2/10/19

HomeWork # 2

1. Solve using masters method

a) $T(n) = 2T(n/4) + 7$

Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, let $T(n)$ be

$$T(n) = aT(n/b) + f(n)$$

1. $f(n) = O(n^{\log_b a - \epsilon})$ for $\epsilon > 0$, then
$T(n) = \Theta(n^{\log_b a})$
2. $f(n) = \Theta(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a} \lg n)$
3. $f(n) = \Omega(n^{\log_b a + \epsilon})$ for $\epsilon > 0$, $af(n/b) \leq cf(n)$ for $c < 1$, then $T(n) = \Theta(f(n))$

a) $T(n) = 2T(n/4) + 7$

$a = 2, b = 4, f(n) = 7$

$\log_b a = \log_4 2 = 0.5 = O(n^{\log_4 2 - \epsilon})$

$\exists\ \epsilon > 0$ which is 0.5

Then $\boxed{T(n) = \Theta(n^{\log_4 2}) = \Theta(n^{0.5})}$

b) $T(n) = 3T(n/9) + \sqrt{n}$

$a = 3, b = 9, f(n) = \sqrt{n} = n^{0.5}$

$\log_b a = \log_9 3 = 1/2$.

$f(n) = \Theta(n^{\log_9 3})$.

$\therefore \boxed{T(n) = \Theta(n^{\log_9 3} \log n) = \Theta(\sqrt{n} \log n)}$

c) $T(n) = 2T(n/4) + n \lg n$

$a = 2, b = 4,$ ~~k~~, $\log_b a = \log_4 2 = 0.5$

case 3: $f(n) = \Omega(n^{\log_4 2 + \epsilon}) = \Omega(n^{0.5+\epsilon})$

$\boxed{T(n) = \Theta(n \lg n)}$ $\therefore T(n) = f(n)$

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

$a = 4, b = 2, \log_2 4 = 2$.

case 1: $f(n) = n, \Rightarrow O(n^{\log_b a - \epsilon})$

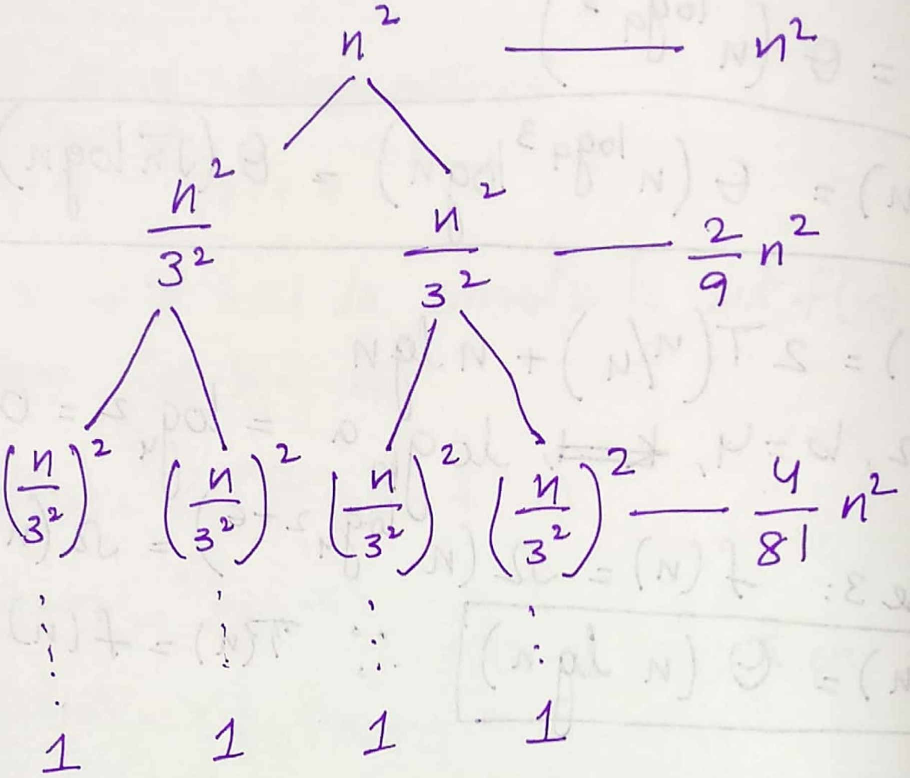
$= O(n^{\log_2 4 - \epsilon})$

$\therefore T(n) = \Theta(\cancel{n \log}\ n^{\log_b a})$

$= \Theta(n^{\log_2 4}) = \Theta(n^2)$

$\boxed{T(n) = \Theta(n^2)}$

2. $T(n) = 2T(n/3) + n^2$



$$T(n) = n^2 + \frac{2}{9}n^2 + \frac{4}{81}n^2 + \dots$$

$$= n^2\left(1 + \frac{2}{9} + \frac{4}{81} + \dots\right)$$

$$= cn^2$$

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

3. Given $l = 0$, $r = n-1$

$\therefore$ substituting in ALG1,

ALG1 (A, 0, n−1)

if $n-1 \leq 10$ —— $T(1)$

$s = 0$, $n = r - l + 1$ —— $T(1)$

for $i = 0$ to $n-1$ —— $T(n)$

$S += A[i]$ —— n

$S += ALG1\left(A, 0, \frac{2n}{3} - 1\right) \quad — \quad T\left(\frac{n}{3} - n - 1\right)$

$S += ALG1\left(A, n/3, n-1\right) \quad — \quad T\left(n - 1 - \frac{n}{3}\right)$

return S

$$T(n) = T\left(\frac{2n}{3} - 1\right) + T\left(\frac{n-1}{3} - \frac{n}{3}\right) + n$$

$$= T\left(\frac{2n}{3}\right) + T\left(\frac{2n}{3}\right) + n$$

$$T(n) = 2T\left(\frac{2n}{3}\right) + n$$

$a = 2,\ b = 3/2\ \ f(n) = n$

~~4. Bad Pair Algori~~

$4 > (3/2)^1 \quad \log_b a = \log_{3/2} 2$

$\Rightarrow f(n) = O(n^{\log_{3/2} 2 - \epsilon})$

$\therefore$ case 1: $T(n) = \Theta(n^{\log_b a})$

$$\boxed{T(n) = \Theta(n^{\log_{3/2} 2})}$$

4. Bad-Pair Algorithm:

This algorithm is done in the concept of merge sort. Divide the array into two halves recursively and compute the bad pairs in the merge functions. Since merge sort time complexity is $O(n \log n)$, the bad pair also has $O(n \log n)$

Scanned by CamScanner

~~Merge~~ Declare global variable count

```
BadPair (A, l, r)
{
  if l < r
  {
    m = l + (r-1)/2;

    BadPair (A, l, m);
    BadPair (A, m+1, r);

    Compute (A, l, m, r);
  }
}

Compute (A, l, m, r)
{
    n1 = m - l + 1;
    n2 = r - m;
   let L [0... n1] and R [0 ... n2] be new arrays
   for 0 to n1
     L[i] = A[l + i];

   for 0 to n2
     R[j] = A [m+1+j]

   i=0, j=0, k=l;
   while  i < n1 and j < n2
      if (L[i] > R[j] + 10) count++.
```

if j+1 < n2
    j++
else if j+1 == n2
    j=0; i++
}
}

5. a) Dividing the input elements into groups of 3 each, we can compute the elements that are greater than m^* and smaller than m^*. The size of sub problem is

$$2\left(\left\lceil \frac{1}{2}\left\lceil \frac{n}{3}\right\rceil\right\rceil - 2\right)$$

$$\therefore n - \left(\frac{n}{3} - 4\right) = \frac{2n}{3} + 4$$

Worst case running time of SELECT becomes

$$T(n) \leq T(\lceil n/3 \rceil) + T\left(\frac{2n}{3} + 4\right) + O(n)$$

$$T(n) \leq c\lceil n/3 \rceil + c\left(\frac{2n}{3} + 4\right) + an$$

$$\leq \frac{cn}{3} + c + \frac{2cn}{3} + 4c + an$$

$$\leq \frac{3cn}{3} + 5c + an$$

$$\leq cn + 5c + an$$

$$\therefore 5c + an \leq 0$$

b) To prove, $T(n) = O(n \lg n)$

$$T(n) = c\left(\frac{n}{3}\right) \lg\left(n/3\right) + c\left(\frac{2n}{3}\right) \log\left(\frac{2n}{3}\right) + O(n)$$

$$n \lg n \leq \frac{cn}{3}\left[\lg n/3 + 2 \lg \frac{2n}{3}\right] + n$$

$$\lg n \leq \frac{c}{3}\left[\lg n - \lg 3 + 2\lg 2n - 2\lg 3\right] + 1$$

$\therefore$ The runtime of SELECT would be $O(n \lg n)$.

Also, $\Omega(n \lg n)$ is the lower bound because they manage to solve the selection problem without sorting. With sorting it requires $\Omega(n \lg n)$ time in the comparison model, even on average.