University of Waterloo
CS240R Spring 2017
Assignment 5 Post mortem

Written Monday, July 31st

Problem 1

Part a) was done well, though some students did not keep the order of their internal nodes' children consistent. We recommend the order NW, NE, SE, SW.

There were two common mistakes in part b). The first one was students who did not use intersection, but rather assigned nodes values based on equality, so 0 and 0 make 1; 1 and 1 make 1. This is not intersection. Intersection is an "and" operator, so only 1 and 1 make 1; 0 and 0 make 0; 1 and 0 make 0. The other mistake was not collapsing all children into one leaf node if they all had the same value.

Problem 2

Part a) was done well, with the majority of students making the right augmentations. Some students made other augmentations such as storing the size of the left and right subtree, or storing how many keys were smaller than the key in the current node and how many were larger. As long as the algorithm is correct and runs in $O(\log n)$ time, they still got the marks.

Almost every student who got part a) right also got part b) right.

Problem 3

The common mistake in part a) was not deriving the final index where the pattern P begins in T.

A few students made mistakes creating the suffix skip array in part b). Students should be careful about the constraint that P[i] $\neq$ P[j] and what they prepend in front of the pattern in order to get the right negative indices.

Many students made mistakes with brackets in part c). Brackets are put around characters when a good suffix skip is done or when a last character jump is done with a character that appears in the pattern. Students should also be aware that they should not continue matching a string once a mismatch is encountered.

In part d), many students gave the right example, but some gave examples that did not result in the worst case. The worst case for this rabin karp example is a string which hashes such that the algorithm needs to compare P with every 3-digit substring in T.

Problem 4

Many students did not submit a program.

Students who achieved $O(n \log n)$ construction time were aware that they had to quickselect for the median, and then partition the points around the median without sorting. Most students who used this algorithm in their program also did the analysis correctly.

Sorting would require $O(n \log n)$ per recursion, so with $O(\log n)$ recursions would result in a runtime of $O(n(\log n)^2)$. Radix sort cannot give an $O(n)$ guarantee since we do not know the maximum size of an input number.