CS 570: Analysis of Algorithms – H4

Submitted by: Indronil Bhattacharjee

Exercise 4.3-6

Choose n1 such that $n \geq n1$ implies $\frac{n}{2} + 17 \leq \frac{3n}{4}$.

We'll find c and d such that $T(n) \leq cn \log n - d$.

$T(n) = 2T(\lfloor \frac{n}{2} \rfloor + 17) + n$

$\leq 2(c(\frac{n}{2} + 17) \lg (\frac{n}{2} + 17) - d) + n$

$\leq cn \lg(\frac{n}{2} + 17) + 17c \lg(\frac{n}{2} + 17) - 2d + n$

$\leq cn \lg(\frac{3n}{4}) + 17c \lg(\frac{3n}{4}) - 2d + n$

$= cn \lg n - d + cn \lg(\frac{3n}{4}) + 17c \lg(\frac{3n}{4}) - d + n.$

Taking $c = -2/\lg(\frac{3n}{4})$ and $d = 34$.

Then we have $T(n) \leq cn \lg n - d + 17c \lg(n) - n$.

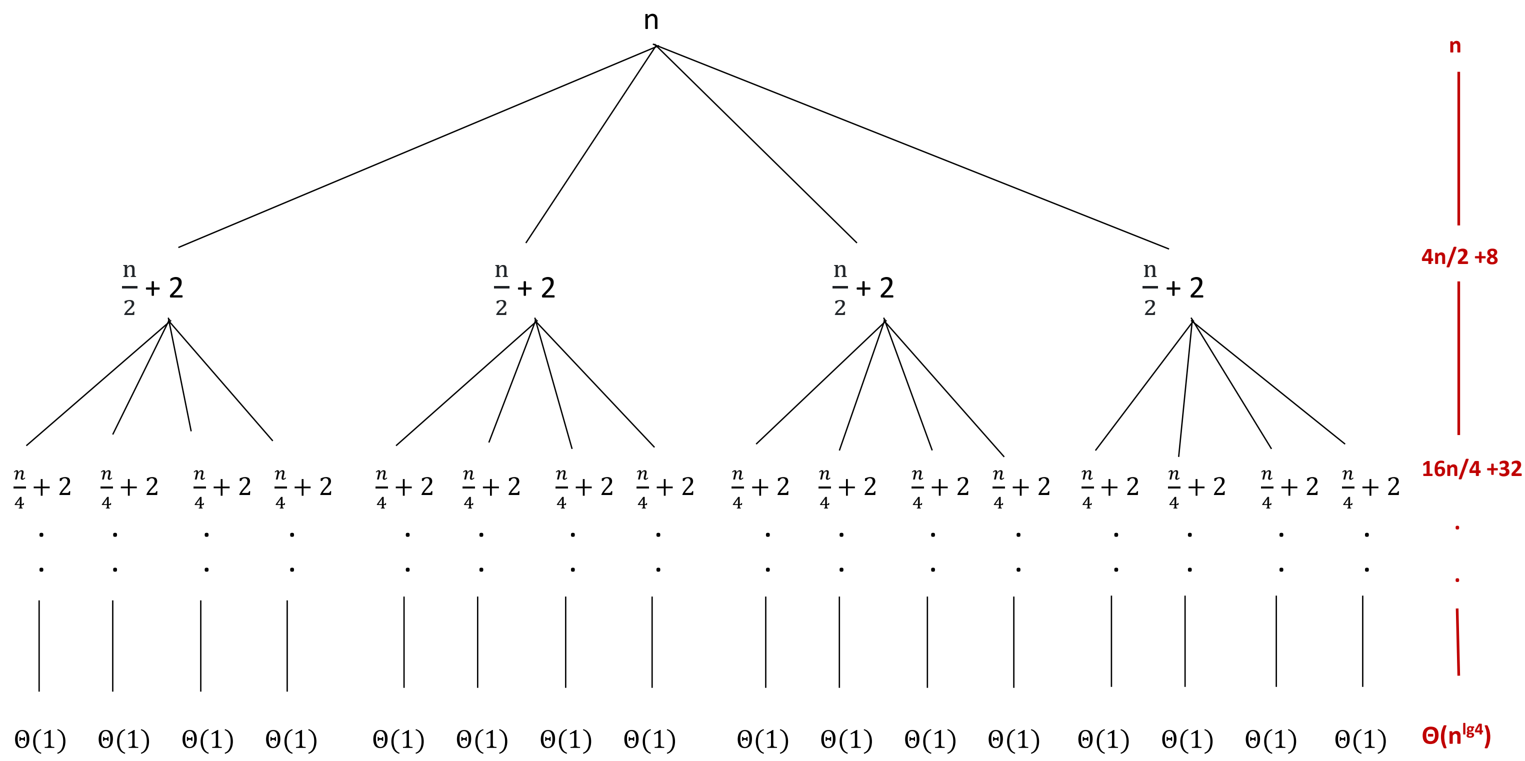
Since $\lg(n) = O(n)$, there exists n2 such that $n \geq n2$ implies $n \geq 17c \lg(n)$.

Letting $n0 = \max\{n1, n2\}$ we have that $n \geq n0$ implies $T(n) \leq cn \lg n - d$.

Therefore $T(n) = O(n \lg n)$.

Exercise 4.4-3

The recurrence $T(n)=4T(\frac{n}{2} + 2) + n$ has the following recursion tree:



Adding up the costs of each level of the tree:

$$T(n) = n + \frac{4n}{2} + 8 + \frac{16n}{4} + 32 + \cdots + \Theta(n^{\lg 4})$$

$$= \sum_{i=0}^{\lg(n-1)} 2^i n + 2\sum_{i=0}^{\lg(n-1)} 4^i + \Theta(n^2)$$

$$= \frac{2^{\lg n}-1}{2-1} n + 2\frac{4^{\lg n}-1}{4-1} + \Theta(n^2)$$

$$= (2^{\lg n} - 1)n + \frac{2}{3}(4^{\lg n} - 1) + \Theta(n^2)$$

$$= (n-1)n + \frac{2}{3}(n^2 - 1) + \Theta(n^2)$$

$$= \Theta(n^2)$$

Based on this calculation, we assume that $T(n) \leq cn^2 - bn$. Substituting this into the recurrence,

$$T(n) \leq 4(c\left(\frac{n}{2} + 2\right)^2 + b\left(\frac{n}{2} + 2\right) + n$$

$$= 4\left(\frac{cn^2}{4} + \frac{4cn}{2} + 4c + \frac{bn}{2} - 2b\right) + n$$

$$= cn^2 + 8cn + 16c - 2bn - 8b + n$$

$$= cn^2 + bn + 8cn - bn + 16c - 8b + n$$

$$= cn^2 + bn - (b - 8c - 1)n - 8(b - 2c)$$

$$\leq cn^2 + bn \text{ [where } b - 8c - 1 \geq 0]$$

Exercise 4.5-4

In the given recurrence, a=4 and b=2.

Hence, $n^{\log_b a} = n^{\log_2 4} = n^2$ and $f(n) = \Theta(n^2 \lg n)$

Now, asymptotically $f(n)=n^2\lg n$ is definitely larger than n^2, but it is not polynomially larger than n^2. So, we cannot apply the master theorem method to this recurrence.

We can use recursion tree to get a good estimate of the asymptotic upper bound of the given reference and then use substitution method to prove that.

Rate of increase in number of subproblems in each recursion = 4

Rate of decrease in subproblem size = 2

Hence in each level of the tree, there are 4i nodes each of cost $c\left(\left(\frac{n}{2^i}\right)2 \cdot lg\left(\frac{n}{2^i}\right)\right)$ at depth i=0, 1, 2, …, lgn.

Hence, total cost of the tree is:

$$T(n) = \sum_{i=0}^{\lg n} 4^i + c\left(\left(\frac{n}{2^i}\right)^2 . \lg\frac{n}{2^i}\right)$$

$$= \sum_{i=0}^{\lg n} 4^i + c\left(\frac{n^2(lg\, n - lg\, 2^i}{2^{2i}}\right)$$

$$= cn^2 \sum_{i=0}^{\lg n} (\lg n - lg\, 2^i)$$

$= cn^2(\sum_{i=0}^{\lg n} \lg n - \sum_{i=0}^{\lg n} lg\, 2^i)$

$= cn^2(\sum_{i=0}^{\lg n} \lg n - \sum_{i=0}^{\lg n} i)$

$= cn^2(\lg n \,.\lg n - \frac{\lg n(\lg n+1)}{2})$

$= cn^2(\frac{(\lg n)^2}{2} - \frac{\lg n}{2})$

$= cn^2.\frac{(\lg n)^2}{2} - cn^2.\frac{\lg n}{2}$

$\leq cn^2.\frac{(\lg n)^2}{2}$

$\leq cn^2.\lg^2 n$

Problem 4.2

Binary search

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

   Now a=1, b=2 and f(n)=1.

   We can use case 2 of the master method because $f(n) = \Theta(n^{\log_2 1}) = \Theta(0)$ and thus the solution to the recurrence is $T(n) =((n^{\log_2 1}) \lg n) = \Theta(\lg n)$

   (By master method)

2. $T(n) \;\; = T(n/2) + cN$

   $= 2cN+T(n/4)$

   $= 3cN+T(n/8)$

   $= \sum_{i=0}^{\lg (n-1)} \frac{2^i cN}{2^i}$

   $= cN \lg n$

   $= \Theta\ (n \lg n)$

3. $T(n) = 2T(n/2) + \Theta(n)$

   Now a=1, b=2 and f(n)=n.

   We can use case 3 of the master method because $f(n) = \Omega(n^{\log_2 1 + \epsilon})$, ϵ =1 >0 and

$$a\,f(n/b) \le cf(n)$$

$$(n/2) \le cn, \text{ holds for } \frac{1}{2} \le c < 1$$

and thus the solution to the recurrence is $T(n) = \Theta(f(n)) = \Theta(n)$

(By master method)

Merge sort

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

   Now a=2, b=2 and f(n)=1.

   We can use case 2 of the master method because $f(n) = \Theta(n^{\log_2 2}) = \Theta(1)$

   and thus the solution to the recurrence is $T(n) = \Theta((n^{\log_2 2}) \lg n) = \Theta(n \lg n)$

   (By master method)

1. $T(n) = 2T(n/2) + cn + 2N$

$$= 4N + cn + 2c(n/2) + 4T(n/4)$$

$$= 8N + 2cn + 4c(n/4) + 8T(n/8)$$

$$= \sum_{i=0}^{\lg(n-1)} (cn + 2^i N)$$

$$= \sum_{i=0}^{\lg(n-1)} cn + N \sum_{i=0}^{\lg(n-1)} 2^i$$

$$= cn \lg n + N \frac{2^{\lg n} - 1}{2 - 1}$$

$$= cn \lg n + nN - N$$

$$= \Theta(nN)$$

$$= \Theta(n^2)$$

2. $T(n) = 2T(n/2) + cn + 2n/2$

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

$$= \Theta(n \lg n) \text{ (Master theorem method)}$$