RECURSION IS HARD...

ESPECIALLY AT THE VERY BEGINNING

HOWEVER, IF YOU PRACTICE ENOUGH, YOU CAN DEVELOP A "RECURSIVE SENSE"

THIS WILL HELP IDENTIFY PROBLEMS WHICH LEND THEMSELVES WELL TO RECURSIVE SOLUTIONS

SO LET'S PRACTICE RECURSION!

BINARY SEARCH

WE'VE SEEN AN ITERATIVE IMPLEMENTATION, NOW LET'S WRITE CODE TO IMPLEMENT BINARY SEARCH RECURSIVELY

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

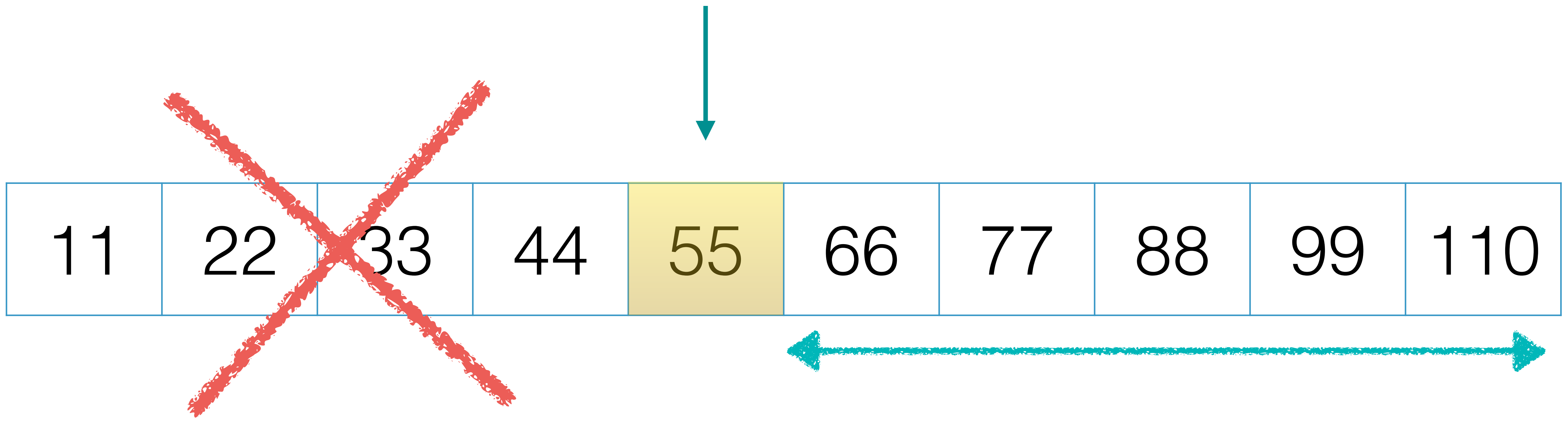
REDUCE YOUR SEARCH AREA TO THE PORTION WHERE THE ELEMENT MIGHT BE PRESENT

CALL BINARY SEARCH ON THAT PORTION!

BINARY SEARCH

SEARCHING FOR: 99

ELEMENT AT THE MIDPOINT



11	22	33	44	55	66	77	88	99	110

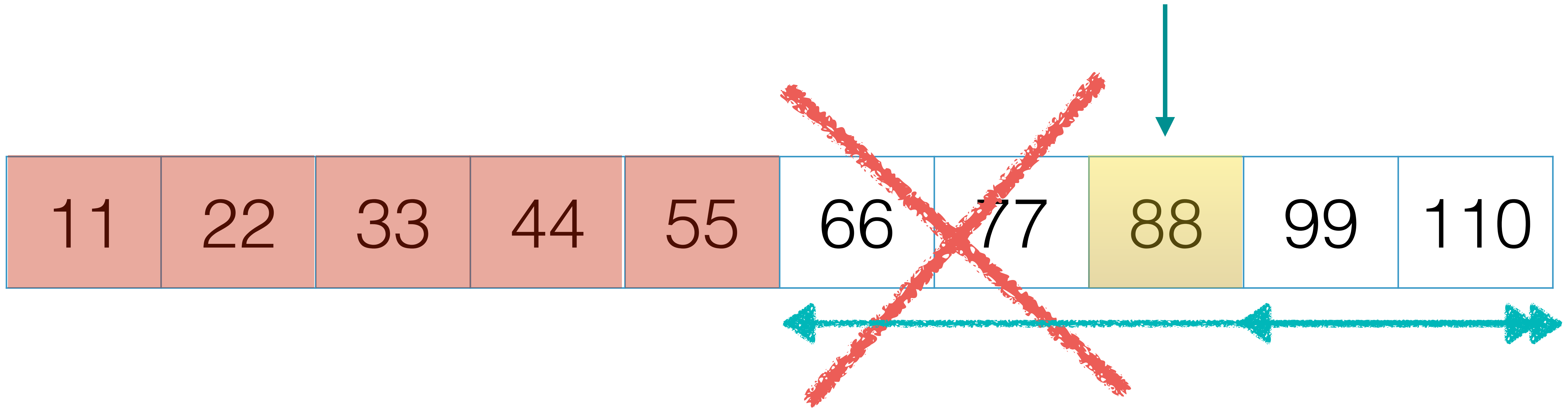
REDUCED SEARCH AREA

99 > 55 SO DISCARD THE
FIRST HALF OF THE LIST

BINARY SEARCH

SEARCHING FOR: 99

ELEMENT AT THE MIDPOINT



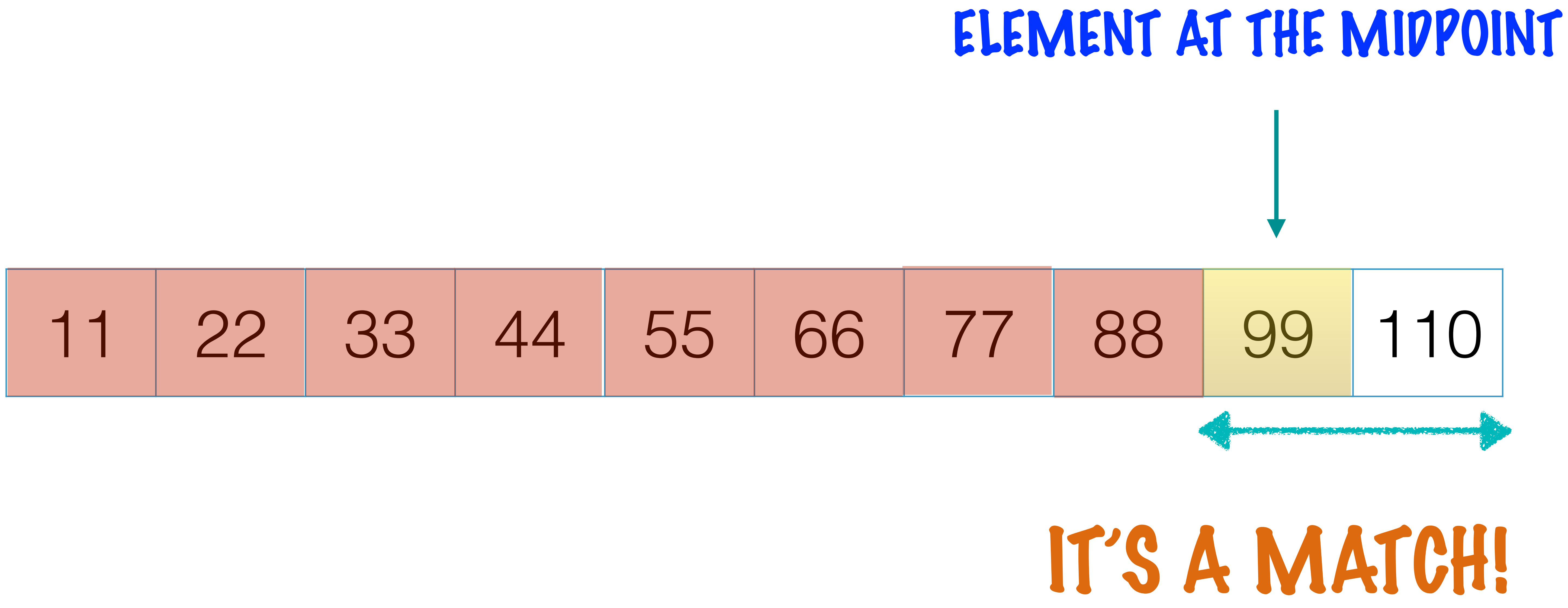
11	22	33	44	55	66	77	88	99	110

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

REDUCED SEARCH AREA

BINARY SEARCH

SEARCHING FOR: 99



ELEMENT AT THE MIDPOINT

11	22	33	44	55	66	77	88	99	110

IT'S A MATCH!

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

WHAT IS THE BASE CASE?

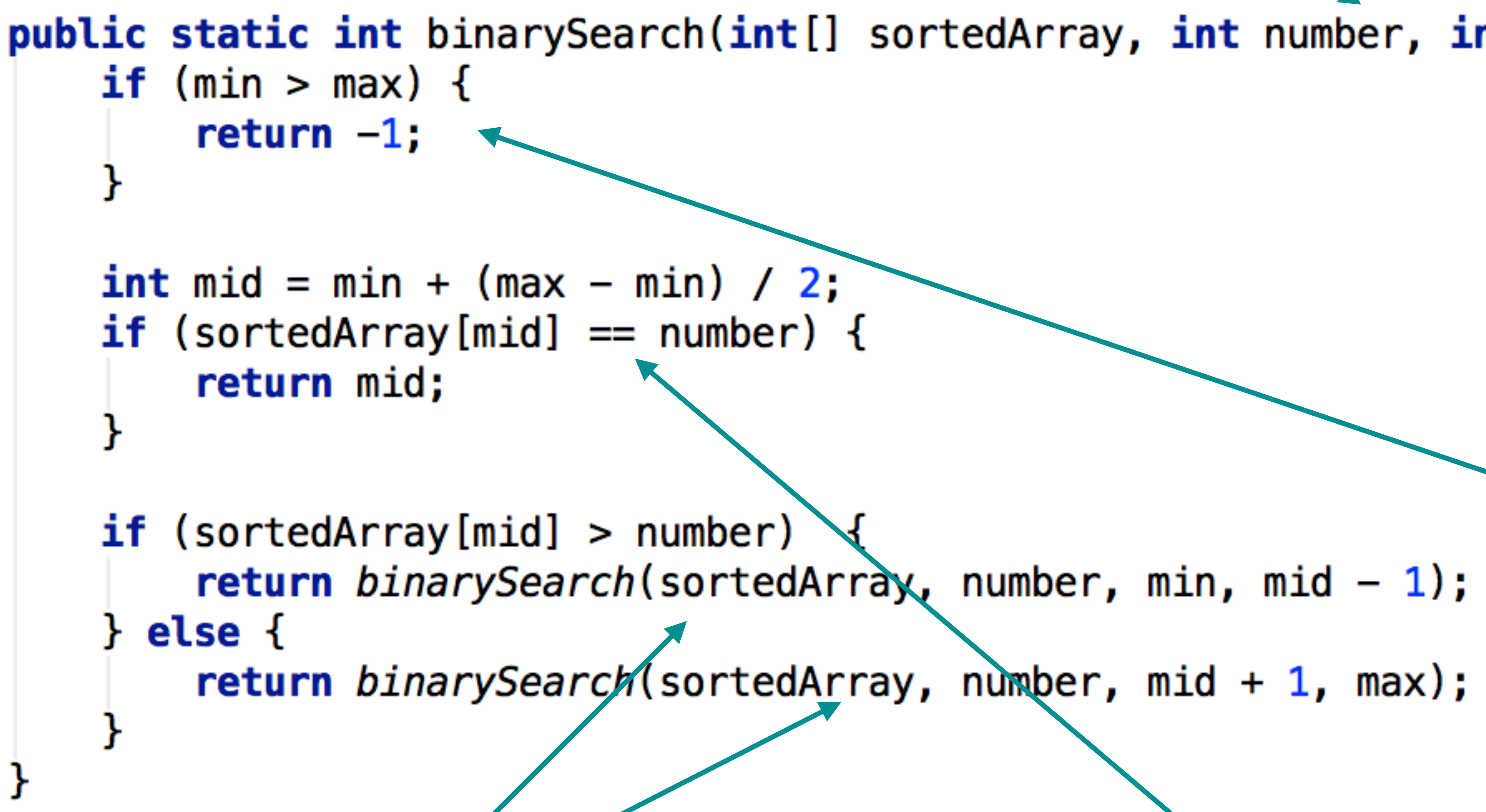
1. THERE IS NO PART OF THE LIST TO SEARCH AND THE ELEMENT HAS NOT BEEN FOUND
2. THE SEARCH ELEMENT HAS BEEN FOUND AT THE MID POINT OF THE PORTION WE'RE SEARCHING

WHAT IS THE RECURSIVE CASE?

BINARY SEARCH SMALLER PORTIONS OF THE LIST WHERE THE ELEMENT MIGHT BE FOUND

RECURSIVE BINARY SEARCH - CODE

THE SEARCH KEY, THE NUMBER WE'RE LOOKING FOR



```
public static int binarySearch(int[] sortedArray, int number, int min, int max) {
    if (min > max) {
        return -1;
    }

    int mid = min + (max - min) / 2;
    if (sortedArray[mid] == number) {
        return mid;
    }

    if (sortedArray[mid] > number)  {
        return binarySearch(sortedArray, number, min, mid - 1);
    } else {
        return binarySearch(sortedArray, number, mid + 1, max);
    }
}
```

THE SEARCH AREA IN THE SORTED LIST, THE MINIMUM AND MAXIMUM INDICES

IF THE INDICES HAVE CROSSED ONE ANOTHER AT THE CENTER, THE TERM IS NOT PRESENT IN THE ARRAY, RETURN -1, BASE CASE OF THE RECURSION

IF THE ELEMENT WE'RE SEEKING IS AT THE MID-POINT RETURN THE INDEX, ALSO A BASE CASE

CALL BINARY SEARCH ON A SMALLER PORTION OF THE ARRAY DEPENDING ON WHETHER THE ELEMENT MIGHT 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)

THE ITERATIVE APPROACH TO BINARY SEARCH MIGHT PERFORM BETTER THAN THE RECURSIVE APPROACH IN TERMS OF SPACE COMPLEXITY. THERE IS NOT RECURSIVE STACK IN THE ITERATIVE APPROACH, WHICH MEANS WE SAVE ON SOME SPACE.