

Linear and Binary Search **Key**

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://chandrinarayan.github.io/LinearSearchWidget/>. You should see a page that looks like this:

Linear Search Widget

Try to find: 1056

Number of Guesses: 0

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]	[15]	[16]	[17]	[18]	[19]	[20]	[21]	[22]	[23]	[24]	[25]	[26]	[27]	[28]	[29]	[30]	[31]
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------

Click on an [index] to reveal the number stored at that index
Refresh your browser to play again

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.

Linear Search Widget

							806																								
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]	[15]	[16]	[17]	[18]	[19]	[20]	[21]	[22]	[23]	[24]	[25]	[26]	[27]	[28]	[29]	[30]	[31]

Try to find: 1056

Number of Guesses: 1

Click on an `[index]` to reveal the number stored at that index
Refresh your browser to play again

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
1 st Try	Answers in this column will vary	Answers in this column will always be 32
2 nd Try		
3 rd Try		
4 th Try		
5 th Try		
6 th Try		
7 th Try		
8 th Try		
9 th Try		
10 th Try		
Lowest number in each column	Answers will vary	32
Highest number in each column	Answers will vary	32
Average number in each column	around 16	32

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

It should about double each of those numbers

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

The high, low and average would all be 64

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

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.

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 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
1 st Try	Answers in this column will vary, but should be around 5 or less	Answers in this column should always be 5 or greater
2 nd Try		
3 rd Try		
4 th Try		
5 th Try		
6 th Try		
7 th Try		
8 th Try		
9 th Try		
10 th Try		
Lowest number in each column	Typically a little less than 5	5 or greater
Highest number in each column	around 5	5 or greater
Average number in each column	Typically a little less than 5	5 or greater

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

They should all increase by about 1 guess

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

It would take 6 guesses to determine it 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?

It would take 7 guesses to determine the name was not in the array

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

It would take 8 guesses to determine the name was not in the array

Each person in your group should submit a copy of this completed worksheet to the school loop drop box for the assignment.