Searching

Overview

- We often need to search for an item in a collection
- In this module, we will learn about how to search for an element in an array
- Example:
    - Find the cat named Garfield inside an array named *shelter*

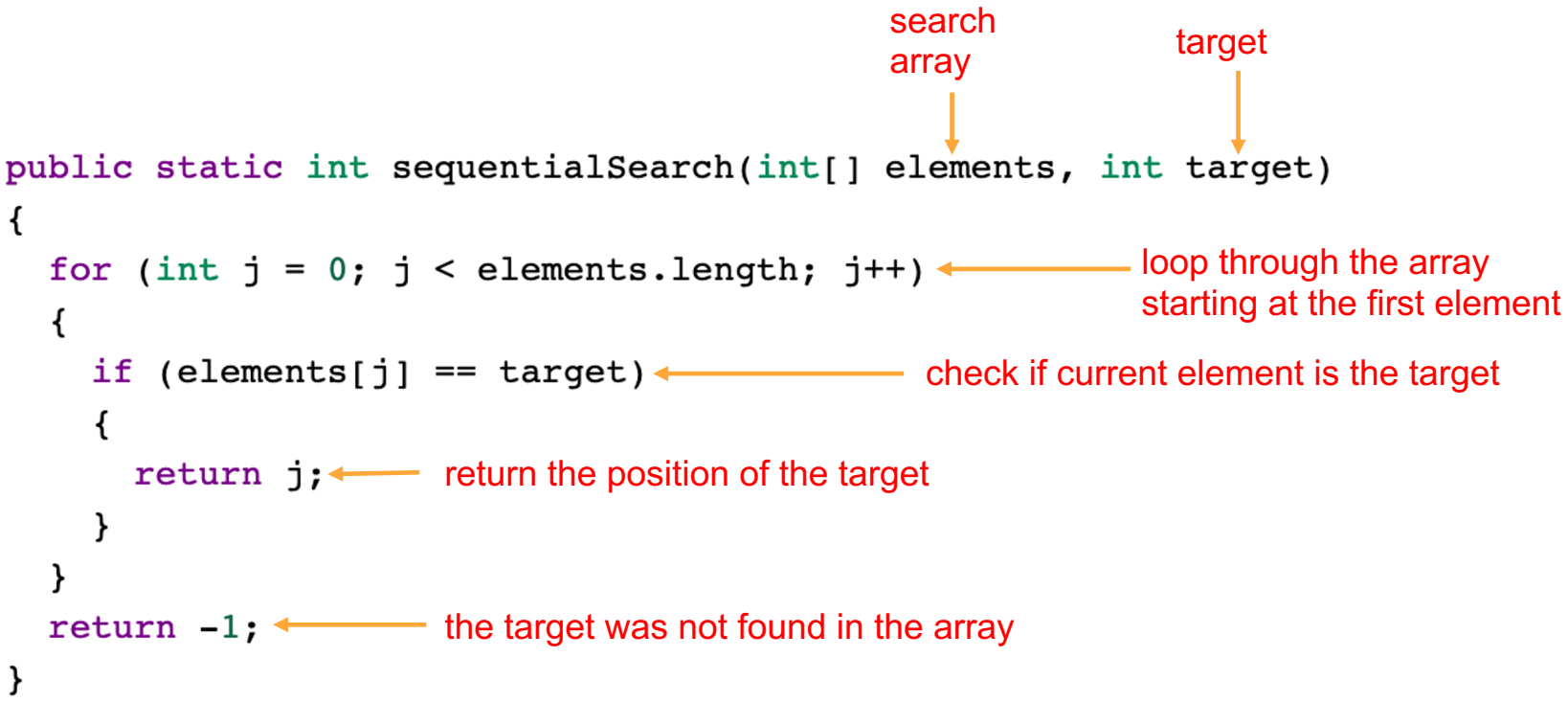
Learning Objectives

- **To be able to use linear search to find an element inside an array**
- To be able to use binary search to find an element inside an array
- To be able to know when to use linear search and when to use binary search

Linear Search

- Used to search for a value (the target) in an **unsorted array**
- It uses a loop to iterate over the values
- Starts at the first element and move to the next element until the target is found
- Returns the position of the target if it was found in the array
- Returns -1 if the target was not found in the array

Linear Search: array



Learning Objectives

- To be able to use linear search to find an element inside an array or an ArrayList
- **To be able to use binary search to find an element inside an array**
- **To be able to know when to use linear search and when to use binary search**

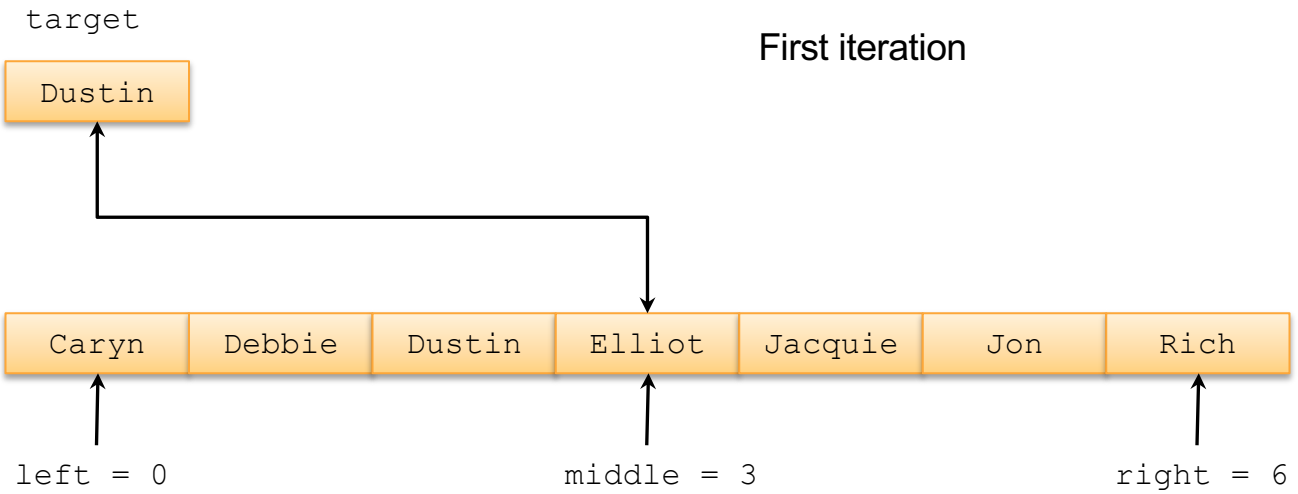
Binary Search

- Used to search for a value (the target) in a **sorted array**
- Keeps dividing the array in half
- Compares the target with the value at the middle index (middle element)
- If the target is less than the middle element, then we search the target in the **left half of the array** (the positions before the middle element)
- If the target is greater than the middle element, then we search the target in the **right half of the array** (the positions before the middle element)

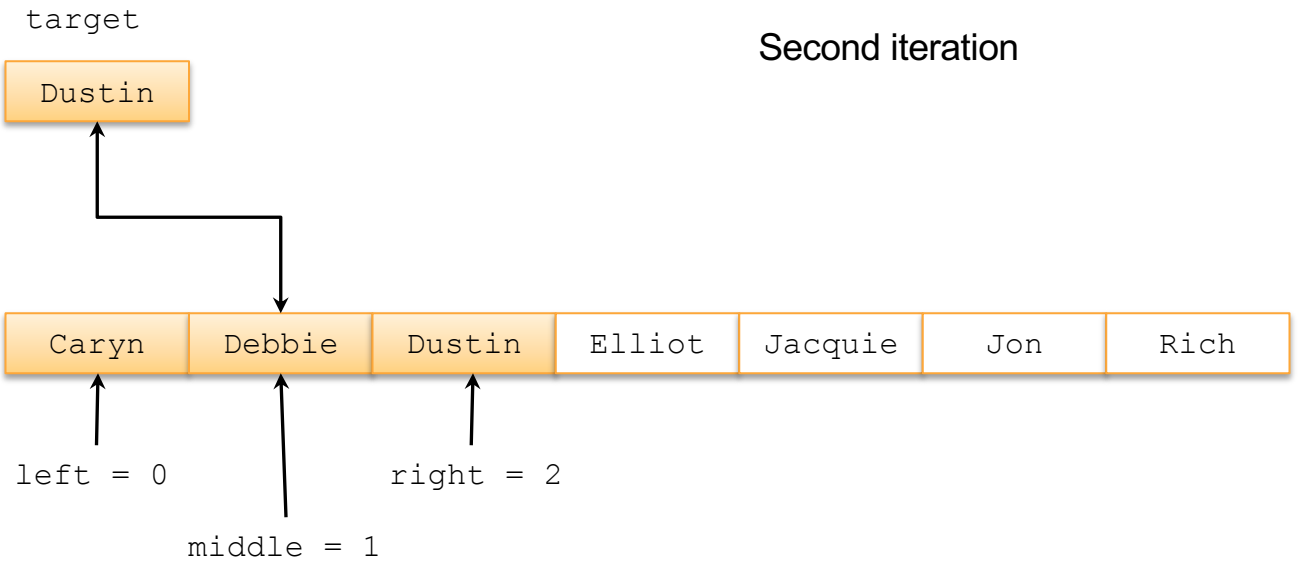
Binary Search

- Returns the position of the middle element if it is equal to the target
- Returns -1 if the target was not found in the array

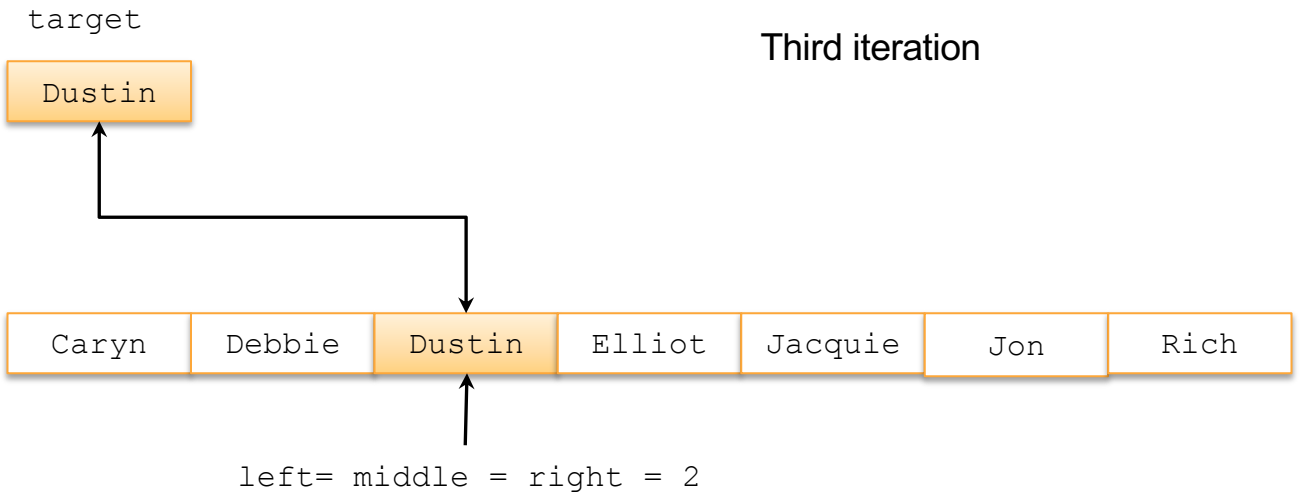
Binary Search



Binary Search



Binary Search



Binary Search

search array

target

```
public static int binarySearch(String[] elements, String target) {
   int left = 0;
   int right = elements.length - 1;
   while (left <= right)
   {
      int middle = (left + right) / 2;
      if (target.compareTo(elements[middle]) < 0)
      {
         right = middle - 1;
      }
      else if (target.compareTo(elements[middle]) > 0)
      {
         left = middle + 1;
      }
      else {
         return middle;
      }
    }
    return -1;
}
```

keep searching until no space left

compute middle position

move right before middle when target < middle element

move left after middle when target > middle element

Return middle when target == middle element

the target was not found in the array

Linear Search vs. Binary Search

- Binary search is faster than linear search
- Binary search runs on sorted data
- Linear search runs on unsorted data

Linear Search vs. Binary Search

- **Runtime analysis**: how many comparisons will it take to determine that the target is not in the array?

Length of the array	Linear Search	Binary Search
2	2	2
4	4	3
8	8	4
16	16	5
100	100	7