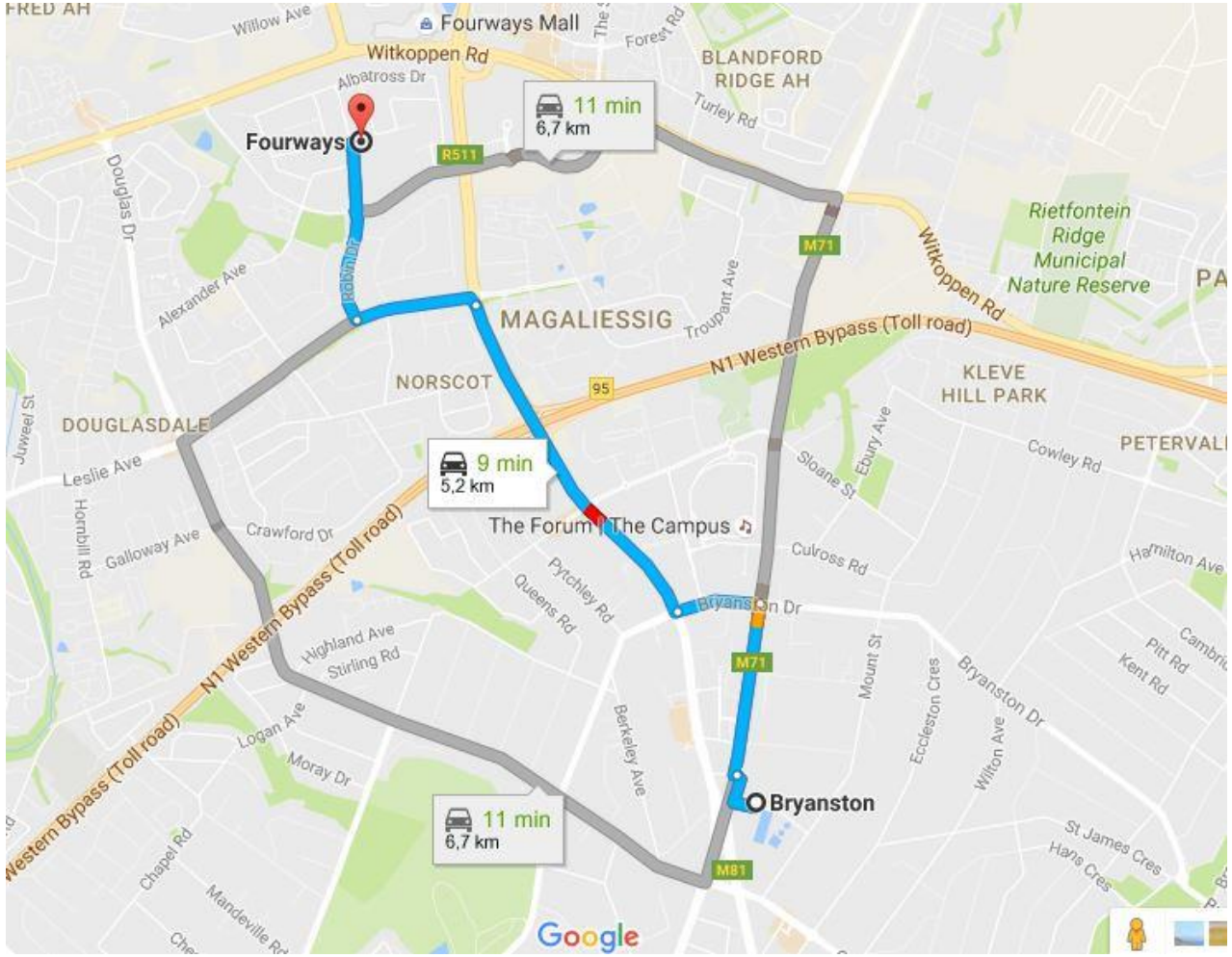
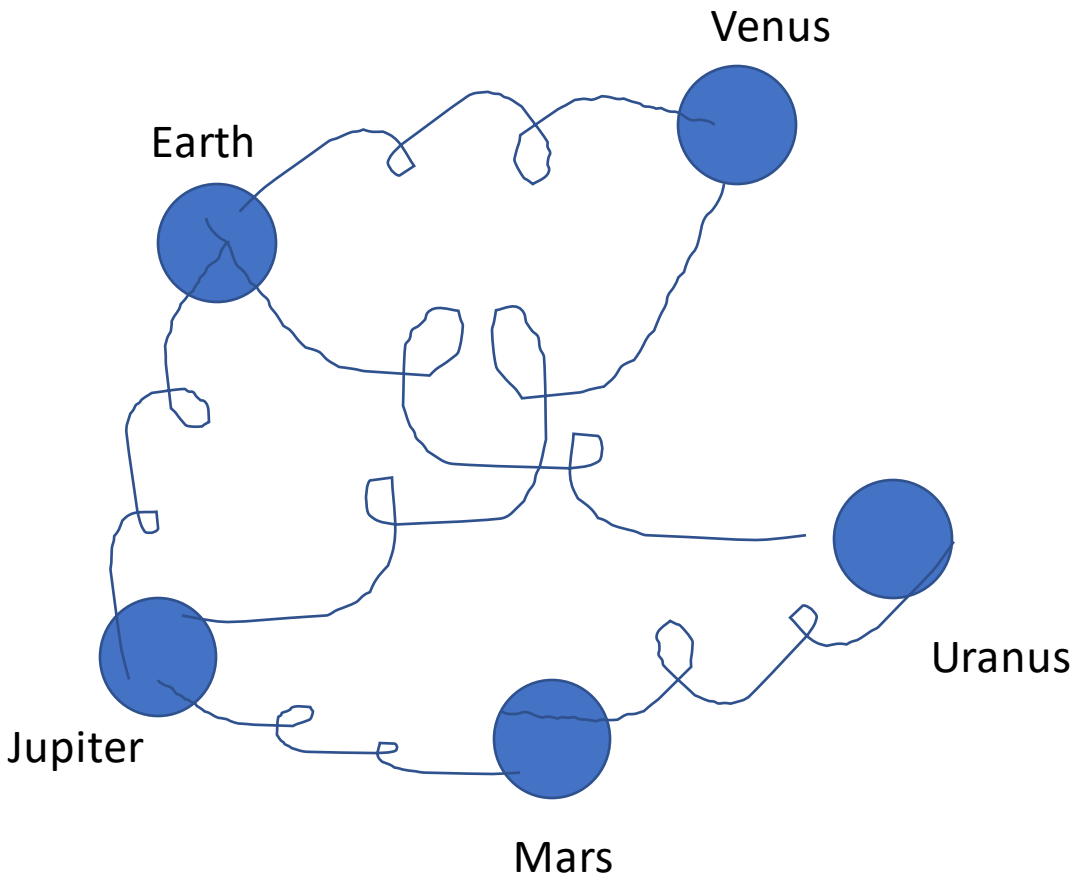
Wormhole problem

Muhamed_20180539@fci.Helwan.edu.eg


Fourways Mall
Witkoppen Rd
Albatross Dr
Fourways
11 min
6,7 km
R511
BLANDFORD RIDGE AH
Turley Rd
Rietfontein Ridge Municipal Nature Reserve
M71
Witkoppen Rd
MAGALIESSIG
Troupant Ave
N1 Western Bypass (Toll road)
NORSCOT
95
KLEVE HILL PARK
DOUGLASDALE
9 min
5,2 km
The Forum
The Campus
Sloane St
Ebury Ave
Cowley Rd
PETERVALE
Culross Rd
Hamilton Ave
Bryanston Dr
Pytchley Rd
Queens Rd
Highland Ave
Stirling Rd
Logan Ave
Moray Dr
Berkeley Ave
Mount St
Eccleston Cres
Wilton Ave
Bryanston
11 min
6,7 km
M81
St James Cres
Harris Cres
Chapel Rd
Mandeville Rd
Google


Venus
Earth
Uranus
Jupiter
Mars

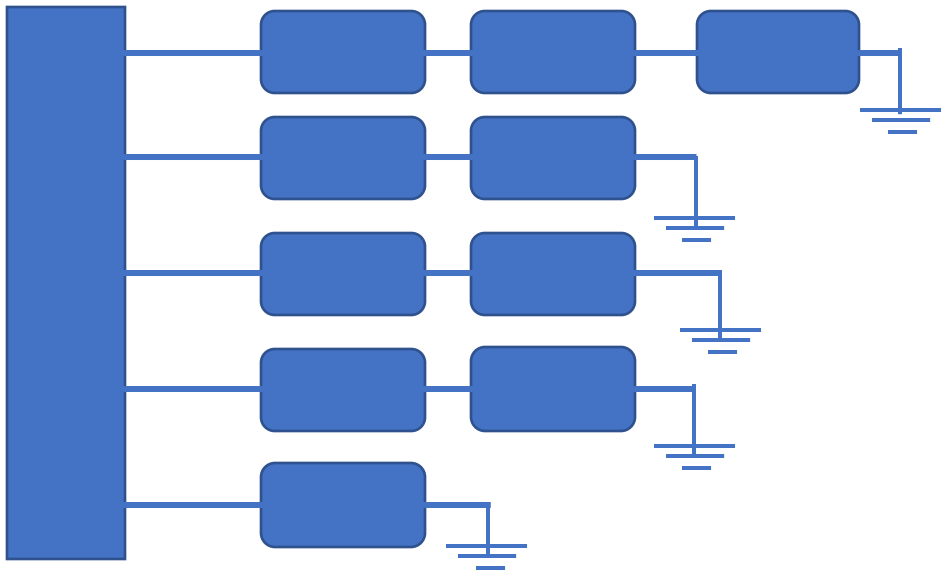
Data Structure

After reading the problem description I asksed my self, what type of data structures will I use to solve this problem !!

it's Graph !!!

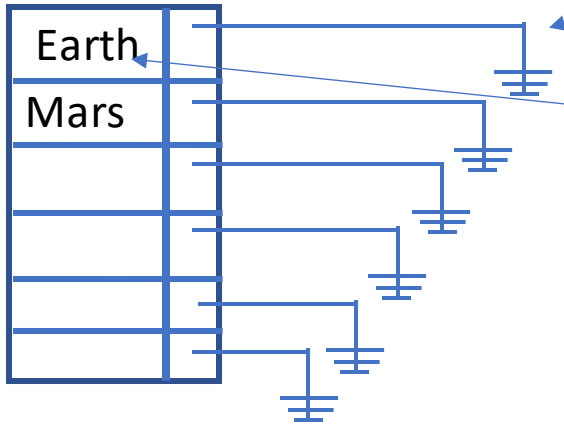
There are planets as Vertexes

And there are wormholes as Edges



```
1   #ifndef GRAPH_H_INCLUDED
2   #define GRAPH_H_INCLUDED
3   #include <stdlib.h>
4   #include<string.h>
5   #define maxvertex 1000
6   #define IN 9999
7   typedef char graphentry[25];
8
9
10  typedef struct edge{
11  int endpoint;
12  int cost;
13  struct edge *nextedge;
14  }Edge;
15
16  typedef struct vertex{
17  graphentry entry;
18  Edge * firstedge;
19  }Vertex;
20
21  typedef struct graph{
22   int n;
23   int E;
24   Vertex entry[maxvertex];
25  }Graph;
26
27
28
29  void creategraph(Graph *);
30  int dijkstra(Graph *,graphentry ,graphentry );
31  void AddEdge(Graph*,int,int,int);
32  void removeEdge(Graph * , graphentry, graphentry );
33  void AddVertex(Graph * , int ,graphentry);
34  void removeVertex(Graph * , graphentry);
35
36  #endif // GRAPH_H_INCLUDED
```

Graph methods

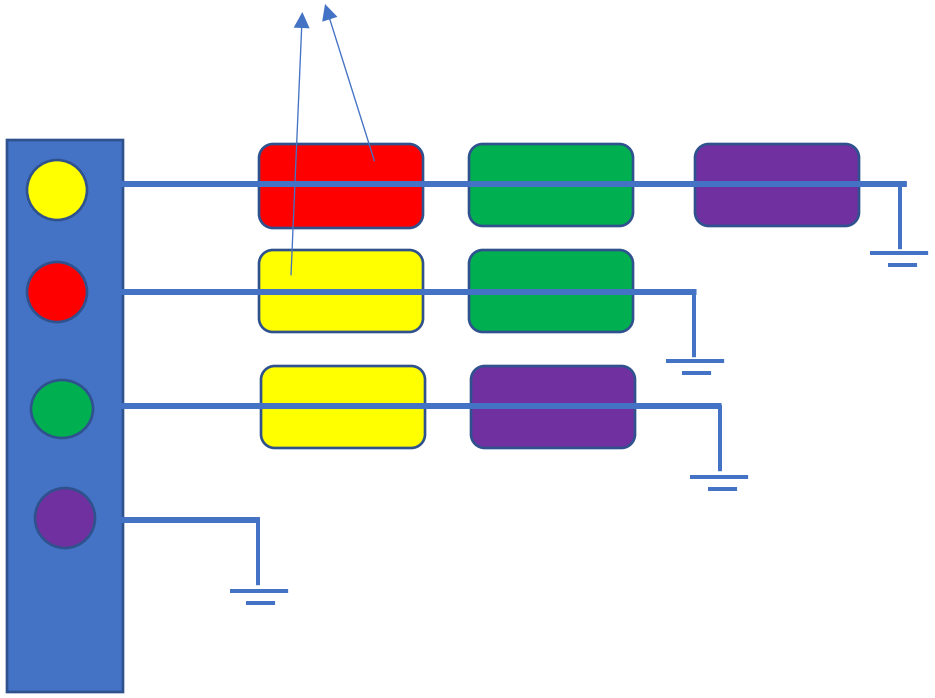



```
2
3     void creategraph(Graph *pg)
4     {
5         for (int i =0 ; i<maxvertex; i++)
6         {
7             pg->entry[i].firstedge=NULL;
8         }
9         pg->n=0;
10        pg->E=0;
11
12    }
```

```
131   void AddVertex(Graph *pg, int i,graphentry name)
132   {
133       strcpy(pg->entry[i].entry,name);
134       pg->n++;
135   }
136
137   void removeVertex(Graph *pg, graphentry vertex)
138   {
139       int vertec_index=findindex(&pg,vertex);
140       Edge *q=pg->entry[vertec_index].firstedge;
141       while(q)
142       {
143           removeEdge(&pg,pg->entry[vertec_index].entry,pg->entry[q->endpoint].entry);
144           q=q->nextedge;
145       }
146       pg->n --;
147   }
148
```

wormholes

Same Edge



```
90  void AddEdge(Graph *pg,int startpoint,int endpoint,int cost)
91  {
92
93      Edge *t = (Edge*)malloc(sizeof(Edge));
94      t->endpoint=endpoint;
95      t->cost=cost;
96
97      Edge *q = (Edge*)malloc(sizeof(Edge));
98      q->endpoint=startpoint;
99      q->cost=cost;
100
101     t->nextedge=pg->entry[startpoint].firstedge;
102     pg->entry[startpoint].firstedge=t;
103     q->nextedge=pg->entry[endpoint].firstedge;
104     pg->entry[endpoint].firstedge=q;
105     pg->E+=2;
106
107 }
```

```
108 void removeEdge(Graph *pg, graphentry start,graphentry end)
109 {
110     int temp;
111     int startpoint = findindex(&pg,start);
112     int endpoint = findindex(&pg,end);
113     for (int i=0; i<2; i++)
114     {
115         Edge *q=pg->entry[startpoint].firstedge;
116         Edge *t;
117         while(q->nextedge->endpoint != endpoint)
118             q=q->nextedge;
119
120         t=q->nextedge;
121         q->nextedge=q->nextedge->nextedge;
122         free(t);
123
124         temp = startpoint;
125         startpoint = endpoint ;
126         endpoint = temp;
127     }
128  pg->E-=2;
129 }
```

Main.c

1- Create graph

2- Bulid Graph

1)take the number of Vertexes from user

2)take name of vertexes

3)take all wormhole costs

```c
#include <stdio.h>
#include <stdlib.h>
#include "graph.h"

int main()
{
    Graph g;
    creategraph(&g);
    graphentry startpoint,target,newstartpoint;
    int shourtpath,n;
    int w,i,j;
    //welcome to my problem
    printf("\t\t\t\t\t\twormhole problem\n");
    printf("\t\t\t The Shortest Path between two planet( DIJKSTRA'S ALGORITHM) \n\n");
    //input the number of planets
    printf("\tEnter the number of planets :  ");
    scanf("%d",&n);
    graphentry vertexs[n];
    //input the names of planets
    printf("\tEnter your vertexs names.\n");
    for(int count=0 ; count<n; count++)
    {
        printf("\tVetex[%d]: ",count+1);
        scanf("%s",vertexs[count]);
        AddVertex(&g,count,vertexs[count]); // add  Vertex to graph
    }

    //inputs all time costs of wormholes
    for(i =0; i<n; i++)
    {
        for(j=0; j<n; j++)
        {
            if(j>i)
            {
                printf("\tthe Time_cost between %s and  %s: ",vertexs[i],vertexs[j]);
                scanf("%d",&w);
                if(w!=0)
                    AddEdge(&g,i,j,w); //add edge in my graph
            }
        }
    }
```

Main.c

3- take source and target planet from user

4- call Dijkstra Algorthim to find and print the shortest path and thi fun will return the time cost wich

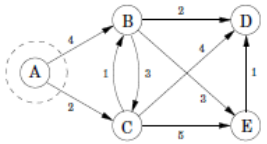
5- check if happen mal function take the new source vertex so go to step 5

End

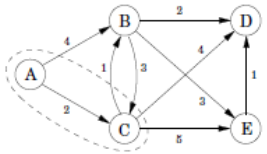
```
44      printf("graph has built\n");
45       printf("\n\tEnter your source : ");
46       scanf("%s",startpoint);
47       printf("\tEnter your target : ");
48       scanf("%s",target);
49       printf("\n\n\n");
50
51           shourtpath = dijkstra(&g,startpoint,target);//calling Dijkstra algorithm to find
52           printf("\n\n\n\t\t\t\tyour path will take : %d unit of time\n ",shourtpath); //ou
53           //malfunction part
54           printf("\n\n\t if happen malfunction press 1  or 0 if you arrived :");
55           scanf("%d",&i);
56           if(i)
57           {
58               printf("\tplease enter your location : ");
59               scanf("%s",newstartpoint);
60               printf("\n\n\n");
61               shourtpath = dijkstra(&g,newstartpoint,target); //calling Dijkstra algorithm
62               printf("\n\n\n\t\t\t\tyour path will take : %d unit of time ",shourtpath);//o
63           }
64
65       printf("\n\n\n\t\t\t\t\t\tthanks\n\n\n");
66
67  }
```

Dijkstra Algorithm

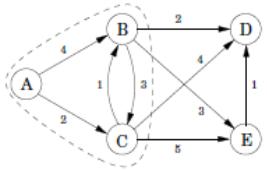
Find shortest path from one source



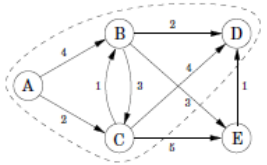
A: 0	D: ∞
B: 4	E: ∞
C: 2	



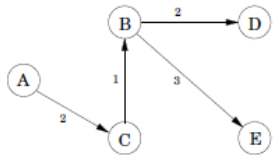
A: 0	D: 6
B: 3	E: 7
C: 2	



A: 0	D: 5
B: 3	E: 6
C: 2	



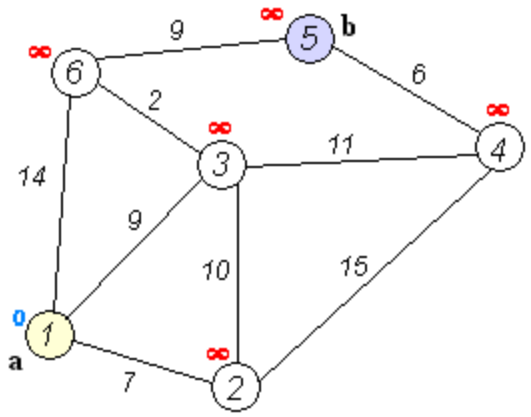
A: 0	D: 5
B: 3	E: 6
C: 2	




```
15  {
24  int dijkstra(Graph *pg,graphentry sourcename,graphentry targetname )
25  {
26      int start= findindex(pg,sourcename);
27      int target= findindex(pg,targetname);
28      Edge *t = pg->entry[start].firstedge;
29      int dist[pg->n],prev[pg->n],i,m,min,d,j=0;
30      int selected[pg->n];
31      graphentry path[pg->n];
32      for(i=0; i< pg->n ; i++)  //complexity O(v)
33      {
34          dist[i] = IN;
35          prev[i] = -1;
36          selected[i]=0;
37      }
38      selected[start]=1;
39      dist[start] = 0;
40
```

```
int findindex(Graph *pg,graphentry point)
{
    for(int i=0; i< pg->n; i++)
    {
        if (strcmp(pg->entry[i].entry,point)==0)
        {
            return i ;
        }
    }
}
```

Core part




```
41      while(!selected[target]&& j++ !=pg->n)//complexity sO(v) path on all Vertex
42      {
43          min = IN;
44          m = 1;
45          while(t)  //complexity O(v-1) in case of complete graph
46          {
47              i=t->endpoint;
48              d = dist[start] +t->cost;
49              if(d< dist[i]&&(selected[i]==0))
50              {
51                  dist[i] = d;
52                  prev[i] = start;
53              }
54              if(min>dist[i] && (selected[i]==0))
55              {
56                  min = dist[i];
57                  m = i;
58              }
59
60              t=t->nextedge;
61          }
62          start = m;
63          t=pg->entry[start].firstedge;
64          selected[start] =1;
65      }
66      if (selected[target]){
67          pathshow(pg,prev,path,target);
68      }else{
69         printf("\t\t\t\tthe target is isolated planet for your location");
70      }
71
72      return dist[target];
73  }
```

Show the shortest path

```
73  void pathshow(Graph *pg,int *prev,graphentry *path,int start ){
74
75      int j=0;
76      while(start != -1) //complexity O(v)
77      {
78          strcpy(path[j++],pg->entry[start].entry);
79          start = prev[start];
80      }
81
82      printf("\t\t\t\tthe best path from %s",path[--j]);
83      for(int i = j-1 ; i>=0; i--)
84      {
85          printf(" --to--> %s",path[i]);
86      }
87
88  }
```

Record

1- what is wormhole problem ? (00:00:00 to 00:37:00)

2- why I used Graph as data Structuer? (00:37:00 to 00:53:00)

3- why undirected graph ? (00:53:00 to 01:11:00)

4-why adjacent list not matrix ? (01:11:00 to 02:39:00)

5-why I used Dijkstra algorithm & what is Dijkstra ? (02:39:00 to 03:32:00)

6- My solution to solve this Ploblem (03:32:00 to 05:16:74)

Link : 20180539_voice