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
Kasper Lindberg, Lucas Slot, Hongjie Chen, Manuel Wiedmer

11 November 2024

Algorithms & Data Structures Exercise sheet 8 HS 24

The solutions for this sheet are submitted on Moodle until 17 November 2024, 23:59.

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.

The solutions are intended to help you understand how to solve the exercises and are thus more detailed than what would be expected at the exam. All parts that contain explanation that you would not need to include in an exam are in grey.

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

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

- For $v \in V$, the **degree** $\deg(v)$ of v (german "Knotengrad") is the number of edges that are incident to v.
- 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 $0 \le i \le k-1$. We say that v_0 and v_k are the **endpoints** (german "Startknoten" and "Endknoten") of the walk. The **length** of the walk $(v_0, v_1, \ldots, v_k)$ is k.
- A sequence of vertices $(v_0, v_1, \ldots, v_k)$ is a **closed walk** (german "Zyklus") if it is a walk, $k \ge 2$ and $v_0 = v_k$.
- A sequence of vertices $(v_0, v_1, \ldots, v_k)$ 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 $(v_0, v_1, \ldots, v_k)$ 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.
- An **Eulerian walk** (german "Eulerweg") is a walk that contains every edge exactly once.
- A **closed Eulerian walk** (german "Eulerzyklus") is a closed walk that contains every edge exactly once.
- A **Hamiltonian path** (german "Hamiltonpfad") is a path that contains every vertex.
- A **Hamiltonian cycle** (german "Hamiltonkreis") is a cycle that contains every vertex.
- For $u, v \in V$, we say u **reaches** v (or v is **reachable** from u; german "u erreicht v") if there exists a walk with endpoints u and v, or equivalently, there exists a path with endpoints u and v.
- A **connected component** of G (german "Zusammenhangskomponente") is an equivalence class of the (equivalence) relation defined as follows: Two vertices $u, v \in V$ are equivalent if u reaches v.

- A graph G is **connected** (german "zusammenhängend") if for every two vertices $u, v \in V$, u reaches v, or equivalently, if there is only one connected component.
- A graph G is a **tree** (german "Baum") if it is connected and has no cycles.

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

A group of $n \geq 3$ people wants to play the following *telephone game*: A random player in the group is given a message. The goal is to communicate this message to each member of the group. Each player is allowed to make one phone call and receive one phone call. Furthermore, a player can only make calls to other players who are in their *contact list* (you may assume that if player a is in the contact list of player b, then player b is also in the contact list of player a).

In this exercise, we care about the following question: *under what circumstances is it possible for the group to win the game, regardless of the starting player?* You may assume the group communicated a strategy beforehand, and each player is aware of the contents of each other player's contact list.

(a) Model the telephone game using a graph. Indicate carefully what the vertices and edges of this graph are. Then, give a necessary and sufficient condition for the game to be winnable (regardless of the starting player) using terminology from the lecture. Briefly argue the correctness of your condition.

Solution:

We consider a graph $G = (V, E)$ whose vertices $\{1, 2, \dots, n\}$ represent the players, and whose edges are

$$E = \{\{i, j\} : \text{ player } i \text{ has player } j \text{ in their contact list}\}.$$

This edge set is well-defined because we assumed the contact lists are symmetric.

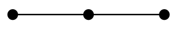
Because players are allowed to make and receive at most one call, a valid telephone game corresponds to a path in G which starts at the vertex representing the starting player, and contains all vertices, that is, a *Hamiltonian path.* For this reason, a necessary and sufficient condition for the game to be winnable regardless of the starting player is:

$$\text{For each } v \in V, \text{ there is a Hamiltonian path in } G \text{ which starts at } v.$$

(b) Give an example of a situation where the game is winnable for some, but not all starting players. Describe your example by drawing the graph that models it according to part (a).

Solution:

For example, we could look at the three player game described by a path graph:



Starting at either of the outer vertices, there is a Hamiltonian path, but there is no Hamiltonian path that starts from the middle vertex.

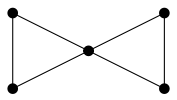
(c) Someone claims the game is always winnable if the following conditions hold:

- Each player has at least two other players in their contact list;
- For any two players a and b, it is possible for the message to reach player b if player a was the starting player.

Translate these conditions to your graph model of part (a) using terminology from the lecture. Then show the claim is false when $n = 5$.

Solution:

Let $G = (V, E)$ be the graph that models the telephone game as in part (a). The first condition means that each vertex in G has degree at least 2. The second condition means that G is connected. To see that these conditions are not sufficient when $n = 5$, consider the following graph:



This graph is connected, and each vertex has degree at least 2. However, there is no Hamiltonian path starting from the center vertex.

(d)* In a variant of the game for advanced players, the last person to learn the message has to call back the starting player to let them know everything went according to plan. Model this *advanced telephone game* using a graph as in part (a). Then, show that even if the (normal) telephone game is winnable regardless of the starting player, this does not mean the advanced telephone game is also winnable.

Hint: Look up the *Petersen graph.*

Solution:

Let $G = (V, E)$ be the graph that models the telephone game as in part (a). The advanced telephone game is winnable if and only if G has a Hamiltonian cycle (the starting player is irrelevant). To show the statement, we need a graph which does not contain a Hamiltonian cycle, but has a Hamiltonian path starting at each of its vertices. It turns out that this property holds for the well-known Petersen graph (see e.g. `https://en.wikipedia.org/wiki/Petersen_graph`) , which can easily be checked by hand.

Guidelines for correction:

The exercise consist of the following elements:

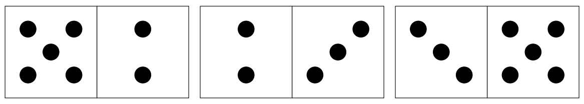
- Definition of vertices and edges in part (a);
- Condition in part (a) + brief explanation;
- Example in part (b);
- Translation of conditions in part (c);
- Counterexample in part (c);

Award 1 point if all elements are correct; award 1/2 point if at least 3 elements are correct. If the initial graph definition / condition in part (a) is (slightly) incorrect, you can still earn points for items (b), (c) if the solution is consistent.

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 like in the example below.



(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 find an equivalent problem that can easily be modeled as a graph problem. 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, and if we have a line with them, then we can just remove them. Thus, both problems are equivalent.

Then, we will model our modified problem as a graph problem. Consider a graph G with n vertices, labeled $\{1, \ldots, n\}$. Each domino tile $[x|y]$ is represented by an edge between the vertices x and y. By construction, the resulting graph G is a complete graph K_n. A complete line (of *all* tiles) corresponds to an Eulerian walk in K_n. Thus, we need to decide whether K_n has an Eulerian walk or not. To this end, we only need to check whether (i) the graph is connected, and (ii) expect for 2 vertices, all other vertices have even degrees.

A complete graph is defined as a graph in which every pair of vertices is connected by an edge. As every possible pair $[x|y]$ exists as a domino tile, G is a complete graph.

K_n is obviously connected because it is a complete graph. 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 $n = 6$.

The fact that a connected graph has an Eulerian walk if and only if all except for 2 vertices have even degrees is very useful whenever we talk about Eulerian walks in graphs and you should always remember this fact.

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 $|V| = n$. 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
```

Proof of correctness: 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.

Proof of runtime: 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.

There is actually a (faster and better) algorithm with runtime $O(\deg(v_0))$ where v_0 is the starting vertex.

Algorithm 2 Star-finding algorithm

```
if countNeighbors() $= n - 1$ then
    break
for $i = 1$ to countNeighbors() do
    moveTo($i$)
    if countNeighbors() $= n - 1$ then
        break
    else
        undo()
```

Note that the above algorithm does not need to use flags at all. The algorithm first checks if the starting vertex is a star. If yes, then we are done. Otherwise, the algorithm iterates over its neighbors and checks if one of them is a star. If there is a star in the graph, then the algorithm is guaranteed to find it, as a star is a neighbor of any other vertex. Since the algorithm only goes through the list of neighbors of v_0, the runtime is $O(\deg(v_0))$.

Exercise 8.4 *Introduction to Trees.*

In this exercise the goal is to prove a few basic properties of trees (for the definition of a tree, see Definition 1).

(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 $P = (v_0, v_1, v_2, \dots, v_{k-1}, v_k)$ in G. Let a be an endpoint of P, say $a = v_0$. A similar argument also holds for $a = v_k$. 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 of a such that $b \neq v_1$. Now, consider the walk $P' = (b, v_0, v_1, \dots, v_k)$. 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 $i \in [k]$ such that $b = v_i$. But now, $(b, v_0, v_1, \dots, v_i)$ 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. Note, as the induction hypothesis only applies to trees, we must first show that our modified graph is a tree.

We prove that G' is connected: Let $a, b \in V \setminus \{u\}$. Since G is a tree, there exists a path P in G with endpoints a, b. Note that the removal of u doesn't affect this, as leaves can only be endpoints in paths. Hence, P is also a path in G', meaning a and b are connected in G'. Since a and b were arbitrary, G' is connected.

We prove that G' has no cycles: To prove this by contradiction, we assume 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.

Therefore, G' is a tree and contains $|V \setminus \{u\}| = (n + 1) - 1 = n$ vertices. Then, we can apply the induction hypothesis and conclude $|E \setminus \{u, v\}| = n - 1$. Therefore, $|E| = n$, which completes the induction step and the proof.

This inductive proof is an example for the fact that it is sometimes easier to argue about deleting a vertex (in this case a leaf) from a graph instead of adding the $n + 1$-st vertex although adding a vertex might seem to be the more intuitive approach to this task. This idea of adding a new leaf to a tree on n vertices and then arguing that it contains n edges might seem easier at first glance, however, it is important to note that here one further needs to show that any tree on $n + 1$ vertices can indeed be constructed by adding a leaf to a tree on n vertices, which is essentially what the proof in our solution does. This is a recurring theme in the context of inductive proofs on graphs: in the inductive step it is often easier to argue about deleting a vertex/edge etc. from a graph instead of adding a vertex/edge to the graph arising from the induction hypothesis because with this approach it is often harder to argue that any graph on $n + 1$ vertices can be constructed this way. The next task is another example for this.

(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 a tree.

Induction hypothesis: Assume **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, as no connected graph with at least 2 vertices can have 0-degree vertices. Let u be this 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.

We prove that G' is connected: 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.

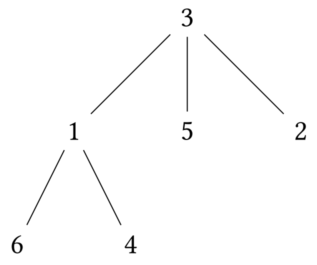
Therefore, we can apply the induction hypothesis on G' and conclude G' is a tree. Finally, we must show that G is also a tree: We know that G is connected. Again, we show by contradiction that G cannot contain any cycles. A cycle in G must be fully contained in G' (since it cannot contain a leaf), which is impossible since G' is a tree. Therefore, G must be a tree and the property holds for $n+1$.

(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 $\{a_1, b_1\}, \{a_2, b_2\}, \dots, \{a_m, b_m\}$ (i.e., the algorithm has access to $2m$ integers $a_1, \dots, a_m, b_1, \dots, b_m$, where each edge of G is given by its endpoints a_i and b_i). You can assume that the graph is valid (specifically, $1 \le a_i, b_i \le n$ and $a_i \ne b_i$). 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.

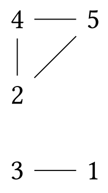
Hint: *Use part (c). There exists a (relatively) simple $O(n+m)$ solution. However, the official solution is $O(n \cdot m)$ for brevity and uses recursion to check if G is connected.*

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 3

```
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")
```

This algorithm is built using part (c). It checks if the graph is connected by starting at a single vertex and marking all reachable vertices. If all vertices are marked, we can conclude that the tree is connected. Then we check the number edges. If $|E| = n - 1$, we use (c) and conclude that G is a tree.

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 $v \neq w \in V$ and suppose that G is a tree. Prove that if P_1 and P_2 are paths that both start at v and end at w, then $P_1 = P_2$.

Solution:

Let $P_1 = (v_0, v_1, \dots, v_k)$ where $v_0 = v$ and $v_k = w$; let $P_2 = (v'_0, v'_1, \dots, v'_{k'})$ where $v'_0 = v$ and $v'_{k'}$; suppose for sake of contradiction that $P_1 \neq P_2$. Then let $v_i \in P_1$ and $v'_i \in P_2$ be the first vertex in which they differ. Also let $v_j \in P_1$ and $v'_{j'} \in P_2$ be the the next time after index i that the paths coincide, meaning $v_j = v'_{j'}$. There must be such indexes because $v_k = v'_{k'} = w$.

Then consider the new walk

$$v_{i-1}, v_i, v_{i+1}, \dots, v_{j-1}, v_j, v_{j'-1}, \dots, v'_{i+1}, v'_i, v_{i-1}.$$

Note by the minimality condition of the indexes j and j', all vertices except the v_{i-1} are used once. And since this is just subpaths of P_1 and P_2 stitched together where they meet, we know we have a closed walk with all vertices except the beginning and end are distinct. But this means we have a cycle, contradicting the assumption that G is a tree. Hence $P_1 = P_2$.

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) If every vertex of G has at least $\lceil n/2 \rceil$ neighbors, then G is connected.

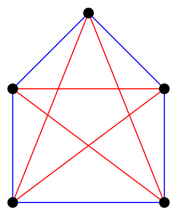
Solution:

This statement is true.
Suppose for sake of contradiction that a graph G with every vertex having degree $\lceil n/2 \rceil$ is not connected. This means there are vertices $u, v \in V$ which do not have a path between them. Let U be the subset of vertices containing u and its neighbors and similarly V be the subset of vertices containing v and its neighbors. By the degree condition, U must contain at least $\lceil n/2 \rceil + 1$ vertices and similarly for V. But then U and V must overlap in some vertex because $\lceil n/2 \rceil + 1 + \lceil n/2 \rceil + 1 > n$. Say w is the overlapping vertex in both U and V, but then either $\{u, v\} \in E$ or we can construct a path u, w, v, which is a contradiction. Hence G is connected.

(c) If G contains a Hamiltonian cycle C, then any other Hamiltonian cycle of G must contain an edge from C.

Solution:

This statement is false.
Consider the complete graph on 5 vertices,



Then the blue edges and red edges each correspond to Hamiltonian cycles which do not share any edge.

(d) For every graph G with $n \geq 2$, there must be at least two vertices with the same degree.

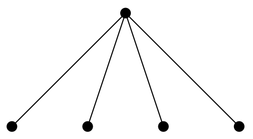
Solution:

This statement is true. Suppose for sake of contradiction that all vertices do not share the same degree. As the degree of a vertex must be between 0 and $n-1$ and we have n vertices, for every $0 \leq i \leq n-1$ there must exist some vertex with degree i. But then there is a vertex with degree $n-1$, meaning it has an edge to every other vertex. But this contradicts the fact that there must also exist a vertex with degree 0. Hence, at least two vertices must share the same degree.

(e) Suppose in a connected graph G, for every path of length at least 2, the sum of the degrees of the vertices in the path is even. Then G has an Eulerian walk.

Solution:

This statement is false. A counterexample is given by the following graph:

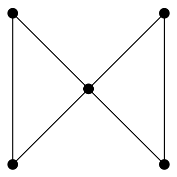


Then any path of length at least 2 must be a path of length 2 which goes from a degree-1 vertex, to the center degree-4 vertex and back to a degree-1 vertex. Thus it satisfies the condition that the sum of the degrees is even. However, for this same reason no Eulerian walk can exist since we must start and end at degree-1 vertices as to not retrace the same edge, but there are more than 2 such degree-1 vertices.

(f) Let G be a connected graph. Suppose that deleting any edge of G does not disconnect the graph. Then deleting any vertex of G does not disconnect the graph. (When deleting a vertex, we also remove all edges incident to the vertex.)

Solution:

This statement is false. Consider the graph on 5 vertices



Then one can check that deleting any one edge does not disconnect the graph, however, deleting the middle vertex does.

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.