CS 473 ✧ Spring 2017

Midterm 1

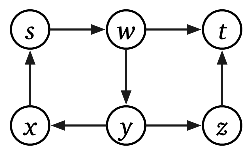
February 21, 2016

---

1. Recall that a ***walk*** in a directed graph G is an arbitrary sequence of vertices $v_0 \to v_1 \to \cdots \to v_k$, such that $v_{i-1} \to v_i$ is an edge in G for every index i. A ***path*** is a walk in which no vertex appears more than once.

   Suppose you are given a directed graph $G = (V, E)$ and two vertices s and t. Describe and analyze an algorithm to determine if there is a walk in G from s to t whose length is a multiple of 3.

   For example, given the graph shown below, with the indicated vertices s and t, your algorithm should return TRUE, because the walk $s \to w \to y \to x \to s \to w \to t$ has length 6.



   [Hint: Build a (different) graph.]

2. A ***shuffle*** of two strings X and Y is formed by interspersing the characters into a new string, keeping the characters of X and Y in the same order. A ***smooth*** shuffle of X and Y is a shuffle of X and Y that never uses more than two consecutive symbols of either string. For example,

   - `prDoYgNAraMmmIiCng` is a smooth shuffle of `DYNAMIC` and `programming`.
   - `DYprNogrAaMmmICing` is a shuffle of `DYNAMIC` and `programming`, but it is not a smooth shuffle (because of the substrings `ogr` and `ing`).

   Describe and analyze an algorithm to decide, given three strings X, Y, and Z, whether Z is a smooth shuffle of X and Y.

3. (a) Describe an algorithm that simulates a fair coin, using independent rolls of a fair three-sided die as your only source of randomness. Your algorithm should return either HEADS or TAILS, each with probability $1/2$.

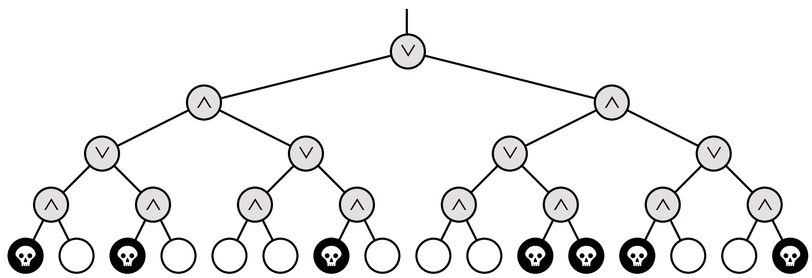
   (b) What is the expected number of die rolls performed by your algorithm in part (a)?

   (c) Describe an algorithm that simulates a fair three-sided die, using independent fair coin flips as your only source of randomness. Your algorithm should return either 1, 2, or 3, each with probability $1/3$.

   (d) What is the expected number of coin flips performed by your algorithm in part (c)?

4. Death knocks on Dirk Gently's door one cold blustery morning and challenges him to a game. Emboldened by his experience with algorithms students, Death presents Dirk with a complete binary tree with 4^n leaves, each colored either black or white. There is a token at the root of the tree. To play the game, Dirk and Death will take turns moving the token from its current node to one of its children. The game will end after $2n$ moves, when the token lands on a leaf. If the final leaf is black, Dirk dies; if it's white, Dirk lives forever. Dirk moves first, so Death gets the last turn.

   (Yes, this is precisely the same game from Homework 3.)

   Unfortunately, Dirk slept through Death's explanation of the rules, so he decides to just play randomly. Whenever it's Dirk's turn, he flips a fair coin and moves left on heads, or right on tails, confident that the Fundamental Interconnectedness of All Things will keep him alive, unless it doesn't. Death plays much more purposefully, of course, always choosing the move that maximizes the probability that Dirk loses the game.

   (a) Describe an algorithm that computes *the probability* that Dirk wins the game against Death.

   (b) Realizing that Dirk is not taking the game seriously, Death gives up in desperation and decides to also play randomly! Describe an algorithm that computes *the probability* that Dirk wins the game again Death, assuming *both* players flip fair coins to decide their moves.

   For both algorithms, the input consists of the integer n (specifying the depth of the tree) and an array of 4^n bits specifying the colors of leaves in left-to-right order.