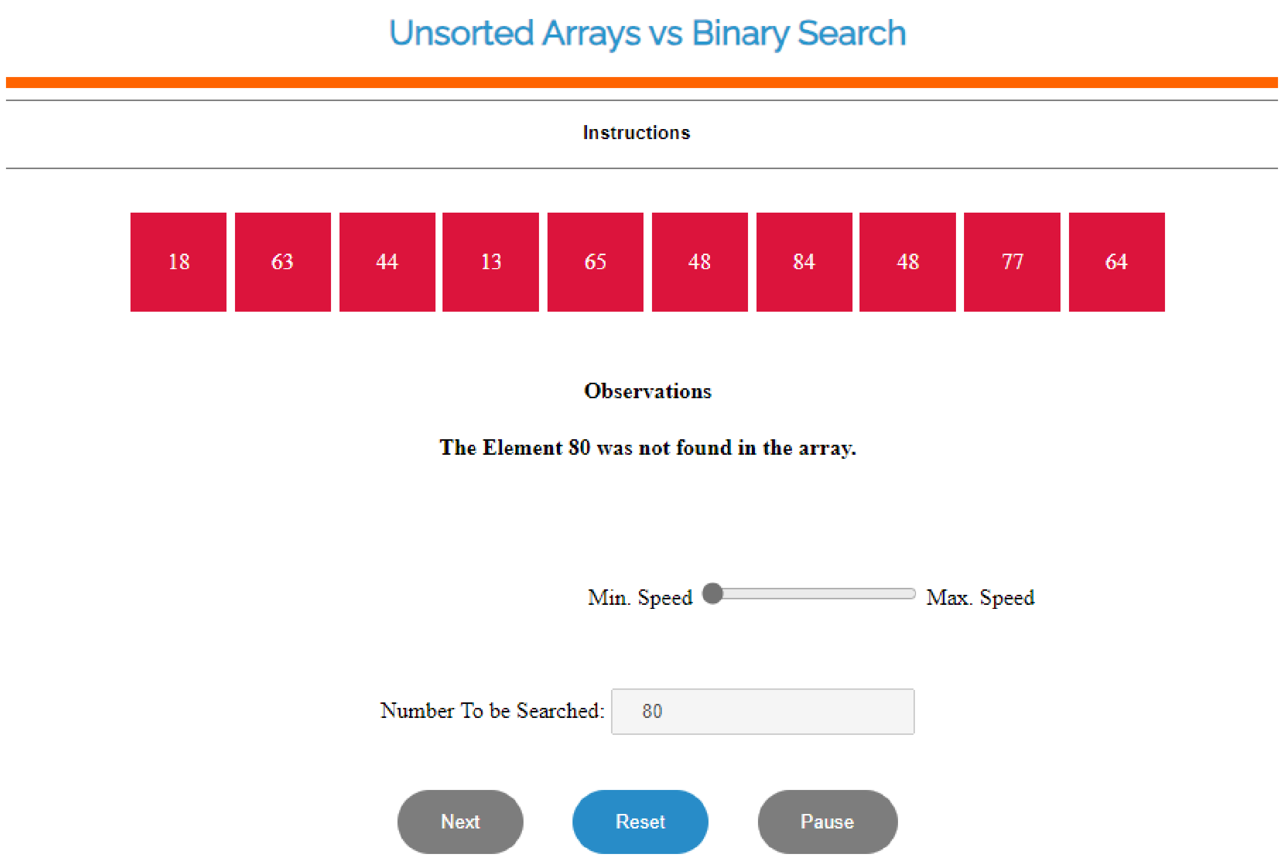
Linear Search



Unsorted Arrays vs Binary Search

Instructions

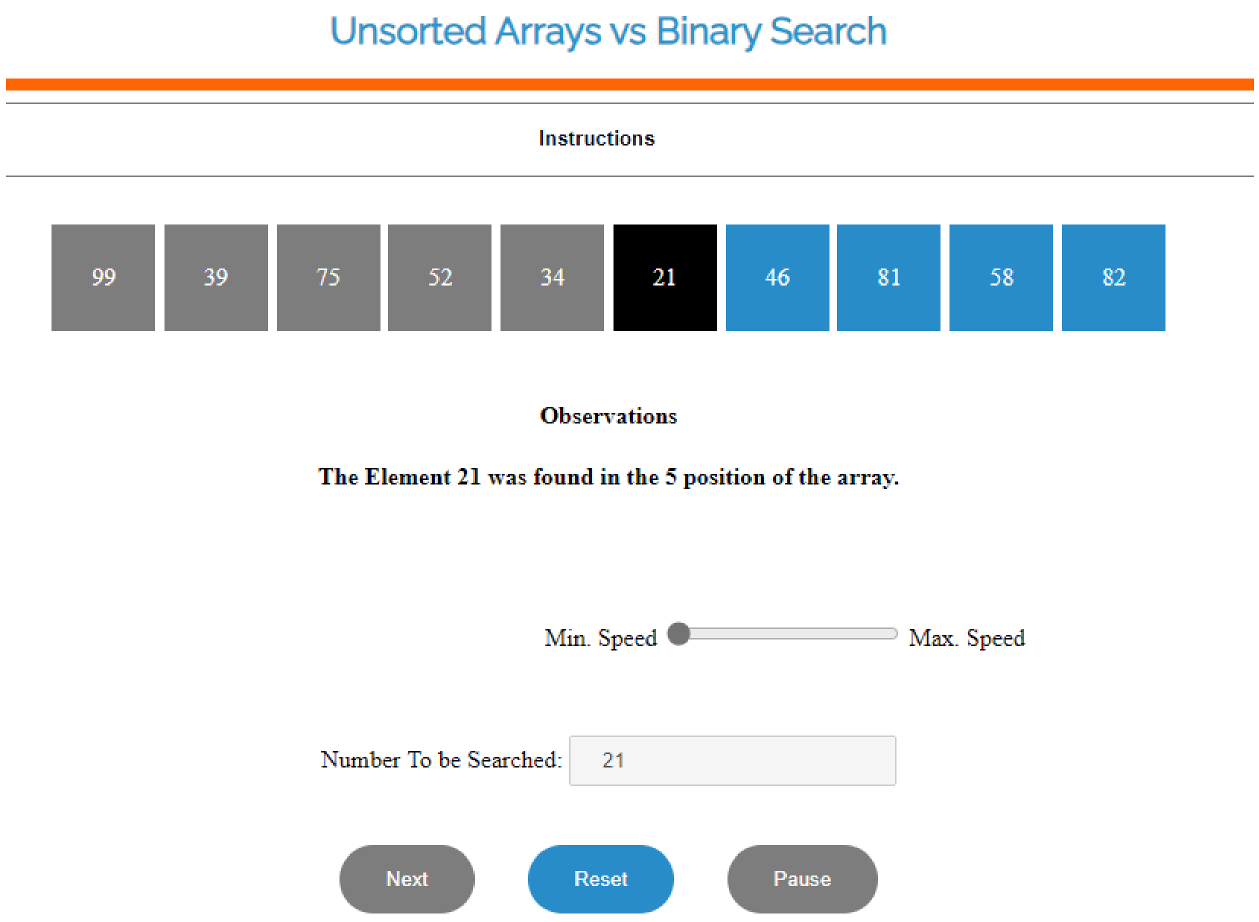
18	63	44	13	65	48	84	48	77	64

Observations

The Element 80 was not found in the array.

Min. Speed Max. Speed

Number To be Searched: 80



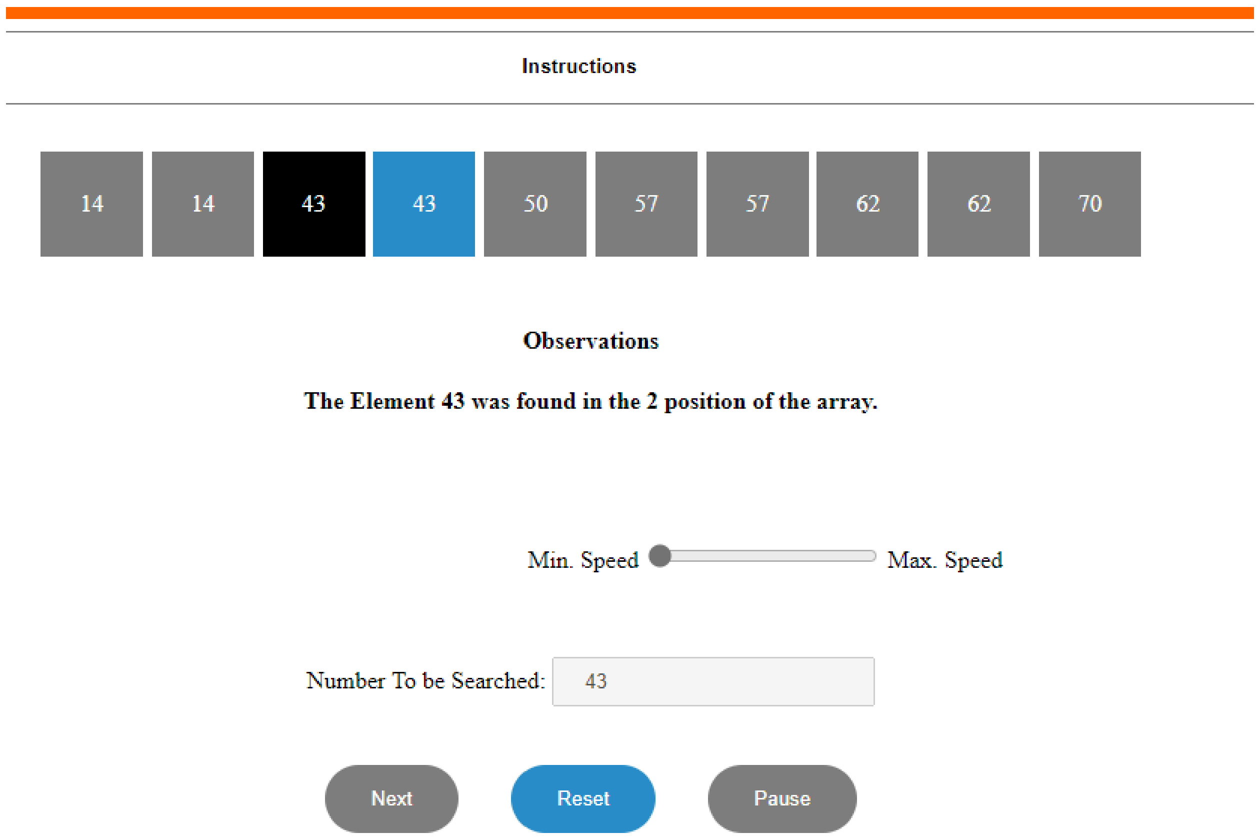
Unsorted Arrays vs Binary Search

Observations

The Element 21 was found in the 5 position of the array.

Number To be Searched: 21

Binary Search



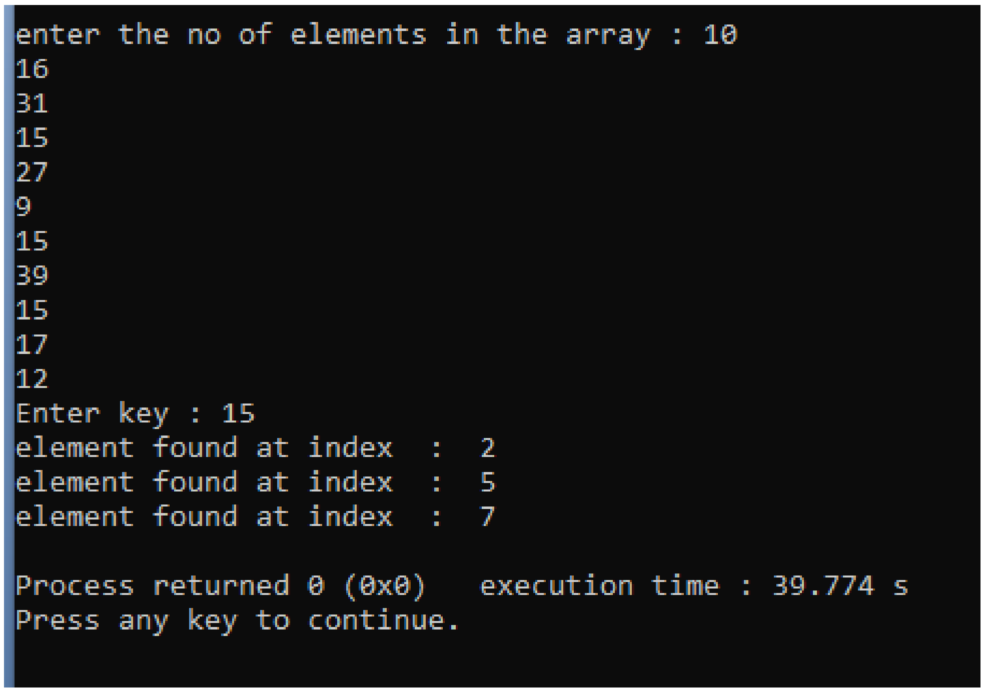
Observations

The Element 43 was found in the 2 position of the array.

Number To be Searched: 43

1.

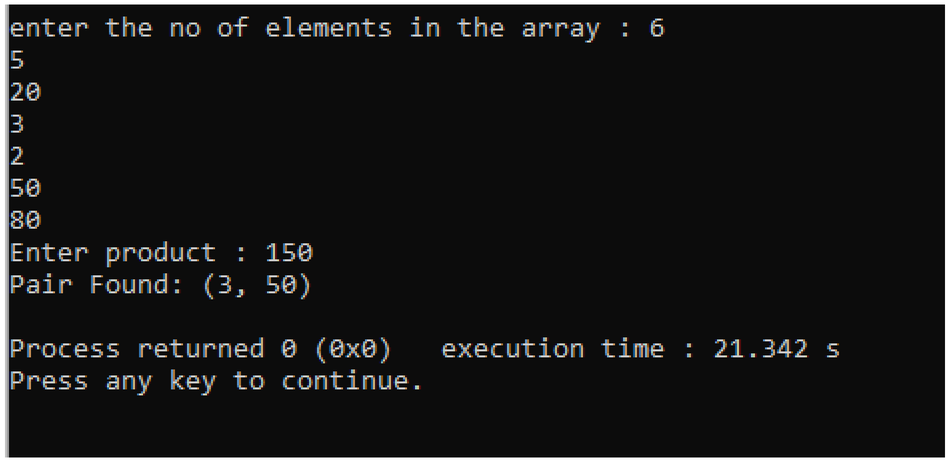
```
#include <iostream>

using namespace std;

int main()

{


    int n;

    cout<<endl<<"enter the no of elements in the array : ";

    cin>>n;

    int arr[n];

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

    {

        cin>>arr[i];

    }

    int k;

    cout<<"Enter key : ";

    cin>>k;

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

    {

        if(arr[i]==k)

        {

            cout<<"element found at index  :  "<<i<<endl;

        }
    }

}
```



```
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.
```

2.

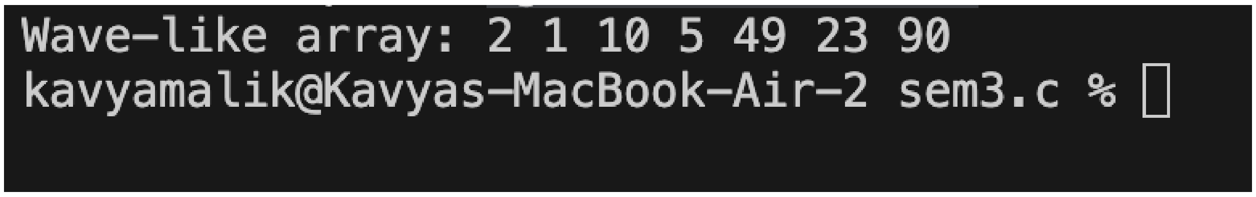
```
#include <iostream>

using namespace std;


void product( int arr[], int size, int n) {

    int f=1;


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

        for (int j = i + 1; j < size; j++) {
            if (arr[i] != 0 && arr[j] != 0 && arr[i] * arr[j] == n) {

                cout << "Pair Found: (" << arr[i] << ", " << arr[j] << ")" << endl;

                f=0;


            }

        }

    }

    if (f==1)

    {

         cout << "No Pair Found" << endl;


    }


}

int main()

{


    int n;

    cout<<endl<<"enter the no of elements in the array : ";

    cin>>n;

    int arr[n];

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

    {

        cin>>arr[i];
    }

    int p;

    cout<<"Enter product : ";

    cin>>p;

    product(arr, n, p);

}
```



```
enter the no of elements in the array : 6
5
20
3
2
50
80
Enter product : 150
Pair Found: (3, 50)

Process returned 0 (0x0)   execution time : 21.342 s
Press any key to continue.
```

3.

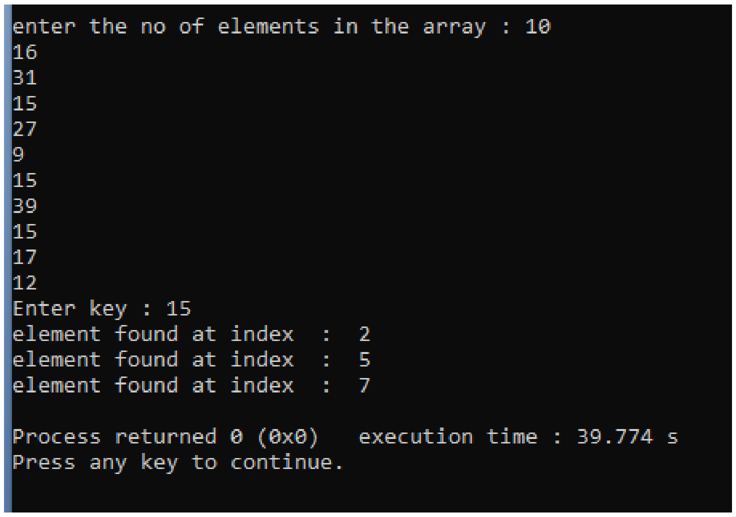
```
#include <iostream>
using namespace std;
void bubbleSortAscending(int arr[], int n) {
    for (int i = 0; i < n - 1; ++i) {
        for (int j = 0; j < n - i - 1; ++j) {
            if (arr[j] > arr[j + 1]) {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                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
kavyamalik@Kavyas-MacBook-Air-2 sem3.c %
```

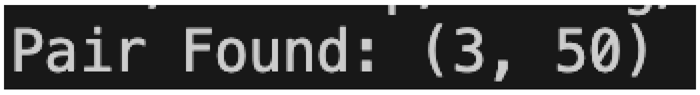
4.

```
#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;
}
```



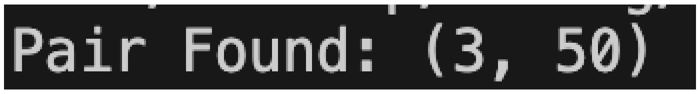
```
enter the no of elements in the array : 10
16
31
15
27
9
15
39
15
17
12
Enter key : 15
element found at index  :  2
element found at index  :  5
element found at index  :  7

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)
```