Name: Shuo Wu Student ID:452666 wustl key: wushuo

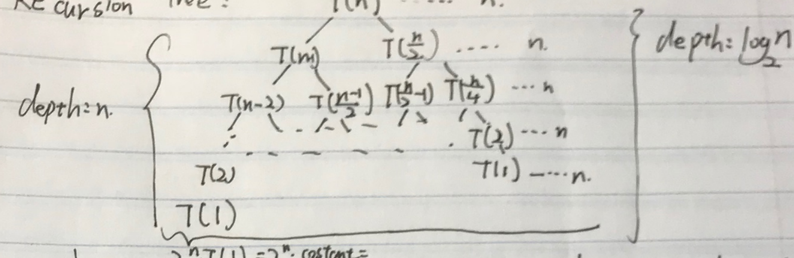
CSE541 HW2

Q1,

HW2

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

Recursion tree:



The recursion tree ~~looks~~ on the left there is a long branch which length is "n", on the right the length of the branch is $\log_2 n$.

So for upper bond, I guess should be $\leq T(n) = 2T(n-1) + n$

us~~ing master method~~ So $T(n) = 2[T(n-1)] + n$

$= 2[2T(n-2) + (n-1)] + n.$

$= 2^n \cdot \text{constant} + n^2 = 2^n$

$\therefore T(n) = O(2^n)$

※Substitution method.

$T(n) = T(n-1) + T(\frac{n}{2}) + n,$

$\leq 2^{n-1} + 2^{\frac{n}{2}} + n$ (when $n >>$ constant)

$\leq 2^{n-1} + 2^{n-1} = 2^n$

$\therefore T(n) \leq 2^n.$ where $n >> 1$,

also $O(2^n)$ is a tight bound, Since if we have any polynomial upper bound like $T(n) \leq cn^k$ $\therefore T(n) = T(n-1) + T(\frac{n}{2}) + n$

$\leq c(n-1)^k + c\frac{n^k}{2^k} + n$

$\leq cn^k(1 + \frac{1}{2^k}) > cn^k$

$\therefore$ this is $O(2^n)$ a pretty tight bound.