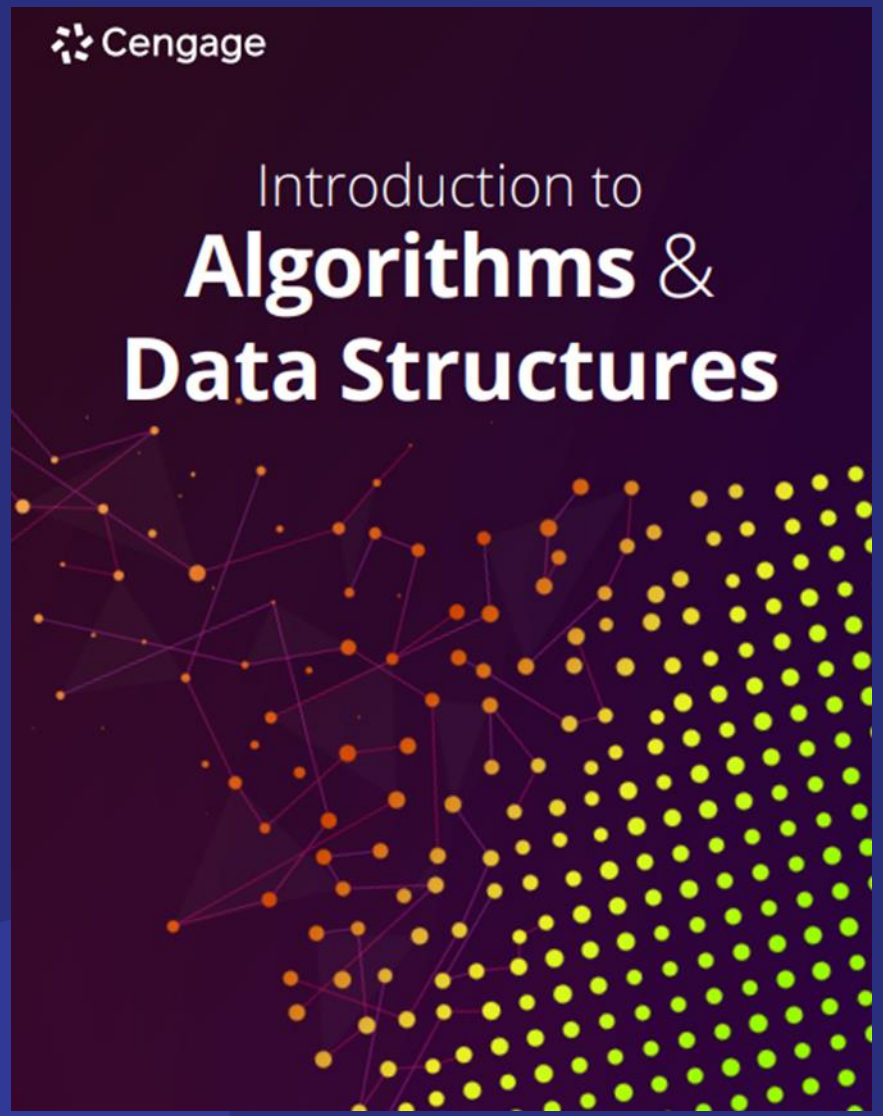


Introduction to Algorithms and Data Structures, 1e

Chapter 5: Search Algorithms

Chapter Objectives

- When you complete this chapter, you will be able to:
  - 5.1 Discuss various search algorithms.
  - 5.2 Apply sequential search algorithms to a problem.
  - 5.3 Apply a binary search algorithm to a problem.
  - 5.4 Apply a recursive binary search algorithm to a problem.

Section 5.1

Introduction to Search Algorithms

5.1 Introduction to Search Algorithms (1 of 2)

- Data structures are created to organize information
- The operation of retrieving a specific data item is called a **search process**
- It is common to refer to the data stored using the user-defined data type as **records** or **nodes** of the larger data structure
- A **search algorithm** is designed to provide a way to access a particular record or node in a data structure
- A **key** is a set of fields used to compare two objects in search and sort algorithms
- The key of interest is called the **target value**

5.1 Introduction to Search Algorithms (2 of 2)

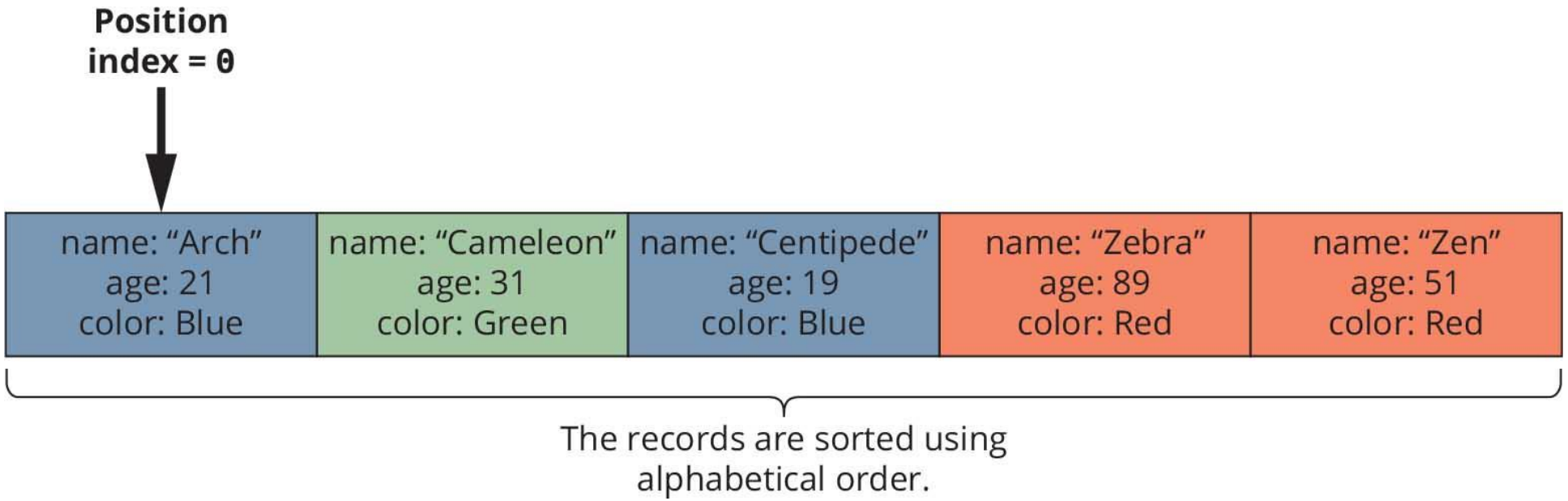


Figure 5-1 A set of records sorted alphabetically using the name field

Finding a Friend's Phone Number

- When searching for a friend's contact information on your phone, you most likely use a search algorithm
  - In most modern devices, more than one type of algorithm and a significant number of data structures are used

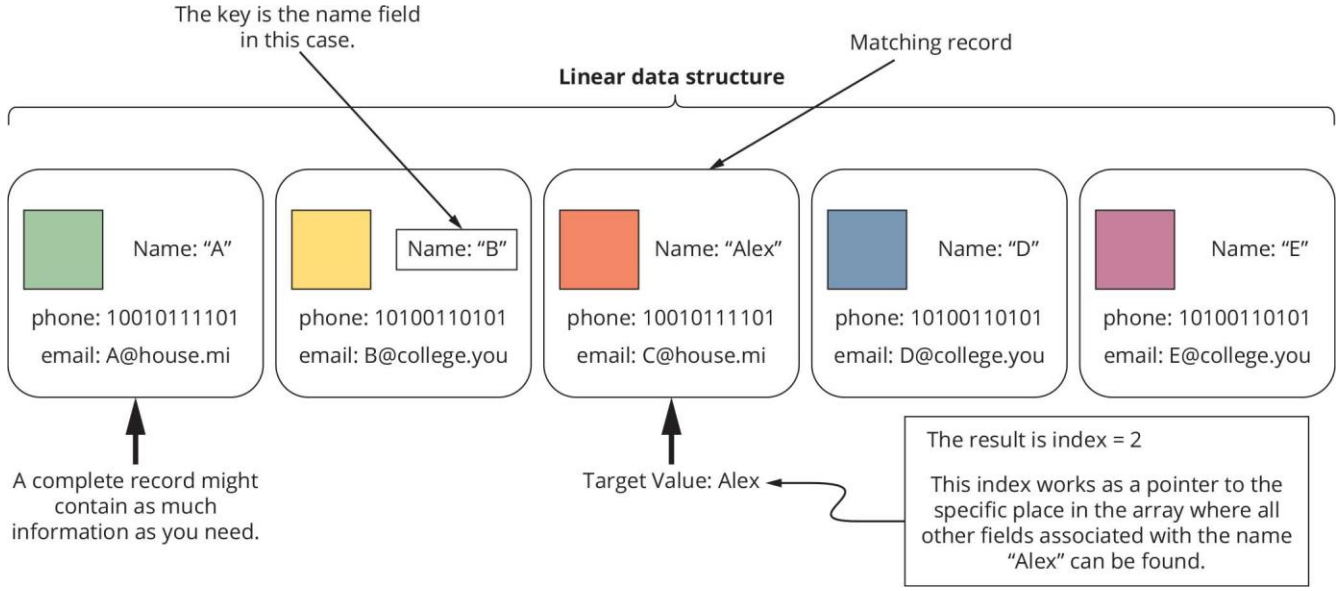


Figure 5-2 A search algorithm must provide a way to retrieve the record with the key value that matches the target value

Sorted and Unsorted Searches

- When implementing a search algorithm, it is essential to know whether the records are sorted
  - If the records are sorted, you can scan a couple of names and deduce if you have to look forward or backward
  - When searching in an array of unsorted records, you must check each record until you find the desired one

Section 5.2

Sequential Search

5.2 Sequential Search

- **Sequential search** is a search algorithm implemented in a linear data structure where the data structure is traversed until the record with the matching key is found

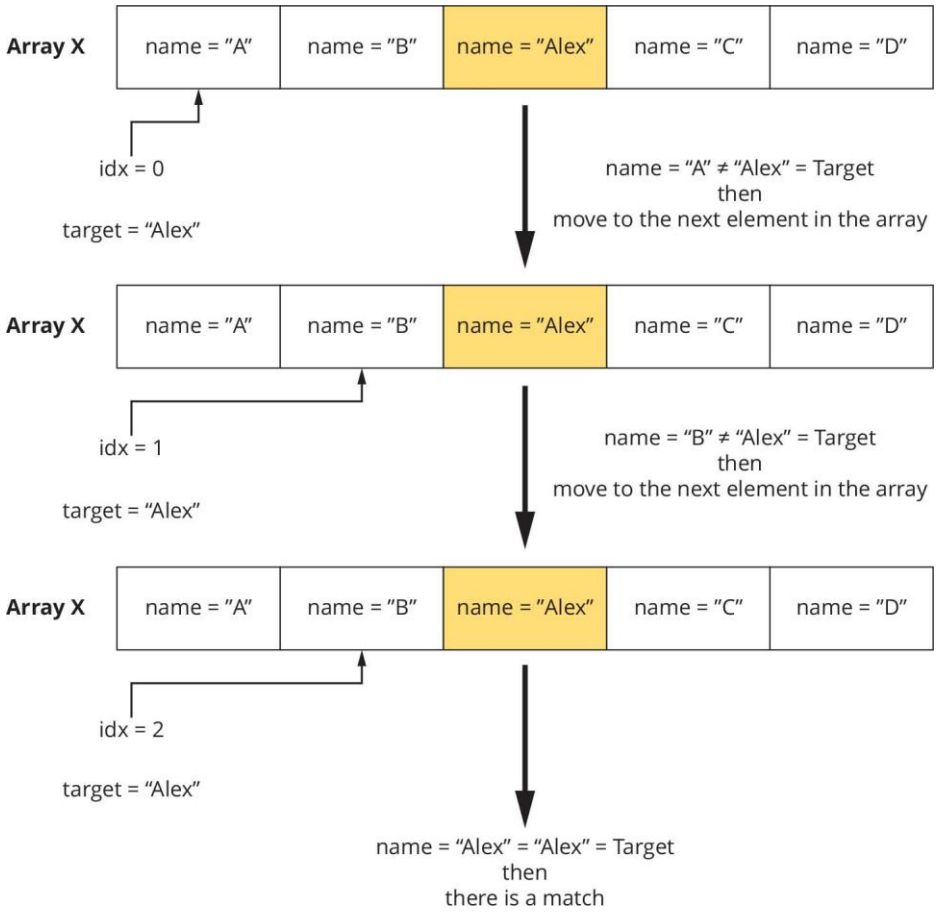


Figure 5-4 Sequential search over an unsorted array

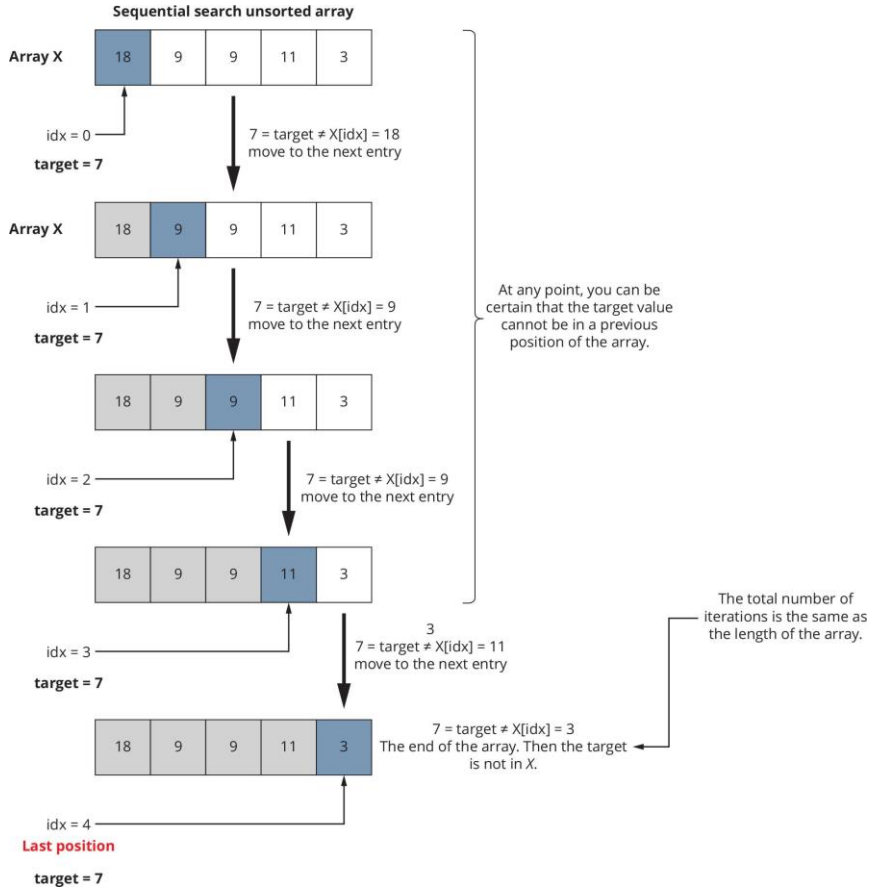
Sequential Search Algorithm Details

- Suppose `X` is an array of integers and `target` is the target value
  - An implementation of a sequential search algorithm that returns the index of the first occurrence of the target value is described below

```
sequential_search(X, target)
    N = len(X)
    for j = 0 to N-1 step 1
        if X[j] = target
            return j
    return Null
```

Time Complexity of Sequential Search (1 of 2)

Figure 5-5 Worst case for sequential search in the unsorted case



Time Complexity of Sequential Search (2 of 2)

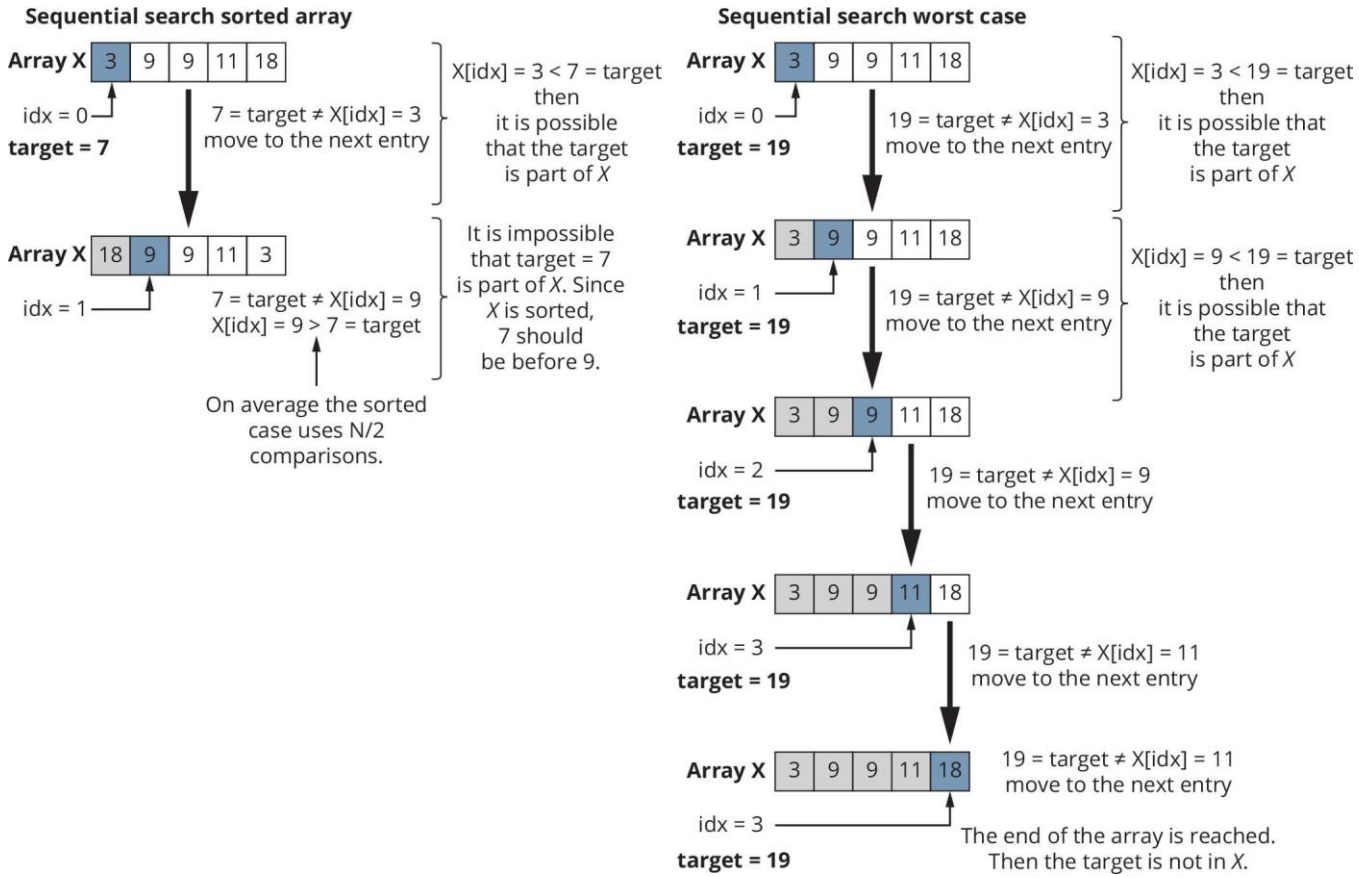


Figure 5-6 Worst case for sequential sort in the sorted case

Activity 5.1: Knowledge Check

1. A(n) _____ algorithm is designed to provide a way to access a particular record or node in a data structure.
2. A set of fields used to compare two objects in search and sort algorithms is called a(n) _____.
3. A key of interest is called the _____ value.

Activity 5.1: Knowledge Check Answers

1. A(n) _____ algorithm is designed to provide a way to access a particular record or node in a data structure.

   Answer: search

2. A set of fields used to compare two objects in search and sort algorithms is called a(n) _____.

   Answer: key

3. A key of interest is called the _____ value.

   Answer: target

Section 5.3

Binary (Interval) Search

5.3 Binary (Interval) Search

- **Binary search** is a search algorithm in sorted arrays that finds the target value by splitting the array in two at each iteration
  - The algorithm starts by finding the element in the middle of the array

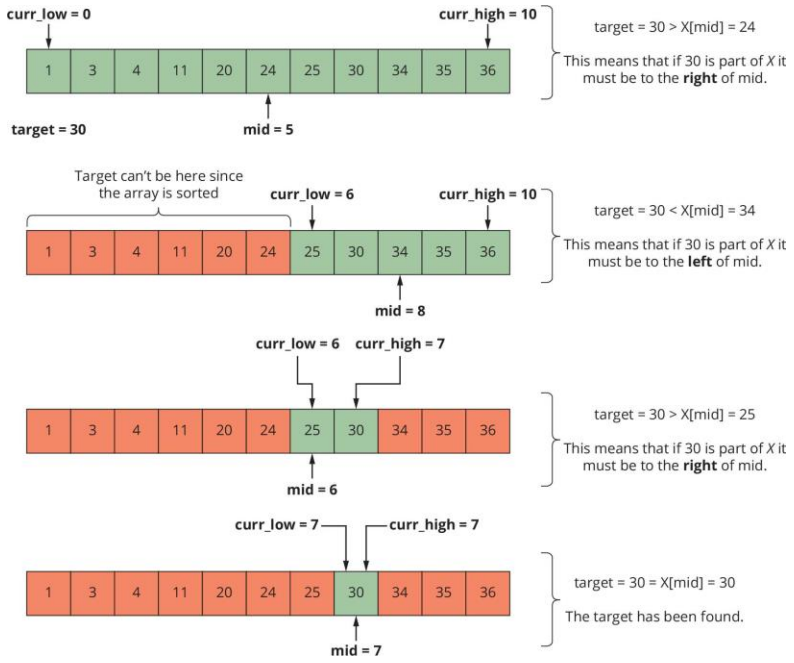


Figure 5-7 Basic binary search procedure

Searching Without Checking All Elements (1 of 2)

- A binary search algorithm doesn't need to check every element in the data structure to find the target, even if the element is not present

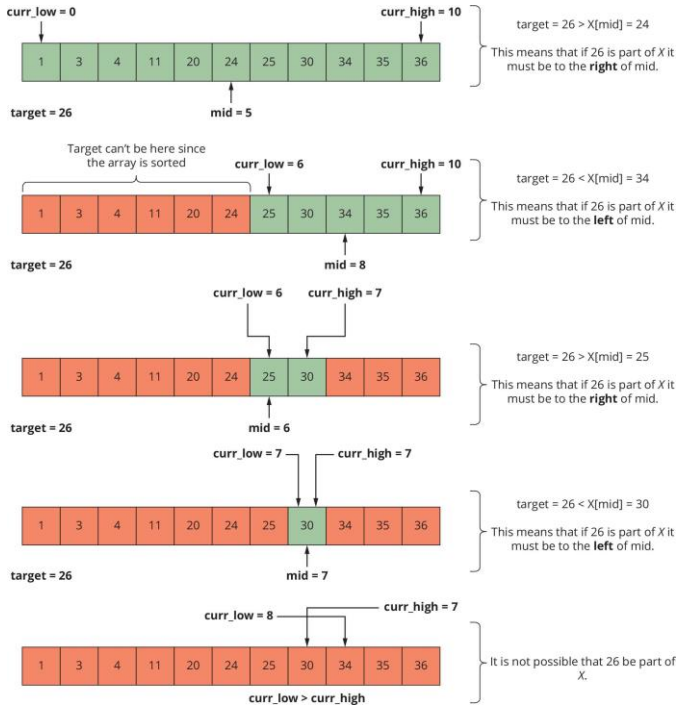
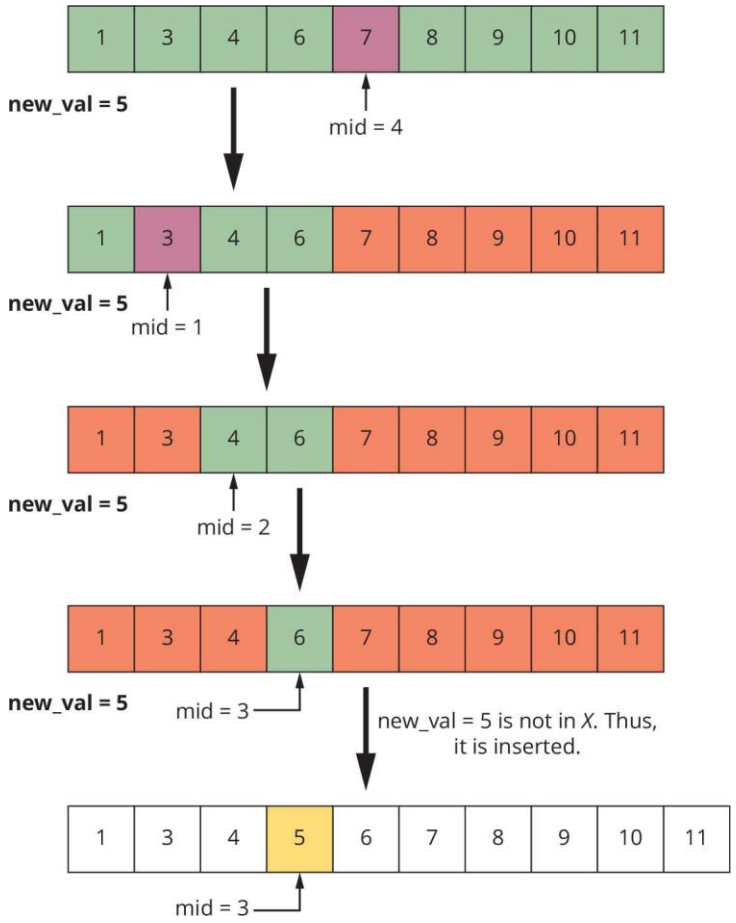


Figure 5-8 Binary search indices used to know where the sub-array starts and ends and provide a way to see that an element is not in the array

Searching Without Checking All Elements (2 of 2)

Figure 5-9 Inserting an element using binary search



Binary Search Algorithm

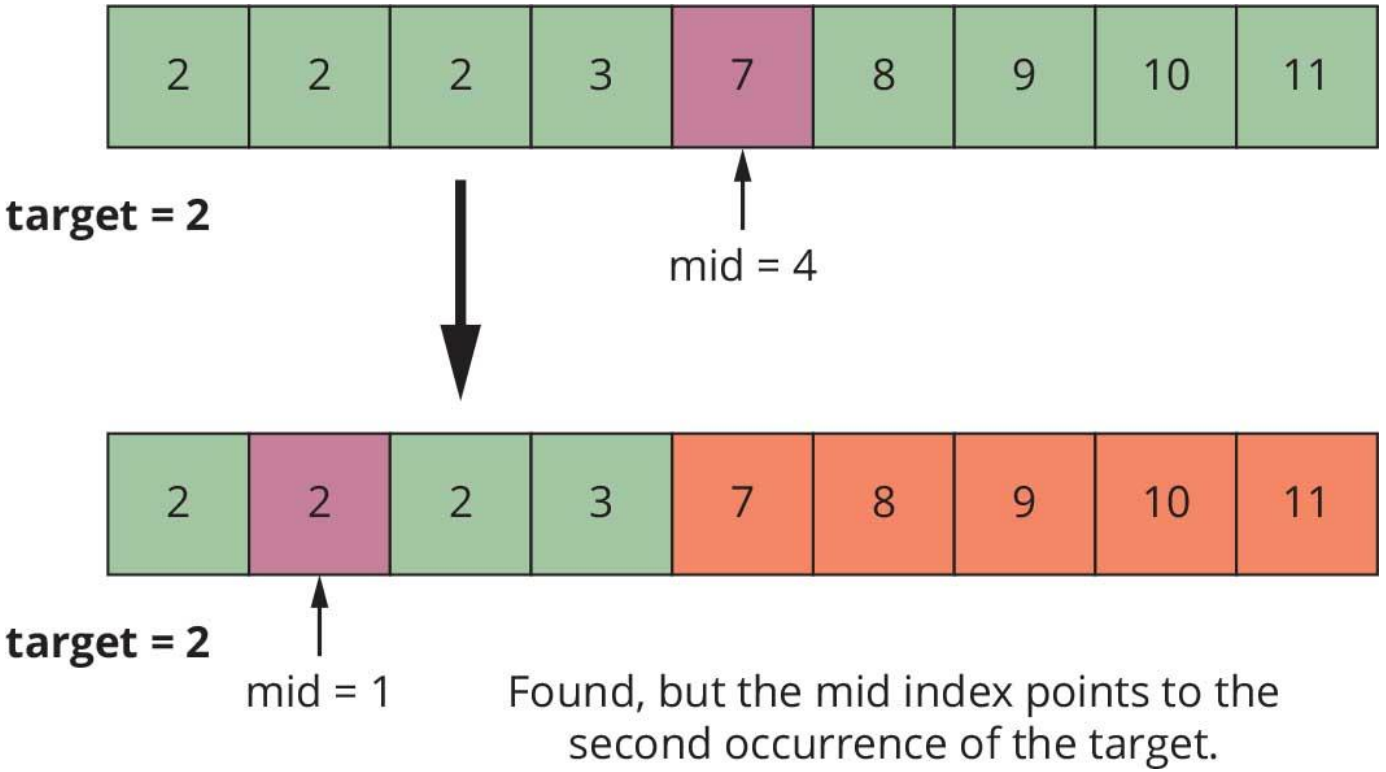


Figure 5-10 Binary search might not give the index of the first occurrence of a value

Time Complexity of Binary Search

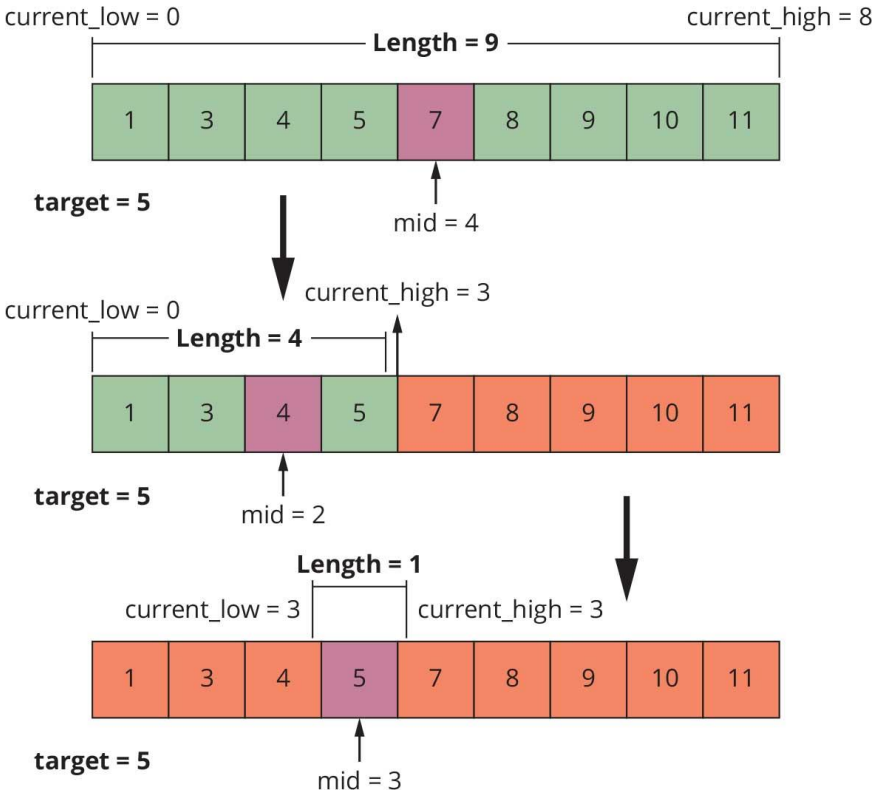


Figure 5-11 At each iteration, the length of the interval where the target value might be is reduced by half

Section 5.4

Recursive Binary Search Method

5.4 Recursive Binary Search Method

- When using recursion, it is necessary to have a base case, the case in which the function will no longer call itself (recurse)
- The other part is the recursion, which is the mechanism or description of how you split the current case into smaller cases
- The binary search algorithm can be described using recursion
  - Finding the mid element and comparing it to the target value is repeated each time to a smaller sub-array
  - The binary search algorithm is an excellent candidate to be described using recursion

Binary Search Using Recursion

- The simplest case in the binary search method is when the array has only one element
  - In such a case, the only thing to do is to compare the element to the target value and return the corresponding index
- The binary search recursion algorithm consists of doing a binary search in the two sub-arrays: the one from `low` to `mid-1` or `mid+1` to `high`, depending on which of those could contain the target value

Space Complexity of Binary Search (1 of 2)

- In most computer architectures, the parameters are stored in memory when you call a function
  - Calling a function requires a memory space equal to the data used inside the function
    - For a recursive function, each call requires reserving new space in memory for the parameters

Space Complexity of Binary Search (2 of 2)

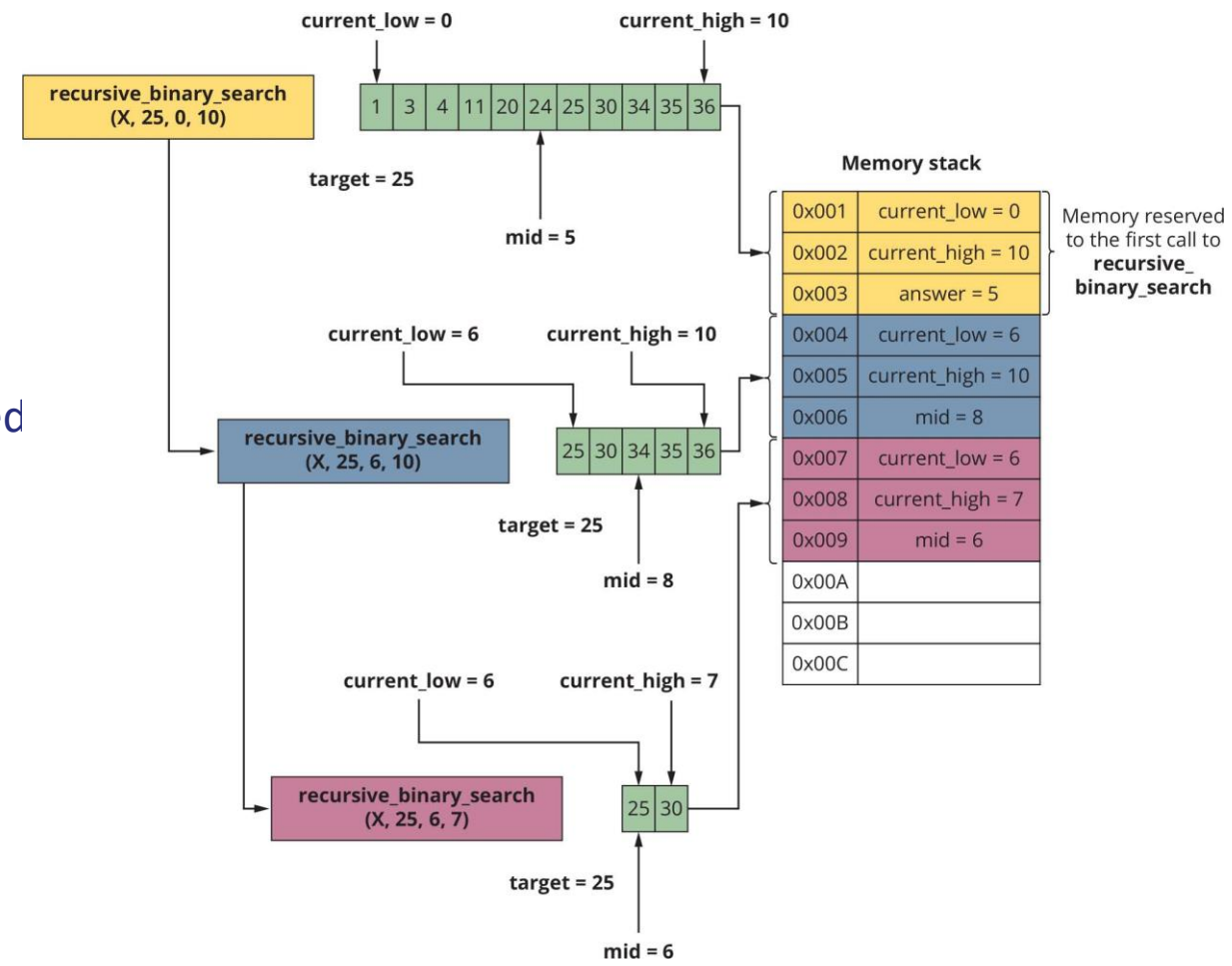


Figure 5-12 More memory is required each recursive call during the search

Activity 5.2: Knowledge Check

1. Which search algorithm consists of sorted arrays that find the target value by splitting the array in two at each iteration?
2. The binary search algorithm starts by finding the element in the _____ of the array.
3. True or False: Binary search is an algorithm that must check every element in the data structure to find the target.

Activity 5.2: Knowledge Check Answers

1. Which search algorithm consists of sorted arrays that find the target value by splitting the array in two at each iteration?

   Answer: binary search

2. The binary search algorithm starts by finding the element in the _____ of the array.

   Answer: middle

3. True or False: Binary search is an algorithm that must check every element in the data structure to find the target.

   Answer: False

Activity 5.3: Discussion Questions

- Explain the importance of search algorithms.
- Discuss how search algorithms are used to retrieve data.
- Review sequential, binary, and recursive binary search algorithms, list their uses, and describe how each of them is applied to problems.

Self-Assessment

- What previous experience do you have using the search algorithms described in the chapter?
- Give an example of how search algorithms are used in everyday life (e.g., using a dictionary).
- Do you think you will apply what you've learned about search algorithms in your future career?

Summary

Click the link to review the objectives for this presentation.

Link to Objectives