Tsinghua University

DISCRETE OPTIMISTIC

Fall 2024

Homework 1

Lin Zejin March 12, 2025

- **Collaborators:** I finish this homework by myself.

Problem 1. If d_j is the sequence $-1, 1, 2, 3, 4, 4$, then all of this are legal. However, first if we choose $d_1 = -1$, now none of $d_2 \sim d_5$ is available, which is what the algorithm will find, *i.e.* $S = \{1, 6\}$. However, $S = \{2, 3, 4, 5\}$ is a larger viewable set. So the algorithm is not correct.

Denote $\mathrm{OPT}[j]$ as the size of optimal solution among $d_j, d_{j+1}, \cdots, d_n$. Then the Bellman equation will be

$$\mathrm{OPT}[j] = \min_{t \text{ reachable from } j} \{\mathrm{OPT}[j+1], \mathrm{OPT}[t] + 1\}$$

where $\mathrm{OPT}[n] = 1$. The detail is below:

$\mathrm{OPT}[n] = 1$.
for j from $n-1$ to 1 **do**
  $\mathrm{OPT}[j] = \min_{t \text{ reachable from } j} \{\mathrm{OPT}[j+1], \mathrm{OPT}[t] + 1\}$.
end for
return $\mathrm{OPT}[1]$

Problem 2. Denote $\mathrm{OPT}[i]$ as the minimum number of rounds when the node i call all his subordinates (do not need direct subodinates)

If node i has direct subordinates $j_1, \cdots, j_k$, WLOG we asume $\mathrm{OPT}[j_1] \geq \mathrm{OPT}[j_2] \geq \cdots \geq \mathrm{OPT}[j_k]$ Then easy to comfirm that the Bellman equation is like

$$\mathrm{OPT}[i] = \max_{1 \leq t \leq k} \{\mathrm{OPT}[t] + t\}$$

So the algorithm will be The answer is $\mathrm{OPT}(n)$ where n is the root node.

Algorithm 1 $\mathrm{OPT}(i)$

---

compute $\mathrm{OPT}[j] = \mathrm{OPT}(j)$ for each j direct subordinate of i
sort $\mathrm{OPT}[j]$ decreasingly. Obtain $\mathrm{opt}_1, \cdots, \mathrm{opt}_k$.
$\mathrm{OPT}[i] = \max\limits_{1 \leq t \leq k} \{\mathrm{OPT}[t] + t\}$

Problem 3. Denote $\mathrm{OPT}[n][k]$ as the maximum possible return of k-shot strategy in $1 \sim n$ days.

The Bellman equation will be

$$\mathrm{OPT}[n][k] = \min\{\mathrm{OPT}[n-1][k], \min_{1\le j\le n-1}\{\mathrm{OPT}[j-1][k-1] + p[n] - p[j]\}\}$$

in which the first term is the case that we do not sell on the last day and the second term is the case that we sell on the last day.

So the algorithm will be

Algorithm 2 OPT

for j from 0 to n, t from 0 to k **do**
  if $j = 0$ **or** $t = 0$ **then**
    $\mathrm{OPT}[j,t] \leftarrow 0$
  end if
  $\mathrm{OPT}[j,t] \leftarrow \min\{\mathrm{OPT}[j-1,t], \min_{1\le l\le n-1}\{\mathrm{OPT}[l-1,t-1] + p[j] - p[l]\}\}$
end for

The answer is $1000 \times \mathrm{OPT}(n,k)$ and the time complexity is $O(n^2k)$. The actual k-shot strategy can be given by tracing the choice of each state (n,k).

Problem 4. Denote $\mathrm{OPT}[n][H]$ as the maximum total grade, given the functions $f_1, \cdots, f_n$ and total hours H. We can enumerate the hours spending on the last project to obtain a Bellman equation

$$\mathrm{OPT}[n][H] = \max_{0\le j\le H} \mathrm{OPT}[n-1][H-j] + f_n(j)$$

Then there is an possible algorithm that has time complexity $O(nH)$.

Algorithm 3 OPT

for k from 0 to n, t from 0 to H **do**
  if $k = 0$ **or** $t = 0$ **then**
    $\mathrm{OPT}[k,t] \leftarrow 0$
  end if
  $\mathrm{OPT}[k,t] \leftarrow 0$
  for l from 0 to t **do**
    if $\mathrm{OPT}[k,t] \le \mathrm{OPT}[k-1,t-j] + f_k(j)$ **then**
      $\mathrm{OPT}[k,t] \leftarrow \mathrm{OPT}[k-1,t-j] + f_k(j)$
      $\mathrm{REC}[k,t] = j$.
    end if
  end for
end for

Then $\frac{1}{n}\mathrm{OPT}[n,H]$ is the maximum average grades and $\mathrm{REC}[k,t]$ is the spending hours at project k at the state of (k,t), so we can decide the hours that spend on each project by tracing it.

The time complexity is $O(H^2n)$

Problem 5. (a) Indeed, we can prove every schedule can be scheduled in increasing order of their deadlines.

If jobs $(s_i, s_i + t_i)$ is a proper schedule, and $s_1 < s_1 + t_1 \le s_2 < s_2 + s_2 \le \cdots \le s_k < s_k + t_k$. Assume $d_i > d_{i+1}$. Then

$$s_i < s_i + t_i \le s_{i+1} < s_{i+1} + t_{i+1} \le d_{i+1} < d_i$$

Therefore

$$s_i < s_i + t_{i+1} = s_i + t_{i+1} \leq s_i + t_i + t_{i+1} < d_i$$

since $d_i > s_{i+1} + t_{i+1} \geq s_i + t_i + t_{i+1}$.

So we can exchange the order of job i, $i+1$. By Bubble sort algorithm, we can obtain an available schedule that execute in increasing order of their deadlines.

(b) We only need to consider the schedule that execute in increasing order of their deadlines. Denote $\mathrm{OPT}[n][D]$ as the optimal size of jobs $1, 2, \cdots, n$ and final dealines D.

WLOG, we may assume $d_1 \leq d_2 \leq \cdots \leq d_n$. Each time we consider whether we choose job n or not. Since job n has the last deadlines in that state, we can arrange for it to be the last to execute. So the Bellman equation is

$$\mathrm{OPT}[n][D] = \begin{cases} 0 & t_n > \min\{d_n, D\} \\ \max\{\mathrm{OPT}[n-1][D], \mathrm{OPT}[n-1][\min\{d_n, D\} - t_n] + 1\} & \end{cases}$$

So the algorithm is

Algorithm 4 OPT

for k from 0 to n, t from 0 to D **do**
  if $k = 0$ **or** $t = 0$ **or** $t_k > \min\{d_k, t\}$ **then**
    $\mathrm{OPT}[k, t] \leftarrow 0$
  end if
  $\mathrm{OPT}[k, t] \leftarrow \mathrm{OPT}[k-1, t]$.
  $\mathrm{REC}[k, t] = 0$.
  if $\mathrm{OPT}[k, t] < \mathrm{OPT}[k-1][\min\{d_k, t\} - t_k] + 1$ **then**
    $\mathrm{OPT}[k, t] \leftarrow \mathrm{OPT}[k-1][\min\{d_k, t\} - t_k] + 1$
    $\mathrm{REC}[k, t] = 1$.
  end if
end for

Then $\mathrm{OPT}[n, D]$ is the optimal solution and $\mathrm{REC}[k, t]$ marks the choice in the state (k, t) which can be traced to form a avaible schedule.

Its time complexity is $O(nD)$

Problem 6. f