

# **CENG 481 GRAF TEORİ VE UYGULAMALARI**

## **Hafta 11**

**Prof. Dr. Tufan TURACI**  
**tturaci@pau.edu.tr**

# Hafta 11

## Konular

### 1- Graflarda Uzaklık (Distance in Graphs)

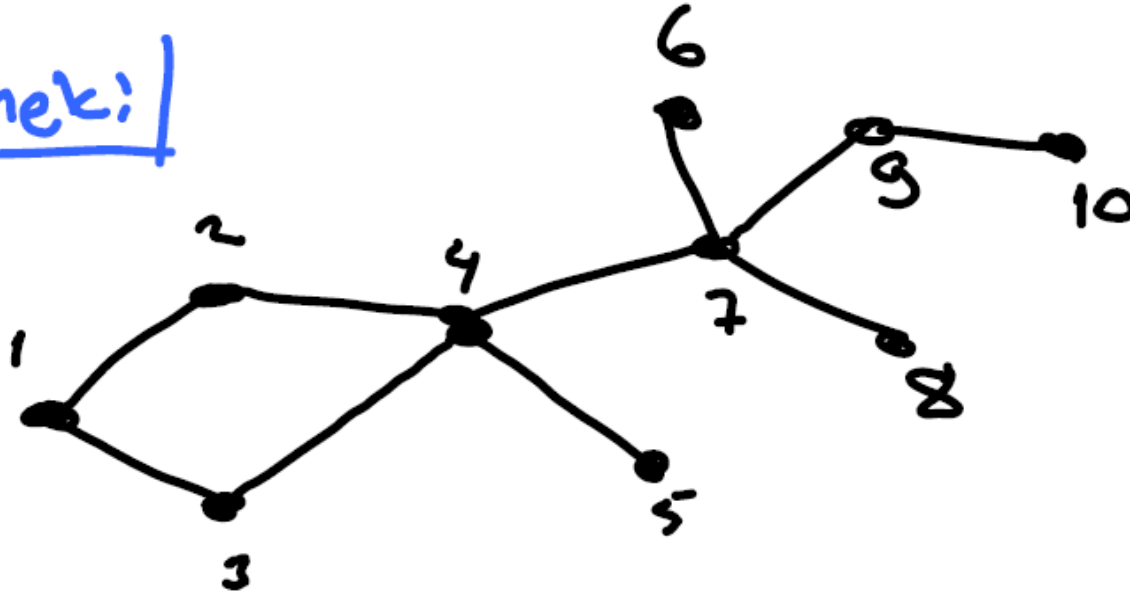
**Tanım 1:** Bir  $G$  grafında  $u$  ve  $v$  gibi iki tepe arasındaki yollar içinde minimum uzunluğu olanın uzunluğuna;  $u$  ve  $v$  nin **uzaklığı (distance)** denir ve  $d(u,v)$  (veya  $d_G(u,v)$ ) biçiminde gösterilir.

**Tanım 2:**  $n$  tepeli bir  $G$  grafında, grafın tepeleri  $\{v_1, v_2, \dots, v_n\}$  olsun.  $G$  grafının **komşuluk matrisi**  $A(G) = [a_{ij}]$  dir.

$$A(G) = \begin{cases} a_{ij} = 1, & v_i \text{ ile } v_j \text{ komşu ise;} \\ a_{ij} = 0, & \text{aksi halde.} \end{cases}$$

$A(G) = [a_{ij}]$  nin satır ve sütunları grafın tepelerine karşılık gelir.

Örnek:



G

Verilen G grafında tüm tepe  
çiftleri arasındaki uzaklıklar,  
bulunuz.

$$d(1,2) = 1$$

$$d(1,3) = 1$$

$$d(1,4) = 2$$

$$d(1,5) = 3$$

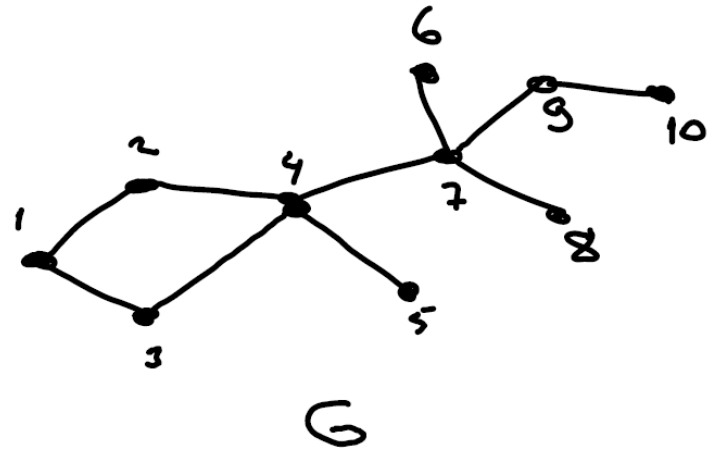
$$d(1,6) = 4$$

$$d(1,7) = 3$$

$$d(1,8) = 4$$

$$d(1,9) = 4$$

$$d(1,10) = 5$$



$$d(2,3) = 2$$

$$d(2,4) = 1$$

$$d(2,5) = 2$$

$$d(2,6) = 3$$

$$d(2,7) = 2$$

$$d(2,8) = 3$$

$$d(2,9) = 3$$

$$d(2,10) = 4$$

$$d(3,4) = 1$$

$$d(3,5) = 2$$

$$d(3,6) = 3$$

$$d(3,7) = 2$$

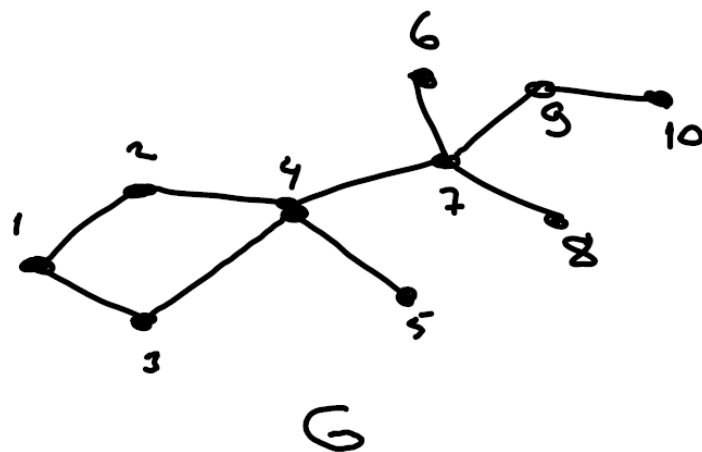
$$d(3,8) = 3$$

$$d(3,9) = 3$$

$$d(3,10) = 4$$

$$\begin{aligned} d(4,5) &= 1 \\ d(4,6) &= 2 \\ d(4,7) &= 1 \end{aligned}$$

$$\begin{aligned} d(4,8) &= 2 \\ d(4,9) &= 2 \\ d(4,10) &= 3 \end{aligned}$$



$$\begin{aligned} d(5,6) &= 3 \\ d(5,7) &= 2 \\ d(5,8) &= 3 \end{aligned}$$

$$\begin{aligned} d(5,9) &= 3 \\ d(5,10) &= 4 \end{aligned}$$

$$\begin{aligned} d(7,8) &= 1 & d(7,10) &= 2 \\ d(7,9) &= 1 \end{aligned}$$

$$\begin{aligned} d(6,7) &= 1 \\ d(6,8) &= 2 \end{aligned}$$

$$\begin{aligned} d(6,9) &= 2 \\ d(6,10) &= 3 \end{aligned}$$

$$\begin{aligned} d(8,9) &= 2 \\ d(8,10) &= 3 \\ d(9,10) &= 1 \end{aligned}$$

# Graflarda Uzaklık Algoritmaları

## 1. Floyd Algoritması

Floyd Algoritması, graf üzerindeki her bir tepe için diğer tepelere olan en kısa yolları ve bu yolların uzaklıklarını bulmak için kullanılan bir algoritmadır. En kısa yolu bulmak için en genel algoritma Floyd'un algoritmasıdır. Grafin bitişiklik matrisi şeklinde tutulması durumunda bu algoritma  $O(n^3)$  karmaşıklığında olmaktadır.

# Algoritma:

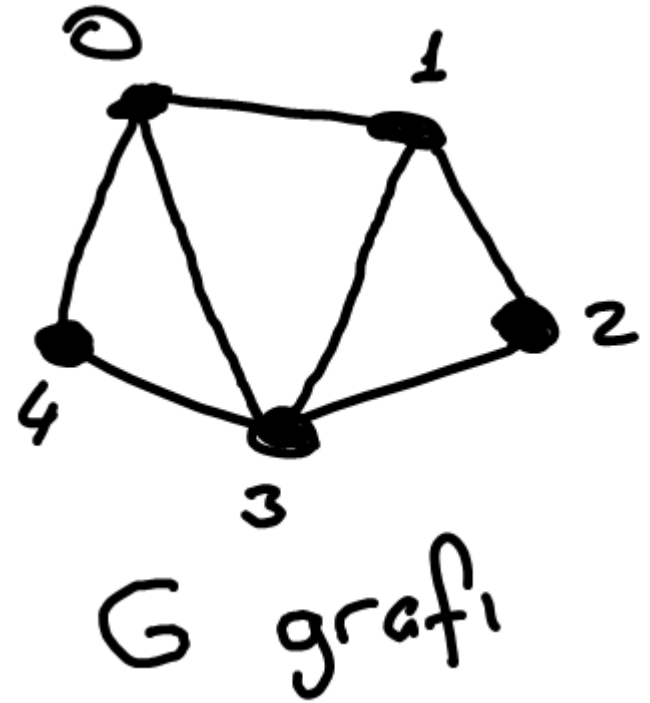
```
for i=1 to n
  for j=1 to n
    if  $A[i,j] \neq 0$  then  $D[i,j]=A[i,j]$ 
      else  $D[i,j]=\infty$ 
  repeat
repeat
for k=1 to n
  for i=1 to n
    for j=1 to n
      if  $(D[i,k]+D[k,j] < D[i,j])$  then  $D[i,j]=D[i,k]+D[k,j]$ 
    repeat
  repeat
repeat
```



## C kodu: Yanda verilen G grafi için...

```
#include <stdio.h>
#include <conio.h>
#define MAX 1000
int main()
{
    int i, j, k, n=5; int D[5][5];
    int A[5][5] = {{0, 1, 0, 1, 1},
                   {1, 0, 1, 1, 0},
                   {0, 1, 0, 1, 0},
                   {1, 1, 1, 0, 1},
                   {1, 0, 0, 1, 0}};

    for (i = 0; i < n; i++){
        for (j = 0; j < n; j++) {
            if (A[i][j] != 0) D[i][j] = A[i][j];
            if (i == j) D[i][j] = 0;
            if ((i != j) && A[i][j] == 0) D[i][j] = MAX;
        }
    }
```



```

for (k = 0; k < n; k++) {
    for (i = 0; i < n; i++){
        for (j = 0; j < n; j++){
            if ((D[i][k] + D[k][j]) < D[i][j]) D[i][j]=D[i][k] + D[k][j];

        }
    }
}

```

```

printf("Tepe ciftleri arasinda en kisa yollar:\n\n");

```

```

for (i = 0; i < n; i++) {
    for (j = 0; j < n; j++) {
        if (i==j) printf("%d - %d uzaklik = %d ", i,j,0);
        if (i!=j) printf("%d - %d uzaklik = %d ", i,j,D[i][j]);
        printf("\n");
    }
}

```

```

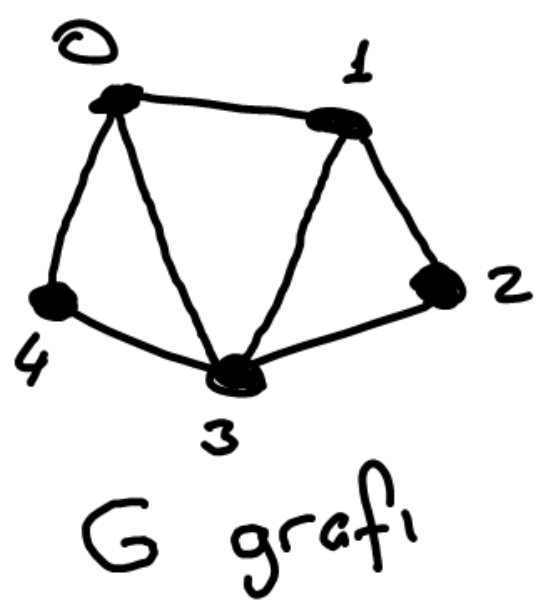
getch();
return 0;
}

```

Tepe ciftleri arasinda en kisa yollar:

0	-	0	uzaklik = 0
0	-	1	uzaklik = 1
0	-	2	uzaklik = 2
0	-	3	uzaklik = 1
0	-	4	uzaklik = 1
1	-	0	uzaklik = 1
1	-	1	uzaklik = 0
1	-	2	uzaklik = 1
1	-	3	uzaklik = 1
1	-	4	uzaklik = 2
2	-	0	uzaklik = 2
2	-	1	uzaklik = 1
2	-	2	uzaklik = 0
2	-	3	uzaklik = 1
2	-	4	uzaklik = 2
3	-	0	uzaklik = 1
3	-	1	uzaklik = 1
3	-	2	uzaklik = 1
3	-	3	uzaklik = 0
3	-	4	uzaklik = 1
4	-	0	uzaklik = 1
4	-	1	uzaklik = 2
4	-	2	uzaklik = 2
4	-	3	uzaklik = 1
4	-	4	uzaklik = 0

-----  
Process exited with return value 0



## 2. Dijkstra Algoritması

- Dijkstra algoritması, bir tepeden diğer tüm tepelere en kısa yolları hesaplar.
- Ağırlıklı ve yönlü graflar için geliştirilmiştir.
- Dijkstra algoritmasının zaman karmaşıklığı genel olarak  $O(m \log n)$  şeklindedir.
- Dijkstra algoritması en kısa yolları Greedy yaklaşımı ile belirler.
- Yani bir düğümden diğer bir düğüme geçerken olası en iyi yerel çözümü göz önüne alır.
- Her seferinde bir sonraki düğüme ilerleme Greedy yaklaşımına göre yapılır.

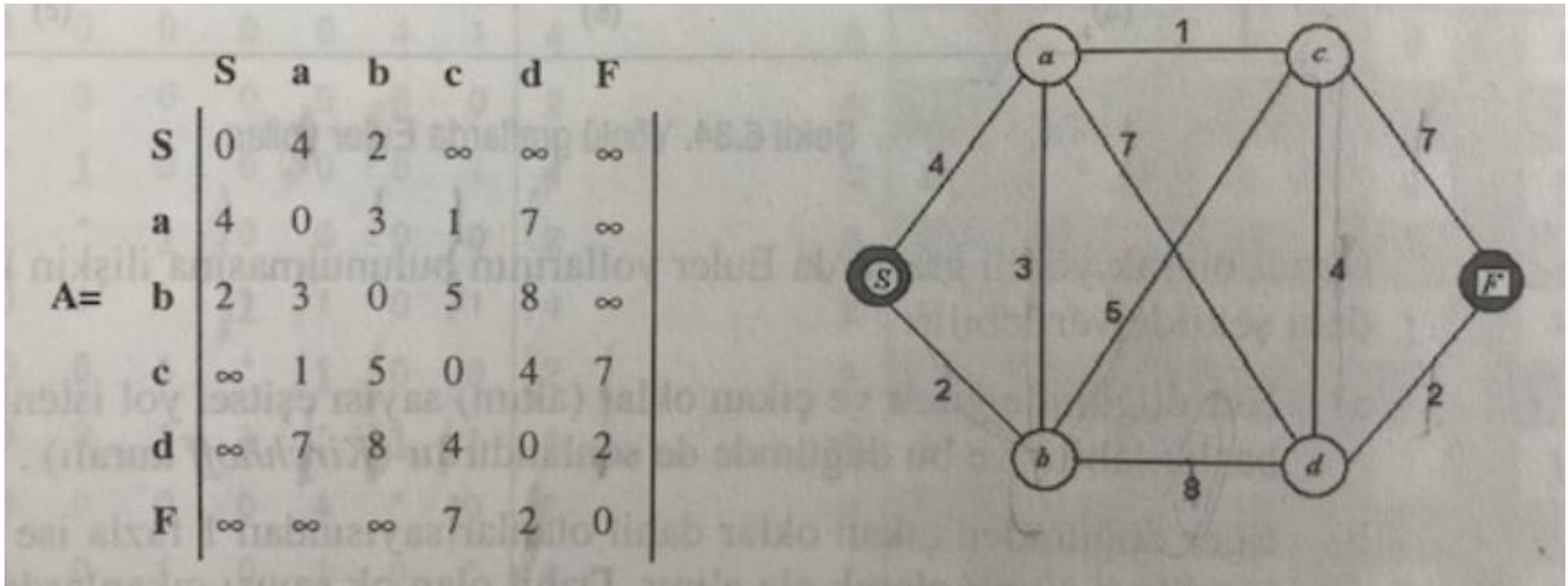
# Algoritma:

```
Dijkstra_uzaklık (graf G, düğüm v0)
// v0 başlangıç düğüm
{
    S = {v0};
    for(i=1; i<n; i++)
        D[i]=C[v0,i];          // C[i,j] - maliyet matrisi
    for (i=0; i<n-1; i++)
    {
        V-S'den öyle bir w düğümü seçelim ki D[w] minimum olsun;
        w'yi S'ye ekle;

        for v∈V - S
            D[v]=min(D[v], D[w]+C[w,v]);    // uzaklıkları yeniden ayarla
    }
}
```

**C kodu çalışma olarak bırakılmıştır.**

**Örnek:** Örnek, 6 numaralı kaynaktan alınmıştır.  
S den F ye en kısa yolu bulunuz.



S den uzaklıklar...

Tepe	S	a	b	c	d	F
Uzaklık	0	4	2	$\infty$	$\infty$	$\infty$

1. adım: a,b,c,d,F arasından minimum olan b=2

Tepe	S	a	<sup>*</sup> b	c	d	F
Uzaklık	0	4	2	$\infty$	$\infty$	$\infty$

$$(b,a) \text{ için } D(a) = \min(4, 2+3) = 4$$

$$(b,c) \text{ için } D(c) = \min(\infty, 2+5) = 7$$

$$(b,d) \text{ için } D(d) = \min(\infty, 2+8) = 10$$

$$(b,f) \text{ için } D(f) = \min(\infty, \infty) = \infty$$

Böylece; yeni tablo aşağıdaki gibidir.

	* S a b c d f					
Tepeler	S	a	b	c	d	f
S	0	4	2	$\infty$	$\infty$	$\infty$
b	0	4	2	7	10	$\infty$



2. adım: a,c,d,F arasından minimum olan a=4

Tepeles	S	* a	* b	c	d	F
S	0	4	2	$\infty$	$\infty$	$\infty$
b	0	<u>4</u>	2	7	10	$\infty$

$$\min \{a, c, d, F\} = 4$$

$$(a, c) \text{ için } D(c) = \min(7, 4+1) = 5$$

$$(a, d) \text{ için } D(d) = \min(10, 4+7) = 10$$

$$(a, F) \text{ için } D(F) = \min(\infty, \infty) = \infty$$

2. Adım sonrası yeni tablo aşağıdaki gibidir.

Tepeks						
	S	<sup>*</sup> a	<sup>*</sup> b	c	d	F
S	0	4	2	$\infty$	$\infty$	$\infty$
b	0	4	2	7	10	$\infty$
a	0	4	2	5	10	$\infty$

3. adım: c,d,F arasından minimum olan c=5

Tepeler	S	<del>a</del>	<del>b</del>	<del>c</del>	d	F
S	0	4	2	$\infty$	$\infty$	$\infty$
b	0	4	2	7	10	$\infty$
a	0	4	2	5	10	$\infty$

$$\min \{c, d, F\} = 5$$

$$(c, d) \text{ için } D(d) = \min(10, 5+4) = 9$$

$$(c, F) \text{ için } D(F) = \min(\infty, 5+7) = 12$$

3. Adım sonrası yeni tablo aşağıdaki gibidir.

Tepe ler	S	<sup>*</sup> a	<sup>*</sup> b	<sup>*</sup> c	d	F
S	0	4	2	$\infty$	$\infty$	$\infty$
b	0	4	2	7	10	$\infty$
a	0	4	2	5	10	$\infty$
c	0	4	2	5	9	12

4. adım: d,F arasından minimum olan d=9

Tepe	S	<sup>*</sup> a	<sup>*</sup> b	<sup>*</sup> c	<sup>*</sup> d	F
S	0	4	2	$\infty$	$\infty$	$\infty$
b	0	4	2	7	10	$\infty$
a	0	4	2	5	10	$\infty$
c	0	4	2	5	9	12

$$\min(d, F) = 9$$

$$(d, F) \text{ için } D(F) = \min(12, 9+2) = 11$$

4. Adım sonrası yeni tablo aşağıdaki gibidir.

Tepe	S	<del>*</del> a	<del>*</del> b	<del>*</del> c	<del>*</del> d	F
		a	b	c	d	
S	0	4	2	$\infty$	$\infty$	$\infty$
b	0	4	2	7	10	$\infty$
a	0	4	2	5	10	$\infty$
c	0	4	2	5	9	12
d	0	4	2	5	9	11

5. adım: F son tepe F=11 için tablo aşağıdaki gibidir.

Tepe	S	<sup>*</sup> a	<sup>*</sup> b	<sup>*</sup> c	<sup>*</sup> d	F
S	0	4	2	$\infty$	$\infty$	$\infty$
b	0	4	2	7	10	$\infty$
a	0	4	2	5	10	$\infty$
c	0	4	2	5	9	12
d	0	4	2	5	9	11
F	0	4	2	5	9	<b>11</b>

S den diğer tüm tepelere en kısa uzaklıklar

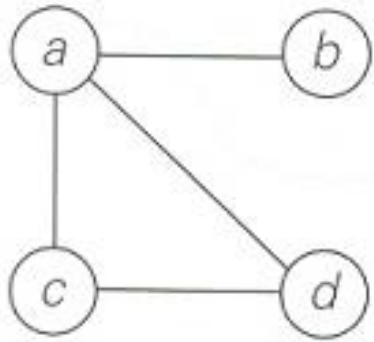
Tepeler	S'den en kısa yollar	Min. uzaklık
a	$S \rightarrow a$	4
b	$S \rightarrow b$	2
c	$S \rightarrow a \rightarrow c$	5
d	$S \rightarrow a \rightarrow c \rightarrow d$	9
F	$S \rightarrow a \rightarrow c \rightarrow d \rightarrow F$	11



# Bir Grafta Yolların Sayısı

- Tanım:

- Bir grafta iki köşe arasındaki farklı yolları sayma.
- Bir grafiğin  $i$ . tepe noktasından  $j$ . tepe noktasına  $k > 0$  uzunluğundaki farklı yolların sayısının,  $A^k$ 'nin  $(i, j)$ . Elemanına eşit olduğunu kanıtlamak kolaydır, burada  $A$ , grafiğin komşuluk matrisidir.



$$A = \begin{matrix} & \begin{matrix} a & b & c & d \end{matrix} \\ \begin{matrix} a \\ b \\ c \\ d \end{matrix} & \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} \end{matrix}$$



$$A^2 = \begin{matrix} & \begin{matrix} a & b & c & d \end{matrix} \\ \begin{matrix} a \\ b \\ c \\ d \end{matrix} & \begin{bmatrix} 3 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 2 \end{bmatrix} \end{matrix}$$

Bir graf

Grafın komşuluk matrisi A

$A^2$



A ve  $A^2$  matrislerinin elemanları 1 ve 2 uzunluklu yolların sayısını temsil eder.

- Böylece problem, komşuluk matrisinin uygun bir gücünü hesaplamak için bir algoritma ile çözülebilir.
- Problem matris üssüne indirgenmiştir.
- $A^k$  nasıl hesaplanır?

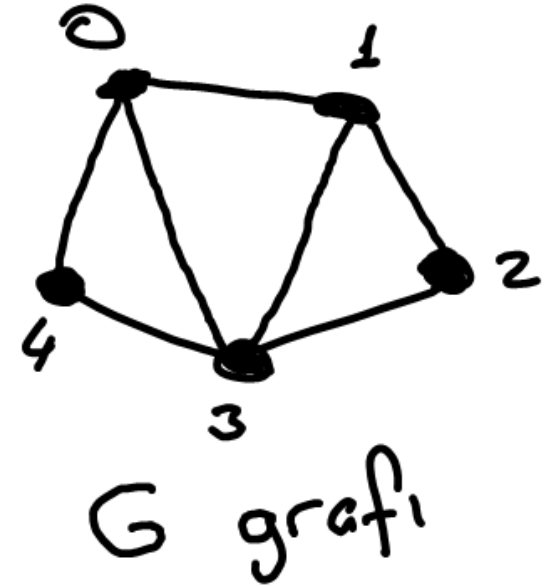
# C kodu: Yanda verilen G grafi için...

## Liste yöntemi ile veri girişi yapalım...

```
#include<stdio.h>
#include<conio.h>
int main()
{ int a[100][100],t[7][7]={0,0,0,1,1,2,3},{1,3,4,2,3,3,4}};
int d[100][100],v[100][100],s[100][100];
int i,j,k,m,top,b,h,f,enb,enk,n,s1,s2,s3,sum1,sum2,sum3;
```

```
m=1;
printf("G nin tepe sayisini giriniz: ");
scanf ("%d",&n);
```

```
for (i=0;i<n;i++)
{
for (j=0;j<n;j++)
{
a[i][j]=0;
}}
```



```
for (i=0;i<7;i++)
{
    a[t[0][i]][t[1][i]]=1;
    a[t[1][i]][t[0][i]]=1;
} // komşuluk matrisi elde edildi...
```


```
for ( i = 0; i <n; i++ ) {

    for ( j = 0; j <n; j++ )
        printf( "%d", a[ i ][ j ] );

    printf( "\n" );} // komşuluk matrisini yazdırdık...
```


```
for (i=0;i<n;i++)
{
    for (j=0;j<n;j++)
    {
        v[i][j]=a[i][j]; // komşuluk matrisinin bir kopyası alındı...
    }
}
printf( "\n" );
```

```
if (m==1)
{

for (i=0;i<n;i++)
{
for (j=0;j<n;j++)
{
if ((a[i][j]==1) || (i==j)) d[i][j]=a[i][j];
else d[i][j]=500; 
```

```
}}
```

```
} // komşuluk matrisi ilk adımda uzaklık matrisine atanıyor...
```

f=0; 

```
for (i=0;i<n;i++)
```

```
{
```

```
  for (j=0;j<n;j++)
```

```
  {
```

```
    if (d[i][j]==500) f=f+1;
```

```
  }
```

```
if (f==0) goto T; // sonsuz olan uzaklık var mı kontrol ediliyor...
```

L:

```
if (m>1) {
```

```
  for (i=0;i<n;i++)
```

```
  {
```

```
    for (j=0;j<n;j++)
```

```
    {
```

```
      if ((d[i][j]==500) && (a[i][j]==1)) d[i][j]=m;
```



```
    }
```

```
  }
```

```
}
```

```
for (i=0;i<n;i++)  
    {  
        for (j=0;j<n;j++)  
            {top=0;  
              for (b=0;b<n;b++)  
                  {  
                      top = (top + (a[i][b]*v[b][j]));  
                  }  
              s[i][j]=top;  
            }  
    }
```

```
for (i=0;i<n;i++)  
    {  
        for (j=0;j<n;j++)  
            {  
                if ((i==j) || (s[i][j]==0)) s[i][j]=0;  
                else s[i][j]=1;  
            }  
    }
```



```
for (i=0;i<n;i++)
{
    for (j=0;j<n;j++)
    {
        if (i==j) a[i][j]=0;
        else a[i][j]=s[i][j];}}

m=m+1;
goto L;
T:
printf ("\n");
printf ( "G nin uzaklik matrisi\n");
for (i=0;i<n;i++)
{
    for (j=0;j<n;j++)
    {
        if (d[i][j]==500) printf ("%d",0);
        else
            printf ("%d",d[i][j]);
    }printf( "\n" );}
getch();
return 0;
}
```

G nin tepe sayisini giriniz: 5

01011

10110

01010

11101

10010

G nin uzaklik matrisi

01211

10112

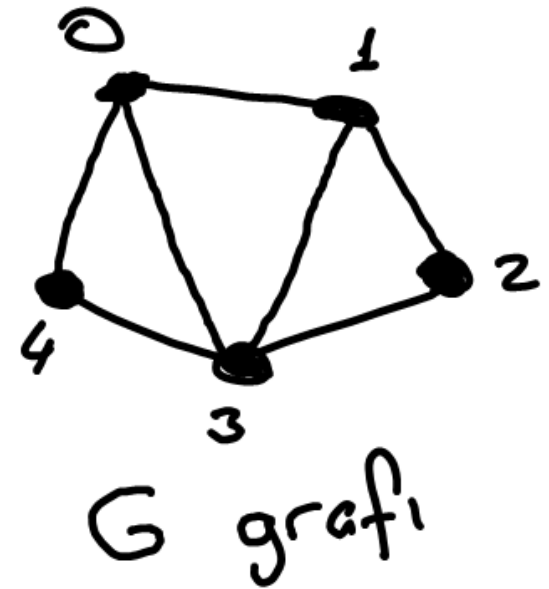
21012

11101

12210

-----  
Process exited with return value 0

Press any key to continue . . .



**Tanım 3:** Dışmerkezlilik (**eccentricity**), her tepenin diğer tepelere olan uzaklıklarının en büyük değeridir ve  $e(v)$  (veya  $e_G(v)$ ) biçiminde gösterilir. En büyük dışmerkezlilik değerine **çap (diameter)** denir,  $diam(G)$  biçiminde gösterilir. En küçük dışmerkezlilik değerine **yarıçap (radius)** denir ve  $r(G)$  biçiminde gösterilir.

**Tanım 4:** Dışmerkezlilik değeri yarıçapa eşit olan tepelere **merkez tepeler (central vertices)** denir.



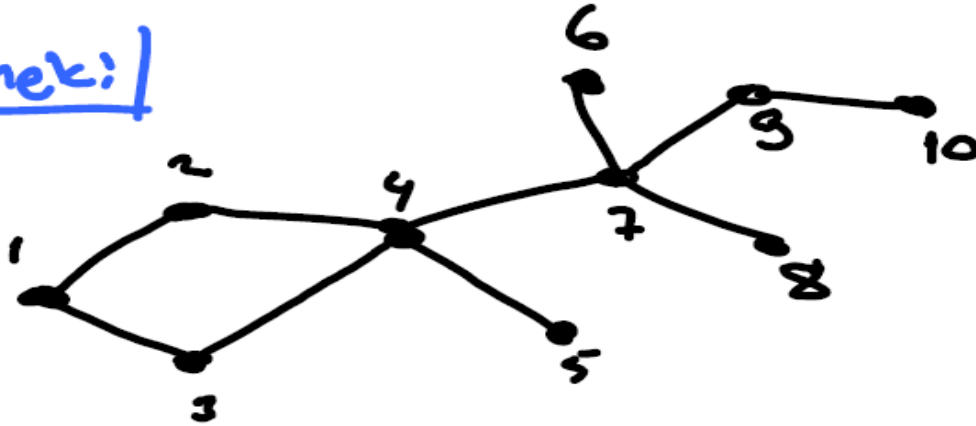
**Tanım 5:** Dışmerkezlilik değeri çapa eşit olan tepe veya tepelere **kıyı tepeler (peribheral vertices)** denir.



# Eccentricity (Distanz)

$$e(v) = \max \{ d(v, u) \mid v, u \in V(G) \}$$

Örnek:



$$e(1) = 5$$

$$e(2) = 4$$

$$e(3) = 4$$

$$e(4) = 3$$

$$e(5) = 4$$

$$e(6) = 4$$

$$e(7) = 3$$

$$e(8) = 4$$

$$e(9) = 4$$

$$e(10) = 5$$

Böylece;

$$\text{diam}(G) = 5 \quad (\text{Çap})$$

$$r(G) = 3 \quad (\text{Yarı çap})$$

Merkez tepeler;

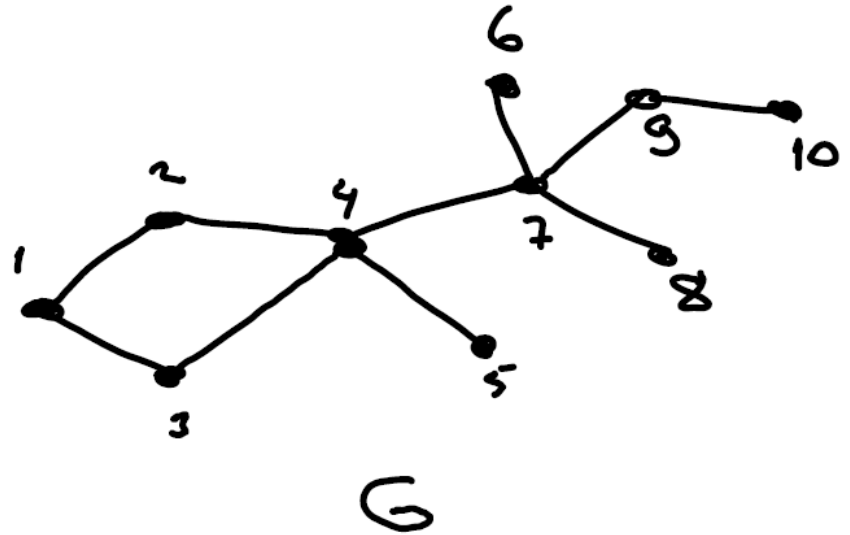
$$C(G) = \{v \mid e(v) = r(G)\}$$

$$C(G) = \{4, 7\}$$

Kıyı tepeler;

$$P(G) = \{v \mid e(v) = \text{diam}(G)\}$$

$$P(G) = \{1, 10\}$$



**C kodu:** Eccentricity için yukarıdaki koda aşağıdaki kod parçası eklenir.

```
for (i=0;i<n;i++)
{
    for (j=0;j<n;j++)
    {
        if (j==0) enb=d[i][j];
        if ( d[i][j] >= enb ) enb=d[i][j];
    }
    e[i][0]=enb;
```

```
printf ("\n");
printf ("Tum tepelerin eccenricity degerleri\n");

for (i=0;i<n;i++)
{
    for (j=0;j<1;j++)
    {
        printf ("%d---> %d",i+1, e[i][j]);

    }printf ("\n");}
```

G nin tepe sayisini giriniz: 5

01011

10110

01010

11101

10010

G nin uzaklik matrisi

01211

10112

21012

11101

12210

Tum tepelerin eccentricity degerleri

0---> 2

1---> 2

2---> 2

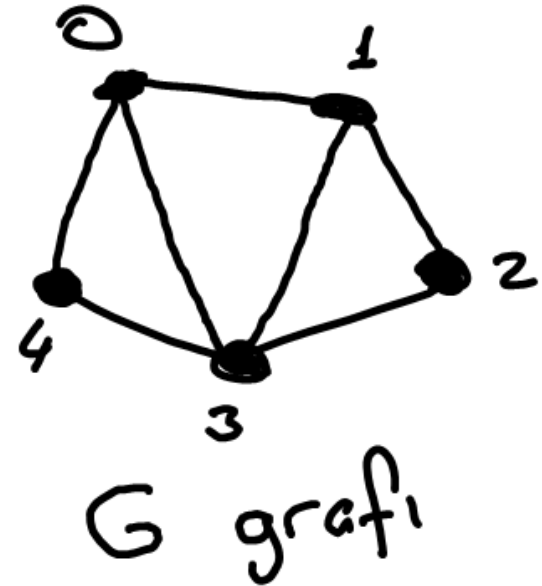
3---> 1

4---> 2

-----

Process exited with return value 0

Press any key to continue . . .



**Eccentricity deęerlerini kullanarak grafın apı, yarıapı,  
merkez ve kıyı tepeleri kolayca bulunabilir...**



## KAYNAKLAR

- [1] Chartrand, G.-Lesniak, L., (1986) : *Graphs and Digraphs*, Wadsworth & Brooks, California
- [2] West D.B. (2001) : *Introduction to Graph Theory*, Prentice Hall, USA.
- [3] Graf Teoriye Giriş, Şerife Büyükköse ve Gülistan Kaya Gök, Nobel Yayıncılık
- [4] Discrete Mathematical Structures for Computer Science, Ronald E. Prather, Houghton Mifflin Company, (1976).
- [5] Christofides, N., 1986. Graph Theory an Algorithmic Approach, Academic Press, London
- [6] ALGORİTMALAR (Teoriden Uygulamalara), Vasif V. NABİYEV, Seçkin Yayıncılık