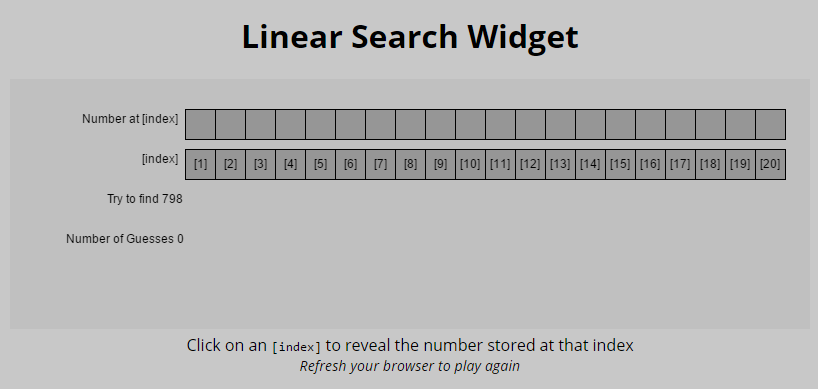
*Linear and Binary Search*

*AP Computer Science Principles*

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

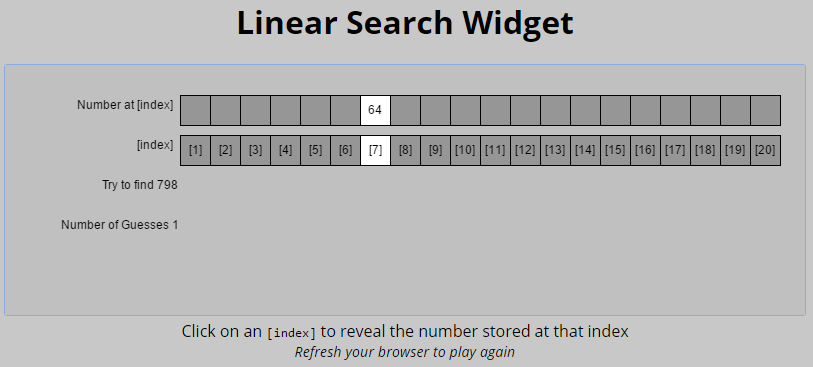
**Linear Search**

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

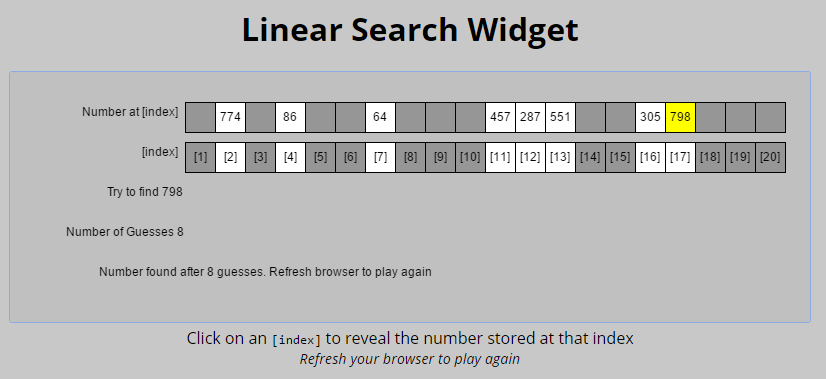


The 20 empty boxes represent a list of numbers. We don’t yet know what numbers are stored in the list. Our goal is to find a particular number (or determine if the number is not in the list).

In the screen shot above, we need to find the number 798. It could be anywhere in the list. 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 64 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 list. In this example, I found the number I was trying to find after 8 guesses.



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 list, and would record the number of guesses in the “Number NOT found” column.

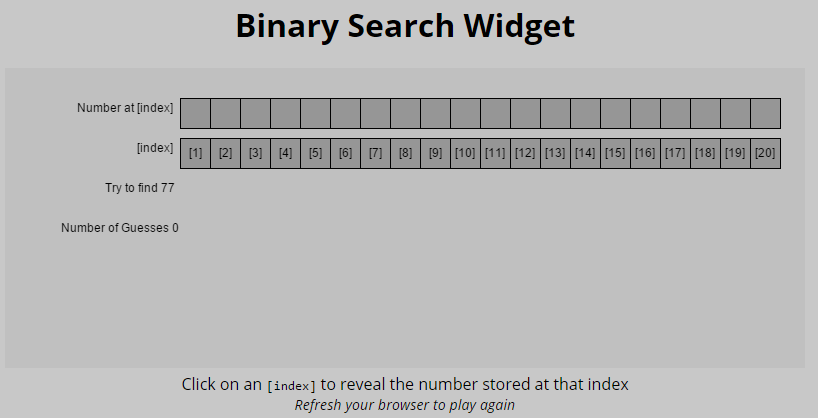
|  |  |  |
| --- | --- | --- |
|  | Number Found after # guesses | Number NOT Found after # guesses |
| 1st Try |  |  |
| 2nd Try |  |  |
| 3rd Try |  |  |
| 4th Try |  |  |
| 5th Try |  |  |
| 6th Try |  |  |
| 7th Try |  |  |
| 8th Try |  |  |
| 9th Try |  |  |
| 10th Try |  |  |
|  |  |  |
| Lowest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

If the size of the list was doubled to 40 numbers rather than 20, what we be the on the lowest, highest and average number of guesses to find a number that was in the list?

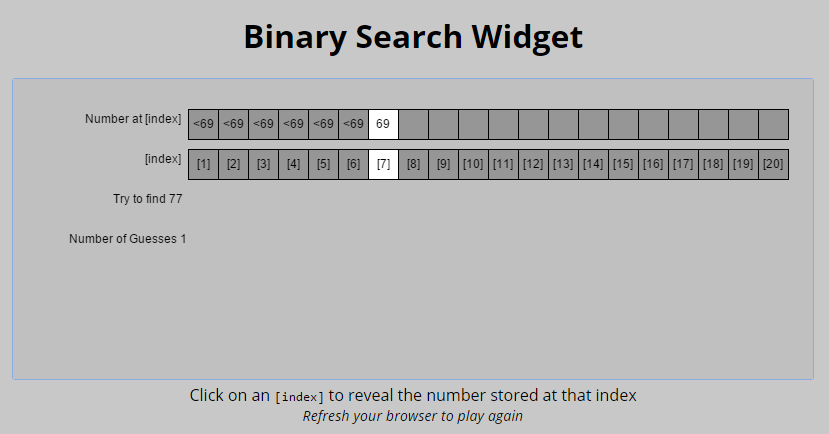
If the size of the list was doubled to 40 numbers rather than 20, what we be the on the lowest, highest and average number of guesses to determine that a number was NOT in the list?

**Binary Search**

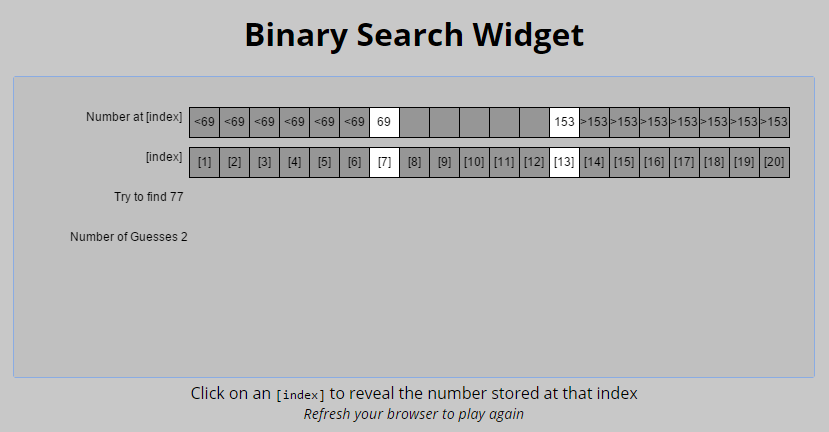
The second searching technique we will explore is binary search. Start by opening a web browser and visiting . You should see a page that looks like this:



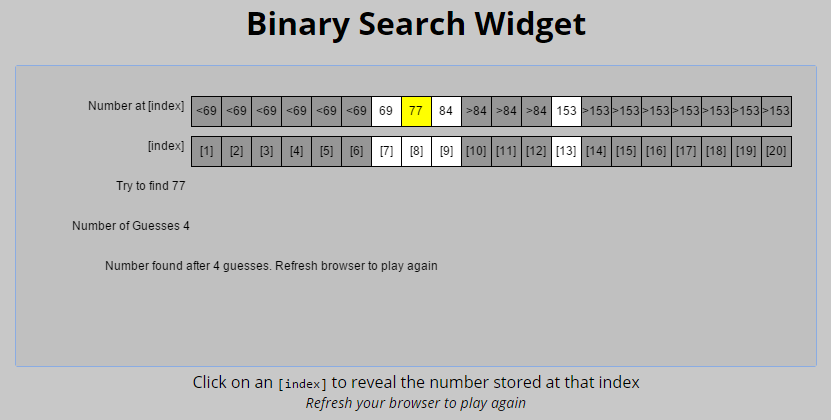
In the Binary Search widget the numbers are in ***sorted order*** from smallest to largest. When we click on the [index] it not reveals the value of the number stored, but it also shows us that we can eliminate to possibilities on one side of the number. In this example, we were looking for the number 77. When I clicked on index [7] it revealed the number at that index was 69. Since that is smaller than the number we are looking for, we can eliminate index [7] and all the positions to the left.



Each time I click on an index, the widget will reveal the value at that index and eliminate other possible indexes. On my next guess I clicked on index [13] and that revealed the number 153 which is larger than the number I’m trying to find.



My third guess was index [8]. After four guesses I found the number at index [7].



With your partners, do at least 10 binary searches. Record your results below. Again, if you find the number in the list, record the number of guesses in the “Number Found” column. If If you determined that the number wasn’t in the list, record the number of guesses in the “Number NOT found” column. If you are careful to divide the list 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 list was doubled to 40 numbers rather than 20, what we be the on the lowest, highest and average number of guesses to find a number that was in the list?

If the size of the list was doubled to 40 numbers rather than 20, what we be the on the lowest, highest and average number of guesses to determine that a number was NOT in the list?

Suppose you have a sorted list 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 list?

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