

CS 152

Computer Programming
Fundamentals

Searching and Sorting Arrays

Brooke Chenoweth

University of New Mexico

Fall 2025

Find index of value

```
/**
 * Find location of given value in an array.
 * @param x Value to search for
 * @param values Array to search in
 * @return Index of x in values, or -1 if not present
 */
public static int linearSearch(int x, int[] values) {
    for(int i = 0; i < values.length; i++) {
        if(x == values[i]) {
            return i;
        }
    }
    return -1;
}
```

Linear Search Complexity

- If the array has 100 items, how many comparisons do I need to do to find the value?
 - worst case?
 - average case?
- What about if the array has 1000 items?
- The number of comparisons scales *linearly* with the size of the array.
- You'll sometimes see this written as $O(n)$ where n is the size of the array.
- Can we do better?
- What if the array is sorted?

Searching sorted array

```
public static int binarySearch(int x,
                                int[] sortedValues) {
    int low = 0;
    int high = sortedValues.length - 1;
    while(low <= high) {
        int mid = (low+high)/2;
        int midVal = sortedValues[mid];
        if(x < midVal) {
            high = mid-1;
        } else if(x > midVal) {
            low = mid + 1;
        } else {
            return mid;
        }
    }
    return -1;
}
```

Binary Search Complexity

- The number of comparisons scales with the $\log_2$ of size of the array.
- You'll sometimes see this written as $O(\log n)$ where n is the size of the array.
- Doubling the size of the array only adds one more comparison!
- This is great, but data isn't always sorted when we get it.
- How can we sort the array?

Find index of largest

```
public static int indexOfLargest(int[] array) {  
    int largestIndex = 0;  
    for(int i = 0; i < array.length; i++) {  
        if(array[i] > array[largestIndex]) {  
            largestIndex = i;  
        }  
    }  
    return largestIndex;  
}
```

Find index of largest in range

```
public static int indexOfLargest(int[] array, int n) {
    int largestIndex = 0;
    for(int i = 0; i < n; i++) {
        if(array[i] > array[largestIndex]) {
            largestIndex = i;
        }
    }
    return largestIndex;
}
```

Swap two elements

```
public static void swap(int[] array, int a, int b) {  
    int temp = array[a];  
    array[a] = array[b];  
    array[b] = temp;  
}
```


Selection sort

```
public static void selectionSort(int[] values) {  
    for(int i = 0; i < values.length; i++) {  
        int endIndex = values.length - i - 1;  
        int maxIndex = indexOfLargest(values,  
                                     values.length - i);  
        swap(values, endIndex, maxIndex);  
    }  
}
```

Selection Sort Complexity

- How does sorting scale with the size of the array?
- The number comparisons scales with the square of the size of the array.
- You'll sometimes see this written as $O(n^2)$ where n is the size of the array.
- This sorting algorithm takes *quadratic* time.
- There are better/faster algorithms out there. (mergesort, quicksort, heapsort, etc.)