15-756: Randomized Algorithms
Homework 2

Due: Wednesday Feb 4, 11:59pm ET. Upload solution to Gradescope (accessible via Canvas website).

1. Consider an unbiased random walk, starting somewhere in the range $[-n, n]$, and taking steps of the form $+1$ and -1 with equal probability. In class, we proved that the expected time for the random walk to reach either n or $-n$ is $O(n^2)$. Prove that, with probability at least $1 - 1/n$, the random walk reaches n or $-n$ within time $O(n^2 \log n)$.

2. Design and analyze an $O(n^3 \log n/\ell)$-time algorithm that, given an undirected and unweighted n-node graph, computes for all pairs of vertices u, v an estimate $D(u, v)$, such that with probability at least $1 - 1/n$, for all pairs of vertices u, v for which the shortest path has length at least ℓ, the estimate $D(u, v)$ is equal to the true distance $d(u, v)$ between u and v. (Your algorithm does not need to know for which pairs $u, v, D(u, v)$ is the correct distance, and it doesn't have to guarantee anything about the estimate $D(u, v)$ for vertices u, v that are distance less than ℓ apart.)

3. Suppose you are given a fair coin (that is, it lands heads/tails with probability 1/2 each) and want to use it to "simulate" a coin that lands heads with probability exactly 1/3. Specifically, you will design an algorithm whose only access to randomness is by flipping the fair coin (repeatedly, if desired), and your algorithm should return "heads "with probability exactly 1/3 and "tails" with probability exactly 2/3.

   (a) Prove that it is impossible to do this if the algorithm is only allowed to flip the fair coin at most $1,000,000,000$ times. [HINT: Read the next two parts of the problem first.]

   (b) Design an algorithm for the above task that flips the fair coin a finite number of times in expectation.

   (c) Show that for any value v in the interval $[0, 1]$, there is an algorithm that flips a fair coin at most 2 times in expectation, and outputs "heads" with probability v and "tails" with probability $1 - v$. Note: if you do this part correctly, you can write "follows from (c)" in part (b) and get full credit for both parts. [HINT: Think about representing the desired probability in its binary representation.]

4. **Anonymous Survey (Optional):** Fill out this survey to give feedback on the HW.

Problem 1

Let T be the random variable for the time the random walk reaches n or $-n$. Then from the problem statement we have $\mathbb{E}[T] \leq c(n^2)$.
By Markov's inequality we have

$$\Pr[T \geq k\mathbb{E}[T]] \leq \frac{1}{k} \implies \Pr[T \geq 2cn^2] \leq \frac{1}{2}.$$

Then consider if we've ran this trial k times, where each trial we run for $2cn^2$ steps. We end the trial when we reach n or $-n$. We start each trial where the last one ended. Then the probability we fail after k trials is is at least $\left(\frac{1}{2}\right)^k$ as each trial fails with at least probability $\frac{1}{2}$.
We want the failure probability to be at most $\frac{1}{n}$ so

$$\left(\frac{1}{2}\right)^k \leq \frac{1}{n} \implies k \geq \log_2 n.$$

So we want to run the trial for $\log_2 n$ times (won't always be integer but we can just round up). So the total number of steps we take is

$$O(n^2) \cdot O(\log_2(n)) = O(n^2 \log n).$$

From this procedure our chance of success is at least $1 - \frac{1}{n}$.

Problem 2

Consider the shortest path from $u \to v$, say $P_{u,v}$ then such a path has length at least ℓ (we don't care about results for distances shorter than ℓ). Then along this path there are at least ℓ vertices. Let S be a set of $\frac{3n \log n}{\ell}$ vertices randomly chosen from the graph and we want it so that with high probability $P_{u,v}$ contains a vertex in S.
We run BFS on all vertices in S to compute the shortest path distance $d(s, v)$ for all $v \in V(G)$. Then for any pair of vertices u, v we have

$$D(u, v) = \min_{s \in S} d(u, s) + d(s, v).$$

When we ran the BFS on all vertices in S, so the total run time is

$$O(|S|(|E| + |V|)) = O\left(\frac{3n \log n}{\ell}(n + n^2)\right) = O(n^3 \log n/\ell).$$

Then to compute the minimum distance across all pairs of vertices, there are $\binom{n}{2} = O(n^2)$ pairs of vertices so the total run time is

$$O(|S|)O(n^2) = O(n^3 \log n/\ell).$$

To prove that this algorithm is correct with very high probability, we know

$$d(u, v) = d(u, s) + d(s, v) \iff s \in P_{u,v}.$$

So for the algorithm to fail for a single pair u, v we need all $s \in S$ to not be on $P_{u,v}$, so the probability is

$$\Pr[\forall s \in S, s \notin P_{u,v}] = \prod_{s \in S} \Pr[s \notin P_{u,v}] \leq \prod_{s \in S} \left(1 - \frac{\ell}{n}\right) = \left(1 - \frac{\ell}{n}\right)^{\frac{3n \log n}{\ell}}.$$

Note: We can multiply because we sample S independently. Using $(1 - x) \leq e^{-x}$ we have

$$\left(1 - \frac{\ell}{n}\right)^{\frac{3n \log n}{\ell}} \leq e^{-\frac{3n \log n}{\ell} \cdot \frac{\ell}{n}} = e^{-3 \log n} = \frac{1}{n^3}.$$

Taking the union bound over all pairs of vertices we have

$$\Pr[\text{Algorithm fails for any pair } u, v] = \sum_{u,v} \Pr[\text{Algorithm fails for pair } u, v] \leq \binom{n}{2} \cdot \frac{1}{n^3} < n^2 \cdot \frac{1}{n^3} = \frac{1}{n}.$$

So we can conclude that the algorithm is correct with probability at least $1 - \frac{1}{n}$.

Problem 3

(a) If the algorithm is only allowed to flip a fair coin $K := 1000000000$ times then there are at most 2^K possible sequences of heads and tails. Since the coin is fair, each sequence has probability $\frac{1}{2^K}$ of occurring. So if x is the number of outcomes that result in outputting heads then if we set $v = \frac{1}{3}$ then we need

$$\frac{x}{2^K} = \frac{1}{3} \iff x = \frac{2^K}{3}.$$

However x must be an integer so 2^K must be divisible by 3 so a contradiction.

(b) Follows from (c)

(c) Let the binary representation of v be $v = (0.b_1b_2\dots)_2$. We perform the following algorithm:

The Algorithm: Initialize $i = 1$.

- Loop:
  (a) Flip the fair coin. Let the result be r_i (Encode Tails=0, Heads=1).
  (b) Compare the random bit r_i with the i-th bit of the target v (denoted b_i).
  - **Case 1 ($r_i < b_i$):** Since $r_i = 0$ and $b_i = 1$, the random number generated is strictly less than v. Return HEADS.
  - **Case 2 ($r_i > b_i$):** Since $r_i = 1$ and $b_i = 0$, the random number generated is strictly greater than v. Return TAILS.
  - **Case 3 ($r_i = b_i$):** The bits match. Increment i and **continue** to the next flip.

Correctness (Probability of Heads): The algorithm returns HEADS at step i only if we matched the first $i-1$ bits (probability $(1/2)^{i-1}$) and then encountered the case $r_i < b_i$. The condition $r_i < b_i$ is only possible if the target bit $b_i = 1$ and we flip $r_i = 0$ (probability $1/2$). Therefore, the total probability of returning HEADS is:

$$\Pr[\text{Heads}] = \sum_{i:b_i=1} \Pr[\text{Match } i-1 \text{ bits}] \cdot \Pr[r_i = 0] = \sum_{i:b_i=1} \left(\frac{1}{2}\right)^{i-1} \cdot \frac{1}{2} = \sum_{i=1}^{\infty} \frac{b_i}{2^i} = v.$$

Runtime Analysis (Expected Flips): Let T be the random variable for the number of flips performed. The algorithm stops at step i if and only if $r_i \neq b_i$. Since the coin is fair, $\Pr[r_i = b_i] = 1/2$ and $\Pr[r_i \neq b_i] = 1/2$.

$$\mathbb{E}[T] = \sum_{i=1}^{\infty} i \cdot \Pr(T = i) = \sum_{i=1}^{\infty} i \cdot \left(\frac{1}{2}\right)^{i-1} \cdot \frac{1}{2} = \sum_{i=1}^{\infty} i \left(\frac{1}{2}\right)^{i}.$$

Using the identity $\sum_{i=1}^{\infty} ix^i = \frac{x}{(1-x)^2}$ with $x = 1/2$:

$$\mathbb{E}[T] = \frac{1/2}{(1-1/2)^2} = \frac{1/2}{1/4} = 2.$$

Note: The identity is derived from $\frac{1}{1-x} = \sum_{i=0}^{\infty} x^i \implies \frac{d}{dx}\left(\frac{1}{1-x}\right) = \sum_{i=0}^{\infty} ix^{i-1}$. So $\frac{x}{(1-x)^2} = \sum_{i=0}^{\infty} ix^{i-1}$.