Homework 8

Due Saturday, November 7

Since we haven't introduced any new algorithms this week, there will be no implementation part.

Written Part

1. In Hw 7 problem 7c you described an algorithm that recovered the first k bits of the discrete log modulo p, assuming that $p-1$ is divisible by 2^k. Prove the correctness of this algorithm. In particular, there is some ambiguity when you take the square root in last week's algorithm. Why does the assumption that $p-1$ is divisible by 2^k alleviate this ambiguity?

Proof. We will give an exposatory proof outlining the idea of the algorithm at the same time. We will point out that this proof can be significantly streamlined and shortened, but we included many details and potential pitfalls that were encountered in office hour conversations, in order to have a better explanation of the underlying ideas. We begin by writing:

$$\log_g a = \varepsilon_0 + \varepsilon_1 \cdot 2 + \varepsilon_2 \cdot 2^2 + \varepsilon_3 \cdot 2^3 + \cdots \qquad \varepsilon_i \in \{0,1\}.$$

We first compute ε_0 by computing $\left(\frac{a}{p}\right) = a^{\frac{p-1}{2}} \mod p$, noticing that it is 0 if and only if $\log_g a$ is even if and only if the legendre symbol is 1. We then define:

$$a' = \begin{cases} a & \varepsilon_0 = 1 \\ g^{-1}a & \varepsilon_0 = 1. \end{cases}.$$

The first thing we notice is:

$$\log_g a' = \varepsilon_1 \cdot 2 + \varepsilon_2 \cdot 2^2 + \varepsilon_3 \cdot 2^3 + \cdots$$

Indeed, if $\varepsilon_0 = 0$ then $a' = a$ and this is immediate. Otherwise $\varepsilon_0 = 1$ and:

$$\begin{aligned} \log_g a' &= \log_g(g^{-1}a) \\ &= -\log_g g + \log_g a \\ &= -1 + 1 + \varepsilon_1 \cdot 2 + \varepsilon_2 \cdot 2^2 + \varepsilon_3 \cdot 2^3 + \cdots \\ &= \varepsilon_1 \cdot 2 + \varepsilon_2 \cdot 2^2 + \varepsilon_3 \cdot 2^3 + \cdots \end{aligned}$$

as desired. Next we use the given square root computation algorithm to compute $b = \sqrt{a'}$. In particular, we see that

$$2\log_g b = \log_g(b^2) = \log_g a'.$$

The dream, then is that $\log_g b = (\log_g a')/2$. That is:

$$\log_g b = \varepsilon_1 + \varepsilon_2 \cdot 2 + \varepsilon_3 \cdot 2^2 + \cdots \tag{1}$$

If this were true, then we could feed b to the beginning of this algorithm to find ε_1 and continue in this fashion. But, **Equation 1 is FALSE** (this is why it is in red). Indeed, the value of the discrete log is not in $\mathbb{Z}$, but in $\mathbb{Z}/(p-1)\mathbb{Z}$, and since $p-1$ is even by assumption, we **do not have division by 2**. (We point out that is should come as somewhat of a relief, because otherwise we would have described an algorithm to solve the discrete log in logarithmic time, and that wouldn't be great for our internet security).

Instead, let $\ell = \log_g a'$. We do know that $\log_g b$ is a solution to the congruence:

$$2x \equiv \ell \mod p-1.$$

Indeed, we studied solutions to these sorts of congruences in HW2 Problem 7, and we showed that if there is one solution then there are $\gcd(2, p-1) = 2$. Only one of these is the naive 'division by 2,' and we have no way of knowing whether it is $\log_g b$. Nevertheless, we also showed in HW2 Problem 7 that given one solution $\log_g b$, the other solution is

$$\log_g b + \frac{p-1}{\gcd(2, p-1)} = \log_g b + \frac{p-1}{2} = x_0 + 2^{k-1}m,$$

where in the last equality we use that $p - 1 = 2^k m$. In particular, we see that,

$$\log_g b \equiv \varepsilon_1 + \varepsilon_2 \cdot 2 + \cdots \mod 2^{k-1}.$$

But the reduction of an element $\mod 2^t$ precisely captures the first t digits of its binary expansion. In particular, we see that:

$$\log_g b = \varepsilon_1 + \varepsilon_2 \cdot 2 + \cdots + \varepsilon_{k-1} 2^{k-2} + \tau_k \cdot 2^{k-1} + \tau_{k+1} \cdot 2^k + \cdots,$$

for some τ_j not necessarily equal to the ε_j. In particular, if we continue the algorithm with b, we will indeed be correctly computing ε_i for $i = 1, 2, \cdots, k-1$.

For the general case, we let a_i be what we have on our i'th step of the loop (indexed so that $a = a_0$). Then by induction we have:

$$\log_g a_i = \varepsilon_i + \varepsilon_{i+1} \cdot 2 + \cdots + \varepsilon_{k-1} \cdot 2^{k-1+i} + \tau_k \cdot 2^{k+i} + \cdots,$$

where the τ_j are not necessarily equal to the ε_j. We can recover ε_i using the Legendre symbol and then let

$$a_i' = \begin{cases} a_i & \varepsilon_i = 0 \\ g^{-1} a_i & \varepsilon_i = 1 \end{cases}$$

and let a_{i+1} be a square root of a_i. Then arguing as above shows:

$$\begin{aligned} \log_g a_{i+1} &= \varepsilon_{i+1} + \varepsilon_{i+1} \cdot 2 + \cdots + \varepsilon_{k-1} \cdot 2^{k-2+i} + \tau_k \cdot 2^{k-1+i} + \cdots \\ &+ \tau_{k+i-1} \cdot 2^{k-2} + \gamma_{k+i} \cdot 2^{k-1} + \gamma_{k+i+1} 2^k + \cdots \end{aligned}$$

where τ_k through τ_{k+i-1} don't necessarily change but τ_{k+i} and onwards will. Of course, this doesn't matter, because we only need the ε_i not to change for $i < k$ for the induction to go through. The point here is that after the first step we have more precision than necessary (although that precision was already lost in the first step and therefore can't be recovered). □

A few conversations from office hours suggested an alternative proof mechanism which we fully flesh out here. It has the advantage of not needing to take the (slightly ambiguous) square root, while also highlighting more clearly where the 2-divisibility of $p - 1$ plays an essential role (specifically in step (2)(i)). Here is a sketch of the algorithm.

```
(1) Set $a_0 = a$.
(2) Loop through $i = 0, \cdots, k-1$.
      (i) Compute $\ell_i \equiv a_i^{\frac{p-1}{2^{i+1}}} \mod p$.
      (ii) If $\ell_i = 1$
            • Set $\varepsilon_i = 0$.
            • Set $a_{i+1} = a_i$
      (iii) If $\ell_i = -1$.
            • Set $\varepsilon_i = 1$.
            • Set $a_{i+1} = g^{-2^i} a_i$
```

We first point out that the exponent $\frac{p-1}{2^{i+1}}$ in step (2)(i) is an integer. Indeed, we are assuming that $2^k | p-1$, so the exponent is an integer precisely when $i < k$, but that is precisely what we are looping through. The correctness of the algorithm now follows from the following rather straightforward observations.

Lemma 1. *Suppose $2^{i+1} | p-1$ and $b^{2^i} = c$. Then:*

$$\left(\frac{b}{p}\right) = c^{\frac{p-1}{2^{i+1}}}.$$

Proof. We compute directly, importantly using that fact that $\frac{p-1}{2^{i+1}}$ is an integer:

$$\left(\frac{b}{p}\right) = b^{\frac{p-1}{2}} = (b^{2^i})^{\frac{p-1}{2^{i+1}}} = c^{\frac{p-1}{2^{i+1}}}.$$

□

We point out that Lemma 1 also solves our ambiguity problem from the first algorithm. That is, with some bookkeeping it shows that the Legendre symbol computation step is insensitive to the choices of roots we take. Lemma 1 has the following reformulation.

Lemma 2. *Let $2^{i+1} | p-1$ and $c \in \mathbb{F}_{p^*}$ have the following binary expansion of its log:*

$$\log_g c = \varepsilon_i 2^i + \varepsilon_{i+1} 2^{i+1} + \cdots .$$

Then $\ell = c^{\frac{p-1}{2^{i+1}}} \in \{\pm 1\}$. Furthermore, $\ell = 1$ if $\varepsilon_i = 0$, and $\ell = -1$ otherwise.

Proof. We construct a 2^ith root of c:

$$b = g^{\varepsilon_i + \varepsilon_{i+1} \cdot 2 + \cdots}.$$

Then since $\log_g b^{2^i} = 2^i \log_g b = \log_g c$ we see that $b^{2^i} = c$ (because the discrete log is bijective). Therefore by Lemma 1 we see that:

$$c^{\frac{p-1}{2^{i+1}}} = \left(\frac{b}{p}\right) \in \{\pm 1\},$$

proving the first part of the claim. The second follows immediately since the legendre symbol $\left(\frac{b}{p}\right)$ precisely computes the parity of the log of b. □

Lemma 3. *For each i, we have:*

$$\log_g a_i = \varepsilon_i 2^i + \varepsilon_{i+1} 2^{i+1} + \cdots .$$

Proof. We proceed by induction on i. For $i = 0$ this is immediate. In general, we may assume that:

$$\log_g a_i = \varepsilon_i 2^i + \varepsilon_{i+1} 2^{i+1} + \varepsilon_{i+1} 2^{i+1} + \cdots ,$$

and show that the result holds for a_{i+1}. By Lemma 2, $\varepsilon_i = 0$ precisely when $a_i^{\frac{p-1}{2^{i+1}}} = 1$, in which case $a_{i+1} = a_i$ and we are done. Otherwise $a_{i+1} = g^{-2^i} a_i$ and $\varepsilon_i = 1$, and we may directly compute:

$$\begin{aligned} \log_g a_{i+1} &= \log_g g^{-2^i} a_i \\ &= \log_g g^{-2^i} + \log_g a_i \\ &= -2^i + 2^i + \varepsilon_{i+1} 2^{i+1} + \varepsilon_{i+2} 2^{i+2} + \cdots \\ &= \varepsilon_{i+1} 2^{i+1} + \varepsilon_{i+2} 2^{i+2} + \cdots \end{aligned}$$

completing the proof. □

By Lemma 2 and Lemma 3 together immediately imply that our computations of ε_i in steps (2)(ii) and/or (2)(iii) of our algorihm are correct, proving the correctness of our algorithm.

2. We've given several proofs of Fermat's Little Theorem. This exercise outlines another one that is of a very different flavor. Throughout we fix a prime number p.

   (a) Let j be an integer with $1 \leq j \leq p - 1$. Prove that $\binom{p}{j}$ is divisible by p.

   Proof. By definition:

   $$\binom{p}{j = \frac{p!}{j!(p-j)!}.}$$

   If $0 < j < p$ then both $j, p - j < p$, so that the p in the numerator cannot be cancelled (since it is prime it is not divisible by any smaller numbers.). □

   (b) For any integers a, b, show that:

   $$(a + b)^p \equiv a^p + b^p \mod p.$$

   (This identity is often called the *freshman's dream* by jaded calculus professors).

   Proof. This follows immediately from the binomial theorem and part (a). In particular, the binomial theorem says the coefficient of $a^j b^{p-j}$ in the expansion is $\binom{p}{j}$. If $j \neq 0, p$ then this is divisible by p (by part (a)), so it becomes 0 once we mod out by p. Therefore the only remaining terms are a^p and b^p. □

   (c) Prove Fermat's Little Theorem: $a^p \equiv a \mod p$ by induction on a using part (b) with $b = 1$.

Proof. For $a = 0, 1$ the result is trivial. In general, we may assume $a^p \equiv a \mod p$ and prove the result for $a + 1$. By part (b) we have:

$$(a+1)^p \equiv a^p + 1^p \equiv a + 1 \mod p,$$

completing the proof. □

3. Suppose we flip a coin 10 times. Compute the probability of the following event.

(a) The probability that the first and last coins are both heads.

Proof. Let C_i be the event where the ith coin flip is heads. Then certainly $Pr(C_i) = \frac{1}{2}$. If $i \neq j$ then C_i and C_j are independant. Therefore:

$$Pr(C_1 \cap C_{10}) = Pr(C_1)Pr(C_{10}) = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}.$$

□

(b) The probability that at least one of the first and last coins is heads.

Proof. We are using 4(b) here.

$$Pr(C_1 \cup C_{10}) = Pr(C_1) + Pr(C_{10}) - Pr(C_1 \cap C_{10}) = \frac{1}{2} + \frac{1}{2} - \frac{1}{4} = \frac{3}{4}$$

□

(c) The probability that exactly 5 coin tosses are heads.

Proof. The probability that any one given outcome (of 10 flips) occurs is $\frac{1}{2^10}$. To have 5 heads, we must choose exactly 5 of the flips to be heads, which we can do in $\binom{10}{5}$ ways. Therefore the probability of 5 heads is $\binom{10}{5} \cdot \frac{1}{2^{10}}$. This is approximately .2461. □

(d) The probability that exactly k coin tosses are heads.

Proof. Arguing exactly as in part (c) we get $\binom{10}{k}\frac{1}{2^{10}}$. □

(e) The probability that an even number of coin tosses are heads.

Proof. Let E_k be the event where we flip k heads. These are all disjoint for distinct k. We use 4(e) below to compute

$$\begin{aligned} Pr(E_0 \cup E_2 \cup E_4 \cup E_6 \cup E_8 \cup E_{10}) &= \sum_{i=0}^{5} Pr(E_{2i}) \\ &= \frac{1}{2^{10}} \sum_{i=0}^{5} \binom{10}{2i}. \end{aligned}$$

Since 10 is rather small one can compute this directly to be $\frac{1}{2}$. Nevertheless, something deeper is going on here. Indeed $\sum_{i=0}^{5} \binom{10}{2i} = 2^9$ is a direct consequence part (3) of the following theorem.

Theorem 1. *The following identities hold for any n.*

(1) $\sum_{k=0}^{n} \binom{n}{k} = 2^n$

(2) $\sum_{k=0}^{n} (-1)^k \binom{n}{k} = 0$

(3) $\sum_{k\geq 0} \binom{n}{2k} = 2^{n-1}$

(4) $\sum_{k\geq 0} \binom{n}{2k+1} = 2^{n-1}$

We will put this on HW 10 perhaps. □

(f) The probability that an odd number of coin tosses are heads.

Proof. The event of odd numbers of heads is the complement of the event of even number of heads, so that by 4(a), the value is also $\frac{1}{2}$. □

4. We let $Pr : \Omega \to \mathbb{R}$ be a probability theory.

(a) Let E be an event, and E^c its complement. Prove $Pr(E^c) = 1 - Pr(E)$.

Proof. Since $E \cup E^c = \Omega$ are disjoint, $Pr(E \cup E^c) = 1$. But by part (b) the left hand side is $Pr(E) + Pr(E^c)$ giving the result. □

(b) Let E and F be disjoint events. Prove that

$$Pr(E \cup F) = Pr(E) + Pr(F).$$

Proof. Notice that if $\omega \in E \cup F$ then either $\omega \in E$ or $\omega \in F$ but not both. This implies that:

$$\sum_{\omega \in E \cup F} Pr(\omega) = \sum_{\omega \in E} Pr(\omega) + \sum_{\omega \in F} Pr(\omega),$$

giving the result. □

(c) Let E and F be any two events (not necessarily disjoint). Prove that

$$Pr(E \cup F) = Pr(E) + Pr(F) - Pr(E \cap F).$$

Proof. The equation in part (b) does not hold in this case, in particular because if $\omega \in E \cap F$, then we have $Pr(\omega)$ added twice on the right hand side, once for ω's appearance in E, and once more of its appearance in F. Therefore we must subtract off the one extra appearance, giving that

$$\sum_{\omega \in E \cup F} Pr(\omega) = \sum_{\omega \in E} Pr(\omega) + \sum_{\omega \in F} Pr(\omega) - \sum_{\omega \in E \cap F} Pr(\omega),$$

as desired. □

(d) Let E_1, E_2, and E_3 be events. Prove that:

$$\begin{aligned} Pr(E_1 \cup E_2 \cup E_3) = \quad & Pr(E_1) + Pr(E_2) + Pr(E_3) - Pr(E_1 \cap E_2) - Pr(E_1 \cap E_3) \\ & -Pr(E_2 \cap E_3) + Pr(E_1 \cap E_2 \cap E_3). \end{aligned}$$

Proof. We will make use of the following lemma.

Lemma 4. *Intersection and union follow a distributive law. That is, if T, S_1, S_2 are sets, then:*

$$T \cap (S_1 \cup S_2) = (T \cap S_1) \cup (T \cap S_2).$$

Similarly, if $T, S_1, \cdots, S_n$ are sets, then:

$$T \cap (S_1 \cup \cdots \cup S_n) = (T \cap S_1) \cup \cdots \cup (T \cap S_n).$$

Proof. If an element is on the left side of the equation, it is contained in T and (at least one of) S_1 or S_2, so that it must be in (at least one of $T \cap S_1$ or $T \cap S_2$ proving the left is a subset of the right. Conversely, if an element is on the right, it is either in $T \cap S_1$ or $T \cap S_2$, so in particular it is in T and one of the S_1 or S_2. This proves the right is a subset of the left.

The general case follows by induction on n, reducing the the case of 2 sets by considering the sets $(S_1 \cdots, S_{n-1})$ and S_n. Indeed:

$$\begin{aligned} T \cap ((S_1 \cup \cdots \cup S_{n-1}) \cup S_n) &= (T \cap (S_1 \cup \cdots \cup S_{n-1})) \cup (T \cap S_n) \\ &= (T \cap S_1) \cup \cdots \cup (T \cap S_{n-1}) \cup (T \cap S_n), \end{aligned}$$

where the first equality is the case of 2 sets and the second is the inductive hypothesis. □

With this in hand we can apply part (c) to prove:

$$\begin{aligned} Pr(E_1 \cup E_2 \cup E_3) &= Pr(E_1 \cup (E_2 \cup E_3)) \\ &= Pr(E_1) + Pr(E_2 \cup E_3) - Pr(E_1 \cap (E_2 \cup E_3)) \\ &= Pr(E_1) + Pr(E_2) + Pr(E_3) - Pr(E_2 \cap E_3) - Pr(E_1 \cap (E_2 \cup E_3)). \end{aligned}$$

We use the distributive law of Lemma 4 and part (c) again to compute the last term:

$$\begin{aligned} Pr(E_1 \cap (E_2 \cup E_3)) &= Pr((E_1 \cap E_2) \cup (E_1 \cap E_3)) \\ &= Pr(E_1 \cap E_2) + Pr(E_1 \cap E_3) - Pr(E_1 \cap E_2 \cap E_3). \end{aligned}$$

This proves the result. □

(e) Let $E_1, E_2, \cdots, E_n$ be n events. We say that the events are *pariwise disjoint* if $E_i \cap E_j = \emptyset$ for all $i \neq j$. Show that if the events are pairwise disjoint then:

$$Pr(E_1 \cup E_2 \cup \cdots \cup E_n) = Pr(E_1) + Pr(E_2) + \cdots + Pr(E_n).$$

Proof. This follows via induction on n reducing to the two events $(E_1 \cup \cdots \cup E_{n-1})$ and E_n and part (b). □

(f) Let $E_1, \cdots, E_n$ be n (not necessarily disjoint) events. Conjecture a general formula for $Pr(E_1 \cup E_2 \cup \cdots \cup E_n)$ in terms of the probability of the E_i and their various intersections. This is called the *inclusion-exclusion* principle.

Proof. The formula is:

$$Pr\left(\bigcup_{i=1}^{n} E_i\right) = \sum_{i=1}^{n} (-1)^{i-1} \left(\sum_{1 \leq j_1 < j_2 < \cdots < j_i \leq n} Pr\left(\bigcap_{k=1}^{i} E_{j_k}\right)\right).$$

It follows by induction on n for the two events $E_1 \cup \cdots E_{n-1}$ and E_n, aruing exactly as in part (d) and using the general case in Lemma 4. □

5. Let E, F be events.

(a) Show that $Pr(E|E) = 1$. Explain in words why this is reasonable.

Proof. $Pr(E|E) = \frac{Pr(E \cap E)}{Pr(E)}$. Since $E \cap E = E$, this is 1. This is reasonable, because E is certain to occur under the assumption that E occurs. □

(b) Suppose that E and F are disjoint. Show that $Pr(E|F) = 0$. Explain in words why this is reasonable.

Proof. $Pr(E|F) = \frac{Pr(E \cap F)}{Pr(F)} = \frac{Pr(\emptyset)}{Pr(F)} = 0$. This makes sense because there are no outcomes in both E and F, so once we know an outcome is in F, the probability it is in E is 0. □

(c) Let $F_1, \cdots, F_n$ be pairwise disjoint and suppose $F_1 \cup \cdots \cup F_n = \Omega$. Prove the following decomposition formula:

$$Pr(E) = \sum_{i=1}^{n} Pr(E|F_i)Pr(F_i).$$

Proof. We use the general case of Lemma 4 to see that:

$$E = E \cap \Omega = E \cap (F_1 \cup \cdots \cup F_n) = (E \cap F_1) \cup \cdot \cup (E \cap F_n).$$

Furthermore, the $E_\cap F_i$ are pariwise disjoint because the F_i are. Therefore:

$$\begin{aligned} Pr(E) &= Pr((E \cap F_1) \cup \cdots \cup (E \cap F_n)) \\ &= \sum_{i=1}^{n} Pr(E \cap F_i) \\ &= \sum_{i=1}^{n} Pr(E|F_i)Pr(F_i) \end{aligned}$$

where the second to last inequality is 4(e) and the last one is the definition of conditional probability. □

(d) Prove the following general version of Bayes' formula:

$$Pr(F_i|E) = \frac{Pr(E|F_i)Pr(F_i)}{\sum_{j=1}^{n} Pr(E|F_j)Pr(F_j)}.$$

Proof. The vertion of Bayes' formula for 2 events says

$$Pr(E)Pr(F_i|E) = Pr(E|F_i)Pr(F_i).$$

Solving for $Pr(F_i|E)$ and plugging in the value of $Pr(E)$ from part (e) gives the desired result. □

6. This is the famous *Monty Hall Problem*. Ralph is on a game show, and Monty Hall gives Ralph the choice of a prize, behind one of 3 closed doors. Monty tell's Ralph that behind 2 of the doors are goats, and behind the third is a new car. Ralph chooses a door, and then Monty opens one of the remaining 2 doors revealing a goat! Monty then asks Ralph: *would you rather stick to the door you chose? Or switch to the other closed door?*

(a) If Ralph always sticks with the same closed door, what are his chances of winning a car? What about if Ralph always switches? What is Ralph's best strategy?

Proof. The important observation to make is that Monty revealing a goat behind one of the remaining doors gives you no information about Ralph's initial choice, since we knew Monty was going to reveal a goat whether Ralph picked a car or a goat on the first step. Since a goat has been revealed, there is one other door. It contains either a car or a goat. In particular, it contains a car if and only if Ralph picked a goat on the first step. In particular,

$$Pr(\text{Ralph wins a car if he switches}) = Pr(\text{Ralph picked a goat with first guess}) = 2/3.$$

Similarly,

$$Pr(\text{Ralph wins a car if he stays}) = Pr(\text{Ralph picked a car with the first guess}) = 1/3.$$

What the better strategy is depends on whether you'd like a goat or a car I suppose. If you're going for a car, switch, if you'd prefer a goat, stay. □

(b) More generally, suppose that there are N doors, M cars, and Monty hall reveals K goats after Ralphs first choice. Compute the probabilities:

$$Pr(\text{Ralph wins a car} \mid \text{Ralph sticks}),$$

$$Pr(\text{Ralph wins a car} \mid \text{Ralph switches}).$$

Which is the better strategy? (Letting $N = 1000, M = 1, K = 998$ makes the solution to part (a) seem less paradoxical).

Proof. We first take care of the boundary cases where $K = 0$ (so Ralph always wins a car), or $M = 0$ (so Ralph always wins a goat). The other boundary case where $K = N - M$ in untenable since if you pick a goat first Monty cannot reveal K more of them. Now we may suppose $K, M > 0$ and $K + M < N$. In this situation we observe that Monty revealing K goats gives us no information about the first door Ralph picked. Therefore:

$$Pr(\text{Ralph wins a car} \mid \text{Ralph sticks}) = Pr(\text{Ralph picked a car on first guess}) = M/N.$$

For the second case, the probability that Ralph wins a car after switching depends on his first guess. That is, there are $N - K - 1$ possible doors to switch too, and they contain either M or $M - 1$ cars depending on Ralphs first guess:

$$Pr(\text{Ralph wins a car} \mid \text{Ralph switches and picked a car on the first guess}) = \frac{M-1}{N-K-1},$$

and

$$Pr(\text{Ralph wins a car} \mid \text{Ralph switches and picked a goat on the first guess}) = \frac{M}{N-K-1}.$$

Let's introduce some notation. E = Ralph wins a car. F = Ralph picked a car on his first guess. G = Ralph switches. Since F and F^c are disjoint, and since F and G are independent, we use 5(c) and the definition of conditional probability to compute:

$$\begin{aligned}
Pr(E|G) &= \frac{Pr(E \cap G)}{Pr(G)} \\
&= \frac{Pr(E \cap G|F)Pr(F)}{Pr(G)} + \frac{Pr(E \cap G|F^c)Pr(F^c)}{Pr(G)} \\
&= \frac{Pr(E \cap G \cap F)PrF}{Pr(F \cap G)} + \frac{Pr(E \cap G \cap F^c)Pr(F^c)}{Pr(F^c \cap G)} \\
&= Pr(E|G \cap F)PrF + Pr(E|G \cap F^c)PrF^c.
\end{aligned}$$

But all these we have computed.

$$\begin{aligned}
Pr(E|G \cap F) &= \frac{M-1}{N-K-1} \\
Pr(E|G \cap F^c) &= \frac{M}{N-K-1} \\
Pr(F) &= \frac{M}{N} \\
Pr(F^c) &= \frac{N-M}{M}
\end{aligned}$$

Therefore:

$$\begin{aligned}
Pr(\text{Ralph wins a car} \mid \text{Ralph switches}) &= \frac{M-1}{N-K-1}\frac{M}{N} + \frac{M}{N-K-1}\frac{N-M}{N} \\
&= \frac{MN-M}{N^2-NK-N}
\end{aligned}$$

□

7. In this exercise we study the probability of success of a Monte Carlo algorithm in quite a bit more generality that we considered in class. Let $\mathcal{S}$ be a set (of integers), and $\mathscr{A}$ an interesting property of elements of $\mathcal{S}$. Suppose that:

$$Pr(x \in \mathcal{S} \text{ is } \textbf{not } \mathscr{A}) = \delta.$$

Suppose that you have a Monte-Carlo algorithm that takes as input a random number r and some $m \in \mathcal{S}$ and returns `Yes` or `No` satisfying:

(1) If the algorithm returns `Yes` m is *definitely* $\mathscr{A}$.

(2) If m has A, then the property that the algorithm returns `Yes` is at least P.

(a) Express conditions (1) and (2) as conditional probabilities

Let $E = m$ is **not** $\mathscr{A}$ and $F_1 =$ The algorithm returns `No`. Then the conditions are

(1) $Pr(E^c|F_1^c) = 1$

(2) $Pr(F_1^c|E^c) \geq P$

(b) Suppose we run the algorithm N times on a fixed $m \in \mathcal{S}$, and the algorithm returns `No` each time. Derive a lower bound in terms of δ, P and N for the probabilit that m is *not* $\mathscr{A}$. (In class we did this for $\delta = .01$ and $P = 1/2$. Here you will have to be more careful about distinguishing P and $1 - P$.)

Proof. Let $F_N =$ the algorithm returns `Yes` N times. We'd like to compute $Pr(E|F_N)$. By Bayes' theorem:

$$Pr(E|F_N) = \frac{Pr(F_N|E)Pr(E)}{Pr(F_N|E)Pr(E) + Pr(F_N|E^c)Pr(E^c)}.$$

We know $Pr(E) = \delta$, and $Pr(E^c) = 1 - \delta$. Furthermore, since $Pr(E^c|F_1^c) = 1$, this implies that $Pr(E^c|F_N^c) = 1$. This implies that if $\omega \in F_N^c$ then $\omega \in E^c$. The contrapositive is $\omega \notin E^c$ implies $\omega \notin F_N^c$, that is $\omega \in E \implies \omega \in F_N$. That is, once an outcome is in E it is guaranteed to be in F_N so that This implies that $Pr(F_N|E) = 1$. It remains to compute $Pr(F_N|E^c)$.

$$Pr(F_N|E^c) = Pr(F_1|E^c)^N = (1 - Pr(F_1^c|E^c))^N = (1 - P)^N.$$

Pluggin this in to Bayes' theorem gives:

$$Pr(E|F_N) \geq \frac{\delta}{\delta + (1 - \delta)(1 - P)^N}.$$

□

8. We can now compute the probability of correctness for `probablyPrime`. Recall that if n is a composite number, then 75% of integers between 2 and $n - 1$ are Miller-Rabin witnesses to the compositeness of n. You will also need the prime number theorem, which we interpret as saying the probability of an integer n being prime is approximately $\ln(n)/n$.

Due to the error in the interpretation of the prime number theorem we delay this discussion until next week.

(a) Suppose `probablyPrime(n)` returns `True`. Compute the probability that n is prime.

(b) Suppose instead of running the Miller-Rabin test on 20 potential witnesses, `probablyPrime` runs the test on N potential witnesses. If `probablyPrime(n)` returns `True`, compute the probability that n is prime in terms of N.