Pairwise independence

**Definition**

The random variables $X_1, X_2, \ldots, X_n$ are said to be **pairwise independent** if, for all $i \neq j$ and any values $a, b$

$$Pr((X_i = a) \cap (X_j = b)) = Pr(X_i = a)Pr(X_j = b)$$

Pairwise independence is a much weaker requirement than mutual independence.
A random bit $Y$ is uniform if $\Pr(Y = 0) = \Pr(Y = 1) = \frac{1}{2}$. We show a method to derive $m = 2^b - 1$ uniform and pairwise independent bits from $b$ mutually independent uniform random bits $X_1, \ldots, X_b$. Enumerate the $m = 2^b - 1$ nonempty subsets of $\{1, 2, \ldots, b\}$ in some order and let $S_j$ denote the $j$th subset. Define $Y_j$ as

$$Y_j = \left( \sum_{i \in S_j} X_i \right) \mod 2.$$
Lemma

The $Y_j$ are pairwise independent uniform bits.

Proof: We use the method of deferred decisions to show that $Y_j$ is a uniform bit. Let $z$ be the largest element in $S_j$. Then whatever the parity of the sum of the first $|S_j| - 1$ bits of $S_j$ is the sum of this number $a$ and $z$ will be 0, resp. one with probability $\frac{1}{2}$ since $z$ is independent of the other bits in $S_j$ and uniform.
Now let $Y_k$ and $Y_r$ be two of the random variables and let $S_k, S_r$ be the corresponding sets. As $S_r \neq S_k$ we can pick $z \in S_r \setminus S_k$. Consider, for any values of $c, d \in \{0, 1\}$

$$\Pr(Y_r = d | Y_k = c).$$

We claim that this equals $\frac{1}{2}$. Again we use deferred decisions: After revealing $S_k \cup S_r - \{z\}$ the variable $Y_k$ is determined but $Y_r$ is not so conditioning on $Y_k = c$ does not change that $Y_r$ is equally likely to be 0 as 1, since $z$ is uniform and independent of all other bits.
We argued that $Pr(Y_r = d|Y_k = c) = \frac{1}{2}$. Hence

$$Pr((Y_k = c) \cap (Y_r = d)) = Pr(Y_r = d|Y_k = c)Pr(Y_k = c)$$

$$= \frac{1}{4}$$

$$= Pr(Y_r = d)Pr(Y_k = c)$$

As this holds for all choices of $k, r$ and all choices of $c, d$ we have proved pairwise independence.
Recall the randomized algorithm for finding as large cut in a graph $G = (V, E)$: assign each vertex $v \in V$ a random color from $\{0, 1\}$ and keep all edges that are properly colored (with 0 and 1). The expected size of this cut is $m/2$, where $m = |E|$. Suppose now that we have $Y_1, Y_2, \ldots < Y_n$ pairwise independent bits, where $n = |V|$. Define the cut by putting all vertices with $Y_i = 0$ on one side and those with $Y_j = 1$ on the other side.
How many edges cross the cut?
For each edge $ij \in E$ let $Z_{ij}$ be the random variable that is 1 if $ij$ crosses the cut and zero otherwise and let $Z = \sum_{ij \in E} Z_{ij}$ be the number of edges crossing the cut.
Since $Y_i$ and $Y_j$ are pairwise independent

$$Pr(Z_{ij} = 1) = Pr(Y_i \neq Y_j) = \frac{1}{2}$$

So

$$E[Z] = E\left[ \sum_{ij \in E} Z_{ij} \right] = \sum_{ij \in E} E[Z_{ij}] = m/2$$
How many random bits did we need? Only \( b = \log_2 (n + 1) \)! (we need \( b \) such that \( 2^b - 1 \geq n \))

By the probabilistic method, there is some setting of the \( b \) bits so that the resulting \( Y_i \)'s define a cut with at least \( m/2 \) edges accross.

Thus we can try all the \( 2^b = O(n) \) possible values of the \( b \) bits:

For a given choice of values to these

- Calculate the values of \( Y_1, Y_2, \ldots, Y_n \)
- Run though all edges and keep those \( ij \) where \( Y_i \neq Y_j \).
- If we get at least \( m/2 \) edges stop, otherwise take the next choice of values for the \( b \) bits
Running time:

- It takes $O(n)$ time to generate all the $2^b$ different bit-settings.
- For a given bitstring ($b$-bits) we can find the values of each $Y_i$ in time $O(nb) = O(n \log n)$.
- Now we can count edges across (and find those) in time $O(m)$.

Altogether our algorithm finds a good cut in time $O(n^2 \log n + nm)$.

The $\log n$ factor can be removed by ordering the vertices appropriately (lexicographical ordering of subsets of $\{1, 2, \ldots, b\}$).
Running time is worse than our derandomized algorithm using conditional expectations!
BUT: this new algorithm can be parallellized easily: use $n$ processors, one for each setting of the $b$ bits. This gives an $O(m)$ parallel algorithm, same complexity as the other derandomized algorithm.
If we used $O(nm)$ processors, one per combination of an edge and a setting of bits, we can decide, for each edge in constant time whether it crosses the cut and then collect the results (one result for each of the $O(n)$ bit-settings) in time $O(\log n)$ (we can find the sum of $n$ numbers in time $O(\log n)$ using $O(n)$ processors).
Perfect Hashing

Goal: Store a static disctionary of \( n \) items in a table of \( O(n) \) space such that any search takes \( O(1) \) time.
### Universal hash functions

**Definition**

Let $U$ be a universe with $|U| \geq n$ and $V = \{0, 1, \ldots, n-1\}$. A family of hash functions $\mathcal{H}$ from $U$ to $V$ is said to be $k$-universal if, for any elements $x_1, x_2, \ldots, x_k$, when a hash function $h$ is chosen uniformly at random from $\mathcal{H}$,

$$\Pr(h(x_1) = h(x_2) = \ldots = h(x_k)) \leq \frac{1}{n^{k-1}}.$$
Example of 2-Universal Hash Functions

Universe \( U = \{0, 1, 2, \ldots, m - 1\} \)
Table keys \( V = \{0, 1, 2, \ldots, n - 1\} \), with \( m \geq n \).
A family of hash functions obtained by choosing a prime \( p \geq m \),

\[
h_{a,b}(x) = ((ax + b) \mod p) \mod n,
\]
and taking the family

\[
\mathcal{H} = \{h_{a,b} \mid 1 \leq a \leq p - 1, 0 \leq b \leq p\}.
\]

Lemma

\( \mathcal{H} \) is 2-universal.
Lemma

Assume that \( m \) elements are hashed into an \( n \) bin chain hashing table, using a hash function \( h \) chosen uniformly at random from a 2-universal family. For an arbitrary element \( x \), let \( X \) be the number of items at the bin \( h(x) \).

\[
E[X] \leq \begin{cases} 
\frac{m}{n} & \text{if } x \notin S \\
1 + \frac{m-1}{n} & \text{if } x \in S.
\end{cases}
\]

Proof.

Let \( X_i = 1 \) if the \( i \)-th element of \( S \) is in the same bin as \( x \) and 0 otherwise. \( \Pr(X_i = 1) \leq 1/n \)

If \( x \notin S \), \( E[X] = E[\sum_{i=1}^{m} X_i] = \sum_{i=1}^{m} E[X_i] \leq m/n \),

If \( x \in S \) (assume \( x \) is \( s_1 \)),

\[
E[X] = E[\sum_{i=1}^{m} X_i] = 1 + \sum_{i=2}^{m} E[X_i] \leq 1 + (m - 1)/n.
\]
Lemma

If \( h \in \mathcal{H} \) is chosen uniformly at random from a 2-universal family of hash functions mapping the universe \( U \) to \( [0, n-1] \), then for any set \( S \subset U \) of size \( m \), the probability of \( h \) being perfect is at least \( 1/2 \) when \( n \geq m^2 \).

Proof.

Let \( s_1, s_2, \ldots, s_m \) be the \( m \) items of \( S \). Let \( X_{ij} \) be 1 if the \( h(s_i) = h(s_j) \) and 0 otherwise. Let \( X = \sum_{1 \leq i < j \leq n} X_{ij} \).

\[
E[X] = E \left[ \sum_{1 \leq i < j \leq n} X_{ij} \right] = \sum_{1 \leq i < j \leq m} E[X_{ij}] \leq \binom{m}{2} \frac{1}{n} < \frac{m^2}{2n},
\]

Markov’s inequality yields \( \Pr(X \geq m^2/n) \leq \Pr(X \geq 2E[X]) \leq \frac{1}{2} \). When \( n \geq m^2 \), \( \Pr(X < 1) \geq 1/2 \), and a randomly chosen hash function is perfect with probability at least \( 1/2 \).
The two-level approach gives a perfect hashing scheme for \( m \) items using \( O(m) \) bins.

Level I: use a hash table with \( n = m \). Let \( X \) be the number of collisions,

\[
\Pr(X \geq m^2/n) \leq \Pr(X \geq 2\mathbb{E}[X]) \leq \frac{1}{2}.
\]

When \( n = m \), there exists a choice of hash function from the 2-universal family that gives at most \( m \) collisions.
Level II: Let $c_i$ be the number of items in the $i$-th bin. There are $\binom{c_i}{2}$ collisions between items in the $i$-th bin, thus

$$
\sum_{i=1}^{m} \binom{c_i}{2} \leq m.
$$

For each bin with $c_i > 1$ items, we find a second hash function that gives no collisions using space $c_i^2$. The total number of bins used is bounded above by

$$
m + \sum_{i=1}^{m} c_i^2 \leq m + 2 \sum_{i=1}^{m} \binom{c_i}{2} + \sum_{i=1}^{m} c_i \leq m + 2m + m = 4m.
$$

Hence the total number of bins used is only $O(m)$. 
Definition

A family of $k$-perfect hash functions from $\{1, 2, \ldots, n\}$ to $\{1, 2, \ldots, k\}$, where $k < n$ is a family $\mathcal{H}$ of hash functions such that for every subset $S$ of $\{1, 2, \ldots, n\}$ with $|S| = k$ at least one of the hash functions $h \in \mathcal{H}$ is perfect on $S$, that is $h$ is a 1-1 map of $S$ onto $\{1, 2, \ldots, k\}$.

Theorem (Schmidt and Segal, 1990)

For all $n, k$ with $n > k$ there exists a $k$-perfect family $\mathcal{H}$ of hash functions of size $2^{O(k)} \log^2 n$ (we can specify each function in $\mathcal{H}$ with $O(k) + 2 \log \log n$ bits). For each function $h \in \mathcal{H}$ and $i \in \{1, 2, \ldots, n\}$ we can calculate $h(i)$ in $O(1)$ time.
Derandomizing color-coding algorithms

What we need is a family of $k$-colorings of $G$ such that for each $V' \subset V$ with $|V'| = k$ there is at least one of the colorings where all vertices of $V'$ receive distinct colors. This is exactly the property of a $k$-perfect family of hash functions. So the derandomization is done by going through the $2^{O(k) \log^2 n}$ different functions in such a family and for each of these testing, using e.g. the dynamic programming algorithm for $k$-path, whether there is a colorful $k$-path. Since $\mathcal{H}$ is $k$-perfect, if $G$ does have a $k$-path, at least one of the hash functions will reveal this path (it will become colorful).