Amena Tajammul

CS 430 Homework #2

1. **(2 pts) Prove, by induction on k, that level k of a binary tree has less than or equal to 2k nodes (root level has k=0).**

Prove: l(k) <= 2^k, 0<= k <= n
Base case: when k=0, nodes = 2^0 = 1 node max in the binary tree root
Assuming the statement to be correct, we then use induction. The maximum number of leaves to replace a previous leaf is 2. So,

    l(k+1) = 2*l(k)

Using induction,

    l(k+1) = 2*2^k = 2^(k+1)

Therefore, our claim is true because k can be any number greater than or equal to zero.

2. **(2pts) Show that the solution of the recurrence relation T(n) = 2T(n/2) + n is Ω(nlgn).**

First, we have to prove that T(n) <= c*n*log(n)
Assuming the statement above to be correct,

    T(n/2) <= c*(n/2)*log(n/2)

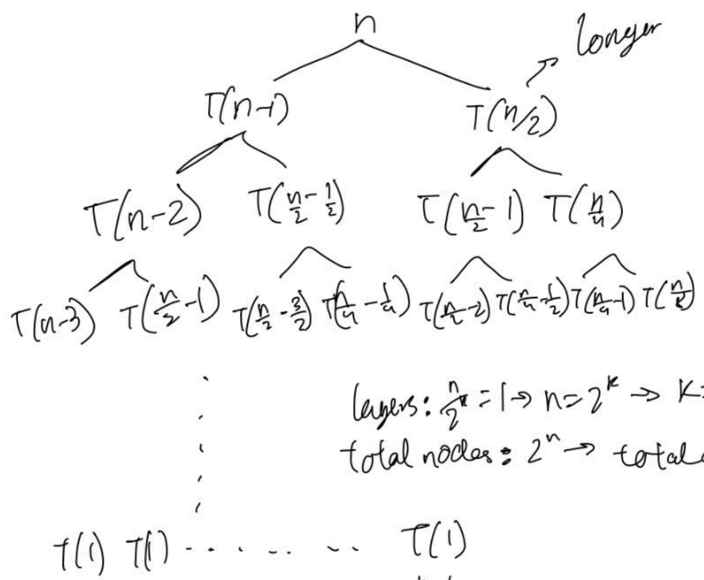
Then,

    T(n) <= 2(c*(n/2)*log(n/2)) + n
    T(n) <= c*n*log(n/2) + n
    T(n) <= c*n*log(n) - c*n*log(2) + n
    T(n) <= c*n*log(n) - c*n + n
    T(n) <= c*n*log(n), for c>=1

Base case: When n=1, T(1)<=0, which we can't have so we need to change the base case. Instead, we can do n=2, T(2) = 2*T(1)+2 = 4. When n=3, T(3) = 2*T(1) + 3 = 5. We need both because as n increases, the recurrence will always go back to either T(2) or T(3).
To find a value for c that satisfies the constraints of the base cases, we can try c=1. When n=2, 4<=2log(2) which is not true. So we can try c=2. When n=2, 4<=2*2log(2) is also not true. We can try increasing it to c=10. When n=2, 4<=10*2log(2), which is true. When n=3, 5<=10*3log(3).
Therefore, we proved that T(n) <= 10*(n)*log(n) for all n>=2. We can then say T(n) = Ω(n*log(n))

3. **(4pts) Use a recursion tree to guess the asymptotic upper bound on the recurrence relation: T(n)=T(n-1)+T(n/2)+n. Then use the substitution method to show your guess is correct.**

Amena Tajammul



$\frac{3n}{2} - 1$

$\frac{9n}{4} - \frac{7}{2}$

$\frac{27n}{8} - \frac{27}{4}$

layers: $\frac{n}{2^k} = 1 \to n = 2^k \to k = \log_2 n$

total nodes: $2^n \to$ total cost $= \Theta(2^n)$

cost per layer $= \left(\frac{3n}{2}\right)^k - \left(\frac{5}{2}\right)^{k-1} - 1$

$= \left(\frac{3n}{2}\right)^{\log_2 n} - \left(\frac{5}{2}\right)^{\log_2 n - 1} - 1$

$= \frac{(3n)^{\log_2 n}}{2^{\log_2 n}} - \frac{5^{\log_2 n - 1}}{\frac{2^{\log_2 n}}{2}} - 1$

$= 3^{\log_2 n} \cdot n^{\log_2 n - 1} - \left(\frac{2}{5}\right) 5^{\log_2 n} - 1$

$= n^{\log_2 3} \cdot n^{\log_2 n - 1} - \frac{2}{5} n^{\log_2 5} - 1$

$= n^{\log_2 (3n) - 1.58} - \frac{2}{5} n^{2.32} - 1$

$T(n) \leq \left(n^{\log_2(3n) - 1.58} - \frac{2}{5} n^{2.32} - 1\right)(\log_2 n) + \Theta(2^n)$

$T(n) \leq n^{\log n} \cdot \log n - n^2 \log n - \log n + \Theta(2^n)$

$T(n) \leq \Theta(2^n)$ fastest growing so upper bound

T(n) <= c*2^n - 4n
T(n) <= c*(2^(n-1)) - 4(n-1) + c*2^(n/2) - 4(n/2) + n = c*(2^(n-1) + 2^(n/2)) - 5n + 4
T(n) <= c*(2^(n-1) + 2^(n/2)) - 4n = c*(2^(n-1) + 2^(n-1)) - 4n
T(n) <= c*2^n - 4n = Θ(2^n)

Subsequently,

T(n) >= c*n^2*T(n) >= c*n^2
T(n) >= c(n-1)^2 + c(n/2)^2 + n = (5/4)*c*n^2 + (1-2c)*n + c
T(n) >= cn^2 + (1-2c)*n + c
T(n) >= cn^2 = Ω(n^2)

4. **(2pts) Please recall Binary Search. To search for a value k in a sorted array A by binary search, we check the midpoint of A against k to halve the size of the remaining portion. Repeat this procedure until we find k in A or verify k's nonexistence in A. What is the recurrence relation of this algorithm? And is its asymptotic bound Θ(lgn)? Use the master theorem to show your solution.**

Binary search algorithm has a recurrence relation of T(n) = T(n/2) + O(1)

Using the master theorem, $a=1$, $b=2$, and $f(n) = 1$. $Log_b(a) = 0$ and n^0 is 1. Therefore, case 2 applies and the upper $\Theta((n^0)*lgn) = \Theta(lgn)$