CSE221 Assignment 1

Mohammad Ariful Islam

Sec: 02

ID: 20101192

(1) (a) ~~log(logx), logx, x!~~

$\log(\log n)$, $\log n$, $\sqrt{n}$, ~~nlogn~~ n, $n\log n$, $n^{\frac{3}{2}}$, $n^2\log n$, n^2, 2^n, $n!$, n^3, e^{n+1}

(b) (i) $n^2 + 15n - 3 = \Theta(n^2)$

if $c_1 = 1$ and $n_0 = 1$

$c_1 n^2 \leq n^2 + 15n - 3$ for all $n \geq 1$

if $c_2 = 3$ and $n_0 = 10$

$c_2 n^2 \geq n^2 + 15n - 3$ for all $n \geq 10$

$\therefore$ True

(ii) $4n^3 - 7n^2 + 15n - 3 = \Theta(n^3)$

if $c_1 = 3$ and $n_0 = 1$

$c_1 n^3 \leq 4n^3 - 7n^2 + 15n - 3$ for all $n \geq 1$

if $c_2 = 5$ and $n_0 = 5$

$c_2 n^3 \geq 4n^3 - 7n^2 + 15n - 3$ for all $n \geq 5$

(iii) $T(n) = 4T(\frac{n}{2}) + n = \Theta(n^2)$

First, $T(n) = 4T(\frac{n}{2}) + n$

Using Master theorem, $T(n) = aT(\frac{n}{b}) + cn^k$, $T(1) = c$

$a = 4$, $b = 2$, $k = 1$, $c = 1$, $T(1) = 1$

$\Rightarrow b^k = 2^1 < a$ $\therefore$ $T(n) = \cancel{\#}\ n^{\log_b a}$

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

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

if $c_1 = 1$ and $n \geq 0$;

$c_1 n^2 \leq n^2$

if $c_2 = 3$ and $n > 0$;

$c_2 n^2 \geq n^2$

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

(iv) $T(n) = 2T(\frac{n}{2}) + n^3$

$T(n) = aT(\frac{n}{b}) + cn^k$ $\therefore$ comparing, we get:

$a = 2$, $b = 2$, $k = 3$

$b^k = 2^3 > a$ $\therefore$ $T(n) = n^k = n^3$

(v) $T(n) = T(n/4) + T(5n/8) + n = O(n)$ $\Rightarrow T(n) = T(\frac{n}{4}) + \{T(n)\}_1$

$\{T(n)\}_1 = T(\frac{n}{8/5}) + n$

$a = 1, \quad b = \frac{8}{5}, \quad k = 1$ Since $b^k > a$

$\therefore \{T(n)\}_1 = O(n)$

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

$a = 1, \; b = 4, \; k = 1$

Since $b^k > a \; (4 > 1)$

$T(n) = O(n)$

(vi) $T(n) = T(\frac{n}{3}) + T(\frac{n}{9/4}) + n = O(n)$

$T(n) = T(\frac{n}{3}) + \{T(n)\}_1$

$\{T(n)\}_1 = T(\frac{n}{9/4}) + n$

$a = 1, \quad b = \frac{9}{4}, \quad k = 1$

$b^k > a \quad (\frac{9}{4} > 1)$ $\therefore \{T(n)\}_1 = O(n)$

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

$a = 1, \quad b = 3, \quad k = 1$

$3^1 > 1 \quad (b^k > a)$

$\therefore \quad T(n) = O(n)$

(c)(1)

```
count = 0 → O(1)
for(i=1, i<=n, i*=2)
    for (j=1, j<=i; j++)
        count++; → O(1)
```

for the first loop:

Step	i		j
0	1	$= 2^0$	1
1	2	$= 2^1$	2
2	4	$= 2^2$	4
3	8	$= 2^3$	8
4	16	$= 2^4$	16
⋮	⋮		
k	2^k		2^k

$$\frac{2^k}{n} = 1 \Rightarrow 2^k = n$$

$$k = \log_2 n$$

$\therefore \cancel{\{T(n)\}_2} = \theta(\log_2 n) \leftarrow$ for first loop

$\cancel{\{T(n)\}_1} = \theta(\log_2 n * O(1))$

$= O(\log_2 n)$

$\cancel{T(n) =}\ O(1) + O\{\log_2 n\ (\log_2 n)\}$

$= \theta\{\log_2 n \cdot \log_2 n\}$

$\theta(\log_2 n . \log_2 n)$

(c)(2) $p = 3$ $\theta(1)$

while $(p < n)$

$p = p * p$

Step	p
0	3
1	9
2	81
3	6561
⋮	⋮
k	3^{2^k}

$$\frac{3^{2^k}}{n} = 1 \Rightarrow 3^{2^k} = n$$

$$2^k \log_3 3 = \log_3 n \Rightarrow 2^k = \log_3 n$$

$$\Rightarrow k \log_2 2 = \log_2(\log_3 n)$$

$$k = \log_2(\log_3 n)$$

$$\cancel{T(n)} = \theta\{\log_2(\log_3 n) + 1\}$$

Time Complexity $\longrightarrow$ $\theta(\log_2 n(\log_3 n))$

(d)(i) In a ternary search for n values:

n step 0

$\downarrow$

$\frac{n}{3}$ step 1

$\downarrow$

$\frac{n}{3^2}$ step 2

$\downarrow$

$\frac{n}{3^3}$ step 3

⋮

$\frac{n}{3^k}$ k

$\frac{n}{3^k} = 1$

We see that every step, the search area is becoming one-thirds of the initial value.

For ~~getting mid1 and mid2~~ the subproblems for each function call, the time complexity is $O(1)$

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

(b) $3^k = n$

$k = \log_3 n$

$\therefore$ Time complexity $= O(\log_3 n)$

(2)(a)

```
def binary-search(A, val, L, R):
    if L>R:
        return R+1

    M = (L+R)//2
    if val == A[M] and M+1 < len(A):
        if A[M+1] == val and M+1== len(A)-1:
            return len(A) - 1

        elif A[M+1] == val
            return binary-search(A, val, M+1, R)

        else:
            M+1
    elif val==A[M] and M== len(A)-1 :
        return M+1
    elif val > A[M]:
        return binary=search(A, val, M+1, R)

    else:
        return binary-search(A, val, L, M-1)

def noOfElements(size1, size 2, list1, list2):
    for i in list2:
        value= binary_search(list1, i, 0, size1- 1)
        print(num, end=" ")

    print()
```

(b) For the ~~method~~ function binary-search:

problems

$n \leftarrow$ step 0

$\frac{n}{2}$ step 1

$\frac{n}{4}$ step 2

$\vdots$

$\frac{n}{2^k}$ step k

$\frac{n}{2^k} = 1$ when it stops recurring

$n = 2^k$

$\log_2 n = k$

and since it is called inside a loop of size2 and if we consider size2 as n, then

$O(n \log_2 n)$