1 Question 1

Consider a game graph $G = (V, E, v_0, F)$, where the (finite) set of vertices $V = V_1 \bigcup V_2$ is partitioned into set of vertices V_1 (belonging to player 1) and set of vertices V_2 (belonging to player 2), E being the set of edges of G. We are also given a start vertex $v_0 \in V$, and a set $F \subset V$ of good (target) states. Denote $E(v)$ the set of all succesor vertices for v, i.e $E(v) = \{v' \in V | (v, v') \in E\}$. We assume that $\forall v, E(v) \neq \varnothing$, i.e every state has a successor state (no deadlocks). Let Π denote the set of infinite paths in G. For $\pi \in \Pi$, $\pi = v_0 v_1 ...$, let us denote the set of states that appear infinitely often in π as

$$inf(\pi) = \{v \in V | \forall i \geq 0, \exists j \geq i, v_i = v\}$$

The play π is a win for player 1 if $inf(\pi) \cap F \neq \varnothing$, and otherwise it is a win for player 2 (i.e a loss for player 1). Describe an efficient algorithm to find which player wins, and to extract a memoryless winning strategy, given such a game.

1.1 Algorithm 1

First, at iteration 1, we calculate the set of states F_1' where player 1 can *force* to reach a state in F at least once. In other words, find the states from which player 1 has a strategy to reach a state in F at least once against all strategies of player 2.

Then, we calculate the set of all vertices T_1 from which player 2 has a strategy to reach $V \setminus F_1'$ (regardless of what player 1 does). It is easy to see that $V \setminus F_1'$ is winning for player 2. Also, T_1 is winning for player 2 as well. Observe that the winning set of nodes for player 1 have to be contained in $V \setminus T_1$ and hence we only need to consider the game on the reduced graph $G' = (V', E') := G \setminus T_1$.

We continue repeating this procedure for the reduced game graph G', at iteration 2, where we calculate the set of states F_2' where player 1 can force to reach a state in $F \cap V'$ on the reduced graph G' . We calculate the set of all vertices T_2 from which player 2 has a strategy to reach $V \setminus F_2'$ on the subgraph G' and then we reduce the game graph to $G'' := G' \setminus T_2$.

We repeat this process until we cannot remove any more vertices (reach a fixed point), in which case the set of remaining nodes is the set of winning vertices for player 1. Note that since F is finite, there must always be the case that the algorithm terminates. Observe that at each iteration i, by construction, F_i' and T_i partition the game graph.

We claim that after the final iteration k (considering that after k iterations the algorithm terminates), the set F_k' consists only of winning states for player 1. Suppose that there is a state $v_i \in F_k'$ such that $v_i \in V_2$ and has an outgoing edge towards T_k (it is a player 2's state in F_k' that has an outgoing edge in T_k). But by construction of T_k, it must be the case that v_i would have been removed

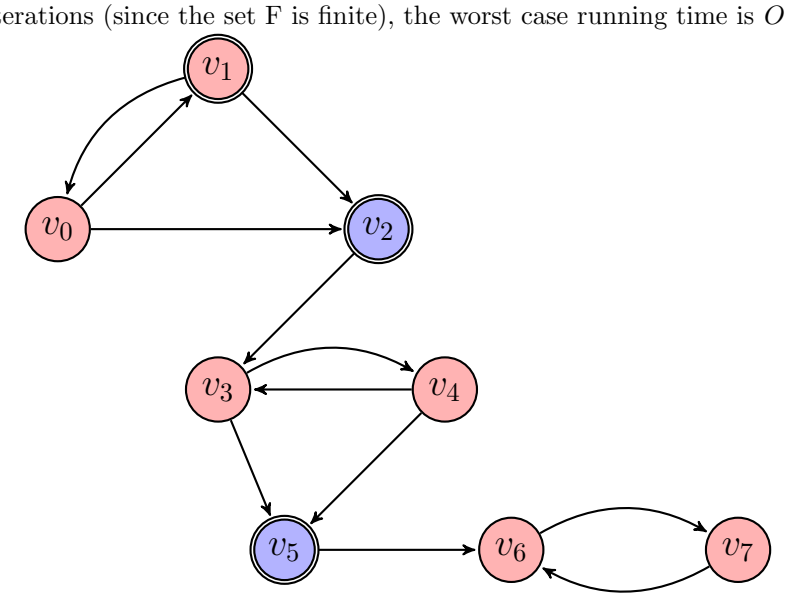
from the game graph in a previous iteration. Contradiction. Therefore, the set F'_k must only contain winning states for player 1. If $v_0 \in F'_k$, then player 1 wins, otherwise, player 2 wins.

Now we show that player 1 has a memoryless winning strategy for all states in F'_k.

- for any state $v_i \in F'_k \cap F$, player 1 is guaranteed to stay in F'_k
- for any state $v_i \in F'_k \setminus F$, use a memoryless deterministic strategy to reach F

By construction of T_k, player 2 also has a memoryless winning strategy.

Since at each iteration i, computing the set F'_i requires $O(|E|)$ time (by use of algorithm described in class), and because the algorithm runs $O(|V|)$ iterations (since the set F is finite), the worst case running time is $O(|E||V|)$.



$G = (V, E)$, where $V = \{v_0, v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$. The red nodes belong to player 1, whereas the blue nodes belong to player 2. $F = \{v_1, v_2, v_5\}$.

Iteration 1.

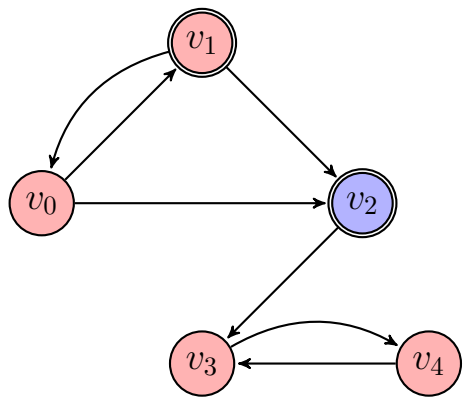
$F'_1 = F \bigcup \{v_0, v_3, v_4\}$.

$V \setminus F'_1 = \{v_6, v_7\} \neq \varnothing$ (we still can remove vertices)

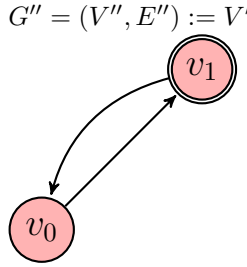
$T_1 = (V \setminus F'_1) \bigcup \{v_5\}$

$G' = (V', E') := G \setminus T_1$. $F \cap V' = \{v_1, v_2\}$

Then, G' becomes:



Iteration 2:
$F_2' = (F \cap V') \bigcup \{v_0\} = \{v_0, v_1, v_2\}$
$V' \setminus F_2' = \{v_3, v_4\} \neq \varnothing$ (we still can remove vertices)
$T_2 := (V' \setminus F_2') \bigcup \{v_2\}$
Iteration 3 :
$G'' = (V'', E'') := V' \setminus T_2$. $F \cap V'' = \{v_1\}$ Then, G'' becomes:



$F_3' = (F \cap V'') \bigcup \{v_0\} = \{v_0, v_1\}$
$V'' \setminus F_3' = \varnothing$ (we cannot remove any more vertices!)
Return the winning set for player 1, i.e $\{v_0, v_1\}$.

1.2 Algorithm 2

Let us compute F_i , for $i \geq 1$, to be the set of good (target) states $f \in F$ from which player 1 can force to visit F at least i times, i.e the set of final states such that player 1 has a strategy to visit F at least i times, regardless of what player 2 does. Then, we show that (in lemma 1.1):

$$... \subseteq F_3 \subseteq F_2 \subseteq F_1 \subseteq F_0 = F$$

We define:

$$F_0 = F$$

$$F_{i+1} = F \cap Win_1'(F_i)$$

where for any set of nodes $S \subseteq V$,
$Win_1'(S) = \{v \in V | pl.1\ can\ force\ reaching\ S\ starting\ from\ v\ in\ one\ or\ more\ moves\}$

Note that computing Win_1' boils down to applying the easy bottom up algorithm found in class.

Denote $\mathcal{F} = \bigcap_i F_i$

Lemma 1.1. $F_{k+1} \subseteq F_k, \forall k \geq 0$

Proof. Induction on k.

Base case: $k = 0$. $F_1 = F \cap Win_1'(F) \subseteq F$

Induction step:

By induction hypothesis (I.H), $F_k \subseteq F_{k-1}$. Observe that Win_1' is monotonically decreasing.

From the definition, we have that $F_{k+1} = F \cap Win_1'(F_k)$ and using I.H and the fact that Win_1' is monotone, $F_{k+1} = F \cap Win_1'(F_k) \subseteq F \cap Win_1'(F_{k-1}) = F_k$ □

Since F is finite, it must be the case that there exists a k, such that $\mathcal{F} = \bigcap_i F_i = \bigcap_i^k F_i = F_k$. In other words, $\mathcal{F} = F \bigcap F_1 \bigcap F_2 \bigcap ... \bigcap F_k$, where for every $j > k$, $F_j = F_k$, i.e all terms F_j after F_k will be in fact equal to F_k.

We show that the states in $Win_1'(\mathcal{F})$ are winning for player 1, by showing how to construct a memoryless strategy. We know that $\exists k$ s.t $F_{k+1} = F_k = F \bigcap Win_1'(F_k)$. Therefore, using this fact and the way F_k is constructed, we have that for a node v_i belonging to player 1 and $v_i \in F_k$, player 1 will choose an edge that will lead to $Win_1'(F_k)$. If a node v_i belongs to player 2 and $v_i \in F_k$, all edges will go back to $Win_1'(F_k)$.

If $v_i \in Win_1'(\mathcal{F}) \setminus \mathcal{F}$, player 1 uses a memoryless strategy (by algorithm in class) to reach $\mathcal{F}$.

Now we need to compute a memoryless winning strategy of player 2 for all states in $V \setminus Win_1'(\mathcal{F})$. The argument is similar to the one developed for player 1 winning strategy, needing to prove that from $V \setminus Win_1'(\mathcal{F})$, player 1 can force at most only a finite number of visits to F.

Remark. *Algorithm 1 and Algorithm 2 do the same thing, by computing the winning set for the corresponding player. The only difference is the fact that the algorithm 1 computes ad-hoc at each iteration i the states from which player 1 can force to reach F at least i times (on game graph G), whereas algorithm 2 prepares at each iteration i the set of* ***final*** *nodes from which player 1 can guarantee to reach F at least i times and then, after it reaches a fixpoint (at iteration k), it calculates the set from which player 1 can force to reach $\mathcal{F}$.*

Each F_i at iteration i of algorithm 2 requires $O(|E|)$ running time for being computed, and since there are $O(|V|)$ iterations, the running time of the algorithm 2 is $O(|E||V|)$.