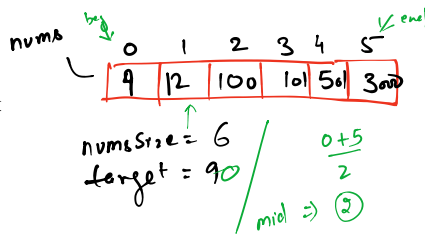


```
int search(int* nums, int numsSize, int target) {
```

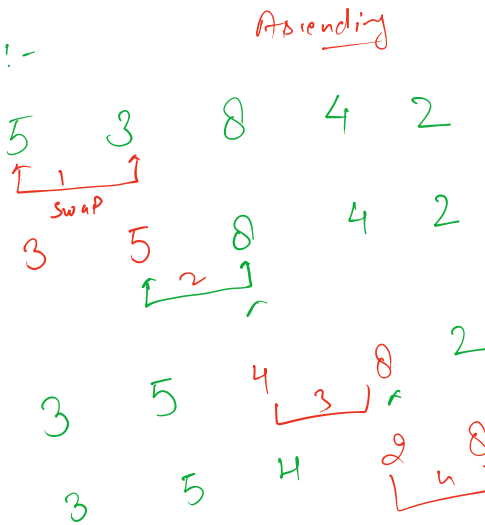
```
    int beg = 0;
    int end = numsSize - 1;
    int mid;
    while (beg <= end) {
        mid = (beg + end) / 2;
        if (target == nums[mid])
            return mid;
        else if (nums[mid] > target)
            end = mid - 1;
        else
            beg = mid + 1;
    }
    return -1;
}
```



Sequential Search — Complexity = $O(n)$
Binary Search — Complexity = $O(\log n)$

Bubble Sort :-

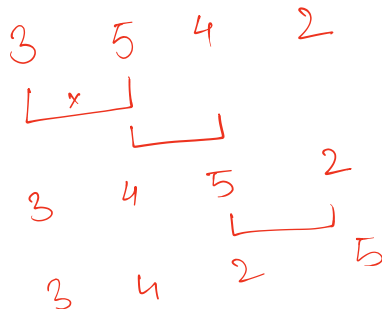
Pass 1



Size = 5
(4) (n-1)

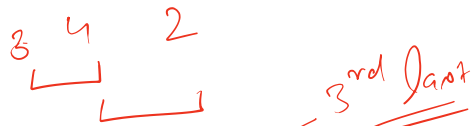
In first Pass
The largest element got
its position.

Pass 2



In 2nd Pass
2nd last element
Got its position

Pass 3

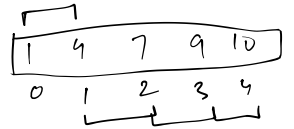


Pass 4

3 2 (4) ← --
 3 2
 └─┬─┘ 4th
 2 (3)

dthdthf

```
void bubbleSort(int* arr, int n) {
    int flag;
    for(int i=0; i<n-1; i++) {
        flag = 0;
        for(int j=0; j<n-1-i; j++) {
            if(arr[j] > arr[j+1]) {
                flag = 1;
                int temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
        if(!flag) break;
    }
}
```



flag = 0,
i = 0

```
void bubbleSort(int* arr, int n) {
    int flag;
    for(int i = 0; i < n-1; i++) {
        flag = 0;
        for(int j = 0; j < n-1-i; j++) {
            if(arr[j] > arr[j+1]) {
                flag = 1;
                int temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
        if(!flag)
            break;
    }
}
```

Selection Sort:-