Theory Assignment – 1

Name: Maruf Morshed

ID: 20101299

Section: 5

Course: CSE221

Course Instructor: Shaily Roy

Ans no. 1(a)

Here,

$$\log(\log(n)) < \log(n) < \sqrt{n} < n < n.\log(n) < n^{3/2}$$

$$< n^2 < n^2.\log(n) < n^3 < 2^n < e^{n+1} < n!$$

(Ans.)

Ans no. 1(b)

i) Given,

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

$\therefore f(n) = n^2 + 15n - 3$ and $g(n) = n^2$

Here, in case of upper bound,

$n^2 + 15n - 3 \leq c \cdot n^2$

or, $n^2 \leq c \cdot n^2$

$\therefore$ For all values of $c \geq 1$ the inequality is True.

Again, for lower bound,

$n^2 + 15n - 3 \geq cn^2$

or, $n^2 \geq cn^2$

$\therefore$ For $c < 1$, the inequality is true.

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

[Proved]

ii) Given,

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

We get,

$f(n) = 4n^3 - 7n^2 + 15n - 3$ and $g(n) = n^3$

Now, in case of upper bound,

$4n^3 - 7n^2 + 15n - 3 \leq cn^3$

or, $4n^3 \leq cn^3$

$\therefore$ for $c \geq 4$, this inequality is true.

Again, in case of lower bound,

$4n^3 - 7n^2 + 15n - 3 \geq cn^3$

or, $4n^3 \geq cn^3$

$\therefore$ for $c < 4$ the inequality is true.

$\therefore$ $4n^3 - 7n^2 + 15n - 3 = \theta(n^3)$.

[proved]

III) Given,

$$T(n) = 4T\left(\frac{n}{2}\right) + n = \theta(n^2)$$

Comparing the following eqn with the master's theorem (divide and conquer), we get,

$$a = 4, \quad b = 2, \quad c = 1, \quad k = 1$$

$$\therefore b^k = 2^1 = 2$$

$$\therefore b^k < a \qquad [2 < 4]$$

$\therefore$ Time complexity will be, $T(n) = \theta(n^{\log_2 4})$

$$= \theta(n^2) \qquad [\log_2 4 = 2]$$

IV) Given,

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

Comparing the following eqn with the master's theorem, we get,

$$a = 2, \quad b = 2, \quad c = 1, \quad k = 3$$

Here,
$b^k = 2^3 = 8$

$\therefore b^k > a \quad [8 > 2]$

$\therefore$ Time complexity, $T(n) = \theta(n^k)$

$= \theta(n^3)$

$\therefore T(n) = 2T(\frac{n}{2}) + n^3 = \theta(n^3)$.

[Proved]

v) Given,
$T(n) = T(\frac{n}{4}) + T(\frac{5n}{8}) + n = O(n)$

Now, taking,
$T(n) = T(\frac{n}{8/5}) + n \quad \ldots (1)$

Comparing eq^n (1) to master theorem,
$a = 1, \quad b = 8/5, \quad c = 1, \quad k = 1$

Now,

$b^k = (8/5)^1$

$= 1.6$

$\therefore b^k > a$

$\therefore n^k = n$ [k = 1]

Again,

$T(n) = T(\frac{n}{4}) + n$ ... (ii) [$T(n) = T(\frac{5n}{8}) + n$ = n]

Comparing eqn (ii) with master theorem

$a = 1, \quad b = 4, \quad c = 1, \quad k = 1$

$\therefore b^k = (4)^k$ [a = 1]

$\therefore b^k > a$

$\therefore T(n) = n^k$

$= n$

$= O(n)$

[Proved]

vi) Given,

$$T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{4n}{9}\right) + n$$

Here, taking second part,

$$T(n) = T\left(\frac{n}{9/4}\right) + n \quad \ldots (1)$$

Comparing eqn (1) with master theorem,

$a = 1$, $b = 9/4$, $c = 1$, $k = 1$.

$\therefore b^k = (9/4)^1$

$= 2.25$

$\therefore b^k > a$

$\therefore T(n) = n^k$

$= n \quad [k = 1]$

Again,

$T(n) = T\left(\frac{n}{3}\right) + n \quad \ldots (11) \quad \left[\because T\left(\frac{4n}{5}\right) + n = n\right]$

Comparing eqn (11) with master theorem,

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

$\therefore b^k = 3^1$

$= 3$

$\therefore b^k > a \qquad [a = 1]$

$\therefore T(n) = n^k$

$= O(n)$

[Proved].

Ans no. 1(c)

1) Given,

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

Here,

k	j	no. of loops
1	1	$= 2^0$
2	1	$2 = 2^1$
	2	
4	1	$4 = 2^2$
	2	
	3	
	4	
8	1	$8 = 2^3$
	2	
	:	
	8	
n		$n = 2^k$

∴ Time complexity $= 2^0 + 2^1 + 2^2 + 2^3 + \ldots + 2^k$

$= 2^{k+1} - 1$

$= 2 \cdot 2^k - 1$

$= 2n - 1$

$= O(n)$ (Ans)

2) Given,

```
p = 3
while (p < n)
    p = p * p
```

P.T.O

Here,

Step no.	p
1	$3 = 3^1$
2	$3 \times 3 = 3^2$
3	$3 \times 3 \times 3 \times 3 = 3^4$
4	$3 \times 3 \times 3 \times 3^5 = 3^8$
5	$43046721 = 3^{16}$
⋮	
k^{th}	$= 3^{2^{(k-1)}}$

Here, the term is a geometric series where the first term is 2 and ratio is 2.

series = $1+2+4+8+16 \ldots$

For tight bound, the loop was executed k times,

$\therefore p \geq n$

or, $3^{2^{(k-1)}} \geq n$

or, $3^{2^{(k-1)}} = n$ [assuming]

or, $2^{k-1} = \log_3(n)$

$$\text{or, } k-1 = \log_2(\log_3(n))$$

$$\text{or, } k = \log_2(\log_3(n)) + 1$$

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

(Ans.)

Ans no. 1(d)

i) The recurrence function of the given code is,

$$T(n) = T.\left(\frac{n}{3}\right) + 1 \quad \text{(Ans.)}$$

ii) from (i) we got,

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

By applying substitution,

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

or, $$T\left(\frac{n}{3}\right) = T\left(\frac{n}{3}/3\right) + 1 + 1$$

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

P.T.O

Here,

$T(n) = T(n/3) + 1 \Rightarrow$ step 2

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

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

$= T(\frac{n}{3^4}) + 4 \Rightarrow$ step 5

$= T(\frac{n}{3^k}) + k \Rightarrow$ step k .... (11)

Now, assuming,

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

or, $n = 3^k$

or, $k = \log_3(n)$.

From eq^{n}(11) we get,

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

or, $T(n) = T(1) + \log_3(n)$

$$= 1 + \log_3(n) \approx O(\log_3 n)$$

(Ans.)

Ans. no. 1 (a) (searching)

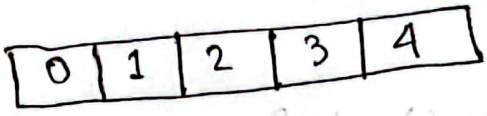
```
def binarySearch (list, x):
    if x >= list [-1]
        return len(list)

    left = 0
    right = len(list) - 1

    while left <= right:
        mid = (left + right) // 2
        if list[mid] > x:
            if list [mid - 1] <= x:
                return mid

            else:
                right = mid - 1

        else:
            left = mid + 1


a, b = input().split()
list 1 = input().split()
list 2 = input().split()
Final Array = []
```

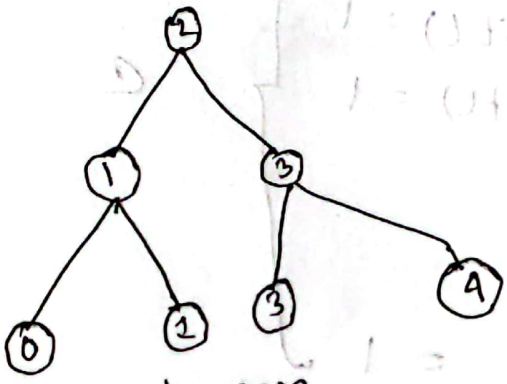
```
a= int(a)
b= int(b)
i= 0
j= 0
while   i < a:
        temp = int(list1[i])
        list1[i] = temp
        i += 1
while   j <b:
        temp= int (list2[j])
        list2[j] = temp
        j+= 1

for i in ~~rang~~ list2:
        a= binary Search (list1, i)
        finalArray. append (a)

print (finalArray)
```

Ans no. 1(b)

Discussing the time complexity of the code,

1) In the binary search,





Now, in case of a worst-case scenario, we need to check the bottom layer of the tree. We know, the height of the tree is $\log(n)$.

∴ Time complexity of the search $= \log(n)$.

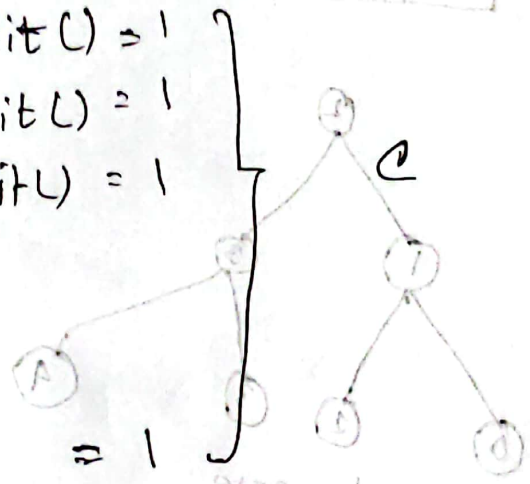
P.T.O

∴ Time complexity of the binary search = log(n)

Analysing the ~~the~~ whole code,

```
a, b = input().split() = 1
list1 = input().split() = 1
list2 = input().split() = 1
.
.
.
j = 0
```

= 1



```
while i < a:          =  O(n)
    int conversion.

while j < a:          =  O(n)
    int conversion

for i in list1:                    → n
    a = binarySearch(list2, i)     → log(n)
    finalArray.append(a)           → 1
```

∴ Total ~~time~~ time = $n \cdot \log(n) + n + n + c$ ← all other input/output

complexity

$\approx O(n \cdot \log(n))$

(Ans.)