Gebze Technical University
Computer Engineering

CSE 222
2017 Spring

HOMEWORK HW09 REPORT

MEHMET GÜROL ÇAY
121044029

Course Assistant:

1. Problem Solutions Approach

- ***public int addRandomEdgesToGraph (int edgeLimit);***
  Bu metot implement edilirken. *isEdge(int src, int dst);* metodundan yararlandım. Rastgele oluşturduğum edge'in graphta olup olmadığını kontrol ettim eğer yoksa ekleme işlemini gerçekleştirdim. Her eklenen edge'i saymak için bir sayaç tuttum ve bunu en son return ettim. ***This*** objesi kullarak hangi graph olduğunu düşünmeden hareket edebildim. Çünkü her iki graph tipinde de ***edgeIterator()*** metodunu kullanabiliyordum.
- ***public int [] breadthFirstSearch (int start);***
  Bu metot kitap kaynak kodlarından yararlanılarak implement edilmiştir. Start vertex'i alarak başlayan bu metotta başlanılan vertex'den aşağıya doğru tüm vertex'ler gezilir ve parent array'i doldurulur. Sonrasında bu array return edilir.
- ***public Graph [] getConnectedComponentUndirectedGraph ();***
  Çözüm geliştirilememiştir.
- ***public boolean isBipartiteUndirectedGraph ();***
  Bu kısımda iki küme tutulmuştur. ArrayList olarak tuttuğum bu kümelerin her bir elemanı string barındırmaktadır. Burdaki asıl amacım bize verilmiş bir graph'ı iki kümeye ayırmaktır. En basit yaklaşım ile ele almış olduğum bir vertex'e "RED" string ataması yaparken bu vertex'in komşularına "BLUE" string ataması yaptım. Sonrasında ise her "RED" kelimesini barındıran U küme elemanını, "BLUE" kelimesini barındıran V kümesinin elemanları ile karşılaştırdım. Herhangi bir aynı renk kelimesi ile karşılaşılması durumunda false, karşılaşılmaması durumunda ise true return ettim.
- ***public void writeGraphToFile (String fileName);***
  Herbir vertex'i vertex sayısı kadar dolaşırken her bir vertex'i komşusu kadar iterated ettim ve dosyaya yazdım.

2. Test Cases

ListGraph ve MatrixGraph objelerini dosyadan okuyarak oluşturduktan sonra edgeLimit'e 5 ve 10 vererek ***addRandomEdgesToGraph*** metodunu test ettim. Test sonuçları 3. bölümdeki ekran görüntülerindeki gibidir.

Yine dosyadan okunan graph üzerinde start vertex'e 0(sıfır) verilerek ***breadthFirstSearch*** metodu test edilmiştir.

İsBipartiteUndirectedGraph metodu için ise kendi oluşturduğum graph üzerinde test ettim. Elle kontrol ettikten sonra true beklediğim sonucu metot da true return etti.

3. Running and Results

```
ListGraph is testing...
Read graph from file:
[(0, 1): 1.0]
[(0, 3): 1.0]
[(1, 2): 1.0]
[(1, 4): 1.0]
[(1, 5): 1.0]
[(2, 5): 1.0]
[(3, 6): 1.0]
[(4, 6): 1.0]
[(4, 7): 1.0]
[(5, 7): 1.0]
[(6, 8): 1.0]
[(7, 8): 1.0]

edgeLimit: 10
Number of edges added: 4
New list graph
[(0, 1): 1.0]
[(0, 3): 1.0]
[(0, 8): 1.0]
[(1, 2): 1.0]
[(1, 4): 1.0]
[(1, 5): 1.0]
[(2, 5): 1.0]
[(3, 6): 1.0]
[(4, 6): 1.0]
[(4, 7): 1.0]
[(4, 1): 1.0]
[(5, 7): 1.0]
[(5, 0): 1.0]
[(5, 4): 1.0]
[(6, 8): 1.0]
[(7, 8): 1.0]
```

```
MatrixGraph is testing...
Read graph from file:
[(0, 1): 1.0]
[(0, 3): 1.0]
[(1, 2): 1.0]
[(1, 4): 1.0]
[(1, 5): 1.0]
[(2, 5): 1.0]
[(3, 6): 1.0]
[(4, 6): 1.0]
[(4, 7): 1.0]
[(5, 7): 1.0]
[(6, 8): 1.0]
[(7, 8): 1.0]

edgeLimit: 5
Number of edges added to graph 1
New matrix graph
[(0, 1): 1.0]
[(0, 3): 1.0]
[(1, 2): 1.0]
[(1, 4): 1.0]
[(1, 5): 1.0]
[(2, 5): 1.0]
[(3, 6): 1.0]
[(3, 8): 1.0]
[(4, 6): 1.0]
[(4, 7): 1.0]
[(5, 7): 1.0]
[(6, 8): 1.0]
[(7, 8): 1.0]
```

```
Breadth First Search testing:
Graph:
[(0, 1): 1.0]
[(0, 3): 1.0]
[(1, 2): 1.0]
[(1, 4): 1.0]
[(1, 5): 1.0]
[(2, 5): 1.0]
[(2, 10): 1.0]
[(3, 6): 1.0]
[(4, 6): 1.0]
[(4, 7): 1.0]
[(4, 3): 1.0]
[(5, 7): 1.0]
[(6, 8): 1.0]
[(6, 6): 1.0]
[(7, 8): 1.0]
[(8, 8): 1.0]
[(9, 10): 1.0]
[(9, 4): 1.0]

Breath First Search testing; start is 0:
>1 0 1 0 1 1 3 4 6 -1 2
```

```
BipartiteUndirectedGraph is testing...
Graph:
[(0, 1): 1.0]
[(0, 3): 1.0]
[(0, 5): 1.0]
[(1, 0): 1.0]
[(1, 2): 1.0]
[(2, 1): 1.0]
[(2, 5): 1.0]
[(3, 0): 1.0]
[(3, 4): 1.0]
[(4, 3): 1.0]
[(4, 5): 1.0]
[(5, 0): 1.0]
[(5, 2): 1.0]
[(5, 4): 1.0]

isBipartite: true
```