Lecture 7: Recursion tree (4.4)

CS303: Algorithms

Last update: January 21, 2014

1 Review

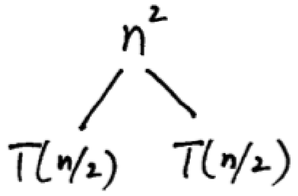
- Analysis of non-recursive algorithms
- Substitution method for recursive algorithms

Example
Find the class for $T(n) = T(n-1) + 1$ using substitution method.
Guess:$T(n)$ is $O(n)$
Proof:
if $T(k) \leq ck$ for all $k \leq n$, then $T(n) = T(n-1) + 1 \leq c(k-1) + 1 = ck - (c-1)$. Thus if I pick a c as 2, $T(n) \leq ck$. Thus induction is proved.
base case: $T(1)$ is $\Theta(1) < c$. So c has to be large enough (bigger than the time to process the T(1) trivial time)

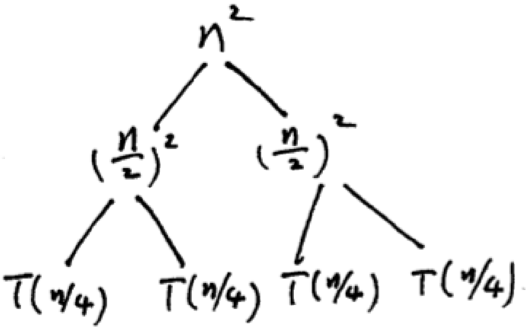
2 Recurrence tree

It is more intuitive but less strict. It is usually used as a means to guess the form of $T(n)$ before proving it using the substitution method.
The overall strategy is to summarize the work load at every level of the tree and find a pattern. Sometimes the tree height matters and sometimes it doesn't. (more on this when we discuss the master therorem)
E.g. $T(n) = 2T(n/2) + n^2$

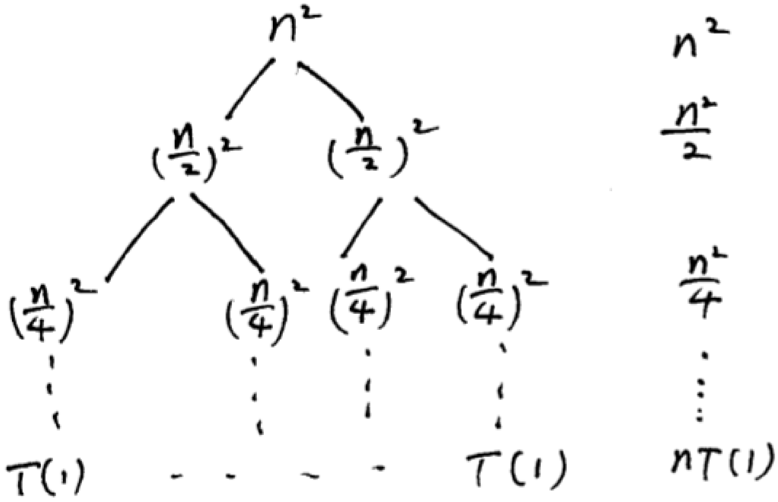
$$T(n) = 2T(n/2) + n^2$$



Since $T(n/2) = 2T(n/4) + (\frac{n}{2})^2$
the above tree is



repeat



Thus
$T(n) = (n^2 + \frac{n^2}{2} + \frac{n^2}{4} + ...) + nT(1) = n^2(1 + \frac{1}{2} + \frac{1}{4} + ...) + nT(1) \leq 2n^2 + n\Theta(1) = 2n^2 + \Theta(n) \in O(n^2)$

Exercise Use recursion tree to find the solution for $T(n) = 2T(n/2) + n$.