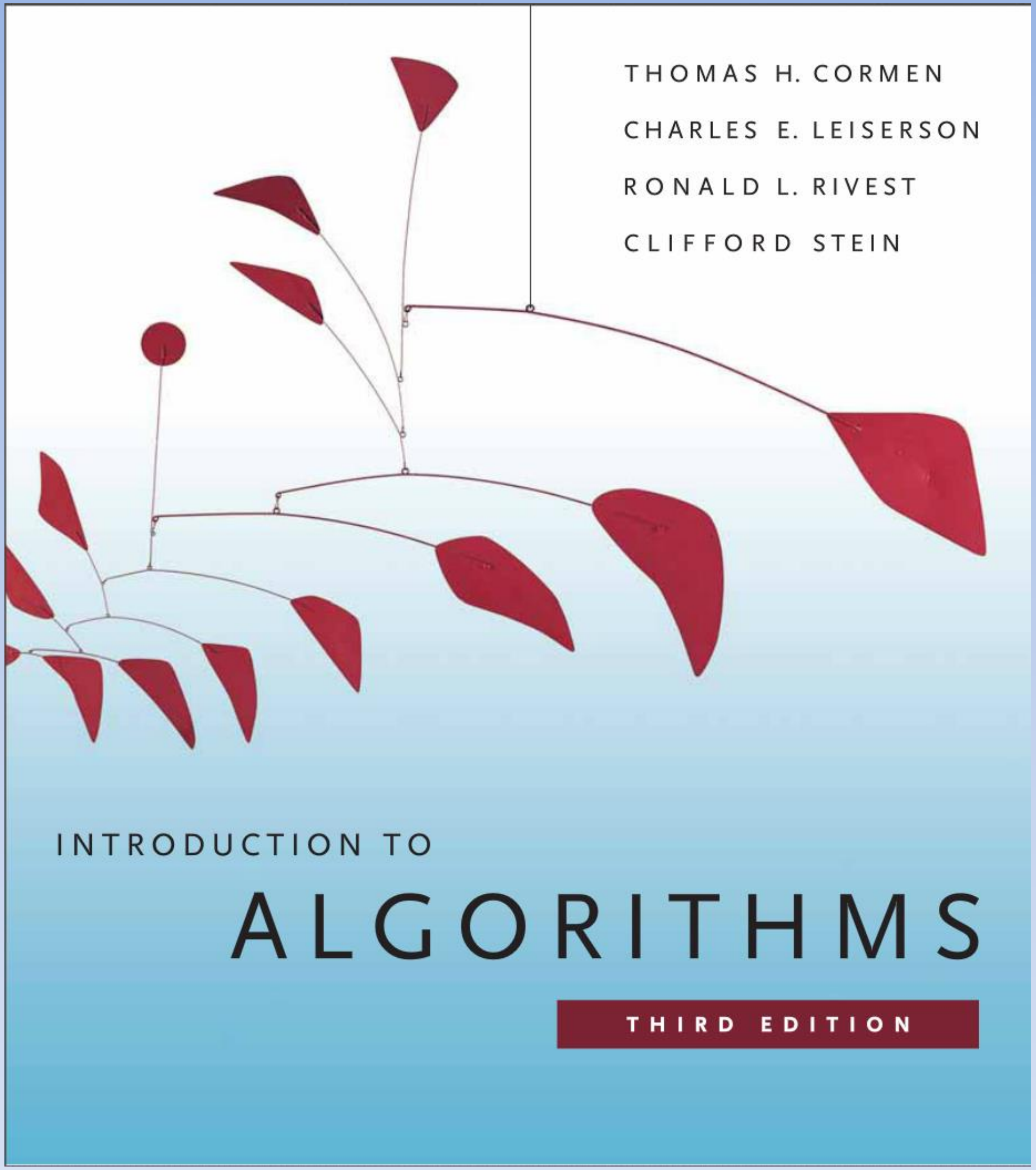

THOMAS H. CORMEN

CHARLES E. LEISERSON

RONALD L. RIVEST

CLIFFORD STEIN

INTRODUCTION TO

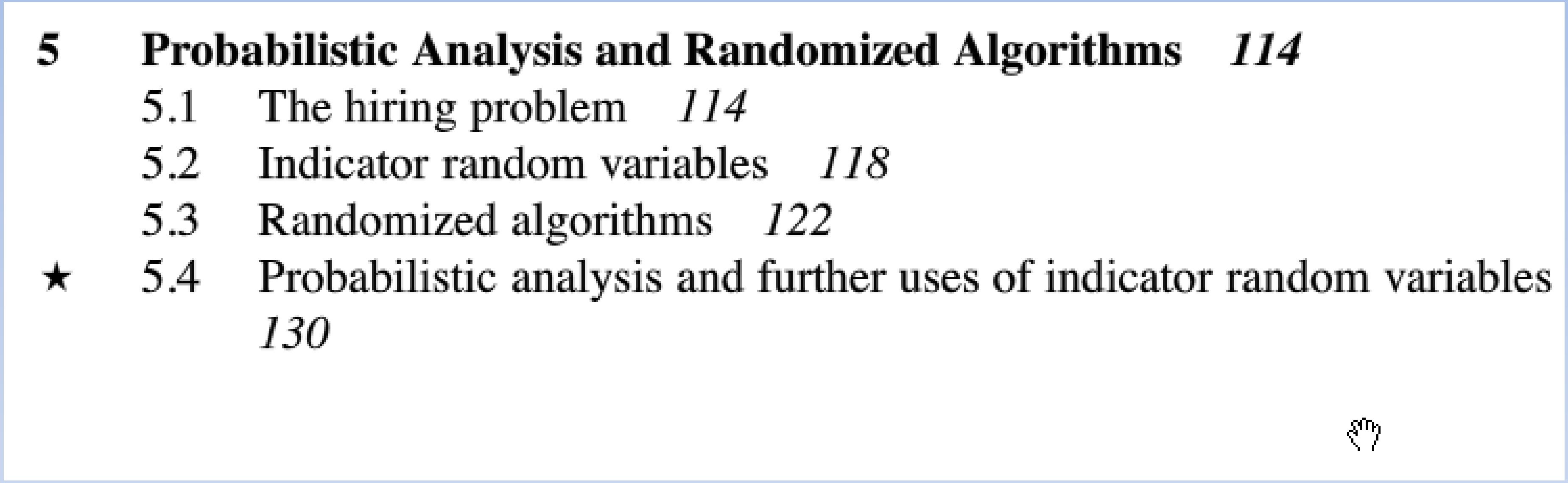
ALGORITHMS

THIRD EDITION

Chapter 5

Probabilistic Analysis & Randomized Algorithms

Chapter 5



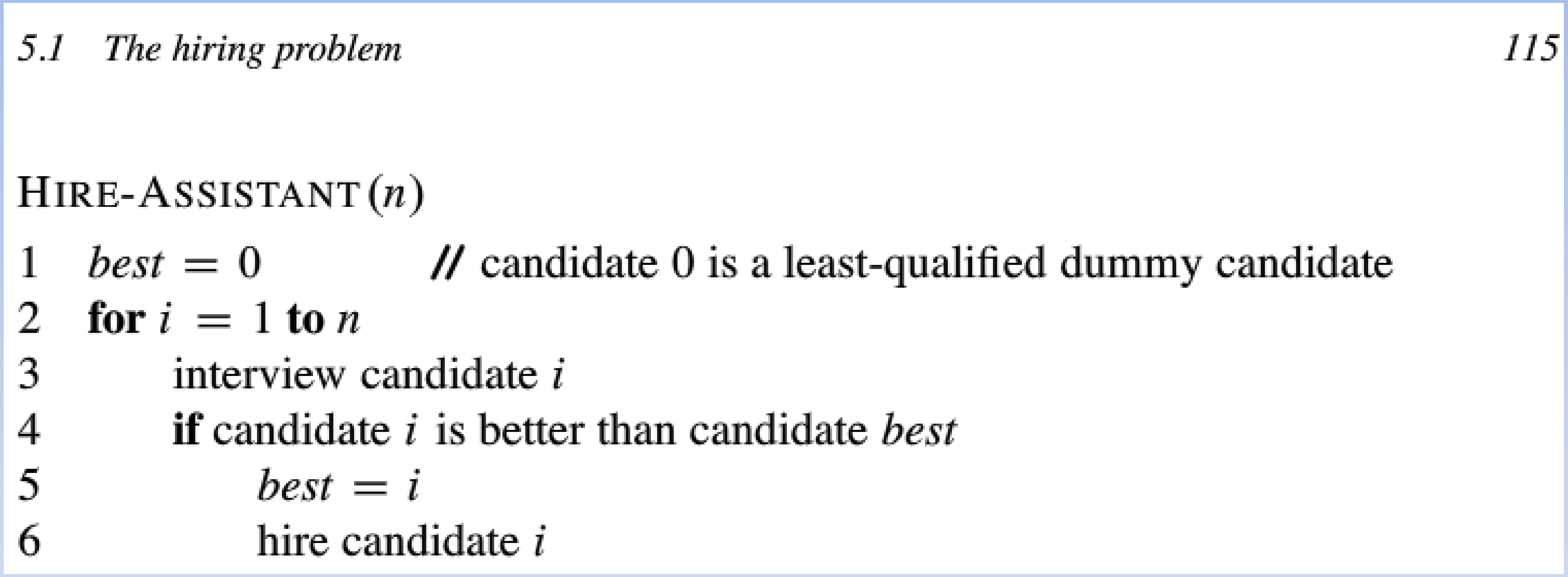
The Hiring Problem

- You need a new Assistant
- Finding an assistant on your own has failed.
  - You decide to go for help.
- You hire an Employment Agency to send you one candidate each day.
  - A small fee for each candidate interview
- After each interview you decide whether or not to hire candidate.

The Hiring Problem: Approach 1

- After each interview:
- IF: Candidate is no better qualified than current assistant THEN do nothing.
- ELSE IF: Candidate is better qualified than current assistant THEN:
  - FIRE current assistant
  - HIRE candidate!
- Estimate cost of strategy!

Hire-Assistant


5.1 *The hiring problem* 115

```
HIRE-ASSISTANT(n)
1  best = 0           // candidate 0 is a least-qualified dummy candidate
2  for i = 1 to n
3      interview candidate i
4      if candidate i is better than candidate best
5          best = i
6          hire candidate i
```

- Analysis will focus on costs from interviewing & hiring – Not Runtime!
- Analytic techniques the same!

Hire-Assistant & Winning!

- Model costs from algorithm
- Interviewing Costs are low: c_i
- Hiring costs are high: c_h
- Given:
  - n candidates interviewed
  - m candidates hired
- Cost associated with algorithm:
  - $O(c_i n + c_h m)$
- Model for common computation paradigm:
  - Finding the maximum or minimum value in a sequence
  - “WINNING” value maintained as items examined.

Hire-Assistant w/ Worst-case Analysis

- Consider behavior when candidates arrive in the worst possible order for algorithm
- IF Candidates arrive in increasing order of quality, THEN Everyone Hired!
  - Total Hiring Cost: $O(c_h n)$
- Obvious Question: What's likely in a typical or average case????

Probability & Science

- Which view of our universe
  - Complex Watch
  - Uncertainty

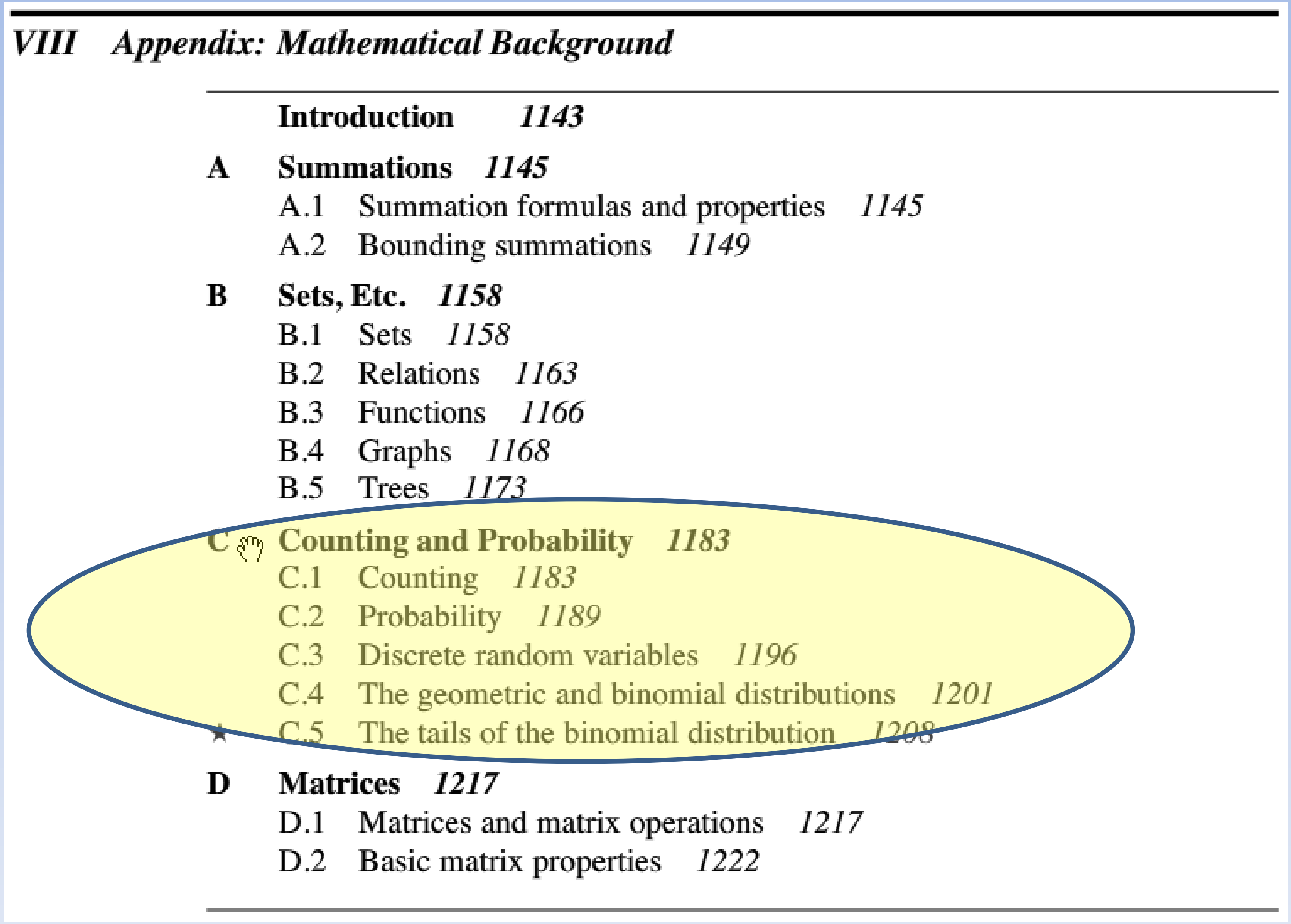
Average-Case Running Time w/ Probabilistic Analysis

- Typical case for Hire-Assistant requires:
  - Assumptions about distribution of inputs
  - Averages over distribution of possible inputs
- Running times derived labeled: Average-Case Running Time
- Probabilistic analysis is the use of probability in the analysis of problems.
  - Expected runtime given assumption about input distribution

Hire-Assistant: Random Order

- Assume Candidates arrive in a random order.
- FORMALLY:
  - – Candidates can be ranked imposing a total order.
  - – Each candidate from pool of n can be assigned unique number from 1 through n.
  - – Using rank(i) to denote rank of applicant, list <rank(1),rank(2),…,rank(n)> permutation of <1,2,…,n>
  - – Assume that for our candidate list, each list of ranks equally likely.
    - Ranks form a Uniform Random Permutation

WARNING
Some Probability Theory Required



VIII Appendix: Mathematical Background

Probability Concepts

- (discrete) random variable: is a function from a finite or countable sample space to the real numbers.
  - It associates a real number with each possible outcome of an experiment.
  - X_i = outcome of flipping the ith coin.
- Expectation of a random variable is:

$$E[X] = \sum_x x \cdot \Pr\{X = x\}$$

- Expectation of the sum of a set of random variables:
  - $X = X_1 + X_2 + \cdots + X_n$
  - $E[X] = E[X_1 + X_2 + \cdots + X_n] = E[X_1] + E[X_2] + \cdots + E[X_n]$
  - Useful!

Indicator Variables

- $$I\{A\} = \begin{cases} 1 & \text{if } A \text{ occurs ,} \\ 0 & \text{if } A \text{ does not occur.} \end{cases}$$

- Define indicator random variable X_H associated with a fair coin tossed coming up heads:

$$\begin{aligned} X_H &= I\{Heads\} \\ &= \begin{cases} 1 & \text{if Heads occurs ,} \\ 0 & \text{if Tails occurs} \end{cases} \end{aligned}$$

Hiring Problem

- X_i is the indicator random variable associated with the event in which the ith candidate is hired.

$$\begin{aligned} X_i &= I\{candidate\ i\ is\ hired\} \\ &= \begin{cases} 1 & if\ candidate\ i\ is\ hired\ , \\ 0 & if\ candidate\ i\ is\ not\ hired \end{cases} \end{aligned}$$

- Number candidates hired:
  - $X = X_1 + X_2 + \cdots + X_n$

Hiring Problem

- $E[X_i]$ = Pr{candidate i is hired}
- i is hired when better than all i-1 earlier candidates.
- Each candidate has equal chance of being the best,
  - since they are assumed to arrive randomly.
- Candidate i has probability of 1/i of being best
  - since each of the i candidates have an equal chance of being the best.
- SO:
  - $E[X_i] = 1/i$

Hiring Problem

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

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

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

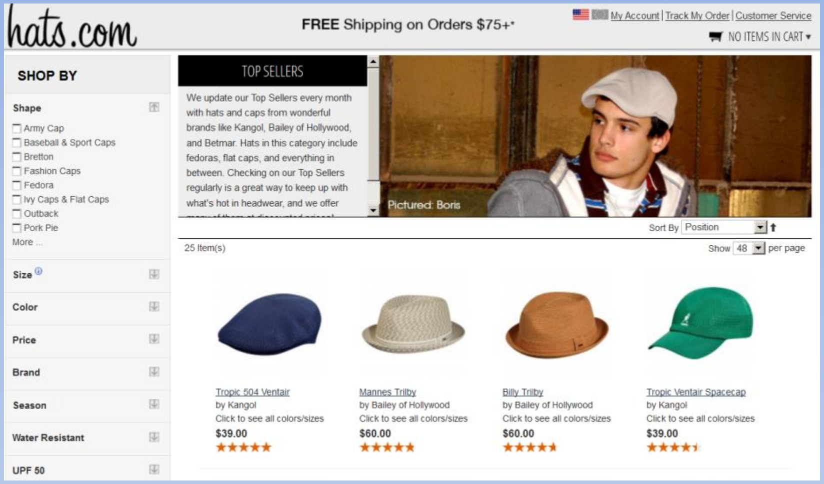
$$= \ln n + O(1)$$

- SO: Expected number of candidates hired is only $\ln n$
  – NOT n!

Hire-Assistant Average-Case Running Time

- Given a total of N candidates
- Expected number of candidates hired is $\ln n$
- Average-Case total hiring cost is $O(c_h \ln n)$
- Worst-Case hiring cost is $O(c_h n)$

The Hat-Check Problem



We update our Top Sellers every month with hats and caps from wonderful brands like Kangol, Bailey of Hollywood, and Betmar. Hats in this category include fedoras, flat caps, and everything in between. Checking on our Top Sellers regularly is a great way to keep up with what's hot in headwear, and we offer

- Each of n customers gives a hat to a hat-check person at a restaurant.
- The hat-check person gives the hats back to the customers in a random order.
- What is the expected number of customers who get back their own hat?
- Use indicator random variables to solve the hat-check problem.

The Hat-Check Problem

- Important Idea: We are looking for **fixed points** in a random permutations:
  - Values that are the same before and after a permutation.
  - Hats distributed to back to same person after random shuffle.
- Interested in the Expected Number of fixed points.
- Enumeration one approach for solving… but painstaking process!!

The Hat-Check Problem w/ Indicator Variables

- Indicator Variables:
- Define random variable X that equals the number of customers that get their own hat back:
  - Compute E[X]
- X_i = I{customer i gets his own hat}
- $X = X_1 + X_2 + \ldots + X_n$

The Hat-Check Problem w/ Indicator Variables

- Ordering of hats is random
  - each customer has probability 1/n of getting their own hat back.
  - Therefore: $E[X_i]=1/n$
- Therefore:

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

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

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

$$= 1$$

- We Expect One Person to get their own hat back!

The Hat-Check Problem w/ Indicator Variables

- Note that the indicator random variable are NOT independent!
  - If n=2 and X_1=1, then X_2 must also equal 1.
  - If n=2 and X_1=0, then X_2=0
- Despite Dependence:
  - $P(X_i=1) = 1/n$ for all i
  - Linearity of expectation holds!
- Indicator Variable technique available despite dependence!

Essentials Idea w/ Probabilistic Analysis

- With knowledge or assumptions about the distribution of inputs:
  - Develop expectation of performance results over distribution.
- Technique Requires a reasonable characterization of input distribution.

Randomized Algorithm

- Distribution of inputs might not be known, or not easy to model computationally.
- Solution:
  - Impose Input distribution in algorithm w/ Randomization!

Randomized Algorithm w/ Hiring Problem

- In some cases, probabilistic analysis can guide development of randomized algorithm
- THEN:
  - Instead of assuming distribution…
  - IMPOSE Distribution
- Impose Distribution in Hiring Problem by randomly permuting candidates!
  - Enforces equal likelihood permutation property!

Hire-Assistant:

- Algorithm is deterministic
  - Particular inputs always elicit same output
- Given the rank list: A_1=<1,2,3,4,5,6,7,8,9,10>
  - Hire-Assistant will hire a new assistant 10 times!
- Given the rank list: A_2=<10,9,8,7,6,5,4,3,2,1>
  - A new assistant is only hired in the first iteration!
- Given the rank list: A_3=<5, 2, 1, 8, 4, 7, 10, 9, 3, 6>
  - New Assistant hired 3 times!

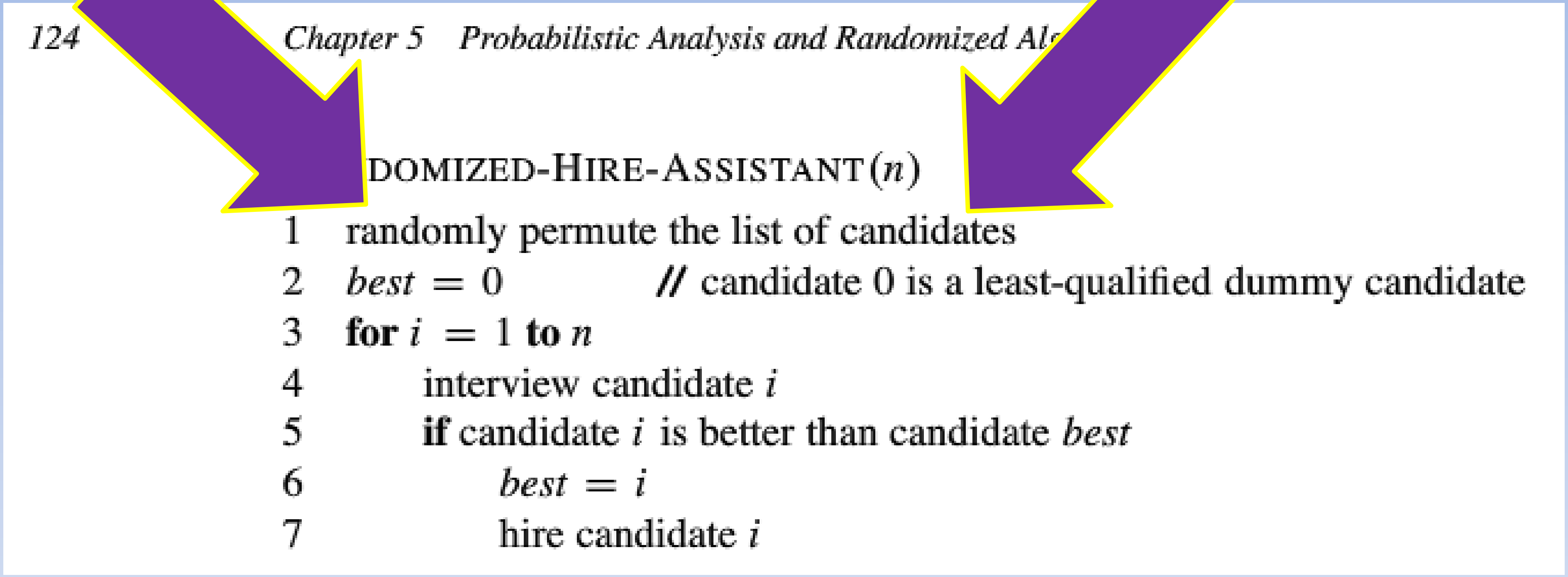
Hire-Assistant:

- Our algorithms has different performance with different inputs:
- Expensive Inputs:
  - A_1=<1,2,3,4,5,6,7,8,9,10>
- Inexpensive Inputs:
  - A_2=<10,9,8,7,6,5,4,3,2,1>
- Moderately Expensive
  - A_3=<5, 2, 1, 8, 4, 7, 10, 9, 3, 6>

Randomized Hire-Assistant

- First: Randomly Permute Candidate List
- Randomization in Algorithm:
  - NOT in Distribution
- NOW given candidate input: A_3
  - Performance Undetermined
    - First Time Permutation ➔ A_1
    - Second Time Permutation ➔ A_2
- Randomized Hire-Assistant behavior varies dependent upon random choices
  - NO ONE Input elicits its worst-case behavior!

Randomized-Hire-Assistant



124 Chapter 5 Probabilistic Analysis and Randomized Al

DOMIZED-HIRE-ASSISTANT(n)

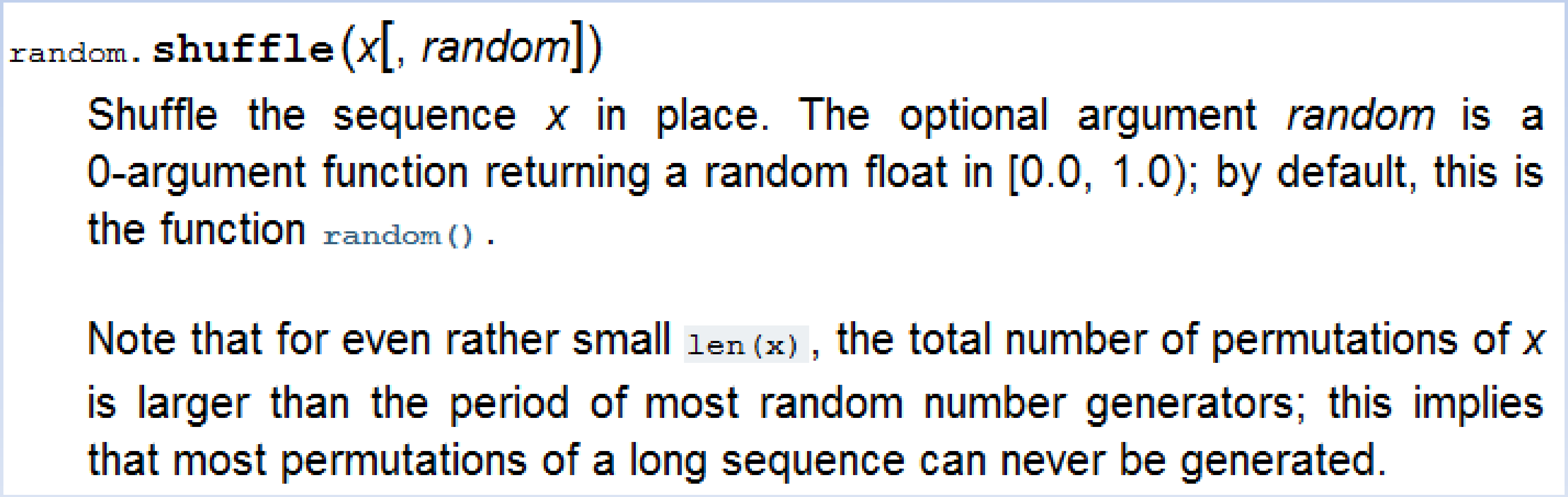
```
randomly permute the list of candidates
$best = 0$          // candidate 0 is a least-qualified dummy candidate
for $i = 1$ to $n$
    interview candidate $i$
    if candidate $i$ is better than candidate $best$
        $best = i$
        hire candidate $i$
```

Randomized Algorithm

- Algorithm is randomized if its behavior is determined in part by values produced by a random-number generator.
  - **Random(a, b)** returns an integer r, where r is between a and b and each of the b-a+1 possible value of r is equally likely
  - In practice, **Random** is implemented by a *pseudorandom-number generator*,
    - deterministic method returning numbers that “look” random and pass statistical tests.

Randomized Algorithms w/ Shuffling

- Many randomized algorithms randomize input by permuting input array.
- Capability built into Python:



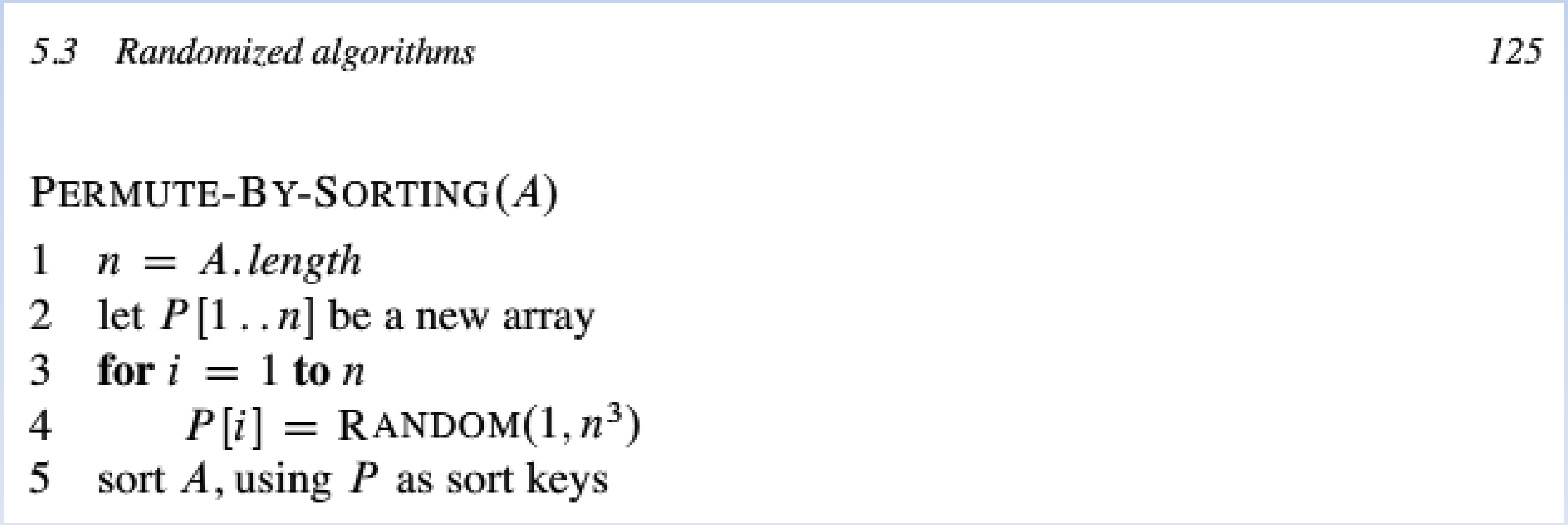
`random.`**`shuffle`**(*x*[, *random*])

Shuffle the sequence *x* in place. The optional argument *random* is a 0-argument function returning a random float in [0.0, 1.0); by default, this is the function `random()`.

Note that for even rather small `len(x)`, the total number of permutations of *x* is larger than the period of most random number generators; this implies that most permutations of a long sequence can never be generated.

Randomly Permuting Arrays w/ Permute-By-Sorting

- First method:
  - Assign each element of the array a random priority.
  - Sort element according to priorities.



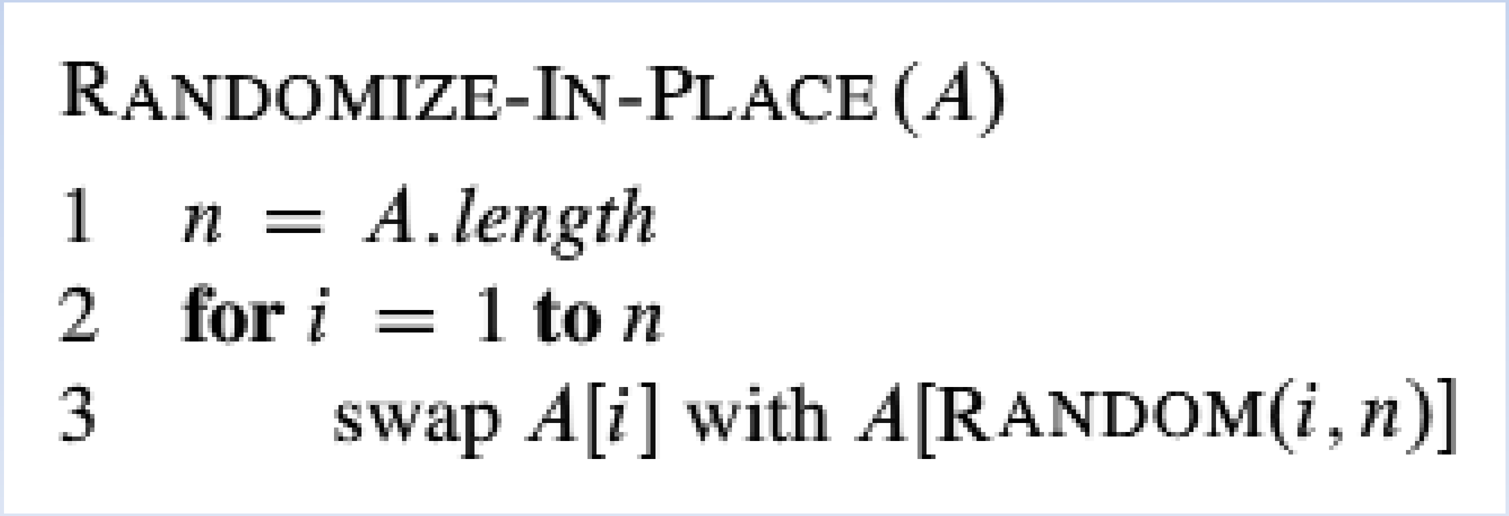
```
PERMUTE-BY-SORTING(A)
1  n = A.length
2  let P[1..n] be a new array
3  for i = 1 to n
4      P[i] = RANDOM(1, n^3)
5  sort A, using P as sort keys
```

Randomly Permuting Arrays w/ Permute-By-Sorting

- Time-consuming step is the sort.
- Using a comparison sort will require at least $n \lg n$
  - Using Radix Sort on a range of number from 0 to n^3-1 ?

Randomly Permuting Arrays w/ Randomize-In-Place

- A better method permutes the array in place.
  - Method used by Python
- Element A[i] is chosen at random from the elements between i and n.



```
RANDOMIZE-IN-PLACE(A)
n = A.length
for i = 1 to n
        swap A[i] with A[RANDOM(i, n)]
```

Randomly Permuting Arrays w/ Randomize-In-Place

- K-permutation on a set of n elements is a sequence containing k of the n elements with no repetitions:
  - n!/(n-k)!
- Uses loop invariant:
  - Just prior to the ith iteration of the for loop
  - For each possible (i-1)-permutation of the n elements
  - The subarray A[1..i-1] contains this (i-1)-permutation with probability (n-i+1)!/n!.

Randomly Permuting Arrays w/ Randomize-In-Place

- Initialization: 0 case is true w/ empty list
- Maintenance:
  - E_1 is the event with first i-1 iterations created (i-1)-permutation of $<x_1, \ldots, x_{i-1}>$
    - $P(E_1) = (n-i+1)!/n!$
  - E_2 is the event that i^{th} iteration puts x_i in position A[i]
  - i-permuation $<x_1, \ldots, x_i>$ occurs when E_1 and E_2 occur.
  - $Pr\{E_1 \cap E_2\} = Pr\{E_2 | E_1\}P\{E_1\}$

CONDITIONAL PROBABILITIES

Randomly Permuting Arrays w/ Randomize-In-Place

- $\Pr\{E_1 \cap E_2\} = \Pr\{E_2 \mid E_1\}P\{E_1\}$
- $\Pr\{E_1 \cap E_2\} = 1/(n-i+1)$ since x_i chosen randomly from n-i+1 values.

$$= \frac{1}{n-i+1} \cdot \frac{(n-i+1)!}{n!}$$

$$= \frac{(n-i)!}{n!}$$

Randomly Permuting Arrays w/ Randomize-In-Place

- Termination: i = n+1, so A[1..n] is given n-permutation with probability :

$$= \frac{(n-i+1)!}{n!}$$
$$= \frac{(0)!}{n!}$$
$$= \frac{1}{n!}$$

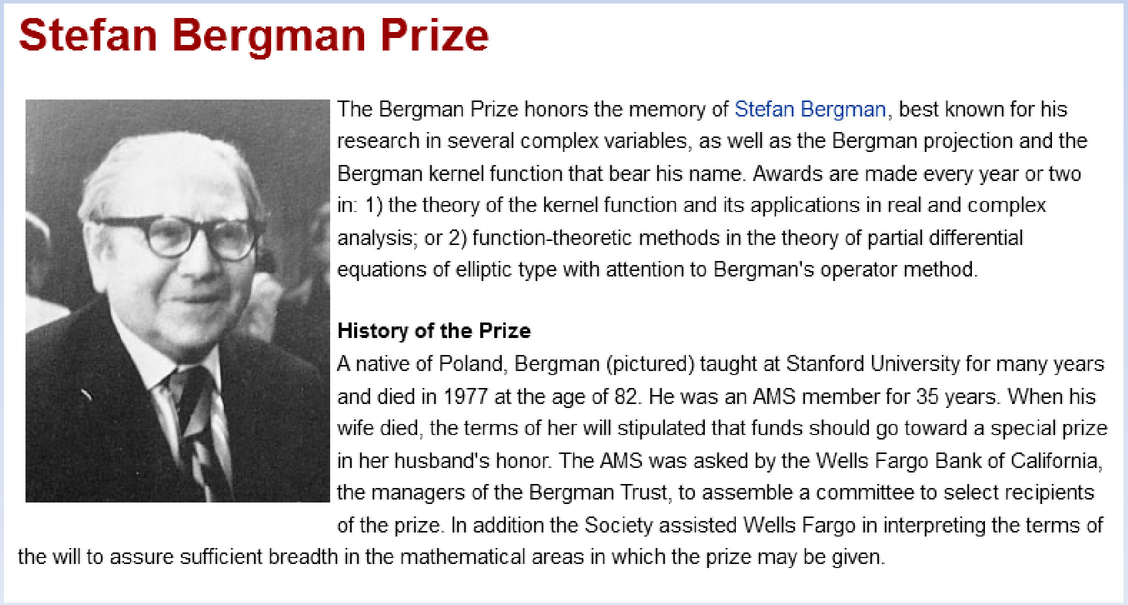
- Uniform Random Permutation!

Any Excuse is a Good Excuse for a Paradox!

- Any time is a good time for a paradox!

The Birthday Paradox!

- In 1939, Richard von Mises first proposed what we know today as the birthday problem.
  - He wondered, "How many people must be in a room before the probability that some share a birthday, ignoring the year and ignoring leap days, becomes at least 50 percent?"
  - We believe that this is one of the most explored probability problems in classrooms today.
- Arthur C. Clarke's novel *A Fall of Moondust*, published in 1961, contains a section where the main characters, trapped underground for an indefinite amount of time, are celebrating a birthday and find themselves discussing the validity of the Birthday problem.
  - As stated by a physicist passenger: "If you have a group of more than twenty-four people, the odds are better than even that two of them have the same birthday."
  - Eventually, out of 22 present, it is revealed that two characters share the same birthday,
  - **That birthday is May 23.**

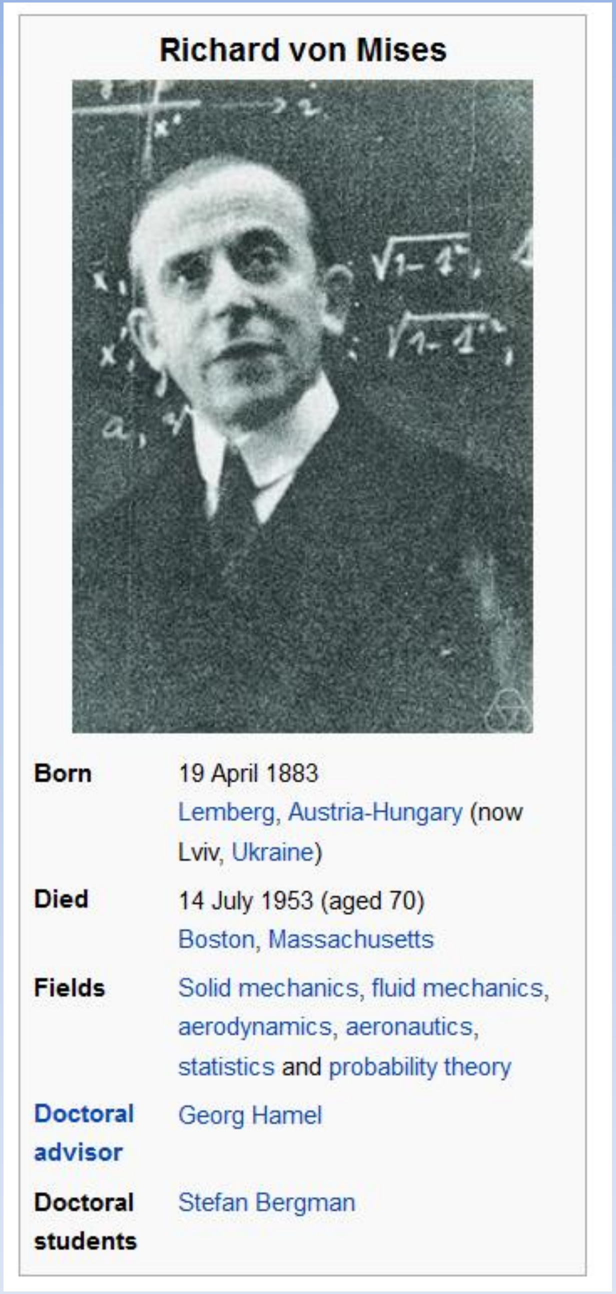


Stefan Bergman Prize

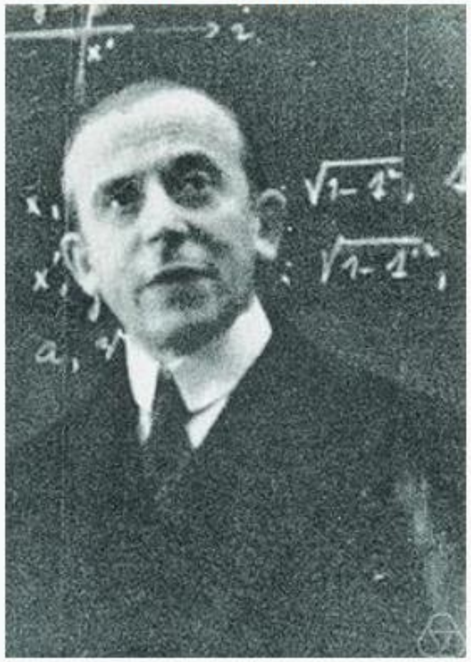
The Bergman Prize honors the memory of Stefan Bergman, best known for his research in several complex variables, as well as the Bergman projection and the Bergman kernel function that bear his name. Awards are made every year or two in: 1) the theory of the kernel function and its applications in real and complex analysis; or 2) function-theoretic methods in the theory of partial differential equations of elliptic type with attention to Bergman's operator method.

History of the Prize

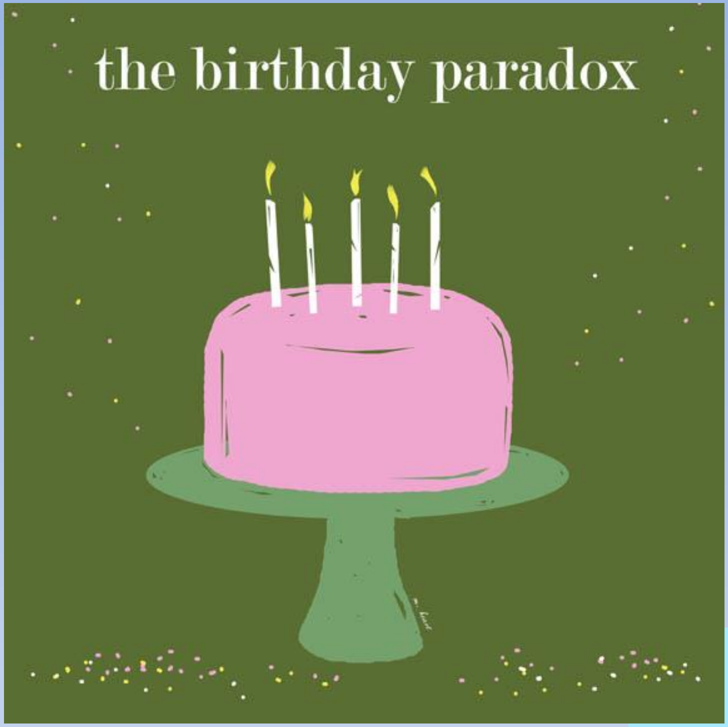
A native of Poland, Bergman (pictured) taught at Stanford University for many years and died in 1977 at the age of 82. He was an AMS member for 35 years. When his wife died, the terms of her will stipulated that funds should go toward a special prize in her husband's honor. The AMS was asked by the Wells Fargo Bank of California, the managers of the Bergman Trust, to assemble a committee to select recipients of the prize. In addition the Society assisted Wells Fargo in interpreting the terms of the will to assure sufficient breadth in the mathematical areas in which the prize may be given.



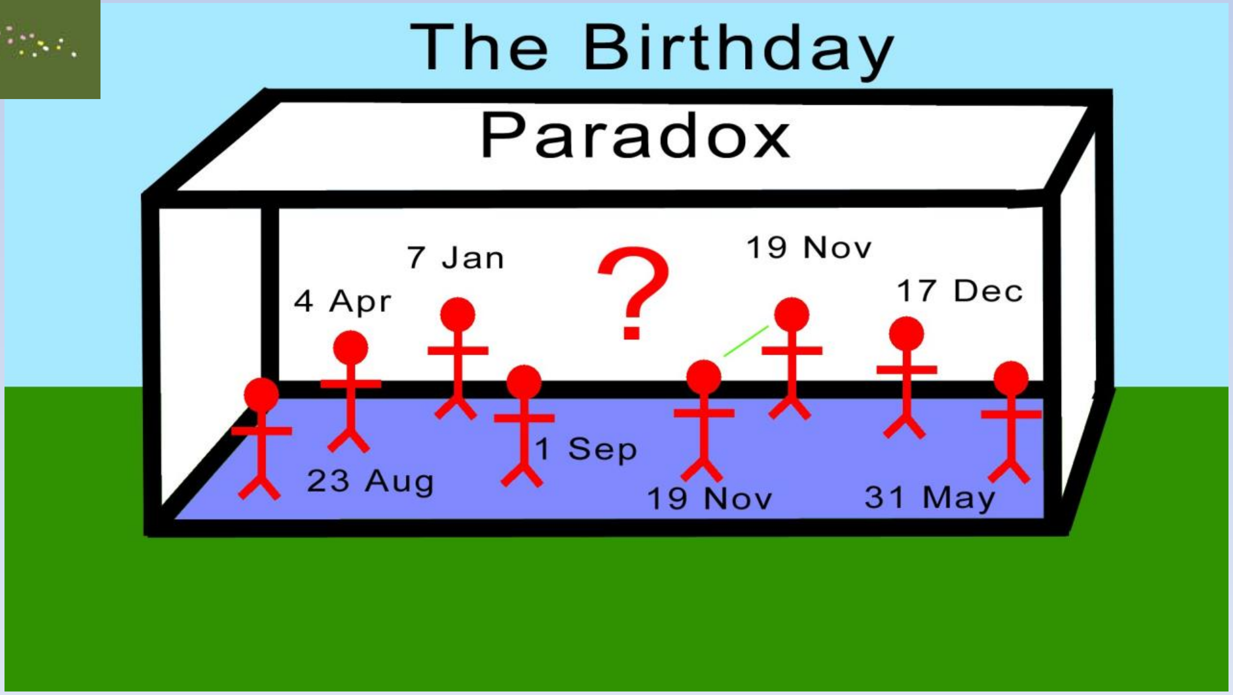
Richard von Mises



Born	19 April 1883 Lemberg, Austria-Hungary (now Lviv, Ukraine)
Died	14 July 1953 (aged 70) Boston, Massachusetts
Fields	Solid mechanics, fluid mechanics, aerodynamics, aeronautics, statistics and probability theory
Doctoral advisor	Georg Hamel
Doctoral students	Stefan Bergman

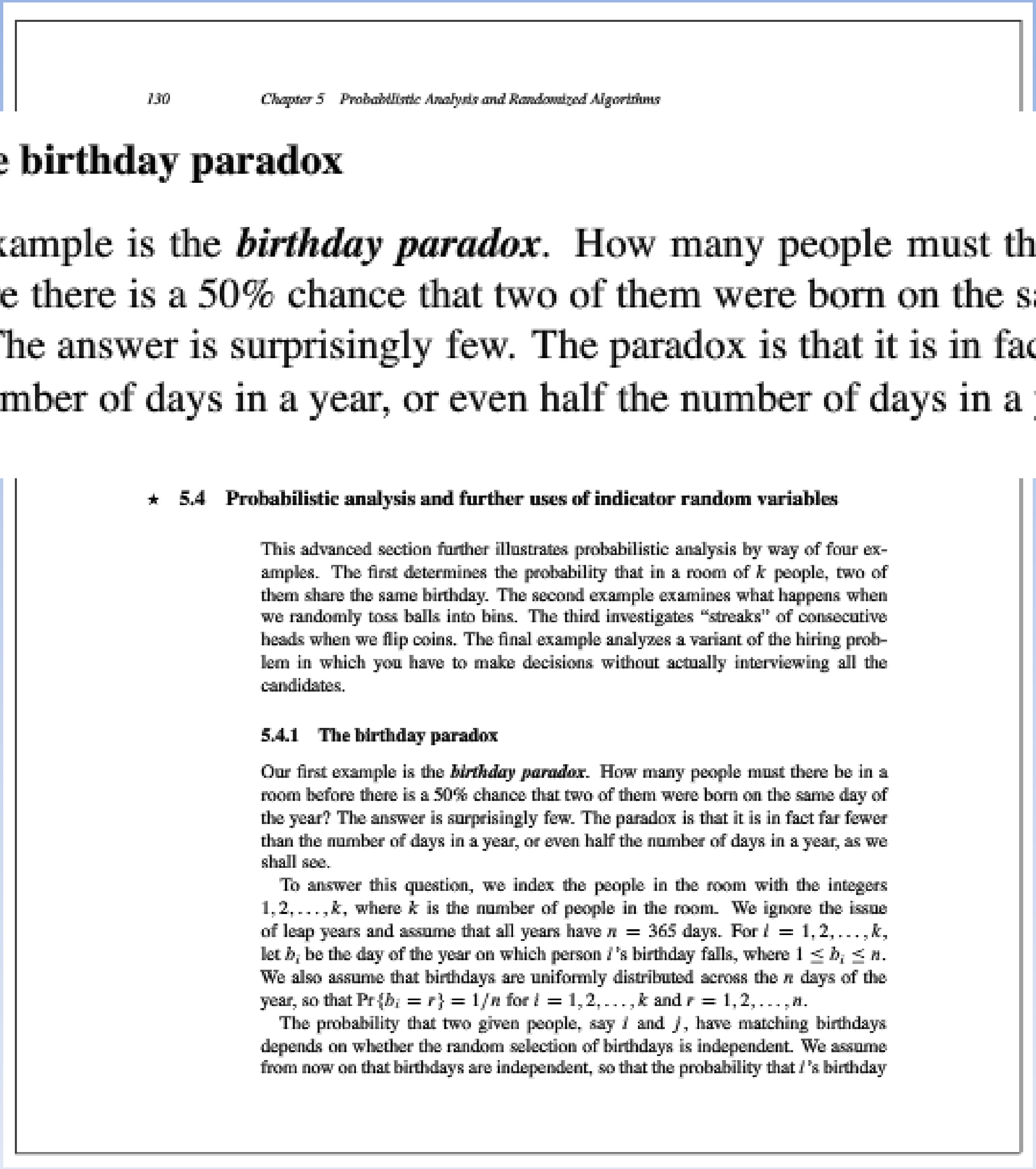

the birthday paradox

JULY 13TH
JULY 13TH


The Birthday Paradox
7 Jan
?
19 Nov
4 Apr
17 Dec
23 Aug
1 Sep
19 Nov
31 May

5.4.1 The birthday paradox

Our first example is the ***birthday paradox***. How many people must there be in a room before there is a 50% chance that two of them were born on the same day of the year? The answer is surprisingly few. The paradox is that it is in fact far fewer than the number of days in a year, or even half the number of days in a year, as we shall see.



★ 5.4 Probabilistic analysis and further uses of indicator random variables

This advanced section further illustrates probabilistic analysis by way of four examples. The first determines the probability that in a room of k people, two of them share the same birthday. The second example examines what happens when we randomly toss balls into bins. The third investigates "streaks" of consecutive heads when we flip coins. The final example analyzes a variant of the hiring problem in which you have to make decisions without actually interviewing all the candidates.

5.4.1 The birthday paradox

Our first example is the ***birthday paradox***. How many people must there be in a room before there is a 50% chance that two of them were born on the same day of the year? The answer is surprisingly few. The paradox is that it is in fact far fewer than the number of days in a year, or even half the number of days in a year, as we shall see.

To answer this question, we index the people in the room with the integers $1, 2, \ldots, k$, where k is the number of people in the room. We ignore the issue of leap years and assume that all years have $n = 365$ days. For $i = 1, 2, \ldots, k$, let b_i be the day of the year on which person i's birthday falls, where $1 \leq b_i \leq n$. We also assume that birthdays are uniformly distributed across the n days of the year, so that $\Pr\{b_i = r\} = 1/n$ for $i = 1, 2, \ldots, k$ and $r = 1, 2, \ldots, n$.

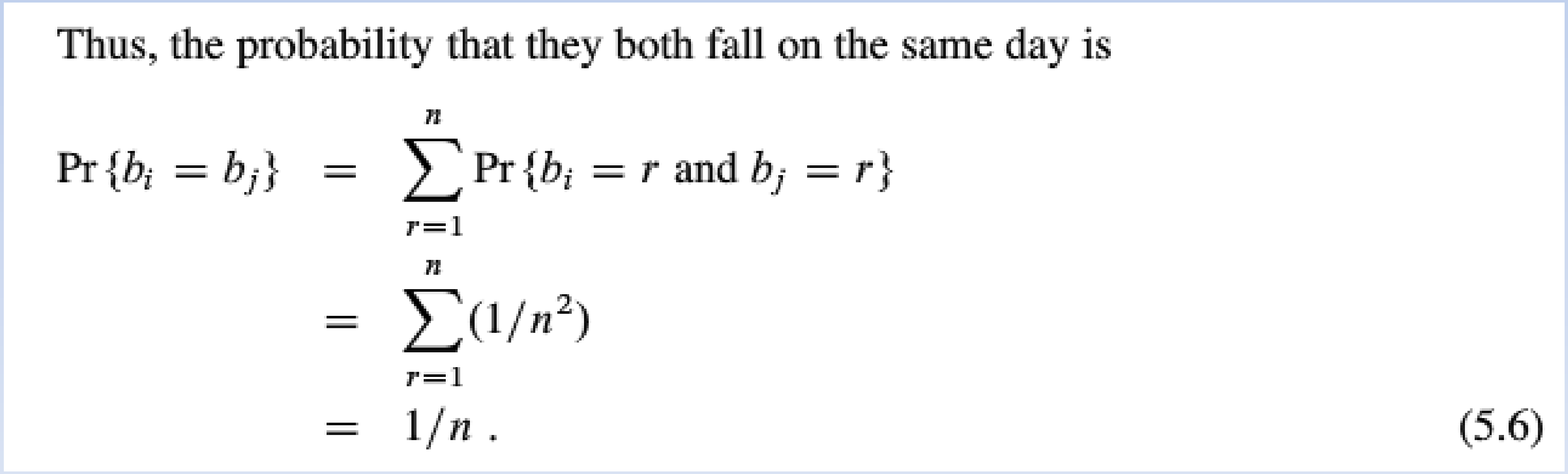
The probability that two given people, say i and j, have matching birthdays depends on whether the random selection of birthdays is independent. We assume from now on that birthdays are independent, so that the probability that i's birthday

Preliminaries

- Index people w/ integers
  - 1, 2, …, k
- Ignore Leap Years w/ days
  - 1, 2, …, n=365
- b_i is i's birthday!
  - $1 \leq b_i \leq n$
- Birthdays are uniformly distributed across all days:
  - $P\{b_i=r\} = \frac{1}{n}$ for i=1,…,k and r = 1,…, n

Birthdays are Independent

- Therefore the probability that two people have the same birthday r:
  - $P\{b_i=r \text{ \& } b_j=r\} = P\{b_i=r\} * P\{b_j=r\} = 1/n^2$



Thus, the probability that they both fall on the same day is

$$\begin{aligned} \Pr\{b_i = b_j\} &= \sum_{r=1}^{n} \Pr\{b_i = r \text{ and } b_j = r\} \\ &= \sum_{r=1}^{n} (1/n^2) \\ &= 1/n . \end{aligned} \tag{5.6}$$

More Intuitively

- The probability that two people have the same birthday can be stated:
  - Give person i has birthday r, what is the probability that person j has the same birthday.
    - 1/n

How Likely 2 out of k have same birthday?

- As is frequently the case… Lets change the question.
- How likely k people have DISTINCT birthdays??
  - – Now: 2 out of k are the same otherwise!
    - 1 – (Probability k are DISTINCT)

- The event that k people have distinct birthdays is:

$$B_k = \bigcap_{i=1}^{k} A_i$$

  - where A_i is the event that person i's birthday is different from person j's for all j<i

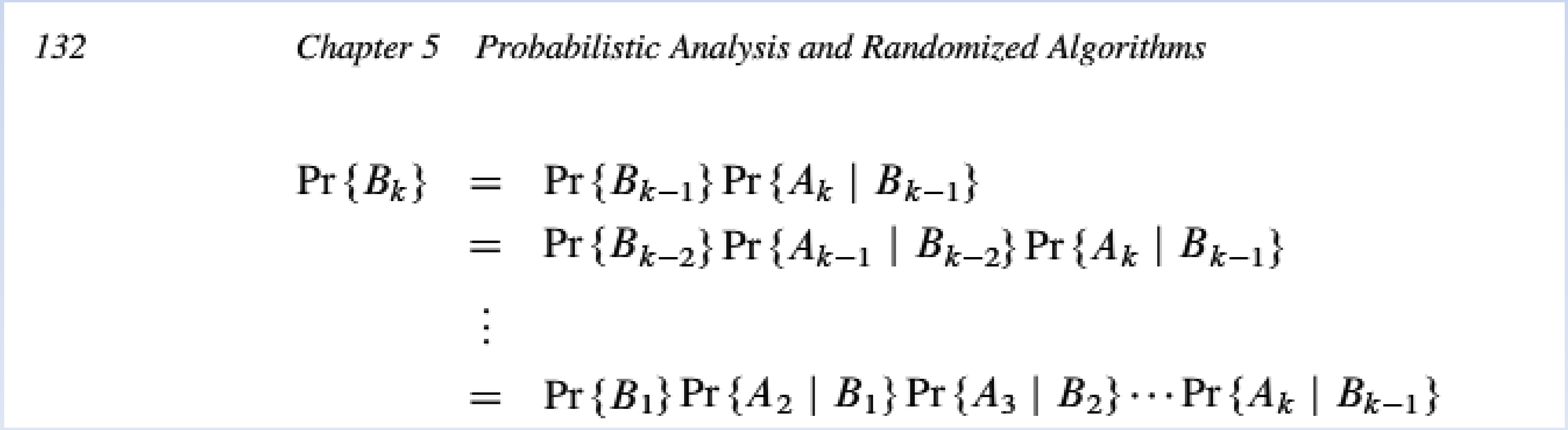
- $B_k = A_k \cap B_{k-1}$
  - Event with k people have distinct birthdays is the intersection of the events
    - k-1 people with distinct birthday
    - person k having a birthday distinct from the k-1 others.
- We All Know:
  - $P\{A \cap B\} = P\{A\}P\{B|A\}$
- SO: $P\{A_k \cap B_{k-1}\} = P\{B_{k-1}\}P\{A_k|B_{k-1}\}$
- SO: $P\{B_k\} = P\{B_{k-1}\}P\{A_k|B_{k-1}\}$

$P\{B_k\} = P\{B_{k-1}\}P\{A_k|B_{k-1}\}$

- Intuitively :
  - The probability that our k people have distinct birthday equals the product of:
    - probability that k-1 are unique
    - probability the $b_k \neq b_i, for\ i \leq k-1$

$P\{B_k\} = P\{B_{k-1}\}P\{A_k|B_{k-1}\}$

- $P\{B_1\} = P\{A_1\} = 1$
  - 1 person has to have a DISTINCT birthday!
- $P\{A_k|B_{k-1}\} = \frac{n-(k-1)}{n} = \frac{(n-k+1)}{n}$
  - $n-(k-1)$ days out of the n=365 days in the year, are not yet taken for birthdays!



$$\begin{aligned}
\Pr\{B_k\} &= \Pr\{B_{k-1}\}\Pr\{A_k \mid B_{k-1}\} \\
&= \Pr\{B_{k-2}\}\Pr\{A_{k-1} \mid B_{k-2}\}\Pr\{A_k \mid B_{k-1}\} \\
&\vdots \\
&= \Pr\{B_1\}\Pr\{A_2 \mid B_1\}\Pr\{A_3 \mid B_2\}\cdots\Pr\{A_k \mid B_{k-1}\}
\end{aligned}$$

$$\begin{aligned}
\Pr\{B_k\} &= \Pr\{B_{k-1}\}\Pr\{A_k \mid B_{k-1}\} \\
&= \Pr\{B_{k-2}\}\Pr\{A_{k-1} \mid B_{k-2}\}\Pr\{A_k \mid B_{k-1}\} \\
&\;\vdots \\
&= \Pr\{B_1\}\Pr\{A_2 \mid B_1\}\Pr\{A_3 \mid B_2\}\cdots\Pr\{A_k \mid B_{k-1}\} \\
&= 1\cdot\left(\frac{n-1}{n}\right)\left(\frac{n-2}{n}\right)\cdots\left(\frac{n-k+1}{n}\right) \\
&= 1\cdot\left(1-\frac{1}{n}\right)\left(1-\frac{2}{n}\right)\cdots\left(1-\frac{k-1}{n}\right).
\end{aligned}$$

- NOW we use:
  - $1 + x \leq e^x$
    - REMEMBER: e^x =

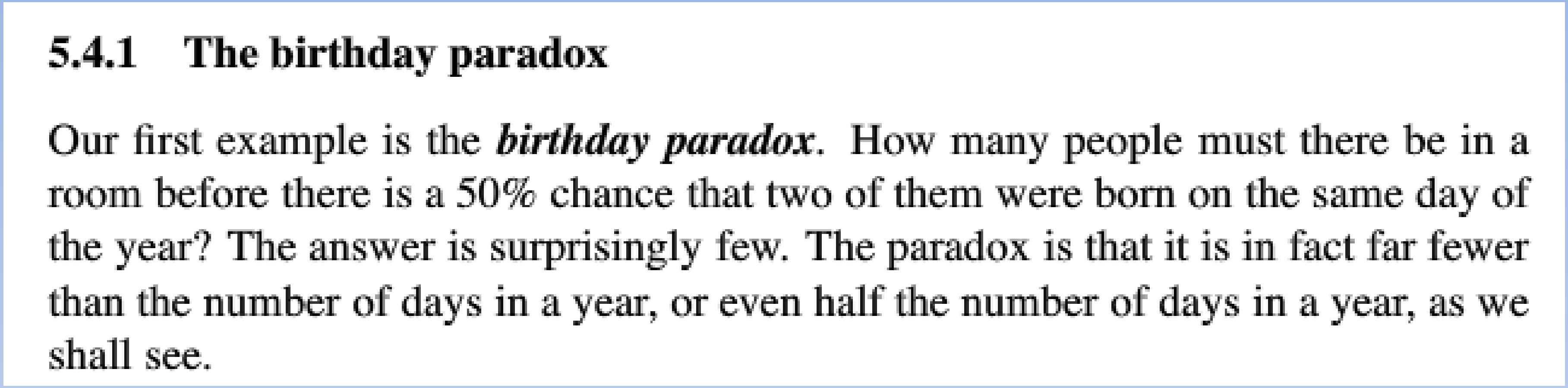
$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots = \sum_{i=0}^{\infty} \frac{x^i}{i!}\,, \qquad (3.11)$$

$$\begin{aligned}
\Pr\{B_k\} &= \Pr\{B_{k-1}\}\Pr\{A_k \mid B_{k-1}\} \\
&= \Pr\{B_{k-2}\}\Pr\{A_{k-1} \mid B_{k-2}\}\Pr\{A_k \mid B_{k-1}\} \\
&\vdots \\
&= \Pr\{B_1\}\Pr\{A_2 \mid B_1\}\Pr\{A_3 \mid B_2\}\cdots\Pr\{A_k \mid B_{k-1}\}
\end{aligned}$$

$$\Pr\{B_k\} \le e^{-1/n}e^{-2/n}\cdots e^{-(k-1)/n}$$

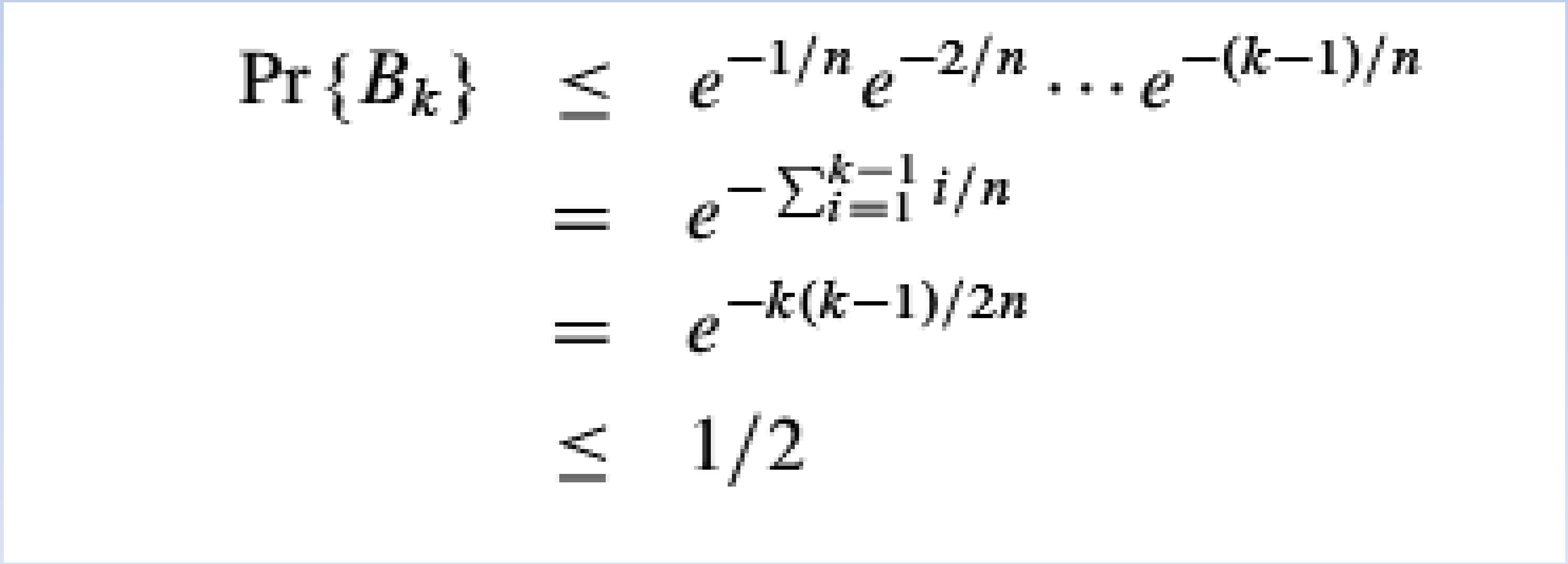
$$= e^{-\sum_{i=1}^{k-1} i/n}$$

$$\sum_{i=1}^{k}\frac{i}{n} = \frac{1}{n}\sum_{i=1}^{k} i = \frac{(i+1)i}{2n}$$



5.4.1 The birthday paradox

Our first example is the ***birthday paradox***. How many people must there be in a room before there is a 50% chance that two of them were born on the same day of the year? The answer is surprisingly few. The paradox is that it is in fact far fewer than the number of days in a year, or even half the number of days in a year, as we shall see.



$$\begin{aligned}
\Pr\{B_k\} &\le e^{-1/n}e^{-2/n}\cdots e^{-(k-1)/n} \\
&= e^{-\sum_{i=1}^{k-1} i/n} \\
&= e^{-k(k-1)/2n} \\
&\le 1/2
\end{aligned}$$

$$\begin{aligned}
\Pr\{B_k\} &\le e^{-1/n} e^{-2/n} \cdots e^{-(k-1)/n} \\
&= e^{-\sum_{i=1}^{k-1} i/n} \\
&= e^{-k(k-1)/2n} \\
&\le 1/2
\end{aligned}$$

- True when:

$$\frac{-k(k-1)}{2n} \le \ln\frac{1}{2}$$

$$k^2 - k - 2n\ln 2 \ge 0$$

$$1k^2 - 1k - 2n\ln\frac{1}{2} \ge 0$$

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = \frac{1 \pm \sqrt{1 + 4 * 2\text{n} * \ln(2)}}{2} = 22.99994315123396$$

where n=365 k ≥ 22.99994315123396

One more time w/ Indicator Variables

$$\begin{aligned} X_{ij} &= \mathrm{I}\{\text{person } i \text{ and person } j \text{ have the same birthday}\} \\ &= \begin{cases} 1 & \text{if person } i \text{ and person } j \text{ have the same birthday}, \\ 0 & \text{otherwise}. \end{cases} \end{aligned}$$

- Probability two people have matching birthdays is 1/n.

$$\begin{aligned} \mathrm{E}[X_{ij}] &= \Pr\{\text{person } i \text{ and person } j \text{ have the same birthday}\} \\ &= 1/n . \end{aligned}$$

Indicator Variables

- X is random variable counting the number of pairs of distinct birthdays.

$$X = \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} X_{ij}$$

Indicator Variables

- X is random variable counting the number of pairs of distinct birthdays.

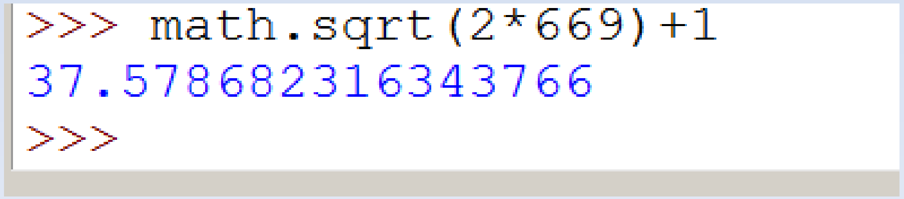
$$X = \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} X_{ij}$$

$$\begin{aligned}
\mathrm{E}[X] &= \mathrm{E}\left[\sum_{i=1}^{k-1} \sum_{j=i+1}^{k} X_{ij}\right] \\
&= \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \mathrm{E}[X_{ij}] \\
&= \binom{k}{2} \frac{1}{n} \\
&= \frac{k(k-1)}{2n} .
\end{aligned}$$

Indicator Variables

- We expect one pair of mathcing birthdays when $k(k-1) \geq 2n$
- $k \geq \sqrt{2n} + 1$ insures E[X] =1
  - With n=365, k $\geq$ 28
- On mars where years are 669 days we require:

- We expect one pair of mathcing birthdays when $k(k-1) \geq 2n$
- $k \geq \sqrt{2n} + 1$ insures E[X] =1
  - With n=365, k $\geq$ 28
- On mars where years are 669 days we require:



```
>>> math.sqrt(2*669)+1
37.578682316343766
>>>
```