T-301-REIR, REIKNIRIT
HAUST 2019
D5 - GRAPHS

Problem 1. *You have managed to learn all friendship connections in the social network Basehook (Bh). You want to forward a message from one person to another, but have observed that the impact of a message depends a lot on along how many connections it has to travel. You therefore want to compute the following: Given a sequence of friendship connections and specific persons A and B, determine the smallest k such that there is a sequence of persons $X_0, X_1, X_2, \ldots, X_k$ where $A = X_0$, $B = X_k$, and X_i and X_{i+1} are Bh-friends, for $i = 0, 1, \ldots, k-1$. Describe briefly an efficient solution in words to this problem, using the concepts of this course, and give its time complexity.*

Solution. Form a (undirected, unweighted) graph from the friendship connections. BFS run from node A gives then the shortest distance to node B. The time complexity is linear in the number of persons and the total number of friendship connections.

Problem 2. *(Continued) You have studied this further and discovered that not all friendships are equal. Strong friendship connections are special, and using such friendship connections doesn't cost anything so to speak: the message can travel along arbitrarily many strong friendships connections without its impact becoming weaker. We call other connections weak. You want to compute the fewest number of weak friendship connections needed to route a message from given person A to person B (possibly using many strong connections).*

Solution. First solution: (Sort the friendships by strength in linear time using Counting sort) Run Connected Components on the graph formed by the strong connections, and store the nodes of each component in a list. This allows each node to know its component, as well as access to the other nodes in its component. Then run BFS on the graph formed by the weak connections, where each node is replaced by its component id.

Second solution: Modify BFS to maintain two queues: Q, with those at the same distance from A as the current node considered, and Q', those of distance that is one greater. When visiting a node, add the other nodes of its component to Q, but its other neighbors to Q'. We finish processing Q before turning to the nodes in Q'.

Problem 3. *Modify* `BreadthFirstPaths.java` *to compute the number of shortest paths between two given vertices v and w in a given digraph. This replaces the method* `boolean hasPathTo(inv v)` *with* `int nrOfPathsTo(inv v)`. Hint*: Note that if vertex w is of distance*

Date: October 11, 2019.

k from v, and it is adjacent to vertices a, b and c of distance $k-1$ from v, then each shortest $v-w$ path runs through one of a, b, and c.

Solution. See the following code:

```
private void bfs() {
    Queue<Integer> q = new Queue<Integer>();
    for (int v = 0; v < G.V(); v++) distTo[v] = INFINITY;
    distTo[s] = 0;
    marked[s] = true;
    countPaths[s] = 1;                                  /* NEW */
    q.enqueue(s);

    while (!q.isEmpty()) {
        int v = q.dequeue();
        for (int w : G.adj(v)) {
            if (!marked[w]) {
                distTo[w] = distTo[v] + 1;
                marked[w] = true;
                countPaths[w] = 1;                      /* NEW */
                q.enqueue(w);
            } else if (distTo[w] == distTo[v] + 1)      /* NEW */
                countPaths[w] += countPaths[v];         /* NEW */
        }
    }
}
```

Problem 4. *Suppose the weight of every edge in a weighted graph is decreased by one. Explain (in 20 words or less) why the MST computed by Kruskal does not change.*

Solution. This does not change the relative order between edges, and therefore not the decisions made by Kruskal's algorithm.

Problem 5. *Consider a graph with distinct edge weights such that Prim and Kruskal select the edges of the spanning tree T in* opposite *order. How must T look like?*

Solution. T will be a path, with the edge weights in reverse order of their distance from the source. This can be argued by induction.

Class Exercises

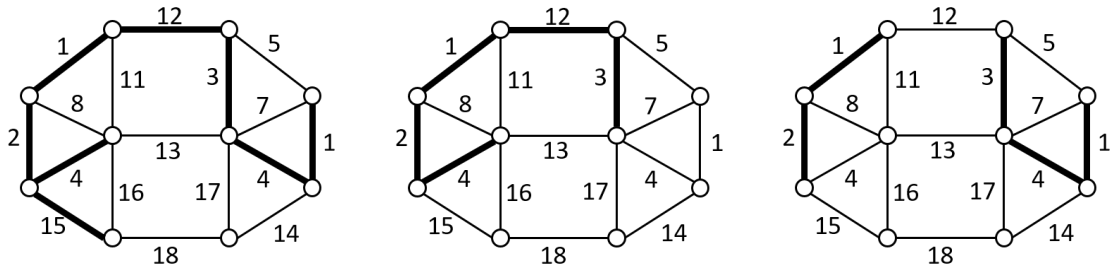
These questions will be addressed during exercise class. They are not to be turned in.

Problem 6. *Consider the graph given in* `t11.txt`*. a) Construct the adjacency list. b) Perform* `DepthFirstOrder`*, constructing the preorder, postorder, and the reverse postorder of the graph. c) Does the graph have a topological order? d) Find the strong components, found by* `KosarajuSCC`*.*

Solution. d) 789, 0125, 46, 10 (in topological order of the component graph

Problem 7. *Consider the weighted graph in* `mst.txt`*. a) Give the order in which Kruskal's method adds edges to the spanning tree. b) Give the order in which Prim's method adds edges to the spanning tree.*

Problem 8. *In each of the three graphs, indicate whether the black edges could constitute a partial execution of Prim's algorithm, Kruskal's algorithm, both, or neither.*



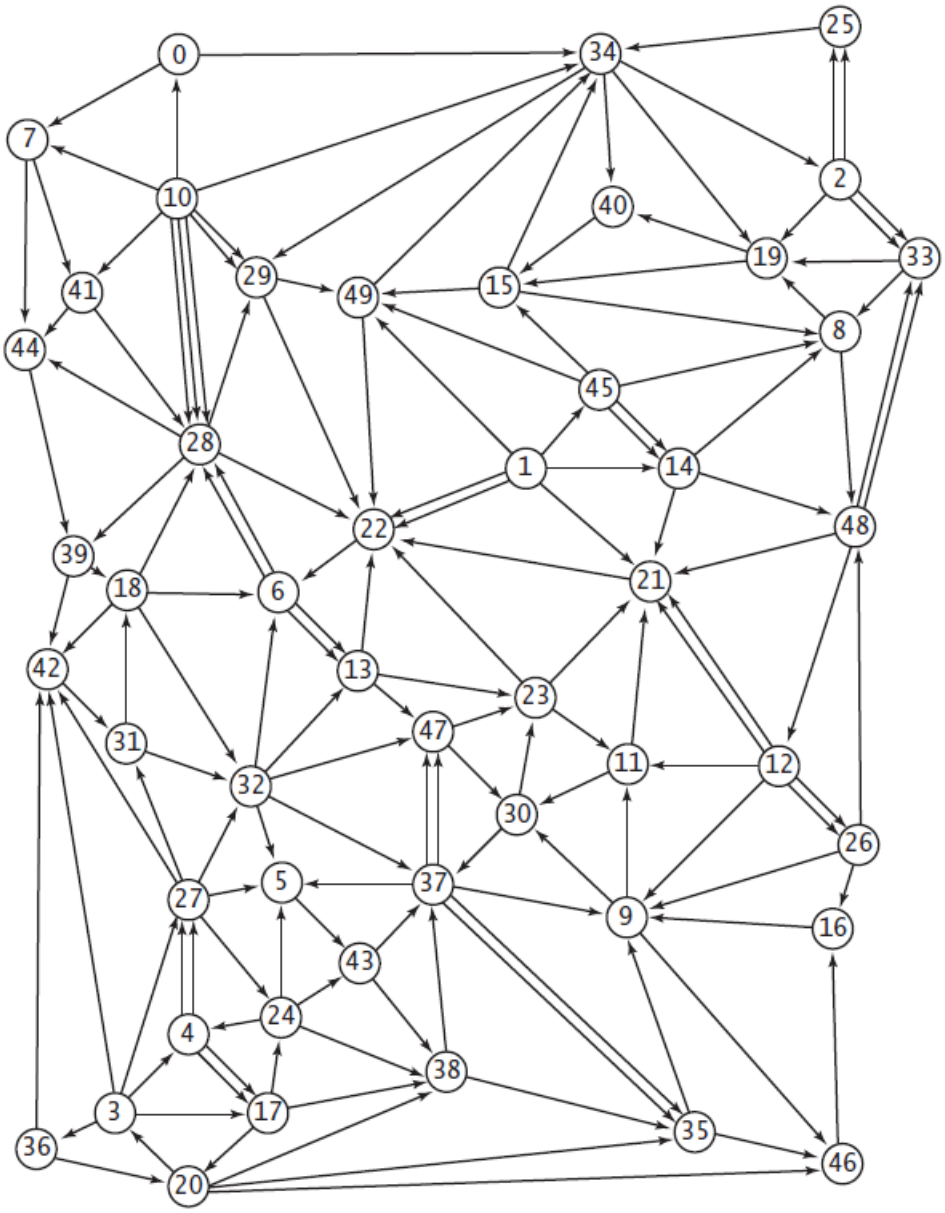
Solution. a) Both, b) Prim only, c) Kruskal only

Problem 9. *Prove carefully that the following algorithm finds an MST: Start with a graph containing all the edges; repeatedly go through the edges in decreasing order of weight; for each edge, check if deleting that edge will disconnect the graph; if not, delete it.*

Solution. Ein leið er að nota skurðareiginleikann og beita honum á hvern legg sem var valinn af aðferðinni inn í tréð. Fyrst leggurinn var valinn, þá var hann síðasti leggurinn sem var eftir yfir einhvern skurð, en þar sem aðeins dýrari leggjum hafði verið eytt, var hann ódýrasti leggurinn yfir skurðinn. Þannig finnum við að hver leggur sem var skilinn eftir var ódýrasti leggurinn yfir skurð, og á því heima í MST.

Önnur leið væri að rökstyðja að aðferð Kruskals velur aldrei dýrasta legg í neinni rás. Aðferðin að ofan hendir út aðeins leggjum sem eru þeir dýrustu í einhverri rás.

Problem 10. *Try to find the strong components of the following graph (*`mediumDG.txt`*, p.591). (You may run the program, but first explore by hand.)*



School of Computer Science, Reykjavik University, Menntavegi 1, 101 Reykjavik

Email address: mmh@ru.is