CENG 315 THE3 REPORT

NAME,SURNAME: Furkan Doğan

ID : 2098937

1) Proposed Algorithm & Pseudocode

To solve this problem, I choose Dijkstra shortest path algorithm. After I get inputs from the3.inp I am creating adjacency matrix and converting it to adjacency list to save the graph (I do it so because there could occur case of multiple graphs, I choose minimal edge from multiple edges).Then I am looking for all permutations of Vertices which I need to visit and at the ok function I am calculating all the possible permutations which are okay like (First Room -> Second Ammo Room -> Key Room -> First Ammo Room-> Scientist Room -> Chamber Room) and I am applying Dijkstra for each subsequent pair of vertices from given permutation. At Dijkstra I am creating two sets to save distances for each of Even and Odd times. Then I take values from beginnings of sets and perform relaxations. I am also saving the parent of the current Vertex to restore the path. Also, I am using Vector to keep if I have key to visit Scientist Room or Chamber Room if I don’t have key I am not going from there. After Dijkstra I am using restored path to calculate final ammo and if it is better than best ammo, I update it.

**function** ok(permutation)

temp = {}

for each permutation

if permutation is none ammo

temp += permutation

if temp sorted

return true

else

false

**function** Dijkstra(source,destination,keys,starting\_time(0 for even,1 for odd))

Initialize all the distances to MAX\_VALUE

Create to priority queue for odd and even times

if starting\_time is 1

odd distance to source from source is 0

else

even distance to source from source is 0

while odd or even queue is not empty

if non-empty odd queue

u = minimum of odd queue

for each neighbor v of u

if odd distance from u to v + weigth(u,v) is better than even distance to v

perform relaxation and set even parent of v to be u

if non-empty even queue

u = minimum of even queue

for each neighbor v of u

if even distance from u to v + weigth(u,v) is better than odd distance to v

perform relaxation and set odd parent of v to be u

if even distance to destination is less than odd distance

if even distance to destination not equal to infinity

restore path from even distance using even and odd parents

return (path,even end time)

else

if odd distance to destination not equal to infinity

restore path from even distance using even and odd parents

return (path,even end time)

return (empty path,random time)

**function** main()

read from the3.inp

create adjacency matrix

create adjacency list from adjacency matrix

create permutation list

best\_ammo = 0

best\_path = {}

map permutation list to list of {first room,key room,scientist room,chamber room,ammo1,ammo2}

do

{

Keys= {}

Path = {}

Next\_time = 1

If ok(permutation)

For each i in permutation

Path,Next\_time = Dijkstra(i,i+1,keys,next\_time)

If key found

Save the key

Calculate final ammo using the path

If final ammo is better than best ammo

best\_ammo = final\_ammo

best\_path = path

}while(next permutation exist)

Return best\_ammo,size of best\_path,best\_path

2) Complexity Analysis

Since we calculate all permutation and apply Dijkstra for okay ones, we have 56 permutations need to be considered. For Dijkstra since we use set so the complexity is O(i\*logV) where i is number of edges and V is number of vertexes. So our complexity for worst case is O(56\*5\*i\*logV) where 56 is number of permutations 5 is number of Dijkstra to apply for each permutations and it is equal to Θ (i\*logV).