

# Randomization Basics

10 June 2025

Prof. Dr. Sebastian Wild

## 7 Randomization Basics

- 7.1 Motivation
- 7.2 Randomized Selection
- 7.3 Recap of Probability Theory
- 7.4 Probabilistic Turing Machines
- 7.5 Classification of Randomized Algorithms
- 7.6 Tail Inequalities and Concentration Bounds
- 7.7 Concentration in Action

## 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.
- ↪ Here: Let's compromise on “always”, i. e., allow algorithms to occasionally **fail**!

# 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.

↪ Here: Let's compromise on “always”, i. e., allow algorithms to occasionally **fail**!

- ⚡ A *deterministic* algorithm  $A$  that fails on input  $x$  will *always* fail for  $x$ .

↪ What if we require a solution for such an input  $x$ ? We get **nothing** from  $A$ !

- ▶ Must use a form of *nondeterminism*.

# 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.

↪ Here: Let's compromise on “always”, i. e., allow algorithms to occasionally **fail**!

- ⚡ A *deterministic* algorithm  $A$  that fails on input  $x$  will *always* fail for  $x$ .

↪ 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.

↪ *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) *hashing*  
Cheap surrogate question, mostly correct, but sometimes wrong.
  - ▶ Protect against **adversarial inputs**  
We make our (algorithm's) behavior unpredictable, so it is harder to exploit us.

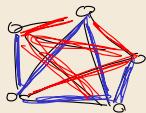
# Why Could Randomization Help?

- ▶ 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 is 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
  - ↪ If algorithm succeeds with prob.  $> 0$ , object must exist!



# Ramsey theory

complete graph on  $n$  vertices



Claim:

$\exists$  monochromatic clique  
of size  $\geq R(n)$

$$R(n) \approx \lg n$$

# Average-Case Analysis vs. Randomized Algorithms

## Average-Case Analysis

- ▶ algorithm is **deterministic**  
same input, same computation

## Randomized Algorithm (here)

- ▶ algorithm is **not** deterministic  
same input, potentially different comp.

# Average-Case Analysis vs. Randomized Algorithms

## Average-Case Analysis

- ▶ algorithm is **deterministic**  
same input, same computation
- ▶ input is chosen according to some  
**probability distribution**

## Randomized Algorithm (here)

- ▶ algorithm is **not** deterministic  
same input, potentially different comp.
- ▶ input is chosen **adversarially** (worst-case  
inputs)

(  
oblivious adversary  
(can't see random bits)

# 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

# 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

*example: sorting by first shuffle*

*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

- ▶ **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 ( $k$ th smallest element).
- but 0-based & counting dups* (arrow pointing to  $k$ )
- ▶  $k = \lfloor n/2 \rfloor \rightsquigarrow$  median;       $k = \lfloor n/4 \rfloor \rightsquigarrow$  lower quartile  
     $k = 0 \rightsquigarrow$  minimum;       $k = n - \ell \rightsquigarrow \ell$ th largest



# Introductory Example – Quickselect

## Selection by Rank

- ▶ **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 ( $k$ th smallest element).
- ▶  $k = \lfloor n/2 \rfloor \rightsquigarrow$  median;     $k = \lfloor n/4 \rfloor \rightsquigarrow$  lower quartile  
 $k = 0 \rightsquigarrow$  minimum;     $k = n - \ell \rightsquigarrow$   $\ell$ th largest

---


```
1 procedure quickselect( $A[0..n)$ ,  $k$ ):  
2    $l := 0$ ;  $r := n$   
3   while  $r - l > 1$   
4      $b :=$  random pivot from  $A[l..r)$   
5      $j :=$  partition( $A[l..r)$ ,  $b$ )  
6     if  $j \geq k$  then  $r := j - 1$   
7     if  $j \leq k$  then  $l := j + 1$   
8   return  $A[k]$ 
```

---

- ▶ simple algorithm:  
determine rank of random element,  
recurse

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


- ▶ worst case:  $\Theta(n^2)$
- ▶  $O(n)$  also possible deterministically,  
but algorithm 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  $2n - o(n)$  comparisons in the worst case. ◀

Proof omitted.

# 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  $2n - o(n)$  comparisons in the worst case. ◀

Proof omitted.

The following weaker result is easier to see:

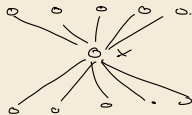
## Theorem 7.2 (Blum et al. (1973))

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

Proof: Two types of comparisons

(1) certificate  
comparisons

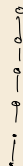
$n-1$



$n=11$

always necessary for correct algorithm

sorting



$n$  elements  
 $n-1$  comps

# A Median Adversary

(2) "nonessential" comparisons

Proof (Theorem 7.2):

(most part of certificate)

in particular, comparisons between  $L$  and  $S$

$m = \text{true median}$

$L = \{x : x > m\}$

$S = \{x : x < m\}$

$(|S| = |L|)$

Given a deterministic algorithm  $A$ ,  
we (the adversary) try to answer  
comparison queries by  $A$  in the  
least useful way (for  $A$ )

Here: maintain elements in 3 sets,  $S$ ,  $L$  and  $U$  (undecided)

initially all in  $U$

query " $x \leq y$ " if  $x$  and  $y$  not in same set, answer  $S < U < L$

$\left. \begin{array}{l} x, y \in S \\ x, y \in L \end{array} \right\}$  arbitrary answer

$x, y \in U$   $x < y$ , put  $x$  to  $S$ ,  $y$  into  $L$

$\Rightarrow$  created one non-essential comp for A  
remove 2 elements from U

$\Rightarrow \geq \frac{n-1}{2}$  non-essential comparisons



# 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!

# Randomized Selection

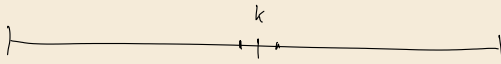
- ▶ Can prove: Randomized Quickselect uses in expectation  $\sim (2 \ln 2 + 2)n \approx 3.39n$  comparisons to find the median
- ▶ But we can do better!

---

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

---

- ▶ Variant of Quickselect with huge sample
- ▶ Analysis sketch:
  - ▶ partition costs  $1.5n$  comparisons



# 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!

---

```
1 procedure floydRivest( $A[\ell..r]$ ,  $k$ ):  
2    $n := r - \ell$   
3   if  $n < n_0$  return quickselect( $A$ ,  $k$ )  
4    $s := \frac{1}{2}n^{2/3}$  // all numbers to be rounded  
5    $sd := \frac{1}{2}\sqrt{\ln(n)s(n-s)/n}$   
6    $S[0..s] :=$  random sample from  $A$   
7    $\hat{k} := s \frac{k}{n}$   
8    $p :=$  floydRivest( $S$ ,  $\hat{k} - sd$ )  
9    $q :=$  floydRivest( $S$ ,  $\hat{k} + sd$ )  
10   $(i, j) :=$  partition  $A$  around  $p_0$  and  $p_1$   
11  if  $i == k$  return  $A[i]$   
12  if  $j == k$  return  $A[j]$   
13  if  $k < i$  return floydRivest( $A[\ell..i]$ ,  $k$ )  
14  if  $k > j$  return floydRivest( $A[j..r]$ ,  $k$ )  
15  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)  $j - i = o(n)$
- $\rightsquigarrow$  all recursive calls expected  $o(n)$  cost



# 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!

---

```
1 procedure floydRivest( $A[\ell..r]$ ,  $k$ ):  
2    $n := r - \ell$   
3   if  $n < n_0$  return quickselect( $A$ ,  $k$ )  
4    $s := \frac{1}{2}n^{2/3}$  // all numbers to be rounded  
5    $sd := \frac{1}{2}\sqrt{\ln(n)s(n-s)/n}$   
6    $S[0..s] :=$  random sample from  $A$   
7    $\hat{k} := s \frac{k}{n}$   
8    $p :=$  floydRivest( $S$ ,  $\hat{k} - sd$ )  
9    $q :=$  floydRivest( $S$ ,  $\hat{k} + sd$ )  
10   $(i, j) :=$  partition  $A$  around  $p_0$  and  $p_1$   
11  if  $i == k$  return  $A[i]$   
12  if  $j == k$  return  $A[j]$   
13  if  $k < i$  return floydRivest( $A[\ell..i]$ ,  $k$ )  
14  if  $k > j$  return floydRivest( $A[j..r]$ ,  $k$ )  
15  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)  $j - i = o(n)$

$\rightsquigarrow$  all recursive calls expected  $o(n)$  cost

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

$\rightsquigarrow$  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

# 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*  $(\Omega, \mathbb{P})$ :

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

fair die

$$\Omega = \left\{ \begin{array}{|c|} \hline 1 \\ \hline \end{array}, \begin{array}{|c|} \hline 2 \\ \hline \end{array}, \begin{array}{|c|} \hline 3 \\ \hline \end{array}, \begin{array}{|c|} \hline 4 \\ \hline \end{array}, \begin{array}{|c|} \hline 5 \\ \hline \end{array}, \begin{array}{|c|} \hline 6 \\ \hline \end{array} \right\}$$

$$\mathbb{P}\left[\begin{array}{|c|} \hline 1 \\ \hline \end{array}\right] = \frac{1}{6}$$

# Probability Spaces

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

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

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

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

# 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*.



# 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

- ▶  $\mathbb{P}[\bar{A}] = 1 - \mathbb{P}[A]$  counter-probability ( $\bar{A} = \Omega \setminus A$ )
- ▶  $\mathbb{P}[\bigcup A_i] \leq \sum_i \mathbb{P}[A_i]$  the union bound (a.k.a. Boole's inequality a.k.a.  $\sigma$ -subadditivity)

# 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

- ▶  $\mathbb{P}[\bar{A}] = 1 - \mathbb{P}[A]$  counter-probability ( $\bar{A} = \Omega \setminus A$ )
- ▶  $\mathbb{P}[\bigcup A_i] \leq \sum_i \mathbb{P}[A_i]$  the *union bound* (a.k.a. Boole's inequality a.k.a.  $\sigma$ -subadditivity)
- ▶  $\{A_1, \dots, A_k\}$  (*mutually independent*)  $\iff \mathbb{P}[\bigcap_i A_i] = \prod_i \mathbb{P}[A_i]$

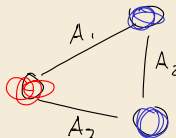
An infinite set of events is mutually independent if every finite subset is so.

*k-wise independence* means that only all size- $k$  subsets are independent.

pairwise indep.  $\exists i, j \in [k] \quad \mathbb{P}[A_i \cap A_j] = \mathbb{P}[A_i] \cdot \mathbb{P}[A_j]$

$\nRightarrow$  mutual independence

$\{A_1, A_2, A_3\}$  2-wise indep.  
but not mut. indep.



$A_i =$  edge has  
2-colors  
at endpoints

# 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

- ▶  $\mathbb{P}[\bar{A}] = 1 - \mathbb{P}[A]$  counter-probability ( $\bar{A} = \Omega \setminus A$ )
- ▶  $\mathbb{P}[\bigcup A_i] \leq \sum_i \mathbb{P}[A_i]$  the *union bound* (a.k.a. Boole's inequality a.k.a.  $\sigma$ -subadditivity)

- ▶  $\{A_1, \dots, A_k\}$  (*mutually independent*)  $\iff \mathbb{P}[\bigcap_i A_i] = \prod_i \mathbb{P}[A_i]$

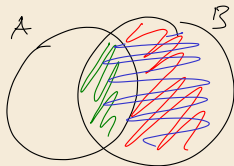
An infinite set of events is mutually independent if every finite subset is so.

*k-wise independence* means that only all size- $k$  subsets are independent.

- ▶ *conditional probability* for  $A$  given  $B$ :  $\mathbb{P}[A | B] = \frac{\mathbb{P}[A \cap B]}{\mathbb{P}[B]}$   
generally undefined if  $\mathbb{P}[B] = 0$

- ▶ *law of total probability*: If  $\Omega = B_1 \dot{\cup} B_2 \dot{\cup} \dots$  is a partition of  $\Omega$ , we have

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



# Random Variables

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

für die

$$\Omega = \left\{ \begin{array}{|c|} \hline \bullet \\ \hline \end{array}, \begin{array}{|c|} \hline \bullet \bullet \\ \hline \end{array}, \begin{array}{|c|} \hline \bullet \bullet \bullet \\ \hline \end{array}, \begin{array}{|c|} \hline \bullet \bullet \bullet \bullet \\ \hline \end{array}, \begin{array}{|c|} \hline \bullet \bullet \bullet \bullet \bullet \\ \hline \end{array}, \begin{array}{|c|} \hline \bullet \bullet \bullet \bullet \bullet \bullet \\ \hline \end{array} \right\}$$

$$\mathbb{P}[\begin{array}{|c|} \hline \bullet \\ \hline \end{array}] = \frac{1}{6}$$

$$X : \Omega \rightarrow \{1, \dots, 6\}$$

$$Y : \Omega \rightarrow \{\text{even}, \text{odd}\}$$

# Random Variables

*Random variables* (r.v.)  $X : \Omega \rightarrow \mathcal{X}$ ; 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  $\mathbb{1}_A(\omega) = [\omega \in A] = \begin{cases} 1 & \omega \in A \\ 0 & \text{else} \end{cases}$

# Random Variables

*Random variables* (r.v.)  $X : \Omega \rightarrow \mathcal{X}$ ; 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  $\mathbb{1}_A(\omega) = [\omega \in A]$
- ▶  $F_X(x) = \mathbb{P}[X \leq x]$  is the *cumulative distribution function (CDF)*.
- ▶  $X$  is *discrete* if  $X(\Omega) = \{X(\omega) : \omega \in \Omega\}$  is countable.  $\mathcal{X} = \mathbb{N}$
- ▶ 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*.

# Random Variables

*Random variables* (r.v.)  $X : \Omega \rightarrow \mathcal{X}$ ; 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  $\mathbb{1}_A(\omega) = [\omega \in A]$
- ▶  $F_X(x) = \mathbb{P}[X \leq x]$  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*.

$$X \stackrel{\mathcal{D}}{=} Y + Z$$

## Equality in distribution:

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

# Independent Random Variables

## Independence:

- ▶ Consider *vector*  $X = (X_1, \dots, X_k)$  as single function from  $\Omega$  to  $\mathbb{R}^k$ .  
CDF/PMF/PDF of  $X$  is called *joint CDF/PMF/PDF* of  $X_1, \dots, X_k$ .
- ▶ 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)



# Independent Random Variables

## Independence:

- ▶ Consider *vector*  $X = (X_1, \dots, X_k)$  as single function from  $\Omega$  to  $\mathbb{R}^k$ .  
CDF/PMF/PDF of  $X$  is called *joint CDF/PMF/PDF* of  $X_1, \dots, X_k$ .
- ▶ 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  $X_1, X_2, \dots$
- ▶ a sequence of *i.i.d.* r.v.  $X_1, X_2, \dots$  (<sup>mutually</sup>*independent and identically distributed*)  
has  $X_i \stackrel{\mathcal{D}}{=} X_1$  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.

# 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  $\mathbb{E}[X \mid A] = \sum_x^{\mathcal{X}} \mathbb{P}[X = x \mid A]$  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  $Y$

$$\mathbb{E}[X \mid Y = y] = \sum_{x \in \mathcal{X}} x \cdot \mathbb{P}[X = x \mid \{Y = y\}]$$

# 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  $Y$

$$\mathbb{E}[X \mid Y = y] = \sum_{x \in \mathcal{X}} x \cdot \mathbb{P}[X = x \mid \{Y = y\}]$$

- ▶ 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

$$\mathbb{E}[X \mid Y = y] = \int_{\mathcal{X}} x \cdot f_{X|Y}(x, y) dx \quad \text{with} \quad f_{X|Y}(x, y) = \frac{f_{(X,Y)}(x, y)}{f_Y(y)}$$

# 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  $Y$

$$\mathbb{E}[X \mid Y = y] = \sum_{x \in \mathcal{X}} x \cdot \mathbb{P}[X = x \mid \{Y = y\}]$$

- ▶ 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

$$\mathbb{E}[X \mid Y = y] = \int_{\mathcal{X}} x \cdot f_{X|Y}(x, y) dx \quad \text{with} \quad f_{X|Y}(x, y) = \frac{f_{(X,Y)}(x, y)}{f_Y(y)}$$

- ▶ With  $g(y) := \mathbb{E}[X \mid Y = y]$  we obtain a *new r.v.*  $\underline{\mathbb{E}[X \mid Y]} = g(Y)$ .
- ▶ *law of total expectation*:  $\mathbb{E}[X] = \mathbb{E}_Y[\mathbb{E}_X[X \mid Y]]$ .

# Famous Distributions

## discrete

- ▶ *Bernoulli r.v.*  $X \stackrel{\mathcal{D}}{=} \text{B}(p) \rightsquigarrow \mathbb{P}[X = 1] = p, \mathbb{P}[X = 0] = 1 - p$
- ▶ *Binomial r.v.*  $Y \stackrel{\mathcal{D}}{=} \text{Bin}(n, p) \rightsquigarrow Y = X_1 + \dots + X_n \text{ for } X_1, \dots, X_n \text{ i.i.d. } X_i \stackrel{\mathcal{D}}{=} \text{B}(p)$
- ▶ *discrete uniform r.v.*  $X \stackrel{\mathcal{D}}{=} \mathcal{U}([0..n)) \rightsquigarrow \mathbb{P}[X = i] = \frac{1}{n} \text{ for } i \in [0..n) \quad (\text{else } 0) \quad \text{disc} \stackrel{\mathcal{D}}{=} \mathcal{U}(\{1..6\})$
- ▶ *Geometric r.v.*  $X \stackrel{\mathcal{D}}{=} \text{Geo}(p) \rightsquigarrow \mathbb{P}[X = k] = (1 - p)^{k-1}p \text{ for } k \in \mathbb{N}_{\geq 1}$

## continuous

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

(of course there are many more)

## 7.4 Probabilistic Turing Machines



# Model of Computation

## Definition 7.3 (Probabilistic Turing Machine)

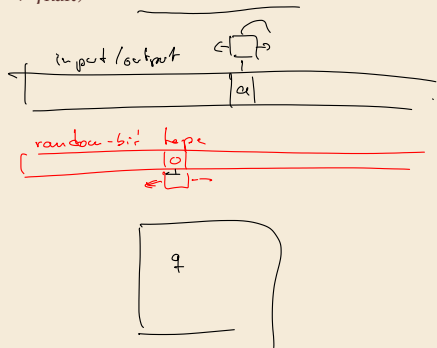
A *probabilistic Turing Machine* (PTM)  $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, q_{\text{halt}})$  is a deterministic TM with an additional read-only tape, filled with random bits.

The *transition function*  $\delta$  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*  $d_r \in \{L, R, N\}$



# Model of Computation

## Definition 7.3 (Probabilistic Turing Machine)

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

The *transition function*  $\delta$  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.  $B(\frac{1}{2})$  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  $\sigma$
- ▶ *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**  $\rho$ , computation is deterministic:  
 $M(x, \rho) = y$  if  $(q_0 x, q_0 \rho) \vdash^* (q_{\text{halt}} y, \rho' q_{\text{halt}} \rho'')$

# 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  $\sigma$
- ▶ *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**  $\rho$ , computation is deterministic:  
 $M(x, \rho) = y$  if  $(q_0 x, q_0 \rho) \vdash^* (q_{\text{halt}} y, \rho' q_{\text{halt}} \rho'')$

$\rightsquigarrow$  *Randomized computation of PTM:*

**random variable**  $M(x, B_0 B_1 B_2 \dots)$  where  $B_0, B_1, B_2, \dots$  are i.i.d.  $B(\frac{1}{2})$  distributed

---

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

- ▶ 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?

## Warmup: Rejection Sampling

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

---

```
1 procedure rollDie():
2   do
3     Draw 3 random bits  $b_2, b_1, b_0$ 
4     // Interpret as binary representation of a number in  $[0..7]$ 
5      $n = \sum_{i=0}^2 2^i b_i$ 
6   while ( $n = 0 \vee n = 7$ )
7   return  $n$ 
```

---

## Warmup: Rejection Sampling

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

---

```
1 procedure rollDie():
2   do
3     Draw 3 random bits  $b_2, b_1, b_0$ 
4     // Interpret as binary representation of a number in  $[0..7]$ 
5      $n = \sum_{i=0}^2 2^i b_i$ 
6   while ( $n = 0 \vee n = 7$ )
7   return  $n$ 
```

---

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

## Warmup: Rejection Sampling

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

---

```
1 procedure rollDie():
2   do
3     Draw 3 random bits  $b_2, b_1, b_0$ 
4     // Interpret as binary representation of a number in  $[0..7]$ 
5      $n = \sum_{i=0}^2 2^i b_i$ 
6     while  $(n = 0 \vee n = 7)$ 
7     return  $n$ 
```

---

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

$$\# \text{ iterations} \stackrel{D}{=} \text{Geo}\left(\frac{3}{4}\right)$$

**Termination:** *Infinite* runs possible!

$$\text{ / } \mathbb{E}[\text{Geo}(p)] = \frac{1}{p}$$

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

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

- ⚡ don't ever terminate, no output

- ↪ Clearly don't want that (just as before)

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

# What can go wrong?

What can go wrong in a randomized computation?

- ▶ Computation could run into a deterministic infinite loop (as for deterministic TM)

- ⚡ don't ever terminate, no output

- ↪ 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)

- ↪ 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  $\infty$  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.

►  $\{0, 1\}^*$  (the set of **finite** bitstrings) is countably infinite (=discrete)


► But the set of *infinite strings* ( $\omega$ -language) is not!

$\{0, 1\}^\omega = \{b_0 b_1 \dots : b_i \in \{0, 1\}\} = \{b : b : \mathbb{N}_0 \rightarrow \{0, 1\}\}$  surjectively maps to  $[0, 1) \subset \mathbb{R}$

$b \mapsto 0.b_0 b_1 b_2 \dots$

# 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.
  - ▶  $\{0, 1\}^*$  (the set of **finite** bitstrings) is countably infinite (=discrete)
  - ▶ But the set of *infinite strings* ( $\omega$ -language) is not!  
 $\{0, 1\}^\omega = \{b_0 b_1 \dots : b_i \in \{0, 1\}\} = \{b : b : \mathbb{N}_0 \rightarrow \{0, 1\}\}$  surjectively maps to  $[0, 1) \subset \mathbb{R}$   
 $b \mapsto 0.b_0 b_1 b_2 \dots$
- ▶ Config  $(\alpha q \beta, \rho q \sigma)$  for PTM needs  $\sigma \in \{0, 1\}^\omega$  in general

# 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.
  - ▶  $\{0, 1\}^*$  (the set of **finite** bitstrings) is countably infinite (=discrete)
  - ▶ But the set of **infinite strings** ( $\omega$ -language) is not!  
 $\{0, 1\}^\omega = \{b_0 b_1 \dots : b_i \in \{0, 1\}\} = \{b : b : \mathbb{N}_0 \rightarrow \{0, 1\}\}$  surjectively maps to  $[0, 1) \subset \mathbb{R}$   
 $b \mapsto 0.b_0 b_1 b_2 \dots$
- ▶ Config  $(\alpha q \beta, \rho q \sigma)$  for PTM needs  $\sigma \in \{0, 1\}^\omega$  in general
- ▶ Define the random variable  $Time_M(x) \in \mathbb{N}_0 \cup \{\infty\}$  on the *Bernoulli probability space*
  - ▶ generators:  $\{\pi_x : x \in \{0, 1\}^*\}$  where  $\pi_x = \{xw : w \in \{0, 1\}^\omega\} \subseteq \{0, 1\}^\omega$
  - ▶ Bernoulli  $\sigma$ -algebra: smallest  $\mathcal{F}$  containing all  $\{\pi_x\}_x$  that is closed under countable union and complement
  - ▶  $\mathbb{P}[\pi_x] = 2^{-|x|}$

rollDie terminates iff  $\rho \in T \subseteq \{0, 1\}^\omega$   $\mathbb{P}[T]$

# 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.
  - ▶  $\{0, 1\}^*$  (the set of **finite** bitstrings) is countably infinite (=discrete)
  - ▶ But the set of **infinite strings** ( $\omega$ -language) is not!  
 $\{0, 1\}^\omega = \{b_0 b_1 \dots : b_i \in \{0, 1\}\} = \{b : b : \mathbb{N}_0 \rightarrow \{0, 1\}\}$  surjectively maps to  $[0, 1) \subset \mathbb{R}$   
 $b \mapsto 0.b_0 b_1 b_2 \dots$
- ▶ Config  $(\alpha q \beta, \rho q \sigma)$  for PTM needs  $\sigma \in \{0, 1\}^\omega$  in general
- ▶ Define the random variable  $Time_M(x) \in \mathbb{N}_0 \cup \{\infty\}$  on the *Bernoulli probability space*
  - ▶ generators:  $\{\pi_x : x \in \{0, 1\}^*\}$  where  $\pi_x = \{xw : w \in \{0, 1\}^\omega\} \subseteq \{0, 1\}^\omega$
  - ▶ Bernoulli  $\sigma$ -algebra: smallest  $\mathcal{F}$  containing all  $\{\pi_x\}_x$  that is closed under countable union and complement
  - ▶  $\mathbb{P}[\pi_x] = 2^{-|x|}$

$\rightsquigarrow$  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. worst case

Moreover, we define the *expected time* as

$$time_{nondet}() = \infty$$

$$\mathbb{E}\text{-}time_M(x) = \mathbb{E}[time_M(x)] = \mathbb{E}_{\rho}[\inf\{t \in \mathbb{N}_0 : (q_0x, q_0\rho) \vdash^t (q_{halt}y, \rho'q_{halt}\rho'')\}]$$

$\uparrow$   
random

Similarly

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



## (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

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



- ▶ 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  $\Sigma$ , the *random-bit cost* for an input  $x \in \Sigma^*$  is denote by

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

and similarly

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

Further, the *expected random-bit cost* are defined as

$\mathbb{E}\text{-}random_M(x) = \mathbb{E}_\rho[random_M(x)]$  and

$\mathbb{E}\text{-}Random_M(n) = \sup\{\mathbb{E}\text{-}random_M(x) : x \in \Sigma^n\}$



# 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
- ↪ 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*  $f : \Sigma^* \rightarrow \Gamma^*$ .

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

# Las Vegas

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

$\leadsto$  Covers *decision problems*  $L \subseteq \Sigma^*$  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  $f : \Sigma^* \rightarrow \Gamma^*$  if for all  $x \in \Sigma^*$  holds

- (a)  $\mathbb{P}[time_A(x) < \infty] = 1$  (*terminate almost surely*)
- (b)  $A(x) \in \{f(x), \boxed{?}\}$  (answer always *correct* or “*don’t know*”)
- (c)  $\mathbb{P}[A(x) = f(x)] \geq \frac{1}{2}$  (*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  $f : \Sigma^* \rightarrow \Gamma^*$  can be transformed into a randomized algorithm  $B$  for  $f$  so that for all  $x \in \Sigma^*$  holds

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

(b)  $\mathbb{E}\text{-time}_B(x) \leq 2 \cdot \text{time}_A(x)$

(can never output  $\boxed{?}$ , not "just"  
almost surely)



# Don't Know vs. Won't Terminate

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

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

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

(b)  $\mathbb{E}\text{-time}_B(x) \leq 2 \cdot \text{time}_A(x)$

**Proof:**

See exercises.



# Don't Know vs. Won't Terminate

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

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

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

(b)  $\mathbb{E}\text{-time}_B(x) \leq 2 \cdot \text{time}_A(x)$

**Proof:**

See exercises.

## Theorem 7.8 (Termination Enforcible)

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

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

(b)  $\text{time}_A(x) \leq 2 \cdot \mathbb{E}\text{-time}_B(x)$  (always terminates).



# Don't Know vs. Won't Terminate

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

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

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

(b)  $\mathbb{E}\text{-time}_B(x) \leq 2 \cdot \text{time}_A(x)$

**Proof:**

See exercises.

## Theorem 7.8 (Termination Enforcible)

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

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

(b)  $\text{time}_A(x) \leq 2 \cdot \mathbb{E}\text{-time}_B(x)$  (always terminates).

**Proof:**

See exercises.

# Las Vegas Variants

↪ 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 7.7)

**(B)** *Always-Terminating Las Vegas algorithms* (output of Theorem 7.8)

# 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:

---

```
1 procedure rollDieLasVegas:
2   Draw 3 random bits  $b_2, b_1, b_0$ 
3    $n = \sum_{i=0}^2 2^i b_i$  // Interpret as binary representation of a number in  $[0 : 7]$ 
4   if ( $n = 0 \vee n = 7$ )
5     return ?
6   else
7     return  $n$ 
```

---

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

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

# To Err is Algorithmic

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

# 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  $f : \Sigma^* \rightarrow \Gamma^*$

- ▶ with *bounded error* if  $\exists \varepsilon > 0 \forall x \in \Sigma^* : \mathbb{P}[A(x) = f(x)] \geq \frac{1}{2} + \varepsilon$ .
- ▶ with *unbounded error* if  $\forall x \in \Sigma^* : \mathbb{P}[A(x) = f(x)] > \frac{1}{2}$ .

# 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  $f : \Sigma^* \rightarrow \Gamma^*$

- ▶ with *bounded error* if  $\exists \varepsilon > 0 \forall x \in \Sigma^* : \mathbb{P}[A(x) = f(x)] \geq \frac{1}{2} + \varepsilon.$
- ▶ with *unbounded error* if  $\forall x \in \Sigma^* : \mathbb{P}[A(x) = f(x)] > \frac{1}{2}.$

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.
- ↪ need a way to **compare** random variables!

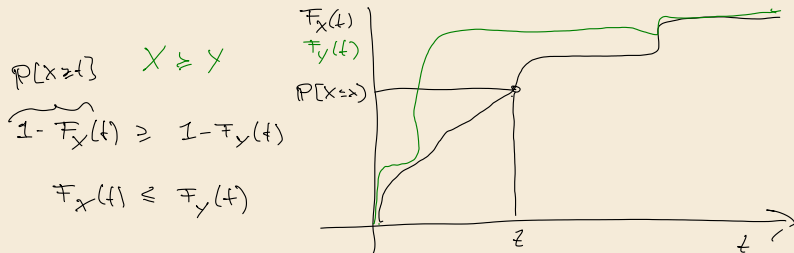


# How To Summarize A Random Variable?

- ▶ 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.
- ⇒ need a way to **compare** random variables!

- ▶ One option: *stochastic dominance*

- ▶ If  $\forall t : \mathbb{P}[\overset{X}{time}_A(x) \geq t] \geq \mathbb{P}[\overset{Y}{time}_B(x) \geq t]$ ,  
we say  $time_A(x)$  (weakly) stochastically dominates  $time_B(x)$  (on input  $x$ )
- ▶ Would rather use  $B$  then!



# 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.

↪ need a way to **compare** random variables!

- ▶ One option: *stochastic dominance*
  - ▶ If  $\forall t : \mathbb{P}[\text{time}_A(x) \geq t] \geq \mathbb{P}[\text{time}_B(x) \geq t]$ ,  
we say  $\text{time}_A(x)$  (weakly) stochastically dominates  $\text{time}_B(x)$  (on input  $x$ )
  - ▶ Would rather use  $B$  then!
- 👎 dominance rarely true for real algorithms
- 👎 no prediction of running time / comparison with explicit bound

# 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.


↪ need a way to **compare** random variables!

- ▶ One option: *stochastic dominance*


- ▶ If  $\forall t : \mathbb{P}[\text{time}_A(x) \geq t] \geq \mathbb{P}[\text{time}_B(x) \geq t]$ ,  
we say  $\text{time}_A(x)$  (weakly) stochastically dominates  $\text{time}_B(x)$  (on input  $x$ )


- ▶ Would rather use  $B$  then!

-  dominance rarely true for real algorithms

-  no prediction of running time / comparison with explicit bound

↪ 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
- ↪ both have  $\mathbb{E}\text{-time}(x) = 2$
- ▶ probably want  $A \dots$  certainly would want to be able to distinguish them!

# 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
  - ↪ both have  $\mathbb{E}\text{-time}(x) = 2$
  - ▶ probably want  $A \dots$  certainly would want to be able to distinguish them!
- ▶ **Goal:** Strengthen algorithms so  $\text{time}(x)$  rarely far from  $\mathbb{E}\text{-time}(x)$ 
  - ▶ formally: bound probability that  $X$  (far) exceeds  $\mathbb{E}[X]$ 
    - ↪ *concentration bounds* a.k.a. *tail inequalities*
  - 👍 can then compare these typical times again
  - 👍 also obtain more reliable algorithms
  - ↪ Let's establish some tools for that!

# With High Probability

⚠ not always the same definition

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

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

- ▶ Very strong notion: failure probability smaller than any polynomial

$\rightsquigarrow$  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]

- 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  $n \rightarrow \infty$ .

**Proof (Lemma 7.11):**

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

$$\mathcal{F}_i \equiv \text{if } A \text{ fails} \quad B = \bigcup_{i=1}^n A_i \quad \text{claim: } \overline{B} \text{ w.h.p.}$$

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



## With High Probability [2]

**Proof (Lemma 7.11):**

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

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

▪  
▪  
▪  
▪  
▪  
▪  
▪  
▪  
▪  
▪

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

## With High Probability [2]

**Proof (Lemma 7.11):**

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

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

Then  $\mathbb{P}[\cup_{i=1}^{n^d} F_i]$

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



## With High Probability [2]

**Proof (Lemma 7.11):**

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

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

$$\text{Then } \mathbb{P}[\cup_{i=1}^{n^d} F_i] \leq \sum_{i=1}^{n^d} \mathbb{P}[F_i]$$

union bound

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



## With High Probability [2]

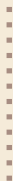
**Proof (Lemma 7.11):**

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

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

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

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



## With High Probability [2]

**Proof (Lemma 7.11):**

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

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

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

$\rightsquigarrow$  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

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

**Proof:** Let  $a > 0$  given, define  $I := \mathbb{1}_{\{X \geq a\}} = [X \geq a] = \begin{cases} 1 & X \geq a \\ 0 & X < a \end{cases}$

$$I \leq \frac{X}{a} \quad \begin{array}{l} \text{if } X \geq a \leadsto I=1 \text{ and } \frac{X}{a} \geq 1 \quad \checkmark \\ \text{if } X < a \leadsto I=0 \text{ and } X \geq 0 \leadsto \frac{X}{a} \geq 0 \quad \checkmark \end{array}$$

take  $\mathbb{E}$   $\nearrow$

$$\frac{\mathbb{E}[X]}{a} \geq \mathbb{E}[I] = \mathbb{P}[X \geq a]$$

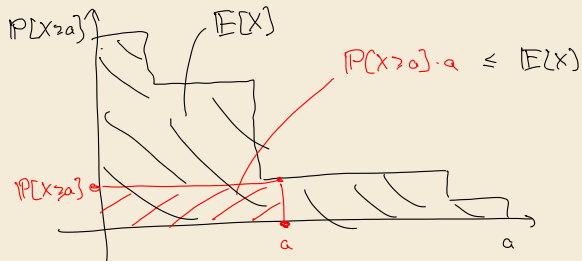
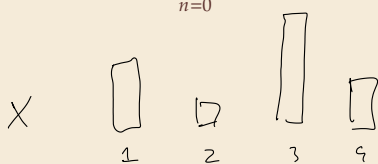
Since  $X \geq 0$  implies  $\mathbb{E}[X] \geq 0$ , nicer equivalent form:  $\forall a > 0 : \mathbb{P}[X \geq a\mathbb{E}[X]] \leq \frac{1}{a}$

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

# Markov's Inequality Visually

If  $X \in \mathbb{N}_0$ , we can visualize Markov's Inequality nicely.

Recall:  $X \in \mathbb{N}_0$  implies  $\mathbb{E}[X] = \sum_{n=0}^{\infty} n \cdot \mathbb{P}[X = n] = \sum_{n=0}^{\infty} \mathbb{P}[X \geq n]$ .



# 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  $X$ !



# 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  $X$ .  $\mathbb{E}[X]$  1st moment
- ▶  $\mathbb{E}[|X - \mathbb{E}[X]|^k]$  is the *kth centered moment* of  $X$ .
- ▶ The *variance* of  $X$  is the second centered moment:  $\text{Var}[X] = \mathbb{E}[(X - \mathbb{E}[X])^2]$
- ▶ The *standard deviation* of  $X$  is  $\sigma[X] = \sqrt{\text{Var}[X]}$ . ◀

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

$$\forall a > 0 : \mathbb{P}[|X - \mathbb{E}[X]| \geq a] \leq \frac{\text{Var}[X]}{a^2}.$$

**Proof:** Apply Markov to  $Y = (X - \mathbb{E}[X])^2$

$$\mathbb{P}[Y \geq b] \leq \frac{\mathbb{E}[Y]}{b}$$
$$\mathbb{P}[Y \geq b] = \mathbb{P}[(X - \mathbb{E}[X])^2 \geq b] = \mathbb{P}[|X - \mathbb{E}[X]| \geq \sqrt{b}] \leq \frac{\mathbb{E}[Y]}{b} = \frac{\text{Var}[X]}{a^2}$$

↑  
take  $\sqrt{\phantom{x}}$  on  
both sides

# Convergence in Probability

## Corollary 7.15 (Chebychev Concentration)

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

►  $\mathbb{E}[X_n]$  and  $\text{Var}[X_n]$  exist for all  $n$  and

►  $\sigma[X_n] = o(\mathbb{E}[X_n])$  as  $n \rightarrow \infty$ .

Then holds

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

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

# Convergence in Probability

## Corollary 7.15 (Chebychev Concentration)

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

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

Then holds

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

This means  $\frac{X_n}{\mathbb{E}[X_n]}$  *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.

## 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  $X_1, \dots, X_n \in \{0, 1\}$  be *(mutually) independent* with  $X_i \stackrel{\mathcal{D}}{=} B(p_i)$ .

Define  $\underline{X} = X_1 + \dots + X_n$  and  $\mu = \mathbb{E}[X_1] + \dots + \mathbb{E}[X_n] = p_1 + \dots + p_n$ . Then holds

$$(1) \quad \forall \delta > 0 : \mathbb{P}[X \geq (1 + \delta)\mu] < \left( \frac{e^\delta}{(1 + \delta)^{1 + \delta}} \right)^\mu$$

$$(2) \quad \forall \delta \in (0, 1] : \mathbb{P}[X \geq (1 + \delta)\mu] \leq \exp(-\mu\delta^2/3)$$

$$\delta = 1 \quad \frac{e}{2^2} = \frac{e}{4} < \frac{3}{4}$$

**Proof:** Again Markov... Let  $t > 0$   
 $e^{t \cdot \cdot}$  on both sides  
 $(1) \Rightarrow (2)$   
 $Y = e^{t \cdot X} \geq 0$

$$Y_i = e^{t X_i}$$

$X_1, \dots, X_n$  unabh. indep.

$$\mathbb{P}[X \geq (1 + \delta)\mu] \stackrel{e^{t \cdot \cdot} \text{ on both sides}}{\leq} \mathbb{P}[Y \geq e^{t(1 + \delta)\mu}] \stackrel{\text{Markov}}{\leq} \frac{\mathbb{E}[Y]}{e^{t(1 + \delta)\mu}}$$

$$\mathbb{E}[Y] = \mathbb{E}[e^{t \cdot X}] = \mathbb{E}[e^{t \sum_{i=1}^n X_i}] = \mathbb{E}\left[\prod_{i=1}^n e^{t \cdot X_i}\right] = \prod_{i=1}^n \mathbb{E}[e^{t X_i}]$$

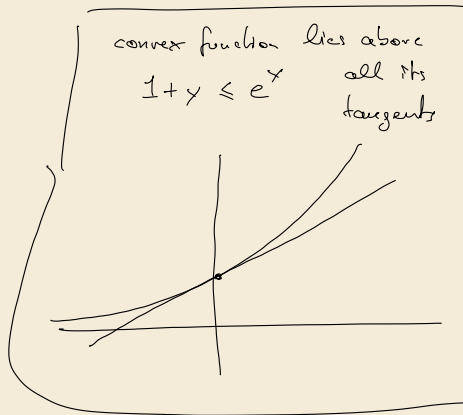
## Concentration III – Chernoff Bounds [2]

Proof (cont.):

$$\begin{aligned}\mathbb{E}[e^{tX_i}] &= 1 \cdot \mathbb{P}\{X_i=0\} + e^t \mathbb{P}\{X_i=1\} \\ &= (1-p_i) + e^t p_i\end{aligned}$$

$$\begin{aligned}\mathbb{E}[Y] &= \prod_{i=1}^n \left( \underbrace{1 + p_i (e^t - 1)}_{\leq e^{p_i(e^t - 1)}} \right) \\ &\leq \exp\left(\sum_{i=1}^n p_i (e^t - 1)\right) \\ &= \exp\left((e^t - 1) \underbrace{\sum p_i}_{= \mu}\right) \quad (*)\end{aligned}$$

$$\begin{aligned}\mathbb{E}[X \geq (1+\delta)\mu] &\leq \frac{\mathbb{E}[Y]}{e^{t(1+\delta)\mu}} \quad (t > 0) \\ &\leq \frac{e^{\mu(e^t - 1)}}{e^{\mu t(1+\delta)}} = \frac{(e^{e^t - 1})^\mu}{(e^{t(1+\delta)})^\mu}\end{aligned}$$



with  $t = \ln(1+s) > 0$

$$= \frac{e^{\mu(1+s)-1}}{(1+s)^{\mu(1+s)}} = \left( \frac{e^s}{(1+s)^{1+s}} \right)^{\mu}$$

$\square$



# 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  $X \stackrel{\mathcal{D}}{=} \text{Bin}(n, p)$ . Then

$$\forall \delta \geq 0 \quad : \quad \mathbb{P} \left[ \left| \frac{X}{n} - p \right| \geq \delta \right] \leq 2 \exp(-2\delta^2 n)$$

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

## **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?

## 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  $\boxed{?}$

# 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  $?$

**Theorem 7.18 (Majority Voting)**  $\exists \varepsilon \forall x \quad \mathbb{P}[A(x) = f(x)] \geq \frac{1}{2} + \varepsilon$

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*. (t odd)

**Proof:** (Assume  $t = \log^2 n$ )

$$\varepsilon = o(n^{-c}) \quad \forall c$$

$$X_1, X_2, \dots, X_t \stackrel{\text{iid}}{\sim} X_i = [\text{ith run succeeds}] \stackrel{\text{d}}{=} \mathcal{B}(p) \quad p \geq \frac{1}{2} + \varepsilon$$

$$X = X_1 + \dots + X_t \stackrel{\text{d}}{=} \text{Bin}(t, p)$$

for bound suffices to assume  $\boxed{p = \frac{1}{2} + \varepsilon}$

$$\text{majority vote fails} \quad \Leftrightarrow \quad X \leq \left\lfloor \frac{t}{2} \right\rfloor$$

# Application 1: Majority Voting for Monte Carlo [2]

**Proof:**  $\mathbb{P}[X \leq \lfloor \frac{t}{2} \rfloor] \leq \mathbb{P}[X \leq \frac{t}{2}] = \mathbb{P}\left[p - \frac{X}{t} \geq p - \frac{1}{2}\right]$

$$\leq \mathbb{P}\left[\left|\frac{X}{t} - p\right| \geq \underbrace{p - \frac{1}{2}}_{=\epsilon > 0}\right]$$

$$\leq 2e^{-2\epsilon^2 t} \quad (*)$$

**Corollary 7.17 (Chernoff Bound for Binomial Distribution)**

Let  $X \stackrel{d}{=} \text{Bin}(n, p)$ . Then

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

To show w.h.p. let  $c \in \mathbb{N}$  arbitrary since

$$n^c \cdot \mathbb{P}[\text{MV fails}] \xrightarrow{!} 0 \quad \leadsto \quad \mathbb{P}[\text{MV fails}] = o(n^{-c})$$

$$n^c \cdot \mathbb{P}[\text{MV fails}] \stackrel{(*)}{\leq} \exp\left(\underbrace{c \ln(n) - 2\epsilon^2 t}_{\rightarrow -\infty}\right) \rightarrow 0 \quad \leadsto \quad \text{MV doesn't fail w.h.p.}$$

since  $t = \omega(\log n) \rightarrow -\infty$

## 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 ( $\text{Random}_A(n) = \Theta(n)$  as  $n \rightarrow \infty$ ), 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  $n \rightarrow \infty$ . ◀

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

**Proof:** success prob  $> \frac{1}{2}$        $\text{Random}_A(n) = n$

$\Rightarrow A$  now has  $2^{\text{Random}_A(n)} = 2^n$  different runs on same input  $x$   
(each has prob.  $2^{-n}$ )

same approach as above      MV fails if  $X \leq \frac{t}{2}$       for  $X \stackrel{D}{=} \text{Bin}(t, p)$

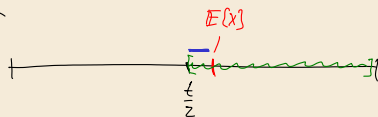
$$p > \frac{1}{2} \quad \varepsilon$$
$$\leadsto p = \frac{1}{2} + 2^{-n}$$

## Application 2: Majority Voting for Unbounded Error [2]

**Proof (cont.):** We will show  $\underbrace{P[X \in (\frac{t}{2}, \mathbb{E}[X])]}_{\text{success range}} \stackrel{!}{\leq} \varepsilon \cdot t$

Since Bin is symmetric

$$P[X \leq \mathbb{E}[X]] \geq \frac{1}{2}$$



$$\begin{aligned} P[\text{MV fails}] &= P[X \leq \frac{t}{2}] \geq P[X < \frac{t}{2}] \\ &= P[X \leq \mathbb{E}[X]] - P[X \in (\frac{t}{2}, \mathbb{E}[X])] \\ &\geq \frac{1}{2} - \varepsilon t \end{aligned}$$

$$\delta > \frac{1}{2}$$

$$\text{If } \boxed{t = n^c} \quad \boxed{\varepsilon = 2^{-n}} \quad = \quad \frac{1}{2} - \underbrace{2^{-n} n^c}_{\rightarrow 0} \quad \rightarrow \quad \frac{1}{2} < \delta$$

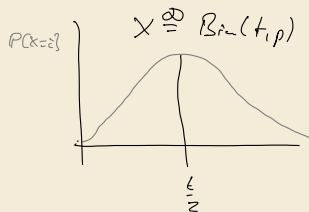
$\Rightarrow$  no polynomial number of repetitions suffice



## Application 2: Majority Voting for Unbounded Error [3]

Proof (cont.): Still to show  $\mathbb{P}[X \in (\frac{t}{2}, \mathbb{E}[X])] \leq \varepsilon t 3^{\varepsilon t}$

$$\mathbb{P}[X \in (\frac{t}{2}, \mathbb{E}[X])] = \sum_{i=\frac{t}{2}+1}^{\frac{t}{2}+\varepsilon t} \underbrace{\binom{t}{i} \left(\frac{1}{2}+\varepsilon\right)^i \left(\frac{1}{2}-\varepsilon\right)^{t-i}}_{\mathbb{P}[X=i]}$$



largest  
summand

$$\leq \sum_{i=\frac{t}{2}+1}^{\frac{t}{2}+\varepsilon t} \underbrace{\binom{t}{t/2}}_{\leq 2^t = 4^{t/2}} \left(\frac{1}{2}+\varepsilon\right)^{t/2} \left(\frac{1}{2}-\varepsilon\right)^{t/2}$$

$$= \varepsilon \cdot t \cdot \underbrace{\left(4 \left(\frac{1}{2}+\varepsilon\right) \left(\frac{1}{2}-\varepsilon\right)\right)^{t/2}}_{1-4\varepsilon^2 \leq 1}$$

$$\leq \varepsilon \cdot t$$

# 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!)

↪ Can naturally lead to an approximately optimal answer

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

↪ same methodology applies

## Application 3: Can we trust Quicksort's expectation?

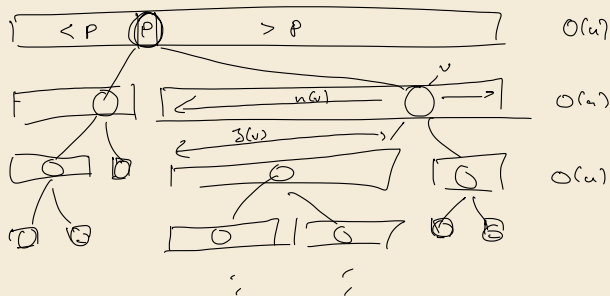
### Theorem 7.20 (Quicksort Concentration)

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  $O(n \log n)$ .

each step picks uniformly random pivot



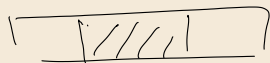
"Where's the binomial dist?"

$v$  node in recursion tree

$n(v)$  # elements in  $v$ 's subproblem

$z(v)$  random # of elements in left recursive call  $z(v) = n(v, \text{left})$

" $v$  balanced" if  $\frac{1}{4} \leq \frac{z(v)}{n(v)} \leq \frac{3}{4}$



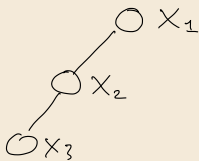
(otherwise "failure")

Any recursion tree for  $n$  elements can contain at most

$$\log_{3/4} \left( \frac{1}{n} \right) = \log_{4/3} (n) \leq 3.5 \ln(n) \quad n \cdot \left( \frac{3}{4} \right)^b \leq 1$$

balanced nodes

$X_i = [v_i \text{ is balanced}]$



left path shows  $P[\text{left path long}]$  small  
union bound over all  $n$  paths  $\rightarrow$  height small

$$X = X_1 + \dots + X_d \stackrel{D}{=} \text{Bin}(d, \frac{1}{2})$$

$$P[d \geq \overset{\text{large enough}}{c \cdot \ln(n)}] = P[X \leq 3.5 \ln(n)]$$

$$E[X] = \frac{1}{2} d$$

$$\mathbb{E}[X] > 3.5 \ln(n) \quad \text{for } d \geq 14 \ln(n)$$

$$\Rightarrow \text{apply Chernoff to } \mathbb{P}[|X - \mathbb{E}[X]| \geq 2\mathbb{E}[X]]$$

(spending you the detailed computation)

⚠  $X_i$  not independent and not  $\mathcal{B}(\frac{1}{2})$

because pivot rank has to be integral

$\rightarrow$  depends on  $u(r)$

can be fixed by clever patch to  $X_i$

$$\Rightarrow \text{height} \geq 42 \ln(n) \quad \text{w.p. } O(n^{-7.4})$$

# 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