CS 5800 Assignment 2 Solutions

Course Staff

July 26, 2017

1 Median of two sorted arrays

We solve this problem through a divide-and-conquer algorithm in which, at each iteration, we discard approximately half of the entries in each array. We are ultimately left with either two or four median candidates, of which we select the lower-middle value, according to the definition given in the problem statement.

```
1: procedure MedianTwoArrays(A, B, n)
2:     return MedianHelper(A, 1, n, B, 1, n)

3: procedure MedianHelper(A, s_A, t_A, B, s_B, t_B)
4:     ▷ invariant: t_A − s_A = t_B − s_B
5:     if s_A = t_A then
6:         return min{A[s_A], B[s_B]}
7:     else if s_A + 1 = t_A then
8:         T ← Sorted({A[s_A], A[t_A], B[s_B], B[t_B]})
9:         return T[2]
10:    else
11:        p ← ⌊(s_A + t_A)/2⌋
12:        q ← ⌊(s_B + t_B)/2⌋
13:        k ← p − s_A
14:        if A[p] < B[q] then
15:            ▷ discard first half of A, second half of B
16:            return MedianHelper(A, p, t_A, B, s_B, t_B−k)
17:        else
18:            ▷ discard second half of A, first half of B
19:            return MedianHelper(A, s_A, t_A−k, B, q, t_B)
```

Note that, as suggested by the invariant of our helper function, we must take care to discard an equal number of elements from each array at every recursive step.

The recurrence for the number of comparisons performed is (roughly) $T(n) = T(n/2) + 1$, which gives a complexity of $O(\log_2 n)$.

2 LCS example

We compute the matrix for sequences $a = 4, 2, 6, 9, 0$ and $b = 2, 5, 9, 0, 4$.

$$L = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 2 & 2 & 2 \\ 0 & 1 & 1 & 2 & 3 & 3 \end{pmatrix}$$

We see that each element is at least as great as the elements above it and to the left. When the sequence items corresponding to a matrix element have equal values, we may be able to use that equality as part of a better subsolution.

3 Number-choosing game

The key to this problem is its symmetry. Final scores are zero-sum, and each player has the same objective. We call a final score, which is the sum of a player's numbers minus the sum of their opponent's, a *relative score*.

Let $S(i, j)$ where $1 \leq i \leq j \leq n$ denote the best relative score a player can guarantee by having first pick at the subsequence $a_i, ..., a_j$. There are $n(n+1)/2$ subproblems, and we can write the recurrence as:

$$S(i, i) = a_i$$
$$S(i, j) = \max\{a_i - S(i{+}1, j),\ a_j - S(i, j{-}1)\}$$

That is, if a player chooses one end of the sequence, her score will be that element minus her opponent's best possible score for the remaining elements.

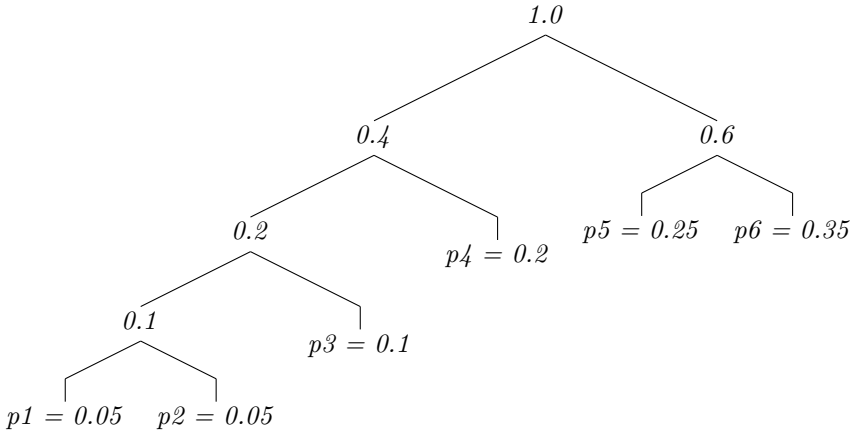
Our final answer will be $S(1, n)$.

We can calculate these subsolutions in a double loop over the indexes. Since each subsolution can be calculated in constant time, the total complexity is $O(n^2)$.

4 Huffman codes

4.a

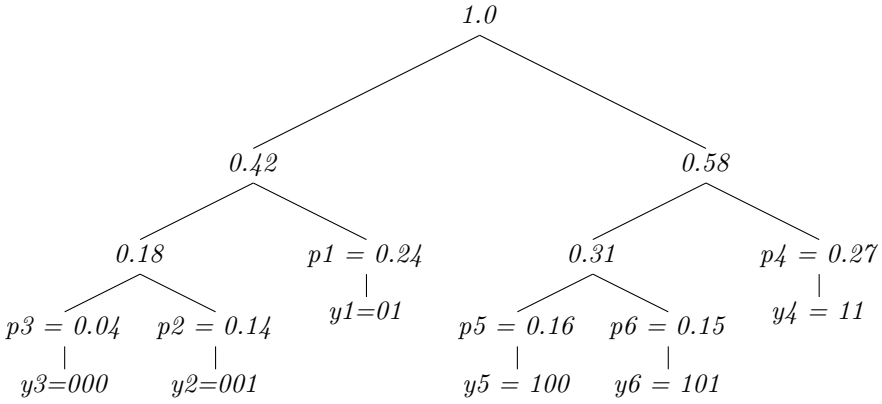
The tree we get after performing Huffman's algorithm is:



Assigning codes based on this tree, we get the Prefix Code (0000, 0001, 001, 01, 10, 11)

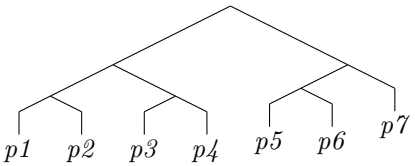
4.b

p = (0.24, 0.14, 0.04, 0.27, 0.16, 0.15) is one probability vector that satisfies given conditions. For the greaph below, a right branch denotes a '1' and a left branch denotes a '0'.



4.c

Consider this 7-leafed tree:



For this tree, $l_1 = l_2 = l_3 = l_4 = l_5 = l_6 = 3, l_7 = 2$.
Let $p = (p_1, p_2, p_3, p_4, p_5, p_6, p_7)$ be the probability vector. We know that,

$$\sum_{i=1}^{7} p_i = 1 \implies \sum_{i=1}^{6} p_i = 1 - p_7$$

In every probability vector, there will be at least one character that has the highest frequency (unless all characters have the same frequency). We can assume that the frequencies $p_1, p_2, ...p_7$ are sorted so that $p_1 \leq p_2 \leq p_3 ... \leq p_7$ since given a probability vector we can label the probabilities as we wish without changing the problem.

In the given tree to be efficient (closer to optimal) the character with highest frequency (p_7) should be the one that is assigned the code with 2 bits, as shown in diagram.

$$f(p) = \sum_{i=1}^{7} p_i l_i = \sum_{i=1}^{6} p_i l_i + p_7 \times l_7 = 3 \times \sum_{i=1}^{6} p_i + p_7 \times 2 = 3 \times (1 - p_7) + 2p_7 = 3 - 3p_7 + 2p_7 = 3 - p_7$$

$f(p) = 3 - p_7$, where p_7 is the largest probability in the probability vector p.

The function $3 - p_7$ is a strictly decreasing function (because the derivative is -1), so for larger values of p_7 the value of the function would be smaller. To get maxima of the function, we will take the smallest possible value for p_7 which is $\frac{1}{7}$.

$$\therefore f(p) \leq 3 - \frac{1}{7} = \frac{20}{7} = \frac{40}{14} < \frac{41}{14}$$

$$\implies f(p) < \frac{41}{14}$$

However, the tree we have used for our analysis may not be optimal. But we know for a fact that the optimal tree must be atleast as efficient as the given tree, which means

$$f_{opt}(p) \leq f(p) < \frac{41}{14} \implies f_{opt}(p) < \frac{41}{14}$$

Q.E.D.

5 Kruskal's algorithm example

Edges in sorted order - ab, bc , eb , ec , cd , da , ae

E'	Connected Components	Edges	Remark
	a,b,c,d,e		
ab	ab,c,d,e	ab = 1	
ab,bc	abc,d,e	bc = 2	
ab,bc,eb	abce,d	eb = 3	
ab,bc,eb	abce,d	ec = 4	In the same component
ab,bc,eb,cd	abced	cd = 5	Finished

6 Unique MST (updated solution)

6.a

Consider the trees $T_1 = (V, E_1)$ and $T_2 = (V, E_2)$, and let $e_1 = (u, v)$ be an edge such that $e_1 \in E_1 - E_2$. If we remove the edge e_1 from T_1, we get a forest with 2 connected components (since we proved that in any forest G we have $|E| = |V| - c(G)$). One of the connected components contains u and the other contains v. Let T_u and T_v be the corresponding trees. Now, since T_2 is also a spanning tree, it follows that there is a path that connects u and v in T_2.

Assume that:

$$u \to v_1 \to v_2 \to \cdots \to v_n = v$$

is the path from u to v in T_2. Since $u \in T_u$ and $v \in T_v$, there must be an edge $e2 = (v_i, v_{i+1})$ such that $v_i \in T_u$ and $v_{i+1} \in T_v$. Clearly, the edge e_2 connects the 2 connected components that corresponds to T_u and T_v and therefore $(T_1 - \{e_1\}) \cup \{e_2\}$ is a spanning tree.

On the other hand, removing e_2 from T_2 would create 2 (possibly different) connected components T_{v_i} and $T_{v_{i+1}}$. Adding the edge e_1 would connect these 2 connected components since we will have a path:

$$v_{i+1} \to v_{i+2} \to \cdots \to v \to u \to v_1 \to \cdots \to v_i$$

that connects v_{i+1} and v_i (and can replace e_2 in any path that contains e_2). It follows that $(T2 - \{e_2\}) \cup \{e_1\}$ is a spanning tree.

6.b

Let us assume there are two different MST T_1 and T_2. Let $e_1, e_2, \ldots, e_m$ be the edges of T_1 and let $e'_1, e'_2, \ldots, e'_m$ be the edges of T_2. Assume the edges are ordered by their weights so that we have:

$$w(e_1) < w(e_2) < \cdots < w(e_m)$$

and

$$w(e'_1) < w(e'_2) < \cdots < w(e'_m).$$

Let i be the maximal index such that $w(e_i) \neq w(e'_i)$. Assume without loss of generality that $w(e_i) > w(e'_i)$. By part (a) we can find an edge $e \in T_2$ such that $e \notin T_1$ and $(T_1 - \{e_i\}) \cup \{e\}$ is a spanning tree. Since for indices greater then i we have equality in the weights of the edges and hence the edges are the same, the edge e must be one of the edges $e_1, e_2, \ldots, e_{i-1}$. It follows that $w(e) \leq w(e'_i) < w(e_i)$ and hence $(T_1 - \{e_i\}) \cup \{e\}$ is a spanning tree, whose weight is smaller than the weight of T_1. Since T_1 is a MST, this is a contradiction.