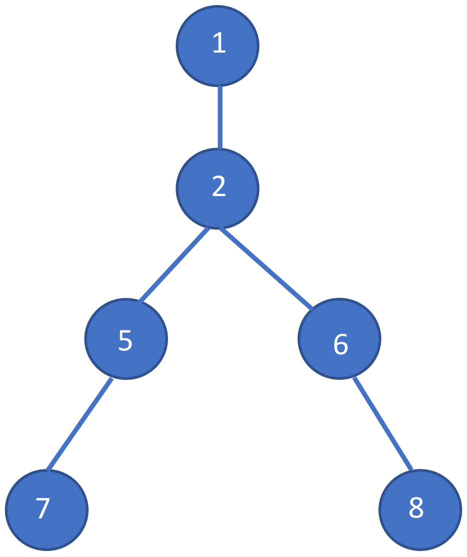
5) We first construct a graph $G = (V, E)$ for this situation. The nodes V are the actors and there is an edge $(u, v) \in E$ if actors u and v have appeared in some movie together. This procedure is $O(n^2)$, where n is the number of actors. It is obvious that when you're on an actor's node, for the next step your opponent can only pick one of the node's neighbors.

Now we're given a game instance where actors (i.e. nodes) $v_1, v_2, \dots, v_i$ have been visited, let $W = \{v_1, v_2, \dots, v_i\}$. We now compute all possible paths from v_i to each node u_j in $V \setminus W$, if there is a path from v_i to u_j. Let $n = |V|$ and $m = |E|$. This procedure's runtime is $O(n^2(n + m))$ because the runtime of computing every possible path between two nodes is $O(n(n + m))$ due to BFS being $O(n + m)$ and visiting $O(n)$ nodes from each vertex, and we're doing that $O(n)$ times.

After generating these paths, we get rid of the ones where the last node visited still have neighbors who weren't visited earlier in the path. This procedure is $O(n^2)$ because for each of the $O(n)$ paths, we see if its $O(n)$ neighbors have been visited.

With these paths, we construct a tree. For example, if two paths looked like [1→ 2→ 5→ 7→] and [1→ 2→ 6→ 8→], then their tree would look like



This procedure is of runtime $O(n^2)$.

After the tree has been constructed, we compute who can force a win, if anyone can. If the last move made in the game instance we were given was by P_i, then P_{1-i} can force a win if there's any tree whose root is at the 2nd level from the top where all of its leaf nodes are at a level where P_{i-1} places. To know if P_i can force a win, we check if all the leaf nodes are at a level where P_i places. This procedure is $O(n^2)$ because we just check the parity of the length of each path.

The whole algorithm's runtime is $O\left(n^2(n+m)\right) = \boldsymbol{O(n^3 + n^2 m).}$

This algorithm is **correct** because the tree is a correct representation of all the possible choices the players can make for the rest of the game, when proceeding from the current instance of the game. Each path down the tree is a sequence of actors who have co-starred with the actors immediately preceding and following him, and the leaf nodes for each path is when the game can no longer proceed because all of the leaves' co-stars have already been mentioned earlier in the game. The procedure for determining whether P_{1-i} can force a win is correct because the topmost level is P_i, so the 2nd topmost layer is P_{1-i}'s choice, and he can choose to go down any branch from there because it's his turn, and if and only if there is a branch where all of whose paths end at his turn, can he win; if there's even one path where it ends at P_i's turn, he cannot guarantee a win. The procedure for determining whether P_i can force a win is correct because at the current turn (P_{1-i}'s turn), P_i is at the whim of P_{1-i}, so if and only if all of the paths in the tree end at P_i's turn can he guarantee a win.