Design and Analysis of Algorithm (DAA)

Greedy Algorithms

(Job Scheduling with Deadline)

Dr. Dayal Kumar Behera

School of Computer Engineering
KIIT Deemed to be University, Bhubaneswar, India

What is Job Sequencing Problem?

Given a list of Jobs where each job has a deadline and associated profit if the job is completed before the deadline.

Assumption:

- Uniprocessor System (single processor capable of performing one job at a time)
- No Preemption: Jobs cannot be preempted mid-execution
- Each job takes one unit of time.(could be 1 hour or 1 month or any unit)
- Minimum deadline for any job is 1

Job Sequencing with Deadline

n jobs or tasks

Each job i has:

deadline : d_i

Profit or gain : p_i

Goal: *complete the jobs or tasks in such a way that the profit is maximum.*

Mathematical Interpretation

The objective is to maximize the total profit earned by scheduling the jobs within their respective deadlines.

Maximize : $\sum_{1 \le i \le n} p_i x_i$

Subject to: $\sum_{1 \le i \le n} x_i = \max(d_i)$

x_i is a binary decision variable:

$$x_i = \begin{cases} 1 & if\ job\ i\ is\ scheduled(completed)\ within\ its\ deadline \\ 0 & otherwise \end{cases}$$

Approach to Solution

A feasible solution would be a subset of jobs where each job of 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.

Approach to Solution: Brute-force Approach

Step 1: **Generate all subsets** of the jobs:

Step 2: Subset Evaluation and Finding the profit

- Check deadlines for each subset:

  [We need to ensure that each job in a subset can be completed within its deadline. If two jobs are scheduled for the same time slot (conflicting deadlines), that subset will be invalid]

- Calculate the total profit for valid subsets and find the maximum profit.

Example

index	1	3	4
Jobs (J)	J1	J2	J3
Deadline(D)	2	2	1
Profit (P)	50	10	25

Solve the job scheduling problem with the goal of maximizing profit without missing the deadlines using a brute-force approach.

Hints: Generate all possible subsets of jobs and find the most optimal sequence that maximizes the profit.

Brute-force solution

index	1	3	4
Jobs (J)	J1	J2	J3
Deadline(D)	2	2	1
Profit (P)	50	10	25

Generate all subsets of the jobs:

The possible subsets are as follows:

{}

{J1}, {J2}, {J3}

{J1, J2}, {J1, J3}, {J2, J3}

{J1, J2, J3}

Brute-force solution

index	1	3	4
Jobs (J)	J1	J2	J3
Deadline(D)	2	2	1
Profit (P)	50	10	25

Subset Evaluation and Finding the profit:

Subset: Ø
Profit: 0 (no jobs)

Subset: {J1}
Deadline: 2 (Job J1 can be scheduled within the 2nd time slot)
Profit: 50

Subset: {J1, J2}
Deadline conflict, as J1 and J2 both need the same time slot.
Invalid subset (both jobs cannot be scheduled)

……..

The subset with the maximum profit is {J1, J3} with a total profit of 75.

Brute-force solution: Complexity

Generating all subsets takes $O(2^n)$.

For each subset, checking deadlines and calculating profit takes $O(n)$.

Overall time complexity: $O(2^n \cdot n)$

This is an exponential time complexity, which makes this approach inefficient for large numbers of jobs.

Can it be improved?

Job Sequencing with Deadline

Greedy Algorithm Steps:

- Sort the jobs in decreasing order of their profit.
- For each job taken from sorted list do
  Find a suitable time slot in the Gantt chart
- calculate the profit for all the allocated feasible jobs.

Job Sequencing with Deadline

Greedy:

Sort the jobs in decreasing order of profits: $p_1 \geq p_2 \geq . . . \geq p_n$

for t : 1..n

    S(t) $\leftarrow$ 0

//end for

for i : 1..n

    Schedule job i in the latest possible free slot meeting its deadline;
    if there is no such slot, do not schedule i.

//end for

Problem: Not necessarily all the job going to be scheduled. But maintaining time slots for all jobs. (Space complexity increases)

Can it be reduced?

Job Sequencing with Deadline

Greedy:

Sort the jobs in decreasing order of profits: $p_1 \geq p_2 \geq . . . \geq p_n$

```
d ← max di
for t : 1..d
        S(t) ← 0
end for
for i : 1..n
    [find the latest possible free slot meeting the deadline]
end for
```

Maximum time slots = maximum deadline

Job Sequencing with Deadline

Greedy:

Sort the jobs in decreasing order of profits: $p_1 \geq p_2 \geq . . . \geq p_n$

d ← max di

for t : 1..d

    S(t) ← 0

end for

for i : 1..n

    //find the latest possible free slot meeting the deadline

    Find the largest t such that (S(t) = 0 and t ≤ di), if found S(t) ← i

end for

Job Sequencing with Deadline

Algorithm GreedyJobSeqWD(T, D, P)

Input: Three arrays T, D, P, each of size n containing task, deadlines and profits for each job

Output: The Job sequence in the form of an array timeslot "S" and the maximum profit "Profit"

Sort the jobs in decreasing order of profits: $p_1 \geq p_2 \geq . . . \geq p_n$

```
d ← max di
Profit ←  0
for t : 1..d
        S(t) ← 0
end for
for i : 1..n
        k ← D[i]
        while k >= 1
        if S[k]==0 then
            S[k] ← T[i]
            Profit = Profit + P[i]
        //end if
        k ← k -1
end for
Return (S, Profit)
```

Example

index	1	2	3	4	5
Task (T)	**T1**	**T2**	**T3**	**T4**	**T5**
Deadline(D)	**3**	**1**	**4**	**2**	**2**
Profit (P)	**10**	**30**	**20**	**50**	**40**

Step 1: sort the jobs according to the profit in the decreasing order

index	1	2	3	4	5
Task (T)	**T4**	**T5**	**T2**	**T3**	**T1**
Deadline(D)	**2**	**2**	**1**	**4**	**3**
Profit (P)	**50**	**40**	**30**	**20**	**10**

Example

(Sorted List)

index	1	2	3	4	5
Task (T)	**T4**	**T5**	**T2**	**T3**	**T1**
Deadline(D)	**2**	**2**	**1**	**4**	**3**
Profit (P)	**50**	**40**	**30**	**20**	**10**

Step 2: Maintain timeslot array S whose size is same as the maximum deadline among all the jobs. Initial value of S=0.

Max deadline: 4

Index	1	2	3	4
S	0	0	0	0

Example

index	1	2	3	4	5
Task (T)	**T4**	**T5**	**T2**	**T3**	**T1**
Deadline(D)	**2**	**2**	**1**	**4**	**3**
Profit (P)	**50**	**40**	**30**	**20**	**10**

Step 3: find the latest possible free slot meeting the deadline

i=1

Index	1	2	3	4
S		T4		

```
for i =1
        k ← D[i]            2
        while k >= 1        True
          if S[k]==0        True
            S[k] ← T[i]     S[2] = T4
            Profit = Profit + P[i]
          //end if
        k ← k -1
end for
Return (S, Profit)
```

Example

index	1	2	3	4	5
Task (T)	T4	T5	T2	T3	T1
Deadline(D)	2	2	1	4	3
Profit (P)	50	40	30	20	10

i= 2

Index	1	2	3	4
S	T5	T4		

```
for i =2
        k ← D[i]            2
        while k >= 1        True
          if S[k]==0        False
            S[k] ← T[i]
            Profit = Profit + P[i]
          //end if
        k ← k -1            1
end for
Return (S, Profit)
```

Example

index	1	2	3	4	5
Task (T)	T4	T5	T2	T3	T1
Deadline(D)	2	2	1	4	3
Profit (P)	50	40	30	20	10

i=3

T2 with deadline 1, there is no free slot. Hence, T2 is not scheduled.

Index	1	2	3	4
S	T5	T4		

Example

index	1	2	3	4	5
Task (T)	**T4**	**T5**	**T2**	**T3**	**T1**
Deadline(D)	**2**	**2**	**1**	**4**	**3**
Profit (P)	**50**	**40**	**30**	**20**	**10**

i=4

T3 with deadline 4, can be scheduled in timeslot 4.

Index	1	2	3	4
S	T5	T4		T3

Example

index	1	2	3	4	5
Task (T)	T4	T5	T2	T3	T1
Deadline(D)	2	2	1	4	3
Profit (P)	50	40	30	20	10

i=5

T1 with deadline 3, can be scheduled in timeslot 3.

Index	1	2	3	4
S	T5	T4	T1	T3

Job Sequence: T5→T4→T1→T3

Max Profit: 120

Another Example

Jobs	Profit	Deadline
J1	85	5
J2	25	4
J3	16	3
J4	40	3
J5	55	4
J6	19	5
J7	92	2
J8	80	3
J9	15	7

Step 1: Sorting in decreasing order of profit

Jobs	Profit	Deadline
J7	92	2
J1	85	5
J8	80	3
J5	55	4
J4	40	3
J2	25	4
J6	19	5
J3	16	3
J9	15	7

1	2	3	4	5	6	7
J4	**J7**	**J8**	**J5**	**J1**		**J9**

The optimal sequence is J4, J7, J8, J5, J1, J9.

Profit: 40+92+80+55+85+15=36740+92+80+55+85+15=367

Greedy: Using Array

Sorting n jobs takes $O(n\log n)$.

checking deadlines and calculating profit takes in two nested loops takes $O(n^2)$.

Overall time complexity: $n\log n + n^2 = O(n^2)$

Can it be improved?

Job Sequencing using Disjoint-set

Data-Structure: Disjoint-Set (Tree representation)

Initial Sets: maxDeadline + 1 (0 to maxDeadline)

FIND-SET returns the latest available time slot

The time slot returned is always maximum (free slots).

Call **UNION(t-1, t)** If we want to assign a time slot ‘t’ to a job and make ‘t-1’ as the representative. (‘t-1’ is the parent of ‘t’).

This means that all future queries for time slot t would now return the latest time slot available for set represented by t-1.

Job Sequencing using Disjoint-set

Algorithm GreedyJobSeqWD(T, D, P)

Input: Three arrays T, D, P, each of size n containing task, deadlines and profits for each job

Output: The Job sequence in the form of a selected sets "S" and the maximum profit "Profit"

Sort the jobs in decreasing order of profits: $p_1 \geq p_2 \geq . . . \geq p_n$

```
d ← max di
Profit ← 0
S ← { }
for t : 0..d
        MAKE-SET(t)
end for
for i : 1..n
        t = FIND-SET(D[i])
        if t > 0 then
            S ← S U {T[i]}
            Profit = Profit + P[i]
            UNION(FIND-SET(t-1), t)
        end if
end for
return (S, Profit)
```

Example

(Jobid, deadline, profit):

(a, 2, 100) (b, 1, 19) (c, 2, 27) (d, 1, 25) (e, 3, 15)

The optimal sequence is a, c, e.

Profit: 100 + 27 + 15 = 142

Greedy: Using Disjoint-Set

Sorting the jobs: Sorting the jobs based on profit takes O(nlogn), where n is the number of jobs.

Disjoint Set Operations: Both the **FIND-SET** and **UNION** operations take almost constant time.

Overall time complexity: O(nlogn).


Each of your
actions will
have an
impact on your
future.


Once you know
who is walking
with you on your path.
you will never
be afraid.

Thank you