CS 577: Introduction to Algorithms Fall 2019

Homework 1

Instructor: Dieter van Melkebeek TA: Andrew Morgan

This homework covers graph primitives and the divide and conquer paradigm. In order to get feedback on problem 3, you need to submit it via Canvas by **11:59pm on 9/17**. Please refer to the homework guidelines on Canvas for detailed instructions.

Warm-up problems

1. In a tree T, the distance between two vertices is the number of edges on the unique simple path between them. The *depth* of a nonempty rooted tree T is the largest distance between any vertex and the root, and the *diameter* of T is the largest distance between any pair of vertices.

   The algorithm below purports to compute the diameter of a given nonempty rooted tree.

   Argue that FindDiameterAndDepth and FindDiameter

   (a) correctly implement their specifications and

   (b) run in linear time.

Algorithm 1

Input: T, where T is a nonempty rooted tree
Output: the diameter of T

```
procedure FindDiameter(T)
    (D, d) ← FindDiameterAndDepth(T)
    return D
```

Input: T, where T is a nonempty rooted tree
Output: (D, d) where D is the diameter of T and d is the depth of T

```
procedure FindDiameterAndDepth(T)
    if T's root has no children then
        return (0, 0)
    else
        T_1, ..., T_k ← the subtrees of T rooted at the children of the root of T
        for i = 1 to k do
            (D_i, d_i) ← FindDiameterAndDepth(T_i)
        d ← 1+ largest d_i
        if k = 1 then d' ← 0 else d' ← 1+ second largest d_i including repetitions
        D ← max(D_1, ..., D_k, d + d')
        return (D, d)
```

2. Consider the problem of powering an integer:

   Input: (a, b) with $a, b \in \mathbb{Z}$ and $b \geq 1$

   Output: a^b, which we define as $a^b \doteq \underbrace{a \cdot a \cdot \cdots \cdot a}_{b \text{ times}}$, where "$\cdot$" denotes multiplication.

   Design an algorithm for this problem that uses at most $O(\log b)$ multiplications. Hint: First consider the case where b is a power of 2.

Feedback problem

3. You are given a complete binary tree T with $n = 2^d$ leaves, where each leaf contains an integer value. Reading the leaf values from left to right yields a sequence of integers. The question is how small we can make the number of inversions in that sequence by applying any number of operations of the following type: Select an internal vertex and swap the two child subtrees.

   For example, if the sequence of leaf values is $(4, 2, 1, 3)$, then a swap at the root followed by a swap at the right child of the root turns the sequence into $(1, 3, 2, 4)$, which has only one inversion. See Figure 1. It is impossible to do better, so the answer for this particular example is 1.

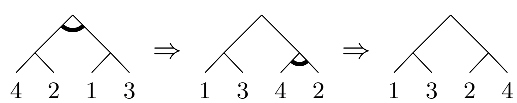

Figure 1: Example for Problem 3

   Design an $O(n \log n)$ algorithm for this problem.

Additional problem

4. Consider the following computational problem:

   Input: Array $A[1, \ldots, n]$ of positive integers.

   Output: Array $C[1, \ldots, n]$ where $C[i]$ is the number of $j \in \{1, \ldots, i-1\}$ with $A[j] \geq A[i]$.

   For example, if $A = [8, 12, 10, 9, 10, 12, 7]$ then $C = [0, 0, 1, 2, 2, 1, 6]$.

   Design an $O(n \log n)$ algorithm for this problem.

Challenge problem

5. Show that every comparison-based algorithm for problem 3 requires $\Omega(n \log n)$ comparisons.

Programming problem

6. SPOJ problem Insertion Sort (problem code CODESPTB).