**REPORT**

Project 1 :- **Savvy Traveler**

Group Member:

Anant Jain - [jain.anant@csu.fullerton.edu](mailto:jain.anant@csu.fullerton.edu)

Harsh Bodgal - [harsh481997@csu.fullerton.edu](mailto:harsh481997@csu.fullerton.edu)

Hemil Prajapati - [hemilhp22@csu.fullerton.edu](mailto:hemilhp22@csu.fullerton.edu)

Graphical user interface, text

Description automatically generated

**Pseudocode for given project description part 1:-**

edges, probability,start,end (variable used to iterate all the nodes )

src, dst (source, destination)

##taking a list, to store the nodes and there probability of a given example,

## for loop to visit each node and taking the probability from source to destination and storing that in queue

g = default(list)

for i in range(len(edges)):

src, dst = edges[i][0], edges[i][1]

prob = probability[i]

## appending the probability of destination from source to list g()

g[src].append((dst, prob))

g[dst].append((src, prob))

## using double ended queue to insert and remove the value of a particular nodes

q = deque()

## comparing the nodes’ probability with its parent probability

## initializing the maximum probability

maxProb = defaultdict(float)

parent = defaultdict(int)

parent[start] = start

## while loop to remove the minimum probability than the maxProb

while q:

node, prob = q.popleft()

if maxProb[node] > prob:

continue

else :

maxProb[node] = prob

for adj, nextProb in g[node]:

if maxProb[adj] < (prob \* nextProb):

parent[adj] = node;

q.append((adj, (prob \* nextProb)))

path = []

while(end!=parent[end]):

path.append(end)

end = parent[end];

## explicitly assigning the probability of each nodes

edges =[[0, 1], [0, 2], [0, 3], [1, 2], [1, 4], [1, 5], [2, 5], [3, 5], [3, 6], [4, 5], [4, 7], [5, 6], [5, 7], [6, 7]]

probability = [0.8, 0.7, 0.9, 0.8, 0.6, 0.6, 0.9, 0.6, 0.8, 0.8, 0.6, 0.7, 0.7, 0.9]

start = 5

end = 0

**Pseudocode for given project description part 2:-**

## taking source and destination

## explicitly initializing the nodes and the probability

point\_edges = [['A','B'], ['B','C'], ['A','C'], ['A','D'], ['D','G'], ['G','H'], ['E','H'], ['B','E'], ['C','F'],['B','F'],['D','F'],['B','C'],['E','F'],['G','F']]

probability = [0.8,0.8,0.7,0.9,0.8,0.9,0.6,0.6,0.9,0.6,0.6,0.8,0.8,0.7]

edges = [] #contains edges in terms of 0,1,2..

points = [] #contains unique Nodes in terms if its Name

nodes = [] #Contains unique Nodes in terms of 0,1,2..

count = 0 # to count unique Nodes

## for loop to visit each node and taking the probability from source to destination and storing that in queue

g = default(list)

for i in range(len(edges)):

src, dst = edges[i][0], edges[i][1]

prob = probability[i]

## appending the probability of destination from source to list g()

g[src].append((dst, prob))

g[dst].append((src, prob))

## using double ended queue to insert and remove the value of a particular nodes

q = deque()

## mapping nodes to its index to find the maximum probability of a particular nodes

for i in point\_edges:

for j in i:

# steps performed if array is not empty

# inserts node and its index in points and nodes respectivelyt if not already exists

if len(points) > 0:

flag = 0

for k in points:

if k == j:

flag = 1

if flag == 0:

points.append(j)

nodes.append(count)

count = count + 1

# appends when array is empty (at index 0)

else:

points.append(j)

nodes.append(count)

count = count + 1

# loops into nodes and calls fuction for every node and finds max probability path.

for w in range(len(nodes)):

result = 0

for i in range(len(nodes)):

if i != w:

start = i

end = w

result = result + solve(edges, probability, start, end)

if result > max:

max = result

maxIndex = w

print(points[maxIndex])

**Description on how to run the code:-**

**To run \*.py files(max\_prob\_module1\_ex1.py,** **max\_prob\_module1\_ex2.py,** **max\_prob\_module2\_ex1.py,** **max\_prob\_module2\_ex2.py,** **max\_prob\_module3\_ex1.py,** **max\_prob\_module1\_ex2.py), are the input files that contain the appropriate code for the project followed by computing the most reliable city which can be executed as python “filename.py” into the command line then hit enter.** **Else, we can also use the IDEs like Jupyter, Pycharm, Spyder, Atom to execute the code by hitting the run command. Steps: User have to provide the graph detailed inputs for each of the above-mentioned python files and run the files and desired output is shown in the output window.**

**Code and Output for example 1, part 1 given in the project document :-**

Graphical user interface, text, application

Description automatically generated

**Code and Output for example 1, part 2 given in the project document :-**

Graphical user interface, text, application, Word

Description automatically generated

**Code and Output for example 2, part 1 given in the project document :-**

**Graphical user interface, text, application, Word

Description automatically generated**

**Code and Output for example 2, part 2 given in the project document :-**

**Graphical user interface, text, application, Word

Description automatically generated**

**Code and Output for example 3, part 1 given in the project document :-**

**Graphical user interface, text, application, Word

Description automatically generated**

**Code and Output for example 3, part 2 given in the project document :-**

Graphical user interface, text, application, Word

Description automatically generated