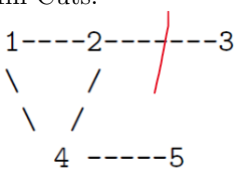


Please limit your answer to the following problems to at most 1/2 a page each.

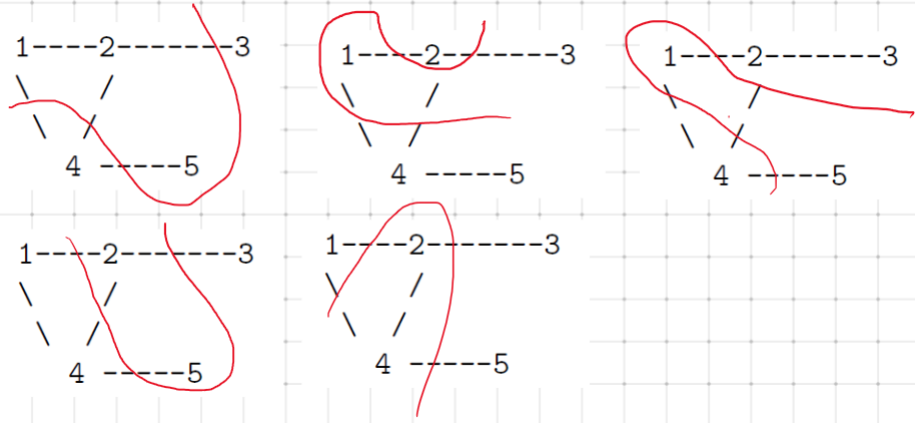
**Problem 1.** Consider the graphs  $G$  given.

- i) State the size of the min-cut  $G$ , 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.

Min-Cuts:



Max-Cuts:



The min-cut size is 1, and there are 2 min-cuts, the max-cut size is 4.

- 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.

Probability of last contraction being 2-3 or 4-5 is  $4! \times 2/5! = \frac{2}{5}$

**Problem 2.**

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

The probability of the Randomize Min-Cut algorithm of finding a min-cut is  $2/(n(n-1))$ .

Proof:

If a graph has a min-cut of size  $k$ , then each vertex must have a minimum degree of  $k$ . Therefore the amount of edges that the graph can have is  $|E| \geq \frac{kn}{2}$ .

For a min-cut, the probability of a contraction not being an edge in the min-cut is

$1 - \frac{k}{|E|} \geq 1 - \frac{2}{n}$ . For each additional step  $i$ , the conditional probability of the new edge not being a min-cut

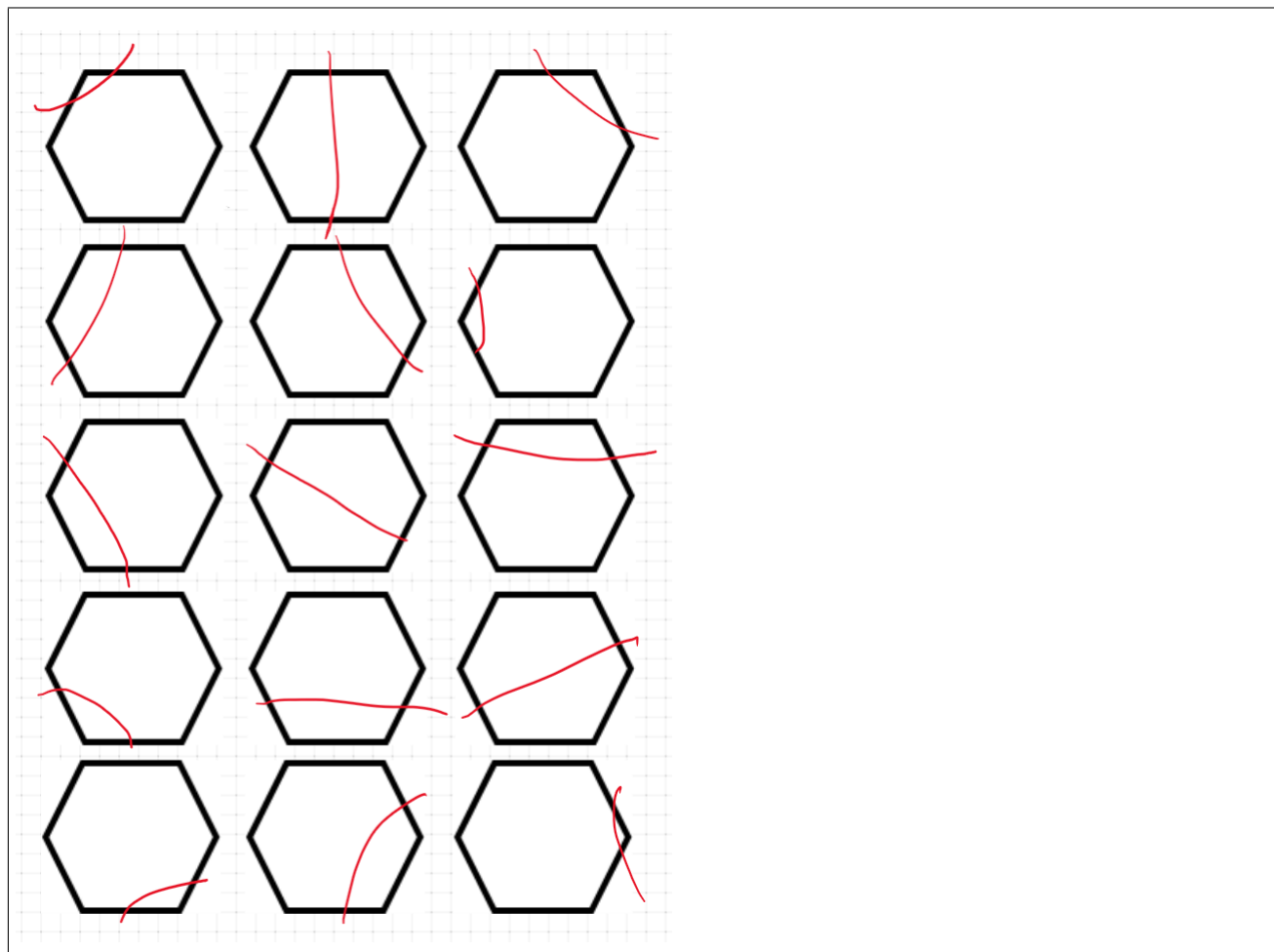
$Pr \geq 1 - \frac{2}{n-i}$

Therefore, the probability of success  $Pr = (1 - \frac{2}{n})(1 - \frac{2}{n-1})(1 - \frac{2}{n-2}) \dots (1 - \frac{2}{3}) \geq \frac{2}{n(n-1)}$

Since for each min-cut, there is a  $\frac{2}{n(n-1)}$  probability of the algorithm succeeding, there must be at most  $1 / \frac{2}{n(n-1)} = \frac{n(n-1)}{2}$  min-cuts, as the sum of the probability is bound to 1.

(Can be summed because each min-cut's probability is independent of one another)

- ii) Give an example of a graph of at least  $k > 5$  vertices which has at least  $k(k-1)/2$  min-cuts.



**Problem 3.** Assume you have a list of  $N$ ,  $N > 10,000$ , numbers from 1 to 100 and you know that more than .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 times and error bounds.

Pick  $k$  random number from the list.  $T$  is the number that appears the most often.

The probability of this being wrong decreases with a higher  $k$ .

The worst case scenario is 0.7 of the list is  $T$ , and 0.3 is another number  $S$ .

The probability of getting it wrong is the poisson distribution equation

$$Pr = \sum_{i=0}^{k/2} \binom{k}{i} 0.7^i 0.3^{k-i}$$

Thus  $Pr > 0.8$  when  $k \geq 8$ .

Thus if  $k = 8$ , our algorithm will have a probability of being correct at least 80% of the time

- 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  $\mathcal{O}(\log N)$ . Why or why not? Explain your reasoning.

There is no Las Vegas algorithm capable that works in  $\mathcal{O}(\log N)$  time.

Any random sampling algorithm will always have a chance of picking numbers that are not the majority. This probability can be reduced by increasing the sampling amount, up to  $0.3N + 1$  samplings with the lowest error rate (0%). However, as the size of the list grows, the sampling requirements also increases to keep it at 0% at  $\mathcal{O}(N)$  time. Thus, there is no algorithm that can never make an error and also complete in  $\mathcal{O}(\log N)$  time.

**Problem 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 edge) 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.

Take a graph  $G$ , where there are two  $n$ -complete graphs  $g_1, g_2$ , and a single edge between one vertex of  $g_1$  and  $g_2$  (lets call them  $i, j$ ). The min-cut of this graph must be the cut on the edge between  $i$  and  $j$

The algorithm will error out whenever it contracts vertices where one belongs in  $g_1$  and one belongs in  $g_2$ .

As there are a total of  $2n$  vertices, there will need to be  $2n-2$  contractions. ( $n-1$  for  $g_1, g_2$  each.)

The chance of picking correct is therefore  $Pr = \frac{(n-1)!(n-1)!}{2n!/2} = \frac{2(n-1)(n-2)(n-3)\dots(n-n+1)}{(2n)(2n-1)(2n-2)\dots(2n-n+1)(2n-n)} \approx \frac{1}{n^2 \times 2^n}$ ,

Thus, we have shown that there is at least one type of graph in which the algorithm has an exponentially small probability of getting the correct min-cut.

This is different to the original algorithm because there is only the single edge between  $i$  and  $j$  that will cause an error if contracted.