





**PART 1**  
**SEARCH**  
**ADD-REMOVE**



```
public boolean product_is_in(int branch,String furniture,char model,String color){
```



```
    int model_number=(int)model;
```

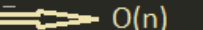

```
    model_number-=65;
```



```
    if(furniture.equals("chairs")){   $O(n)$   
        if(color.equals("blue")){   $O(n)$   
            return (katalog[branch].chairs[model_number][0]>0);   $\Theta(1)$ 
```


```
        }  
        else if(color.equals("red")){  
            return (katalog[branch].chairs[model_number][1]>0);   $\Theta(1)$ 
```


```
        }  
        else if(color.equals("yellow")){   $O(n)$   
            return (katalog[branch].chairs[model_number][2]>0);   $\Theta(1)$ 
```



```
        }  
        else if(color.equals("green")){   $O(n)$   
            return (katalog[branch].chairs[model_number][3]>0);   $\Theta(1)$ 
```



```
        }  
        else if(color.equals("red")){   $O(n)$   
            return (katalog[branch].chairs[model_number][4]>0);   $\Theta(1)$ 
```



```
        }  
        else if(color.equals("brown")){   $O(n)$   
            return (katalog[branch].chairs[model_number][5]>0);   $\Theta(1)$ 
```



```
        }  
        else  
            return false;   $\Theta(1)$ 
```



```
    }  
    else if(furniture.equals("desks")){   $O(n)$ 
```


```
        if(color.equals("blue")){   $O(n)$   
            return (katalog[branch].desks[model_number][0]>0);   $\Theta(1)$ 
```


```
        }  
        else if(color.equals("red")){   $O(n)$   
            return (katalog[branch].desks[model_number][1]>0);   $\Theta(1)$ 
```



```
        }  
        else if(color.equals("yellow")){   $O(n)$   
            return (katalog[branch].desks[model_number][2]>0);   $\Theta(1)$ 
```

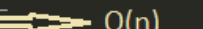
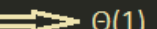
```
        }  
        else if(color.equals("green")){   $O(n)$   
            return (katalog[branch].desks[model_number][3]>0);   $\Theta(1)$ 
```


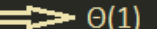
```
        }  
        else if(color.equals("red")){   $O(n)$   
            return (katalog[branch].desks[model_number][4]>0);   $\Theta(1)$ 
```

```
        }  
        else  
            return false;   $\Theta(1)$ 
```

```
    }  
    else if(furniture.equals("meeting_tables")){   $O(n)$ 
```

```
        if(color.equals("blue")){   $O(n)$   
            return (katalog[branch].meeting_tables[model_number][0]>0);   $\Theta(1)$ 
```

```
        }  
        else if(color.equals("red")){   $O(n)$   
            return (katalog[branch].meeting_tables[model_number][1]>0);   $\Theta(1)$ 
```

```
        }  
        else if(color.equals("yellow")){   $O(n)$   
            return (katalog[branch].meeting_tables[model_number][2]>0);   $\Theta(1)$ 
```

```
        }  
        else  
            return false;   $\Theta(1)$ 
```












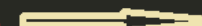
  $O(n)$

  $O(n)$

  $O(n)$

  $O(n)$

```

public void add_product(int branch_id,String product,int color_id,int model_id)throws ahmets_exceptions{
    if(branch_id>katalog.length+1){   $\Theta(1)$    $\Theta(1)$ 
        throw new ahmets_exceptions("INCORRECT BRANCH ID PLEASE TRY AGAIN");   $\Theta(1)$ 
    }
    if(product.equals("chairs")){   $O(n)$    $O(n)$ 
        katalog[branch_id].chairs[model_id][color_id]++;   $\Theta(1)$ 
    }
    if(product.equals("desks")){   $O(n)$    $O(n)$ 
        katalog[branch_id].desks[model_id][color_id]++;   $\Theta(1)$ 
    }
    if(product.equals("meeting_tables")){   $O(n)$    $O(n)$ 
        katalog[branch_id].meeting_tables[model_id][color_id]++;   $\Theta(1)$ 
    }
}
/**

```




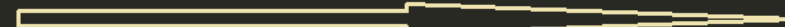





$T(n)=O(n)$

```

/**
 *The desired product is added to the desired branch.just model
 *@param branch_id,product,model_id
 */

```

```

public void add_product(int branch_id,String product,int model id )throws ahmets_exceptions{
    if(branch_id>katalog.length+1){   $\Theta(1)$    $\Theta(1)$ 
        throw new ahmets_exceptions("INCORRECT BRANCH ID PLEASE TRY AGAIN");   $\Theta(1)$ 
    }
    if(product.equals("bookcases")){   $O(n)$    $O(n)$ 
        katalog[branch_id].bookcases[model_id]++;   $\Theta(1)$ 
    }
    if(product.equals("office_cabinets")){   $O(n)$    $O(n)$ 
        katalog[branch_id].office_cabinets[model_id]++;   $\Theta(1)$ 
    }
}

```

$T(n)=O(n)$

```

/**
 *The desired product is deleted to the desired branch for model and color

```

```
public void remove_product(int branch_id,String product,int model_id,int color_id)throws ahmets_exceptions{
```

```
    if(product.equals("chairs")){  $\Rightarrow O(n)$   
        if(katalog[branch_id].chairs[model_id][color_id]<1){  $\Rightarrow \Theta(1)$   
            throw new ahmets_exceptions("THIS PRODUCT ALREADY NO");  $\Rightarrow \Theta(1)$   
        }  
        katalog[branch_id].chairs[model_id][color_id]--;  $\Rightarrow \Theta(1)$ 
```

$\Rightarrow O(n)$

```
    }  
    if(product.equals("desks")){  $\Rightarrow O(n)$   
        if(katalog[branch_id].desks[model_id][color_id]<1){  $\Rightarrow \Theta(1)$   
            throw new ahmets_exceptions("THIS PRODUCT ALREADY NO");  $\Rightarrow \Theta(1)$   
        }  
        katalog[branch_id].desks[model_id][color_id]--;  $\Rightarrow \Theta(1)$ 
```

$\Rightarrow O(n)$

```
    }  
    if(product.equals("meeting_tables")){  $\Rightarrow O(n)$   
        if(katalog[branch_id].meeting_tables[model_id][color_id]<1){  $\Rightarrow \Theta(1)$   
            throw new ahmets_exceptions("THIS PRODUCT ALREADY NO");  $\Rightarrow \Theta(1)$   
        }  
        katalog[branch_id].meeting_tables[model_id][color_id]--;  $\Rightarrow \Theta(1)$ 
```

$\Rightarrow O(n)$

```
    }
```

$O(n)$

```
}
```

```
/**
```

```
*The desired product is deleted to the desired branch.just model
```

```
*@param branch_id,product,model_id
```

```
*/
```

```
public void remove_product(int branch id,String product,int model_id )throws ahmets_exceptions{
```

```
    if(product.equals("bookcases")){  $\Rightarrow O(n)$   
        if(katalog[branch_id].bookcases[model_id]<1){  $\Rightarrow \Theta(1)$   
            throw new ahmets_exceptions("THIS PRODUCT ALREADY NO");  $\Rightarrow \Theta(1)$   
        }  
        katalog[branch_id].bookcases[model_id]--;  $\Rightarrow \Theta(1)$ 
```

$\Rightarrow O(n)$

```
    }  
    if(product.equals("office_cabinets")){  $\Rightarrow O(n)$   
        if(katalog[branch_id].office_cabinets[model_id]<1){  $\Rightarrow \Theta(1)$   
            throw new ahmets_exceptions("THIS PRODUCT ALREADY NO");  $\Rightarrow \Theta(1)$   
        }  
        katalog[branch_id].office_cabinets[model_id]--;  $\Rightarrow \Theta(1)$ 
```

$\Rightarrow O(n)$

$O(n)$

```
}
```



## Part 2

a) Big-O is an Asymptotic notation for the worst case, or ceiling of growth for a given function. It gives us an asymptotic upper bound for the growth rate of runtime of an algorithm. So we can say "at least"

b)  $F(n) \leq F(n) + g(n)$  and  $g(n) \leq F(n) + g(n)$

$$\max(F(n), g(n)) \in O(F(n) + g(n))$$

$$F(n) + g(n) \leq 2 \max(F(n), g(n))$$

$$\max(F(n), g(n)) \in \Omega(F(n) + g(n))$$

we get that

$$\max(F(n), g(n)) \in \Theta(F(n) + g(n))$$

Note that

$$\max(F(n), g(n)) = \begin{cases} F(n) & \text{if } F(n) \geq g(n) \\ g(n) & \text{if } g(n) \geq F(n) \end{cases}$$

if  $F(n) = 10n$  and  $g(n) = n^2$ , we get that

$$\max(F(n), g(n)) = \begin{cases} 10n & \text{if } n \leq 10 \\ n^2 & \text{if } n \geq 10 \end{cases}$$



## Part 2

C) if  $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = c \neq 0 \Rightarrow f(n) = O(g(n))$

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

$$\lim_{n \rightarrow \infty} \frac{2^{n+1}}{2^n} = \lim_{n \rightarrow \infty} \frac{2 \cdot 2^n}{2^n} = 2 \text{ so true}$$

2)  $2^n = O(2^n)$

$$\lim_{n \rightarrow \infty} \frac{2^n}{2^n} = \lim_{n \rightarrow \infty} \frac{2^n \cdot 2^n}{2^n} = \infty \text{ so it is false}$$

3)  $f(n) = O(n^2) \Rightarrow 0 \leq f(n) \leq Cn^2$

$g(n) = O(n^2) \Rightarrow C_1 n^2 \leq g(n) \leq C_2 n^2$

$0 \leq f(n) \cdot g(n) \leq C \cdot C_2 \cdot n^4$

$O(n^4) = f(n) \cdot g(n) \rightarrow C_3 \cdot n^4 \leq f(n) \cdot g(n) \leq C_4 \cdot n^4 \quad \times$

$O(n^4) = f(n) \cdot g(n) \rightarrow 0 \leq f(n) \cdot g(n) \leq C_5 \cdot n^4 \quad \checkmark$



### Part 3

$$n^{1.01}, n \log^2 n, 2^n, \sqrt{n}, \log^3 n, \underbrace{n \cdot 2^n}_{\text{exponential}}, \underbrace{3^n}_{\text{exponential}}, \underbrace{2^{n+1}}_{\text{exponential}}, 5^{\log_2 n}, \log n$$

exponential  $> n^x > \text{linear} > \text{logarithmic}$

- The growth rate can be compared with the limit

$$\begin{aligned} \lim_{N \rightarrow \infty} \frac{F(N)}{g(N)} &= 0 \Rightarrow F(N) = o(g(N)) \\ &= C \neq 0 \Rightarrow F(N) = \Theta(g(N)) \\ &= \infty \Rightarrow g(N) = o(F(N)) \end{aligned}$$

Compare  $\Rightarrow 2^n, n \cdot 2^n, 3^n, 2^{n+1} \rightarrow \text{exponential group}$

$$\lim_{n \rightarrow \infty} \frac{2^n}{2^{n+1}} = \lim_{n \rightarrow \infty} \frac{1}{2} \cdot \frac{2^n}{2^n} = \frac{1}{2} \text{ constant so growth rate } 2^n = 2^{n+1}$$

$$\lim_{n \rightarrow \infty} \frac{2^n}{n \cdot 2^n} = \frac{1}{n} = 0 \text{ so growth rate } 2^n \cdot n > 2^n$$

$$\lim_{n \rightarrow \infty} \frac{n \cdot 2^n}{3^n} = 0 \text{ so growth rate } 3^n > n \cdot 2^n$$

$$\underline{\underline{3^n > n \cdot 2^n > 2^{n+1} = 2^n}}$$

Compare  $\Rightarrow \log^3 n, \log n \rightarrow \text{logarithmic group}$

$$\lim_{n \rightarrow \infty} \frac{\log^3 n}{\log n} = \frac{\log n \cdot \log^2 n}{\log n} = \infty \text{ so growth rate } \log^3 n > \log n$$



Compare  $\Rightarrow n^{1.01}, \sqrt{n}, 5^{\log_2 n}, n \cdot \log^2 n$

$$\lim_{n \rightarrow \infty} \frac{\sqrt{n}}{n \log^2 n} = \lim_{n \rightarrow \infty} \frac{1}{\sqrt{n} \cdot \log^2 n} = 0 \quad \text{so growth rate } n \cdot \log^2 n > \sqrt{n}$$

$$\lim_{n \rightarrow \infty} \frac{n \cdot \log^2 n}{n^{1.01}} = \lim_{n \rightarrow \infty} \frac{\log^2 n}{n^{0.01}} = 0 \quad \text{so growth rate } n^{1.01} > \log^2 n$$

$$\lim_{n \rightarrow \infty} \frac{n^{1.01}}{5^{\log_2 n}} = \lim_{n \rightarrow \infty} \frac{n^{1.01}}{n^{\log_2 5}} = 0 \quad \text{so growth rate } 5^{\log_2 n} > n^{1.01}$$

$\log_2 5 \sim 2$

$$5^{\log_2 n} > n^{1.01} > n \cdot \log^2 n > \sqrt{n}$$

$$\lim_{n \rightarrow \infty} \frac{3^n}{5^{\log_2 n}} = \infty$$

$$3^n > 2^n \cdot n > 2^{n+1} = 2^n$$

$$\log n^3 > \log n$$

$$\lim_{n \rightarrow \infty} \frac{\sqrt{n}}{\log^3 n} = \infty$$

$$3^n > 2^n \cdot n > 2^{n+1} = 2^n > 5^{\log_2 n} > n^{1.01} > n \cdot \log^2 n > \sqrt{n} > \log n^3 > \log n$$



## Part 4

1) Find the minimum-valued item

```
int P_1(int array[], int n) {
```

```
    int min = array[0];  $\rightarrow \phi(1)$ 
```

```
    for (int i = 1; i < n; i++)  $\rightarrow \phi(n)$ 
```

```
        if (min > array[i])  $\rightarrow \phi(1)$ 
```

```
            min = array[i];  $\rightarrow \phi(1)$ 
```

$\phi(1), \phi(n) = \phi(n)$

```
    return min;  $\rightarrow \phi(1)$ 
```

```
}
```

$T(n) = \phi(n)$

2) 

```
int P_2(int array[], int n) {
```

```
    int min, tmp;  $\rightarrow \phi(1)$ 
```

```
    for (int i = 0; i < n-1; i++) {  $\rightarrow \phi(n)$ 
```

```
        min = i;  $\rightarrow \phi(1)$ 
```

```
        for (int j = i+1; j < n; j++) {  $\rightarrow \phi(n)$ 
```

```
            if (array[j] < array[min])  $\rightarrow \phi(1)$ 
```

```
                min = j;
```

```
        }
```

```
        if (min != i) {
```

```
            tmp = array[i];
```

```
            array[i] = array[min];
```

```
            array[min] = tmp;
```

```
        }
```

```
    if (n % 2)  $\rightarrow \phi(1)$ 
```

```
        return (array[n/2] + array[n/2 - 1]) / 2;  $\rightarrow \phi(1)$ 
```

```
    return array[n/2];  $\rightarrow \phi(1)$ 
```

```
}
```

$T(n) = \phi(n^2)$



3)

Find two elements whose sum is equal to a given value

bool P-3 (int array[], int n, int value) {

$T_{2b}(n) = \Theta(1)$   
 $T_{2w}(n) = \Theta(n)$

```

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if (array[i] + array[j] == value && i != j) →  $\Theta(1)$ 
            return true; →  $\Theta(1)$ 
        }
    }

```

$T_{1b}(n) = \Theta(1)$   
 $T_{1w}(n) = \Theta(n)$

return false; →  $\Theta(1)$

}

$$T_w(n) = T_{1w}(n) \cdot T_{2w}(n) = \Theta(n) \cdot \Theta(n) = \Theta(n^2)$$

$$T_b(n) = T_{1b}(n) \cdot T_{2b}(n) = \Theta(1) \cdot \Theta(1) = \Theta(1)$$

→  $T(n) = O(n^2)$



4)

Assume there are two ordered arraylist of  $n$  elements. Merge these two list to get a single list in increasing order

Void P-4 (int array1[], int array2[], int array3[], int n) {

int i=0, j=0, k=0;  $\rightarrow O(1)$

for (; i < n && j < n; k++) {  $\rightarrow T_1(n) = O(n)$

if (array1[i] < array2[j])  
array3[k] = array1[i++]; }  $O(1)$

else  
array3[k] = array2[j++]; }  $O(1)$

}

while (i < n)  $\rightarrow O(n)$

array3[k++] = array1[i++];  $\rightarrow O(1)$

$T_2(n) = O(n)$

while (j < n)  $\rightarrow O(n)$

array3[k++] = array2[j++];  $\rightarrow O(1)$

$T_3(n) = O(n)$

}

$T(n) = T_1(n) + T_2(n) + T_3(n) = O(n)$

$T_w(n) = O(n)$   $\xrightarrow{\text{constant not important}}$   $T_b(n) = T_w(n)$   
 $= T_1(n) = O(n)$



## Part 5)

a) int p-1 (int array[])

{

return array[0] \* array[2];  $\rightarrow O(1)$

}

$T(n) = O(1)$   $S(n) = O(1)$

b)

int p-2 (int array[], int n)

{

int sum = 0  $\rightarrow O(1)$

For (int i = 0; i < n; i = i + 5)  $\rightarrow O(n)$

sum += array[i] \* array[i];  $\rightarrow O(1)$

return sum;  $\rightarrow O(1)$

}

$T(n) = O(n)$   $S(n) = O(1)$

0, 5, 10, 15, ..., n

$5k = n$   
 $k = \frac{n}{5}$   $O(\frac{n}{5})$   
constant

c)

void p-3 (int array[], int n)

{

$O(n \log n)$  { For (int i = 0; i < n; i++)  $\rightarrow O(n)$

For (int j = 0; j < i; j = j \* 2)  $\rightarrow O(\log n)$

printf ("%d", array[i] \* array[j]);  $\rightarrow O(1)$  }  $O(\log n)$

}

$T(n) = O(n \log n)$   $S(n) = O(n)$



d)

```
void p_4(int array[], int n)
```

```
{
```

```
    if (p_2(array, n) > 1000)  $\rightarrow O(n)$ 
```

```
        p_3(array, n);  $\rightarrow O(n \cdot \log n)$ 
```

```
    else
```

```
        printf("%d", p_1(array) * p_2(array, n))  $\rightarrow O(n)$ 
```

```
}
```

$$T(n) = O(n \cdot \log n) \quad S(n) = O(n)$$