

Gebze Technical University

Computer Engineering

CSE 222
2017 Spring

HOMEWORK 9 REPORT

Gözde DOĞAN
131044019

Course Assistant: Ahmet SOYYİĞİT

İçindekiler

1. Problem Solution Approach	3
1. public int addRandomEdgesToGraph (int edgeLimit);	3
2. public int [] breadthFirstSearch (int start);	3
3. public Graph[] getConnectedComponentUndirectedGraph ();	4
4. public boolean isBipartiteUndirectedGraph ();	5
5. public void writeGraphToFile (String fileName);	5
2. Test Cases	6
3. Running Command and Results	7
1.addRandomEdgesToGraph:	7
2. breadthFirstSearch:	8
3.getConnectedComponentUndirectedGraph:	9
4.isBipartiteUndirectedGraph:	10
5.writeGraphToFile:	11
1. inputFileForDirectedMatrixGraph:	11
2. outputFileForDirectedMatrixGraph:	11
3. inputFileForDirectedListGraph:	12
4. outputFileForDirectedListGraph:	12
5. inputFileForUndirectedMatrixGraph	13
6. outputFileForUndirectedMatrixGraph	13
7. inputFileForUndirectedListGraph	14
8. outputFileForUndirectedListGraph	14

1. Problem Solution Approach

- ListGraph ve MatrixGraph class'ları için test işleminde işimi kolaylaştırması açısından toString metotları yazdım.

1. public int addRandomEdgesToGraph (int edgeLimit);

- Gelen edgeLimit değeri ile 0 arasında random bir sayı seçilir.
- Bu random sayı graph'a eklenecek edge sayısını belirtiyor. (numOfInsertEdge)
- numOfInsertEdge kadar, random olarak numOfVertex ve 0 arasında source ve target değerleri seçilir.
- Seçilen source ve target arasında weight değeri 1.0 olan bir edge oluşturulur.
- Oluşturulan edge graph'a eklenir.
- Insert işlemi numOfInsertEdge(random olarak seçilen eklenecek edge sayısı) kadar devam etti.
- numOfInsertEdge'e ulaşıldığında işlem biter.

2. public int [] breadthFirstSearch (int start);

- breadthFirstSearch metodu kitabın breadthFirstSearch class'ından alındı.
- Kitaptaki metot undirected graphlar için uygun olmadığı için düzenlemeler yapıldı.
- Bir private breadthFirstSearch metodu tanımlandı ve search işlemini gerçekleştiren ve kitaptan alınıp düzenlenen breadthFirstSearch metodu bu metot.
- private void breadthFirstSearch(int start, int[] visited, int[] parent);
 - start: search işleminin başlayacağı vertex
 - visited: ziyaret edilen vertexler üzerinden tekrar bir arama işlemi başlatılmasını diye tutulmuş bir array (visited olanlar 1, diğerleri -1)
 - parent: vertexlere nereden gidildiğini tutan array (undirected graphlarda gidilemeyen vertexlerin kalma ihtimali olduğu için bu metoda yollanan bir parametre)
- public breadthFirstSearch metodu start vertexi alır
 - int[] order: start vertexten başlayarak vertex değerleri bu arrayde tutulur. Gidilemeyen ihtimaline karşı order arrayinin devamı da 0. vertexten başlayarak start vertex'e kadar olan vertexler ile doldurulur.
 - int[] parent: vertexler'e hangi vertexlerden gidildiğini tutan array. Bu array'in bütün elemanları -1 ile initialize edilir.
 - int[] visited: ziyaret edilen vertexlere bir daha gidilmesini engellemek adına tutulmuş bir array (gidilen vertexlerin değerleri 1, gidilemeyen vertexlerin değerleri -1)

- public olan metot da order arrayindeki bütün değerlerden başlayarak bir search yapılır. (Daha önce ziyaret edilmiş vertexler dışındaki vertexler üzerinde yapılmasına dikkat edilir.)
- breadthFirstSearch algoritması;
 - başlangıç vertex'i , ardından başlangıç vertex'ine 1 adım uzaklıktaki vertex'ler, ardından başlangıç vertex'ine 2 adım uzaklıktaki vertexler şeklinde graph'ı arar.
 - Bu işlemi queue kullanarak gerçekleştirir
 - Queue'ya başlangıç vertex'i eklenir ilk olarak ve queue boş olana kadar gerekli işlem gerçekleştirilir.
 - Queue boş değilse, iterator ile sağlanan vertex'lerde ilerleme işlemi gerçekleştirilir.
 - Gidilen vertex'ler henüz identified değilse queue'ya eklenir.
 - Bu işlem bütün elemanlar dolaşıldığında sonlanır.
 - İşlem sonlandığında hangi vertex'e nerden ulaşıldığını tutan bir parent arrayi return edilir.

3. public Graph[] getConnectedComponentUndirectedGraph();

- private Graph addGraph(String FileName, int start, boolean[] visited, int indexV) şeklinde bir metot yazıldı.
- addGraph metodu, start vertexten başlayarak daha önce gezilmemiş vertexleri bir array e attı.
- Gezme işlemi bittiğinde(daha ileri gidilemediğinde) tutulan arrayi bir dosyaya yazdı.
- Dosya adı graphMATRIXorLIST_indexArrayOfGraph.txt şeklinde oluşturuldu.
- Dosyaya yazma işlemi sonucu oluşan dosyalar örnek olarak ödev klasörünün içinde var.
- Dosyaya yazma işlemi de bittiğinde createGraph işlemi gerçekleştirildi.
- Oluşturulan bu graph return edilip, çağırıldığı fonksiyonda gerekli yere insert edildi.
- getConnectedComponentUndirectedGraph metodunda öncelikle graph'ın undirected olup olmadığı kontrol edildi ve undirected ise işlem gerçekleştirildi. Directed ise null return edildi.
- Graph arrayi oluşturuldu.
- Yazılacak dosya adı oluşturuldu.
- addGraph metodu bütün unconnected graphlar için gerçekleştirebilsin diye her vertex üzerinden çağırıldı.
- İşlem bittiğinde graph arrayi return edildi.

4. public boolean isBipartiteUndirectedGraph ();

- stack yapısı kullanıldı.
- Renklendirme olayı uygulandı. (1 ve 0 değerleri verilerek)
- Bir integer arrayi tutuldu. Renklendirme için.
- Renklendirilen değerler stack'e atıldı.
- Stackten eleman çekildi, iterator ile bu elemanın komşuları arandı ve renklendirmesi yapıldı. (red-black tree gibi.)
- Stackten çekilen elemanın rengi ile komşularının rengi aynı olursa false return edildi. (bipartite değildir.) Renkler aynı değilse bir sonraki komşuya, stackteki bir sonraki elemana bakıldı.
- <http://www.geeksforgeeks.org/bipartite-graph/> sitesinden yararlanıldı.

5. public void writeGraphToFile (String fileName);

- breadthFirstSearch algoritmasına benzer bir algoritma kullanıldı.
- Komşu olanların komşuluğunu belirlemek adına bir integer arrayi(chilids, 2 boyutlu) tutuldu.
- Bu işlem bittikten sonra arraydeki değerlere göre graph bir dosyaya yazdırıldı.

2. Test Cases

- Bütün metotlar directed, undirected, list graph ve matrix graph için test edildi.
- addRandomEdgesToGraph metodunda random işlemler olduğu için birkaç defa test edildi. (Her test edilişinde farklı sayı ve farklı eklemelerin gerçekleşeceği gösterildi.)
- Kitaptan alınan breadthFirstSearch metodu directed graphlar için çalışıyor, bunun undirected graphlarda da çalışabilmesi sağlandı ve test işlemini geçti.
- getConnectedComponentUndirectedGraph metodu sadece undirected graphlar üzerinde null dışında bir sonuç vermeli. Bunu undirected ve directed graphlarda deneyerek gösterilmesini sağladım.
- isBipartiteUndirectedGraph metodu hem directed hem undirected graphlar üzerinde çalışmalı ve bunun için directed ve undirected graphlar üzerinde test edildi.
- writeGraphToFile metodu bir graph'ı verilen bir dosyaya yazar. Yine directed, undirected, ListGraph ve MatrixGraph için test edildi.
- Test işlemlerinin sonuçları (Ekran görüntüleri) Random commands and results başlığı altında gösterildi.

3. Running Command and Results

1.addRandomEdgesToGraph:

DIRECTED and UNDIRECTED GRAPHS:

```
TESTING addRandomEdgesToGraph For MatrixGraph:
new Edge: (1 ,0)
new Edge: (0 ,2)
new Edge: (6 ,4)
new Edge: (2 ,7)
numOfInsertedEdges: 4
MatrixGraph:
[(0, 1), (0, 2), (0, 3), (1, 0), (1, 2), (1, 4), (1, 5), (2, 5), (2, 7), (3, 6), (4, 6), (4, 7), (5, 7), (6, 4), (6, 8), (7, 8)]

TESTING addRandomEdgesToGraph For ListGraph:
new Edge: (1 ,1)
numOfInsertedEdges: 1
ListGraph:
[(0, 1), (0, 3), (1, 2), (1, 4), (1, 5), (1, 1), (2, 5), (3, 6), (4, 6), (4, 7), (5, 7), (6, 8), (7, 8)]
```

```
TESTING addRandomEdgesToGraph For MatrixGraph:
new Edge: (2 ,7)
new Edge: (6 ,5)
numOfInsertedEdges: 2
MatrixGraph:
[(0, 1), (0, 3), (1, 2), (1, 4), (1, 5), (2, 5), (2, 7), (3, 6), (4, 6), (4, 7), (5, 7), (6, 5), (6, 8), (7, 8)]

TESTING addRandomEdgesToGraph For ListGraph:
new Edge: (0 ,7)
numOfInsertedEdges: 1
ListGraph:
[(0, 1), (0, 3), (0, 7), (1, 2), (1, 4), (1, 5), (2, 5), (3, 6), (4, 6), (4, 7), (5, 7), (6, 8), (7, 8)]
```

```
TESTING addRandomEdgesToGraph For MatrixGraph:
new Edge: (0 ,0)
new Edge: (4 ,2)
new Edge: (4 ,4)
numOfInsertedEdges: 3
MatrixGraph:
[(0, 0), (0, 1), (0, 3), (1, 2), (1, 4), (1, 5), (2, 5), (3, 6), (4, 2), (4, 4), (4, 6), (4, 7), (5, 7), (6, 8), (7, 8)]

TESTING addRandomEdgesToGraph For ListGraph:
new Edge: (2 ,3)
numOfInsertedEdges: 1
ListGraph:
[(0, 1), (0, 3), (1, 2), (1, 4), (1, 5), (2, 5), (2, 3), (3, 6), (4, 6), (4, 7), (5, 7), (6, 8), (7, 8)]
```

2. breadthFirstSearch:

DIRECTED GRAPHS:

Edge ekledikten önce;

```
TESTING breadthFirstSearch For MatrixGraph: [(parent, child),.....]
[(-1, 0), (0, 1), (1, 2), (0, 3), (1, 4), (1, 5), (3, 6), (4, 7), (6, 8), (7, 9),
(7, 10), (9, 11), (10, 12), (12, 13)]

TESTING breadthFirstSearch For ListGraph:
[(-1, 0), (0, 1), (1, 2), (0, 3), (1, 4), (1, 5), (3, 6), (4, 7), (6, 8)]
```

Edge ekledikten sonra;

```
TESTING breadthFirstSearch For MatrixGraph after add edge: [(parent, child),.....]
[(-1, 0), (0, 1), (1, 2), (0, 3), (1, 4), (1, 5), (3, 6), (4, 7), (6, 8), (7, 9), (7, 10),
(9, 11), (10, 12), (12, 13)]

TESTING breadthFirstSearch For ListGraph after add edge:
[(-1, 0), (0, 1), (1, 2), (0, 3), (1, 4), (1, 5), (3, 6), (4, 7), (6, 8)]
```

AYNI

UNDIRECTED GRAPHS:

Edge eklenmeden önce;

```
TESTING breadthFirstSearch For MatrixGraph: [(parent, child),.....]
[(-1, 0), (0, 1), (1, 2), (-1, 3), (6, 4), (1, 5), (3, 6), (4, 7), (6, 8), (-1, 9), (9,
10), (9, 11), (10, 12), (12, 13)]

TESTING breadthFirstSearch For ListGraph:
[(-1, 0), (0, 1), (1, 2), (-1, 3), (6, 4), (1, 5), (3, 6), (4, 7), (6, 8)]
```

Edge eklendikten sonra;

```
TESTING breadthFirstSearch For MatrixGraph after add edge: [(parent, child),.....]
[(-1, 0), (0, 1), (1, 2), (-1, 3), (6, 4), (1, 5), (3, 6), (4, 7), (6, 8), (10, 9), (4,
10), (9, 11), (10, 12), (12, 13)]

TESTING breadthFirstSearch For ListGraph after add edge:
[(-1, 0), (0, 1), (1, 2), (0, 3), (7, 4), (1, 5), (3, 6), (1, 7), (7, 8)]
```


3.getConnectedComponentUndirectedGraph:

DIRECTED:

```
TESTING DIRECTED getConnectedComponentUndirectedGraph For MatrixGraph:
Graph: [(0, 1), (0, 3), (1, 2), (1, 4), (1, 5), (2, 5), (3, 6), (4, 6), (4, 7), (4, 11),
(5, 7), (6, 8), (7, 8), (7, 9), (7, 10), (9, 10), (9, 11), (10, 12), (11, 7), (11, 10),
(12, 13)]
isBipartiteUndirectedGraph: null

TESTING DIRECTED getConnectedComponentUndirectedGraph For ListGraph:
Graph: [(0, 1), (0, 3), (1, 2), (1, 4), (1, 5), (2, 5), (3, 6), (3, 2), (4, 6), (4, 7),
(5, 7), (6, 8), (6, 0), (7, 8)]
isBipartiteUndirectedGraph: null
```

UNDIRECTED:

```
TESTING getConnectedComponentUndirectedGraph For MatrixGraph:
getConnectedComponentUndirectedGraph:
0.graph:
[(1, 0), (2, 1), (5, 1), (5, 2)]
1.graph:
[(6, 3), (6, 4), (7, 4), (8, 6), (8, 7)]
2.graph:
[(10, 9), (11, 9), (12, 10), (13, 12)]

TESTING getConnectedComponentUndirectedGraph For ListGraph:
Graph: [(0, 1), (1, 0), (1, 2), (1, 5), (2, 1), (2, 5), (3, 6), (4, 6), (4, 7), (5, 1),
(5, 2), (6, 3), (6, 4), (6, 8), (7, 4), (7, 8), (8, 6), (8, 7)]
getConnectedComponentUndirectedGraph:
0.graph:
[(0, 1), (1, 0), (1, 2), (1, 5), (2, 1), (2, 5), (5, 1), (5, 2)]
1.graph:
[(3, 6), (4, 6), (4, 7), (6, 3), (6, 4), (6, 8), (7, 4), (7, 8), (8, 6), (8, 7)]
```

4.isBipartiteUndirectedGraph:

DIRECTED:

```
TESTING isBipartiteUndirectedGraph For MatrixGraph:
Graph: [(0, 1), (0, 3), (1, 2), (1, 4), (1, 5), (2, 5), (3, 6), (4, 6), (4, 7), (4, 11),
        (5, 7), (6, 8), (7, 8), (7, 9), (7, 10), (9, 10), (9, 11), (10, 12), (11, 7), (11, 10),
        (12, 13)]
isBipartiteUndirectedGraph: false

TESTING isBipartiteUndirectedGraph For ListGraph:
Graph: [(0, 1), (0, 3), (1, 2), (1, 4), (1, 5), (2, 5), (3, 6), (3, 2), (4, 6), (4, 7),
        (5, 7), (6, 8), (6, 0), (7, 8)]
isBipartiteUndirectedGraph: false
```

UNDIRECTED:

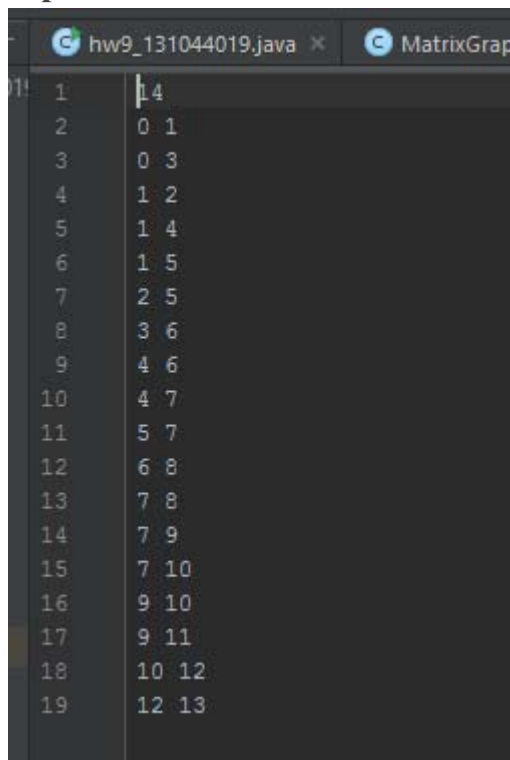
```
TESTING isBipartiteUndirectedGraph For MatrixGraph:
Graph: [(1, 0), (2, 1), (5, 1), (5, 2), (6, 3), (6, 4), (7, 4), (8, 6), (8, 7), (10, 4),
        (10, 9), (11, 9), (12, 10), (13, 12)]
isBipartiteUndirectedGraph: false

TESTING isBipartiteUndirectedGraph For ListGraph:
Graph: [(0, 1), (0, 3), (1, 0), (1, 2), (1, 5), (1, 7), (2, 1), (2, 5), (3, 6), (3, 5), (3,
        0), (4, 6), (4, 7), (5, 1), (5, 2), (5, 3), (6, 3), (6, 4), (6, 8), (7, 4), (7, 8), (7,
        1), (8, 6), (8, 7)]
isBipartiteUndirectedGraph: false
```

5.writeGraphToFile:

DIRECTED:

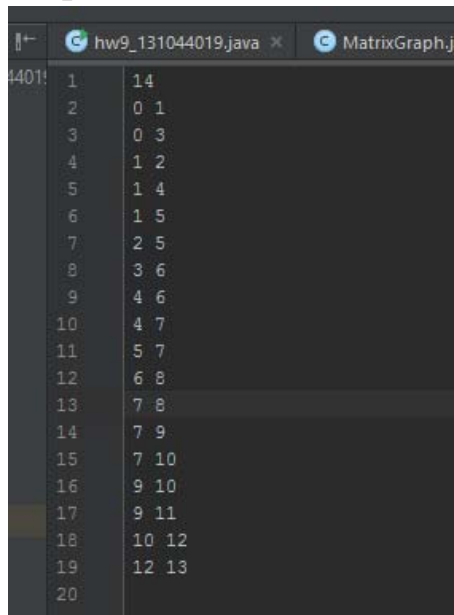
1. inputFileForDirectedMatrixGraph:



A screenshot of a Java IDE window titled 'hw9_131044019.java'. The code displays a list of 19 directed edges for a graph. Each edge is represented by a line with a source node, a target node, and a weight. The edges are as follows:

Edge ID	Source	Target	Weight
1	1	4	1
2	2	0	1
3	3	0	3
4	4	1	2
5	5	1	4
6	6	1	5
7	7	2	5
8	8	3	6
9	9	4	6
10	4	7	4
11	5	7	5
12	6	8	6
13	7	8	7
14	7	9	7
15	7	10	7
16	9	10	9
17	9	11	9
18	10	12	10
19	12	13	12

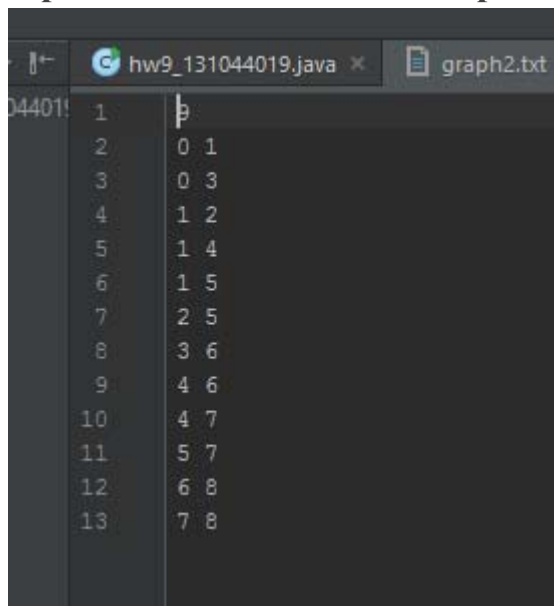
2. outputFileForDirectedMatrixGraph:



A screenshot of a Java IDE window titled 'hw9_131044019.java'. The code displays a list of 19 directed edges for a graph, identical to the first screenshot. Each edge is represented by a line with a source node, a target node, and a weight. The edges are as follows:

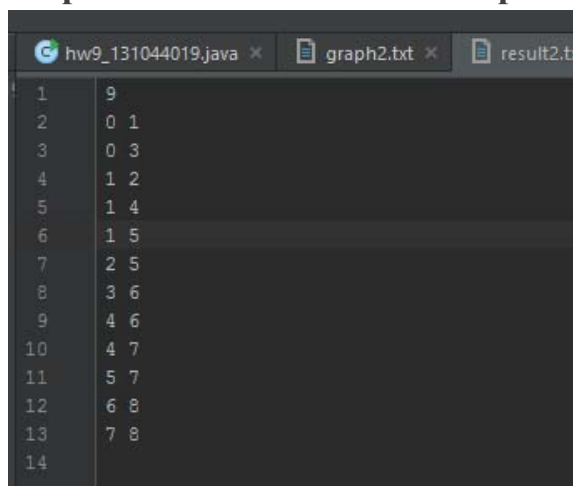
Edge ID	Source	Target	Weight
1	1	4	1
2	2	0	1
3	3	0	3
4	4	1	2
5	5	1	4
6	6	1	5
7	7	2	5
8	8	3	6
9	9	4	6
10	4	7	4
11	5	7	5
12	6	8	6
13	7	8	7
14	7	9	7
15	7	10	7
16	9	10	9
17	9	11	9
18	10	12	10
19	12	13	12

3. inputFileForDirectedListGraph:



```
hw9_131044019.java x graph2.txt
044019: 1 | b
        2 | 0 1
        3 | 0 3
        4 | 1 2
        5 | 1 4
        6 | 1 5
        7 | 2 5
        8 | 3 6
        9 | 4 6
       10 | 4 7
       11 | 5 7
       12 | 6 8
       13 | 7 8
```

4. outputFileForDirectedListGraph:

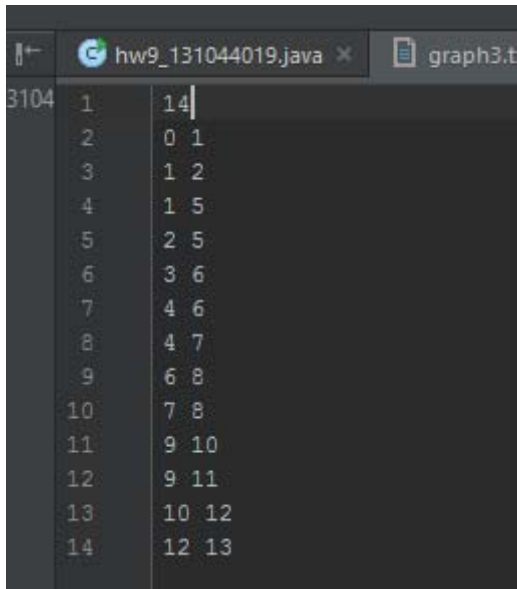


```
hw9_131044019.java x graph2.txt x result2.txt
1 | 9
2 | 0 1
3 | 0 3
4 | 1 2
5 | 1 4
6 | 1 5
7 | 2 5
8 | 3 6
9 | 4 6
10 | 4 7
11 | 5 7
12 | 6 8
13 | 7 8
14 |
```

UNDIRECTED:

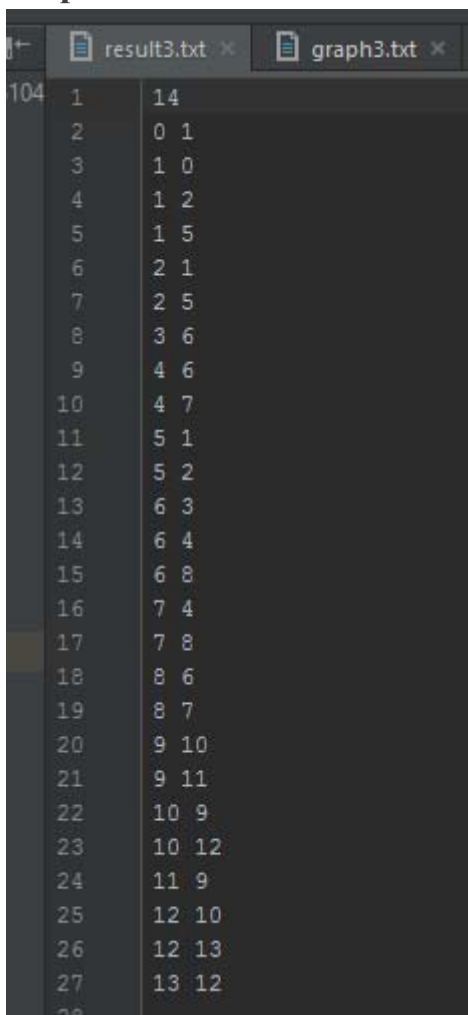
- undirected graphlarda 1,5 arası edge var ise 5,1 arası da olacağı için output dosyalarında bunlarda gösterilmistir.

5. inputFileForUndirectedMatrixGraph



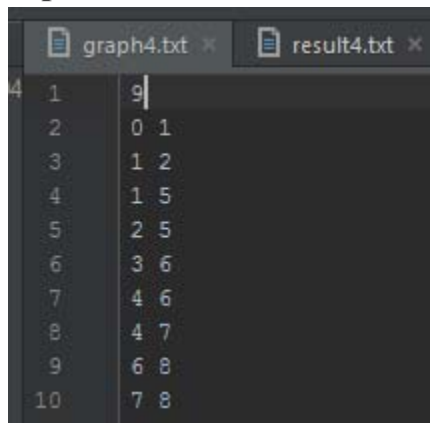
1	14
2	0 1
3	1 2
4	1 5
5	2 5
6	3 6
7	4 6
8	4 7
9	6 8
10	7 8
11	9 10
12	9 11
13	10 12
14	12 13

6. outputFileForUndirectedMatrixGraph



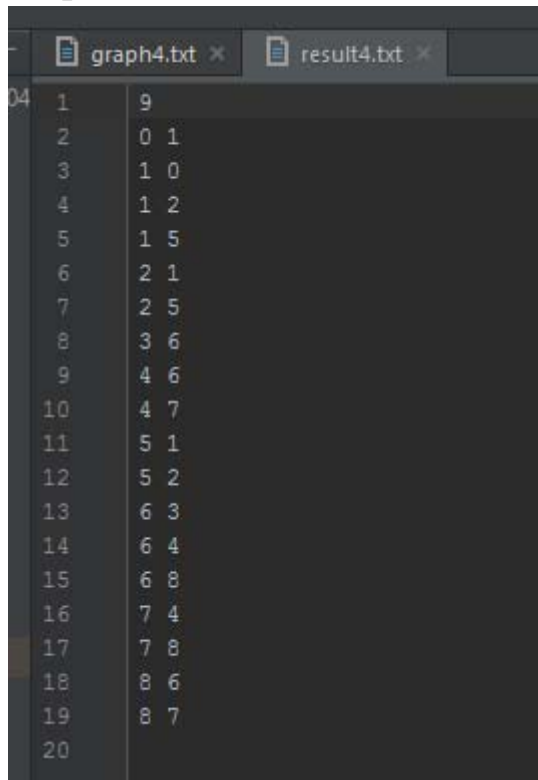
1	14
2	0 1
3	1 0
4	1 2
5	1 5
6	2 1
7	2 5
8	3 6
9	4 6
10	4 7
11	5 1
12	5 2
13	6 3
14	6 4
15	6 8
16	7 4
17	7 8
18	8 6
19	8 7
20	9 10
21	9 11
22	10 9
23	10 12
24	11 9
25	12 10
26	12 13
27	13 12

7. inputFileForUndirectedListGraph



```
graph4.txt x result4.txt x
1 9
2 0 1
3 1 2
4 1 5
5 2 5
6 3 6
7 4 6
8 4 7
9 6 8
10 7 8
```

8. outputFileForUndirectedListGraph



```
graph4.txt x result4.txt x
1 9
2 0 1
3 1 0
4 1 2
5 1 5
6 2 1
7 2 5
8 3 6
9 4 6
10 4 7
11 5 1
12 5 2
13 6 3
14 6 4
15 6 8
16 7 4
17 7 8
18 8 6
19 8 7
20
```