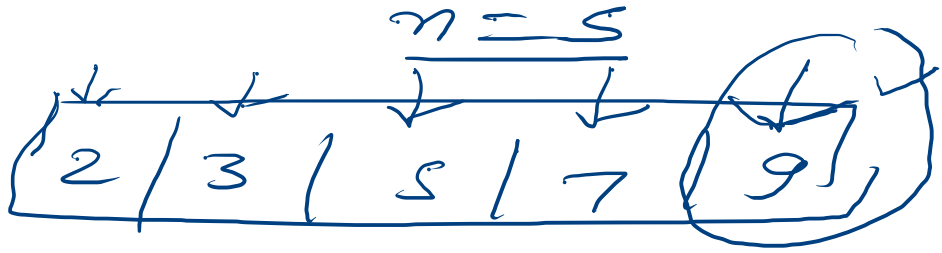
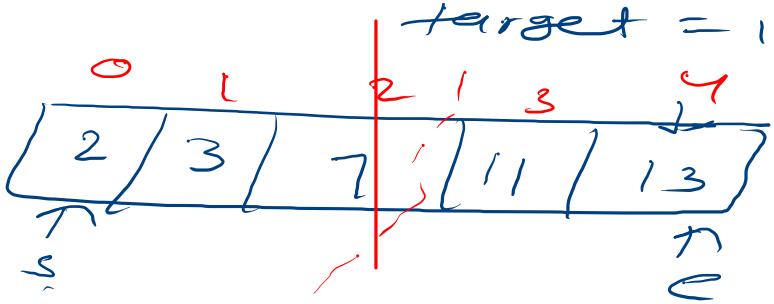
# (1) Binary search



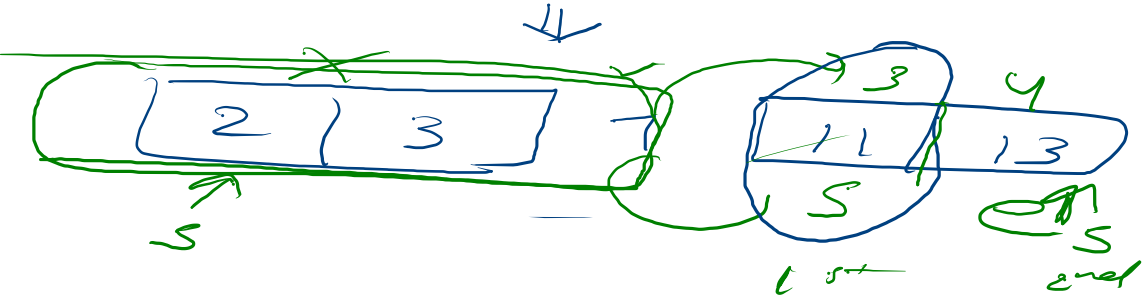

tar = 9 → n operations

## Conditions

(1) When your array is sorted apply binary search




$$\frac{0+4}{2} = \frac{4}{2} \Rightarrow 2$$



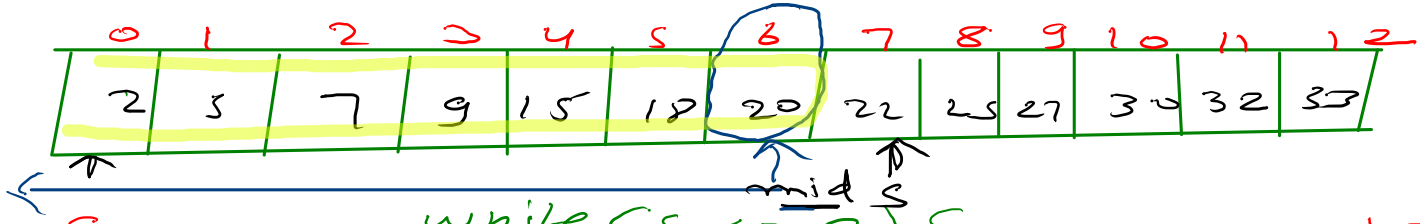

$$\frac{3+4}{2}$$

mid = 13 { equal, greater, smaller

if (arr[mid] == target

return mid;

}



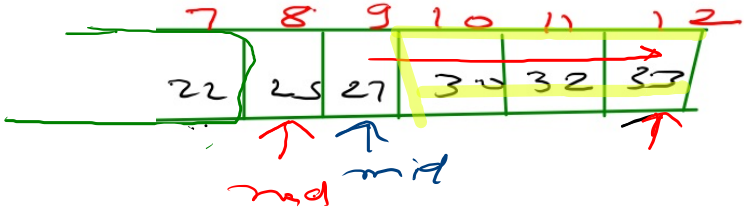

while(s <= e){}

tar = 25

s = 0

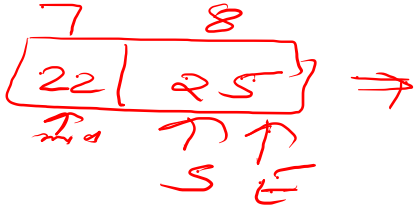
e = 12

calculate the mid



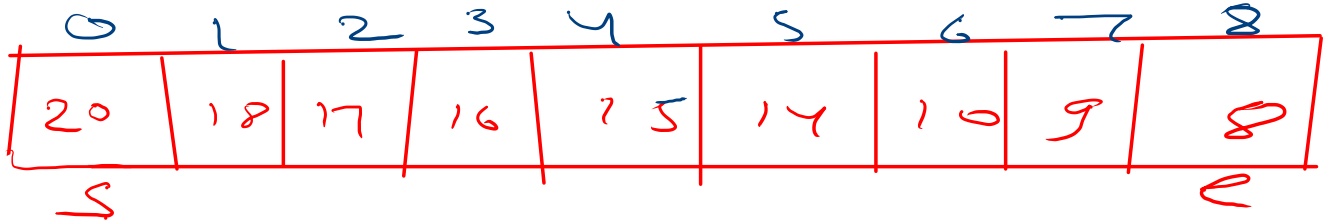

$\Rightarrow \frac{7+12}{2} = 9$

25




$\Rightarrow \frac{7+8}{2} = 7$

$\Rightarrow \frac{8+8}{2} = 8$




target = 9

mid ⇒ $\frac{0+8}{2} = 4$

15

arr[mid] < tar ⇒ s = mid + 1

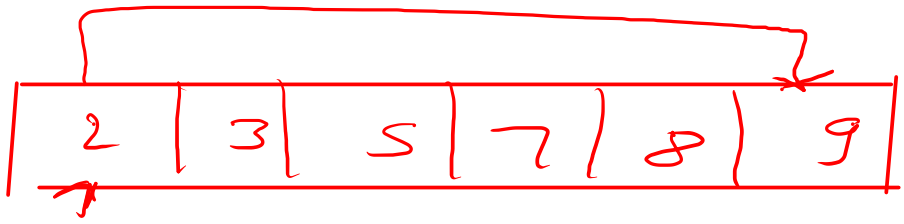
arr[mid] > tar ⇒ e = mid - 1

arr[mid] > target

s = mid + 1

arr[mid] < target

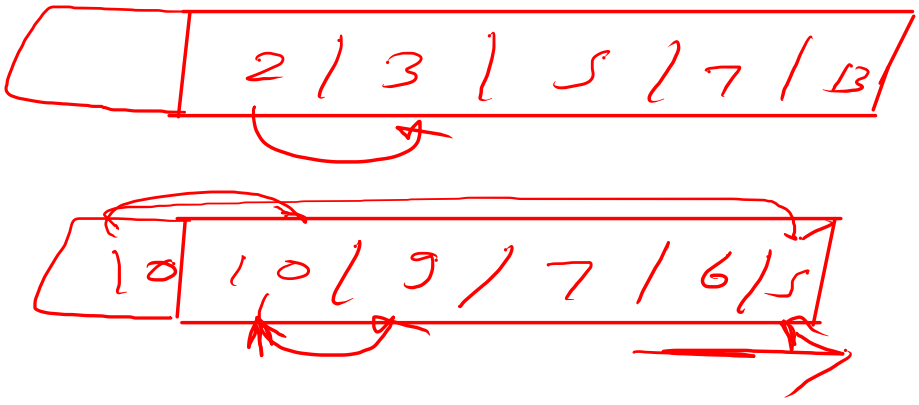
e = mid - 1




Increasing


Decreasing

```
if (arr[0] <= arr[n-1]),
        Increasing()
else,
        Decreasing.
}
```



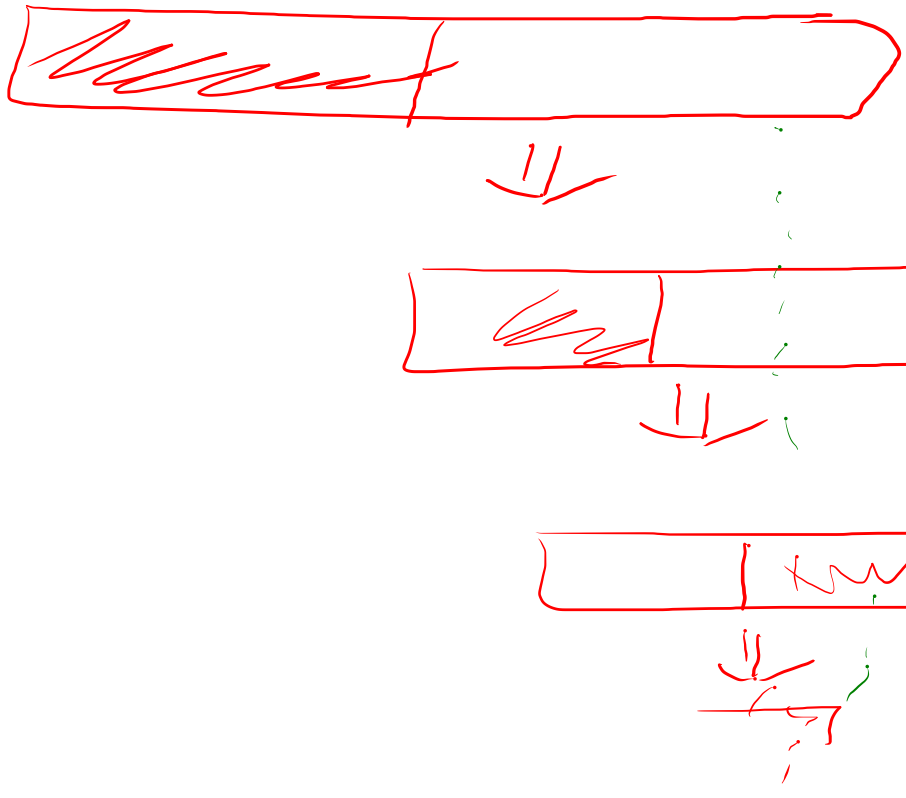

Decr

⇒ IH > last

⇒ IS+1 < last ↑ Incres

10 ≤ 5

Order Agnostic :→ When you don't know your array is sorted or not and you have calculate the index of target element



$N \quad 2^0$

$\frac{N}{2} \Rightarrow \frac{N}{2^1}$

$\frac{N}{4} \Rightarrow \frac{N}{2^2}$

$\frac{N}{8} \Rightarrow \frac{N}{2^3}$

$1 \quad = \quad \frac{N}{2^k}$

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

Best

| 0 | 1 | 2 |
|---|---|---|
| 2 | 3 | 5 |

$N = 2^k$

take log on both side

$\log N = \log(2^k)$

$\log N = k \log 2$

$$k = \frac{\log N}{\log 2} \Rightarrow \log_2$$

$O(\log n)$

1 0 0 0 0 0 0 = arr size

→ last position

$\log_2(1000000)$

| Comparision | |
|---|---|
| Linear search | Binary search |
| 1 0 0 0 0 0 0 | 20 |

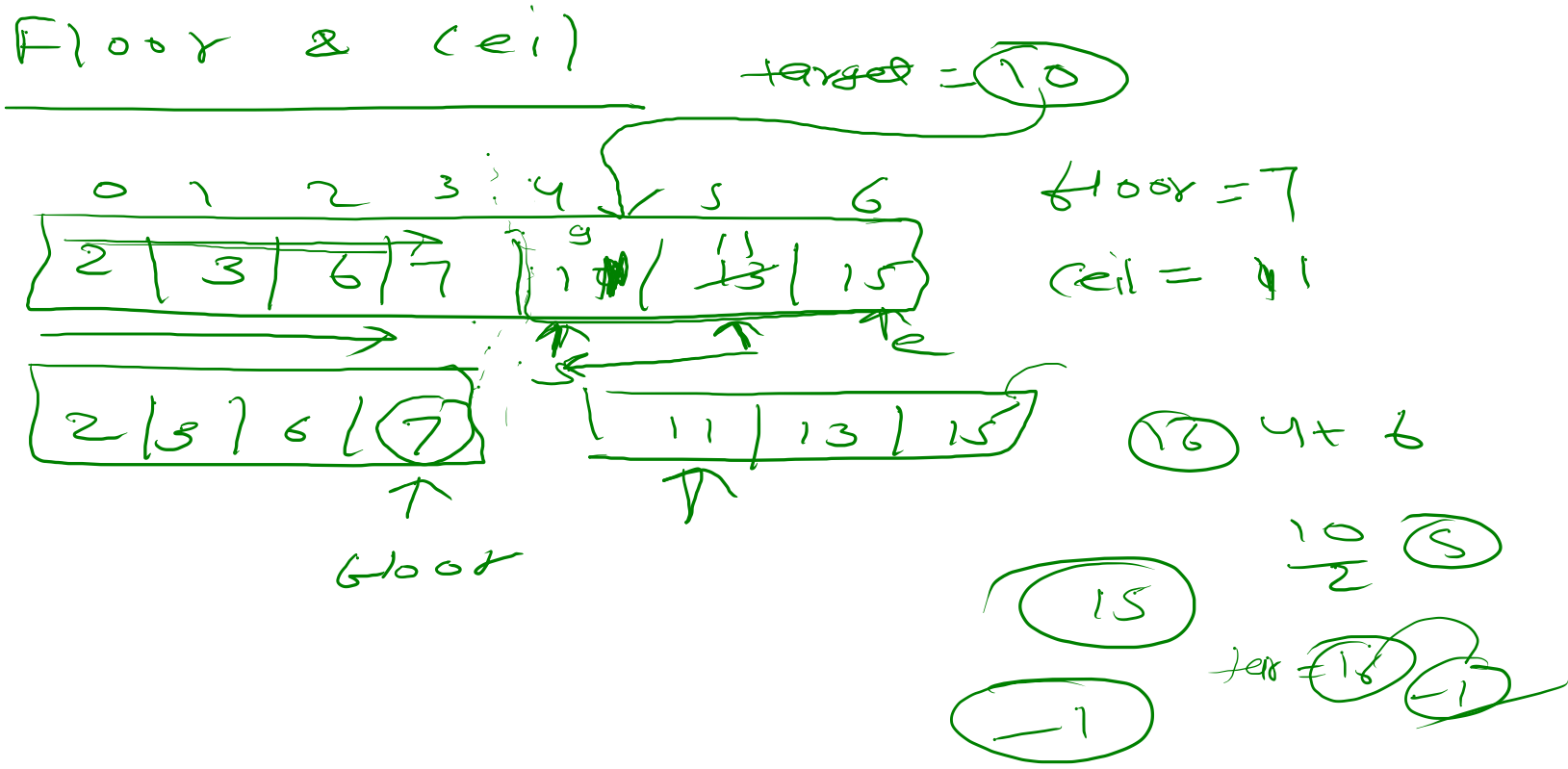
# Binary Search in an Array

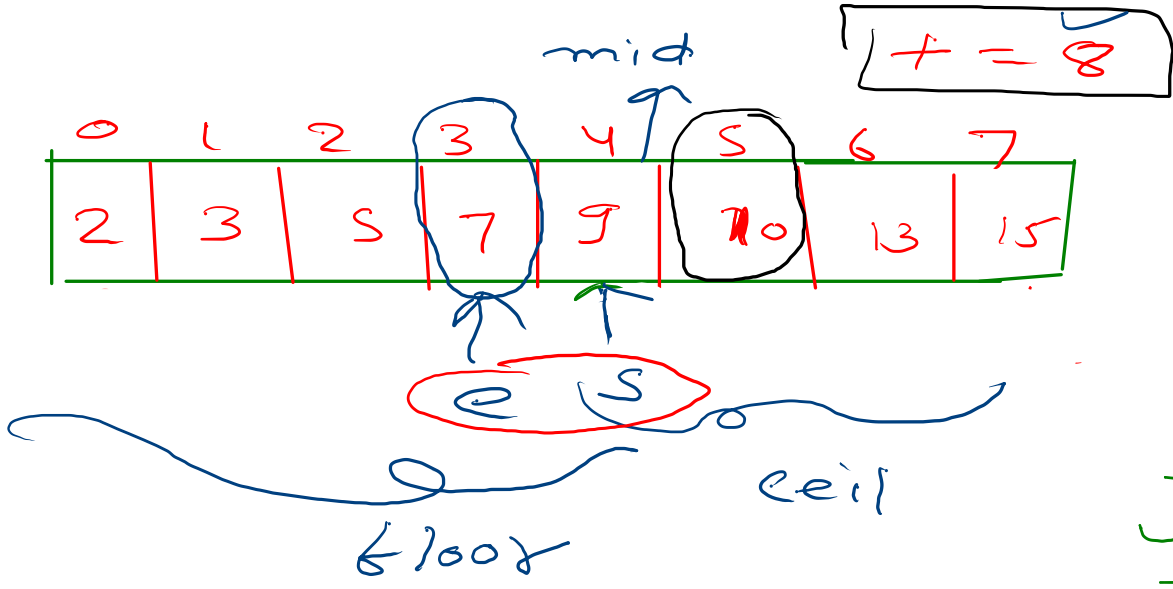
```java
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;

public class Solution {

    public static void main(String[] args) {
        /* Enter your code here. Read input from STDIN. Print output to STDOUT. Your class should be named Solution. */
        Scanner sc  = new Scanner(System.in);
        int n = sc.nextInt();
        int[] arr = new int[n];
        for(int i=0;i<n;i++){
            arr[i] = sc.nextInt();
        }
        int tar = sc.nextInt();
        System.out.println(binarySearchIncaresing(arr,tar));
    }
    public static int binarySearchIncaresing(int[] arr,int target){
        // Assume the pointers
        int start=0;
        int end  = arr.length-1;
        int ans=-1;
        // looping condition
        while(start<=end){
            // calculate the mid
            int mid = (start+end)/2;
            // that will check the equal condition
            if(arr[mid]==target){
                    ans = mid;
                    break;
            }
            // It will check when your mid is less the target
            else if (arr[mid]<target) {
                  start = mid+1;
            }
            // It will check when your mid greater than target
            else{
                   end = mid-1;
            }
        }
        return ans;
    }
}
```

# Floor & Ceil


target = 10
0 1 2 3 4 5 6
2 | 3 | 6 | 7 | 9 | 11 | 15
2 | 3 | 6 | 7
11 | 13 | 15
Floor
Floor = 7
Ceil = 11
16
4+6
10/2
5
15
-1
ans = 16 - 1

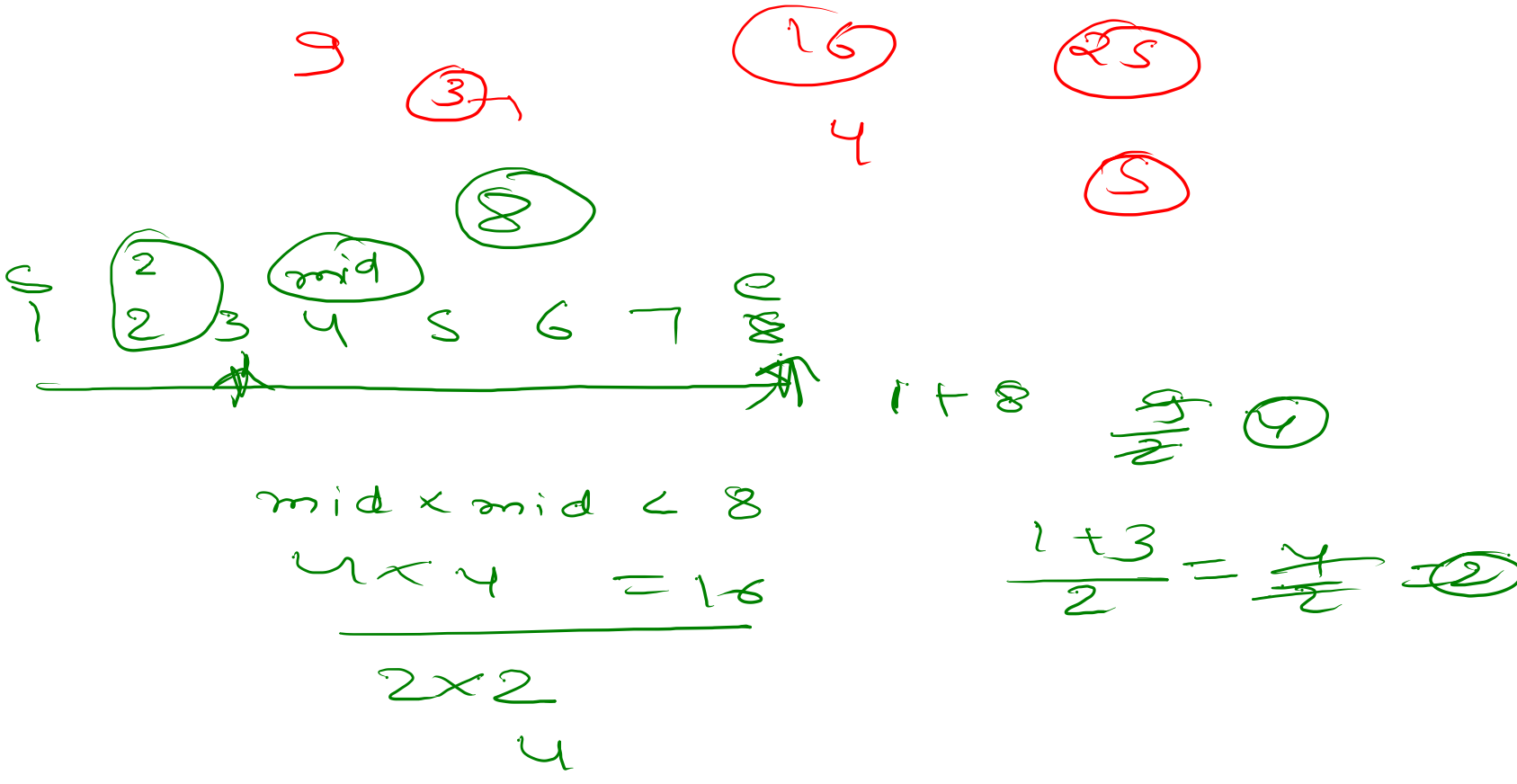



$\frac{0+7}{2} \Rightarrow 3$

$\frac{4+7}{2} = 5$

$\frac{4+4}{2} = 4$

16


9
3
16
4
25
5
8
2
2
mid
S
1
2
3
4
5
6
7
e
8
1 + 8
9/2
mid x mid < 8
4 x 4 = 16
2 x 2
4
1 + 3/2 = 4/2