GIT Department of Computer Engineering
CSE 222/505 - Spring 2022
Homework 4 Report

Emircan Demirel
1901042674

1. SYSTEM REQUIREMENTS

Functional Requirements:

The user can make recursive string search to get index of i^{th} occurrence of the substring.

The user could enter an interval to and search numbers in this range in the given sorted integer array.

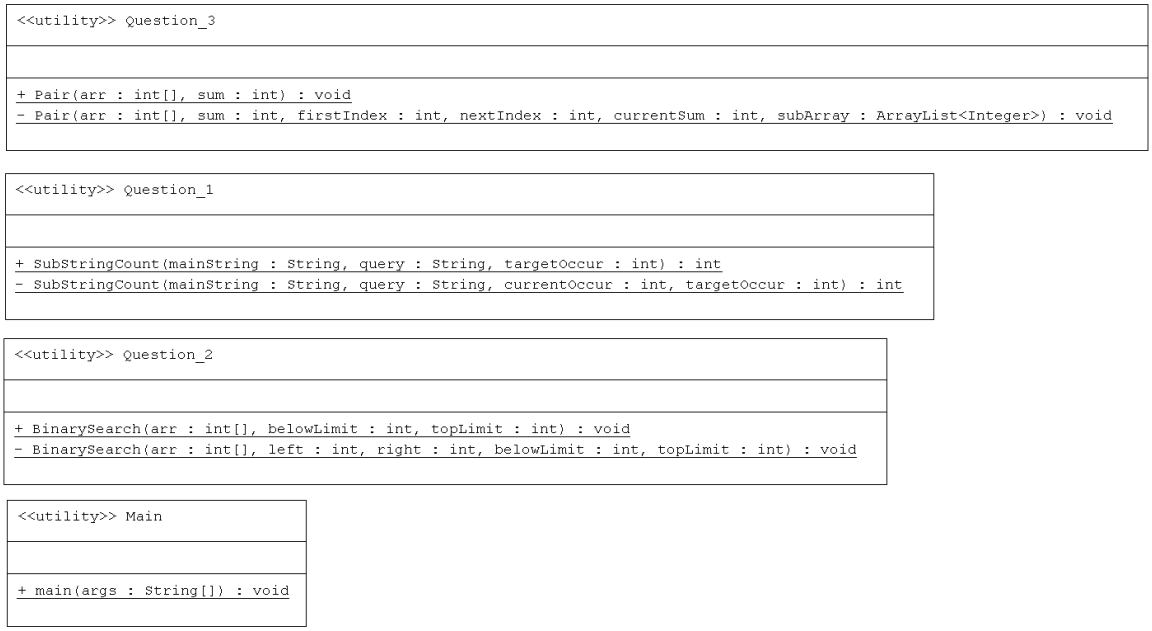
The user could search contiguous subarray/s that the sum of its/theirs items is equal to a given integer value in the given unsorted integer array.

Non-Functional Requirements:

Java-SE13 or higher should be installed and running properly.

Program must handle errors.

2. CLASS DIAGRAMS



3. PROBLEM SOLUTION APPROACH

Q1 -) According to the problem, the recursive method should return index of target occurrence in the main string. At the end of the search operation, If target occurrence value bigger than current occurrences of query string, method should return -1. To solve this by using recursion, I used substring() and equals() method of String class.

If the main string's length is 0 or the query string is larger than the main string, base case will work, and method will return -1. On the other hand, program will search query string by splitting main string pieces. In every search index parameter increases. If searched

substring is equal to query string current Occur parameter increases and if current Occur parameter is equal to targetOccur method will return index of searched substring.

Q2 -) Problem wants us to write a recursive algorithm to find the number of items in the given sorted array between two given integer values. I decided to write a recursive binary search algorithm, taking advantage of the fact that it is a sorted array. If calculated mid index value of the array is in the interval of the given integer values, the method will print mid index value to the console, in each step.

Q3 -) According to the problem I have to write a recursive algorithm to find contiguous subarray/s that the sum of its/theirs items is equal to a given integer value. To traverse in the array easily I create two parameters firstIndex and nextIndex. To find contiguous elements in wanted conditions, program increases nextIndex value and compare the currentSum value to the sum value and add element in nextIndex to the subArray, in each step. When nextIndex value reach the end of the array or currentSum is larger than sum value, program increases firstIndex value then assigned this value to nextIndex and create new subArray. When the desired conditions are met, the program will print the subArray to the console.

4. TEST CASES

Q1 -)

1. Compile -> Valid String & Valid Query & Target Occurrence is smaller than number of total occurrences
2. Compile -> Valid String & Valid Query & Target Occurrence is equal to the number of total occurrences
3. Compile -> Valid String & Valid Query & Target Occurrence is larger than number of total occurrences
4. Compile -> Valid String & Invalid Query & Valid Target Occurrence
5. Compile -> Invalid String & Valid Query & Valid Target Occurrence

Q2 -)

1. Compile -> Valid Array (contain numbers in given range) & Smaller belowLimit & Larger topLimit
2. Compile -> Valid Array (does not contain numbers in given range) & Smaller belowLimit & Larger topLimit
3. Compile -> Valid Array & Larger belowLimit & Smaller topLimit
4. Compile -> Valid Array & equal belowLimit and topLimit

Q3 -)

1. Compile -> Array contains contiguous subarray & positive sum value
2. Compile -> Array does not contains contiguous subarray & positive sum value
3. Compile -> Array contains contiguous subarray & negative sum value
4. Compile -> Array does not contains contiguous subarray & negative sum value

5. RUNNING AND RESULTS

Q1.1)

```
-------------
Q1 - )
mainString: emircanmirmir
query: mir
target occurrence: 1
index of target occurrence: 1
```

Q1.2)

```
-------------
Q1 - )
mainString: emircanmirmir
query: mir
target occurrence: 3
index of target occurrence: 10
```

Q1.3)

```
-------------
Q1 - )
mainString: emircanmirmir
query: mir
target occurrence: 7
index of target occurrence: -1
```

Q1.4)

```
-------------
Q1 - )
mainString: emircanmirmir
query:
target occurrence: 7
index of target occurrence: -1
```

Q1.5)

```
-------------
Q1 - )
mainString:
query: mir
target occurrence: 7
index of target occurrence: -1
```

Q2.1)

```
-------------
Q2 - )
sorted array: -4 0 5 6 7 8 9 21 44
below limit: 6
top limit: 9
elements in the given range: 7 6 9 8
```

Q2.2)

```
-------------
Q2 - )
sorted array: -4 0 5 6 7 8 9 21 44
below limit: 12
top limit: 16
elements in the given range:

Process finished with exit code 0
```

Q2.3)

```
-------------
Q2 - )
sorted array: -4 0 5 6 7 8 9 21 44
below limit: 9
top limit: 6
elements in the given range:

Process finished with exit code 0
```

Q2.4)

```
-------------
Q2 - )
sorted array: -4 0 5 6 7 8 9 21 44
below limit: 9
top limit: 9
elements in the given range:

Process finished with exit code 0
```

Q3.1)

```
-------------
Q3 - )
unsorted array: 9 -8 5 0 3 4 2 7
sum: 9
subarrays:
[9]
[0, 3, 4, 2]
[3, 4, 2]
[2, 7]

Process finished with exit code 0
```

Q3.2)

```
-------------
Q3 - )
unsorted array: 1 4 2 5 -4 11
sum: 9
subarrays:

Process finished with exit code 0
```

Q3.3)

```
-------------
Q3 - )
unsorted array: 1 4 2 5 -4 -7 11
sum: -11
subarrays:
[-4, -7]

Process finished with exit code 0
```

Q3.4)

```
-------------
Q3 - )
unsorted array: 1 4 2 5 -4 -7 11
sum: -1
subarrays:

Process finished with exit code 0
```