Data Structures (Spring 2020)

Minimum Spanning Tree (11th Lab)

2020.05.29

Seoul National University
Database Systems Lab

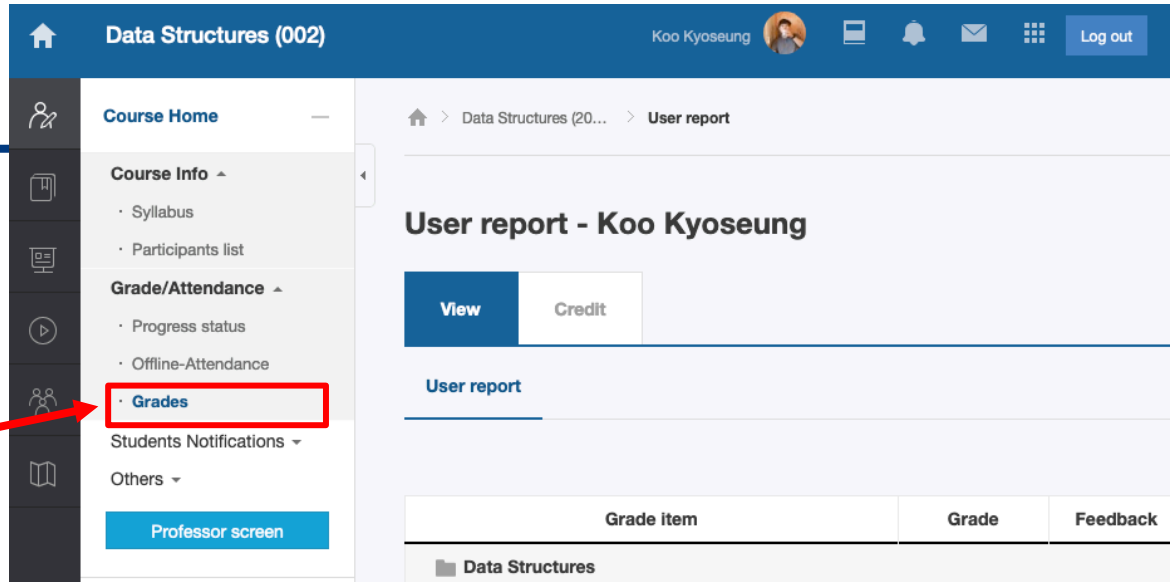
Today's Lab

- Minimum Spanning Tree
  - Prim's Algorithm
  - Kruskal's Algorithm

- 2nd Midterm Claim Session

Announcement

- 2nd Midterm score
  - Check out your score at eTL



- Claim sessions
  - Online session (This class): May 29 Lab class (16:00 ~ 17:50)
  - Offline session: Visit our office (Engr. Bldg. 301, Room 418) 13:30~15:00 on Tue June 2

Minimum Spanning Tree

- MST: a spanning tree whose sum of edge weights is as small as possible.
  - *Kruskal's* and *Prim's algorithm* is **greedy algorithms** that finds a minimum spanning tree for a weighted undirected graph.
  - It finds a subset of the edges that forms a tree that includes every vertex, where the total weight of all the edges in the tree is minimized.
  - Minimum spanning trees have direct applications in the design of networks, including computer networks, telecommunications networks, transportation networks, water supply networks, and electrical grids.

From: https://en.wikipedia.org/wiki/Minimum_spanning_tree

Algorithms for finding MST

- Greedy algorithms for a weighted undirected graph.
  - *Prim*: Finding the nearest and smallest edge connecting different set.
  - *Kruskal*: Finding the most minimum edge without cycle (i.e. different class).

Algorithm 9 (Kruskal's Minimum Spanning Tree)

```
Assign each vertex to a separate class;
F = {e ∈ E | in an increasing order of weights};
while(number of printed edges < |V|-1) {
   Pick an edge uv ∈ F in the order;
   if (u and v are in different classes) {
      print uv;
      merge their classes;
   }
}
```

Algorithm 8 (Prim's Minimum Spanning Tree (Improved version))

```
S = {s};                                   // s is an arbitrary start vertex.
for each v ∈ V − S {
   d[v], neighbor[v] = { w_sv, s;          if sv ∈ E
                       { ∞, s or null;     otherwise
}
while(V − S ≠ ∅) {
   find v ∈ V − S such that d[v] is minimum;
   print v, neighbor[v];                   // This edge becomes part of MST.
   S = S ∪ {v};
   for each u ∈ V − S such that vu ∈ E {
      if (w_uv < d[u]) {                   // uv is a new fringe edge.
         d[u] = w_uv; // the shortest distance to ANY vertex in S.
         neighbor[u] = v;
      }
   }
}
```

From: Bongi Moon, "Lecture Notes on Data Structures: Graph"

GraphMatrix spec

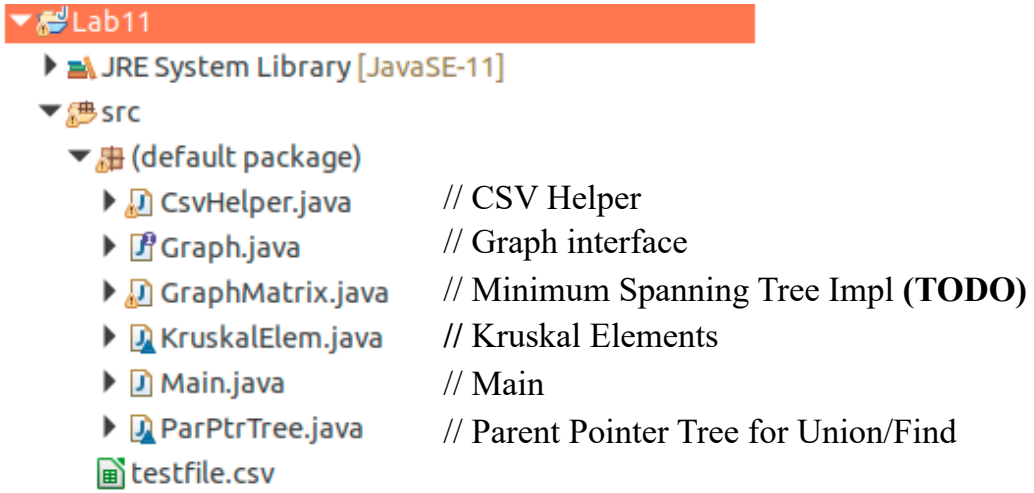
- **public static void Prim(GraphMatrix g, int s);**
  - GraphMatrix g: GraphMatrix instance
  - int s: source vertex index

- **public static void Kruskal(GraphMatrix g);**
  - GraphMatrix g: GraphMatrix instance
  - You can use Collections.sort() with ArrayList of KruskalElem type.

- You should print MST edge results in above functions.
- Assume that the graph is undirected.

Exercises

- Fill the blank of codes
  - Update your code in `GraphMatrix.java` ("`// TODO: `" section)
  - Write `Prim()`, `Kruskal()` method



Project Structure

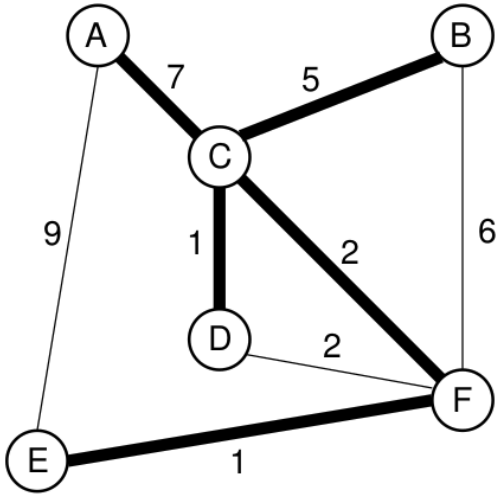
```java
public class Main {
    public static void main(String args[]) {
        String path = args[0];

        // Graph initialize
        GraphMatrix gm = new GraphMatrix(path);

        // Prim's Test
        System.out.println("======= Prim =======");
        GraphMatrix.Prim(gm, 0);

        // Kruskal Test
        System.out.println("======= Kruskal =======");
        GraphMatrix.Kruskal(gm);
    }
}
```

Main.java

Exercises

- Result

```
$ java Main testfile1.csv
======= Prim =======
2 0
3 2
5 2
4 5
1 2
======= Kruskal =======
2 3
4 5
2 5
1 2
0 2
```

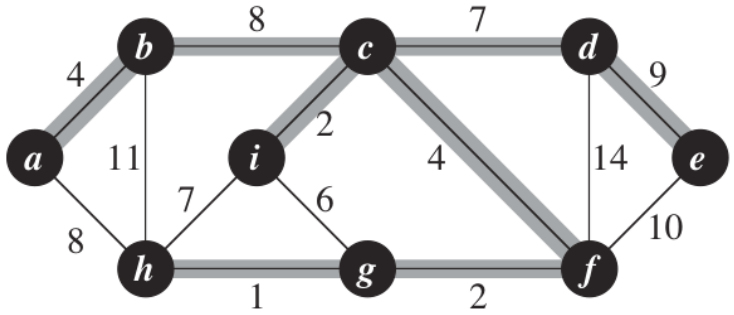


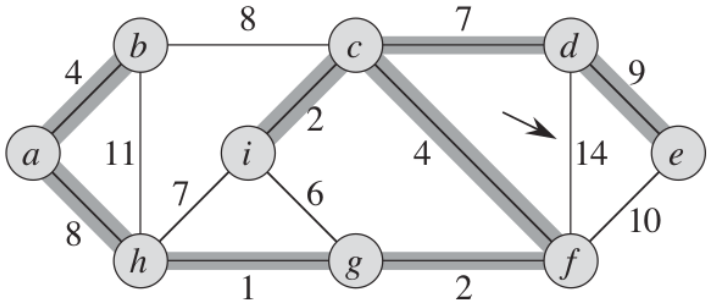
From: Cliff A. Shaffer, Data Structures and Algorithm Analysis, Edition 3.2 (Java version)

Exercises

- Result

```
$ java Main testfile2.csv
======= Prim =======
1 0
2 1
8 2
5 2
6 5
7 6
3 2
4 3
======= Kruskal =======
6 7
2 8
5 6
0 1
2 5
2 3
0 7
3 4
```





From: Thomas H. Cormen ed. *Introduction to Algorithms (3rd)*. 3rd ed. Cambridge, Mass: MIT Press, 2009.