Experiment No. 7
Job sequencing with deadline
Date of Performance: 14/03/2024
Date of Submission: 21/03/2024

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-

Jobs	J1	J2	J3	J4	J5	J6
Deadlines	5	3	3	2	4	2
Profits	200	180	190	300	120	100

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?

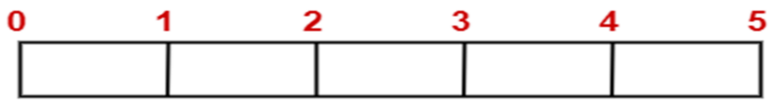
Solution:

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

Jobs	J4	J1	J3	J2	J5	J6
Deadlines	2	5	3	3	4	2
Profits	300	200	190	180	120	100

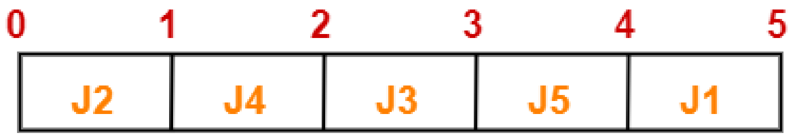
Step-02:

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



Gantt Chart

- 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.



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.

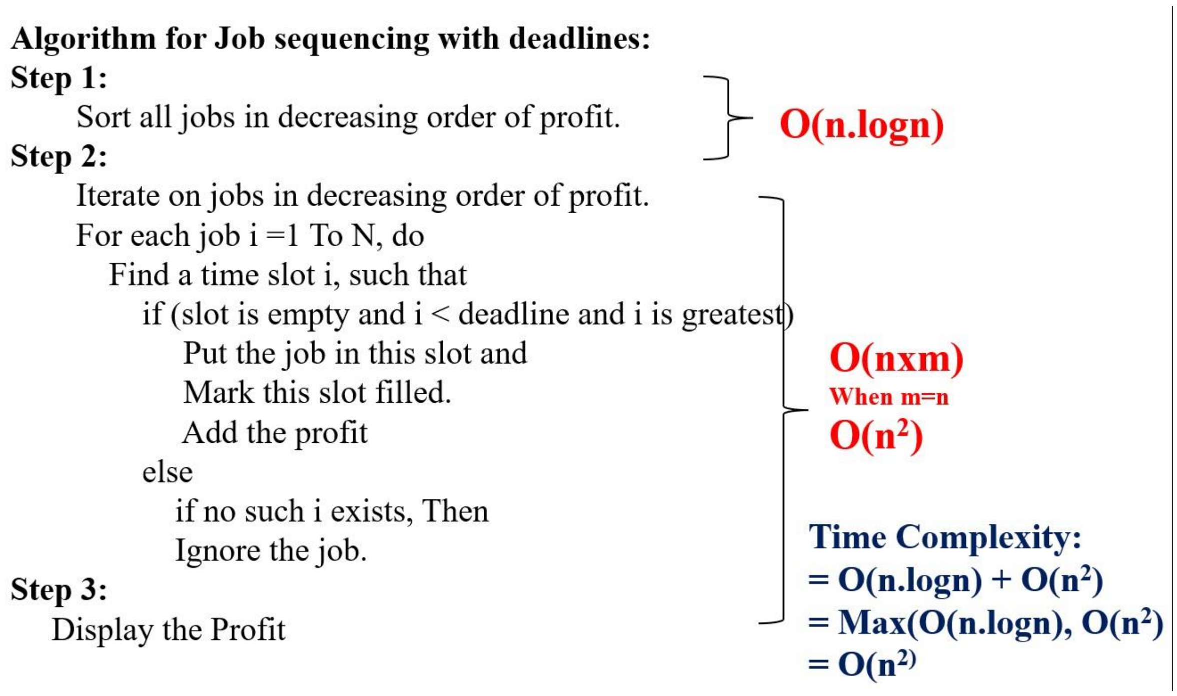
Maximum earned profit = Sum of profit of all the jobs in optimal schedule

= 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



Algorithm for Job sequencing with deadlines:

```
Step 1:
    Sort all jobs in decreasing order of profit.          } O(n.logn)
Step 2:
    Iterate on jobs in decreasing order of profit.
    For each job i =1 To N, do
        Find a time slot i, such that
            if (slot is empty and i < deadline and i is greatest)
                Put the job in this slot and
                Mark this slot filled.                    } O(nxm)
                Add the profit                              When m=n
            else                                            O(n^2)
                if no such i exists, Then
                Ignore the job.
Step 3:
    Display the Profit
```

O(n.logn)

O(nxm)

When m=n

$O(n^2)$

Time Complexity:

$= O(n.logn) + O(n^2)$

$= Max(O(n.logn), O(n^2))$

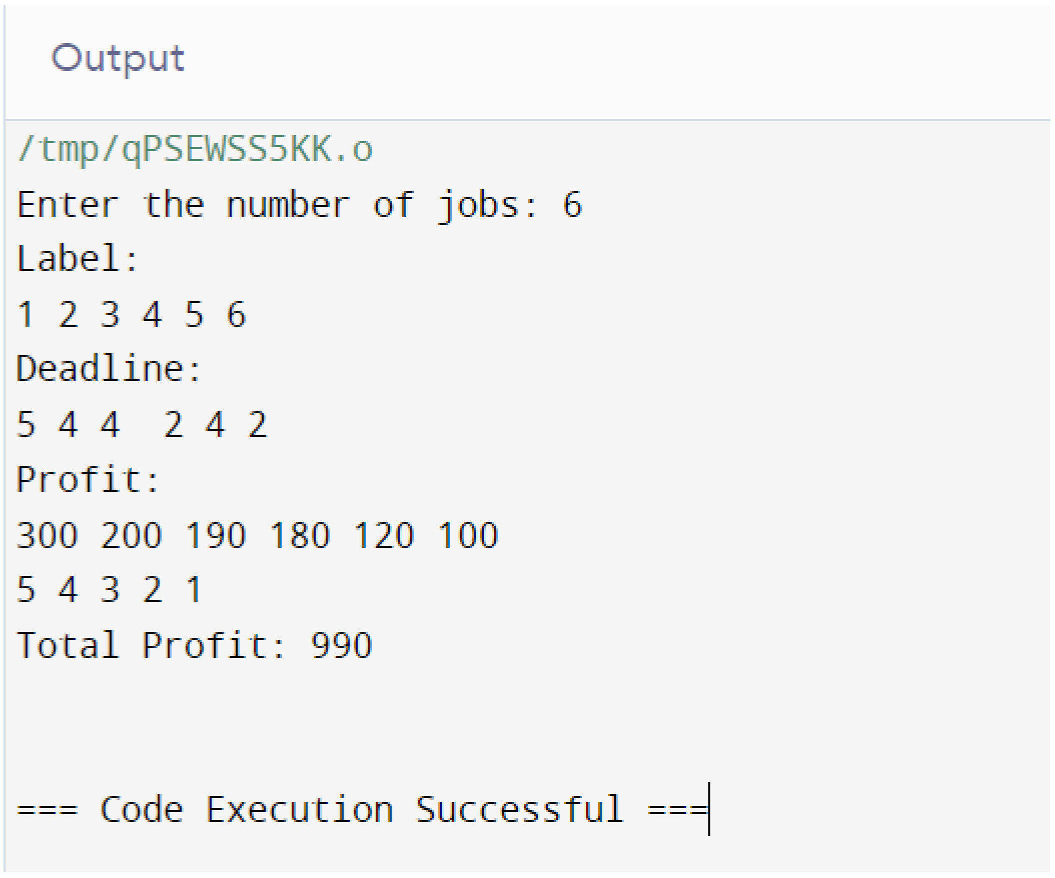
$= O(n^2)$

Code:

```
#include <stdio.h>

#include <stdlib.h>


void swap(int *a, int *b) {

    int temp = *a;

    *a = *b;

    *b = temp;

}


void jobSquencing(int label[], int profit[], int deadline[], int n) {


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

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

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

                swap(&profit[j], &profit[j + 1]);

                swap(&deadline[j], &deadline[j + 1]);

                swap(&label[j], &label[j + 1]);

            }

        }

    }




    int maxDeadline = 0;

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

```
    if (deadline[i] > maxDeadline) {
        maxDeadline = deadline[i];
    }
}


int schedule[maxDeadline + 1];
for (int i = 0; i <= maxDeadline; i++) {
    schedule[i] = -1;
}


int totalProfit = 0;
for (int i = 0; i < n; i++) {
    for (int j = deadline[i]; j > 0; j--) {
        if (schedule[j] == -1) {
            schedule[j] = label[i];
            totalProfit += profit[i];
            break;
        }
    }
}


for (int i = 1; i <= maxDeadline; i++) {
        if (schedule[i] != -1) {
            printf("%d ", schedule[i]);
        }
    }
    printf("\nTotal Profit: %d\n", totalProfit);
}

int main() {
    int i,n,a;
    int label[100];
    int profit[100];
    int deadline[100];
    printf("Enter the number of jobs: ");
    scanf("%d", &n);
    printf("Label: \n");
    for(i=0;i<n;i++){
        scanf("%d", &a);
        label[i]=a;
    }
    printf("Deadline: \n");
    for(i=0;i<n;i++){
        scanf("%d", &a);
        deadline[i]=a;
    }
    printf("Profit: \n");
    for(i=0;i<n;i++){
        scanf("%d", &a);
        profit[i]=a;
    }

    jobSquencing(label, profit, deadline, n);

    return 0;
}
```

Output:



Output

```
/tmp/qPSEWSS5KK.o
Enter the number of jobs: 6
Label:
1 2 3 4 5 6
Deadline:
5 4 4  2 4 2
Profit:
300 200 190 180 120 100
5 4 3 2 1
Total Profit: 990


=== Code Execution Successful ===
```

Conclusion:

Job sequencing with deadlines is a crucial problem in scheduling tasks efficiently. This algorithm ensures that jobs are completed within their respective deadlines while maximizing profit. By considering job deadlines and their profits, the algorithm schedules jobs accordingly. The time complexity of the algorithm is O(n log n), where n is the number of jobs, making it efficient for practical purposes. Efficient job sequencing contributes to optimal resource utilization and timely project completion.