PACS Part 2, Lecture 1

Useful Equalities and Inequalities

- Union bound: $\mathbb{P}(\bigcup_{i=1}^{\infty} A_i) \leq \sum_{i=1}^{\infty} \mathbb{P}(A_i)$
- Markov's inequality: $\mathbb{P}(X \geq z) \leq \mathbb{E}X/z$ if X and z are nonnegative
- $1 - x \leq e^{-x}$
- Law of total probability: $\mathbb{P}(E) = \sum_{i=1}^{\infty} \mathbb{P}(E \mid A_i) \cdot \mathbb{P}(A_i)$ if $(A_i)_{i \geq 1}$ is a partition of Ω
- Linearity of expectation: $\mathbb{E} \sum_{i=1}^{n} X_i = \sum_{i=1}^{n} \mathbb{E}X_i$
- $\mathbb{E}X = \sum_{k=0}^{\infty} \mathbb{P}(X > k)$ if $X \in \{0, 1, 2, \dots\}$

Coupon Collector: Expectation

We want to collect all of n different coupons. At every time step, we get an i.i.d. coupon chosen uniformly at random from the set of all possible coupons. What is the number X of time steps until we have every coupon at least once?

First answer: expectation. We can write $X = \sum_{i=1}^{n} X_i$ where X_i is the number of time steps needed to go from $i-1$ unique collected coupons to i. By linearity of expectation, it suffices to compute each $\mathbb{E}X_i$ and sum them up.

Each X_i follows a geometric distribution with success probability $p_i = 1 - \frac{i-1}{n} = \frac{n-i+1}{n}$. The expectation is thus $\mathbb{E}X_i = 1/p_i = \frac{n}{n-i+1}$. Summing these, we get $\mathbb{E}X = n \sum_{i=1}^{n} \frac{1}{n-i+1} = n \sum_{i=1}^{n} \frac{1}{i} = nH(n)$, where $H(n)$ is the nth harmonic number.

Since $H(n) = \log n + O(1)$ as $n \to \infty$, we get $\mathbb{E}X \sim n \log n$.

Chebyshev's Inequality

$$\mathbb{P}(|X - \mathbb{E}X| \geq a) \leq \frac{\mathrm{Var}(X)}{a^2}$$

Proof: use Markov on $(X - \mathbb{E}X)^2$

Coupon Collector: Variance

Since the X_i are independent, we have $\mathrm{Var}(X) = \sum_{i=1}^{n} \mathrm{Var}(X_i)$. The variance of the geometric random variable X_i is equal to $\mathrm{Var}(X_i) = (1 - p_i)/p_i^2 \leq 1/p_i^2$. Thus:

$$\mathrm{Var}(X) \leq \sum_{i=1}^{n} \frac{n^2}{(n-i+1)^2} \leq n^2 \sum_{i=1}^{n} \frac{1}{i^2} \leq n^2 \frac{\pi^2}{6} = O(n^2)$$

We thus get $\mathbb{P}(|X - nH(n)| \geq nH(n)) = O(1/\log^2 n)$.

Quicksort

This is the quicksort algorithm to sort the list S of distinct elements from a totally ordered universe.

1. If S has at most one element return S.
2. Choose a pivot element $s \in S$.
3. Compare each element of S to s and construct lists S_1 and S_2 with the elements that are less than, resp. greater than, s.
4. Recursively sort S_1 and S_2.
5. Return S_1, x, S_2.

Making the choice in step 2 an independent uniformly random one, this is a randomized algorithm.

Let $x_1, \ldots, n_n$ be the input values and $y_1, \ldots, y_n$ the same values in increasing order. Let X be the number of comparisons, and let X_{ij} be the indicator variable whether y_i and y_j are every compared (for $i < j$).

We have $\mathbb{E}X = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \mathbb{E}X_{ij} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} = \sum_{k=2}^{n} \sum_{i=1}^{n+1-k} \frac{2}{k} = \sum_{k=2}^{n} (n+1-k)\frac{2}{k} = (2n+2)\sum_{k=1}^{n} \frac{1}{k} - 4n = 2n \log n + O(n)$.

Randomized Median Algorithm

Assume a set of n distinct elements from a totally ordered universe. For simplicity, we assume that n is odd.

1. Pick a multiset of $\lceil n^{3/4} \rceil$ elements in S, independently and uniformly at random with replacement.
2. Sort the multiset R.
3. Let d be the $\lfloor \frac{1}{2} n^{3/4} - \sqrt{n} \rfloor$th smallest element of R.
4. Let u be the $\lceil \frac{1}{2} n^{3/4} + \sqrt{n} \rceil$th smallest element of R.
5. Compute $C = \{x \in S \mid d \leq x \leq u\}$, $\ell_d = |\{x \in S \mid x < d\}|$, and $\ell_u = |\{x \in S \mid x > u\}|$.
6. If $\ell_d > n/2$ or $\ell_u > n/2$, then FAIL.
7. If $|C| \leq 4n^{3/4}$, then sort C, else FAIL.
8. Output the $(\lfloor n/2 \rfloor - \ell_d + 1)$th smallest element of C.

Idea: sample d and u such that the median m is between d and u, then order the set C of elements between d and u and find the index of the median in C (after having counted the elements smaller than d).

Consider the three following error events:

- $\mathcal{E}_1$: $Y_1 = |\{r \in R \mid r \leq m\}| < \frac{1}{2} n^{3/4} - \sqrt{n}$
- $\mathcal{E}_2$: $Y_2 = |\{r \in R \mid r \geq m\}| < \frac{1}{2} n^{3/4} - \sqrt{n}$

- $\mathcal{E}_3$: $|C| > 4n^{3/4}$

The event $\mathcal{E}_1$ is equivalent to $\ell_d > n/2$ and does not happen with high probability: Denoting by X_i the indicator variable whether the ith sample for R is $\leq m$, we have that $\mathcal{E}_1$ is equivalent to $Y_1 = \sum_{i=1}^{n^{3/4}} X_i < \frac{1}{2}n^{3/4} - \sqrt{n}$. Since Y_1 is a sum of i.i.d. Bernoulli trials, we have:

$$\mathrm{Var}(Y_1) = n^{3/4}\left(\frac{1}{2} + \frac{1}{2n}\right)\left(\frac{1}{2} - \frac{1}{2n}\right) \leq \frac{1}{4}n^{3/4}$$

Applying Chebyshev's inequality then gives:

$$\mathbb{P}(\mathcal{E}_1) \leq \frac{\mathrm{Var}(Y_1)}{n} \leq \frac{1}{4}n^{-1/4}$$

The event $\mathcal{E}_2$ is equivalent to $\ell_u > n/2$ and satisfies the same probability bound.

The event $\mathcal{E}_3$ also does not happen with high probability: The event $\mathcal{E}_3$ implies either the event $\mathcal{E}_{3,1}$: at least $2n^{3/4}$ elements of C are $> m$ or the event $\mathcal{E}_{3,2}$ with $> m$ replaced by $< m$. The event $\mathcal{E}_{3,1}$ implies that the order of u in S is at least $\frac{1}{2}n + 2n^{3/4}$, i.e., that R has at least $\frac{1}{2}n^{3/4} - \sqrt{n}$ samples among the $\frac{1}{2}n - 2n^{3/4}$ largest elements of S. Denoting by X_i the indicator variable whether the ith sample for R is among the $\frac{1}{2}n - 2n^{3/4}$ largest elements of S, we have

$$\mathbb{E}X = \mathbb{E}\sum_{i=1}^{n^{3/4}} X_i = \frac{1}{2}n^{3/4} - 2\sqrt{n}$$

and

$$\mathrm{Var}(X) = n^{3/4}\left(\frac{1}{2} - 2n^{-1/4}\right)\left(\frac{1}{2} + 2n^{-1/4}\right)$$

Applying Chebyshev's inequality and the union bound then gives:

$$\mathbb{P}(\mathcal{E}_3) \leq 2\frac{\mathrm{Var}(X)}{n} \leq \frac{1}{2}n^{-1/4}$$

By the union bound, the probability of the algorithm failing is at most $n^{-1/4}$.

Las Vegas from Monte Carlo

As stated, the algorithm is a Monte Carlo algorithm, i.e., it doesn't always give the correct answer when it terminates. Since we can check the correctness of the median in linear time, we can easily transform it into a Las Vegas algorithm, i.e., an algorithm that always gives the correct answer, but whose running time may be randomized.

The correctness of each iteration of the algorithm is an i.i.d. Bernoulli trial with success probability $p \geq 1 - n^{-1/4}$. The expected running time is thus at most:

$$O(n) \cdot \sum_{k=0}^{\infty}(1-p)^k = O(n) \cdot \frac{1}{p} = O(n)$$