CST209

Object-oriented Programming C++

(Week 12)

Prepared by Dr. Teo Bee Guan

Presented by Venantius Kumar Sevamalai

Content

- Search Algorithms
- Sort Algorithms

Search Algorithms

- A search algorithm is a method of locating a specific item in a larger collection of data.

- For example, it's very common for programs not only to store and process data stored in arrays, but to search arrays for specific items.

Linear Search

- The linear search is a very simple algorithm. Sometimes called a sequential search , it uses a loop to sequentially step through an array, starting with the first element.

- It compares each element with the value being searched for and stops when either the value is found or the end of the array is encountered.

In-class Practice: Example 1

In-class Exercise 1

Modify the linear search function from the previous code example by converting it a function template that can work for either an integer or a char array.

Test your function with an array of integer and an array of characters.

Inefficiency of the Linear Search

- The advantage of the linear search is its simplicity. It is very easy to understand and implement.

- Its disadvantage, however, is its inefficiency.

- If the array being searched contains 20,000 elements, the algorithm will have to look at all 20,000 elements in order to find a value stored in the last element.

Binary Search

- The binary search is an algorithm that is much more efficient than the linear search.

- Its only requirement is that the values in the array be **sorted in order**.

- Instead of testing the array's first element, this algorithm starts with the element in the middle. If that element happens to contain the desired value, then the search is over.

https://yongdanielliang.github.io/animation/web/BinarySearchNew.html

In-Class Exercise 2

Write a binarySearch function based on the pseudocode below:

```
Set first index to 0.
Set last index to the last subscript in the array.
Set found to false.
Set position to -1.
While found is not true and first is less than or equal to last
   Set middle to the subscript halfway between array[first]
   and array[last].
   If array[middle] equals the desired value
      Set found to true.
      Set position to middle.
   Else If array[middle] is greater than the desired value
      Set last to middle - 1.
   Else
      Set first to middle + 1.
   End If.
End While.
Return position.
```

The function should have three parameters: array, size of array and target value. Test your function with an integer array.

The Efficiency of the Binary Search

- Obviously, the binary search is much more efficient than the linear search.
- Every time it makes a comparison and fails to find the desired item, it eliminates half of the remaining portion of the array that must be searched.
- For example, consider an array with 1,000 elements. If the binary search fails to find an item on the first attempt, the number of elements that remains to be searched is 500. If the item is not found on the second attempt, the number of elements that remains to be searched is 250. This process continues until the binary search has either located the desired item or determined that it is not in the array.
- With 1,000 elements, this takes no more than 10 comparisons.

Sorting Algorithms

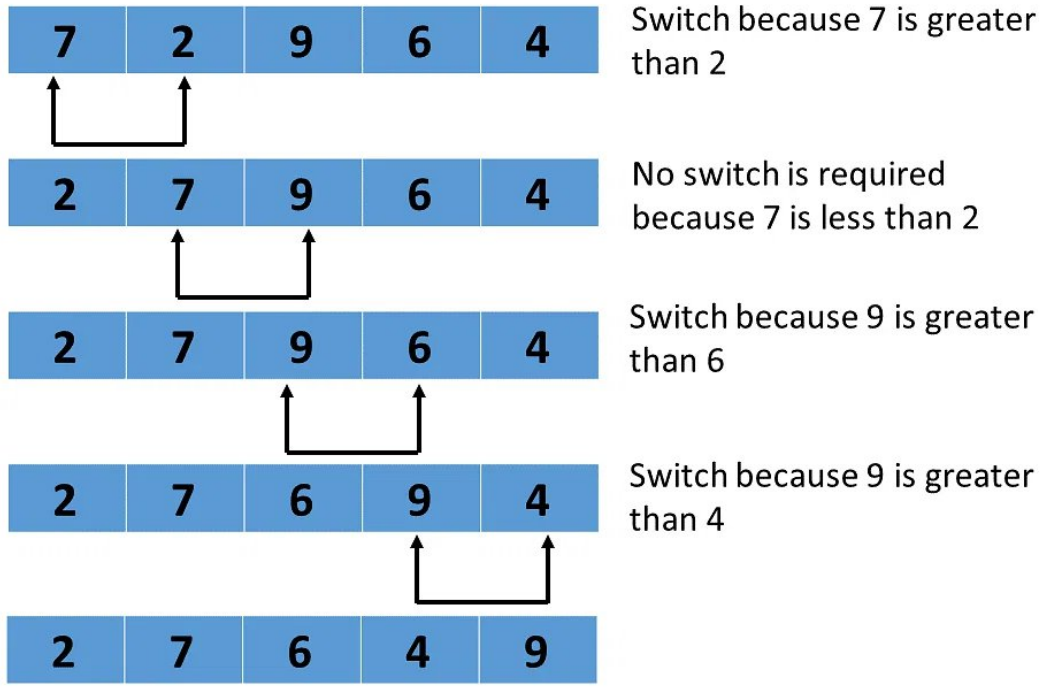
- Often the data in an array must be sorted in some order. Customer lists, for instance, are commonly sorted in alphabetical order. Student grades might be sorted from highest to lowest. Product codes could be sorted so all the products of the same color are stored together.

- To sort the data in an array, the programmer must use an appropriate sorting algorithm .

- A sorting algorithm is a technique for scanning through an array and rearranging its contents in some specific order.

Bubble Sort

- Bubble sort works by examining each set of adjacent elements in the string, from left to right, switching their positions if they are out of order.



https://visualgo.net/en/sorting

In-Class Exercise 3

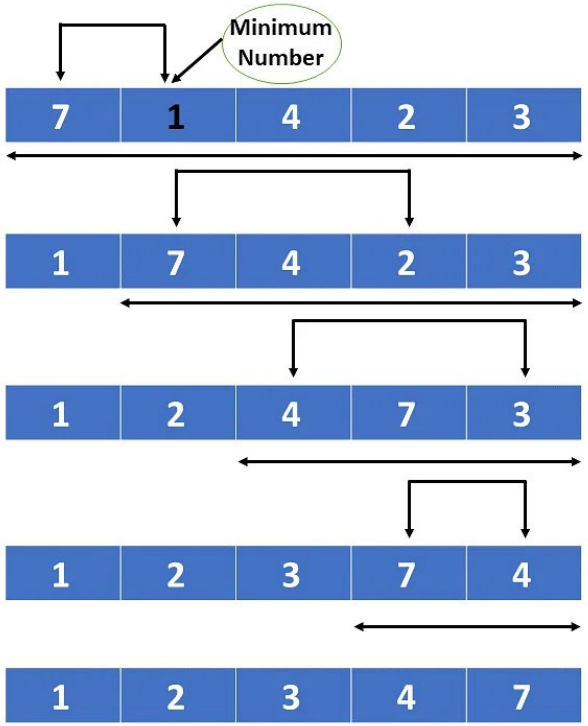
Write a bubble sort function based on the pseudocode below:

```
Do
    Set swap flag to false.
    For count is set to each subscript in array from 0 through the
            next-to-last subscript
        If array[count] is greater than array[count + 1]
            Swap the contents of array[count] and array[count + 1].
            Set swap flag to true.
        End If.
    End For.
While any elements have been swapped.
```

The function should have two parameters: array, size of array. Test your function with an integer array.

Selection Sort

- Selection sort selects the smallest element from an unsorted array in each iteration and places that element at the beginning of the unsorted array.



https://visualgo.net/en/sorting

In-Class Exercise 4

Write a selection sort function based on the pseudocode below:

```
For startScan is set to each subscript in array from 0 through the
          next-to-last subscript
   Set index variable to startScan.
   Set minIndex variable to startScan.
   Set minValue variable to array[startScan].

   For index is set to each subscript in array from (startScan + 1)
          through the last subscript
      If array[index] is less than minValue
         Set minValue to array[index].
         Set minIndex to index.
      End If.
   End For.
   Set array[minIndex] to array[startScan].
   Set array[startScan] to minValue.
End For.
```

The function should have two parameters: array, size of array. Test your function with an integer array.

See you next class