Problem 10-1 (How NOT to Read All Edges) 8 Points

(a) (5 points) Design \( O(n) \) algorithm to test if a given undirected graph \( G \) is acyclic. Notice, the running time of your algorithm should not depend on the number of edges \( m \)!

(Hint: Could you argue faster termination of a regular DFS tester on undirected graph?)

(b) (3 points) Extend the above algorithm to actually print the cycle, in case \( G \) is cyclic.

Problem 10-2 (Rescue Mission) 8 (+4) points

A big earthquake in the island of Coco destroyed many (but not all) two-way roads connecting some of the \( n \) houses present in the island. The Red Cross would like to check all the \( n \) houses to ensure that all the inhabitants are safe. The Red Cross can fly in an inspector to any particular house \( u \) by a helicopter. However, landing the helicopter is very dangerous, so the officials would like to minimize the number of such landings. Luckily, once the inspector is delivered to any node \( u \), he can drive to any neighboring node \( v \), if the road \((u,v)\) is not destroyed (but not otherwise). Unfortunately, the inspector can only tell if the road \((u,v)\) is safe after arriving at \( u \), so one cannot plan the whole rescue operation in advance. In particular, the Red Cross does not even know in advance the number \( m \) of safe roads. Nevertheless, help Red Cross to design a rescue operation with the smallest number of helicopter landings.

(a) (8 points) Design a strategy that minimizes the number of landings. Prove that your strategy always gives an optimal, i.e., no other plan has fewer landings.

(b) (4 points) [Extra credit] Design a plan that has minimal number of landings, and less than \( 2n \) trips along safe roads.

Problem 10-3 (Avoiding Double Infections) 10 points

You are given a directed graph \( G = (V,E) \) on \( n \) nodes and \( m \) edges, where the node set \( V = A \cup B \) consists of two disjoint subsets \( A \) and \( B \) of sizes \( n_1 \) and \( n_2 \) (so \( n = n_1 + n_2 \)). Nodes in \( A \) are “healthy”, while nodes in \( B \) are “infected”. Given a source \( s \in A \), your goal is to compute the shortest distance from \( s \) to every other healthy node which can pass through at most one infected node (i.e., if the path from \( s \) to \( v \) contains at most one infected \( u \), this is OK, but if it contains two or more, this path is not allowed when computing the shortest distance).

Define the following directed graph \( G' = (V',E') \) on \( 2n_1 + n_2 \) nodes. The vertex set of \( V' \) of \( G' \) is \( V' = A_1 \cup B' \cup A_2 \), where \( A_1 \) and \( A_2 \) are two copies of healthy nodes \( A \), and \( B' \) is a copy of \( B \). Two nodes in \( A_1 \) are connected in \( G' \) if and only if they are connected in \( G \), the same between
two nodes in \( A_2 \). The nodes in \( B \) are not connected to each other. For every original incoming edge \((a, b) \in E\), where \( a \in A \) and \( b \in B \), we put an edge \((a_1, b) \in E'\), where \( a_1 \) is the copy of \( a \) in \( A_1 \). Similarly, for every original outgoing edge \((b, a) \in E\), where \( a \in A \) and \( b \in B \), we put an edge \((b, a_2) \in E'\), where \( a_2 \) is the copy of \( a \) in \( A_2 \).

(a) Let \( n', m' \) be the number of vertices and edges in \( G' \). Show that \( n' \leq 2n \) and \( m' \leq 2m \).

(b) Recall our original problem of computing the required shortest distance in \( G \) from \( s \) to every other healthy node \( a \in A \) which can pass through at most one infected node \( b \in B \). Call this distance \( a[\text{dis}] \). Let \( s_1 \) and \( s_2 \) be the two copies of \( s \) in \( G' \). Using one “appropriate” BFS call on \( G' \), show how to compute the values \( a[\text{dis}] \). Specifically, say what is the starting node (call it \( s' \)) of your BFS call in \( G' \). Also, after your BFS call computed shortest distances \( v'.d \) from \( s' \) to \( v' \), for every \( v' \in V' \), show how to compute the desired values \( a[\text{dis}] \) for the problem at hand (i.e., write an explicit formula for \( a[\text{dis}] \) using appropriate \( v'.d \) values). Justify your algorithm.

(c) Show that the running time of your procedure is \( O(m + n) \).

**Problem 10-4 (Sokoban) 15 Points**

An \( n \times n \)-grid is a graph \( G \) with \( n^2 \) vertices \( \{(i, j) \mid 1 \leq i, j \leq n\} \) and edges such that two vertices \( u \) and \( v \) are connected if and only if they are at distance 1.

The game of Sokoban is formalized as follows: Let \( S = (V, E) \) be a subgraph of the \( n \times n \)-grid, i.e., \( S \) can be obtained by deleting all vertices of the grid not contained in \( V \) and the edges adjacent to them. Imagine that at the onset of the game, some vertices \( I \subseteq V \) on subgrid \( S \) have a crate on them and that there is one special vertex \( w \in V \setminus I \) where Waldo stands. Moreover, there is a set of target positions \( T \subseteq V \) with \( k := |T| = |I| \) for the crates. Waldo’s goal is to move the crates in such a way that in the end there is a crate on each vertex in \( T \).

At any point, Waldo can move freely about the subgrid vertices not covered by crates. He can move an adjacent crate by standing behind it and pushing by one vertex, provided there is no other crate in the way. For example, if Waldo is at \((i, j) \in V\), there is a crate at \((i + 1, j) \in V \) and no no crate at \((i + 2, j) \), he can push the crate to \((i + 2, j) \in V \); Waldo ends up at \((i + 1, j) \) in such a case.

(a) (5 points) Show that the number of different states of this game is at most \((n^2)^{k+1}\).

(b) (3 points) Describe a winning state of this game and argue that there are at most \( n^2 k! \) of them.

(c) (7 points) Devise an \( O\left((n^2)^{k+1}\right)\)-algorithm that on input \( S = (V, E) \) figures out the fastest way for Waldo to solve this problem (i.e., to move the crates from their initial positions \( I \) to their target positions \( T \)) or reports that there is no solution.

Argue why your algorithm works and achieves the desired running time.

(Hint: Using one of the algorithms you have seen in class, explore a state graph corresponding to Sokoban, in which the vertices are the various states and two states are connected if and

---

1 Feel free to watch the animation on Wikipedia.
only if in one move of the game (either Waldo only or Waldo and a crate change positions) one can get from one to the other.

**Problem 10-5 (BFS/DFS on a Heap)**  

Consider a full binary tree of height $h$, i.e., with $2^{h+1} - 1$ nodes.

(a) (2 points) Let $h = 3$. Draw the tree and assign to each node the discovery time during a **BFS** starting at the root (which is discovered at time 1) of the heap and visiting children left to right.

(b) (2 points) Let $h = 3$. Draw the tree and assign to each node the discovery time during a **DFS** starting at the root (which is discovered at time 1) of the heap and visiting children left to right.

(c) [Extra credit] (4 points) Describe a permutation $f$ on $\{1, \ldots, 2^{h+1} - 1\}$ that maps the BFS discovery time to the DFS discovery time for each node. Write a closed formula for $f(x)$ in terms of $x$, $h$, and the binary representation $b_0, \ldots, b_{h+1}$ of $x$ (i.e., $x = \sum_{i=0}^{h+1} b_i \cdot 2^i$).