"Job Sequencing With Deadlines"

.....a Greedy Algorithm

Define-

The sequencing of jobs on a single processor with deadline constraints is called as Job Sequencing with Deadlines.

Here-

- You are given a set of jobs.
- Each job has a defined deadline and some profit associated with it.
- The profit of a job is given only when that job is completed within its deadline.

Assumptions-

- Only one processor is available for processing all the jobs.
- Processor takes one unit of time to complete a job.
- Non-preemption is there.

The problem states-

"How can the total profit be maximized if only one job can be completed at a time?"

Approach to Solution-

- A feasible solution would be where each job in the subset gets completed within its deadline.
- Value of the feasible solution would be the sum of profit of all the jobs contained in the subset.
- An optimal solution of the problem would be a feasible solution which gives the maximum profit.

Greedy Algorithm-

Greedy Algorithm is adopted to determine how the next job is selected for an optimal solution.

The greedy algorithm described in the next slide always gives an optimal solution to the job sequencing problem.

Steps to solve-

Step-01:

- Sort all the given jobs in decreasing order of their profit.

Step-02:

- Check the value of maximum deadline.
- Draw a Gantt chart where maximum time on Gantt chart is the value of maximum deadline.

Step-03:

- Pick up the jobs one by one.
- Put the job on Gantt chart as far as possible from 0 ensuring that the job gets completed before its deadline.

Example Problem-

Given the jobs, their deadlines and associated profits as shown, answer the following questions-

- Write the optimal schedule that gives maximum profit.
- What is the maximum earned profit?

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

Solution-

Step1: Sort all the given 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

Step2:

Value of maximum deadline = 5.

So, draw a Gantt chart with maximum time on Gantt chart = 5 units as shown-

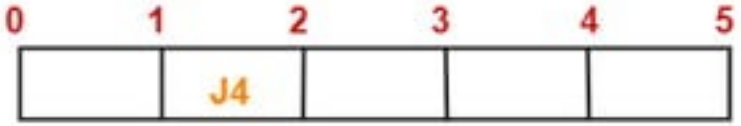
Gantt Chart

Now,

- We take each job one by one in the order they appear in Step1.
- We place the job on Gantt chart as far as possible from 0.

Step3:

- We take job J4.
- Since its deadline is 2, so we place it in the first empty cell before deadline 2 as-



Step4:

- We take job J1.
- Since its deadline is 5, so we place it in the first empty cell before deadline 5 as-

Step5:

- We take job J3.
- Since its deadline is 3, so we place it in the first empty cell before deadline 3 as-

Step6:

- We take job J2.
- Since its deadline is 3, so we place it in the first empty cell before deadline 3.
- Since the second and third cells are already filled, so we place job J2 in the first cell as-

Step7:

•Now, we take job J5.

•Since its deadline is 4, so we place it in the first empty cell before deadline 4 as-

Now,

- The only job left is job J6 whose deadline is 2.
- All the slots before deadline 2 are already occupied.
- Thus, job J6 can not be completed.

The optimal schedule is-

J2 , J4 , J3 , J5 , J1

This is the required order in which the jobs must be completed in order to obtain the maximum profit.

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

Time Complexity Analysis:

Time for sorting on basis of profits = O (n log n)

Next, two loops are used one within the other, for which complexity = O (n^2)

Hence, the overall complexity of this algorithm becomes O(n^2).

Program-

```
#include<iostream>
#include<algorithm>
using namespace std;
struct Job                                  //stores details of each job
{  char id;
   int dead;
   int profit;  };

bool comparison(Job a, Job b)               //used in sorting acc. to profits
{   return (a.profit > b.profit); }

void printJobScheduling(Job arr[], int n)
{
   sort(arr, arr+n, comparison);
   int result[n];
   bool slot[n];
   for (int i=0; i<n; i++)
      slot[i] = false;
```

```
 for (int i=0; i<n; i++)
   {  for (int j=min(n, arr[i].dead)-1; j>=0; j--)
     {  if (slot[j]==false)
       { result[j] = i;
         slot[j] = true;
         break;
       }  }  }
   for (int i=0; i<n; i++)
     if (slot[i])
       cout << arr[result[i]].id << " ";
}
int main()
{
   Job arr[] = { {'a', 5, 200}, {'b', 3, 180}, {'c', 3, 190}, {'d', 2, 300},
                 {'e', 4, 120}, {'f', 2, 100} };
   int n = sizeof(arr)/sizeof(arr[0]);
   cout << "Following is maximum profit sequence of jobs \n";
   printJobScheduling(arr, n);
   return 0;
}
```

Thank You