Serie 8

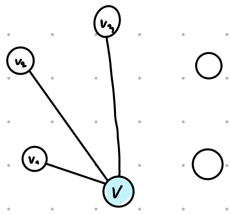
Exercise 8.1 - Introduction to graphs

In this exercise, we want to prove the following statement: Among any six people, there are either three that all know each other or three that all do not know each other (or both). We assume that this relation is symmetric, so if person A knows person B, then also B knows A. We model the problem as a graph. We define $G = (V, E)$ to be a graph on 6 vertices, where the vertices correspond to the six people and two people are connected by an edge if they know each other.

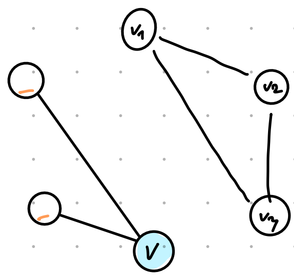
(a) Prove the above statement, i.e. that in every possible graph on 6 vertices, there are three vertices that are all pairwise adjacent or there are three vertices that are all pairwise not adjacent.
Hint: *Start with one vertex and notice that this vertex is either adjacent to (at least) three vertices or not adjacent to (at least) three vertices.*

(a)

Fall 1: $\deg(v) \geq 3$

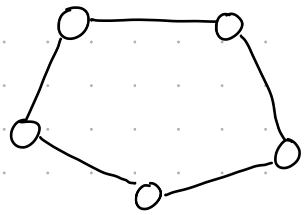


Fall 2: $\deg(v) \leq 2$



(b) Is the statement also true for five people? In other words, does the following hold: For any graph $G = (V, E)$ with 5 vertices, there are either three vertices that are all pairwise adjacent or there are three vertices that are all pairwise not adjacent (or both). Provide a proof or a counterexample.

(b)



Exercise 8.5 - Short questions about graphs

In the following, let $G = (V, E)$ be a graph, $n = |V|$ and $m = |E|$.

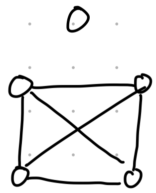
(a) Let $v \neq w \in V$. Prove that if there is a walk with endpoints v and w, then there is a path with endpoints v and w.

v, w $\quad W: \langle v = v_0, v_1, \dots, v_k = w \rangle$, weil $v \neq w$ $k \geq 1$.

$v_i = v_j$ $\quad 0 \leq i < j \leq k$ $\quad \Rightarrow \quad W': \langle v = v_0, v_1, \dots, v_i = v_j, v_{j+1}, \dots, v_k = w \rangle$ ein Weg zw. v und w

Länge $k - (j-i) < k$

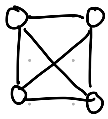
(b) Every graph with $m \geq n$ is connected.



$m = 6$

$n = 5$

(c) If G contains a Hamiltonian path, then G contains a Eulerian walk.

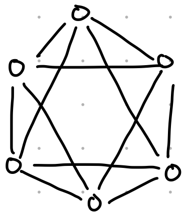


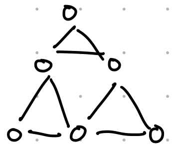
(d) If every vertex of a non-empty graph G has degree at least 2, then G contains a cycle.

G enthält keinen Kreis (Annahme). $\Rightarrow$ Jede ZHK Baum $\Rightarrow$ $|E| \leq n-1$

Jeder Knoten hat Grad mindestens 2 $\Rightarrow$ G hat mindestens $(2 \cdot n)/2 = n$ Kanten (Widerspruch)

(e) Suppose in a graph G every pair of vertices v, w has a common neighbour (i.e., for all distinct vertices v, w, there is a vertex x such that $\{v, x\}$ and $\{w, x\}$ are both edges). Then there exists a vertex p in G which is a neighbour of every other vertex in G (i.e., p has degree $n - 1$).





(f) Let G be a connected graph with at least 3 vertices. Suppose there exists a vertex v_{cut} in G so that after deleting v_{cut}, G is no longer connected. Then G does not have a Hamiltonian cycle. (Deleting a vertex v means that we remove v and any edge containing v from the graph).

Annahme: Sei G ein Graph mit Hamiltonkreis $\langle v_1, v_2, v_3, \dots, v_n, v_1 \rangle$. Sei $v_1 = v_{cut}$

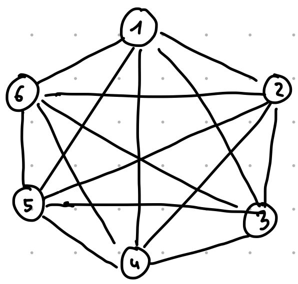
Wenn wir $v_1 = v_{cut}$ aus G entfernen $\Rightarrow$ G' enthält Hamiltonpfad, nämlich $\langle v_2, v_3, \dots, v_n \rangle$.

$\Rightarrow$ G' ist zusammenhängend $\Rightarrow$ Widerspruch

Exercise 8.2 - 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?

deg(v) soll gerade sein

Ungerades n $\rightarrow$ deg(v) ist gerade $\forall v \in V$ $\Rightarrow$ es existiert Eulerweg

+ falls $n = 2$

Exercise 8.3 - Star search reloaded

A *star* in an undirected graph $G = (V, E)$ is a vertex that is adjacent to all other vertices. More formally, $v \in V$ is a star if and only if $\{\{v, w\} \mid w \in V \setminus \{v\}\} \subseteq E$.

In this exercise, we want to find a star in a graph G by walking through it. Initially, we are located at some vertex $v_0 \in V$. Each vertex has an associated flag (a Boolean) that is initially set to `False`. We have access to the following constant-time operations:

- `countNeighbors()` returns the number of neighbors of the current vertex
- `moveTo(`i`)` moves us to the ith neighbor of the current vertex, where $i \in \{1..$`countNeighbors()`$\}$
- `setFlag()` sets the flag of the current vertex to `True`
- `isSet()` returns the value of the flag of the current vertex
- `undo()` undoes the latest action performed (the movement or the setting of last flag)

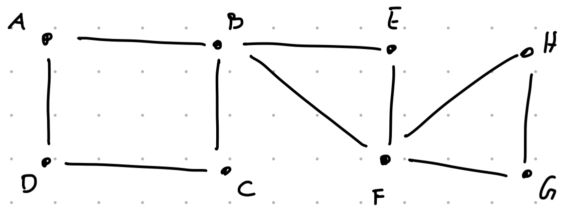
Assume that G has exactly one star and $|G| = n$. Give the pseudocode of an algorithm that finds the star, i.e., your algorithm should always terminate in a configuration where the current vertex is a star in G. Your algorithm must have complexity $O(|V| + |E|)$, and must not introduce any additional datastructures (no sets, no lists etc.). Show that your algorithm is correct and prove its complexity. The behavior of your algorithm on graphs that do not contain a star or contain several stars can be disregarded.

wenn v.countNeighbors() == n-1 => v ist Star

Vorlesung Recap

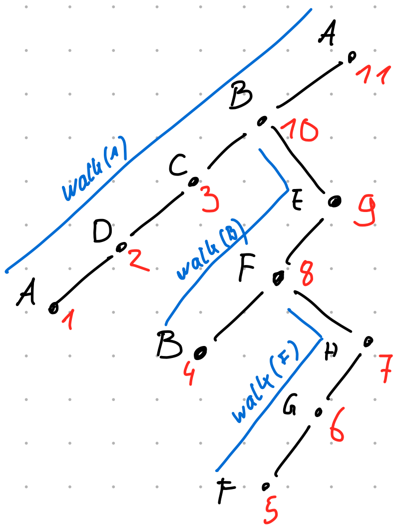
- Eulerweg ist ein Weg, der jede Kante genau 1 mal benutzt
  - -> 1. Alle Kanten in einer ZHK
  - -> deg(v) muss gerade sein bei allen Knoten ausser 2
- Eulerzyklus ist ein Zyklus, der -II-
  - -> 1. Alle Kanten in einer ZHK
  - -> deg(v) muss gerade sein bei allen Knoten
- Hamiltonpfad ist ein Pfad, der jeden Knoten durchläuft
- Hamiltonkreis ist -II-

Eulerzyklus finden



Euler (G) :

- Leere Liste Z
- alle Kanten unmarkiert
- Euler Walk (u_0) für jeden Knoten
- return Z



Euler Walk (u):

for $uv \in E$, nicht markiert
  markiere Kante uv
  EulerWalk (v)
$z \leftarrow u$

Graph Definitionen

Ungerichteter Graph: $G = (V, E)$, $V = \{v_1, \dots, v_n\}$, $E \subseteq \{\{u,v\} \mid u,v \in V \; u \neq v\}$

$\{u,v\} = \{v,u\}$

Gerichteter Graph: $G = (V, E)$, $V = \{v_1, \dots, v_n\}$, $E \subseteq V \times V$

$(u,v) \neq (v,u)$

v Nachfolger von u ist

u Vorgänger von v ist

v adjazent zu u: Ungerichtet: $\Leftrightarrow \{u,v\} \in E$

Gerichtet: $(u,v) \in E$

Nachbarschaft von v: Ungerichtet: $\mathcal{N}(v) = \{u \in V \mid \{u,v\} \in E\}$

Gerichtet: $\mathcal{N}^-(v) = \mathcal{N}_{in}(v) = \{u \in V \mid (u,v) \in E\}$

$\mathcal{N}^+(v) = \mathcal{N}_{out}(v) = \{u \in V \mid (v,u) \in E\}$

Grad von v:

Ungerichtet: $\deg(v) = |\mathcal{N}(v)|$

Gerichtet: $\deg_{in}(v) = \deg^-(v) = |\mathcal{N}_{in}(v)|$

$\deg_{out}(v) = |\mathcal{N}_{out}(v)|$

Adj. Matrix $(A)_{uv} = 1 \Leftrightarrow (u,v) \in E$

Adj. Liste Adj[u] gibt Liste aller <u>Nachfolger</u> von u aus

Topologische Sortierung

$\exists$ topologische Sort. $\Leftrightarrow$ $\nexists$ gerichteter Zyklus

<u>visit(u):</u>

```
markiere u
for nachfolger v, unmarkiert:
    visit(v)
Füge u zur top. Sort. hinzu
```

<u>DFS(G):</u>

```
alle Knoten unmarkiert
for u0 ∈ V unmarkiert
    visit(u0)
```

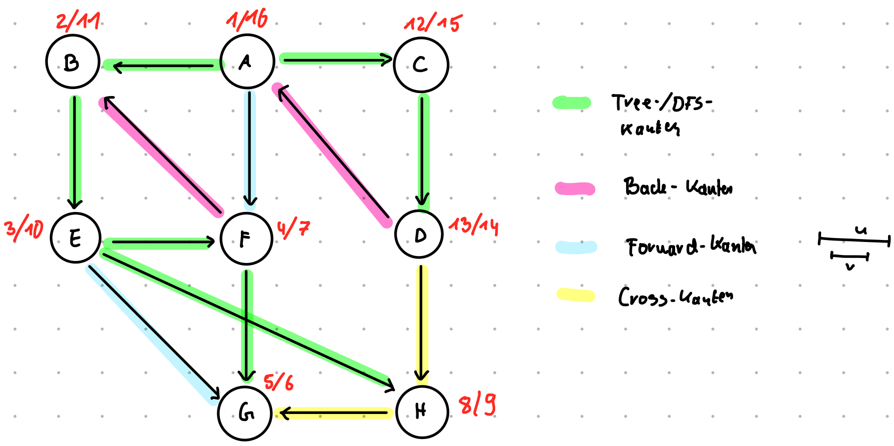
DFS

visit(u):

pre[u] ← T ; T++
markiere u
for nachfolger v, unmarkiert:
    visit(v)
post[u] ← T ; T++

DFS(G):

T ← 1
alle Knoten unmarkiert
for $u_0 \in V$ unmarkiert
    visit(u_0)



- Falls G azyklisch, dann ist umgekehrt post-order eine top. Sort.

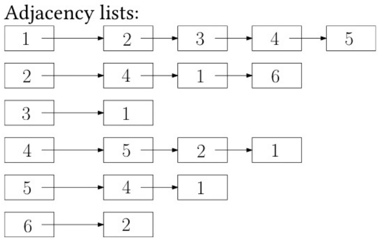
- $\exists$ back Kante $\Rightarrow$ $\exists$ gerichteter Zyklus

Exercise: Data structures for graphs

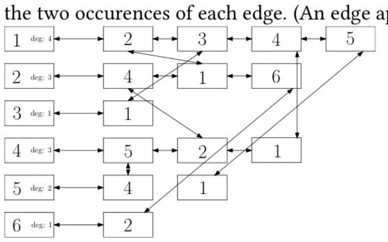
Consider three types of data structures for storing a graph G with n vertices and m edges:

a) Adjacency matrix. $\quad (A)_{uv} = 1 \iff (u,v) \in E$

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).

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

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

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

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

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

(f) 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.

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

For the last two queries, describe your algorithm.

	adjacency matrix	adjacency lists	improved adjacency lists
space	$\Theta(n^2)$	$\Theta(n+m)$	$\Theta(n+m)$
$\deg(v)$	$\Theta(n)$	$\Theta(1+\deg(v))$	$\Theta(1)$
find $u \in \mathcal{N}(v)$	$\Theta(n)$	$\Theta(1)$	$\Theta(1)$
$\{u,v\} \overset{?}{\in} E$	$\Theta(1)$	$\Theta(1+\min\{\deg(u),\deg(v)\})$	$\Theta(1+\min\{\deg(u),\deg(v)\})$
delete $\{u,v\}$	$\Theta(1)$	$\Theta(1+\deg(u)+\deg(v))$	$\Theta(1+\min\{\deg(u),\deg(v)\})$
find $u \in \mathcal{N}(v)$ and delete $\{u,v\}$	$\Theta(n)$	$\Theta(1+\deg(u))$	$\Theta(1)$
Insert $\{u,v\}$	$\Theta(1)$	$\Theta(1+\min\{\deg(u),\deg(v)\})$	$\Theta(1+\min\{\deg(u),\deg(v)\})$
delete v	$\Theta(n^2)$ $\Theta(n)$	$\Theta(n+m)$	$\Theta(1+\deg(v)\ (+n))$