SEARCHING

▼ 1. Linear Searching

In this algorithm we iterate through the array element by element starting from the 1st index.

```
#include <stdio.h>

int linearSearch(int arr[], int size, int key) {
  for (int i = 0; i < size; i++) {
    if (arr[i] == key) {
      return i;
    }
  }
  return -1;
}

int main() {
  int arr[] = {1, 3, 5, 7, 9};
  int key = 5;

  int index = linearSearch(arr, sizeof(arr) / sizeof(arr[0]),

  if (index == -1) {
    printf("Element not found.\n");
  } else {
    printf("Element found at index %d.\n", index);
  }

  return 0;
}
```

TIME COMPLEXITY:

- Best: O(1)
    - This is when the element to be found is at the start of the array.
- Worst: O(n)
    - This is when the element to be found is at the end of the array.
- Average: O(n/2)

SPACE COMPLEXITY:

O(1)

▼ 2. Binary Searching

In this searching algorithm, the array has to be sorted.

We keep on dividing the array into subarrays by comparing the element with the middle element until the ending index is greater than or equal to the starting index.

```
/*
Condition - Array has to be sorted
          - do until l <= r
          - starting and ending index shouldn't cross
*/

#include <stdio.h>


void printSearchSpace(int arr[], int s, int e){
    printf("The new search space from %d - %d find the array

    for(int i=s; i<e; i++){
        printf("%d ",arr[i]);
    }
    printf("\n");
}


int binarySearch(int arr[], int key, int start, int end){

    while(start <= end){

        printSearchSpace(arr,start,end);

        int mid = (start + end)/2;

        if(arr[mid] == key){
            return mid;
        }

        if(arr[mid] < key){
            start = mid+1;
        }else{
            end = mid-1;
        }
    }
    return -1;
}




int main() {

    int arr[] = {3,5,8,11,15,19,21,34,56};
    int n = sizeof(arr)/sizeof(arr[0]);

    int key;
    printf("Enter the key you want to search in the Array: ")
    scanf("%d",&key);
    printf("\n");

    int result = binarySearch(arr, key, 0, n);

    printf("\n");

    if(result==-1){
        printf("The key %d is not present in the array.\n",ke
    }else{
        printf("The key %d is present at the index %d.\n",key
    }


    return 0;
}
```

TIME COMPLEXITY:

- Best: O(1)
    - This is when the element to be found is at the middle of the array.
- Worst: O(log n)
    - This is when the element to be found is either at the start or end of the array.
- Average: O(log n)

SPACE COMPLEXITY:

O(1)