**K. J. Somaiya College of Engineering, Mumbai-77**

(Autonomous College Affiliated to University of Mumbai)

https://ssl.gstatic.com/ui/v1/icons/mail/images/cleardot.gif

**Batch: B1 Roll No.: 1711072**

**Experiment No. 4**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **Title:** Implementation ofSingle source shortest path by Greedy strategy |

**Objective:** To learn the Greedy strategy of solving the problems for different types of problems

**CO to be achieved:**

|  |  |
| --- | --- |
| Sr. No | Objective |
| CO 1 | Compare and demonstrate the efficiency of algorithms using asymptotic complexity notations. |
| CO 2 | Analyze and solve problems for divide and conquer strategy, greedy method, dynamic programming approach and backtracking and branch & bound policies. |
| CO 3 | Analyze and solve problems for   different string matching algorithms. |

**Books/ Journals/ Websites referred:**

1. **1. Ellis horowitz, Sarataj Sahni, S.Rajsekaran,” Fundamentals of computer algorithm”, University Press**
2. **T.H.Cormen ,C.E.Leiserson,R.L.Rivest and C.Stein,” Introduction to algortihtms”,2nd Edition ,MIT press/McGraw Hill,2001**
3. **https://www.mpi-inf.mpg.de/~mehlhorn/ftp/ShortestPathSeparator.pdf**
4. **en.wikipedia.org/wiki/Shortest\_path\_problem**
5. **www.cs.princeton.edu/~rs/AlgsDS07/15ShortestPaths.pdf**

**Pre Lab/ Prior Concepts:**

Data structures, Concepts of algorithm analysis

**Historical Profile:**

Sometimes the problems have more than one solution. With the size of the problem, every time it’s not feasible to solve all the alternative solutions and choose a better one. The greedy algorithms aim at choosing a greedy strategy as solutioning method and proves how the greedy solution is better one.

Though greedy algorithms do not guarantee optimal solution, they generally give a better and feasible solution.

The path finding algorithms work on graphs as input and represent various problems in the real world.

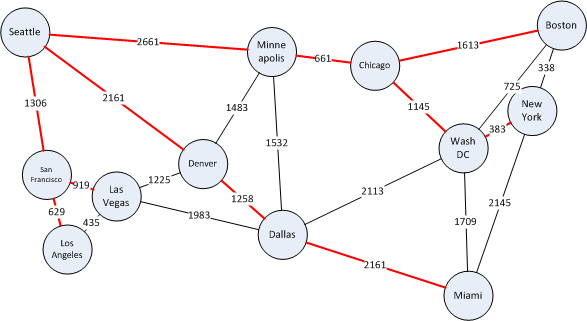
**New Concepts to be learned:**

Application of algorithmic design strategy to any problem, Greedy method of problem solving Vs other methods of problem solving, optimality of the solution

**Algorithm**:



**Example Graph:**



**Solution (Implementation in Python):**

class Graph():

def \_\_init\_\_(self, vertices):

self.V=vertices

self.graph=[[0 for col in range(vertices)] for row in range(vertices)]

def printSolution(self, dist):

print ("Vertex \tDistance from Source")

for node in range(self.V):

print (node, "\t\t\t", dist[node])

def minDistance(self, dist, SPT):

min = float('inf')

min\_index=0

for v in range(self.V):

if dist[v] < min and SPT[v] == False:

min = dist[v]

min\_index = v

return min\_index

def dijkstra(self,source):

dist= [float('inf')] \* self.V

dist[source]=0

SPT=[False] \* self.V

for \_ in range(self.V):

u=self.minDistance(dist, SPT)

SPT[u]=True

for v in range(self.V):

if self.graph[u][v] > 0 and SPT[v] == False and dist[v] > dist[u] + self.graph[u][v]:

dist[v] = dist[u] + self.graph[u][v]

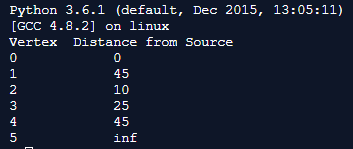
self.printSolution(dist)

g=Graph(6)

g.graph=[[0,50,10,0,45,0],[0,0,15,0,10,0],[20,0,0,15,0,0],[0,20,0,0,35,0],[0,0,0,30,0,0],[0,0,0,3,0,0]]

g.dijkstra(0)

**Screenshot:**



**Time Complexity for single source shortest path:**

The main logic of single source shortest path is defined in the function dijkstra(self, source). The main logic consists of nested for loops which both run for total number of vertices. If number of vertices is |V|, then the Time complexity of this algorithm is given by:

**T(n)=O(|V|2)**

The complexity of the algorithm can be improved by using Fibonacci heaps, Binary heaps, adjacency list or Disjoint data structure. If number of edges is “E” and number of vertices is |V|, then the time complexity on using any these data structures will be approximately:

**T(n)=O(E\*log(|V|)**

**CONCLUSION:**

**The program for Dijkstra’s shortest path algorithm was successfully implemented in Python and works for all test cases.**