Homework #3 Solutions

Contact TAs: `ada01@csie.ntu.edu.tw`

Problem 1

One pawn

First, lets consider the simplified version of the problem where there is only one pawn while all other conditions remain the same. This problem can be converted to a graph problem as follows: Vertices: The set of all possible pawn positions (i, j). This is the set of all cells without obstacles on them. Edges: For any given cell C_{ij} , there can be at most 4 valid moves that can be performed when the pawn is present on that cell. Since the pawn can only move horizontally or vertically, let the cells which can be reached from C_{ij} in one move be: C_{ij_1},C_{ij_2},C_{i_1j},C_{i_2j}. We add directed edges from C_{ij} to all the 4 cells.

Note that our graph needs to be directed since it may not be possible to reach a cell back. For example, look at the edge labeled 4 in the figure. Now given the starting position of the pawn, we want to find whether the target can be reached. This translates to finding whether a path exists from the source(starting position) to the target in the graph.

Correctness: The number of vertices is the number of pair of squares which is $O(n^2)$. The number of edges is at most 4 times the number of vertices and hence is $O(n^2)$. Finding each edge can take $O(n)$ time. Finally, DFS takes $O(|V| + |E|)$. Thus, the total time complexity is $O(n^3)$.

two pawns

Now we tackle the original problem which has two pawns, both of which can be moved. Imagine what would happen if we used the same graph as the simplified problem. While adding the edges for a given pawn position, we do not know the position of the other pawn. Note that this information is necessary since the other pawn acts as an obstacle when the current pawn is being moved. Thus, we need to consider the position of both the pawns.

We now construct the graph for the problem with two pawns: Vertices: The set of all possible pair of pawn positions (i,j,k,l). (i,j) denotes the position for pawn 1 while (k,l) denotes the position for pawn 2. Edges: For a given vertex (i,j,k,l), there are at most four moves for each of the two pawns. Since only one pawn can move at a time, there are 8 possible moves from a given pair of pawn positions. We add edges for these 8 moves. Again, we translate this to a graph problem. The source is pair of starting positions for the two pawns. The target is slightly tricky: there are multiple target vertices. Any vertex in which one of the pawns is in the target square is a target vertex.

Algorithm

```
Input: Puzzle board, pawn starting positions, target square
Output: Can one of the pawns reach target square: Yes/No
Let V be the set of vertices
Let E be the set of the edges
For each square (i,j):
    For each square (k,l):
        If (i,j) != (k,l):
            Add (i,j,k,l) to V

For each (i,j,k,l) in V:
    For all (i',j') reachable from (i,j) and all (k',l') reachable from (k,l) in o
        Add ((i,j,k,l) -> (i',j',k,l)) to E
        Add ((i,j,k,l) -> (i,j,k',l')) to E

Use DFS from starting pawn positions to find all reachable vertices
```

Correctness: The number of vertices is the number of pair of squares which is $O(n^4)$. The number of edges is at most 8 times the number of vertices and hence is $O(n^4)$. Finding each edge can take $O(n)$ time. Finally, DFS takes $O(|V| + |E|)$. Thus, the total time complexity is $O(n^5)$.

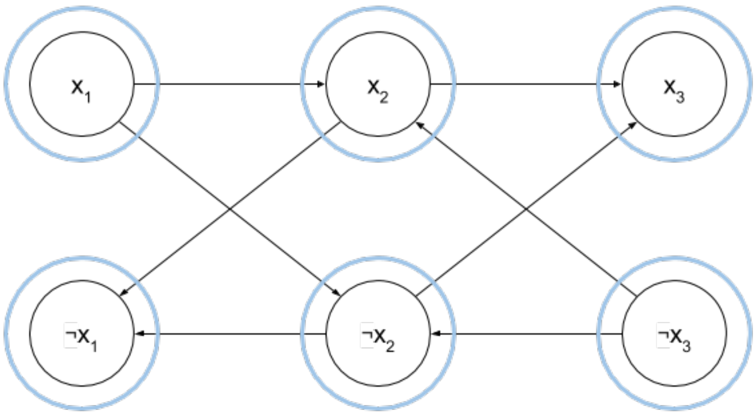
Problem 2

(1) 1. Call $DFS(G)$ to compute finishing times for each vertex
 2. Compute transpose graph G^T
 3. Call $DFS(G^T)$ in the order of decreasing finishing times
 4. Each tree in the step 3 is a strongly connected component

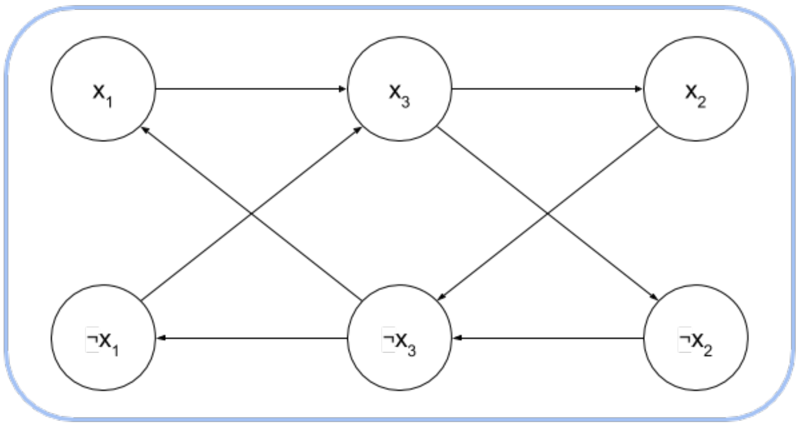
(2) 1. Call $SCC(G)$ on implication graph
 2. Check whether any strongly connected component contains both a variable and its negation. If so, return unsatisfiable, else, return satisfiable.

Explanation:
If a strongly connected component contains both x and $\neg x$, there exist both a path $x \to ... \to \neg x$ and a path $\neg x \to ... \to x$. To satisfy $x \to ... \to \neg x$, x must be $false$. To satisfy $\neg x \to ... \to x$, x must be $true$. It is a contradiction $\Rightarrow\Leftarrow$

(3) (a) Every vertex is a strongly connected component. Satisfiable.



(b) The whole graph is in a strongly connected component. Unsatisfiable.


x1
x3
x2
¬x1
¬x3
¬x2