DTC Bus Guide implemented through graphs and trie

Report submitted in partial fulfillment of the requirement for the degree of

Bachelor of Technology

In

Computer Science & Engineering

By

SAKSHAM ARORA

(05415002717)

Maharaj Surajmal Institute of Technology

Affiliated to Guru Gobind Singh Indraprastha University

Janak Puri, New Delhi-110058

2017-2021

Certificate by Student

This is to certify that Report entitled “**DTC Bus Guide implemented through Graphs And Trie**” which is submitted in partial fulfilment of the requirement for the award of degree B.Tech. in Computer Science And Engineering to MSIT, GGSIP University, Dwarka, Delhi comprises only my original work and due acknowledgement has been made in the text to all other material used.

Date: 14 Sept 2019 **Name of Student**

SAKSHAM ARORA (05415002717)

ACKNOWLEDGEMENT

First and foremost, we would like to thank God Almighty for making us capable and fortunate enough that we have been given a chance to pursue an innovative and knowledgeable course like B.Tech. from one of the most reputed universities in our country.

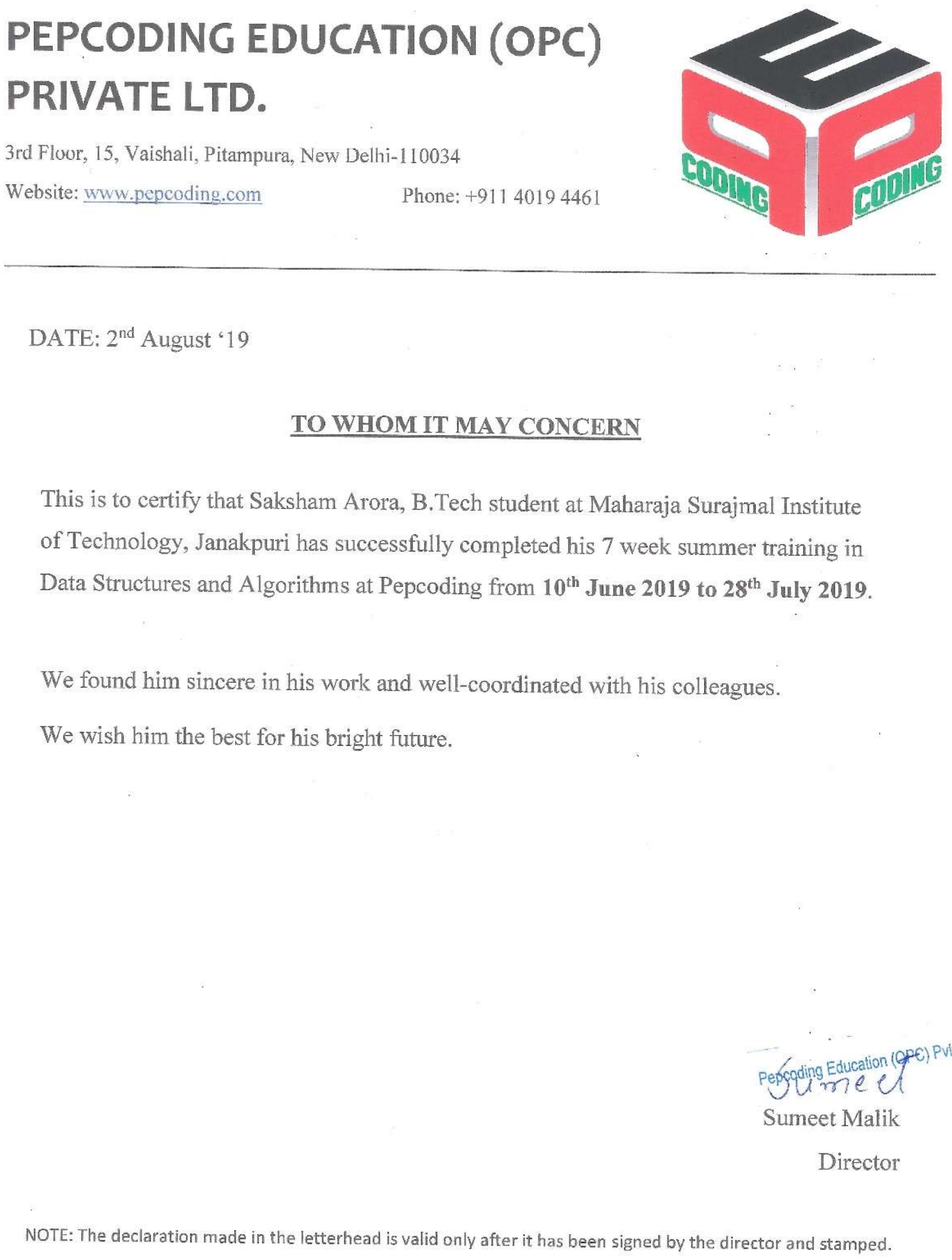
We are very grateful to our director **Dr. K.P. Chaudhary** for giving us the beautiful opportunity of working on this project and to the other faculty members, related to the project, for providing their valuable time and attention to help us in the project.

We are also thankful to **Dr. Sapna Malik** for her most valuable and significant guidance in our understanding of the project and in elaborating our view of studying the project's details and getting the right vision for its implementation.

We would also like to thank our Head of Department, **Dr. Koyel Datta Gupta,** who provided a wide vision of the scope of the project which was very important for us.

At last, we would like to thank all the faculty members and lab in-charge who were directly or indirectly involved in the development of this project by providing much-needed support and cooperation.

Certificate By Organisation



PEPCODING EDUCATION (OPC) PRIVATE LTD.

3rd Floor, 15, Vaishali, Pitampura, New Delhi-110034

Website: www.pepcoding.com Phone: +911 4019 4461

DATE: 2nd August '19

TO WHOM IT MAY CONCERN

This is to certify that Saksham Arora, B.Tech student at Maharaja Surajmal Institute of Technology, Janakpuri has successfully completed his 7 week summer training in Data Structures and Algorithms at Pepcoding from **10th June 2019 to 28th July 2019**.

We found him sincere in his work and well-coordinated with his colleagues.

We wish him the best for his bright future.

Pepcoding Education (OPC) Pvt. Ltd.

Sumeet

Sumeet Malik

Director

NOTE: The declaration made in the letterhead is valid only after it has been signed by the director and stamped.

Abstract

This report presents the five tasks, completed while making this project by the knowledge gained from the summer training done at PepCoding which are listed below:

1) Collecting the data of the buses in Delhi, including their routes.
2) Making of trie of all the Bus stands, and assigning them their keys.
3) Making of Graph of all the bus stands, and connecting them with the buses common to them.
4) Designing an algorithm to find the route between the two bus stands, and modifying it to find the route with minimum changes, and minimum stops as well.
5) Implementing GUI.

Trie helped to achieve better speed at searching for a prefix than the usual text processing/Brute force.

Breadth-First Algorithm(BFS) with some modifications is being deployed to find the routes from one place to another, as it yields good results.

A user-friendly GUI is made and the application is made as a .jar file so as to make it platform-independent.

All the tasks are completed successfully, and the project is completed and is running with no error found.

TABLE OF CONTENTS

List of figures

List of Tables

Chapter 1 : Introduction

1.1 Needs and Objectives:-

Needs:

1. The need of this project is due to the fact, that people in Delhi are not familiar with the bus routes in Delhi, as they are very complex. Most of the people are unaware that which bus to board and where to interchange to reach their destination by applying minimum efforts, according to their needs.
2. Due to the unavailability of proper application for a DTC bus, this application is a must for people who ocassionally or are new to Delhi Public Transport.

Objectives:

1. Collect the data from the web to complete your database and store it as required.
2. Implement Trie to store all the Bus stand, for better search results.
3. Make a graph for the Delhi bus route.
4. Implement appropriate algorithm to find the route with minimum bus stops.
5. Implement appropriate algorithm to find the route with minimum changes.
6. Enhance efficiency.
7. Implement a user-friendly Interface
8. Make an application that is platform-independent.

2.1 Methadology

The methodology, which was used to make this application and accomplish all the objective are as follows :

- **Phase -1 : Initiation**

  The initiation of a project or a system begins when a need or a problem is identified. In this project, the problem was identified a real life-based problem of difficulty in finding the appropriate bus route.

- **Phase -2 : Data Collection**

  In this phase, the data was collected, as soon as the problem was identified. The data was needed to be stored, and manipulated according to the developer. The most difficult part of this phase was to maintain the quality of data which was being collected.

- **Phase -3 : Framework Designing**

  The physical characteristics of the system are designed during this phase. The physical characteristics of the system are specified and a detailed design is prepared. In this phase according to the requirement and functionalities required by the user are laid out. All the structures according to the need are designed for both user interface side and database side.

- **Phase -4 : Development**

  In this phase the detailed specifications produced during the design phase are translated into hardware, communication, and executable software. Backend of the application is prepared according to the framework designing in the above phase.

- **Phase -5 : Implementing GUI**

  A graphical user interface for the user is implemented in this phase. Using this interface, user will be able to use this software. Efforts are made to make the interface user-friendly.

- **Phase -6 : Testing**

  In this phase the software result after development phase is tested and all the problems are noted down. It is analysed that the system fulfilled the demands of most of the users. It was tested on daily travellers and people who rarely travels by Delhi Buses.

2.2 Software Used:-

- For Programming - VS Code
- For Interface - Net Beans IDE
- Software Required - JDK 1.80
- Language Used - Java 8
- Database : Text file

Hardware Used

- Hard disk : 40 GB
- Ram : 8 GB
- OS : Windows/Mac OS

Hardware Required

- Hard disk : 100 MB
- Ram : 512 MB
- OS : Platform Independent

Software Required: JDK 1.80

2.2.1 VS Code:

Visual Studio Code is a lightweight but powerful source code editor that runs on your desktop and is available for Windows, macOS, and Linux. It comes with built-in support for JavaScript, TypeScript, and Node.js and has a rich ecosystem of extensions for other languages (such as C++, C#, Java, Python, PHP, Go) and runtimes (such as .NET and Unity).

It is highly customizable, allowing users to change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality.

Visual Studio Code includes multiple extensions for FTP, allowing the software to be used as a free alternative for web development. Code can be synced between the editor and the server, without downloading any extra software.[1]

2.2.2 Net Beans IDE:

NetBeans IDE is an open-source integrated development environment. NetBeans IDE supports the development of all Java application types (Java

SE (including JavaFX), Java ME, web, EJB and mobile applications) out of the box[2].

All the functions of the IDE are provided by modules. Each module provides a well-defined function, such as support for the Java language, editing, or support for the CVS versioning system, and SVN. NetBeans contains all the modules needed for Java development in a single download, allowing the user to start working immediately. Modules also allow NetBeans to be extended.

The NetBeans Platform is a framework for simplifying the development of Java Swing desktop applications. The NetBeans IDE bundle for Java SE contains what is needed to start developing NetBeans plugins and NetBeans Platform based applications; no additional SDK is required.

The platform offers reusable services common to desktop applications, allowing developers to focus on the logic specific to their application. Among the features of the platform are:

- User interface management (e.g. menus and toolbars)
- User settings management
- Storage management (carries out efficient storage)
- Window management
- Wizard framework (supports step-by-step dialogs)
- NetBeans Visual Library
- Integrated development tools

2.2.3 JDK 1.80:

The Java Development Kit (JDK) is an implementation of either one of the Java Platform, Standard Edition, Java Platform, Enterprise Edition, or Java Platform, Micro Edition platforms released by Oracle Corporation in the form of a binary product aimed at Java developers on Solaris, Linux, macOS or Windows. The JDK includes a private JVM and a few other resources to finish the development of a Java Application.

The JDK also comes with a complete Java Runtime Environment, usually called a private runtime, due to the fact that it is separated from the "regular" JRE and has extra contents. It consists of a Java Virtual Machine and all of the class libraries present in the production environment, as well as additional libraries only useful to developers, such as the internationalization libraries and the IDL libraries.[3]

2.2.4 Java 8

We chose Java because of its following features:

1 In Java, everything is an Object. Java can be easily extended since it is based on the Object model.

2 In Java, everything is an Object. Java can be easily extended since it is based on the Object model.

3 Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform independent byte code. This byte code is distributed over the web and interpreted by the Virtual Machine (JVM) on whichever platform it is being run on.

4 With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.

5 Being architecture-neutral and having no implementation dependent aspects of the specification makes Java portable. Compiler in Java is written in ANSI C with a clean portability boundary, which is a POSIX subset.

1.4 About Organisation

PepCoding is a professional institute for learning software and programing skills, in the field of Computer Science.

It offers courses in various skills, like JAVA Foundation course, Web-Development course, Interview preparation course for IT industry, and many other courses.

PepCoding is an amazing institution to learn programming skills, where one can gather in-depth knowledge of data structures, algorithms and other technical subjects in a really interesting and fun manner.

Their distinctive features are

- Identifying the most effective way to design and deliver learning programs to the participant for particular requirements.

- Defining the right strategy and operating model.
- Customization of programs enabling “Learn with Fun”
- Platform to share best practices and replicate learning.
- Great Collection of Online Resources , Content and Video lectures for the Students.

Chapter 2: Project Design

2.1 Introduction

Project design is an early phase of the project where a project's key features, structure, criteria for success, and major deliverables are all planned out. The point is to develop one or more designs that can be used to achieve the desired project goals.

In this phase, the complex activity of system development is divided into several smaller sub-activities, which coordinate with each other to achieve the main objective of system development.

2.2 Problem Description:

Collect data of all the buses in Delhi and find:-

A. The shortest path between two bus stands, that is, which bus to board from which stand and where to interchange to reach the destination.
B. All the alternatives, for the route such as route with minimum changes and minimum stops, both of them should be displayed.
C. Show route for all the buses.
D. Show all the buses which leave from a particular bus stand.

2.3 Data Flow Diagram:

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both.

It shows how data enters and leaves the system, what changes the information, and where data is stored.

The objective of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communication tool between a system analyst and any person who plays a part in the order that acts as a starting point for redesigning a system. The DFD is also called as a data flow graph or bubble chart[5].

2.3.1 Symbols used in DFD

External Entity

An external entity is a source or destination of a data flow which is outside the area of study. Only those entities which originate or receive data are represented on a business process diagram. The symbol used is an oval containing a meaningful and unique identifier

Process

A process shows a transformation or manipulation of data flows within the system. The symbol used is a rectangular box which contains 3 descriptive elements:
Firstly, an identification number appears in the upper left-hand corner. This is allocated arbitrarily at the top level and serves as a unique reference.
Secondly, a location appears to the right of the identifier and describes where in the system the process takes place. This may, for example, be a department or a piece of hardware. Finally, a descriptive title is placed in the center of the box. This should be a simple imperative sentence with a specific verb, for example 'maintain customer records' or 'find driver'.

Data Flow

A data flow shows the flow of information from its source to its destination. A data flow is represented by a line, with arrowheads showing the direction of flow. Information always flows to or from a process and may be written, verbal or electronic. Each data flow may be referenced by the processes or data stores at its head and tail, or by a description of its contents.

Data Store

A data store is a holding place for information within the system:
It is represented by an open-ended narrow rectangle. Data stores may be long-term files such as sales ledgers, or may be short-term accumulations: for example, batches of documents that are waiting to be processed. Each data store should be given a reference followed by an arbitrary number.

2.3.2 DFD

- **Level-0 DFD**

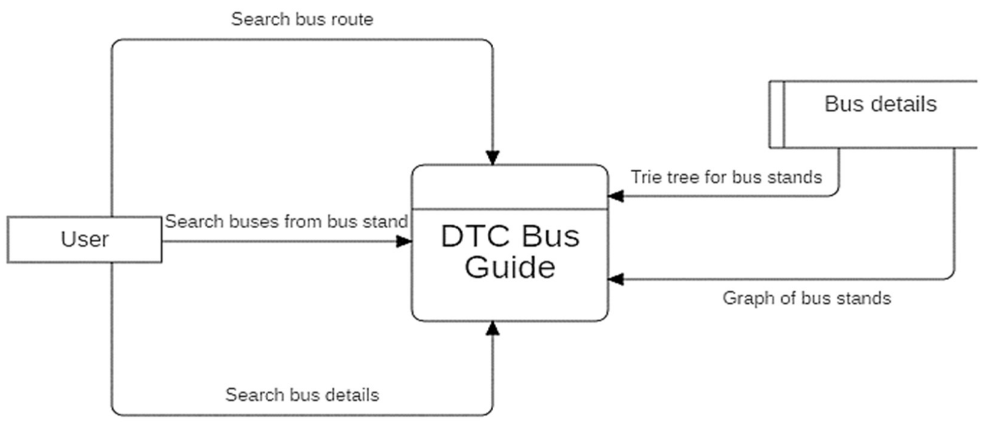


Fig 2.1: Level-0 DFD

- **Level-1 DFD**

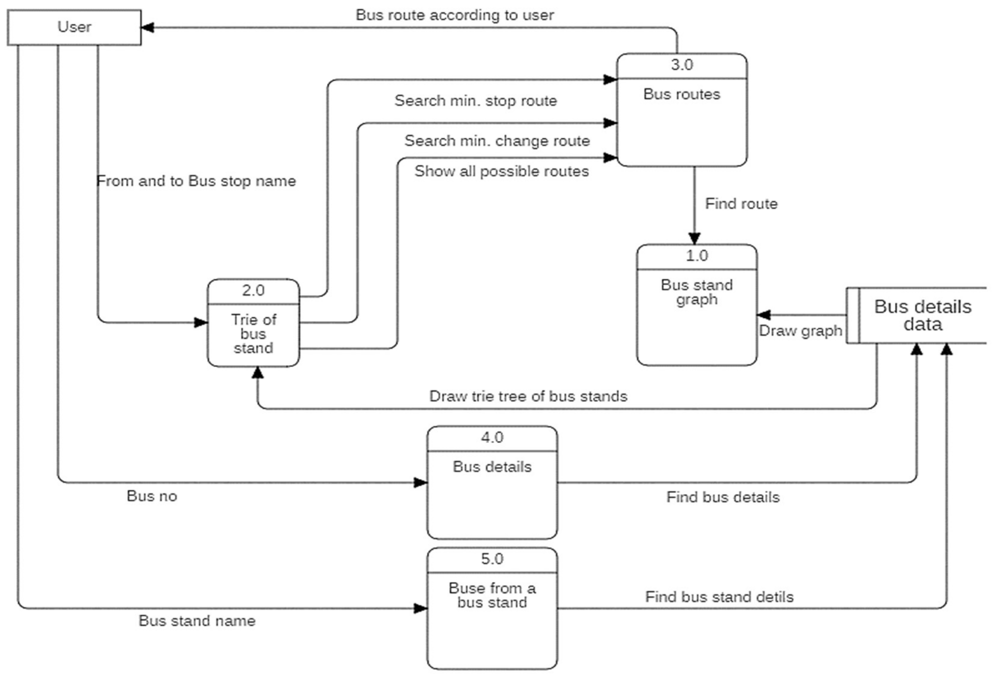


Fig 2.2: Level-1 DFD

2.4 Project Architecture

The software is an implementation of a three-tier model. These three tiers are represented below

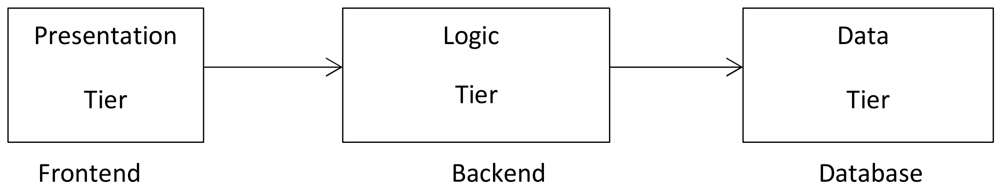


Fig 2.3: Project Architecture

2.4.1 Presentation Tier

The presentation tier is the front-end part of the application, and this tier acts as an interface between the user and the backend. This is made graphical, so known as graphical user interface(GUI). It is made by using libraries such as Swing, and AWT included in the Java Development Kit (JDK). This interface is the intermediate between the user and the system, and it implements the abstract nature of software, hiding the implementation from the user.

2.4.2 Logic Tier

It is the middle tier, also known as logic level, of the architecture. This is where all the “magic” happens, the logic behind the software is implemented here. It controls the functionality of the application. Also known as the application tier it contains the business logic that drives the software's capabilities. It is the most difficult part of the project to implement.

2.4.3 Data Tier

The data tier consists of a database and a program for managing read and write access to a database. This tier may also be referred to as the storage tier. It provides a layer of data abstraction and also protects the data from unauthorized access from a user. Collecting data is one of the time-consuming part of any project, quality of data affects the result of the project, so it is important to ensure the quality of data. The text file is oldest technique of storing data, which has been used in this project.

2.5 Text File System in Java

There are multiple ways of writing and reading a text file. this is required while dealing with many applications.

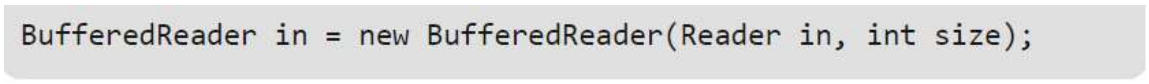
There are several ways to read a plain text file in Java e.g. you can use FileReader, BufferedReader or Scanner to read a text file. Every utility provides something special e.g. BufferedReader provides buffering of data for fast reading, and Scanner provides parsing ability.

We can also use both BufferReader and Scanner to read a text file line by line in Java. Then Java SE 8 introduces another Stream class java.util.stream. Stream which provides a lazy and more efficient way to read a file[6].

2.5.1 Buffered Reader

This method reads text from a character-input stream. It does buffering for efficient reading of characters, arrays, and lines.
The buffer size may be specified, or the default size may be used. The default is large enough for most purposes.

In general, each read request made of a Reader causes a corresponding read request to be made of the underlying character or byte stream. It is therefore advisable to wrap a BufferedReader around any Reader whose read() operations may be costly, such as File Readers and InputStreamReaders.

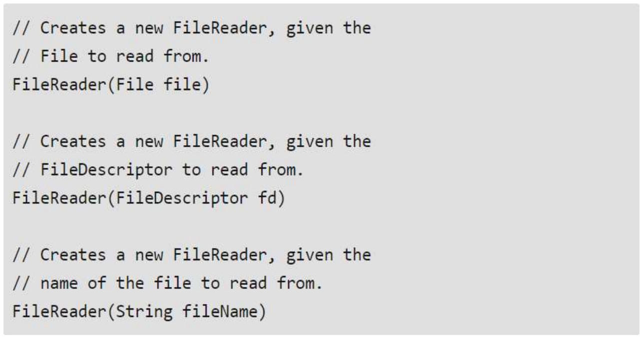


```
BufferedReader in = new BufferedReader(Reader in, int size);
```

Figure 2.4: BufferedReader Syntax[11]

2.5.2 File Reader Class

Convenience class for reading character files. The constructors of this class assume that the default character encoding and the default byte-buffer size are appropriate.



```
// Creates a new FileReader, given the
// File to read from.
FileReader(File file)

// Creates a new FileReader, given the
// FileDescriptor to read from.
FileReader(FileDescriptor fd)

// Creates a new FileReader, given the
// name of the file to read from.
FileReader(String fileName)
```

Figure 2.5: File Reader Class Syntax[11]

2.5.3 Scanner Class

Scanner is a class in java.util package used for obtaining the input of the primitive types like int, double, etc. and strings. It is the easiest way to read input in a Java program, though not very efficient if you want an input method for scenarios where time is a constraint like in competitive programming.

- To create an object of Scanner class, we usually pass the predefined object System.in, which represents the standard input stream. We may pass an object of class File if we want to read input from a file.
- To read the numerical values of a certain data type XYZ, the function to use is nextXYZ(). For example, to read a value of type short, we can use nextShort().
- To read strings, we use nextLine().
- To read a single character, we use next().charAt(0). next() function returns the next token/word in the input as a string and charAt(0) function returns the first character in that string.

Sometimes, we have to check if the next value we read is of a certain type or if the input has ended (EOF marker encountered).
Then, we check if the scanner's input is of the type we want with the help of hasNextXYZ() functions where XYZ is the type we are interested in. The function

returns true if the scanner has a token of that type, otherwise false. For example, in the below code, we have used hasNextInt(). To check for a string, we use nextLine(). Similarly, to check for a single character, we use hasNext().charAt(0).

2.6 Graphs

Graphs are mathematical structures that represent pairwise relationships between objects. A graph is a flow structure that represents the relationship between various objects.

It consists of the following two components:

1. A finite set of vertices also called nodes.

2. A finite set of ordered pairs of the form (u, v) called as edge. The pair is ordered because (u, v) is not same as (v, u) in case of a directed graph(digraph). The pair of the form (u, v) indicates that there is an edge from vertex u to vertex v. The edges may contain weight/value/cost.

Graphs are used to represent many real-life applications: Graphs are used to represent networks. The networks may include paths in a city or telephone network or circuit network. Graphs are also used in social networks like LinkedIn, Facebook. For example, In Facebook, each person is represented with a vertex(or node). Each node is a structure and contains information like person id, name, gender, and locale. Following is an example of an undirected graph with 5 vertices.

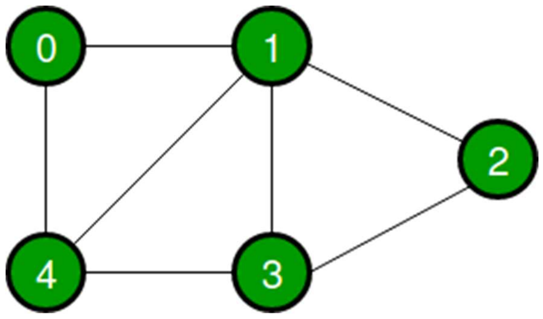


Fig 2.6: Graph[12]

2.6.1 Representation

The following two are the most commonly used representations of a graph.

1. Adjacency Matrix

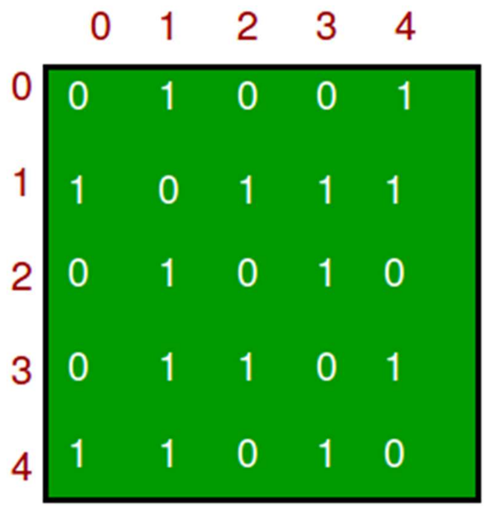
2. Adjacency List

There are other representations also like, Incidence Matrix and Incidence List. The choice of graph representation is situation-specific. It totally depends on the type of operations to be performed and ease of use.

2.6.1.1 Adjacency Matrix

Adjacency Matrix is a 2D array of size V x V where V is the number of vertices in a graph. Let the 2D array be adj[][], a slot adj[i][j] = 1 indicates that there is an edge from vertex i to vertex j. The adjacency matrix for the undirected graph is always symmetric. Adjacency Matrix is also used to represent weighted graphs. If adj[i][j] = w, then there is an edge from vertex i to vertex j with weight w.

The adjacency matrix for the above example graph is:



$$\begin{array}{c|ccccc} & 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 2 & 0 & 1 & 0 & 1 & 0 \\ 3 & 0 & 1 & 1 & 0 & 1 \\ 4 & 1 & 1 & 0 & 1 & 0 \end{array}$$

Fig 2.7: Adjacency Matrix[12]

Pros: Representation is easier to implement and follow. Removing an edge takes O(1) time. Queries like whether there is an edge from vertex ‘u’ to vertex ‘v’ are efficient and can be done O(1).

Cons: Consumes more space O(V^2). Even if the graph is sparse(contains less number of edges), it consumes the same space. Adding a vertex is O(V^2) time. Please see this for a sample Python implementation of the adjacency matrix.

2.6.1.2 Adjacency List:

An array of lists is used. The size of the array is equal to the number of vertices. Let the array be array[]. An entry array[i] represents the list of vertices adjacent to the ith vertex. This representation can also be used to represent a weighted graph. The weights of edges can be represented as lists of pairs. Following is adjacency list representation of the above graph[7].

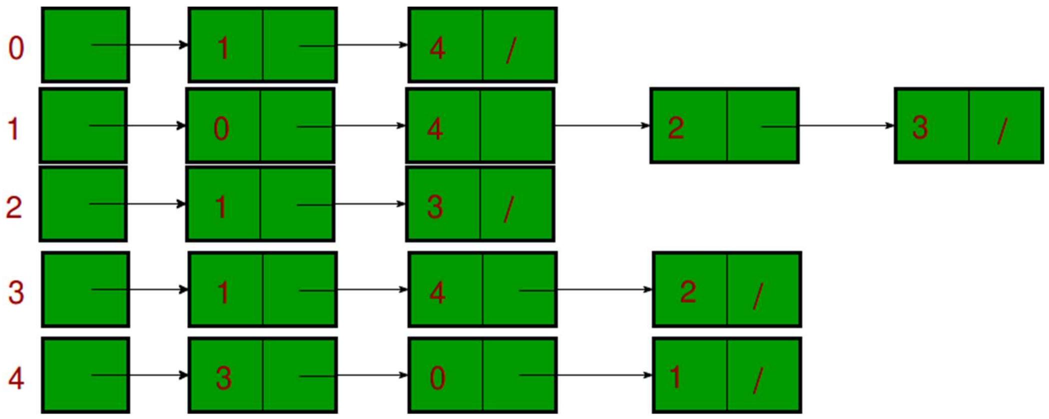


Fig 2.8: Adjacency List[12]

2.6.2 Traversal:

Graph traversal refers to the process of visiting (checking and/or updating) each vertex in a graph. Such traversals are classified by the order in which the vertices are visited. Broadly there are two types of traversal used for traversing a graph which are:

- Depth-First Search (DFS)
- Breadth-First Search (BFS)

2.6.2.1 Depth First Search :

A depth-first search (DFS) is an algorithm for traversing a finite graph. DFS visits the child vertices before visiting the sibling vertices that is, it traverses the depth of any particular path before exploring its breadth. A stack (or recursion) is generally used when implementing the algorithm.

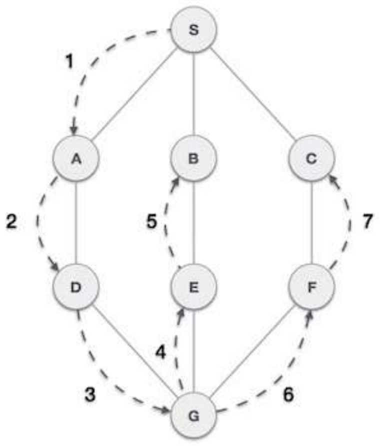


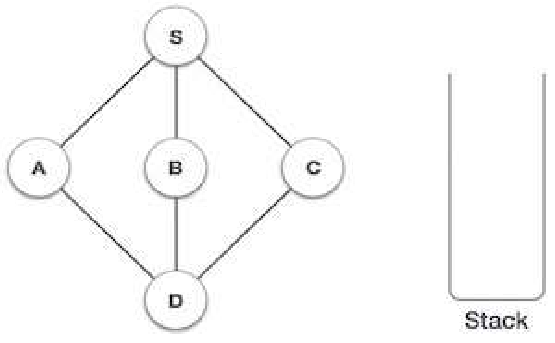
Fig 2.9: DFS Iteration[13]

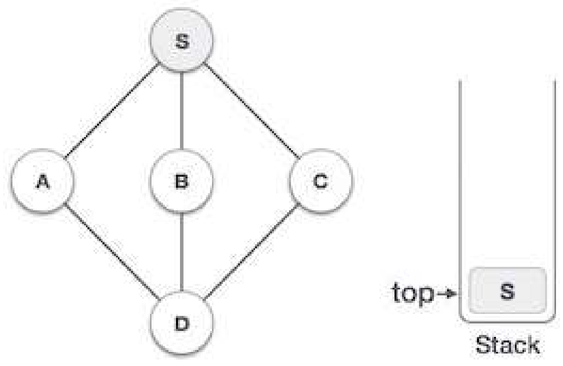
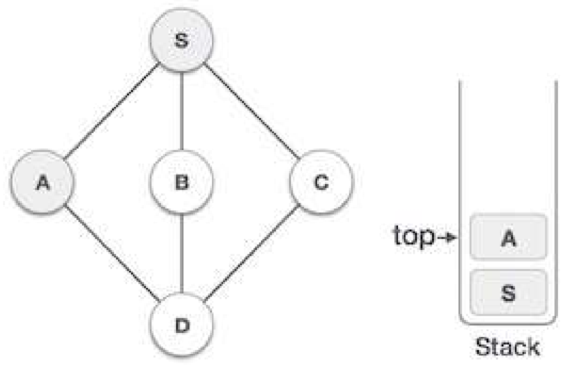
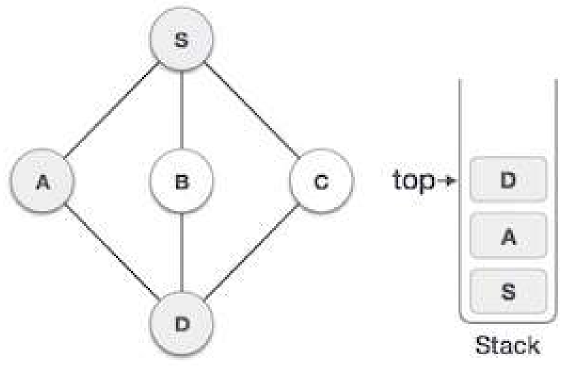
As in the example given above, DFS algorithm traverses from S to A to D to G to E to B first, then to F and lastly to C. It employs the following rules.

Rule 1 − Visit the adjacent unvisited vertex. Mark it as visited. Display it. Push it in a stack.

Rule 2 − If no adjacent vertex is found, pop up a vertex from the stack. (It will pop up all the vertices from the stack, which do not have adjacent vertices.)

Rule 3 − Repeat Rule 1 and Rule 2 until the stack is empty.

Step	Traversal	Description
1		Initialize the stack.

2		Mark S as visited and put it onto the stack. Explore any unvisited adjacent node from S. We have three nodes and we can pick any of them. For this example, we shall take the node in alphabetical order.
3		Mark A as visited and put it onto the stack. Explore any unvisited adjacent node from A. Both S and D are adjacent to A but we are concerned for unvisited nodes only.
4		Visit D and mark it as visited and put onto the stack. Here, we have B and C nodes, which are adjacent to D and both are unvisited. However, we shall again choose in alphabetical order.

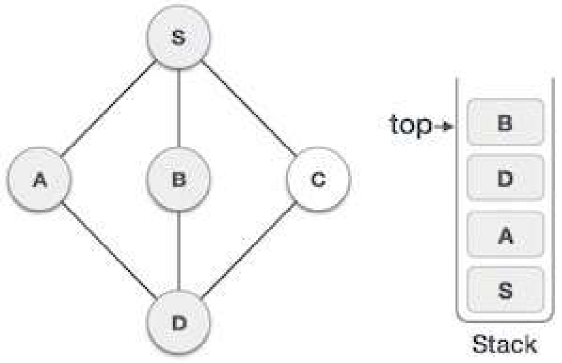
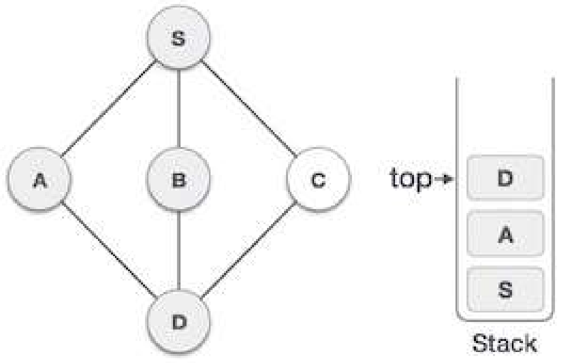
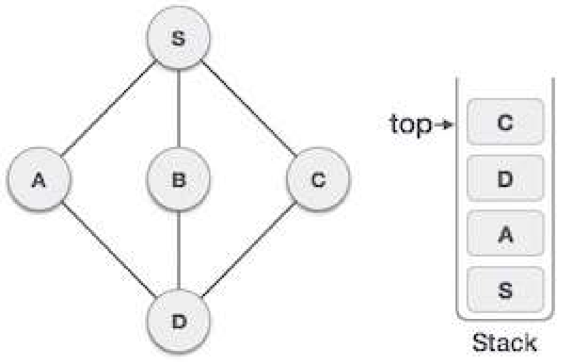
5		We choose B, mark it as visited and put onto the stack. Here B does not have any unvisited adjacent node. So, we pop B from the stack.
6		We check the stack top for return to the previous node and check if it has any unvisited nodes. Here, we find D to be on the top of the stack.
7		An only, unvisited adjacent node is from D is C now. So we visit C, mark it as visited and put it onto the stack.

Table 2.1: DFS Iteration[13]

As C does not have any unvisited adjacent node so we keep popping the stack until we find a node that has an unvisited adjacent node. In this case, there's none and we keep popping until the stack is empty[8].

2.6.2.2 Breadth-First Search:

A breadth-first search (BFS) is a technique for traversing a finite graph. BFS visits the sibling vertices before visiting the child vertices, and a queue is used in the search process. This algorithm is often used to find the shortest path from one vertex to another.

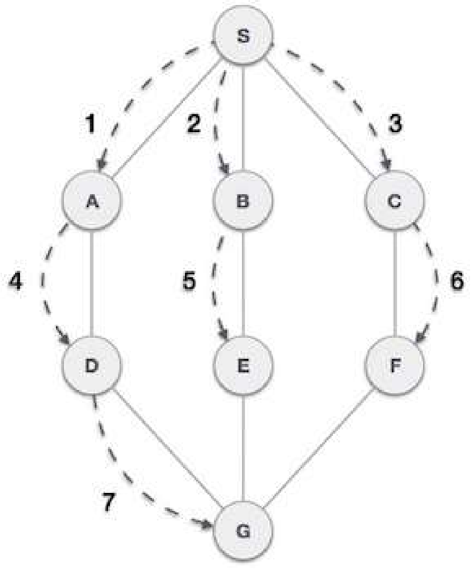


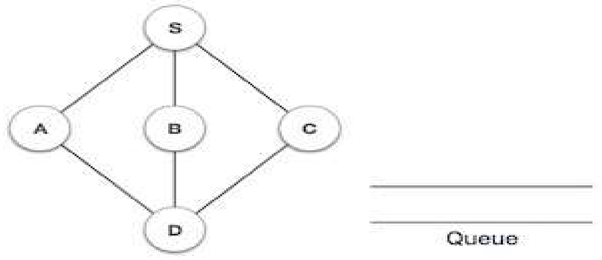
Fig 2.10: BFS Iteration[14]

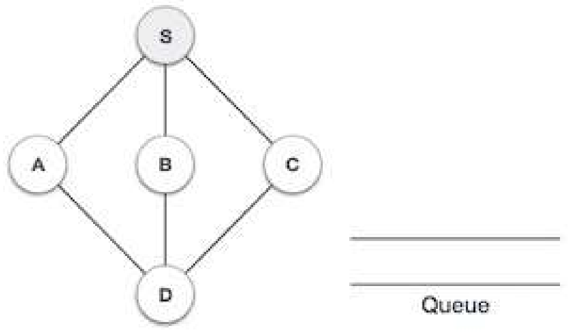
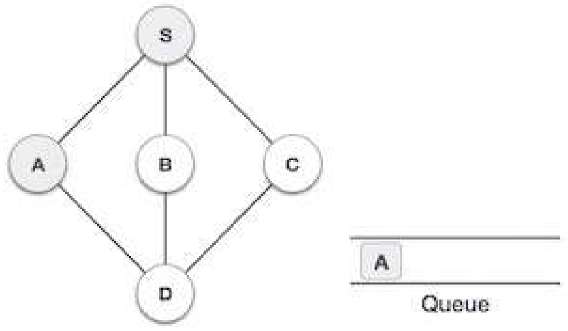
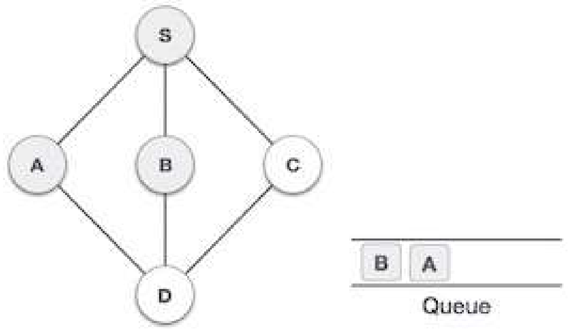
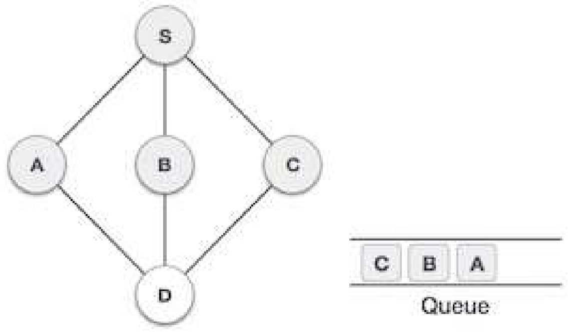
As in the example given above, the BFS algorithm traverses from A to B to E to F first then to C and G lastly to D. It employs the following rules.

Rule 1 − Visit the adjacent unvisited vertex. Mark it as visited. Display it. Insert it in a queue.

Rule 2 − If no adjacent vertex is found, remove the first vertex from the queue.

Rule 3 − Repeat Rule 1 and Rule 2 until the queue is empty.

Step	Traversal	Description
1		Initialize the queue.

2		We start from visiting S (starting node), and mark it as visited.
3		We then see an unvisited adjacent node from S. In this example, we have three nodes but alphabetically we choose A, mark it as visited and enqueue it.
4		Next, the unvisited adjacent node from S is B. We mark it as visited and enqueue it.
5		Next, the unvisited adjacent node from S is C. We mark it as visited and enqueue it.

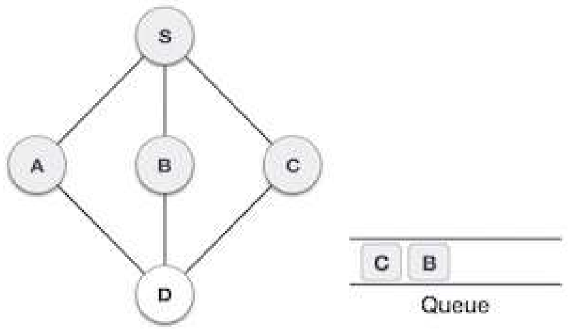
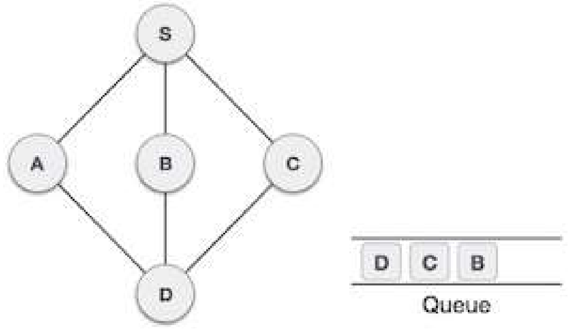
6		Now, S is left with no unvisited adjacent nodes. So, we dequeue and find A.
7		From A we have D as an unvisited adjacent node. We mark it as visited and enqueue it.

Table 2.2 BFS Iteration[14]

At this stage, we are left with no unmarked (unvisited) nodes. But as per the algorithm we keep on dequeuing in order to get all unvisited nodes. When the queue gets emptied, the program is over[9].

2.7 Trie

A Trie is a special tree-like data structure, wherein characters of a string are stored as nodes. Trie is an efficient information re*Trie*val data structure. Using Trie, search complexities can be brought to an optimal limit (key length). If we store keys in the binary search tree, a well-balanced BST will need time proportional to M * log N, where M is maximum string length and N is number of keys in tree. Using Trie, we can search the key in O(M) time.

It consists of nodes and edges. Each node consists of a maximum of 26 children and edges connect each parent node to its children. These 26 pointers are nothing but pointers for each of the 26 letters of the English alphabet A separate edge is maintained for every edge.

Strings are stored in a top to bottom manner on the basis of their prefix in a trie. All prefixes of length 1 are stored at until level 1, all prefixes of length 2 are sorted at until level 2 and so on.[10]

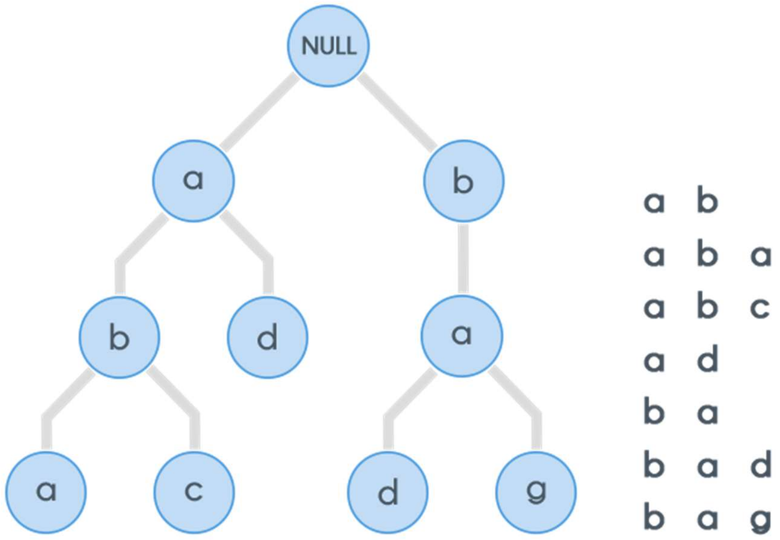


Fig 2.11: Trie[15]

Insert and search costs O(key_length), however the memory requirements of Trie where N is number of keys in Trie. The searching cost of a prefix is O(prefix_size).

Chapter-3: Implementation

3.1 Code Structure

- The project is implemented using 4 classes are defined: trieData (for implementation of trie), graphMaker (To construct a graph and give results), DtcBus (class to do the linking of the above two classes, and load data). MainScreen (to implement an interface).
- trieData
  - It contains two classes one ‘op’ which is for the output of the trie prefix search having members: integer value to store busStand ID, and a string to store bus stand name
  - The second class is named “node” and it stores a character value, integer value for storing ID, ArrayList of type node to store links with other nodes.
  - insert(): To insert a string in the trie.
  - Search(): To search a prefix in the trie created, return an ArrayList of type ‘op’.
- graphMaker
  - It contains graph which is an ArrayList of type ArrayList of ‘edge’.
  - It contains five classes area, DTC, edge, pp, resultPair.
  - Objects of class ‘area’ contains the details of a bus stand: areaName, areaCode, busIn, busOut.
  - Objects of class ‘DTC’ contains the details of a bus: code, busRouteNo, start, end, direction, ArrayList of type string to store intermediate Stops, ArrayList of type integer to store id’s of intermediate Stops.
  - Class ‘edge’ is used to construct graph.
  - Class ‘pp’ is used in the algorithms as a queue to find a route from one bus stand to another.

    - Class ‘resultPair’ is used to return the result generated by the search function.
    - searchAreaIfExistsOrNot(): return the index where area is found in the Array of strings named aName if found else returns -1.
    - searchDtcBust(): return the index where DtcBus name is found in the Array of strings named bName if found else returns -1.
    - displayDTC(): Displays the bus name and its route.
    - checkNbg(): Returns Boolean by checking whether the given ids of the bus Stand are neighbors or not.
    - bfsMinStop(): returns object of class ‘resultPair’ by giving route with minimum stop.
    - bfsMinChange(): returns an object of class ‘resultPair’ by giving route with minimum change.
    - bfsAllRoute(): returns ArrayList of class ‘resultPair’ by giving maximum of 10 routes if possible.
- DtcBus
    - In this class objects of class trie and graphMaker are created, and loading of data from the text file is done.
- MainScreen
    - The Graphical user interface(GUI) is implemented by using Swing and AWT libraries of java.

3.2 Implementation Blocks

The objective of this part is to implement all the details we studied in the project designing part of the report. The implementation of all important modules has been explained below.

3.2.1 BFS Route Search (Minimum Change)

1. Initialize Boolean array visited[] of size equal to total no of bus stands
2. Initialize queue and insert source bus stand id into queue
3. Repeat step 4 to while queue.size() > 0

```
4. Remove first element from queue and rem <- first element
5. Check if first element is visited  then continue
6. If there is direct bus from rem to destination stop
    a. Check if total no of changes are less than minimum changes ,set
       minimum change equal to present change
    b. Insert present stop id and bus id in arraylist
    c. Set present total stop equal to total intermediate stop
7. Insert all non visited next bus stop of all buses that are going from rem bus top
8. Return arraylist that contain id of bus stop and bus stand
9. Exit
```

3.2.2 BFS Route Search (Minimum Stops)

```
1. Initialize Boolean array visited[] of size equal to total no of bus stands
2. Initialize queue and insert source bus stand id into queue
3. Repeat step 4 to 8 while queue.size() > 0
4. Remove first element from queue and rem <- first element
5. Check if first element is visited  then continue
6. If there is direct bus from rem to destination stop
    a. Check if total no of stops are less than minimum stop ,set minimum stop
       equal to present stop
    b. Insert present stop id and bus id in arraylist
    c. Set present total change equal to rem total change + 1
7. Insert all non visited next bus stop of all buses that are going from rem bus top
8. Return arraylist that contain id of bus stop and bus stand
9. Exit
```

3.2.3 Trie Insertion

```
1. If length of input string equals 0, the return.
2. Set string t = x in lowercase.
3. Set index = t[0]
    a. if index >= 48 and index <=5, set index = index+=25-48;
```

```
        b. else t[0] = '.' , set index = 36
        c. else t[0] = '#',  set index = 37
        d. else t[0] = '/',  set index = 38
        e. else t[0] = '(' or ')' ,  set index = 39
        f. else t[0] – 'a' < 0,  set index = 40
    4. if the size of child ArrayList is shorter than index,
        a. Then add null till size is index+1, then add child and then call the the
           function recursively with substring(1) of the input string.
    5. Else make a recursive call, wth substring(1) of the input string.
    6. Exit
```

3.2.4 Trie Search

```
    1. If reached a root that has no children, then
        a. If length of the string to be searched equals 0, then return an ArrayList
           after adding psf.
        b. Else check is length is 1 and if that character exists in the ArrayList
           then also return an Arraylist after adding psf.
        c. Else return an emty Arraylist.
    2. If length of the string to be searched equals 0, then make call to all the childs
       in the child ArrayList of that node where child is not null.
    3. Make a call to the first character of the string to be searched.
    4. In all the recusive calls made, take output in the ArrayList, and copy all the
       strings to a result ArrayString.
    5. Return res.
    6. Exit
```

Chapter 4: Result and Discussion

4.1 Result

All the modules have been successfully implemented and further details have been discussed below.

4.1.1 Main Screen

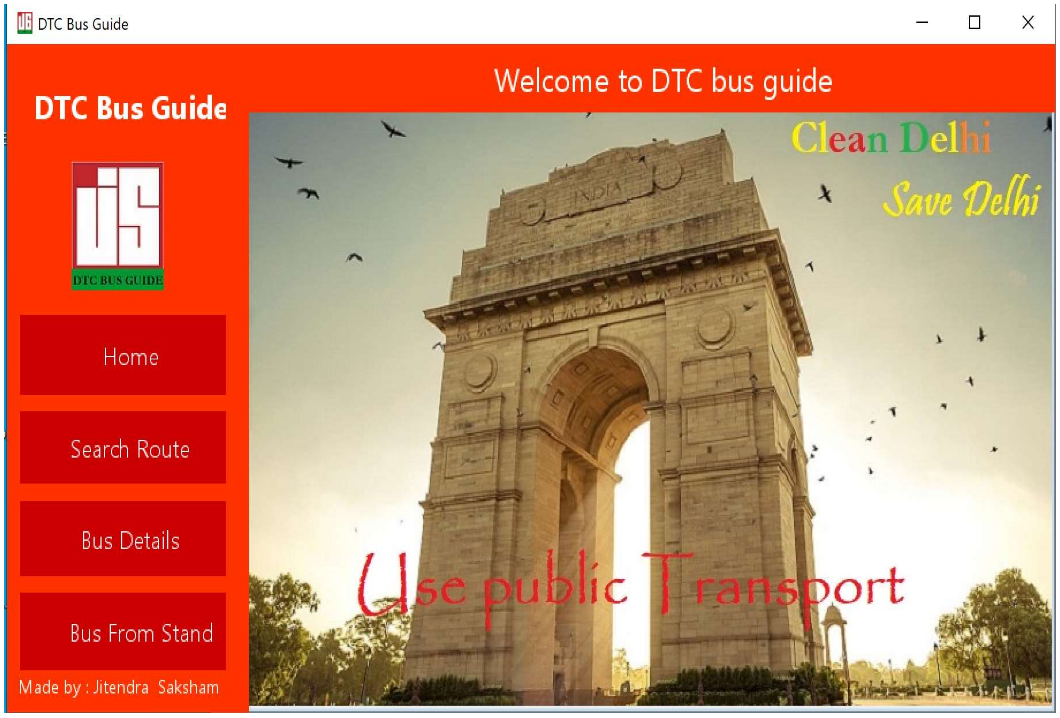


Fig 4.1: Main Screen

The screenshot provided above is the home page of the application. Home has four buttons left alligned with modules name written on them. On clicking any button the respective module gets open. All the options are explained below:

1. **Home**

   By clicking, this button user will br brought to the home page, no matter where he is.

2. **Search Route**

   This option will open a window, where user can enter the destination and the start point and con choose from a variety of options of route.

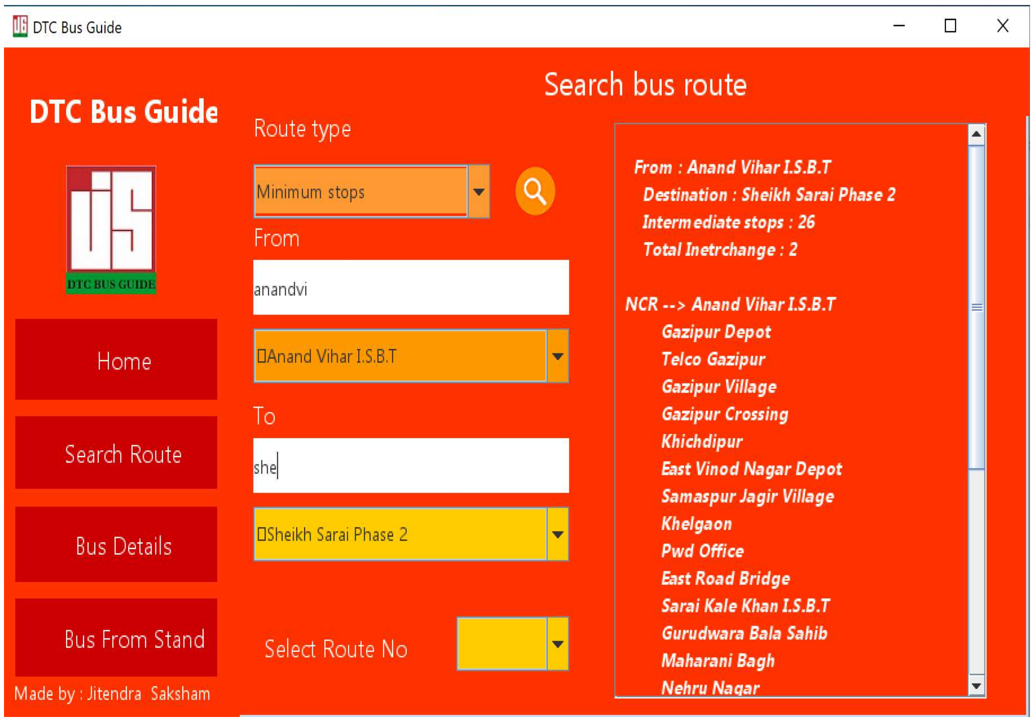
3. **Bus Details**

   This option will open a window where a user can select a particular bus and follow its route.

4. **Bus Stand Info**

   This option will open a window where a user can search for a particular bus Stand and get to know all the Buses which leaves that stand. Atmost 15 Buses will be shown as result.

4.1.2 Search Screen



anandvi

Anand Vihar I.S.B.T

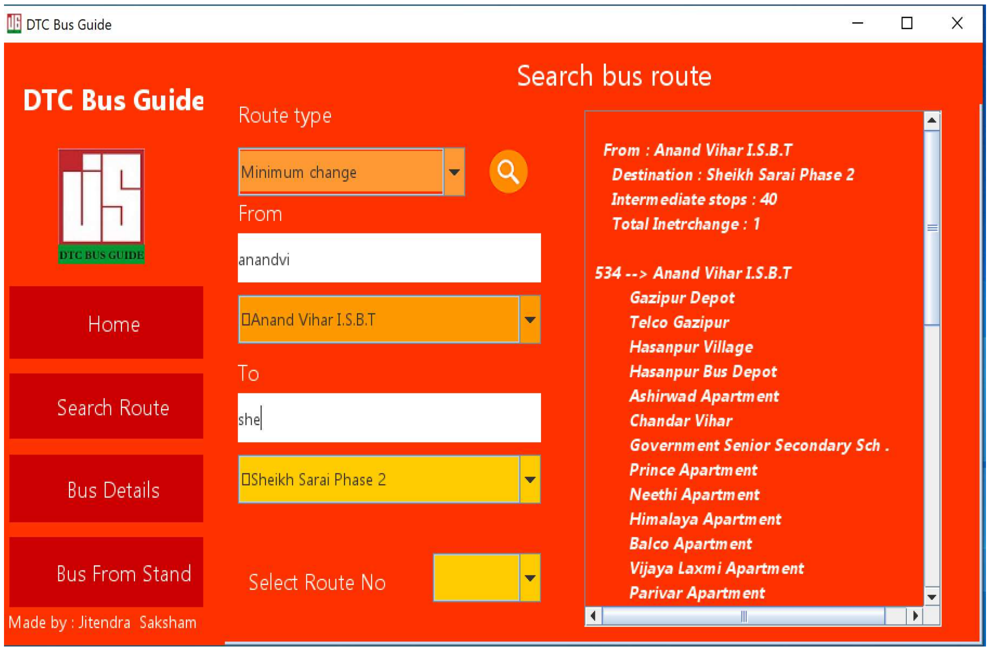
she

Sheikh Sarai Phase 2

From : Anand Vihar I.S.B.T
Destination : Sheikh Sarai Phase 2
Intermediate stops : 26
Total Inetrchange : 2

NCR --> Anand Vihar I.S.B.T
Gazipur Depot
Telco Gazipur
Gazipur Village
Gazipur Crossing
Khichdipur
East Vinod Nagar Depot
Samaspur Jagir Village
Khelgaon
Pwd Office
East Road Bridge
Sarai Kale Khan I.S.B.T
Gurudwara Bala Sahib
Maharani Bagh
Nehru Nagar

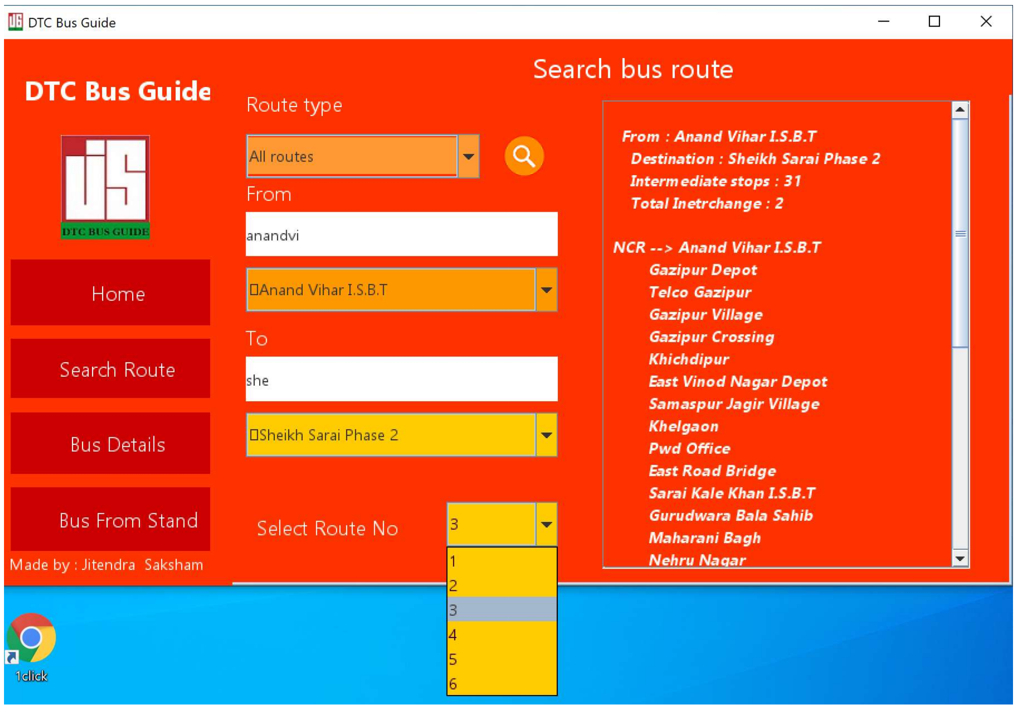
Fig 4.2: Search Screen(A)



```
From : Anand Vihar I.S.B.T
  Destination : Sheikh Sarai Phase 2
  Intermediate stops : 40
  Total Inetrchange : 1

534 --> Anand Vihar I.S.B.T
      Gazipur Depot
      Telco Gazipur
      Hasanpur Village
      Hasanpur Bus Depot
      Ashirwad Apartment
      Chandar Vihar
      Government Senior Secondary Sch .
      Prince Apartment
      Neethi Apartment
      Himalaya Apartment
      Balco Apartment
      Vijaya Laxmi Apartment
      Parivar Apartment
```

Fig 4.3: Search Screen(B)



```
From : Anand Vihar I.S.B.T
  Destination : Sheikh Sarai Phase 2
  Intermediate stops : 31
  Total Inetrchange : 2

NCR --> Anand Vihar I.S.B.T
      Gazipur Depot
      Telco Gazipur
      Gazipur Village
      Gazipur Crossing
      Khichdipur
      East Vinod Nagar Depot
      Samaspur Jagir Village
      Khelgaon
      Pwd Office
      East Road Bridge
      Sarai Kale Khan I.S.B.T
      Gurudwara Bala Sahib
      Maharani Bagh
      Nehru Nagar
```

Fig 4.4: Search Screen(C)

The screenshots provided above is the Search page of the application. It has two text box, and three dropdowns, and a display box.

1. **Route Type**

   It is a drop down which will help you select the type of route to be displayed, that is, route with minimum stop, minimum change, and all routes.

2. **From**

   In this, the autocomplete feature of the trie is being used, and it based on the prefix search, in this the bus Stand from where user have to board the bus has to be mentioned.

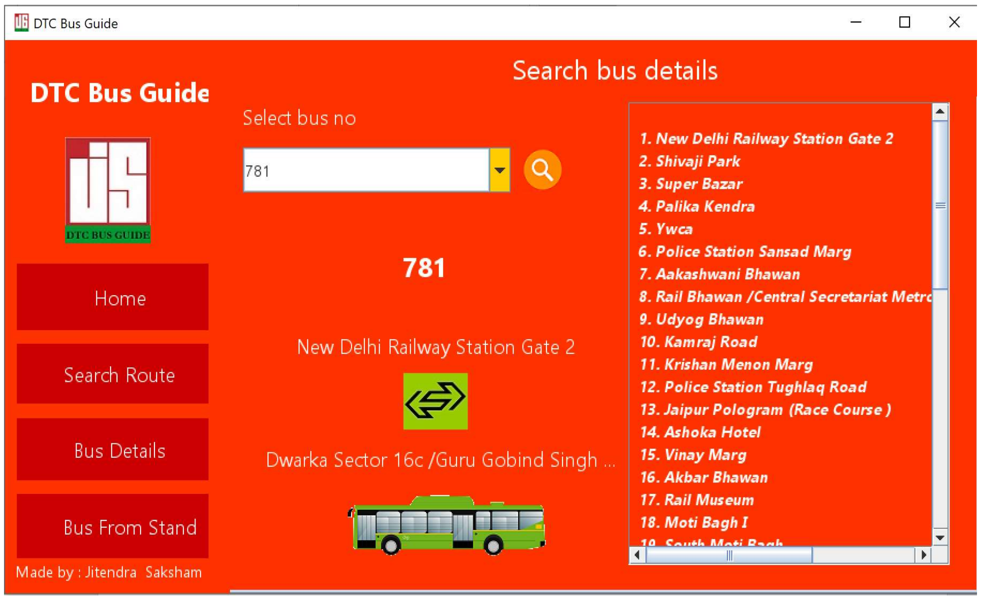
3. **To**

   In this, the autocomplete feature of the trie is being used, and it based on the prefix search. In this the bus Stand where user have to deboard, that is, the destination has to be mentioned.

3. **Display Box**

   The result of the search will be displayed, in the display Box on the rigt side of the window.

4.1.3 Bus info Screen



781

781

New Delhi Railway Station Gate 2

Dwarka Sector 16c /Guru Gobind Singh ...

1. New Delhi Railway Station Gate 2
2. Shivaji Park
3. Super Bazar
4. Palika Kendra
5. Ywca
6. Police Station Sansad Marg
7. Aakashwani Bhawan
8. Rail Bhawan /Central Secretariat Metrc
9. Udyog Bhawan
10. Kamraj Road
11. Krishan Menon Marg
12. Police Station Tughlaq Road
13. Jaipur Pologram (Race Course)
14. Ashoka Hotel
15. Vinay Marg
16. Akbar Bhawan
17. Rail Museum
18. Moti Bagh I

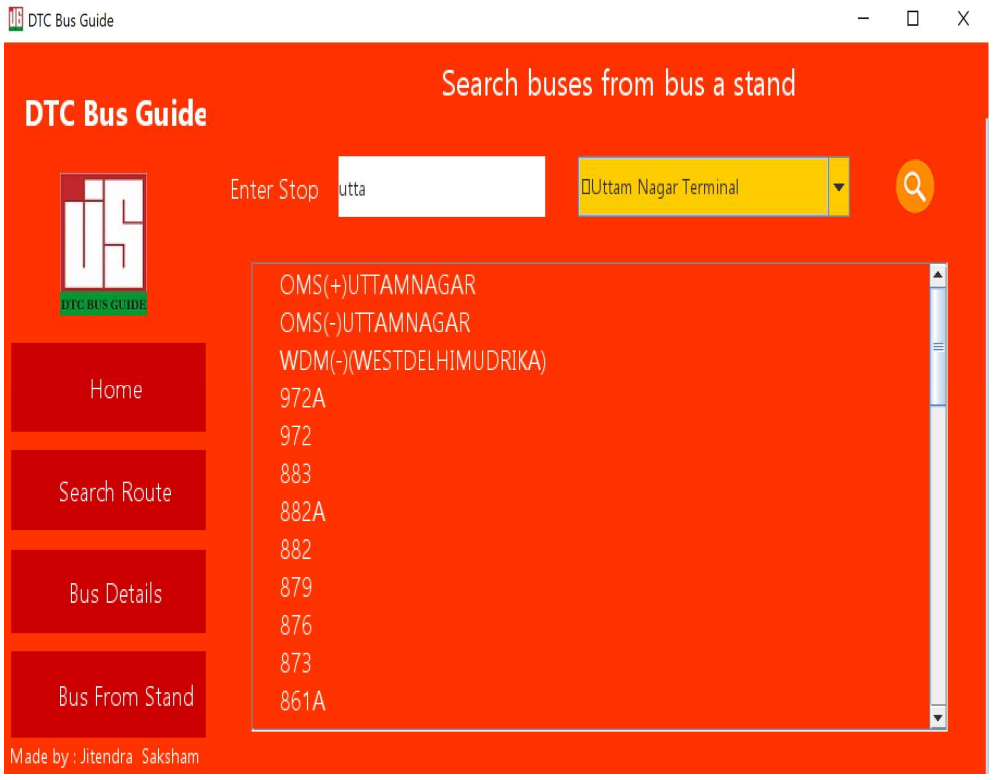
Fig 4.5: Bus Details Screen

The screenshot provided above is the Bus Info of the application. It has one text box, and one dropdowns, and a display box.

You need to select the bus from the dropdown to see its route.

The route of the bus is displayed on the display box which is on the right side of the window.

4.1.4 Bus From Stand



OMS(+)UTTAMNAGAR
OMS(-)UTTAMNAGAR
WDM(-)(WESTDELHIMUDRIKA)
972A
972
883
882A
882
879
876
873
861A

Fig 4.6: Bus From Stand Screen

The screenshot provided above is the Bus From Stand of the application. It has one text are, and one dropdown, and a text box.

User need to type the prefix of the bus Stand name and it will be autocompleted and then user need to click the search icon and then the buses related to that bus Stand will be shown in the text area, it the total number of buses exceed the limit that is allocated in the text area, then a scroll bar will b shown.

4.2 Discussion

Now we will discuss the advantages and disadvantages of all the techniques used, and why we used it.

4.2.1 Using BFS

The greatest thing about BFS is that it is **complete**. That means BFS will **always** find a solution to a problem if a solution exists. If a path exists that connects two nodes in a finite graph, BFS will find it no matter how much time it takes.
BFS explores nodes one depth level at a time. It starts from a node, then checks the neighbors of the initial node, then the neighbors of the neighbors and so on.
Completeness is a nice-to-have feature for an algorithm, but in the case of BFS, this comes at a high cost. The execution time of BFS is fairly slow because the complexity of the algorithm is **exponential** — O(n^2). What's worse are the memory requirements. That's because BFS has to keep track of all the nodes it explores. When applied to huge graphs, the expensive memory requirements make BFS unfeasible.

4.2.2 Why not use DFS?

BFS has a nice property that it will check all the edges from the root and keep the distance from the root to the other nodes as minimal as possible, but dfs simply jumps to the first adjacent node and goes depth-wise. You can modify DFS to get the shortest path, but you will only end up in an algorithm that is of higher time complexity or will end up doing the same thing BFS does. As the shortest path requires a queue to be used to get results in a viable time.

4.2.3 Using Trie

Space-Efficient: If you're storing lots of words that start with similar patterns, tries may reduce the overall storage cost by storing shared prefixes once.
Efficient Prefix Queries: Tries can quickly answer queries about words with shared prefixes.

4.2.4 Trie Vs Sets

- For the trie, you'd have to walk from the root of the trie through k nodes, one character at a time.
- For the hash set, you have to compute a hash value from all k characters of the string in order to index into the underlying array.

Chapter-5 : Future Scopes And Conclusions

5.1 Future Scope

Each and every module of the application has been written in a modular way, which makes it easy to upgrade.

Following upgrades could be made in the future:

- Admin portal can be made by giving admin the rights to to make changes in the database such as adding new buses, etc.
- Timing of buses can also be added.
- Cost of travelling ca also be added.
- Adding a specific stop , that is , a bus should stop at that middle stand, that is prioritizing some bus stops.
- This software can also be used in DTC buses, by integrating this to kiosks for the customers to search the route appropriate to him/her.
- We could use Machine Learning and Artificial Intelligence to predict what route a user wants to go and show that to him as soon as he launches the application.
- We could make it online, by the help of Web development.

5.2 Conclusions

All the objectives were completed in the given/allocated frame of time. It is also concluded that BFS should be used while tackling the shortest path problems. Trie should be used while implementing prefix search.

Tasks Accomplished

- Successfully collected all the data from the web to make a complete database for the buses in Delhi.
- Implemented User-friendly interface, for the application
- Successfully Implemented all the algorithms by modifying the breadth-first search algorithm.

- Successfully decreased the time complexity for the prefix search from O(n*p)-> O(p).

References

[1] Lardinois, Frederic (April 29, 2015). "Microsoft Launches Visual Studio Code, A Free Cross-Platform Code Editor For OS X, Linux And Windows". TechCrunch.
Available: https://en.wikipedia.org/wiki/Visual_Studio_Code
[Accessed: Aug. 31, 2019]

[2] "HTML5 Web Development Support". netbeans.org. Retrieved 2 August 2017.
Available: https://en.wikipedia.org/wiki/NetBeans
[Accessed: Aug. 31, 2019]

[3] "Java SE 7 Features and Enhancements". Oracle Corporation. Retrieved 1 January 2013.
Available: https://en.wikipedia.org/wiki/Java_Development_Kit
[Accessed: Aug. 31, 2019]

[4] Available: https://www.tutorialspoint.com/java/index.htm
[Accessed: Aug. 31, 2019]

[5] Available: https://www.edrawsoft.com/Data-Flow-Diagram-Symbols.php
[Accessed: Aug. 31, 2019]

[6] Available:https://www.geeksforgeeks.org/different-ways-reading-text-file-java/
[Accessed: Aug. 31, 2019]

[7] Available: https://www.geeksforgeeks.org/graph-and-its-representations/
[Accessed: Sep. 01, 2019]

[8] Available:https://www.tutorialspoint.com/data_structures_algorithms/depth_first_traversal.htm
[Accessed: Sep. 01, 2019]

[9] Available:https://www.tutorialspoint.com/data_structures_algorithms/breadth_first_traversal.htm
[Accessed: Sep. 01, 2019]

[10] Available:https://www.hackerearth.com/practice/datastructures/advanced-data-structures/trie-keyword-tree/tutorial/
[Accessed: Sep. 01, 2019]

[11] Available: https://www.geeksforgeeks.org/different-ways-reading-text-file-java/

[Accessed: Sep. 01, 2019]

[12] Available: https://www.geeksforgeeks.org/graph-and-its-representations/

[Accessed: Sep. 01, 2019]

[13] Available:https://www.tutorialspoint.com/data_structures_algorithms/depth_first_traversal.htm

[Accessed: Sep. 01, 2019]

[14] Available:https://www.tutorialspoint.com/data_structures_algorithms/breadth_first_traversal.htm

[Accessed: Sep. 02, 2019]

[15] Available: https://www.hackerearth.com/practice/data-structures/advanced-data-structures/trie-keyword-tree/tutorial/

[Accessed: Sep. 02, 2019]