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

Berkan AKIN
171044073

1)

Big O notation

$T(N) = O(f(n))$

$T(N) \leq c\, f(N)$ when $N \geq n_0$

Omega notation

$T(N) = \Omega(f(N))$

$T(N) \geq c\, f(N)$

Theta notation

$O(f(n)) = \Omega(f(n)) = \Theta(f(n))$

a) $\log_2 n^2 + 7 = O(n)$

$\log_2 n^2 \leq cn$

$c = 7000 \quad n_0 = 7$

$\log_2 .7^2 \leq 7000.7$ True

a) True

b) $\sqrt{n(n+1)} = \Omega(n)$

$\sqrt{n^2+n} \geq cn$

$n_0 = 5 \quad c = 1$

$\sqrt{26} \geq 5$ True

b) True

a)	True
b)	True
c)	True

c) $n^{n-1} = \Theta(n^n)$

$n^{n-1} = O(n^n)$

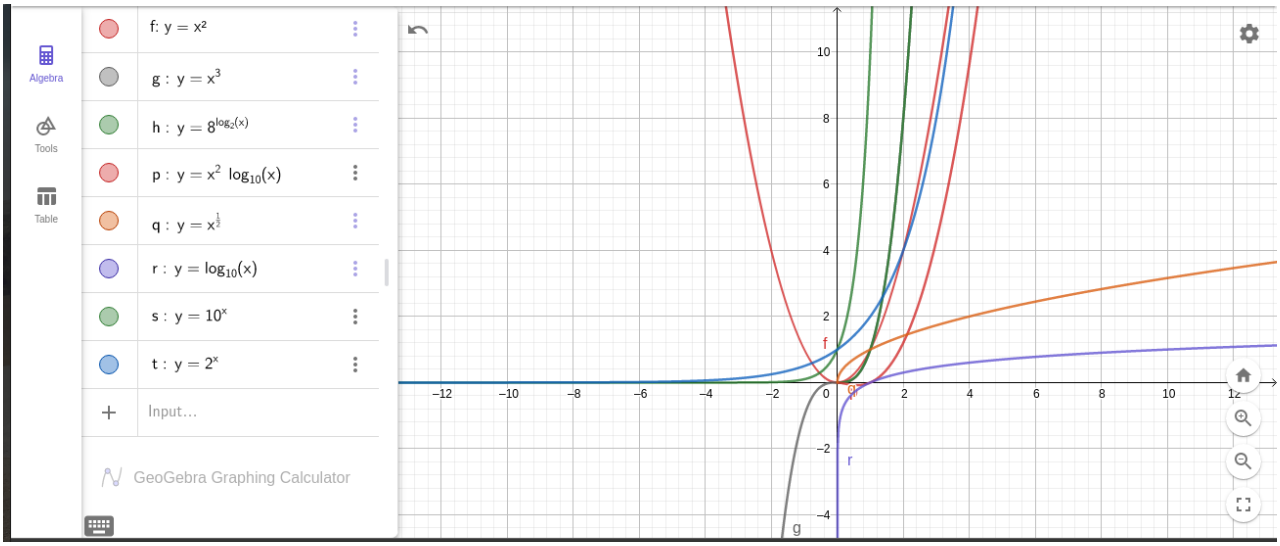
$n^{n-1} = \Omega(n^n)$ $\Rightarrow$ $O(n^n) = \Omega(n^n) = \Theta(n^2)$ True

c) = True

CamScanner ile tarandı

2) When I do the necessary operations for the functions and make a graph, the result is as follows.

$$10^n > 2^n > 8^{\log_2 n} = n^3 > n^2 \log n > n^2 > \log n$$



3)
a)
```
int p_1(int my_array[]){
    for(int i=2; i<=n; i++){
        if(i%2==0){
            count++;
        }else{                1
            i=(i-1)*i         1
        }
    }
}
```

step/exec	freq	Total
2	n(n-1)	$2\log(\frac{n}{2})$
1		
1		

1, 2, 1, 2, 42, …, k

$k.(k-1) \le n$

$(k-1)k^2 - k - 1 = n(k-1)$

$k^2 - k - k = n - k$

$(k-1)^3 = n(k-1)$

$(k-1)k = n$

$k-1 = \sqrt{n} - 1$

Best case

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

step exec	freq	total
1	1	1
1	1	1
2	n+1	2n+2
1	n	n
1	n	n
1	n	n
1	n	n
1	n	n
1	n	n
		5n+4

$T(n) = O(5n+4) = O(n)$

$T(n) = \Omega(5n+4) = \Omega(n)$

$O(n) = \Omega(n) = \Theta(n)$

worst case	Average	best case
O(n)	O(n)	O(n)

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

step	Freq	total
1	1	1
		1

$O(1) = \Omega(1) = \Theta(1)$

Best case	Average case	worst case
O(1)	O(1)	O(1)

4)

a) "The running time of algorithm A is at least $O(n^2)$."

Answer

Big O notation determine worst case. we can prediction algorithm work most time for worst case. This sentence wrong because best running time indicate Ω notation instate of Big O notation.

6)

I) $2^{n+7} = \Theta(2^n)$ False / $O(2^{n+7}) = O(2^n)$

$\Omega(2^{n+7}) = \Omega(2^n)$

this statement is false because dont intersection 2^n function

II) $2^{2n} = \Theta(2^n)$ False $O(2^{2n}) = O(4^n)$

$\Omega(2^{2n}) = \Omega(4^n)$

$2^n \leq 4^n$

this statement is false because dont intersect on 2^n function

III) $f(n) = O(n^2)$ $g(n) = \Theta(n^2)$

$f(n) * g(n) = \Theta(n^4)$

f(n) function degree least 2 g(n) function degre constant 2. we dont know theta notation f(n) so we cant prove this statement.

False

5)

a) $T(n) = 2T(n/2) + n$ $T(1) = 1$

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

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

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

$T(n) = T(\frac{n}{2^k}) + kn$

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

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

$n = 2^k$

$k = \log n$

$T(n) = T(1) + n\log n$

$O(n\log n)$

CamScanner ile tarandı

6) $T(n) = 2T(n-1) + 1 \quad T(0) = 0$

$T(1) = 1$
$T(2) = 3$
$T(3) = 7$
$T(4) = 15$

$O(2^n)$

$T(n) = 2^n - 1$

7)

```
void pairs(int *arr, int size, int sum, int *pair){
    int i=0, k=0;
    for(i=0; i<size; i++){
        for(k=i+1; k<size; k++){
            if(sum==(arr[i]+arr[k])){
                pair[0]=arr[i];
                pair[1]=arr[k];
                return;
            }
        }
    }
}
```

	freq	step exec	total
for(i=0; i<size; i++){	2	$n+1$	$2n+2$
for(k=i+1; k<size; k++){	2	$n(n+1)$	$2n^2+2$
if(sum==(arr[i]+arr[k])){	2	$n(n-1)$	$2n^2+2$
pair[0]=arr[i];	1	$n.(n-1)$	n^2-1
pair[1]=arr[k];	1	$n.(n-1)$	n^2-1
return;	1	$n.(n-1)$	n^2-1
			$7n^2-1$

$T(n) = 7n^2 - 1$

$O(7n^2 - 1) = O(n^2)$

worst case $O(n^2)$

Average case $O(n^2)$

Best case $O(n^2)$

7)

```
T(n)→ void recur(int *arr, int size, int sum, int *pair){
    int i=0, k=0;
    if(size==0){                                  1
        return;                                   1
    }
    for(k=0; k<size; k++){                        n+1
        if(sum==(arr[i] +arr[k])){                n
            pair[0] =arr[i];                      n
            pair[1] =arr[k];                      n
        }
    }
    return recur(&arr[1], size-1, sum, pair);     T(n-1)
}
```

size == 0 => T(0) = 1

$T(n) = T(n-1) + 4n + 3$

$T(1) = T(0) + 7$

$T(1) = 7$

$T(2) = 18$

$T(3) = 33$

⋮

$k =$

Best case
$O(1)$

Worst case
$O(n)$

Average case
$O(1) \leq av \leq O(n)$

CamScanner ile tarandı