Warm-up

Problem 1. If you repeatedly perform an experiment whose outcome has a Geometric Distribution with probability of success p, what is the expected number of times you need to repeat the experiment before you get a success?

Solution 1. The expected number of repetitions required before you get a success is $1/p$.

Problem solving

Problem 2. Suppose you only have access to a biased coin that lands heads with probability p and tails with probability $(1-p)$. Show how to design a fair coin without even knowing what p is. How many times does your algorithm flip the biased coin in expectation?

Solution 2. Flip the coin twice. If the outcomes are HT, output Heads. If the outcomes are TH, output Tails. If the outcomes are HH or TT, try again.

The probability of not having to try again is $2p(1-p)$. Therefore, we expect that the number of tries is $1/2p(1-p)$ because that is the expected value of the Geometric Distribution with probability $2p(1-p)$. Each try involves two coin flips of the biased coins, so the expected number of coin flips is therefore $1/p(1-p)$.

Problem 3. Suppose you only have access to a fair coin that lands heads with probability $1/2$ and tails with probability $1/2$. Show how to design an algorithm for uniformly sampling an integer from $\{1,\dots,n\}$. How many times does your algorithm flip the coin in expectation?

Solution 3. Let $k = \lfloor \log_2 n \rfloor + 1$. Flip the fair coin k many times. Interpret the sequence of heads and tails as a binary number, i.e.,

$$\text{HHTHTTH} \to 1101001.$$

It is easy to see that the number is distributed uniformly from $\{0,\dots,2^k-1\}$. Note that $\log_2 n \le k \le \log_2 n + 1$, so

$$n-1 \le 2^k - 1 \le 2n-1.$$

If the number constructed happens to be in the range $\{1,\dots,n\}$, then we output it. Otherwise, we try again.

The probability of not having to try again is at least $1/2$. Therefore, the expected number of tries is 2 because the expected value of a Geometric Distribution with probability $1/2$ is 2. Each try involves flipping $\log_2 n + 1$ fair coins, so the expected number of coin flips is $2(\log_2 n + 1)$.

Problem 4. Suppose you only have access to a random generator for sampling real numbers from the interval $[0, 1]$. Show how to design an algorithm for uniformly sampling points (x, y) from the square $[-1, 1]^2$ (i.e., $-1 \leq x \leq 1$ and $-1 \leq y \leq 1$). How many samples does your algorithm need in expectation?

Solution 4. We sample two numbers x' and y' and return $x = 2x' - 1$ and $y = 2y' - 1$. Since the random generator generates a number from the real interval $[0, 1]$, all x' and y' are equally likely, which implies that all x and y are equally likely. Hence, this generates a point in the square uniformly at random.

Problem 5. Suppose you only have access to a random generator for sampling real numbers from the interval $[0, 1]$. Show how to design an algorithm for uniformly sampling points (x, y) from the unit radius disk centered at the origin (i.e., $x^2 + y^2 \leq 1$). How many samples does your algorithm need in expectation?

Solution 5. Sample a point from the square, if the point lies inside the disk output that; otherwise, reject it and try again.

The probability of not having to try again is $\pi/4$ (the area of the disk is π and that of the square is 4), so the expected number of tries is $4/\pi$. Each try takes two samples, so this approach requires $8/\pi$ samples overall.

Problem 6. Suppose we want to add a new operation to the existing skip list implementation. This operation, RANGESEARCH($k1$, $k2$), returns all the items with keys in the range $[k1, k2]$. Design this operation and show that it runs in expected time $O(\log n + s)$, where n is the number of elements in the skip list and s is the number of items returned.

Solution 6. Since every element is stored at the lowest layer of the skip list and we are allowed to report the s elements in $O(s)$ time, a simple approach to this problem is to search for $k1$ and after that traverse the bottom layer of the skip list until we reach a node with key greater than $k2$. Note that our search ends at a node with key at most $k1$, so we need to check if it should be reported before proceeding to scan its successors.

This algorithm is correct, since it starts from a node that's at most $k1$ and then scans the bottom level of the skip list until it reaches a node with key larger than $k2$, reporting all nodes in between.

This algorithm's running time is determined by the length of the search path to $k1$ and the number of elements we traverse until we reach a node with key greater than $k2$. Recall that the expected height of a skip list if $O(\log n)$, so that's the time we spend searching for $k1$. While reporting all nodes in the range $[k1, k2]$, we note that if there are s such nodes to report, we check at most $s + 2$ nodes: the s nodes that need to be reported and at most one node before the range (where our search for $k1$ ends) and one node after the range (where our scan of the lowest level of the skip lists ends). Hence, we spend $O(s)$ time reporting these nodes. In total this leads to an $O(\log n + s)$ expected time algorithm.