7 Randomization Basics

10 June 2025

Prof. Dr. Sebastian Wild

CS627 (Summer 2025)
Philipps-Universität Marburg
version `2025-06-16 23:24 H`

Outline

7 Randomization Basics

7.1 Motivation

Computational Lottery?

- If we are faced with solving an NP-hard problem and known smart algorithms are too slow, we likely have to compromise on what "solving" means.
- Classical algorithms are *always* and *exactly* correct.

$\rightsquigarrow$ Here: Let's compromise on "always", i. e., allow algorithms to occasionally **fail**!

ϟ A *deterministic* algorithm A that fails on input x will ***always*** fail for x.

  - $\rightsquigarrow$ What if we require a solution for such an input x? We get **nothing** from A!
  - Must use a form of *nondeterminism*.

- ***Randomization:*** Use *random bits* to guide computation.

$\rightsquigarrow$ *Instead of always failing on some rare inputs, we rarely fail on any input.*

can make this arbitrarily rare

Why Could Randomization Help?

- Main intuitive reason: (can be) much easier to be 99.999999% correct than 100%
  How can this manifest itself?
  - **Faster and simpler algorithms**
    Random choice can allow to sidestep tricky edge cases
  - We can use **fingerprinting** (a.k.a. checksums)
    Cheap surrogate question, mostly correct, but sometimes wrong.
  - Protect against **adversarial inputs**
    We make our (algorithm's) behavior unpredictable, so it us harder to exploit us.

- Also: *probabilistic method* for proofs
  - Goal: Prove existence of discrete object with some property
  - Idea: Design randomized algorithm to find one
  - $\rightsquigarrow$ If algorithm succeeds with prob. > 0, object must exist!

Average-Case Analysis vs. Randomized Algorithms

Average-Case Analysis

- algorithm is **deterministic**
  same input, same computation
- input is chosen according to some **probability distribution**
- cost given as expectation over inputs

Randomized Algorithm (here)

- algorithm is **not** deterministic
  same input, potentially different comp.
- input is chosen **adversarially** (worst-case inputs)
- cost given as expectation over random choices of algorithm

Confusingly enough, the analysis (technique) is often the same!

But: Implications are quite different; randomization is much more versatile and robust.

7.2 Randomized Selection

Separation Example

- Before we introduce randomization more formally, let's see a successful example
- Here, not a "hard" problem, but a showcase where randomization makes something possible that is *provably*

Introductory Example – Quickselect

Selection by Rank

but 0-based & counting dups

- **Given:** array $A[0..n)$ of numbers and number $k \in [0..n)$.
- **Goal:** find element that would be in position k if A was sorted (kth smallest element).
  - $k = \lfloor n/2 \rfloor \rightsquigarrow$ median; $k = \lfloor n/4 \rfloor \rightsquigarrow$ lower quartile
    $k = 0 \rightsquigarrow$ minimum; $k = n - \ell \rightsquigarrow$ ℓth largest

```
procedure quickselect(A[0..n), k):
    l := 0;  r := n
    while r − l > 1
        b := random pivot from A[l..r)
        j := partition(A[l..r), b)
        if j ≥ k then r := j − 1
        if j ≤ k then l := j + 1
    return A[k]
```

- simple algorithm:
  determine rank of random element,
  recurse

over random choices

$\rightsquigarrow$ $O(n)$ time in expectation

- worst case: $\Theta(n^2)$
- $O(n)$ also possible deterministically,
  but algorithms is more involved

median of medians

A closer look at Selection

While all within $\Theta(n)$, we do get a strict separation for selecting the median.

Theorem 7.1 (Bent & John (1985))

Any **deterministic** comparison-based algorithm for finding the median of n elements uses at least $\mathbf{2}n - o(n)$ comparisons in the worst case. ◀

Proof omitted.

The following weaker result is easier to see:

Theorem 7.2 (Blum et al. (1973))

Any deterministic comparison-based algorithm for finding the median of n elements uses at least $n - 1 + (n - 1)/2 \sim 1.5n$ comparisons in the worst case. ◀

A Median Adversary

Proof (Theorem 7.2):

Randomized Selection

- Can prove: Randomized Quickselect uses in expectation $\sim (2\ln 2 + 2)n \approx 3.39n$ comparisons to find the median
- But we can do better!

procedure floydRivest($A[\ell..r)$, k):
    $n := r - \ell$
    if $n < n_0$ **return** quickselect(A, k)
    $s := \frac{1}{2}n^{2/3}$ *// all numbers to be rounded*
    $sd := \frac{1}{2}\sqrt{\ln(n)s(n-s)/n}$
    $S[0..s)$:= random sample from A
    $\hat{k} := s\frac{k}{n}$
    $p :=$ floydRivest(S, $\hat{k} - sd$)
    $q :=$ floydRivest(S, $\hat{k} + sd$)
    (i, j) :=partition A around p_0 and p_1
    if $i == k$ **return** $A[i]$
    if $j == k$ **return** $A[j]$
    if $k < i$ **return** floydRivest($A[\ell..i)$, k)
    if $k > j$ **return** floydRivest($A[j..r)$, k)
    return floydRivest($A[i..j)$, k)

- Variant of Quickselect with huge sample
- Analysis sketch:
  - partition costs $1.5n$ comparisons
  - Everything on sample has cost $o(n)$
  - by the choice of parameters, with prob $1 - o(1)$:
    (a) $i < k < j$ after partition
    (b) $j - i = o(n)$
  - ⇝ all recursive calls expected $o(n)$ cost

⇝ Randomized median selection with $1.5n + o(n)$ comparisons

⇝ Separation from deterministic case!

Power of Randomness

- Selection by Rank shows two aspects of randomization:
  - A simpler algorithm by avoiding edge cases (like an initial order giving bad pivots)
  - Protection against adversarial inputs
    (inputs constructed with knowledge about the algorithm)

    Here randomization provably more powerful than any thinkable deterministic algorithm!

    constant factor for #cmps

- What can we gain for (NP-)hard problems?
- But first, let's define things properly.

7.3 Recap of Probability Theory

Probability Theory

- We will quickly revisit some key terms from probability theory
  - Single place to look up notation etc.
- Much will focus on discrete probability, but some continuous tools useful, too

Probability Spaces

Discrete probability space $(\Omega, \mathbb{P})$:

- $\Omega = \{\omega_1, \omega_2, \ldots\}$ a (finite or) *countable* set
- $\mathbb{P} : 2^\Omega \to [0,1]$ a discrete probability measure, i. e.,
  - $\mathbb{P}[\Omega] = 1$
  - $\mathbb{P}[A] = \sum_{\omega \in A} \mathbb{P}[\omega] \quad \rightsquigarrow \quad \mathbb{P}$ determined by $w_i = \mathbb{P}[\omega_i]$.

General probability space $(\Omega, \mathcal{F}, \mathbb{P})$:

- Ω is a set of points (the universe)
- $\mathcal{F} \subseteq 2^\Omega$ is a σ-algebra, i. e., (discrete case: $\mathcal{F} = 2^\Omega$; $\Omega = \mathbb{R}$: Borel σ-algebra $\mathcal{B}$ generated by (a, b))
  - $\emptyset \in \mathcal{F}$
  - closed under complementation: $A \in \mathcal{F} \implies \overline{A} = \Omega \setminus A \in \mathcal{F}$
  - closed under *countable* union: $A_1, A_2, \ldots \in \mathcal{F} \implies \bigcup_{i=1}^{\infty} A_i \in \mathcal{F}$
- $\mathbb{P} : \mathcal{F} \to [0,1]$ is a probability measure, i. e., ($\Omega = \mathbb{R} \quad \rightsquigarrow \quad$ Lebesgue measure $\lambda((a, b)) = b - a$)
  - $\mathbb{P}[\Omega] = 1$
  - If $A_1, A_2, \ldots \in \mathcal{F}$ are pairwise *disjoint* then $\mathbb{P}\left[\bigcup_{i=1}^{\infty} A_i\right] = \sum_{i=1}^{\infty} \mathbb{P}[A_i]$

Events

something we can assign a probability to

$A \in \mathcal{F}$ is called an *event* of $(\Omega, \mathcal{F}, \mathbb{P})$; also a *measurable set*.

Basic properties

- $\mathbb{P}\left[\overline{A}\right] = 1 - \mathbb{P}[A]$ counter-probability $\quad (\overline{A} = \Omega \setminus A)$
- $\mathbb{P}\left[\bigcup A_i\right] \le \sum_i \mathbb{P}[A]$ the *union bound* (a.k.a. Boole's inequality a.k.a. σ-subadditivity)
- $\{A_1, \ldots, A_k\}$ *(mutually) independent* $\iff \mathbb{P}\left[\bigcap_i A_i\right] = \prod_i \mathbb{P}[A_i]$

  An infinite set of events is mutually independent if every finite subset is so.

  k-wise independence means that only all size-k subsets are independent.
- *conditional probability* for A given B: $\mathbb{P}[A \mid B] = \mathbb{P}[A \cap B] / \mathbb{P}[B]$
  generally undefined if $\mathbb{P}[B] = 0$
- *law of total probability*: If $\Omega = B_1 \dot{\cup} B_2 \dot{\cup} \cdots$ is a partition of Ω, we have

$$\mathbb{P}[A] = \sum_{\substack{i \\ \mathbb{P}[B_i] \neq 0}} \mathbb{P}[A \mid B_i] \cdot \mathbb{P}[B_i].$$

Random Variables

Random variables (r.v.) $X : \Omega \to \mathcal{X}$; often $\mathcal{X} = \mathbb{R}$ (in general spaces: only *measurable* functions)

Basic properties and conventions:

- event $\{X = x\}$ is defined as $\{\omega \in \Omega : X(\omega) = x\}$.
- For event A define the indicator r.v. $\mathbb{1}_A$ via $\mathbb{1}_A(\omega) = [\omega \in A]$
- $F_X(x) = \mathbb{P}[X \le x]$ is the *cumulative distribution function (CDF)*.
- X is *discrete* if $X(\Omega) = \{X(\omega) : \omega \in \Omega\}$ is countable.
- for discrete r.v. X define $f_X(n) = \mathbb{P}[X = n]$ the *probability mass function (PMF)*.
- If F_X is everywhere differentiable, X is *continuous*.
  Then $f_X = F'_X$ is its *probability density function*.

Equality in distribution:

- We write $X \overset{\mathcal{D}}{=} Y$ if $F_X = F_Y$

Independent Random Variables

Independence:

- Consider *vector* $\boldsymbol{X} = (X_1, \ldots, X_k)$ as single function from Ω to $\mathbb{R}^k$.
  CDF/PMF/PDF of $\boldsymbol{X}$ is called *joint CDF/PMF/PDF* of $X_1, \ldots, X_k$.
- r.v.s *independent* $\iff$ joint PMF/PDF *factors*:
  X and Y independent $\iff$ $\mathbb{P}[X = x \wedge Y = y] = \mathbb{P}[X = x] \cdot \mathbb{P}[Y = y]$ for all x, y.

  (Naturally follows from independent events)

i.i.d. sequences

- We often talk about sequences of random variables $X_1, X_2, \ldots$
- a sequence of *i.i.d.* r.v. $X_1, X_2, \ldots$ (*independent and identically distributed*)
  has $X_i \overset{\mathcal{D}}{=} X_1$ and $\{X_i\}_{i \geq 1}$ are mutually independent
  - typical example: sequence of coin tosses (with same coin)

Expected Values

Expectation of an $\mathcal{X}$-valued r.v. X, written $\mathbb{E}[X]$, is given by

- $\mathbb{E}[X] = \sum_{x \in \mathcal{X}} x \cdot f_X(x)$ for *discrete* X with PMF f_X,
- $\mathbb{E}[X] = \int_{x \in \mathcal{X}} x \cdot f_X(x)\, dx$ for *continuous* X with PDF f_X.
- undefined if sum does not converge / integral does not exist.

Properties:

- *linearity*: $\mathbb{E}[aX + bY] = a\mathbb{E}[X] + b\mathbb{E}[Y]$ (X, Y r.v. and a, b constants)
  even if X and Y are not independent
  only for *finite* sums / linear combinations!
- X and Y *independent* $\implies \mathbb{E}[X \cdot Y] = \mathbb{E}[X] \cdot \mathbb{E}[Y]$.

Conditional Expectation

Similar to conditional *probability*, we can define conditional *expectations*.

- *conditional expectation* on event $\mathbb{E}[X \mid A] \;=\; \sum_x \mathbb{P}[X = x \mid A]$ for *discrete* X.
  for general A, continuous definition problematic

- *conditional expectation* on $\{Y = y\}$, written $\mathbb{E}[X \mid Y = y]$.
  - for *discrete* X and Y
  $$\mathbb{E}[X \mid Y = y] \;=\; \sum_{x \in \mathcal{X}} x \cdot \mathbb{P}\big[X = x \,\big|\, \{Y = y\}\big]$$
  - for *continuous* X and Y, use the joint density $f_{(X,Y)}$ and define the *marginal density* of Y as $f_Y(y) = \int_{x \in \mathcal{X}} f(x, y)\, dx$. Then
  $$\mathbb{E}[X \mid Y = y] \;=\; \int_{\mathcal{X}} x \cdot f_{X|Y}(x, y)\, dx \qquad \text{with} \qquad f_{X|Y}(x, y) = \frac{f_{(X,Y)}(x, y)}{f_Y(y)}$$

- With $g(y) := \mathbb{E}[X \mid Y = y]$ we obtain a *new r.v.* $\mathbb{E}[X \mid Y] = g(Y)$.

- *law of total expectation*: $\mathbb{E}[X] \;=\; \mathbb{E}_Y\big[\mathbb{E}_X[X \mid Y]\big]$.

Famous Distributions

discrete

- *Bernoulli r.v.* $X \overset{\mathcal{D}}{=} \mathrm{B}(p) \quad \rightsquigarrow \quad \mathbb{P}[X = 1] = p, \ \mathbb{P}[X = 0] = 1 - p$
- *Binomial r.v.* $Y \overset{\mathcal{D}}{=} \mathrm{Bin}(n, p) \quad \rightsquigarrow \quad Y = X_1 + \cdots + X_n$ for $X_1, \ldots, X_n$ i.i.d. $X_i \overset{\mathcal{D}}{=} \mathrm{B}(p)$
- discrete *uniform r.v.* $X \overset{\mathcal{D}}{=} \mathcal{U}([0..n)) \quad \rightsquigarrow \quad \mathbb{P}[X = i] = \frac{1}{n}$ for $i \in [0..n)$ (else 0)
- *Geometric r.v.* $X \overset{\mathcal{D}}{=} \mathrm{Geo}(p) \quad \rightsquigarrow \quad \mathbb{P}[X = k] = (1 - p)^{k-1} p$ for $k \in \mathbb{N}_{\geq 1}$

continuous

- continuous uniform $X \overset{\mathcal{D}}{=} \mathcal{U}([0, 1)) \quad \rightsquigarrow \quad f_X(x) = 1$ for $x \in [0, 1)$ (else 0)

(of course there are many more)

7.4 Probabilistic Turing Machines

Model of Computation

Definition 7.3 (Probabilistic Turing Machine)

A *probabilistic Turing Machine* (PTM) $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, q_{\mathrm{halt}})$ is a deterministic TM with an additional read-only tape, filled with random bits.
The *transition function* δ takes as input

- the current state q
- the current tape symbol a
- the current *random-tape symbol* $r \in \{0, 1\}$

and outputs

- the next state q'
- the new tape symbol b
- the tape-head movement $d \in \{L, R, N\}$
- the *random-tape head movement* $d_r \in \{L, R, N\}$ ◀

Intended semantics: random tape filled with i.i.d. $\mathrm{B}(\frac{1}{2})$ r.v.

Randomized Computation

- *Configuration* of PTM: $(\alpha q \beta, \rho q \sigma)$
  $\alpha q \beta$ normal TM config
  $\rho\sigma$ content of random tape, with head on first bit of σ
- *computation relation* $\vdash$ similar to TM
  content of random tape unchanged, heads can move independently
- *function computed* by PTM M:
  for input x and **fixed random bits** ρ, computation is deterministic:
  $M(x, \rho) \;=\; y \;$ if $\;(q_0 x, q_0 \rho) \;\vdash^\star\; (q_{\text{halt}} y, \rho' q_{\text{halt}} \rho'')$

⇝ ***Randomized computation of PTM:***
random variable $M(x, B_0 B_1 B_2 \ldots)$ where $B_0, B_1, B_2, \ldots$ are i.i.d. $\mathrm{B}(\frac{1}{2})$ distributed

⇝ Write $\mathbb{P}[M(x) = y] \;=\; \sum_b \mathbb{P}[B_0 B_1 \ldots = b] \cdot [M(x, b) = y]$

- Hope: PTM M so that correct output computed with high probability

Warmup: Rejection Sampling

We assume only random *bits*. How to simulate, say, a fair (6-sided) die?

```
procedure rollDie():
    do
        Draw 3 random bits b_2, b_1, b_0
        // Interpret as binary representation of a number in [0..7]
        n = ∑_{i=0}^{2} 2^i b_i
    while (n = 0 ∨ n = 7)
    return n
```

Correctness: Every output $1, \ldots, 6$ equally likely by construction.

Termination: *Infinite* runs possible!

***Expected* Running Time:** Leave loop with probability $\frac{6}{8} = \frac{3}{4}$ in each iteration

$\rightsquigarrow$ in expectation, only $\dfrac{4}{3} = \displaystyle\sum_{i \geq 1} i \cdot \left(\frac{1}{4}\right)^{i-1} \frac{3}{4}$ repetitions.

rollDie is a correct and practically efficient algorithm.

What can go wrong?

What can go wrong in a randomized computation?

- Computation could run into a deterministic infinite loop (as for deterministic TM)
  - ϟ don't ever terminate, no output
  - $\rightsquigarrow$ Clearly don't want that (just as before)

    (annoyingly undecidable to check ... also just as before)

- Computation could repeatedly have branches that keep looping (as for rollDie)
  - $\rightsquigarrow$ For every t, there is a probability $p > 0$ to run for more than t time steps
  - This is a new option that deterministic TMs didn't have

    ... but nondeterministic TMs did, and we just defined running time to be ∞ there!

    So, is that a problem? Or is it not??

Random Termination

Key question: *What is the probability space for the running time of the PTM simulating rollDie?*

- Note: this could indeed be a problem.
  - $\{0,1\}^\star$ (the set of **finite** bitstrings) is countably infinite (=discrete)
  - But the set of ***infinite strings*** (ω-*language*) is not!
    $\{0,1\}^\omega \;=\; \big\{b_0 b_1 \ldots : b_i \in \{0,1\}\big\} \;=\; \big\{b \;:\; b : \mathbb{N}_0 \to \{0,1\}\big\}$ surjectively maps to $[0,1) \subset \mathbb{R}$
    $b \mapsto 0.b_0 b_1 b_2 \ldots$
- Config $(\alpha q \beta, \rho q \boldsymbol{\sigma})$ for PTM needs $\sigma \in \{0,1\}^{\boldsymbol{\omega}}$ in general

- Define the random variable $Time_M(x) \in \mathbb{N}_0 \cup \{\infty\}$ on the *Bernoulli probability space*
  - generators: $\big\{\pi_x : x \in \{0,1\}^\star\big\}$ where $\pi_x = \big\{xw : w \in \{0,1\}^\omega\big\} \subseteq \{0,1\}^\omega$
  - Bernoulli σ-algebra: smallest $\mathcal{F}$ containing all $\{\pi_x\}_x$ that is closed under countable union and complement
  - $\mathbb{P}[\pi_x] = 2^{-|x|}$

$\leadsto$ expectations over $\rho \in \{0,1\}^\omega$, the infinite initial random-bit tape input are well-defined

(Expected) Time

Definition 7.4 (PTM running time)

For a PTM M, we define $time_M(x)$ as for nondeterministic TMs as the supremum of time steps over all computations.
Moreover, we define the *expected time* as

$$\mathbb{E}\text{-}time_M(x) \;=\; \mathbb{E}[time_M(x)] \;=\; \mathbb{E}_\rho\left[\inf\{t \in \mathbb{N}_0 : (q_0 x, q_0 \rho) \vdash^t (q_{\text{halt}} y, \rho' q_{\text{halt}} \rho'')\right]$$

Similarly

$$\mathbb{E}\text{-}Time_M(n) \;=\; \sup\left\{\mathbb{E}\text{-}time_M(x) : x \in \Sigma^n\right\}$$

◀

- We can of course also study full distribution of $time_M(x)$
- Useful property of expected time:
  $\mathbb{E}\text{-}time_M(x) < \infty \quad \text{iff} \quad \mathbb{P}[time_M(x) = \infty] = 0$

A New Complexity Measure: Random Bits

Definition 7.5 (Random-bit complexity)

For a PTM M computing with input alphabet Σ, the *random-bit cost* for an input $x \in \Sigma^\star$ is denote by

$$random_M(x) \quad = \quad \sup\left\{|\rho'| : (xq_0, q_0\rho) \vdash^\star (\alpha q\beta, \rho' q\rho'') \vdash^\star (q_{\text{halt}}y, \rho' q_{\text{halt}}\rho'')\right\}$$

and similarly

$$Random_M(n) \quad = \quad \sup\left\{random_M(x) : x \in \Sigma^n\right\}.$$

Further, the *expected random-bit cost* are defined as
$\mathbb{E}\text{-}random_M(x) = \mathbb{E}_\rho[random_M(x)]$ and
$\mathbb{E}\text{-}Random_M(n) = \sup\left\{\mathbb{E}\text{-}random_M(x) : x \in \Sigma^n\right\}$ ◀

Randomization vs. Nondeterminism

- Superficially similar concepts
- Key difference: meaning of number of computations of TM
  - nondeterministic TM: accept if **some (single)** accepting computation is possible
  - randomized TM: accept if **most** possible computations are accepting

$\rightsquigarrow$ nondeterminism = purely theoretical construction (overly powerful yardstick)

- randomization = widely applied efficient design technique

7.5 Classification of Randomized Algorithms

Las Vegas

Consider here the general problem to compute some *function* $f : \Sigma^\star \to \Gamma^\star$.

$\rightsquigarrow$ Covers *decision problems* $L \subseteq \Sigma^\star$ by setting $\Gamma = \{0, 1\}$ and $f(w) = \begin{cases} 1 & w \in L \\ 0 & w \notin L \end{cases}$

Definition 7.6 (Las Vegas Algorithm)

A randomized algorithm A is a *Las-Vegas (LV) algorithm* for a problem $f : \Sigma^\star \to \Gamma^\star$ if for all $x \in \Sigma^\star$ holds

(a) $\mathbb{P}\left[time_A(x) < \infty\right] = 1$ (*terminate* almost surely)

(b) $A(x) \in \{f(x), \boxed{?}\}$ (answer always *correct or "don't know"*)

(c) $\mathbb{P}\left[A(x) = f(x)\right] \geq \frac{1}{2}$ (*correct half the time*)

Don't Know vs. Won't Terminate

Theorem 7.7 (Don't know don't needed)

Every Las Vegas algorithm A for $f : \Sigma^\star \to \Gamma^\star$ can be transformed into a randomized algorithm B for f so that for all $x \in \Sigma^\star$ holds

(a) $\mathbb{P}\big[B(x) = f(x)\big] \;=\; 1$ (always correct)

(b) $\mathbb{E}\text{-}time_B(x) \;\leq\; 2 \cdot time_A(x)$ ◀

Proof:
See exercises. ■

Theorem 7.8 (Termination Enforcible)

Every randomized algorithm B for $f : \Sigma^\star \to \Gamma^\star$ with $\mathbb{P}[B(x) = f(x)] = 1$ can be transformed into a Las Vegas algorithm A for f so that for all $x \in \Sigma^\star$ holds

(a) $\mathbb{P}\big[A(x) = f(x)\big] \;\geq\; \frac{1}{2}$

(b) $time_A(x) \;\leq\; 2 \cdot \mathbb{E}\text{-}time_B(x)$ (always terminates). ◀

Proof:
See exercises. ■

Las Vegas Variants

$\rightsquigarrow$ *Can trade expected time bound for worst-case bound by allowing "don't know" and* ***vice versa****!*

Both types are called commonly LV algorithms; where helpful, we distinguish:

(A) *Always-Decisive Las Vegas algorithms* (output of Theorem **??**)

(B) *Always-Terminating Las Vegas algorithms* (output of Theorem **??**)

Las Vegas Examples

rollDie by rejection sampling is Las Vegas of unbounded worst-case type.

Easy to transform into Las Vegas according to Definition 7.6:

```
procedure rollDieLasVegas:
    Draw 3 random bits b_2, b_1, b_0
    n = Σ_{i=0}^{2} 2^i b_i   // Interpret as binary representation of a number in [0 : 7]
    if (n = 0 ∨ n = 7)
        return ?
    else
        return n
```

Other famous examples: *(randomized) Quicksort* and *Quickselect*

- always correct *and*
- $time(n) = O(n^2) < \infty$
- much better average:
  - $\mathbb{E}\text{-}time_{QSort}(n) = \Theta(n \log n)$
  - $\mathbb{E}\text{-}time_{QSelect}(n) = \Theta(n)$

To Err is Algorithmic

Sometimes sensible to allow *wrong / imprecise* answers . . .
but random should not mean *arbitrary*!

Definition 7.9 (Monte Carlo Algorithm)

A randomized algorithm A is a *Monte Carlo algorithm* for $f : \Sigma^\star \to \Gamma^\star$

- with *bounded error* if $\quad \exists \varepsilon > 0 \, \forall x \in \Sigma^\star \; : \; \mathbb{P}[A(x) = f(x)] \ge \frac{1}{2} + \varepsilon.$
- with ***un**bounded error* if $\quad \forall x \in \Sigma^\star \; : \; \mathbb{P}[A(x) = f(x)] > \frac{1}{2}.$

Seems like a minuscule difference? We will see it is vital!

7.6 Tail Inequalities and Concentration Bounds

How To Summarize A Random Variable?

- running time of randomized algorithm is a **random variable**
- For two randomized algorithms A and B, we'd like to decide which is better
  Whether A is faster than B is also random.

$\rightsquigarrow$ need a way to **compare** random variables!

- One option: *stochastic dominance*
  - If $\forall t : \mathbb{P}[time_A(x) \ge t] \ge \mathbb{P}[time_B(x) \ge t]$,
    we say $time_A(x)$ (weakly) stochastically dominates $time_B(x)$ (on input x)
  - Would rather use B then!

  dominance rarely true for real algorithms

  no prediction of running time / comparison with explicit bound

$\rightsquigarrow$ look at **expected value** $\mathbb{E}\text{-}time(x)$ (randomized version of average case)

  simple (one number); reflects typical case

  not always reliable / representative

When Expectation Isn't Enough

- Two hypothetical algorithms:
  - A takes 1 step in half the cases and 3 steps otherwise
  - B takes 1 step in 99% of cases and **101** steps otherwise
  - $\rightsquigarrow$ both have $\mathbb{E}\text{-}time(x) = 2$
  - probably want A ... certainly would want to be able to distinguish them!

- **Goal:** Strengthen algorithms so $time(x)$ rarely far from $\mathbb{E}\text{-}time(x)$
  - formally: bound probability that X (far) exceeds $\mathbb{E}[X]$
    - $\rightsquigarrow$ *concentration bounds* a.k.a. *tail inequalities*
  - 👍 can then compare these typical times again
  - 👍 also obtain more reliable algorithms

  $\rightsquigarrow$ Let's establish some tools for that!

With High Probability

Definition 7.10 (With high probability)

We say

- an event $X = X(n)$ happens *with high probability (w.h.p.)* when
  $\forall c : \mathbb{P}[X(n)] = 1 \pm O(n^{-c})$ as $n \to \infty$.
- a random variable $X = X(n)$ is *in $O(f(n))$ with high probability (w.h.p.)* when
  $\forall c\ \exists d : \mathbb{P}[X \le df(n)] = 1 \pm O(n^{-c})$ as $n \to \infty$.
  (This means, the constant in $O(f(n))$ may depend on c.) ◀

- Very strong notion: failure probability smaller than any polynomial

$\leadsto$ If A succeeds w.h.p. then also polynomially many repetitions of A succeed w.h.p.

Lemma 7.11 (Repetitions w.h.p.)

Suppose A is an algorithm that w.h.p. does not fail.
In n^d *independent* repetitions of A on inputs of size n, w.h.p. *no* repetition fails. ◀

With High Probability [2]

Proof (Lemma ??):

Let c from the definition of w.h.p. be given.

The event F_i that the ith run of A fails happens with probability $O(n^{-(c+d)})$ by definition.

Then $\mathbb{P}[\cup_{i=1}^{n^d} F_i] \;\le\; \sum_{i=1}^{n^d} \mathbb{P}[F_i] \;=\; n^d \cdot O(n^{-(c+d)}) \;=\; O(n^{-c})$. ∎

$\rightsquigarrow$ events that happen with high probability can be combined

Concentration I – Markov's Inequality

Theorem 7.12 (Markov's Inequality)

Let $X \in \mathbb{R}_{\geq 0}$ be a r.v. that assumes only *weakly positive* values. Then holds

$$\forall a > 0 \quad : \quad \mathbb{P}[X \geq a] \;\leq\; \frac{\mathbb{E}[X]}{a}$$

Proof:

Since $X \geq 0$ implies $\mathbb{E}[X] \geq 0$, nicer equivalent form: $\forall a > 0 \;:\; \mathbb{P}\big[X \geq a\mathbb{E}[X]\big] \;\leq\; \dfrac{1}{a}$

Mnemonic: With probability at least $\frac{1}{a}$, won't exceed expectation by a factor a.

Moments

- Markov's Inequality is tight (for some r.v.), but not usually a strong concentration result.
- but we can apply it to $f(X)$ for any (positive) *function* of X!

Towards this, we consider moments of r.v.:

Definition 7.13 (Moments, variance, standard deviation)

For a random variable $X \in \mathbb{R}$,

- $\mathbb{E}[X^k]$ is the *kth moment* of X.
- $\mathbb{E}\left[\left|X - \mathbb{E}[X]\right|^k\right]$ is the *kth centered moment* of X.
- The *variance* of X is the second centered moment: $\mathrm{Var}[X] = \mathbb{E}\left[(X - \mathbb{E}[X])^2\right]$
- The *standard deviation* of X is $\sigma[X] = \sqrt{\mathrm{Var}[X]}$. ◀

Note: None of the moments is guaranteed to exist!

Concentration II – Chebychev's Inequality

Using second moments, we obtain a stronger concentration inequality.

Theorem 7.14 (Chebychev's Inequality)

Let X be a random variable. We have

$$\forall a > 0 \quad : \quad \mathbb{P}\Big[\big|X - \mathbb{E}[X]\big| \ge a\Big] \;\le\; \frac{\mathrm{Var}[X]}{a^2}.$$

Proof:

Convergence in Probability

Corollary 7.15 (Chebychev Concentration)

Let $X_1, X_2, \ldots$ be a sequence of random variables and assume

- $\mathbb{E}[X_n]$ and $\mathrm{Var}[X_n]$ exist for all n and
- $\sigma[X_n] = o(\mathbb{E}[X_n])$ as $n \to \infty$.

Then holds

$$\forall \varepsilon > 0 \quad : \quad \mathbb{P}\left[\left|\frac{X_n}{\mathbb{E}[X_n]} - 1\right| \geq \varepsilon\right] \;\to\; 0 \qquad (n \to \infty).$$

◀

This means $\dfrac{X_n}{\mathbb{E}[X_n]}$ *converges in probability* to 1.

We're getting there, but this is still a far cry from our "with high probability"!
(superpolynomial convergence rate in above limit)

Concentration III – Chernoff Bounds

Stronger concentration inequalities require more assumptions on distribution of X.
A classical one is that X consists of many **small and independent** parts.

Theorem 7.16 (Chernoff Bound for Bernoulli trials)

Let $X_1, \ldots, X_n \in \{0,1\}$ be *(mutually) independent* with $X_i \overset{\mathcal{D}}{=} \mathrm{B}(p_i)$.
Define $X = X_1 + \cdots + X_n$ and $\mu = \mathbb{E}[X_1] + \cdots + \mathbb{E}[X_n] = p_1 + \cdots + p_n$. Then holds

$$\begin{aligned} \forall \delta > 0 \quad &: \quad \mathbb{P}[X \geq (1+\delta)\mu] \ < \ \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu} \\ \forall \delta \in (0,1] \quad &: \quad \mathbb{P}[X \geq (1+\delta)\mu] \ \leq \ \exp(-\mu\delta^2/3) \end{aligned}$$

◀

Proof:

Concentration III – Chernoff Bounds [2]

Proof (cont.):

Chernoff Bound for Binomial Distribution

The algorithmically most widely used special case has identical coin flips.

Corollary 7.17 (Chernoff Bound for Binomial Distribution)

Let $X \overset{\mathcal{D}}{=} \mathrm{Bin}(n,p)$. Then

$$\forall \delta \ge 0 \;:\; \mathbb{P}\left[\left|\frac{X}{n} - p\right| \ge \delta\right] \;\le\; 2\exp(-2\delta^2 n)$$ ◀

$\rightsquigarrow$ $\boxed{\mathrm{Bin}(n,p) \;\in\; np \pm n^{0.501} \;\text{ w.h.p.}}$

7.7 Concentration in Action

Application 1: Majority Voting for Monte Carlo

Monte Carlo algorithms are allowed to err half the time.
That sound unusable in practice ... can we improve upon that?

Idea: Use t *independent* repetitions of A on x.
If at least $\lceil t/2 \rceil$ runs (i. e., an absolute majority) yield result y, return y, otherwise return $\boxed{?}$

Theorem 7.18 (Majority Voting)

Let A be a Monte Carlo algorithm for f with *bounded* error. Then, a *majority vote* of $t = \omega(\log n)$ repetitions of A is correct *with high probability*.

Proof:

Application 1: Majority Voting for Monte Carlo [2]

Proof:

Application 2: Majority Voting for Unbounded Error

Theorem 7.19 (Majority Voting with unbounded error)

There are Monte Carlo algorithms A with *unbounded* error that use only a linear number of random bits ($Random_A(n) = \Theta(n)$ as $n \to \infty$), so that a guarantee for successful *majority votes* with fixed probability $\delta \in (\frac{1}{2}, 1)$ requires the number of repetitions t to satisfy $t = \omega(n^c)$ for *every* constant c as $n \to \infty$. ◀

That means, probability amplification for ***unbounded*** error Monte Carlo methods requires a *superpolynomial* number of repetitions and is not in general tractable.

Proof:

Application 2: Majority Voting for Unbounded Error [2]

Proof (cont.):

Application 2: Majority Voting for Unbounded Error [3]

Proof (cont.):

Randomized Algorithms for Optimization Problems

- For algorithms solving an **optimization problem**,
  "wrong" answers may just be suboptimal solutions
- but we can compute their cost!
  (Wouldn't want to follow a majority vote if that's provably worse!)

$\rightsquigarrow$ Can naturally lead to an approximately optimal answer

- For this chapter: The only correct output = optimal solution

$\rightsquigarrow$ same methodology applies

Application 3: Can we trust Quicksort's expectation?

Theorem 7.20 (Quicksort Concentration)
The height of the recursion tree of (randomized) Quicksort is w.h.p. in $O(\log n)$. ◀

Corollary 7.21
The number of comparisons of randomized Quicksort is w.h.p. in $O(n \log n)$. ◀

Summary

- randomization can provably improve performance
  - not clear in general by how much
- Randomized computation can be modeled formally as *Probabilistic Turing Machines*
  - Bernoulli probability space to handle non-terminating runs
- *Las Vegas* algorithms use randomization internally, but never give wrong results
- *Monte Carlo* algorithms are allowed to output wrong results with small probability
- Chernoff bounds give strong concentration results for independent repetitions