Job Sequencing with Deadline

Job scheduling algorithm is applied to schedule the jobs on a single processor to maximize the profits.

The greedy approach of the job scheduling algorithm states that, "Given 'n' number of jobs with a starting time and ending time, they need to be scheduled in such a way that maximum profit is received within the maximum deadline".

Job Scheduling Algorithm

Set of jobs with deadlines and profits are taken as an input with the job scheduling algorithm and scheduled subset of jobs with maximum profit are obtained as the final output.

Algorithm

```
Step1 –  Find the maximum deadline value from the input set
of jobs.
Step2 –  Once, the deadline is decided, arrange the jobs
in descending order of their profits.
Step3 –  Selects the jobs with highest profits, their time
periods not exceeding the maximum deadline.
Step4 –  The selected set of jobs are the output.
```

Examples

Consider the following tasks with their deadlines and profits. Schedule the tasks in such a way that they produce maximum profit after being executed −

S. No.	1	2	3	4	5
Jobs	J1	J2	J3	J4	J5
Deadlines	2	2	1	3	4
Profits	20	60	40	100	80

Step 1

Find the maximum deadline value, dm, from the deadlines given.

```
dm = 4.
```

Step 2

Arrange the jobs in descending order of their profits.

S. No.	1	2	3	4	5
Jobs	J4	J5	J2	J3	J1
Deadlines	3	4	2	1	2
Profits	100	80	60	40	20

The maximum deadline, d_m, is 4. Therefore, all the tasks must end before 4.

Choose the job with highest profit, J4. It takes up 3 parts of the maximum deadline.

Therefore, the next job must have the time period 1.

Total Profit = 100.

Step 3

The next job with highest profit is J5. But the time taken by J5 is 4, which exceeds the deadline by 3. Therefore, it cannot be added to the output set.

Step 4

The next job with highest profit is J2. The time taken by J5 is 2, which also exceeds the deadline by 1. Therefore, it cannot be added to the output set.

Step 5

The next job with higher profit is J3. The time taken by J3 is 1, which does not exceed the given deadline. Therefore, J3 is added to the output set.

```
Total Profit: 100 + 40 = 140
```

Step 6

Since, the maximum deadline is met, the algorithm comes to an end. The output set of jobs scheduled within the deadline are **{J4, J3}** with the maximum profit of **140**.

Example

Following is the final implementation of Job sequencing Algorithm using Greedy Approach –

C C++ Java Python

</> Open Compiler

```
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>

// A structure to represent a Jobs
typedef struct Jobs {
   char id; // Jobs Id
   int dead; // Deadline of Jobs
   int profit; // Profit if Jobs is over before or on deadline
} Jobs;

// This function is used for sorting all Jobss according to
// profit
int compare(const void* a, const void* b){
   Jobs* temp1 = (Jobs*)a;
   Jobs* temp2 = (Jobs*)b;
   return (temp2->profit - temp1->profit);
}

// Find minimum between two numbers.
int min(int num1, int num2){
   return (num1 > num2) ? num2 : num1;
}
int main(){
   Jobs arr[] = {
      { 'a', 2, 100 },
      { 'b', 2, 20 },
      { 'c', 1, 40 },
      { 'd', 3, 35 },
      { 'e', 1, 25 }
   };
   int n = sizeof(arr) / sizeof(arr[0]);
   printf("Following is maximum profit sequence of Jobs: \n");
   qsort(arr, n, sizeof(Jobs), compare);
   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
       for (int j = min(n, arr[i].dead) - 1; j >= 0; j--) {

          // Free slot found
          if (slot[j] == false) {
             result[j] = i;
             slot[j] = true;
             break;
          }
       }
    }

    // Print the result
    for (int i = 0; i < n; i++)
       if (slot[i])
          printf("%c ", arr[result[i]].id);
    return 0;
 }
```

Output

```
Following is maximum profit sequence of Jobs:
c a d
```