Searching: Sections 14.5-14.6

Ted Samore

John MacAslan

Eric Vernon

Methods of Searching

- Linear Search
  - Step through each piece of data individually
  - Data does not need to be sorted
- Binary Search
  - Continually cuts the dataset in half
  - Requires the data to be sorted

Linear Search

- O(n)
- Example:

An array consists of {4, 2, 6, 8, 10, 11}, and a user wants to find the index of the number 8.

The searching algorithm will check 4, and see that 4 does not equal 8. It then checks 2, 6, and finally 8 before returning "3".

If the user searches for any element not inside the dataset, the algorithm will return "-1".

Binary Search

- O(log n)
- Requires presorted data
- Example:

An array consists of {2, 4, 7, 9, 11, 17, 25}, and a user wants to find the index of the number 7.

The algorithm will examine the middle element, 4. Since 9 is greater than 4, the algorithm knows it only needs to consider elements with smaller indices than 9 - the subset {2, 4, 7}.

It then repeats the process and examines the middle element, 4. Since 4 equals 4, it returns the correct index of "1".

If the user searches for any element not inside the dataset, the algorithm will return "-1".

Arrays.binarySearch

- The *Arrays* class has a nifty static function, *binarySearch*.
- If the object is not found, it tells you where the object would go.
- Useful for keeping an array sorted.
- Example:

```
int[] a = {1, 4, 9};
int v = 7;
int pos = Arrays.binarySearch(a, v);
// pos = (-k - 1) = -3
```

When to Use Which

- If your data is already sorted:
  - Use a binary search!
- If your data is not already sorted:
  - Are you going to need to search this data *multiple times?*
    - If yes, it's worth it to sort the data, then binary search
    - If not, just use a linear search

Activity

Demo

Wrap-Up

- Questions?
- Problems with the quiz?