TELEPHONE NETWORK

PROJECT REPORT

Curriculum for Applied Learning

Data Structures and Algorithms – CSE2003

Yash Gupta 15BCE2073

B. Tech.
in
Computer Science and Engineering
School of Computer Engineering

VIT®
UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)
VELLORE ▪ CHENNAI

May, 2016

ABSTRACT

The aim of this project is to create a real-time application by implementing data structures. To achieve this aim, and to demonstrate our knowledge in this subject, I have designed a telephone network system.

In a telephone network, the lines have bandwidth BW. The bandwidth of a telephone line is the highest frequency supported by the line, and it represents the amount of information that can be transmitted by the line. We want to route the call via the highest BW. The network will be designed using the graph data structure, implemented using linked list. The vertices of the graph will represent switching stations, the edges will represent transmission lines, and the weighted of the edges will represent the BW of the line. This will be done in the C language.

Table of Contents

1. Introduction

1.1 Problem Statement

The problem requires us to design and implement a telephone network using graphs. The nodes of the graph will act as cell towers or switching stations. The edges will be the routes between the towers and the weights of the edges will be the bandwidth of that route. In a telephone network, the bandwidth is the maximum information that can be transferred through a route. Thus, calls are routed via the route having maximum bandwidth to enable maximum transfer of information, and thus, maximum efficiency. A menu driven program has to be designed which will input the number of nodes, their names, adjacency matrix, and the source and destination node. The source and destination nodes have to be distinct and valid nodes. The output will be the longest path from source to destination, and the bandwidth along that path, and the user will have the option to quit, or continue either with the same network, or with a different network.

2. Project Plan

2.1 Software Used

The Telephone Network has been implemented in the C language. For this purpose, we had used the Atom text editor, and the code was compiled and executed in the Dev-C++ compiler.

2.2 Reason / Functionality of Language

The Telephone Network has been implemented in the C language. The reason for choosing this particular language was its simplicity, open-source nature and the fact that our class lectures are imparted with respect to the C language.

2.3 Data Structure Used

The Telephone Network uses graphs for implementation. The graph data structure is used because of the following key reasons:

- The algorithm used for this problem is a modified form of the Dijkstra Algorithm, and this algorithm requires the use of graphs.
- The concept of nodes and weighted edges exist only in graphs.

3. Design: Functions/Modules

The Telephone Network accepts the number and names of nodes, the adjacency matrix and the source and destination nodes. It then displays the longest path from source to destination in terms of the weight of the edges. For this purpose, we have used 2 primary modules:

- void dijkstra (int G[MAX][MAX], int n, int startnode, char name[MAX])

  This function accepts the adjacency matrix (G[MAX][MAX]), the number of nodes (n), the source node (startnode) and the array of the names of the nodes (name[MAX]). It then creates an array to keep track of the nodes visited (visited[MAX]), an array to keep track of the preceding node of each node from the source (pred[MAX]), an array to store the maximum distance of each node (distance[MAX]), and a 2-D array (cost[MAX][MAX]) as the cost matrix. Then, in a loop which runs (n-1) times, the distance of each node from the current node is calculated, and the distance matrix is updated if the calculated distance is greater than the previously calculated distance. In the end, it asks for a destination node from the user, and this should be an existing node and distinct from the source node. If these conditions are not satisfied, error message is displayed and it prompts the user again. Finally, maximum distance and the route from the source node to destination is displayed.

  This function implements a modified form of Dijkstra's Algorithm, where instead of calculating the shortest path, we will find the longest path.

- int main()

  This is the driver function for the program. It accepts the number of nodes and their names from the user. Then it accepts the corresponding adjacency matrix, and the source node. Finally it calls the above function and passes all of this information as parameters. This happens inside a loop and after the function call, the user is given an option to exit, or continue, either with the same network, or to input a new set of nodes and adjacency matrix.

4. Key / Challenging Logic

The key point in the Telephone Network was to decide upon the algorithm to be used. Many longest path algorithms exist, such as the Hamiltonian Path problem or the Travelling Salesman problem. However, the Hamiltonian Path algorithm is a brute force algorithm, and so, it is highly inefficient as the number of nodes increases. The Travelling Salesman algorithm is NP-complete, and so, as the number of nodes increases, its complexity grows exponentially. Thus, longest path algorithms were too complex to be used. The remaining approach was to modify a shortest path algorithm such that it gave the longest path. The most efficient shortest path algorithms are Johnson's algorithm, Floyd-Warshall algorithm, Bellman-Ford algorithm, Best-First Search algorithm, A* search algorithm, minimum spanning tree algorithm and Dijkstra's algorithm. Each of these algorithms has certain disadvantages which are:

- Johnson's algorithm finds the minimum distance between all pairs of nodes, thereby increasing the complexity. Moreover, it makes use of Dijkstra's algorithm itself.
- Floyd-Warshall has a complexity of $O(|V|^3)$, which is much greater than Dijkstra's. Moreover, it cannot give us the details of the path taken.
- It has complexity $O(|V|.|E|)$, which is slower than Dijkstra's algorithm.
- Best-First search is an informed graph search, meaning it should only be used when there is some prior knowledge about the graph. Moreover, it uses the heuristics function to increase its efficiency, whereas Dijkstra's is always optimal. A* search is just a special case of BFS.
- A minimum spanning tree requires all nodes to be visited at least once, and then it calculates the minimum total weight. However, the length of a path between any two nodes in the MST may not be the shortest distance between them. Moreover, shortest path does not require all the nodes to be visited.

Thus, Dijkstra's algorithm was the choice for the implementation purpose.

We have modified this algorithm in two specific ways. Firstly, the cost matrix was initialized with -9999 (instead of 0), where there was no edge. Secondly, the cost matrix can be updated even when a node has already been visited unlike the conventional rule in Dijkstra's algorithm. We have also come up with a way to display the shortest path by keeping a track of the predecessor node of each node beginning from the source node.

5. Errors Encountered

While designing the telephone network, the errors generated were because of the modifications to Dijkstra's algorithm to change it into a longest path algorithm. Due to these changes, the path from the source to destination was often redundant in nature, with some nodes being visited more than once, especially in cases where the graph had cycles in it. To avoid this, two major changes had to be done in the algorithm.

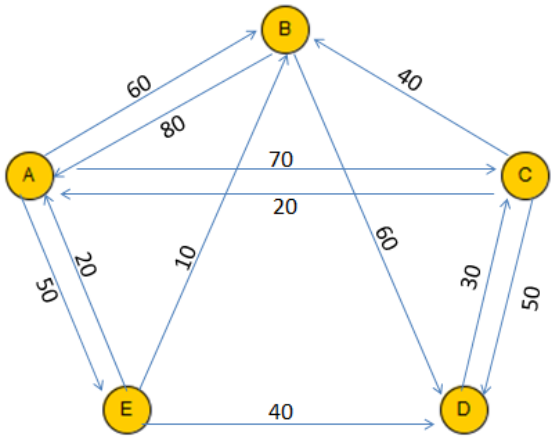
- Originally, in Dijkstra's algorithm, the cost matrix is not updated with new values of distance for a node if that node has already been visited. However, since we modified Dijkstra's algorithm to give the longest path, we removed this condition, because even if a node has been visited, a longer path may arise at any instant, and that path will have to be considered because Dijkstra's is a greedy algorithm.
- When the distance value for a node is being updated, and that node has already been visited before, then we check whether in the path of that particular node, the current node had already been involved. If it is true, it is an indication of repeated visits to the same node, and updating of the distance value will not take place.

These were logical errors and were quickly identified and rectified. Our modification to Dijkstra's algorithm is still not perfect though. It holds good for unidirectional graphs, but fails in some cases when a bidirectional graph is given.

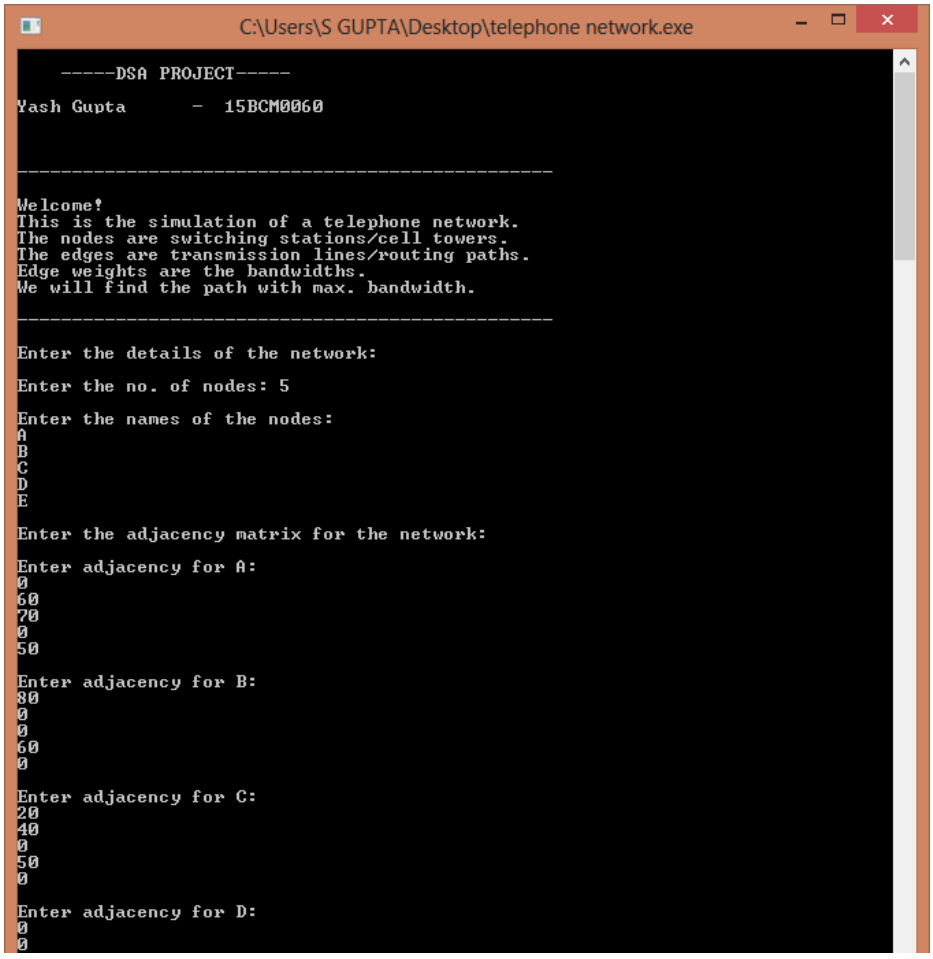
This is because Dijkstra's algorithm is an NP-complete problem, meaning its solution can be successfully verified in polynomial time, but a solution may or may not be found in polynomial time.

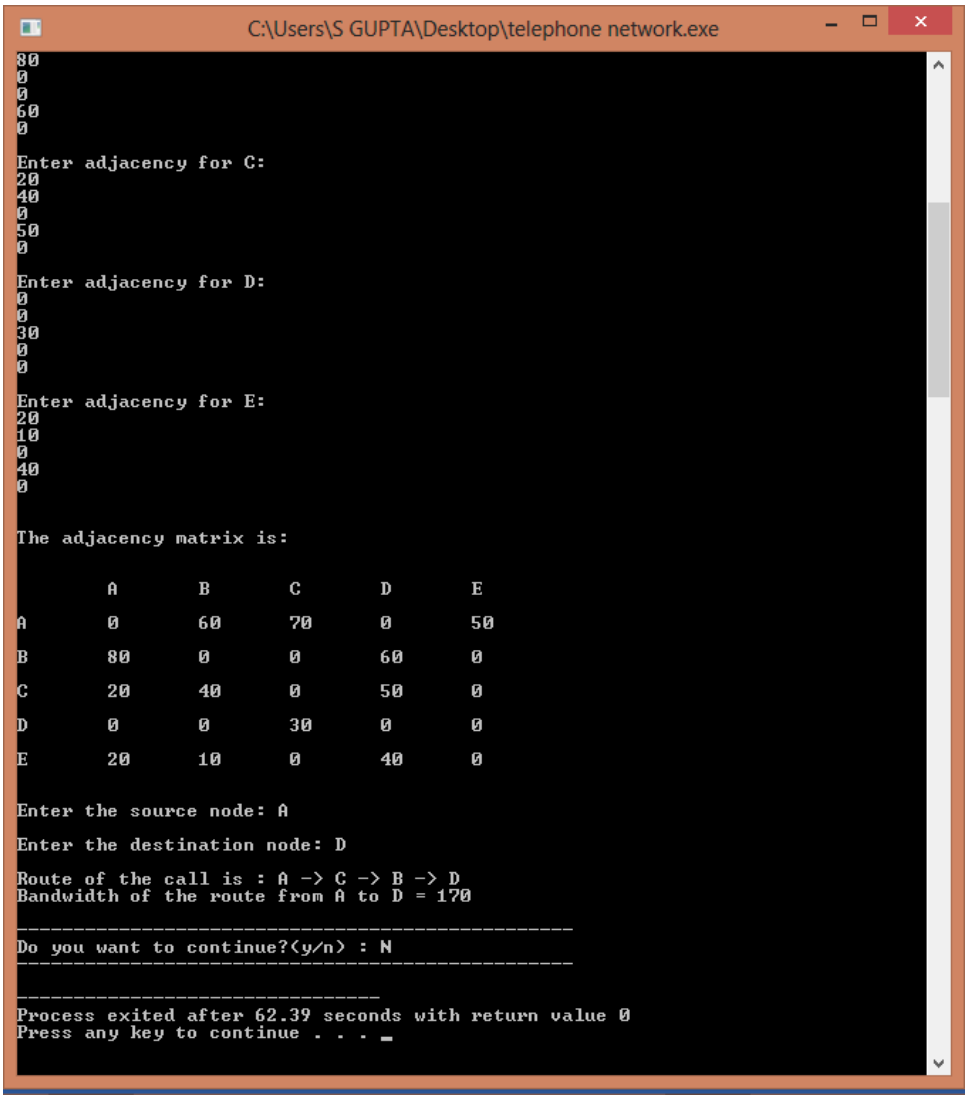
6. Test Cases

The snapshots below show a graph, its adjacency matrix and the output when the shortest path is found in that graph:



	A	B	C	D	E
A	0	60	70	0	50
B	80	0	0	60	0
C	20	40	0	50	0
D	0	0	30	0	0
E	20	10	0	40	0





Thus, first the adjacency matrix is accepted, then displayed, and then the source and destination nodes are accepted and the route of the call, along with the bandwidth is displayed.

7. Conclusion

In this project, the telephone network is based primarily on Dijkstra's algorithm. I have made my very own contribution towards the versatility of Dijkstra's algorithm, by taking a shortest path algorithm, and engineering it into its exact opposite, along with a slight modification so that it displays the path it is taking.

Data structures are used in every branch of the information technology sector, and are underlying some of the most common technology we use today, from checking our mail to browsing the internet and so on. The analysis and design of algorithms in itself is a vast field, and to have made even a slight contribution to it has been overwhelming. This project has provided me with an insight into mainstream industrial applications of classroom knowledge, and I feel glad to have taken this opportunity.

8. References

- https://cs.stackexchange.com, an online question and answer forum. <For referring issues with the code>
- http://en.wikipedia.org, the free online encyclopedia. <For referring doubts related to algorithms>