**Yuan Gao z5239220 Q4**

You are given a set of jobs where each job has a deadline and profit . 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 , every element contains two numbers which are deadline and profit.

Step 1:

Sort all jobs in decreasing order based on profit

Step 2:

Create a array as every time unit has a slot

Step3:

Iterate array and find slot

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

Greedy proof:

Assume but is considered before

When

Swap and won't change the total profit.

When

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

Assume and but is placed after

If and both into the , swap and do not have different result.

If in the s but is not, putting closer to its deadline, it may give place of slot for . Hence, the greedy method can increase the total profit.

Time complexity:

Sort all jobs costs

Iterate array and find empty slot cost

The total time complexity is

Code:

1. /\*
2. Two-dimensional array jobs,
3. every element contains two numbers which are deadline and profit[deadline,profit].
4. day means that the worker has total work days
5. \*/
6. **int** maxProfit(vector<vector<**int** >> jobs, **int** day){
7. **int** res = 0;
8. //Create a selected\_work array as every time unit has a slot
9. //the default value is -1
10. vector<**int**> selected\_work(day, -1);
12. //Sort all jobs in decreasing order based on profit
13. sort(jobs.begin(), jobs.end(), [](vector<**int**> &a, vector<**int**> &b){
14. **return** a[1] > b[1];
15. });
17. //Iterate jobs array and find slot
18. **for**(auto i: jobs){
19. **int** tmpNum = i[0] - 1;
20. //If slot which before the deadline is empty, this job can be choose
21. **while**(tmpNum >= 0){
22. //If slot is empty, put the profit in it
23. **if**(selected\_work[tmpNum] == -1){
24. selected\_work[tmpNum] = i[1];
25. res += i[1];
26. **break**;
27. }**else**{
28. //else find another empty slot
29. --tmpNum;
30. }
31. }
32. }
34. **return** res;
35. }