CS 624: Analysis of Algorithms

Assignment 3

Suggested solutions

1. Given $n > 2$ distinct numbers, what is the minimum number of comparisons needed to find a number that is neither the maximum nor the minimum?

   Solution: We already saw that we can find the minimum and maximum simultaneously using $C \leq \lfloor \frac{3n}{2} \rfloor$ comparisons. Finding a number that is neither the minimum nor the maximum takes only three comparisons. We know that if all the numbers are distinct then the first three numbers contain at least one number that is not the minimum or the maximum. Therefore at most you need to compare A[1] to A[2], A[1] to A[3] and A[2] to A[3] and return the middle element. In some cases you may stop after two comparisons. Say, if in the scenario above A[1] turns out to be in the middle between A[2] and A[3], there is no need to compare them to one another.

2. Find the i largest numbers in sorted order using various methods:

   (a) Sort the numbers and list the i largest – sorting is $O(n \log n)$ and then list i numbers, so overall $O(n \log n)$.

   (b) Building a max-PQ takes $O(n)$, followed by i extract-max, each one logarithmic. Overall $O(n + i \log n)$ (notice we don't know how i relates to n, so we have to use both variables in the run time).

   (c) In the worst case finding the i^{th} largest number can take $O(n^2)$. Partition is $O(n)$ and sorting is $O(i \log i)$. The average case, though, is $O(n+i \log i)$. So on average this method is the fastest.

3. Since an edge connects two different vertices and by definition a simple loop contains no vertex more than once, no edge can repeat more than once since it would have to connect the same two vertices (in which case the simple loop would have two repeated vertices which cannot occur by definition).

4. Two things have to be proved:

   - That there is a path from x to y – this follows from the definition of a tree as connected and undirected, so there is a path between every two distinct vertices.
   - That the path is unique – Let us assume there are two or more paths from x to y. As hinted in the question, the two different paths would constitute a loop if we go from x to y in one path and then backtrack to x through the other path. However, the loop doesn't have to be simple since the two paths may share several vertices. However, it can be shown that such a loop would have a simple loop inside of it. Let us denote one of the two paths $x, v_1, v_2, ..., y$ and the other path $x, u_1, u_2, ...y$. Let us look at the last vertex the two paths share before y – call it w (w can be x, or any other vertex before y). We can then follow path 1 from w to y and then backtrack through y to w through

the other path. This loop is simple as we denoted w to be the last shared vertex these paths have before y. Since a tree is defined to have no simple loops, it follows that two unique paths cannot exist, then, between any two vertices.

5. Again, we need to show two things

   - That every node has at most one parent: If a node x had more than one parent, then there would be at least two distinct vertices (say y and z) which fit the definition of a parent: That is, there is a simple path from r to x through y, such that there is one edge between y and x and there is a simple path from r to x through z such that there is one edge between z and x. But this would constitute two simple paths between r and x, which contradicts the previous question.
   - That every node except the root has exactly one parent: If a non-root node x had no parent, it means there would be no path from the root to x of size at least 1, which contradicts the connectivity of the tree. The root does not have a parent, because there is only a path of size 0 from the root to itself, so a root is its own ancestor but by definition, not its own parent.

6. If y is an ancestor of x then there is a simple path from r to x through y. Similarly, if x is an ancestor of y then there is a simple path from r to y through x. If x and y were not the same node this would imply that there is more than one simple path between r and each of x and y, the ones described above. Since we showed in previous exercise that this is impossible, x and y must be the same node for that path from r to be unique.

7. If a and b are both ancestors of x, then they are on the path from r to x by definition of ancestor. x is not the root and therefore that path has at least one edge (this would be if a is the root and b is x or vice versa. Otherwise there are at least two edges). Only one simple path exists between r and x according to previous results, and a and b are on that path, as both are ancestors of x. Therefore, there exist a path from r to a through b (if b precedes a on the path) or vice versa. In the former case, a is an ancestor of b. In the latter, b is an ancestor of a.

8. There is a unique path from r to x and a unique path from r to y. Ancestors of x are all the nodes on the path from r to x (including r and x), and ancestors of y are all the nodes on the path from r to y (including r and y),

9. Let us substitute $T(n) = \alpha n + \beta$ into equation 1 and get $\beta = c$. We then substitute into equation 2 and get:

   $T(n) = c + T(k) + T(n-k-1) + v = c + \alpha k + \beta + \alpha(n-k-1) + \beta + v$

   Opening the brackets and substitute $\beta = c$ we get:

   $T(n) = c + 2\beta + \alpha n - \alpha + v = \alpha n + 3c + v - \alpha$

   Since $T(n) = \alpha n + \beta$, we can substitute the free term and get $3c + v - \alpha = \beta = c \Rightarrow \alpha = 2c + v$. Therefore, $T(n) = (2c + v)n + c$, as suggested.

10. Exercise 4.1 in the Lecture 7 handout.

    (a) If a is an ancestor of x and $a.key > x.key$, then x is in the left subtree of a. Hence, all the subtree rooted at x is also at the left subtree of a, which means every descendant d of x is also smaller than a.

    (b) If a is a successor of x then it is either an ancestor or a descendant of it. Let's assume by contradition that it's neither. According to exercise 1.6, a and x have a least common ancestor y. Since $a > x$ (it's the successor of x), then x is on the left subtree of y and a is on its right subtree. But that would make y in between a and x, making y the successor. Therefore, the least common ancestor of a and x must be either a or x themselves.

(c) According to the previous exercise a, the successor of x is either a descendant or an ancestor. If it's a descendant, it must be the minimum of the right subtree (since anything else on the right subtree must be larger than a). Let's show it can't be an ancestor in this case. If it were an ancestor, x must be on its left subtree (since as a successor $a > x$. However, since x has a non-empty right subtree, then any node on that subtree must be $> x$, but still $< a$ (since it's on a's left subtree). So, a can't be a successor to x.

(d) If the right subtree of x is empty, and if x has a successor y, then y is the lowest ancestor of x whose left child is also an ancestor of x: According to part (b) y must be an ancestor of x (since x has no bigger descendants than itself). First, x must be on the left subtree of y (since $y > x$). Let's show first that no lower ancestor of x can be the successor: y is the the lowest ancestor whose left child z is also an ancestor. This means that x is on the right subtree of z (since y is the lowest one that has x on the left). This makes $x < z$ so z can't be x's successor. It can't also be any ancestor of y. Let's assume by contradiction that there is an ancestor of y, say z, that is the successor of x. This means $z > x$, so x is on z's left subtree, but then so is y, since y is an ancestor of x. We know that x is on y's left subtree, which would put y between x and z, contradicting the assumption that z is an ancestor.

(e) If x has a right child, then the successor of x does not have a left child: According to (c) above, if x has a right subtree, the successor a must be on that subtree. a cannot have a left child since that left child, z, would be $< a$ but $> x$ (since it's on x's right subtree), making it the successor instead of a.

(f) This is exactly the same claim only switch left and right.