Chapter Three Assignment Solutions

CS310

October 15, 2013

1. Prove : $T(n) = T(n-1) + \Theta(n) = \Theta n^2$ by substitution

Suppose the guess, $T(n) = \Theta(n^2)$, is true. In order to prove that T(n) is $\Theta(n^2)$, one must show, through substitution, that there exist constants that make cn^2 both an upper and lower bound for the recurrence.

$\frac{c_1}{2}(n-1)^2 - 1 \subset \Theta(n^2)$ so one can substitute it.

$T(n) <= \frac{c_1}{2}(n-1)^2 - 1 + c_1 n$

$T(n) <= \frac{c_1}{2}(n^2 - 2n + 1 - 1) + c_1 n$

$T(n) <= \frac{c_1}{2}n^2 - c_1 n + c_1 n$

$T(n) <= \frac{c_1}{2}n^2$

At this point, we have proven that $T(n) \subset \mathrm{O}\big(n^2\big)$

$T(n) >= \frac{c_3}{2}(n-1)^2 - 1 + c_3 n$

$T(n) >= \frac{c_3}{2}(n^2 - 2n + 1 - 1) + c_3 n$

$T(n) >= \frac{c_3}{2}n^2 - c_3 n + c_3 n$

$T(n) >= \frac{c_3}{2}n^2$

At this point, we have proven that $T(n) \subset \Omega(n^2)$ Since T(n) is also $\subset \mathrm{O}\big(n^2\big), T(n) \subset \Theta(n^2)$.

2.

3. Although Quicksort's asymptotic behavoir beats that of InsertionSort, real-world use cases may not be approximately asymptotic. If, as in the case of the bank,the average case invovles InsertionSort performing a constant number of swaps:

$T(n) = kT_{shift} + T(n-1)$ where $k, T_{shift} \subset \Theta 1$

$T(n) = \sum_{i=1}^{n} \Theta 1$

$T(n) \subset \Theta n$

Quicksort, on the other hand, will still take $\Theta n * \log n$ time.

4. Since $0 < \alpha <= 1/2$, $1/2 <= (1 - \alpha) < 1$

Because the recurrence terminates at n = 1, $T(1) = T(\alpha^{h_l} * n)$ and $= T((1 - \alpha)^{h_r} * n)$ are expressions for the lefthand and righthand heights of the tree (where the left child of a node is $\alpha * parent$). $(1 - \alpha) > \alpha$ means that $h_r > h_l$. Because any product, $\alpha^n * (1 - \alpha)^m$, is less than $(1 - \alpha)^{n+m}$ and greater than α^{n+m}, the corresponding height would be between h_r (the maximum) and h_l (the minimum).

$T(1) = T(\alpha^{h_{min}} * n)$

$k_1 = k_2 * \alpha^{h_{min}} * n$ where $k_1, k2 \subset \Theta(1)$

$\log_\alpha(k_1/(k_2 * n)) = h_{min}$

$h_{min} = k_3 * \frac{\log(1/n)}{\log(\alpha)}$ // by log arithmetic rules

$h_{min} = k_3 * -\frac{\log(n)}{\log(\alpha)}$ // by log arithmetic rules

$T(1) = T((1 - \alpha)^{h_{max}} * n)$

$k_5 = k_4 * (1 - \alpha)^{h_{max}} * n$

$\log_{(1-\alpha)}(k_1/(k_2 * n)) = h_{max}$

$h_{max} = k_6 * \frac{\log(1/n)}{\log(1-\alpha)}$ // by log arithmetic rules

$h_{max} = k_6 * -\frac{\log(n)}{\log(1-\alpha)}$ // by log arithmetic rules

5.

6.

7. Definition of $\Omega(T_2)$: set of all T_1 $\lim_{n\to\infty} \frac{T_1(n)}{T_2(n)} > 0$

In this case, $T_2(n) = \Theta(n * \log n)$.

Suppose randomized quicksort selects the maximum element of the un-

sorted subset for every iteration: $T_1(n) = T_{left} + T_{right} + \Theta(n)$ $T_1(n) = T(1) + T(n-1) + \Theta(n)$. In homework problem one, this recurrence was shown to be $\Theta(n^2)$. $\lim_{n\to\infty} \frac{\Theta(n^2)}{\Theta(n \log n)} > 0$ is true so the runtime of randomized quicksort is big omega of n lg n.

8.

9.

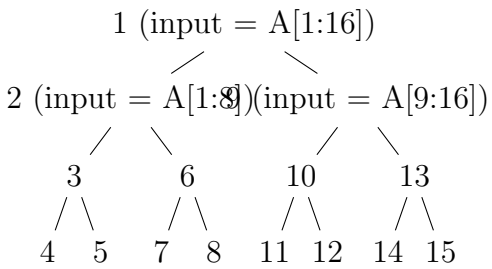
Making quicksort tail recursive.

(a) The correctness of a sorting procedure is proven by demonstrating that the loop invariant is, in fact, invariant. The loop invariants for Quicksort are:

I If $p <= k <= i$, then $A[k] < A[i+1]$

II If $i+1 < k <= j-1$, then $A[k] > A[i+1]$

The postcondition of Partition is a list with the loop invariant met. If and only if, at recursive depth n, Partition has been called (n-1) times on the input size, will the loop invariant work towards a sorted list. In the following graph, the depth of a node represents the number of times Partition has been called on the local input. The number in the node represents the sequential order of the call to Partition. If the value of every node is greater than the value of its parent, the correctness condition is met.



The correctness condition is met.

(b) The stack depth will be $\Theta(n)$ if the pivot is always the maximum element.

(c)

```
while p less than r do
    q = Randomized-Partition(A,p,r);
    Tail-Recursive-Quicksort(A,lower,upper);
    p = q + 1;
end
```

This modification makes the worst case stack depth $\Theta(lgn)$ because Randomized-Parition always puts a fraction on either side of the pivot. See the text for the definition of Randomized-Partition.