Homework 3 Solutions

CSC 140 – Advanced Algorithm Design and Analysis

If you find any errors in the solution writeup, please let me know. Ask in class or come to office hours if you need any further help understanding the problems and their solutions.

1) (a) $n^4 \in \Omega(n^{\log_2 2+\varepsilon})$ for all $0 < \varepsilon \leq 3$, so $T(n) \in \Theta(n^4)$.

(b) $7n/10 = n/(10/7)$, so the logarithm in question is $\log_{10/7} 1$ which despite the unusual base is 0, because $\log 1$ is always 0. $n^1 \in \Omega(n^{0+\varepsilon})$ for all $0 < \varepsilon \leq 1$, so $T(n) \in \Theta(n)$.

(c) $n^2 \in \Theta(n^{\log_4 16})$, so $T(n) \in \Theta(n^2 \log n)$.

(d) $\log_3 7 \approx 1.77$, so $n^2 \in \Omega(n^{\log_3 7+\varepsilon})$ for all $0 < \varepsilon \leq 0.22$, so $T(n) \in \Theta(n^2)$.

(e) $\log_2 7 \approx 2.81$, so $n^2 \in O(n^{\log_2 7-\varepsilon})$ for all $0 < \varepsilon \leq 0.80$, so $T(n) \in \Theta(n^{\log_2 7}) \approx \Theta(n^{2.81})$.

(f) $\sqrt{n} \in \Theta(n^{\log_4 2})$, so $T(n) \in \Theta(\sqrt{n} \log n)$.

(g) The master theorem does not work on this recurrence. Either via repeated substitution or by drawing a recursion tree it's easy to see that $T(n) = n^2 + (n-2)^2 + (n-4)^2 + \cdots$. Asking Wolfram Alpha to solve this sum for me ("sum (n-2i)^2, for i=0 to n/2") we get $(n/6)(n+1)(n+2)$ which is $\Theta(n^3)$.

2) The row sums are 1-2-4-8-16-etc, so they are doubling. The short side of the tree has problem sizes decreasing by 2 each time, so the tree is complete through depth $n/2$. The sum $1 + 2 + 4 + \cdots 2^{n/2}$ is $2^{n/2+1} - 1$. If the tree were complete through n levels, the row sums would sum to $2^{n+1} - 1$. So, the lower bound for `fib` running time is $\Omega(2^{n/2})$ and the upper bound is $O(2^n)$. Regardless, both are exponential, so this is a bad way to implement `fib`.

3) You can do this either by having non-recursive work dominate and be in $\Theta(n^2)$ or have the number of leaves dominate and be in $\Theta(n^2)$. I'll do one of each.

For the number of leaves to be quadratic we need $\log_b a = 2$, so for example $T(n) = 4T(n/2)$ or $T(n) = 9T(n/3)$. Here's a toy algorithm with recurrence $T(n) = 4T(n/2) + 1$.

```
int quadratic_leaves(int n) {
    if (n==0)
        return 1;
    return quadratic_leaves(n/2) + quadratic_leaves(n/2) +
           quadratic_leaves(n/2) + quadratic_leaves(n/2);
}
```

For the non-recursive work to dominate and be quadratic we need $\log_b a < 2$ and something non-recursive to take $\Theta(n^2)$ time, so for example $T(n) = 2T(n/2) + n^2$.

```
int quadratic_nonrecursive(int n) {
    if (n==0)
        return 1;
    int sum = 0;
    for (int i=0; i<n*n; i++)
        sum += i;
    return quadratic_nonrecursive(n/2) + quadratic_nonrecursive(n/2);
}
```

4) Your plot should look undeniably like the right side of a parabola.

5) My solution has recurrence relation $T(n) = 3T(n/4) + 1$. Since $1 \in O(n^{\log_4 3-\varepsilon})$ this means my algorithm is in $\Theta(n^{\log_4 3})$, or about $\Theta(n^{0.79})$. This is worst case and occurs when the key is not in the array. Also, note that this is an upper bound since the subproblems are slightly smaller than $n/4$.