Searching Algorithms

MD. MUKTAR HOSSAIN

LECTURER

DEPT. OF CSE

VARENDRA UNIVERSITY

Searching

Searching algorithms are essential tools in computer science used to locate specific items within a collection of data.

There are so many searching algorithms and these are followings:

- Linear Search
- Binary Search
- Two Pointers Technique
- Interpolation Search
- Exponential Search
- Fibonacci Search etc.

Here, we'll discuss linear search and binary search.

Linear Search

Linear search algorithm, also known as sequential search algorithm, is a simple searching algorithm that traverses a list or array sequentially to find a target element.

Algorithm:

1. Start at the first element of the list.
2. Compare the target element with the current element.
3. If the current element matches the target, return the index of the element.
4. If the current element does not match, move to the next element.
5. Repeat steps 2–4 until the target is found or the list ends.
6. If the target is not found after checking all elements, return an indication that the target is not present (e.g., -1 or None).

Linear Search Cont'd

Let, arr[] = {12, 3, 43, 5, 67, 34, 23, 45, 21, 4}

and x = 5.

12	3	43	5	67	34	23	45	21	4

```
int LS(int arr[], int N, int x)
{
    for (int i = 0; i < N; i++)
        if (arr[i] == x)
            return i;
    return -1;
}
```

Linear Search Cont'd

x = 5
i = 0
arr[i] = 12

12	3	43	5	67	34	23	45	21	4

```
int LS (int arr[], int N, int x)
{
    for (int i = 0; i < N; i++)
        if (arr[i] == x)
            return i;
    return -1;
}
```

Linear Search Cont'd

x = 5
i = 0
arr[i] = 12

12	3	43	5	67	34	23	45	21	4

```
int LS (int arr[], int N, int x)
{
    for (int i = 0; i < N; i++)
        if (arr[i] == x)
            return i;
    return -1;
}
```

Linear Search Cont'd

x = 5
i = 1
arr[i] = 13

12	3	43	5	67	34	23	45	21	4

```
int LS (int arr[], int N, int x)
{
    for (int i = 0; i < N; i++)
        if (arr[i] == x)
            return i;
    return -1;
}
```

Linear Search Cont'd

x = 5
i = 1
arr[i] = 13

12	3	43	5	67	34	23	45	21	4

```
int LS (int arr[], int N, int x)
{
    for (int i = 0; i < N; i++)
        if (arr[i] == x)
            return i;
    return -1;
}
```

Linear Search Cont'd

x = 5
i = 2
arr[i] = 43

12	3	43	5	67	34	23	45	21	4

```
int LS (int arr[], int N, int x)
{
    for (int i = 0; i < N; i++)
        if (arr[i] == x)
            return i;
    return -1;
}
```

Linear Search Cont'd

x = 5
i = 2
arr[i] = 43

12	3	43	5	67	34	23	45	21	4

```
int LS (int arr[], int N, int x)
{
    for (int i = 0; i < N; i++)
        if (arr[i] == x)
            return i;
    return -1;
}
```

Linear Search Cont'd

x = 5
i = 3
arr[i] = 5

12	3	43	5	67	34	23	45	21	4

```
int LS (int arr[], int N, int x)
{
    for (int i = 0; i < N; i++)
        if (arr[i] == x)
            return i;
    return -1;
}
```

Linear Search Cont'd

x = 5
i = 3
arr[i] = 5

12	3	43	5	67	34	23	45	21	4

So, Location = 3

```
int LS (int arr[], int N, int x)
{
    for (int i = 0; i < N; i++)
        if (arr[i] == x)
            return i;
    return -1;
}
```

Linear Search Cont'd

Time Complexity:

Best Case: In the best case, the key might be present at the first index. So the best case complexity is *O(1)*

Worst Case: In the worst case, the key might be present at the last index i.e., opposite to the end from which the search has started in the list. So the worst-case complexity is *O(N)* where N is the size of the list.

Average Case: *O(N)*

Binary Search

Binary search is a search algorithm used to find the position of a target value within a sorted array.

Algorithm:

- Initialize two pointers: low (start of the list) and high (end of the list).
- While low is less than or equal to high:
  - Calculate the middle index:

$$mid = \frac{low + high}{2}$$

  - Compare the target with the middle element:
    - If the target matches the middle element, return mid.
    - If the target is less than the middle element, left half used for search by setting high = mid - 1.
    - If the target is greater than the middle element, right half used for search by setting low = mid + 1.
- If the target is not found, return an indication that the target is not present (e.g., -1 or None).

Binary Search Cont'd

Let, arr[] = {3, 4, 5, 12, 21, 23, 34, 43, 45, 67}

and x = 5.

3	4	5	12	21	23	34	43	45	67

```
int BS(int arr[], int L, int H, int x){
while (L <= H) {
        int mid = ( L + H) / 2;
if (arr[mid] == x)
            return mid;
if (arr[mid] < x)
            L = mid + 1;
else
            H = mid - 1;
    }
return -1;
}
```

Binary Search Cont'd

x = 5
L = 0, H = 9
mid= 4
arr[i] = 21

3	4	5	12	21	23	34	43	45	67

```
int BS(int arr[], int L, int H, int x){
while (L <= h) {
        int mid = L + (H - L) / 2;
if (arr[mid] == x)
            return mid;
if (arr[mid] < x)
            L = mid + 1;
else
            H = mid - 1;
      }
return -1;
}
```

Binary Search Cont'd

x = 5
L = 0, H = 3
mid= 4
arr[i] = 21

3	4	5	12	21	23	34	43	45	67

```
int BS(int arr[], int L, int H, int x){
while (L <= h) {
        int mid = L + (H - L) / 2;
if (arr[mid] == x)
            return mid;
if (arr[mid] < x)
            L = mid + 1;
else
            H = mid - 1;
    }
return -1;
}
```

Binary Search Cont'd

x = 5
L = 0, H = 3
mid= 1
arr[i] = 4

3	4	5	12	21	23	34	43	45	67

```
int BS(int arr[], int L, int H, int x){
while (L <= h) {
        int mid = L + (H - L) / 2;
if (arr[mid] == x)
            return mid;
if (arr[mid] < x)
            L = mid + 1;
else
            H = mid - 1;
    }
return -1;
}
```

Binary Search Cont'd

x = 5
L = 2, H = 3
mid= 1
arr[i] = 4

3	4	5	12	21	23	34	43	45	67

```
int BS(int arr[], int L, int H, int x){
while (L <= h) {
        int mid = L + (H - L) / 2;
if (arr[mid] == x)
            return mid;
if (arr[mid] < x)
            L = mid + 1;
else
            H = mid - 1;
    }
return -1;
}
```

Binary Search Cont'd

x = 5
L = 2, H = 3
mid= 2
arr[i] = 4

3	4	5	12	21	23	34	43	45	67

```
int BS(int arr[], int L, int H, int x){
while (L <= h) {
        int mid = L + (H - L) / 2;
if (arr[mid] == x)
            return mid;
if (arr[mid] < x)
            L = mid + 1;
else
            H = mid - 1;
    }
return -1;
}
```

Binary Search Cont'd

x = 5
L = 2, H = 3
mid= 2
arr[i] = 4

3	4	5	12	21	23	34	43	45	67

So, Location = 2

```
int BS(int arr[], int L, int H, int x){
while (L <= h) {
        int mid = L + (H - L) / 2;
if (arr[mid] == x)
            return mid;
if (arr[mid] < x)
            L = mid + 1;
else
            H = mid - 1;
    }
return -1;
}
```

Binary Search Cont'd

Time Complexity:

Best Case: $O(1)$

Worst Case: $O(\log N)$

Average Case: $O(\log N)$

Linear vs Binary

Home Task

1. Differentiate between Linear search and Binary search algorithm.

2. Advantages and Disadvantages of Linear search and Binary search algorithm.

Thank You