

Wormhole Problem

Part 1

A) The inhabitants of a far off star system have developed a transportation system between their planets. There are n planets in the system. Some planets have a wormhole between them, which allows you to travel between planet i to planet j in $t_{i,j}$ seconds. All times $t_{i,j}$ are positive (you can't travel back in time). Some pairs of planets may not have a wormhole between them. You are currently at planet 0 and wish to reach planet $n - 1$. However, your ship's navigation commands are starting to malfunction. When you are at a planet and tell the ship to go through a certain wormhole, it may misinterpret the command and instead choose a completely random outgoing wormhole instead. Luckily, you know that you that this will happen at most once. You don't know when the malfunction might happen, but you are going to assume it will happen at the most inconvenient time. You want to choose a strategy that will minimize your total travel time in the very worst case (no matter when the malfunction happens or which outgoing wormhole it takes you on). Design an efficient algorithm for this. Specifically, your algorithm should output at what time you can be guaranteed to reach planet $n-1$, if we use the optimal strategy. You don't need to prove your algorithm correct or justify the stated running time. You may assume it's possible to reach planet $n-1$ from any other planet via some sequence of wormholes.

B) In the space there are a graph which has n planets (Vertexes) and those vertexes connected with each other by E wormholes (edges), each wormhole connect two planets with each other, but some pairs of planets may not have a wormhole between them ,so if I want to travel from planet 1 to planet $n-1$ I will path through sequence of Wormholes , and this will take some units of time (cost or weight).

The problem is how to find the shortest path between two planets and how many units of time this path will take.

I will use Dijkstra algorithm to solve this problem, it will help me to find the shortest path from start planet to all other planets.

C) And if happen malfunction in my ship that makes some changes on the right shortest path which lead to target planet, then we should find the new shortest path from the planet which happen in this Malfunction, and find new time cost which that new path will take.

Part 2: Pseudo code

Dijkstra (*pg, source name, target name)

Start <-Index (source name) //initialize

Target <-Index (target name) //initialize

T<- First edge in source vertex //initialize

Int l , m, min, d, j =0; //some counters and used variable

For i=0 to graphSize do //initialize

 dist[i]<-Infinite

 prev[i]<- (-1)

 selected[i] <- 0

END FOR

selected [start] <- 1

dist [start] <- 0

WHILE Not selected[target] and j++ not equal graph size do // j visited planet counter

```

Min <- Infinte // infinite = 9999
M <- 1
WHILE t not equal null do
    i =t <- endpoint
    d=dist[start] + t -> cost
    IF d < dist [i] and not selected [i] then
        dist[i] <- d
        prev[i] <- start
    ENDIF
    IF min > dist [i] and selected [i] equal 0 then
        min <- dist [i]
        m <- i
    ENDIF
    t = t -> nextedge
ENDWHILE
start <- m
t = pg -> entry[start].firstedge
selected [start] <- 1
ENDWHILE
If selected[target] then
    Pathshow(); //printing the shortest path
else
    Print 'the target Is isolated from your location'
End if

```

Return MinimumTimeCost

Part 3: Time complexity

The core part is two while loops, the first while loop moving between vertexes and inside this loop there are another while loop it moves in all vertex's Edges

The core part in my algorithm

```
WHILE !selected[target] && j++ != pg->n do // first loop
```

```
    Min <- ln
```

```
    M <- 1
```

```
    WHILE t not equal null do //the second loop
```

```
        i = t <- endpoint
```

```
        d = dist[start] + t -> cost
```

```
        IF d < dist [i] and selected [i] equal 0 then
```

```
            dist[i] <- d
```

```
            prev[i] <- start
```

```
        ENDIF
```

```
        IF min > dist [i] and selected [i] equal 0 then
```

```
            min <- dist [i]
```

```
            m <- i
```

```
        ENDIF
```

```
        t = t -> nextedge
```

```
    ENDWHILE
```

```
    start <- m
```

```
    t = pg -> entry[start].firstedge
```

```
    selected [start] <- 1
```

The worst case: In case of complete graph

That's make each vertex has $E = (V-1)$.

Note: E->number of Edge

, V->number of vertexes

In The first loop:

The worst case happens when
the algorithm moving to all vertexes
to calculate the shortest path so the
time complexity = $O(v)$

In the second loop:

Each vertex has $(v-1)$ edges (worst case)

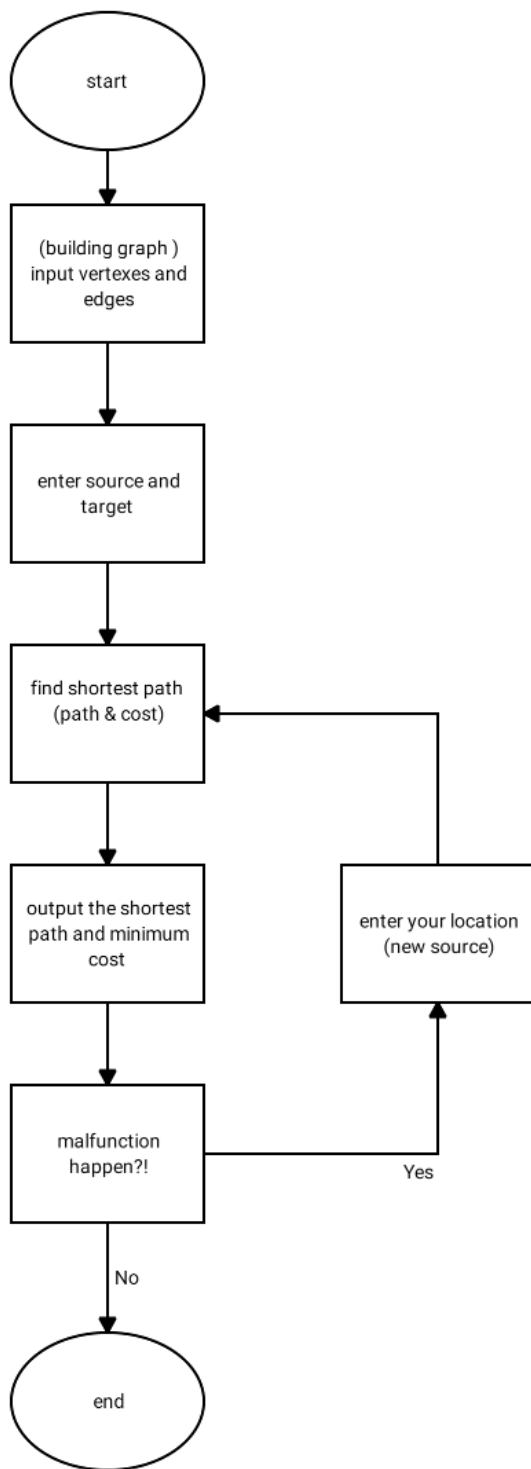
$$\sum_{k=1}^{V-1}(1) = (V-1)-1+1 = O(V-1)$$

The final complexity of my algorithm

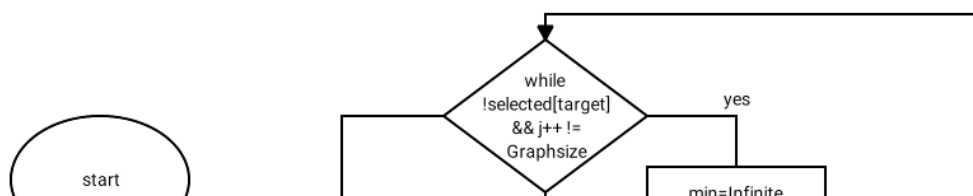
$$= O(V(V-1)) = O(V^2 - V) = O(V^2)$$

Part 4 : A) high level design

Block diagram



Dijkstra FlowChart



c) Snapshot from Result

"C:\Users\lenovo\first dsproject\algorithmstask\bin\Debug\algorithmstask.exe"

```
wormhole problem
The Shortest Path between two planet( DIJKSTRA'S ALGORITHM)

Enter the number of planets : 8
Enter your vertexs names.
Vetex[1] :Mercury
Vetex[2] :Venus
Vetex[3] :Earth
Vetex[4] :Mars
Vetex[5] :Jupiter
Vetex[6] :Saturn
Vetex[7] :Uranus
Vetex[8] :Neptune
enter the cost between planet Mercury to planet Venus : 41
enter the cost between planet Mercury to planet Earth : 67
enter the cost between planet Mercury to planet Mars : 34
enter the cost between planet Mercury to planet Jupiter : 0
enter the cost between planet Mercury to planet Saturn : 69
enter the cost between planet Mercury to planet Uranus : 24
enter the cost between planet Mercury to planet Neptune : 78
enter the cost between planet Venus to planet Earth : 58
enter the cost between planet Venus to planet Mars : 62
enter the cost between planet Venus to planet Jupiter : 64
enter the cost between planet Venus to planet Saturn : 5
enter the cost between planet Venus to planet Uranus : 45
enter the cost between planet Venus to planet Neptune : 81
enter the cost between planet Earth to planet Mars : 27
enter the cost between planet Earth to planet Jupiter : 61
enter the cost between planet Earth to planet Saturn : 91
enter the cost between planet Earth to planet Uranus : 95
enter the cost between planet Earth to planet Neptune : 42
enter the cost between planet Mars to planet Jupiter : 27
enter the cost between planet Mars to planet Saturn : 36
enter the cost between planet Mars to planet Uranus : 91
enter the cost between planet Mars to planet Neptune : 4
enter the cost between planet Jupiter to planet Saturn : 2
enter the cost between planet Jupiter to planet Uranus : 53
enter the cost between planet Jupiter to planet Neptune : 92
enter the cost between planet Saturn to planet Uranus : 82
enter the cost between planet Saturn to planet Neptune : 21
enter the cost between planet Uranus to planet Neptune : 16

Enter your source : Earth
Enter your target : Uranus
```

This is the inputs part

The User should enter the number of planets (Vertexes)

Then the name of each planet

Then all Time costs for each wormhole. //note that 0 value means no wormhole exists

and after building the graph,

User should enter his location and his target planet.

```
Graph has built..
Enter your source : Earth
Enter your target : Uranus

the best path from Earth --to--> mars --to--> Neptune --to--> Uranus

your path will take : 47 unit of time

if happen malfunction press 1 or 0 if you arrived :1
please enter your location : Saturn

the best path from Saturn --to--> Neptune --to--> Uranus

your path will take : 37 unit of time

thanks

rocess returned 0 (0x0)  execution time : 18.628 s
ress any key to continue.
```

After inputs part the program will find the minimum Time-cost and shortest path between the start planet and the target planet.

b) system performance

The input is the new start planet which happen in that malfunction so the program will find the minimum Time cost and shortest path between the new start planet and the target planet.

thanks