Tallinna Tehnikaülikool

Individuaaltöö aines "Algoritmid ja andmestruktuurid"

Hanna Kristin Ojaveer

Rühm: IADB33

Juhendaja : Jaanus Pöial

Tallinn 2020

Sisukord

Ülesande püstitus

Ülesandeks on koostada meetod, mis etteantud Euleri graafi servad nummerdaks vastavalt servade järjekorrale Euleri ahela läbimisel.

Lahenduse kirjeldus

Euleri graafiks nimetatakse sidusat suunamata graafi, kui selles leidub kinnine ahel, mis sisaldab iga serva ühe korra. **Euleri ahel on** vastav ahel.

> Kui sidusa graafi servade hulk esitub paarikaupa lõikumatute tsüklite ühendina, siis on tegu Euleri graafiga. Pool-Euleri graafiks saab nimetada sidusat graafi, kui selles on täpselt kaks tippu, mille aste on paarituarvuline.

Servade nummerdamiseks on loodud meetod *enumerate_graphs_edges*, aga enne nummerdama hakkamist tuleb külgnevusstruktuuri abil esitatud graaf üle kontrollida.

Graafi kontrollimiseks kasutatakse meetodit *isEulerGraph*, mis tagastab boolean väärtuse. Meetodis *isEulerGraph* luuakse *verticel_list,* et sealt kokku loendada tipud, millel on paaritu arv kaari. Kui kaari on rohkem kui kaks, siis *isEulerGraph* meetod tagastab väärtuse false, mis näitab et tegu ei ole Euleri graafiga. Sellisel juhul antakse kasutajale veateade.

Pärast kontrollimist ja kinnitamist, et graafi puhul on vajalikud tingimused täidetud, alustatakse selle graafi servade nummerdamist. Kuna Euleri graafi serva on lubatud läbida ainult üks kord, siis selle jaoks on lisatud Arc klassi muutuja visited, mille järgi kontrollitakse, kas serva on juba varasemalt läbitud (boolean visited = false). Kui visited = true, ehk seda kaart on juba varasemalt läbitud, siis see asendatakse kaarega, mis väljub samast tipust, aga mida pole veel külastatud. Nii läbitakse ja nummerdatakse kõik servad.

Programmi kasutusjuhend

Programmi kasutamine on väga lihtne, kui Graph tüüpi objekt on juba olemas. Siis on vaja vaid kutsuda välja klassist *GraphTask* meetod *enumerate_graphs_edges*, mis nummerdab graafi servad. Kui graafi pole, siis tuleks see eelnevalt ise luua run() meetodis. Selleks on eraldi juhend.

Ise graafi loomine:

1) ```
   Graph new_graph = new Graph( s: "new graph");
   ```
   loome uue graafi, sulgudesse kirjutada uue graafi nimetus.

2) ```
   Vertex a = new Vertex( s: "A");
   ```
   Tipu A loomine. Tegevust tuleb korrata kuni on loodud soovitud hulk tippe. Tippudele tuleks määrata erinevad nimed.

3) ```
   new_graph.first = a;
   ```
   Graafi läbimise alguspunkti määramine.

4) ```
   a.next = b;
   ```
   Määrame järjestatud tipud.

5) ```
   Arc ab = new Arc( s: "AB");
   ```
   Serva AB loomine. Tegevust tuleb korrata kuni on loodud vajalik hulk servi.

6) ```
   a.first = ab;
   ```
   Määrame seose tipu ja esimese läbitava serva vahel. Tegevust tuleb korrata iga tipuga.

7) ```
   ab.next = ac;
   ```
   Määrame seose tipu ja selle teiste servadega. Tegevust tuleb korrata iga tipu ja servaga, millele pole veel seost määratud.

8) ```
   ab.target = b;
   ```
   Määrame seose serva ja lõputipuga. Tegevust tuleb korrata kõigi servade ja tippudega.

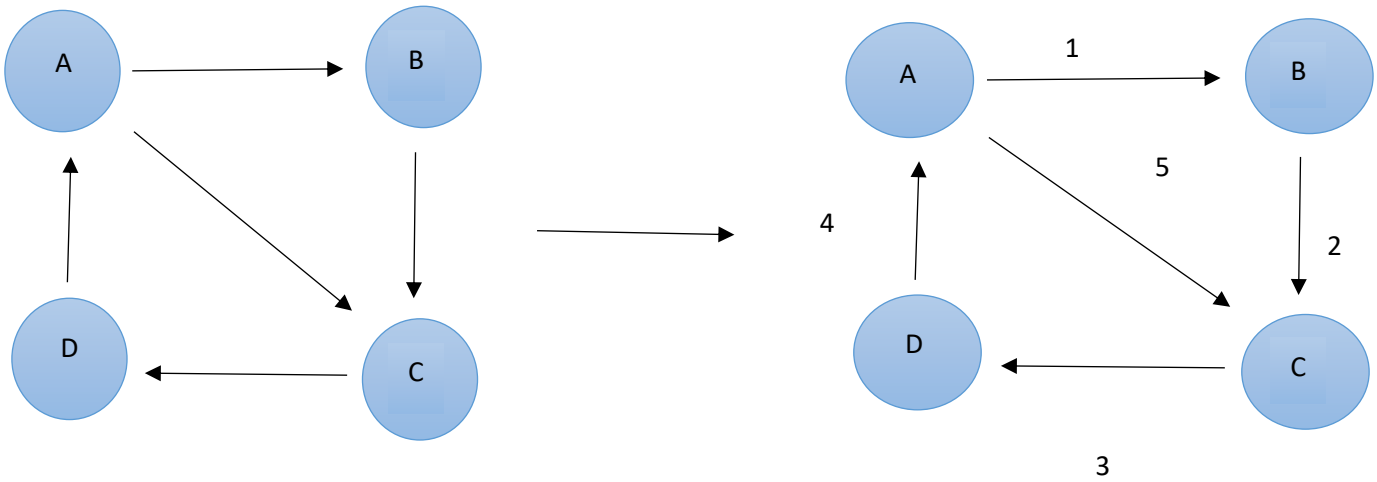
9) ```
   enumerate_graphs_edges(new_graph);
   ```
   Käivitame programmi, kus kontrollime, kas tegu on Euleri graafiga ja nummerdame selle servad vastavalt nende järjekorrale Euleri ahela läbimisel.
   ```

Testimiskava

Test 1

Testitakse 4 punktilise Euleri graafi nummerdamist, servade läbimine on näidatud vasakpoolsel joonisel ja oodatava tulemuse illustratsioon on paremal.

Test 1 joonis



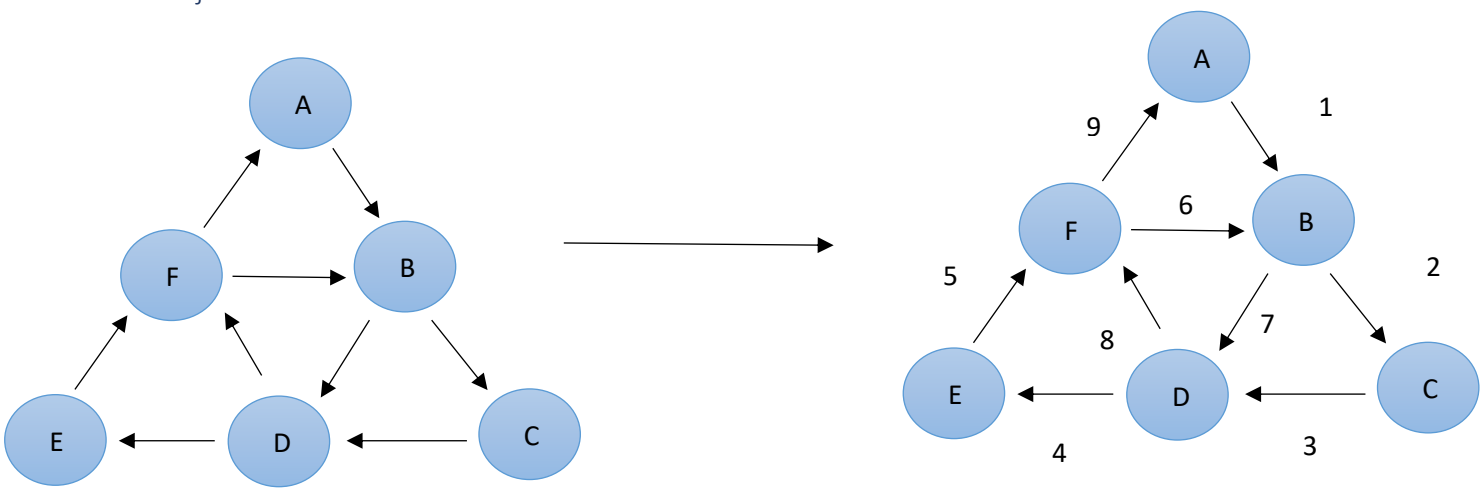
Test 1 tulemus

```
Test 1
A --> AB (A -- 1 --> B) AC (A -- 5 --> C)
B --> BC (B -- 2 --> C)
C --> CD (C -- 3 --> D)
D --> DA (D -- 4 --> A)
```

Test 2

Testitakse 6 punktilise Euleri graafi nummerdamist, servade läbimine on näidatud vasakpoolsel joonisel ja oodatava tulemuse illustratsioon on paremal.

Test 2 joonis



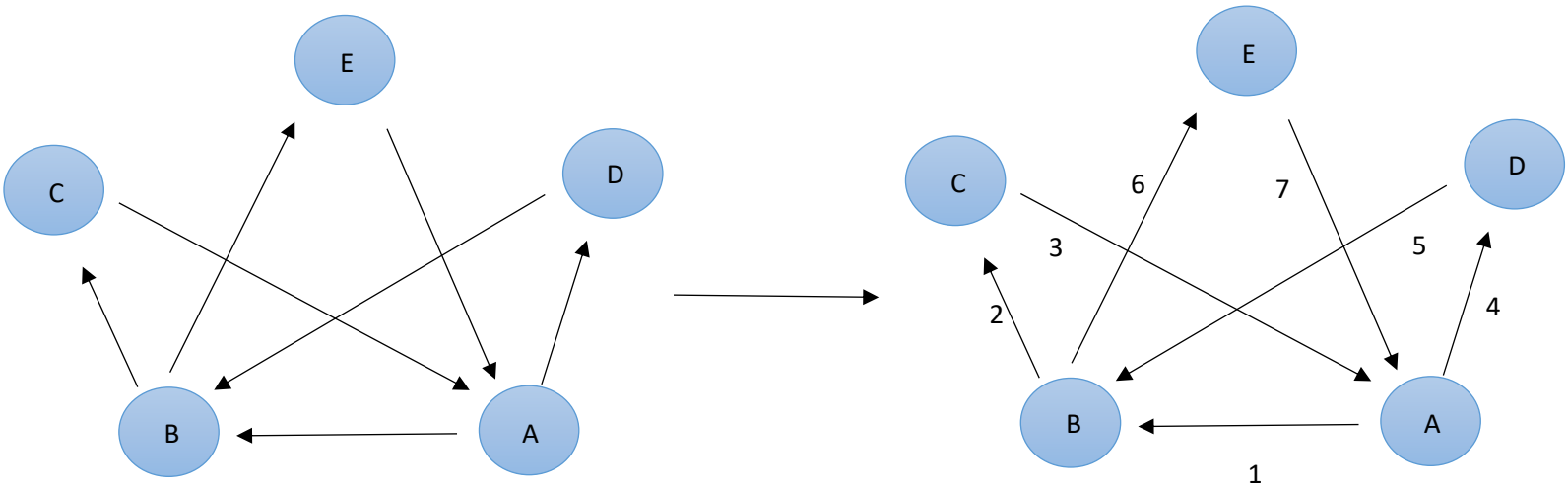
Test 2 tulemus

```
Test 2
A --> AB (A -- 1 --> B)
B --> BC (B -- 2 --> C) BD (B -- 7 --> D)
C --> CD (C -- 3 --> D)
D --> DE (D -- 4 --> E) DF (D -- 8 --> F)
E --> EF (E -- 5 --> F)
F --> FB (F -- 6 --> B) FA (F -- 9 --> A)
```

Test 3

Testitakse 5 punktilise Euleri graafi nummerdamist, servade läbimine on näidatud vasakpoolsel joonisel ja oodatava tulemuse illustratsioon on paremal.

Test 3 joonis



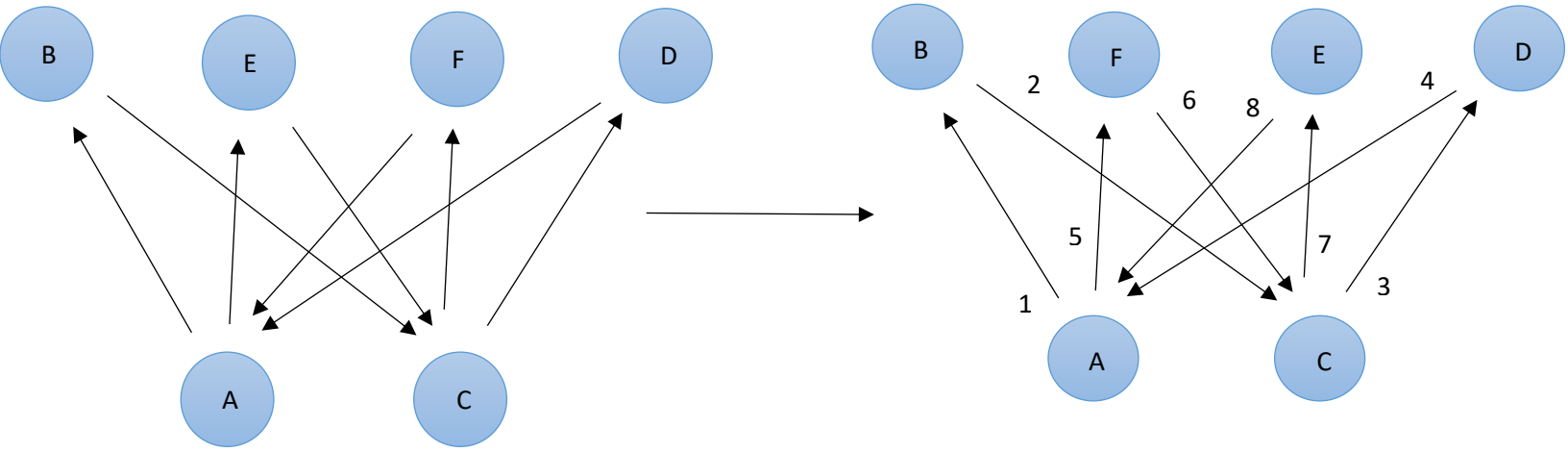
Test 3 tulemus

```
Test 3
A --> AB (A -- 1 --> B) AD (A -- 4 --> D)
B --> BC (B -- 2 --> C) BE (B -- 6 --> E)
C --> CA (C -- 3 --> A)
D --> DB (D -- 5 --> B)
E --> EA (E -- 7 --> A)
```

Test 4

Testitakse 6 punktilise Euleri graafi nummerdamist, servade läbimine on näidatud vasakul joonisel ja oodatava tulemuse illustratsioon on paremal.

Test 4 joonis



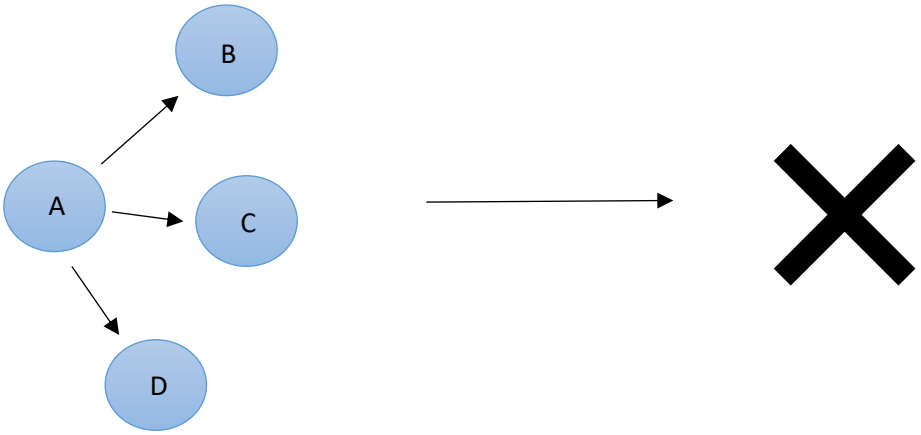
Test 4 tulemus

```
Test 4
A --> AB (A -- 1 --> B) AE (A -- 5 --> E)
B --> BC (B -- 2 --> C)
C --> CD (C -- 3 --> D) CF (C -- 7 --> F)
D --> DA (D -- 4 --> A)
E --> EC (E -- 6 --> C)
F --> FA (F -- 8 --> A)
```

Test 5

Testitakse, kas kood tunneb ära, et tegu ei ole Euleri graafiga

Test 5 joonis



Test 5 tulemus

```
java.lang.RuntimeException: Graph 5 isn't Euler graph!!!

    at GraphTask.enumerate_graphs_edges(GraphTask.java:44)
    at GraphTaskTest.test5(GraphTaskTest.java:206) <10 internal calls>
    at java.base/java.util.concurrent.FutureTask.run(FutureTask.java:264)
    at java.base/java.lang.Thread.run(Thread.java:834)
```

Test 5 tulemuseks pidigi kood viskama vea, sest tegu ei ole Euleri graafiga.

Kasutatud kirjandus:

Teooria: https://research.cyber.ee/~peeter/teaching/graafid02s/loeng2.pdf

https://research.cyber.ee/~peeter/teaching/graafid03s/graafid.pdf

https://www3.cs.stonybrook.edu/~pfodor/courses/CSE260/L28_Graphs.pdf

Euleri graafid: https://mathworld.wolfram.com/EulerianGraph.html

Lisad:

GraphTask.java

```java
import java.util.ArrayList;
/** Container class to different classes, that makes the whole
 * set of classes one class formally.
 */
public class GraphTask {

   /** Main method. */
   public static void main (String[] args) {
      GraphTask a = new GraphTask();
      a.run();
      }

   /** Actual main method to run examples and everything. */
   public void run() {

      Graph graph = new Graph( s: "2 point");
      Vertex a = new Vertex( s: "A");
      Vertex b = new Vertex( s: "B");
      graph.first = a;
      Arc ab = new Arc( s: "AB");
      a.first = ab;
      ab.target = b;

      enumerate_graphs_edges(graph);
      System.out.println(graph);
   }
```

```java

   /** Nummerdab graafi servad järjekorrale vastalt*/
   public void enumerate_graphs_edges(Graph graph) {
      boolean euler_graph = isEulerGraph(graph);
      if (!euler_graph) {
         throw new RuntimeException( graph.id + " isn't Euler graph!!! Insert new graph or change existing one");
      }
      Vertex vertex = graph.first;
      int count = 1;
      for (int i = 0; i < graph.count_arcs(); i++) {
         if (!vertex.first.visited) {
            vertex.first.nr = count;
            vertex.first.visited = true;
            vertex = vertex.first.target;
         } else {
            Arc arc = vertex.first.next;
            while (arc.visited) {
               arc = arc.next;
            }
            arc.nr = count;
            vertex = arc.target;
         }
         count += 1;
      }
   }
```

```java

    /** Kontrollin, kas tegemist on Euleri graafiga*/
    private boolean isEulerGraph(Graph graph) {
        ArrayList<Vertex> vertices_list = graph.vertices_list();
        int counter = 0;
        for (Vertex vertex : vertices_list) {
            if (vertex.counter % 2 != 0) {
                counter += 1;
            }
        }
        return counter <= 2;
    }

    /** Kujutab tippu graafis*/
    public static class Vertex {

        private String id;
        public Vertex next;
        public Arc first;
        private int counter = 0;


        Vertex (String s, Vertex v, Arc e) {
            id = s;
            next = v;
            first = e;
        }

        public Vertex(String s) { this (s,  v: null,  e: null); }

        @Override
        public String toString() { return id; }

    }


```

```java
/** Arc represents one arrow in the graph. Two-directional edges are
 * represented by two Arc objects (for both directions).
 */
public class Arc {

   private String id;
   public Vertex target;
   public Arc next;
   public int nr = 0;
   private boolean visited; // kontrollimiseks, kas kaart on juba läbitud

   Arc (String s, Vertex v, Arc a) {
      id = s;
      target = v;
      next = a; }

   public Arc(String s) { this (s, v: null, a: null); }

   @Override
   public String toString() { return id; }

   }

   /** Kujutab graafi*/
public class Graph {

   private String id;
   public Vertex first;

   Graph (String s, Vertex v) {
      id = s;
      first = v;
   }

   public Graph(String s) { this (s, v: null); }

```

```java
    /** Tagastab tippude listi */
    public ArrayList<Vertex> vertices_list() {
        ArrayList<Vertex> vertices_list = new ArrayList<>();
        Vertex vertex = this.first;
        while (vertex != null) {
            vertices_list.add(vertex);
            Arc arc = vertex.first;
            while (arc != null) {
                arc.target.counter += 1;
                vertex.counter += 1;
                arc = arc.next;
            }
            vertex = vertex.next;
        }
        return vertices_list;
    }

    /** Loendab kaared kokku, et teada saada, mitu ühendust punktide vahel on*/
    public int count_arcs() {
        int counter = 0;
        Vertex vertex = this.first;
        while ( vertex != null) {
            Arc arc = vertex.first;
            while ( arc != null){
                counter += 1;
                arc = arc.next;
            }
            vertex = vertex.next;
        }
        return counter;
    }

```

```
166         @Override
167         public String toString() {
168             String nl = System.getProperty ("line.separator");
169             StringBuffer sb = new StringBuffer (nl);
170             sb.append (id);
171             sb.append (nl);
172             Vertex v = first;
173             while (v != null) {
174                 sb.append (v.toString());
175                 sb.append (" -->");
176                 Arc a = v.first;
177                 while (a != null) {
178                     sb.append (" ");
179                     sb.append (a.toString());
180                     sb.append (" (");
181                     sb.append (v.toString());
182                     sb.append(" -- ").append(a.nr).append(" --> ");
183                     sb.append (a.target.toString());
184                     sb.append (")");
185                     a = a.next;
186                 }
187                 sb.append (nl);
188                 v = v.next;
189             }
190             return sb.toString();
191         }
192
193     }
194
195 }
196
197
```

GraphTaskTest.java

```java
import static org.junit.Assert.*;
import org.junit.Test;

public class GraphTaskTest {

   private GraphTask.Vertex vertexA = new GraphTask.Vertex( s: "A");
   private GraphTask.Vertex vertexB = new GraphTask.Vertex( s: "B");
   private GraphTask.Vertex vertexC = new GraphTask.Vertex( s: "C");
   private GraphTask.Vertex vertexD = new GraphTask.Vertex( s: "D");
   private GraphTask.Vertex vertexE = new GraphTask.Vertex( s: "E");
   private GraphTask.Vertex vertexF = new GraphTask.Vertex( s: "F");


   @Test (timeout=20000)
   public void tester1() {
      GraphTask graph = new GraphTask();
      GraphTask.Graph graphTest = graph.new Graph( s: "Test 1", vertexA);
      GraphTask.Arc ArcBC = graph.new Arc( s: "BC", vertexC,  a: null);
      GraphTask.Arc ArcCD = graph.new Arc( s: "CD", vertexD,  a: null);
      GraphTask.Arc ArcDA  = graph.new Arc( s: "DA", vertexA,  a: null);
      GraphTask.Arc ArcAC = graph.new Arc( s: "AC", vertexC,  a: null);
      GraphTask.Arc ArcAB = graph.new Arc( s: "AB", vertexB, ArcAC);
      vertexA.next = vertexB;
      vertexB.next = vertexC;
      vertexC.next = vertexD;
      vertexA.first = ArcAB;
      vertexB.first = ArcBC;
      vertexC.first = ArcCD;
      vertexD.first = ArcDA;
      graph.enumerate_graphs_edges(graphTest);
      assertEquals( message: "Arc " + ArcAB + " number should be 1",  expected: 1, ArcAB.nr);
      assertEquals( message: "Arc " + ArcBC + " number should be 2",  expected: 2, ArcBC.nr);
      assertEquals( message: "Arc " + ArcCD + " number should be 3",  expected: 3, ArcCD.nr);
      assertEquals( message: "Arc " + ArcDA + " number should be 4",  expected: 4, ArcDA.nr);
      assertEquals( message: "Arc " + ArcAC + " number should be 5",  expected: 5, ArcAC.nr);
      System.out.println(graphTest);
   }
```

```java
@Test (timeout=20000)
public void tester2() {
   GraphTask graphTask = new GraphTask();
   GraphTask.Graph testGraph = graphTask.new Graph( s: "Test 2", vertexA);
   GraphTask.Arc ArcAB = graphTask.new Arc( s: "AB", vertexB,  a: null);
   GraphTask.Arc ArcBC = graphTask.new Arc( s: "BC", vertexC,  a: null);
   GraphTask.Arc ArcCD = graphTask.new Arc( s: "CD", vertexD,  a: null);
   GraphTask.Arc ArcDE = graphTask.new Arc( s: "DE", vertexE,  a: null);
   GraphTask.Arc ArcEF = graphTask.new Arc( s: "EF", vertexF,  a: null);
   GraphTask.Arc ArcFB = graphTask.new Arc( s: "FB", vertexB,  a: null);
   GraphTask.Arc ArcBD = graphTask.new Arc( s: "BD", vertexD,  a: null);
   GraphTask.Arc ArcDF = graphTask.new Arc( s: "DF", vertexF,  a: null);
   GraphTask.Arc ArcFA = graphTask.new Arc( s: "FA", vertexA,  a: null);
   vertexA.next = vertexB;
   vertexB.next = vertexC;
   vertexC.next = vertexD;
   vertexD.next = vertexE;
   vertexE.next = vertexF;
   vertexA.first = ArcAB;
   vertexB.first = ArcBC;
   vertexC.first = ArcCD;
   vertexD.first = ArcDE;
   vertexE.first = ArcEF;
   vertexF.first = ArcFB;
   ArcBC.next = ArcBD;
   ArcDE.next = ArcDF;
   ArcFB.next = ArcFA;
   graphTask.enumerate_graphs_edges(testGraph);
   assertEquals( message: "Arc " + ArcAB + " number should be 1",  expected: 1, ArcAB.nr);
   assertEquals( message: "Arc " + ArcBC + " number should be 2",  expected: 2, ArcBC.nr);
   assertEquals( message: "Arc " + ArcCD + " number should be 3",  expected: 3, ArcCD.nr);
   assertEquals( message: "Arc " + ArcDE + " number should be 4",  expected: 4, ArcDE.nr);
   assertEquals( message: "Arc " + ArcEF + " number should be 5",  expected: 5, ArcEF.nr);
   assertEquals( message: "Arc " + ArcFB + " number should be 6",  expected: 6, ArcFB.nr);
   assertEquals( message: "Arc " + ArcBD + " number should be 7",  expected: 7, ArcBD.nr);
   assertEquals( message: "Arc " + ArcDF + " number should be 8",  expected: 8, ArcDF.nr);
   assertEquals( message: "Arc " + ArcFA + " number should be 9",  expected: 9, ArcFA.nr);
   System.out.println(testGraph);
}
```

```
80      @Test (timeout=20000)
81      public void tester3() {
82         GraphTask graphTask = new GraphTask();
83         GraphTask.Graph testGraph = graphTask.new Graph( s: "Test 3", vertexA);
84         GraphTask.Arc ArcAB = graphTask.new Arc( s: "AB", vertexB,  a: null);
85         GraphTask.Arc ArcBC = graphTask.new Arc( s: "BC", vertexC,  a: null);
86         GraphTask.Arc ArcCA = graphTask.new Arc( s: "CA", vertexA,  a: null);
87         GraphTask.Arc ArcAD = graphTask.new Arc( s: "AD", vertexD,  a: null);
88         GraphTask.Arc ArcDB = graphTask.new Arc( s: "DB", vertexB,  a: null);
89         GraphTask.Arc ArcBE = graphTask.new Arc( s: "BE", vertexE,  a: null);
90         GraphTask.Arc ArcEA = graphTask.new Arc( s: "EA", vertexA,  a: null);
91         vertexA.next = vertexB;
92         vertexB.next = vertexC;
93         vertexC.next = vertexD;
94         vertexD.next = vertexE;
95         ArcAB.next = ArcAD;
96         ArcBC.next = ArcBE;
97         vertexA.first = ArcAB;
98         vertexB.first = ArcBC;
99         vertexC.first = ArcCA;
100        vertexD.first = ArcDB;
101        vertexE.first = ArcEA;
102        graphTask.enumerate_graphs_edges(testGraph);
103        assertEquals( message: "Arc " + ArcAB + " number should be 1",  expected: 1, ArcAB.nr);
104        assertEquals( message: "Arc " + ArcBC + " number should be 2",  expected: 2, ArcBC.nr);
105        assertEquals( message: "Arc " + ArcCA + " number should be 3",  expected: 3, ArcCA.nr);
106        assertEquals( message: "Arc " + ArcAD + " number should be 4",  expected: 4, ArcAD.nr);
107        assertEquals( message: "Arc " + ArcDB + " number should be 5",  expected: 5, ArcDB.nr);
108        assertEquals( message: "Arc " + ArcBE + " number should be 6",  expected: 6, ArcBE.nr);
109        assertEquals( message: "Arc " + ArcEA + " number should be 7",  expected: 7, ArcEA.nr);
110        System.out.println(testGraph);
111     }
112
```

```java
@Test (timeout=20000)
public void tester4() {
   GraphTask graphTask = new GraphTask();
   GraphTask.Graph testGraph = graphTask.new Graph( s: "Test 4", vertexA);
   GraphTask.Arc ArcAB = graphTask.new Arc( s: "AB", vertexB,  a: null);
   GraphTask.Arc ArcBC = graphTask.new Arc( s: "BC", vertexC,  a: null);
   GraphTask.Arc ArcCD = graphTask.new Arc( s: "CD", vertexD,  a: null);
   GraphTask.Arc ArcDA = graphTask.new Arc( s: "DA", vertexA,  a: null);
   GraphTask.Arc ArcAE = graphTask.new Arc( s: "AE", vertexE,  a: null);
   GraphTask.Arc ArcEC = graphTask.new Arc( s: "EC", vertexC,  a: null);
   GraphTask.Arc ArcCF = graphTask.new Arc( s: "CF", vertexF,  a: null);
   GraphTask.Arc ArcFA = graphTask.new Arc( s: "FA", vertexA,  a: null);
   vertexA.first = ArcAB;
   vertexB.first = ArcBC;
   vertexC.first = ArcCD;
   vertexD.first = ArcDA;
   vertexE.first = ArcEC;
   vertexF.first = ArcFA;
   vertexA.next = vertexB;
   vertexB.next = vertexC;
   vertexC.next = vertexD;
   vertexD.next = vertexE;
   vertexE.next = vertexF;
   ArcAB.next = ArcAE;
   ArcCD.next = ArcCF;
   graphTask.enumerate_graphs_edges(testGraph);
   assertEquals( message: "Arc " + ArcAB + " number should be 1",  expected: 1, ArcAB.nr);
   assertEquals( message: "Arc " + ArcBC + " number should be 2",  expected: 2, ArcBC.nr);
   assertEquals( message: "Arc " + ArcCD + " number should be 3",  expected: 3, ArcCD.nr);
   assertEquals( message: "Arc " + ArcDA + " number should be 4",  expected: 4, ArcDA.nr);
   assertEquals( message: "Arc " + ArcAE + " number should be 5",  expected: 5, ArcAE.nr);
   assertEquals( message: "Arc " + ArcEC + " number should be 6",  expected: 6, ArcEC.nr);
   assertEquals( message: "Arc " + ArcCF + " number should be 7",  expected: 7, ArcCF.nr);
   assertEquals( message: "Arc " + ArcFA + " number should be 8",  expected: 8, ArcFA.nr);
   System.out.println(testGraph);
}
```

```
149
150     @Test (expected = RuntimeException.class)
151     public void tester5() {
152         GraphTask graphTask = new GraphTask();
153         GraphTask.Graph testGraph = graphTask.new Graph( s: "5", vertexA);
154         GraphTask.Arc arcAB = graphTask.new Arc( s: "AB", vertexB,  a: null);
155         GraphTask.Arc arcAC = graphTask.new Arc( s: "AC", vertexC,  a: null);
156         GraphTask.Arc arcAD = graphTask.new Arc( s: "AD", vertexD,  a: null);
157         vertexA.first = arcAB;
158         vertexA.next = vertexB;
159         vertexB.next = vertexC;
160         vertexC.next = vertexD;
161         arcAB.next = arcAC;
162         arcAC.next = arcAD;
163         graphTask.enumerate_graphs_edges(testGraph);
164
165     }
166
167 }
168
```

Testide tulemused

Test 1

```
A --> AB (A -- 1 --> B) AC (A -- 5 --> C)
B --> BC (B -- 2 --> C)
C --> CD (C -- 3 --> D)
D --> DA (D -- 4 --> A)
```

Test 2

```
A --> AB (A -- 1 --> B)
B --> BC (B -- 2 --> C) BD (B -- 7 --> D)
C --> CD (C -- 3 --> D)
D --> DE (D -- 4 --> E) DF (D -- 8 --> F)
E --> EF (E -- 5 --> F)
F --> FB (F -- 6 --> B) FA (F -- 9 --> A)
```

Test 3

```
A --> AB (A -- 1 --> B) AD (A -- 4 --> D)
B --> BC (B -- 2 --> C) BE (B -- 6 --> E)
C --> CA (C -- 3 --> A)
D --> DB (D -- 5 --> B)
E --> EA (E -- 7 --> A)
```

Test 4

```
Test 4
A --> AB (A -- 1 --> B) AE (A -- 5 --> E)
B --> BC (B -- 2 --> C)
C --> CD (C -- 3 --> D) CF (C -- 7 --> F)
D --> DA (D -- 4 --> A)
E --> EC (E -- 6 --> C)
F --> FA (F -- 8 --> A)
```

Test 5

```
java.lang.RuntimeException: Graph 5 isn't Euler graph!!!

    at GraphTask.enumerate_graphs_edges(GraphTask.java:44)
    at GraphTaskTest.test5(GraphTaskTest.java:206) <10 internal calls>
    at java.base/java.util.concurrent.FutureTask.run(FutureTask.java:264)
    at java.base/java.lang.Thread.run(Thread.java:834)
```