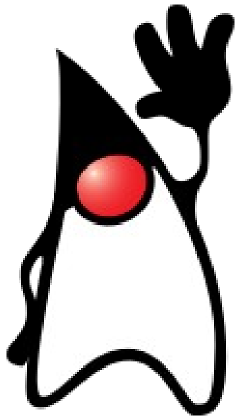
Data Structures (Spring 2020)

Shortest Paths (10th Lab)

2020.05.22
Seoul National University
Database Systems Lab

Today’s Lab

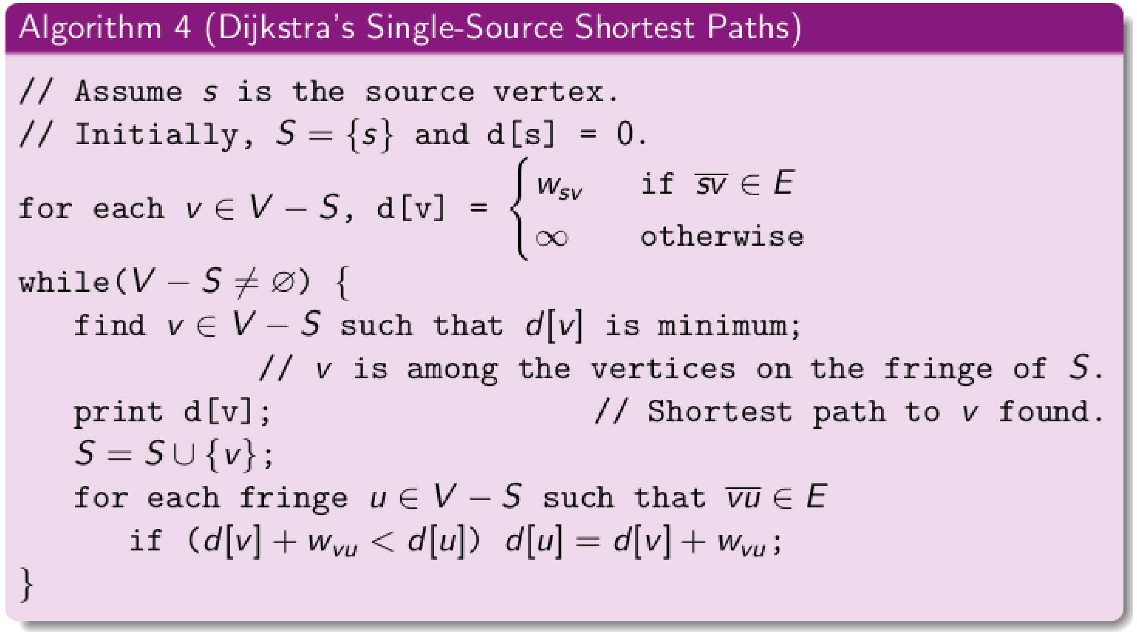
- Dijkstra's Algorithm for Single-Source
- Floyd's Algorithm for All-Pair



Dijkstra's Algorithm for Single-Source

- Dijkstra's Algorithm is an algorithm for finding the shortest paths between nodes in a graph.
  - Fixes a single node as the "source" node and finds shortest paths from the source to all other nodes in the graph.
  - All weights are assumed to be **non-negative.**

$$d(s, v_i) = min_{v_j \in S}\{d(s, v_j) + w_{ji}\}$$



Algorithm 4 (Dijkstra's Single-Source Shortest Paths)

```
// Assume $s$ is the source vertex.
// Initially, $S = \{s\}$ and d[s] = 0.
for each $v \in V - S$, d[v] = $\begin{cases} w_{sv} & \text{if } \overline{sv} \in E \\ \infty & \text{otherwise} \end{cases}$
while($V - S \neq \varnothing$) {
    find $v \in V - S$ such that $d[v]$ is minimum;
                // $v$ is among the vertices on the fringe of $S$.
    print d[v];                // Shortest path to $v$ found.
    $S = S \cup \{v\}$;
    for each fringe $u \in V - S$ such that $\overline{vu} \in E$
        if $(d[v] + w_{vu} < d[u])$ $d[u] = d[v] + w_{vu}$;
}
```

From: https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm
From: Bongi Moon, "Lecture Notes on Data Structures: Graph"

Floyd's Algorithm for All-Pair

- An algorithm for finding shortest paths in a weighted graph with positive or negative edge weights (but with no negative cycles).
    - Floyd proposed an algorithm with a dynamic programming flavor.

$$A^{k+1}[i,j] = min\{A^k[i,j], A^k[i,k] + A^k[k,j]\}.$$

Algorithm 6 (Floyd's All-Pair Shortest Paths)

```
// Assume $n$ vertices are indexed from 0 to n-1.
for(i=0; i < n ;i++)                          // initialize $A^0$ matrix.
   for(j=0; j < n ;j++)
      A[i,j] = $\begin{cases} 0 & \text{if } i = j \\ w_{ij} & \text{if } \overline{v_i v_j} \in E \\ \infty & \text{otherwise} \end{cases}$
for(k=0; k < n ;k++)                          // compute $A^{k+1}$ matrix.
   for(i=0; i < n ;i++)
      for(j=0; j < n ;j++)
         if (A[i,j] > A[i,k]+A[k,j]) A[i,j] = A[i,k]+A[k,j];
```

From: https://en.wikipedia.org/wiki/Floyd%E2%80%93Warshall_algorithm
From: Bongi Moon, "Lecture Notes on Data Structures: Graph"

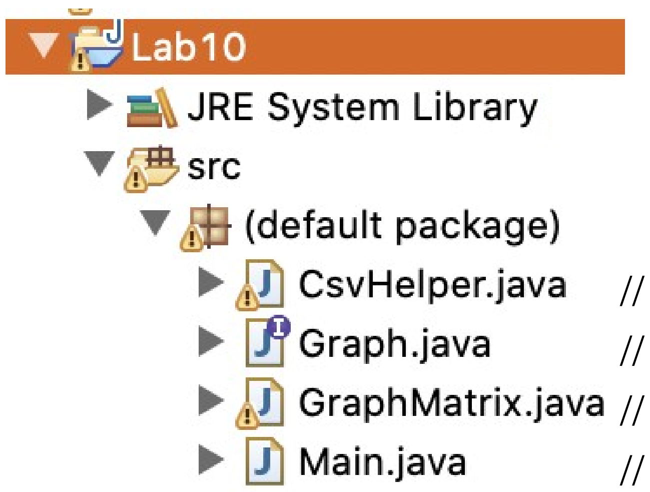
GraphMatrix spec

- **public static boolean** Dijkstra(GraphMatrix g, **int** s, **int**[] d);
    - GraphMatrix g: GraphMatrix instance
    - int s: source vertex index
    - int[] d: shortest distances from s
    - <return>: is there any negative edges?

- **public static boolean** Floyd(GraphMatrix g, **int**[][] d);
    - GraphMatrix g: GraphMatrix instance
    - int[][] d: shortest distances matrix
    - <return>: is there any negative cycles?

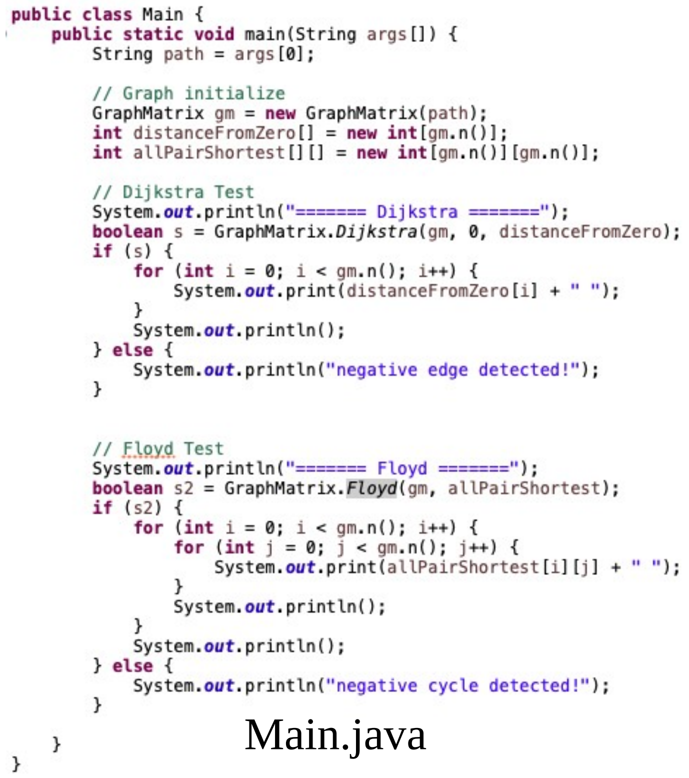
Exercises

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



// CSV Helper
// Graph interface
// Shortest Path Impl **(TODO)**
// Main

Project Structure



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

        // Graph initialize
        GraphMatrix gm = new GraphMatrix(path);
        int distanceFromZero[] = new int[gm.n()];
        int allPairShortest[][] = new int[gm.n()][gm.n()];

        // Dijkstra Test
        System.out.println("======= Dijkstra =======");
        boolean s = GraphMatrix.Dijkstra(gm, 0, distanceFromZero);
        if (s) {
            for (int i = 0; i < gm.n(); i++) {
                System.out.print(distanceFromZero[i] + " ");
            }
            System.out.println();
        } else {
            System.out.println("negative edge detected!");
        }


        // Floyd Test
        System.out.println("======= Floyd =======");
        boolean s2 = GraphMatrix.Floyd(gm, allPairShortest);
        if (s2) {
            for (int i = 0; i < gm.n(); i++) {
                for (int j = 0; j < gm.n(); j++) {
                    System.out.print(allPairShortest[i][j] + " ");
                }
                System.out.println();
            }
            System.out.println();
        } else {
            System.out.println("negative cycle detected!");
        }

    }
}
```

Main.java

Exercises

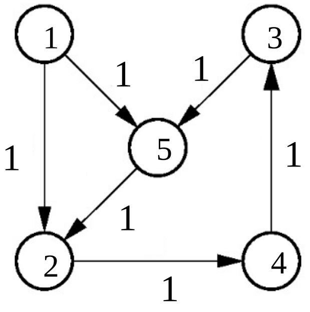
- Result

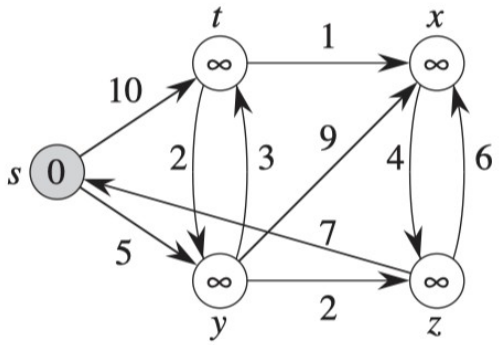
Dijkstra Alg did not consider negative value!

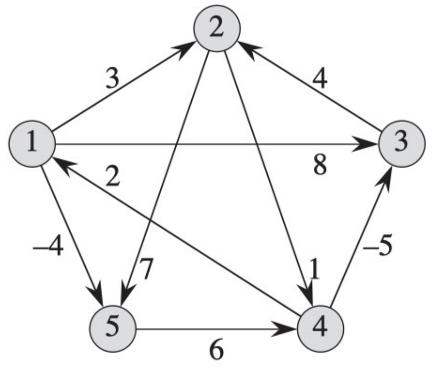
```
$ java Main test1.csv
======= Dijkstra =======
0 1 3 2 1
======= Floyd =======
0 1 3 2 1
2147483647 0 2 1 3
2147483647 2 0 3 1
2147483647 3 1 0 2
2147483647 1 3 2 0
```

```
$ java Main ita_246.csv
======= Dijkstra =======
0 8 9 5 7
======= Floyd =======
0 8 9 5 7
11 0 1 2 4
11 19 0 16 4
9 3 4 0 2
7 15 6 12 0
```

```
$ java Main ita_251.csv
======= Dijkstra =======
negative edge detected!
======= Floyd =======
0 1 -3 2 -4
3 0 -4 1 -1
7 4 0 5 3
2 -1 -5 0 -2
8 5 1 6 0
```







From: Bongi Moon, "Lecture Notes on Data Structures: Graph"
Cormen, Thomas H., and Thomas H. Cormen, eds. *Introduction to Algorithms (2nd)*. 2nd ed. Cambridge, Mass: MIT Press, 2001.

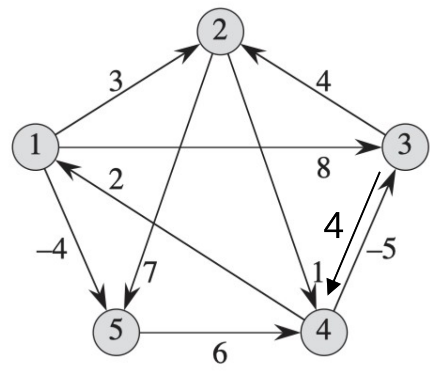
Exercises

- Result

Dijkstra Alg did not consider negative value!

```
$ java Main ita_251_neg.csv
======= Dijkstra =======
negative edge detected!
======= Floyd =======
negative cycle detected!
```

Floyd Alg did not consider negative cycle!



From: Bongi Moon, "Lecture Notes on Data Structures: Graph"
Cormen, Thomas H., and Thomas H. Cormen, eds. *Introduction to Algorithms (2nd)*. 2nd ed. Cambridge, Mass: MIT Press, 2001.