REPORT Assignment 4

## TRIE:

--------------compile----------------

$triemain.cpp

------------instruction------------

file with name as “name\_number.txt’ containg info about names and numbers should be there in same folder as main file.

---------------------details---------------

I have used uncompressed trie for storing phonebook . Order compilexity for insert operation is O(m) and search opeartion is also order O(m) where m is the length of string. Therefore max time complexity for any operation is O(m)(Number of operation for printing top 5 are O(5m) ,therefore O(m)). Top five names will be printed to input string using preoder traversal. Nunber of contacts with that string is already stored in that node ,therefore no extra time complexity is there for finding the total number of names with that string.

Each node pointer contains 26 pointers to its children ,’0’ being ‘a’ and 25 being ‘z’.

Number for each node is initialized to 0 and number is updated whenever a name ends at that node. This way whenever a number is not zero we know that the name ends at that node.

## GRAPH:

------------compile--------------

$graph.cpp

------------instruction------------

file with name as “courses.txt’ containg info about vertices with string and duration and another file with name “dependencies.txt” containg edge information as given in question should be there in same folder as the main file.

------------details------------

Vertex class contains vector of children storing the vetices to which the node is pointing.

Created a adjacency list for storing the graph. Used map for storing node pointer and string to find pointer to node if string is given used in insert edge function. For insert vertex a new object is added in map with its string a pointer to a new node.

Order :

for checking DAG, every node is checked for cycle once and function is returned if node is already visited ensuring that node is not visited twice .Therefore order is O(v+e).

for giving path,we recursively store max path lengths for each node which give max course length to end node. We take maximum of max paths of all children and add node’s duration to get its max path.For last node,max path is it’s own duration. In this also , we store if a node is visited before so that every node’s max path is calculated only once and oredr reamins O(v+e). At last we return the max path of root node , which is the first node in file as given in question.