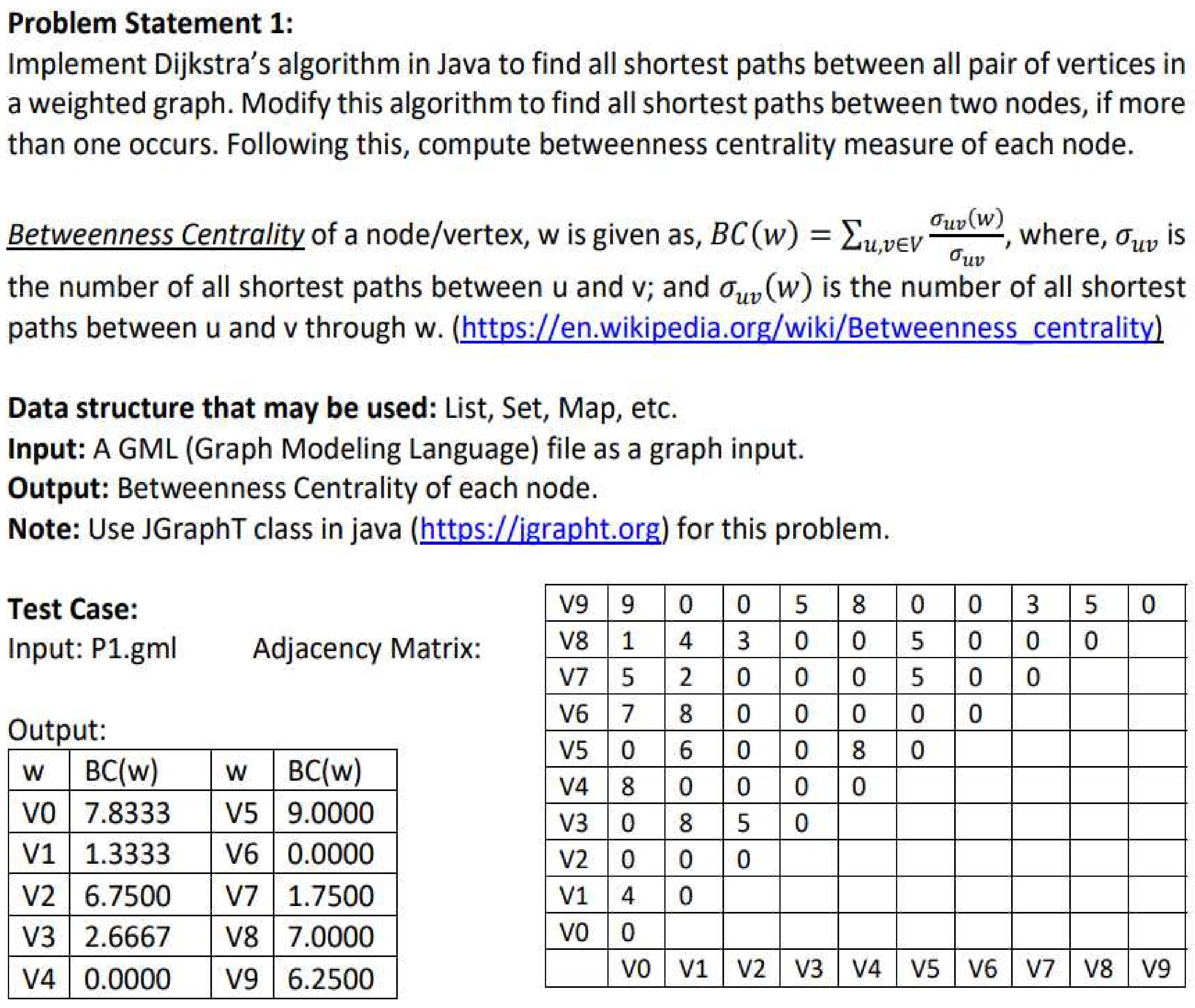
CSN-261
ASSIGNMENT-8

- Name-Kavya Barnwal
- Enrollment No.-18114039
- Course-B.Tech-CSE
- Email: kbarnwal@cs.iitr.ac.in
- Contact:6205125517

==

Problem Statement 1:



Problem Statement 1:
Implement Dijkstra's algorithm in Java to find all shortest paths between all pair of vertices in a weighted graph. Modify this algorithm to find all shortest paths between two nodes, if more than one occurs. Following this, compute betweenness centrality measure of each node.

Betweenness Centrality of a node/vertex, w is given as, $BC(w) = \sum_{u,v \in V} \frac{\sigma_{uv}(w)}{\sigma_{uv}}$, where, σ_{uv} is the number of all shortest paths between u and v; and $\sigma_{uv}(w)$ is the number of all shortest paths between u and v through w. (https://en.wikipedia.org/wiki/Betweenness_centrality)

Data structure that may be used: List, Set, Map, etc.
Input: A GML (Graph Modeling Language) file as a graph input.
Output: Betweenness Centrality of each node.
Note: Use JGraphT class in java (https://jgrapht.org) for this problem.

Test Case:
Input: P1.gml

Adjacency Matrix:

V9	9	0	0	5	8	0	0	3	5	0
V8	1	4	3	0	0	5	0	0	0	
V7	5	2	0	0	0	5	0	0		
V6	7	8	0	0	0	0	0			
V5	0	6	0	0	8	0				
V4	8	0	0	0	0					
V3	0	8	5	0						
V2	0	0	0							
V1	4	0								
V0	0									
	V0	V1	V2	V3	V4	V5	V6	V7	V8	V9

Output:

w	BC(w)	w	BC(w)
V0	7.8333	V5	9.0000
V1	1.3333	V6	0.0000
V2	6.7500	V7	1.7500
V3	2.6667	V8	7.0000
V4	0.0000	V9	6.2500

=======================================

Data Structure And Algorithm Used :

Data Structures Used :

Adjacency List(Linked list)

Linked List for Nodes and edges.

Algorithms Used :

1) Create a set *sptSet* (shortest path tree set) that keeps track of vertices included in shortest path tree, i.e., whose minimum distance from source is calculated and finalized. Initially, this set is empty.

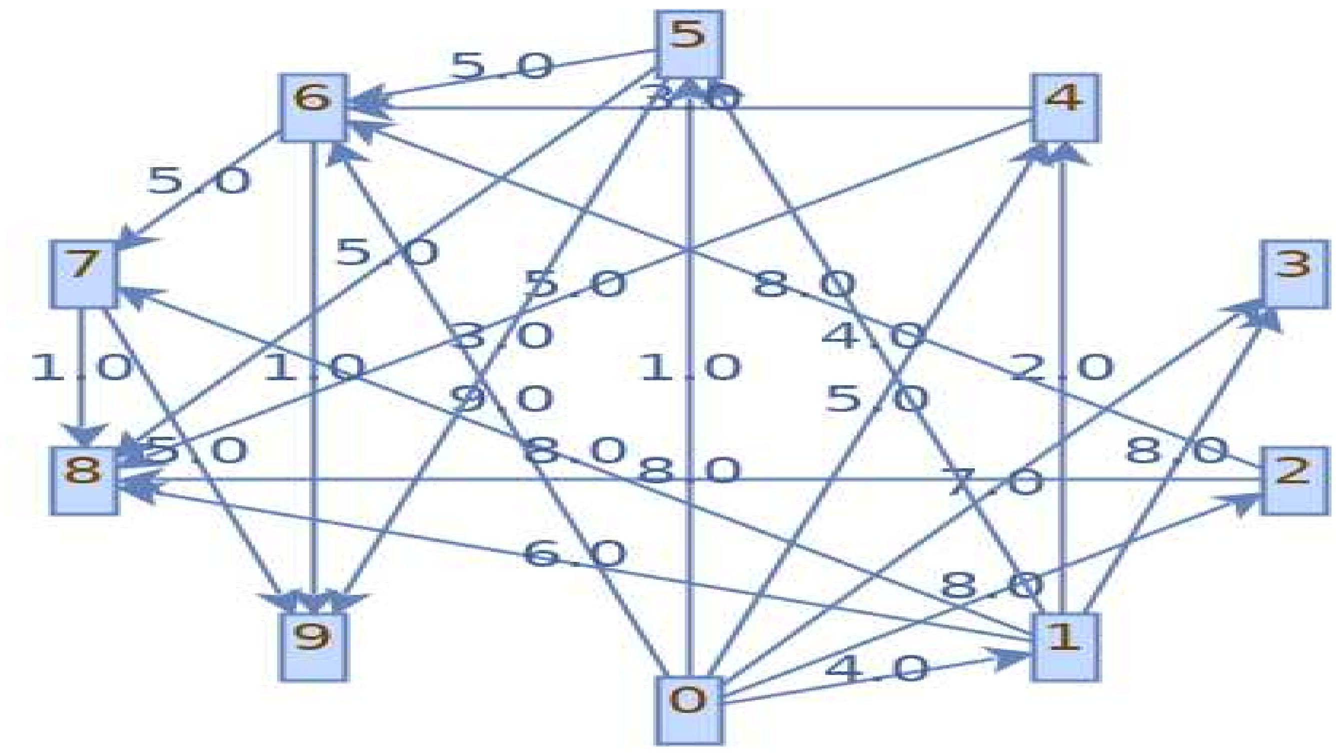
2) Assign a distance value to all vertices in the input graph. Initialize all distance values as INFINITE. Assign distance value as 0 for the source vertex so that it is picked first.

3) While *sptSet* doesn't include all vertices

….**a)** Pick a vertex u which is not there in *sptSet* and has minimum distance value.

….**b)** Include u to *sptSet*.

….**c)** Update distance value of all adjacent vertices of u. To update the distance values, iterate through all adjacent vertices. For every adjacent vertex v, if sum of distance value of u (from source) and weight of edge u-v, is less than the distancevalue of v, then update the distance value of v.



Drive Link for the complete olders of project files is here:

Link

======================================

Problem Statement 2:

Create a project/program in Java called Unscramble Word. Given a string of ‘N’ characters

print all the words present in a dictionary of length ‘M’ such that 3 < M <= N.

Use dictionary present in Linux @ /usr/share/dict/words.

Implement this code in java and the student may use inbuilt data structures such as Maps,Sets, etc. (For fast execution, use of Trie is suggested).

Input: A String

Output: All unscrambled words of given string present in the dictionary categorized by length of word. Also print the total number of words of each length.

Test Case:

Input: “great”

Output:

Length: 5 greta, grate, great, retag, targe Count: 5

Length: 4 ager, gate, gear, geta, grat, rage, rate, tare, tear Count: 9

==

Data Structure And Algorithm Used :

Data Structures Used :

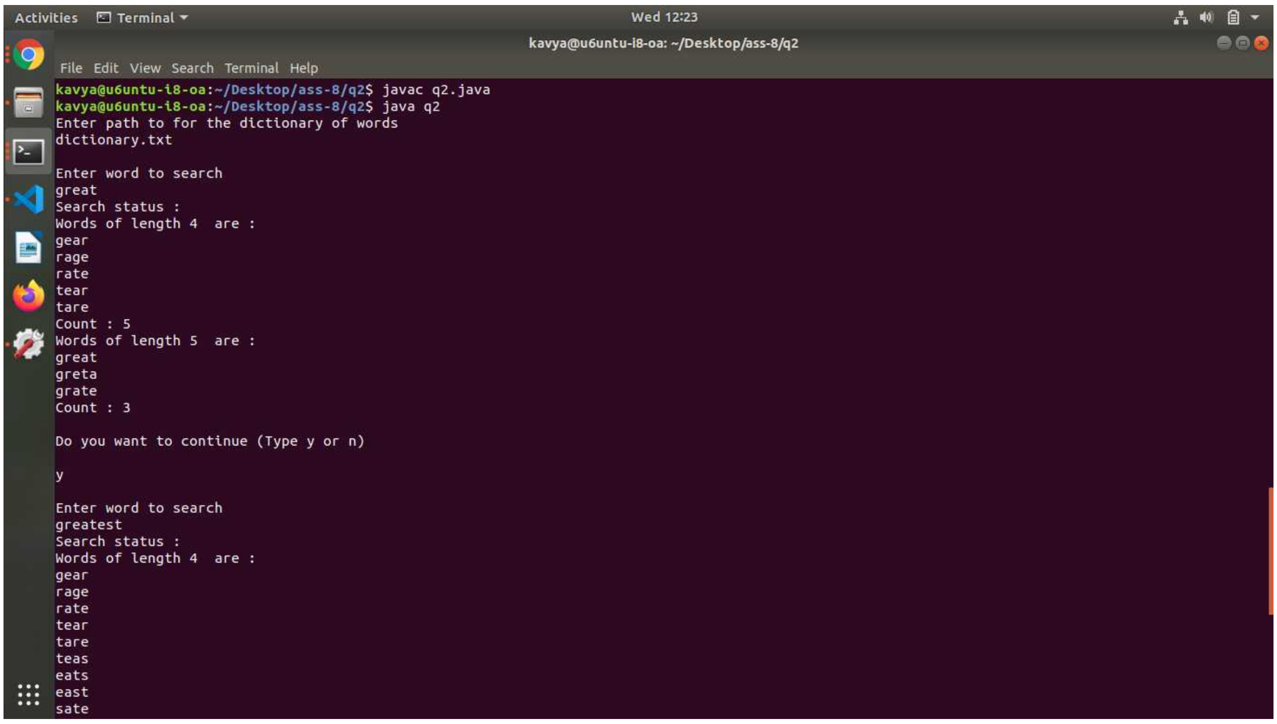
- Trie data structure used to store the words of dictionary
- HashMap<String,Integer> used to store the wods printed to avoid multiple times printing of same words.
- Arrays used to fetch the words from dictionary.

Algorithms Used :

First ,I’ve extracted all possible words ,from the given word to search, of different length ,them srambled all possiblities.

For each scrambled word ,I’ve checked against the dictionary made, whether he word is present or not, which is implemented using HashTrie.

Also the prined words ust not repeat so, It’s taken care by using a HashMap which contains all the words which were printed previously , thus ensuring repitition of printing words.


Activities
Terminal
Wed 12:23
kavya@u6untu-i8-oa: ~/Desktop/ass-8/q2
File Edit View Search Terminal Help

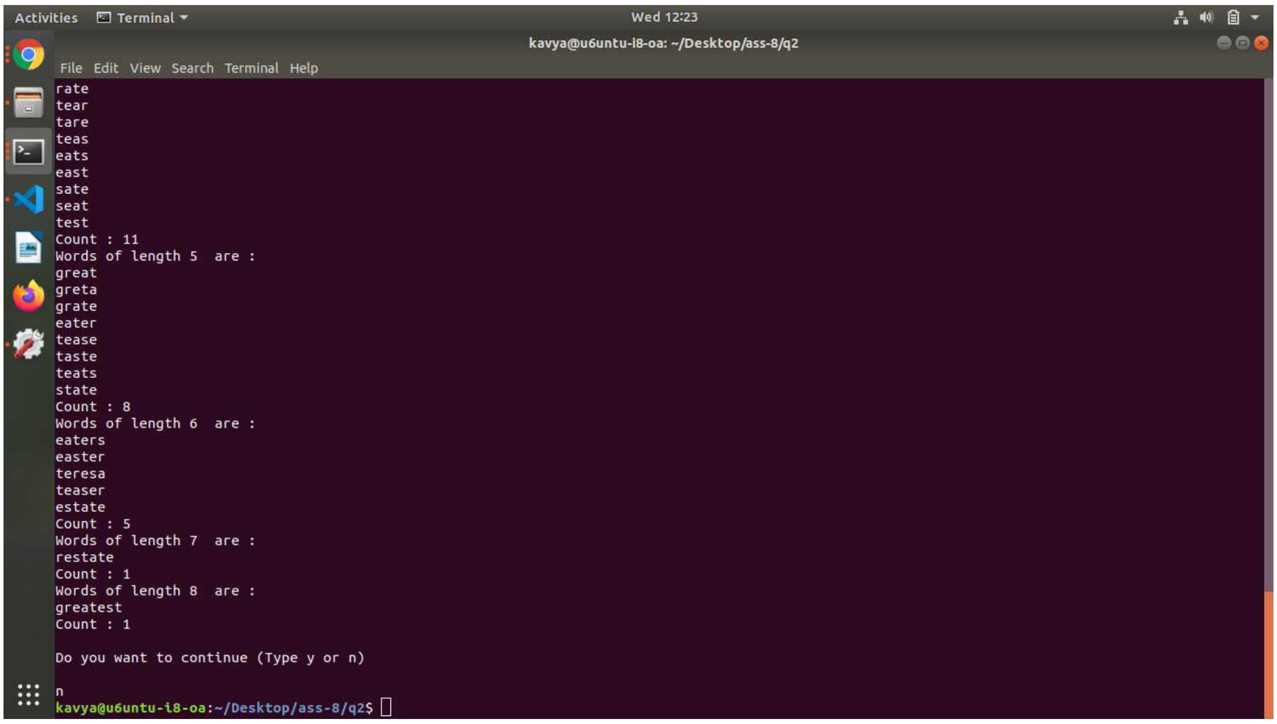
```
kavya@u6untu-i8-oa:~/Desktop/ass-8/q2$ javac q2.java
kavya@u6untu-i8-oa:~/Desktop/ass-8/q2$ java q2
Enter path to for the dictionary of words
dictionary.txt

Enter word to search
great
Search status :
Words of length 4  are :
gear
rage
rate
tear
tare
Count : 5
Words of length 5  are :
great
greta
grate
Count : 3

Do you want to continue (Type y or n)

y

Enter word to search
greatest
Search status :
Words of length 4  are :
gear
rage
rate
tear
tare
teas
eats
east
sate
```


Activities
Terminal
Wed 12:23
kavya@u6untu-i8-oa: ~/Desktop/ass-8/q2
File Edit View Search Terminal Help

```
rate
tear
tare
teas
eats
east
sate
seat
test
Count : 11
Words of length 5  are :
great
greta
grate
eater
tease
taste
teats
state
Count : 8
Words of length 6  are :
eaters
easter
teresa
teaser
estate
Count : 5
Words of length 7  are :
restate
Count : 1
Words of length 8  are :
greatest
Count : 1

Do you want to continue (Type y or n)

n
kavya@u6untu-i8-oa:~/Desktop/ass-8/q2$ 
```

===

Problem Statement 3:

Given an integer array having N number of elements, write a C++ program using hash map (using STL) to find the length of the largest subarray from the given input array, where the summation of the elements of the subarray is equal to n. In the output, if any solution exists then print the starting and ending index (with respect to given input array) of the largest subarray and also print its length. Otherwise, print “Not Found”, as described in the following output.

Input:

N = 8

15 0 2 -3 1 5 3 -2

n = 5

Output:

Length of longest subarray is 5

Index from 1 to 5

===

Data Structure And Algorithm Used :

Data Structures Used :

- Arrays used to store the given values.
- UnOrdered HashMap from c++ STL to store the accumulated sum upto a given index.

Algorithms Used :

- Initialize sum = 0 and maxLen = 0.
- Create a hash table having (sum, index) tuples.
- For i = 0 to n-1, perform the following steps:

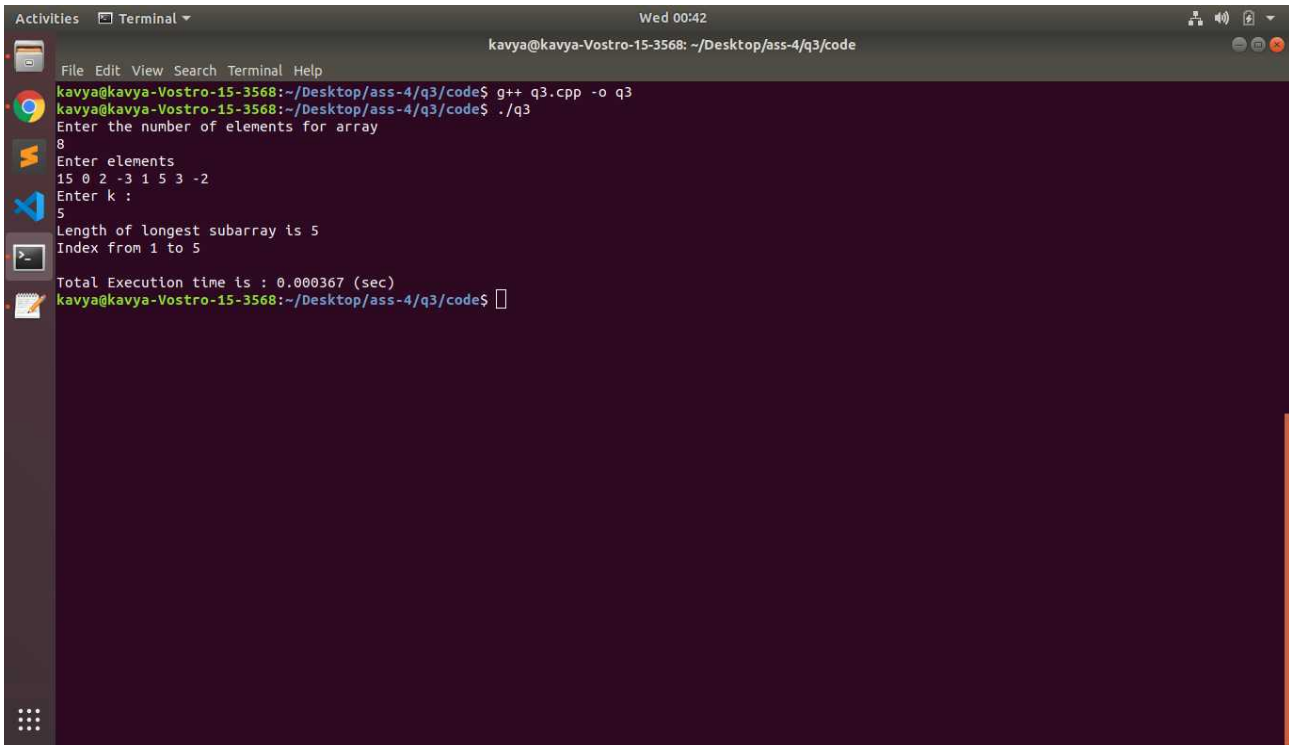
    Accumulate arr[i] to sum.

    If sum == k, update maxLen = i+1.

    Check whether sum is present in the hash table or not. If not present, then add it to the hash table as (sum, i) pair.

    Check if (sum-k) is present in the hash table or not. If present, then obtain index of (sum-k) from the hash table as index. Now check if maxLen < (i-index), then update maxLen = (i-index).

- Return maxLen.


Activities
Terminal
Wed 00:42
kavya@kavya-Vostro-15-3568: ~/Desktop/ass-4/q3/code
File Edit View Search Terminal Help

```
kavya@kavya-Vostro-15-3568:~/Desktop/ass-4/q3/code$ g++ q3.cpp -o q3
kavya@kavya-Vostro-15-3568:~/Desktop/ass-4/q3/code$ ./q3
Enter the number of elements for array
8
Enter elements
15 0 2 -3 1 5 3 -2
Enter k :
5
Length of longest subarray is 5
Index from 1 to 5

Total Execution time is : 0.000367 (sec)
kavya@kavya-Vostro-15-3568:~/Desktop/ass-4/q3/code$ 
```

==