Lecture 8: More recurrence tree / Master theorem (4.5)

CS303: Algorithms

Last update: January 26, 2014

1 Review

- Recurrence tree method: intuitive but less strict
- Recurrence tree approach is usually used before officially proving the efficiency using substitution method.

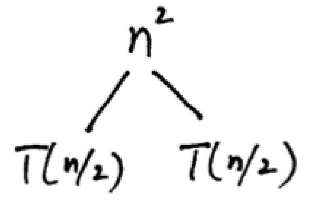
2 More examples

2.1 Proof the effiency of $T(n) = T(n-1) + 1$ follows $O(n)$ using substitution method

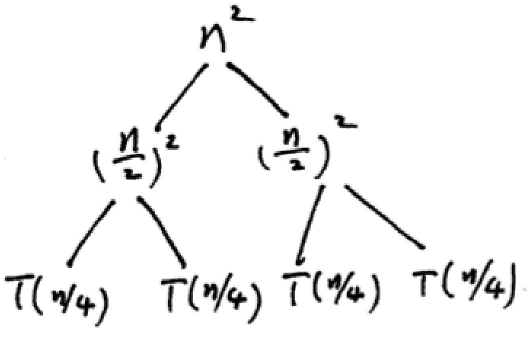
Guess:$T(n)$ is $O(n)$ because we did a recurrence tree last time on this $T(n)$
Proof:
if $T(k) \leq ck$ for all $k \leq n$, then $T(n) = T(n-1) + 1 \leq c(k-1) + 1 = ck - (c-1)$. Thus if I pick a c as 2, $T(n) \leq ck$. Thus induction is proved.
base case: $T(1)$ is $\Theta(1) < c$. So c has to be large enough (bigger than the time to process the T(1) trivial time)

2.2 Find the efficiency of $T(n) = T(n/2) + T(n/4) + n^2$

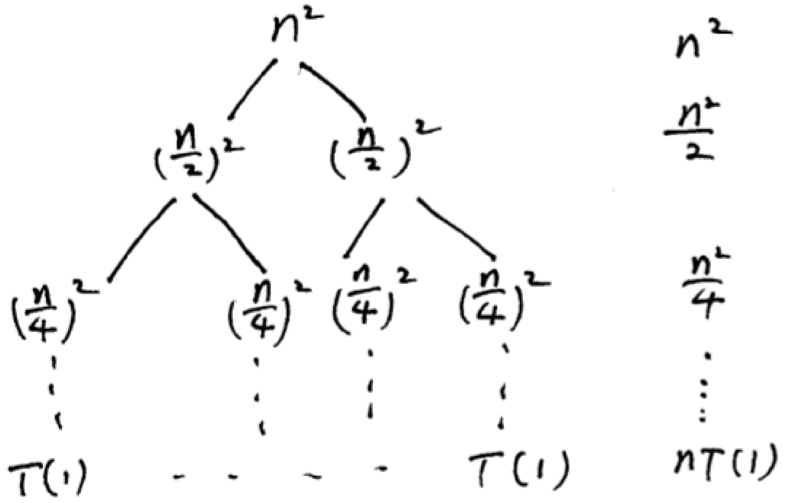
$$T(n) = 2T(n/2) + n^2$$



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



repeat



3 Master Theorem

This theorem can be proved and it is available in the book. We will cover the ideas of proof here. The good part of the theorem is it is very straightforward. The con is it is restrictive, only good for some special form of recurrence.
If $T(n) = aT(n/b) + f(n)$ where we divide the original job into a jobs of size n/b. Requirements

1. $a >= 1, b > 1$
2. $f(n)$ should be asymptotically positive (i.e. $f(n)$ is positive when n is large)

Idea: compare $f(n)$ with $n^{log_b(a)}$

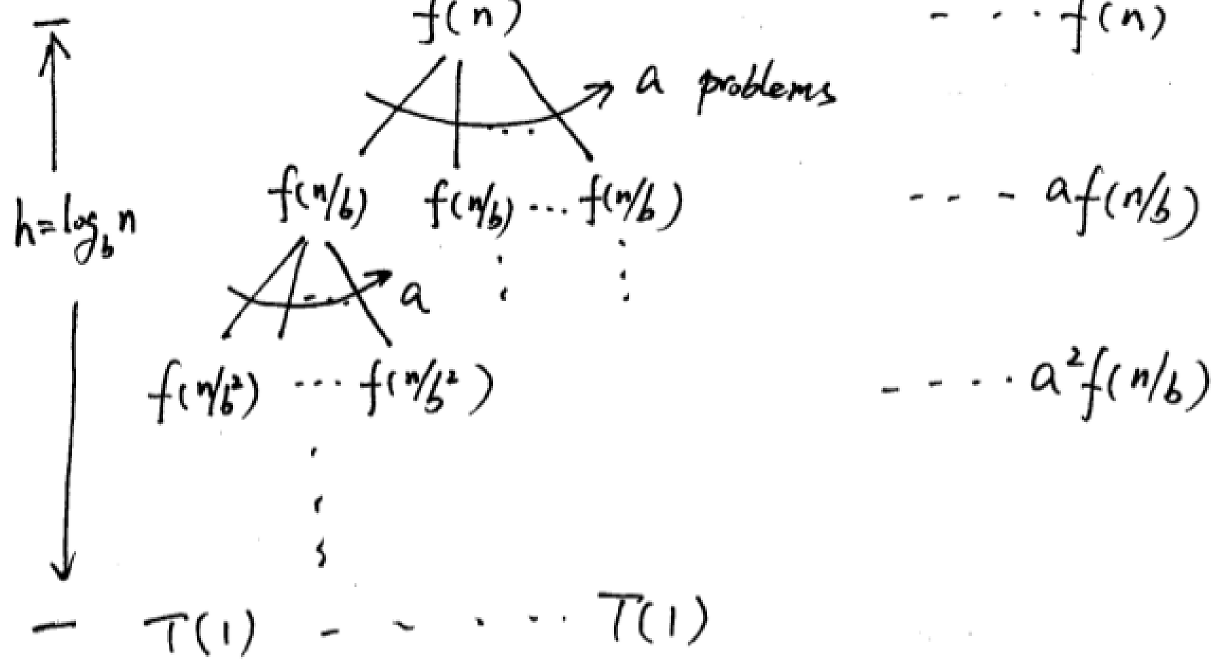
- Case 1: when $f(n)$ is $O(n^{log_b(a)-\epsilon}$ for some $\epsilon > 0)$ (i.e. when $f(n)$ is polynomially smaller than $n^{log_b(a)}$) Then $T(n) = \Theta(n^{log_b(a)})$
- Case 2: if $f(n) = \Theta(n^{log_b(a)} lg^k n)$ for some $k \geq 0$ (i.e. when $f(n)$ is the same as $\Theta(n^{log_b(a)})$ up to poly-log factors) Then $T(n) = \Theta(n^{log_b(a)} lg^{(k+1)} n)$
- Case 3: if $f(n) = \Omega(n^{log_b(a)+\epsilon})$ for some $\epsilon > 0$ and $af(n/b) < cf(n)$ where $c < 1$. (i.e. work load gets smaller when we go down the tree) Then $T(n) = \Theta(f(n))$

3.1 Examples

1. $T(n) = 4T(n/2) + n$
   $a = 4, b = 2, f(n) = n$
   $n^{log_b(a)} = n^2$, then $T(n) = \Theta(n^2)$
2. $T(n) = 4T(n/2) + n^2$
   $a = 4, b = 2, f(n) = n^2$
   Then $T(n) = \Theta(n^2 lg(n))$
3. $T(n) = 4T(n/2) + n^3$
   $a = 4, b = 2, f(n) = n^3$
   Then $T(n) = \Theta(n^3)$
4. $T(n) = 4T(n/2) + n^2/lgn$
   master theorem doesn't apply since $k < 0$

3.2 Proof of mater theorem

$$T(n) = a\,T(n/b) + f(n)$$



Height of the tree $h = \log_b n$

# of leaves : $a^h = a^{\log_b n} = n^{\log_b a}$

For case 3: the cost decrease geometrically and the cost is baiscally a constant times $f(n)$

For case 1: the cost increases thus the big one dominates

For case 2: the cost are pretty much the same as we go down the tree. Thus the cost is $f(n) \times height$.