FirstName LastName
StudentID, acct@ucsc.edu

CMPS 102 — Spring 2020 – Homework 1

Updated Ver. 3(04-05)
Four problems, 25 points, due Friday April 10 (11:59 PM) on Gradescope.

Before you begin the assignment, please read the following carefully.

- Read the *Homework Guidelines*.
- Every part of each question begins on a new page. Do not change this.
- This does not mean that you should write a full page for every question. Your answers should be short and precise. Lengthy and wordy answers will lose points.
- Do not change the format of this document. Simply type your answers as directed.
- You are **not** allowed to work in teams.

I have read and agree to the collaboration policy. – FirstName LastName, email@ucsc.edu
Collaborators:

---

1. (Total: 8 pts) A grad student comes up with the following algorithm to sort an array $A[1..n]$ that works by first sorting the first 2/3rds of the array, then sorting the last 2/3rds of the (resulting) array, and finally sorting the first 2/3rds of the new array.

```
1:  function G-SORT(A, n)                          ▷ takes as input an array of n numbers, A[1..n]
2:      G-sort-recurse(A, 1, n)
3:  end function
4:  function G-SORT-RECURSE(A, ℓ, u)
5:      if u − ℓ ≤ 0 then
6:          return                                  ▷ 1 or fewer elements already sorted
7:      else if u − ℓ = 1 then                      ▷ 2 elements
8:          if A[u] < A[ℓ] then                     ▷ swap values
9:              temp ← A[u]
10:             A[u] ← A[ℓ]
11:             A[ℓ] ← temp
12:         end if
13:     else                                        ▷ 3 or more elements
14:         size ← u − ℓ + 1
15:         twothirds ← ⌈(2 ∗ size)/3⌉
16:         G-sort-recurse(A, ℓ, ℓ + twothirds − 1)
17:         G-sort-recurse(A, u − twothirds + 1, u)
18:         G-sort-recurse(A, ℓ, ℓ + twothirds − 1)
19:     end if
20: end function
```

a (4 pts). First, prove that the algorithm correctly sorts the numbers in the array (in increasing order). After showing that it correctly sorts 1 and 2 element intervals, you may make the (incorrect) assumption that the number of elements being passed to *G-sort-recurse* is always a multiple of 3 to simplify the notation (and drop the floors/ceilings).

Solution.

b (1 pts). Next, Derive a recurrence for the algorithm's running time (or number of comparisons made).

Solution.

c (3 pts). Finally, obtain a good asymptotic upper bound (big-O) for your recurrence.

Solution.

"I have read and agree to the collaboration policy." – FirstName LastName, email@ucsc.edu
Collaborators:

---

2. (Total: 8 pts) Induction Proof correctness.

   Recall that a full binary tree contains (A) just a single leaf node, or (B) is an internal node (the root) connected to two disjoint subtrees, which are themselves full binary trees.

   First consider the following claim and proof. First, think about if the theorem is true or not, and if the proof is correct or not (do not include these preliminary thoughts in your answer).

   Claim 1. *In any full binary tree, the number of leaf nodes is one greater than the number of internal nodes.*

   Proof. (??) By induction on number of internal nodes.

   For all $n \geq 0$, let IH(n) be the statement: "all full binary trees having exactly n internal nodes have $n + 1$ leaf nodes."

   Base Case: Show IH(0). Every full binary tree with zero internal nodes is formed by case (A) of the definition, and thus consists of just a single leaf node. Therefore, every full binary tree with 0 internal nodes has exactly 1 leaf node, and IH(0) is true.

   Inductive Step: Assume $k > 0$ and IH(k) holds to show that IH$(k + 1)$ also holds.

   Consider an arbitrary full binary tree T with k internal nodes. By the inductive hypothesis T has $k + 1$ external nodes. Create a $k + 1$ internal node tree T' by removing a bottom leaf node in T and replacing it with an internal node connected to two children that are leaves. T' has one more internal node than T, and $2 - 1 = 1$ more external node than T. Therefore T' has $k + 1$ internal nodes and $(k + 1) + 1 = k + 2$ external nodes, proving $P(k + 1)$. □

   Now consider the following claim.

   Claim 2. *For all n, all full binary trees with n internal nodes have height $n - 1$.*

   We have the following "proof" of the claim.

   Proof. ?? By induction on n. For each $n \geq 1$, let IH(n) be the statement "all full binary trees with n internal nodes have height $n - 1$".

   Base Case: Show IH(0). Every full binary tree with zero internal nodes is formed by case (A) of the definition, and thus consists of just a single leaf node. Therefore, every full binary tree with 0 internal nodes has only one leaf (which is also the root). Thus the longest root-to-leaf path has length 0, and IH(0) is true.

   Inductive Step: Let $n \geq 1$ and show that IH(n) implies IH$(n + 1)$. Let T be an arbitrary full binary tree with n internal nodes. Create the binary tree T' having $n+1$ internal nodes by removing a bottom leaf node in T and replacing it with an internal node connected to two children that are leaves. The longest root-to-leaf path in T' is thus one greater than the longest root-to-leaf path in T, so the height of T' is the height of T plus 1. Furthermore, by the inductive hypothesis IH(n), the height of T is $n - 1$. Therefore T' has $n + 1$ internal nodes and height $n - 1 + 1 = n$, showing IH$(n + 1)$. □

a (1 pt). Give a counter-example showing that claim 2 is false. (Recall that the height of a binary tree is the length of the longest root-to-leaf path).

Solution.

b (3 pts). Identify and clearly describe the flaw in the proof of claim 2.

Solution.

c (4 pts). Now go back to the proof of the first claim.
Does it have a flaw or hidden assumption? (1 pt)
Either clearly describe the flaw/assumption, or argue that the proof is correct (3 pts).

Solution.

"I have read and agree to the collaboration policy." – FirstName LastName, email@ucsc.edu
Collaborators:

---

3. (Total: 6 pts) Asymptotic notation: Exercise 5 of Chaper 2: Prove or disprove 3 asymptotic implications: If $f(n)$ is in $O(g(n))$ then is it always true that:
a (2pts). $\log_2(f(n))$ is in $O((\log_2(g(n))$.

   Solution.

b (2 pts). $2^{f(n)}$ is in $O(2^{g(n)})$.

Solution.

c (2 pts). $f(n)^2$ is in $O(g(n)^2)$.

Solution.

"I have read and agree to the collaboration policy." – FirstName LastName, email@ucsc.edu
Collaborators:

---

4. (3 pts) Least counterexample: Give a proof by contradiction using the *least counterexample* method that:

$$\sum_{i=0}^{n} i = \frac{n(n+1)}{2} \qquad \text{for all } n \geq 0.$$

Solution.

Recommended exercises (not to be turned in)

1. The solved exercises in Chapters 1, 2, and 5 (Divide and Conquer).
2. Exercises 1 and 2 in Chapter 1 of the text.
3. Exercises 3 and 4 in Chapter 2 of the text.