EECS 340: Assignment 5

Shaochen (Henry) ZHONG, sxz517

Due and submitted on 04/06/2020
EECS 340, Dr. Koyutürk

Problem 1

Hi TA! I did **(ii)** and **(vi)** on **Problem 1(a)**. Thanks.

(a)

(i) The last operation base on operator o_{last} for $o_{\text{last}} \in \{o_1, o_2, o_3, ..., o_{n-1}\}$.

(ii) Assuming the last operation will be $T[i,j]\ o_{\text{last}}\ T[j+1,k] = T_{\text{final}}$, where T_{final} provides the optimal output for $x_1o_1x_2o_2...x_{n-1}o_{n-1}x_n$. There must be:

- $T[i,j]$ provides the optimal output for $x_io_ix_{i+1}o_{i+1}...x_j$.
- $T[j+1,k]$ provides the optimal output for $x_{j+1}o_{j+1}x_{j+2}o_{j+2}...x_k$.

Proof. Without lost of generality, assume $T[i,j]$ is not optimal for $x_io_ix_{i+1}o_{i+1}...x_j$, there must be an $T'[i,j]$ where $T'[i,j] > T[i,j]$. Then we may have $T'[i,j]\ o_{\text{last}}\ T[j+1,k] > T_{\text{final}}$.
However, it is known that $T[i,j]$ provides the optimal output for $x_io_ix_{i+1}o_{i+1}...x_j$, for $T'[i,j] > T[i,j]$ to exist, $T[i,j]$ is no longer optimal. Thus we may say $T[i,j]$ is optimal by contradiction. □

(iii)

- Subproblem: $T[i,j]$ for the optimal output of $x_io_ix_{i+1}o_{i+1}...x_j$.
- Overall problem: $T[1,n]$ for the optimal output of $x_1o_1x_2o_2...x_{n-1}o_{n-1}x_n$.

(iv)

$$T[i,k] = \begin{cases} x_i & \text{if } \ i = k \\ max\{T[i,j]\ o_j\ T[j+1,k]\} & \text{for } \ i \le j \le k \end{cases}$$

(v) We have $T(n) = \Theta(n^3)$. As there are $\Theta(n^2)$ subproblems and for each subproblem of $T[i,k]$, there are $\Theta(k-i)$ choices (which is essential $\Theta(n)$). Also each choice takes $\Theta(1)$ too execute.
$\Longrightarrow T(n) = \Theta(n^2) \cdot \Theta(n) \cdot \Theta(1) = \Theta(n^3)$

(vi) For X being a list of operands `[x_1, x_2, x_3, ... , x_n]`, likewise O being a list of operators `[o_1, o_2, o_3,... , o_n-1]`.

(Discussed with Yuhui ZHANG on this)

Algorithm 1 Largest-Output(X, O, n)

```
1: Create two empty n × n 2D arrays: E and S.
2: procedure LARGEST-OUTPUT(X, O, n)
3:     for i ← 1 to n do
4:         E[i, i] ← X[i]                          ▷ Get operands on diagonal
5:     for cur ← 2 to n do
6:         for i ← 1 to n − cur + 1 do
7:             j = i + cur − 1
8:             E[i, j] ← −∞
9:             for k ← i to j − 1 do
10:                holder ← eval("E[i, k] + O[k] + E[k + 1, j]
11:                if holder > E[i, j] then
12:                    E[i, j] = holder
13:                    S[i, j] = k
14:    Return E, S
```

Algorithm 2 Display(X, O, s, i, j)

```
1: procedure DISPLAY(X, O, s, i, j)
2:     if i == j then
3:         print X[i]
4:     else
5:         print Display(X, O,s, i, s[i,j] )
6:         print O[s[i, j]]
7:         print Display(X, O,s, s[i,j]+1, j )
```

(b)

(i) Determine the last coin c_{last} which will be added to the change set $\{c_1, c_2, c_3, ..., c_{\text{last}-1}\}$. Where the sum of elements of such set plus c_{last} should be equal to t.

(iii)

- Subproblem: $f(v)$ provides the minimum number of coins to make change for v, given n types of coin denominations are available. There must be $v \in [0, t]$ and v must be equal to the sum of some coins from the given n types of coin denominations, to represent a value that is achievable by given coins.
- Overall problem: $f(t)$ provides the minimum number of coins to make change for t, given n types of coin denominations are available.

(iv) Let n_c be a type of coin from the provided n types of coins, which means n_c can potentially hold n types of values. We will try all n_c(s) as we don't know which coin to use for the c_1.

$$f(v) = \begin{cases} 0 & \text{if } v = 0 \\ min\{f(v - n_c) + 1\} & \text{if } v > 0 \end{cases}$$

(v) We have $T(n) = \Theta(nt)$. There are $\Theta(t)$ subproblems, as we can divide at most t groups of coin(s) for a total value of t (as for all coins being value 1), and for each group there are at most n possible choices to iterate (thus $\Theta(n)$). Each choice takes $\Theta(1)$ to execute.
$\implies T(n) = \Theta(t) \cdot \Theta(n) \cdot \Theta(1) = \Theta(nt)$

(c)

(i) Determine what edit to make (or not making any edit) when reached the last character of X.

(iii) Let $X_k = x_1x_2...x_k$ and likewise for Y_k.

- Subproblem: $D(i, j)$ provides the shortest edit distance between X_i and Y_j.
- Overall problem: $D(m, n)$ provides the shortest edit distance between X_m and Y_n.

(iv)

$$D(i,j) = \begin{cases} j & \text{if } i = 0 \\ i & \text{if } j = 0 \\ D(i-1, j-1) & \text{if } x_i = y_j \\ min\{D(i, j-1) + 1, D(i-1, j) + 1, D(i-1, j-1) + 1\} & \text{otherwise} \end{cases}$$

(v) We have $T(n) = \Theta(mn)$. A matrix of $m \cdot n$ is filled when two strings are listing their possible reductions of chars, thus there are $\Theta(mn)$ subproblems. Each subproblem takes $\Theta(1)$ to execute as all edit options cost the same.
$\implies T(n) = \Theta(mn) \cdot \Theta(1) = \Theta(mn)$

Problem 2

(a)

Proof. Suppose we have r_{ij} for the optimal exchange between currencies $i \to j$, with an substructure of $r_{ij} = r_{ik} \cdot r_{kj}$ (for $k \in [i, j]$). We want to show both r_{ik} and r_{kj} must be the optimal exchange between currencies $i \to k$ and $k \to j$.

Without lost of generality, assume we have r'_{ik} where $r'_{ik} > r_{ik}$, then there must be $r'_{ik} \cdot r_{kj} > r_{ij}$. Which suggest r_{ij} is no longer optimal for the exchange between currencies $i \to j$ and the assumption is void. Thus, we proved substructure r_{ik} must be at its optimal structure by contradiction. And an optimal r_{ij} is only obtainable with optimal substcutures. □

(b)

It is certainly possible. Assume the optimal exchange path found by the abovementioned procedure is a long path with a tiny amount of gain after each exchange, e.g., for r_{az} we have to do $r_{ab} \cdot r_{bc} \cdot ... \cdot r_{yz}$, where after each exchange the accumlated value become 1.0001 times of before, and an direct exchange rate between currencies $a \to z$ is 1.0001 as well.

Supposely the best exchange path with no exchange fee is to do as many of positive exchanges as possible, as the accumlated values will only be larger and larger, result in a maximum amount of currency z at the end. But if we set $f(k) = 0.5$ for all exchanges, the more exchange we do, the more money we lost – since the tiny amount of gain of one exchange cannot cover the high exchange fee. In this case, the best exchange path would be a direct exchange between currencies $a \to z$ is 1.

Since the optimal exchange path with $f(k) = 0.5$ is not the same as the optimal path produced by the abovementioned procedure, we have proven the statement.