**Aim:** Implementation of Job sequencing with Deadlines

**Theory:**

In job sequencing problem, the objective is to find a sequence of jobs, which is completed within their deadlines and gives maximum profit.

Solution

Let us consider, a set of ***n*** given jobs which are associated with deadlines and profit is earned, if a job is completed by its deadline. These jobs need to be ordered in such a way that there is maximum profit.

It may happen that all of the given jobs may not be completed within their deadlines.

Assume, deadline of **ith** job ***Ji*** is ***di*** and the profit received from this job is ***pi***. Hence, the optimal solution of this algorithm is a feasible solution with maximum profit.

Thus, D(i)>0 for 1⩽i⩽n.

Initially, these jobs are ordered according to profit, i.e. p1⩾p2⩾p3⩾...⩾pn

**Complexity Analysis :**

In this algorithm, we are using two loops, one is within another. Hence, the complexity of this algorithm is **O(n2)**

**Code:**

#include <stdio.h>

#include <stdlib.h>

struct Job

{

    int jobid;

    float profit;

    int deadline;

};

int comparator(const void \*a, const void \*b)

{

    struct Job \*j = (struct Job \*)a;

    struct Job \*j1 = (struct Job \*)b;

    return (j1->profit - j->profit);

}

int minValue(int x, int y)

{

    if (x < y)

        return x;

    return y;

}

int main()

{

    int i, j, n, dmax = 0, counter = 0;

    float total\_profit = 0;

    printf("Enter number of processes:");

    scanf("%d", &n);

    struct Job \*arr = (struct Job \*)malloc(n \* sizeof(struct Job));

    int alloc[n];

    for (i = 0; i < n; i++)

    {

        printf("Enter profit and deadine for job %d :", i + 1);

        scanf("%f %d", &arr[i].profit, &arr[i].deadline);

        arr[i].jobid = i + 1;

        alloc[i] = -1;

        if (arr[i].deadline > dmax)

        {

            dmax = arr[i].deadline;

        }

    }

    qsort(arr, n, sizeof(struct Job), comparator);

    for (i = 1; i <= n; i++)

    {

        for (j = minValue(dmax, arr[i - 1].deadline); j >= 1; j--)

        {

            if (alloc[j] == -1)

            {

                alloc[j] = i - 1;

                counter++;

                total\_profit += arr[i - 1].profit;

                break;

            }

        }

        if (counter == dmax)

        {

            break;

        }

    }

    printf("Sequence:");

    for (i = 1; i <= dmax; i++)

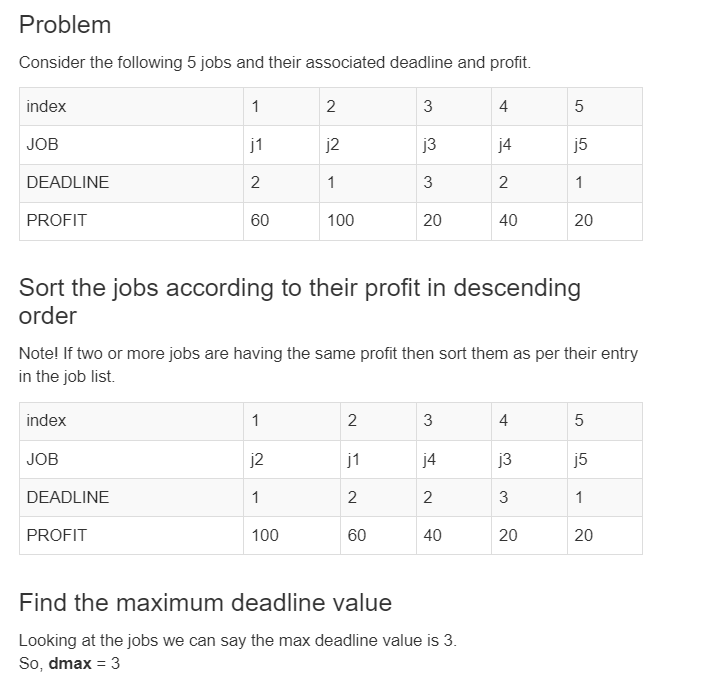
    {

        printf("Job %d ", arr[alloc[i]].jobid);

    }

    printf("\nTotal profit= %f", total\_profit);

}



From this set of jobs, first we select ***J2***, as it can be completed within its deadline and contributes maximum profit.

* Next, ***J1*** is selected as it gives more profit compared to ***J4***.
* In the next clock, ***J4*** cannot be selected as its deadline is over, hence ***J3*** is selected as it executes within its deadline.
* The job ***J5*** is discarded as it cannot be executed within its deadline.

Thus, the solution is the sequence of jobs (***J2, J1, J3***), which are being executed within their deadline and gives maximum profit.

Total profit of this sequence is **100 + 60 + 20 = 180**.

**Output:** 