

# Data Structure and Algorithm Practicum

## Searching



**Name**

Dicha Zelianivan Arkana

**NIM**

2241720002

**Class**

1i

**Department**

Information Technology

**Study Program**

D4 Informatics Engineering

---

# 1 Sequential Search Method

## 1.1 Questions

1. What is the difference of method `displayData()` and `displayPosition()` in `StudentSearch` class?

The former shows the entire data of the student while the latter only shows the position of the student.

2. What is the function of `break` in this following program code?

```
if (listStudents[i].nim == search) {  
    position = i;  
    break;  
}
```

It's used to break the loop when we found the matching student's nim.

3. If inserted NIM data is not sorted from smallest to biggest value, will the program encounter an error? Is the result still correct? Why is that?

Because linear search doesn't depend on the sorting, it won't throw an error. It works by checking each item from start to finish.

# 2 Binary Search Method

## 2.1 Questions

1. Show the program code in which runs the divide process

```
if (search == listStudents[mid].nim) {  
    return mid;  
} else if (listStudents[mid].nim > search) {  
    return findBinarySearch(search, left, mid - 1);  
} else {  
    return findBinarySearch(search, mid + 1, right);  
}
```

It splits the list into 2 parts where the start and end changes based on the value of the NIM.

- 
2. Show the program code in which runs the conquer process

```
int mid;
if (right >= left) {
    mid = (left + right) / 2;
    if (search == listStudents[mid].nim) {
        return mid;
    } else if (listStudents[mid].nim > search) {
        return findBinarySearch(search, left, mid - 1);
    } else {
        return findBinarySearch(search, mid + 1, right);
    }
}
return -1;
```

It finds the middle part, compares it, and then conquer one of its side if it does not match. If it does match then return the value.

3. If inserted NIM data is not sorted, will the program crash? Why?

It will not crash, but it will display incorrect data since binary search depends on the sorting of the list.

If inserted NIM data is sorted from largest to smallest value (e.g 20215, 20214, 20212, 20211, 20210) and element being searched is 20210. How is the result of binary search? Does it return the correct one? If not, then change the code so that the binary search executed properly

It says that it didn't found the data.

4. Modify program above so that the students amount inserted is matched with user input

We can make the `StudentSearch` class accepts a parameter through its constructor

```
System.out.println("Insert the amount of the data");
int amountStudent = s.nextInt();
SearchStudent data = new SearchStudent(amountStudent);
```

and then on the class constructor, we can change it like so

```
public Student[] listStudents;
private int index = 0;

public SearchStudent(int amount) {
    listStudents = new Student[amount];
}
```

---

## 3 Review Divide and Conquer

## 4 Assignments

1. Modify the searching program above with these requirements:
  - Before we search using binary search, we have to sort the data first. You can use whichever sorting algorithm that you are comfortable with.

We can sort the data first before trying to search using binary search as such:

```
public int findBinarySearch(int search, int left, int right) {
    sortData();
    int mid;
    if (right >= left) {
        mid = (left + right) / 2;
        if (search == listStudents[mid].nim) {
            // if this is the last occurrence of search, return it
            if (mid == listStudents.length - 1 || listStudents[mid + 1].nim > search)
                return mid;
        } else {
            // continue searching in the right sub-array
            return findBinarySearch(search, mid + 1, right);
        }
    } else if (listStudents[mid].nim > search) {
        return findBinarySearch(search, left, mid - 1);
    } else {
        return findBinarySearch(search, mid + 1, right);
    }
}

return -1;
}

private void sortData() {
    for (int i = 0; i < listStudents.length - 1; i++) {
        for (int j = 0; j < listStudents.length - i - 1; j++) {
            if (listStudents[j].nim > listStudents[j + 1].nim) {
                // swap
                Student temp = listStudents[j];
                listStudents[j] = listStudents[j + 1];
                listStudents[j + 1] = temp;
            }
        }
    }
}
```

---

2. Modify the searching above with these requirements:

- Search by student's name with Sequential Search Algorithm

```
public int findSeqSearch(String search) {  
    int position = -1;  
    for (int i = 0; i < listStudents.length; i++) {  
        if (listStudents[i].name == search) {  
            position = i;  
            break;  
        }  
    }  
    return position;  
}
```

- How is the output of the program if there is any duplicate name?  
It will give the first found name and ignore the last one because sequential search works from start to end, starting at index 0
- There is 2d array as follows

Index	0	1	2	3	4
0	45	78	7	200	80
1	90	1	17	100	50
2	21	2	40	18	65

Based on data above, create a program to search data in 2d array, which the data to be searched is defined by user input (using sequential search). We can iterate through the 2d array by the row and columns. It will turn our loop from doing  $O(n)$  into  $O(n^2)$  shown on the code below

```
public int[] findInMatrix(int search, int[][] matrix) {  
    for (int i = 0; i < matrix.length; i++) {  
        for (int j = 0; j < matrix[i].length; j++) {  
            if (matrix[i][j] == search) return new int[]{i, j};  
        }  
    }  
    return null;  
}
```

We return null if it's not found and return a tuple of the [row, column] position if it's found

---

3. There is 1D array as follows:

5. 0	1	2	3	4	5	6	7	8	9
12	17	2	1	70	50	90	17	2	90

Create a program to sort the array, search & display the biggest value, and print the amount of biggest value available alongside with its position.

```
public class SingleDimensionArray {
    private static void displayData(int[] data) {
        for (int i = 0; i < data.length; i++) {
            System.out.printf("%s ", data[i]);
        }
        System.out.println();
    }

    private static void displayLargestValue(int[] data) {
        int[] sortedData = sortAscending(data);
        int[] largestValue = findLargestValue(sortedData);
        System.out.println("Largest value position\t: " + largestValue[0]);
        System.out.println("Largest value\t\t\t: " + largestValue[1]);
    }

    // returns a tuple of index and value
    private static int[] findLargestValue(int[] data) {
        int largestPosition = 0;
        int largest = data[largestPosition];
        for (int i = 1; i < data.length; i++) {
            if (data[i] > largest) {
                largestPosition = i;
                largest = data[largestPosition];
            }
        }
        return new int[]{largestPosition, largest};
    }

    private static int[] sortAscending(int[] data) {
        // uses insertion sort
        for (int i = 1; i < data.length; i++) {
            int tmp = data[i];
            int j = i - 1;
            while (j >= 0 && data[j] > tmp) {
                data[j + 1] = data[j];
                j--;
            }
            data[j + 1] = tmp;
        }
        return data;
    }
}
```

---

```

private static int[] sortDescending(int[] data) {
    // uses insertion sort
    for (int i = 1; i < data.length; i++) {
        int tmp = data[i];
        int j = i - 1;
        while (j >= 0 && data[j] < tmp) {
            data[j + 1] = data[j];
            j--;
        }
        data[j + 1] = tmp;
    }
    return data;
}

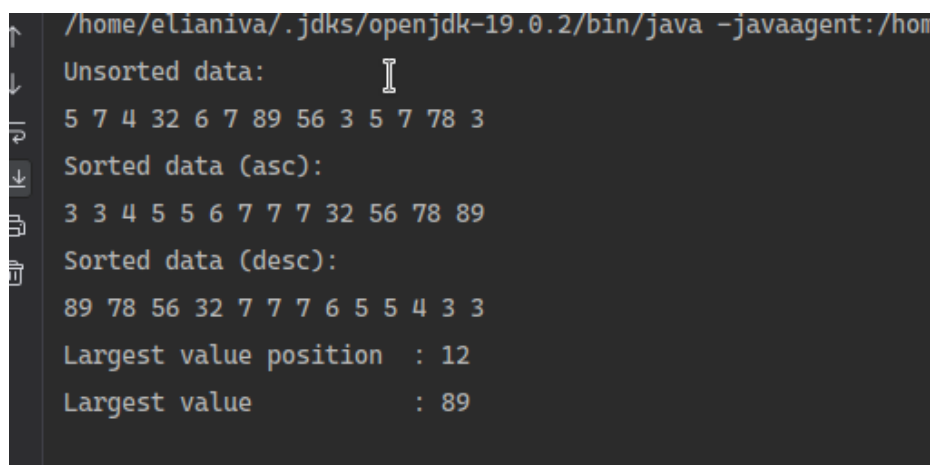
public static void main(String[] args) {
    int[] data = {5, 7, 4, 32, 6, 7, 89, 56, 3, 5, 7, 78, 3};
    System.out.println("Unsorted data: ");
    displayData(data);

    System.out.println("Sorted data (asc):");
    int[] sortedDataAscending = sortAscending(data);
    displayData(sortedDataAscending);

    System.out.println("Sorted data (desc):");
    int[] sortedDataDescending = sortDescending(data);
    displayData(sortedDataDescending);

    displayLargestValue(data);
}
}

```



```

/home/elianiva/.jdk/openjdk-19.0.2/bin/java -javaagent:/hom
Unsorted data:
5 7 4 32 6 7 89 56 3 5 7 78 3
Sorted data (asc):
3 3 4 5 5 6 7 7 7 32 56 78 89
Sorted data (desc):
89 78 56 32 7 7 7 6 5 5 4 3 3
Largest value position : 12
Largest value          : 89

```