ID: 12859191938
Name: Arda Tarım
Section: 1
Assignment number: 2

Problem Statement and Code Design

There are two questions in the assigment and I need to use Directed Graph structure to solve them. I used my custom made IterableList class which I created for assigment 1 to implement the Directed Graph. This class is basically a generic LinkedList/Queue that can return an Iterable.

In the first question, I have 5 different files DirectedGraph.java, FileRead.java, Valuefinder.java, HW2_solution.java and the .txt file. In the second question, I have 4 different files, FileRead.java, Valuefinder.java, HW2_Q2_solution.java and .txt file.

Finally, I created the test cases to test the algorithms. The tests will be used to find where algorithms fail.

Implementation and Functionality

HW2_solution:

First I initialized my FileRead, Valuefinder and DirectedGraph. The directed graph is initialized with the data from .txt file. Then I get the startVertex data from the user.

For printing the result, I used an algorithm with a basic nested loop.

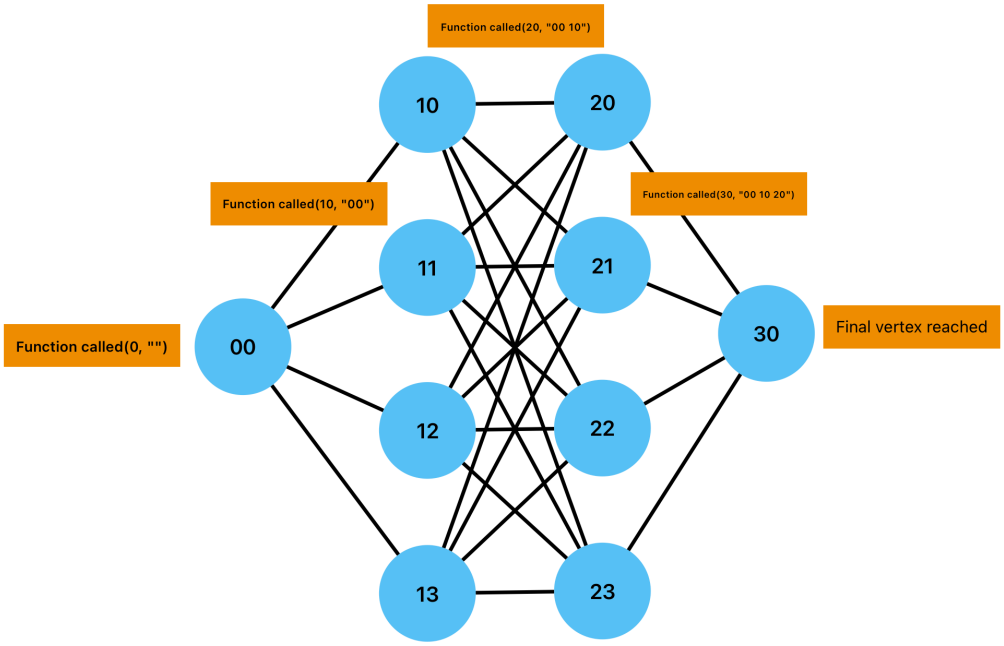
```
// Printing the result
String result;
// For each adjacent vertex from the starting point
for (int adjacent : graph.adj(startVertex)) {

    // Each line begins with startVertex
    result = startVertexStr;

    // Each line has the current adjacent
    result += " " + Integer.toString(adjacent);

    // For each adjacent there is second step
    for (int adjacent2 : graph.adj(adjacent)) {

        String print = result + " " + Integer.toString(adjacent2);
        // Printing the result
        System.out.println(print);
    }
```

For each adjacent vertex from the starting point I add the startVertex and adjacent to “result”. Then, for each adjacent vertex to the adjacent vertex, I add the adjacent 2 to the “result”. I save this information to “print” and result is printed. “result” string’s value is reset in the start of each iteration.

HW_2_Q2_solution:

Firstly, I initialized FileRead and Valuefinder classes and created a NeuralNetwork(Directed Graph) from the Valufinder object. Then I call the recursive findConnections() function from the 00 vertex. Function has 2 parameters vertex(int) and path(string). The results from the recursive function are saved to a String array to get sorted and printed. String array is sorted with Comparator, prioritizing the lenght then the lexicographic order of the items.

The recursive function first checks if it has reached the final vertex. If the reached vertex is final vertex it returns nothing. If the vertex is not the final vertex, vertex data is added to the path and function is called for each adjacent vertex. If the vertex has no adjacent vertices (meaning connection is not established until the final vertex), function adds the current path data to the paths array.



This is an example of how recursive function travels through the network.

DirectedGraph:

Directed Graph is the basic implementation of a directed graph using the Digraph API from the book. Main functions of the class are Constructer(valuefinder), addEge(a,b) and adj(a). Directed Graph can be initialized with a Valuefinder object. addEdge(a,b) method adds an edge from the vertex a to vertex b. Adj(v) method returns an IterableList containing all the vertices directed from the vertex v.

There is also an inner class called IterableList. It is a generic class that can return an Iterable and works as a LinkedList. It includes basic functions like add(), poll(), isEmpty().

FileRead:

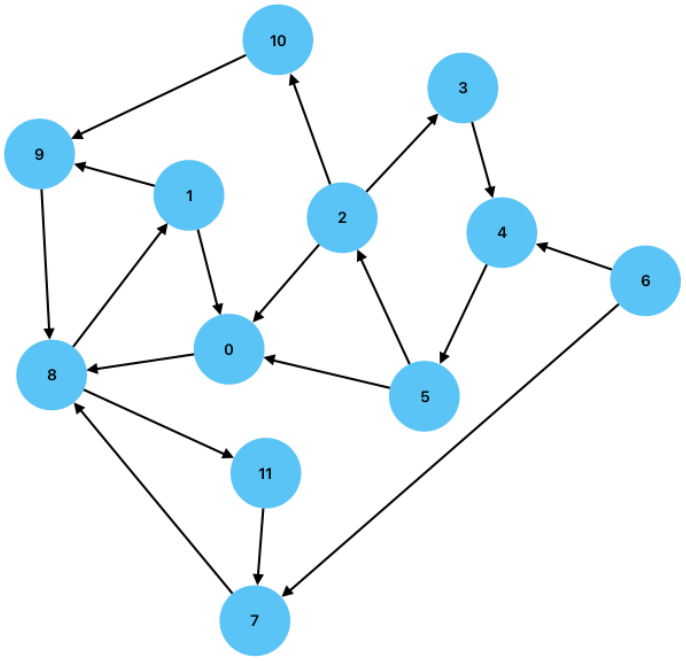
FileRead class is created to read data from .txt files. It is initialized with constructor that takes the “file path” as a parameter. The .txt file is opened and read when the readFile() function is called. With this function, integers are read from the file with a scanner and then added to a queue. The readInt() method returns the next integer from the queue. Because of the FIFO principle of the queue, first scanned integer is returned first from the queue.

Valuefinder:

Valuefinder is a basic wrapper around the FileRead class. It can be initialized with a FileRead object. It has three basic methods to create graphs, getV(), getE(), getNext(). Graph constructor can Access number of vertices and edges with getV() and getE() methods. getNext() method will return the next integer in line.

Testing

Testing for question 1



Start Vertex: 8
8
8 1 9
8 1 0
8 11 7

(Expected output)

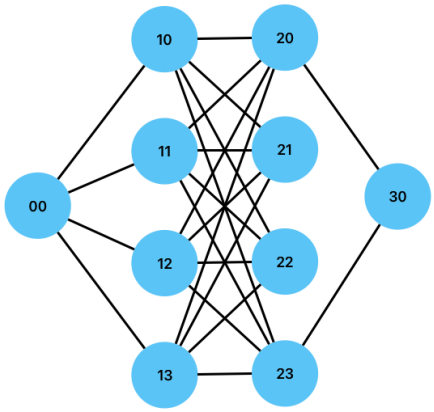
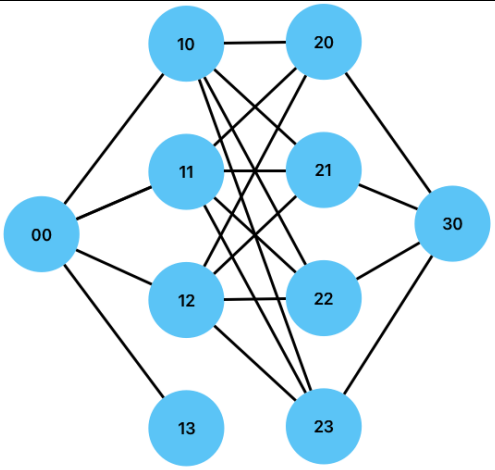
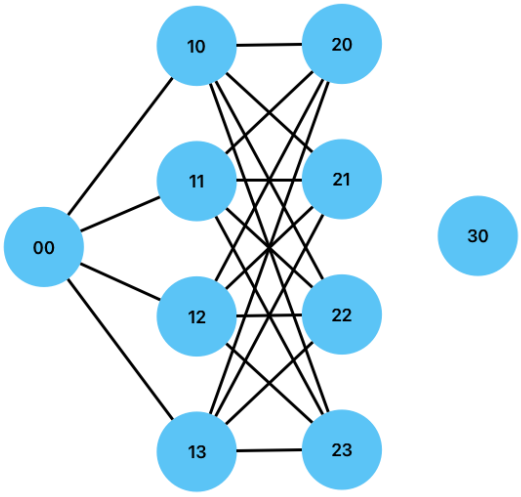
Start Vertex: 5
5
5 0 8
5 2 10
5 2 3
5 2 0

(Expected output)

Start Vertex: 11
11
11
11 7 8

(Expected output)

Testing for question 2

	**Result is** **0 10 21** **0 10 22** **0 11 21** **0 11 22** **0 12 21** **0 12 22** **0 13 21** **0 13 22** **(Expected output)**
	**Result is** **0 13** **(Expected output)**
	**Result is** **0 10 20** **0 10 21** **0 10 22** **0 10 23** **0 11 20** **0 11 21** **0 11 22** **0 11 23** **0 12 20** **0 12 21** **0 12 22** **0 12 23** **0 13 20** **0 13 21** **0 13 22** **0 13 23** **(Expected output)**

Final Assessments

Which parts were the most challenging for you?

In my opinion understanding and solving the first question was easy. I already had an undirected graph for assigment 1 and I quickly created a directed graph with my previous knowledge. Then, I easily created the logic for 2-step algorithm.

The second question was a lot harder, my directed graph didn't work because I used an array for adj. list. But vertices in the question are not ordered from 0 to N. Instead I had to create a new directed graph called Neural Network and used a hashmap to hold adj. list. Also, finding the correct algorithm for finding the question was hard. I tried multiple approaches with arrays, lists and recursion. I know recursion was the correct approach but creating the correct recursion algorithm wasn't easy.

What did you like about the assignment? What did you learn from it?

The assigment helped me to learn directed graph and understand all the basics. I liked the second question because it was hard for me to solve and I think it greatly improved my recursion skills.

References

Algorithms, FOURTH EDITION, Robert Sedgewick and Kevin Wayne

https://www.geeksforgeeks.org/java-string-compareto-method-with-examples/