını atıyor.
65.                                 //Önceki düğüm dizisinin bütün değerlerine başlangıç
    düğümünün değerini atıyor. Bu izlenilen yolu tutmamız için yapılıyor.
66.                                 //Ziyaret dizisinin bütün değerlerine de 0 değerini
    atıyor.Bu şu an için ziyaret edilen hiçbir noktanın olmadığını göstermek için yapılıyor
67.
68.                                 for(i=1; i<=n; i++){
69.                                         printf(" *");
70.                                         uzaklik[i] = maliyet[baslangic][i];
71.                                         oncekidugum[i] = baslangic;
72.                                         ziyaret[i] = 0;
73.                                 }
74.
75.                                 printf("\n");
76.
77.                                 uzaklik[baslangic] = 0;                 //Başlangıç
    düğümünün kendisine olan uzaklığını 0 yapıyor1
78.                                 ziyaret[baslangic]=1;                   //Başlangıç
    noktasının ziyaret edildiği bilgisini veriyor
79.
80.                                 /*Burda while döngüsü bütün değerlerin uzaklıklarının
    ölçülmesi için yapılıyor.
81.                                 Normalden 1 değer az dönüyor çünkü ilk değerlerin
    atamaları yapıldı. */
82.
83.                                 count = 1;

84.                                 while(count < n){
85.                                         minuzaklik = SONSUZ;
86.                                         for(i=1; i<=n; i++){
87.                                                 printf(" *");
88.                                                 if(uzaklik[i] < minuzaklik &&
    !ziyaret[i]){                   //Uzaklık değerlerinin belirlenen minimum uzaklıktan
    küçük olup olmadığı ve o noktanın daha önce ziyaret
89.                                                         minuzaklik = uzaklik[i];
                                                //edilip edilmediği kontrol ediliyor. Eğer
    şartlar sağlanıyorsa yeni minimum değerimize bu uzaklık ekleniyor.
90.                                                         sonrakidugum = i;

91.                                                 }

      //Şartları sağlayan en optimum değer bulunduktan sonra o noktanın ziyaret edildiği,
    diziye işleniyor.
92.                                         }
93.                                         printf("\n");

94.                                         ziyaret[sonrakidugum] = 1;
95.
96.                                         for(i=1; i<=n; i++){
97.                                                 printf(" *");
98.                                                 /*Eğer ziyaret edilmeyen bir noktada;
    minimum uzaklık ve belirlediğimiz sonraki düğümün uzaklığının toplamı
99.                                                 başlangıç düğümümüzün uzaklıklarınndan
    küçükse, yeni uzaklık değerimiz bu iki değerin toplamı olur.
100.                                                        Önceki düğüm dizimizde de bu
    değer atanır.  */
101.                                                        if(ziyaret[i] == 0){


102.                                                                if(minuzaklik +
    maliyet[sonrakidugum][i] < uzaklik[i]){
103.
      uzaklik[i] = minuzaklik + maliyet[sonrakidugum][i];
104.
      oncekidugum[i] = sonrakidugum;
105.                                                                }
106.                                                        }
```

```
107.                                                  }
108.
109.                                                  count++;
110.                                          }
111.                                          printf("\n \n");
112.
113.
114.                                  }
115.
116.                                  //Bu döngülerde başlangıç düğümümüzün diğer düğümlere en
    kısa yolu ve ilerledikleri yol ekrana yazdırılır.
117.
118.                                  for(i=1; i<=n; i++){
119.                                          printf(" *");
120.                                          if(i != baslangic){
121.                                                  printf("%d.dugume uzakligi = %ld\n",i,
    uzaklik[i]);
122.                                                  printf("Gidilen yol : %d ", i);
123.
124.                                                  j = i;
125.                                                  while(j != baslangic){
126.                                                          printf(" *");
127.                                                          j = oncekidugum[j];
128.                                                          printf("<-- %d ", j);
129.                                                  }
130.                                          }
131.
132.                                          printf("\n");
133.                                  }
134.                                  printf("\n \n");
135.
136.
137.                                  surebit = clock();
138.
139.                                  printf("\n Algoritmanin calisma suresi(mikrosaniye) = %Lf
    \n \n", ((long double)(surebit-surebas)/(1000 * CLOCKS_PER_SEC)));
140.
141.                                  /*Bütün istediğimiz değerleri girdikten sonra
    karşılaştırma yapabilmek için
142.                                  bulduğumuz süreleri bir diziye atıyor */
143.
144.                                  sure[k] = (long double)(surebit-surebas)/ (1000 *
    CLOCKS_PER_SEC);
145.                                  k++;
146.                                  sayac++;
147.                          }
148.
149.                          else if(selection == 2){
150.                                  if(sayac >= 1){
151.
152.                                          //Tuttuğumuz bütün değerler için bir bar
    diyagramı yapıyor.
153.                                          printf("Algoritmanin Calisma Suresinin Bar
    Diagrami\n");
154.                                          k=0;
155.                                          while(k<sayac){
156.                                                  printf("%d.degerler  ", k+1);
157.                                                  for(l=0; l<sure[k]/2; l = l  +
    0.000001){
158.                                                          printf("|");
159.                                                  }
160.                                                  printf("\n");
161.                                                  k++;
162.                                          }
163.
164.                                          //Elde ettiğimiz zamanın ortalamasını alıyor ve
    bunu şekillendiriyor.
165.
166.                                          toplamsure = 0;
167.                                          for(k=0; k<sayac; k++)
```

```
168.                                                    toplamsure = toplamsure + sure[k];
169.
170.                                            ortsure = toplamsure / sayac;
171.                                            printf("Ort. sure   ");
172.
173.                                            for(l=0; l<ortsure/2; l = l + 0.000001)
174.                                                    printf("|");
175.
176.                                            //Yıldızlarla  hesapladığımız karmaşıklığı
    veriyor.
177.
178.                                            printf("\n\nKarmasiklik  = O(N^2)\n");
179.                                    }
180.
181.                                    else
182.                                            printf("Analizler Icin Programi
    Calistiriniz!!");
183.
184.                            }
185.
186.                            //Programı sonlandırma
187.
188.                            else if(selection==3){
189.                                    printf("Cikis Yapiliyor..\n");
190.                            }
191.
192.                            //Menüdeki değerlerden farklı bir değer girme durumunda ekrana
    yazırıyor
193.
194.                            else
195.                                    printf("Yanlis deger! Yeniden deneyiniz\n");
196.                    }
197.
198.                    return 0;
199.
200.  }
201.
```

Ekran Çıktıları

1.Değerler

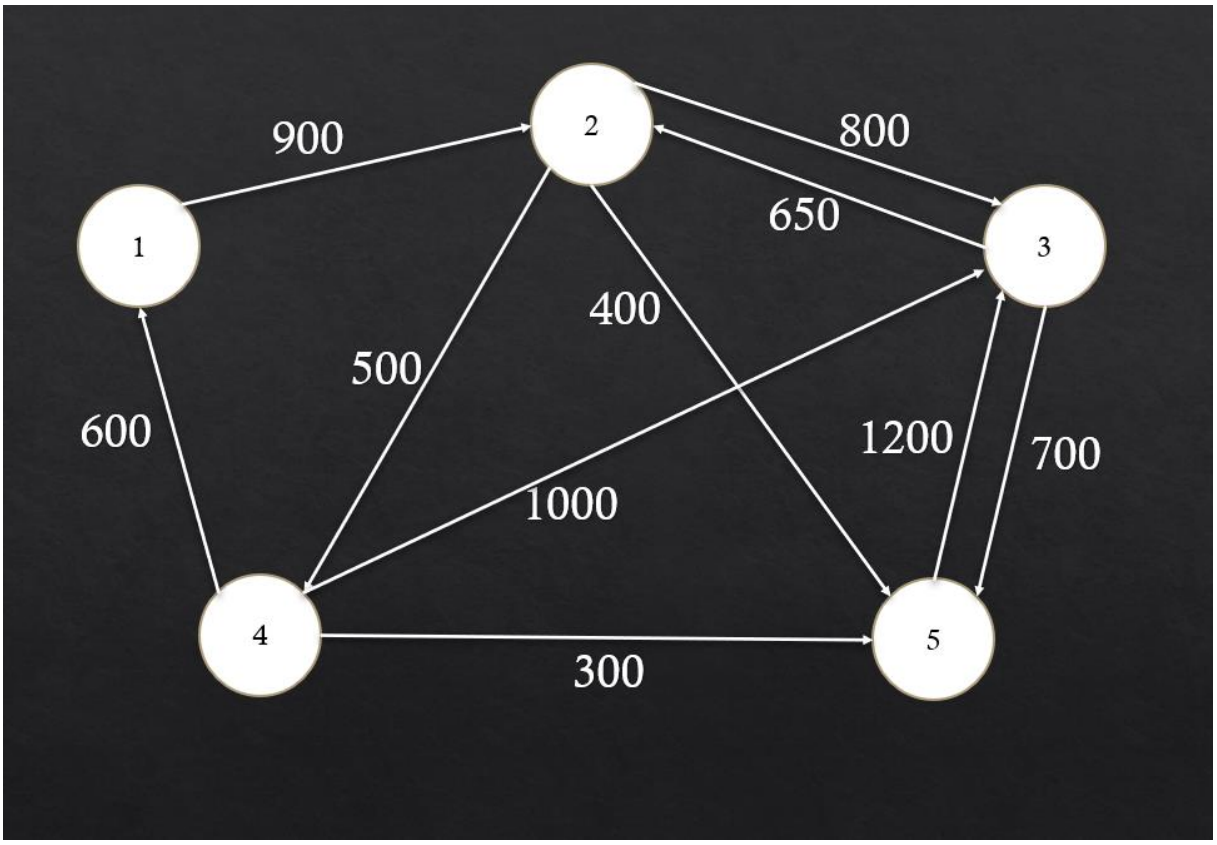


4 numaralı düğüme göre en kısa mesafeyi bulmak için:

Düğüm	**1**	**2**	**3**	**4**	**5**	**6**
4	Sonsuz	Sonsuz	600	0	200	750
5(200)	1100	1000	450	0	200	600
3(450)	1000	1000	450	0	200	600
6(600)	1000	950	450	0	200	600
2(950)	1000	950	450	0	200	600
1(1000)	1000	950	450	0	200	600

```
yigit@OYS: ~/Desktop
2.dugumun 5.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =800
2.dugumun 6.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =350
3.dugumun 1.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =550
3.dugumun 2.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =0
3.dugumun 4.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =600
3.dugumun 5.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =250
3.dugumun 6.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =0
4.dugumun 1.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =0
4.dugumun 2.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =0
4.dugumun 3.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =600
4.dugumun 5.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =200
4.dugumun 6.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =750
5.dugumun 1.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =900
5.dugumun 2.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =800
5.dugumun 3.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =250
5.dugumun 4.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =200
5.dugumun 6.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =400
6.dugumun 1.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =0
6.dugumun 2.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =350
6.dugumun 3.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =0
6.dugumun 4.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =750
6.dugumun 5.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =400
Baslangic dugumunu giriniz = 4
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 * * * * * *
 * * * * * *
 * * * * * * * * * * * *
 * * * * * * * * * * * *
 * * * * * * * * * * * *
 * * * * * * * * * * * *
 * * * * * *

 *1.dugume uzakligi = 1000
Gidilen yol : 1  *<-- 3  *<-- 5  *<-- 4
 *2.dugume uzakligi = 950
Gidilen yol : 2  *<-- 6  *<-- 5  *<-- 4
 *3.dugume uzakligi = 450
Gidilen yol : 3  *<-- 5  *<-- 4
 *
 *5.dugume uzakligi = 200
Gidilen yol : 5  *<-- 4
 *6.dugume uzakligi = 600
Gidilen yol : 6  *<-- 5  *<-- 4


  Algoritmanin calisma suresi(mikrosaniye) = 0.000021
```

2.Değerler

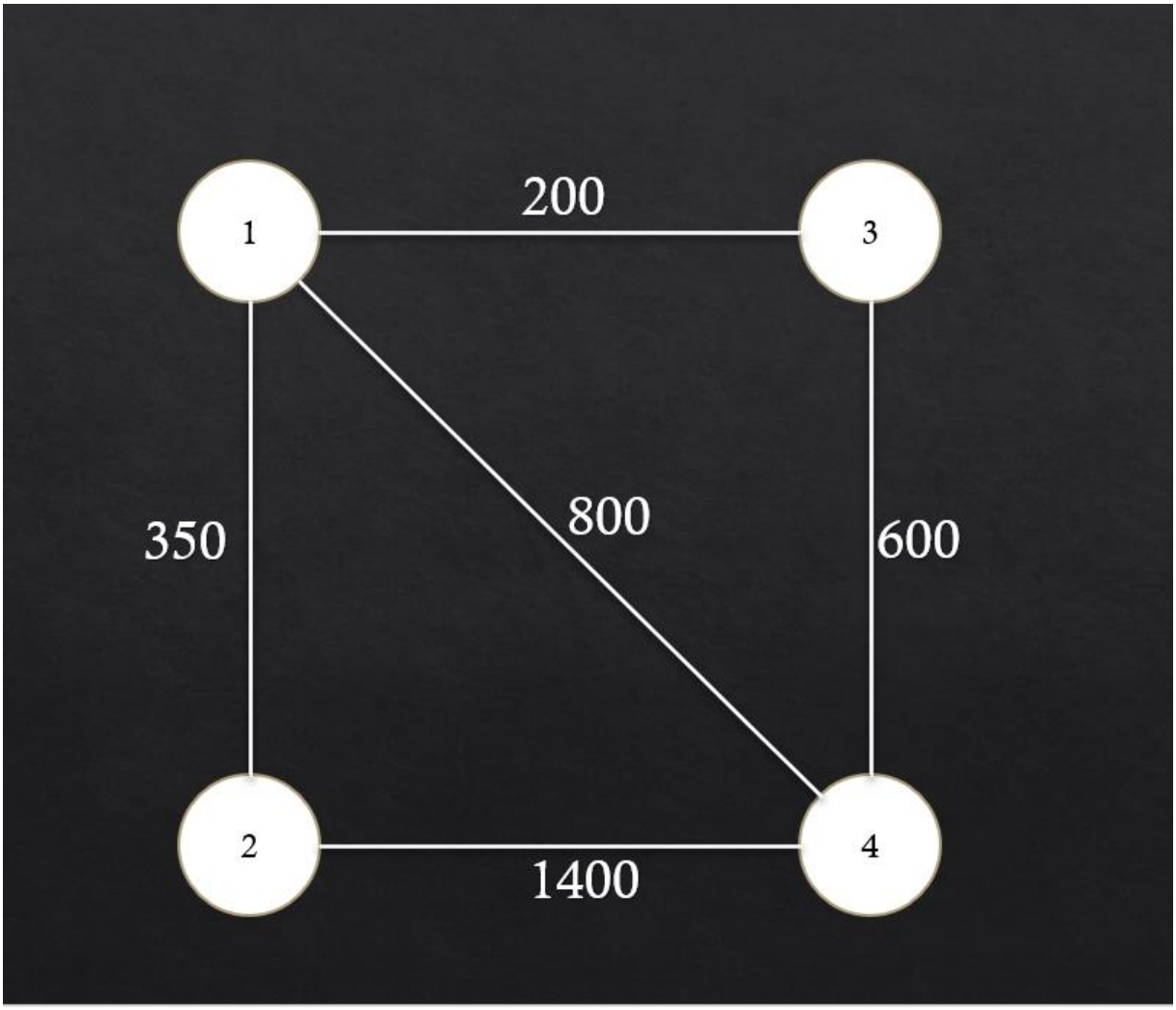


1 numaralı düğüme göre en kısa mesafeyi bulmak için:

Düğüm	**1**	**2**	**3**	**4**	**5**
1	0	900	Sonsuz	Sonsuz	Sonsuz
2(900)	0	900	1700	1400	1300
5(1300)	0	900	1700	1400	1300
4(1400)	0	900	1700	1400	1300
3(1700)	0	900	1700	1400	1300

```
yigit@OYS: ~/Desktop
(1)Progami Baslat
(2)Programi Calistirdiktan Sonra Analizler
(3)Cikis
1
Dugum sayisini giriniz = 5
1.dugumun 2.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =900
1.dugumun 3.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =0
1.dugumun 4.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =0
1.dugumun 5.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =0
2.dugumun 1.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =0
2.dugumun 3.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =800
2.dugumun 4.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =500
2.dugumun 5.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =400
3.dugumun 1.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =0
3.dugumun 2.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =650
3.dugumun 4.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =0
3.dugumun 5.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =700
4.dugumun 1.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =600
4.dugumun 2.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =0
4.dugumun 3.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =1000
4.dugumun 5.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =300
5.dugumun 1.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =0
5.dugumun 2.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =0
5.dugumun 3.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =1200
5.dugumun 4.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =0
Baslangic dugumunu giriniz = 1
 * * * * * * * * * * * * * * * * * * * * * * * * * *
 * * * * *
 * * * * *
 * * * * * * * * * *
 * * * * * * * * * *
 * * * * * * * * * *
 * * * * *

 *
 *2.dugume uzakligi = 900
Gidilen yol : 2  *<-- 1
 *3.dugume uzakligi = 1700
Gidilen yol : 3  *<-- 2  *<-- 1
 *4.dugume uzakligi = 1400
Gidilen yol : 4  *<-- 2  *<-- 1
 *5.dugume uzakligi = 1300
Gidilen yol : 5  *<-- 2  *<-- 1


  Algoritmanin calisma suresi(mikrosaniye) = 0.000016
```

3.Değerler



2 numaralı düğüme göre en kısa mesafeyi bulmak için:

Düğüm	**1**	**2**	**3**	**4**
2	350	0	800	1400
1(350)	350	0	550	1400
3(550)	350	0	550	1150
4(1150)	350	0	550	1150

```
Menu:
(1)Progami Baslat
(2)Programi Calistirdiktan Sonra Analizler
(3)Cikis
1
Dugum sayisini giriniz = 4
1.dugumun 2.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =350
1.dugumun 3.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =200
1.dugumun 4.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =0
2.dugumun 1.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =350
2.dugumun 3.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =800
2.dugumun 4.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =1400
3.dugumun 1.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =200
3.dugumun 2.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =800
3.dugumun 4.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =600
4.dugumun 1.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =0
4.dugumun 2.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =1400
4.dugumun 3.dugume uzakligini giriniz(Baglanti yoksa 0 giriniz) =600
Baslangic dugumunu giriniz = 2
 * * * * * * * * * * * * * * * *
 * * * *
 * * * *
 * * * * * * * *
 * * * * * * * *
 * * * *

 *1.dugume uzakligi = 350
Gidilen yol : 1  *<-- 2
 *
 *3.dugume uzakligi = 550
Gidilen yol : 3  *<-- 1  *<-- 2
 *4.dugume uzakligi = 1150
Gidilen yol : 4  *<-- 3  *<-- 1  *<-- 2


 Algoritmanin calisma suresi(mikrosaniye) = 0.000012
```

```
Menu:
(1)Progami Baslat
(2)Programi Calistirdiktan Sonra Analizler
(3)Cikis
2
Algoritmanin Calisma Suresinin Bar Diagrami
1.degerler  ||||||||||
2.degerler  ||||||||
3.degerler  |||||||
Ort. sure   |||||||||

Karmasiklik  = O(N^2)

Menu:
(1)Progami Baslat
(2)Programi Calistirdiktan Sonra Analizler
(3)Cikis
3
Cikis Yapiliyor..
```

Kaynakça

- https://cs.indstate.edu/~rjaliparthive/dijkstras.pdf
- https://bilgisayarkavramlari.com/2010/05/13/dijkstra-algoritmasi-2/#:~:text=Bilgisayar%20bilimlerinde%20kullanılan%20ve%20algoritmayı,shortest%20path)%20bulmak%20için%20kullanılır.&text=Dijkstra%20algoritması%20herhangi%20bir%20şekildeki,giden%20en%20kısa%20yolu%20hesaplar.
- https://www.halildurmus.com/2020/10/26/dijkstra-algoritmasi/
- https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/
- https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm
- https://www.youtube.com/watch?v=ZtBMVxyD8A0&t=41s
- https://www.youtube.com/watch?v

Video Linki

https://youtu.be/hN-kzPaJCmI