CSE 321 HOMEWORK 2

Emre Sezer
1901042640

1) $X(n) \in \begin{cases} \Theta(n^d) & \text{if } a < b^d \\ \Theta(n^d \log n) & \text{if } a = b^d \\ \Theta(n^{\log_b a}) & \text{if } a > b^d \end{cases}$ for all n

$X(n) = aX(\frac{n}{b}) + f(n)$; $n = b^k$ $(k = 1, 2, \dots)$

If $f(n) \in \Theta(n^d)$ where $d \geq 0$

$a \geq 1$, $b > 1$

a) $T(n) = 16T(\frac{n}{4}) + n!$

$a = 16$, $b = 4$, $f(n) = n! = \Theta(n \lg n)$

for $\forall n$ $n^1 > \log n$, $\log n \in o(n)$

$1 < d < 2$

$a = 16$
$b = 4$
$a = b^2$

$16 = 4^2 > 4^d$

Since, d is smaller than 2 for every n, this inequality is true.

$T(n) \in \Theta(n^2)$ [Case 3]

$a \geq 1$, $b > 1$, $d \geq 0$

b) $T(n) = \sqrt{2}\,T(\frac{n}{4}) + \log n$

$a = \sqrt{2} = 2^{\frac{1}{2}}$, $b = 4$, $f(n) = \log n$

$d < 0{,}01$, $\log n < n^1$

d is so small that i assume $d = 0$.

$\sqrt{2} > 4^d$, $\sqrt{2} > 4^0 = 1$

$T(n) \in \Theta(n^{\log_4 \sqrt{2}}) = \Theta(n^{\frac{1}{4}})$

$a \geq 1$, $b > 1$, $f(n) \in \Theta(\log n)$

[Case 3] $d \geq 0$

c) $T(n) = 8T(\frac{n}{2}) + 4n^3$

$a = 8$, $b = 2$, $f(n) = 4n^3$

$f(n) \in \Theta(n^3)$, $d = 3$

$8 = 2^3$, $a = b^d$

$a \geq 1$, $b > 1$, $d \geq 0$

[Case 2]

$T(n) \in \Theta(n^3 . \log n)$

d) $T(n) = 64T(\frac{n}{8}) - n^2 \log n$ $a=64, b=8, f(n) = -n^2 \log n$

$f(n)$ is a negative function, So, it is not non-decreasing.

$\lim_{n \to \infty} f(n) = -\infty$. Master Theorem can be applied to non-decreasing functions. Because of that this problem cannot be solved by using Master theorem.

---

e) $T(n) = 3T(\frac{n}{3}) + \sqrt{n}$ $a=3, b=3, f(n) = \sqrt{n} \in \Theta(\sqrt{n})$

$n^d = n^{\frac{1}{2}}, d = \frac{1}{2}$

$a \geq 1, b > 1, d \geq 0$

$3 > 3^{\frac{1}{2}}, 3 > \sqrt{3}$

$T(n) \in \Theta(n^{\log_3 3}) = \Theta(n^1) = \Theta(n)$ [Case 3]

---

f) $T(n) = 2^n T(\frac{n}{2}) - n^n$ $f(n) = -n^n$

Answer is the same as part d).

This problem cannot be solved by using Master theorem.

---

g) $T(n) = 3T(\frac{n}{3}) + \frac{n}{\log n}$ $a=3, b=3, f(n) = \frac{n}{\log n}$

$f(n) \in \Theta(n^d \log^p n)$ $a \geq 1, b > 1$

$f(n) \in \Theta(n^1 \log^{-1} n)$ $d \geq 0$

$\log_b a = \log_3 3 = 1$

$d = 1, p = -1$

From: if $\log_b a$

if $p > -1$ $\Theta(n^d \log^{p+1} n)$

if $p = -1$ $\Theta(n^d \log \log n)$

if $p < -1$ $\Theta(n^d)$

$T(n) \in \Theta(n \log \log n)$

[Case 2]

2) $T(n) = aT(\frac{n}{b}) + f(n)$

a) $X(n) = 9X(\frac{n}{3}) + \Theta(n^2)$ $a=9, b=3, f(n) \in O(n^2)$

$n^d = n^2$, $d=2$

$9 = 3^2$, $a = b^d$ $a \geq 1, b > 1, d \geq 0$

$X(n) \in \Theta(n^d \log n) = \Theta(n^2 \log n)$ [Case 2]

b) $Y(n) = 8T(\frac{n}{2}) + f(n)$ $a=8, b=2, f(n) \in O(n^3)$

$n^d = n^3$, $d=3$

$8 = 2^3$, $a = b^d$ $a \geq 1, b > 1, d \geq 0$

$X(n) \in \Theta(n^d \log n) = \Theta(n^3 \log n)$ [Case 2]

c) $Z(n) = 2T(\frac{n}{4}) + f(n)$ $a=2, b=4, f(n) \in O(\sqrt{n})$

$n^d = n^{\frac{1}{2}}$, $d = \frac{1}{2}$

$2 = 4^{\frac{1}{2}} = b^d$ $a \geq 1, b > 1, d \geq 0$

$Z(n) \in \Theta(n^d \log n) = \Theta(\sqrt{n} \log n)$ [Case 2]

$X(n) \in O(n^2 \log n)$, $Y(n) \in O(n^3 \log n)$, $Z \in O(\sqrt{n} \log n)$

for every $n \geq 0$ $Y(n) = n^3 \log n > X(n) = n^2 \log n > Z(n) = \sqrt{n} \log n$

I would choose $Z(n)$. Because, it's time complexity is smaller which would be more efficent.

3) a)

i)

Time complexity of merge sort for worst, best and average cases are same and $\Theta(n\log n)$. That means, maximum number of comparisons wouldn't change for any 8-element array. For every n sized array, there will be n time comparisons for best, worst and average cases.

ii)

Explanation for this part is same as part i). That means, minimum number of comparisons wouldn't change for any 8-element array. For evey n sized array, there will be n times comparisons for best, worst and average cases.

b)

i) arr = [4,5,6,7,8,3,2,1] pivot is the first element

$\boxed{4}$ 5 6 7 8 3 2 1

↓ after swaps

3 2 1 | 4 | 8 7 6 5

$\boxed{3}$ 2 1 $\boxed{8}$ 7 6 5

1 2 3 5 7 6 | 8

5 | $\boxed{7}$ 6 | 8

1 2 3 4 5 6 7 8

I put middle element to the first place. Then, 5 6 7 8 3 2 1. At the first step all swaps will happen and array will look like this: 1 2 3 | 4 | 8 7 6 5. Right side of the 4 is reverse sorted. So, there has to be more swaps. With that approach there will be max amounts of swap operations.

ii) arr = [1,2,3,4,5,6,7,8]   pivot is the last element.

1 2 3 4 5 6 7 $\boxed{8}$ → 8

1 2 3 4 5 6 $\boxed{7}$ → 7 8

1 2 3 4 5 $\boxed{6}$ → 6 7 8

⋮

1 → 1 2 3 4 5 6 7 8

At each step left pointer will be out of bounds and will point pivot's position. There will be no swops needed. Array will be swapped with 0 swops if i pick arr above and assume pivot is the last element of the array.

4) algorithm (left, right)

```
mid = (left + right) / 2  ⎫
if A[mid] == 0            ⎬ Θ(1)
    return mid            ⎭
else
  if A[mid] > 0  } Θ(1)
    right = mid
    algorithm(left, right) } T(n/2)
  else
    left = mid  } Θ(1)
    algorithm(left, right) } T(n/2)
```

I apply Master Theorem.

$$T(n) = 2T\left(\frac{n}{2}\right) + f(n) \quad , f(n) \in \Theta(1)$$

$a=2$, $b=2$

$n^d = 1$, $d=0$

$a \geq 1$, $b > 1$, $d \geq 0$

$$T(n) \in \Theta\left(n^{\log_b a}\right) = \Theta\left(n^{\log_2 2}\right) = \Theta(n)$$

[Case 1]

Scanned with CamScanner

5)

I suppose there are 2 arrays named Gifts and Boxes with equal length.

a) 1. Select a random element x from Boxes array.

2. Use x as an input of rearrange function which will be partitioning the Gifts array for this step. It will compare each element in Gifts with x. Basically x will act like a pivot.

3. 2. Step will return y. y will act like a pivot. Use y as an input of rearrange function which will be partitioning the Boxes array for this step. It will compare each element in Boxes with y.

4. If length of partitioned sub-arrays are smaller or equal to 1 finish the program. Else, Jump to step 1.

b) $T(n) = 2T(\frac{n}{2}) + f(n)$ $\quad a=2, b=2, f(n) \in \Theta(n)$

$n^d = n, d=1$

$a \geq 1, b > 1, d \geq 0$

$2^1 = 2, \ b^d = a$

$T(n) \in \Theta(n \log n)$ [Case 2]