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 ti, j seconds. All times ti, 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 I , 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 <- In
    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)

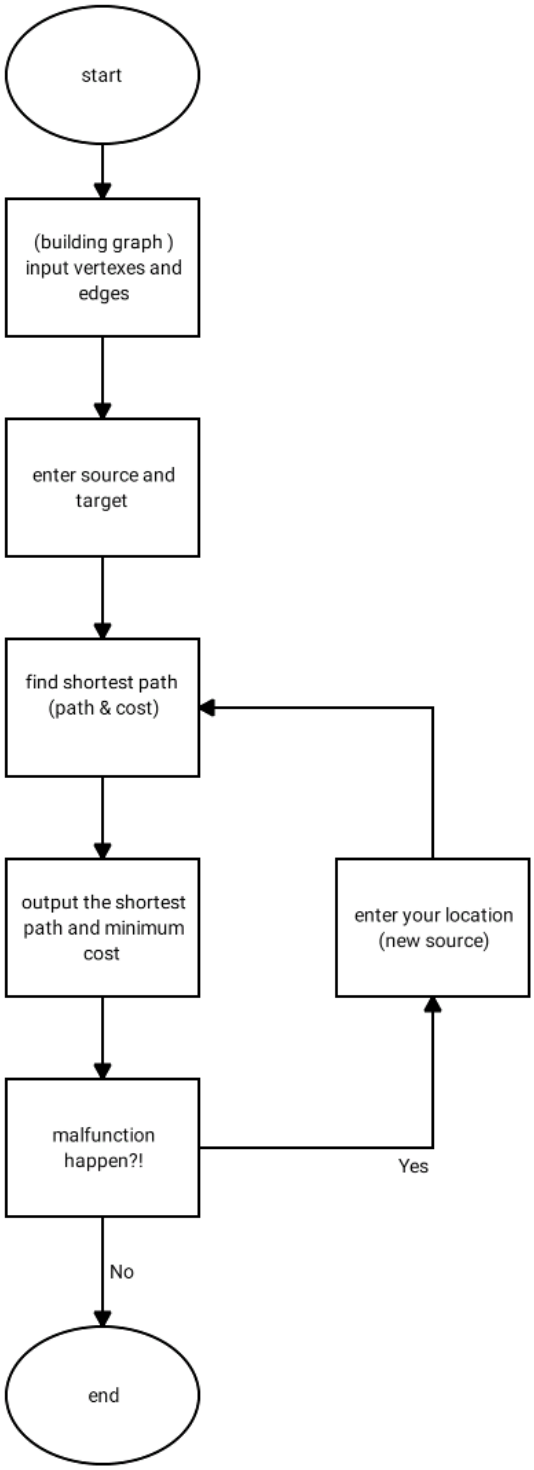
$\sum_{k=1}^{V-1}(1)$ = (V-1)-1+1 = O(V-1)

The final complexity of my algorithm

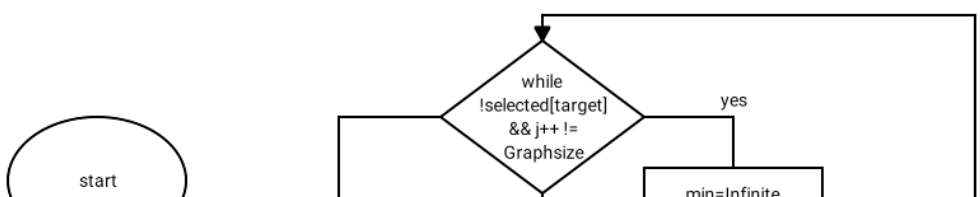
= O(V (V-1))= O($V^2 - V$) = O(V^2)

Part 4 : A) high level design

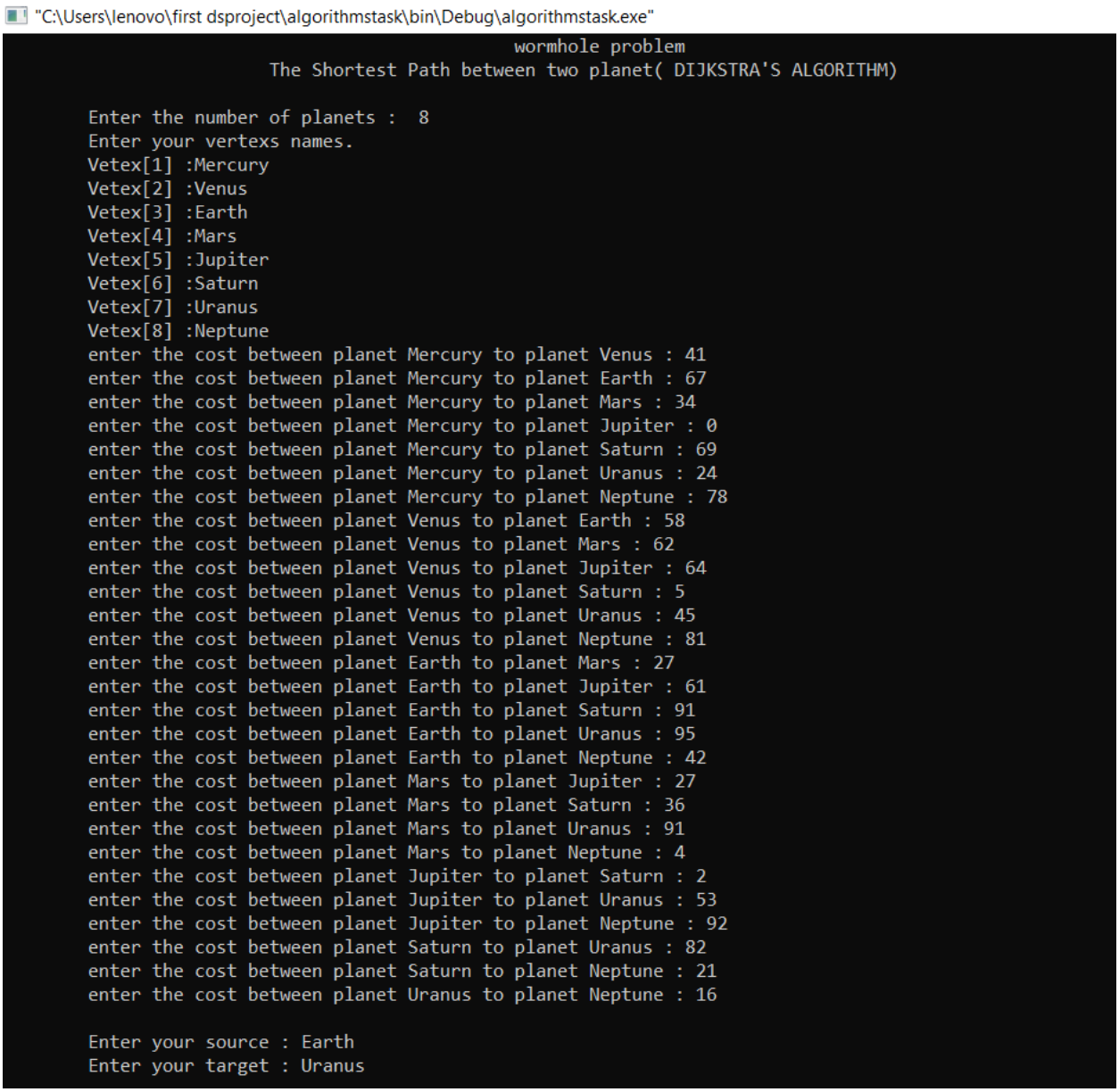
Block diagram



Dijkstra FlowChart



c) Snapshot from Result



This is the inputs part

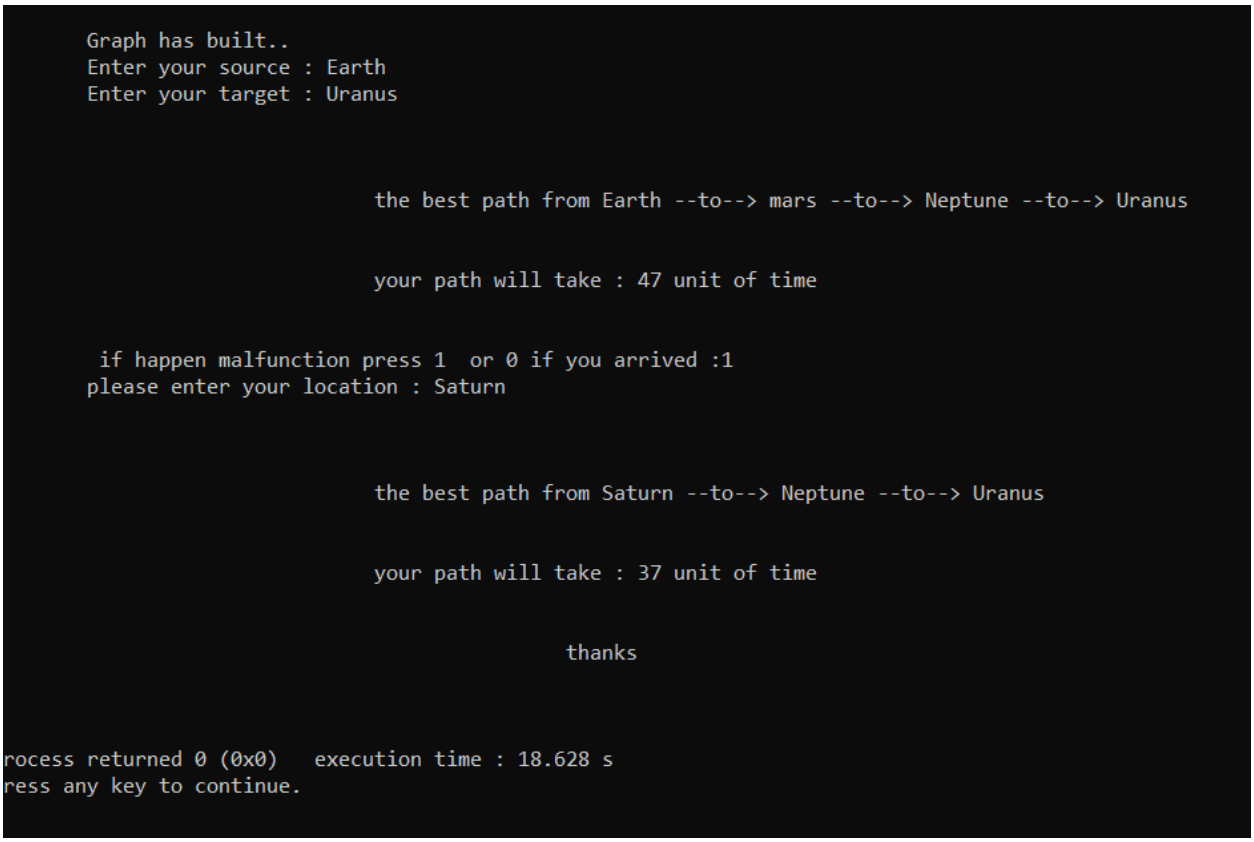
The User should enter the number of planets (Vertexes)

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.



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

In case of malfunction

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

b) system performance

time complexity of my all project is equal my algorithm equal O(v^2)