|  |
| --- |
| **Experiment No. 7** |
| **Job sequencing with deadline** |
| Date of Performance: 14/03/24 |
| Date of Submission: 21/03/23 |

## Experiment No. 7

**Title:** Job Sequencing with deadline

**Aim:**

To study and implement Job Sequencing with deadline Algorithm

Objective: To introduce Greedy based algorithms

**Theory:**

* Job sequencing 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”.
* We are given n-jobs, where each job is associated with a deadline Di and a profit Pi if the job if finished before the deadline.
* We have single CPU with Non-Primitive Scheduling.
* With each job we assume arrival time is 0, burst time of each job requirement is 1.
* Select a Subset of 'n' jobs, such that, the jobs in the subset can be completed within deadline and generate maximum profit.

Strategy to solve job sequencing with deadlines problem:

Step 1: Arrange the list based on descending order of profits. Read the profits array

from left to right.

Step 2: Fill up the job array using the deadlines.

Step 2.1: If the job array has vacant position at the location indicated by the

deadline, then insert the pi at corresponding index in job array.

Step 2.2: If it is not vacant then search for the less than current deadline indexes

in the job array.

Step 2.3: If empty location is found the insert pi otherwise discard that job.

Step 3: Finally read the job array to get the optimal sequence.

**Example:**

A grid of numbers and letters

Description automatically generatedGiven the jobs, their deadlines and associated profits as shown-

Answer the following questions-

1. Write the optimal schedule that gives maximum profit.
2. Are all the jobs completed in the optimal schedule?
3. What is the maximum earned profit?

**Solution:**

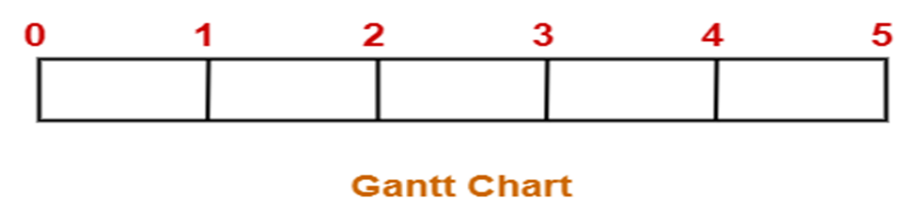
**Step-01:** Sort all jobs in decreasing order of their profit.

A grid of numbers and letters

Description automatically generated

**Step-02:**

* Value of maximum deadline = 5.
* Draw a Gantt chart with maximum time on Gantt chart = 5 units



* Take each job one by one in the order they appear in Step-01 and place the job on Gantt chart as far as possible from 0.

A rectangular object with numbers and letters

Description automatically generated

Here, only job left is job J6 whose deadline is 2.

All the slots before deadline 2 are already occupied.

Thus, job J6 cannot be completed.

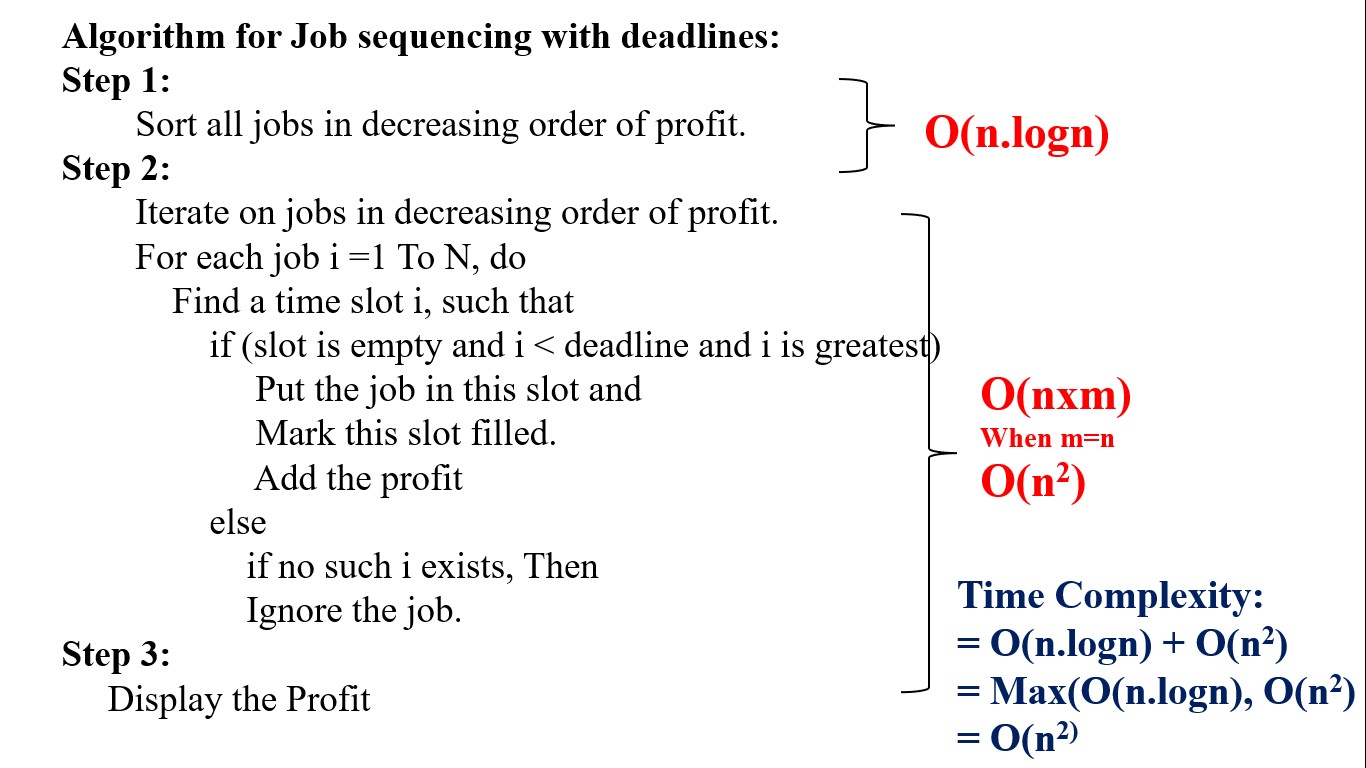
**Maximum earned profit = Sum of profit of all the jobs in optimal schedule**

= Profit of job J2 + Profit of job J4 + Profit of job J3

+ Profit of job J5 + Profit of job J1

= 180 + 300 + 190 + 120 + 200

= 990 units



**Code:**

**#include <stdio.h>**

**struct Job {**

**int id;**

**int deadline;**

**int profit;**

**};**

**void jobSequencing(struct Job jobs[], int n) {**

**int maxDeadline = 0;**

**for (int i = 0; i < n; i++) {**

**if (jobs[i].deadline > maxDeadline)**

**maxDeadline = jobs[i].deadline;**

**}**

**struct Job temp;**

**for (int i = 0; i < n - 1; i++) {**

**for (int j = 0; j < n - i - 1; j++) {**

**if (jobs[j].profit < jobs[j + 1].profit) {**

**temp = jobs[j];**

**jobs[j] = jobs[j + 1];**

**jobs[j + 1] = temp;**

**}**

**}**

**}**

**int result[maxDeadline];**

**for (int i = 0; i < maxDeadline; i++)**

**result[i] = -1;**

**int totalProfit = 0;**

**for (int i = 0; i < n; i++) {**

**for (int j = jobs[i].deadline - 1; j >= 0; j--) {**

**if (result[j] == -1) {**

**result[j] = jobs[i].id;**

**totalProfit += jobs[i].profit;**

**break;**

**}**

**}**

**}**

**printf("Job sequence: ");**

**for (int i = 0; i < maxDeadline; i++) {**

**if (result[i] != -1)**

**printf("%d ", result[i]);**

**}**

**printf("\nTotal profit: %d\n", totalProfit);**

**}**

**int main() {**

**int n;**

**printf("Enter the number of jobs: ");**

**scanf("%d", &n);**

**struct Job jobs[n];**

**printf("Enter the deadline and profit of each job:\n");**

**for (int i = 0; i < n; i++) {**

**printf("Job %d: ", i + 1);**

**scanf("%d %d", &jobs[i].deadline, &jobs[i].profit);**

**jobs[i].id = i + 1;**

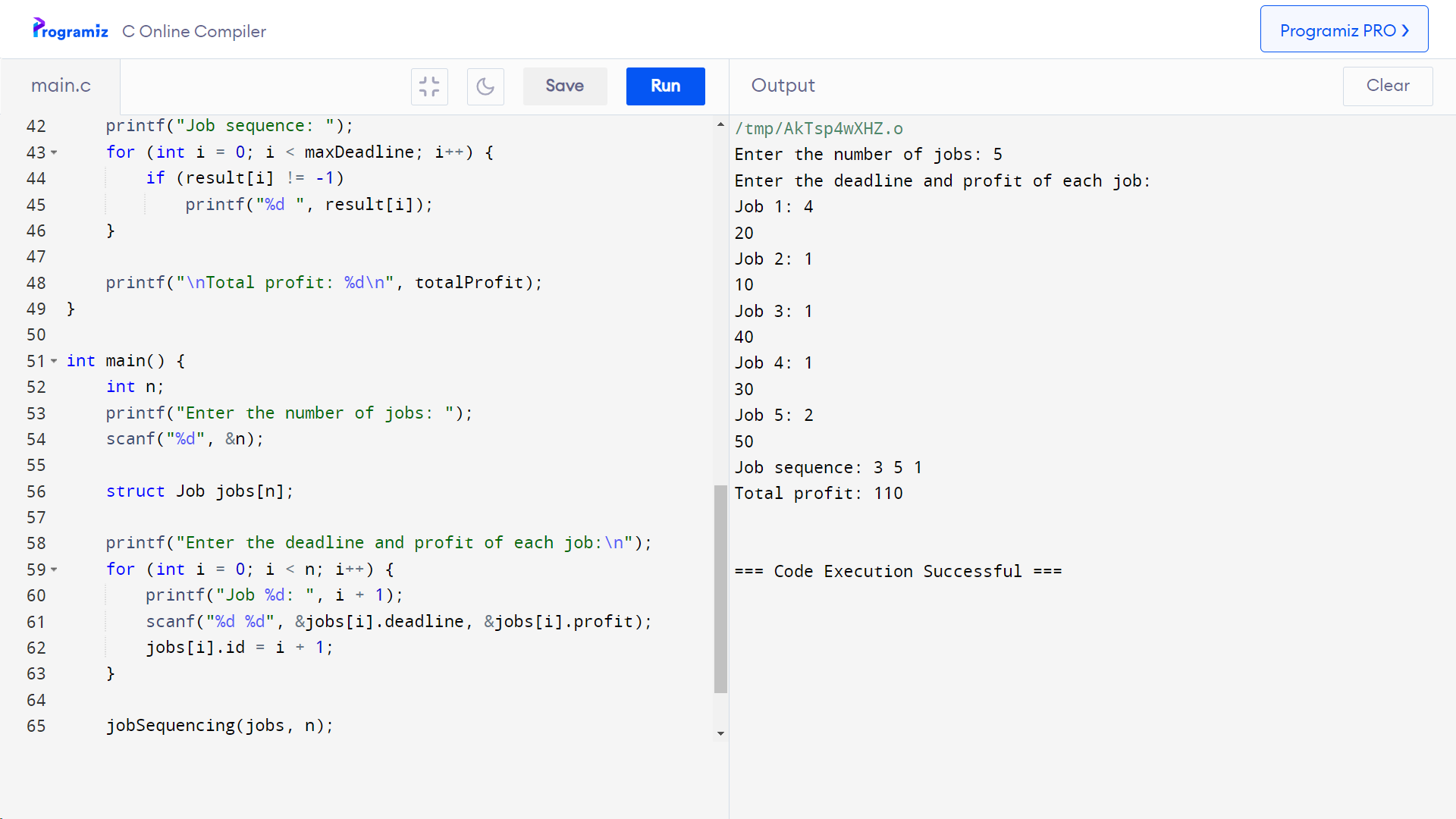
**}**

**jobSequencing(jobs, n);**

**return 0;**

**}**

**Output:**



**Conclusion:**

**The provided code implements the Job Sequencing with Deadline algorithm in C, enabling users to input job deadlines and profits. It then computes and outputs the optimal job sequence and total profit.**