CS/MATH 335: PROBABILITY, COMPUTING, AND GRAPH THEORY
HOMEWORK 5
DUE IN CLASS MONDAY, OCTOBER 17, 2016

(Taylor Heilman) and Matt Iammarino

Partner Problems

Roughly in order of difficulty

(1) Mitzenmacher 1.25

a.) Running the algorithm twice the number edge contractions will be twice the number of running the algorithm once, hence we get $2(n-2)$ edge contractions.

The probability of not finding a min cut in one run is $1 - \frac{2}{n(n-1)}$, so the probability of not getting a min cut when the algorithm is run twice is simply $(1 - \frac{2}{n(n-1))})^2$. Given the fact that $1 - x \leq e^{-x}$ then $(1 - \frac{2}{n(n-1))})^2 \leq (e^{-\frac{2}{n(n-1))}})^2$

Therefore Pr(getting min cut)$\geq (e^{-\frac{2}{n(n-1))}})^2$

b.) The number of edge contractions when running the algorithm until there are k vertices is equal to $(n-k)$. Running the algorithm on the k vertices l different times will have $l(k-2)$ edge contractions. Hence, the total number of edge contractions is $(n-k) + l(k-2)$

The probability that the algorithm produces a min-cut when we go from n to k vertices is $\sum_{i=1}^{n-k}(\frac{n-i-1}{n-i+1}) = \frac{k(k-1)}{n(n-1)}$. Then the probability that at least one of the l trials produces a min-cut is 1− the probability that none produce a min-cut, which is $(1 - \frac{2}{k(k-1)})^l$. So the probability that at least one of the l trials produces a min-cut is $1 - (1 - \frac{2}{k(k-1)})^l$. Thus, the probability that both of these things happen is..
Pr(Finding a min cut) $\geq (1 - (1 - \frac{2}{k(k-1)})^l)(\frac{k(k-1)}{n(n-1)})$.

c. In order to find the optimal values of k and l to produce the best probability, we first had to understand our constraints. We needed the number of edge contractions to be equal to the number of edge contractions by only running the algorithm twice. So we have that..
$(n-k) + l(k-2) = 2(n-2)$

We struggled to optimize this result to get the best k, l, and n values for the probability of.. Pr = $(1 - (1 - \frac{2}{k(k-1)})^l)(\frac{k(k-1)}{n(n-1)})$.

We tried running a Python program in order to optimize this result based upon the constraints by testing multiple k and l values, which would then decide the n value. Our problem was that we realized in order to maximize the probability, we wanted the l to be large because that would cause the biggest number in the first part of the probability, but then k needed to be smaller in the first part of the probability product and bigger in the second part. The problem that we found was that our n value kept increasing with increasing k and l values so the smallest k and l values always outputted the highest probability. Here is an example of our code..

Our Code for the Python program is attached.

From this, we got our optimal answers to be ..
Prob = 0.3518518518518518
$l = 3$

$k = 3$

$n = 4$

Prob = 0.6632956796462868

$l = 2.5$

$k = 2.5$

$n = 2.75$

(2) Mitzenmacher 5.7

a.) The conditional probability that bin 1 has one ball given that exactly one ball fell into the first three bins is easily computed. Since there are 3 possible bins that 1 ball can land in, each bin's individual probability of receiving the single ball is $\frac{1}{3}$.

b.) Notice that when there are n balls and n bins the expected number of balls to land in a bin is equal to $\frac{n}{n} = 1$. Given that bin 2 receives 0 balls we can think of the problem as throwing n balls into $n - 1$ bins, hence the expected number of balls in bin 1 is equal to $\frac{n}{n-1}$.

c.) When thinking of how to express the probability that bin 1 has more balls than bin 2 we thought of every situation where bin 1 has more balls than bin 2. The first possibility is bin 2 has 0 balls and bin 1 has anywhere from 1 to n balls.The next possibility is bin 2 having 1 ball and bin 1

having anywhere from 2 to $n-1$ balls and so on. By summing up the probabilities of each case we can find the total probability. We can write out the expression of the probability using summations. Letting X be a random variable representing the number of balls in bin 1 and Y represent the number of balls in bin 2 we get:

$$Pr(X > Y) = \sum_{i=0}^{\lfloor \frac{n-1}{2} \rfloor} \sum_{j=i+1}^{n-i} Pr(X = j \cap Y = i)$$

The outer summation represents the balls in bin 2. It only goes up to the floor of $\frac{n-1}{2}$ because in order for bin 1 to have more balls than bin 2, bin 2 has to have less than half of all the balls, otherwise bin 1 cannot have more balls in it than bin 2. The inner summation represents the balls in bin 1 and start at $i+1$ because the balls in bin 1 need to be at least 1 greater than bin 2. The inner summation only goes up to $n-i$ because i is the number of balls in bin 2, hence $n-i$ is the number of balls left.

(3) Mitzenmacher 5.2

For this Social Security question, we decided to treat it as the birthday problem with just a different number of days of the year. In the case when we wanted to find how many people it would take for it to be more than likely than not that two people have the same last four digits, we thought that the last four digits could be numbers 0000 – 9999. This results in $10,000$ numbers. Using the equation from the book, we have the approximation for the halfway point is $\frac{m^2}{2n} = ln(2)$, or $m = \sqrt{(}2n * (ln(2))$. Using this approximation, $n = 10000$ and we get that $m = \sqrt{(}20000 * (ln(2)) = 117.7$. From this approximation, we plugged values around 118 into the equation, $\frac{(\binom{10000}{x})(x!)}{10000^x}$. We got the final answer to be $x = 119$ and $\frac{(\binom{10000}{119})(119!)}{10000^{1}19} = .494$. This means that there is a .494 probability that no two people have the same last four digits, which implies that it is more than likely that two people have the same last four digits.

We did a very similar approach to an entire SSN. Now the range was 000000000 – 999999999. This is 1000000000 numbers and using our approximation again we get.. $m = \sqrt{(}2(1000000000) * ln(2)) = 37232.9$. Once again trying values around this approximation into the equation, $\frac{(\binom{1000000000}{x})(x!)}{1000000000^x}$, we got that $x = 37234$ would produce $\frac{(\binom{1000000000}{37234})(37234!)}{1000000000^{3}7234} = .4999$. This means that there is a .4999 probability that no two people have the same SSN, which implies that it is more than likely that two people have the same SSN.

Notice that if the SSN was 13 digits long, then the last four digits would not change and we would get the same answer as 9 digits. The probability of having a duplicate number would change however because the range would be 0000000000000 – 9999999999999. This would cause 10000000000000 numbers. Once again, using our approximation we get

that $m = \sqrt{(}20000000000000 * (ln(2))) = 3723297$. Plugging in numbers around this value into the equation, $\frac{((\binom{10000000000000}{x}))(x!)}{10000000000000^x}$, we get that $x = 3723298$ to get the prob $\frac{((\binom{1000000000}{32723298}))(32723298!)}{1000000000^3723298} = .499999$. This means that there is a .499999 probability that no two people have the same SSN of 13 digits, which implies that it is more than likely that two people have the same SSN of 13 digits.

(4) Mitzenmacher 5.3

First, we thought about the probability while throwing each individual ball into the n bins such that no bin has more than 1 ball. On the first throw, we have a probability of 1.0 that we would throw a ball into a bin that doesn't already have a ball. On the second throw, one bin will have a ball in it, and thus the probability of throwing that ball into an empty bin has decreased from $\frac{n}{n}$ to $\frac{n-1}{n}$. This pattern will continue for all of the balls that we throw. Since we have $c_1 \sqrt{(n)}$ balls, the final probability for the last ball will be $\frac{n-(c_1 \sqrt{(n))+1}}{n}$. We then want to take the product of all of these probabilities to find the total probability that no bin has more than one ball. So..

Pr(no two balls into same bin) = $\frac{n}{n} * \frac{n-1}{n} *.... * \frac{n-(c_1 \sqrt{(n))+1}}{n}$

Our final answer for the first part needs to be that ..
$(\frac{n}{n} * \frac{n-1}{n} *.... * \frac{n-(c_1 \sqrt{(n))+1}}{n}) \leq \frac{1}{e}$

For the second part, our final answer needs to be that for n sufficiently large.. $(\frac{n}{n} * \frac{n-1}{n} *.... * \frac{n-(c_2 \sqrt{(n))+1}}{n}) \geq \frac{1}{2}$

We were not able to use the hints to get these inequalities to hold and reduce them into the proper form.

When we couldn't use the hints, we then tried to use the Poisson Approximation approach in the attached scrap work. That did not work either, but we thought we would attach anyway.

Individual Problems

(5) Mitzenmacher 1.24

Notice we find a 3 way cut set by randomly contracting edges until there are 3 vertices left. The remaining edges in the 3 vertex graph is the cut set.

Thus if we want to find an r-way cut set We can randomly contract edges until there are r vertices remaining.

Notice the fact from the book that a minimum cut is produced by the algorithm, such that the algorithm terminates when 2 vertices remain, with probability $\frac{1}{n^2}$. We get this probability by computing $\frac{1}{\binom{n}{2}}$ We can use this information to figure out the probability a min cut is produced when the algorithm terminates when there are r vertices remaining. When finding an r way cut set we will have $n-r$ contractions. Each edge contraction has a probability at most $\frac{r}{n}$ that the randomly chosen edge is in the minimum cut. The probability that we never contract a min cut edge through all $n-k$ contractions is at least $\frac{1}{\binom{n}{k}}$ which simplifies to $\frac{k(k-1)}{n(n-1)}$. Hence, Pr(finding a minimum r-way cut set) $\geq (\frac{k}{n})^2$.

For this problem I used information found in our text book as well as information found in an article "An $O(n^2)$ Algorithm for Minimum Cuts" by David R Karger and Clifford Stein.