Searching

CS 1181 Computer Science II

Search

- Given a collection (e.g. array, list, etc.) of values and a target value, determine whether or not the target is in the list.

- If the value is in the list, return its index
- If the value is not in the list, return -1

- If the value occurs more than once in the list, you can return any valid index

- int[] arr = {11, 21, 37, 39, 42, 68, 103, 109}
- search(arr, 42) => 4
- search(arr, 57) => -1

Search

- int[] arr = {11, 21, 37, 39, 42, 68, 103, 109}
- search(arr, 42) => 4
- search(arr, 57) => -1

- What strategies can you think of to implement the search method?

Linear Search

- Start at the beginning of the array (index 0) and ask “is this value the target?”
- If so, return 0
- If not, move to index 1 and repeat
- If you get to the end of the array and haven’t found the target, return -1

- int[] arr = {11, 21, 37, 39, 42, 68, 103, 109}
- search(arr, 68) => 5
- search(arr, 57) => -1

- Let’s code this…

Binary Search

- If the numbers are sorted, we can do better
- Binary search is like the "guess what number I'm thinking of" game.
- Compare the middle array value to the target
  - If equal, return the index of the middle value
  - If the middle value is higher than the target, repeat for the first half of the array
  - If the middle value is lower than the target, repeat for the last half of the array

Binary Search

- int[] arr = {11, 21, 37, 39, 42, 68, 103, 109}
- search(arr, 68) => 5
  - 39 < 68 => search second half (42, 68, 103, 109)
  - 68 == 68 => search its index, which is 5
- search(arr, 57) => -1
  - 39 < 57 => search second half (42, 68, 103, 109)
  - 68 > 57 => search first half (42)
  - 42 < 57 => search second half, but it's empty so return -1
- Let's code this…

Binary Search

- What would be some good test cases to thoroughly test your binary search method?
- Implement these test cases

More realistic implementations

- If you wanted to search an array of doubles instead of ints, what would need to change?
- This is a great use case for generics.
- Re-implement linearSearch and binarySearch using generics.
- What conditions must the data type meet in each case?

Which search is more work?

- We'll define "work" as the number of times we need to loop (or the number of comparisons we need to make)

- How many times do we need to loop for this?
- int[] arr = {11, 21, 37, 39, 42, 68, 103, 109}
- linearSearch(arr, 68) => 5

- What is the least number of times we need to loop for linear search? When does this happen?

- What about the most number of times we need to loop for linear search?

- If instead of 8 items, the list had 32 items, what would be the most number of times we would need to loop?

- What if the list had n items?

Which search is more work?

- Same questions but for **binary** search:

- How many times do we need to loop for this?
- int[] arr = {11, 21, 37, 39, 42, 68, 103, 109}
- binarySearch(arr, 68) => 5

- What is the least number of times we need to loop for binary search? When does this happen?

- What about the most number of times we need to loop for binary search?

- If instead of 8 items, the list had 32 items, what would be the most number of times we would need to loop?

- What if the list had n items?

Question

- When comparing the search algorithms, we based our comparison on the number of times that each algorithm looped in the worst case. What other ways could be compare two implementations of the same algorithm?

Time Complexity

- Empirical versus theoretical
- Best, worst, or average case