Searching

AP Computer Science

Linear Search

Card Example

- We will demonstrate how a linear search works using cards

Linear Search in Java

- With an array of primitives

```
public int linearSearch(int[] arr, int key)
{
    for(int i = 0; i < arr.length; i++) {
        if(key == arr[i])
        {
            return i; // Found target,return index
        }
    }
    return -1;        // Failed to find target
}
```

Linear Search in Java

- With an array of objects

```
public int linearSearch(String[] arr, String key)
{
   for(int i = 0; i < arr.length; i++) {
      if(key.equals(arr[i]))
      {
         return i; //Found target,return index
      }
   }
   return -1;       //Failed to find target
}
```

Linear Search in Java

- With an ArrayList of objects

```
public int search(ArrayList<Dot> arr, Dot key)
{
   for(int i = 0; i < arr.size(); i++) {
      if(key.equals(arr.get(i)))
      {
         return i; //Found target,return index
      }
   }
   return -1;       //Failed to find target
}
```

Accessing Elements Arrays vs ArrayLists

- One problem many of you are having is the difference between accessing an element of an Array vs ArrayList
- With an array make sure you use **[ ]** to access an element

```
int[] arr = {1, 2};
System.out.println(arr[0]);
System.out.println(arr[1]);
```

- With an ArrayList make sure you use **get()** to access an element

```
ArrayList<Integer> nums = new ArrayList<Integer>();
nums.add(1);
nums.add(2);
System.out.println(nums.get(0));
System.out.println(nums.get(1));
```

Linear Search 2D Array

```
public int countNums(int[][] arr, int key)
{
    int count = 0;
    for(int r = 0; r < arr.length; r++) {
        for(int c = 0; c < arr[r].length; c++) {
            if(key == arr[r][c])
            {
                count++;
            }
        }
    }
    return count;
}
```

Linear Search Limitations

- Are you happy with linear search for your small set of cards?
- Would you still want to use linear search for a set of data containing 1000, 10000, 100000, or 1000000 items?
- How would you approach finding an item in a set this big?

Quick aside

- How many guesses would it take you to figure out a number someone picks between 1 and 100? What about 1 and 1000?
  - The only feedback they would provide is if your guess is correct, high, or low

Binary Search

Card Example

- We will demonstrate how a binary search works using cards now
- Pay attention because we are going to ask you to write the algorithm for a binary search when we are done

Binary Search Pseudocode

```
binary search (array, target)
    low  = 0                          // first index
    high = array.length - 1  // last index
    while low <= high
        // calculate the midpoint index
        middle = low + (high - low) / 2
        if array[middle] == target
            return middle
        else if array[middle] < target
            low = middle + 1
        else
            high = middle - 1
    return -1
```

Write Binary Search Algorithm

- Take the card sorting example shown and the pseudocode provided
  - Write a binary search on paper
- Exchange your search algorithm with the group to your left
- Use the algorithm given to search a set of cards
- Does it work?
  - If not, what needs to be changed?

```
public int binarySearch(int[] arr, int key) {
    int low = 0;
    int high = arr.length - 1;
    // your code here
}
```

Binary Search Pseudocode

```
binary search (array, target)
    low  = 0                          // first index
    high = array.length - 1  // last index
    while low <= high
        // calculate the midpoint index
        middle = low + (high - low) / 2
        if array[middle] == target
            return middle
        else if array[middle] < target
            low = middle + 1
        else
            high = middle - 1
    return -1
```

Binary Search in Java

```
// precondition: array arr[] is sorted
public int rank(int[] arr, int key) {
   int low = 0;
   int high = arr.length - 1;
   while (low <= high) {
      int mid = low + (high - low) / 2;
      if (key == arr[mid])
         return mid;
      else if (key < arr[mid])
         high = mid - 1;
      else  // key > arr[mid]
         low = mid + 1;
   }
   return -1;
}
```

With primitives

Binary Search in Java

```
// precondition: array arr[] is sorted
public int rank(String[] arr, String key) {
   int low = 0;
   int high = arr.length - 1;
   while(low <= high) {
      int mid = low + (high - low) / 2;
      if(key.compareTo(arr[mid]) == 0)
         return mid;
      else if(key.compareTo(arr[mid]) < 0)
         high = mid - 1;
      else // key.compareTo(arr[mid]) > 0
         low = mid + 1;
   }
   return -1;
}
```

With objects

Search Algorithm Efficiency

Name	Best Case	Avg. Case	Worst Case
Linear Search	1	`array.length` `list.size()`	`array.length` `list.size()`
Binary Search	1	$\log_2$`(array.length)` $\log_2$`(list.size())`	$\log_2$`(array.length)` $\log_2$`(list.size())`

Binary Search Requirements

- As you saw with the example of an unsorted set of cards a binary search requires the set to be sorted
- This is another reason why implementing **`Comparable`** and writing a **`compareTo(obj)`** method in your classes is so important

Arrays Methods

Commonly Used Arrays Methods	
sort(a)	Sorts the specified array a into ascending order.
binarySearch(a, key)	Searches the specified array a for the specified value key using the binary search algorithm.
equals(a, a2)	Returns true if the two specified arrays are equal to one another.
toString(a)	Returns a string representation of the contents of the specified array.

```
import java.util.Arrays;
```

Sorting an Array

- There are two things to notice here:
  - Arrays.sort()
  - Arrays.toString()
- What is the output?

```
int[] arr = {8, 5, 3, 9, 4};
System.out.println(Arrays.toString(arr));
Arrays.sort(arr);
System.out.println(Arrays.toString(arr));
```

Output
[8, 5, 3, 9, 4] [3, 4, 5, 8, 9]

Binary Search with Arrays

- What is the output?

```
int[] arr = {8, 5, 3, 9, 4};
Arrays.sort(arr);
System.out.println(Arrays.binarySearch(arr, 8));
System.out.println(Arrays.binarySearch(arr, 7));
System.out.println(Arrays.binarySearch(arr, 3));
System.out.println(Arrays.binarySearch(arr, 22));
```

- binarySearch() will return one of two things:
  - The index of the item if it is in the Array
  - If the item is not in the Array it returns **-1 + -(index where it would be)**

Output
3
-4
0
-6

Arrays equals()

- What is the output?

```
int[] arr1 = {8, 5, 3, 9, 4};
int[] arr2 = {3, 4, 5, 8, 9};
int[] arr3 = arr1;
System.out.println(Arrays.equals(arr1, arr2));
Arrays.sort(arr1);
System.out.println(Arrays.equals(arr1, arr2));
System.out.println(Arrays.equals(arr1, arr3));
```

- Two arrays are equal if they contain the same elements in the **same order**

Output
false **true** **true**

List Methods

Commonly Used List Methods	
indexOf(o)	Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.
contains(o)	Returns true if the list o contains the specified element.
equals(o)	Compares the specified object with this list for equality.
toArray()	Returns an **array of objects** containing all of the elements in this list in proper sequence (from first to last element).

```
import java.util.List;
```

List indexOf() and contains()

- What is the output?

```
List<Integer> one = new ArrayList<Integer>();
one.add(8);
one.add(4);
one.add(2);
System.out.println(one.indexOf(8));
System.out.println(one.indexOf(7));
System.out.println(one.contains(4));
System.out.println(one.contains(7));
```

Output

0
-1
true
false

List equals()

- What is the output?

```
List<Integer> one = new ArrayList<Integer>();
List<Integer> two = new ArrayList<Integer>();
one.add(8);
one.add(4);
one.add(2);
two.add(4);
two.add(2);
two.add(8);
Collections.sort(one);
System.out.println(one.equals(two));
Collections.sort(two);
System.out.println(one.equals(two));
```

Output
false true

Collections Methods

Commonly Used Collections Methods	
sort(list)	Sorts the specified list into ascending order, according to the natural ordering of its elements.
binarySearch(list,key)	Searches the specified list for the specified object using the binary search algorithm.

```
import java.util.Collections;
```

Sorting Collections

- What is the output?

```
List<Integer> one = new ArrayList<Integer>();
List<Integer> two = new ArrayList<Integer>();
one.add(8);
one.add(4);
one.add(2);
two.add(4);
two.add(2);
two.add(8);
System.out.println(Arrays.toString(one.toArray()));
Collections.sort(one);
System.out.println(Arrays.toString(one.toArray()));
Collections.sort(two);
System.out.println(Arrays.toString(two.toArray()));
```

Output
[8, 4, 2] [2, 4, 8] [2, 4, 8]

Binary Search with Collections

- binarySearch() works the same as with Arrays
- Make sure you have the import Collections
- Does the ArrayList need to be sorted?
- What is the output?

```
List<Integer> one = new ArrayList<Integer>()
one.add(8);
one.add(4);
one.add(2);
Collections.sort(one);
System.out.println(Collections.binarySearch(one,8));
System.out.println(Collections.binarySearch(one,5));
System.out.println(Collections.binarySearch(one,2));
System.out.println(Collections.binarySearch(one,7));
```

Output
2
-3
0
-3