COMPSCI 130B Discussion

03/04/2021

Kha-Dinh (Jacob) Luong

Office hours: F 9-11 AM

Objectives

- HW3 Solutions

Q1

- Simple extension to the LCS problem on 2 strings.
- $OPT[i_1, \ldots, i_k]$: the length of the LCS of $s_1[0: i_1]$, ..., $s_k[0: i_k]$.
- Recurrence:

If $s_1[i_1] = s_2[i_2] = \cdots = s_k[i_k]$:

$$OPT[i_1, \ldots, i_k] = OPT[i_1 - 1, \ldots, i_k - 1] + 1$$

Else:

$$OPT[i_1, \ldots, i_k] = \max(OPT[i_1 - 1, i_2, \ldots, i_k], OPT[i_1, i_2 - 1, \ldots, i_k], \ldots, OPT[i_1, i_2, \ldots, i_k - 1])$$

Q1

- Dimension: $O(n^k)$
- Time complexity: $O(kn^k)$ because filling in each cell is $O(k)$

Q2

- $OPT[i_1, \ldots, i_k]$: the length of the longest common contiguous substring that ends at i_1 in s_1, i_2 in s_2, … , i_k in s_k.
- Recurrence:

If $s_1[i_1] = s_2[i_2] = \cdots = s_k[i_k]$ and $i_1, i_2, \ldots, i_k \neq 0$:

$$OPT[i_1, \ldots, i_k] = OPT[i_1 - 1, \ldots, i_k - 1] + 1$$

Else:

$$OPT[i_1, \ldots, i_k] = 0$$

Q2

- Dimension: $O(n^k)$
- Time complexity: $O(kn^k)$ because filling in each cell is $O(k)$

Q3

- Let the sorted list of symbols and their frequencies s and f, respectively.
- Let $TREE[i, j]$ stores the root of the optimal subtree of the subset of symbols $s[i, j]$.
- Average path length of $TREE[i, j]$:

$$\frac{\sum_{x=i}^{j} f_x l_x}{\sum_{x=i}^{j} f_x}$$

where l_x is the path length of symbol s_x in the tree.

Q3

- Combining 2 trees $TREE[i,k]$ and $TREE[k+1,j]$ using a function called $combineTree$:

$$\begin{tikzcd}
 & TREE[i,j] \arrow[dl, dash] \arrow[dr, dash] & \\
TREE[i,k] & & TREE[k+1,j]
\end{tikzcd}$$

- Calculating the cost of the combined tree:
    - Cost of $TREE[i,k] = \frac{\sum_{x=i}^{k} f_x l_x}{\sum_{x=i}^{k} f_x}$; Cost of $TREE[k+1,j] = \frac{\sum_{x=k+1}^{j} f_x l_x}{\sum_{x=k+1}^{j} f_x}$
    - Cost of $TREE[i,j] = \frac{\sum_{x=i}^{j} f_x l_x}{\sum_{x=i}^{j} f_x} + 1 = \frac{\sum_{x=i}^{k} f_x l_x + \sum_{x=k+1}^{j} f_x l_x}{\sum_{x=i}^{k} f_x + \sum_{x=k+1}^{j} f_x} + 1$

Q3

- Calculating the cost of the combined tree:
  - Cost of $TREE[i,k] = \frac{\sum_{x=i}^{k} f_x l_x}{\sum_{x=i}^{k} f_x}$; Cost of $TREE[k+1,j] = \frac{\sum_{x=k+1}^{j} f_x l_x}{\sum_{x=k+1}^{j} f_x}$
  - Cost of $TREE[i,j] = \frac{\sum_{x=i}^{j} f_x l_x}{\sum_{x=i}^{j} f_x} + 1 = \frac{\sum_{x=i}^{k} f_x l_x + \sum_{x=k+1}^{j} f_x l_x}{\sum_{x=i}^{k} f_x + \sum_{x=k+1}^{j} f_x} + 1$
- To make this update $O(1)$, keep 2 tables $NUMER[i,j]$ and $DENOM[i,j]$:
  - $NUMER[i,j] = \sum_{x=i}^{j} f_x l_x$
  - $DENOM[i,j] = \sum_{x=i}^{j} f_x$
- Updating the cost for $TREE[i,j]$ from $TREE[i,k]$ and $TREE[i+1,j]$:
  - $DENOM[i,j] = DENOM[i,k] + DENOM[k+1,j]$
  - $NUMER[i,j] = NUMER[i,k] + NUMER[k+1,j] + DENOM[i,j]$
  - Cost of $TREE[i,j] = \frac{NUMER[i,j]}{DENOM[i,j]}$

Q3

- Base case:
  - $TREE[i,i] = TreeNode(s[i])$
  - $NUMER[i,i] = 0$ (path length = 0)
  - $DENOM[i,i] = f[i]$
- Recurrence:
  - $k_{min} = argmin_{i \leq k \leq j} \frac{NUMER[i,k]+NUMER[k+1,j]}{DENOM[i,k]+DENOM[k+1,j]} + 1$
  - $TREE[i,j] = combineTree(TREE[i,k_{min}], TREE[k_{min}+1,j])$
  - $DENOM[i,j] = DENOM[i,k] + DENOM[k+1,j]$
  - $NUMER[i,j] = NUMER[i,k] + NUMER[k+1,j] + DENOM[i,j]$

Q3

- Assuming both subtrees are valid prefix encoding, the algorithm adds a different prefix to each tree, so the two trees should not share any prefixes between each other.
- In the base case it is valid prefix encoding because there is only 1 character.
- As a result, the prefix coding produced by DP is valid.

Q3

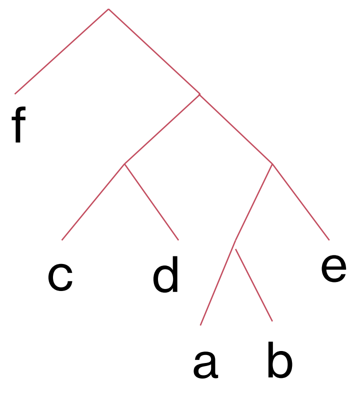
- We will prove that DP is optimal.
- Notice some properties about the class of trees found by Huffman and the class of trees found by DP:
    - Trees found by Huffman: The total frequency of the nodes on the left subtree is greater than the total frequency of the nodes on the right subtree.
    - Trees found by DP: The frequency of any node in the left subtree is greater than the frequency of any node in the right subtree.
- It is easy to see that the set of trees found by DP is a subset of the set of trees found by Huffman so there is a reason to believe that DP is not as optimal as Huffman. However, if we can show that there is a way to transform any tree found by Huffman into a tree that DP can find without increasing the average path length then DP is also optimal.

Q3

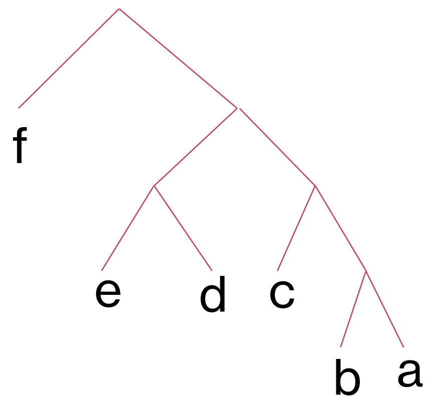
- Procedure:
    - Consider an arbitrary Huffman encoding tree with each node (both leaf nodes and inner nodes) keeping track of the maximum frequency of any node in its subtree.
    - For each layer, rearrange the node in the order of decreasing maximum frequency.
    - This procedure does not change the average path length of the tree because each node stays on the same level before and after the procedure. In addition, each node on the left subtree now has a frequency no less than that of any node on the right subtree.

Q3

- Example: a:5, b:9, c:12, d:13, e:16, f:45.



Tree produced by Huffman



Tree transformed by the procedure

Q3

- Proof by contradiction
- Assume that there exists a node l in some left subtree at level i, and node r in the right subtree at level j such that freq(r) > freq (l). Moreover, we assume that i is the smallest such layer.
- We consider the following three cases:
- **j > i** : From the properties of the huffman tree, we know that frequency of any node closer to the root is greater than the node farther from the root; otherwise, we can swap the two nodes to obtain a better huffman encoding. Hence, this case is not possible.
- **j = i** : At the end of our procedure, at each layer of the tree the frequencies will decrease from left to right (since our procedure ensures that). Hence, this case is not possible.
- **j < i**: Let's analyse subtree rooted at node l and node r. Since, the frequency stored at r is greater than the frequency stored at l, there must be a leaf node in the tree rooted at r which has this frequency, let's call that leaf node t (let the level of t be k). Note that k must be strictly less than i; otherwise, at the i-th layer containing l, our algorithm would have swapped the left and right trees since the right tree node stores a higher frequency than freq(l).

Q3

- For any optimal Huffman tree, there exists a tree with the same average path length that belongs to the class of trees that DP considers. Because DP finds the most optimal tree among all the trees in the class, it will find an optimal tree.
- Dimension: $O(n^2)$
- Time complexity: $O(n^3)$

Q4

- Let $OPT[i]$ be the minimum sum of any decreasing subsequence of the string $s[0{:}\,i]$.
- Base case: $OPT[0] = s[0]$
- Recurrence:

$$OPT[i] = \min(OPT[i-1], min_{0\leq j<i}(s[i] + OPT[j]\ such\ that\ s[j] > s[i]))$$

Q4

- Dimension: $O(n)$
- Time complexity: $O(n^2)$ because filling in each cell is $O(n)$

Q5

- Let (x_s, y_s) be the starting position. Let $OPT[i, j]$ be the minimum cost to go from the start position to (i, j).
- Base case: $OPT[x_s, y_s] = 0$
- Recurrence:

$$OPT[i, j] = \min(OPT[i + x_{di}, j + y_{di}] + cost([i + x_{di}, j + y_{di}], [i, j]))$$

such that $0 \leq i + x_{di}, j + y_{di} < n$ and $(x_{di}, y_{di}) \in \{(-1, -1), (0, -1), (1, -1)\}$

Q5

- Dimension: $O(n^2)$
- Time complexity: $O(n^2)$ because filling each cell takes constant time.

Q6

- Let $OPT[i, j]$ be the maximum success probability when we consider the first i stages and budget j.
- Base cases:
  - $OPT[0, j] = 1$
  - $OPT[i, 0] = 0$
- Recurrence:

$$OPT[i, j] = max_{0 \le j' \le j}(OPT[i-1, j-j'] \times (1 - p_i^{\left\lfloor \frac{j\prime}{c_i} \right\rfloor}))$$

Q6

- Dimension: $O(nB)$
- Time complexity: $O(nBB)$ since we spend $O(B)$ time per entry.