DESIGN AND ANALYSIS OF ALGORTIHMS

CIA-1

HARSHATH

21011102037

## PRIMS ALGORTIHM

The given problem cannot be solved using prims algorithm because it is a directed graph.

In Prims algorithm we consider graphs in which traversal from one node to any other node is possible in order to find the minimum spanning tree, ie, the graph should be connected In the given graph we can observe that it is not possible to traverse from node D to any other node. Therefore prims algorithms cannot be used to solve directed graphs.

## KRUSKALS ALGORTIHM

For each step in kruskals algorithm we check if the minimum spanning forms a cycle or not. in directed graph only closed loops are said to form cycles but the kruskals algorithm assumes the MST to form cycles even if the loops aren’t closed.

## DIJSKSTRAS ALGORTIHM

DIJSKSTRAS algorithm works fine with directed graphs .

import sys

print("DIJSKSTRAS ALGORITHM")

vertices = [[0, 1, 1,0 , 0],

[0, 0, 1, 1, 1],

[0, 1, 0, 1, 1],

[0, 0, 0, 0, 0],

[0, 0, 0, 1, 0],

]

edges = [[0, 4, 2, 0, 0],

[0, 0, 3, 2, 3],

[1, 2, 0, 4, 5],

[0, 0, 0, 0, 0],

[0, 0, 0, 5, 0],

]

def to\_be\_visited():

global visited\_and\_distance

v = -10

for index in range(num\_of\_vertices):

if visited\_and\_distance[index][0] == 0 \

and (v < 0 or visited\_and\_distance[index][1] <=

visited\_and\_distance[v][1]):

v = index

return v

num\_of\_vertices = len(vertices[0])

visited\_and\_distance = [[0, 0]]

for i in range(num\_of\_vertices-1):

visited\_and\_distance.append([0, sys.maxsize])

for vertex in range(num\_of\_vertices):

to\_visit = to\_be\_visited()

for neighbor\_index in range(num\_of\_vertices):

if vertices[to\_visit][neighbor\_index] == 1 and \

visited\_and\_distance[neighbor\_index][0] == 0:

new\_distance = visited\_and\_distance[to\_visit][1] \

+ edges[to\_visit][neighbor\_index]

if visited\_and\_distance[neighbor\_index][1] > new\_distance:

visited\_and\_distance[neighbor\_index][1] = new\_distance

visited\_and\_distance[to\_visit][0] = 1

i = 0

for distance in visited\_and\_distance:

print("Distance of ", chr(ord('a') + i),

" from source vertex: ", distance[1])

i = i + 1