Job sequencing with deadlines

Job sequencing with deadlines

- There are n jobs to be processed on a machine.
- Each job i has an integer deadline $d_i \geq 0$ and profit $p_i \geq 0$.
- Pi is earned if the job is completed within its deadline.
- The job is completed if it is processed on a machine for unit time.
- Only one machine is available for processing jobs.
- Only one job is processed at a time on the machine.

Job sequencing with deadlines

- Problem: n jobs, S={1, 2, …, n}, each job i has a deadline $d_i \geq 0$ and a profit $p_i \geq 0$. We need one unit of time to process each job and we can do at most one job each time. We can earn the profit p_i if job i is completed by its deadline.

i	1	2	3	4	5
p_i	20	15	10	5	1
d_i	2	2	1	3	3

The optimal solution = {1, 2, 4}.

The total profit = 20 + 15 + 5 = 40.

Example(Obtain an optimal sequence for the given jobs)

N =5 (p1,p2,p3,p4,p5) = (20,15,10,5,3) and (d1,d2,d3,d4,d5) = (2,2,1,3,3).

Job	Profit	Deadline
1	20	2
2	15	2
3	10	1
4	5	3
5	3	3

Step1: Arrange according to descending order of profit (it already sorted in descending order in question)

SNO	Job	Profit	Deadline
1	Job 1	20	2
2	Job 2	15	2
3	Job 3	10	1
4	Job 4	5	3
5	Job5	3	3

Job Sequencing with deadline

- Step1 :Create an array J[ ] which stores the jobs. Initally J[ ] will be

1	2	3	4	5
0	0	0	0	0

- Step 2: Add ith job in array J[ ] at the index denoted by its deadline di. First job is job1.The deadline for this job is 2. Hence insert job1 in the array J[ ] at index 2

1	2	3	4	5
0	Job1	0	0	0

Job Sequencing with deadline

- Step 3: Next job is Job 2 its deadline is 2 but its already occupied. So check for the previous slot . Its empty so insert at index 1.

1	2	3	4	5
Job2	Job1	0	0	0

- Step 4: Next job is Job 3 its deadline is 1 but its already occupied. So job 3 is not part of the solution
- Step 5:Next job is Job 4 its deadline is 3 so insert at index 3.

1	2	3	4	5
Job2	Job1	Job4	0	0

Job Sequencing with deadline

- Step 6: Next job is Job 5 its deadline is 3 but its already occupied. So job 5 is not part of the solution
- So final sequence is

1	2	3	4	5
Job2	Job1	Job4	0	0

- So final optimum solution is Job 1 ,2 and 4 and processing sequence is **<2,1,4>** and the total profit earned is **40**

Job sequencing with deadlines

- A feasible solution is a subset of jobs J such that each job is completed by its deadline.
- An optimal solution is a feasible solution with maximum profit value.

Example : Let $n = 4$, $(p_1,p_2,p_3,p_4) = (100,10,15,27)$, $(d_1,d_2,d_3,d_4) = (2,1,2,1)$

Feasible solution

Sr.No.	Feasible Solution	Processing Sequence	Profit value
(i)	(1,2)	(2,1)	110
(ii)	(1,3)	(1,3) or (3,1)	115
(iii)	(1,4)	(4,1)	127 is the optimal one
(iv)	(2,3)	(2,3)	25
(v)	(3,4)	(4,3)	42
(vi)	(1)	(1)	100
(vii)	(2)	(2)	10
(viii)	(3)	(3)	15
(ix)	(4)	(4)	27

Job Sequencing with deadline

Example

Job	Profit	Deadline
1	100	2
2	10	1
3	15	2
4	27	1

Step1: Arrange according to descending order of profit

SNO	Job	Profit	Deadline
1	Job 1	100	2
2	Job 4	27	1
3	Job 3	15	2
4	Job 2	10	1

Job Sequencing with deadline

- Create an array J[ ] which stores the jobs. Initally J[ ] will be

1	2	3	4
0	0	0	0

- Add ith job in array J[ ] at the index denoted by its deadline di
- First job is job1.The deadline for this job is 2. Hence insert job1 in the array J[ ] at index 2

1	2	3	4
0	Job1	0	0

Job Sequencing with deadline

- Next job is p4. its deadline is 1. So insert at index 1

1	2	3	4
Job4	Job1	0	0

- Next job is Job 3 its deadline is 2 but its already occupied. So job 3 is not part of the solution
- Next job is job2, its deadline is 1 but its already occupied. So job 3 is not part of the solution
- So final optimum solution is Job 1 and Job 4 and processing sequence is **<4,1>** and the total profit earned is **127**

Job Sequencing with deadline

- Maximum number of jobs that can be completed is

  = Min(n,maxdelay(di))

- Consider 5 jobs with profits (p1,p2,p3,p4,p5) = (20,15,10,5,1) and maximum delay allowed (d1,d2,d3,d4,d5) = (2,2,1,3,3).
- Here maximum number of jobs that can be completed is

  = Min(n,maxdelay(di))

  = Min(5,3)

  = 3.

Algorithm

- Algorithm JS(d,p,n)

// d[1…n] be deadiline where di>0

//p[1…n] be profit

// The jobs are ordered such that p[1] >= p[2]>= ……>=p[n]

Initialize Jobset[1..n] to zero // Jobset contains the jobs included in the optimal solution which will give maximum profit

Initialize slot[1…n] = false

Algorithm

```
for(i=1;i<=n ;i++)
{
for (int j = deadline[i]; j>0; j--)
    If(! Slot[j])
    {
       Slot[ j ] = true;
       Jobset[ j ] = i;
       break;
    }
Return jobset
```

Complexity

- Time Complexity of job sequencing with deadlines algorithm is $O(n^2)$, because the basic operation of computing sequence is within two nested loops.

More examples

- Solve following job sequencing problem

  N =7 profits(p1,p2…p7) = {3,5,20,18,1,6,30) and deadline (d1, d2…..d7) =(1,3,4,3,2,1,2)

- Solve following job sequencing problem

  N =5 profits(p1,p2…p5) = {100,19,27,25,15) and deadline (d1, d2…..d7) =(2,1,2,1,3)