Randomized Algorithms

1 Preliminaries

1.1 Definitions of Probability

Definition 1. A **probability space** (Ω, p) is a finite set Ω with $p(\omega) \geq 0 \; \forall \omega \in \Omega$ and $\sum_{\omega \in \Omega} p(\omega) = 1$.

Definition 2. An **event** $\mathcal{E} \subseteq \Omega$ has $\Pr[\mathcal{E}] = \sum_{\omega \in \mathcal{E}} p(\omega)$.

Definition 3. A **random variable** is a function $X : \Omega \to \mathbb{R}$, and an **indicator random variable** is a function $X : \Omega \to \{0, 1\}$. An indicator random variable $X_{\mathcal{E}}$ of $\mathcal{E}$ has:

$$X_{\mathcal{E}}(\omega) = \begin{cases} 1 & \omega \in \mathcal{E} \\ 0 & \text{o.w.} \end{cases}$$

Definition 4. The **expectation** of X is:

$$\mathbb{E}[X] = \sum_i i \Pr[X = i]$$

$$= \sum_{\omega \in \Omega} X(\omega) p(\omega)$$

Proposition 5. *Expectation is <u>always</u> linear:*

$$\mathbb{E}[\alpha X + \beta Y] = \alpha \mathbb{E}[X] + \beta \mathbb{E}[Y]$$

Definition 6. Events $\mathcal{E}_1, \ldots, \mathcal{E}_n$ are **independent** iff for all sets $S \subseteq \{1, \ldots, n\}$, $\Pr[\bigwedge_{i \in S_i} \mathcal{E}_i] = \prod_{i \in S} \Pr[\mathcal{E}_i]$. The events are **pairwise independent** if this holds for all S with $|S| = 2$.

Proposition 7. *The **union bound** is as follows: for any events $\mathcal{E}_1, \ldots, \mathcal{E}_n$, $\Pr[\bigcup_i \mathcal{E}_i] \leq \sum_i \Pr[\mathcal{E}_i]$. This bound is tight when the $\mathcal{E}_i$ are mutually exclusive.*

1.2 Definitions of Randomized Algorithms

Definition 8. A **randomized algorithm** is an algorithm which gets to generate randomness to make decisions.

Randomized algorithms are often faster or simpler than their deterministic counterparts. Note that the input is still deterministic/worst case; in the contrasting probabalistic analysis, the input is randomized and the algorithm is deterministic.

Definition 9. A **Monte Carlo algorithm** is a randomized algorithm with guaranteed runtime and random correctness.

Definition 10. A **Las Vegas** algorithm is a randomized algorithm with guaranteed correctness and random runtime.

Definition 11. **RP** (randomized polynomial time) is the complexity class containing problems for which there exists a randomized polytime algorithm A such that:

1. If $x \notin X$, A always returns "No".
2. If $x \in X$, A returns "Yes" with probability $p \geq \frac{1}{2}$.

It is clear that $\mathbf{P} \subseteq \mathbf{RP} \subseteq \mathbf{NP}$, but it is an open question whether $\mathbf{P} = \mathbf{RP}$ or $\mathbf{RP} = \mathbf{NP}$.

2 Randomized Approximation Algorithm for 3SAT

2.1 The Airplane Seat Problem

Given n people and n assigned seats on an airplane, generate a uniformly random matching between people and seats. Let X be the random variable representing the number of people who randomly select their assigned seat. What is $\mathbb{E}[X]$?

Define indicator random variables:

$$X_i = \begin{cases} 1 & \text{if person } i \text{ got their seat} \\ 0 & \text{o.w.} \end{cases}$$

Then:

$$X = \sum_i X_i$$

So by expectation:

$$\begin{aligned} \mathbb{E}[X] &= \sum_i \mathbb{E}[X_i] \\ &= \sum_i \Pr[i] \\ &= \sum_i \frac{1}{n} \\ &= 1 \end{aligned}$$

2.2 Johnson's Algorithm

Find a 3SAT assignment approximately maximizing the number of satisfied clauses. We can't do the opposite (approximately minimize the amount of violated clauses) because it is **NP**-hard to distinguish between 0 and 1 violated clauses; if we have 0, we have solved an **NP**-hard problem.

Let Y be the random variable representing the number of satisfied clauses. Define indicator random variables:

Algorithm 1 Johnson's Algorithm

for all variables X_i in the 3SAT formula **do**
    Let $X_i = \begin{cases} \text{true} & \text{with probability } \frac{1}{2} \\ \text{false} & \text{o.w.} \end{cases}$
end for

$$Y_c = \begin{cases} 1 & \text{if } c \text{ is true} \\ 0 & \text{o.w.} \end{cases}$$

Then:

$$Y = \sum_c Y_c$$

So by expectation:

$$\mathbb{E}[Y] = \sum_c \mathbb{E}[Y_c]$$

$$= \sum_c \Pr[c]$$

$$= \sum_c (1 - \Pr[\neg c])$$

By independence:

$$= \sum_c (1 - \frac{1}{2}^3)$$

$$= \sum_c \frac{7}{8}$$

$$= \frac{7}{8} m$$

We know that OPT satisfies at most m clauses, so this is a $\frac{7}{8}$-approximation in expectation. Interestingly, this also proves that for every 3SAT formula, there exists an assignment satisfying $\frac{7}{8}$ of all clauses by the **probabilistic method**.

Definition 12. The probabilistic method is a way to prove the existence of an object with some property by sampling it randomly from a carefully designed distribution, then show that the randomly drawn object has a probability $p \geq 0$ of having the desired property (i.e., expander graphs, 3SAT formulae).

Theorem 13. *(Hastad): Unless **P**=**NP**, there is no polytime $\frac{7}{8} + \mathcal{E}$-approximation for 3SAT for any $\mathcal{E} \geq 0$ by the PCP theorem.*

3 Randomized Linear Time Algorithm for Median Finding

3.1 The Geometric Distribution

Imagine we flip a coin with probability p to land on heads until it comes up heads. Let X be the random variable representing the number of flips until we get a heads.

$$X \sim \mathcal{G}(p)$$

$$\mathbb{E}[X] = \frac{1}{p}$$

3.2 The Coupon Collector Problem

Given n items, each round we get a copy of a uniformly random item. How many rounds will it take until we have at least one copy of every item? Let X be the random variable representing the number of steps until we have at least one copy of each item. Define phase i to last from the first time we have $i-1$ distinct items until the first time we have i. Then, let X_i be the random variable representing the number of steps in phase i.

$$X = \sum_{i=1}^{n} X_i$$

$$\mathbb{E}[X] = \sum_{i=1}^{n} \mathbb{E}[X_i]$$

Note that:

$$X_i \sim \mathcal{G}(\frac{n-i+1}{n})$$

$$\mathbb{E}[X_i] = \frac{n}{n-i+1}$$

So:

$$\mathbb{E}[X] = \sum_{i=1}^{n} \frac{n}{n-i+1}$$

$$= n\sum_{i=1}^{n} \frac{1}{i}$$

$$= nH(n) \text{ where } H(n) \text{ is the } n^{th} \text{ harmonic number}$$

$$= \Theta(n \log n)$$

Algorithm 2 Median Finding Algorithm

if $|S| = 1$ **then**
  Return S_0
end if
Choose a pivot p uniformly at random.
Let $S^- = \{i \in S | i < p\}$
* Let $S^+ = \{i \in S | i \geq p\}$
if $k = 1 + |S^-|$
* Return S_0^+
elseif $k < 1 + |S^-|$
* Return (S^-, k)
else
* Return (S^+, k)
endif

3.3 Median Finding

Given a set S of n unsorted numbers, find the number in position $\lfloor \frac{n+1}{2} \rfloor$ or $\lceil \frac{n+1}{2} \rceil$ in the sorted order. Sorting the entire list gives $\mathcal{O}(n \log n)$, but we want $\mathcal{O}(n)$. The initial algorithm idea is similar to Quicksort: we choose a pivot, divide the set into S^- and S^+, and recurse in the correct partition. We can generalize this problem to select the element (S, k) which is the k^{th} smallest element of S.

We are interested in the expectation of the runtime of this algorithm $\mathbb{E}[T(n)]$. Note that $T(n) = \Theta(n) + T(n')$ where $n' = |S^-|$ or $|S^+|$, whichever we recurse on.

With probability $\frac{1}{2}$, p is in the middle half of S (call this a good pivot). Thus, each partition will be at most $\frac{3}{4}$ the size of S. In other words, $n' \leq \frac{3}{4}n$ with probability $\frac{1}{2}$. As a consequence of this fact, we know that our algorithm will terminate after $\log_{\frac{4}{3}} n$ rounds of good pivots.

Let X be the random variable representing the total number of steps of the algorithm. Define phase i to last from the first time $|S| \leq \frac{n}{\frac{4}{3}^{i-1}}$ until the first time $|S| \leq \frac{n}{\frac{4}{3}^{i}}$. Then, let X_i be the random variable representing the number of steps in phase i.

Combining all facts above, we have:

$$X \leq \sum_{i=1}^{\log_{\frac{4}{3}} n} \Theta(n * \frac{3}{4}^{i-1}) X_i$$

By expectation:

$$\mathbb{E}[X] = \sum_{i=1}^{\log_{\frac{4}{3}} n} \Theta(n * \frac{3}{4}^{i-1}) \mathbb{E}[X_i]$$

Note that:

$$X_i \leq \mathcal{G}(\frac{1}{2})$$

$$\mathbb{E}[X_i] \leq 2$$

So:

$$\mathbb{E}[X] \leq 2\Theta(n) \sum_{i=1}^{\infty} \frac{3}{4}^{i-1}$$

$$= 8\Theta(n)$$

$$= \Theta(n)$$