**Lab5**

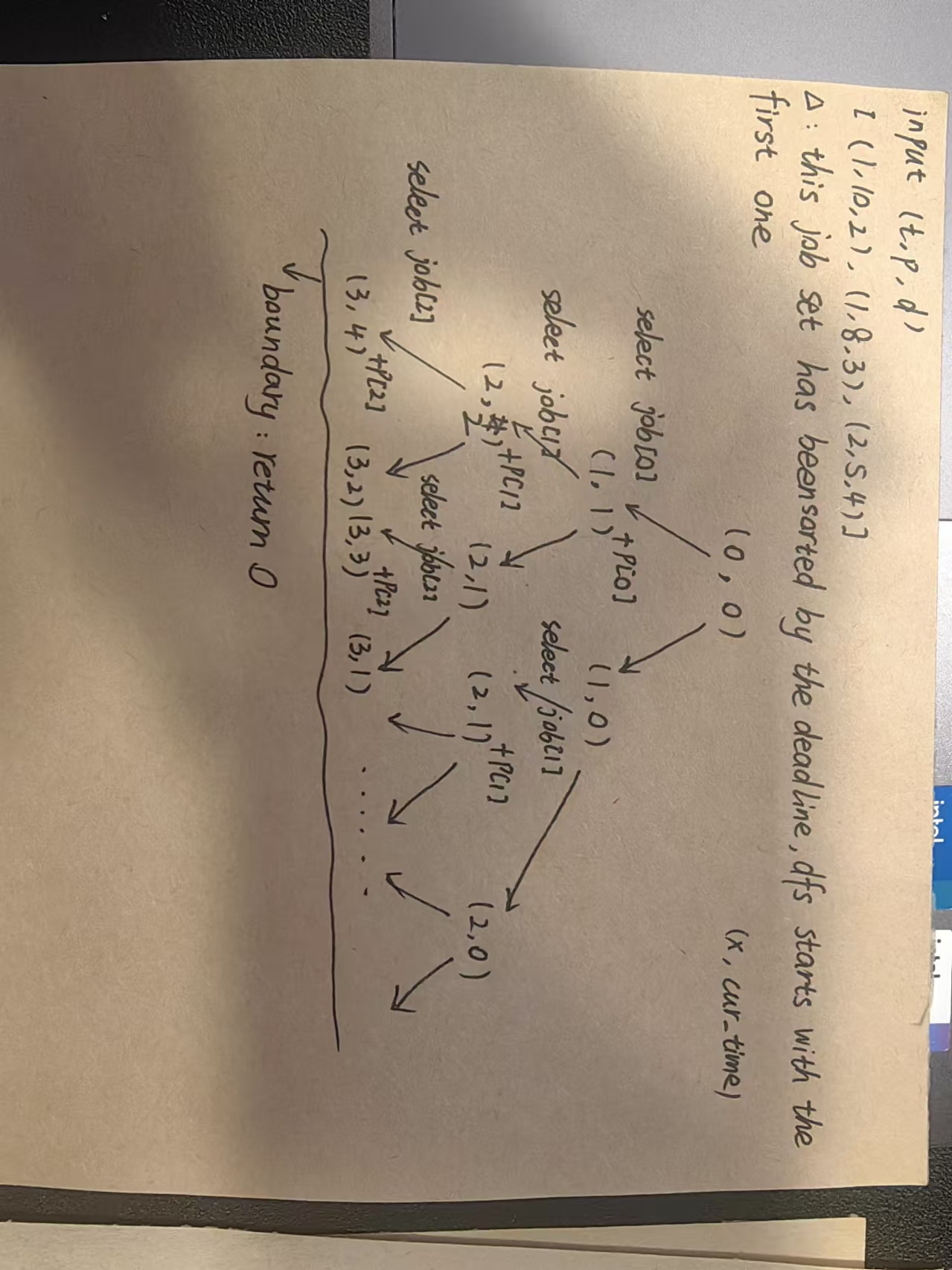
1. Design Method and Explanation

The experiment’s purpose is to find the max profit among a set of jobs with given profit,process time and deadline,we can use the DP method to achieve it since this problem has locally optimal solution.In order to make the procedure of finding the state transition equation clearer and see how to improve the algorithm’s efficiency obviously,I decided to start from the dfs(brute force),memrised search(memorandum),recursion(DP) with inverted order,recursion(DP) orderly,finally to the optimised DP(scrolling array) algorithm.Following is explanation to these algorithms.

First,in order to find the best schedule with max profit,all strategies need to sort the job set with their deadline.

The dfs function starts from the first job.This is a recursive function which parameter are the count number of the job be concidered(from 0 to n - 1,n is the size of job set) and current time recording how much time has been used due to those selected jobs before.The function dfs(x,cur\_time) returns locally max profit.Boundary condition is when x reach n,function returns 0.When it didn’t reach the boundary,we first judge that whether we can benefit from the current job,if not return dfs(x + 1,cur\_time),showing that we can’t select this job,simply skip it and consider the next one.If we can benefit from this job,we have two options:one is don’t select this one for long-term consideration,another is select it,spending the process time it costs and gain the profit it brings.At this time,to find the max profit,the function return the bigger one of them.

The recursion tree is basically like this,we can find that the function dfs(0,0) will give us the final answer.But through this diagram,we can apparrently see that there are a lot of repeated fuction call,which cost a lot of time.



So we choose the memorised algorithm to optimise it,which just using an arrey to memorise the return value of the dfs function.So we only invoke each dfs function with different parameters once,because the once we called one specific function,store the return value to memory array.Afterwards when we call it again,just return the array value directly.

Although we use the memory array to cope with the repeated function call,the recursion function still costs plenty of time to run out for the final answer.We can find a state transition equation from this recursion course,then we can change it to a DP algorithm.

Due to subproblems before are from the end of the job set,for the purpose of making it easier,consider the DP algorithm with inverted order first.Using a two dimension array f.f[i][j] represcents that when all the jobs are [0:n - 1],the max profit for the job[i:n - 1] with start time j.Then use the state transition equation to assign value for the array f.According to the definition before,f[0][0] would be the final answer.Then we can use the data in array f to find the job schedule giving the max profit,and print it.

Then consider how to write a orderly DP algorithm.Using two dimension array f.f[i][j] represcents that when all the jobs are [0:n - 1],the max profit for the job[0:i] within time j.Then use the state transition equation to assign value for the array f.According to the definition before,f[n - 1][max\_ddl] would be the final answer.The reason to choose the max deadline in the job set as the boundary of time j is that we can’t benefit anymore when one job completed after the max deadline.Then we can use the data in array f to find the job schedule giving the max profit,and print it.

Based on the orderly DP algorithm,we can have some optimisation on the space.We can find that in the orderly DP,once some row of f array’s data is used,we don’t use it anymore.So we can change the two dimension array f to one dimension.Thus get the optimised DP algorithm.

1. Experiment Data and Time&Space Analysis
2. Dfs

In the worst case,we will explore all the job schedule.For that every job has two options:select or don’t select,we have 2^n different schedules. The time complexity of dfs is O(2^n).

For the recursive function,we need stack to store the recursion state,which costs space.Because the boundary condition is that when x reach n,the function returns 0.So the recursive call depth is n.We also need a vector jobs to store the job set,which needs space of O(n).So the space complexity can be O(n).

1. memorised search:

Time complexity mainly comes from the memorandum array.The recursion depth is n,cur\_time is at most max\_ddl.So the time complexity is around O(n \* max\_ddl).

We need vector jobs to store information of job set,space complexity is O(n).Memorandum arrsy’s size is n \* max\_ddl.So the space complexity is O(n \* max\_ddl).

1. DP with inverted order:

Using memset to initialise f array——O(n \* max\_ddl).Nested for loop——O(n \* max\_ddl).Construct jobList——O(n).So time complexity is O(n \* max\_ddl).

F array’s size is n \* max\_ddl.Vector jobs’s size is n.So space complexity is O(n \* max\_ddl).

1. Orderly DP:

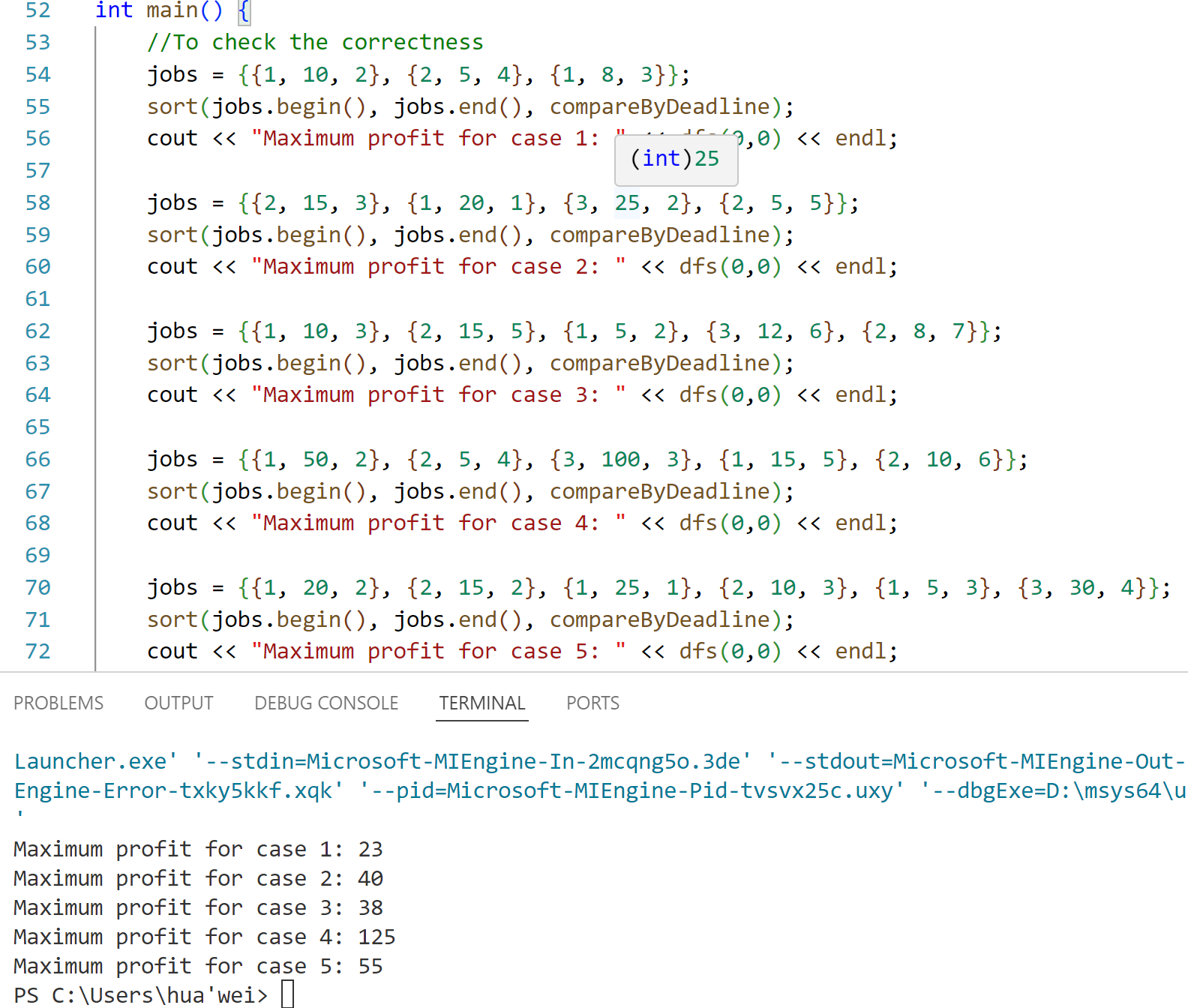
Same as DP with inverted order.

1. Optimised DP:

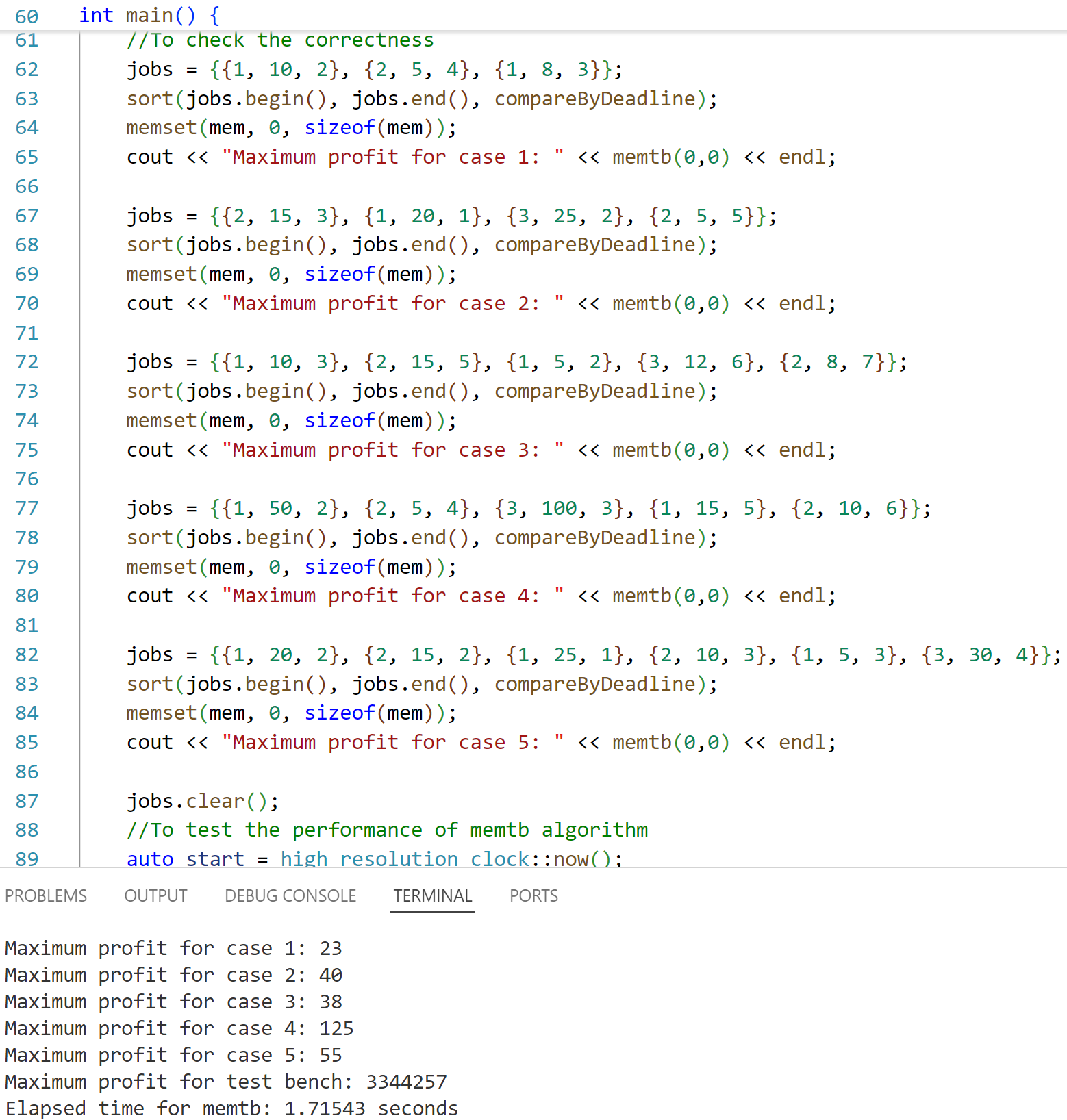
Using memset to initislise f——O(max\_ddl).Nested for loop(Dynamic programming)——O(n \* max\_ddl).So time complexity iS O(n \* max\_ddl).

Array f’s size is max\_ddl.Vector jobs’s size is n.So space complexity is O(max(n,max\_ddl)).

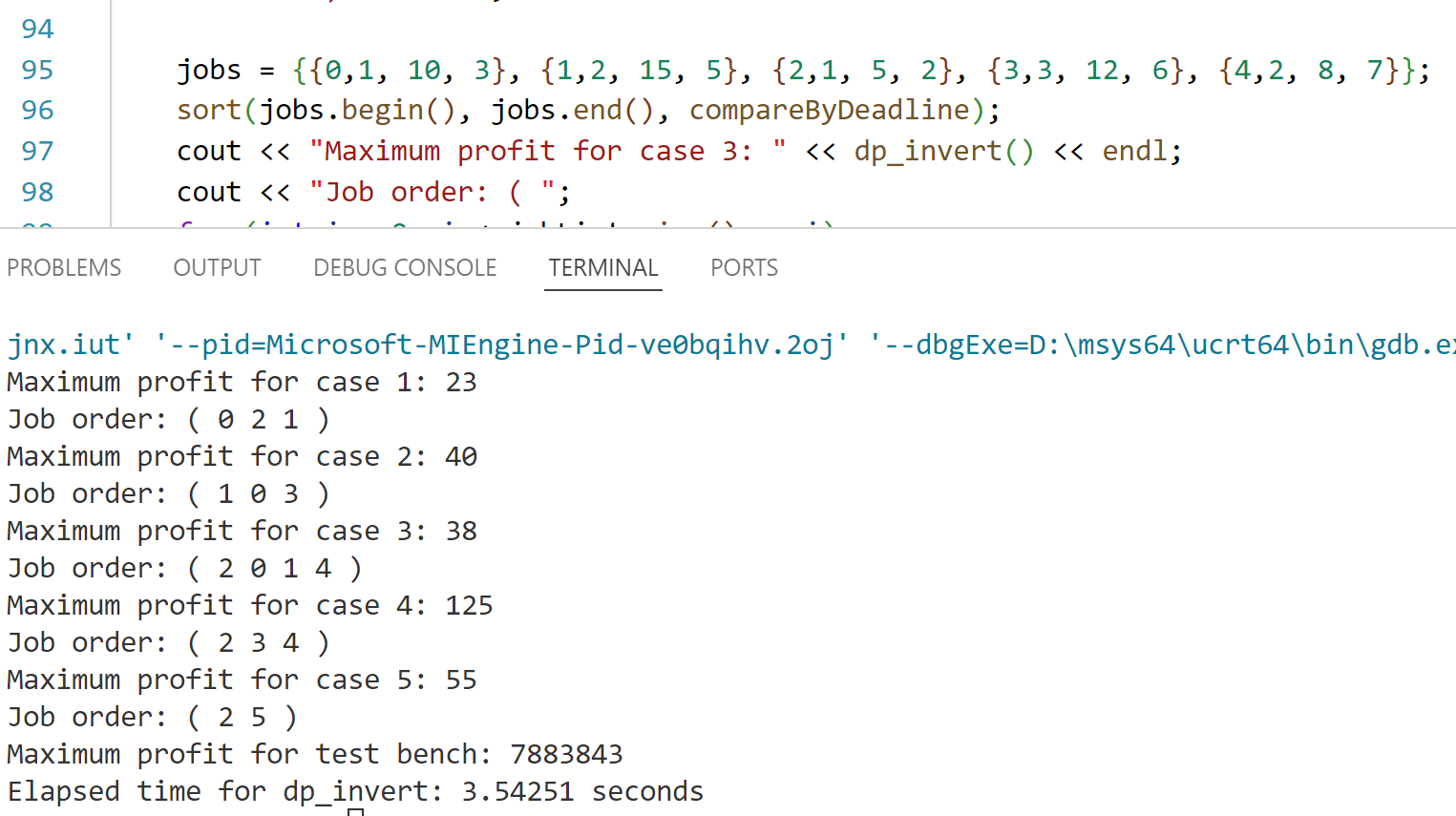
1. Experiment Screenshots
2. Dfs



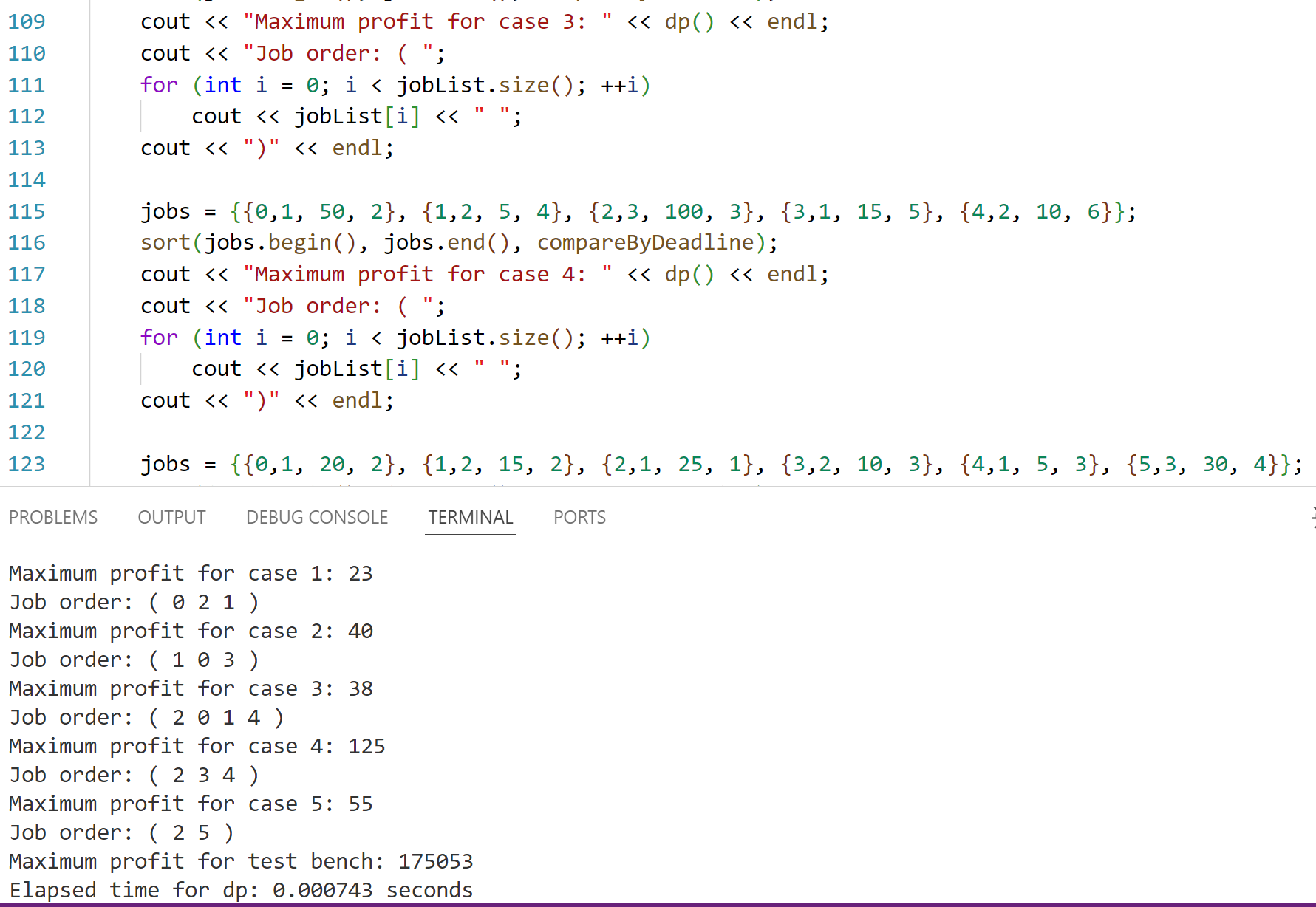
1. Memtb



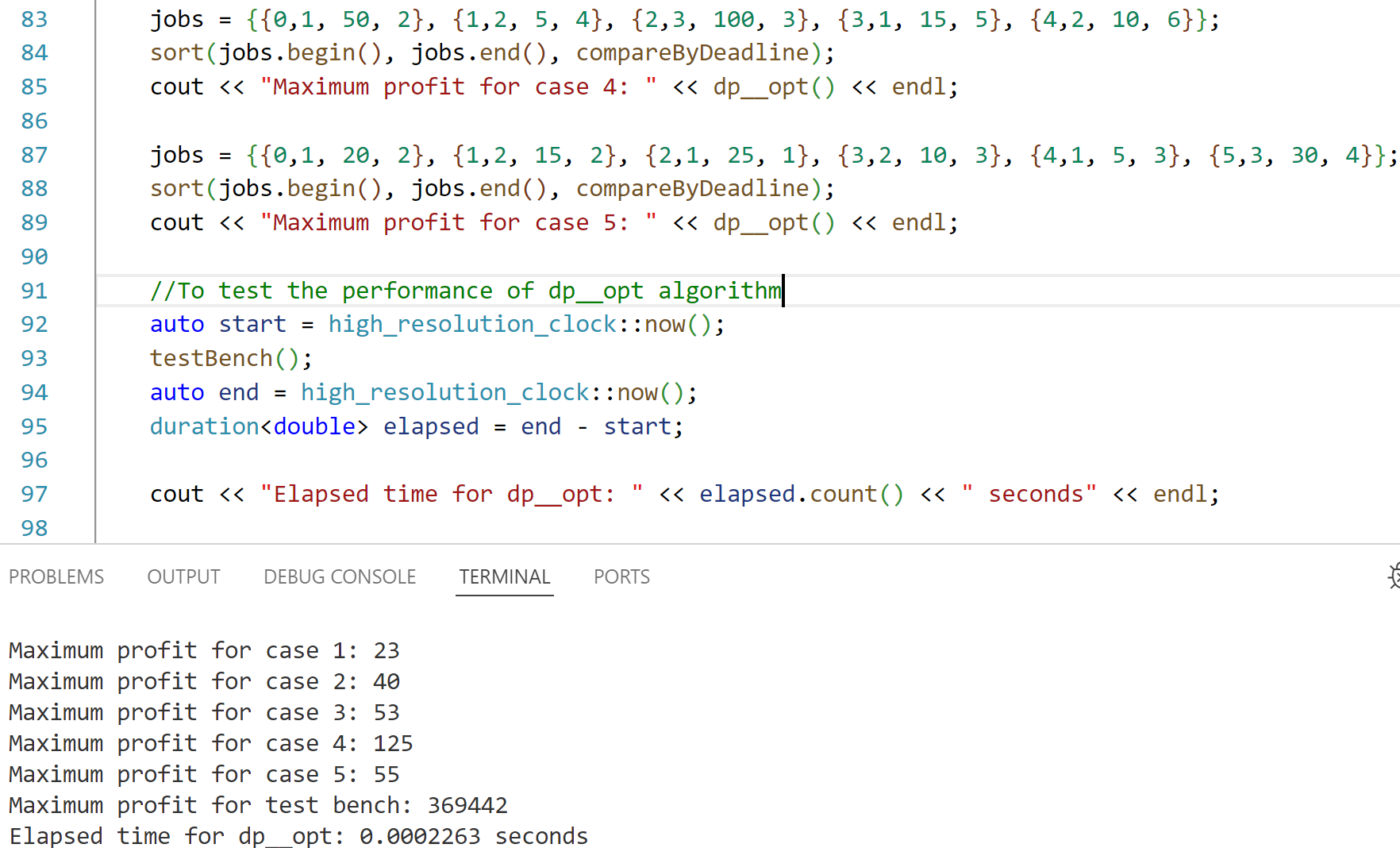
1. DP\_inv



1. DP



1. DP\_opt



1. File Explanation
2. Dfs:maxPro\_dfs.cpp
3. Memtb:maxPro\_memtb.cpp
4. DP\_inv:maxPro\_dp.cpp
5. DP:maxProfit.cpp
6. DP\_opt:maxPro\_opt.cpp