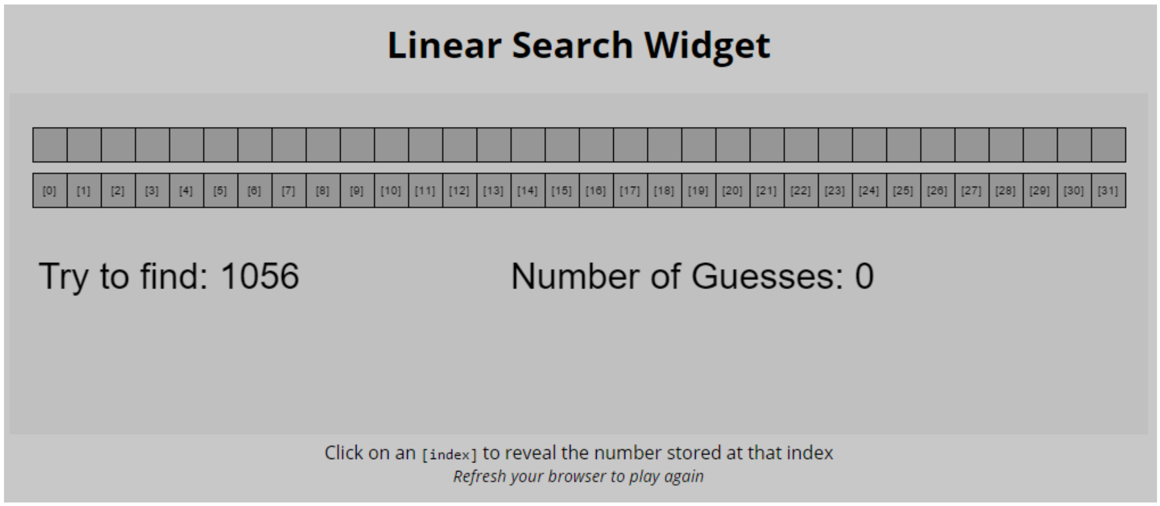
Linear and Binary Search
CPJava

In this assignment you and a partner or two will use two "widgets" to explore the different searching techniques of linear and binary search.

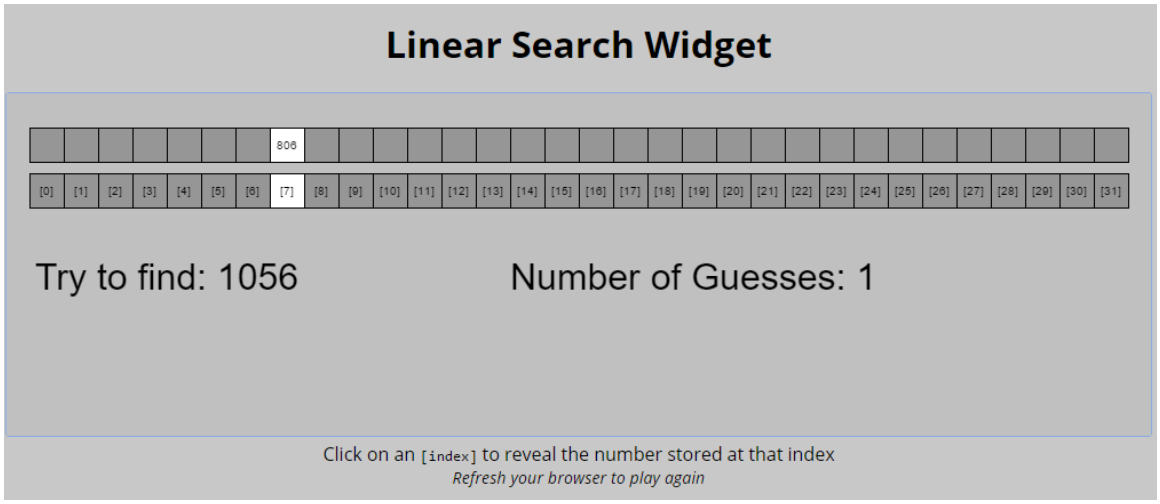
Linear Search

The first searching technique we will explore is linear search. Start by opening a web browser and visiting https://chandrunarayan.github.io/LinearSearchWidget/. You should see a page that looks like this:



The 32 empty boxes represent an array of numbers. We don't yet know what numbers are stored in the array. Our goal is to find a particular number (or determine if the number is not in the array).

In the screen shot above, we need to find the number 1056. It could be anywhere in the array. To find it, we will click on the index numbers in the bottom row. For instance, in this example when I clicked on the index [7], it revealed that the number 806 was stored at that index.



We need to keep clicking until we find the number we are looking for. The Widget will keep track of the number of guesses it takes to find the number or to determine that the number is not in the array.

With a partner or two, do at least 10 linear searches. Record your results below. For example, if I found a number after 8 guesses on the 1st try, I would record 8 under "Number Found." If I determined that the number wasn't in the array, and would record the number of guesses in the "Number NOT found" column.

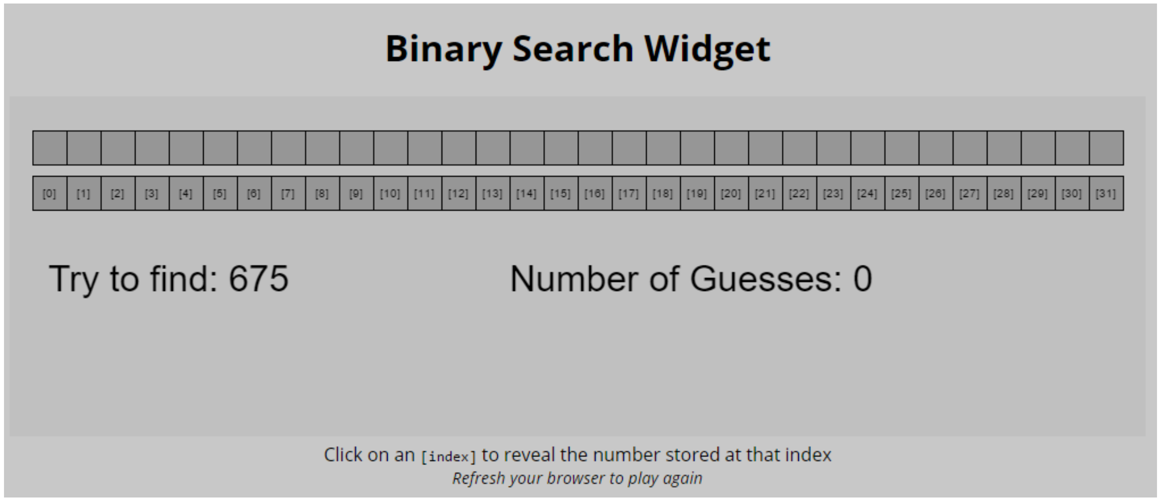
	Number Found after # guesses	Number NOT Found after # guesses
1st Try		
2nd Try		
3rd Try		
4th Try		
5th Try		
6th Try		
7th Try		
8th Try		
9th Try		
10th Try		
Lowest number in each column		
Highest number in each column		
Average number in each column		

If the size of the array was doubled to 64 numbers rather than 32, what we be the on the lowest, highest and average number of guesses to find a number that was in the array?

If the size of the array was doubled to 64 numbers rather than 32, what we be the on the lowest, highest and average number of guesses to determine that a number was NOT in the array?

Binary Search

The second searching technique we will explore is binary search. Start by opening a web browser and visiting https://chandrunarayan.github.io/BinarySearchWidget/. You should see a page that looks like this:



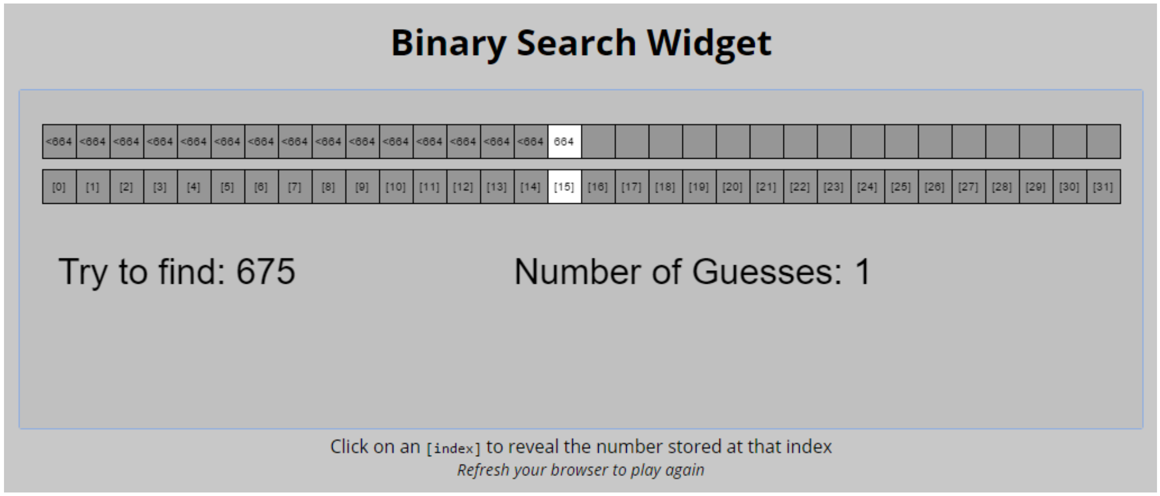
Try to find: 675

Number of Guesses: 0

Click on an [index] to reveal the number stored at that index

Refresh your browser to play again

In the Binary Search widget the numbers are in ***sorted order*** from smallest to largest. When we click on the [index] it not only reveals the value of the number stored, but it also shows us that we can eliminate to possibilities on one side of the number. In this example, we were looking for the number 675. When I clicked on index [15] it revealed the number at that index was 664. Since that is smaller than the number we are looking for, we can eliminate index [15] and all the positions to the left.



Try to find: 675

Number of Guesses: 1

Click on an [index] to reveal the number stored at that index

Refresh your browser to play again

With your partners, do at least 10 binary searches. Record your results below. Again, if you find the number in the array, record the number of guesses in the "Number Found" column. If If you determined that the number wasn't in the array, record the number of guesses in the "Number NOT found" column. If you are careful to divide the array in half with each guess, you can minimize the number of guesses needed.

	Number Found after # guesses	Number NOT Found after # guesses
1st Try		
2nd Try		
3rd Try		
4th Try		
5th Try		
6th Try		
7th Try		
8th Try		
9th Try		
10th Try		
Lowest number in each column		
Highest number in each column		
Average number in each column		

If the size of the array was doubled to 64 numbers rather than 32, what we be the on the lowest, highest and average number of guesses to find a number that was in the array?

If the size of the array was doubled to 64 numbers rather than 32, what we be the on the lowest, highest and average number of guesses to determine that a number was NOT in the array?

Suppose you have a sorted array of 128 names, and you're searching through it using a binary search. What's the maximum number of steps ("guesses") it would take to either find the name or determine it was not in the array?

Suppose you double the size of the array to 256 names. What is the maximum number of steps now?

Each person in your group should submit a copy of this completed worksheet as a new document to the Google assignment