CS 170 Algorithms Fall 2014 David Wagner HW 1

Due Sept. 5, 6:00pm

Instructions. This homework is due Friday, September 5, at 6:00pm electronically. It must be submitted electronically via Pandagrader (not in person, in the drop box, by email, or any other method). We recommend you submit your homework by 5:00pm, as the Pandagrader website sometimes becomes overloaded and slow in the final hour before the deadline.

You are welcome to form small groups (up to four people) to work through the homework, but you **must** write up all your solutions strictly by yourself, and you must acknowledge any ideas you got from others (including from books, papers, web pages, etc.) Please read the collaboration policy on the course web page.

On the first page of your submission, put your name, your student ID number, your class account userid, the homework number (HW1), and your study partners for this homework or "none" if you had no partners. Note that you will need a class account, and you will need to log into it and complete the registration process.

Each problem should begin on a new page. The pages of your homework submissions must be in order (all pages of problem 1 in order followed by all pages of problem 2 in order, etc...). See the web site for detailed instructions on how to prepare your PDF file and upload to Pandagrader. Make sure to select pages for each question after uploading in Pandagrader. If you fail to complete the process of page selection, your PDF may be marked as late and therefore not accepted. You risk receiving no credit for any homework that does not adhere to these guidelines.

No late homeworks will be accepted. No exceptions. Please don't ask for extensions. We don't mean to be harsh, but we prefer to make model solutions available shortly after the due date, which makes it impossible to accept late homeworks.

1. (30 pts.) Getting started

The goal of this question is to ensure that you have read the course policies.

(a) Please read the course policies on the web page, especially the course policies on collaboration. If you have any questions, ask on Piazza about it. Once you have done this, please write "I understand the course policies." to get credit for this problem.

(b) You've been working with a homework partner, and together you've managed to solve both problems on the homework. It's been a true joint effort—neither of you could have solved either problem on your own. Your homework partner suggests that you write up the solution for Problem 1, and he'll write up the solution for Problem 2, then you can swap write-ups to see how the other wrote up the solution, as a way to learn from each other. Is this allowed?

2. (35 pts.) Proof of correctness

Define $P(x) = a_n x^n + \cdots + a_2 x^2 + a_1 x + a_0$. Prove the correctness of the following algorithm for evaluating the polynomial P at the value $x = c$, i.e., for computing $P(c)$:

Algorithm HORNER$((a_0, a_1, \ldots, a_n), c)$:
1. Set $y := a_n$.
2. For $i := 0, 1, 2, \ldots, n-1$:
3. Set $y := c \times y + a_{n-i-1}$.
4. Return y.

Hint: Identify an invariant that holds at the start of each iteration of the loop, use proof by induction to show that it is a valid invariant, and then finally show why the invariant guarantees this algorithm will produce the correct output.

3. (35 pts.) Prove this algorithm correct

We say that v is a *majority* value for the list L if $> 50\%$ of the elements of L have the value v. Let's prove that, if L has a majority value, then the following algorithm will output it. (If L does not have a majority value, then the algorithm is allowed to output anything it wants, without restriction.)

Algorithm BM$(A[0..n-1])$:
1. Set $c := 0$. Set $v :=$ null.
2. For $i := 0, 1, 2, \ldots, n-1$:
3. If $c = 0$, set $v := A[i]$.
4. If $v = A[i]$, set $c := c + 1$, else set $c := c - 1$.
5. Output v.

We'll guide you through a proof that this algorithm is correct, with the following steps:

(a) Prove that c never goes negative.

(b) The following is an invariant of the algorithm:

> At the start of any iteration of the loop, the elements of $A[0..i-1]$ can be partitioned into two groups: a group U_i of at least c instances of the value v, and a group P_i of elements that can be paired off so that the two elements in each pair differ.

For example, consider the input $[2,3,3]$. After one iteration, $[2]$ can be partitioned into the group $U_1 = [2]$ and $P_1 = []$. After two iterations, $[2,3]$ can be partitioned into the group $U_2 = []$ and $P_2 = [2,3]$. After two iterations, $[2,3,3]$ can be partitioned into the group $U_3 = [3]$ and $P_3 = [2,3]$.

Prove by induction that this invariant holds at the start of each iteration of the loop.

Hint: Show the base case, then the inductive step. Break the inductive step down into three cases: (i) $c = 0$, (ii) $c > 0$ and $v = A[i]$, (iii) $c > 0$ and $v \neq A[i]$. For each case, use the inductive hypothesis to explain how U_{i+1} and P_{i+1} can be formed.

(c) Prove why the invariant from part (b) implies that the algorithm is correct.

Hint: Consider what happens if the array has a majority element.

4. (3 pts.) Optional bonus problem: Check my proof

(This is an *optional* bonus challenge problem. Only solve it if you can't get enough of the algorithms goodness. We reserve the right not to grade bonus problems if necessary.)

If $G = (V, E)$ is an undirected graph and $S \subseteq V$ is a set of vertices, call the edge (u, v) *good* if one of u, v is in S and the other is not in S; and call the set S *nice* if it makes at least half of the edges of the graph good. We

want an algorithm that, given an undirected graph $G = (V, E)$, outputs a nice set S. Consider the following algorithm and proof of correctness:

Algorithm MAKENICE(G):
1. Set $S := \emptyset$.
2. Loop until S is nice:
3. Let w be a vertex whose degree is maximal (resolve ties arbitrarily).
4. Set $S := S \cup \{w\}$.
5. Delete all edges that have w as an endpoint.
6. If S is nice (for the original graph), return S.

Proof: Imagine keeping track of the number of good edges (from the original graph), as the algorithm executes. We will show that the number of good edges increases in each iteration of the loop.

In particular, when we add w to S in step 4, every edge currently in the graph that has w as an endpoint will shift from non-good to good.

(Why? Well, the algorithm immediately deletes all edges whenever one of their endpoints enters S, so whenever execution reaches step 3, all remaining edges connect two vertices in $V \setminus S$. In particular, all edges that haven't been deleted yet are non-good. But the ones that have w as an endpoint will immediately become good when w is added to S.)

Since we strictly increase the number of good edges each time we execute an iteration of the loop, eventually the number of good edges must exceed $|E|/2$. □

Is this proof of correctness valid? Write "Valid" or "Invalid." If you write "Valid", analyze the running time of the algorithm. If you write "Invalid", explain what is wrong (list the first step in the reasoning that doesn't follow).