EECS 336

Litong Wang

Homework 4

May 4, 2016

Problem 1

Solution:

I think the greedy algorithm discussed in the paper is incorrect. The algorithm proposed in the paper always picks the symbol that comes earlier in string Y.

The paper claims that it leaves as much opportunity as possilble for the remaining symbols to be select. However, we can easily find counter example to prove it wrong. For example, consider these two input array:

$$X = [A, B, B, B]$$

$$Y = [B, A, B, B]$$

In the first iteration, accoding to the algorithm presented in the paper. We need to discard A in X, then the result of this algorithm for the longest common subsequence will be BB. However, we can easily note that the correct result should be ABB.

Problem 2

Solution:

Let $C_1(T)$ to be the cost function mentioned in class and $C_2(T)$ to be the cost function presented in textbook.

$$C_1(T) = \sum_{i=1}^{n} P_i \times level(a_i) + \sum_{i=0}^{n} Q_i \times (level(e_i) - 1)$$

$$C_2(T) = \sum_{i=1}^{n} P_i \times level(a_i) + \sum_{i=0}^{n} Q_i \times level(e_i)$$

We can easily see that

$$C_2(T) - C_1(T) = \sum_{n=i}^{j} Q_n$$

During the construction of the optimal binary tree, we will use the following formula:

$$C(i,j) = \min_{i \le n \le j} \{C(i,k-1) + C(k+1,j) + \sum_{m=i}^{j} P_m + \sum_{m=i-1}^{j} Q_m\}$$

Assume we find the optimal solution $k' \in [i,j]$ by using $C_2(T)$, then we can say that for any $k \in [i,j]$:

$$C_2(i,k-1) + C_2(k+1,j) \le C_2(i,k'-1) + C_2(k'+1,j)$$

$$\Leftrightarrow$$

$$C_1(i,k-1) + \sum_{n=i}^{k-1} Q_n + C_1(k+1,j) + \sum_{n=k+1}^{j} Q_n \le C_1(i,k'-1) + \sum_{n=i}^{k'-1} Q_n + C_1(k'+1,j) + \sum_{n=k'+1}^{j} Q_n$$

$$\Leftrightarrow$$

$$C_1(i,k-1) + C_1(k+1,j) \le C_1(i,k'-1) + C_1(k'+1,j)$$

We can see that we will choose the same k' to be our optimal solution using $C_1(T)$. Hence, these two cost model will produce the same optimal binary tree.

Problem 3

The Shortest Maximum-Weight Common Subsequence Problem:

Solution:

Table:

- $W[i,j]$ = the maximum-weight of a common subsequence of the prefix $x_1, x_2, \cdots, x_i$ and $y_1, y_2, \cdots, y_j$
- $L[i,j]$ = the shortest length of a maximum-weight common subsequence of the prefix $x_1, x_2, \cdots, x_i$ and $y_1, y_2, \cdots, y_j$

$0 \le i \le m$

$0 \le j \le n$

Output for the problem:

The shortest maximum-weight of a common subsequence.

Base Case:

- $i = 0, W[i,j] = 0, L[i,j] = 0$
- $j = 0, W[i,j] = 0, L[i,j] = 0$

Recursion Case:

- $1 \le i \le m$ *and* $1 \le j \le n$
- If $x_i = y_j$ then $W[i,j] = weight(x_i) + W[i-1, j-1], L[i,j] = L[i-1,j-1] + 1$

- If $x_i \neq y_j$, $W[i-1,j] > W[i,j-1]$ then $W[i,j] = W[i-1,j], L[i,j] = L[i-1,j]$
- If $x_i \neq y_j$, $W[i,j-1] > W[i-1,j]$ then $W[i,j] = W[i,j-1], L[i,j] = L[i,j-1]$
- If $x_i \neq y_j$, $W[i,j-1] = W[i-1,j]$ then $W[i,j] = W[i,j-1] = W[i-1,j], L[i,j] = \min\{L[i,j-1], L[i-1,j]\}$

Correctness:

Table $W[i,j]$ represent the maximum weight of the common subsequence of prefix $x_1, x_2, \cdots, x_i$ and $y_1, y_2, \cdots, y_j$. If the $x_i = y_j$, then we must append x_i to the result common subsequence S. If we don't append x_i, and we get a result S'. We must can find a common subsequence S with weight is larger than weight of S'. In case of $x_i \neq y_j$, we should let $W[i,j] = \max\{W[i,j-1], W[i-1,j]\}$. If we don't choose the maximum weight, then we cannot meet the goal finding the "maximum" weight. If $x_i \neq y_j$ & $W[i,j-1] = W[i-1,j]$, In order to find the shortest maximum weight common subsequence, we need to take the length of the result into consideration. In order to achieve the result with mimimum length, clealy we should choose $L[i,j] = \min\{L[i,j-1], L[i-1,j]\}$.

Space complexity:

- The table $W[i,j]$ has $m \times n$ entries, so the space complexity is $\mathcal{O}(mn)$
- The table $L[i,j]$ has $m \times n$ entries, so the space complexity is $\mathcal{O}(mn)$

Time complexity:

- The runtime of computing each entry of $W[i,j]$ and $L[i,j]$ is $\mathcal{O}(1)$
- The total runtime of this algorithm is $\mathcal{O}(mn)$

The Minimum-Weight Longest Common Subsequence Problem:

Solution:

Table:

- $W[i,j]$ = the minimum-weight of a common subsequence of the prefix $x_1, x_2, \cdots, x_i$ and $y_1, y_2, \cdots, y_j$
- $L[i,j]$ = the longest length of a maximum-weight common subsequence of the prefix $x_1, x_2, \cdots, x_i$ and $y_1, y_2, \cdots, y_j$

$0 \leq i \leq m$

$0 \leq j \leq n$

Output for the problem:

The longest common subsequence with minimum-weight

Base Case:

- $i = 0, W[i,j] = 0, L[i,j] = 0$
- $j = 0, W[i,j] = 0, L[i,j] = 0$

Recursion Case:

- $1 \leq i \leq m$ *and* $1 \leq j \leq n$
- If $x_i = y_j$ then $L[i,j] = L[i-1,j-1] + 1, W[i,j] = W[i-1,j-1] + weight(x_i)$
- If $x_i \neq y_j$, $L[i-1,j] > L[i,j-1]$ then $L[i,j] = L[i-1,j], W[i,j] = W[i-1,j]$
- If $x_i \neq y_j$, $L[i,j-1] > L[i-1,j]$ then $L[i,j] = L[i,j-1], W[i,j] = W[i,j-1]$
- If $x_i \neq y_j$, $L[i,j-1] = L[i-1,j]$ then $L[i,j] = L[i,j-1] = L[i-1,j], W[i,j] = \min\{W[i,j-1], W[i-1,j]\}$

Correctness:

Table $L[i,j]$ represent the maximum length of the common subsequence of prefix $x_1, x_2, \cdots, x_i$ and $y_1, y_2, \cdots, y_j$. If the $x_i = y_j$, then we must append x_i to the result common subsequence S. If we don't append x_i, and we get a result S'. We must can find a common subsequence S with length is larger than length of S'. In case of $x_i \neq y_j$, we should let $L[i,j] = \max\{L[i,j-1], L[i-1,j]\}$. If we don't choose the maximum length, then we cannot meet the goal finding the "maximum" length. If $x_i \neq y_j$ & $L[i,j-1] = L[i-1,j]$, In order to find the longest minimum weight common subsequence, we need to take the weight of the result into consideration. In order to achieve the result with mimimum length, clealy we should choose $W[i,j] = \min\{W[i,j-1], W[i-1,j]\}$.

Space complexity:

- The table $W[i,j]$ has $m \times n$ entries, so the space complexity is $\mathcal{O}(mn)$
- The table $L[i,j]$ has $m \times n$ entries, so the space complexity is $\mathcal{O}(mn)$

Time complexity:

- The runtime of computing each entry of $W[i,j]$ and $L[i,j]$ is $\mathcal{O}(1)$
- The total runtime of this algorithm is $\mathcal{O}(mn)$

Problem 4

Solution:

Table:

- $Total[i]$ = the minimum total cost for traveling from post 1 to post i.
- $1 \leq i \leq n$

Output for the problem:

- $Total[n]$

Base Case:

- $i = 1$. Then $Total[i] = 0$
- $i = 2$ Then $Total[i] = fee[1,2]$

Recursion Case:

- $i < j - 1$
- $Total[i] = Total[k] + fee[k, i]$ minimized over $k = 2, \cdots, i - 1$

Algorithm:

Data: $1, n, fee[i, j]$
Result: The minimum total cost
if $n = 2$ **then**
  $return\ A[2] = fee[1, 2]$;
end
$A[1] = 0$;
$i = 2$;
while $i \leq n$ **do**
  while $j < i$ **do**
    $min = \min\{min, A[j] + fee[j, i]\}$;
  end
  $A[i] = min$;
  $i++$;
end
$return\ A[n]$;
Correctness:

The recurrence in this algorithm can be justified by considering an optimal sequence of renting at some post $n > 1$. This sequence must contain a last rental from some other post i to post n. The cost of this optimal sequence must be the cost of the optimal sequence ends at post i, plus the cost of renting from i to n. Since all cases for this last rental are considered, the recurrence correctly computes the costs of optimal rental sequence.

Space complexity:

- Because we have n entries, so the space complexity is $\mathcal{O}(n)$

Time complexity:

- The runtime of computing each entry is $\mathcal{O}(n)$
- The total runtime of this algorithm is $\mathcal{O}(n^2)$

Problem 5

Solution: