CSC 211: Computer Programming
Binary Search

Michael Conti

Department of Computer Science and Statistics
University of Rhode Island

Spring 2022

THINK BIG WE DO™

Linear search

· Two fundamental problems in CS

  ✓ Sorting and Searching

· **Linear (sequential) search** is a method for finding a **value** within a sequence

· A naive solution works by sequentially checking each element until a match is found

  ✓ it also stops when there are no more elements to check

  ✓ performs at most **n** comparisons for sequences of length **n**

  ✓ considered slow for finding elements in collections of data

```
bool lin_search(int *A, int key, int len) {

    for (int i = 0 ; i < len; i++){

        if (A[i] == key){

            return true;

        }
    }

    return false;
}
```
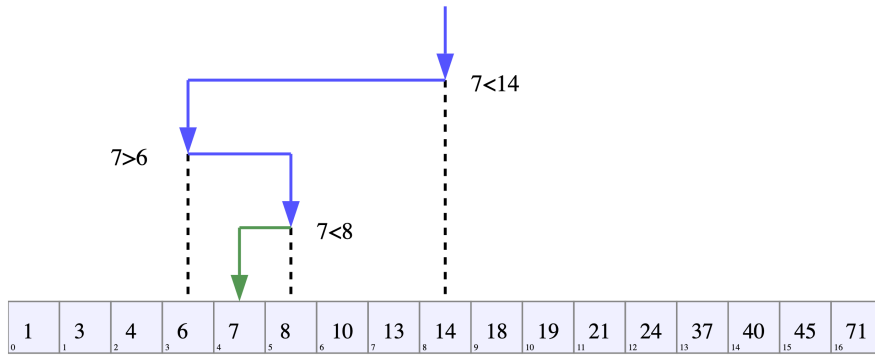
Binary search

· Search algorithm to find a giving value within a **sorted array**

· The algorithm compares the value to the middle element

  ✓ if they are not equal, one of the half is eliminated and the search continues on the other half

  ✓ repeat until value is found or no more elements are left (value is not in the array)

· Binary search is **faster than linear search**

Binary search

Looking for 7

7<14

7>6

7<8

1	3	4	6	7	8	10	13	14	18	19	21	24	37	40	45	71
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Binary Search

0	1	2	3	4	5	6	7	8	9	10	11	12
1	2	5	10	15	20	22	30	35	40	43	48	51

low   high

k = 48?

Binary Search

0	1	2	3	4	5	6	7	8	9	10	11	12
1	2	5	10	15	20	22	30	35	40	43	48	51

low   mid   high

k = 48?

Binary Search

0	1	2	3	4	5	6	7	8	9	10	11	12
1	2	5	10	15	20	22	30	35	40	43	48	51

low   high

k = 48?

Binary Search

0	1	2	3	4	5	6	7	8	9	10	11	12
1	2	5	10	15	20	22	30	35	40	43	48	51

low   mid   high

k = 48?

Binary Search

0	1	2	3	4	5	6	7	8	9	10	11	12
1	2	5	10	15	20	22	30	35	40	43	48	51

low mid high

↵

k = 48?

Binary Search

0	1	2	3	4	5	6	7	8	9	10	11	12
1	2	5	10	15	20	22	30	35	40	43	48	51

low   high

k = 22?

k = 0?

k = 55?

```c
// returns index of element k in A
// returns NOT_FOUND if element not in A
int bin_search(int *A, int lo, int hi, int k) {
    // base case
    if (hi < lo) {
        return NOT_FOUND;
    }
    // calculate midpoint index
    int mid = lo + ((hi-lo)/2);
    // key found?
    if (A[mid] == k) {
        return mid;
    }
    // key in upper subarray?
    if (A[mid] < k) {
        return bin_search(A, mid+1, hi, k);
    }
    // key is in lower subarray?
    return bin_search(A, lo, mid-1, k);
}
```

Call stack

0	1	2	3	4	5	6	7	8	9
1	2	5	10	15	20	22	30	35	40

```c
#define NOT_FOUND -1

int bsch(int *A, int lo, int hi, int k) {
    if (hi < lo) {
        return NOT_FOUND;
    }
    int mid = lo + ((hi-lo)/2);
    if (A[mid] == k) {
        return mid;
    }
    if (A[mid] < k) {
        return bsch(A, mid+1, hi, k);
    }
    return bsch(A, lo, mid-1, k);
}

int main() {
    int arr[] = {1,2,5,10,15,20,22,30,35,40};
    int idx = bsch(arr, 0, 9, 1);
}
```

https://bit.ly/36cEQWK

13

Google Research Blog

https://research.googleblog.com/2006/06/extra-extra-read-all-about-it-nearly.html

The latest news from Research at Google

"The version of binary search that I wrote for the JDK (java.util.Arrays) contained the same bug. It was reported to Sun recently when it broke someone's program, after lying in wait for nine years or so."

Extra, Extra - Read All About It: Nearly All Binary Searches and Mergesorts are Broken

Friday, June 02, 2006

Posted by Joshua Bloch, Software Engineer

overflow

```c
int mid = (low + high) / 2;
```

```c
int mid = lo + ((hi-lo) / 2);
```  👍

14

Find peak in unimodal arrays

Unimodal arrays

- An array is (**strongly**) **unimodal** if it can be split into an increasing part followed by a decreasing part

1	2	5	16	20	18	17	16	15	12	10	8	5

- An array is (**weakly**) **unimodal** if it can be split into a nondecreasing part followed by a nonincreasing part

1	2	5	5	15	20	22	22	35	38	13	8	5

16

Find the peak (strongly unimodal)

1	2	5	8	15	20	22	20	15	12	10	8	5
1	2	5	8	15	20	22	20	15	12	10	8	5
1	2	5	8	15	20	22	20	15	12	10	8	5
1	2	5	8	15	20	22	20	15	12	10	8	5
1	2	5	8	15	20	22	20	15	12	10	8	5
1	2	5	8	15	20	22	20	15	12	10	8	5
1	2	5	8	15	20	22	20	15	12	10	8	5
1	2	5	8	15	20	22	20	15	12	10	8	5

Find the peak (weakly unimodal)

1	2	5	5	15	20	22	22	35	38	13	8	5
1	2	5	5	15	20	22	22	35	38	13	8	5
1	2	5	5	15	20	22	22	35	38	13	8	5

Two recursive calls

More recursive examples …

Print all binary strings of length N

‣ How many strings should be printed? What is the algorithm?

Print all strings of length N made up of 'abcd's

- How many strings should be printed? What is the algorithm?