DET102 Data Structures and Algorithms

Lecture 05: Searching Algorithms

Searching Problem

- Given a set of data, decide whether there exists some data x.

- Searching method
  - Linear search
  - Binary search
  - Hashing

User doesn't exist ?

Wrong password

Linear Search

- Search from the beginning.
  - If found, find the location of x.
  - If not found, return some special value.

- Applied to all types of data.
- Time complexity: O(n)

Exercise 1 : Linear Search

- **Problem description**: You are given a sequence of *n* integers S and a sequence of different *q* integers T. Write a program which outputs C, the number of integers in T which are also in the set S.
- **Input:** In the first line *n* is given. In the second line, *n* integers are given. In the third line *q* is given. Then, in the fourth line, *q* integers are given.
- **Output:** Print C in a line.

Constraints
- $n \leq 10000$
- $q \leq 500$
- $0 \leq$ an element in S $\leq 10^9$
- $0 \leq$ an element in T $\leq 10^9$

Source: https://onlinejudge.u-aizu.ac.jp/problems/ALDS1_4_A

Sample Input 1

```
5
1 2 3 4 5
3
3 4 1
```

Sample Output 1

```
3
```

Sample Input 2

```
3
3 1 2
1
5
```

Sample Output 2

```
0
```

Sample Input 3

```
5
1 1 2 2 3
2
1 2
```

Sample Output 3

```
2
```

Binary Search

- Given a **sorted** list and an item x, check whether x exists in the list.

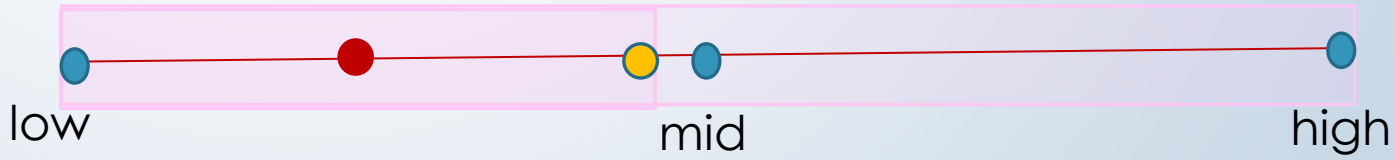
1, 3, 5, 7, 13, 18, 37, 65

1. Find the middle one
2. Compare x with the middle one
3. Ignore half of the elements

http://www.cs.armstrong.edu/liang/animation/web/BinarySearch.html

Binary Search

- We consider three cases:
  - If the target equals data[mid], then we have found the item we are looking for, and the search terminates successfully.
  - If target < data[mid], then we recur on the first half of the sequence, that is, on the interval of indices from low to mid−1.
  - If target > data[mid], then we recur on the second half of the sequence, that is, on the interval of indices from mid+1 to high.
- An unsuccessful search occurs if low > high, as the interval [low,high] is empty.



mid=(low+high)/2

floor function

mid=(low+high)/2 ceiling function

Binary Search implementation

- Iterative binary search
  - stop when the key is found or having looped for enough rounds.
- Recursive binary search
  - stop when the key is found or low>high

https://www.geeksforgeeks.org/binary-search/

Binary Search

- Complexity:
  - consider an array with n elements,
    - Find the middle one
    - Compare it with the target
    - Remove a half

    } O(1)
  - Consider an array with n/2 elements,
  - Consider an array with n/4 elements,
  - …
  - Consider an array with 1 element

} $(\text{Log}_2 n + 1)$ times

Pseudo code: 1

```
int BinarySearch(A[], low, high,key){
    mid=(low+high)/2;
    if(low>high)
        return -1;      // key is not found.
    if(A[mid]==key)
        return mid;   //find now
    if(A[mid]<key)   //search the left part
        return BinarySearch(A, low, mid-1, key);
    if(A[mid]>key)   //search the right part
        return BinarySearch(A, mid+1,high, key);
}
```

Pseudo code: 2

```
int BinarySearch(A[], low, high, key){
    int mid;
    while(low<=high){
        mid=(low+high)/2;
        if(key==A[mid])
            return mid;    //return the index of the key
        if(key<A[mid])
            high=mid-1;
        if(key>A[mid])
            low=mid+1;
    }
    return -1;              // not found, return an invalid index
}
```

Time complexity

- Let T(n) be the time used to do binary search on an array with n items.

```
int BinarySearch(A[], low, high,key){
      mid=(low+high)/2;
      if(low>high)
            return 0;            // key is not found.
      if(A[mid]==key)
            return 1;   //find now
      if(A[mid]<key)  //search the left part
            return BinarySearch(A, low, mid-1, key);
      if(A[mid]>key)  //search the right part
            return BinarySearch(A, mid+1,high, key);
}
```

$T(n)= T(n/2) + c$
$T(1)=O(1)$

Time complexity

$T(n)= T(n/2) + c$

$T(1)=O(1)$

$T(n)=O(\log_2 n)$

$T(n)=T(n/2)+c$

$=T(n/4)+ c + c$

$=T(n/8)+ c + c + c$

....

$=T(n/2^k)+ k*c$

If $k=\log_2 n$, then $2^k=n$, and $n/2^k=1$.

Therefore, $T(n)=T(1)+k*c = O(\log_2 n)$ where $k=\log_2 n$

Exercise 2： Binary Search

- **Problem description**: You are given a sequence of *n* integers S and a sequence of different *q* integers T. Write a program which outputs C, the number of integers in T which are also in the set S.
- **Input:** In the first line *n* is given. In the second line, *n* integers are given. In the third line *q* is given. Then, in the fourth line, *q* integers are given.
- **Output:** Print C in a line.

Constraints

- Elements in S is sorted in ascending order
- $n \leq 100000$
- $q \leq 50000$
- $0 \leq$ an element in S $\leq 10^9$
- $0 \leq$ an element in T $\leq 10^9$

Source: https://onlinejudge.u-aizu.ac.jp/problems/ALDS1_4_B

Sample Input 1

```
5
1 2 3 4 5
3
3 4 1
```

Sample Output 1

```
3
```

Sample Input 2

```
3
1 2 3
1
5
```

Sample Output 2

```
0
```

Sample Input 3

```
5
1 1 2 2 3
2
1 2
```

Sample Output 3

```
2
```

Compare

Number of items	Linear Search	Binary Search
100	100	7
10 000	10 000	14
1 000 000	1 000 000	20
	O(n)	O(logn)