1. **Half-priced hash.** In class, we studied a family of hash functions based on taking inner products. That family was a family of hash functions from $\mathbb{Z}_m^t$ to $\mathbb{Z}_m$ indexed by $\mathbb{Z}_m^t$. For each hash function index $\lambda = (\lambda_1, \ldots, \lambda_t) \in \mathbb{Z}_m^t$, and each key $a = (a_1, \ldots, a_t) \in \mathbb{Z}_m^t$, the hash function was defined as $h_\lambda(a) := \sum_i a_i \lambda_i$, which requires $t$ multiplications to evaluate. In this problem, you are to analyze a variant hash which cuts the number of multiplications in half.

Assume $t$ is even, so $t = 2s$. Hash function indices and keys have the same structure as above, but the hash function is defined as follows:

$$h'_\lambda(a) := \sum_i (a_{2i-1} + \lambda_{2i-1}) (a_{2i} + \lambda_{2i}).$$

So, for example, for $t = 4$, we have

$$h'_\lambda(a) = (a_1 + \lambda_1)(a_2 + \lambda_2) + (a_3 + \lambda_3)(a_4 + \lambda_4).$$

Your task is to show that the family of hash functions $\{h'_\lambda\}_{\lambda \in \mathbb{Z}_m^t}$ is a universal family.

Hint: Your proof should mimic the one given in class for the inner-product based family. Namely, consider two distinct keys $a = (a_1, \ldots, a_t)$ and $b = (b_1, \ldots, b_t)$, and show that the number of hash function indices $(\lambda_1, \ldots, \lambda_t)$ which satisfy

$$\sum_{i=1}^s (a_{2i-1} + \lambda_{2i-1})(a_{2i} + \lambda_{2i}) = \sum_{i=1}^s (b_{2i-1} + \lambda_{2i-1})(b_{2i} + \lambda_{2i}).$$

is at most $m^{t-1}$. To keep the notation simple, you may first want to do the calculation for the case $t = 4$.

2. **Pretty good hash.** Let $h : \mathcal{U} \to \mathcal{V}$ be a hash function, mapping from some (finite) universe $\mathcal{U}$ of keys to a (finite) set of slots $\mathcal{V}$. For a set $Q \subseteq \mathcal{U}$ and an element $a \in Q$, we say that $h$ isolates $a$ in $Q$ if the only element of $Q$ that hashes to the slot $h(a)$ is $a$ itself, i.e.,

$$\text{for all } b \in Q: \quad h(a) = h(b) \implies a = b.$$

Now recall the notion of a perfect hash function. Using the above terminology, we can say that $h$ is a perfect hash function for $Q$ if $h$ isolates every element of $Q$. Consider the following, weaker property: let us say that $h$ is a pretty good hash function for $Q$ if $h$ isolates at least $|Q|/2$ elements of $Q$.

Your task is to design an efficient, probabilistic algorithm that takes as input a set $Q = \{a_1, \ldots, a_n\}$ of $n$ distinct keys, and finds a hash function that is pretty good for $Q$.

To this end, assume that $\{h_\lambda\}_{\lambda \in \Lambda}$ is a universal family of hash functions from $\mathcal{U}$ to $\mathcal{V}$. Assume that $\mathcal{V} = \{0, \ldots, m\}$, where $4n \leq m \leq 8m$. You may assume that you can choose $\lambda \in \Lambda$ uniformly at random in time $O(1)$, and that you can evaluate $h_\lambda(a)$ at any point $a \in \mathcal{U}$ in time $O(1)$.

On input $Q$ as above, your algorithm should find $\lambda \in \Lambda$ such that $h_\lambda$ is pretty good for $Q$. The expected running time of your algorithm should be $O(n)$.

You may wish to follow the following outline:

(a) Suppose $R$ is uniformly distributed over $\Lambda$. Let $X$ be the number of $a_i$’s that are not isolated by $h_R$.

Show that $E[X] \leq n(n-1)/m$.

Hint: use indicator variables and linearity of expectation.

(b) Now use Markov’s inequality and the assumption that $m \geq 4n$, to show that a random hash function $h_R$ is pretty good with probability at least $1/2$.

(c) Using part (b), and the assumption that $m \leq 8n$, design an algorithm that actually finds a pretty good hash function in expected time $O(n)$. 

Due: May 8
3. **Composed hash.** Suppose \( h_\lambda \lambda \in \Lambda \) is an \( \epsilon \)-universal family of hash functions from \( U \) to \( V \). Further, suppose that \( \{h'_\lambda\}_\lambda \in \Lambda' \) is an \( \epsilon' \)-universal family of hash functions from \( U' \) to \( V' \), where \( V \subseteq U' \). Show that

\[
\{h'_\lambda \circ h_\lambda\}_{(\lambda, \lambda') \in \Lambda \times \Lambda'}
\]

is an \((\epsilon + \epsilon')\)-universal family of hash functions from \( U \) to \( V' \). (Here, \( h'_\lambda \circ h_\lambda \) is the usual composition of functions \( h'_\lambda \) and \( h_\lambda \), so that \( (h'_\lambda \circ h_\lambda)(a) = h'_\lambda(h_\lambda(a)) \).)

4. **2D hash.** In class, we presented a \((t - 1)/m\)-universal hash family based on polynomial evaluation. This exercise develops a two-dimensional variant. The universe of keys \( \mathcal{U} \) consists of all \( t \times t \) matrices over \( \mathbb{Z}_m \), where \( m \) is prime. We write such a matrix \( A \in \mathcal{U} \) as \( A = (a_{ij}) \), where the indices \( i \) and \( j \) run from 0 to \( t - 1 \). The set of hash function indices \( \Lambda \) consists of pairs \((\lambda_1, \lambda_2) \in \mathbb{Z}_m \times \mathbb{Z}_m \). For \( \lambda = (\lambda_1, \lambda_2) \in \Lambda \) and \( A = (a_{ij}) \in \mathcal{U} \), define

\[
h_\lambda(A) = \sum_{i=0}^{t-1} \sum_{j=0}^{t-1} a_{ij} \lambda_1^i \lambda_2^j. \tag{1}
\]

Show that \( \{h_\lambda\}_{\lambda \in \Lambda} \) is \( 2(t - 1)/m \)-universal.

Hint: re-write the right-hand side of (1) as

\[
\sum_{i=0}^{t-1} \lambda_1^i \left( \sum_{j=0}^{t-1} a_{ij} \lambda_2^j \right),
\]

and then apply the result of the previous exercise (with \( V = \mathcal{U}' = \mathbb{Z}_m^t \)), making use of the fact that the usual polynomial evaluation hash is \((t - 1)/m\)-universal.

5. **2D pattern matching.** In the 2D pattern matching problem, you are given an \( n \times n \) array \( A \) and a \( t \times t \) array \( B \), where \( t \leq n \), and you want to determine if \( B \) appears as a subarray within \( A \). For example, the array

\[
B = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}
\]

appears as a subarray of

\[
A = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 1 & 2 & 3 \\ 2 & 3 & 4 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}.
\]

Adapt the Karp/Rabin pattern matching algorithm using the 2D hash function from the previous exercise to give a probabilistic algorithm that solves this problem. The expected running time should be \( O(n^2 + n^2t^3/m) \), where \( m \) is the prime used in the above hash function. For reasonable choices of \( t \) and \( m \), the first term will dominate, and so the expected running time will be \( O(n^2) \).

Hint: you will have to somehow adapt the “rolling hash” idea of Karp/Rabin to the 2D hash.

6. **Cuckoo hashing.** We have a cuckoo graph \( G \) with \( m \) vertices \( 0, \ldots, m-1 \) (corresponding to hash table slots) and \( n \) edges \( e_1, \ldots, e_n \) (corresponding to keys). Each (undirected) edge \( e_i \) is chosen at random as \( \{u_i, v_i\} \) by selecting the endpoints \( u_i \) and \( v_i \) uniformly and independently from \( \{0, \ldots, m-1\} \). Define \( \alpha := n/m \), which is the “load factor”.

(a) For \( k \geq 1 \), define \( p_k \) to be the probability that \( G \) contains a simple cycle of length \( k \). Here, a simple cycle of length \( k \) is a path \((s_0, s_1, \ldots, s_{k-1}, s_0)\), where the vertices \( s_0, \ldots, s_{k-1} \) are distinct.

Show that \( p_1 \leq \alpha \) and \( p_2 \leq \alpha^2 \).

(b) Show that \( p_k \leq (2\alpha)^k/k \) for all \( k \geq 1 \), with \( p_k \) defined as in part (a).

Notes: The results from part (a) already imply this bound for \( k = 1 \) and \( k = 2 \). We proved this bound for \( k = 3 \) in class. Generalize that proof to arbitrary \( k \).

(c) Prove that for every fixed vertex \( s_0 \), and every \( \ell \geq 1 \), the probability that \( G \) contains a “simple loop” of length \( \ell \) starting at \( s_0 \) is at most \( \ell(2\alpha)^\ell/m \).

Here, a “simple loop” of length \( \ell \) starting at \( s_0 \) is a path \((s_0, \ldots, s_{\ell-1}, s_j)\), where \( j < \ell \) and the vertices \( s_0, \ldots, s_{\ell-1} \) are distinct.