

CS 381: Assignment #1

Due on Thursday, September 11th, 2014

Prof. Grigorescu 12:00pm

Yao Xiao(xiao67)

Problem 1

$(\log n)^{0.3}$, $(\log n)^6$, \sqrt{n} , $\{n \log n, \log n^n\}$, $n(\log n)^4$, n^2 , n^{21} , 2^n , $n!$

Problem 2

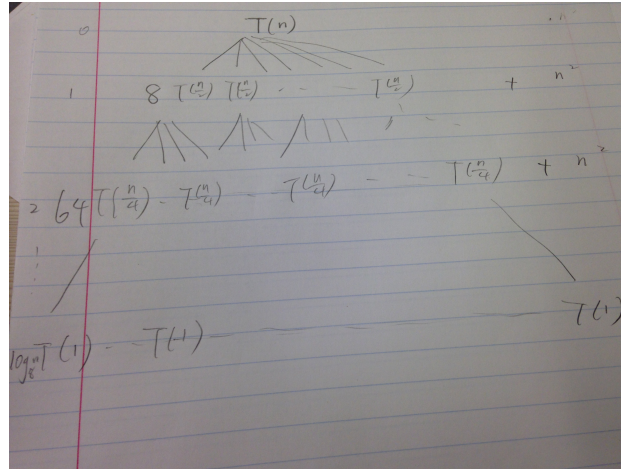


Figure 1. Recursion Tree

What we can get from Figure 1 is :

$$\begin{aligned}
 T(n) &= 8T\left(\frac{n}{2}\right) + n^2 && \text{Level 0} \\
 8T\left(\frac{n}{2}\right) &= 64T\left(\frac{n}{4}\right) + n^2 && \text{Level 1} \\
 64T\left(\frac{n}{4}\right) &= 512T\left(\frac{n}{8}\right) + n^2 && \text{Level 2} \\
 &\dots && \\
 T(n) &= 8^{\log_2 n} T(1) + \log_2 n n^2
 \end{aligned}$$

So we can conclude that $T(n) = \Theta(n^3)$

From Master theorem: $T(n) = \Theta(n^{\log_2 8}) = \Theta(n^3)$ since $a = 8$ $b = 2$ $f(n) = n^2$ where $c = 2$ $c < \log_b a$

Proof:

Basis: $T(1) = 1$; $T(1) = 1^3 + \log_2 1 * 1 = 1 + 0 = 1$

Induction: Assume $T(n) = n^3 + \log_2 n * n^2$ is true

From the original definition:

$$\begin{aligned}
 T(n+1) &= 8\left(T\left(\frac{n+1}{2}\right)\right) + (n+1)^2 \\
 &= 8\left[\left(\frac{n+1}{2}\right)^3 + \log_2 \frac{n+1}{2} * \left(\frac{n+1}{2}\right)^2\right] + (n+1)^2 \\
 &= 8(n+1)^3 + 2\log_2 \frac{n+1}{2} * (n+1)^2 + (n+1)^2 \\
 &\dots \text{Some Magic} \dots \\
 &= (n+1)^3 + \log_2(n+1) * (n+1)^2
 \end{aligned}$$

So by the basis and induction. We conclude that $T(n) = n^3 + \log_2 n * n^2$

Problem 3

Apparently $F(n) = F(n-1) + F(n-2)$; $F(0) = 1$; $F(1) = 1$; $F(2) = 2$

So We can get

$F(0)=1$

$F(1)=1$

$F(2)=2$

$F(3)=3$

$F(4)=5$

$F(5)=8$

It Fibonacci sequence.

We keep going on(I wrote a small program) And get the result $F(12)=233$

Problem 4

1. If I can create a boolean array from [minnum,maxnum]

For i from first number to last number
if $arr[i+k]=true$ or $arr[i-k]=true$
the element exists.
print it out && break

If not exists, print out not exists

The complexity is $O(n)$

2. If I can not create an array.

MergeSort the array
For i from first number to last number
Do binary search for $i+d$ and $i-d$
If not exists, print out not exists

The complexity is $O(n \log n)$

Problem 5

1. Scan the array, find the minimum one, set the element as ∞ . Repeat it 3 times.
2. Use $O(n \log n)$ sorting method. return the \sqrt{n} element.
3. Create a Min-Heap , $O(n)$, Delete a node every time, $O(\sqrt{n})$. So the total complexity is $O(N)$