Dynamic Programming --- Telescope Scheduling

Telescope Scheduling

Coin in a line

0/1 Knapsack

1. Telescope Scheduling

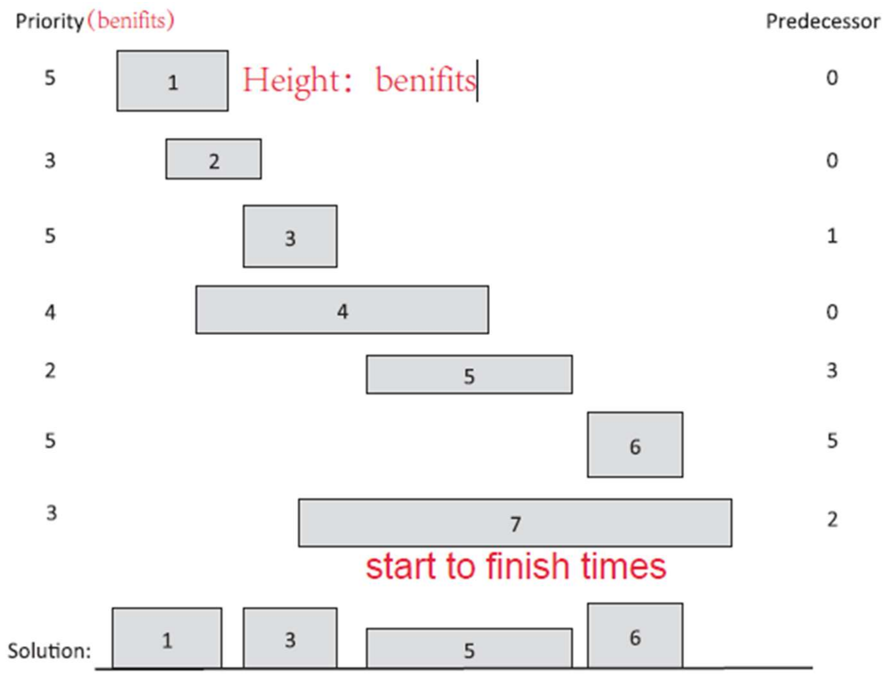


Figure 12.6: The telescope scheduling problem. The left and right boundary of each rectangle represent the start and finish times for an observation request. The height of each rectangle represents its benefit. We list each request's benefit on the left and its predecessor on the right. The requests are listed by increasing finish times. The optimal solution has total benefit 17.

2. Explain the problem

Telescope Scheduling problem is that Given a list L, of observation requests, we need to schedule these observation requests **in a non-conflicting way** and **maximize the total benefit of the observations**.

si start time

fi finish time

bi benefit

pred(i) predecessor, to be the largest index, j<I, make request I and j don't conflict. If no such index, return 0.

3. Algorithm

```
$B[0] \leftarrow 0$
**for** $i = 1$ to $n$ **do**
    $B[i] \leftarrow \max\{B[i-1], B[P[i]] + b_i\}$
```

After this algorithm completes, the benefit of the optimal solution will be $B[n]$

Runtime: O(n)

4. Explain the algorithm

Simple subproblem:

Bi = the maximum benefit that can be achieved with the first i requests in L. So, as a boundary condition, we get that Bo = 0.

Subproblem Optimality

The core to solve TS question is to make sure whether Bi should include observation i or not.

Case 1:

If including observation i, then B[i] = B[p[i]] + bi

B[p[i]] is the total maximum benefit without conflicting to i observation before.

Case 2:

If not including observation i, then B[i] = B[i-1]

We choose the greater value as Bi between Case 1 and Case 2.

Thus Bi = Max{B[i-1], B[p[i]] + bi}

Subproblem Overlap

```
B[0] <- 0
for i <-1 to n do
    B[i] = Max{B[i-1], B[p[i]] + bi}
Return B[n]
```

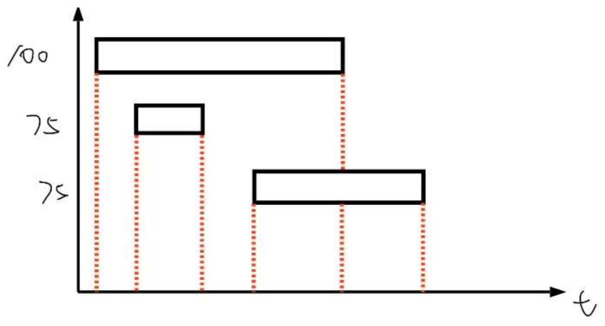
Therefore, the maximum value of Task Scheduling problem is B[n].

5. Is it greedy or dynamic or some other type of algorithm, explain why?

False Start 1: Brute Force, take O(n*2^n) time,

False Start 2: Greedy Method

Proof by counter example:



-- give an example or given an instance of the problem, solve it

见ipad