Homework 2

Berkan AKIN
171044073

Part 1:

I. Searching a product

```
public Furniture findFurniture(int productId) {
      for(int i=0;i<data.getFurnitureNumber();i++) {                    n+1
            if(data.getFurniture(i).getProductNumber()== productId) {   n
                  return data.getFurniture(i);                          1
            }
      }
      return null;                                                      +     1
```

$T_w(n) = 2n+2 = O(g(n))$ $Q(g(n))$
$T_b(n) = +3 = \Omega(g(1))$

II. remove product

```
public Boolean removeFurniture(int productId) {                                 3
      int i=0,index=0 ,flag=0;                                                  1
      Furniture removeFurniture = findFurniture(productId);                     1
      Furniture tmp[]= new Furniture[data.getFurnitureNumber()-1];              n+1
      for(i=0;i<data.getFurnitureNumber();i++) {                                n
            if(removeFurniture.equals(data.getFurniture(i))) {                  n
                  index=i;                                                      n
                  flag=1;
            }
      }
      if(flag==1) {                                                             m
            for(i=0;i<index;i++) {                                              m
                  tmp[i]=data.getFurniture(i);
            }
            for(i=index+1;i<data.getFurnitureNumber();i++) {                    (n+1)
                  tmp[i-1]=data.getFurniture(i);                                n
            }
            data.addFurnitureNumber(-1);                                        1
            data.FurnitureSwap(tmp);                                            1
                  return true;                                                  1
            }
            else {                                                              1
                  return false;                                                 1
            }
      }
```

+

$T_w(g(n)) = 3n+6 +2m+1 +2n+4$
worst case so “index” equals last index $m = n$
$T_w(g(n)) = 7n+11 = O(n)$

$T_b(g(n)) = 19 = \Omega(n)$

Add Product

```
public Boolean addFurniture(Furniture newFurniture) {                    1
      int i=0;                                                           1
      data.addFurnitureNumber(1);                                        1
      Furniture tmp[]= new Furniture[data.getFurnitureNumber()];         1

      for(i=0;i<data.getFurnitureNumber()-1;i++) {                      n+1
            tmp[i]= data.getFurniture(i);                                n
      }
      tmp[i]=newFurniture;                                               1
      data.FurnitureSwap(tmp);                                           1
      return true;                                                       1
      }
```

+

$T_w(n)=3 +2n+1 +3 =2n+7$
$=O(n)$
$T_b(n)= 7$
$=\Omega(1)$

Part 2:

Q1-)Answer

It is meaningless since the larger the value of n, the larger the time-complexity.

Q2)Answer

prove

1-) Since we are requiring both f and g to be asymptotically non-negative, suppose that we are past some n_1 where both are non-negative (take the max of the two bounds on the n corresponding to both f and g. Let $c_1 = 0.5$ and $c_2=1$.

2-) $0 \leq 0.5(f(n) + g(n)) \leq 0.5(\max(f(n), g(n)) + \max(f(n), g(n)))$

3-) $\max(f(n), g(n)) \leq \max(f(n), g(n)) + \min(f(n), g(n)) = (f(n) + g(n))$

Q3)Answer

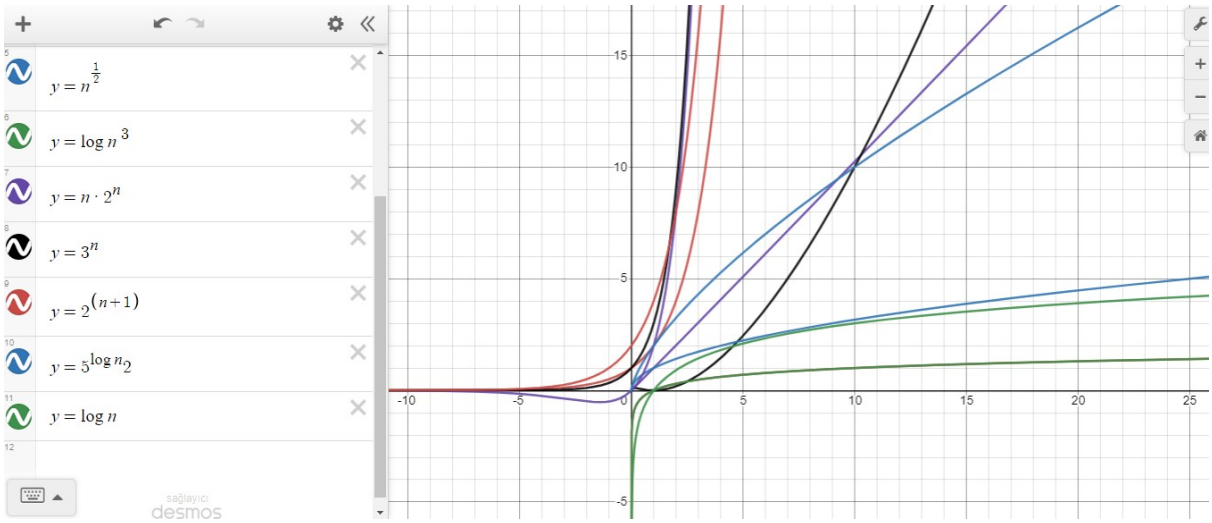
1)2^{n+1} = $O(n^{n+1})$ 1 is unimportant value which ignored. True

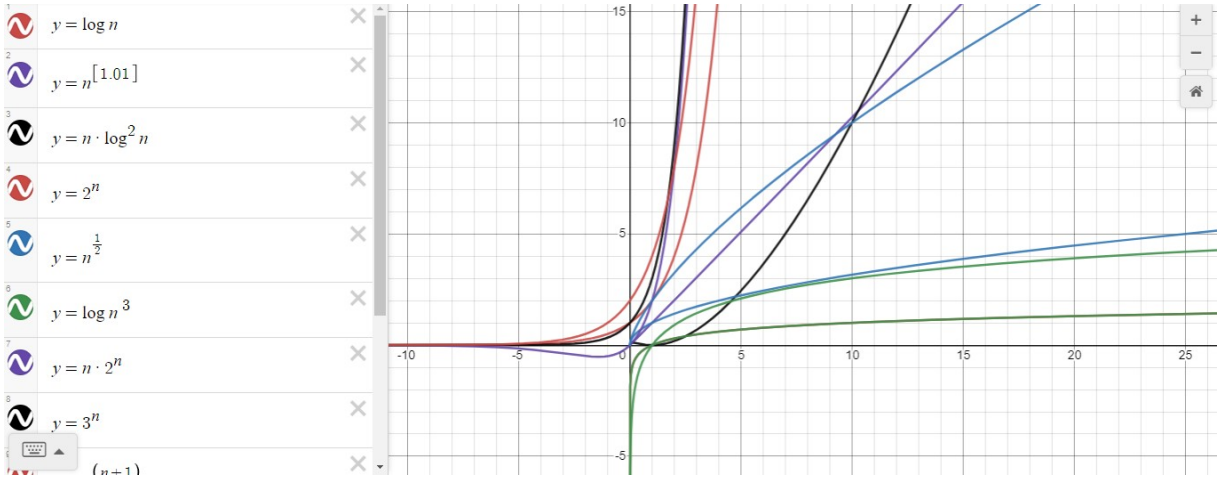
2)2^{2n} = $O(2^n)$ 2 coeficient unimportant value which ignored. True

3)False each one diffirent function O() and Q() functions isnt equal to multiplication

Part 3:
I drew each function, awere of order

$3^n > n2^n > 2^{n+1} > 2^n > 5^{\log_2 n} > n\log^2 n > n^{1.01} > \sqrt{n} > (\log n)^3 > \log n$





Part 4:

1-)

```
SET Max to array[0]                 1
FOR i = 1 to array length – 1       n+1
 IF array[i] > Max THEN             n
      SET Max to array[i]           1
      ENDIF                         1
ENDFOR                              1
PRINT Max                           1
```

+

T_w**(n)**=2n +7 =O(n)
T_b**(n)**=7=Ω(1)

2-)

Array A	**1**
Size N	1
SORT(A)	n
middle = (N + 1) / 2	1
DISPLAY A[middle] as median	1

+

T_w(n)=n+4 = O(n)
T_b(n) =5 = Ω(1)

3-)

```
unordered_set s                                          1
for(i=0 to end )                                         n+1
if(s.find(target_sum - arr[i]) == s.end)                 n
   insert(arr[i] into s)                                 n
  else                                                   n
   print arr[i], target-arr[i]                           n
```

T_w =3n+1 =O(n) +

T_b =3n+1 = O(n)

4-)

Part 5:
Space Complexcity
1)

```
int p_1 (int array[]):                    array =n
{
return array[0] * array[2])
}
```

+

S(n)=n

2)

```
int p_2 (int array[], int n):          array=n
{                                         n=1   // n is parameter
Int sum = 0                              sum=1
for (int i = 0; i < n; i=i+5)             i=1
sum += array[i] * array[i])
return sum
```

```
}                                          +

                                          S(n) =n+3
```

3)

```
void p_3 (int array[], int n):
{
for (int i = 0; i < n; i++)                          array =n
  for (int j = 0; j < i; j=j*2)                   n = 1  // n is parameter
    printf("%d", array[i] * array[j])                  i=1
}                                                +      j=1

                                                      S(n)=n+3
```

4-)

```
void p_4 (int array[], int n):                            array =n
{                                                             n=1
If (p_2(array, n)) > 1000)                              2 // p_2() variable
     p_3(array, n)                                      2 // p_3() variable
else
     printf("%d", p_1 (array) * p_2 (array, n))
}                                                  +

                                                    S(n) =n+5
```

Time Complexity

Part 5:

a)

```
int p_1 (int array[]):
 {
 return array[0] *
array[2];
 }
```

Step/exec	Freq	Total
2	1	2

b)

```
int p_2 (int array[], int n):
{
Int sum = 0
for (int i = 0; i < n; i=i+5)
 sum += array[i] * array[i])
return sum
 }
```

Step/exec	Freq	Total
1	1	1
2	n+1	2n+2
2	n	2n
1	1	1

c)

```
void p_3 (int array[], int n):
 {
 for (int i = 0; i < n; i++)
        for (int j = 0; j < i; j=j*2)
                printf("%d", array[i] *
        array[j])
}
```

Step/exec	Freq	Total
2	n+1	2n+2
2	nlogn	2nlogn
1	logn	logn

d)

```
void p_4 (int array[], int n):
{
        If (p_2(array, n)) > 1000)
                p_3(array, n)
        else
                printf("%d", p_1(array) * p_2(array, n))
}
```

$T3(n) = \theta(n)$

$T1(n) = \theta(n\log n)$

$T2(n) = \theta(n)$

$T(n)$