Binary Search

Definition:

Binary Search is an efficient algorithm for finding an item from a sorted list of items. It works by repeatedly dividing the search interval in half. If the value of the search key is less than the item in the middle of the interval, the search focuses on the left half; otherwise, it focuses on the right half. This process continues until the item is found or the interval is empty.

Key Characteristics:

- ★ *Requires Sorted Array:* Binary search operates on a sorted array.
- ★ *Divide and Conquer:* The algorithm repeatedly divides the array into halves.
- ★ *Time Complexity:* O(log n), where n is the number of elements in the array.
- ★ *Space Complexity:* O(1) for the iterative approach, O(log n) for the recursive approach due to the call stack.

How It Works:

- ★ **Initial Setup**: Define two pointers, start (beginning of the array) and end (end of the array).
- ★ **Middle Calculation**: Calculate the middle index: mid = start + (end - start) / 2.
- ★ **Comparison**:
  - ○ If the middle element is equal to the target value, return the middle index.
  - ○ If the middle element is less than the target value, adjust the start pointer to mid + 1.
  - ○ If the middle element is greater than the target value, adjust the end pointer to mid - 1.
- ★ **Repeat**: Continue the process until start exceeds end.

Example:

Find k.

```
int binarySearch(const vector<int> &arr, int k){

    int start = 0;

    int end = arr.size()-1;
```

```
    while (start <= end){
        int mid = start + (end - start) / 2;
        if (arr[mid] == k){
            return mid;
        }else if (arr[mid] < k){
            start = mid + 1;
        }else{
            end = mid - 1;
        }
    }
    return -1;
}
```

Advantages:

- ★ *Efficiency:* Significantly faster than linear search for large datasets.
- ★ *Simplicity*: Conceptually straightforward and easy to implement.

Disadvantages:

- ★ *Requires Sorted Array:* The array must be sorted beforehand.
- ★ *Not Suitable for Linked Lists:* Binary search is less effective for linked lists due to the lack of random access.

Use Cases:

- ★ *Large Datasets*: Particularly effective for searching in large sorted arrays.
- ★ *Database Indexing:* Commonly used in database indexing to speed up search queries.
- ★ *Real-time Systems:* Used in real-time systems where search speed is critical.

Conclusion:

Binary search is a powerful and efficient search algorithm when dealing with sorted arrays. Its logarithmic time complexity makes it ideal for large datasets, providing

quick and reliable search results. Understanding and implementing binary search is a fundamental skill for computer science and software engineering.
