CS 70 Discrete Mathematics and Probability Theory
Summer 2017 Lu, Moulos, and Tang

DIS 1C

1 Fibonacci for Home

Recall, the Fibonacci numbers, defined recursively as

$F_1 = 1$, $F_2 = 1$ and $F_n = F_{n-2} + F_{n-1}$.

Prove that every third Fibonacci number is even. For example, $F_3 = 2$ is even and $F_6 = 8$ is even.

2 Natural Induction on Inequality

Prove that if $n \in \mathbb{N}$ and $x > 0$, then $(1+x)^n \geq 1 + nx$

3 A Tricky Game

(a) CS 70 course staff invite you to play a game: Suppose there are n^2 coins in a $n \times n$ grid ($n > 0$), each with their heads side up. In each move, you can pick one of the n rows or columns and flip over all of the coins in that row or column. However, you are not allowed to re-arrange them in any other way. You have an unlimited number of moves. If you happen to reach a configuration where there is exactly one coin with its tails side up, you will win the game. Are you able to win this game? Find all values of n for which you can win the game, and prove your statement. In other words, for each value of n that you listed, prove that you can win the game; then, prove that it is impossible to win the game for all other values of n.

(b) (Optional) Now, suppose we change the rules: If the number of "tails" is between 1 and $n-1$, you win. Are you able to win this game? Does that apply to all n? Prove your answer.

4 Binary Search

We have an array A, with sorted values from left to right. There are n values. Two arbitrary numbers are picked from 1 to n, denoting the start and end index of the search range, respectively. Given a value x, we want to use a binary search algorithm to see if this value is contained in the specified range. If x is found in this range, then the program should return the index of where it is in the array. However, if it is not found, it will return 0. An outline of the algorithm goes like this:

(1) Divide the search range in half and look at $A[m]$ where m is the middle index. If $A[m] = x$, we've found it and the program returns index m.

(2) If not, and $A[m] < x$, search the remaining right part of the array by setting the range from index $(m+1)$ to b. If $A[m] > x$, set the range from a to $(m-1)$.

(3) If the search range is empty, return 0. Otherwise, repeat from step (1).

(a) Write pseudocode (be as detailed as you can) of a recursive function that would follow the steps outlined above. No explicit `for` or `while` loops are allowed.

(b) We want to know if this pseudocode is doing what it's supposed to, returning a single index if x is in the search range or returning 0 if it's not. Try using induction to prove that your pseudocode is correct. Please outline all the steps in the proof.