NOW THAT THE ELEMENTS ARE SORTED IN THE ORDER WE WANT....

HOW DO WE SEARCH A SORTED LIST?

HOW DO WE SEARCH A SORTED LIST?

A NAIVE WAY TO SEARCH
FOR AN ELEMENT IN A LIST
IS TO CHECK EVERY ELEMENT
TILL WE FIND THE RIGHT ONE

THIS WOULD WORK FOR
BOTH SORTED AND
UNSORTED LISTS AND IS O(N)

FOR A SORTED LIST,
HOWEVER, WE OUGHT TO DO
MUCH BETTER!

BINARY SEARCH

CHOOSE AN ELEMENT IN AT THE MID-POINT OF A SORTED LIST

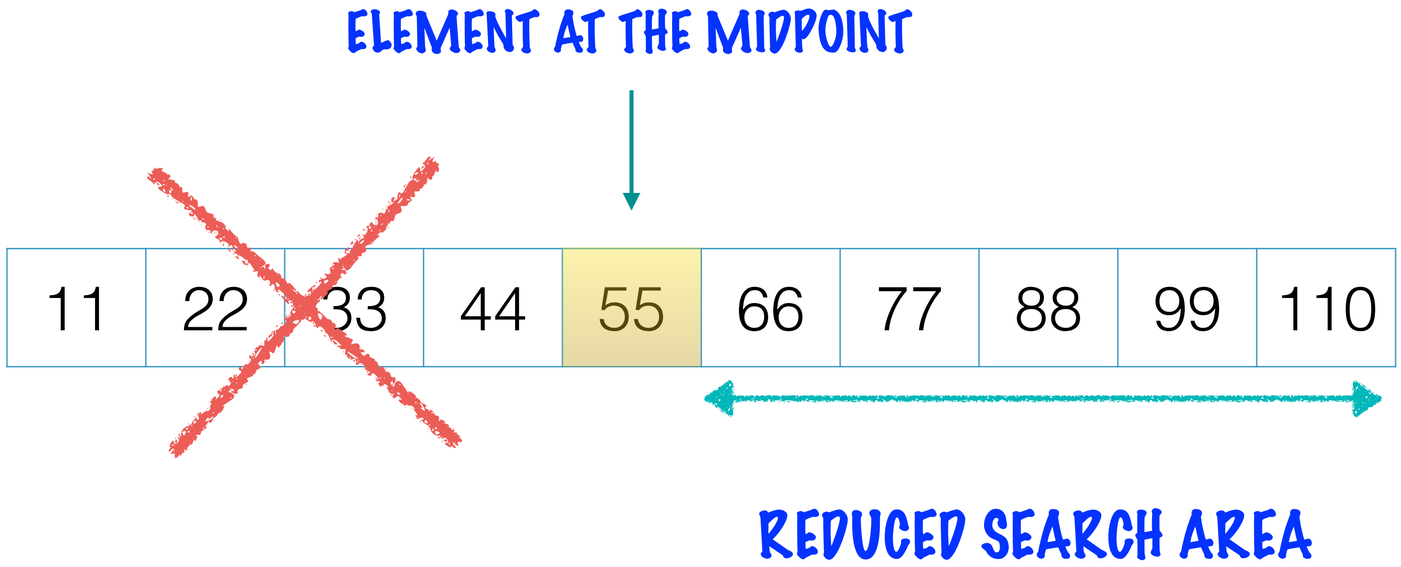
CHECK WHETHER IT'S SMALLER THAN OR GREATER THAN THE ELEMENT YOU ARE LOOKING FOR

IF THE ELEMENT AT THE MID-POINT IS LARGER THAN THE ELEMENT YOU ARE SEARCHING FOR

HALVE THE PORTION OF THE LIST YOU NEED TO SEARCH BY ONLY CONSIDERING ELEMENTS BEFORE THE MIDPOINT

BINARY SEARCH

SEARCHING FOR: 99

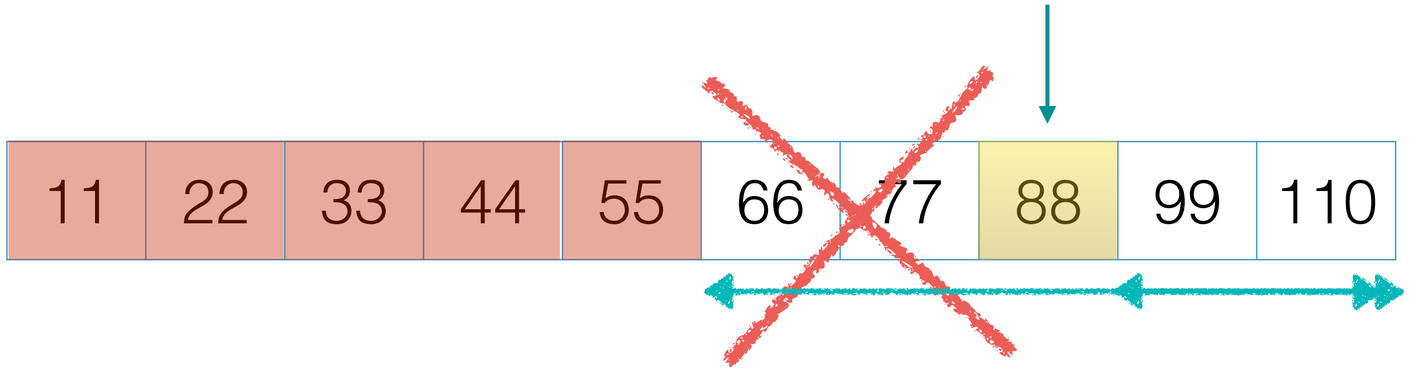


99 > 55 SO DISCARD THE FIRST HALF OF THE LIST

BINARY SEARCH

SEARCHING FOR: 99

ELEMENT AT THE MIDPOINT

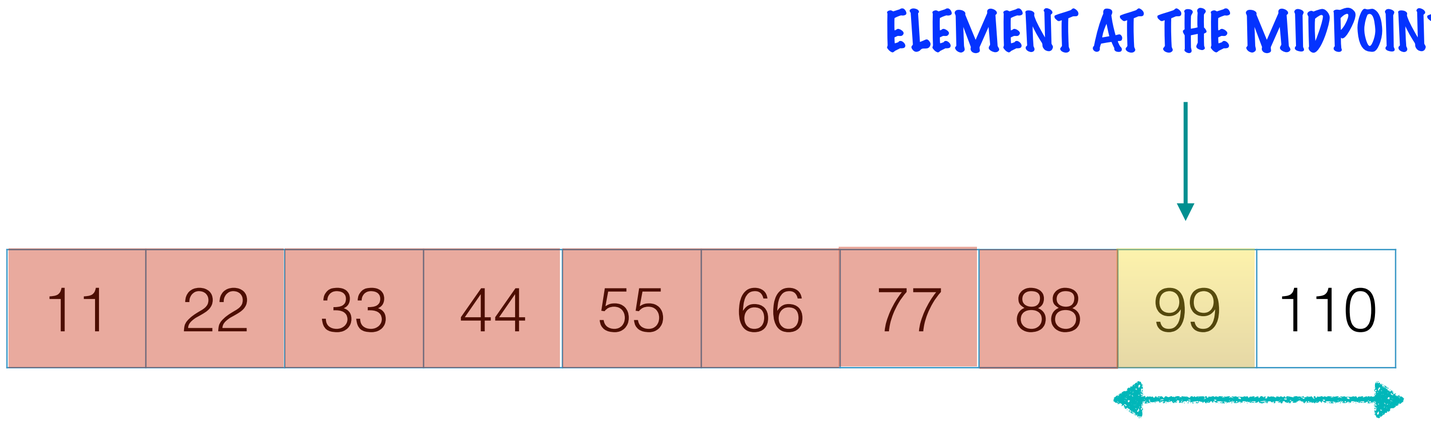


99 > 88 SO DISCARD THE FIRST HALF OF THE REDUCED LIST

REDUCED SEARCH AREA

BINARY SEARCH

SEARCHING FOR: 99



THE ELEMENT AT THE MIDPOINT IS THE ONE WE'RE LOOKING FOR, OUR SEARCH HAS ENDED

BINARY SEARCH - CODE

```
public static int binarySearch(int[] sortedList, int number) {
    int min = 0;
    int max = sortedList.length - 1;

    while (min <= max) {
        int mid = min + (max - min) / 2;
        if (sortedList[mid] == number) {
            return mid;
        }
        if (sortedList[mid] > number) {
            max = mid - 1;
        } else {
            min = mid + 1;
        }
    }

    return -1;
}
```

THE SEARCH KEY, THE NUMBER WE'RE LOOKING FOR

START THE SEARCH CONSIDERING THE ENTIRE LIST

FIND THE MID-POINT OF THE LIST. MIN AND MAX REPRESENT THE PORTION OF THE LIST TO SEARCH

IF THE ELEMENT WE'RE SEEKING IS AT THE MID-POINT RETURN THE INDEX

HALVE THE PORTION OF THE LIST WE'RE SEARCHING BASED ON WHETHER THE ELEMENT WILL BE FOUND IN THE FIRST OR SECOND HALF OF THE LIST

BINARY SEARCH

BY HALVING THE SEARCH AREA AT EVERY STEP, BINARY SEARCH WORKS MUCH FASTER THAN LINEAR SEARCH

THE COMPLEXITY OF BINARY SEARCH IS O(LOG N)