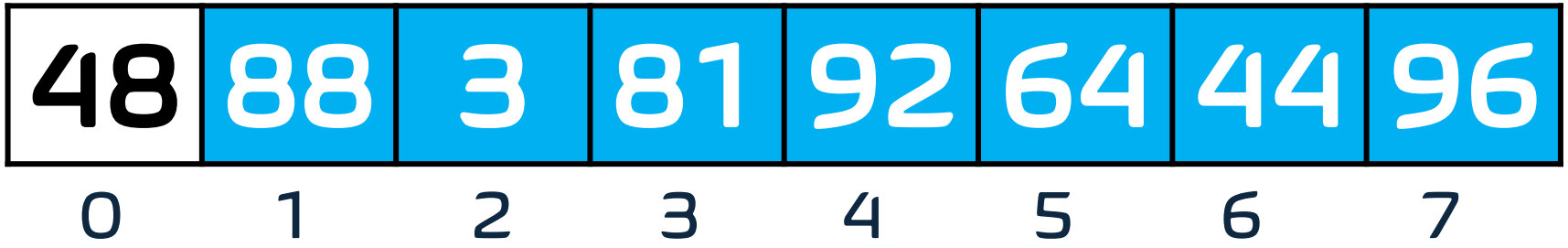
Analysis of Search Algorithms

Lec3, CMSC 142

Sequential Search

Sequential Search

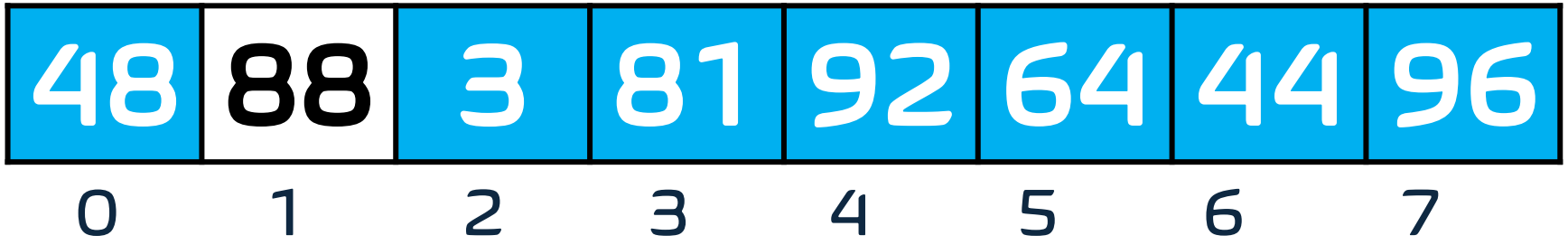
Target = 92



92 = 48 ?

Sequential Search

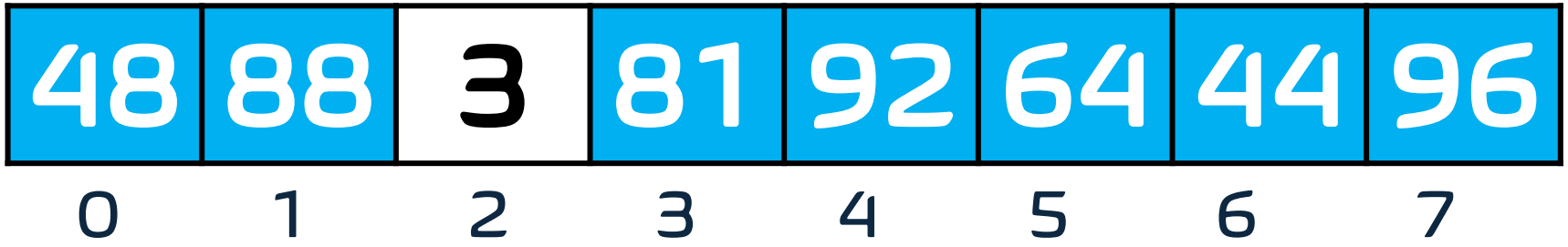
Target = 92



92 = 88 ?

Sequential Search

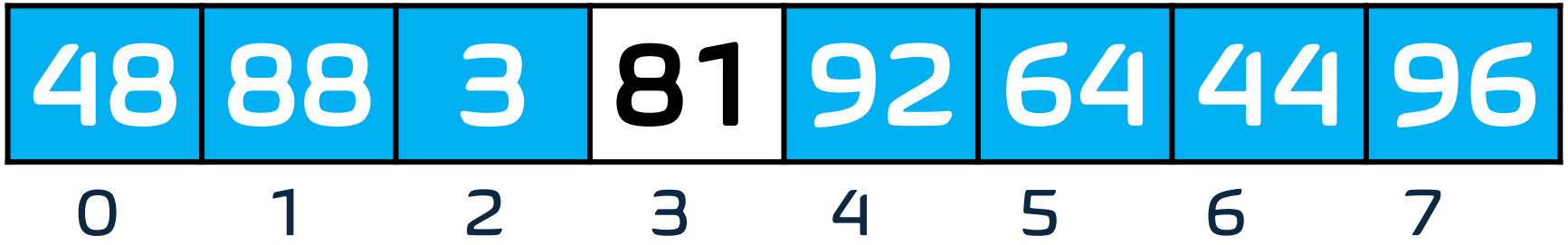
Target = 92



92 = 3 ?

Sequential Search

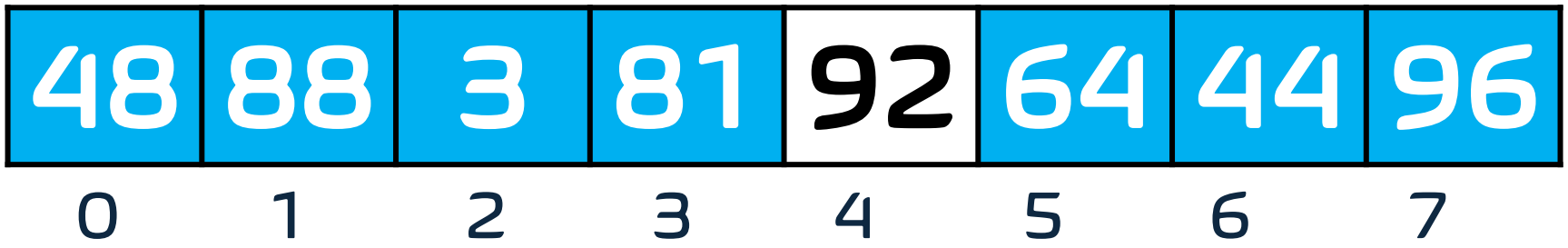
Target = 92



92 = 81 ?

Sequential Search

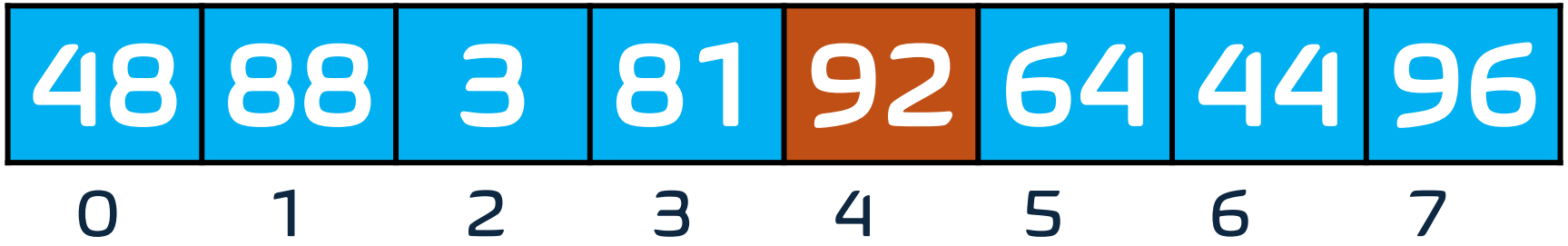
Target = 92



92 = 92 ?

Sequential Search

Target = 92



92 = 92 ?

Sequential Search Algorithm

```
LinearSearch(A, size, target):
    for i = 0 to size-1:
        if (A[i]==i):
            return i
    return -1
```

Sequential Search

- Best Case: O(1)
- Average Case: O(n)
- Worst Case: O(n)

Sequential Search

- Also called Linear search
- Simplest of all searching algorithms
- Uses brute-force approach to locate a target element in a collection

Sequential Search

List of Steps

1. Start at first element of collection
2. Examine each subsequent element
3. Stop if either the matching element is found, or each element of the collection has been examined

Sequential Search

Used when?

- When collection may or may not be ordered
- With no further knowledge about the collection, Sequential Search just gets the job done in a brute-force manner

Sequential Search

- For small collections of unordered elements, Sequential Search is easy to implement and reasonably efficient

- Worst case is when you search for an item not in the collection: inspect every element only to come up empty handed.

Sequential Search Variations

Move to front on success

- Suitable if there's increased likelihood that item being searched for will be searched again
- Also, there's a desire to avoid the cost of moving lots of elements

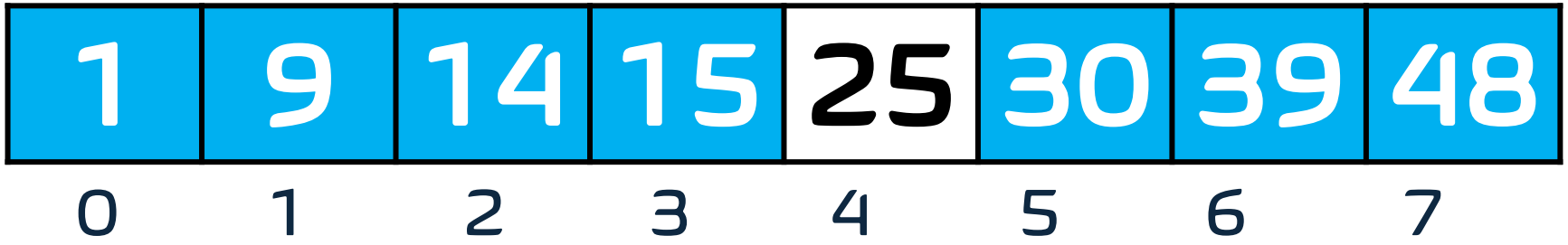
Move to end on success

- Used if element is unlikely to be searched for multiple times
- Moving element to the end when found will improve performance of future searches

Binary Search

Binary Search

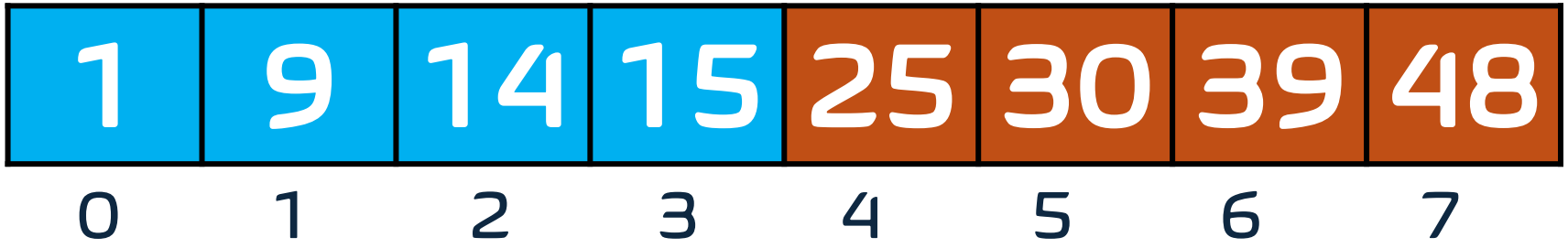
Target = 14



14 < 25 ?

Binary Search

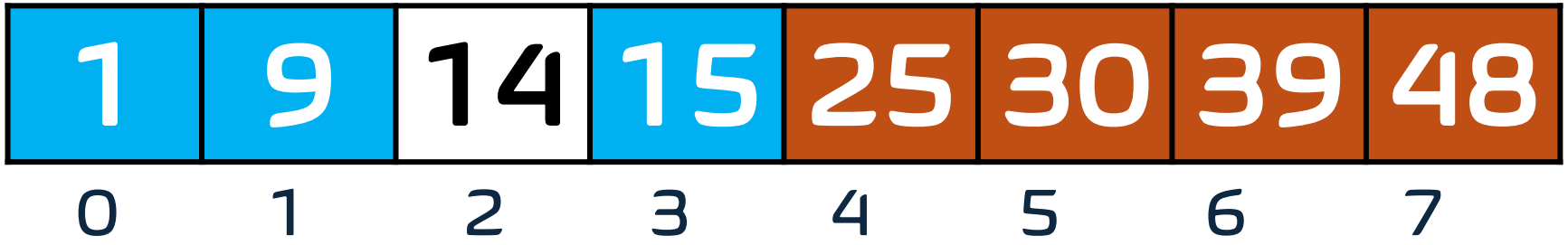
Target = 14



14 < 25 ? YES

Binary Search

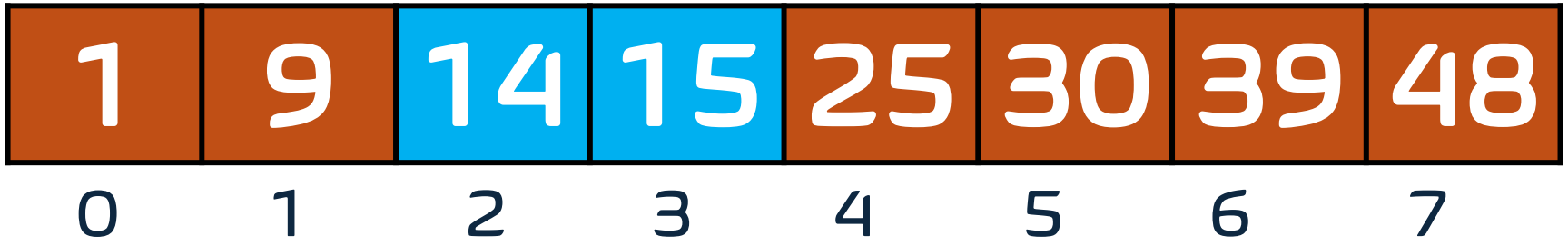
Target = 14



14 < 14 ?

Binary Search

Target = 14



14 < 14 ? NO

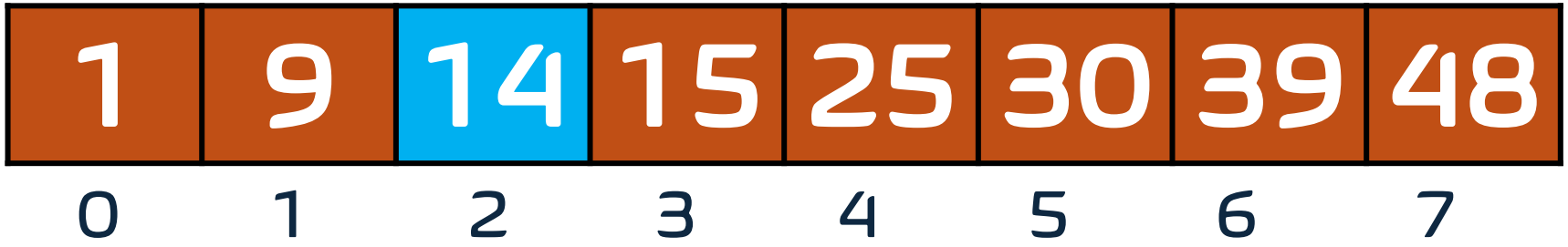
Binary Search

Target = 14

14 < 15 ?

Binary Search

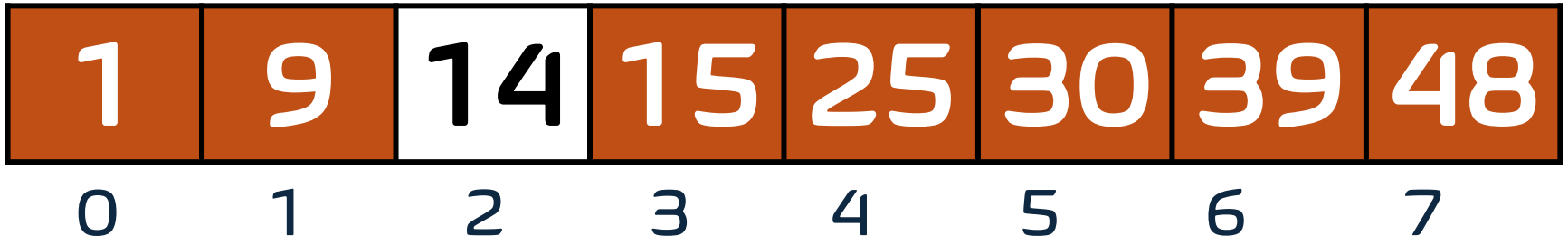
Target = 14



14 < 15 ? YES

Binary Search

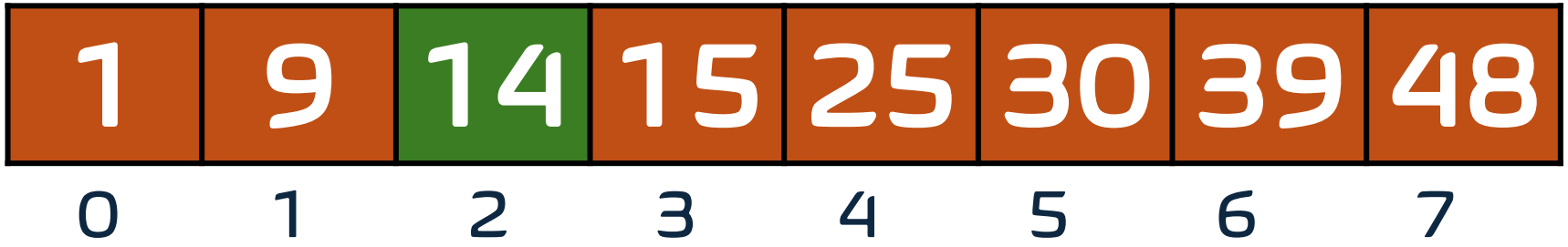
Target = 14



14 = 14 ?

Binary Search

Target = 14



14 = 14 ? YES

Binary Search Algorithm

```
Binary_Search(A, t):
    low = 0
    high = n-1
    while low ≤ high:
        ix = ceil ( (low+high) / 2 )
        if t == A(ix):
            return true
        else if t < A(ix):
            high = ix – 1
        else:
            low = ix + 1
    return false
```

Binary Search

- Best Case: O(1)
- Average Case: O(log n)
- Worst Case: O(log n)

Binary Search

List of Steps

1. Divide the sorted collection in half until the sought-for item is found or until it is determined that the item isn't in the collection.
2. Adds a small amount of complexity for large performance gains.

Log N

Given 8 elements, what is the maximum no. of iterations it will take to find the element? 3

- How about 16 elements? 4
- How about 64 elements? 6

Sorting and Searching

- In some cases, doing a preliminary sorting of the collection can help make the search faster

- But there are costs for maintaining a sorted collection, especially if you have frequent insertions or deletions