GIT Department of Computer Engineering
CSE 222/505 - Spring 2022
Homework 2 Report

Yunus Emre Yumşak
1801042659

1)

a) $log_2n < n$ since O is upper bound it is a true statement.

$$log_2n \leq c * (n) \; \forall \, n \geq n_0$$

b) $\sqrt{n(n+1)} = \sqrt{n^2+n} = \Omega(n)$ it is a true statement.

$$\sqrt{n(n+1)} \geq c * (n) \; \forall \, n \geq n_0$$

c) $c_1 * (n^n) \;\; \forall \, n \leq n^{n-1} \leq c_2 * (n^n) \;\; \forall \, n$ thus it is true.

2)

If $\lim_{n \to \infty} \frac{f(n)}{g(n)} = \infty$ f(n)'s growth rate is bigger or vice versa. If we apply this for the given equations we get:

$\log \, n < \sqrt{n} < n^2 < n^2 \log n < n^3 = 8^{log_2n} < 2^n < 10^n$

$8^{log_2n} = n^{log_28} = n^{3* \, log_22} = n^3$

3) What is the time complexity of the following programs? Use most appropriate asymptotic notation. Explain by giving details.

a)

```
int p_1 ( int my_array[]){
        for(int i=2; i<=n; i++){
                if(i%2==0){
                        count++;      1
                } else{
                        i=(i-1)i;     1
                }
        }
}
```

$n-1 \Rightarrow O(n)$

b)

```
int p_2 (int my_array[]){
        first_element = my_array[0];      1
        second_element = my_array[0];     1
        for(int i=0; i<sizeofArray; i++){
                if(my_array[i]<first_element){
                        second_element=first_element;
                        first_element=my_array[i];      1
                }else if(my_array[i]<second_element){
                        if(my_array[i]!= first_element){
                                second_element= my_array[i];   1
                        }
                }
}
```

$n \Rightarrow O(n)$

c)

```
int p_3 (int array[]) {
        return array[0] * array[2];
}
```

$1 \Rightarrow \Theta(1)$

d)

```
int p_4(int array[], int n) {
        Int sum = 0
        for (int i = 0; i < n; i=i+5)
                sum += array[i] * array[i];
        return sum;
}
```

1, $1 \} n$, 1

$n+2 \Rightarrow O(n)$

e)

```
void p_5 (int array[], int n){
        for (int i = 0; i < n; i++)
                for (int j = 1; j < i; j=j*2)
                        printf("%d", array[i] * array[j]);
}
```

$1 \} n \} n \} n^2 \Rightarrow O(n^2)$

f)

```
int p_6(int array[], int n) {
        If (p_4(array, n)) > 1000)
                p_5(array, n)
        else printf("%d", p_3(array) * p_4(array, n))
}
```

$\rightarrow O(n)$

$\rightarrow O(n^2)$

$O(n)$

$O(n^2)$

g)

```
int  p_7( int n ){
        int i = n;
        while (i > 0) {
                for (int j = 0; j < n; j++)
                        System.out.println("*");
                i = i / 2;
        }
}
```

1, $1 \} n \} \log n$

$\rightarrow O(n + \log n)$

$= O(n)$

h)

```
int  p_8( int n ){
        while (n > 0) {
                for (int j = 0; j < n; j++)
                        System.out.println("*");
                n = n / 2;
        }
}
```

$1 \} n \} \log n$

$O(n + \log n)$

$= O(n)$

4)

a) "The running time of algorithm A is at least $O(n^2)$". Is wrong because Big-oh notation is used for upper bound.

b)

I. $2^{n+1} = O\ (2^n)$

$2^n \leq c * (2^n)\ \forall\, n \geq n_0$ is true thus the statement is true

II. $2^{2n} = O\ (2^n)$

$4^n \leq c * (2^n)\ \forall\, n \geq n_0$ is not true because $4^n > c * (2^n)$.

III. If it were $O(n^4)$ it would be true but since it is Θ notation it could be less than $\Theta(n^4)$.

5)

a) T(n) = 2T(n/2) + n, T(1) = 1
T(n) = 2 (2T(n/4) + n/2) + n
T(n) = 2 (4T(n/8) + n/4) + n/2 + n
.
.
.

$\mathrm{T(n)} = 2(\, 2^{k-1}T\left(\frac{n}{2^k}\right) + \frac{n}{2^{k-1}} + \cdots + \frac{n}{2^2} + \frac{n}{2} + n$

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

$n = 2^k\ and\ k = logn$

$\mathrm{T(n)} = 2(\frac{n}{2} + n)$

$\mathrm{T(n)} = (n)$

b) T(n)=2 T(n-1)+1, T(0)=0
T(n)=2 (2T(n-2)+1)+1 => T(n)=4T(n-2)+3
T(n)=4 (2T(n-3)+1)+3 => T(n)=8T(n-3)+7
T(n)=k 2T(n-k)+(k+1) Assume n-k = 0
T(n)=n 2T(0)+(n+1)
T(n)=n 2*0 + n+1
T(n)=n+1 => T(n)=(n)

6)

```
for (i = 0; i < ARRAY_SIZE; i++){
    for (j = i; j < ARRAY_SIZE; j++){

        if ((array[i]+array[j]==x) && (i != j)){
            printf("(%d , %d) \n",array[i] , array[j] ); 1
        }

    }
}
```

$n \quad n \Rightarrow O(n^2)$

```
cse312@ubuntu:~/Desktop/data$ ./output
(1 , 9)
(2 , 8)
(3 , 7)
(4 , 6)
fun() took 52.000000 unit of time to execute with 20 elements.
```

```
(1 , 9)
(2 , 8)
(3 , 7)
(4 , 6)
fun() took 442.000000 unit of time to execute with 400 elements.
```

Experimental result is better than theoretical result.

7)

```
Func(array){
    If(size_of(array)==1) return;                       → 1
    Print(" check_sum(array[0], array[1,,,,,,n])");     → n        } O(n^2)
    Func(array[1,,,,,,,,,,n]);                          → n
}
```