CS361 Algorithm Lab 4

What to do

1. Write a program to implement the following DFA.
   a. Q = {s, q_1, q_2, r_1, r_2}
   b. s is the start state
   c. A = {q_1,r_1} are the accept states
   d. Σ = {a,b}
   e. σ is defined by the following table:

	a	b
s	q_1	r_1
q_1	q_1	q_2
q_2	q_1	q_2
r_1	r_2	r_1
r_2	r_2	r_1

Show me the output (accepting the string or not) for the following strings:

I keep getting a java.lang.NullPointerException at DFA.transition(DFA.java:86) at DFA.main(DFA.java:125) **And I don't know why, but I am getting a result for 1 3 and 5… Guess it only does the odds…**

1. Ababa
   1. Accepted string: ababa
2. baba // This should be not accepted.
3. aababaab
   3. Unaccepted string: aababaab
4. babaabaaabb // This should be accepted.
5. ε (the empty string)
   5. Unaccepted string: e

```
1  import java.util.*;
2
3  /**
4   * @author Nathan Stark
5   *
6   */
7  public class DFA {
8
9      //this is to define the alphabet.
10     private String  alph;
11     //this field is to hold our final states.
12     private ArrayList<State> stateF;
13     //This field represents our Q set of states.
14     private ArrayList<State> states;
15     //This is the input to be tested.
16     private char[] input;
17     //The start state.
18     private State start;
19
20     /**
21      * Constructor for the deterministic finite automata
22      *
23      * @param alphabet sets the alphabet of this machine.
24      * enter only the characters in the alphabet you intend
25      * to use.
26      * @param start The start state.
27      */
28     public DFA(String alphabet, State start){
29         this.start=start;
30         alph = alphabet;
31         states = new ArrayList<State>();
32         stateF = new ArrayList<State>();
33
34     }
```

```
32          stateF = new ArrayList<State>();
33
34      }
35
36      /**
37       * Enter in states that you want example: "q1", "q2" and so on,
38       * as individual strings.
39       * @param newState The states that you want to have in
40       * the machine.
41       */
42      public void addFinal(State newState){
43          stateF.add(newState);
44      }
45
46      /**
47       * Enter in states that will be part of the machine
48       * @param newState The states that will make up the
49       * machine.
50       */
51      public void addState(State newState){
52          states.add(newState);
53      }
54
55      /**
56       * This is to simulate the transition function.
57       */
58      public boolean transition(String put){
59          char c;
60          int i;
61          boolean acceptance = false;
62          for (i = 0; i < alph.length(); i++) {
63              c = alph.charAt(i);
```

```
62          for (i = 0; i < alph.length(); i++) {
63              c = alph.charAt(i);
64              if(!acceptance){
65                  for(int t=0;t<put.length();t++){
66                      if ((put.charAt(t) != c)&&(!acceptance)){
67                          acceptance = false;
68                      }else{
69                          acceptance = true;
70                      }
71                  }
72              }
73          }
74          if(!acceptance){
75              return acceptance;
76          }
77          State stat = start;
78
79          acceptance=false;
80          input=put.toCharArray();
81          for(i=0;i<input.length;i++){
82              String g = String.valueOf(input[i]);
83              //if(!stat.passChar(g).equals(null)){
84                  //return false;
85              //}
86              stat= stat.passChar(g);
87          }
88          if(stateF.contains(stat)){
89              acceptance=true;
90          }
91          return acceptance;
92      }
93
```

```java
        return acceptance;
    }

    /**
     * @param args
     */
    public static void main(String[] args) {
        State s = new State();
        State q1 = new State();
        State q2 = new State();
        State r1 = new State();
        State r2 = new State();

        s.addNextState("a", q1);
        s.addNextState("b", r1);
        q1.addNextState("a", q1);
        q1.addNextState("b", q2);
        q2.addNextState("a", q1);
        q2.addNextState("b", q2);
        r1.addNextState("a", r2);
        r1.addNextState("b", r1);
        r2.addNextState("a", r2);
        r1.addNextState("b", r1);

        DFA myDFA = new DFA("ba",s);
        //my final state.
        myDFA.addFinal(q1);
        myDFA.addFinal(r1);

        if(myDFA.transition("ababa")){
            System.out.println("1. Accepted string: ababa");
        }else{
            System.out.println("1. Unaccepted string: ababa");
        }
        if(myDFA.transition("baba")){
```

```java
            System.out.println("1. Unaccepted string: ababa");
        }
        if(myDFA.transition("baba")){
            System.out.println("2. Accepted string: baba");
        }else{
            System.out.println("2. Unaccepted string: baba");
        }
        if(myDFA.transition("aababaab")){
            System.out.println("3. Accepted string: aababaab");
        }else{
            System.out.println("3. Unaccepted string: aababaab");
        }
        if(myDFA.transition("babaabaaabb")){
            System.out.println("4. Accepted string: babaabaaabb");
        }else{
            System.out.println("4. Unaccepted string: babaabaaabb");
        }
        if(myDFA.transition("")){
            System.out.println("5. Accepted string: e");
        }else{
            System.out.println("5. Unaccepted string: e");
        }
    }
```

CS361Labs.java | Lab4.java | DFA.java | State.java

```java
import java.util.HashMap;
import java.util.Map;



/**
 * @author Nathan Stark
 *
 */
public class State {
    //This is a map for the next states given the next char passed in.
    Map<String,State> nextStates;

    /**
     * Constructor that initiates the nextStates HashMap field.
     */
    public State(){
        nextStates = new HashMap<String,State>();
        }
    /**
     * Do the this that DFA's do the fine what state they should be at next.
     *
     * @param putIn The char that is passed at the currant state.
     * @return The next state that is a determined from the char passed in.
     */
    public State passChar(String putIn){
        if(!nextStates.containsKey(putIn)){
            return null;
            }
        return nextStates.get(putIn);
        }

    /**
     * Add a state that this state connects to.
     * @param sta the state that this state connects to when
```

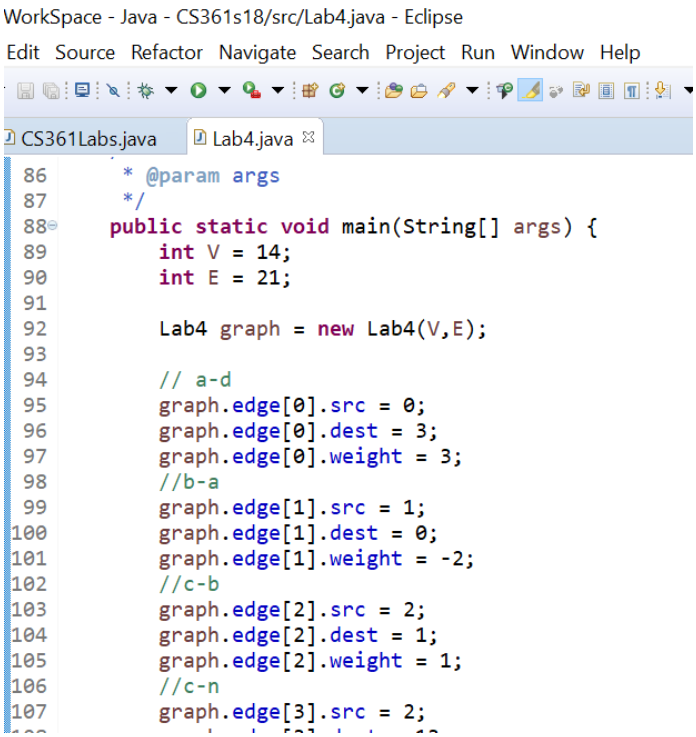
```java
    /**
     * Add a state that this state connects to.
     * @param sta the state that this state connects to when
     * a letter in the alphabet is passed to it.
     * @param pass the letter that is passed to it.
     */
    public void addNextState(String pass,State state){
        nextStates.put(pass,state);
        }
```

2. Implement the Bellman-Ford algorithm. Show commented code.

```
 * @author Nathan Stark
 *
 * The following code was provided by Aakash Hasija.
 * From https://www.geeksforgeeks.org/dynamic-programming-set-23-bellman-ford-algorithm/
 *
 */
public class Lab4 {

    // A class to represent a weighted edge in graph
    class Edge {
        int src, dest, weight;      // Where its coming from, where its going, and how heavy. Respectively.
        Edge() {
            src = dest = weight = 0;
        }
    };

    //fields of Lab 4 in order to implement the Bellman-Ford Algorithm.
    int V, E;
    Edge edge[];

    /**
     * The constructed class, that initializes the values of how many edges
     * and how many Vertices are included on the graph.
     *
     * @param v the amount of vertices that are included with the graph.
     * @param e the amount of edges connecting the vertices.
     */
    Lab4(int v, int e){
        V=v;
        E = e;
        edge = new Edge[e];
        for(int i=0; i<e; ++i){
            edge[i]= new Edge();
        }
    }

    public void bellmanFord(Lab4 graph, int src){
        int i, j;                                   // index variables.
        int V = graph.V, E = graph.E;
        int dist[] = new int[V];                    // Distance array set to the length of how many vertices are on the graph.

        for(i=0;i<V;++i){                           // Init. the distances to unreachable values.
            dist[i] = Integer.MAX_VALUE;            // Note here that we are using the max possible value for integers instead of infinity, b/c infinity
        }
        dist[src] = 0;                              // Starting distance will always be 0.

        for(i=0;i<V;++i){
            for(j=0;j<E;++j){
                int u = graph.edge[j].src;          // Set u to be the vertex that we are coming from.
                int v = graph.edge[j].dest;         // Set v to be the vertex that we are going to.
                int weight = graph.edge[j].weight;  // set the weight to be the weight between u and v.
                if(dist[u]!=Integer.MAX_VALUE && dist[v]+weight<dist[v]){
                    dist[v]=dist[u]+weight;         // set the new weigh if the it passes the check above.
                }
            }
        }

        for(j=0;j<E;++j){
            int u = graph.edge[j].src;              // Set u to be the vertex that we are coming from.
            int v = graph.edge[j].dest;             // Set v to be the vertex that we are going to.
            int weight = graph.edge[j].weight;      // set the weight to be the weight between u and v.
            if (dist[u] != Integer.MAX_VALUE && dist[u]+weight < dist[v]){
                System.out.println("Graph contains negative weight cycle");//detect if there is a negative weight on the graph.
            }
```

```
60          for(j=0;j<E;++j){
61              int u = graph.edge[j].src;                  // Set u to be the vertex that we are coming from.
62              int v = graph.edge[j].dest;                 // Set v to be the vertex that we are going to.
63              int weight = graph.edge[j].weight;          // set the weight to be the weight between u and v.
64              if (dist[u] != Integer.MAX_VALUE && dist[u]+weight < dist[v]){
65                  System.out.println("Graph contains negative weight cycle");//detect if there is a negative weight on the graph.
66              }
67          }
68          printArray(dist,V);                             // call the utility function to show the results.
69      }
70
71      /**
72       * Utility method used to print the results.
73       *
74       * @param dist the distance array.
75       * @param V The Vertex.
76       */
77      private void printArray(int dist[], int V){
78          System.out.println("Vertex   Distance from Source");
79          for(int i=0; i<V;++i){
80              System.out.println(i+"\t\t"+dist[i]);
81          }
82      }
83
84
85      /**
86       * @param args
87       */
88      public static void main(String[] args) {
89          int V = 14;
90          int E = 21;
91
92          Lab4 graph = new Lab4(V,E);
93
94          // a-d
```

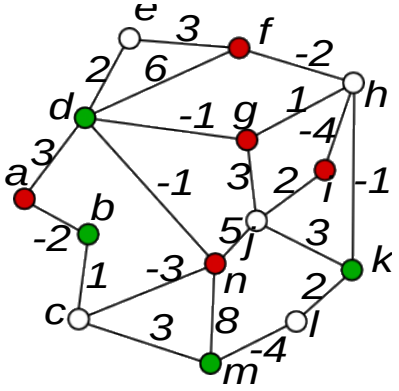
I'm going to omit most of the main method but there should be enough to understand the basic jest of what is happening.



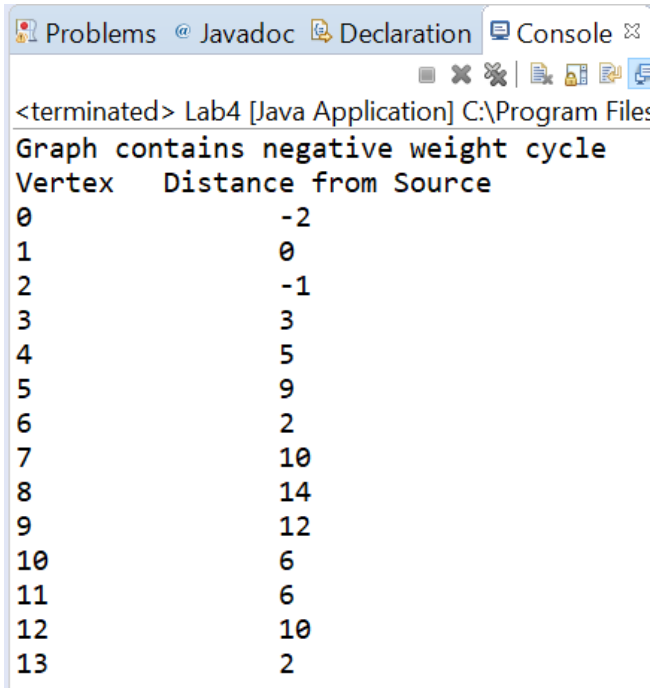

```
86       * @param args
87       */
88      public static void main(String[] args) {
89          int V = 14;
90          int E = 21;
91
92          Lab4 graph = new Lab4(V,E);
93
94          // a-d
95          graph.edge[0].src = 0;
96          graph.edge[0].dest = 3;
97          graph.edge[0].weight = 3;
98          //b-a
99          graph.edge[1].src = 1;
100         graph.edge[1].dest = 0;
101         graph.edge[1].weight = -2;
102         //c-b
103         graph.edge[2].src = 2;
104         graph.edge[2].dest = 1;
105         graph.edge[2].weight = 1;
106         //c-n
107         graph.edge[3].src = 2;
```

```
160         graph.edge[16].dest = 10;
161         graph.edge[16].weight = 3;
162         //l-k
163         graph.edge[17].src = 11;
164         graph.edge[17].dest = 10;
165         graph.edge[17].weight = 2;
166         //m-l
167         graph.edge[18].src = 12;
168         graph.edge[18].dest = 11;
169         graph.edge[18].weight = -4;
170         //n-m
171         graph.edge[19].src = 13;
172         graph.edge[19].dest = 12;
173         graph.edge[19].weight = 8;
174         //n-c
175         graph.edge[20].src = 13;
176         graph.edge[20].dest = 2;
177         graph.edge[20].weight = -3;
178
179         graph.bellmanFord(graph, 0);
180     }
181
182 }
```

3. Show the output (including all of the distances and predecessors) for the Bellman-Ford algorithm on the graph below.



Note that I have started with the letter a and replaced it with the number 0 and have do so with all of the respective letters as well (1 for b 2 for c and so on up to n).


Problems
Javadoc
Declaration
Console
<terminated> Lab4 [Java Application] C:\Program Files
Graph contains negative weight cycle
Vertex Distance from Source
0 -2
1 0
2 -1
3 3
4 5
5 9
6 2
7 10
8 14
9 12
10 6
11 6
12 10
13 2