

Searching Algorithms

by Dharmin Shah and Jeff Carter

CSSE 221-02

Fundamentals of Software Development
Honors

Rose-Hulman Institute of Technology

Why Search?

- To find an element in an list.
- For example, to find a criminal record, the FBI searches from a list of criminal records it has.
- A car dealer searches for a car from his inventory list.

Types of Searches

- There are two types of searches:
 1. Sequential or Linear Search.
 2. Binary Search
- If an array of elements is not ordered, and you want to find a particular element, you use Linear Search.
- If the array is a sorted one, and you want to find an element, using Binary Search is the faster way to search for the element.

Sequential (Linear) Search

- A sequential search of an array starts at the beginning (0th position) of the array and continues until the element that we are searching for is found, or the element is not in the array.
- Each element of the array is visited until the end of the array or until the element is found.

```
public class Search implements Comparable<E>{
```

```
    public int compareTo(E obj){  
        if(this.value == obj.value)  
            return 0;  
        else if(this.value > obj.value)  
            return 1;  
        else  
            return -1;  
    }
```

```
    public int LinearSearch(E[] arr, E element){  
        for(int i = 0; i < arr.length; i++){  
            if(arr[i].compareTo(element)==0)  
                return i;  
        }  
        return -1; }  
}
```

Efficiency of Linear Search

Case	Efficiency
Best Case (the item to be searched is the first one)	O(1)
Worst Case (the item to be searched is the last one)	O(n)
Average	O(n)

- The average number of comparisons for linear search is:

$$\frac{1+2+\dots+n}{n}$$

It is known
that

$$1+2+\dots+n = \frac{n(n+1)}{2}$$

$$\frac{1+2+\dots+n}{n} = \frac{1}{n} \frac{n(n+1)}{2} = \frac{n+1}{2}$$

Sequential (Linear) Search

- Advantages:
 - a. Easy to implement and understand.
 - b. The Array need not be ordered (sorted).
- Disadvantages:
 - a. Inefficient; its average efficiency is $O(n)$ and the efficiency for the worst case scenario is $O(n)$.

Binary Search

- A binary search uses the ‘divide-and-conquer’ strategy.
- It looks at the middle element of a sorted array, and then searches for the left half or the right half depending on the element being searched and the order in which the array was sorted.
- It will repeat the same process after selecting the appropriate half, until the element is found.

Binary Search (array sorted in increasing order)

```
public int BinarySearch(E[] arr, E element){
    int left = 0;
    int right = arr.length-1;
    while(left<=right){
        int middle = (left+right)/2;
        if(element.compareTo(arr[middle])>0){
            left = middle+1;
        } else if(element.compareTo(arr[middle])<0) {
            right = middle-1;
        } else if(element.compareTo(arr[middle])==0) {
            return middle;
        }
    }
    return -1;
}
```

Binary Search (array sorted in decreasing order)

```
public int BinarySearch(E[] arr, E element){
    int left = 0;
    int right = arr.length-1;
    while(left<=right){
        int middle = (left+right)/2;
        if(element.compareTo(arr[middle])>0) {
            right = middle-1;
        } else if(element.compareTo(arr[middle])<0) {
            left = middle+1;
        } else if(element.compareTo(arr[middle])==0) {
            return middle;
        }
    }
    return -1;
}
```

Efficiency Binary Search

11 items

1st try - 6 items

2nd try - 3 items

3rd try - 2 items

4th try - 1 item

32 items

1st try - 16 items

2nd try - 8 items

3rd try - 4 items

4th try - 2 items

5th try - 1 item

250 items

1st try - 125 items

2nd try - 63 items

3rd try - 32 items

4th try - 16 items

5th try - 8 items

6th try - 4 items

7th try - 2 items

8th try - 1 item

512 items

1st try - 256 items

2nd try - 128 items

3rd try - 64 items

4th try - 32 items

5th try - 16 items

6th try - 8 items

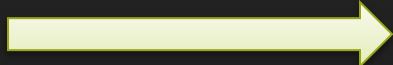
7th try - 4 items

8th try - 2 items

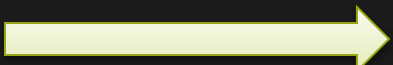
9th try - 1 item

Efficiency of Binary Search

- List of 11 takes 4 tries.
- List of 32 takes 5 tries.
- List of 250 takes 8 tries.
- List of 512 takes 9 tries.

• $8 < 11 < 16$  $2^3 < 11 < 2^4$

• $2^5 = 32$ and $2^9 = 512$

• $128 < 250 < 256$  $2^7 < 250 < 2^8$

Efficiency of Binary Search

- Since,

$$8 = 2^3 \longrightarrow \log_2 8 = 3.$$

- Therefore, we say that the binary search algorithm has an efficiency of $\log_2 n$ time.
- Thus, the efficiency of binary search is:

$$O(\log n)$$

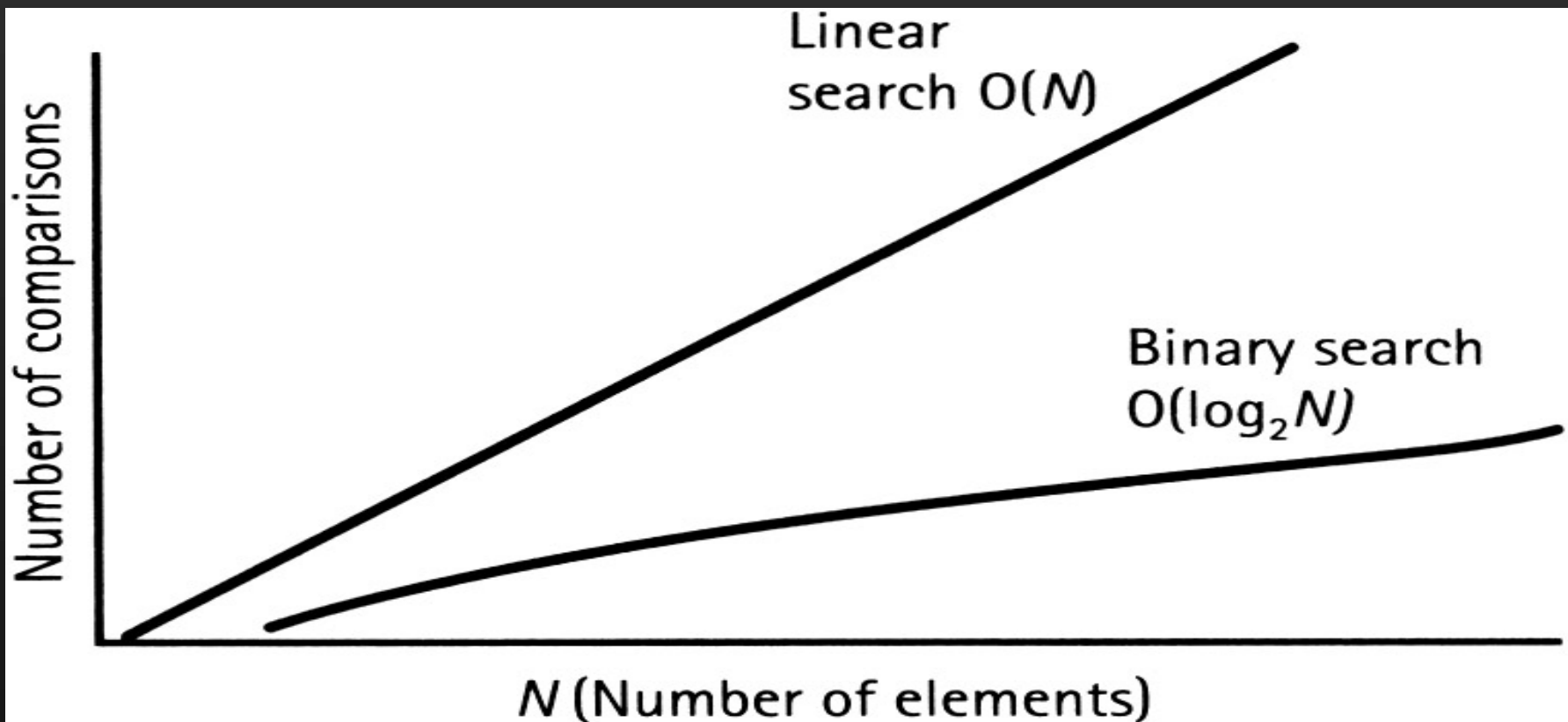
Binary Search

Advantage

More efficient than linear search. $O(\log n)$ efficiency compared to $O(n)$ efficiency of linear search.

Disadvantage

Requires a sorted array.





Demo