Graph Basics: graph traversals

Frédéric Guinand
email:Frederic.Guinand@univ-lehavre.fr
web: http://litis.univ-lehavre.fr/~guinand

1 Algorithms on Graphs

1.1 GraphStream primitives

For processing each node of the graph one solution consists in iterating on the set of Nodes. Obtaining the set of Nodes of the graph in GraphStream can be done by calling the `getNodeSet()` on the graph. Thus iterating of the Nodes is as simple as:

```
for(Node n:  graph.getNodeSet()) { process vertex n }
```

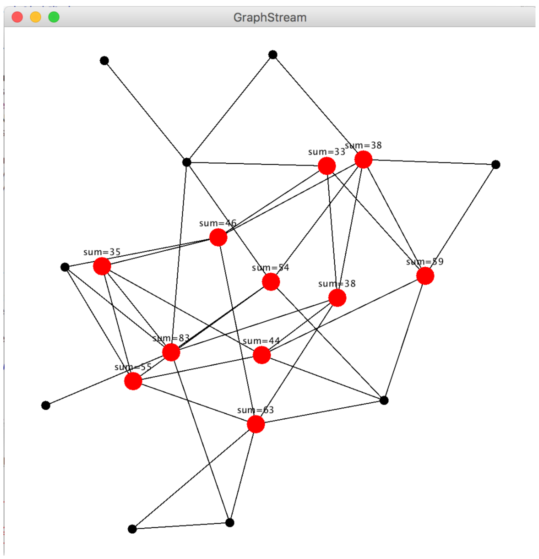
For visiting the neighborhood of a given Node v, it is possible to get an Iterator on its neighbors:

```
Iterator<Node> neighbors = v.getNeighborNodeIterator();
```

1.2 Exercise 1

Download the `firstgraphlab2.dgs` dgs file on the web site and place it in the ”dgs” directory.

1. compute the average degree of this graph.
2. the vertices of this graph all have an attribute ”cost” which value is an integer. For each vertex, sum the costs of all its neighbors and if the sum obtained is greater than a given value (a parameter), then change the style of the vertex (size 30px and color red).

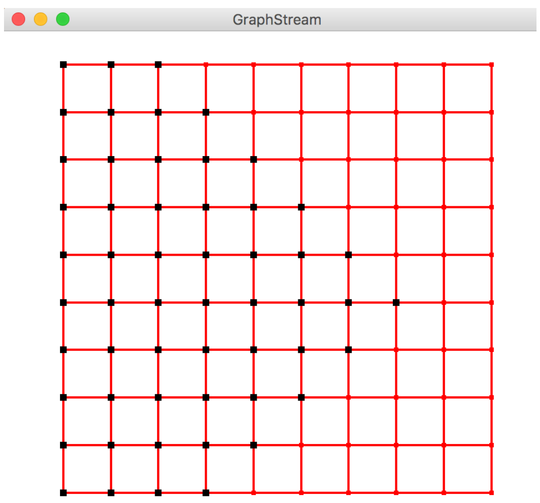


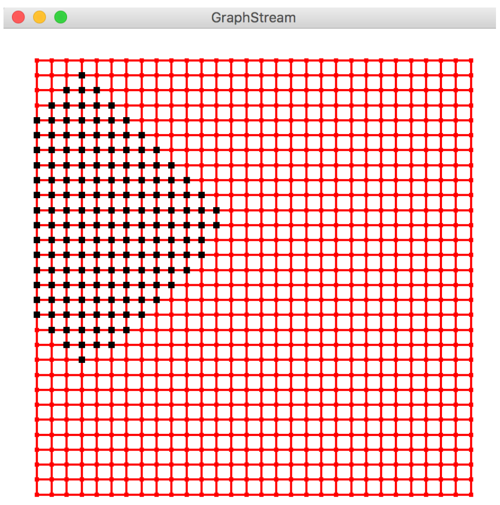
1.3 Exercise 2: Graph Traversal: Breadth-First Search

In java, the class corresponding to a Queue is java.util.Queue, but we can also use ArrayList (java.util.ArrayList
which is more convenient to use.

1. implement the BFS algorithms
2. hightlight the traversal evolution as seen during the lecture

Download the `completegrid_10.dgs` dgs file on the web site and place it in the "dgs" directory and test your algorithm on thes graphs.



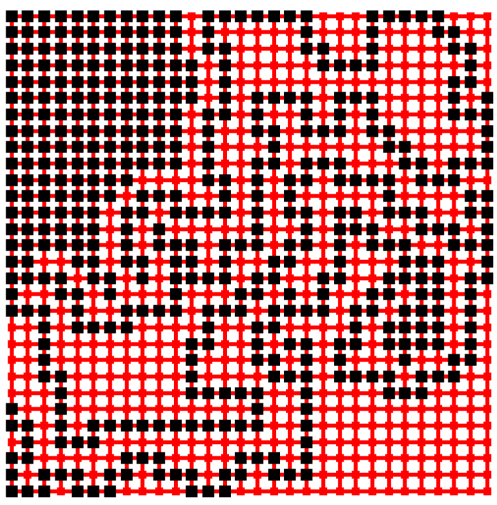


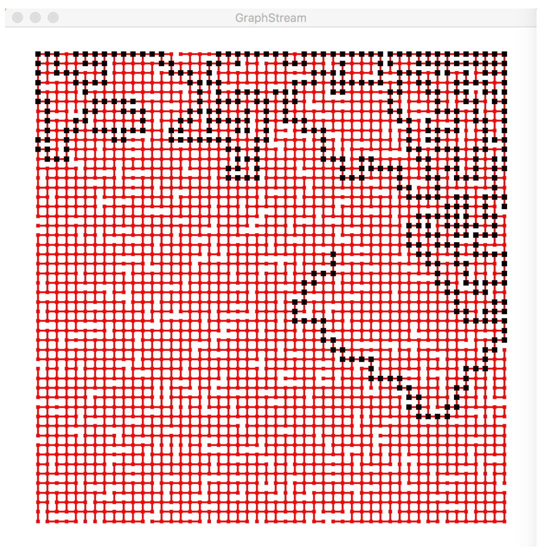
1.4 Exercise 3: Graph Traversal Depth-First Search

In java, the class corresponding to a Stack is java.util.Stack

1. implement the DFS algorithms
2. hightlight the traversal evolution as seen during the lecture

Download the `completegrid_30.dgs` and `uncompletegrid_50-0.12.dgs` dgs files on the web site and place them in the "dgs" directory and test your algorithm on this graph.



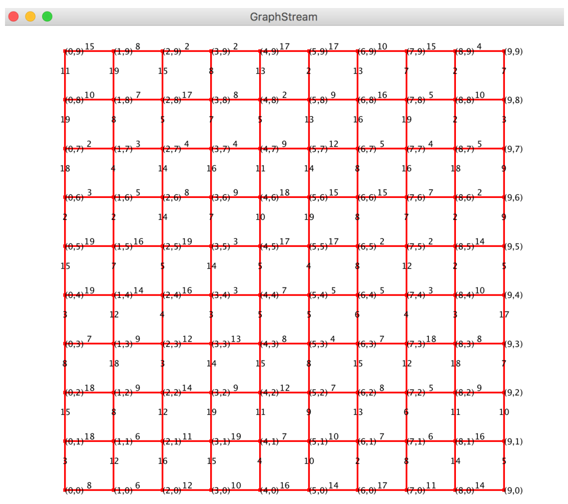


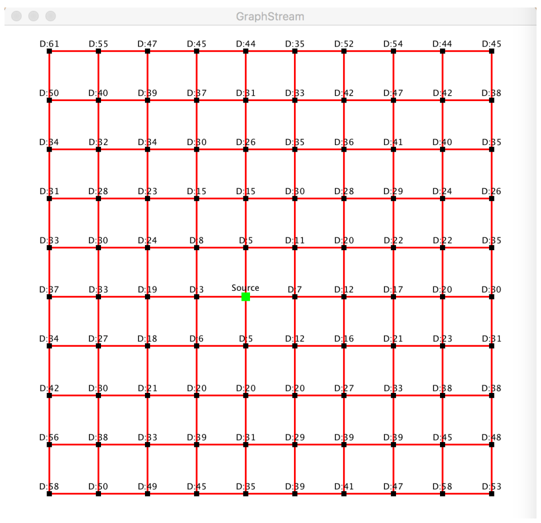
1.5 Exercise 4: Dijkstra

For the PriorityQueue, you can use an ArrayList and manage the ”distance” attribute for inserting elements in the ArrayList.

1. implement the Dijkstra algorithm
2. apply this algorithm starting from a randomly chosen vertex

Download the `gridvaluated_10_1:2.dgs` and `gridvaluated_10_2:20.dgs` for testing your algorithm and display the distance values on the vertices after the application of the `dijkstra` method.

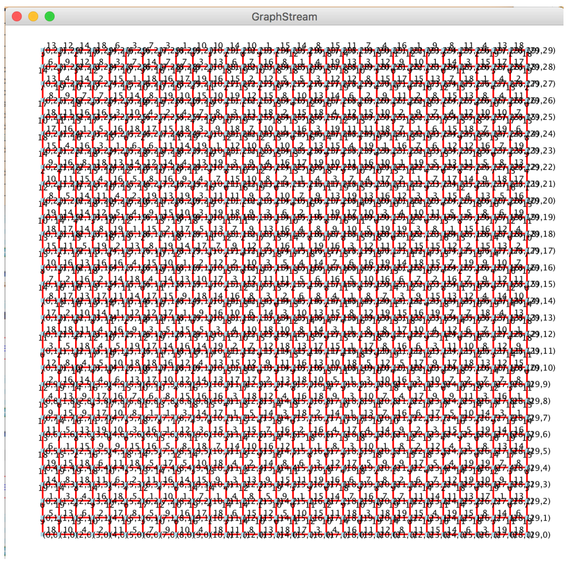


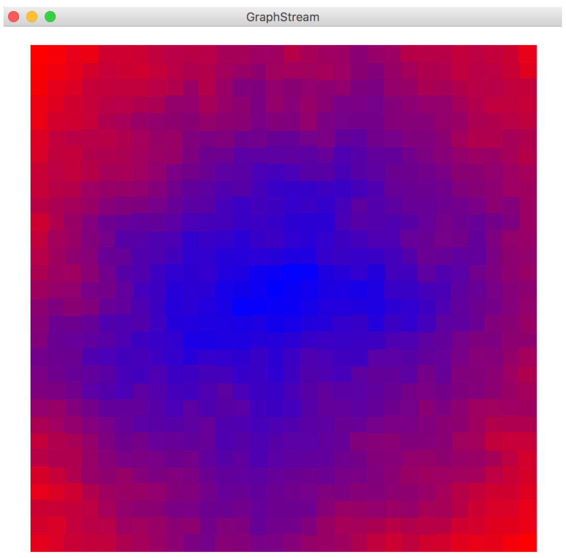


1.6 Exercise 5: Diameter and Radius

1. compute the eccentricity of every vertex and deduce the graph diameter and radius for the graph.
2. highlight the vertices such that the closer to the diameter their eccentricity value, the more red are the vertices, and the closer to the radius their eccentricty value, the more blue they are (as on the picture below).

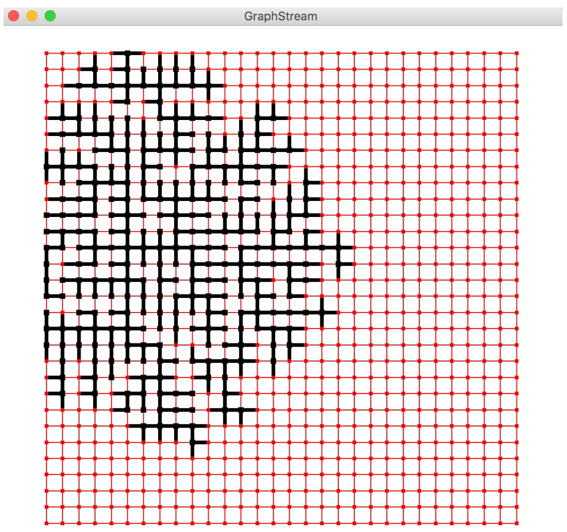
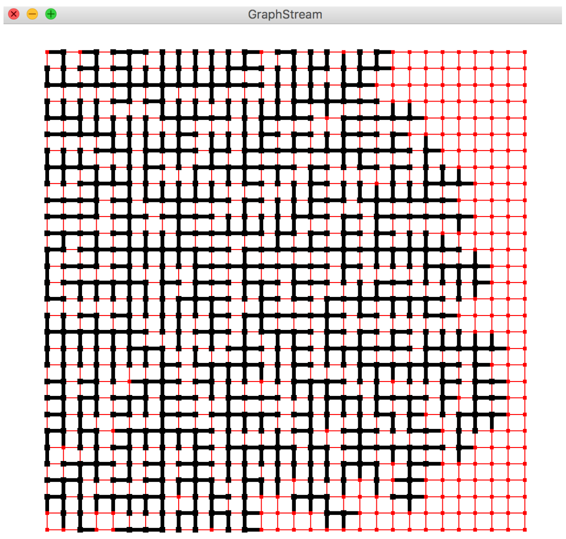
Download the `gridvaluated_30_1:20.dgs` for testing your algorithm.





1.7 Exercise 6: Spanning Trees

1. Using previous algorithms, highlight spanning trees build by the graph traversals


GraphStream

GraphStream