CS 561: Homework 4

Prof. Jared Saia, University of New Mexico

Remember: You are encouraged to work on the homework in groups, but please observe the "Star Trek" rule from the syllabus.

1. Consider a recurrence of the form

$$T(n) = aT(n/b) + f(n),$$

where $a \geq 1$ is an integer and $b > 1$ is a real constant, and $f(n) \geq 0$. Assume, as usual, that $T(n) = f(n) = \Theta(1)$ for all constant values of n. Also assume that there is a constant $K > 1$ such that

$$af(x/b) \geq Kf(x)$$

for every $x \geq b$. Let $\ell = \lfloor \log_b n \rfloor$. At level ℓ, every subproblem has constant size; let $L = \Theta(a^\ell)$ be the total cost of the leaves in the recursion tree.

   (a) Show that $L = \Theta\big(n^{\log_b a}\big)$.

   (b) Prove by induction on i that, for every integer $i \in [0, \ell]$,

$$a^i f(n/b^i) \geq K^i f(n).$$

   Clearly identify the base case, inductive hypothesis, and inductive step.

   (c) Use the preceding inequality to show that the cost of level j is at most a constant multiple of

$$\frac{L}{K^{\ell-j}}$$

   for every $j \in [0, \ell]$. Then sum the level costs and prove that

$$T(n) = \Theta\left(n^{\log_b a}\right).$$

   Be sure to justify both the upper and lower bounds.

2. Consider the recurrence

$$T(n) = T(n/3) + T(2n/3) + cn,$$

where $c > 0$ is a constant.

   (a) Explain why the version of the Master Method stated in class does not directly apply to this recurrence.

   (b) Use a recursion tree to prove that $T(n) = \Theta(n \log n)$. In your proof, show that the sum of the subproblem sizes at every level before the first leaf appears is n, show that both the shortest and longest root-to-leaf paths have length $\Theta(\log n)$, and account for the total cost of the leaves. Clearly justify both the upper and lower bounds.

3. Consider the following recursive algorithm, whose argument is any nonnegative integer n.

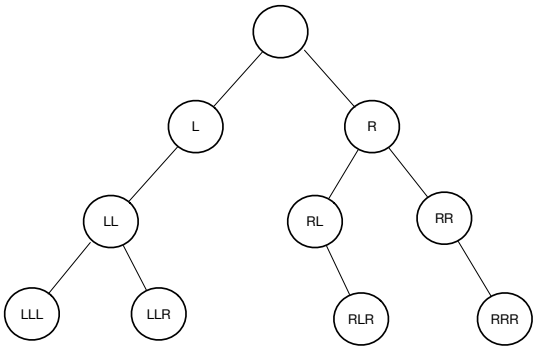
```
RecursiveValue(n)
  if n = 0 then return 2
  if n = 1 then return 5
  return 2 · RecursiveValue(n − 1) − RecursiveValue(n − 2)
```

   (a) Write a recurrence for the *value* returned by RecursiveValue. Use the annihilator method to solve the recurrence exactly, including the constants determined by the base cases, and check your answer.

   (b) Write a recurrence for the *running time* of RecursiveValue. Give a tight asymptotic bound on its solution.

4. Consider a rooted binary tree whose nodes are named as follows. The root is named by the empty string. If a node named σ has a left child, that child is named σL; if it has a right child, that child is named σR.

   Give a recurrence that returns the total number of occurrences of R across the names of all nodes. For example, the following tree contains 10 occurrences of R.



   Hint: For a node v, let $f(v)$ be the number of occurrences of R in the names of nodes in the subtree rooted at v, assuming that the naming starts with the empty string at v. Let $\ell(v)$ and $r(v)$ be the left and right children of v, respectively, or NULL if the corresponding child does not exist. Finally, let $s(v)$ be the number of nodes in the subtree rooted at v, with $s(\text{NULL}) = 0$, and assume that these values are stored at the nodes. Give a base case and a recurrence for $f(v)$.