Recursion

Binary Search Algorithm

Lecture Contents

- Linear Search
- Binary Search
  - Details
  - Algorithm
- Big O Complexity (just a mention)

Linear Search

- Linear search is “brute force”
- Just “walk” through the array until we find the target
- Simple, easy… and maybe best if the array is not already sorted

```
public static int search(int[] a, int target) {
    for (int i=0; i<a.length; i++) {
        if (a[i] == target)
            return i;
    }
    return -1;
}
```

Binary Search

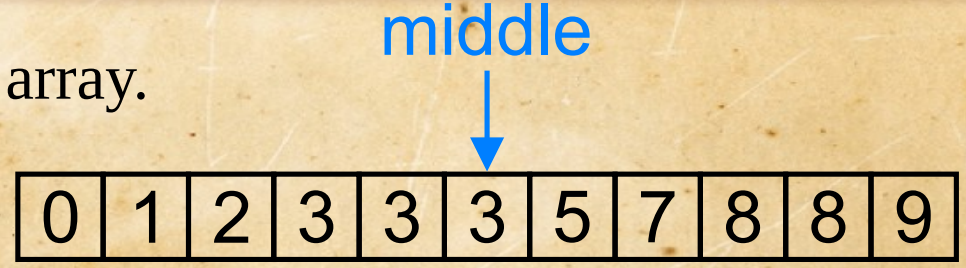
- Details
  - Requires a sorted list
  - Very efficient
    - Divide and conquer

Binary Search

- Find where the value 5 is in our array.

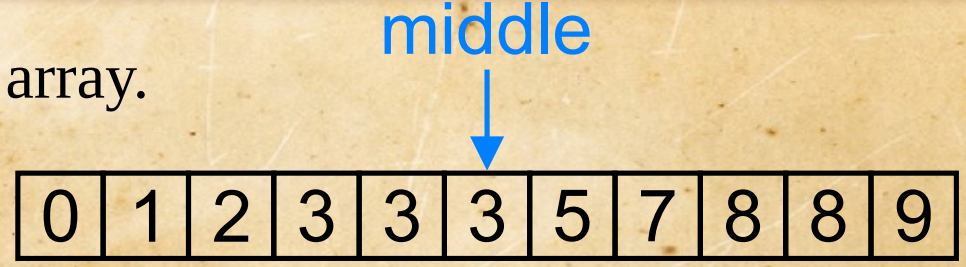
Binary Search

- Find where the value 5 is in our array.
- Algorithm:
  - Find the middle
    - how?



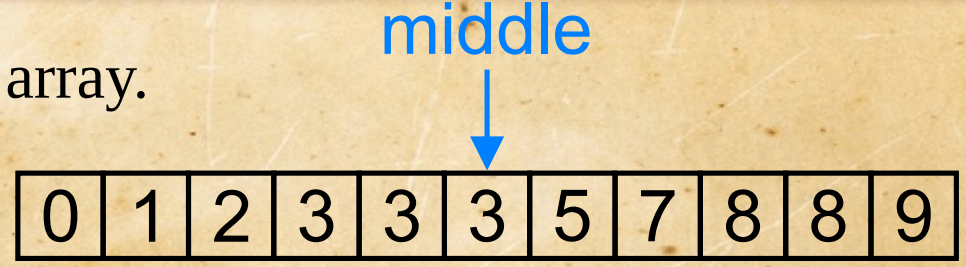
Binary Search

- Find where the value 5 is in our array.
- Algorithm:
  - Find the middle
    - `a.length / 2`
    -



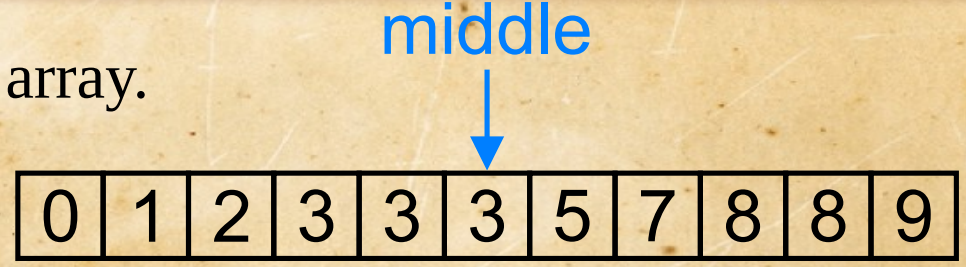
Binary Search

- Find where the value 5 is in our array.
- Algorithm:
  - Find the middle
    - `a.length / 2`
  - Is the middle our target?



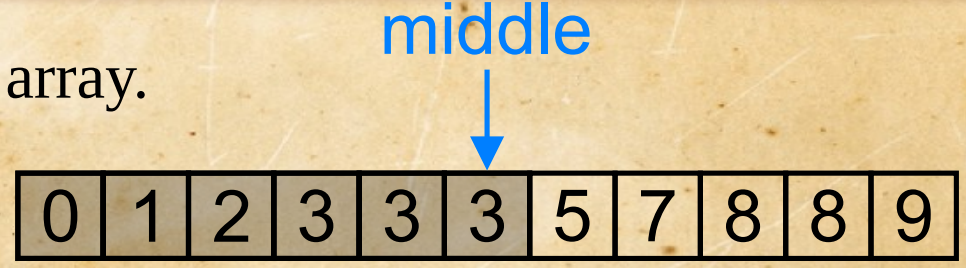
Binary Search

- Find where the value 5 is in our array.
- Algorithm:
  - Find the middle
    - `a.length / 2`
  - Is the middle our target?
    - No, so where do we search?



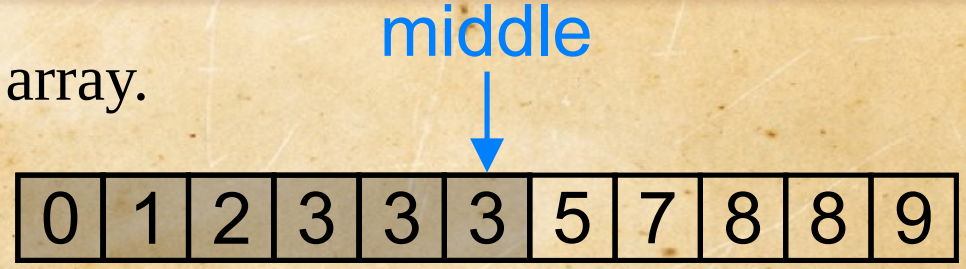
Binary Search

- Find where the value 5 is in our array.
- Algorithm:
  - Find the middle
    - `a.length / 2`
  - Is the middle our target?
    - No, so where do we search?



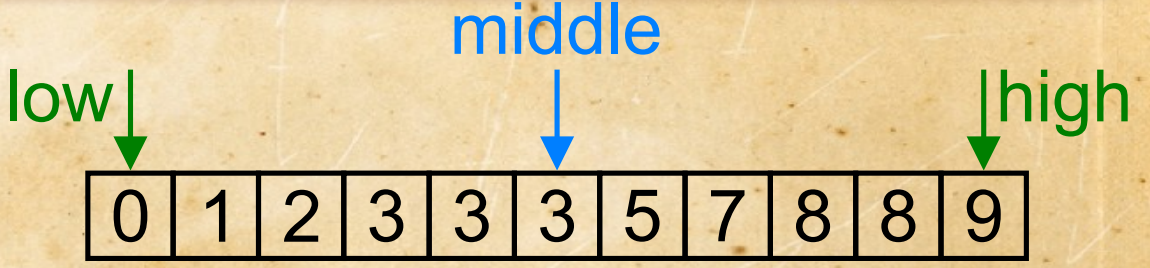
Binary Search

- Find where the value 5 is in our array.
- Algorithm:
  - Find the middle
    - `a.length / 2`
  - Is the middle our target?
    - No, so where do we search?
  - Hmmm… it seems we're going to need to keep track of the bounds of the array that still needs to be searched.



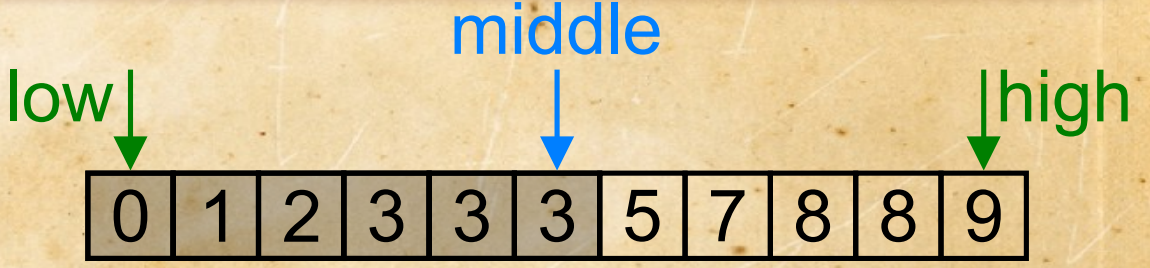
Binary Search

- Find where the value 5 is.
- Algorithm:
  - Find the middle
    - `a.length / 2`
  - Is the middle is the target?



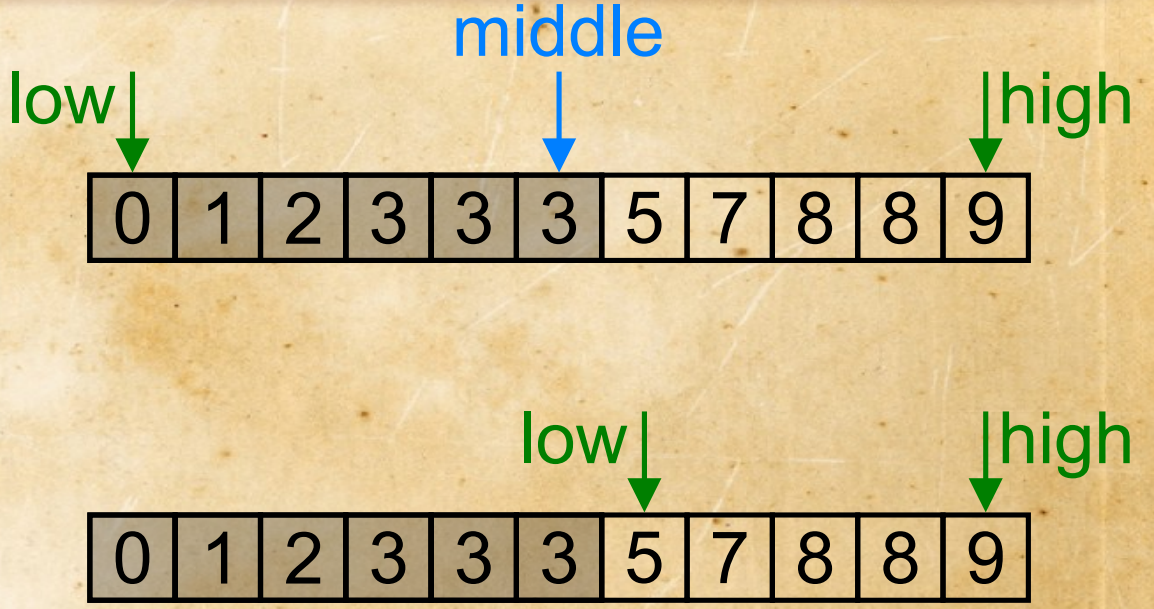
Binary Search

- Find where the value 5 is.
- Algorithm:
  - Find the middle
    - `a.length / 2`
  - Is the middle is the target?



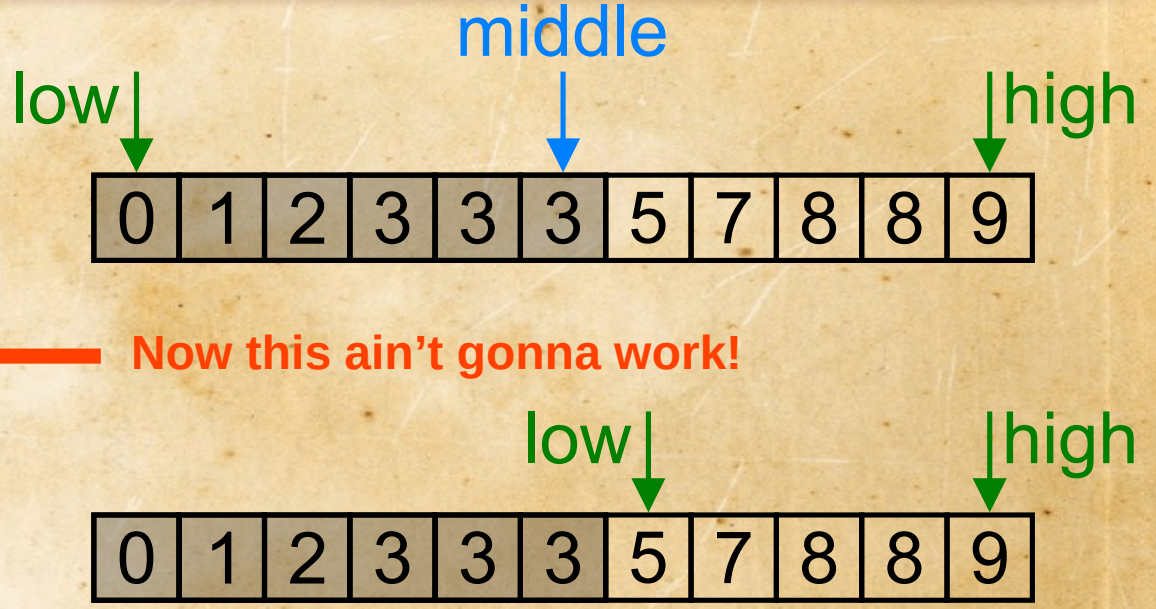
Binary Search

- Find where the value 5 is.
- Algorithm:
  - Find the middle
    - `a.length / 2`
  - Is the middle is the target?



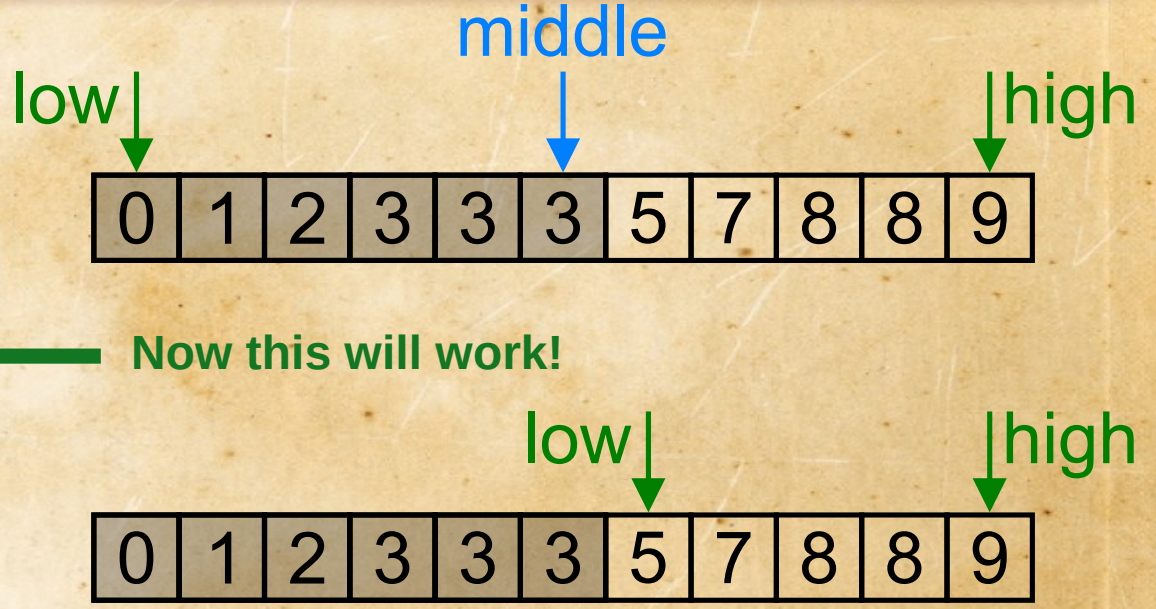
Binary Search

- Find where the value 5 is.
- Algorithm:
  - Find the middle
    - `a.length / 2`
  - Is the middle is the target?



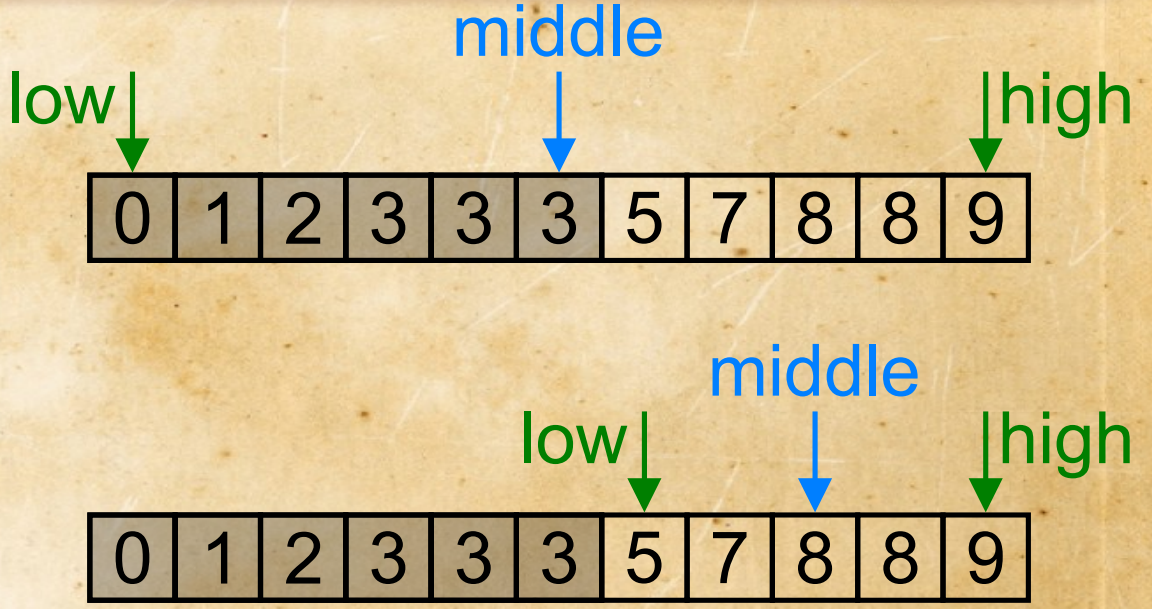
Binary Search

- Find where the value 5 is.
- Algorithm:
  - Find the middle
    - `(high+low) / 2`
  - Is the middle is the target?



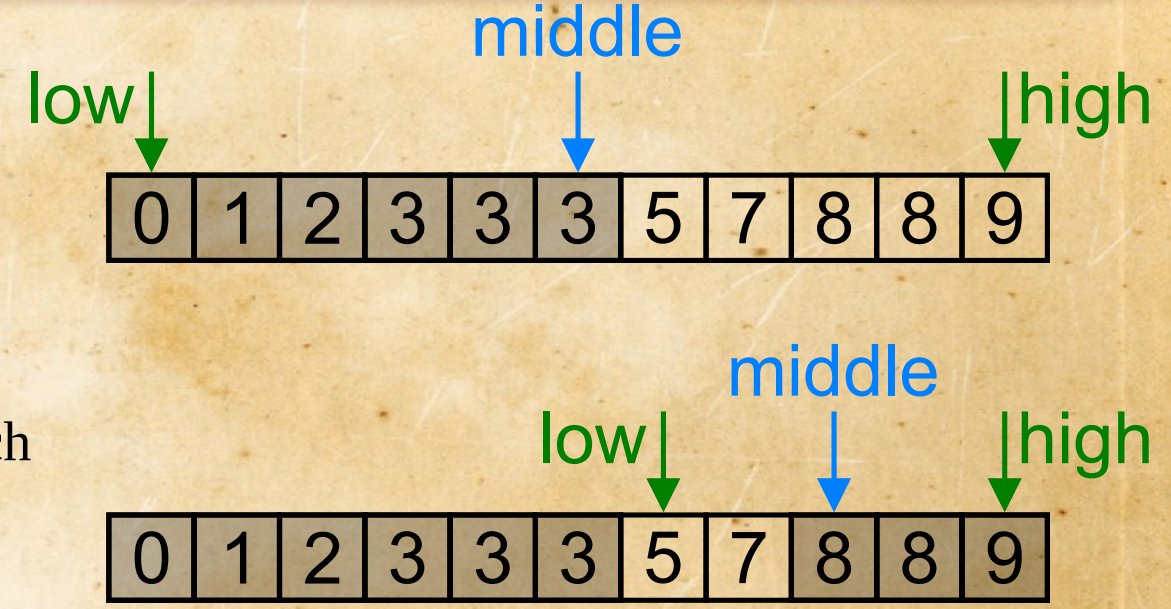
Binary Search

- Find where the value 5 is.
- Algorithm:
  - Find the middle
  - Is the middle is the target?



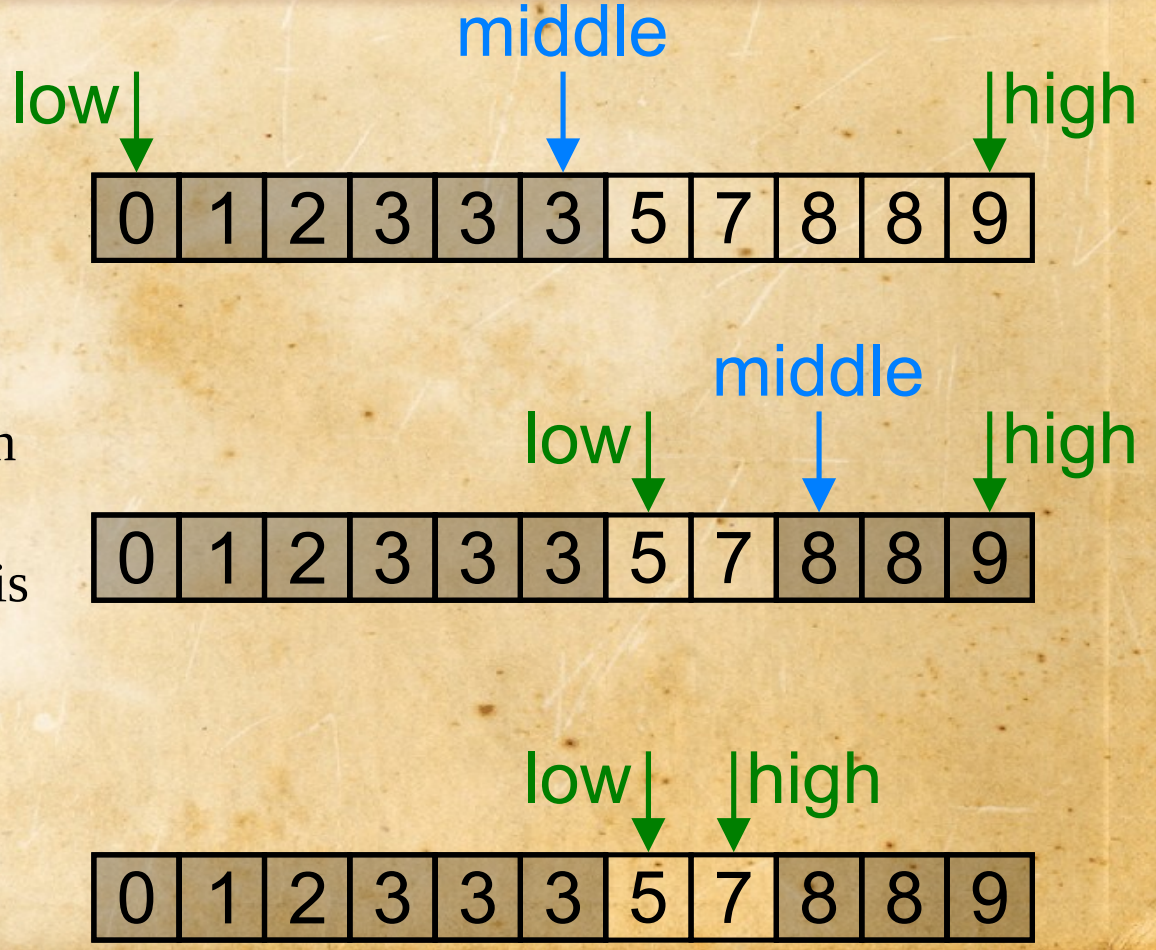
Binary Search

- Find where the value 5 is.
- Algorithm:
  - Find the middle
  - Is the middle is the target?
    - Nope; this time we search the lower half



Binary Search

- Find where the value 5 is.
- Algorithm:
  - Find the middle
  - Is the middle is the target?
    - Nope; this time we search the lower half
    - Where's the middle of this one?

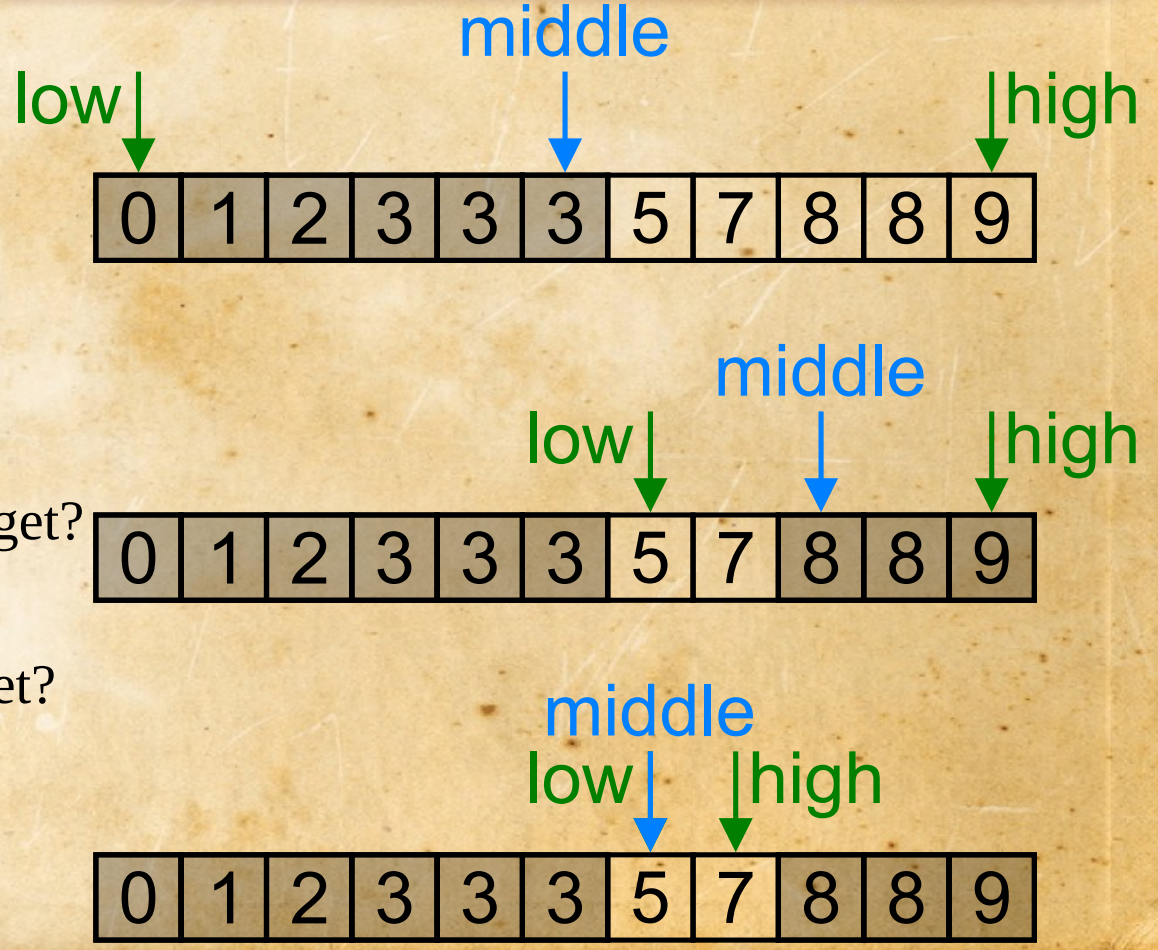


Binary Search

- Find where the value 5 is.
- Algorithm:
  - Find the middle
  - Is the middle is the target?
    - Nope; this time we search the lower half
    - Where's the middle of this one?

Binary Search

- Find where the value 5 is.
- Algorithm:
  - Find the middle
  - Is the middle is the target?
    - If yes, return index
  - Is the middle greater than target?
    - If yes, search lower
  - Is the middle lesser than target?
    - If yes, search upper



Binary Search

- Find where the value 5 is.
- Algorithm:
  - **If no array left, return -1**
  - Find the middle
  - Is the middle is the target?
    - If yes, return index
  - Is the middle greater than target?
    - If yes, search lower
  - Is the middle lesser than target?
    - If yes, search upper

Binary Search

- Find where the value 5 is.
- Algorithm:
  - If no array left, return `-1`
  - Find the middle
  - Is the middle is the target?
    - If yes, return index
  - Is the middle greater than target?
    - If yes, search lower
  - Is the middle lesser than target?
    - If yes, search upper

Binary Search

- What's the longest a ***linear search*** would take to search an array with ***n*** elements in it?

Binary Search

- What's the longest a linear search would take to search an array with ***n*** elements in it?
  - ***n*** comparisons

Binary Search

- What's the longest a linear search would take to search an array with ***n*** elements in it?
  - ***n*** comparisons
- What's the longest a binary search would take to search a sorted array with ***n*** elements in it?

Binary Search

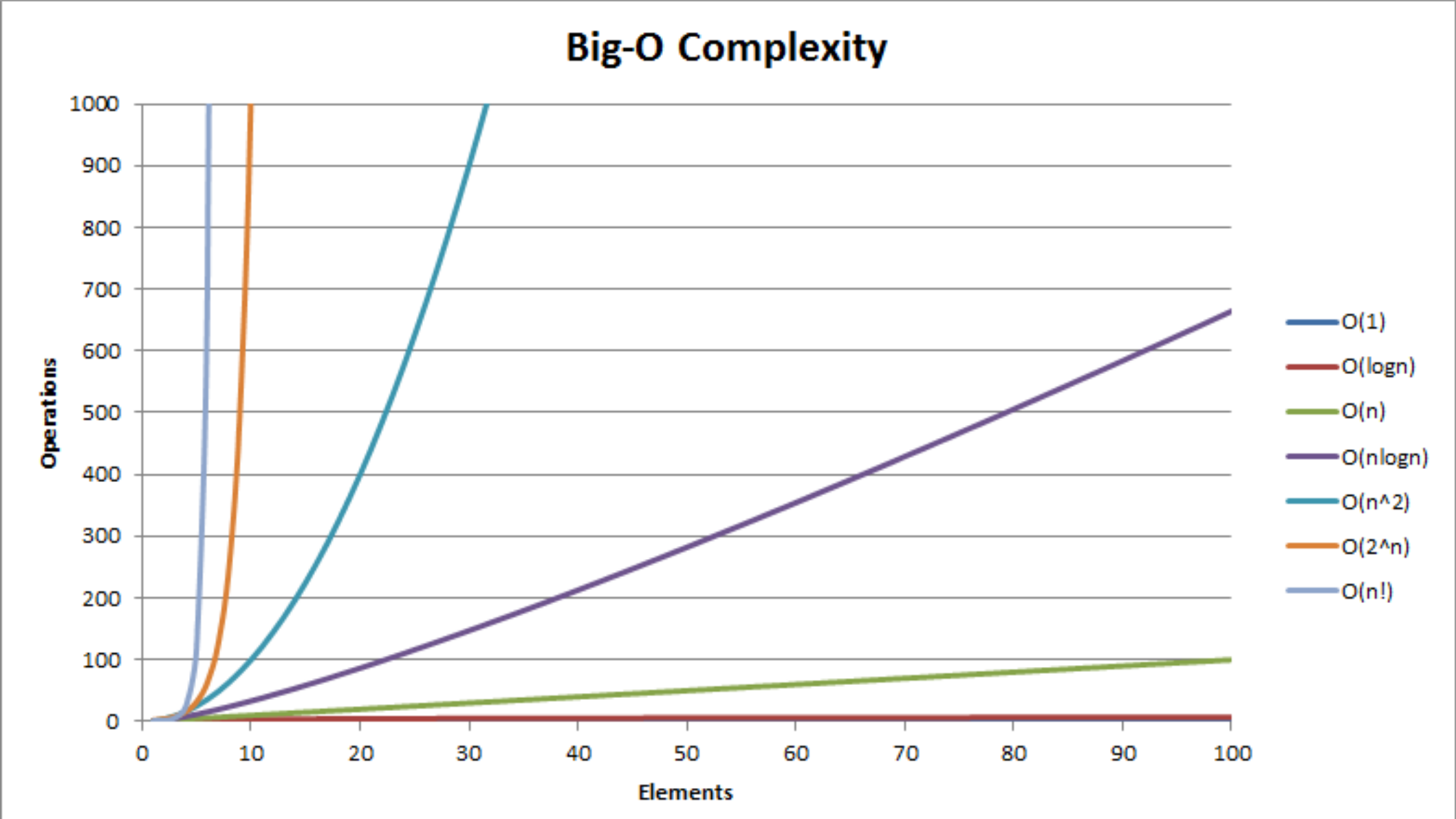
- What's the longest a linear search would take to search an array with ***n*** elements in it?
  - ***n*** comparisons
- What's the longest a binary search would take to search a sorted array with ***n*** elements in it?
  - **log *n*** comparisons

Binary Search

- What's the longest a linear search would take to search an array with ***n*** elements in it?
  - ***n*** comparisons
- What's the longest a binary search would take to search a sorted array with ***n*** elements in it?
  - **log *n*** comparisons
- In computer science, algorithm efficiency is discussed using "***Big O Notation***"

Big O Notation

$O(n)$	linear	Finding an item in an unsorted list or in an unsorted array; adding two *n*-bit integers by ripple carry
$O(n \log^* n)$	*n* log-star *n*	Performing triangulation of a simple polygon using Seidel's algorithm, or the union–find algorithm. Note that $\log^*(n) = \begin{cases} 0, & \text{if } n \leq 1 \\ 1 + \log^*(\log n), & \text{if } n > 1 \end{cases}$
$O(n \log n) = O(\log n!)$	linearithmic, loglinear, quasilinear, or "*n* log *n*"	Performing a fast Fourier transform; fastest possible comparison sort; heapsort and merge sort
$O(n^2)$	quadratic	Multiplying two *n*-digit numbers by schoolbook multiplication; simple sorting algorithms, such as bubble sort, selection sort and insertion sort; (worst-case) bound on some usually faster sorting algorithms such as quicksort, Shellsort, and tree sort


Big-O Complexity
Operations
1000
900
800
700
600
500
400
300
200
100
0
0
10
20
30
40
50
60
70
80
90
100
Elements
O(1)
O(logn)
O(n)
O(nlogn)
O(n^2)
O(2^n)
O(n!)

Binary Search

- Find where the value 5 is.
- Algorithm:
  - If no array left, return `-1`
  - Find the middle
  - Is the middle is the target?
    - If yes, return index
  - Is the middle greater than target?
    - If yes, search lower
  - Is the middle lesser than target?
    - If yes, search upper

Recursion

Binary Search Algorithm