CSE 101: Homework #5

Due on May 16, 2024 at 23:59pm

Professor Jones

Ray Tsai, Kevin Yu

Problem 1

Consider the following divide and conquer algorithm that claims to find an MST when the input is a complete graph G with positive edge weights:

Algorithm Description: Given an undirected complete graph $G = (V, E)$ with positive edge weights where $V = [v_1, \ldots, v_n]$,

- If $n = 1$ then return the empty set of edges.

- Otherwise, split the set of vertices into two sets: $V' = [v_1, \ldots, v_{\lfloor n/2 \rfloor}]$ and $V'' = [v_{\lfloor n/2 \rfloor} + 1, \ldots, v_n]$.

- Create two new graphs $G' = (V', E')$ and $G'' = (V'', E'')$ where $E' \subseteq E$ is the set of edges with both endpoints in V' and $E'' \subseteq E$ is the set of edges with both endpoints in V''.

- Recursively run the algorithm on G' and G'' to get T' and T'', respectively. Find the lightest edge that connects a vertex in T' to a vertex in T'' and call that edge e.

- Return $T' \cup T'' \cup \{e\}$.

Disprove the correctness of this algorithm by giving a counterexample.

Proof. Consider $G = C_4$, where the edge $\{v_3, v_4\}$ has weight 2 and the remaining edges each has weight 1. The algorithm recurses on subgraph G'' with vertex set $V'' = [v_3, v_4]$, so the resulting spanning tree T contains the edge $\{v_3, v_4\}$. Since the resulting spanning tree has 3 edges with an edges of weight 2, the total cost of T is 4. But then $\{\{2, v_i\} : i \neq 2\} \subset E$ spans G with a total weight of 3, as it only uses edges of weight 1. $\qquad\square$

Problem 2

You are given an increasing sequence of integers: $(A[1], A[2], \ldots, A[n])$. Design an algorithm that determines (returns TRUE or FALSE) if there exists an index i such that $A[i] = i$.

Your algorithm should run in $O(\log n)$ time.

Problem 3

You are given a list of n ordered pairs $[(x_1, f_1), \ldots, (x_n, f_n)]$. This list describes a list of length $\sum f_i$ that contains f_1 copies of the value x_1, f_2 copies of the value x_2 and so on.

You wish to find the median value of this list in expected runtime of $O(n)$. (You can assume that $\sum f_i$ is odd.)

Proof. We give a description of the algorithm:

Let $\ell([(x_1, f_1), \ldots, (x_u, f_u)])$ denote the length of the list described by $[(x_1, f_1), \ldots, (x_u, f_u)]$, namely $\sum_{i=1}^{u} f_i$.

We first define $Selection(L = [(x_1, f_1), \ldots, (x_m, f_m)], k)$, which takes in a list L of ordered pairs and an integer k, and outputs the kth smallest number in the list described in L:

If $|L| = 1$, return x_1. Otherwise, pick x_v randomly from L. Split L into L_l, $[(x_v, f_v)]$, and L_r, where L_l contains all the ordered pairs with x_i less than x_v and L_r contains the ordered pairs with x_i greater than x_v. If $k \leq \ell(L_l)$, return $Selection(L_l, k)$. Else, if $k \leq \ell(L_l) + f_v$, return x_v. Otherwise, return $Selection(L_r, k - \ell(L_l) - f_v)$.

Now for finding the median value of the list described in L, we simply run $Selection(L, \lceil \frac{n}{2} \rceil)$.

We now show that the expected runtime for $Selection$ is $O(n)$.

Since we select the pivot x_v uniformly at random, the input list L will be split into a list L_l of length $v-1$ and a list L_r of length $n-v$. Hence, when we recurse on L_l, L_r, it will take time proportional to $max(v-1, n-v)$. Note that if $\frac{n}{4} \leq v - 1 \leq \frac{3}{4}n$, then $max(v-1, n-v) \leq \frac{3}{4}n$. Otherwise, $\frac{3}{4}n \leq max(v-1, n-v) < n$. Let $ET(n)$ denote the expected runtime for $Selection$ on a list of length n. It now follows that

$$ET(n) \leq \frac{1}{2} ET\left(\frac{3}{4}n\right) + \frac{1}{2} ET(n) + cn,$$

where the cn term derived from the splitting process of L. But then

$$ET(n) \leq ET\left(\frac{3}{4}n\right) + cn,$$

and thus

$$ET(n) \in O(n).$$

by the Master Theorem. $\square$

Problem 4

(a) Let $T(n)$ be the runtime of a divide and conquer algorithm on an input of size n. The algorithm has 6 recursive calls each of size $n/4$ and the non-recursive part takes $O(n^{1.5})$ time. Use the Master theorem to find the best Big-Oh runtime.

Proof. We first note that
$$T(n) = 6T(n/4) + cn^{1.5}.$$

By the Master Theorem,
$$T(n) \in O(n^{1.5}),$$

as $6 < 4^{1.5} = 8$. □

(b) Let $R(n)$ be the runtime of a divide and conquer algorithm on an input of size n. The algorithm has 1 recursive call of size $n/2$ and the non-recursive part takes $O(\log n)$ time. Find the best Big-Oh runtime.

Proof. We first note that
$$R(n) = R(n/2) + c \log n.$$

Consider the levels of recurrence of this algorithm. Since the algorithm has 1 recursive call of size $n/2$, there are $\log n$ levels of recurrence, with 1 recursive call per level. It now follows that

$$\begin{aligned}
R(n) &= R(n/2) + c \log n \\
&= \left(R(n/4) + c \log \frac{n}{2} \right) + c \log n \\
&= c \sum_{k=0}^{\log n} \log \frac{n}{2^k} \\
&= c \sum_{k=0}^{\log n} (\log n - k) \\
&= c \log^2 n - c \sum_{k=0}^{\log n} k \\
&= c \log^2 n - \frac{c(\log n + 1) \log n}{2} \\
&\in O\left(\log^2 n \right).
\end{aligned}$$

□

(c) Let $S(n)$ be the runtime of a divide and conquer algorithm on an input of size n. The algorithm has 2 recursive calls each of size $2n/3$ and the non-recursive part takes $O(n)$ time. Find the best Big-Oh runtime.

Proof. We first note that
$$S(n) = 2T(2n/3) + cn.$$

By the Master Theorem,

$$S(n) \in O(n^{\log_{3/2} 2}) = O(n^{\frac{\log 2}{\log 3 - \log 2}}) \approx O(n^{1.71}),$$

as $2 > 3/2$. □