PPT Presentation

By

U.Joshna

[03-02-2022]

Using Arrays,
Explain bubble sort program

Bubble Sort:

Bubble Sort is a sorting algorithm (an algorithm that puts elements of a list in a certain order). The simplest sorting algorithm is Bubble Sort. In the Bubble Sort, as elements are sorted they gradually "bubble up" to their proper location in the array, like bubbles rising in a glass of soda.

To sort an array there will be n-1 passes where n is the number of elements in the array. In the above diagram there are seven elements in an array so there will be 7-1=6 passes.

Array Element	89	76	45	92	67	12
Pass#1	89	76	92	67	45	99
Pass#2	89	92	76	67	99	45
Pass#3	92	89	76	99	67	45
Pass#4	92	89	99	76	67	45
Pass#5	92	99	89	76	67	45
Pass#6	99	92	89	76	67	45

The Bubble Sort works by iterating down an array to be sorted from the first element to the last, comparing each pair of elements and switching their positions if necessary. This process is repeated as many times as necessary, until the array is sorted.

When this first pass through the array is complete, the Bubble Sort returns to elements one and two and starts the process all over again. The Bubble Sort has stopped when it is finished examining the entire array and no "swaps" are needed.

Example:

```
using System;
namespace Bubble__Sort
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] number = { 89, 76, 45, 92, 67, 12, 99 };
            bool flag = true;
            int temp;
            int numLength = number.Length;

         //sorting an array
                    for (int i = 1; (i <= (numLength - 1)) && flag; i++)
                    {
                        flag = false;
```

```
for (int j = 0; j < (numLength - 1); j++)
            {
                if (number[j + 1] > number[j])
                {
                    temp = number[j];
                    number[j] = number[j + 1];
                    number[j + 1] = temp;
                    flag = true;
                }
            }
        }
 //Sorted array
            foreach (int num in number)
            {
                Console.Write("\t {0}", num);
            }
            Console.Read();

        }
```

```
    }
}
```

Output:

```
C:\Users\KOLLI TEJASWI\source\repos\Bubble  Sort\bin\Debug\Bubble  Sort.exe
99      92      89      76      67      45      12
```

Using Arrays,
Explain Linear Search

Linear Search:

- Start from the leftmost element of arr[] and one by one compare x with each element of arr[]
- If x matches with an element, return the index.
- If x doesn’t match with any of elements, return -1.

Example:

```
using System;
namespace LinerSearch

{
    class Program
    {
        static void Main(string[] args)
        {
            int[] a = new int[100];
            Console.WriteLine("Enter the number of elements you want to add in
the array ?");
            string s = Console.ReadLine();
            int x = Int32.Parse(s);
```

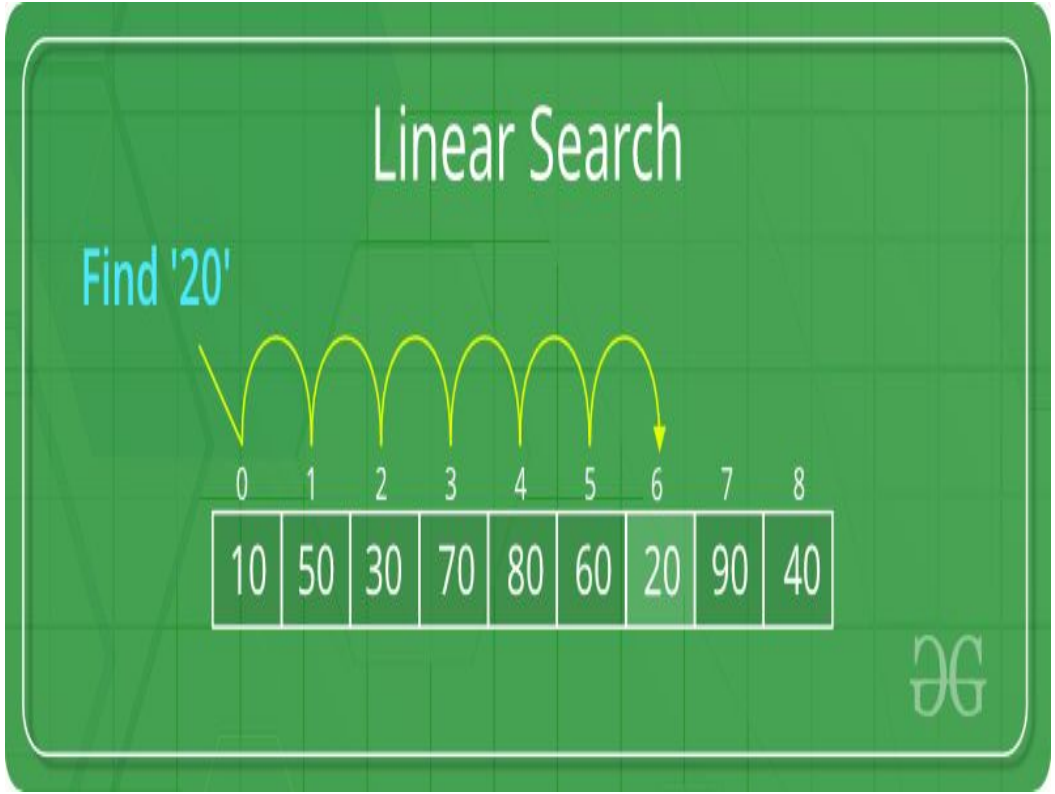
```
Console.WriteLine("1,2,3,4,5,6,7");
Console.WriteLine("\n Enter the array elements \n");
for (int i = 0; i < x; i++)
{
    string s1 = Console.ReadLine();
    a[i] = Int32.Parse(s1);
}
Console.WriteLine("8,9,10,11,12,1314");
Console.WriteLine("Enter the Search element\n");
string s3 = Console.ReadLine();
int x2 = Int32.Parse(s3);
for (int i = 0; i < x; i++)
{
if (a[i] == x2)
            {
                Console.WriteLine("15,16,17,18,19,20,21");
                Console.WriteLine("Search successful");
```

```
  Console.WriteLine("Element {0} found at location {1}\n", x2, i + 1);
                //return;
                Console.ReadLine();
            }
        }
        Console.WriteLine("Entered element not found. Search
unsuccessful");
        Console.ReadLine();
    }
  }
}
```

Output:

```
C:\Users\KOLLI TEJASWI\source\repos\Linear Search\bin\Debug\Linear Search
Enter the number of elements you want to add in the array ?
7
1,2,3,4,5,6,7

 Enter the array elements

8
9
10
11
12
13
14
8,9,10,11,12,13,14
Enter the Search element

12
15,16,17,18,19,20,21
Search successful
Element 12 found at location 5

22
Entered element not found. Search unsuccessful
```

Linear search is rarely used practically because other search algorithms such as the binary search algorithm and hash tables allow significantly faster-searching comparison to Linear search.

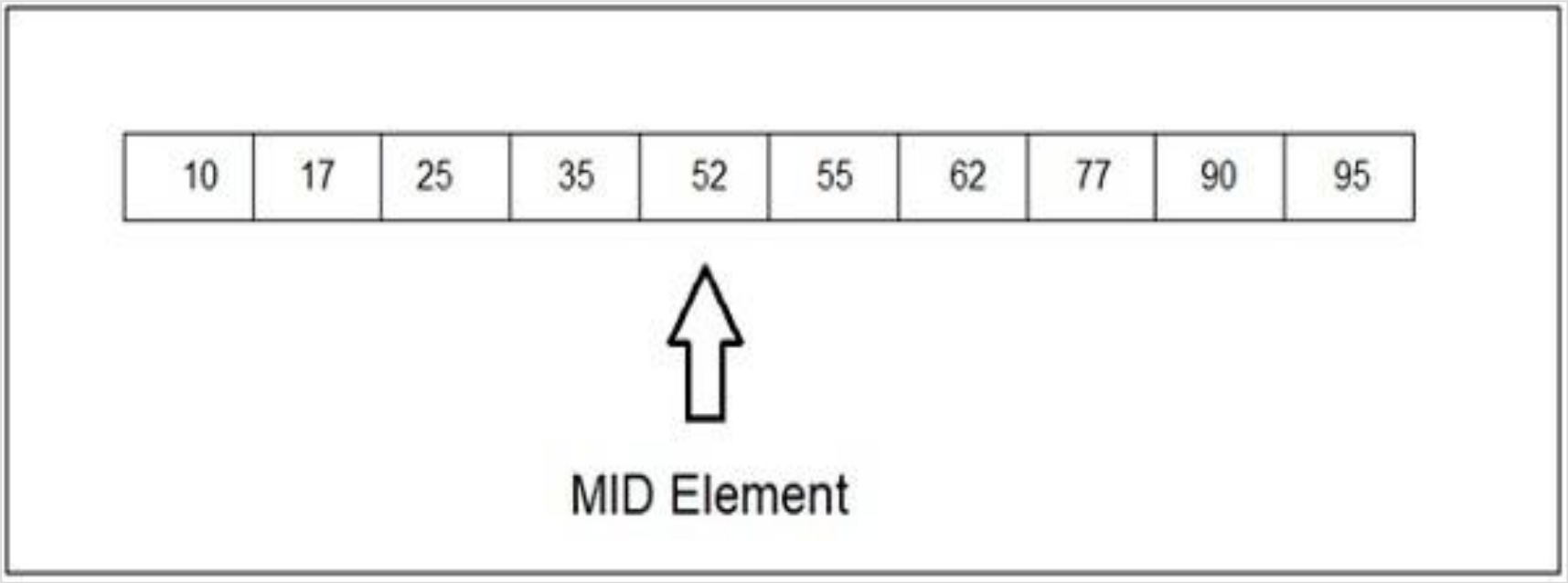


Using Array,
Explain Binary Search

Binary Search:

Binary search works on a sorted array. The value is compared with the middle element of the array. If equality is not found, then the half part is eliminated in which the value is not there. In the same way, the other half part is searched.

Here is the mid element in our array. Let’s say we need to find 62, then the left part would be eliminated and the right part is then searched −



Example:

```
using System;
namespace Binary_Search_2
{
    class Program
    {
        static void Main(string[] args)
        {
            // Create an array of 10 elements
            int[] IntArray = new int[10] { 9,10,4,2,7,2,19,23,15,6};
            // Value to search for

     int target = 7;
                int pos = Array.BinarySearch(IntArray, target);
                if (pos >= 0)
```

```
 Console.WriteLine($"Item {IntArray[pos].ToString()} found at position {pos
+ 1}.");
            else
                Console.WriteLine("Item not found");
            Console.ReadKey();
        }
    }
}
```

Output:

```
C:\Users\KOLLI TEJASWI\source\repos\Binary Search 2\
Item 7 found at position 5.
```

Thank You