Practice 4

Exercise levels:
(L1) Reproduce: Reproduce basic facts or check basic understanding.
(L2) Apply: Follow step-by-step instructions.
(L3) Reason: Show insight using a combination of different concepts.
(L4) Create: Prove a non-trivial statement or create an algorithm or data structure of which the objective is formally stated.

▶ Lecture 10 Union-Find

Exercise 1 *(L2)* Consider the elements $\{x_1, x_2, \ldots, x_8\}$. We call Make-Set($x$) once for each element. Show the status of the data structure after performing each of the following instructions in sequence: Union(x_1, x_2), Union(x_3, x_4), Union(x_3, x_5), Union(x_1, x_7), Union(x_3, x_6), Union(x_1, x_3), Union(x_1, x_8) ...

(a) ...when using Union-by-size.

(b) ...when using Union-by-rank with path compression.

Exercise 2

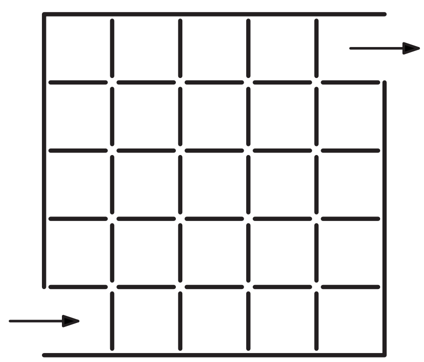
(a) *(L1)* What is the maximum number of Union operations performed during Union-Find?

(b) *(L3)* Consider union-by-size. Let $S = \{x_1, x_2, \ldots, x_{n-1}, x_n\}$ be a set of $n = 2^k$ elements, for $k \in \mathbb{N}$. Describe a sequence of m operations on S (of which n are Make-Set) that achieves the worst-case runtime of $\Theta(m + n \log n)$.

(c) *(L3)* Consider union-by-rank with path compression. What is the maximum height of a node with rank k? What is the minimum height?

(d) *(L3)* Explain, in your own words, why there are at most $\log^* n$ rank groups when using union-by-rank, when $n \geq 1$.

Exercise 3 Let G be a grid of k cells by k cells. Between every two horizontally or vertically adjacent cells there is a wall separating the cells. On the outside there is a wall encompassing the full grid that has two openings (see figure).



We are going to build a maze using the following algorithm:

```
MAZE(G)
for each cell c
    MAKE-SET(c)
for all internal walls w in an arbitrary order
    let c and d be the cells on both sides of w
    if FIND-SET(c) ≠ FIND-SET(d)
        remove wall w
        UNION(c, d)
```

(a) *(L3)* What does it mean when during execution two cells are in the same set?

(b) *(L2)* What is the running time of the algorithm in terms of k when using Union-Find and the implementation of solution 1 from the lectures?

(c) *(L2)* What is the running time of the algorithm in terms of k when using union-by-rank with path compression?

▶ Lecture 11 Elementary graph algorithms

Exercise 4 Consider the following two adjacency matrices:

$$
(1)\quad \begin{array}{c|cccccc|}
 & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline
1 & 0 & 1 & 0 & 0 & 1 & 0 \\
2 & 1 & 0 & 1 & 0 & 0 & 1 \\
3 & 0 & 1 & 0 & 0 & 0 & 1 \\
4 & 0 & 0 & 0 & 0 & 1 & 0 \\
5 & 1 & 0 & 0 & 1 & 0 & 1 \\
6 & 0 & 1 & 1 & 0 & 1 & 0 \\ \hline
\end{array}
\qquad
(2)\quad \begin{array}{c|cccccc|}
 & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline
1 & 0 & 1 & 0 & 0 & 0 & 0 \\
2 & 1 & 0 & 0 & 0 & 0 & 0 \\
3 & 0 & 1 & 0 & 0 & 0 & 1 \\
4 & 1 & 0 & 0 & 1 & 1 & 0 \\
5 & 1 & 0 & 0 & 1 & 0 & 1 \\
6 & 0 & 1 & 1 & 0 & 0 & 0 \\ \hline
\end{array}
$$

For each, answer the following questions:

(a) *(L2)* Can this matrix be the adjacency matrix of an undirected graph? Could it represent a directed graph? Why or why not?

(b) *(L2)* For both adjacency matrices draw the corresponding undirected graph if possible, otherwise draw the corresponding directed graph.

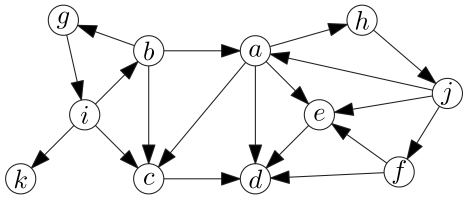
(c) *(L2)* Give the adjacency list representation for both graphs.

Exercise 5 *(L3)* Draw an undirected graph that has 12 edges and at least 6 vertices. Six of the vertices have to have degree exactly 3, all other vertices have to have degree at most 2. Use as few vertices as possible.

Exercise 6 *(L3)* Let G be an undirected graph on the vertices $\{a, b, c, d, e, f, g, h\}$. We do not know which edges are in G but we know that Breadth First Search (BFS), when starting at node a, visits the following edges of the graph in the given order: (a, b), (a, c), (b, h), (c, f), (c, g), (c, e), (c, d). Can the following edges be in the graph G?

1: (b, d), 2: (d, e), 3: (d, h), 4: (a, f)

Exercise 7 *(L2)* Consider the following directed graph:



Perform a Depth-First-Search (DFS) from node a. Whenever there are multiple candidate nodes that could be visited next, choose the one with the smallest label in lexicographic order. Determine all tree edges, back edges, forward edges, and cross edges.

Exercise 8 Consider the following algorithm on an unrooted tree T. (An unrooted tree has no designated root node.) Iteratively remove all the nodes with degree at most 1 until the tree is empty.

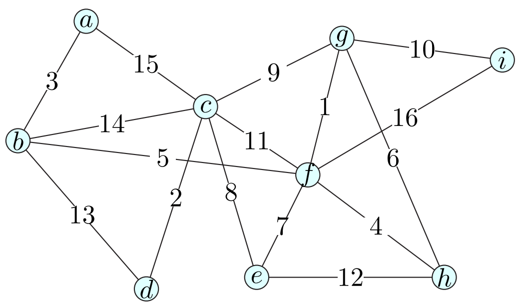
(a) *(L3)* How many nodes could have been removed in the last step? Justify your answer.

(b) *(L4)* The nodes that are removed in the last step are called the *center nodes* of the tree. Assume there is unique center point c. Prove that for this node the longest distance to any other node in the tree is minimized. That is, for any non-center node a, it holds $\max_v d(a, v) > \max_w d(c, w)$, where $d(x, y)$ is the distance between the nodes x and y.

▶ Lecture 12 Minimum spanning trees

Exercise 9

(a) *(L2)* Draw the selected edges after each step of Kruskal's algorithm for the following graph. Do the same for Prim's algorithm when starting from node a.



(b) *(L2)* For each of the edges (a, b), (c, e), (b, f), give a cut of G where the edge is a light edge.

Exercise 10 *(L3)* Let T be a minimum spanning tree of a weighted graph $G = (V, E)$ with distinct positive edge weights. We make a new graph G' that is equal to G but where the weight for each edge $e \in E$ is defined as:

(a) $w_{new}(e) = 2 \cdot w(e)$

(b) $w_{new}(e) = w(e) + 1$

(c) $w_{new}(e) = 1/w(e)$

Is T also a minimum spanning tree for G'? Prove T is a minimum spanning tree or give a counterexample to disprove it.
Note: to prove the statement holds it must be true for any (connected) graph G with any (distinct, positive) weights on the edges. To disprove the statement, a single example suffices.

Exercise 11 Given a graph $G = (V, E)$ and its minimum spanning tree T.

(a) *(L2)* We increase the weight of one edge $e \in E$. Can the minimum spanning tree of G change if e was part of the MST? What if e was not part of the MST? Argue your answers.

(b) *(L3)* You are given a graph $G = (V, E)$, a minimum spanning tree T of G, and an edge $e \in T$. Consider the graph G' obtained from G by removing edge e. That is $G' = (V, E \setminus \{e\})$. Assume G' is still connected. Let $(S, V - S)$ be a cut that respects $T - \{e\}$, and let e' be a light edge crossing $(S, V - S)$ in G'.

Prove that $T' = T - \{e\} + \{e'\}$ is a minimum spanning tree of G'.