Implementation and Analysis of Single source shortest path and All pair shortest path Algorithms

Tarun Kumar
UE143103
Computer Science Engineering
University Institute of Engineering and Technology
Panjab University
Chandigarh, 160012
Email: tarunk.1996@gmail.com

***Abstract*—This practical illustrates about the implementation of Single source shortest path and All pair shortest path algorithms and the approach followed in it. It analyses the time complexities of both the algorithm.**

I. Introduction

A **directed graph** is graph, that is, a set of vertices or nodes that are connected with each other, where all the edges are directed from one vertex to another vertex. A directed graph is also known as digraph .

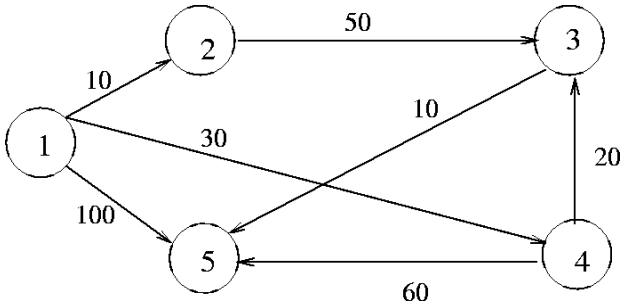

Fig. 1. An example of Directed graph

The above figure 1 has a total of 5 vertices and 7 directed edges in the graph. The number written over the edge is called the weight of the edge (means the cost to visit from one vertex to another vertex using that path or edge).In directed graphs, edges consist of a head pointing towards the direction of the path.

A. Single Source Shortest Path Algorithm

Single source shortest path follows a greedy approach (if all the edges have non-negative weights) for finding the shortest path from a node to every other node. To achieve the goal many algorithm's were developed so far. In this paper, Dijkstra's Shortest path algorithm is implemented for solving the problem. An array (of size n i.e. total number of nodes) is produced showing minimum cost and shortest path possible from origin to every other node.
In Dijkstra's algorithm, an origin node is selected and all the directed edges from that node to other nodes weights is assigned to the cost array.(Note: if the two vertices are not connected the distance between them is taken as infinity). Select the least cost edge from the cost array (Also, create a flag array consisting of all the visited nodes so far) and set flag variable of that node to 1. Now, Update the cost node if, the cost of visiting from origin to current node + cost of edge from current node to remaining nodes is lesser than cost from visiting origin to remaining node present in the cost array. Again, select the least cost among the unused nodes from the cost array and repeat the process until each and every node is not visited once.
The time complexity of Dijkstra's shortest path algorithm is $O(VlgV + E)$.(V represents total number of vertices and E represents total number of edges) But in here the dominating factor is E as the bound value of E is $O(V^2)$).(This is the case when multigraphs are not considered where two set of vertices have multiple edges between them).

B. All Pair Shortest Path Algorithm

All pair shortest path follows a dynamic approach. All pair shortest path problem is determined as a Matrix A such that $A(i, j)$ is the length of the shortest path from vertex i to vertex j. Firstly, Matrix $A^0(i, j)$ is build which is generally the cost of directed edges from node i to node j. After that Matrix $A^1(i, j)$ is build where the cost of visiting i to j is $min\{cost.A^0(i, j), cost.A^0(i, 1) + cost.A^0(1, j)\}$. The process is repeated n time where n is total number of vertices in the graph. The Final resultant matrix $A^n(i, j)$ is the required matrix which gives the All pair shortest path problem's solution.
The time complexity of All Pair Shortest Path Algorithm is $O(V^3)$ as it consists of 3 loops in it.

II. Approach

Single source shortest path problem as specified in it's name is used to find the shortest path from origin vertex to all the other vertices. In this practical, Dijkstra's algorithm is used. To analyse the the working and complexity of the algorithm for different values of V (i.e total number of vertices) random values are provided to the edges weight using random function.

The time required to find the shortest path from an origin node to all the other nodes are noted down for different values of V.

Similarly, for All pair shortest path problem. The algorithm specified in the later section of the paper is used to find the minimum distance from any node to any other node. To analyse the working and complexity of the algorithm, random numbers are used to assign the weight to the edges. The algorithm is executed for different values of V and time required to execute the code is noted down to plot a graph and analyse it.

III. Algorithms and Implementation

A. Single Source Shortest Path Algorithm

```
for(i=0;i<n;i++)
{
    for(j=0;j<n;j++)
    {
       if(k!=-1)
       {
         b[j]=a[z][j];
       }

       else
       {
        if(b[j]>b[k]+a[k][j])
        {
            b[j]=b[k]+a[k][j];
        }
       }
     }
     min=10000;
     for(j=0;j<n;j++)
     {
       if(min>b[j]&&!c[j])
       {
          min=b[j];
          x=j;
       }
     }
     c[x]=1;
     k=x;
}
```

B. All Pair Shortest Path Algorithm

```
for(k=-1;k<n;k++)
{
    for(i=0;i<n;i++)
    {
       for(j=0;j<n;j++)
       {
          if(k==-1)
          {
            b[i][j]=a[i][j];
          }
          else if(b[i][j]>b[i][k]+b[k][j])
          {
            b[i][j]=b[i][k]+b[k][j];
          }

       }
    }
}
```

IV. Graphs and Tables

Following are results obtained after executing the program:

TABLE I
ALL PAIR SHORTEST PATH RESULT

Total number of vertices	Time required (in ticks)
2	1
3	1
4	1
5	1
6	1
7	3
8	3
9	5
10	6
11	8
12	11
13	14
14	16
15	18
16	22
17	27
18	31
19	36
20	45
21	47
22	56
23	61
24	69
25	77

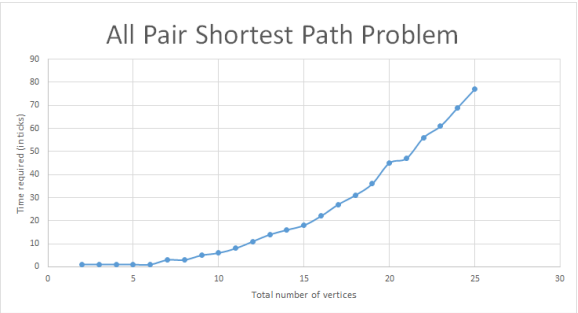

Fig. 2. Graph obtained from table 1 results

TABLE II
SINGLE SOURCE SHORTEST PATH RESULT

Total number of vertices	Time required (in ticks)
2	0
3	0
4	0
5	1
6	1
7	1
8	1
9	1
10	1
11	1
12	1
13	2
14	2
15	2
16	2
17	2
18	3
19	3
20	3
21	3
22	3
23	4
24	4
25	4

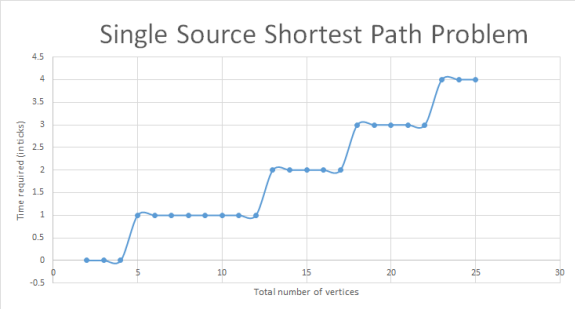

Fig. 3. Graph obtained from the results of table 2

V. CONCLUSION

- The time complexity of Dijkstra's Algorithm for finding single source shortest path is $O(V^2)$.
- The time complexity of All pair shortest path algorithm is $O(V^3)$.

VI. ATTACHMENTS

The Program code used for performing All Pair Shortest Path Problem is here : http://pastebin.com/0azSff0S
The Program code used for performing Single Source Shortest Path Problem is here : http://pastebin.com/8ppfsbwQ