CS180 Homework 1

Due: 8:00pm, 1/17/2019

1. In the class we showed how to construct the binary representations for numbers $1, 2, 3, ..., n$ by doing binary addition:

   ```
   BINARYONETON(n)
     X ← 0
     for i ← 1 to n
         starting from right to left in X, find the first digit that is 0 and assume it is the k^th digit
         X ← flip the k^th digit of X to 1 and flip 1, 2, ..., (k−1)^th digit of X to 0
         print X
   ```

   The bit complexity of an algorithm counts the number of bit operations in terms of the length of the input. What is the bit complexity of the algorithm BINARYONETON?

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

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

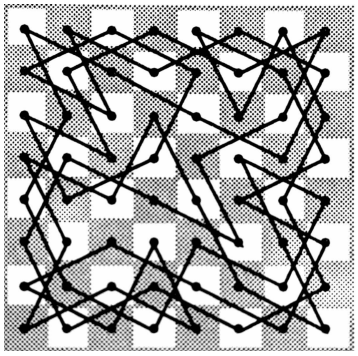
   Since the third row from the right and the second row from the right, each has odd numbers of ones, the table is favorable.

   (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, in order to make the table unfavorable.

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

   (c) Design an algorithm that always wins the game if given a favorable table.

3. Recall that a knight can make one of eight legal moves. A closed knight's tour on an 8×8 chessboard, that is, a circular tour of knight's moves that visits every square of the chessboard exactly once before returning to the first square. Such a tour is also known as a Eulerian tour.



A closed knight tour for 8×8 chessboard

(a) Given a graph $G = \{V, E\}$. G contains two cycles C_1 and C_2 where each cycle visits half of the vertices exactly once and C_1 and C_2 do not share any vertex. Suppose there exists a circle S in G made of four edges: $S = \{(v_a, v_b), (v_b, v_c), (v_c, v_d), (v_d, v_a)\}$, where (v_a, v_b) is an edge in cycle C_1 and (v_c, v_d) is an edge in cycle C_2, while the two other edges are neither in C_1 nor in C_2. Show that then there exists a single cycle in G that visits every vertex in G exactly once.

(b) Give an algorithm generates a closed knight's tour on a 16×16 chessboard. (Hint: you may use a closed knight's tour for 8×8 as a starting point)

(c) Give an algorithm generates a closed knight's tour on any $2^k \times 2^k$ chessboard for all $k \geq 3$.

4. Given an array $B[1..n]$ that stores a binary representation of a positive integer, where each $B[i]$ is either 0 or 1. For example, $B = [1, 1, 0]$, which represents 6 in binary.

(a) Design a recursive algorithm that evaluates the actual value of the array B represents. Your algorithm is NOT allowed to use any pre-calculated table for powers of 2. For example, your algorithm cannot use $2^3 = 8$ or $2^4 = 16$ as a known fact without actually calculating it.

(b) Unfold the recursive algorithm (in which a procedure call on itself with a different parameter) into an *iterative* algorithm that does not call any procedure.

★ Express your algorithm in a well-structured manner. Pseudo code in the textbook has good examples to follow. Avoid using a long continuous piece of text to describe your algorithm. Start each problem on a NEW page. Unless specified, you should justify the time complexity of your algorithm and why it works. For grading, we will take into account both the correctness and the clarity. Your answers are supposed to be in a simple and understandable manner and sloppy answers are expected to receiver fewer points.

★ Homework assignments are due on Gradescope. Email attachments or paper submissions are NOT acceptable.

★ Raw photo is NOT acceptable. Upload your homework as a PDF scan by using a scanner or mobile scanner app. Match each problem with your answer on Gradescope. Use dark pen or pencil and your handwriting should be clear and legible.

★ We recommend using LaTeX, LyX or other word processing software for writing the homework. This is **NOT** a requirement but it helps us to grade and give feedback.