# How to solve: Probabilistic methods in computer science

Patrik Žnidaršič

Compiled December 14, 2025

## 1 Probability

The following probability distributions could prove useful:

| Name | PMF | Expected value | Variance |
|---|---|---|---|
| Geom$(p)$ | $(1-p)^{k-1}p$ | $p^{-1}$ | $\frac{1-p}{p^2}$ |
| Bin$(n,p)$ | $\binom{n}{k}p^k(1-p)^{n-k}$ | $np$ | $np(1-p)$ |
| Po$(\lambda)$ | $\frac{\lambda^k e^{-\lambda}}{k!}$ | $\lambda$ | $\lambda$ |

We know some basic tricks for computing with the expected value:

- Definition:
$$E(f(X)) = \sum_k f(k)P(X=k).$$
- Linearity: $E(aX+bY) = aE(X)+bE(Y)$ for any $a,b \in \mathbb{R}$ and random variables $X, Y$.
- $E(X \cdot Y) = E(X)E(Y)$ *if* $X, Y$ *are independent.*
- Markov's inequality: if $X \geq 0$, then for all $a > 0$,
$$P(X \geq a) \leq \frac{E(X)}{a}.$$

Additionally, remember the union bound for probability:

$$P\left(\bigcup_i D_i\right) \leq \sum_i P(D_i).$$

Equality holds if the events $D_i$ are pairwise disjoint.

The HARMONIC NUMBERS $H_n$ are defined as

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

We have $H_n \leq 1 + \log n$ for all $n$.

### 1.1 Confusion

Sometimes, the source of probability is questionable. Consider the following example: we're given a trinary tree with each leaf assigned a (deterministic, but unknown) 0/1 value. The other nodes of the tree are assigned their value by a majority vote of their children. We wish to determine the value of the root node in sublinear time with a randomized algorithm.

The solution is that for each node, we check two of its children randomly. If we're lucky, they'll have the same value, so we won't need to compute the third child. There is nothing random about the tree; it is only the steps that the algorithm takes which are randomized. We therefore cannot compute directly the probability that we select two children with an equal value (that depends on the values of the children), but we can bound it. If we consider the cases for what the children could be (000, 001, 011, 111 up to permutation), then we see that the probability in question is at most 1/3; from this, we can compute a bound for the expected value of visited leaves.

## 2 Complexity classes

We can split randomized algorithms into two types: Las Vegas algorithms always return the correct answer, but its running time is a random variable. A Monte Carlo algorithm on the other hand can return an incorrect answer. Monte Carlo algorithms for a decision problem $\pi$ can be further split into different types:

- One-sided error:
  - Type 1: If $\omega \in \pi$, then we return "yes" with probability $\geq \frac{3}{4}$. If $\omega \notin \pi$, then return "no" with probability 1.
  - Type 2: The opposite.
- Two-sided error: Type 3: If $\omega \in \pi$, then return "yes" with probability $\geq \frac{3}{4}$. If $\omega \notin \pi$, then return "no" with probability $\geq \frac{3}{4}$.

Note that $\frac{3}{4}$ is an arbitrary constant; for types 1 and 2, anything above 0 would work, and for type 3, anything above $\frac{1}{2}$. Based on this classification, we define the following complexity classes:

- RP: decision problems for which there exists a Monte Carlo algorithm of type 1 with polynomial expected time.
- co-RP: decision problems for which there exists a Monte Carlo algorithm of type 2 with polynomial expected time.

- BPP: decision problems for which there exists a Monte Carlo algorithm of type 3 with polynomial expected time.
- ZPP: decision problems for which there exists a Las Vegas algorithm with polynomial expected time.

We've shown that $\pi \in \text{ZPP}$ if and only if there exists a randomized algorithm $M$ for $\pi$ which finishes in polynomial time, and which either returns the correct answer or, with probability at most $\frac{1}{2}$, "Don't know." Additionally, we have $\text{ZPP} = \text{RP} \cap \text{co-RP}$.

Define $\text{BPP}(p)$ as BPP, except that we require the error probability to be at most $1-p$, so that $\text{BPP} = \text{BPP}(\frac{3}{4})$. Then it holds that $\text{BPP} = \text{BPP}(p)$ for any $p \in (\frac{1}{2}, 1)$.

## 3 Bounding probabilities

### 3.1 General bounds

Remember Markov's inequality; if $X \geq 0$ and $a > 0$, then

$$P(X \geq a) \leq \frac{E(X)}{a}.$$

Taking $a = 1$ gives us the FIRST MOMENT METHOD

$$P(X > 0) \leq E(X).$$

A similar bound is the SECOND MOMENT METHOD

$$P(X = 0) \leq \frac{\text{var}(X)}{E(X)^2}.$$

Note that the last two bounds only work if $X$ is integer-valued (and positive).

We can also use Chebyshev's bound: If $X$ has a finite variance $\sigma^2$ and expected value $\mu$, then

$$P(|X - \mu| \geq a\sigma) \leq \frac{1}{a^2}$$

for any $a > 0$. Or alternatively:

$$P(|X - \mu| \geq a) \leq \frac{\sigma^2}{a^2}.$$

### 3.2 Chernoff bounds

**Theorem 3.1** (Chernoff). *Let $X_1, X_2, \ldots, X_n$ be mutually independent random variables taking values 0 and 1. Let $p_i$ be the probability that $X_i = 1$. Define $X = \sum_i X_i$ and $\mu = E(X) = p_1 + \cdots + p_n$. For each $\delta \in (0,1)$ we have*

- $P(X - \mu \geq \delta\mu) \leq \exp(-\mu\delta^2/3)$,

- $P(\mu - X \geq \delta\mu) \leq \exp(-\mu\delta^2/3)$,
- $P(|X - \mu| \geq \delta\mu) \leq 2\exp(-\mu\delta^2/3)$.

The theorem can be used to bound probabilities of the form $P(X \gtrless C)$ for some $C$. First, write $X$ as a sum of mutually independent Bernoulli random variables (they are usually the same, but they don't have to be) and compute $E(X)$. Then, take $\delta = C/E(X) - 1$ and plug it into the theorem. Depending on which bound you want, you might need to use either the first or second assertion.

## 3.3 $(\varepsilon, \delta)$-approximations

We say that a randomized algorithm is an $(\varepsilon, \delta)$-APPROXIMATION for $V$ if the for the output $X$ of the algorithm, the following holds:

$$P(|X - V| \leq \varepsilon V) \geq 1 - \delta.$$

Using Chernoff's bounds, we've shown that if $Y = \frac{1}{m}\sum_{i=1}^{m} Y_i$ is a sum of mutually independent and identically distributed Bernoulli random variables with $E(Y_i) = \mu$, then $P(|Y - \mu| < \varepsilon\mu) \geq 1 - \delta$ if

$$m \geq \frac{3\log(2/\delta)}{\varepsilon^2 \mu}.$$

## 3.4 Polynomials

Sometimes, we can describe a randomized algorithm as if it is either implicitly or explicitly evaluating a certain polynomial at random values. The following theorem is helpful in the analysis of such algorithms.

**Theorem 3.2** (Schwarz, Zippel)**.** *Let $p(x_1, \ldots, x_n)$ be a polynomial in $\mathbb{F}[x_1, \ldots, x_n]$ of degree $d > 0$. Let $S \subseteq \mathbb{F}$ be a finite set. If $r_1, \ldots, r_n \in S$ are selected uniformly at random, then*

$$P(p(r_1, \ldots, r_n) = 0) \leq \frac{d}{|S|}.$$

Usually, it is easy to find a polynomial to check if two things – sets of numbers, string, etc. are equal. We somehow set these things to be roots of two polynomials, then evaluate them many times in random points. If all evaluations are the same, report that the things are equal; otherwise, they are different. We can then use the theorem to compute the probability that we report incorrectly, or determine how many times we need to repeat the experiment to get the probability below a certain bound. Useful polynomials are products of monomials, sums of $a_i x^i$, or determinants of matrices.

## 4 Random graphs

We've defined a random graph $G(n, p)$ as a graph on vertices $\{1, 2, \ldots, n\}$ for which each pair $\{i, j\}$ of numbers is an edge with probability $p$. For a fixed $p$, these graphs tend to gain connectivity properties as $n \to \infty$. For example, taking an arbitrary $m$, the graph will be $m$-connected with high probability for any large enough $n$.

We say that a random graph has property $\mathcal{P}$ ALMOST SURELY if the probability that $G_{n,p}$ has this property converges to 1 as $n \to \infty$. Here, $p$ can depend on $n$ if we so choose.

## 5 Markov chains

**Definition 5.1.** A MARKOV CHAIN is an infinite sequence $X_0, X_1, X_2, \ldots$ of random variables with values in a finite set $\Omega$ such that for all $t$,

$$P(X_{t+1} = i \mid X_t = i_t, \ldots, X_0 = i_0) = P(X_{t+1} = i \mid X_t = i_t)$$

and for any $t, t'$, $P(X_{t+1} = i \mid X_t = j) = P(X_{t'+1} = i \mid X_{t'} = j)$.

The transition from $X_t$ to $X_{t+1}$ is given by a transition matrix $P = [p_{ij}]_{ij}$, where $p_{ij} = P(X_{t+1} = j \mid X_t = i)$. We say that a row-vector (distribution) $\pi$ is stationary if $\pi = \pi P$.

Define the hitting time $h_{ij}$ for $i \to j$ as the expected number of steps required to reach state $j$, starting from state $i$.

With $G_P$, we denote the directed graph representing $P$, i.e. the graph on $\Omega$ with edges $i \to j$ whenever $p_{ij} > 0$. We say that the Markov chain is STRONGLY CONNECTED if $G_P$ is strongly connected. The chain is APERIODIC if the greatest common divisor of the length of all closed walks in $G_P$ is 1.

**Theorem 5.2.** *Consider a strongly connected Markov chain. Then*

- *there exists a unique stationary distribution $\pi = (\pi_i)_{i \in \Omega}$,*
- *for every $i \in \Omega$, $h_{ii} = \frac{1}{\pi_i}$,*
- *for all $i \in \Omega$,*
$$\lim_{t \to \infty} \frac{N(i, t, q_0)}{t} = \pi_i,$$
*where $N(i, t, q_0)$ is the number of visits to $i$ in $t$ steps if $X_0$ is selected according to distribution $q_0$,*
- *If $G_P$ is aperiodic, then $\lim q_0 P^t = \pi$ for any distribution $q_0$.*

# 6 Examples

Some of the examples of randomized algorithms covered in lectures are listed here, along with their properties

## 6.1 Minimum cut

Remember that a cut $\delta(U)$ is the set of edges from $U$ to $V \setminus U$. If we're looking for a minimum cut, we're interested in $|\delta(U)|$, not $|U|$.

**Proposition 6.1.** *Algorithm 1 returns a minimum cut with probability at least* $2/n(n+1)$.

**Algorithm 1** RandMinCut

```
$G_0 = G$ and $i = 0$.
while $|V(G_i)| > 2$ do
    Let $e_i$ be a random edge of $G_i$.
    Contract edge $e_i$ to get $G_{i+1} = G_i/e_i$, preserving parallel edges.
    $i = i + 1$.
end while
Let $u, v$ be the two vertices of $G_i$.
Define $U$ as the set of all vertices merged to $u$.
Return $\delta(U)$.
```

## 6.2 Area estimate

Let $B_1, \ldots, B_m$ be some rectangles in the plane. We wish to estimate the area of their union with an $(\varepsilon, \delta)$-approximation. To do this, define

$$R_i = B_i \setminus \bigcup_{j<i} B_j.$$

Then the area of the union of $B_i$ is equal to the sum of the areas of $R_i$, so we only need to compute those. We will sample from $\bigcup_i R_i$. First, select a random index $a$ such that

$$P(a = i) = \frac{|B_i|}{\sum_j |B_j|}.$$

Then take a random point in $B_i$ (uniformly) and check if it is in $R_i$. Count the number of hits of $R_i$, then estimate

$$\left| \bigcup_i B_i \right| \approx \frac{\text{number of hits}}{\text{number of tries}} \sum_i |B_i| .$$

To get an $(\varepsilon, \delta)$-approximation, we need

$$\text{number of tries} \geq \frac{3 \log(2/\delta)}{\varepsilon^2} \sum_i |B_i| .$$

This algorithm can be easily adapted to compute the hypervolume of a union of hyper-rectangles.

**Algorithm 2** Area estimate

```
Let
    $$k = \frac{3\log(2/\delta)}{\varepsilon^2} \sum_i |B_i| .$$
Let $y = 0$
for $i = 1, \ldots, k$ do
    Choose $a \in \{1, \ldots, n\}$ with
        $$P(a = i) = \frac{|B_i|}{\sum_j |B_j|}$$
    Choose $p \in B_a$ uniformly at random
    if $p \notin B_1 \cup \ldots \cup B_{a-1}$ then
        Increment $y$
    end if
end for
Return $\frac{y}{k} \sum_i |B_i|$
```

## 6.3 DNF counting

A formula $\tilde{F}$ in CNF is not satisfiable if and only if its negation $\neg\tilde{F}$ (in DNF) has $2^n$ solutions. We have an $(\varepsilon, \delta)$-approximation for this number of solutions.

Let $F \equiv C_1 \vee \ldots \vee C_k$, where $C_i \equiv l^i_1 \wedge \ldots \wedge l^i_{k_i}$ and $l^i_j$ are literals. Define $\phi(C_i)$ to be the set of $n$-tuples $(r_1, \ldots, r_n) \in 2^n$ for which $C_i(r_1, \ldots, r_n) \equiv \top$. Then take $\tilde{\phi}(C_i) = \{i\} \times \phi(C_i)$. Algorithm 3 gives an $(\varepsilon, \delta)$-approximation of the number of solutions for $F$.

**Algorithm 3** DNF counting

```
Let $x = 0$
Set
    $$m = \left\lceil \frac{3k}{\varepsilon^2} \log(2/\delta) \right\rceil$$
for $i = 1, \ldots, m$ do
    Choose $(i, a) \in \bigcup_{j=1}^{k} \tilde{\phi}(C_j)$ uniformly at random
    if $(i, a) \notin \tilde{\phi}(C_1) \cup \ldots \cup \tilde{\phi}(C_{i-1})$ then
        Increment $x$
    end if
end for
Return $\frac{x}{m} \sum_i |\phi(C_i)|$
```

### 6.4 Rand 2-SAT

Consider the 2-SAT problem, where we have boolean variables $x_1, \dots, x_n$ and clauses $C_1, \dots, C_m$ of the form $C_i \equiv l_1^i \vee l_2^i$, where $l_j^i$ are literals. We wish to find an assignment for the variables $x_i$ such that all clauses will be satisfied. The problem can actually be solved deterministically in polynomial time, but we will still consider a randomized algorithm.

Taking $k = 8n^2$, algorithm **??** will return the correct answer (whether the problem is satisfiable or not) with probability at least $\frac{3}{4}$. In analyzing the algorithm, we define a Markov chain in the following way: Let $B^*$ be some satisfying assignment. Then an assignment $B$ is in state $V_j$ if exactly $j$ of its values are equal to those values in $B^*$.

With some simple but not straightforward though, we can see that in each step of the algorithm, the probability that we change to a higher state is at least $\frac{1}{2}$, as in $B^*$, there are precisely three possibilities for what the two variables in question could have assigned. We can then compute that the expected number of steps to reach $V_n$ is $\leq 2n^2$, so we can conclude with Markov's inequality.

**Algorithm 4** Rand 2-SAT

```
Choose arbitrary $B_0 = (b_1, \dots, b_n) \in 2^n$
Let $i = 0$
for $j = 1, \dots, k$ do
    if $B_i$ satisfies $C_1 \wedge \dots \wedge C_m$ then
        Return "yes"
    else
        Choose some $C_j$ that is false with $B_i$
        Choose a random variable $x_\alpha$ in $C_j$
        Copy $B_i$ to $B_{i+1}$, but switch the value of $x_\alpha$
    end if
    Increment $i$
end for
Return "no"
```