Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

FS 2024

Prof. Dr. Roger Wattenhofer

Exam
Principles of Distributed Computing

Tuesday, August 20, 2024
8:30 – 10:30

Do not open or turn until told to by the supervisor!

The exam lasts 120 minutes, and there is a total of 120 points. The maximal number of points for each question is indicated in parentheses. Your answers must be in English. Be sure to always justify your answers. Algorithms can be specified in high-level pseudocode or as a textual description. You do not need to give every last detail, but the main aspects need to be there. Big-O notation is acceptable when giving algorithmic complexities. Some questions will ask you to fill in answers in a template. If you decide to start over you will find fill-in replacements at the end of the examination booklet. Please write legibly, illegible answers will not be graded.

Please write down your name and Legi number (your student ID) in the following fields.

Family Name	**First Name**	**Student Number**

Exercise	**Achieved Points**	**Maximum Points**
1 - Multiple Choice		18
2 - Coloring		28
3 - Maximal Independent Set		28
4 - Graph Neural Networks		22
5 - Labeling		24
Total		**120**

1 Multiple Choice (18 points)

Indicate whether the following statements are true or false by ticking the corresponding boxes. Justify your answers.

A) [9] Consider the radius R and diameter D of an undirected graph, as well as the radii of its nodes.

	TRUE	**FALSE**
For connected bipartite graphs with an odd number of nodes, the graph diameter is exactly twice the graph radius ($D = 2R$). *Justification:*	☐	☐
For rings, the graph diameter equals the graph radius ($D = R$). *Justification:*	☐	☐
In connected graphs with an even number of nodes, for every node u there is at least one other node v with the same radius as u. *Justification:*	☐	☐

B) [9] Consider any uniform connected graph with n nodes, where each node stores an integer input value between 1 and 100. Let r be an arbitrary node acting as the root.

	TRUE	**FALSE**
Root r can compute the maximum of the node values using one pass of the flooding/echo algorithm with messages' size not exceeding $\mathcal{O}(1)$ bits. *Justification:*	☐	☐
Root r can compute the mean of the node values using one pass of the flooding/echo algorithm with messages' size not exceeding $\mathcal{O}(1)$ bits. *Justification:*	☐	☐
Root r can compute the median of the node values using one pass of the flooding/echo algorithm with messages' size not exceeding $\mathcal{O}(\log n)$ bits. *Justification:*	☐	☐

Solutions

A) [9] Consider the radius R and diameter D of an undirected graph, as well as the radii of its nodes.

	TRUE	FALSE
For connected bipartite graphs with an odd number of nodes, the graph diameter is exactly twice the graph radius ($D = 2R$). *Justification: One can simply construct a bipartite graph with an odd diameter and an odd number of nodes, e.g., a tree with two branches of depth 1 and one of depth 2.*		✓
For rings, the graph diameter equals the graph radius ($D = R$). *Justification: In rings, all nodes have the same radius because of the symmetry. Therefore, the minimum node radius (R) equals the maximum node radius (D).*	✓	
In connected graphs with an even number of nodes, for every node u there is at least one other node v with the same radius as u. *Justification: The radius of the center node of a star with 3 branches is unique.*		✓

B) [9] Consider any uniform connected graph with n nodes, where each node stores an integer input value between 1 and 100. Let r be an arbitrary node acting as the root.

	TRUE	FALSE
Root r can compute the maximum of the node values using one pass of the flooding/echo algorithm with messages' size not exceeding $\mathcal{O}(1)$ bits. *Justification: In the echo phase, each node transmits the greatest value between its own value and those received from its children (if any).*	✓	
Root r can compute the mean of the node values using one pass of the flooding/echo algorithm with messages' size not exceeding $\mathcal{O}(1)$ bits. *Justification: As the graph is uniform, r does not know n. Therefore, its children must transmit the size of their subtrees, which can take $\mathcal{O}(\log n)$ bits.*		✓
Root r can compute the median of the node values using one pass of the flooding/echo algorithm with messages' size not exceeding $\mathcal{O}(\log n)$ bits. *Justification: In the echo phase, each node sends 100 counters (using $\mathcal{O}(\log n)$ bits each) representing the number of occurrences of each value in the subtree rooted at that node. With this full distribution, r can compute the median.*	✓	

2 Coloring (28 points)

Alice has just started studying distributed computing, and she is trying to solve the graph coloring problem on *grid graphs*! For $R, C \geq 2$, a grid graph consists of $N = R \cdot C$ nodes arranged in R rows and C columns. The node on the i-th row and j-th column is denoted by $v_{i,j}$. Each node $v_{i,j}$ has at most four neighbors: the top neighbor $v_{i-1,j}$ (if $i > 1$), the bottom neighbor $v_{i+1,j}$ (if $i < R$), the left neighbor $v_{i,j-1}$ (if $j > 1$), and the right neighbor $v_{i,j+1}$ (if $j < C$).

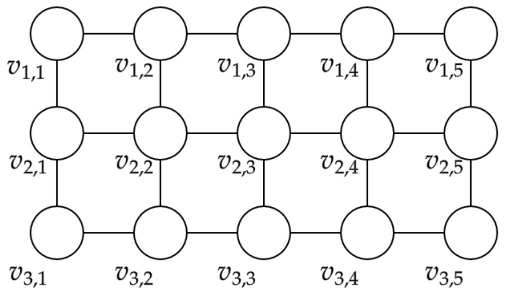


Figure 1: Grid graph for $R = 3$ and $C = 5$.

A) [1] The chromatic number of the grid graph is ....

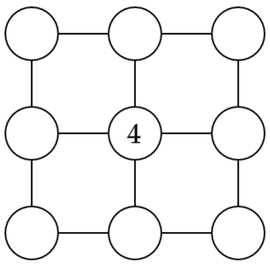
The grid graph models a synchronous network. All nodes are assigned unique identifiers of $O(\log N)$ bits. Initially, every node knows N, its own identifier, and the identifiers of its top, bottom, left, and right neighbors (whenever they exist). In the following, we will help Alice design a distributed algorithm that obtains a proper 4-coloring of the grid graph.

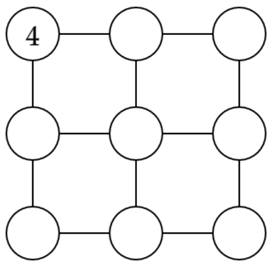
B) [7] Describe a distributed algorithm that computes a proper 9-coloring of the grid graph in $O(\log^\star N)$ communication rounds with colors $0, 1, \ldots, 8$. Explain the algorithm's correctness.

C) [4] Assume we are given a proper 9-coloring of the grid graph. That is, each node additionally holds a color in $\{0, 1, \ldots, 8\}$ such that adjacent nodes hold different colors. Describe a distributed algorithm obtaining a proper 5-coloring in $O(1)$ communication rounds.

So far, we colored the grid graph using 5 colors: $0, \ldots 4$. We now focus on discarding color 4!

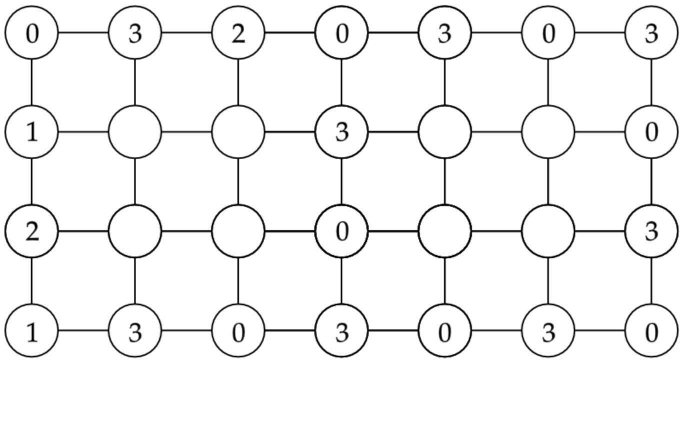
D) [2] Complete the coloring of each of the grid graphs below such that the node holding color 4 **cannot** choose any lower color. If there is no such coloring, explain why.





Alice believes that we should allow a few more nodes to choose new colors in order to help their neighbors holding color 4. Hence, if node $v_{i,j}$ holds color 4 ($i < R$, $j < C$), we will allow nodes $v_{i,j+1}$, $v_{i+1,j+1}$, and $v_{i+1,j}$ to help. These four (including $v_{i,j}$) nodes form a *fixing window for* $v_{i,j}$. Alice claims that if all nodes in such a fixing window discard their colors, obtaining a proper 4-coloring from a 5-coloring should be very easy! Let us take a look.

E) [2] Complete the coloring below using colors in $\{0, 1, 2, 3\}$ such that the final coloring is proper.



The fixing window approach seems to be a good idea! Then, in Tasks **F)** and **G)**, you may assume that we are given a proper 5-coloring of the grid graph where all nodes $v_{i,j}$ around the border of the grid hold colors in $\{0, 1, 2, 3\}$. You may also make use of the following fact that Alice has already proven: *a fixing window can always be recolored using colors in* $\{0, 1, 2, 3\}$ *such that the grid graph remains properly colored.*

F) [4] We first assume that the fixing windows are sufficiently separated. That is, if $v_{i,j}$ and $v_{i',j'}$ hold color 4 in the given 5-coloring, then $|i-i'| \geq 3$ or $|j-j'| \geq 3$. Describe a distributed algorithm that obtains a proper 4-coloring of the grid graph in $O(1)$ communication rounds.

G) [8] We will now discard the assumption of Task **F)**. Alice shows you a distributed algorithm that colors any graph with maximum degree Δ using only Δ^2 colors and within $O(\Delta \log \Delta + \log^\star N)$ communication rounds. Describe a distributed algorithm that, making use of the algorithm given by Alice, obtains a proper 4-coloring of the grid graph in $O(\log^\star N)$ communication rounds.

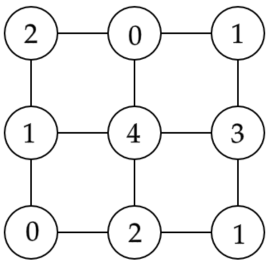
Solutions

A) The chromatic number of the grid graph is **2**.

B) We run the log-star algorithm from the lecture on each row and each column in parallel. In the row (resp. column) invocations, nodes mark their top (resp. left) neighbor as a parent and bottom (resp. right) neighbor as a child. This way, nodes obtain a pair of colors $(c_1, c_2) \in \{0, 1, 2\} \times \{0, 1, 2\}$, which are then mapped to values in $\{0, 1, \ldots, 8\}$ using a bijection. Adjacent nodes in the grid are also adjacent in at least one invocation of the log-star algorithm, therefore they obtain different pairs of colors and hence different colors. The round complexity follows from the round complexity of the log-star algorithm.

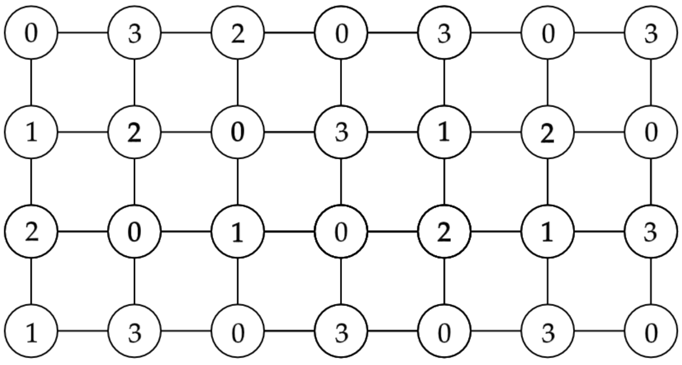
C) Nodes send their color to all neighbors. In round $r = 2 \ldots 5$, nodes holding color $r + 3$ set their color to the lowest available color in $\{0, 1, 2, 3, 4\}$ (always possible as the maximum degree is 4), and send their new color to their neighbors. Since the initial coloring is proper and hence adjacent nodes do not update their color in the same round, the final coloring is also proper.

D) In the first grid, any proper coloring where the neighbors of 4 hold colors 0, 1, 2 and 3 is a good solution, e.g.,



For the second grid, there is no solution: The node holding color 4 only has two neighbors, hence some color in $\{0, 1, 2, 3\}$ is always available.

E) A possible solution is shown below.



F) First, nodes send their initial color to everyone. Nodes receiving color 4 from their left neighbor v announce their bottom neighbor that it is part of the fixing window for v. Then, all nodes in the fixing window for v send the colors they cannot use to v. Node v can then locally compute a proper partial coloring, which exists according to Alice's claim, and sends the new color to the nodes in the fixing window.

G) Fixing windows are first identified similarly to Task **F)**. Nodes announce their fixing windows of all the fixing windows they belong to. This leads to a virtual graph having the fixing windows as vertices, where two fixing windows are adjacent iff they overlap.

Due to the structure of the grid graph, the maximum degree of the virtual graph is bounded by some constant c. Therefore, Alice's algorithm obtains a proper c^2-coloring of the virtual graph in $O(\log^\star N)$ rounds. Then, we run c^2 additional rounds: in round r, fixing windows holding color r proceed as in Task **F)**. This way, only non-overlapping windows are colored in parallel, and Alice's claim ensures fixing windows can be recolored.

3 Maximal Independent Set (28 points)

Consider the following algorithm for finding an MIS you learned in the lectures:

Algorithm 1 Fast MIS 2

```
The algorithm operates in synchronous rounds, grouped into phases.
A single phase is as follows:
1) Each node $v$ chooses a random value $r(v) \in [0, 1]$ and sends it to its neighbors.
2) If $r(v) < r(w)$ for all neighbors $w \in N(v)$, node $v$ enters the MIS and informs its neighbors.
3) If $v$ or a neighbor of $v$ entered the MIS, $v$ terminates ($v$ and all edges adjacent to $v$ are removed from the graph), otherwise $v$ enters the next phase.
```

In this task, we will study the size of the independent set obtained by running this algorithm.

A) [8] Show that the expected size of the maximal independent set obtained by the Fast MIS 2 algorithm on a graph G on n vertices is at least $\frac{n}{d+1}$, where d is the **average** degree of the graph.

Hint: You can use that $\frac{1}{x_1} + \frac{1}{x_2} + \ldots + \frac{1}{x_n} \geq \frac{n^2}{x_1+x_2+\ldots+x_n}$ for positive $x_1, x_2, \ldots, x_n$.

B) [5] Show that the previous bound is tight. More precisely, give an example of a graph G such that, if we denote by n the number of vertices and by d the average degree, any execution of Fast MIS 2 always finds an independent set of size exactly $\frac{n}{d+1}$. Justify your answer.

We also wish to compare the **maximal** independent set found by Fast MIS 2 to any **maximum** independent set of a graph G. A maximum independent set is an independent set of largest possible cardinality (as many nodes as possible), which is also an MIS by definition.

C) [7] Give an example of a graph G such that G contains at least one maximal, but not maximum independent set, and the MIS found by Fast MIS 2 is maximum with probability at least 0.9999 and prove that this is the case.

D) [8] Give an example of a graph G such that the MIS found by Fast MIS 2 is **not** maximum with probability at least 0.9999 and prove that this is the case.

You will earn 3 extra points if G is connected (assuming your proof is correct).

Solutions

A) Let X_v be the indicator random variable for the event "vertex v entered the MIS in phase 1 of Fast MIS 2". Denote the size of the MIS as S. It holds that $S \geq \sum_{v \in V} X_v$ so we have that

$$\mathbb{E}[S] \geq \sum_{v \in V} \mathbb{E}[X_v] = \sum_{v \in V} \frac{1}{1+\deg(v)} \geq \frac{n^2}{n+nd} = \frac{n}{d+1}.$$

where we use linearity of expectation in the first step and the hint in the second.

B) Consider the complete graph on n vertices K_n. Every vertex has degree $n-1$ so $d = n-1$, but every possible MIS the algorithm could output is of size exactly $1 = \frac{n}{d+1}$.

C) Let $G = K_{1,s}$ be the star graph with $s \geq 10^4$ leaves. G only has two possible MIS: either the center of the star (of size 1) or the set of all the leaves L (which is also maximum). Notice that the algorithm terminates after at most 2 phases. The algorithm only outputs $\{v\}$ if the value $r(v) < r(\ell)$ for all leaves ℓ, so with probability $\frac{1}{s+1}$. Otherwise, a leaf node (or multiple) enters the MIS, the center is deleted, then all the leaves that did not enter the MIS in Phase 1 enter it in Phase 2. Therefore, the algorithm selects L with probability

$$1 - \frac{1}{s+1} > 0.9999$$

and we are done.

Note: Another possible example is a graph composed of two sufficiently large cliques that overlap in exactly one vertex.

D) Take G to be the union of $s > 4\log_{3/2} 10$ disjoint paths of length 3, say $T_1, T_2, \dots, T_s$. The unique maximum independent set S consists of the $2s$ endpoints of the T_i, whereas any other MIS is non-maximum. Therefore, the algorithm only chooses S if no midpoint of any of the T_i ever enters the MIS. Let X_v be the indicator random variable for the event "vertex v entered the MIS in phase 1". The probability that a midpoint vertex m_i of any path T_i enters the MIS in step 1 is $\mathbb{P}[X_{m_i} = 1] = \frac{1}{3}$ and as the paths aren't connected, these random variables are independent, so we have

$$\mathbb{P}[\text{Fast MIS 2 finds } S] \leq \mathbb{P}\left[\text{no midpoint } m_i \text{ chosen in step 1}\right] = \left(\frac{2}{3}\right)^s < 0.0001$$

and we are done.

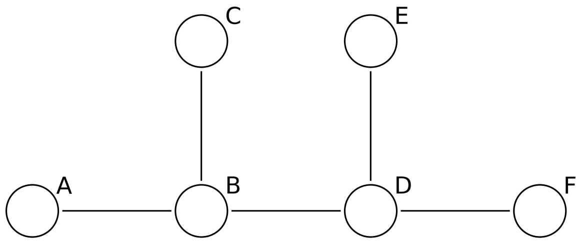
D*) To find a G that is connected, fix $s > 12\log_{3/2} 10$ and let $G = P_{2s+1}$ be the path graph on $2s+1$ vertices labelled $\{1, 2, \dots, 2s+1\}$ in that order on the path. The unique maximum independent set of G is the set of vertices S with odd labels, whereas there are many MISs. The algorithm only outputs S if no vertex of even index ever enters the MIS. Let X_v be the indicator random variable for the event "vertex v entered the MIS in phase 1". Now, consider the set of vertices $T = \{2, 6, 10, \dots\}$ of cardinality at least $s/3$ (those whose label gives remainder 2 when divided by 4) The random variables X_t for $t \in T$ are all independent, and we have that $\mathbb{P}[X_t = 1] = \frac{1}{3}$ (as it has to be the minimum between its neighbours) so we have that

$$\mathbb{P}[\text{Fast MIS 2 finds } S] \leq \mathbb{P}\left[\text{no vertex of } T \text{ chosen in step 1}\right] = \left(\frac{2}{3}\right)^{|T|} < \left(\frac{2}{3}\right)^{\frac{s}{3}} < 0.0001$$

and we are done.

Note: Another possible example is to take a sufficiently large clique and remove a single edge.

4 Graph Neural Networks (22 points)



Consider the social network graph G shown above with nodes representing people. The links can represent friendships, collaborations, or any other type of relationship.

A) [3] Run the Weisfeiler-Lehman (WL) algorithm on G. Assume the nodes have no initial features. For your answer, label the graph above. Note: you are free to use arbitrary numbers as labels — what matters is how they partition the nodes.

In this problem, we consider the task of predicting new links between nodes. Assume that the new-link probability $\Pr(\{v, w\})$ is proportional to the number of common neighbors of v and w.

B) [1] Based on this model, who is more likely to form a link with node A?

○ C ○ E

C) [4] We attempt to formulate the new edge prediction task as

$$\Pr(\{v, w\}) = f(h_v, h_w)$$

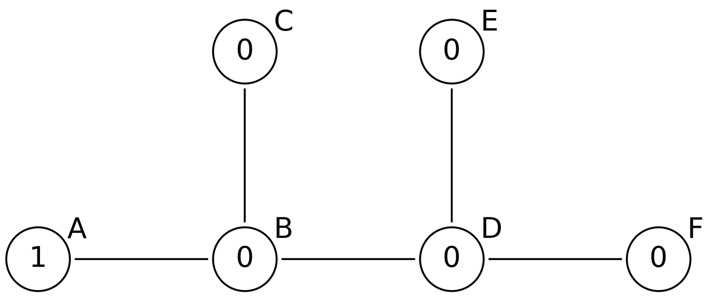
where f is any function, and h_v and h_w are the node embeddings of v and w, computed with a message passing GNN. Assume that the nodes have no initial features. Explain why it is not possible that $f(h_{\mathrm{A}}, h_{\mathrm{C}}) \neq f(h_{\mathrm{A}}, h_{\mathrm{E}})$. Can our approach be used to predict links?

D) [3] Give an example of an undirected graph where our approach predicts the same link probability for all non-edges, that is, $f(h_v, h_w)$ is the same for all $\{v, w\} \in V^2 \setminus E$ with $v \neq w$. The graph should be connected, have at least 2 non-edges and at least 5 nodes.

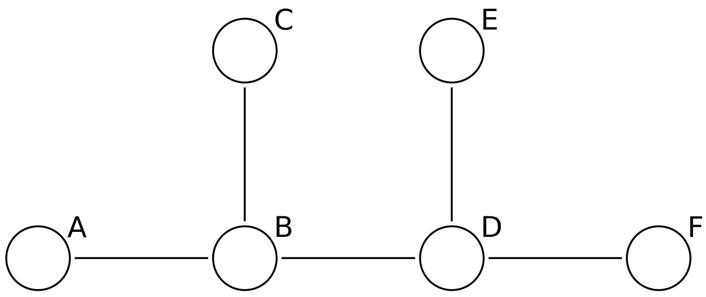
E) [3] Suppose we introduce an *anchor* by assigning the label '1' to one of the nodes and labeling all other nodes with '0'. See below for an example.

The WL algorithm can be extended to graphs with initial features. The labels are initialized with the node features, and the algorithm proceeds as usual. Run the WL algorithm on the anchored graph below. For your answer, fill in node labels for rounds 1 and 2. Note: You are free to use arbitrary numbers as labels — what matters is how they partition the nodes.

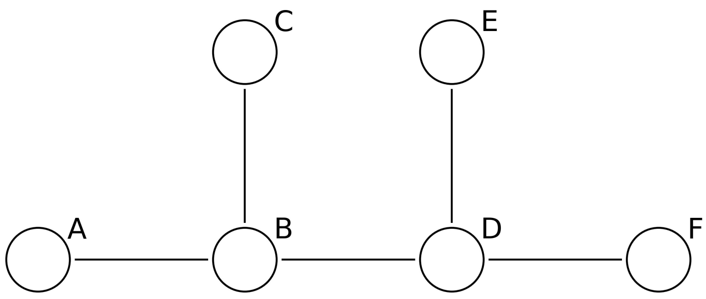
Round 0:



Round 1:



Round 2:



F) [8] Let $G = (V, E)$ be an undirected graph. Anchor a node $x \in V$, as in the previous part. We claim that WL is expressive enough to count the number of common neighbors with the anchor x for any node $v \in V$, denoted $c_v := |N(v) \cap N(x)|$. Specifically, your task is to argue that for any $v, w \in V$, $c_v \neq c_w \implies L_v \neq L_w$ where L_v, L_w are the final labels of v, w in the WL test.

Solutions

A) WL assigns one label to the nodes A, C, E and F. A second label is assigned to B and D.

B) Node C. For C, one out of one of its neighbors are also neighbors of node A, while node E shares no common neighbors with A.

C) By part **a**, we know that nodes C and E have the same label under WL. The distinguishing power of a message passing GNN is bounded by the WL test, meaning the embeddings of C and E must be the same. Hence, $f(h_{\mathrm{A}}, h_{\mathrm{C}}) = f(h_{\mathrm{A}}, h_{\mathrm{E}})$. This implies that a message passing GNN cannot learn to predict links based on the number of common neighbors (without having some additional features).

D) Any graph where WL assigns all nodes the same label works, for example a cycle or in general any k-regular graph.

E) The first round partitions nodes into $\{\mathrm{A}\}, \{\mathrm{B}\}, \{\mathrm{D}\}, \{\mathrm{C}, \mathrm{E}, \mathrm{F}\}$. The second (and the final) partition is $\{\mathrm{A}\}, \{\mathrm{B}\}, \{\mathrm{C}\}, \{\mathrm{D}\}, \{\mathrm{E}, \mathrm{F}\}$.

F) The first label $L_v^{(1)}$ of each $v \in V$ is based on the input feature and a multiset of neighborhood features. For nodes with $x \in N(v)$, the multiset contains a '1' and for others it does not. Hence, for any $v, w \in V$, if $x \in N(v)$ and $x \notin N(w)$, then $L_v^{(1)} \neq L_w^{(1)}$. In essence, first round labels can be partitioned into neighbors of the anchor $\mathcal{X} = \{L_v^{(1)} : v \in N(x)\}$ and other labels. In the second round, each $v \in V$ receives a multiset of labels over its neighborhood. If $c_v \neq c_w$, then the multisets of v and w contain a different number of labels from $\mathcal{X}$, and hence $L_v^{(2)} \neq L_w^{(2)}$. Finally, note that $L_v^{(2)} \neq L_w^{(2)}$ implies $L_v^{(R)} \neq L_w^{(R)}$ for all $R \geq 2$, so the final labels are distinct.

5 Labeling (24 points)

You are given an incomplete directed $n \times n$ grid graph G on n^2 nodes. We want to design a labeling scheme l to determine reachability between two nodes in the graph. More precisely, given only the labels $l(u)$, $l(v)$ of the two vertices u and v, you have to decide whether there exists a (directed) path from u to v, or, in other words, whether u can reach v following the arrows. Note that in the grid graph, an edge can only be present if the two nodes are adjacent in the same column or in the same row. However, it is possible that no edges or that edges in both directions are present between two neighboring nodes.

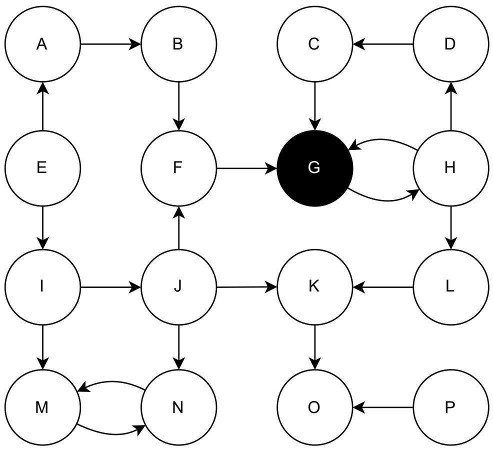
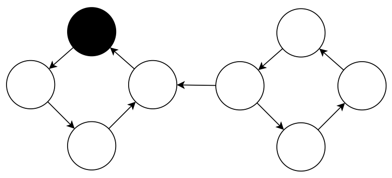


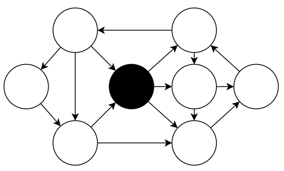
Figure 2: A 4×4 directed grid graph with the black node G. In this graph, node E can reach node K through E, A, B, F, G, H, L, K to include at least one black node. In contrast, node I **cannot** reach nodes E and F with at least one black node on the path.

A) [5] First, assume that there is one special node in the graph that is colored black. We want to determine for any two nodes u, v if there is a way from u to v that contains the black node on the way. Design a labeling scheme using 2-bit labels for this special case.

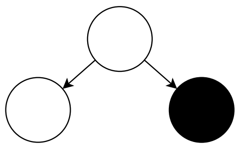
B) [12] Decide for each graph below whether there exists a labeling with 1 bit per node for answering whether there is a way from u to v visiting the black node. If there is such a labeling ("yes"), label the nodes and give the decoder by filling in the corresponding column in the table below. Otherwise ("no"), reason why it is not possible.



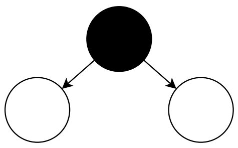
(a) ☐ yes ☐ no



(b) ☐ yes ☐ no



(c) ☐ yes ☐ no



(d) ☐ yes ☐ no

Decoder	(a)	(b)	(c)	(d)
$d(0,0)$				
$d(0,1)$				
$d(1,0)$				
$d(1,1)$				

For "yes": label the graph, fill in the column — each entry **T**rue (if reachable) or **F**alse (otherwise).
For "no": argue why in the space below:

C) [4] There are no black nodes in the grid graph anymore. However, you can assume that n is odd. Furthermore, when answering a query between nodes u, v you can assume that u belongs to the left half (columns 1, ..., $\frac{n-1}{2}$) and v belongs to the right half of the graph (columns $\frac{n-1}{2} + 2$, ..., n), or the other way around. Design a labeling scheme with $\mathcal{O}(n)$-bit labels for reachability. You can reuse any previous subtasks without proof.

D) [3] Now the queried nodes can be located anywhere. Sketch how you could build a labeling scheme with $\mathcal{O}(n \log n)$-bit labels for reachability. You can reuse any previous subtasks without proof. You do not need to prove your construction rigorously.

Solutions

A) **Encode**: Let b be the number of black vertices. For each vertex u we encode for each black vertex b if u can reach it in the b-th position of $reach$ and if the black node can reach u in the b-th position of $from$ using $2b$ bits. If to_u AND $from_v$ is all zeros, there is no such way, otherwise there is at least one path.

B) (a) yes - all left nodes 1 the right nodes 0,
(b) yes - all nodes 1
(c) no - the two leafs have different labels. The center can reach the black node, but not the other, wlog let black be 1, the other 0. But the center can't be 0 (would imply the other leaf can reach the black node) nor 1 (then the black node could reach the center).
(d) yes - black node 1, others 0

Decoder	(a)	(b)	(c)	(d)
d(0,0)	F	T	-	F
d(0,1)	T	T	-	F
d(1,0)	F	T	-	T
d(1,1)	T	T	-	T

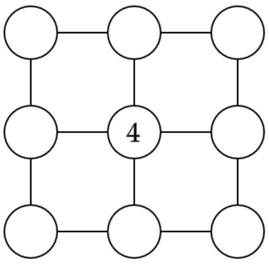
Table 5: For each yes fill in if the decoder output is **T**rue (if reachable) or **F**alse.

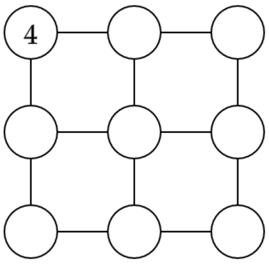
C) We reuse the idea from A), and apply it n times to include the n nodes in the middle as the black nodes and store reachability with $2n$ bits. If there is a path from u to v at least one middle node has to be included in the path. We check all n nodes as possible intermediates and say YES if at least one of them answers YES.

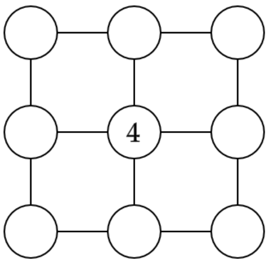
D) We can reuse the idea from C) and recursively apply it to the graph. First, we encode the reachability to all nodes that are in the same or adjacent column directly using n bits and the position of the node - this ensures we can always split into left, middle and right side. With the position we can decide if we can answer or need to recurse. Whenever we recurse, we split the graph into two halves. As we reduce the size of the graph by a constant factor each step, we need $\log n$ steps.

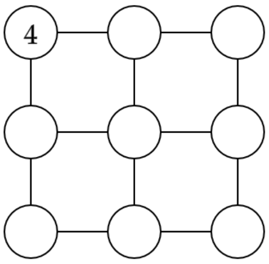
Replacements for Question 2, Parts D) and E)

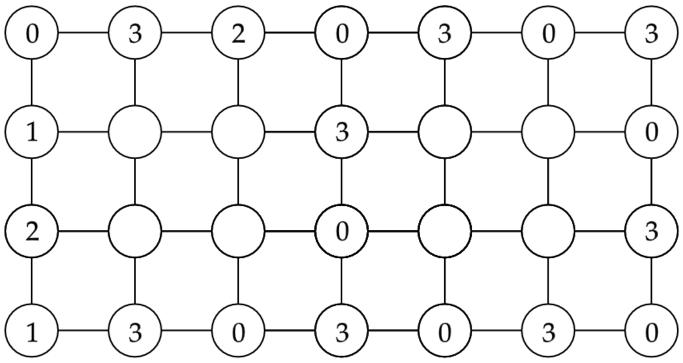
Here you can find replacement templates in case your original solution becomes too messy. If you use these, please *clearly* indicate which ones represent your final answers.

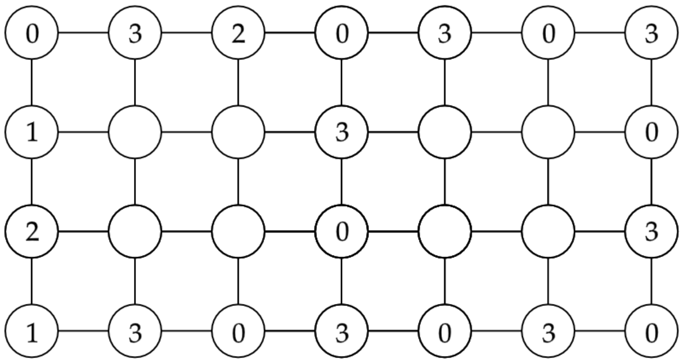






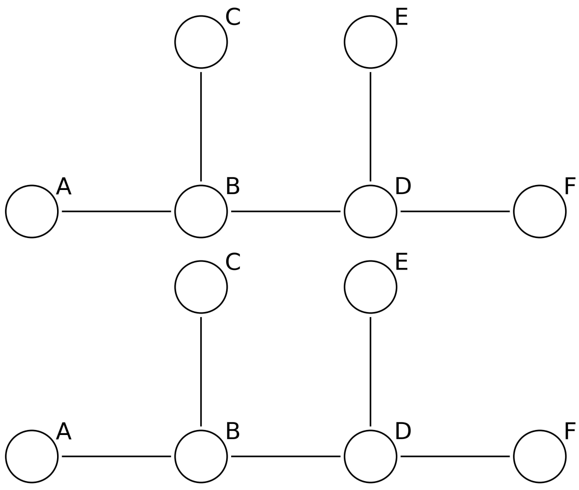




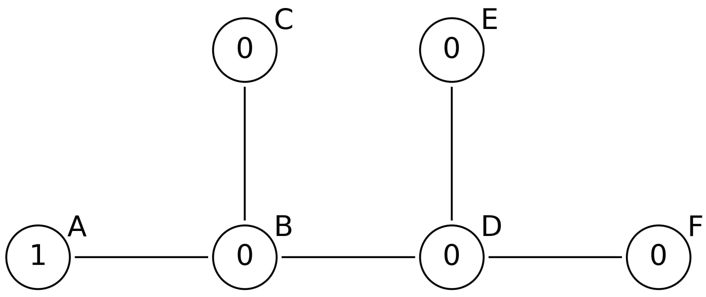


Replacements for Question 4, Parts A) and E)

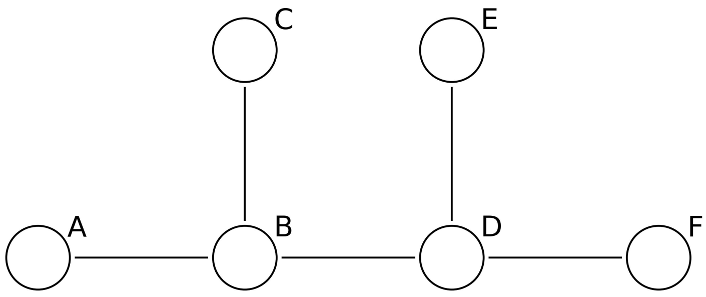
Here you can find replacement templates in case your original solution becomes too messy. If you use these, please *clearly* indicate which ones represent your final answers.



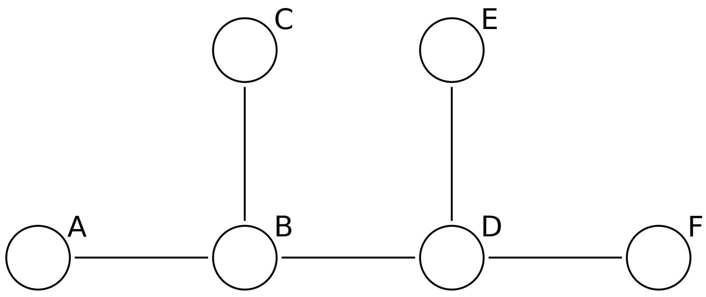
Round 0:



Round 1:

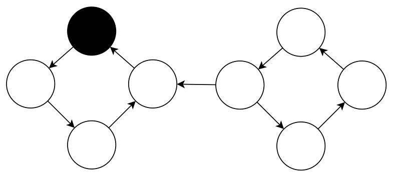


Round 2:

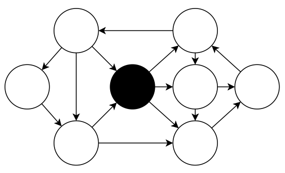


Replacements for Question 5, Part B)

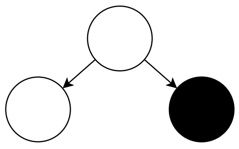
Here you can find replacement templates in case your original solution becomes too messy. If you use these, please *clearly* indicate which ones represent your final answers.



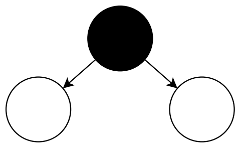
(a) ☐ yes ☐ no



(b) ☐ yes ☐ no



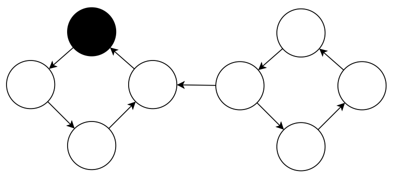
(c) ☐ yes ☐ no



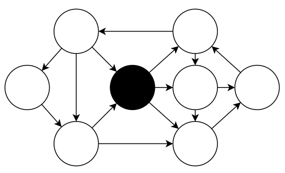
(d) ☐ yes ☐ no

Decoder	(a)	(b)	(c)	(d)
$d(0,0)$				
$d(0,1)$				
$d(1,0)$				
$d(1,1)$				

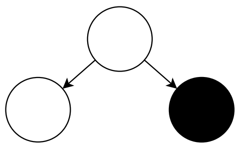
Decoder	(a)	(b)	(c)	(d)
$d(0,0)$				
$d(0,1)$				
$d(1,0)$				
$d(1,1)$				



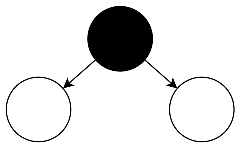
(a) ☐ yes ☐ no



(b) ☐ yes ☐ no



(c) ☐ yes ☐ no



(d) ☐ yes ☐ no