Cop and Robber spoiler

Idea by Vytautas Gruslys. Spoiler by Marijonas Petrauskas

This is a game theory problem. Let's denote the current cop's position as c, current robber's position as r, and who goes next as t ($t \in \{C, R\}$, C is for cop and R is for robber). The winning positions can be defined as follows. Cop wins the game ($W[c, r, t] = \texttt{true}$) in the following cases:

1. Cop and robber are in the same node ($c = r$, $t \in \{C, R\}$);
   Reasoning: Cop has already won the game.

2. It's cop's turn and he has at least one possible move that leads to a winning position ($t = C$, $\exists c'((A[c, c']$ or $c = c')$ and $W[c', r, \neg t])$);
   Reasoning: Cop can make any move. If there is a move that leads to a winning position, he can make it and win the game.

3. It's robber's turn and all possible moves lead to a cop's winning position ($t = R$, $\forall r'(A[r, r']$ and $W[c, r', \neg t])$).
   Reasoning: If the robber can make the cop lose, he will definitely do it. Cop can win only if every possible robber's move leads to a winning position.

Let's now build a directed graph G_s in which each node represents a game state (c, r, t). The graph will have $2N^2$ nodes in total. Let the graph have edges that represent valid game moves:

1. edges from $(c_{\text{before}}, r, C)$ to (c_{after}, r, R) if $(c_{\text{before}} = c_{\text{after}})$ or $A[c_{\text{before}}, c_{\text{after}}]$;

2. edges from $(c, r_{\text{before}}, R)$ to (c, r_{after}, C) if $A[r_{\text{before}}, r_{\text{after}}]$.

Any path in the new graph corresponds to a possible move sequence in the game. We can use the above definition of winning positions to fill in $W[c, r, t]$ array. The easiest way to do it is to do a breadth-first search in the game state graph, starting from the known winning positions (where $c = r$) and following the edges backwards. The only exception is that robber positions should marked as winning only if all moves lead to winning positions. This requirement can be checked on every step, which leads to an $O(N^4)$ solution. Alternatively, the number of winning moves can be tracked in an array to achieve an $O(N^3)$ solution that scores full points. In order to determine whether cop can win the game, we need to find a starting node c such that $W[c, r, C]$ is true for all r. During the breadth-first search we also need to build a table of cop's winning moves that will be later used by nextMove() function. For more details, refer to the sample solution.

Partial solutions

Tree

When the graph is a tree, the cop can always win using a simple strategy. There is only one way for the cop to catch the robber. A simple depth-first search may be used to find the cop's neighbor that is closest to the robber.

Grid

In this subtask the cop can always win too. The first task is to find the width of the grid. Notice that there are 4 corner nodes, each of which has exactly two neighbors. Suppose the dimensions of the grid are $w \times h$, then the corner nodes are 0, $w-1$, $w(h-1)$ and $wh-1$. Thus, if x is the second node with exactly two neighbors, $x = w - 1 \implies w = x + 1$. Now that we know the dimensions of the grid we can identify cop's position (c_x, c_y) and robber's position (r_x, r_y). The winning strategy is:

- If $|c_x - r_x| < |c_y - r_y|$, move towards the robber along the y-axis;
- If $|c_x - r_x| > |c_y - r_y|$, move towards the robber along the x-axis;
- If $|c_x - r_x| = |c_y - r_y|$, wait.