| **Experiment No. 7** |
| --- |
| **Job sequencing with deadline** |
| Date of Performance: 14/3/24 |
| Date of Submission: 21/3/24 |

**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:**

Given the jobs, their deadlines and associated profits as shown-A grid of numbers and letters

Description automatically generated

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:**

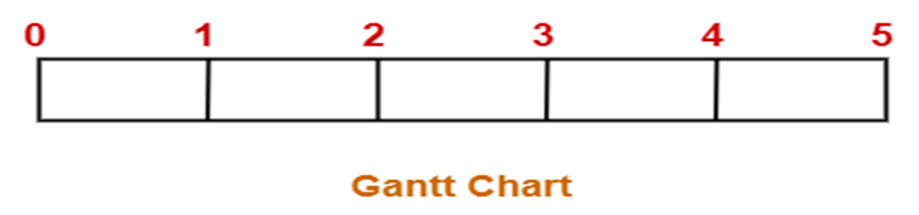
**Step-01:** Sort all jobs in decreasing order of their profit.

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.

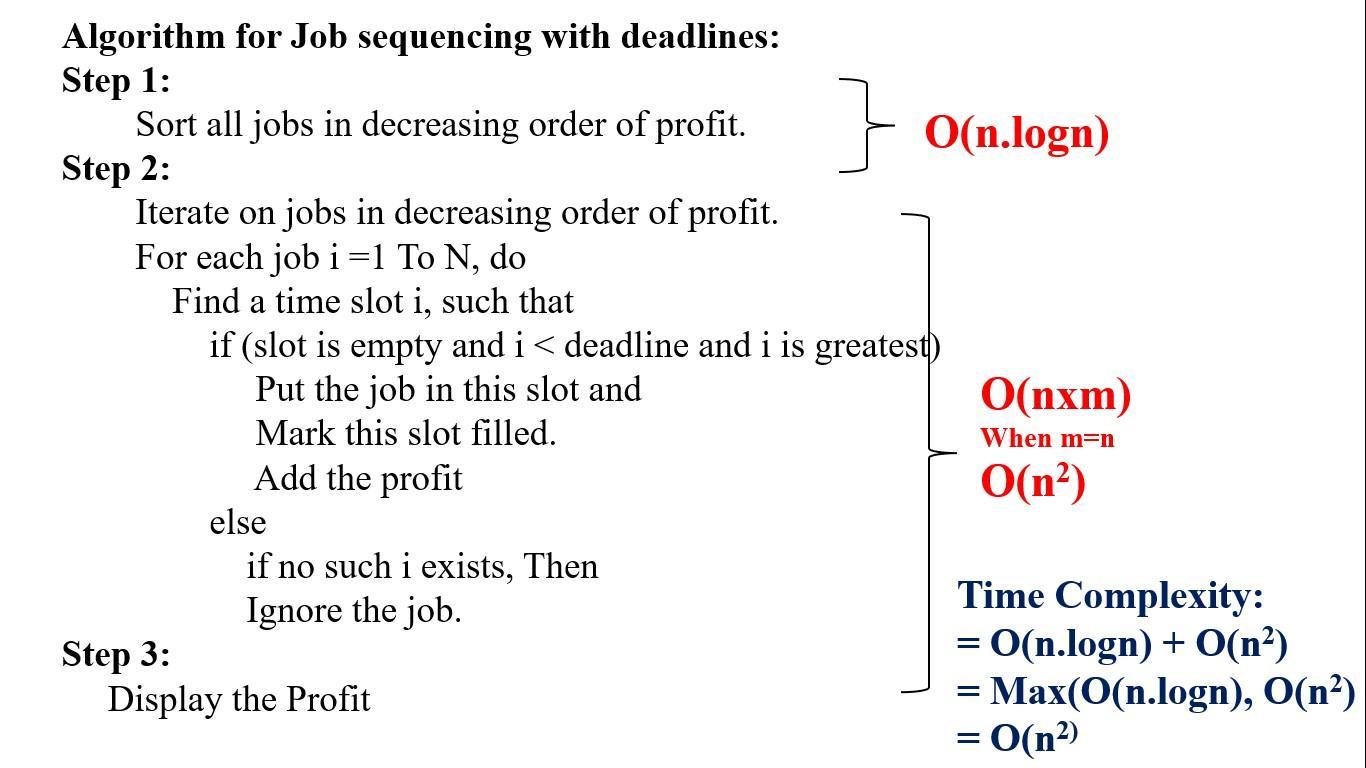
**Maximum earned profit = Sum of profit of all the jobs in optimal schedule**

= 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>

int main()

{

int n;

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

scanf("%d", &n);

int job[10];

int profit[10];

int deadline[10];

int selected[10] = {0};

// Input jobs, profits, and deadlines

printf("Enter the jobs: ");

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

scanf("%d", &job[i]);

}

printf("Enter the profits: ");

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

scanf("%d", &profit[i]);

}

printf("Enter the deadlines: ");

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

scanf("%d", &deadline[i]);

}

// Sort jobs based on profit in descending order (using bubble sort)

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

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

if(profit[j] < profit[j + 1]) {

int temp = profit[j];

profit[j] = profit[j + 1];

profit[j + 1] = temp;

temp = job[j];

job[j] = job[j + 1];

job[j + 1] = temp;

temp = deadline[j];

deadline[j] = deadline[j + 1];

deadline[j + 1] = temp;

}

}

}

// Perform job sequencing

int maxProfit = 0;

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

int j = deadline[i];

while(j > 0 && selected[j] == 1) { // Find the nearest free slot

j--;

}

if(j > 0) { // Slot found

selected[j] = 1;

maxProfit += profit[i];

}

}

// Output the maximum profit and the sequence of jobs

printf("Maximum profit: %d\n", maxProfit);

printf("Job sequence: ");

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

if(selected[i+1] == 1) {

printf("%d ",job[i]);

}

}

printf("\n");

return 0;

}

**Output:**

Enter the number of jobs: 7

Enter the jobs: 1 2 3 4 5 6 7

Enter the profits: 3 5 20 18 1 6 30

Enter the deadlines: 1 3 4 3 2 1 2

Maximum profit: 74

Job sequence: 7 3 4 6

**Conclusion:  
In conclusion, the job sequencing problem involves selecting jobs with associated profits and deadlines to maximize total profit, subject to the constraint that each job must be completed within its deadline.**

**This problem can be efficiently solved using greedy algorithms, such as sorting jobs based on their profits and selecting them in a non-decreasing order of deadlines. Greedy approaches ensure optimal solutions and have a time complexity of O(n log n), where n is the number of jobs. Overall, job sequencing algorithms provide effective solutions for prioritizing tasks to maximize profits while meeting deadlines.**