

# Binary Search Illustrated

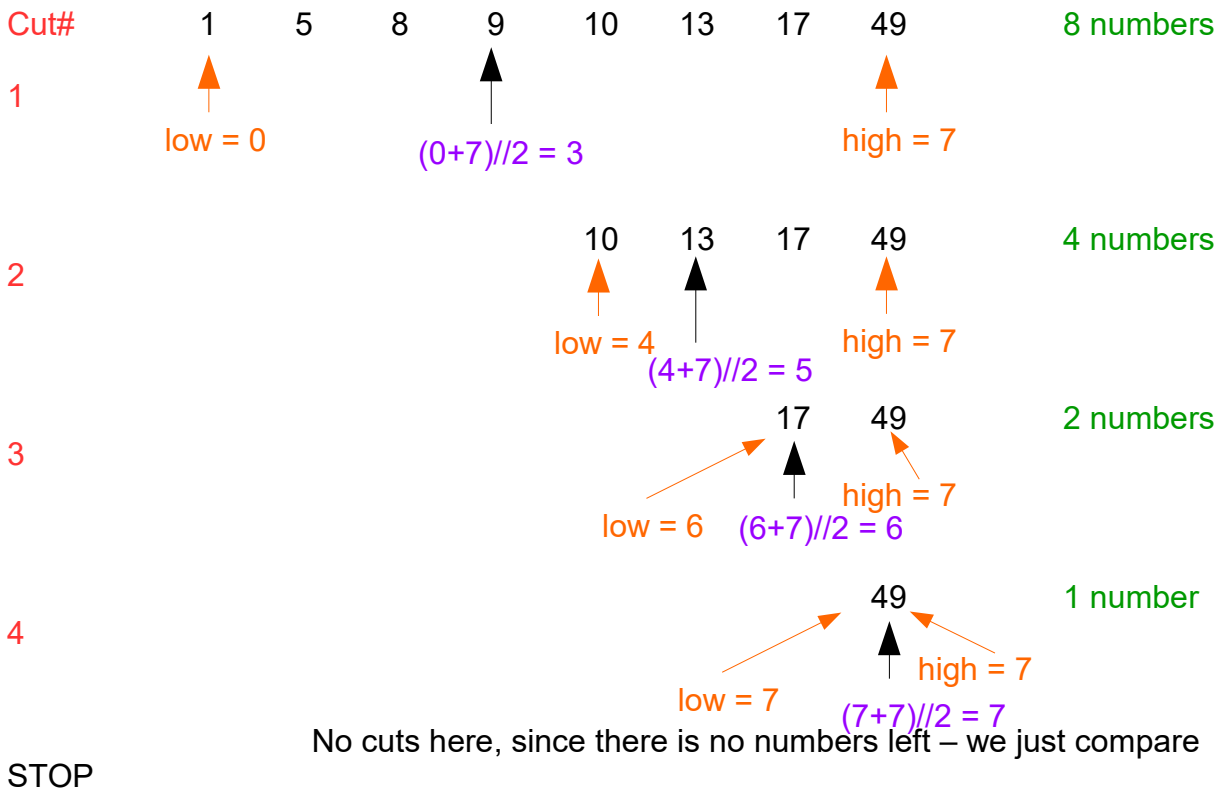
1) for the in-class and homework assignment I asked you to copy three search procedures (using built-in list's operator `index`, `linear search` and `binary search`) and time them on three different numbers for search: 10, 499999, 999999. The `binary search`'s all three times were 0's.

Let's see how many steps does it take a binary search to find 499,999:  
 $(0+999999)//2 = 499999$  – just one step (no cuttings will be done)

To find 999999 all the cuts will have to be done (worst-case scenario),

And to find 10?

2) Now, let's see how the binary search algorithm provided in our book works: let's take a list of 8 integer numbers: 1, 5, 8, 9, 10, 13, 17, 49, and let's try to find 26 (note that they are already sorted as Binary Search has this requirement)



Note: we are considering the worst-case scenario (when the element is not present in the list, and the algorithm takes the longest route to stop)

With binary search we can see the following pattern:

List size	# of cuts	
1	0	
2	1	$2^1=2$
4	2	$2^2=4$
8	3	$2^3=8$
16	4	$2^4=16$

Do you see  $2^{\text{#of cuts}} = n$  dependency?

If we re-write it in logarithmic form:  
 $\log_2 n = \text{\# of cuts}$