Job Sequencing Problem | Set 1 (Greedy Algorithm) - GeeksforGeeks

Given an array of jobs where every job has a deadline and associated profit if the job is finished before the deadline. It is also given that every job takes single unit of time, so the minimum possible deadline for any job is 1. How to maximize total profit if only one job can be scheduled at a time.

Examples:

```
Input: Four Jobs with following deadlines and profits
  JobID    Deadline      Profit
    a        4            20
    b        1            10
    c        1            40
    d        1            30
Output: Following is maximum profit sequence of jobs
        c, a


Input:  Five Jobs with following deadlines and profits
   JobID     Deadline     Profit
     a         2           100
     b         1           19
     c         2           27
     d         1           25
     e         3           15
Output: Following is maximum profit sequence of jobs
        c, a, e
```

We strongly recommend to minimize your browser and try this yourself first.
A **Simple Solution** is to generate all subsets of given set of jobs and check individual subset for feasibility of jobs in that subset. Keep track of maximum profit among all feasible subsets. The time complexity of this solution is exponential.

This is a standard Greedy Algorithm problem. Following is algorithm.

```
1) Sort all jobs in decreasing order of profit.
2) Initialize the result sequence as first job in sorted jobs.
3) Do following for remaining n-1 jobs
.......a) If the current job can fit in the current result sequence
          without missing the deadline, add current job to the result.
          Else ignore the current job.
```

The Following is C++ implementation of above algorithm.

```
// Program to find the maximum profit job sequence from a given array// of jobs with deadlines and profits#include<iostream>#include<algorithm>
using namespace std;
// A structure to represent a job
struct Job
{
    char id;        // Job Id
    int dead;       // Deadline of job
    int profit;   // Profit if job is over before or on deadline
};// This function is used for sorting all jobs according to profit
bool comparison(Job a, Job b)
{
     return (a.profit > b.profit);
}// Returns minimum number of platforms reqquired
void printJobScheduling(Job arr[], int n)
{
    // Sort all jobs according to decreasing order of prfit
    sort(arr, arr+n, comparison);
    int result[n]; // To store result (Sequence of jobs)
    bool slot[n];   // To keep track of free time slots
    // Initialize all slots to be free
    for (int i=0; i<n; i++)
        slot[i] = false;
    // Iterate through all given jobs
    for (int i=0; i<n; i++)
    {
        // Find a free slot for this job (Note that we start
        // from the last possible slot)
        for (int j=min(n, arr[i].dead)-1; j>=0; j--)
        {
            // Free slot found
            if (slot[j]==false)
            {
                result[j] = i;  // Add this job to result
                slot[j] = true; // Make this slot occupied
                break;
            }
        }
    }
    // Print the result
```

```
    for (int i=0; i<n; i++)
        if (slot[i])
            cout << arr[result[i]].id << " ";
}// Driver program to test methods
int main()
{
    Job arr[5] = { {'a', 2, 100}, {'b', 1, 19}, {'c', 2, 27},
                   {'d', 1, 25}, {'e', 3, 15}};
    int n = sizeof(arr)/sizeof(arr[0]);
    cout << "Following is maximum profit sequence of jobs\n";
    printJobScheduling(arr, n);
    return 0;
}
```

Output:

```
Following is maximum profit sequence of jobs
c a e
```

Time Complexity of the above solution is $O(n^2)$. It can be optimized to almost O(n) by using union-find data structure. We will son be discussing the optimized solution.

Sources:
http://ocw.mit.edu/courses/civil-and-environmental-engineering/1-204-computer-algorithms-in-systems-engineering-spring-2010/lecture-notes/MIT1_204S10_lec10.pdf

This article is contributed by **Shubham**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.