Kıvanç Adıgüzel
191180003

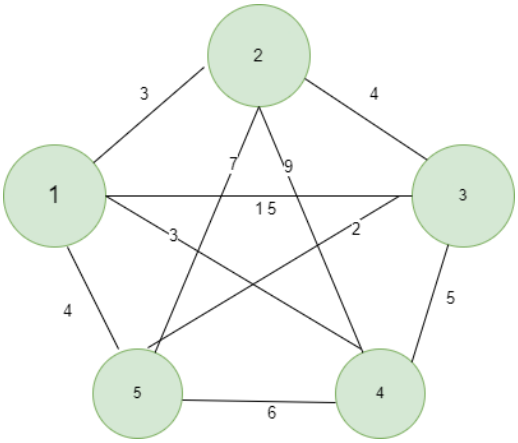
BM218 ALGORİTMALAR

ÖDEV-3

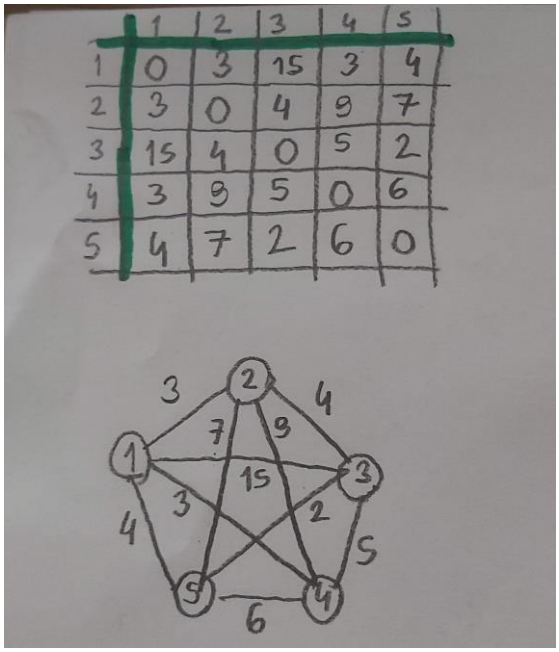
TRAVELING SALESMAN PROBLEM (TSP)

- ✓ $3 < N < 100$ şartına göre (en az 3 — en fazla 100 şehir için);
- ✓ **<u>TSP problem durumunu sağlamak şartı ile</u> N ve E sayılarına göre uygun rastgele bir çizgeler üretilmelidir.** $(genel\ kural: N \leq E \leq \frac{Nx(N-1)}{2})$

N adet şehir ve E adet bağlantı için TSP (*Gezgin Satıcı Probleminin — Traveling Salesman Problem*) probleminin Brute Force yaklaşımı ile çözülmesi beklenmektedir.

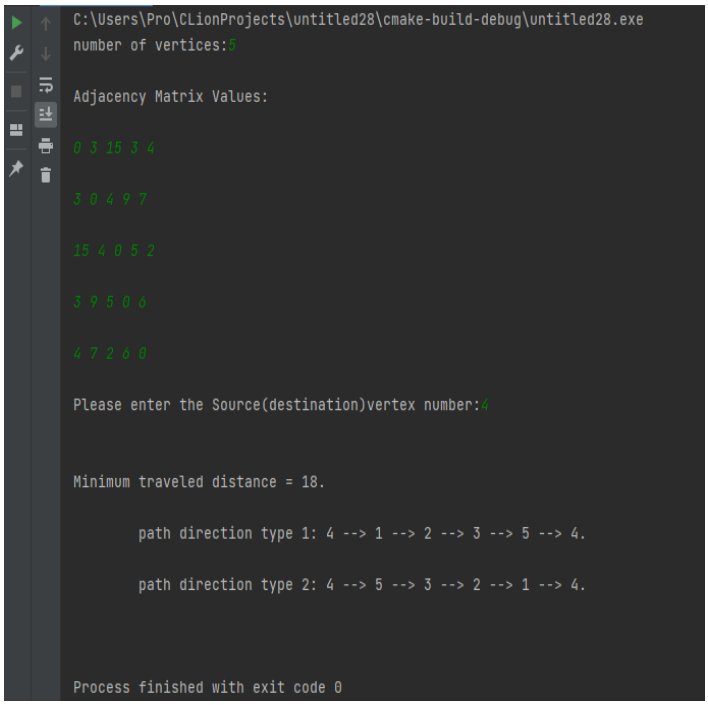


Kodumda kullanacağım şekil



Komşuluk matrisi gösterimi ve şeklin tekrar elle çizimi

Kıvanç Adıgüzel
191180003



ii. Listeleme adımı ve iii. En kısa yol bulma adımı

Sonuç olarak; tüm olasılıklar listelenir ve iki adet en iyi sonucu veren (optimal) liste kaydedilir.

TSP KODUM

```
#include<stdio.h>
void TSP(int*C, int*A, int*path, int*fpath, int *sum, int *fsum, int flag, int n, int b, int a, int *sc)
{
    //ingilizce yorum satırlarını bilinçli bir şekilde silmedim
    int i,k;
    flag++;
    for(k=0;k<n;k++)
        if(*(C+n*flag+k)==0)
        {  //Checking if any node of (flag+1)-th row is not already reached or not.
            *(C+n*flag+k)=k+1;   //Placing the new node in that Vertex.
            *sum=*sum+*(A+n*b+k);   //Updating total covered path distance.
            *(path+flag-1)=k;   //Adding the vertex to salesman path.

            /*Updating the Board w.r.t. the newly covered vertex*/
            if(flag<n){
                for(i=flag+1;i<n;i++)
                    *(C+i*n+k)=k+1;
            }
            /*Recursively call TSP function*/
            if(flag<n-1)
                TSP(C,A,path,fpath,sum,fsum,flag,n,k,a,sc);
            /*Storing new solution to 'fpath'-array if found.extra*/
            if(flag==n-1){
                *sum=*sum+*(A+n*k+a);
                if(*sum==*fsum){
                    *sc=*sc+1;
                    for(i=0;i<n-1;i++)
                        *(fpath+(*sc)*(n-1)+i)=*(path+i);   //Updating final path direction.
                }
                else if(*sum<*fsum){
                    *fsum=*sum; //Updating covered path distance.
                    *sc=0;
                    for(i=0;i<n-1;i++)
                        *(fpath+i)=*(path+i);   //Updating final path direction.
                }
                *sum=*sum-*(A+n*k+a);
            }

            /*Removing the previous node and undoing all its effect*/
            for(i=flag;i<n;i++)
                *(C+n*i+k)=0;
            *sum=*sum-*(A+n*b+k);   //Substructing last added path distance.
        }
}
int main(){
    int n,i,j,a,sc=0;
    printf("number of vertices:");
    scanf("%d",&n);
    int A[n][n],C[n][n],sum=0,fsum=9999,path[n-1],fpath[1000][n-1];
    printf("\nAdjacency Matrix Values:\n");
    for(i=0;i<n;i++){
        printf("\n");
        for(j=0;j<n;j++)
            scanf(" %d",&A[i][j]);
    }

    printf("\nEnter the Source(destination)vertex number:");
    scanf("%d",&a);
    for(i=0;i<n;i++)
        for(j=0;j<n;j++)
            C[i][j]=0;
    for(i=0;i<n;i++)
```

```
        C[i][a-1]=a;

    TSP(C,A,path,fpath,&sum,&fsum,0,n,a-1,a-1,&sc);

    printf("\n\nMinimum traveled distance = %d.",fsum);
    for(i=0;i<=sc;i++){
        printf("\n\n\tpath direction type %d: %d -->",i+1,a);
        for(j=0;j<n-1;j++)
            printf(" %d ---->",fpath[i][j]+1);
        printf(" %d.",a);
    }
    printf("\n\n");
}
```

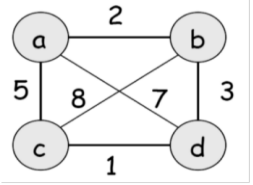
Örnek 2 - program çıktısı:

N = 4 için;

$N \leq E \leq \frac{Nx(N-1)}{2} \rightarrow 4 \leq E \leq \frac{4x(4-1)}{2} \rightarrow 4 \leq E \leq 6$ eşitliğinden rastgele bir şekilde:

E = 6 olsun. Buna göre N = 4 ve E = 6 için çizge oluşturulur ve ağılıklar rastgele atanır:

i. Çizge oluşturma adımı



ii. Listeleme adımı

a → b → c → d → a	2+8+1+7 = 18
a → b → d → c → a	2+3+1+5 = 11 ← optimal
a → c → b → d → a	5+8+3+7 = 23
a → c → d → b → a	5+1+3+2 = 11 ← optimal
a → d → b → c → a	7+3+8+5 = 23
a → d → c → b → a	7+1+8+2 = 18

Sonuç olarak; tüm olasılıklar listelenir ve iki adet en iyi sonucu veren (*optimal*) liste kaydedilir.

iii. En kısa yol bulma adımı:

En kısa yollar: a → b → d → c → a ve a → c → d → b → a

Komşuluk matrisi kullanarak(değerleri kullanıcıya girdirerek) ödevde istenenlere ulaştım.

1.adım komşuluk matrisi bölümünde yer almaktadır.

NOT: Komşuluk Matrisi kullanarak çözüm geliştirmek için izin aldım.