Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

Departement of Computer Science
Markus Püschel, David Steurer
Johannes Lengler, Gleb Novikov, Chris Wendler, Ulysse Schaller

16. November 2020

# Algorithms & Data Structures — Exercise sheet 9 — HS 20

Exercise Class (Room & TA): _____

Submitted by: _____

Peer Feedback by: _____

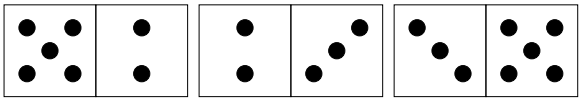
Points: ____

**Submission:** On Monday, 23 November 2020, hand in your solution to your TA *before* the exercise class starts. Exercises that are marked by * are challenge exercises. They do not count towards bonus points.

### Exercise 9.1 *Domino.*

a) A domino set consists of all possible $\binom{6}{2} + 6 = 21$ different tiles of the form $[x|y]$, where $x$ and $y$ are numbers from $\{1, 2, 3, 4, 5, 6\}$. The tiles are symmetric, so $[x|y]$ and $[y|x]$ is the same tile and appears only once.

Show that it is impossible to form a line of all 21 tiles such that the adjacent numbers of any consecutive tiles coincide.



b) What happens if we replace 6 by an arbitrary $n \geq 2$? For which $n$ is it possible to line up all $\binom{n}{2} + n$ different tiles along a line?

### Exercise 9.2 *Post-Corona Party* **(1 point)**.

It is 2021, we have multiple Corona vaccines and the pandemic is over. Imagine you go to a party with your friends. Assume that people shake hands with each other to introduce themselves.

Prove that at each party with at least two participants there are two people that shook hands with the same number of people.

***Hint:*** *Model the problem as a problem defined on a graph. Define the set of vertices and the set of edges in words.*

### Exercise 9.3 *Graph connectivity* **(2 points)**.

In this exercise, you will need to prove or find counterexamples to some statements about the connectivity of graphs. We first need to introduce/recall a few definitions.

**Definition 1.** A *cycle* is a sequence of vertices $v_1, \ldots, v_k, v_{k+1}$ with $k \geq 3$ such that all $v_1, \ldots, v_k$ are distinct, $v_1 = v_{k+1}$ and such that any two consecutive vertices are adjacent.

**Definition 2.** A graph is *connected* if there is a walk between every pair of vertices. It is called *disconnected* otherwise.

**Definition 3.** A vertex $v$ in a connected graph is called a *cut vertex* (or *articulation point*) if the subgraph obtained by removing $v$ (and all its incident edges) is disconnected.

**Definition 4.** An edge $e$ in a connected graph is called a *cut edge* (or *bridge*) if the subgraph obtained by removing $e$ (but keeping all the vertices) is disconnected.

In the following, we always assume that the original graph is connected. Prove or find a counterexample to the following statements:

a) If a vertex $v$ is part of a cycle, then it is not a cut vertex.

b) If a vertex $v$ is not a cut vertex, then $v$ must be part of a cycle.

c) If an edge $e$ is part of a cycle (i.e. $e$ connects two consecutive vertices in a cycle), then it is not a cut edge.

d) If an edge $e$ is not a cut edge, then $e$ must be part of a cycle.

e) If $u$ and $v$ are two adjacent cut vertices, then the edge $e = \{u, v\}$ is a cut edge.

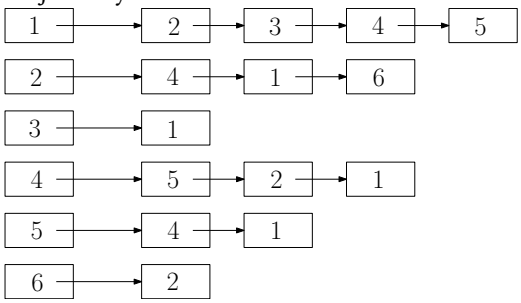
f) If $e = \{u, v\}$ is a cut edge, then $u$ and $v$ are cut vertices. What if we add the condition that $u$ and $v$ have degree at least 2 ?

**Exercise 9.4** *Data structures for graphs.*

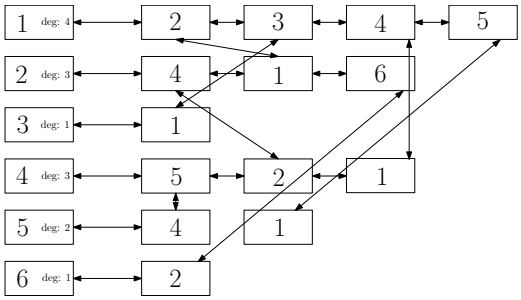
Consider three types of data structures for storing a graph $G$ with $n$ vertices and $m$ edges:

a) Adjacency matrix.

b) Adjacency lists:




c) Adjacency lists, and additionally we store the degree of each node, and there are pointers between the two occurences of each edge. (An edge appears in the adjacency list of each endpoint).




For each of the above data structures, what is the required memory (in Θ-Notation)?

Which runtime (worst case, in Θ-Notation) do we have for the following queries? Give your answer depending on $n$, $m$, and/or $\deg(u)$ and $\deg(v)$ (if applicable).

(i) Input: A vertex $v \in V$. Find $\deg(v)$.

(ii) Input: A vertex $v \in V$. Find a neighbour of $v$ (if a neighbour exists).

(iii) Input: Two vertices $u, v \in V$. Decide whether $u$ and $v$ are adjacent.

(iv) Input: Two adjacent vertices $u, v \in V$. Delete the edge $e = \{u, v\}$ from the graph.

(v) Input: A vertex $u \in V$. Find a neighbor $v \in V$ of $u$ and delete the edge $\{u, v\}$ from the graph.

(vi) Input: Two vertices $u, v \in V$ with $u \neq v$. Insert an edge $\{u, v\}$ into the graph if it does not exist yet. Otherwise do nothing.

(vii) Input: A vertex $v \in V$. Delete $v$ and all incident edges from the graph.

For the last two queries, describe your algorithm.

**Exercise 9.5** *Walks in a graph.*

Recall that a *walk of length* $k$ in a graph $G = (V, E)$ is a sequence of vertices $v_0, v_1, \ldots, v_k$ such that for $1 \leq i \leq k$ there is an edge $\{v_{i-1}, v_i\} \in E$.

Assume that you are given a graph $G = (V, E)$, with $V = \{v_1, \ldots, v_n\}$, and a matrix $M$ that contains for every pair $v_i, v_j \in V$ in the entry $M_{ij}$ the number of walks of length $k$ from $v_i$ to $v_j$. For every pair $v_k, v_\ell \in V$, find a formula for computing the number of walks of length $k + 1$ from $v_k$ to $v_\ell$ from that data.