CS146: Quiz 11
Due Tuesday, April 25, at 7:00AM
10 points

START EARLY!

You will implement some methods in the UndirectedGraph class. The UndirectedGraph class contains two nested classes, Vertex and Node, you should NOT change these classes. The graph will store a list of all the vertices in the graph. Each vertex has a pointer to its adjacent vertices. **To get started, import the starter file, Undirected.java into the graphs package you create in a new Java Project.** Please do not change any of the method signatures in the class, but you can add any helper methods you deem necessary. Implement the methods described below. You are free to test your code however you prefer. These methods should be completed individually using any IDE you are comfortable with. You are free to use the textbook, slides, class notes, and the Java API Documentation, but **DO NOT** consult any other resources.

Vertex Class (DO NOT EDIT)

The vertex class holds information about the vertices in the graph. It has an int val, a Vertex next that refers to the next vertex in the list of vertices (not necessarily an adjacent vertex), and a Node edge that starts the list of adjacent vertices.

Node Class (DO NOT EDIT)

This is a simple class to represent an adjacency list of a vertex in the graph.

UndirectedGraph Class

You will implement the following three methods in the UndirectedGraph class.

```
public boolean addVertex(int value)
```

This method adds a new vertex with val attribute equal to value. It returns true if the vertex is added and false if a vertex with val equal to value already exists. The vertex list (stored in the vertices instance variable) is kept in sorted order based on the value (smallest to largest)

```
public boolean addEdge(int val1, int val2)
```

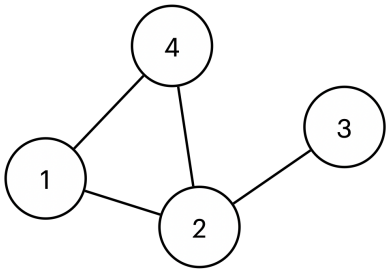
This method adds an undirected edge between the Vertex with val equal to val1 and the Vertex with val equal to val2. You can assume that there exists vertices with vals equal to val1 and val2 (i.e., there are Vertex objects in the vertices list that have vals equal to val1 and val2). The Nodes in the edge list should be stored in ascending order by vert.val. This method returns false if the edge already exists and true if the edge was added. You might want a helper method here that can find a vertex with val v. Hint: that this is an undirected graph so you should create an edge from the Vertex with val1 to the Vertex with val2 and vice versa.

```
public ArrayList<Integer> breadthFirstSearch(int initial)
```

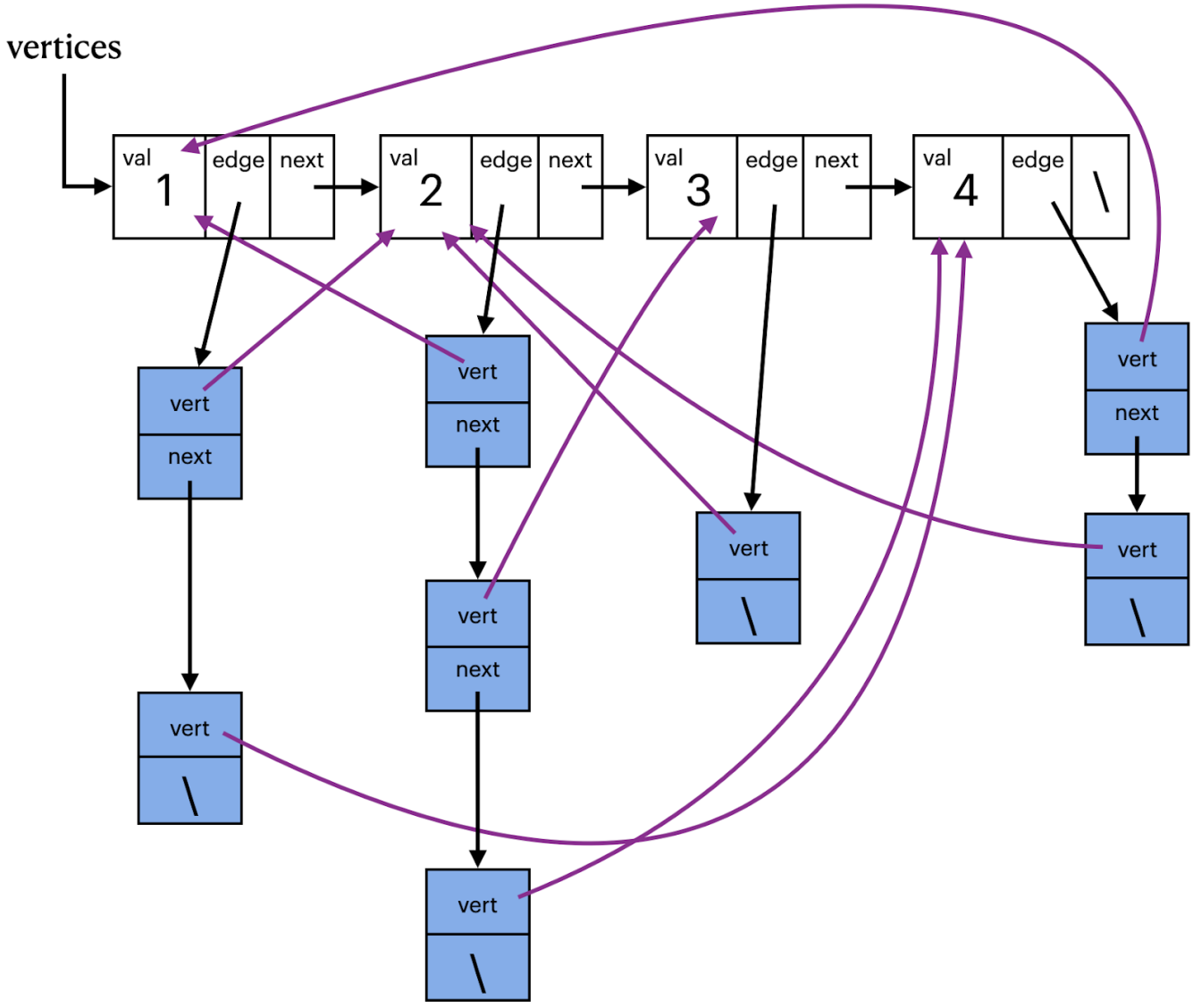
This method should return an ArrayList of integers corresponding to the values associated with the vertices in the graph visited in breadth first order. You can assume that the graph contains a vertex with value initial. Instead of changing the colors of the nodes we will keep track of the values of the nodes we've visited in the visited ArrayList. You must implement this using a queue. For more information about the Queue interface in java see this link. See the textbook (or class slides) for the pseudocode for the breadth first search algorithm.

Example

Suppose I have the below graph.



Represented in the UndirectedGraph class as…



The breadthFirstSearch with argument equal to 3 would result in the following ArrayList: [3,2,1,4]

Submission

Please create a jar or zip file of your project. It should include the java files (UndirectedGraph.java) and submit it on Canvas. If your file includes class files or other extraneous files outside of the package folder and java files, you will receive an automatic one point deduction.