Eidgenössische Technische Hochschule Zürich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

Departement of Computer Science
Johannes Lengler, David Steurer
Lucas Slot, Manuel Wiedmer, Hongjie Chen, Ding Jingqiu

13 November 2023

Algorithms & Data Structures — Exercise sheet 8 — HS 23

The solutions for this sheet are submitted at the beginning of the exercise class on 20 November 2023.

Exercises that are marked by * are challenge exercises. They do not count towards bonus points.

You can use results from previous parts without solving those parts.

We first recall some definitions from the lecture and introduce some new ones.

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

- A sequence of vertices $(v_0, v_1, \dots, v_k)$ (with $v_i \in V$ for all i) is a **walk** (german "Weg") if $\{v_i, v_{i+1}\}$ is an edge for each $0 \le i \le k-1$. We say that v_0 and v_k are the **endpoints** (german "Startknoten" and "Endknoten") of the walk.
- A sequence of vertices $(v_0, v_1, \dots, v_k)$ is a **closed walk** (german "Zyklus") if it is a walk, $k \ge 2$ and $v_0 = v_k$.
- A sequence of vertices $(v_0, v_1, \dots, v_k)$ is a **path** (german "Pfad") if it is a walk and all vertices are distinct (i.e., $v_i \ne v_j$ for $0 \le i < j \le k$).
- A sequence of vertices $(v_0, v_1, \dots, v_k)$ is a **cycle** (german "Kreis") if it is a closed walk, $k \ge 3$ 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.
- A graph G is **connected** (german "zusammenhängend") if for every two vertices $u, v \in V$ there exists a path with endpoints u and v.[1]
- A graph G is a **tree** (german "Baum") if it is connected and has no cycles.

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.

[1]We will see in exercise 8.5 that this definition is equivalent to the one given in the lecture (which was that a graph G is connected if for every two vertices $u, v \in V$ there exists a walk with endpoints u and v)

(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.*

Solution:

Pick any $v \in V$. We distinguish two cases: (i) $\deg(v) \geq 3$ and (ii) $\deg(v) \leq 2$.

(i) If $\deg(v) \geq 3$, consider three neighbors v_1, v_2, v_3 of v. If any of the edges (v_1, v_2), (v_1, v_3) or (v_2, v_3) is present in G, then these two vertices together with v form a set of three vertices that are all pairwise adjacent. Otherwise, the vertices v_1, v_2, v_3 are a set of three vertices that are all pairwise not adjacent. In both cases, the statement holds.

(ii) Similary, if $\deg(v) \leq 2$, we consider three non-neighbors v_1, v_2, v_3 of v. Either one of the edges (v_1, v_2), (v_1, v_3) or (v_2, v_3) is not present in G, in which case these two vertices with v are all pairwise not adjacent, or these three edges are all present in G, in which case v_1, v_2 and v_3 are pairwise adjacent. In any case, the statement also holds in this case.

In summary, no matter what graph on 6 vertices we have, there are either three vertices that are all pairwise adjacent or there are three vertices that are all pairwise not adjacent (or even both).

(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.

Solution:

We consider the graph $G = (V, E)$, where

$$V = \{v_1, v_2, v_3, v_4, v_5\}$$

and

$$E = \{\{v_1, v_2\}, \{v_2, v_3\}, \{v_3, v_4\}, \{v_4, v_5\}, \{v_5, v_1\}\}.$$

In this graph, no set of three vertices are all pairwise adjacent (every vertex is adjacent to only two vertices that are not adjacent to each other). Also, no set of three vertices are all not pairwise adjacent (every vertex is not adjacent to only two vertices that are adjacent to each other). Hence, the statement is false for graphs with 5 vertices.

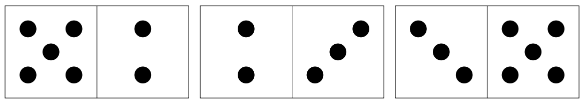
Guidelines for correction:

Award $1/2$ point for a correct proof in (a) and $1/2$ point for a correct counterexample in (b).

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?

Solution:

We directly solve the general problem.

First we note that we may neglect tiles of the form $[x|x]$. If we have a line without them, then we can easily insert them to any place with an x. Conversely, if we have a line with them then we can just remove them. Thus the problems with and without these tiles are equivalent.

Consider the following graph G with n vertices, labelled with $\{1, \ldots, n\}$. We represent the domino tile $[x|y]$ by an edge between vertices x and y. Then the resulting graph G is a complete graph K_n, i.e., the graph where every pair of vertices is connected by an edge. A line of domino tiles corresponds to a walk in this graph that uses every edge at most once, and vice versa. A complete line (of *all* tiles) corresponds to an Eulerian walk in G. Thus we need to decide whether $G = K_n$ has an Eulerian walk or not.

K_n is obviously connected. If n is odd then all vertices have even degree $n - 1$, and thus the graph is Eulerian. On the other hand, if n is even then all vertices have odd degree $n - 1$. If $n \geq 4$ is even, then there are at least 3 vertices of odd degree, and therefore K_n does not have an Eulerian walk. Finally, for $n = 2$, the graph K_n is just an edge and has an Eulerian walk. Summarizing, there exists an Eulerian walk if $n = 2$ or n is odd, and there is no Eulerian walk in all other cases. Hence, it is possible to line up the domino tiles if $n = 2$ or n is odd, and it is impossible otherwise. In particular, it is not possible for $n = 6$.

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.

Solution:

Consider the following algorithm:

Algorithm 1 Star-finding algorithm

```
while countNeighbors() $\neq n-1$ do
    setFlag()
    for $i = 1$ to countNeighbors() do
        moveTo($i$)
        if isSet() then
            undo()
        else
            break
```

In the following, we say that a vertex is *marked* if its flag is set to True. In each iteration of the while loop, a new, previously unmarked vertex is explored (if the vertex was already marked, the movement towards this vertex would have been undone). Hence, in each iteration, either the current vertex has $n-1$ neighbors and the algorithm terminates (case 1), or the number of vertices to be explored decreases by exactly one (case 2), or the current vertex has no unmarked neighbors and we loop forever on this vertex (case 3). Whenever the algorithm reaches the star $s \in V$, it successfully terminates (case 1), since a vertex is a star if and only if it has $n-1$ neighbors. Now, the star s is, by definition, a neighbor of all vertices; in particular, s is always a neighbor of the current vertex. Hence, for case 3 to occur, the star s must have been previously marked. But this never occurs, since the algorithm always terminates when reaching the star. Hence, only cases 1 and 2 can happen, and the number of unmarked vertices decreases by exactly one in each iteration until the star is eventually reached. This proves the correctness of the algorithm.

The cost of each iteration of the while loop is $O(1) + O(1) + \sum_{i=1}^{\deg v}(O(1) + O(1) + O(1)) = O(1) + O(\deg v)$, which sums up to at most $\sum_{v\in V}(O(1)+O(\deg v)) = O(|V|)+O\left(\sum_{v\in V}\deg v\right) = O(|V|)+O(2|E|) = O(|V|+|E|)$ as every vertex is explored at most once.

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.

Solution:

Consider the longest path $P = (v_0, v_1, v_2, \ldots, v_{k-1}, v_k)$ in G. Let $a := v_0$ be an endpoint of P. We claim a is a leaf. Suppose for the sake of contradiction that this is not true, i.e., the degree of a is at least 2 (since the tree has at least two vertices, the degree cannot be 0). Hence, there exists a neighbor $b \neq v_1$ of a. Now, consider the walk $P' = (b, v_0, v_1, \ldots, v_k)$. This walk is longer than P, hence by choice of P, it cannot be a path. Therefore, since b is the only new addition, there must exist an index $i > 1$ such that $b = v_i$. But now, $(b, v_0, v_1, \ldots, v_i)$ is a cycle in G, a contradiction.

(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.

Solution:

We proceed by induction on n.

Base case: When $n = 1$, there can only be $0 = n - 1$ edges. When $n = 2$, there exists a unique tree (two vertices connected by an edge), and that one has $1 = n - 1$ edge. This completes the base case.

Induction hypothesis: Assume that the hypothesis is true for every tree with $n \geq 2$ vertices, i.e. it contains $n - 1$ edges.

Induction step: We now show the property holds for every tree $G = (V, E)$ with $|V| = n + 1$ vertices.

Let u be a leaf in G (it must exist by part (a)), and let v be u's only neighbor in the tree $G = (V, E)$. Consider the graph $G' := (V \setminus \{u\}, E \setminus \{u, v\})$. We first argue that G' is a tree.

Claim: G' is connected.

Proof of Claim: Let $a, b \in V \setminus \{u\}$. Since G is a tree, there exists a path P in G with endpoints a, b. It is immediate that no path can contain a leaf except on its endpoints (or the leaf's only incident edge is used twice). Hence, P is also a path in G'. Thus, a and b are connected in G'. Since a and b were arbitrary, G' is connected. This completes the proof of this claim.

Claim: G' has no cycles.

Proof of Claim: Suppose for the sake of contradiction that P is a cycle in G'. But since G' is a subgraph of G, P is also a cycle in G. However, G is a tree and thus cannot have a cycle. This completes the proof of the second claim.

Combining the two claims, we proved that G' is a tree. It contains $|V \setminus \{u\}| = (n + 1) - 1 = n$ vertices. Hence, by the induction hypothesis, $|E \setminus \{u, v\}| = n - 1$. Therefore, $|E| = n$, which completes the induction step and the proof.

(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.*

Solution:

Suppose G is a tree. By definition, G is connected. By part (b), it has $n - 1$ edges. This completes one direction of the implication.

We now prove the other direction. Suppose G is connected and has $n - 1$ edges. We proceed by induction on n.

Base case: Let $n = 1$. The graph with a single vertex and 0 edges is trivially a tree. Let $n = 2$. There exists one unique graph with 2 vertices and 1 edge, and that graph is also obviously a tree. This completes the base case.

Induction hypothesis: Assume the hypothesis: Every connected graph with $n \geq 2$ vertices and $n - 1$ edges is a tree.

Induction step: We now show the property holds for $n+1$. Let $G = (V, E)$ be a connected graph with $n+1$ vertices and n edges. The average degree in this graph is $2|E|/|V| = 2n/(n+1) < 2$. Hence, there must exist a vertex u with degree 1 (no connected graph with at least 2 vertices can have 0-degree vertices). In other words, u is a leaf and let v be u's only neighbor in G. Consider the graph $G' := (V \setminus \{u\}, E \setminus \{u, v\})$. Clearly, G' has $n-1$ edges.

Claim: G' is connected.

Proof of Claim: Let $a, b \in V \setminus \{u\}$. Since G is connected, there exists a path P in G with endpoints a, b. As in part (b), no path can contain a leaf except on its endpoints. Hence, P is also a path in G' and thus G' is connected. This completes the proof of this claim.

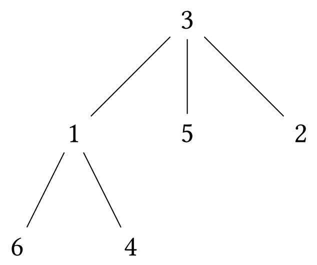
Therefore, we can apply the induction hypothesis on G' and conclude G' is a tree. It is simple to conclude that then G is also a tree: Any cycle in G must be fully contained in G' (since it cannot contain a leaf), and this is impossible since G' is a tree.

(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.

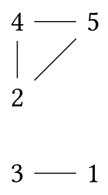
Hint: *Use part (c). There exists a (relatively) simple $O(n+m)$ solution. However, the official solution is $O(n \cdot m)$ for brevity and uses recursion to check if G is connected.*

Example 1: $n = 6$
$m = 5$
$a_1, b_1 = 1, 3$
$a_2, b_2 = 6, 1$
$a_3, b_3 = 3, 5$
$a_4, b_4 = 2, 3$
$a_5, b_5 = 4, 1$



Output: YES

Example 2: $n = 5$
$m = 4$
$a_1, b_1 = 1, 3$
$a_2, b_2 = 4, 5$
$a_3, b_3 = 5, 2$
$a_4, b_4 = 2, 4$



Output: NO

Solution:

Algorithm 2

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

Let $visited[1 \dots n]$ 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 $i \leftarrow 1 \dots m$ 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 $visited[\cdot] = [True, True, \dots, True]$ 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.

Solution:

If there is a walk with endpoints v and w, consider a walk $W : v = v_0, v_1, \dots, v_k = w$ between these two vertices with the shortest length. Since $v \neq w$, $k \geq 1$. If W is not a path, then there are $0 \leq i < j \leq k$ with $v_i = v_j$. But then, $v = v_0, v_1, \dots, v_i = v_j, v_{j+1}, \dots, v_k = w$ is a walk between v and w with length $k - (j - i) < k$, contradicting the choice of the walk. Hence, W is a path, so 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.

Solution:

This statement is false.
Consider the graph G that is a disjoint union of two cycles of length 3, i.e.

$$V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$$

and

$$E = \{\{v_1, v_2\}, \{v_2, v_3\}, \{v_3, v_1\}, \{v_4, v_5\}, \{v_5, v_6\}, \{v_6, v_4\}\}.$$

Then, both $n = m = 6$, but G is not connected since there is for example no path from v_1 to v_4.

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

Solution:

This statement is false.
Consider the complete graph on 4 vertices, i.e.

$$V = \{v_1, v_2, v_3, v_4\}$$

and

$$E = \{\{v_1, v_2\}, \{v_1, v_3\}, \{v_1, v_4\}, \{v_2, v_3\}, \{v_2, v_4\}, \{v_3, v_4\}\}.$$

Then G has a Hamiltonian path v_1, v_2, v_3, v_4. But since every vertex has odd degree 3, it follows that G has no Eulerian walk as we know from the lecture that a graph that has an Eulerian walk has at most 2 vertices with odd degree.

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

Solution:

This statement is true.

Proof using bound on the number of edges in a tree. Assume for a contradiction that G does not contain a cycle. Then each of its connected components must be a tree. By a previous exercise, we know that this implies that the total number of edges in G is at most $n - 1$. But each vertex of G has degree at least 2, implying G has at least $(2 \cdot n)/2 = n$ edges, a contradiction.

Direct proof. We construct a cycle in G as follows. Let v_1 be any vertex of G, and let v_2 be a neighbour of v_1 (i.e., $\{v_1, v_2\}$ is an edge). For $3 \leq i \leq n + 1$, inductively choose vertices v_i so that $\{v_i, v_{i-1}\}$ is an edge, and $v_i \neq v_{i-2}$. This is possible because each vertex of G has degree at least 2, and so v_{i-1} always has at least one neighbour which is not equal to v_{i-2}.

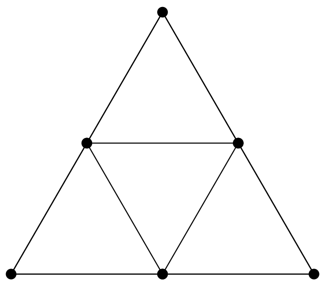
We have thus created a walk in G of length n. As G only has n vertices, this means there exist distinct $1 \leq j, k \leq n+1$ such that $v_j = v_k$, and $v_j, v_{j+1}, \ldots, v_{k-1}$ are all distinct. By construction, $k \geq j + 3$, and so the vertices $v_j, v_{j+1}, \ldots, v_k$ form a cycle (of length $k - j$).

(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 $n - 1$).

Solution:

This statement is false.

A counterexample is given by the following graph:



(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).

Solution:

This statement is true. Suppose for a contradiction that G has a Hamiltonian cycle $v_1, v_2, v_3, \ldots, v_n, v_1$, which we order so that $v_1 = v_{\text{cut}}$. Then, after removing $v_1 = v_{\text{cut}}$ from G, the resulting graph still has a Hamiltonian path, namely $v_2, v_3, \ldots, v_n$. In particular, the resulting graph is still connected.

Guidelines for correction:

For awarding the bonus points, each subexercise (except (a)) should be split into two parts, namely one part is giving the correct answer and the other part is giving a correct proof or counterexample. Subexercise (a) contains only one part, namely the proof of the statement. If at least 2 parts are solved correctly, $1/2$ points should be awarded. If at least 5 parts are solved correctly, 1 point should be awarded. If at least 8 parts are solved correctly, $3/2$ points should be awarded. If all 11 parts are solved correctly, 2 points should be awarded.