CS3230 : Tutorial - 5

Bakh Khoussainov

The deadline is 1pm, 18-Sept-2012.

1. Design an algorithm that given a graph G detects if G has a cycle. Determine the running time of your algorithm.

2. Recall that a path in a graph is **simple** if no vertices in it are repeated. Design an algorithm that given a graph G, and two vertices x and y decides if there is a unique simple path from x to y. Determine the running time of your algorithm.

3. Suppose that P and Q are some properties of graphs. Assume that there are two polynomial time algorithms such that the following hold. The first algorithm detects if a given graph G has property P, and similarly, the second algorithm detects if G has property Q. Prove that there are polynomial time algorithms that can decide each of the following properties: (a) P or Q, (b) P and Q, and (d) not P.

4. A **bipartite graph** is a directed graph $G = (V, E)$ such that there are two sets of vertices V_0 and V_1 for which the following properties are true:

   (a) $V_0 \cap V_1 = \emptyset$ and $V_0 \cup V_1 = V$.

   (b) The set E of edges is a subset of $V_0 \times V_1 \cup V_1 \times V_0$.

   Do the following:

   (a) Design an algorithm that, given a graph, detects if the graph is bipartite. Your algorithm does not have to be efficient.

   (b) Do you think there is an efficient algorithm that detects if a given graph is bipartite? If so, give an idea of the algorithm and provide a pseudo-code.

5. Let (V, E) be a bipartite graph given with V_0 and V_1 (see Problem 4). Let T be a subset of V. Call T the target set. We define **reachability game** Γ as follows. There are two players: *Player* 0 and *Player* 1.

   A play between these two players is described as follows. The play starts at any vertex v_0. Say the vertex is in V_0. In this case, *Player* 0 selects an edge $e = (v_0, v_1)$, and then moves along the edge. Then, *Player* 1 selects an edge $e = (v_1, v_2)$, and moves along the edge. This continues turn by turn. If v_0 is in V_1 then *Player* 1 starts the play. So, a **play** in $G = (V_0 \cup V_1, E)$ is a (finite or infinite) sequence of vertices $v_0, v_1, v_2, \ldots,$ such that (v_0, v_1), (v_1, v_2), $(v_2, v_3), \ldots$ are all edges of G. Note that *Player* 0 moves from vertices in V_0, and *Player* 1 moves from vertices in V_1. We say that Player 0 **wins the play** $v_0, v_1, v_2, \ldots$ if there exists an i such that $v_i \in T$. Otherwise, Player 1 wins the play.

Assume that the game starts from a vertex v. The vertex v is **winning** for Player 0 if, starting from v, the player can reach the set T no matter what moves the opponent makes. Similarly, the vertex v is **winning** for Player 1 if, starting from v, the player can keep the plays out of T forever no matter what moves the opponent makes. Do the following.

(a) Consider the game pictured below. For this game, V_0 is the set of all vertices labeled by letters, and V_1 is the set of all vertices labeled by numbers, and $T = \{a, b\}$. Find all winning vertices for Player 0. Similarly, find all winning vertices for Player 1.

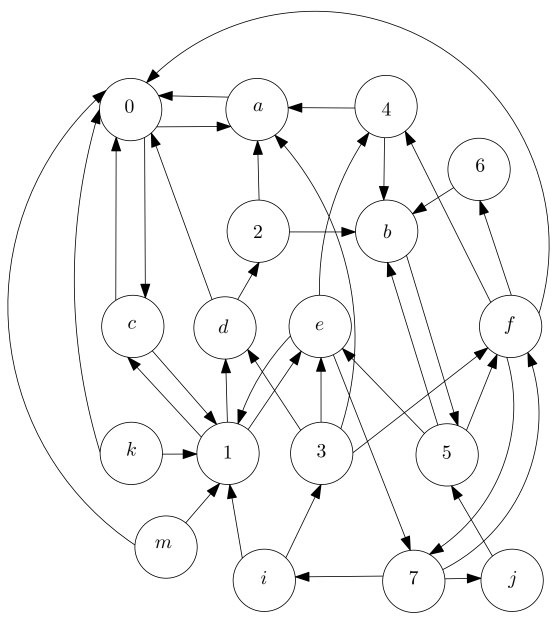

Figure 1: An example of a reachability game

(b) Design an efficient algorithm that, given a game Γ, computes all the winning vertices for Player 0, and computes all winning vertices for Player 1.

(c) Prove that your algorithm that solves the game problem is correct.

(d) Design an algorithm that, given a game Γ and a starting vertex v, builds a winning strategy from the vertex v for the winner.