Jane Downer

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

Base case: $k = 0$

Expect 1 leaf (root node)

$\text{Leaves}(0) \le 2^0 = 1$ ✓

Induction step: Consider a complete binary tree with $k = n$ levels

A complete binary tree with $k = n+1$ levels will have twice as many leaf nodes — fewer, if the tree is not complete.

i.e., we expect $\text{Leaves}(n+1) \ge 2 \cdot \text{Leaves}(n)$

Induction hypothesis: $\text{Leaves}(n) \le 2^n$

$\text{Leaves}(n+1) \le 2^{n+1} = 2^n \cdot 2^1 = 2 \cdot \text{Leaves}(n)$ Q.E.D.

2. (2pts) Show that the solution of the recurrence relation $T(n) = 2T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n$ is Ω (nlgn)

Guess: $T(n) \ge c \cdot n \cdot \lg n$

Base case: $n = 1$: $T(1) = 2T\left(\lfloor \tfrac{1}{2} \rfloor\right) + 1$

$= 2T(0) + 1$

$\ge c \cdot 1 \cdot \lg(1)$

$= 0$

conflict

For $n > 3$, formula does not rely on $T(1)$. So that will work as long as we give base cases for $n = 2$ and $n = 3$.

Keep boundary $T(1)=1$ for sake of new base cases

$$T(2) = 2T\left(\lfloor \tfrac{2}{2} \rfloor\right) + 2 \qquad c(2)\cdot \lg(2) = 2c\cdot 1$$
$$= 2T(1) + 2 \qquad = 2c$$
$$= 2\cdot 1 + 2$$
$$= 4 \longrightarrow T(2) \geq c(2)\cdot \lg(2) \quad \text{for } 0 \leq c \leq 2$$

$$T(3) = 2T\left(\lfloor \tfrac{3}{2} \rfloor\right) + 3 \qquad c(3)\cdot \lg(3) = 3c * x$$

↳ $1 < x < 2$

↳ between $3c$ and $6c$

↳ only works with $0 \leq c \leq 1$

$$= 2T(1) + 3$$
$$= 2\cdot 1 + 3$$
$$= 5 \longrightarrow T(3) \geq c(3)\cdot \lg(3) \quad \text{for } c = 1$$

base cases: $\begin{cases} T(2) = 4 \\ T(3) = 5 \end{cases}$

$$T(n) = 2T\left(\lfloor \tfrac{n}{2} \rfloor\right) + n$$

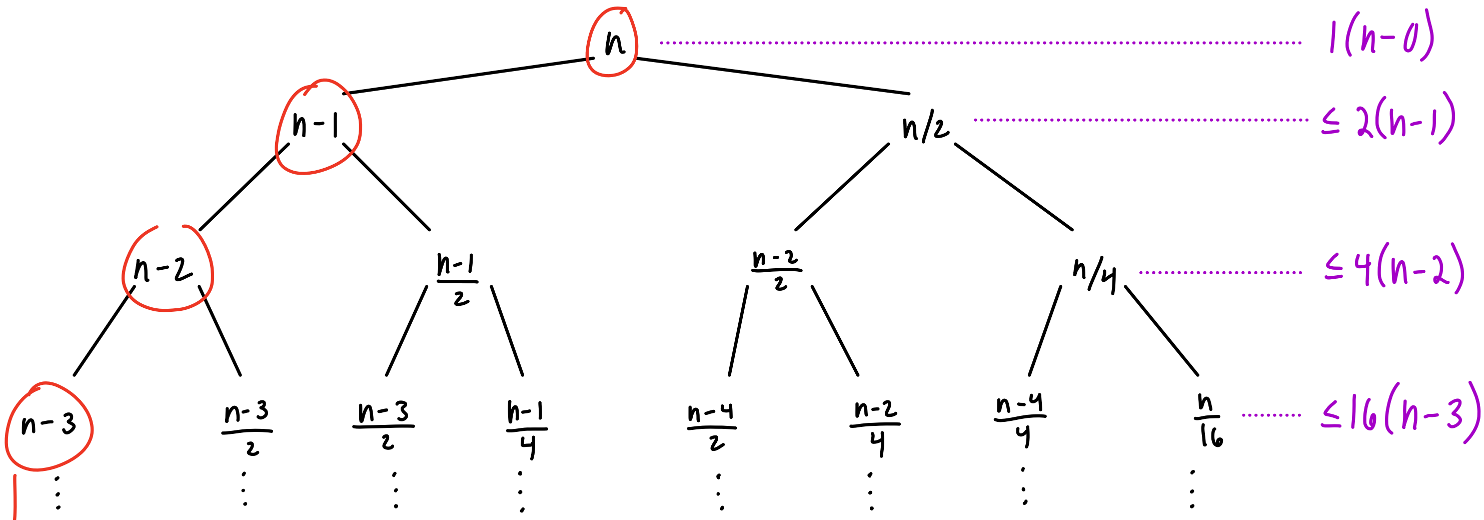
Suppose

$$T\left(\lfloor \tfrac{n}{2} \rfloor\right) = c\cdot \lfloor \tfrac{n}{2} \rfloor \cdot \lg \lfloor \tfrac{n}{2} \rfloor$$

$$T(n) = 2\cdot c \lfloor \tfrac{n}{2} \rfloor \cdot \lg \lfloor \tfrac{n}{2} \rfloor + n$$
$$= cn\cdot \lg \lfloor \tfrac{n}{2} \rfloor + n$$
$$= cn\cdot \lg n - cn\cdot \lg 2 + n$$
$$= cn\cdot \lg n - cn + n$$
$$\geq cn\cdot \lg n \quad \text{for } 0 \leq c \leq 1$$

✓✓ Q.E.D.

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.

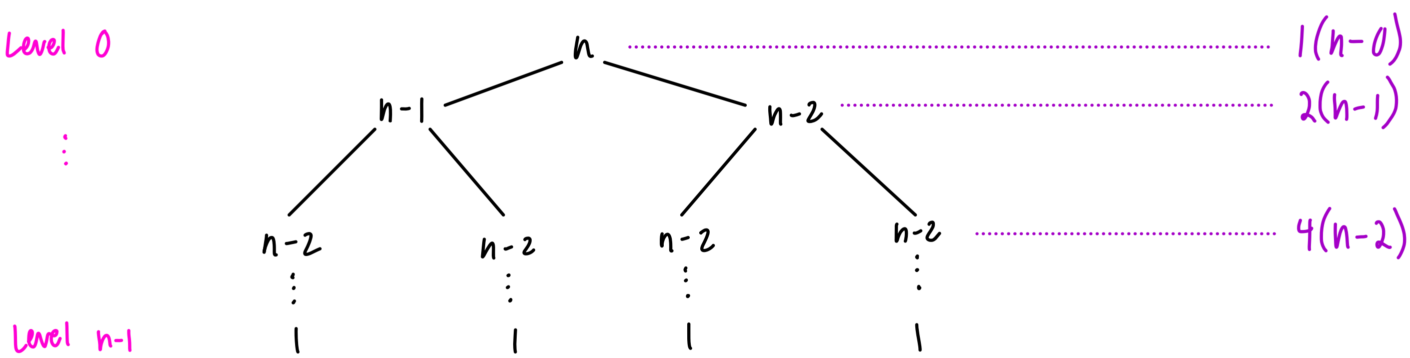


As $n \to \infty$, these terms are the largest in their levels

At each level $k = \underset{\substack{\uparrow \\ \text{root}}}{0} \dots L$, the cost is $\leq 2^k(n-k)$

Therefore, we can say
$$T(n) = T(n-1) + T(n/2) + n$$
$$\leq T(n-1) + T(n-1) + n \longleftarrow \text{call this } T(n)^*$$

For $T(n)^*$, the recursion tree is as follows:



$$cost(T(n)^*) = \overset{\text{first level}}{2^0(n-0)} + 2^1(n-1) + \dots + \overset{\text{last level}}{2^{n-1}\cdot 1}$$
$$= \sum_{k=0}^{n-1} 2^k(n-k)$$
$$= n\sum_{k=0}^{n-1} 2^k - \sum_{k=0}^{n-1} k\cdot 2^k$$

$$= n \cdot \left(\frac{2^{n-1+1} - 1}{2-1}\right) - \left[(n-2) \cdot 2^n + 2\right]$$

$$= n \cdot 2^n - n - n \cdot 2^n + 2 \cdot 2^n - 2$$

$$= -n + 2 \cdot 2^n - 2$$

$$\leq 2 \cdot 2^n$$

$$= O(2^n) \longleftarrow \underline{\text{Guess}}$$

<u>Substitution:</u>

hypothesis: $\begin{cases} T(n-1) \leq c \cdot 2^{n-1} \\ T(n/2) \leq c \cdot 2^{n/2} \end{cases}$

$$T(n) = T(n-1) + T(n/2) + n$$

$$\leq c \cdot 2^{n-1} + c \cdot 2^{n/2} + n$$

$$= \underbrace{\frac{c}{2} \cdot 2^n}_{\text{leading term}} + c\left(\sqrt{2}\right)^n + n$$

$$= O(2^n)$$ ✓✓ Q.E.D.

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 $\Theta(\lg n)$? Use the master theorem to show your solution.

Recurrence Relation:

size of problem: n
\# subproblems per cycle: 1
size of each subproblem: $n/2$
Time to divide problem of size n: $\Theta(1)$
Time to combine subproblems: n/a
Time to directly solve small problem: $\Theta(1)$

$\hookrightarrow T(n) = T(n/2) + \Theta(1)$

Asymptotic Bound:

$\hookrightarrow$ Master Theorem:

$$T(n) = T(n/2) + \Theta(1)$$
$$= a \cdot T(n/b) + f(n) \implies a=1,\ b=2,\ f(n) = 1 = n^0$$

For some positive constant $\epsilon > 0$,

I. $0 = \log_b a - \epsilon$? X

II. $0 = \log_b a$? ✓

III. $0 = \log_b a + \epsilon$? X

Under Case II:

$$T(n) = \Theta(n^{\log_b a} \cdot \lg n)$$
$$= \Theta(n^{\log_2 1} \cdot \lg n)$$
$$= \Theta(n^0 \cdot \lg n)$$
$$= \Theta(\lg n)$$