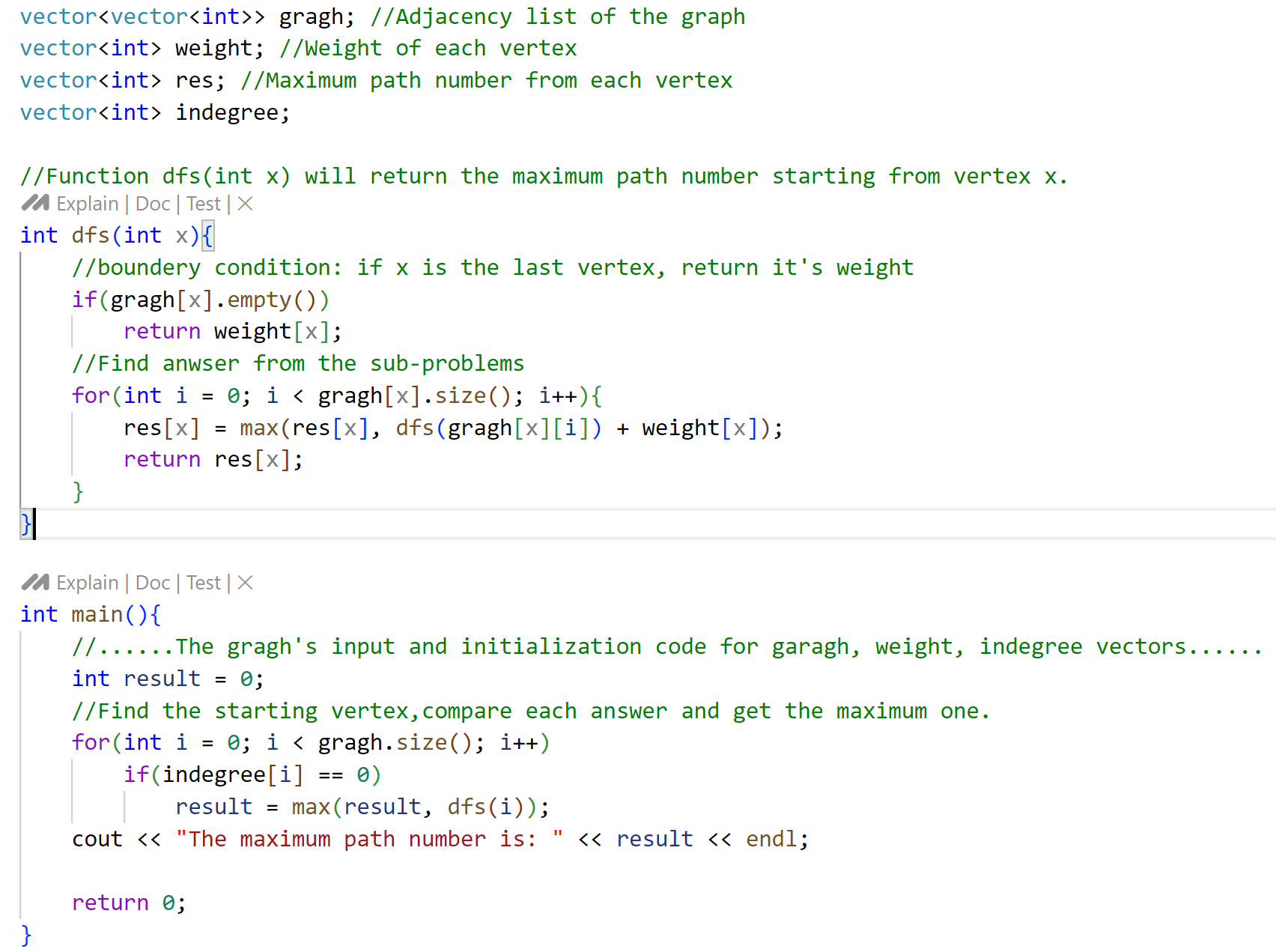
**Lab6-1**

1. Design Thinking

This problem also has locally optimal solution,which means that we can use DP to get the answer.The basic idea is that to find the longest path,we need to find all the possible starting vertices first.Then ignore the starting vertices,try to find the longest path in the relevant gragh.Continue this process untill the relevant gragh is a trivial gragh,the only vertex would be the terminus.We can easily get the weight of terminus,which is also the answer of the base case.Since that every step we get the longest path,the final anwser would also be the correct one.Following is a brief structue of the solution(using dfs to make it clearer first).



1. Data Structure

Apparently,dfs can’t fit the requirement of finding the anwser in linear time.But we can find the state transition equation for DP method through it.

Using a vector named res to store the anwser while res[i] means the maximum path from possible to vertex i.The idea is finding all the starting vertex which property is that in-degree would be 0.For initialisation,res[i] for starting vertex is its weight.Then find the adjacent vertices of these starting vertex,using the longest path data of previous vertices,find the answer for the adjacent vertices and store it to the vector res.Also elimilate the dependence between these adjacent vertices and starting poits.Then check whether these adjacent ones can be a new starting point for the next process,and continue the step untill reach the terminus.

Considering that in each step we always find the starting vertices(in-degree is 0)first,then due with their adjacent ones.Elimilating dependence,try to find starting vetices again......We can choose a queue to store the vertices going to be processed.When one vertex’s in-degree is 0,push it in the queue.When we pop a vertex in the queue,processing it,elimilating dependence between it and its adjacent vertices,during the course,finding the vertices whose in-degree become 0 and push them to the queue.

1. Time/Space Complexity Analysis

Our attention is on the algorithm,so skip the analysis of how much time and space inputting examples costs.

Time Comlexity:

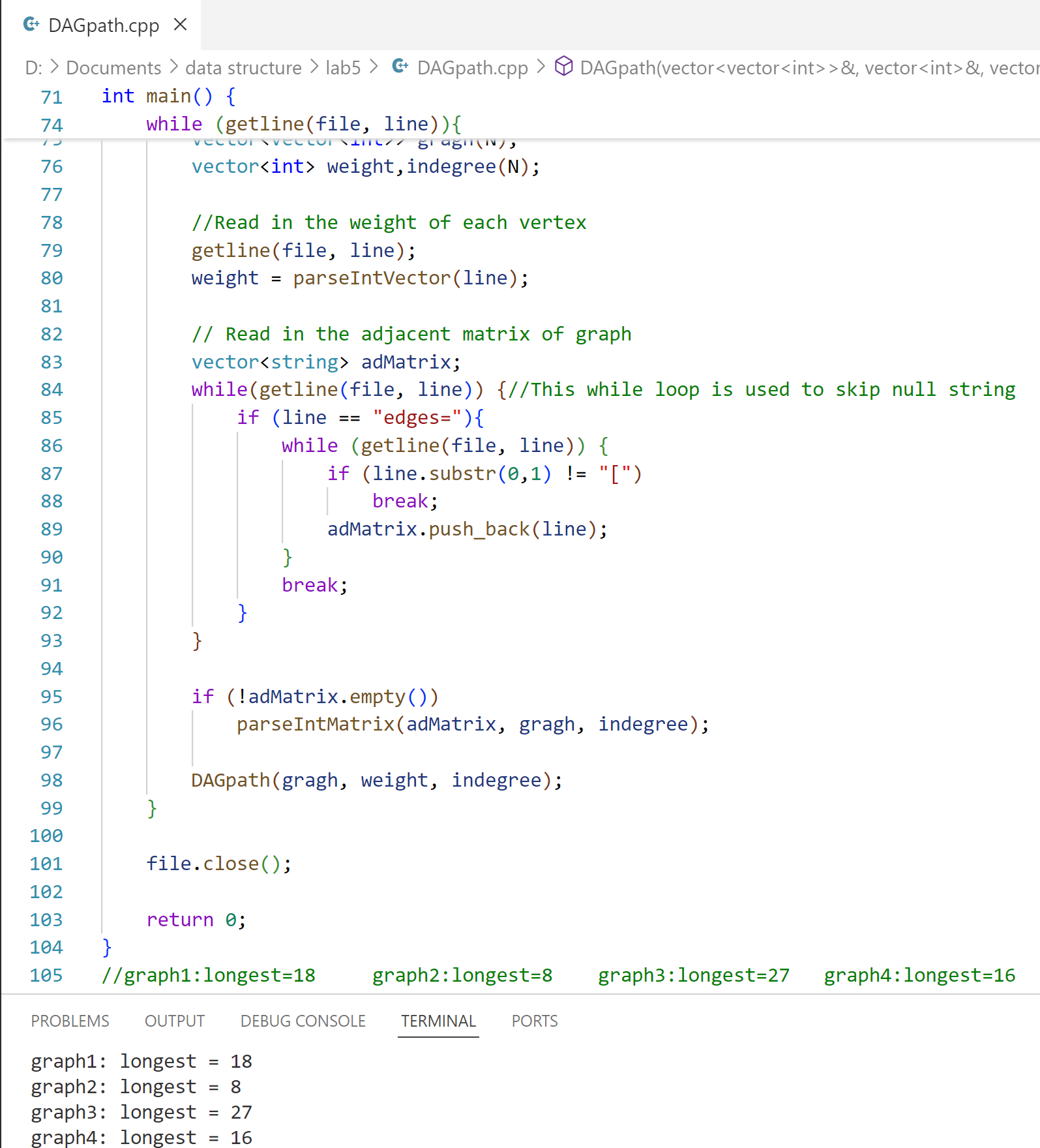
The time complexity is mainly depends on topological sort and dynamic programming.Since thet during this course,every vertex and edge is going to be used at most once,time complexity is O(N + E).N is number od vertices,E is number of edges.

Space Complexity:

Space complexity for adjacenct list is O(N + E),N is number od vertices,E is number of edges.For vector weight,indegree,res vector,are all O(N).In the worst case,queue’s space complexity can be O(N).So space complexity is O(N + E).

All in all,we can meet the standard of finding the answer in linear time.

1. Experiment Screenshot



1. File Input Format Explanation

For the guarantee of program’s correct execution,input file should follow specific format.

File should start with”vertices\_weight=”,with no null string or blank before it.

Datas should fit the format “[X,X,X,X,X,X],”.

There would be one null string between two gragh examples.

