Greedy Algorithms
JOB SEQUENCING WITH DEADLINES

SMITA SANKHE

Assistant Professor

Department of Computer Engineering

Problem Statement

The problem is stated as below.

- There are n jobs to be processed on a machine.
- Each job i has a deadline $d_i \geq 0$ and profit $p_i \geq 0$.
- Pi is earned iff the job is completed by 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.

Problem Statement (Contd..)

- 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, $(p_1,p_2,p_3,p_4) = (100,10,15,27)$, $(d_1,d_2,d_3,d_4) = (2,1,2,1)$

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

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	

GREEDY ALGORITHM TO OBTAIN AN OPTIMAL SOLUTION

- Consider the jobs in the non increasing order of profits subject to the constraint that the resulting job sequence J is a feasible solution.

ALGORITHM

```
Algorithm 4.2: JobSequence(job[1..n], p[1..n], d[1..n], list[1..n], n)
1.   Initialize list with zeros.          // make all n time slots empty
2.   profit = 0;                          // profit = total profit
3.   For i = 1 to n do
4.       k = d[i];                        // set ith job's deadline to k
5.       While( k > 0 )
6.           If( list[k] = 0 )                // if kth slot is vacant,
7.               list[k] = job[i];            // assign ith job to kth slot
8.               add p[i] to profit;          // accumulate total profit
9.               go to step 3;                // exit from while-loop (break)
10.          Endif
11.          Decrement k by 1;            // otherwise get preceding slot
12.      Endwhile
13.  Endfor
14.  Print list
```

COMPLEXITY ANALYSIS OF JS ALGORITHM

- Sorting (sort p_i into decreasing order) and selection of jobs take ***O(nlogn)*** time in total
- Consider n jobs in turn.
- For each job, inserting the job into partial solution using its deadline take ***O(n)*** time.
- Checking whether the new solution is still feasible tales ***O(n)*** time.
- Hence total running time, in worst case is $O(n^2)$

EXAMPLE:

let n = 5, $(p_1,\ldots, p_5) = (20,15,10,5,1)$ and $(d_1,\ldots,d_5) = (2,2,1,3,3)$.

Let n=7 , $(p_1,\ldots, p_7) =(35,30,25,20,15,12,5)$ and (d1…d7)=(3,4,4,2,3,1,2)