

## 1. Print 1D array on console

```
void printArray(int arr[], int n){  
    for(int i = 0 ; i < n ; i++)  
        sysout(arr[i]);  
}
```

No of iterations =  $n$

Time required  $\propto$  No. of iterations

Time  $\propto n$

Time = 1  $\cdot n$  units

Time complexity =  $T(n) = O(n)$

## 2. Print 2D array on console

```
void print2DArray(int arr[][], int m, int n){  
    for(int i = 0 ; i < m ; i++){  
        for(int j = 0 ; j < n ; j++){  
            sysout(arr[i][j]);  
        }  
    }  
}
```

iterations of outer loop =  $m$

iterations of inner loop =  $n$

Total iterations =  $m * n$

Time  $\propto m * n$

Time =  $m * n$  units

Time complexity =  $T(m, n) = O(m * n)$

if row = col =  $n$  Time complexity =  $T(n) = O(n^2)$

### 3. add two numbers

```
int addNumbers(int num1, int num2){  
    return num1 + num2;  
}
```

This algorithm will take constant amount of time irrespective of input/type

Time complexity =  $T(n) = O(1)$

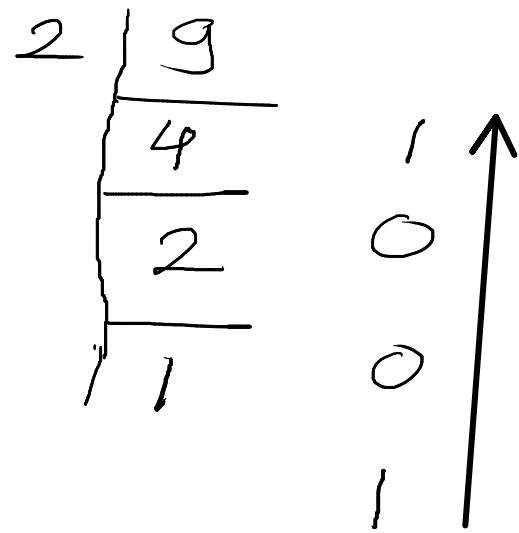
### 4. print table of given number

```
void printTable(int num){  
    for(int i = 1 ; i <= 10 ; i++)  
        sysout(i * num);  
}
```

Loop is going to iterate for constant number of times - constant time requirement

Time complexity =  $T(n) = O(1)$

## 5. print binary of given number



$$(9)_2 = 1001$$

```
void printBinary(int n){
    while(n > 0){
        rem = n % 2;
        sysout(rem);
        n = n / 2;
    }
}
```

n	n > 0	rem
9	T	1
4	T	0
2	T	0
1	T	1
0	F	

$$\begin{aligned} \text{loop var} &= 9, 4, 2, 1, 0 \\ &= n, n/2, n/4, n/8, \dots \\ &= \frac{n}{2^0}, \frac{n}{2^1}, \frac{n}{2^2}, \frac{n}{2^3}, \dots, \frac{n}{2^{\text{itr}}}, \frac{n}{2^{\text{itr}}} \end{aligned}$$

for last time condition will be true for  $n=1$

$$\therefore \frac{n}{2^{\text{itr}}} = 1$$

$$n = 2^{\text{itr}}$$

$$\log n = \log 2^{\text{itr}}$$

$$\text{itr} \log 2 = \log n$$

$$\text{itr} = \frac{\log n}{\log 2}$$

$$\text{Time} \propto \text{itr}$$

$$\text{Time} \propto \frac{\log n}{\log 2}$$

$$\text{Time} = \log n$$

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

**Time complexity :  $O(1)$ ,  $O(\log n)$ ,  $O(n)$ ,  $O(n \log n)$ ,  $O(n^2)$ ,  $O(n^3)$ ,  $O(2^n)$ , .....**

**mod : '+' or '-'**

**>> time complexity in terms of  $n$**

**mod : '/' or '\*'**

**>> time complexity in terms of  $\log n$**

**for(int i = 0 ; i < 10 ; i++)**

**>>  $O(1)$**

**for(int i = 0 ; i < n ; i++)**

**>>  $O(n)$**

**for(int i = n ; i > 0 ; i--)**

**>>  $O(n)$**

**for(int i = 0 ; i < n ; i+=2)**

**>>  $O(n)$**

**for(int i = 0 ; i < n ; i+=20)**

**>>  $O(n)$**

**for(int i = 0 ; i < n ; i++)**

**>>  $O(n^2)$**

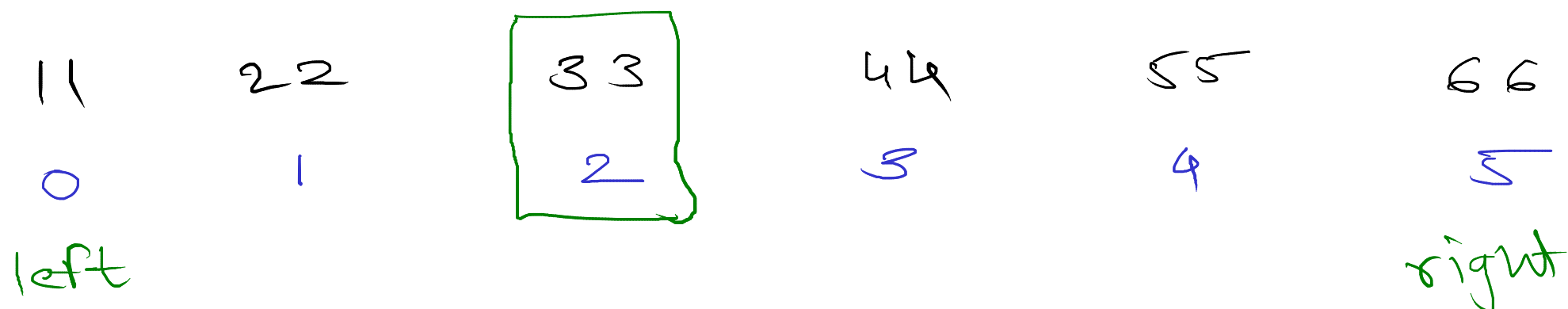
**for(int j = 0 ; j < n ; j++)**

**for(int i = 0 ; i < n ; i++);**

**>>  $O(n)$**

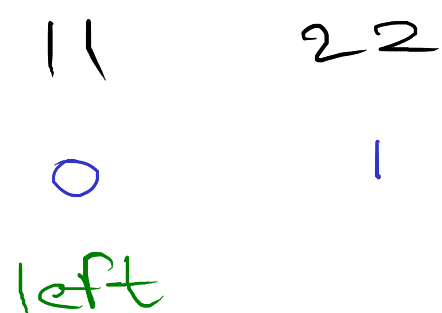
**for(int j = 0 ; j < n ; j++);**

Key = 55



$$\text{mid} = \frac{\text{left} + \text{right}}{2} = \frac{0 + 5}{2} = 2$$

left sub array / left partition



right sub array / right partition



Key = 10

11	22	33	44	55	66
0	1	2	3	4	5
left		mid			right

44	55	66
3	4	5
left	mid	right

```
int binary-search(arr, size, key)
```

```
{  
    int left = 0, right = size - 1, mid;
```

```
    while (left <= right) {  
        mid = (left + right) / 2;
```

```
        if (key == arr[mid])  
            return mid;
```

```
        elseif (key < arr[mid])  
            right = mid - 1;
```

```
        else  
            left = mid + 1;
```

```
    }  
    return -1;
```

66
5
left
right
mid

5	6
right	left