1. Let T be a treap with $n \geq 2$ vertices.

   (a) What is the *exact* expected number of leaves in T?

   (b) What is the *exact* expected number of nodes in T that have two children?

   (c) What is the *exact* expected number of nodes in T that have exactly one child?

   Prove that your answers are correct. *[Hint: What is the probability that the node with the kth smallest search key has no children, one child, or two children?]*

2. Every Halloween, hundreds of ghosts rise from their graves and haunt the houses of Sham-Poobanana. This is not as straightforward as it sounds, for two reasons. First, each ghost can only haunt houses where they spent significant time when they were alive. Second, at most one ghost can haunt each house. There are n ghosts and n houses.

   (a) Describe and analyze an efficient algorithm that either assigns each ghost a distinct house that they can haunt, or correctly reports that such an assignment is impossible. Your input is a two-dimensional boolean array $CanHaunt[1..n, 1..n]$, where $CanHaunt[i, j] = \textsc{True}$ if and only if ghost i can haunt house j.

   (b) Oh, no! Beetlejuice broke into the main office and assigned each ghost to a house they can't haunt! Halloween is ruined! Stay-Puft proposes the following strategy to fix everyone's assignments. At exactly midnight, each ghost will give their assigned house to another ghost that actually wants it. For example, suppose

   - Agnes was assigned house A, but she can only haunt houses C and D.
   - Banquo was assigned house B, but he can only haunt houses A and C.
   - Casper was assigned house C, but he can only haunt houses A and D.
   - Daayan was assigned house D, but she can only haunt houses B and C.

   The ghosts can fix their assignment as follows: Agnes gives house A to Banquo; Banquo gives house B to Daayan; Casper gives house C to Agnes, and Daayan gives house D to Casper.

   Describe and analyze an efficient algorithm to compute an exchange that results in a valid assignment of ghosts to houses. The input to your algorithm is the Boolean array *CanHaunt* from part (a) and a second array $Assigned[1..n]$, where $Assigned[i]$ is the index of the house originally assigned to ghost i. The correct output is an array $GiveTo[1..n]$, where $GiveTo[i] = j$ means ghost i should give their house to ghost j.

   You can assume that $CanHaunt[i, Assigned[i]] = \textsc{False}$ for every index i (that is, no ghost can haunt their assigned house) and that a valid exchange exists.

[Questions 3 and 4 are on the back.]

3. Recall that a family $\mathcal{H}$ of hash functions is ***universal*** if $\Pr_{h\in\mathcal{H}}[h(x) = h(y)] \le 1/m$ for all distinct items $x \neq y$, where m is the size of the hash table. For any fixed hash function h, a ***collision*** is an unordered pair of distinct items $x \neq y$ such that $h(x) = h(y)$.

   Suppose we hash a set of n items into a table of size $m = 2n$, using a hash function h chosen uniformly at random from some universal family. Assume $\sqrt{n}$ is an integer.

   (a) ***Prove*** that the expected number of collisions is at most $n/4$.

   (b) ***Prove*** that the probability that there are at least $n/2$ collisions is at most $1/2$.

   (c) ***Prove*** that the probability that any subset of more than $\sqrt{n}$ items all hash to the same address is at most $1/2$. *[Hint: Use part (b).]*

   (d) Now suppose we choose h at random from a ***4-uniform*** family of hash functions, which means for all distinct items w, x, y, z and all addresses i, j, k, l, we have

$$\Pr_{h\in\mathcal{H}}\Big[h(w) = i \ \wedge\ h(x) = j \ \wedge\ h(y) = k \ \wedge\ h(z) = \ell\Big] = \frac{1}{m^4}.$$

   Prove that the probability that any subset of more than $\sqrt{n}$ items all hash to the same address is at most $O(1/n)$.

[Hint: All four statements have short elementary proofs via tail inequalities.]

4. Your friends are organizing a board game party, and because they admire your personal dice collection, they ask you to bring dice. You choose several beautiful, perfectly-balanced, six-sided dice to bring to the party. Unfortunately, by the time you arrive, the other guests have already chosen a game ("Let's Summon Demons") that requires 20-sided dice! So now you get to improvise.

   (a) Describe an algorithm to simulate one roll of a fair 20-sided die using independent rolls of a fair 6-sided die *and no other source of randomness*. Equivalently, describe an implementation of RANDOM(20), whose only source of randomness is an implementation of RANDOM(6).

   (b) What is the *exact* expected number of 6-sided-die rolls (or calls to RANDOM(6)) executed by your algorithm?

   (c) Derive an upper bound on the probability that your algorithm requires more than N rolls. Express your answer as a function of N.

   (d) Estimate the smallest number N such that the probability that your algorithm requires more than N rolls is less than δ. Express your answer as a function of δ.

   You do ***not*** need to prove that your answers are correct.

Some Useful Inequalities

Suppose X is the sum of random indicator variables $X_1, X_2, \ldots, X_n$.
For each index i, let $p_i = \Pr[X_i = 1] = \mathrm{E}[X_i]$, and let $\mu = \sum_i p_i = \mathrm{E}[X]$.

- **Markov's Inequality:**

$$\Pr[X \geq x] \leq \frac{\mu}{x} \qquad \text{for all } x > 0, \text{ and therefore...}$$
$$\Pr[X \geq (1+\delta)\mu] \leq \frac{1}{1+\delta} \qquad \text{for all } \delta > 0$$

- **Chebyshev's Inequality:** If the variables X_i are pairwise independent, then...

$$\Pr[(X-\mu)^2 \geq z] < \frac{\mu}{z} \qquad \text{for all } z > 0, \text{ and therefore...}$$
$$\Pr[X \geq (1+\delta)\mu] < \frac{1}{\delta^2\mu} \qquad \text{for all } \delta > 0$$
$$\Pr[X \leq (1-\delta)\mu] < \frac{1}{\delta^2\mu} \qquad \text{for all } \delta > 0$$

- **Higher Moment Inequalities:** If the variables X_i are $2k$-wise independent, then...

$$\Pr[(X-\mu)^{2k} \geq z] = O\left(\frac{\mu^k}{z}\right) \qquad \text{for all } z > 0, \text{ and therefore...}$$
$$\Pr[X \geq (1+\delta)\mu] = O\left(\frac{1}{\delta^{2k}\mu^k}\right) \qquad \text{for all } \delta > 0$$
$$\Pr[X \leq (1-\delta)\mu] = O\left(\frac{1}{\delta^{2k}\mu^k}\right) \qquad \text{for all } \delta > 0$$

- **Chernoff's Inequality:** If the variables X_i are fully independent, then...

$$\Pr[X \geq x] \leq e^{x-\mu}\left(\frac{\mu}{x}\right)^x \qquad \text{for all } x \geq \mu, \text{ and therefore...}$$
$$\Pr[X \geq (1+\delta)\mu] \leq e^{-\delta^2\mu/3} \qquad \text{for all } 0 < \delta < 1$$
$$\Pr[X \leq (1-\delta)\mu] \leq e^{-\delta^2\mu/2} \qquad \text{for all } 0 < \delta < 1$$

- **The World's Most Useful Inequality:** $1 + x \leq e^x$ for all x
- **The World's Most Useful Limit:** $\lim_{n\to\infty}\left(1+\frac{1}{n}\right)^n = e$

Hashing Properties

$\mathcal{H}$ is a set of functions from some universe $\mathcal{U}$ to $[m] = \{0, 1, 2, \ldots, m-1\}$.

- **Universal:** $\Pr_{h\in\mathcal{H}}[h(x) = h(y)] \leq \frac{1}{m}$ for all distinct items $x \neq y$
- **Near-universal:** $\Pr_{h\in\mathcal{H}}[h(x) = h(y)] \leq O\left(\frac{1}{m}\right)$ for all distinct items $x \neq y$
- **Strongly universal:** $\Pr_{h\in\mathcal{H}}[h(x) = i \text{ and } h(y) = j] = \frac{1}{m^2}$ for all distinct $x \neq y$ and all i and j
- **2-uniform:** Same as strongly universal.
- **Ideal Random:** Fiction.