CS 630 - Fall 2020
Homework 4

Due: Wednesday, November 25

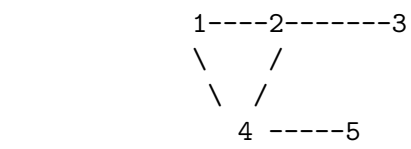
Reading : Read over chapter 34 on NP Completeness and really read Chapter 35 on Approximation Algorithms, Sections 1,2 and 3.

Problems:
1. Consider the graphs G below.

(i). State the size of the min-cut of G, state how many min cuts G has and list them. What is the size of the max cut of G. List all of G's max-cuts.

(ii). Compute the exact probability that the algorithm will find a min cut of graph G after running the contraction algorithm once. Show your work in doing this computation.

Graph G:



2. (i).Prove that the maximum number of min-cuts that a graph
of n vertices can have is O(n(n-1)/2).

(ii). Give an example of a graph of at least k > 5 vertices which has at least k(k-1)/2 min-cuts. (Note: If you find this problem hard for k > 5, try finding an example for k=5. You will get some partial credit for a graph with 5 vertices.)
You should state what the min-cuts are in your graph; either list them or draw a picture.

3. Assume you have a list of N, N > 10,000, numbers from 1 to 100 and you know that more that .7 of the N numbers are the same (say T).

(i). Write a Monte Carlo algorithm which determines the number T with high probability. Specifically your algorithm should,

-run for some constant number of steps not depending on N, and

-output the value T with probability at least .8

You should state the time complexity and error bounds of your algorithm and explain why your algorithm achieves these time and error bounds.

(ii). Is it possible to find an efficient Las Vegas algorithm to solve this problem? That is, is there an algorithm which never makes an error and which finds the majority element in expected time O(log N). Why or why not ? Explain your reasoning.

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 any edges) and contract them into 1 vertex by contracting them into a single vertex.

Show that there are input graphs for which this new min-cut algorithm finds a min-cut with only 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 small 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.)