Greedy_Algorithms

June 15, 2022

[15]:

```
```

```
<IPython.core.display.HTML object>
```

[1]:

```
from IPython.display import Image
```

0.1 Greedy method :

- General method,
- Applications
  - Job sequencing with deadlines,
  - Fractional knapsack problem,
  - Minimum cost spanning trees
    - Prims,
    - Kruskal,
  - Single source shortest path problem
    - Dijkstra.

0.2 Features and Bugs of the Greedy Paradigm

- First, for many problems, it's surprisingly easy to come up with one or even multiple greedy algorithms that might plausibly work. But it can be hard to assess which greedy approach is the most promising.
- Second, the running time analysis is often a one-liner. For example, many greedy algorithms boil down to sorting plus a linear amount of extra processing, in which case the running time of a good implementation would be $O(n \log n)$, where n is the number of objects to be sorted.
- Finally, it's often difficult to figure out whether a proposed greedy algorithm actually returns the correct output for every possible legal input. It is hard to prove the correctness of greedy algorithms because most of them are not correct, meaning there exist inputs for which the algorithm fails to produce the desired output.

0.3 Job Sequencing with deadlines

- Given an array of jobs having a specific deadline and associated with a profit, provided the job is completed within the given deadline. The task is to maximize the profit by arranging the jobs in a schedule, such that only one job can be done at a time.

0.3.1 Example

[2]:
```
Image(filename = "gd5.png", width = "100%")
```

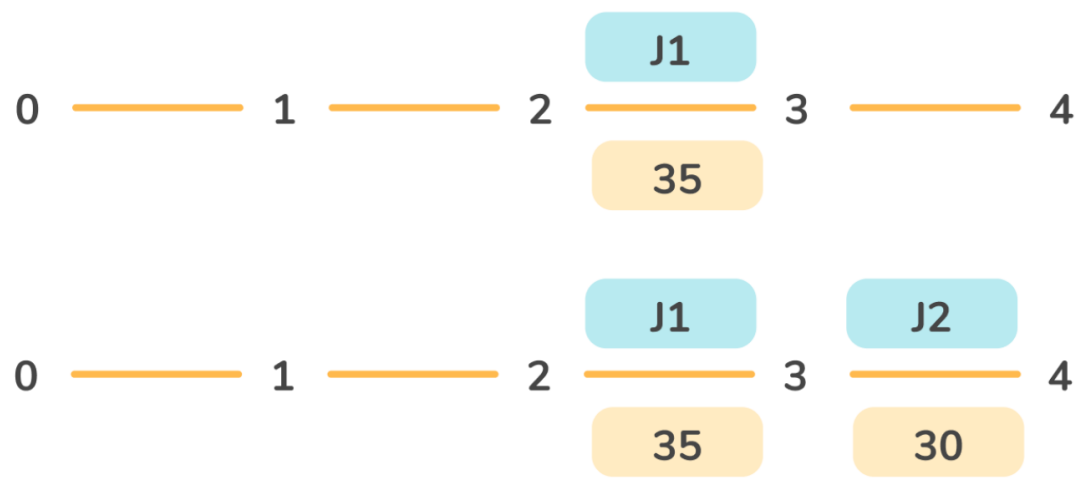
[2]:

n=7

Jobs	J1	J2	J3	J4	J5	J6	J7
Profits	35	30	25	20	15	12	5
Deadlines	3	4	4	2	3	1	2

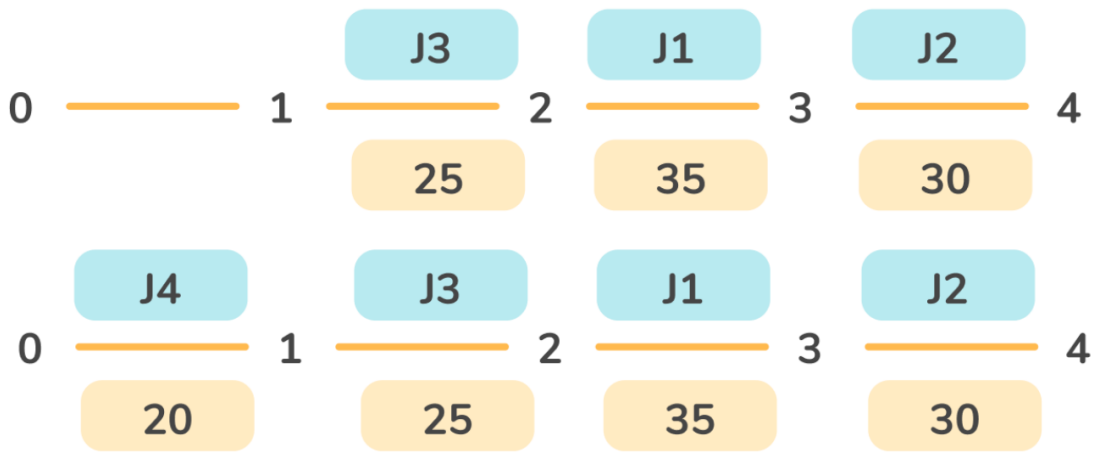
[3]:
```
Image(filename = "gd6.png", width = "100%")
```

[3]:



[4]:
```
Image(filename = "gd7.png", width = "100%")
```

[4]:



Total Profit = 20 + 25 + 35 + 30 = 110

0.4 Algorithm

- Sort the jobs based on decreasing order of profit.
- Iterate through the jobs and perform the following:
  - Choose a Slot i if:
    - Slot i isn't previously selected.
    - i < deadline
    - i is maximum
  - If no such slot exists, ignore the job and continue.

0.5 Example

- Given

[5]:
```
Image(filename = "gd8.png", width = "100%")
```

[5]:

Job ID	1	2	3	4	5
Deadline	2	3	2	1	3
Profit	20	38	16	10	30

- Sort in decreasing order of profits:

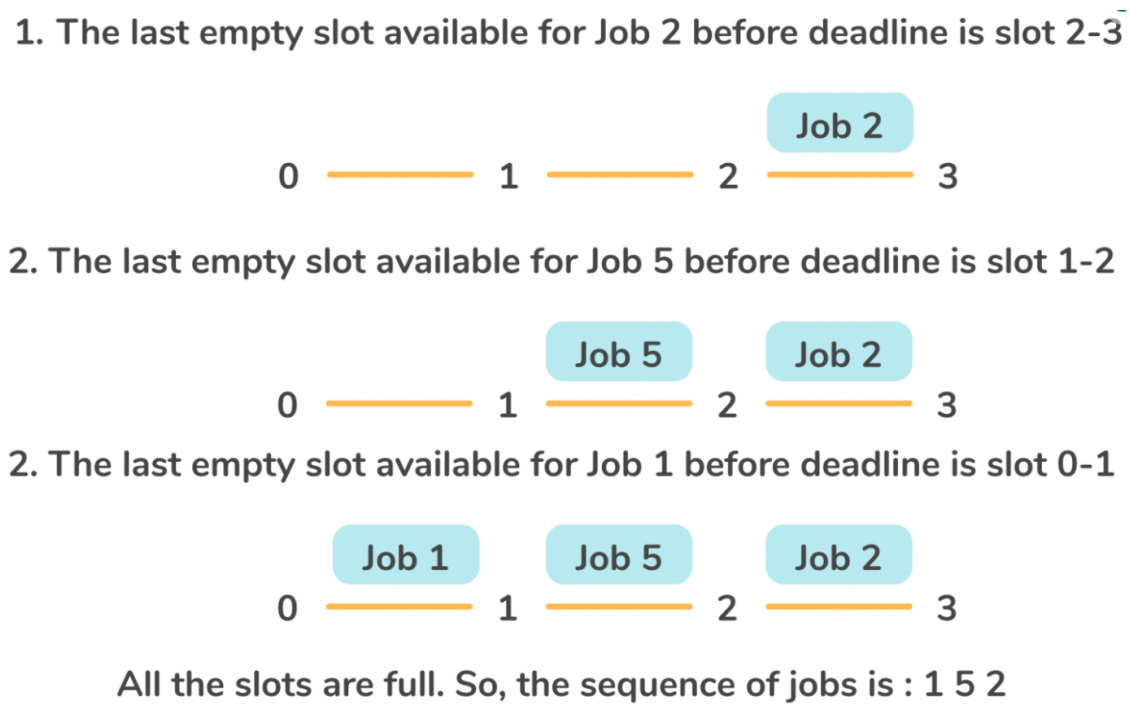
[8]:
```
Image(filename = "gd9.png", width = "100%")
```

[8]:

Job ID	2	5	1	3	4
Deadline	3	3	2	2	1
Profit	38	30	20	16	10

[10]:
```
Image(filename = "gd10.png", width = "100%")
```

[10]:




```
for i = 1 to n do
  Set k = min(dmax, DEADLINE(i))
  //where DEADLINE(i) denotes deadline of ith job

  while k >= 1 do
    if timeslot[k] is EMPTY then
      timeslot[k] = job(i)
      break
    endif
```

```
    Set k = k - 1

  endwhile

endfor
```