Algorithms and Data Structures

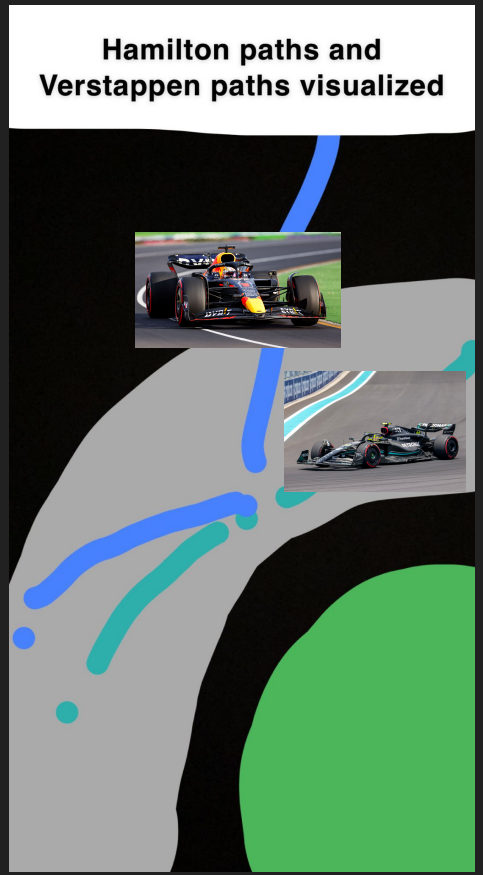
Exercise Session 8

https://n.ethz.ch/~ahmala/and

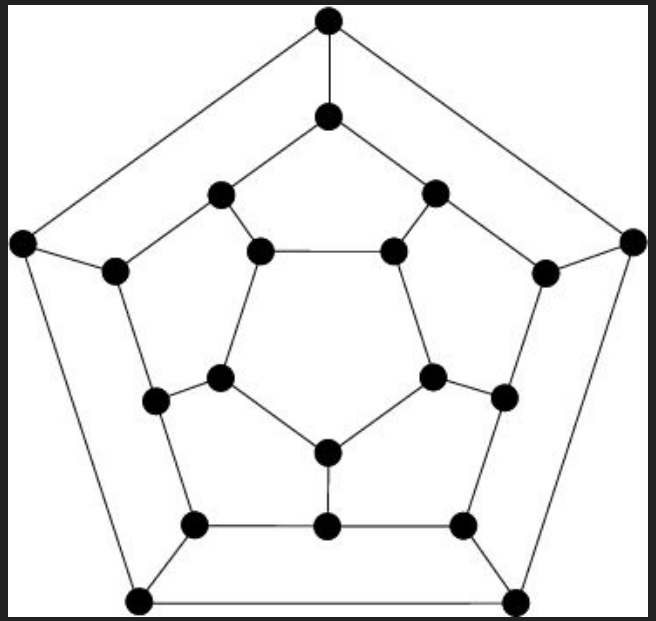
Definition 1. Let $G = (V, E)$ be a graph.

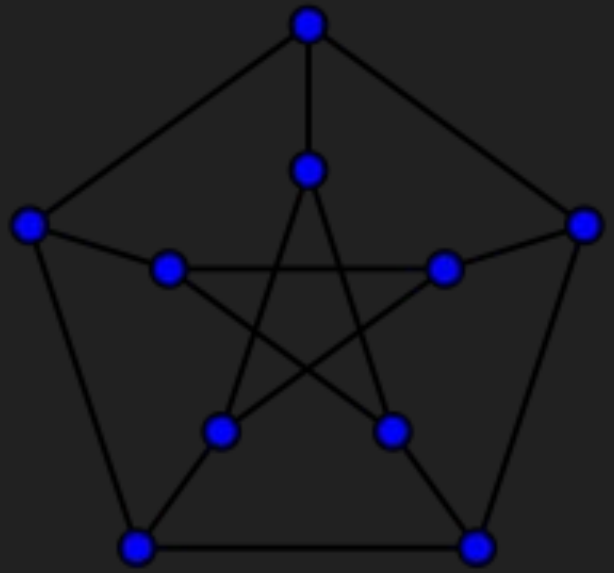
- A sequence of vertices $(v_0, v_1, \ldots, v_k)$ (with $v_i \in V$ for all i) is a **walk** (german "Weg") if $\{v_i, v_{i+1}\}$ is an edge for each $0 \leq i \leq k-1$. We say that v_0 and v_k are the **endpoints** (german "Startknoten" and "Endknoten") of the walk.
- A sequence of vertices $(v_0, v_1, \ldots, v_k)$ is a **closed walk** (german "Zyklus") if it is a walk, $k \geq 2$ and $v_0 = v_k$.
- A sequence of vertices $(v_0, v_1, \ldots, v_k)$ is a **path** (german "Pfad") if it is a walk and all vertices are distinct (i.e., $v_i \neq v_j$ for $0 \leq i < j \leq k$).
- A sequence of vertices $(v_0, v_1, \ldots, v_k)$ is a **cycle** (german "Kreis") if it is a closed walk, $k \geq 3$ and all vertices (except v_0 and v_k) are distinct.

- A Eulerian walk (german “Eulerweg”) is a walk that contains every edge exactly once.
- A Hamiltonian path (german “Hamiltonweg”) is a path that contains every vertex.
- A Hamiltonian cycle (german “Hamiltonkreis”) is a cycle that contains every vertex.

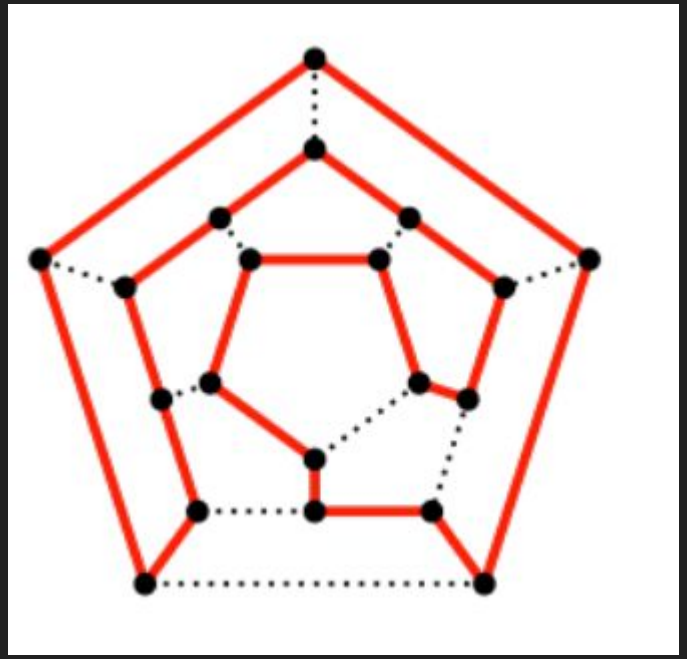


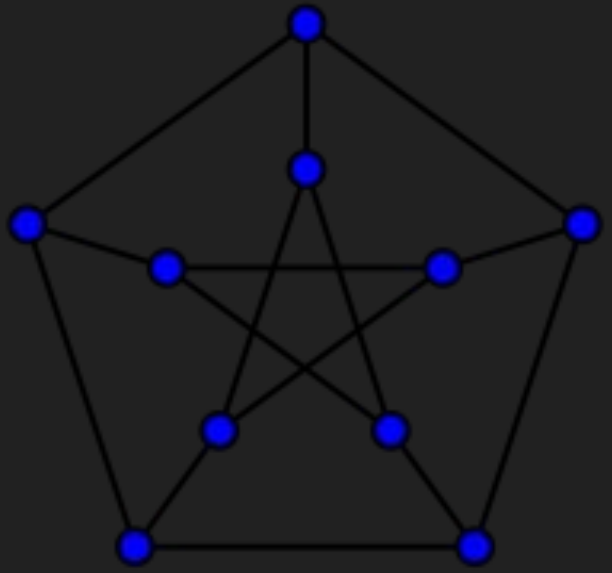
Do they have Hamiltonian Cycle?





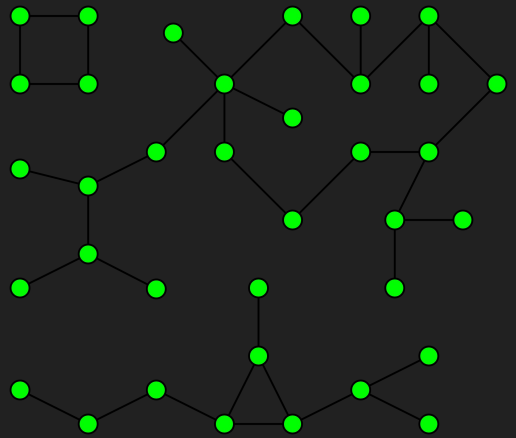
Do they have Hamiltonian Cycle?

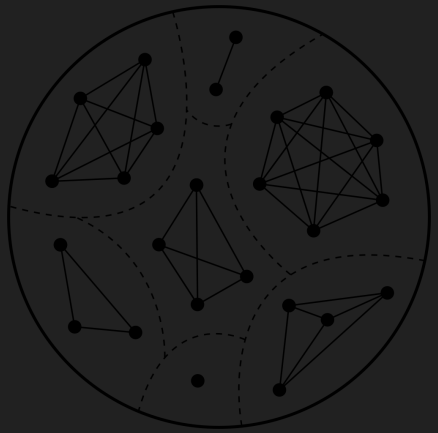




It hasn't

- A graph G is connected (german “zusammenhängend”) if for every two vertices u, v ∈ V there exists a path with endpoints u and
- A graph G is a tree (german “Baum”) if it is connected and has no cycles.





No multigraphs and self loops in AnD

G = (V,E) is a graph with |V| >= 1 Nodes. Following Expressions are equivalent!

a) G is a tree
b) G is connected with no cycles
c) G is connected with |E| = |V| - 1
d) G has no cycles and |E| = |V| - 1
e) For all x,y in V: G has exactly one x-y-Path

More Terminology-1

- Incident: connected(vertex&edge)
- Adjacent: neighboring(vertex&vertex)
- connected/undirected
- Acyclic: no cycles

More Terminology-2

- DAG(this week’s exercise sheet)
- Indegree
- Outdegree
- Forest
- reachable

Exercise 8.1 *Introduction to graphs* **(1 point)**.

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.*

Exercise 8.1 *Introduction to graphs* **(1 point)**.

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.

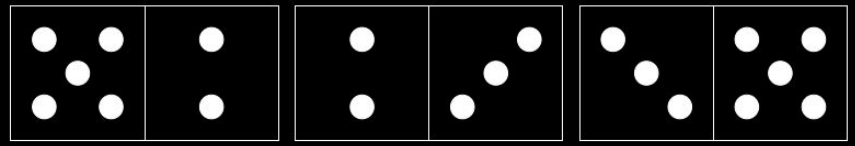
Hint: *Start with one vertex and notice that this vertex is either adjacent to (at least) three vertices or not adjacent to (at least) three vertices.*

Distinguish two cases deg(v)>=3 and deg(v)<=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.

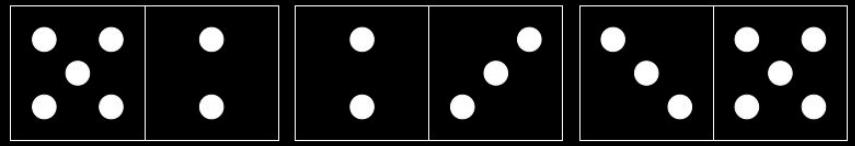
(a) A domino set consists of all possible $\binom{6}{2} + 6 = 21$ different tiles of the form $[x|y]$, where x and y are numbers from $\{1, 2, 3, 4, 5, 6\}$. The tiles are symmetric, so $[x|y]$ and $[y|x]$ is the same tile and appears only once.

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



(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 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.

Example Algorithm

In each iteration:

1. Algorithm terminates
2. Number of vertices to be explored decreases by exactly one
3. current vertex has no unmarked neighbors and we loop forever on this vertex

Algorithm 1 Star-finding algorithm

```
while countNeighbors() ≠ n − 1 do
    setFlag()
    for i = 1 to countNeighbors() do
        moveTo(i)
        if isSet() then
            undo()
        else
            break
```

Exercise 8.4 *Introduction to Trees.*

In this exercise the goal is to prove a few basic properties of trees.

(a) A **leaf** is a vertex with degree 1. Prove that in every tree G with at least two vertices there exists a leaf.

Hint: *Consider the longest path in G. Prove that its endpoint is a leaf.*

(b) Prove that every tree with n vertices has exactly $n - 1$ edges.

Hint: *Prove the statement by using induction on n. In the induction step, use part (a) to find a leaf.*

Disconnect the leaf from the tree and argue that the remaining subgraph is also a tree. Apply the induction hypothesis and conclude.

(c) Prove that a graph with n vertices is a tree if and only if it has $n - 1$ edges and is connected.

Hint: *One direction is immediate by part (b). For the other direction (every connected graph with $n-1$ edges is a tree), use induction on n. First, prove there always exists a leaf by considering the average degree. Then, disconnect the leaf from the graph and argue that the remaining graph is still connected and has exactly one edge less. Apply the induction hypothesis and conclude.*

(d) Write the pseudocode of an algorithm that is given a graph G as input and checks whether G is a tree.

As input, you can assume that the algorithm has access to the number of vertices n, the number of edges m, and to the edges $\{a_1, b_1\}, \{a_2, b_2\}, \ldots, \{a_m, b_m\}$ (i.e., the algorithm has access to $2m$ integers $a_1, \ldots, a_m, b_1, \ldots, b_m$, where each edge of G is given by its endpoints a_i and b_i). You can assume that the graph is valid (specifically, $1 \leq a_i, b_i \leq n$ and $a_i \neq b_i$). The algorithm outputs "YES" or "NO", corresponding to whether G is a tree or not. Your algorithm must always complete in time polynomial in n (e.g., even $O(n^{10}m^{10})$ suffices). You do not have to show the correctness of your algorithm or what the running time of your algorithm is.

Hint: *Use part (c). There exists a (relatively) simple $O(n + m)$ solution. However, the official solution is $O(n \cdot m)$ for brevity and uses recursion to check if G is connected.*

Algorithm 2

Input: integers n, m. Collection of integers $a_1, b_1, a_2, b_2, \ldots, a_m, b_m$.

Let $visited[1 \ldots n]$ be a global variable, initialized to $False$.

function $walk(u)$ ▷ Find all neighbors of u that have not been visited and walk there.
    $visited[u] \leftarrow True$
    for $i \leftarrow 1 \ldots m$ **do** ▷ Iterate over all edges.
        if $a_i = u$ and not $visited[b_i]$ **then**
            $walk(b_i)$
        if $b_i = u$ and not $visited[a_i]$ **then**
            $walk(a_i)$

$walk(1)$ ▷ Find all vertices connected to 1.
$connected \leftarrow True$ if $visited[\cdot] = [True, True, \ldots, True]$ and $connected \leftarrow False$ otherwise
if $connected = True$ and $m = n - 1$ **then** ▷ Use the characterization from part (c).
    Print("YES")
else
    Print("NO")

Exercise 8.5 *Short questions about graphs* **(2 points)**.

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.

For each of the following statements, decide whether the statement is true or false. If the statement is true, provide a proof; if it is false, provide a counterexample.

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

(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.

(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).

Peer Grading(change of the rules this week)

Exercise 8.1

You will find the file to peergrade in your polybox folder

While emailing your peer grading to me please include the group you corrected their work in cc.

https://docs.google.com/spreadsheets/d/1owPsJsd9THBWInwFcVjKCc0f_r6n4pGwwDKMDdwaCjM/edit?usp=sharing