Yuan Gao z5239220 Q4

You are given a set of n jobs where each job i has a deadline $d_i \geq 1$and profit $p_i > 0$. Only one job can be scheduled at a time. Each job takes 1 unit of time to complete. We earn the profit if and only if the job is completed by its deadline. The task is to find the subset of jobs that maximises profit. Your algorithm should run in time.

Create a Two-dimensional array $jobs$, every element contains two numbers which are deadline and profit$[deadline, profit]$.

Step 1:

Sort all jobs in decreasing order based on profit

Step 2:

Create a $selected\_work$ array as every time unit has a slot

Step3:

Iterate $jobs$ array and find slot

If slot which before the deadline is empty, this job can be choose

Greedy proof:

Assume $p_i < p_j$ but p_i is considered before p_j

When $d_i < d_j$

Swap i and j won't change the total profit.

When $d_i \geq d_j$

If there is no empty slot before d_i, get d_j first can increase our total profit.

Assume $p_i = p_j$ and $d_i < d_j$ but p_i is placed after p_j

If p_i and p_j both into the $selected\_work$, swap i and j do not have different result.

If p_j in the $selected\_work$ but p_i is not, putting j closer to its deadline, it may give place of slot for i. Hence, the greedy method can increase the total profit.

Time complexity:

Sort all jobs costs $O(nlogn)$

Iterate $jobs$ array and find empty slot cost $O(n^2)$

The total time complexity is $O(n^2)$

Code:

```
/*
Two-dimensional array jobs,
every element contains two numbers which are deadline and profit[deadline,profit].
day means that the worker has total work days
*/
int maxProfit(vector<vector<int >> jobs, int day){
    int res = 0;
    //Create a selected_work array as every time unit has a slot
```

```cpp
    //the default value is -1
    vector<int> selected_work(day, -1);

    //Sort all jobs in decreasing order based on profit
    sort(jobs.begin(), jobs.end(), [](vector<int> &a, vector<int> &b){
                                        return a[1] > b[1];
    });

    //Iterate jobs array and find slot
    for(auto i: jobs){
        int tmpNum = i[0] - 1;
        //If slot which before the deadline is empty, this job can be choose
        while(tmpNum >= 0){
            //If slot is empty, put the profit in it
            if(selected_work[tmpNum] == -1){
                selected_work[tmpNum] = i[1];
                res += i[1];
                break;
            }else{
                //else find another empty slot
                --tmpNum;
            }
        }
    }

    return res;
}
```