Exercises

Exercises should be completed **on your own.**

1. **(1 pt.)** Suppose that $h : U \to \{0, \ldots, n-1\}$ is a uniformly random function. That is, $h(i)$ is distributed uniformly at random in the set $\{0, \ldots, n-1\}$ for all i, and $h(i)$ are independent for all i. Prove that for any $x \neq y \in U$,

$$\mathbb{P}_h\{h(x) = h(y)\} = \frac{1}{n}.$$

[We are expecting: A short but rigorous proof.]

SOLUTION: One way is to break the sum using the rule

$$\mathbb{P}\{X\} = \sum_{E_i} \mathbb{P}\{X|E_i\}\mathbb{P}\{E_i\},$$

where $E_1, \ldots, E_t$ are events so that with probability one, exactly one of $E_1, \ldots, E_t$ occurs. Using this, we have

$$\begin{aligned}
&\mathbb{P}_h\{h(x) = h(y)\} \\
&= \sum_{i=0}^{n-1} \mathbb{P}_h\{h(x) = h(y)|h(x) = i\}\mathbb{P}_h\{h(x) = i\} \\
&= \sum_{i=0}^{n-1} \mathbb{P}_h\{h(y) = i\}\mathbb{P}_h\{h(x) = i\} \\
&= \sum_{i=0}^{n-1} \frac{1}{n} \cdot \frac{1}{n} \\
&= \frac{1}{n}.
\end{aligned}$$

2. **(2 pt.)** This exercise references the IPython notebook `HW4.ipynb` as well as the files `mysteryA.pickle` and `mysteryB.pickle`.

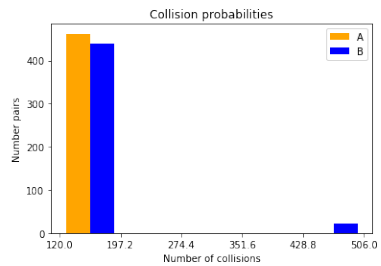
In the IPython notebook, run the cells to load the hash families A and B. Both A and B are lists of functions $h : \{0, \ldots, 21\} \to \{0, 1, 2\}$. As shown in the notebook, at first glance both of these seem like reasonable hash families. **However,** one of them is a universal hash family and one of them is not. Which one is which? Play around with both hash families in Python until you figure it out.

[We are expecting: Your answer, along with compelling numerical evidence (either numbers or a plot), and an explanation about why your numerical evidence is compelling and what it has to do with the definition of a universal hash family.]

SOLUTION: Family A is a good universal hash family. Family B is not. To see this, we compute the collision probabilities for all pairs $x \neq y$. That is, for each x, y, I computed

$$\text{collisionProb(x,y)} = \frac{1}{|H|} \sum_{h \in H} \mathbf{1}\{h(x) = h(y)\}.$$

Then I made a histogram of these collision probabilities. It looked like this:



As we can see, with A all of the collision probabilities are small (actually they are all exactly $154/506 \leq 1/3$), but with B there are some that are really big. (In fact, there are some that have collision probability $506/506 = 1$).

Thus, for A, we have

$$\max_{x \neq y \in U} \mathbb{P}_{h \in A}(h(x) = h(y)) \leq \frac{1}{3},$$

while

$$\max_{x \neq y \in U} \mathbb{P}_{h \in B}(h(x) = h(y)) = 1,$$

so A is the universal hash family and B is not.

Problems

You may talk with your fellow CS161-ers about the problems. However:

- Try the problems on your own *before* collaborating.
- Write up your answers yourself, in your own words. You should never share your typed-up solutions with your collaborators.
- If you collaborated, list the names of the students you collaborated with at the beginning of each problem.

---

1. **(3 pt.)** Your friend has a proposal for a new universal hash function $h : \mathcal{U} \to \{0, \dots, n-1\}$, where $\mathcal{U} = \{0, \dots, n^2 - 1\}$. Your friend thinks that you can skip this whole "choose uniformly from a universal hash family" stuff, and just go with

$$h(x) = x \bmod n.$$

More precisely, your friend says, just take the hash family $\mathcal{H} = \{h\}$ to be the set with just this one function in it.

(a) **(1 pt.)** Your friend doesn't have a very good track record on these homework sets, so you are dubious even before you hear their argument. Prove to your friend that their choice does *not* satisfy the key property of a universal hash family.

[We are expecting: A rigorous proof, using the definition of a universal hash family.]

(b) **(1 pt.)** Even given your proof, your friend plows on. Their first point:

> Let $h = x \bmod n$ be as above. If we choose $x \neq y$ uniformly at random[1] from $\mathcal{U}$, then $\mathbb{P}\{h(x) = h(y)\} \leq \frac{1}{n}$, where the probability is over the random choice of x and y.

Do you agree?

[We are expecting: Whether the statement is true or false, and a convincing argument either way.]

(c) **(1 pt.)** Your friend continues:

> Given the computation above, we have
>
> $$\mathbb{P}\{h(x) = h(y)\} \leq \frac{1}{n}.$$
>
> This is the definition of a universal hash family, so $\{h\}$ must be a universal hash family.

Do you agree?

[We are expecting: Whether this conclusion correctly follows from the statement in part (b), and a convincing argument either way.]

SOLUTION:

(a) The statement can't be true. Consider the choice of $x = 0$ and $y = n$. Then $h(x) = h(y)$ with probability 1, violating the definition of a universal hash family.

(b) This is correct. Informally, this is true for the following reason: if we picked x and y uniformly at random with replacement, then the probability that $x = y \bmod n$ is exactly $1/n$ (by symmetry). If instead we pick them without replacement, the collision probability should only go down, so it is less than $1/n$.
(The above explanation is sufficient for full credit on this problem.)

---

[1] That is, choose x uniformly at random from $\mathcal{U}$ and then choose y uniformly at random from $\mathcal{U} \setminus \{x\}$

More formally, suppose that we choose $x \in \mathcal{U}$ uniformly at random, and then choose $y \in \mathcal{U} \setminus \{x\}$ uniformly at random (that is, uniformly among all the $y \neq x$). Then the probability that $h(x) = h(y)$ can be computed as

$$\begin{aligned}\mathbb{P}\{h(x) = h(y)\} &= \sum_{j=0}^{n-1} \mathbb{P}\{h(x) = h(y) = j\} \\ &= \sum_{j=0}^{n-1} \frac{\text{number of pairs } x \neq y \text{ so that } x = y = j \bmod n}{\text{total number of pairs } x \neq y}.\end{aligned}$$

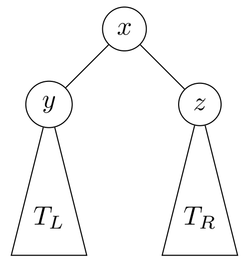
Above, the second equality followed from the fact that if we pick x and y at random, then by the definition of a probability, the probability that $h(x) = h(y) = j$ is the number of (x, y) pairs so that $h(x) = h(y) = j$ (aka, $x = y = j \mod n$), divided by the total number of pairs. Now we just need to count those two quantities to compute the probability.

Consider the number of pairs $x \neq y$ so that $x = y = j \bmod n$. There are n choices for x (because there are n numbers between $0, \ldots, n^2 - 1$ that are equal to any $j \bmod n$), and then $n-1$ choices for y (anything other than x that's equivalent to j). So the numerator in the expression above is $n(n-1)$. On the other hand, the denominator is $n^2(n^2-1)$, since we could choose x to be anything in $\mathcal{U}$ and then y to be anything in $\mathcal{U} \setminus \{x\}$. Thus, this probability is

$$\mathbb{P}\{h(x) = h(y)\} = \sum_{j=0}^{n-1} \frac{n(n-1)}{n^2(n^2-1)} = \frac{n-1}{n^2-1} \leq \frac{1}{n}.$$

(c) This is the problem. That's not the definition of a universal hash family, because the probability should be over the choice of h, not over the choice of x and y.

2. **(5 pt.)** Let T be a Red-Black tree with root x. Let T_L be the subtree rooted at x's left child, and let T_R be the subtree rooted at x's right child.



Decide if each of the following statements are true or false. If it is true, give a proof. If it is false, give a counter-example.

(a) **(2 pt.)** In the set-up above, we must have

$$|T_L| \geq \frac{|T|}{2} - 1 \qquad \text{and} \qquad |T_R| \geq \frac{|T|}{2} - 1, \tag{1}$$

where $|T|$ denotes the number of nodes in T (including the root, not including NIL nodes).

(b) **(3 pt.)** In the set-up above, we must have

$$|T_L| \geq \frac{\sqrt{|T|}}{2} - 1 \qquad \text{and} \qquad |T_R| \geq \frac{\sqrt{|T|}}{2} - 1, \tag{2}$$

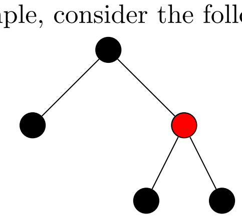
where $|T|$ denotes the number of nodes in T (including the root, not including NIL nodes).

[We are expecting: For each, either a rigorous proof, or an explicit counter-example.]

[HINT: You may use a claim that we proved in class.]

SOLUTION:

(a) The statement is false. For example, consider the following graph:



Then $|T| = 5$ and $|T_L| = 1$, so

$$|T_L| = 1 < \frac{|T|}{2} - 1 = \frac{5}{2} - 1 = 1.5.$$

(b) The statement is true. To prove it, we'll make use of the proof that we saw in class. As in class (and in the lecture notes), let $b(x)$ be the "black-height" of a node x: the number of black nodes on any path from x to NIL, including NIL but not including x. Let $h(x)$ be the height of the node x. (That is, the number of nodes of any color on the longest path from x to NIL, including NIL but not x).
Let x be the root of T, and let y and z be its left and right children (as in the picture). Then we have $|T| \leq 2^{h(x)} - 1$. Moreover, we saw in class that for any node w, the number of nodes in the subtree rooted at w is at least $2^{b(w)} - 1$. Thus,

$$|T_L| \geq 2^{b(y)} - 1 \geq 2^{b(x)-1} - 1 \geq 2^{h(x)/2-1} - 1,$$

using the fact that $b(x) \geq h(x)/2$. So altogether

$$|T_L| \geq 2^{h(x)/2} \cdot \frac{1}{2} - 1 \geq \sqrt{|T|}/2 - 1,$$

as desired. The exact same proof works for T_R.

3. **(6 pt.)** A large flock of T Colorful Geese will migrate south for the winter over the Gates building in the next few weeks. Colorful Geese are an interesting species. They can come in a huge number of colors—say, M colors—but each flock only has m colors represented, where $m < T$. You'd like to be able to answer queries about what colors of geese appeared in the flock. The birds will fly overhead one at a time, and after they have flown by they won't come back again.

   For example, if $T = 7$, $M = 100000$ and $m = 3$, then a flock of T colorful geese might look like:

`seabreeze, seabreeze, brick red, ultraviolet, brick red, ultraviolet, seabreeze`

   You'll see this sequence in order, and only once. After the birds have gone, you'll be asked questions like "How many `brick red` geese were there?" (Answer: 2), or "How many `neon orange` geese were there?" (Answer: 0).

   You have access to a universal hash family $\mathcal{H}$, so that each function $h \in \mathcal{H}$ maps the set of M colors into the set $\{0, \ldots, n-1\}$. For example, one function $h \in \mathcal{H}$ might have $h(\texttt{seabreeze}) = 3$.

   (a) **(3 pt.)** Suppose that $n = 10m$, and you only have space to store n numbers in the set $\{0, \ldots, T\}$, as well as one function h from $\mathcal{H}$. Use the universal hash family $\mathcal{H}$ to create a randomized data structure that fits in this space and that supports the following operations:

   - `Update(color)`: Update the data structure when you see a goose with color `color`.
   - `Query(color)`: Return the number of geese of color `color` that you have seen so far. For each query, your query should be correct with probability at least 9/10. That is, for all colors i,

$$\mathbb{P}\{\texttt{Query}(i) = \text{ the number of geese with color } i\ \} \geq \frac{9}{10}.$$

   You want each of these operations to be done in $O(1)$ time (in the worst case), assuming that you can evaluate a function $h \in \mathcal{H}$ in $O(1)$ time.

   [We are expecting: An explanation of how you will implement your operations, and a short but rigorous proof that your operations meet the requirements.]

   (b) **(3 pt.)** Suppose that you now have ten times the space you had in part (a). Adapt your data structure from part (a) so that the `Query` operation is correct with probability $1 - \frac{1}{10^{10}}$.

   [We are expecting: An explanation of how you will implement your operations, and a short but rigorous proof that your operations meet the requirements.]

   SOLUTION:

   (a) Our data structure will be an array B of length n, where each bucket stores a number between $\{0, \ldots, T\}$, and is initialized to zero. Intuitively, each bucket stores a counter of how many geese were hashed to that bucket. Before the flock flies by, we choose a function $h \in \mathcal{H}$ uniformly at random. We implement the required operations as follows:

   - `Update(color): B[h(color)] ++`
   - `Query(color): Return B[h(color)].`

   Each of these operations takes time $O(1)$. The probability that a single `Query` option fails is the probability that any of the m (or $m-1$ other) colors which did appear collided with the color that was queried. That is, we want

$$\mathbb{P}\{\text{there is a color } x \text{ which appeared, not the same as } \texttt{color}, \text{ so that } h(x) = h(\texttt{color})\}$$

to be small. By the universal hash family property, we have for each color x,

$$\mathbb{P}\{h(x) = h(\texttt{color})\} \le \frac{1}{n}.$$

Thus, by the union bound, the probability that there exists an x which appeared that collides with `color` is at most

$$\mathbb{P}\{\text{there is a color } x \text{ which appeared, not the same as } \texttt{color}\text{, so that } h(x) = h(\texttt{color})\}$$
$$\le m \cdot \mathbb{P}\{h(x) = h(\texttt{color})\} \le \frac{m}{n} = \frac{1}{10}.$$

(b) Instead of keeping a single array B, we will keep 10 arrays $B_0, B_1, \ldots, B_9$, each of size n. We choose 10 hash functions, $h_0, \ldots, h_9$ from $\mathcal{H}$, uniformly and independently. Then our update strategy is:

```
Update(color):
    for i = 0,...,9:
        B_i[ h_i(color) ] ++

Query(color):
    return min_{i = 0,...,9} B_i[ h_i(color) ]
```

To compute the success probability, notice that this returns the correct value as long as the color `color` is isolated in *any* of the 10 tables. Since each of these 10 hash functions are independent, we have:

$$\begin{aligned}
&\mathbb{P}\{\text{for all } i, \text{ there is a color } x \text{ which appeared, not the same as } \texttt{color}\text{, so that } h_i(x) = h_i(\texttt{color})\} \\
&= \left(\mathbb{P}\{\text{there is a color } x \text{ which appeared, not the same as } \texttt{color}\text{, so that } h_i(x) = h_i(\texttt{color})\}\right)^{10} \\
&\le \left(m \cdot \mathbb{P}\{h(x) = h(\texttt{color})\}\right)^{10} \\
&\le \left(\frac{m}{n}\right)^{10} \\
&= \frac{1}{10^{10}}.
\end{aligned}$$