Problem Set Two

Zoe Farmer

February 26, 2024

1. Consider the following Python function:

```
1  def find(a, target):
2      x = 0
3      y = len(a)
4      while x < y:
5          m = (x + y) / 2
6          if a[m] < target:
7              x = m + 1
8          elif a[m] > target:
9              y = m
10         else:
11             return m
12 return -1
```

   Suppose list a has n elements and is sorted.

   (a) Using Θ notation, what is the best case running time as function of n?

      i. $\Theta(1)$ $\rightarrow$ If the target is the middle element the algorithm completes in constant time.

   (b) Using Θ notation, what is the worst case running time as function of n?

      i. $\Theta(\log_2(n))$ $\rightarrow$ If the target is at one end, the algorithm will divide the array in half each time until the target is found.

2. In the classic version of Quicksort, the pivot is always chosen as the last element in the input array. When the pivot is compared to another element, we can use any valid comparison operator. Thus, "sorting" generalizes to any set of inputs over which we can mathematically define comparisons.
   Use this version of the algorithm to sort the following functions by order of asymptotic growth such that the final arrangement of functions $g_1, g_2, \cdots, g_{12}$ satisfies the ordering constraint $g_1 = \Omega(g_2), g_2 = \Omega(g_3), \cdots, g_{11} = \Omega(g_{12})$.

n
n^2
$\left(\sqrt{2}\right)^{\log_2(n)}$
$2^{\log_2^*(n)}$
$n!$
$(\log_2(n))!$
$\left(\frac{3}{2}\right)^n$
$n^{\frac{1}{\log_2(n)}}$
$n\log_2(n)$
$\log_2(n!)$
e^n
1

(a) For each branch of the recursion tree, identify the chosen pivot element; for each level of the recursion tree and after all pivot elements at that level have been moved into their final location, give the global array ordering. (Hint: it may be easier to work out the final ordering by hand and then backtrack through the Quicksort operations to produce the recursion tree.)

i. Solving the ordering first allows us to obtain the array representing the order of the given array:

n	n^2	$\left(\sqrt{2}\right)^{\log_2(n)}$	$2^{\log_2^*(n)}$	$n!$	$(\log_2(n))!$	$\left(\frac{3}{2}\right)^n$	$n^{\frac{1}{\log_2(n)}}$	$n\log_2(n)$	$\log_2(n!)$	e^n	1
5	9	3	4	12	8	10	2	7	6	11	1

We can use Quicksort to sort this array. See Figure 1. Each node is the pivot for that step.
At each step the array is in a certain order, as is displayed in Table 1.

1	[5, 9, 3, 4, 12, 8, 10, 2, 7, 6, 11, 1]
2	[1, 9, 3, 4, 12, 8, 10, 2, 7, 6, 11, 5]
3	[1, 3, 4, 2, 5, 8, 10, 9, 7, 6, 11, 12]
4	[1, 2, 4, 3, 5, 8, 10, 9, 7, 6, 11, 12]
5	[1, 2, 3, 4, 5, 6, 8, 10, 9, 7, 11, 12]
6	[1, 2, 3, 4, 5, 6, 10, 9, 7, 8, 11, 12]
7	[1, 2, 3, 4, 5, 6, 7, 8, 10, 9, 11, 12]
8	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]

Table 1: Global Array at Each Step

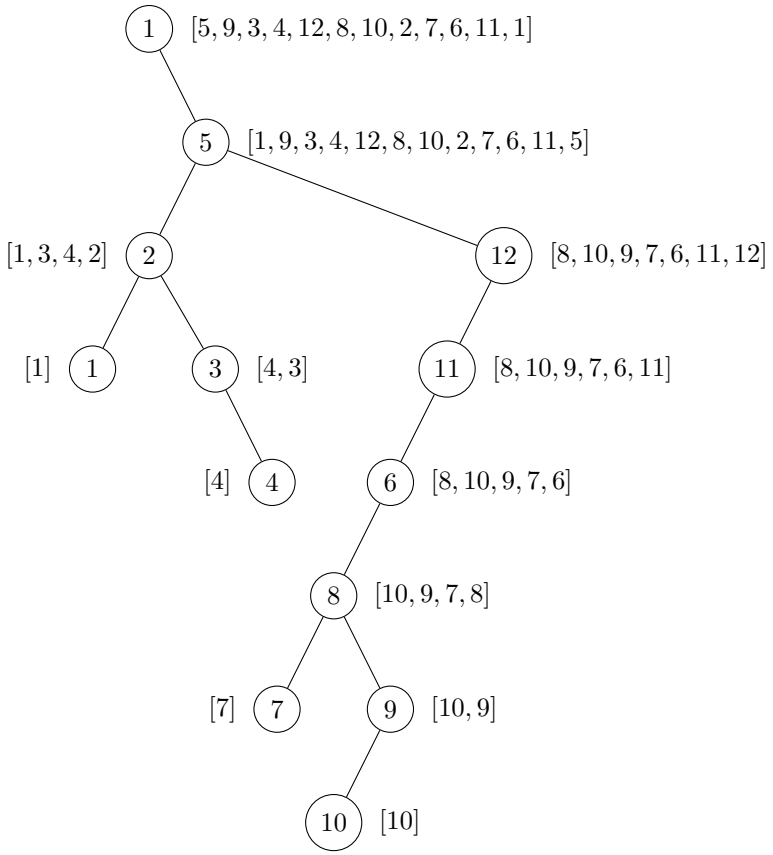

Figure 1: Quicksort Binary Tree with Pivots

(b) Give the final sorted list and identify which pair(s) functions $f(n)$, $g(n)$, if any, are in the same equivalence class, i.e., $f(n) = \Theta(g(n))$.

i. The resulting array is shown below.

1	1
2	$n^{\frac{1}{\log_2(n)}}$
3	$\left(\sqrt{2}\right)^{\log_2(n)}$
4	$2^{\log_2^*(n)}$
5	n
6	$\log_2(n!)$
7	$n \log_2(n)$
8	$(\log_2(n))!$
9	n^2
10	$\left(\frac{3}{2}\right)^n$
11	e^n
12	$n!$

3. Consider the following recursive function `f`, which takes an integer argument `n` and returns some other integer `f(n)`:

```
1  def f(n)
2      if n == 0
3          return 3;
4      else if n == 1
5          return 5;
6      else
7          val = 3 * f(n-1);
8          val = val - 2 * f(n-2);
9          return val;
```

(a) Write down the recurrence relation for the value returned by `f(n)`; identify the base cases.

i.
$$\begin{cases} T(n) = 3 \cdot T(n-1) - 2 \cdot T(n-2) \\ T(0) = 3 \\ T(1) = 5 \end{cases}$$

(b) Using the method of the characteristic polynomial, solve the value recurrence relation exactly. (See `http://bit.ly/1dcsNUJ`)

$$\begin{aligned}
\text{Given: } &\rightarrow & T(n) &= 3 \cdot T(n-1) - 2 \cdot T(n-2) \\
\text{Assume } &\rightarrow & a_n &= cr^n \\
\text{We can represent } T(n) \text{ as } &\rightarrow & a_n &= 3a_{n-1} - 2a_{n-2} \\
\text{Substituting } &\rightarrow & cr^n &= 3cr^{n-1} - 2cr^{n-2} \\
\text{Characteristic Polynomial Roots } &\rightarrow & -r^2 + 3r - 2 &= 0 \rightarrow \text{Roots: } [1, 2] \\
\text{Therefore } &\rightarrow & a_n &= c_1 + c_2 2^n \\
\text{Using our base cases: } &\rightarrow & &\begin{cases} 3 = c_1 + c_2 \\ 5 = c_1 + 2c_2 \end{cases} \\
\text{Therefore } &\rightarrow & c_1 = 1, c_2 &= 2 \\
\text{Therefore } &\rightarrow & \boxed{a_n = 2^{n+1} + 1} & \blacksquare
\end{aligned}$$

(c) Show (prove) by induction that your solution is correct.

i. Let $P(n)$ be our propositional function:

$$\begin{cases} P(n) \equiv 3a_{n-1} - 2a_{n-2} = 2^{n+1} + 1 \\ P(0) \equiv 3 \\ P(1) \equiv 5 \end{cases}$$

ii. Show $P(2)$

$$3 \cdot 5 - 2 \cdot 3 = 2^3 + 1 \Rightarrow 9 = 9$$

iii. Assume $P(k)$. Also assume that $P(k) \Rightarrow P(k+1)$

$$\begin{aligned} 3a_{k-1} - 2a_{k-2} = 2^{k+1} + 1 \Rightarrow 3a_k - 2a_{k-1} &= 2^{k+2} + 1 \\ a_{k-1} &= 2^k + 1 \\ 3 \cdot 2^{k+1} + 3 - 2 \cdot 2^k - 2 &= 2^{k+2} + 1 \\ 3 \cdot 2^{k+1} - 2 \cdot 2^k &= 2^{k+2} \\ 3 \cdot 2^{k+1} - 2^{k+1} &= 2^{k+2} \\ 2^{k+2} &= 2^{k+2} \blacksquare \end{aligned}$$

(d) Under the `RAM` model of computation, write down the recurrence relation for the running time of `f(n)`.

i. The recurrence relation above can be redefined in terms of operations.

$$T(n) = \overbrace{\underbrace{3}_{1} \cdot T(\overbrace{n-1}^{1}) - \underbrace{2}_{1} \cdot T(\overbrace{n-2}^{1})}^{1}$$

As you can see, each level requires 5 operations and is called twice. This allows us to define the the running time of the recurrence relation as

$$R(n) = 2 \cdot R(n-1) + 5$$

This overcounts the total number of operations by one, and so our base case is $R(1) = 5$

(e) Give a succinct explanation in terms of the shape of the recursion tree of why $O(2^n)$ is a trivial upper bound on the running time.

i. At any given node of the binary tree there will either be two children or none, which would indicate that the branch bottomed out at a base case. This means that there are n levels of the tree, and at each level there are 2^n branches.

(f) Using the method of characteristic polynomials, solve the time recurrence relation for a tight upper bound. (Hint: $O(2^n)$ is not tight.)

$$\begin{aligned} \text{Given: } \rightarrow \quad & R(n) = 2 \cdot R(n-1) + 5 \\ \text{Assume } \rightarrow \quad & a_n = cr^n \\ \text{We can represent } R(n) \text{ as } \rightarrow \quad & a_n = 2a_{n-1} + 5 \\ \text{Substituting } \rightarrow \quad & cr^n = 3cr^{n-1} + 5 \\ \text{Characteristic Polynomial Roots } \rightarrow \quad & -r^2 + 2r = 0 \rightarrow \text{Roots: } [0, 2] \\ \text{Therefore } \rightarrow \quad & a_n = c_2 2^n - 1 \\ \text{Using our base case: } \rightarrow \quad & 5 = c_2 \\ \text{Therefore } \rightarrow \quad & \boxed{5(2^n - 1)} \blacksquare \end{aligned}$$

4. Solve the following recurrence relations using the method specified. Show your work.
   (a) $T(n) = T(n1) + n$ by unrolling (tail recursion).
      i. Tail Recursion involved substituting the next step into the current one, and then estimating an answer based on the results.

$$\begin{array}{rcl} T(n) = T(n-1) + n & \rightarrow & T(n-1) = T(n-2) + n - 1 \\ T(n) = T(n-2) + 2n - 1 & \rightarrow & T(n-2) = T(n-3) + n - 2 \\ T(n) = T(n-3) + 3n - 3 & \rightarrow & T(n-3) = T(n-4) + n - 3 \\ T(n) = T(n-4) + 4n - 6 & & \end{array}$$

$$\boxed{T(n) = c + \frac{1}{2}n(n+1)} \blacksquare$$

   (b) $T(n) = 2T\left(\frac{n}{2}\right) + n^3$ by the Master method.
      i. The Master Method states that for a recurrence relation of the form

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

if $f(n) = \Theta(n^c)$ where $c > \log_b(a)$, then $T(n) = \Theta(f(n))$. In this problem, $3 > \log_2(2)$, therefore

$$\boxed{T(n) = \Theta(n^3)}$$

5. Professor Septima Vector thinks she has discovered a remarkable property of binary search trees. Suppose that the search for key k in a binary search tree ends up in a leaf. Consider three (possibly empty) sets: A, the keys to the left of the search path; B, the keys on the search path; and C, the keys to the right of the search path. Professor Vector claims that any three keys $a \in A$, $b \in B$ and $c \in C$ must satisfy $a \leq b \leq c$. Prove that Vector's claim is false by giving a counterexample that is the smallest possible.

   (a) The easiest counterexample given the phrasing of her property is when k is the smallest element in the tree. In this case, $A = \emptyset$ and thusly no key a could possibly be in A, $\nexists a$. However, depending on the meaning of the question it could also be interpreted that this case falls to $b \leq c$ which is true.
   Therefore an alternate counterexample is when there is a zig-zag in the nodes. Please reference Figure 2 for a pictoral representation. If $k = 7$, then $A = 3, 6, B = 10, 5, 7$, and $C = 15$. If $a = 6, b = 5$, and $c = 15$ then Professor Vector's property states that $6 \leq 5 \leq 15$ which is clearly false.

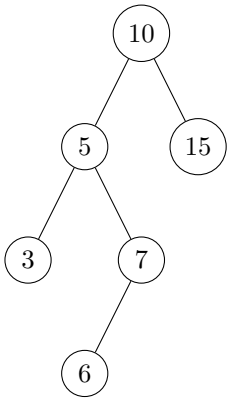

Figure 2: Counterexample Binary Tree