Topic 6.0: Searching & Sorting

Learning Goals (Week 7):

- **Write code that implements linear search on an array**
- **Write code that implements binary search on an array**
- Describe sorting algorithms such as insertion sort in plain English.
- Describe sorting algorithms such as selection sort in plain English.
- Describe sorting algorithms such as merge and quick sort in English.
- Write code that implements insertion sort
- Write code that implements selection sort
- Write code implementing merging two arrays from merge sort

Searching

- How do you search through a phone book?
  - Start on Page 1 and entry 1:
  - look at name. **If** name is corrrect -> Stop **else**
    - go to next entry
  - continue until find entry (or not)
  - What if youre looking for Wundersmith?
    - Would you really start at page 1 entry 1?
- **Hint: probably not since we know it’s sorted by name**

Searching

- What if the phone book wasn’t sorted?
- Then yes, you would probably go entry by entry until you found what you were looking for (or tore your hair out trying to find it)

- Searching:
  - Looking through an array/ArrayList/linked list/any list for a particular item (the “key” value)
- Two ways:
  - The linear search (we’ve done this many times)
  - The binary search (sort of like the ***sorted*** phone book search)

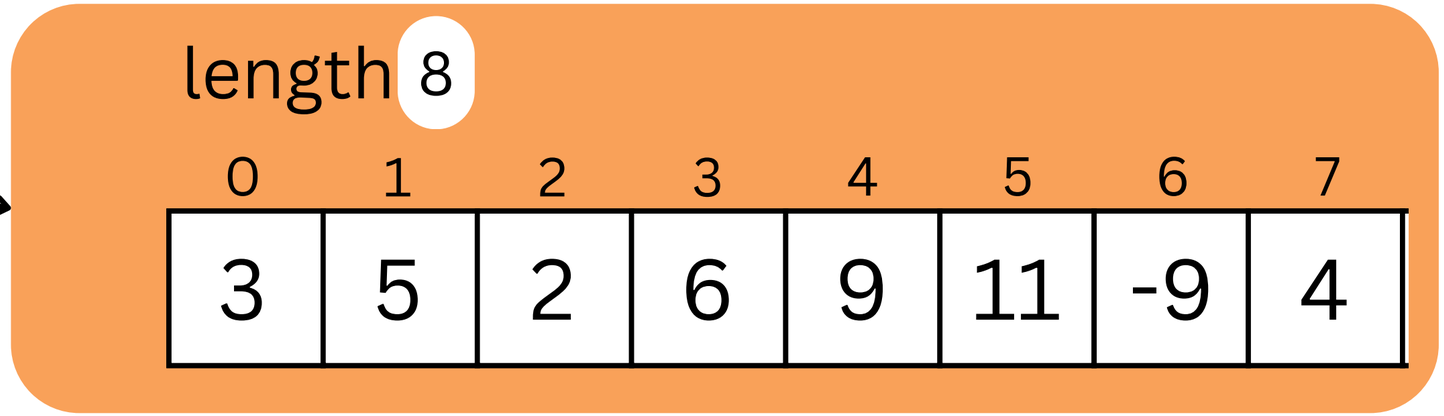
Linear Search

- Searches from beginning to end
- It has to look at each item one after the other until the key is found
- Does **not** require the list to be sorted
- Let’s take a look at in pictures (and then put the LinearSearch.java code into PythonTutor)

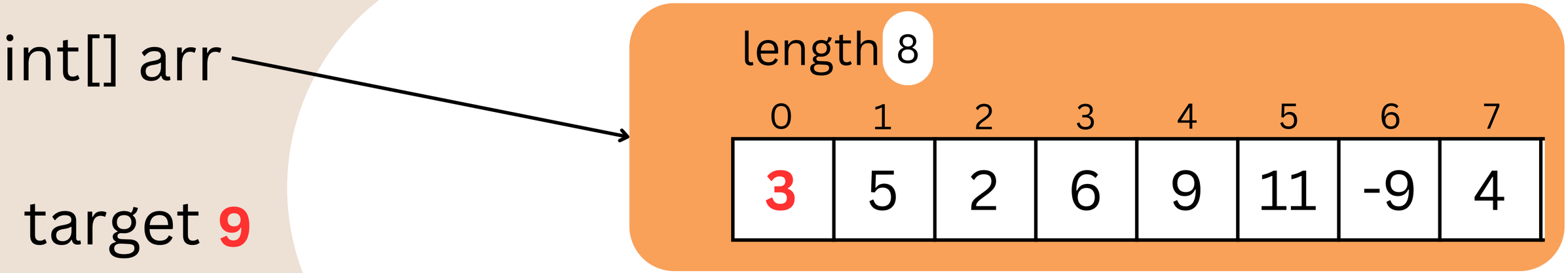
Linear Search

int[] arr

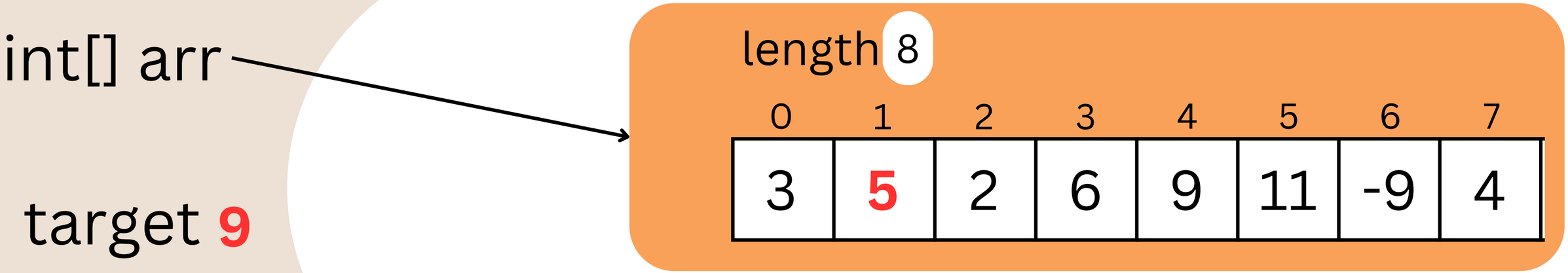
target **9**



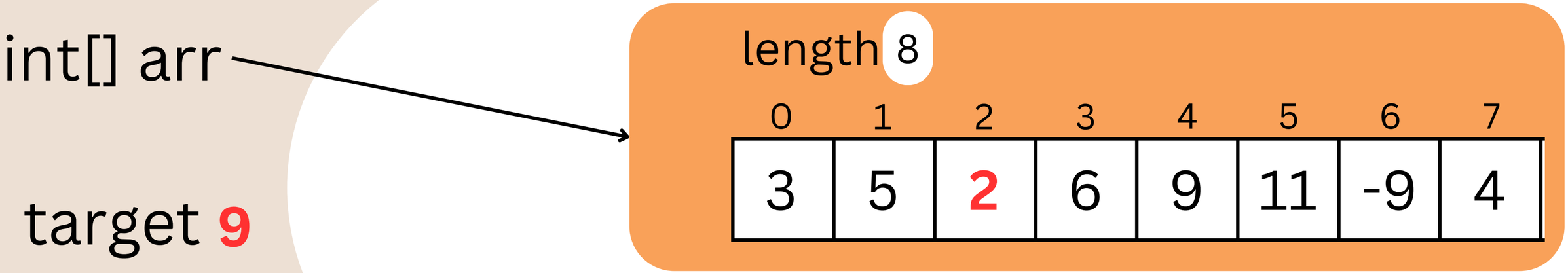
Linear Search



Linear Search



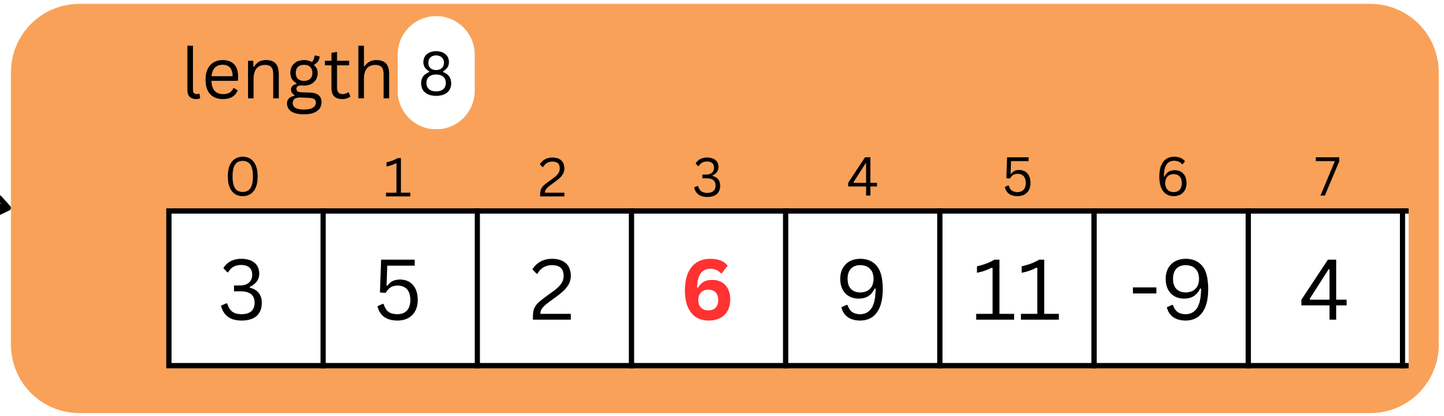
Linear Search



Linear Search

int[] arr

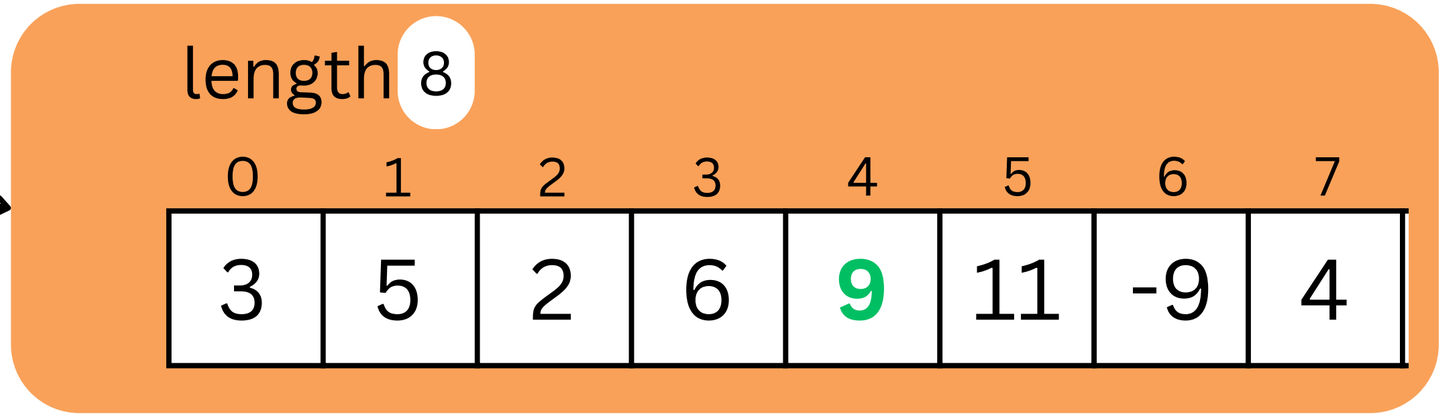
target **9**



Linear Search

int[] arr

target 9



Binary Search

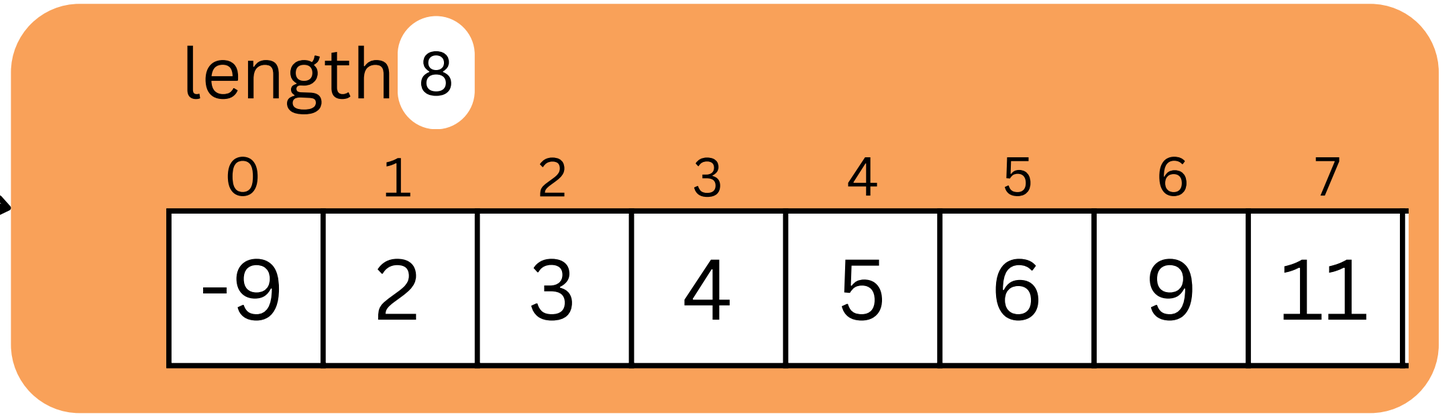
- Divides the list in half repeatedly
- Fast
- **Requires** the list to be **sorted**
- **Requires** fast random access to the list
    - which not all lists allow

- Let’s take a look at in pictures (and then put the BinarySearch.java code into PythonTutor)

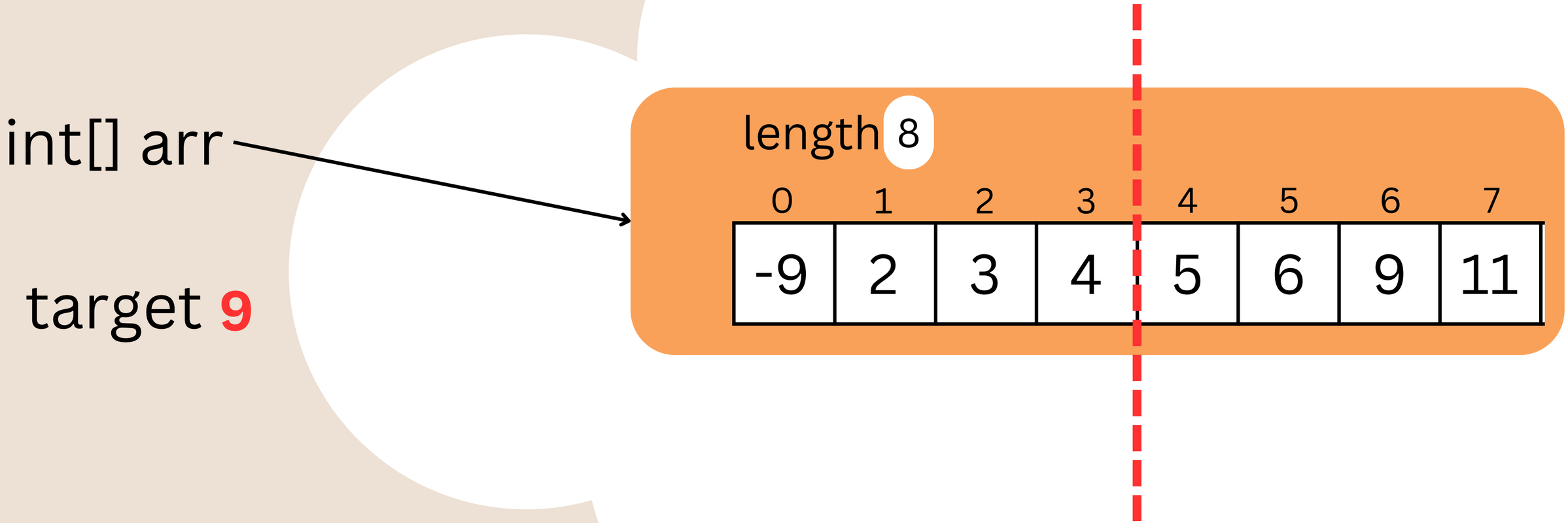
Binary Search

int[] arr

target 9



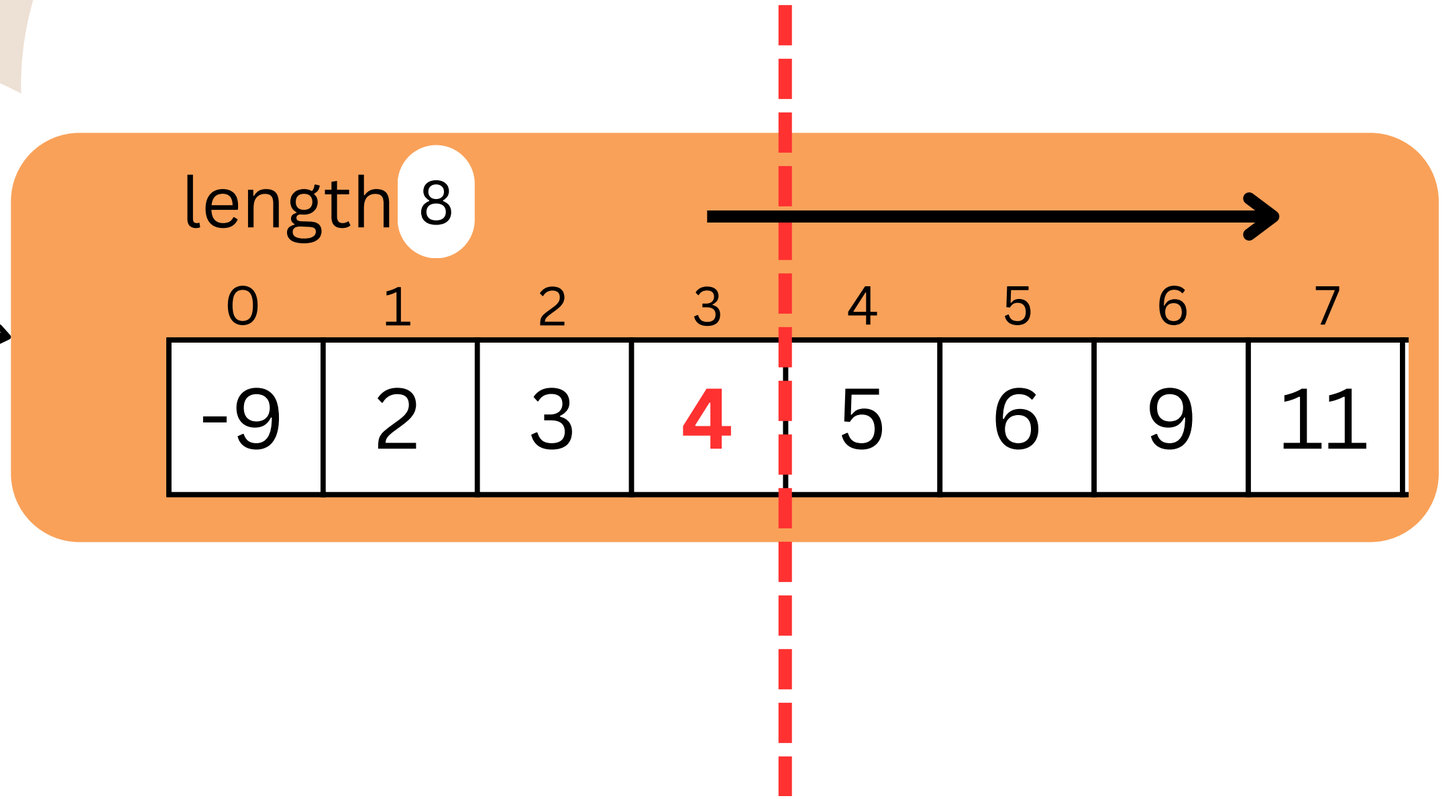
Binary Search



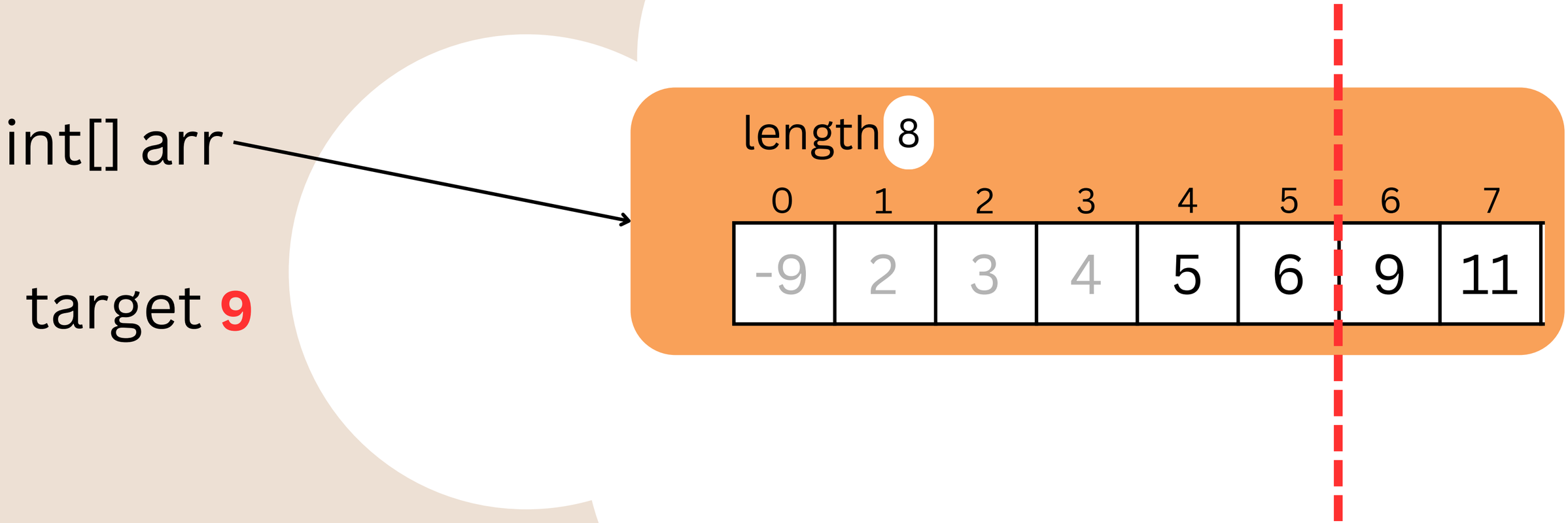
Binary Search

int[] arr

target 9



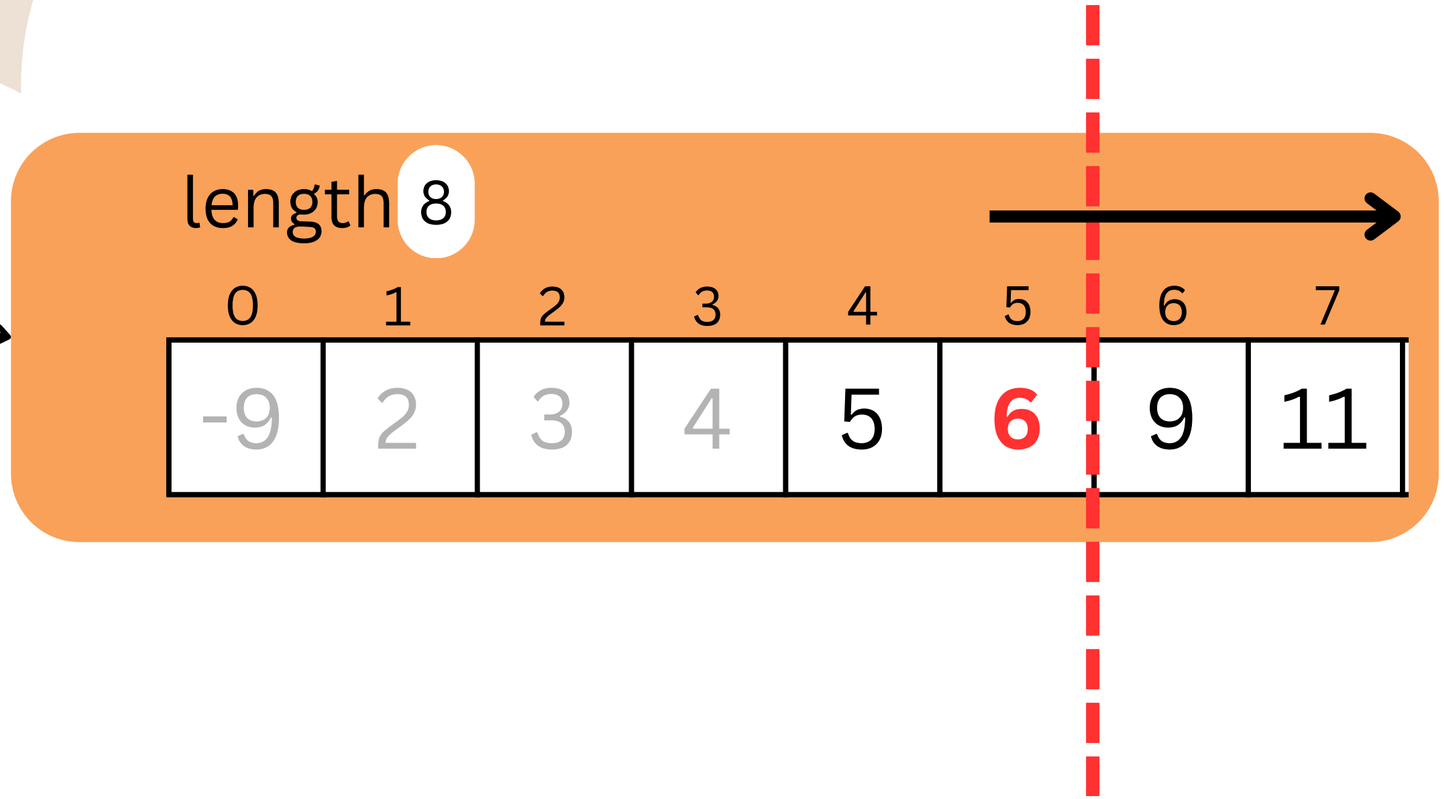
Binary Search



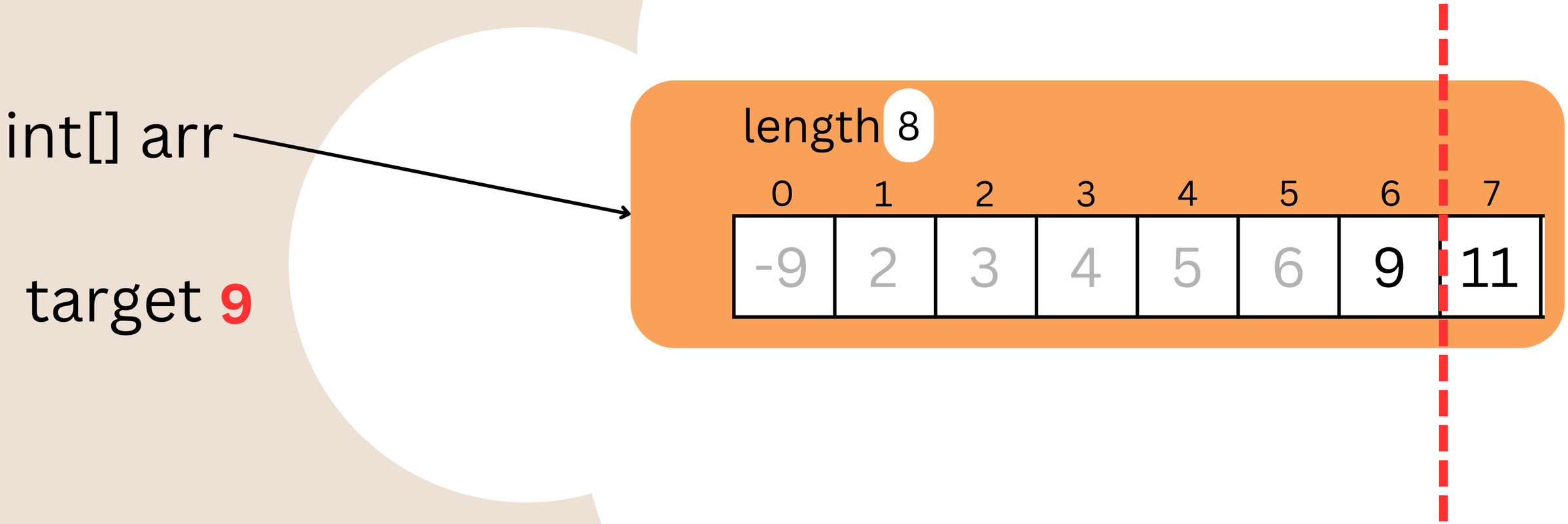
Binary Search

int[] arr

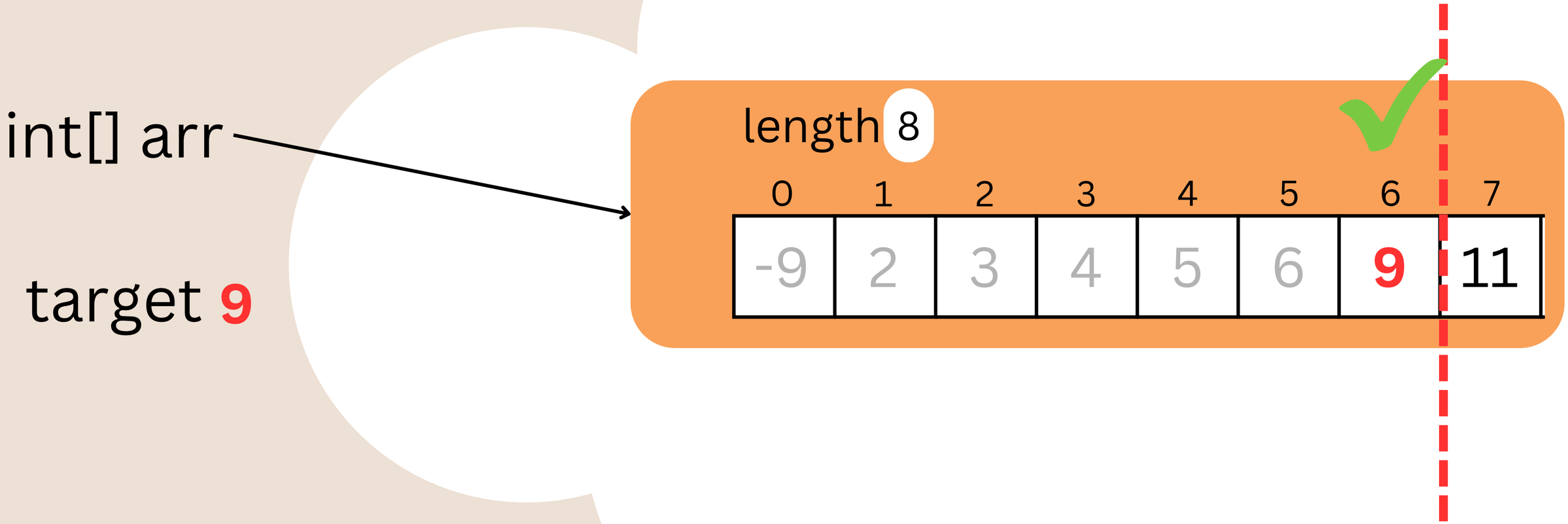
target **9**



Binary Search



Binary Search



Variable Tracking

- **Linear Search**
  - index
  - target
  - array
- **Binary Search**
  - lo index
  - hi index
  - mid index
  - target
  - array

Binary Search: What if we can't find it?

- Watch for hi < lo
  - while(lo < hi) ensures our borders are still tracking the correct range
  - if hi < lo (**impossible**) the range borders crossed and we never found the number

Analyzing Time Complexities

- Which is faster and how do we know?

Analyzing Time Complexities

- Which is faster and how do we know?
- Linear Search takes
- **Linear Time**
  - If we have 10 elements and are looking for an element found at the end of list (in the 10th spot) this takes 10 “time steps”
  - It is possible that we find it in the first spot but it’s also possible to find it in the last spot. Worst case, we need ‘n’ operations for an array of size ‘n’.
- **i.e.** it will do a number of operations that is linear in comparison with the size of the input array/list (in the worst case)

Analyzing Time Complexities

- Which is faster and how do we know?
- Binary Search
  - Problem space shrinks by half each time
    - we throw half the array out each time we iterate

Comparing Relative Speed

- The difference between the two algorithms is **huge**!
- If you double the list size:
  - The linear search doubles the iterations
  - The binary search adds only 1 iteration!

List Size	Linear iterations	Binary iterations
10	10	4
20	20	5
1000	1000	10
1,000,000	1,000,000	20
1,000,000,000	1,000,000,000	30

When to use a Binary Search

- You **need** sorted data
- You can keep it sorted as you build the data OR
- If you get a sorted list by default you can use it to!

- What if we have unsorted data?
  - We could do a linear search OR
  - sort it then do a binary search

- Keeping a list in sorted order OR sorting a list then binary searching it actually takes longer than just doing the linear search

When to use a Binary Search

- Use a binary search if:
    - You happen to have a sorted list already
    - You plan to do a LOT of searching
        - But the list doesn’t change much, so keeping it sorted is easy
    - You have lots of time to sort (overnight, maybe), but when they happen, the searches need to be FAST!
- Planning your requirements is VERY important when choosing the right algorithm

Pause and Practice

- Consider the following array:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
-9	-5	-2	0	0	3	8	10	11	12	23	44	45	90	91

- How many comparisons to a target = 9 have to happen in a linear search?
- How many comparisons to a target = 9 have to happen in a binary search?