CS180 Homework 1

Due: 4:00pm, 1/18/2017

1. A celebrity among n persons is someone who is known by everyone but does not know anyone.

   (a) Give an iterative algorithm to find the celebrity among n people.

```
CANDIDATE(S)
  for i ← 1 to n − 1
      choose two person x and y
      if x know y
          eliminate x from G
      else
          eliminate y from G
      c ← the last person in G
  if ISCELEBRITY(c)
      return c
```

```
ISCELEBRITY(c)
  for every other person i in the group of people
      if i does not knows c or c knows i
          return FALSE
  return TRUE
```

   The algorithm is correct because the CANDIDATE algorithm eliminate the people that are not celebrity and ISCELEBRITY verifies the candidate is a true candidate.

   (b) What is the complexity of your algorithm?

   Assume the relationship between two people can be accessed in constant time, the algorithm runs in $O(n)$ time because each step takes $O(1)$ time and each loop is executed at most n times.

2. Suppose you are playing the game of NIM. This game begins with a placement of n rows of matches on a table. Each row i has m_i matches. Players take turns selecting a row of matches and removing any or all of the matches in that row. Whoever claims the final match from the table wins the game. This game has a winning strategy based on writing the count for each row in binary and lining up the binary numbers (by place value) in columns. We note that a table is *favorable* if there is a column with an odd number of ones in it, and the table is *unfavorable* if all columns have an even number of ones.

   Example: Suppose we start off with three rows of matches of 7, 8 and 9. The binary representations of number of matches are $7 = 0111, 8 = 0111, 9 = 1001$. Therefore, the board will look like this

$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

   Since the third row from the right and the second row from the right, each has odd numbers of ones, the table is favorable.

(a) Prove that, for any favorable table, there exists a move that makes the table unfavorable. Prove also that, for any unfavorable table, any move makes the table favorable for one's opponent. Write the algorithm that on an input of favorable table outputs the row and the number of matches to remove from that row, to make the table unfavorable. **Solution:**

The first thing to prove is that, for any favorable table, there exists a move that makes the table unfavorable for one's opponent. The easiest way to prove that something exists is by construction: that is, show how to take a favorable table and select a move to produce an unfavorable table.

If the table is favorable, this means that there is at least one column that has an odd sum. Inspect the most significant column with an odd count. Select an arbitrary row with a one in that column; this is the pile of matches from which we will remove some. Change the 1 in that column to a 0 (note that this change alone is sufficient to show that the resulting number of matches is smaller than the original count, and thus a valid move, no matter how many other columns we change). Change any other columns, if necessary, to produce even parity. This will produce an unfavorable table for the opponent.

We also need to prove that, given an unfavorable table, any move produces a favorable table for one's opponent. By definition of unfavorable table, there is even parity for each column. However, we may only select one row, and must remove at least one match from that row. As a result, at least one bit will change, and will change in only one row. Any bit change will modify the parity of the column, and as such, this will produce odd parity in at least one column, and thus a favorable table for the opponent.

(b) Given an input of favorable table, can you determine whether there exists multiple ways to make the table unfavorable? How?

If the most significant column with odd ones has three or more "1"s , then you can chose any row with a "1" and make the table unfavourable

(c) Give a winning strategy for this game.

The loser of the game will see no match on the table, which is a unfavorable table. So, a wining strategy is always leave a unfavorable table to the other.

3. Given a connected undirected graph $G = (V, E)$, and let O be the set of nodes with odd degrees. Two paths are edge-disjoint if they do not share any edge. Give an algorithm that partitions O into pairs such that each pair can be connected by a path and all the $|O|/2$ paths connecting pairs are edge disjoint. For a set S, the notation $|S|$ is the number of elements in the set S.

```
PATITION(G)
  While there exist odd nodes that are not paired
      chose two arbitrary odd nodes A and B in the same connect component
      pair A and B, select an arbitrary path between them and remove that path
```

Next, we prove the correctness of the algorithm:

Theorem 1. *The above algorithm gives a correct partition of odd nodes.*

Proof. In the execution of the algorithm, it is possible that after removing the path, the graph becomes several connected components. Hence, in each iteration we always take two odd nodes in the same connected component. We prove our algorithm actually works even if the graph is

not connected. The proof is by induction on the number of edges. The base case is the graph has only one edge, then it is vacuously true since there is no odd node. Assume that the claim is true for any undirected graph that has $|E| \leq m-1$ edges. Consider an undirected graph with m edges. Each component is a connected graph by itself, it should have an even number of odd vertices (since the number of total degree of the component is even). Therefore, if there exists an unpaired odd node, we can always pick two odd nodes in the same component. Moreover, by removing a path between these two odd nodes, the number of edges in the graph is less than m, and so the rest is solved by our induction hypothesis. Since we remove the path once we pair two odd nodes, all paths we used are edge-disjoint. □

4. A sink in a directed acyclic graph (DAG) is a node that has no outgoing edge.

   (a) Given a DAG $G = (V, E)$, prove that there exists a sink node.

   If the DAG has no sink node, then every node has a outgoing edge. We pick an arbitrary node v and choose an arbitrary outgoing edge to go the next node. Since every node has a outgoing edge, we can repeat the process for $|V|+1$ times. Hence, we must visit some node more than once and so it contains a cycle.

   (b) Consider the following strategy to eliminate the sink in a DAG:

```
While there exists a sink s in the graph G
    Reverse all incoming edges of s to outgoing edges
```

   In the worst case, how many steps does the above algorithm run before terminating?

   Consider a DAG with two nodes A and B and an edge $A \to B$. The above algorithm will run for infinite time since there is always a sink.