15B17CI371 – Data Structures Lab

ODD 2024

Week 4-LAB A

Practice Lab

Virtual lab

Linear Search

Instructions

- Click on the next button to start the demo.
- Move the slider to adjust the speed of the demo.
- Click on the pause button if you want to stop and manually click the Next button to have a step by step realization of the process.
- Click on the reset button to start all over with a new set of random numbers!

31	35	11	52	12	48	28	87	24	89

Legend:

- Current Element
- Element Found
- Element Not Found
- Visited Element
- Unvisited Element

Observations

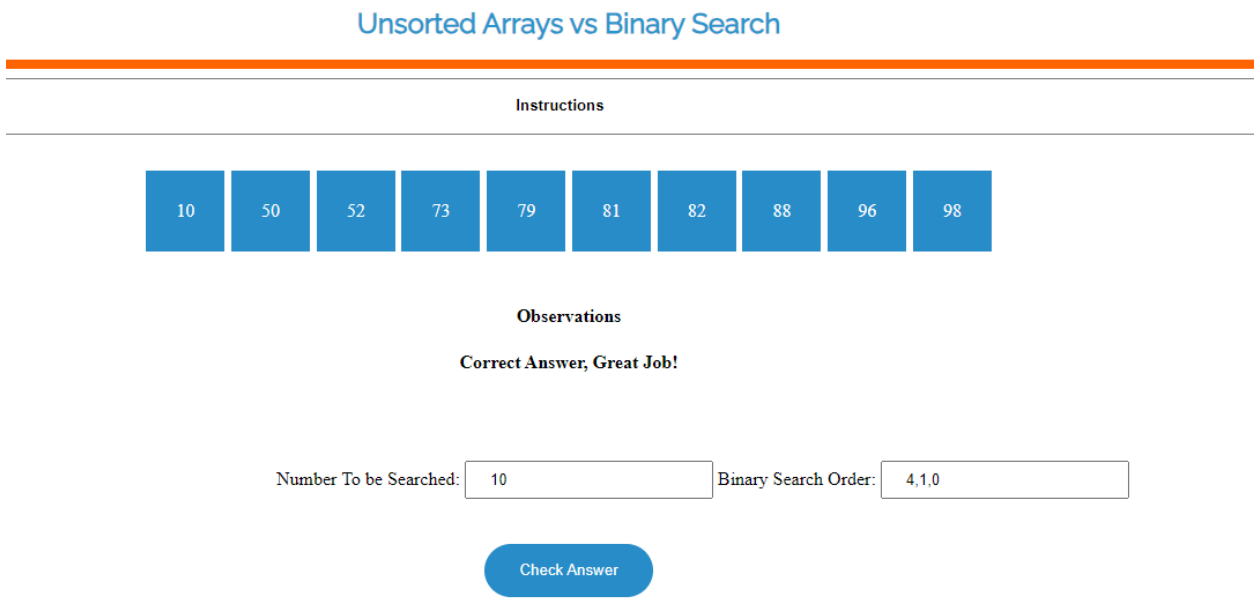
The Element 87 was found in the 7 position of the array.

Min. Speed Max. Speed

Number To be Searched: 87

Next Reset Pause

Binary Search



1.

```
#include<iostream>

using namespace std;

void lsearch( int arr[],int n,int k)

{

   int c=0;

for(int i=0;i<n;i++)

{

   if(arr[i]==k)

   {

      cout<<"Element found at index "<<i;

      cout<<endl;
```

```
        c++;
    }
}
if(c==0)


{
cout<<"Element not found";
}
}
int main()
{
    int n;
    cout<<"Enter the number of elements you want:";
    cin>>n;
    int arr[n];
    cout<<"Enter the elements:";
    for(int i=0;i<n;i++)
    {
       cin>>arr[i];
    }
    cout<<"The elements are:";
    for(int i=0;i<n;i++)
    {
       cout<<arr[i]<<" ";
```

```
    }
    cout<<endl;
    int k;
    cout<<"Enter the key element:";
    cin>>k;
    lsearch(arr,n,k);

}
```

```
Enter the number of elements you want:5
Enter the elements:1
2
2
3
4
The elements are:1 2 2 3 4
Enter the key element:2
Element found at index 1
Element found at index 2

Process returned 0 (0x0)   execution time : 80.867 s
Press any key to continue.
```

```
Enter the number of elements you want:5
Enter the elements:1
2
3
4
5
The elements are:1 2 3 4 5
Enter the key element:6
Element not found
Process returned 0 (0x0)   execution time : 5.343 s
Press any key to continue.
```

2.

```
#include<iostream>
using namespace std;
bool ppair( int arr[],int n,int n1)
{
for(int i=0;i<n;i++)
{

    for(int j=i+1;j<n;j++)
    {

        if(arr[i]*arr[j]==n1)
        {
            cout<<"Pair found:("<<arr[i]<<","<<arr[j]<<")";
            cout<<endl;
            return true;
        }

    }
}
return false;
}
```

```
int main()
{
   int n;
   cout<<"Enter the number of elements you want:";
   cin>>n;
   int arr[n];
   cout<<"Enter the elements:";
   for(int i=0;i<n;i++)
   {
      cin>>arr[i];
   }
   cout<<"The elements are:";
   for(int i=0;i<n;i++)
   {
      cout<<arr[i]<<" ";
   }
   cout<<endl;
   int n1;
   cout<<"Enter the element you want product for:";
   cin>>n1;
   int x=ppair(arr,n,n1);
if(x==0)
```

```
{
    cout<<"Pair not found";
}


}
```

```
Enter the number of elements you want:5
Enter the elements:1
2
3
4
5
The elements are:1 2 3 4 5
Enter the element you want product for:5
Pair found:(1,5)

Process returned 0 (0x0)   execution time : 5.812 s
Press any key to continue.
```

```
Enter the number of elements you want:5
Enter the elements:1
2
3
4
5
The elements are:1 2 3 4 5
Enter the element you want product for:7
Pair not found
Process returned 0 (0x0)   execution time : 4.894 s
Press any key to continue.
```

3.

```
#include <iostream>
using namespace std;
void bubbleSortAscending(int arr[], int n) {
    for (int i = 0; i < n - 1; ++i) {
        for (int j = 0; j < n - i - 1; ++j) {
            if (arr[j] > arr[j + 1]) {
                int temp = arr[j];
```

```
            arr[j] = arr[j + 1];
            arr[j + 1] = temp;
        }
    }
  }
}
void waveSort(int arr[], int n) {
  bubbleSortAscending(arr, n);

  for (int i = 0; i + 1 < n; i += 2) {
    int temp = arr[i];
    arr[i] = arr[i + 1];
    arr[i + 1] = temp;
  }
}
int main() {
  const int size = 7;
  int arr[size] = {10, 90, 49, 2, 1, 5, 23};

  waveSort(arr, size);

  cout << "Wave-like array: ";
  for (int i = 0; i < size; ++i) {
    cout << arr[i] << " ";
  }
  cout << endl;
  return 0;
}
```

```
Wave-like array: 2 1 10 5 49 23 90
```

4.

```
#include <iostream>
#include <algorithm>
using namespace std;
int binarySearch(const int arr[], int size, int key) {
  int left = 0;
  int right = size - 1;

  while (left <= right) {
```

```
        int mid = left + (right - left) / 2;

        if (arr[mid] == key) {
            return mid; // Key found
        }
        if (arr[mid] < key) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }
    return -1;
}

void findAllOccurrences(int arr[], int size, int key) {
    sort(arr, arr + size);
    int index = binarySearch(arr, size, key);

    if (index == -1) {
        cout << "Element not found in the array" << endl;
        return;
    }

       int leftIndex = index;
    while (leftIndex >= 0 && arr[leftIndex] == key) {
        cout << "Element found at index " << leftIndex << endl;
        leftIndex--;
    }

      int rightIndex = index + 1;
    while (rightIndex < size && arr[rightIndex] == key) {
        cout << "Element found at index " << rightIndex << endl;
        rightIndex++;
    }
}
int main() {
    int arr[] = {16, 31, 15, 27, 9, 15, 39, 15, 17, 12};
    int size = sizeof(arr) / sizeof(arr[0]);
    int key = 15;

    findAllOccurrences(arr, size, key);

    return 0;
}
```

```
enter the no of elements in the array : 10
16
31
15
27
9
15
39
15
17
12
Enter key : 15
element found at index  :  2
element found at index  :  5
element found at index  :  7

Process returned 0 (0x0)   execution time : 39.774 s
Press any key to continue.
```

```
#include <iostream>

using namespace std;


void insertionSort(int arr[], int size) {

   for (int i = 1; i < size; ++i) {

      int key = arr[i];

      int j = i - 1;

      while (j >= 0 && arr[j] > key) {

         arr[j + 1] = arr[j];

         --j;

      }

      arr[j + 1] = key;

   }
```

```
}

bool binarySearch(const int arr[], int size, int key) {
    int left = 0;
    int right = size - 1;

    while (left <= right) {
        int mid = left + (right - left) / 2;

        if (arr[mid] == key) {
            return true;
        }
        if (arr[mid] < key) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }
    return false;
}

void findPairWithProduct(int arr[], int size, int n) {
    insertionSort(arr, size);

    for (int i = 0; i < size; ++i) {
```

```
        if (arr[i] == 0) {
            continue;
        }

        if (n % arr[i] == 0) {
            int complement = n / arr[i];

            if (binarySearch(arr, size, complement)) {
                cout << "Pair Found: (" << arr[i] << ", " << complement << ")" << endl;
                return;
            }
        }
    }

    cout << "No pair found" << endl;
}

int main() {
    int arr[] = {5, 20, 3, 2, 50, 80};
    int size = sizeof(arr) / sizeof(arr[0]);
    int n = 150;

    findPairWithProduct(arr, size, n);

    return 0;
```

```
}
```

```
Pair Found: (3, 50)
```

```
#include <iostream>

using namespace std;


void insertionSort(int arr[], int size) {

    for (int i = 1; i < size; ++i) {

        int key = arr[i];

        int j = i - 1;

        while (j >= 0 && arr[j] > key) {

            arr[j + 1] = arr[j];

            --j;

        }

        arr[j + 1] = key;

    }

}


bool interpolationSearch(const int arr[], int size, int key) {

    int left = 0;

    int right = size - 1;


    while (left <= right && key >= arr[left] && key <= arr[right]) {

        if (left == right) {
```

```
            if (arr[left] == key) return true;
            return false;
        }

        int pos = left + ((key - arr[left]) * (right - left)) / (arr[right] - arr[left]);

        if (arr[pos] == key) {
            return true;
        }
        if (arr[pos] < key) {
            left = pos + 1;
        } else {
            right = pos - 1;
        }
    }

    return false;
}

void findPairWithProduct(int arr[], int size, int n) {
    insertionSort(arr, size);

    for (int i = 0; i < size; ++i) {
        if (arr[i] == 0) {
            continue;
```

```
        }

        if (n % arr[i] == 0) {
            int complement = n / arr[i];

            if (interpolationSearch(arr, size, complement)) {
                cout << "Pair Found: (" << arr[i] << ", " << complement << ")" << endl;
                return;
            }
        }
    }

    cout << "No pair found" << endl;
}

int main() {
    int arr[] = {5, 20, 3, 2, 50, 80};
    int size = sizeof(arr) / sizeof(arr[0]);
    int n = 150;
    findPairWithProduct(arr, size, n);
    return 0;
}
```

```
Pair Found: (3, 50)
```