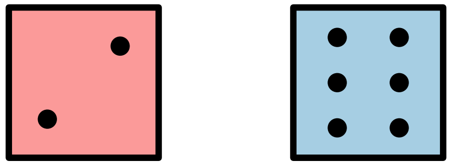
Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

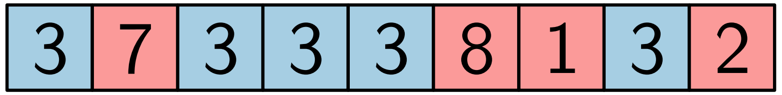
Advanced Algorithms

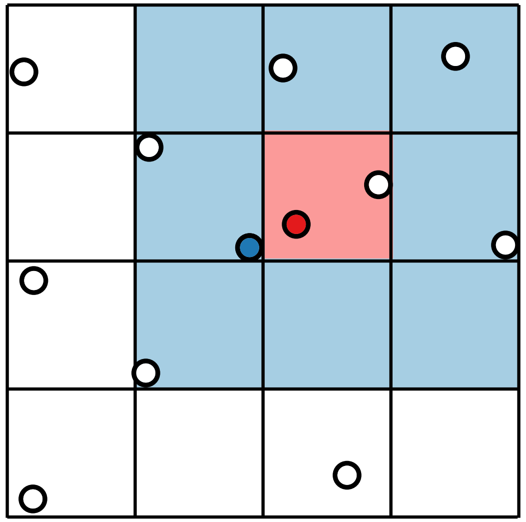
Randomized Algorithms

An introduction

Johannes Zink · WS22







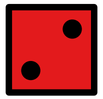
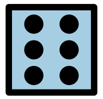
Basic Definitions

A **discrete probability space** (Ω, Pr) is used to model random experiments.

Ω is a countable set of **elementary events** ($=$ outcomes of the experiment).

$\mathsf{Pr} : \Omega \longrightarrow [0,1]$ assigns a **probability** $\mathsf{Pr}(\omega)$ to each $\omega \in \Omega$ s.t. $\sum_{\omega\in\Omega} \mathsf{Pr}(\omega) = 1$.

A set $A \subseteq \Omega$ is called **event**. The **probability** of A is $\mathsf{Pr}[A] = \sum_{\omega\in A} \mathsf{Pr}(\omega)$.

Example. Rolling a red and a blue fair six-sided die.

$\Omega = \{(1,1),(1,2),(1,3),\ldots,(6,6)\}$, $\quad \mathsf{Pr}((i,j)) = \frac{1}{6}\cdot\frac{1}{6} = \frac{1}{36}$ for each $(i,j) \in \Omega$

$A = \{(1,1),(2,2),(3,3),(4,4),(5,5),(6,6)\} =$ rolling a double

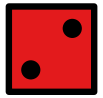
$\mathsf{Pr}(A) = 6 \cdot \frac{1}{36} = \frac{1}{6}$

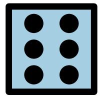
Basic Definitions

A **random variable** is a function $X : \Omega \longrightarrow \mathbb{R}$.

For each $x \in \mathbb{R}$ we define an event $(X = x) = \{\omega \in \Omega \mid X(\omega) = x\}$.

The **expected value** of X is $\mathrm{E}[X] = \sum_{x \in X(\Omega)} x \cdot \Pr[(X = x)]$.





Example. Rolling a red and a blue fair six-sided die.

$$\Omega = \{(1,1),(1,2),(1,3),\ldots,(6,6)\}, \quad \Pr((i,j)) = \frac{1}{6} \cdot \frac{1}{6} = \frac{1}{36} \text{ for each } (i,j) \in \Omega$$

$$X((i,j)) = \max\{i,j\}$$

$$\Pr[(X=1)] = \frac{1}{36}, \Pr[(X=2)] = \frac{3}{36}, \Pr[(X=3)] = \frac{5}{36}, \ldots, \Pr[(X=6)] = \frac{11}{36}$$

$$\mathrm{E}[X] = 1 \cdot \frac{1}{36} + 2 \cdot \frac{3}{36} + 3 \cdot \frac{5}{36} + \cdots + 6 \cdot \frac{11}{36} \approx 4.5$$

Linearity of Expectation

For each set of random variables $X_1, X_2, \ldots, X_n : \Omega \longrightarrow \mathbb{R}$,
we define a random variable $(X_1 + X_2 + \cdots + X_n) : \Omega \longrightarrow \mathbb{R}$ with
$(X_1 + X_2 + \cdots + X_n)(\omega) = X_1(\omega) + X_2(\omega) + \cdots + X_n(\omega)$ for each $\omega \in \Omega$.

Linearity of expectation: $\mathsf{E}[(X_1 + X_2 + \cdots + X_n)] = \mathsf{E}[X_1] + \mathsf{E}[X_2] + \cdots + \mathsf{E}[X_n]$

Proof of correctness:

$$\begin{aligned}\mathsf{E}[(X_1 + X_2 + \cdots + X_n)] &= \sum_{\omega \in \Omega} \Pr[\omega] \cdot \sum_{i=1}^{n} X_i(\omega) = \sum_{\omega \in \Omega} \sum_{i=1}^{n} \Pr[\omega] \cdot X_i(\omega) \\ &= \sum_{i=1}^{n} \sum_{\omega \in \Omega} \Pr[\omega] \cdot X_i(\omega) = \sum_{i=1}^{n} \mathsf{E}[X_i]\end{aligned}$$

Using Indicator Random Variables (I)

Let A be an array filled with n pairwise distinct integers.

How often is the maximum m updated? $\mathcal{O}(n)$ times

$\textsc{FindMax}(A)$

$m := A[1]$
for $i = 2, 3, \ldots, n$
  if $A[i] > m$
    $m := A[i]$
return m

Assume that the integers in A are randomly permuted.

Let X denote the random varibale that counts the number of times m is updated. We define n random variables

$$X_i = \begin{cases} 1 \text{ , if } m \text{ is updated in iteration } i \\ 0 \text{ , otherwise} \end{cases}$$

indicator random variable

Observation. $X = (X_1 + X_2 + \cdots + X_n)$

$$\Pr[(X_i = 1)] = \frac{1}{i} \quad \Rightarrow \quad \mathrm{E}[X_i] = 0 + 1 \cdot \frac{1}{i} = \frac{1}{i}$$

H_n is the n-th harmonic number; $\ln(n+1) \leq H_n \leq \ln(n) + 1.$

$$\mathrm{E}[X] = \mathrm{E}[X_1] + \mathrm{E}[X_2] + \cdots + \mathrm{E}[X_n] = 1 + \frac{1}{2} + \cdots + \frac{1}{n} = H_n \in \Theta(\log n)$$

linearity of expectation

Playing until You Win

A **Bernoulli experiment** has only two outcomes $\Omega = \{\text{failure}, \text{success}\}$.

Let $p = \Pr(\text{success})$ be the **success probability**.

$\Rightarrow q = \Pr(\text{failure}) = 1 - p$ is the **failure probability**.

Suppose we repeat such an experiment multiple times.
Assume the outcomes are independent from each other. $\longleftarrow$ This experiment has a **geometric distribution**

Let X be the random variable that counts the number of rounds until we succeed for the first time.

$$\Pr[(X = j)] = q^{j-1} p$$

geometric series

$$\Rightarrow E[X] = \sum_{j=0}^{\infty} j \cdot q^{j-1} p = p \cdot \frac{d}{dq} \left(\sum_{j=0}^{\infty} q^j \right) = p \cdot \frac{d}{dq} \left(\frac{1}{1-q} \right) = p \cdot \frac{1}{(1-q)^2} = p \cdot \frac{1}{p^2} = \frac{1}{p}$$

Using Indicator Random Variables (II)

Each time you buy groceries at your local supermarket for more than 10 Euro you get a random toy for free. The number of pairwise distinct toys is n.

How often do you have to shop to obtain a toy of each type? $\rightarrow$ random variable X

Observation. Suppose you have already obtained $i-1$ types of toys. Now you continue shopping until you receive a new type of toy. This experiment has a geometric probability distribution! The success probability is $p_i = \frac{n-(i-1)}{n}$.

$X_i =$ number of times you have to shop to obtain the i-th type of toy when you already have $i-1$ types of toys

$\Rightarrow \mathsf{E}[X_i] = \frac{1}{p_i} = \frac{n}{n-(i-1)}$

$\Rightarrow \mathsf{E}[X] = \mathsf{E}[X_1] + \mathsf{E}[X_2] + \cdots + \mathsf{E}[X_n] = n \cdot (\frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{2} + 1) \in \Theta(n \log n)$

Finding a Large Number

Given: An array A of n pairwise distinct natural numbers.

Task: Determine an integer $A[j]$ that is at least as large as the median.

Deterministic approach: Go through all elements, return maximum.

(Actually, it suffices to go through $\lfloor n/2 \rfloor + 1$ elements.)

runtime $\Theta(n)$

Randomized approach:

$\textsc{FindLarge}(A, k \in \mathbb{N})$

$\ell := 0$
for $i = 1, 2, \ldots, k$
  randomly choose $r \in \{1, 2, \ldots, n\}$
  if $A[r] > \ell$
    $\ell := A[r]$
return ℓ

$\textsc{FindLarge}$ has error probability $\leq \frac{1}{2^k}$.

Set $k = c \log_2 n$ for some constant $c > 1$.

$\Rightarrow$ Error probability $\leq \frac{1}{n^c}$

runtime $\mathcal{O}(\log n)$

Remark. We traded correctness for running time.

Finding a Repeated Element

Given: An array A of n natural numbers such that
$\lceil \frac{n}{2} \rceil$ of them are identical and $\lfloor \frac{n}{2} \rfloor$ of them are pairwise distinct.

Task: Find the repeated element.

3	7	3	3	3	8	1	3	2

Deterministic approaches:

Compare each element with every predecessor $\Theta(n^2)$ time

Sort the array, then perform a linear sweep. $\Theta(n \log n)$ time

Compute and report the median. $\Theta(n)$ time

Randomized approach:

FindRepeated(A)
while true **do**
randomly choose $i \in \{1, \ldots, n\}$
randomly choose $j \in \{1, \ldots, n\} \setminus \{i\}$
if $A[i] = A[j]$ **then return** $A[i]$

Success probability in each step

$$\geq \frac{n/2}{n} \cdot \frac{(n/2) - 1}{n - 1} \approx \frac{1}{4}$$

$\Rightarrow$ Expected number of steps ≈ 4

Remark. The algorithm only returns correct answers, but may run forever.

Las Vegas and Monte Carlo Algorithms

Las Vegas algorithm. Returns a correct result, but the running time (and possibly the required space) are random variables.

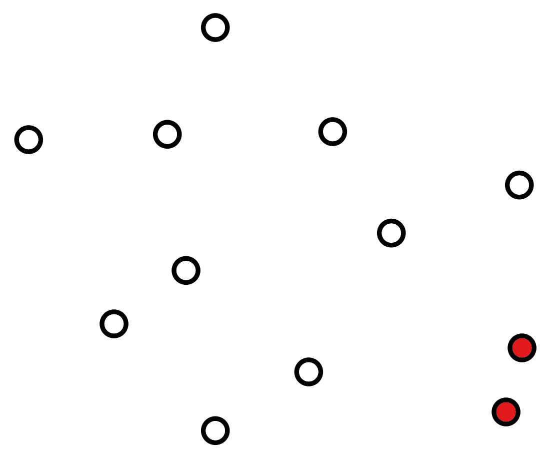
Examples. FindRepeated, RandomizedQuicksort

Monte Carlo algorithm. Returns incorrect result or fails with a certain (small) probability. The running time *may* be a random variable.

Examples. FindLarge, Karger's randomized MinCut algorithm

Remark. A Monte Carlo algorithm can often be turned into a Las Vegas algorithm and vice versa.

Closest Pair



Given: (multi-)set of points $P = \{p_1, p_2, \ldots, p_n\} \subseteq \mathbb{R}^2$.

Task: Find a pair of distinct elements $p_a, p_b \in P$ such that the Euclidean distance $\delta = ||p_a, p_b||$ is minimum.

Deterministic approaches:

Brute-force	$\Theta(n^2)$
Divide and conquer (recall from ADS)	$\Theta(n \log n)$

Lower bound:

Element Uniqueness: Given numbers $a_1, a_2, \ldots, a_n$. Are they pairwise distinct?

There is no $o(n \log n)$ time algorithm for Element Uniqueness.

(under some assumption concerning the arithmetic model)

$\Rightarrow$ There is no $o(n \log n)$ time algorithm for Closest pair.

(under the same assumption concerning the arithmetic model)

Reduction: map each a_i to a point (a_i, a_i) and test if the minimum distance is 0.

A Randomized Incremental Algorithm for Closest Pair

Define $P_i = \{p_1, p_2, \ldots, p_i\}$ and let δ_i be the distance of a closest pair in P_i.

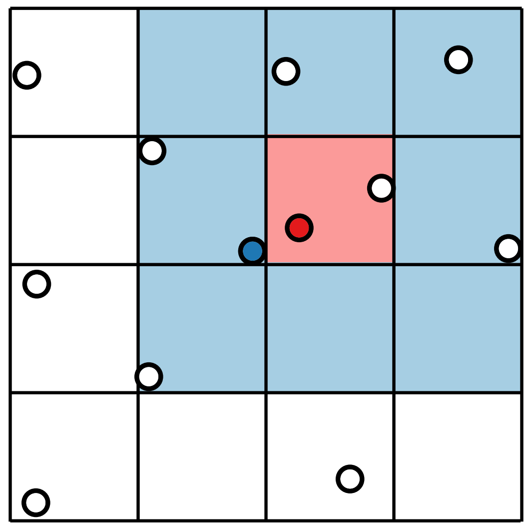
Idea: $\delta_2 = ||p_1, p_2||$. Compute $\delta_3, \delta_4, \ldots, \delta_n$ by adding the points iteratively.

Suppose we have already determined δ_{i-1}.

Consider a square grid with cells of size $\delta_{i-1} \times \delta_{i-1}$.

Add the point p_i. If $\delta_i < \delta_{i-1}$, then p_i must be part of each closest pair p_i, p_j.

Moreover, p_j must lie in the cell of p_i or one of the adjacent cells.



(simple) exercise

Each of these cells contains at most $\mathcal{O}(1)$ points of P_{i-1} ($\Leftarrow$ packing argument).

The coordinates of the cell of p_i can be determined in $\mathcal{O}(1)$ time assuming the floor function can be computed in $\mathcal{O}(1)$ time.

$\Rightarrow$ The test $\delta_i < \delta_{i-1}$ can be performed in $\mathcal{O}(1)$ time assuming P_{i-1} is stored in a suitable dictionary for the nonempty cells (implementable via dynamic perfect hashing).

Backwards Analysis

If $\delta_i = \delta_{i-1}$, we add p_i to the dictionary in $\mathcal{O}(1)$ time.
If $\delta_i < \delta_{i-1}$, the cell size changes and we have to rebuild the dictionary in $\mathcal{O}(i)$ time.
$\Rightarrow$ total runtime $\mathcal{O}(n^2)$.

Randomization: In the beginning, randomly permute the point set P.

Probability that adding p_i to P_{i-1} decreases the minimum distance
$=$ Probability that deleting p_i from P_i increases the minimum distance

How many points p in P_i have the property that the minimum distance in $P_i \setminus \{p\}$ is larger than in P_i? ≤ 2 points

Let X_i be the running time used for adding p_i.
$\Rightarrow \mathsf{E}[X_i] \leq \frac{2}{i} \cdot \mathcal{O}(i) + \frac{i-2}{i} \cdot \mathcal{O}(1) = \mathcal{O}(1)$

Let $X = (X_1 + \cdots + X_n)$ be the total running time used by the algorithm.
$\Rightarrow \mathsf{E}[X] = \mathsf{E}[X_1] + \cdots + \mathsf{E}[X_n] \in \mathcal{O}(n)$

Discussion

Randomized algorithms (often)

- are faster or use less space than deterministic algorithms in practice,
- have expected runtimes beyond deterministic lower bounds,
- are easier to implement/more elegant than deterministic strategies,
- allow for trading runtime against output quality,
- provide a good strategy for games or search in unknown environments.