CS 530 - Fall 2018
Homework 3

Due: Thursday, October 25

Reading : Read the end sections (the last 4 pages) of the posting on the course homepage on probabilistic algorithms.
The url is: http://cs-www.bu.edu/faculty/homer/530f18/rand-alg1.pdf

Problems: All on randomized algorithms

1. Describe a Monte Carlo algorithm which tests, given three polynomials p, q and r in one variable x, whether p(x)q(x) = r(x). Your algorithm should,

(i). have one-sided error, (Explain why it does.)

(ii). have error probability less than or equal to 1/2, (Explain why it does.)

(iii). have running time O(n) where n is the degree of r. (Again, explain why.)

(Hints: You may use the facts that a polynomial of degree n can be evaluated at a point in O(n) steps and that a polynomial of degree n has at most n distinct roots.)

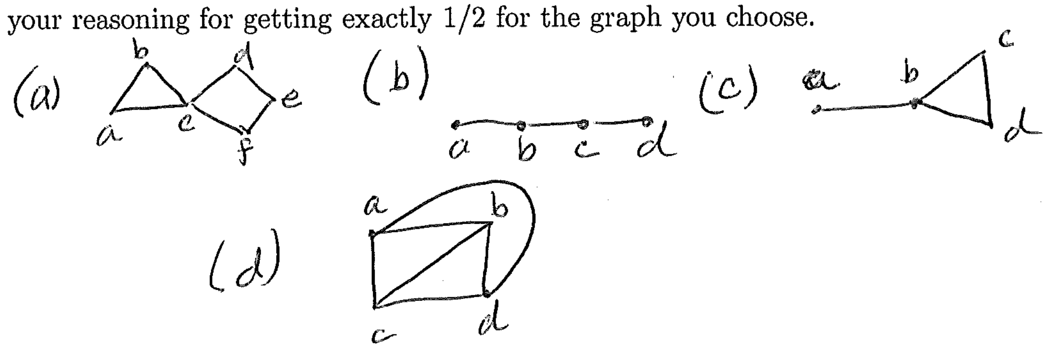
(iv). Show how your algorithm might work given the input polynomials p(x) = $x^2 - 3x + 1$, q(x) = $x + 2$ and r(x) = $x^3 - x^2 - 3x + 2$.

(v). How many random tests would need to be done so that you could ensure (with probability = 1) that the algorithm has output the correct answer to whether p(x)q(x) = r(x)?

2. Assume you have a biased coin which comes up heads with probability p and tails with probability 1-p. Unfortunately you do not know the value p. (i). Design a simple process by which you can use this coin to generate a perfectly unbiased sequence of random bits. Explain briefly why this works.

(ii). Assume that p=1/4. What is the expected number of times you would need to flip your biased coin in order to generate a sequence of 3 random bits. Explain .

3. Try running the contraction algorithm on each of the four graphs below to identify a mincut in the graph.
For which of the 4 graphs is the probability of finding a min cut of the graph exactly 1/2. Explain your reasoning for getting exactly 1/2 for the graph you choose.



4. Let's slightly change the randomized min-cut algorithm discussed in class as follows: Instead of choosing an edge to contract, we randomly choose 2 vertices (which may or may not be connected by edges) and contract them into 1 vertex at each contraction step. .

Show that there are input graphs for which this new min-cut algorithm finds a min-cut with exponentially small probability. That is, describe a specific graph with n vertices for which the contraction algorithm is likely to run for exponentially many iterations (in n) before it finds the min-cut in the graph. Explain your reasoning.

(Note: Even if you can't fully prove or explain why your exponential bound holds, giving the example of the graphs for which the revised contraction algorithm may often fail and saying something about its weak probability of success, i.e. why it doesn't give the same error bound as the randomized min-cut algorithm discussed in class, will be enough for partial credit.)