Julius-Maximilians-
UNIVERSITÄT WÜRZBURG

Lehrstuhl für
INFORMATIK I
Effiziente Algorithmen und wissensbasierte Systeme

Institut für Informatik

Advanced Algorithms

Winter term 2019/20

Lecture 4. Randomized Algorithms

(based on lecture notes of Sabine Storandt)

Steven Chaplick & Alexander Wolff *Chair for Computer Science I*

Randomized Algorithms

Randomized Algorithms

- are faster or use less space than deterministic algorithms in practise,

Randomized Algorithms

- are faster or use less space than deterministic algorithms in practise,
- have theoretical runtimes beyond deterministic lower bounds,

Randomized Algorithms

- are faster or use less space than deterministic algorithms in practise,
- have theoretical runtimes beyond deterministic lower bounds,
- are easier to implement/more elegant than deterministic strategies,

Randomized Algorithms

- are faster or use less space than deterministic algorithms in practise,
- have theoretical runtimes beyond deterministic lower bounds,
- are easier to implement/more elegant than deterministic strategies,
- allow for trading runtime against output quality,

Randomized Algorithms

- are faster or use less space than deterministic algorithms in practise,
- have theoretical 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.

Some Basics

A (discrete) random variable X maps a (finite) set Ω of possible outcomes of a random experiment to some measurable set Ω' of observations (e.g., $\mathbb{N}$ or $\mathbb{R}$).

Some Basics

A (discrete) random variable X maps a (finite) set Ω of possible outcomes of a random experiment to some measurable set Ω' of observations (e.g., $\mathbb{N}$ or $\mathbb{R}$).

Example: dice: $\{⚀, ⚁, ⚂, ⚃, ⚄, ⚅\} \to \{1, 2, 3, 4, 5, 6\}$

Some Basics

A (discrete) random variable X maps a (finite) set Ω of possible outcomes of a random experiment to some measurable set Ω' of observations (e.g., $\mathbb{N}$ or $\mathbb{R}$).

Example: dice: {⚀, ⚁, ⚂, ⚃, ⚄, ⚅} $\rightarrow \{1, 2, 3, 4, 5, 6\}$

The *expected value* of a discrete random variable X is

$$\mathbf{E}[X] =$$

Some Basics

A (discrete) random variable X maps a (finite) set Ω of possible outcomes of a random experiment to some measurable set Ω' of observations (e.g., $\mathbb{N}$ or $\mathbb{R}$).

Example: dice: {⚀, ⚁, ⚂, ⚃, ⚄, ⚅} $\to \{1, 2, 3, 4, 5, 6\}$

The *expected value* of a discrete random variable X is

$$\mathbf{E}[X] = \sum_{i \in \Omega'} i \cdot \mathbf{Pr}[X = i].$$

Some Basics

A (discrete) random variable X maps a (finite) set Ω of possible outcomes of a random experiment to some measurable set Ω' of observations (e.g., $\mathbb{N}$ or $\mathbb{R}$).

Example: dice: $\{$⚀,⚁,⚂,⚃,⚄,⚅$\} \to \{1, 2, 3, 4, 5, 6\}$

The *expected value* of a discrete random variable X is

$$\mathbf{E}[X] = \sum_{i \in \Omega'} i \cdot \mathbf{Pr}[X = i].$$

Example: $\mathbf{E}[\text{fair dice}] =$

Some Basics

A (discrete) random variable X maps a (finite) set Ω of possible outcomes of a random experiment to some measurable set Ω' of observations (e.g., $\mathbb{N}$ or $\mathbb{R}$).

Example: dice: {⚀,⚁,⚂,⚃,⚄,⚅} $\rightarrow \{1, 2, 3, 4, 5, 6\}$

The *expected value* of a discrete random variable X is

$$\mathbf{E}[X] = \sum_{i \in \Omega'} i \cdot \mathbf{Pr}[X = i].$$

Example: $\mathbf{E}[\text{fair dice}] = (1 + 2 + 3 + 4 + 5 + 6)/6 = 3.5$

Some Basics

A (discrete) random variable X maps a (finite) set Ω of possible outcomes of a random experiment to some measurable set Ω' of observations (e.g., $\mathbb{N}$ or $\mathbb{R}$).

Example: dice: $\{⚀,⚁,⚂,⚃,⚄,⚅\} \to \{1,2,3,4,5,6\}$

The *expected value* of a discrete random variable X is

$$\mathbf{E}[X] = \sum_{i \in \Omega'} i \cdot \mathbf{Pr}[X = i].$$

Example: $\mathbf{E}[\text{fair dice}] = (1+2+3+4+5+6)/6 = 3.5$

strange dice: $\{⚀,⚀,⚀,⚅,⚅,⚅\} \to \{1,1,1,6,6,6\}$

Some Basics

A (discrete) random variable X maps a (finite) set Ω of possible outcomes of a random experiment to some measurable set Ω' of observations (e.g., $\mathbb{N}$ or $\mathbb{R}$).

Example: dice: {⚀,⚁,⚂,⚃,⚄,⚅} $\rightarrow \{1, 2, 3, 4, 5, 6\}$

The *expected value* of a discrete random variable X is

$$\mathbf{E}[X] = \sum_{i \in \Omega'} i \cdot \mathbf{Pr}[X = i].$$

Example: $\mathbf{E}[\text{fair dice}] = (1 + 2 + 3 + 4 + 5 + 6)/6 = 3.5$

strange dice: {⚀,⚀,⚀,⚅,⚅,⚅} $\rightarrow \{1, 1, 1, 6, 6, 6\}$

$\mathbf{E}[\text{strange dice}] =$

Some Basics

A (discrete) random variable X maps a (finite) set Ω of possible outcomes of a random experiment to some measurable set Ω' of observations (e.g., $\mathbb{N}$ or $\mathbb{R}$).

Example: dice: {⚀,⚁,⚂,⚃,⚄,⚅} $\rightarrow \{1, 2, 3, 4, 5, 6\}$

The *expected value* of a discrete random variable X is

$$\mathbf{E}[X] = \sum_{i \in \Omega'} i \cdot \mathbf{Pr}[X = i].$$

Example: $\mathbf{E}[\text{fair dice}] = (1 + 2 + 3 + 4 + 5 + 6)/6 = 3.5$

strange dice: {⚀,⚀,⚀,⚅,⚅,⚅} $\rightarrow \{1, 1, 1, 6, 6, 6\}$

$$\mathbf{E}[\text{strange dice}] = (1 + 1 + 1 + 6 + 6 + 6)/6 = 3.5$$

First Success

Let $X\colon \{\text{failure}, \text{success}\} \to \{0, 1\}$ be a random variable.

First Success

Let $X\colon \{\text{failure}, \text{success}\} \to \{0, 1\}$ be a random variable.

Let $p = \mathbf{Pr}[X = 1]$ be the success probability.

First Success

Let $X\colon \{\text{failure}, \text{success}\} \to \{0, 1\}$ be a random variable.

Let $p = \mathbf{Pr}[X = 1]$ be the success probability.

$\Rightarrow$ $q := \mathbf{Pr}[X = 0] = 1 - p$ is the *failure probability* (or *rate*).

First Success

Let $X\colon \{\text{failure}, \text{success}\} \to \{0, 1\}$ be a random variable.

Let $p = \mathbf{Pr}[X = 1]$ be the success probability.

$\Rightarrow$ $q := \mathbf{Pr}[X = 0] = 1 - p$ is the *failure probability* (or *rate*).

Repeat experiment many times.
Assume that outcomes are independent from each other.

First Success

Let $X\colon \{\text{failure}, \text{success}\} \to \{0,1\}$ be a random variable.

Let $p = \mathbf{Pr}[X = 1]$ be the success probability.

$\Rightarrow$ $q := \mathbf{Pr}[X = 0] = 1 - p$ is the *failure probability* (or *rate*).

Repeat experiment many times.
Assume that outcomes are independent from each other.

Random variable Y counts the number of rounds until $X = 1$ for the first time.

First Success

Let $X\colon \{\text{failure}, \text{success}\} \to \{0, 1\}$ be a random variable.

Let $p = \mathbf{Pr}[X = 1]$ be the success probability.

$\Rightarrow$ $q := \mathbf{Pr}[X = 0] = 1 - p$ is the *failure probability* (or *rate*).

Repeat experiment many times.
Assume that outcomes are independent from each other.

Random variable Y counts the number of rounds until $X = 1$ for the first time.

$\Rightarrow \mathbf{Pr}[Y = j] =$

First Success

Let $X\colon \{\text{failure}, \text{success}\} \to \{0,1\}$ be a random variable.

Let $p = \mathbf{Pr}[X = 1]$ be the success probability.

$\Rightarrow$ $q := \mathbf{Pr}[X = 0] = 1 - p$ is the *failure probability* (or *rate*).

Repeat experiment many times.
Assume that outcomes are independent from each other.

Random variable Y counts the number of rounds until $X = 1$ for the first time.

$\Rightarrow \mathbf{Pr}[Y = j] = \; q^{j-1}p$

First Success

Let $X \colon \{\text{failure}, \text{success}\} \to \{0, 1\}$ be a random variable.

Let $p = \mathbf{Pr}[X = 1]$ be the success probability.

$\Rightarrow$ $q := \mathbf{Pr}[X = 0] = 1 - p$ is the *failure probability* (or *rate*).

Repeat experiment many times.
Assume that outcomes are independent from each other.

Random variable Y counts the number of rounds until $X = 1$ for the first time.

$\Rightarrow \mathbf{Pr}[Y = j] = \; q^{j-1}p$

$\Rightarrow \mathbf{E}[Y] =$

First Success

Let $X \colon \{\text{failure}, \text{success}\} \to \{0, 1\}$ be a random variable.

Let $p = \mathbf{Pr}[X = 1]$ be the success probability.

$\Rightarrow$ $q := \mathbf{Pr}[X = 0] = 1 - p$ is the *failure probability* (or *rate*).

Repeat experiment many times.
Assume that outcomes are independent from each other.

Random variable Y counts the number of rounds until $X = 1$ for the first time.

$\Rightarrow \mathbf{Pr}[Y = j] = \; q^{j-1}p$

$\Rightarrow \mathbf{E}[Y] = \; \sum_{j=1}^{\infty} j \cdot q^{j-1}p =$

First Success

Let $X\colon \{\text{failure}, \text{success}\} \to \{0, 1\}$ be a random variable.

Let $p = \mathbf{Pr}[X = 1]$ be the success probability.

$\Rightarrow$ $q := \mathbf{Pr}[X = 0] = 1 - p$ is the *failure probability* (or *rate*).

Repeat experiment many times.
Assume that outcomes are independent from each other.

Random variable Y counts the number of rounds until $X = 1$ for the first time.

$\Rightarrow \mathbf{Pr}[Y = j] = \; q^{j-1}p$

$\Rightarrow \mathbf{E}[Y] = \; \sum_{j=1}^{\infty} j \cdot q^{j-1}p = \; p \cdot (\sum_{j=1}^{\infty} q^j)' =$

First Success

Let $X \colon \{\text{failure}, \text{success}\} \to \{0, 1\}$ be a random variable.

Let $p = \mathbf{Pr}[X = 1]$ be the success probability.

$\Rightarrow$ $q := \mathbf{Pr}[X = 0] = 1 - p$ is the *failure probability* (or *rate*).

Repeat experiment many times.
Assume that outcomes are independent from each other.

Random variable Y counts the number of rounds until $X = 1$ for the first time.

$\Rightarrow \mathbf{Pr}[Y = j] = \; q^{j-1}p$

$\Rightarrow \mathbf{E}[Y] = \; \sum_{j=1}^{\infty} j \cdot q^{j-1}p = \; p \cdot (\sum_{j=1}^{\infty} q^j)' = p \cdot (\frac{1}{1-q})' =$

First Success

Let $X\colon \{\text{failure}, \text{success}\} \to \{0, 1\}$ be a random variable.
Let $p = \mathbf{Pr}[X = 1]$ be the success probability.
$\Rightarrow$ $q := \mathbf{Pr}[X = 0] = 1 - p$ is the *failure probability* (or *rate*).

Repeat experiment many times.
Assume that outcomes are independent from each other.

Random variable Y counts the number of rounds until $X = 1$ for the first time.

$\Rightarrow \mathbf{Pr}[Y = j] = \; q^{j-1}p$

$$\Rightarrow \mathbf{E}[Y] = \; \sum_{j=1}^{\infty} j \cdot q^{j-1}p = \; p \cdot (\sum_{j=1}^{\infty} q^j)' = p \cdot (\tfrac{1}{1-q})' =$$
$$= p \cdot (\tfrac{1}{p})' =$$

First Success

Let $X \colon \{\text{failure}, \text{success}\} \to \{0, 1\}$ be a random variable.
Let $p = \mathbf{Pr}[X = 1]$ be the success probability.
$\Rightarrow$ $q := \mathbf{Pr}[X = 0] = 1 - p$ is the *failure probability* (or *rate*).

Repeat experiment many times.
Assume that outcomes are independent from each other.

Random variable Y counts the number of rounds until $X = 1$ for the first time.

$\Rightarrow \mathbf{Pr}[Y = j] = \; q^{j-1}p$

$$\Rightarrow \mathbf{E}[Y] = \; \textstyle\sum_{j=1}^{\infty} j \cdot q^{j-1}p = \; p \cdot (\sum_{j=1}^{\infty} q^j)' = p \cdot (\frac{1}{1-q})' =$$
$$= p \cdot (\tfrac{1}{p})' = \; p \cdot \tfrac{1}{p^2} =$$

First Success

Let $X\colon \{\text{failure}, \text{success}\} \to \{0, 1\}$ be a random variable.

Let $p = \mathbf{Pr}[X = 1]$ be the success probability.

$\Rightarrow$ $q := \mathbf{Pr}[X = 0] = 1 - p$ is the *failure probability* (or *rate*).

Repeat experiment many times.
Assume that outcomes are independent from each other.

Random variable Y counts the number of rounds until $X = 1$ for the first time.

$\Rightarrow \mathbf{Pr}[Y = j] = \ q^{j-1}p$

$$\Rightarrow \mathbf{E}[Y] = \ \sum_{j=1}^{\infty} j \cdot q^{j-1}p = \ p \cdot (\sum_{j=1}^{\infty} q^j)' = p \cdot (\tfrac{1}{1-q})' =$$

$$= p \cdot (\tfrac{1}{p})' = \ p \cdot \tfrac{1}{p^2} = \ 1/p \qquad \square$$

Linearity of Expectation

Let X and Y be two random variables and $\lambda \in \mathbb{R}$. Then
$\mathbf{E}[X + \lambda \cdot Y] =$

Linearity of Expectation

Let X and Y be two random variables and $\lambda \in \mathbb{R}$. Then
$\mathbf{E}[X + \lambda \cdot Y] = \mathbf{E}[X] + \lambda \cdot \mathbf{E}[Y]$

Linearity of Expectation

Let X and Y be two random variables and $\lambda \in \mathbb{R}$. Then
$\mathbf{E}[X + \lambda \cdot Y] = \mathbf{E}[X] + \lambda \cdot \mathbf{E}[Y]$

Indicator Random Variables

Example I: Guessing cards (without memory).

Linearity of Expectation

Let X and Y be two random variables and $\lambda \in \mathbb{R}$. Then
$\mathbf{E}[X + \lambda \cdot Y] = \mathbf{E}[X] + \lambda \cdot \mathbf{E}[Y]$

Indicator Random Variables

Example I: Guessing cards (without memory). Deck of n cards.

Linearity of Expectation

Let X and Y be two random variables and $\lambda \in \mathbb{R}$. Then
$\mathbf{E}[X + \lambda \cdot Y] = \mathbf{E}[X] + \lambda \cdot \mathbf{E}[Y]$

Indicator Random Variables

Example I: Guessing cards (without memory). Deck of n cards.
Let $X_i \colon \{\text{guessed}, \text{not guessed}\} \to \{0, 1\}$ be a random variable that indicates whether card i was guessed or not $(i = 1, \dots, n)$.

Linearity of Expectation

Let X and Y be two random variables and $\lambda \in \mathbb{R}$. Then
$\mathbf{E}[X + \lambda \cdot Y] = \mathbf{E}[X] + \lambda \cdot \mathbf{E}[Y]$

Indicator Random Variables

Example I: Guessing cards (without memory). Deck of n cards.

Let $X_i \colon \{\text{guessed}, \text{not guessed}\} \to \{0, 1\}$ be a random variable that indicates whether card i was guessed or not ($i = 1, \ldots, n$).

$X_1, \ldots, X_n$ are *indicator* random variables.

Linearity of Expectation

Let X and Y be two random variables and $\lambda \in \mathbb{R}$. Then
$\mathbf{E}[X + \lambda \cdot Y] = \mathbf{E}[X] + \lambda \cdot \mathbf{E}[Y]$

Indicator Random Variables

Example I: Guessing cards (without memory). Deck of n cards.

Let $X_i : \{\text{guessed}, \text{not guessed}\} \to \{0, 1\}$ be a random variable that indicates whether card i was guessed or not ($i = 1, \dots, n$).

$X_1, \dots, X_n$ are *indicator* random variables.

$\Rightarrow \mathbf{E}[X_i] = \mathbf{Pr}[X_i = 1] =_{\text{here}}$

Linearity of Expectation

Let X and Y be two random variables and $\lambda \in \mathbb{R}$. Then
$\mathbf{E}[X + \lambda \cdot Y] = \mathbf{E}[X] + \lambda \cdot \mathbf{E}[Y]$

Indicator Random Variables

Example I: Guessing cards (without memory). Deck of n cards.

Let $X_i \colon \{\text{guessed}, \text{not guessed}\} \to \{0, 1\}$ be a random variable that indicates whether card i was guessed or not ($i = 1, \ldots, n$).

$X_1, \ldots, X_n$ are *indicator* random variables.

$\Rightarrow \mathbf{E}[X_i] = \mathbf{Pr}[X_i = 1] =_{\text{here}} 1/n$

Linearity of Expectation

Let X and Y be two random variables and $\lambda \in \mathbb{R}$. Then
$\mathbf{E}[X + \lambda \cdot Y] = \mathbf{E}[X] + \lambda \cdot \mathbf{E}[Y]$

Indicator Random Variables

Example I: Guessing cards (without memory). Deck of n cards.

Let $X_i : \{\text{guessed}, \text{not guessed}\} \to \{0, 1\}$ be a random variable that indicates whether card i was guessed or not $(i = 1, \ldots, n)$.

$X_1, \ldots, X_n$ are *indicator* random variables.

$\Rightarrow \mathbf{E}[X_i] = \mathbf{Pr}[X_i = 1] =_{\text{here}} 1/n$

Let X count the number of correct guesses.

Linearity of Expectation

Let X and Y be two random variables and $\lambda \in \mathbb{R}$. Then
$\mathbf{E}[X + \lambda \cdot Y] = \mathbf{E}[X] + \lambda \cdot \mathbf{E}[Y]$

Indicator Random Variables

Example I: Guessing cards (without memory). Deck of n cards.
Let $X_i \colon \{\text{guessed}, \text{not guessed}\} \to \{0, 1\}$ be a random variable that indicates whether card i was guessed or not ($i = 1, \dots, n$).

$X_1, \dots, X_n$ are *indicator* random variables.
$\Rightarrow \mathbf{E}[X_i] = \mathbf{Pr}[X_i = 1] =_{\text{here}} 1/n$

Let X count the number of correct guesses.
$\Rightarrow \quad X =$

Linearity of Expectation

Let X and Y be two random variables and $\lambda \in \mathbb{R}$. Then
$\mathbf{E}[X + \lambda \cdot Y] = \mathbf{E}[X] + \lambda \cdot \mathbf{E}[Y]$

Indicator Random Variables

Example I: Guessing cards (without memory). Deck of n cards.
Let $X_i : \{\text{guessed, not guessed}\} \to \{0, 1\}$ be a random variable that indicates whether card i was guessed or not ($i = 1, \ldots, n$).

$X_1, \ldots, X_n$ are *indicator* random variables.
$\Rightarrow \mathbf{E}[X_i] = \mathbf{Pr}[X_i = 1] =_{\text{here}} 1/n$

Let X count the number of correct guesses.
$\Rightarrow \quad X = X_1 + \cdots + X_n$

Linearity of Expectation

Let X and Y be two random variables and $\lambda \in \mathbb{R}$. Then
$\mathbf{E}[X + \lambda \cdot Y] = \mathbf{E}[X] + \lambda \cdot \mathbf{E}[Y]$

Indicator Random Variables

Example I: Guessing cards (without memory). Deck of n cards.

Let $X_i \colon \{\text{guessed, not guessed}\} \to \{0, 1\}$ be a random variable that indicates whether card i was guessed or not ($i = 1, \ldots, n$).

$X_1, \ldots, X_n$ are *indicator* random variables.

$\Rightarrow \mathbf{E}[X_i] = \mathbf{Pr}[X_i = 1] =_{\text{here}} 1/n$

Let X count the number of correct guesses.

$\Rightarrow \quad X = X_1 + \cdots + X_n$

$\Rightarrow \mathbf{E}[X] =$

Linearity of Expectation

Let X and Y be two random variables and $\lambda \in \mathbb{R}$. Then
$\mathbf{E}[X + \lambda \cdot Y] = \mathbf{E}[X] + \lambda \cdot \mathbf{E}[Y]$

Indicator Random Variables

Example I: Guessing cards (without memory). Deck of n cards.

Let $X_i \colon \{\text{guessed}, \text{not guessed}\} \to \{0, 1\}$ be a random variable that indicates whether card i was guessed or not ($i = 1, \ldots, n$).

$X_1, \ldots, X_n$ are *indicator* random variables.

$\Rightarrow \mathbf{E}[X_i] = \mathbf{Pr}[X_i = 1] =_{\text{here}} 1/n$

Let X count the number of correct guesses.

$\Rightarrow \quad X = X_1 + \cdots + X_n$

$\Rightarrow \mathbf{E}[X] = \mathbf{E}[X_1 + \cdots + X_n] =$

Linearity of Expectation

Let X and Y be two random variables and $\lambda \in \mathbb{R}$. Then
$\mathbf{E}[X + \lambda \cdot Y] = \mathbf{E}[X] + \lambda \cdot \mathbf{E}[Y]$

Indicator Random Variables

Example I: Guessing cards (without memory). Deck of n cards.

Let $X_i \colon \{\text{guessed}, \text{not guessed}\} \to \{0, 1\}$ be a random variable that indicates whether card i was guessed or not ($i = 1, \dots, n$).

$X_1, \dots, X_n$ are *indicator* random variables.

$\Rightarrow \mathbf{E}[X_i] = \mathbf{Pr}[X_i = 1] =_{\text{here}} 1/n$

Let X count the number of correct guesses.

$\Rightarrow \quad X = X_1 + \dots + X_n$

$\Rightarrow \mathbf{E}[X] = \mathbf{E}[X_1 + \dots + X_n] = \mathbf{E}[X_1] + \dots + \mathbf{E}[X_n] =$

Linearity of Expectation

Let X and Y be two random variables and $\lambda \in \mathbb{R}$. Then
$\mathbf{E}[X + \lambda \cdot Y] = \mathbf{E}[X] + \lambda \cdot \mathbf{E}[Y]$

Indicator Random Variables

Example I: Guessing cards (without memory). Deck of n cards.

Let $X_i \colon \{\text{guessed}, \text{not guessed}\} \to \{0, 1\}$ be a random variable that indicates whether card i was guessed or not $(i = 1, \ldots, n)$.

$X_1, \ldots, X_n$ are *indicator* random variables.

$\Rightarrow \mathbf{E}[X_i] = \mathbf{Pr}[X_i = 1] =_{\text{here}} 1/n$

Let X count the number of correct guesses.

$\Rightarrow \quad X = X_1 + \cdots + X_n$

$\Rightarrow \mathbf{E}[X] = \mathbf{E}[X_1 + \cdots + X_n] = \mathbf{E}[X_1] + \cdots + \mathbf{E}[X_n] = n \cdot \frac{1}{n} = 1.$

Linearity of Expectation

Let X and Y be two random variables and $\lambda \in \mathbb{R}$. Then
$\mathbf{E}[X + \lambda \cdot Y] = \mathbf{E}[X] + \lambda \cdot \mathbf{E}[Y]$

Indicator Random Variables

Example I: Guessing cards (without memory). Deck of n cards.
Let $X_i \colon \{\text{guessed}, \text{not guessed}\} \to \{0, 1\}$ be a random variable that indicates whether card i was guessed or not ($i = 1, \ldots, n$).

$X_1, \ldots, X_n$ are *indicator* random variables.

$\Rightarrow \mathbf{E}[X_i] = \mathbf{Pr}[X_i = 1] =_{\text{here}} 1/n$

Let X count the number of correct guesses.

$\Rightarrow \quad X = X_1 + \cdots + X_n$

$\Rightarrow \mathbf{E}[X] = \mathbf{E}[X_1 + \cdots + X_n] = \mathbf{E}[X_1] + \cdots + \mathbf{E}[X_n] = n \cdot \frac{1}{n} = 1.$

Note that this is independent of n!

Using Indicator Random Variables

Example II: Guessing cards (*with* memory).

Using Indicator Random Variables

Example II: Guessing cards (*with* memory).

Now $\mathbf{Pr}[X_i = 1]$ depends on the current size of the deck.

Using Indicator Random Variables

Example II: Guessing cards (*with* memory).

Now $\mathbf{Pr}[X_i = 1]$ depends on the current size of the deck.

$\mathbf{E}[X_i] = \mathbf{Pr}[X_i = 1] =$

Using Indicator Random Variables

Example II: Guessing cards (*with* memory).

Now $\mathbf{Pr}[X_i = 1]$ depends on the current size of the deck.

$\mathbf{E}[X_i] = \mathbf{Pr}[X_i = 1] = 1/(n - i + 1)$

Using Indicator Random Variables

Example II: Guessing cards (*with* memory).

Now $\mathbf{Pr}[X_i = 1]$ depends on the current size of the deck.

$\mathbf{E}[X_i] = \mathbf{Pr}[X_i = 1] = 1/(n - i + 1)$

$\Rightarrow \mathbf{E}[X] = \mathbf{E}[X_1 + \cdots + X_n] =$

Using Indicator Random Variables

Example II: Guessing cards (*with* memory).

Now $\mathbf{Pr}[X_i = 1]$ depends on the current size of the deck.

$\mathbf{E}[X_i] = \mathbf{Pr}[X_i = 1] = 1/(n - i + 1)$

$\Rightarrow \mathbf{E}[X] = \mathbf{E}[X_1 + \cdots + X_n] = \frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{2} + 1 =$

Using Indicator Random Variables

Example II: Guessing cards (*with* memory).

Now $\mathbf{Pr}[X_i = 1]$ depends on the current size of the deck.

$\mathbf{E}[X_i] = \mathbf{Pr}[X_i = 1] = 1/(n - i + 1)$

$\Rightarrow \mathbf{E}[X] = \mathbf{E}[X_1 + \cdots + X_n] = \frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{2} + 1 = H_n$

Using Indicator Random Variables

Example II: Guessing cards (*with* memory).

Now $\mathbf{Pr}[X_i = 1]$ depends on the current size of the deck.

$\mathbf{E}[X_i] = \mathbf{Pr}[X_i = 1] = 1/(n - i + 1)$

$\Rightarrow \mathbf{E}[X] = \mathbf{E}[X_1 + \cdots + X_n] = \frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{2} + 1 = H_n$

H_n is the n-th harmonic number;

Using Indicator Random Variables

Example II: Guessing cards (*with* memory).

Now $\mathbf{Pr}[X_i = 1]$ depends on the current size of the deck.

$\mathbf{E}[X_i] = \mathbf{Pr}[X_i = 1] = 1/(n - i + 1)$

$\Rightarrow \mathbf{E}[X] = \mathbf{E}[X_1 + \cdots + X_n] = \frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{2} + 1 = H_n$

H_n is the n-th harmonic number; $\leq H_n \leq$

Using Indicator Random Variables

Example II: Guessing cards (*with* memory).

Now $\mathbf{Pr}[X_i = 1]$ depends on the current size of the deck.

$\mathbf{E}[X_i] = \mathbf{Pr}[X_i = 1] = 1/(n - i + 1)$

$\Rightarrow \mathbf{E}[X] = \mathbf{E}[X_1 + \cdots + X_n] = \frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{2} + 1 = H_n$

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

Using Indicator Random Variables

Example II: Guessing cards (*with* memory).

Now $\mathbf{Pr}[X_i = 1]$ depends on the current size of the deck.

$\mathbf{E}[X_i] = \mathbf{Pr}[X_i = 1] = 1/(n - i + 1)$

$\Rightarrow \mathbf{E}[X] = \mathbf{E}[X_1 + \cdots + X_n] = \frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{2} + 1 = H_n$

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

Using Indicator Random Variables

Example II: Guessing cards (*with* memory).

Now $\mathbf{Pr}[X_i = 1]$ depends on the current size of the deck.

$\mathbf{E}[X_i] = \mathbf{Pr}[X_i = 1] = 1/(n - i + 1)$

$\Rightarrow \mathbf{E}[X] = \mathbf{E}[X_1 + \cdots + X_n] = \frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{2} + 1 = H_n$

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

$\Rightarrow \mathbf{E}[X] = H_n \in \Theta(\log n)$

Using Indicator Random Variables

Example II: Guessing cards (*with* memory).

Now $\mathbf{Pr}[X_i = 1]$ depends on the current size of the deck.

$\mathbf{E}[X_i] = \mathbf{Pr}[X_i = 1] = 1/(n - i + 1)$

$\Rightarrow \mathbf{E}[X] = \mathbf{E}[X_1 + \cdots + X_n] = \frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{2} + 1 = H_n$

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

$\Rightarrow \mathbf{E}[X] = H_n \in \Theta(\log n)$ *Note that this does depend on n!*

Using Indicator Random Variables

Example II: Guessing cards (*with* memory).

Now $\mathbf{Pr}[X_i = 1]$ depends on the current size of the deck.

$\mathbf{E}[X_i] = \mathbf{Pr}[X_i = 1] = 1/(n - i + 1)$

$\Rightarrow \mathbf{E}[X] = \mathbf{E}[X_1 + \cdots + X_n] = \frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{2} + 1 = H_n$

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

$\Rightarrow \mathbf{E}[X] = H_n \in \Theta(\log n)$ *Note that this does depend on n!*

Example III: Collecting goodies

Using Indicator Random Variables

Example II: Guessing cards (*with* memory).

Now $\mathbf{Pr}[X_i = 1]$ depends on the current size of the deck.

$\mathbf{E}[X_i] = \mathbf{Pr}[X_i = 1] = 1/(n - i + 1)$

$\Rightarrow \mathbf{E}[X] = \mathbf{E}[X_1 + \cdots + X_n] = \frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{2} + 1 = H_n$

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

$\Rightarrow \mathbf{E}[X] = H_n \in \Theta(\log n)$ *Note that this does depend on n!*

Example III: Collecting goodies

How often do you have to shop to collect all n goodies?

Using Indicator Random Variables

Example II: Guessing cards (*with* memory).

Now $\mathbf{Pr}[X_i = 1]$ depends on the current size of the deck.

$\mathbf{E}[X_i] = \mathbf{Pr}[X_i = 1] = 1/(n - i + 1)$

$\Rightarrow \mathbf{E}[X] = \mathbf{E}[X_1 + \cdots + X_n] = \frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{2} + 1 = H_n$

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

$\Rightarrow \mathbf{E}[X] = H_n \in \Theta(\log n)$ *Note that this does depend on n!*

Example III: Collecting goodies

How often do you have to shop to collect all n goodies? (X)

Using Indicator Random Variables

Example II: Guessing cards (*with* memory).

Now $\mathbf{Pr}[X_i = 1]$ depends on the current size of the deck.

$\mathbf{E}[X_i] = \mathbf{Pr}[X_i = 1] = 1/(n - i + 1)$

$\Rightarrow \mathbf{E}[X] = \mathbf{E}[X_1 + \cdots + X_n] = \frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{2} + 1 = H_n$

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

$\Rightarrow \mathbf{E}[X] = H_n \in \Theta(\log n)$ *Note that this does depend on n!*

Example III: Collecting goodies

How often do you have to shop to collect all n goodies? (X)

Assume: Each time you get a random goodie.

Using Indicator Random Variables

Example II: Guessing cards (*with* memory).

Now $\mathbf{Pr}[X_i = 1]$ depends on the current size of the deck.

$\mathbf{E}[X_i] = \mathbf{Pr}[X_i = 1] = 1/(n - i + 1)$

$\Rightarrow \mathbf{E}[X] = \mathbf{E}[X_1 + \cdots + X_n] = \frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{2} + 1 = H_n$

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

$\Rightarrow \mathbf{E}[X] = H_n \in \Theta(\log n)$ *Note that this does depend on n!*

Example III: Collecting goodies

How often do you have to shop to collect all n goodies? (X)

Assume: Each time you get a random goodie. You can't choose.

Using Indicator Random Variables

Example II: Guessing cards (*with* memory).

Now $\mathbf{Pr}[X_i = 1]$ depends on the current size of the deck.

$\mathbf{E}[X_i] = \mathbf{Pr}[X_i = 1] = 1/(n - i + 1)$

$\Rightarrow \mathbf{E}[X] = \mathbf{E}[X_1 + \cdots + X_n] = \frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{2} + 1 = H_n$

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

$\Rightarrow \mathbf{E}[X] = H_n \in \Theta(\log n)$ *Note that this does depend on n!*

Example III: Collecting goodies

How often do you have to shop to collect all n goodies? (X)

Assume: Each time you get a random goodie. You can't choose.

$X_i :=$ number of times you must shop to get i-th new goodie.

Using Indicator Random Variables

Example II: Guessing cards (*with* memory).

Now $\mathbf{Pr}[X_i = 1]$ depends on the current size of the deck.

$\mathbf{E}[X_i] = \mathbf{Pr}[X_i = 1] = 1/(n - i + 1)$

$\Rightarrow \mathbf{E}[X] = \mathbf{E}[X_1 + \cdots + X_n] = \frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{2} + 1 = H_n$

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

$\Rightarrow \mathbf{E}[X] = H_n \in \Theta(\log n)$ *Note that this does depend on n!*

Example III: Collecting goodies

How often do you have to shop to collect all n goodies? (X)

Assume: Each time you get a random goodie. You can't choose.

$X_i :=$ number of times you must shop to get i-th new goodie.

$\mathbf{Pr}[X_i = 1] =$

Using Indicator Random Variables

Example II: Guessing cards (*with* memory).

Now $\mathbf{Pr}[X_i = 1]$ depends on the current size of the deck.

$\mathbf{E}[X_i] = \mathbf{Pr}[X_i = 1] = 1/(n - i + 1)$

$\Rightarrow \mathbf{E}[X] = \mathbf{E}[X_1 + \cdots + X_n] = \frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{2} + 1 = H_n$

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

$\Rightarrow \mathbf{E}[X] = H_n \in \Theta(\log n)$ *Note that this does depend on n!*

Example III: Collecting goodies

How often do you have to shop to collect all n goodies? (X)

Assume: Each time you get a random goodie. You can't choose.

$X_i :=$ number of times you must shop to get i-th new goodie.

$\mathbf{Pr}[X_i = 1] = (n - i + 1)/n$

Using Indicator Random Variables

Example II: Guessing cards (*with* memory).

Now $\mathbf{Pr}[X_i = 1]$ depends on the current size of the deck.

$\mathbf{E}[X_i] = \mathbf{Pr}[X_i = 1] = 1/(n - i + 1)$

$\Rightarrow \mathbf{E}[X] = \mathbf{E}[X_1 + \cdots + X_n] = \frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{2} + 1 = H_n$

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

$\Rightarrow \mathbf{E}[X] = H_n \in \Theta(\log n)$ *Note that this does depend on n!*

Example III: Collecting goodies

How often do you have to shop to collect all n goodies? (X)

Assume: Each time you get a random goodie. You can't choose.

$X_i :=$ number of times you must shop to get i-th new goodie.

$\mathbf{Pr}[X_i = 1] = (n - i + 1)/n \Rightarrow \mathbf{E}[X_i] =$

Using Indicator Random Variables

Example II: Guessing cards (*with* memory).

Now $\mathbf{Pr}[X_i = 1]$ depends on the current size of the deck.

$\mathbf{E}[X_i] = \mathbf{Pr}[X_i = 1] = 1/(n - i + 1)$

$\Rightarrow \mathbf{E}[X] = \mathbf{E}[X_1 + \cdots + X_n] = \frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{2} + 1 = H_n$

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

$\Rightarrow \mathbf{E}[X] = H_n \in \Theta(\log n)$ *Note that this does depend on n!*

Example III: Collecting goodies

How often do you have to shop to collect all n goodies? (X)

Assume: Each time you get a random goodie. You can't choose.

$X_i :=$ number of times you must shop to get i-th new goodie.

$\mathbf{Pr}[X_i = 1] = (n - i + 1)/n \Rightarrow \mathbf{E}[X_i] = n/(n - i + 1)$

Using Indicator Random Variables

Example II: Guessing cards (*with* memory).

Now $\mathbf{Pr}[X_i = 1]$ depends on the current size of the deck.

$\mathbf{E}[X_i] = \mathbf{Pr}[X_i = 1] = 1/(n - i + 1)$

$\Rightarrow \mathbf{E}[X] = \mathbf{E}[X_1 + \cdots + X_n] = \frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{2} + 1 = H_n$

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

$\Rightarrow \mathbf{E}[X] = H_n \in \Theta(\log n)$ *Note that this does depend on n!*

Example III: Collecting goodies

How often do you have to shop to collect all n goodies? (X)

Assume: Each time you get a random goodie. You can't choose.

$X_i :=$ number of times you must shop to get i-th new goodie.

$\mathbf{Pr}[X_i = 1] = (n - i + 1)/n \Rightarrow \mathbf{E}[X_i] = n/(n - i + 1)$

$\Rightarrow \mathbf{E}[X] =$

Using Indicator Random Variables

Example II: Guessing cards (*with* memory).

Now $\mathbf{Pr}[X_i = 1]$ depends on the current size of the deck.

$\mathbf{E}[X_i] = \mathbf{Pr}[X_i = 1] = 1/(n - i + 1)$

$\Rightarrow \mathbf{E}[X] = \mathbf{E}[X_1 + \cdots + X_n] = \frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{2} + 1 = H_n$

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

$\Rightarrow \mathbf{E}[X] = H_n \in \Theta(\log n)$ *Note that this does depend on n!*

Example III: Collecting goodies

How often do you have to shop to collect all n goodies? (X)

Assume: Each time you get a random goodie. You can't choose.

$X_i :=$ number of times you must shop to get i-th new goodie.

$\mathbf{Pr}[X_i = 1] = (n - i + 1)/n \Rightarrow \mathbf{E}[X_i] = n/(n - i + 1)$

$\Rightarrow \mathbf{E}[X] = \mathbf{E}[X_1 + \cdots + X_n] =$

Using Indicator Random Variables

Example II: Guessing cards (*with* memory).

Now $\mathbf{Pr}[X_i = 1]$ depends on the current size of the deck.

$\mathbf{E}[X_i] = \mathbf{Pr}[X_i = 1] = 1/(n - i + 1)$

$\Rightarrow \mathbf{E}[X] = \mathbf{E}[X_1 + \cdots + X_n] = \frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{2} + 1 = H_n$

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

$\Rightarrow \mathbf{E}[X] = H_n \in \Theta(\log n)$ *Note that this does depend on n!*

Example III: Collecting goodies

How often do you have to shop to collect all n goodies? (X)

Assume: Each time you get a random goodie. You can't choose.

$X_i :=$ number of times you must shop to get i-th new goodie.

$\mathbf{Pr}[X_i = 1] = (n - i + 1)/n \Rightarrow \mathbf{E}[X_i] = n/(n - i + 1)$

$\Rightarrow \mathbf{E}[X] = \mathbf{E}[X_1 + \cdots + X_n] = n(\frac{1}{n} + \cdots + \frac{1}{2} + 1) =$

Using Indicator Random Variables

Example II: Guessing cards (*with* memory).

Now $\mathbf{Pr}[X_i = 1]$ depends on the current size of the deck.

$\mathbf{E}[X_i] = \mathbf{Pr}[X_i = 1] = 1/(n - i + 1)$

$\Rightarrow \mathbf{E}[X] = \mathbf{E}[X_1 + \cdots + X_n] = \frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{2} + 1 = H_n$

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

$\Rightarrow \mathbf{E}[X] = H_n \in \Theta(\log n)$ *Note that this does depend on n!*

Example III: Collecting goodies

How often do you have to shop to collect all n goodies? (X)

Assume: Each time you get a random goodie. You can't choose.

$X_i :=$ number of times you must shop to get i-th new goodie.

$\mathbf{Pr}[X_i = 1] = (n - i + 1)/n \Rightarrow \mathbf{E}[X_i] = n/(n - i + 1)$

$\Rightarrow \mathbf{E}[X] = \mathbf{E}[X_1 + \cdots + X_n] = n(\frac{1}{n} + \cdots + \frac{1}{2} + 1) = \Theta(n \log n)$

Conditional Probabilities

Las Vegas & Monte Carlo

Example IV: Drug detection (n lockers, $n/2$ with drugs)

Las Vegas & Monte Carlo

Example IV: Drug detection (n lockers, $n/2$ with drugs)

Deterministic approach:

Las Vegas & Monte Carlo

Example IV: Drug detection (n lockers, $n/2$ with drugs)

Deterministic approach:

Need to break $n/2 + 1$ lockers in the worst case.

Las Vegas & Monte Carlo

Example IV: Drug detection (n lockers, $n/2$ with drugs)

Deterministic approach:

Need to break $n/2 + 1$ lockers in the worst case.

If students know your strategy, you must break *exactly* $n/2 + 1$.

Las Vegas & Monte Carlo

Example IV: Drug detection (n lockers, $n/2$ with drugs)

Deterministic approach:

Need to break $n/2+1$ lockers in the worst case.

If students know your strategy, you must break *exactly* $n/2+1$.

Randomization removes the adversary.

Las Vegas & Monte Carlo

Example IV: Drug detection (n lockers, $n/2$ with drugs)

Deterministic approach:

Need to break $n/2+1$ lockers in the worst case.

If students know your strategy, you must break *exactly* $n/2+1$.

Randomization removes the adversary.

RandA:

Las Vegas & Monte Carlo

Example IV: Drug detection (n lockers, $n/2$ with drugs)

Deterministic approach:

Need to break $n/2 + 1$ lockers in the worst case.

If students know your strategy, you must break *exactly* $n/2 + 1$.

Randomization removes the adversary.

RandA: – Compute random permutation of the lockers.

Las Vegas & Monte Carlo

Example IV: Drug detection (n lockers, $n/2$ with drugs)

Deterministic approach:

Need to break $n/2+1$ lockers in the worst case.

If students know your strategy, you must break *exactly* $n/2+1$.

Randomization removes the adversary.

RandA: – Compute random permutation of the lockers.
– Break lockers in this order.

Las Vegas & Monte Carlo

Example IV: Drug detection (n lockers, $n/2$ with drugs)

Deterministic approach:

Need to break $n/2 + 1$ lockers in the worst case.

If students know your strategy, you must break *exactly* $n/2 + 1$.

Randomization removes the adversary.

RandA: – Compute random permutation of the lockers.
– Break lockers in this order.

We break $n/2 + 1$ lockers in w-c, but expect to break fewer.

Las Vegas & Monte Carlo

Example IV: Drug detection (n lockers, $n/2$ with drugs)

Deterministic approach:

Need to break $n/2+1$ lockers in the worst case.

If students know your strategy, you must break *exactly* $n/2+1$.

Randomization removes the adversary.

RandA: – Compute random permutation of the lockers.
– Break lockers in this order.

We break $n/2+1$ lockers in w-c, but expect to break fewer.

RandO: – Compute random permutation of $k \leq \frac{n}{2}+1$ lockers.
– Break lockers in this order.

Las Vegas & Monte Carlo

Example IV: Drug detection (n lockers, $n/2$ with drugs)

Deterministic approach:

Need to break $n/2 + 1$ lockers in the worst case.

If students know your strategy, you must break *exactly* $n/2 + 1$.

Randomization removes the adversary.

RandA: – Compute random permutation of the lockers.
– Break lockers in this order.
We break $n/2 + 1$ lockers in w-c, but expect to break fewer.

RandO: – Compute random permutation of $k \leq \frac{n}{2} + 1$ lockers.
– Break lockers in this order.
We don't damage so many lockers, but may not find any drugs.

Las Vegas & Monte Carlo

Example IV: Drug detection (n lockers, $n/2$ with drugs)

Deterministic approach:

Need to break $n/2+1$ lockers in the worst case.

If students know your strategy, you must break *exactly* $n/2+1$.

Randomization removes the adversary.

RandA: – Compute random permutation of the lockers.
– Break lockers in this order. **Las Vegas Algorithm**
We break $n/2+1$ lockers in w-c, but expect to break fewer.

RandO: – Compute random permutation of $k \leq \frac{n}{2}+1$ lockers.
– Break lockers in this order. **Monte Carlo Algorithm**
We don't damage so many lockers, but may not find any drugs.

Analysis

RandA: expected number of broken lockers $=$

Analysis

RandA: expected number of broken lockers $= 1/(1/2) = 2$

Analysis

RandA: expected number of broken lockers $= 1/(1/2) = 2$

RandO: failure probability for 1 locker $= \quad 1/2$

Analysis

RandA: expected number of broken lockers $= 1/(1/2) = 2$

RandO: failure probability for 1 locker $= \quad 1/2$
failure probability for k lockers $=$

Analysis

RandA: expected number of broken lockers $= 1/(1/2) = 2$

RandO: failure probability for 1 locker $= \quad 1/2$
failure probability for k lockers $= (1/2)^k$

Analysis

RandA: expected number of broken lockers $= 1/(1/2) = 2$

RandO: failure probability for 1 locker $= \quad 1/2$

failure probability for k lockers $= (1/2)^k$

success probability for k lockers $=$

Analysis

RandA: expected number of broken lockers $= 1/(1/2) = 2$

RandO: failure probability for 1 locker $= \quad 1/2$
failure probability for k lockers $= \ (1/2)^k$
success probability for k lockers $= 1 - 2^{-k}$

Analysis

RandA: expected number of broken lockers $= 1/(1/2) = 2$

RandO: failure probability for 1 locker $= \quad 1/2$

failure probability for k lockers $= (1/2)^k$

success probability for k lockers $= 1 - 2^{-k}$

Las Vegas Algorithm

Monte Carlo Algorithm

Analysis

RandA: expected number of broken lockers $= 1/(1/2) = 2$

RandO: failure probability for 1 locker $= \quad 1/2$
failure probability for k lockers $= (1/2)^k$
success probability for k lockers $= 1 - 2^{-k}$

Las Vegas Algorithm

Algorithm returns correct result, but resource (runtime) is a random variable.

Monte Carlo Algorithm

Analysis

RandA: expected number of broken lockers $= 1/(1/2) = 2$

RandO: failure probability for 1 locker $= \quad 1/2$
failure probability for k lockers $= (1/2)^k$
success probability for k lockers $= 1 - 2^{-k}$

Las Vegas Algorithm

Algorithm returns correct result, but resource (runtime) is a random variable.

Monte Carlo Algorithm

Algorithm errs or fails with certain probability, but runtime does not depend on random choices.

Analysis

RandA: expected number of broken lockers $= 1/(1/2) = 2$

RandO: failure probability for 1 locker $= 1/2$

failure probability for k lockers $= (1/2)^k$

success probability for k lockers $= 1 - 2^{-k}$

Las Vegas Algorithm

Algorithm returns correct result, but resource (runtime) is a random variable.

Examples:

Monte Carlo Algorithm

Algorithm errs or fails with certain probability, but runtime does not depend on random choices.

Example:

Analysis

RandA: expected number of broken lockers $= 1/(1/2) = 2$

RandO: failure probability for 1 locker $= \quad 1/2$
failure probability for k lockers $= (1/2)^k$
success probability for k lockers $= 1 - 2^{-k}$

Las Vegas Algorithm

Algorithm returns correct result, but resource (runtime) is a random variable.
Examples: RandomizedQuickSort, RandomizedSelect (Median)

Monte Carlo Algorithm

Algorithm errs or fails with certain probability, but runtime does not depend on random choices.
Example:

Analysis

RandA: expected number of broken lockers $= 1/(1/2) = 2$

RandO: failure probability for 1 locker $= \quad 1/2$
failure probability for k lockers $= (1/2)^k$
success probability for k lockers $= 1 - 2^{-k}$

Las Vegas Algorithm

Algorithm returns correct result, but resource (runtime) is a random variable.
Examples: RandomizedQuickSort, RandomizedSelect (Median)

Monte Carlo Algorithm

Algorithm errs or fails with certain probability, but runtime does not depend on random choices.
Example: Karger's randomized MinCut algorithm

Monte Carlo Example

Example V: Find large number ($\geq$ median, in array of n ints)

Monte Carlo Example

Example V: Find large number ($\geq$ median, in array of n ints)

Deterministic approach:

Monte Carlo Example

Example V: Find large number ($\geq$ median, in array of n ints)

Deterministic approach:

Go through all elements, return maximum.

Monte Carlo Example

Example V: Find large number ($\geq$ median, in array of n ints)

Deterministic approach:

Go through all elements, return maximum.

(Actually, suffices to go through $n/2$ elements.)

Monte Carlo Example

Example V: Find large number ($\geq$ median, in array of n ints)

Deterministic approach:

Go through all elements, return maximum.

(Actually, suffices to go through $n/2$ elements.)

MonteCarloFind(int[] A, int $k \geq 1$)

Monte Carlo Example

Example V: Find large number ($\geq$ median, in array of n ints)

Deterministic approach:

Go through all elements, return maximum.

(Actually, suffices to go through $n/2$ elements.)

MonteCarloFind(int[] A, int $k \geq 1$)

pick $a_1, \dots, a_k \in \{1, \dots, n\}$ u.a.r. [uniformly at random]

Monte Carlo Example

Example V: Find large number ($\geq$ median, in array of n ints)

Deterministic approach:

Go through all elements, return maximum.

(Actually, suffices to go through $n/2$ elements.)

MonteCarloFind(int[] A, int $k \geq 1$)

pick $a_1, \ldots, a_k \in \{1, \ldots, n\}$ u.a.r.

$m = \max\{A[a_1], \ldots, A[a_k]\}$

[uniformly at random]

Monte Carlo Example

Example V: Find large number ($\geq$ median, in array of n ints)

Deterministic approach:

Go through all elements, return maximum.
(Actually, suffices to go through $n/2$ elements.)

MonteCarloFind(int[] A, int $k \geq 1$)
  pick $a_1, \ldots, a_k \in \{1, \ldots, n\}$ u.a.r.
  $m = \max\{A[a_1], \ldots, A[a_k]\}$
  return m

[uniformly at random]

Monte Carlo Example

Example V: Find large number ($\geq$ median, in array of n ints)

Deterministic approach:

Go through all elements, return maximum.
(Actually, suffices to go through $n/2$ elements.)

MonteCarloFind(int[] A, int $k \geq 1$)
pick $a_1, \ldots, a_k \in \{1, \ldots, n\}$ u.a.r.
$m = \max\{A[a_1], \ldots, A[a_k]\}$
return m

[uniformly at random]

The algorithm has error probability $\leq$

Monte Carlo Example

Example V: Find large number ($\geq$ median, in array of n ints)

Deterministic approach:

Go through all elements, return maximum.

(Actually, suffices to go through $n/2$ elements.)

MonteCarloFind(int[] A, int $k \geq 1$)
 pick $a_1, \ldots, a_k \in \{1, \ldots, n\}$ u.a.r.
 $m = \max\{A[a_1], \ldots, A[a_k]\}$
 return m

[uniformly at random]

The algorithm has error probability $\leq 2^{-k}$.

Monte Carlo Example

Example V: Find large number ($\geq$ median, in array of n ints)

Deterministic approach:

Go through all elements, return maximum.

(Actually, suffices to go through $n/2$ elements.)

MonteCarloFind(int[] A, int $k \geq 1$)
  pick $a_1, \ldots, a_k \in \{1, \ldots, n\}$ u.a.r.
  $m = \max\{A[a_1], \ldots, A[a_k]\}$
  return m

[uniformly at random]

The algorithm has error probability $\leq 2^{-k}$.

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

Monte Carlo Example

Example V: Find large number ($\geq$ median, in array of n ints)

Deterministic approach:

Go through all elements, return maximum.

(Actually, suffices to go through $n/2$ elements.)

MonteCarloFind(int[] A, int $k \geq 1$)
pick $a_1, \ldots, a_k \in \{1, \ldots, n\}$ u.a.r.
$m = \max\{A[a_1], \ldots, A[a_k]\}$
return m

[uniformly at random]

The algorithm has error probability $\leq 2^{-k}$.

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

$\Rightarrow$ Error probability $\leq$

Monte Carlo Example

Example V: Find large number ($\geq$ median, in array of n ints)

Deterministic approach:

Go through all elements, return maximum.

(Actually, suffices to go through $n/2$ elements.)

MonteCarloFind(int[] A, int $k \geq 1$)
  pick $a_1, \ldots, a_k \in \{1, \ldots, n\}$ u.a.r. [uniformly at random]
  $m = \max\{A[a_1], \ldots, A[a_k]\}$
  return m

The algorithm has error probability $\leq 2^{-k}$.

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

$\Rightarrow$ Error probability $\leq n^{-c}$

Monte Carlo Example

Example V: Find large number ($\geq$ median, in array of n ints)

Deterministic approach:

Go through all elements, return maximum.
(Actually, suffices to go through $n/2$ elements.)

MonteCarloFind(int[] A, int $k \geq 1$)
  pick $a_1, \ldots, a_k \in \{1, \ldots, n\}$ u.a.r.
  $m = \max\{A[a_1], \ldots, A[a_k]\}$
  return m

[uniformly at random]

The algorithm has error probability $\leq 2^{-k}$.

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

$\Rightarrow$ Error probability $\leq n^{-c}$, runtime $\in$

Monte Carlo Example

Example V: Find large number ($\geq$ median, in array of n ints)

Deterministic approach:

Go through all elements, return maximum.
(Actually, suffices to go through $n/2$ elements.)

runtime $\Theta(n)$

MonteCarloFind(int[] A, int $k \geq 1$)
pick $a_1, \ldots, a_k \in \{1, \ldots, n\}$ u.a.r.
$m = \max\{A[a_1], \ldots, A[a_k]\}$
return m

[uniformly at random]

The algorithm has error probability $\leq 2^{-k}$.

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

$\Rightarrow$ Error probability $\leq n^{-c}$, runtime $\in O(\log n)$

Las Vegas Example

Example VI: Find repeated element
(array of $n \geq 4$ ints, $n/2$ distinct, $n/2$ identical)

Las Vegas Example

Example VI: Find repeated element
(array of $n \geq 4$ ints, $n/2$ distinct, $n/2$ identical)

Deterministic approach:

Las Vegas Example

Example VI: Find repeated element
(array of $n \geq 4$ ints, $n/2$ distinct, $n/2$ identical)

Deterministic approach:

Sort and find repeated element. $\Theta(n \log n)$ time

Las Vegas Example

Example VI: Find repeated element
(array of $n \geq 4$ ints, $n/2$ distinct, $n/2$ identical)

Deterministic approach:

Sort and find repeated element. $\Theta(n \log n)$ time

Faster: Find median. $\Theta(n)$ time

Las Vegas Example

Example VI: Find repeated element
(array of $n \geq 4$ ints, $n/2$ distinct, $n/2$ identical)

Deterministic approach:

Sort and find repeated element. $\Theta(n \log n)$ time

Faster: Find median. $\Theta(n)$ time

LasVegasFindRepeated(int[] A)

Las Vegas Example

Example VI: Find repeated element
(array of $n \geq 4$ ints, $n/2$ distinct, $n/2$ identical)

Deterministic approach:

Sort and find repeated element. $\Theta(n \log n)$ time

Faster: Find median. $\Theta(n)$ time

LasVegasFindRepeated(int[] A)
while true **do**

Las Vegas Example

Example VI: Find repeated element
(array of $n \geq 4$ ints, $n/2$ distinct, $n/2$ identical)

Deterministic approach:

Sort and find repeated element. $\Theta(n \log n)$ time

Faster: Find median. $\Theta(n)$ time

LasVegasFindRepeated(int[] A)
while true **do**
pick $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, n\} \setminus \{i\}$, both u.a.r.

Las Vegas Example

Example VI: Find repeated element
(array of $n \geq 4$ ints, $n/2$ distinct, $n/2$ identical)

Deterministic approach:

Sort and find repeated element. $\Theta(n \log n)$ time

Faster: Find median. $\Theta(n)$ time

LasVegasFindRepeated(int[] A)
while true **do**
pick $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, n\} \setminus \{i\}$, both u.a.r.

Las Vegas Example

Example VI: Find repeated element
(array of $n \geq 4$ ints, $n/2$ distinct, $n/2$ identical)

Deterministic approach:

Sort and find repeated element. $\Theta(n \log n)$ time

Faster: Find median. $\Theta(n)$ time

LasVegasFindRepeated(int[ ] A)
while true **do**
pick $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, n\} \setminus \{i\}$, both u.a.r.
if $A[i] == A[j]$ **then return** $A[i]$

Las Vegas Example

Example VI: Find repeated element
(array of $n \geq 4$ ints, $n/2$ distinct, $n/2$ identical)

Deterministic approach:

Sort and find repeated element. $\Theta(n \log n)$ time

Faster: Find median. $\Theta(n)$ time

LasVegasFindRepeated(int[ ] A)
while true **do**
  pick $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, n\} \setminus \{i\}$, both u.a.r.
  if $A[i] == A[j]$ **then return** $A[i]$

Algorithm always returns correct result –

Las Vegas Example

Example VI: Find repeated element
(array of $n \geq 4$ ints, $n/2$ distinct, $n/2$ identical)

Deterministic approach:

Sort and find repeated element. $\Theta(n \log n)$ time

Faster: Find median. $\Theta(n)$ time

```
LasVegasFindRepeated(int[] A)
   while true do
      pick i ∈ {1, . . . , n} and j ∈ {1, . . . , n} \ {i}, both u.a.r.
      if A[i] == A[j] then return A[i]
```

Algorithm always returns correct result – but may take forever.

Las Vegas Example

Example VI: Find repeated element
(array of $n \geq 4$ ints, $n/2$ distinct, $n/2$ identical)

Deterministic approach:

Sort and find repeated element. $\Theta(n \log n)$ time

Faster: Find median. $\Theta(n)$ time

LasVegasFindRepeated(int[] A)
while true **do**
  pick $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, n\} \setminus \{i\}$, both u.a.r.
  if $A[i] == A[j]$ **then return** $A[i]$

Algorithm always returns correct result – but may take forever.

Success probability $=$

Las Vegas Example

Example VI: Find repeated element
(array of $n \geq 4$ ints, $n/2$ distinct, $n/2$ identical)

Deterministic approach:

Sort and find repeated element. $\Theta(n \log n)$ time

Faster: Find median. $\Theta(n)$ time

LasVegasFindRepeated(int[] A)
while true **do**
  pick $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, n\} \setminus \{i\}$, both u.a.r.
  if $A[i] == A[j]$ **then return** $A[i]$

Algorithm always returns correct result – but may take forever.

$$\text{Success probability} = \frac{n/2}{n} \cdot \frac{n/2 - 1}{n - 1} \approx$$

Las Vegas Example

Example VI: Find repeated element
(array of $n \geq 4$ ints, $n/2$ distinct, $n/2$ identical)

Deterministic approach:

Sort and find repeated element. $\Theta(n \log n)$ time

Faster: Find median. $\Theta(n)$ time

```
LasVegasFindRepeated(int[] A)
  while true do
    pick i ∈ {1, ..., n} and j ∈ {1, ..., n} \ {i}, both u.a.r.
    if A[i] == A[j] then return A[i]
```

Algorithm always returns correct result – but may take forever.

$$\text{Success probability} = \frac{n/2}{n} \cdot \frac{n/2 - 1}{n - 1} \approx \frac{1}{4}.$$

Las Vegas Example

Example VI: Find repeated element
(array of $n \geq 4$ ints, $n/2$ distinct, $n/2$ identical)

Deterministic approach:

Sort and find repeated element. $\Theta(n \log n)$ time

Faster: Find median. $\Theta(n)$ time

```
LasVegasFindRepeated(int[] A)
  while true do
    pick i ∈ {1, ..., n} and j ∈ {1, ..., n} \ {i}, both u.a.r.
    if A[i] == A[j] then return A[i]
```

Algorithm always returns correct result – but may take forever.

Success probability $= \dfrac{n/2}{n} \cdot \dfrac{n/2-1}{n-1} \approx \dfrac{1}{4}$.

$\Rightarrow$ Expected number of iterations $\approx$

Las Vegas Example

Example VI: Find repeated element
(array of $n \geq 4$ ints, $n/2$ distinct, $n/2$ identical)

Deterministic approach:

Sort and find repeated element. $\Theta(n \log n)$ time

Faster: Find median. $\Theta(n)$ time

LasVegasFindRepeated(int[] A)
while true **do**
pick $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, n\} \setminus \{i\}$, both u.a.r.
if $A[i] == A[j]$ **then return** $A[i]$

Algorithm always returns correct result – but may take forever.

$$\text{Success probability} = \frac{n/2}{n} \cdot \frac{n/2 - 1}{n - 1} \approx \frac{1}{4}.$$

$\Rightarrow$ Expected number of iterations $\approx 4 \in O(1)$.

From Las Vegas to Monte Carlo

Theorem. (Markov inequality)
For any non-negative random variable X and $t \geq 1$,

$$\mathbf{Pr}[X > t] \leq$$

From Las Vegas to Monte Carlo

Theorem. (Markov inequality)
For any non-negative random variable X and $t \geq 1$,

$$\mathbf{Pr}[X > t] \leq \mathbf{E}[X]/t.$$

From Las Vegas to Monte Carlo

Theorem. (Markov inequality)
For any non-negative random variable X and $t \geq 1$,

$$\mathbf{Pr}[X > t] \leq \mathbf{E}[X]/t.$$

Equivalently,

$$\mathbf{Pr}[X > t \cdot \mathbf{E}[X]] \leq$$

From Las Vegas to Monte Carlo

Theorem. (Markov inequality)
For any non-negative random variable X and $t \geq 1$,

$$\mathbf{Pr}[X > t] \leq \mathbf{E}[X]/t.$$

Equivalently,

$$\mathbf{Pr}[X > t \cdot \mathbf{E}[X]] \leq 1/t.$$

From Las Vegas to Monte Carlo

Theorem. (Markov inequality)
For any non-negative random variable X and $t \geq 1$,

$$\mathbf{Pr}[X > t] \leq \mathbf{E}[X]/t.$$

Equivalently,

$$\mathbf{Pr}[X > t \cdot \mathbf{E}[X]] \leq 1/t.$$

Let X be the running time of a Las Vegas algorithm and $f(n) = \mathbf{E}[X]$ its expected running time and $\alpha > 1$. Then

$$\mathbf{Pr}[X > \alpha \cdot f(n)] \leq$$

From Las Vegas to Monte Carlo

Theorem. (Markov inequality)
For any non-negative random variable X and $t \geq 1$,

$$\mathbf{Pr}[X > t] \leq \mathbf{E}[X]/t.$$

Equivalently,

$$\mathbf{Pr}[X > t \cdot \mathbf{E}[X]] \leq 1/t.$$

Let X be the running time of a Las Vegas algorithm and $f(n) = \mathbf{E}[X]$ its expected running time and $\alpha > 1$. Then

$$\mathbf{Pr}[X > \alpha \cdot f(n)] \leq 1/\alpha$$

From Las Vegas to Monte Carlo

Theorem. (Markov inequality)
For any non-negative random variable X and $t \geq 1$,

$$\mathbf{Pr}[X > t] \leq \mathbf{E}[X]/t.$$

Equivalently,

$$\mathbf{Pr}[X > t \cdot \mathbf{E}[X]] \leq 1/t.$$

Let X be the running time of a Las Vegas algorithm and $f(n) = \mathbf{E}[X]$ its expected running time and $\alpha > 1$. Then

$$\mathbf{Pr}[X > \alpha \cdot f(n)] \leq 1/\alpha$$

So the probability that the Las Vegas algorithm does not find a solution in the first $\alpha \cdot f(n)$ steps is less than $1/\alpha$, which is the error probability of the respective Monte Carlo algorithm.

Closest Pair

Given a set $P = \{p_1, \ldots, p_n\}$ of points in the plane,
find a pair in $\binom{P}{2}$ whose Euclidean distance is minimum.

Closest Pair

Given a set $P = \{p_1, \ldots, p_n\}$ of points in the plane, find a pair in $\binom{P}{2}$ whose Euclidean distance is minimum.

ADS: Deterministic divide-and-conquer algorithm,
worst-case runtime $O(n \log n)$.

Closest Pair

Given a set $P = \{p_1, \ldots, p_n\}$ of points in the plane,
find a pair in $\binom{P}{2}$ whose Euclidean distance is minimum.

ADS: Deterministic divide-and-conquer algorithm,
worst-case runtime $O(n \log n)$.

Element Uniqueness Problem: Given n numbers, are they unique?

Closest Pair

Given a set $P = \{p_1, \ldots, p_n\}$ of points in the plane,
find a pair in $\binom{P}{2}$ whose Euclidean distance is minimum.

ADS: Deterministic divide-and-conquer algorithm,
worst-case runtime $O(n \log n)$.

Element Uniqueness Problem: Given n numbers, are they unique?
Cannot be solved in $o(n \log n)$ w-c time.
(under some assumption concerning the arithmetic model)

Closest Pair

Given a set $P = \{p_1, \dots, p_n\}$ of points in the plane, find a pair in $\binom{P}{2}$ whose Euclidean distance is minimum.

ADS: Deterministic divide-and-conquer algorithm,
worst-case runtime $O(n \log n)$.

Element Uniqueness Problem: Given n numbers, are they unique?
Cannot be solved in $o(n \log n)$ w-c time.

(under some assumption concerning the arithmetic model)

$\Rightarrow$ Closest Pair cannot be solved in $o(n \log n)$ w-c time.

(under the same assumption concerning the arithmetic model)

A Randomized Incremental Algorithm

Assume: – Can use the floor function in $O(1)$ time.
– Can use hashing in $O(1)$ time.

A Randomized Incremental Algorithm

Assume: – Can use the floor function in $O(1)$ time.
– Can use hashing in $O(1)$ time.

Define: $P_i = \{p_1, \ldots, p_i\}$
$\delta_i =$ distance of the closest pair in P_i.

A Randomized Incremental Algorithm

Assume: – Can use the floor function in $O(1)$ time.
– Can use hashing in $O(1)$ time.

Define: $P_i = \{p_1, \ldots, p_i\}$
δ_i = distance of the closest pair in P_i.

Problem: Given δ_{i-1}, how can we compute δ_i?

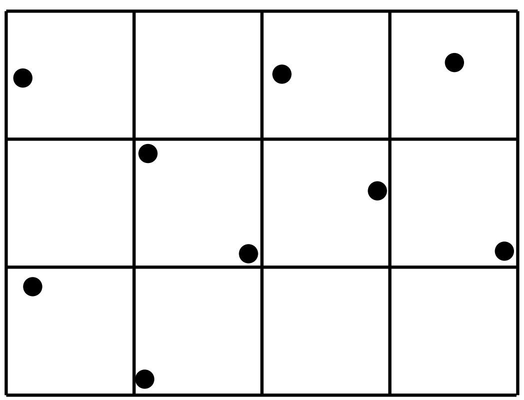
A Randomized Incremental Algorithm

Assume: – Can use the floor function in $O(1)$ time.
– Can use hashing in $O(1)$ time.

Define: $P_i = \{p_1, \ldots, p_i\}$
δ_i = distance of the closest pair in P_i.

Problem: Given δ_{i-1}, how can we compute δ_i?

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



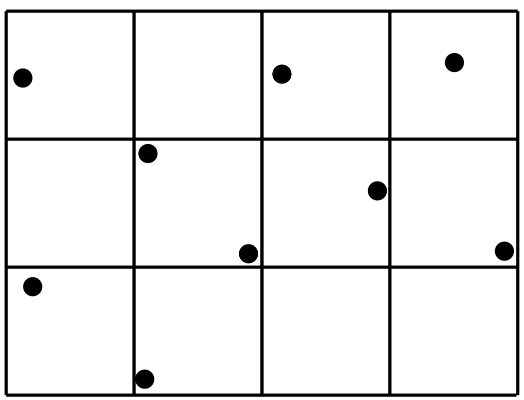
A Randomized Incremental Algorithm

Assume: – Can use the floor function in $O(1)$ time.
– Can use hashing in $O(1)$ time.

Define: $P_i = \{p_1, \ldots, p_i\}$
$\delta_i =$ distance of the closest pair in P_i.

Problem: Given δ_{i-1}, how can we compute δ_i?

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



How many points in P_{i-1} can lie in the same grid cell?

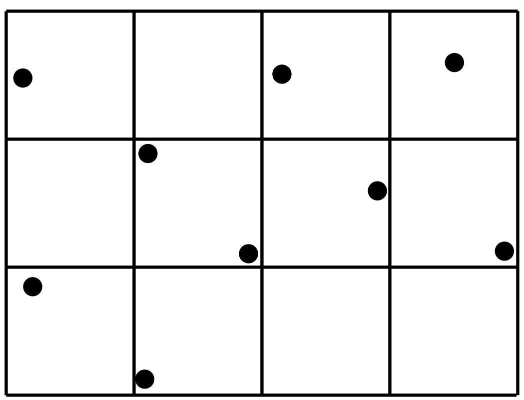
A Randomized Incremental Algorithm

Assume: – Can use the floor function in $O(1)$ time.
– Can use hashing in $O(1)$ time.

Define: $P_i = \{p_1, \ldots, p_i\}$
$\delta_i =$ distance of the closest pair in P_i.

Problem: Given δ_{i-1}, how can we compute δ_i?

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



How many points in P_{i-1} can lie in the same grid cell?

At most 4 (in the corners).

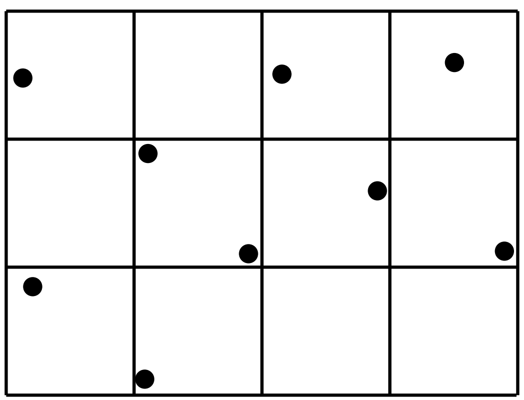
A Randomized Incremental Algorithm

Assume: – Can use the floor function in $O(1)$ time.
– Can use hashing in $O(1)$ time.

Define: $P_i = \{p_1, \dots, p_i\}$
δ_i = distance of the closest pair in P_i.

Problem: Given δ_{i-1}, how can we compute δ_i?

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



How many points in P_{i-1} can lie in the same grid cell?

At most 4 (in the corners).

After finding p_i's cell, need to check only $O(1)$ points in vicinity.

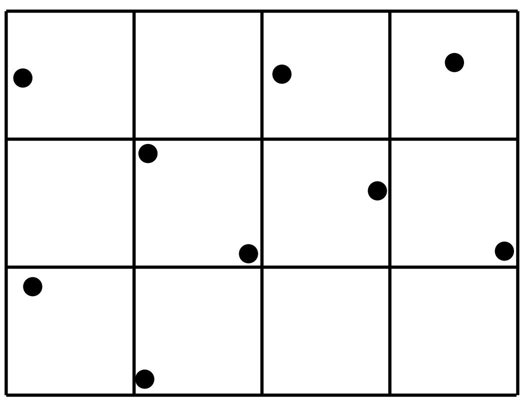
A Randomized Incremental Algorithm

Assume: – Can use the floor function in $O(1)$ time.
– Can use hashing in $O(1)$ time.

Define: $P_i = \{p_1, \ldots, p_i\}$
δ_i = distance of the closest pair in P_i.

Problem: Given δ_{i-1}, how can we compute δ_i?

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



How many points in P_{i-1} can lie in the same grid cell?

At most 4 (in the corners).

After finding p_i's cell, need to check only $O(1)$ points in vicinity.

Cases:

- $\delta_i < \delta_{i-1}$:
- $\delta_i = \delta_{i-1}$:

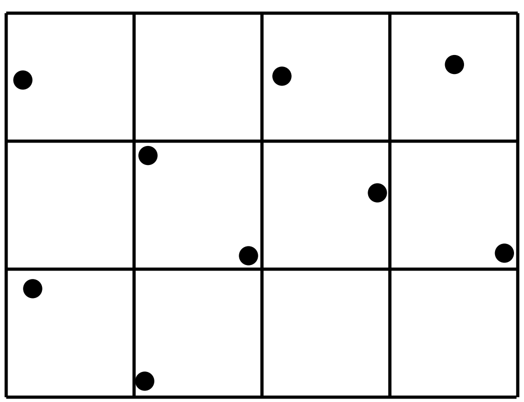
A Randomized Incremental Algorithm

Assume: – Can use the floor function in $O(1)$ time.
– Can use hashing in $O(1)$ time.

Define: $P_i = \{p_1, \dots, p_i\}$
δ_i = distance of the closest pair in P_i.

Problem: Given δ_{i-1}, how can we compute δ_i?

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



How many points in P_{i-1} can lie in the same grid cell?

At most 4 (in the corners).

After finding p_i's cell, need to check only $O(1)$ points in vicinity.

Cases:
- $\delta_i < \delta_{i-1}$: Need to recompute grid in $O(i)$ time.
- $\delta_i = \delta_{i-1}$:

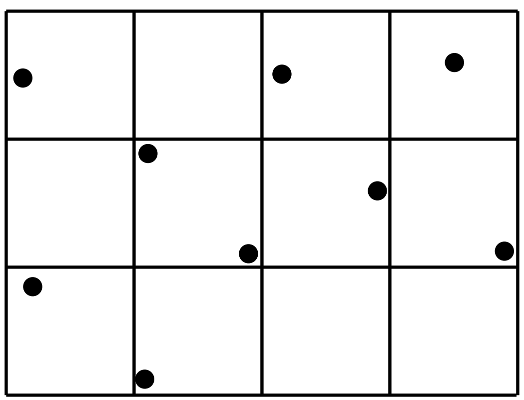
A Randomized Incremental Algorithm

Assume: – Can use the floor function in $O(1)$ time.
– Can use hashing in $O(1)$ time.

Define: $P_i = \{p_1, \dots, p_i\}$
$\delta_i =$ distance of the closest pair in P_i.

Problem: Given δ_{i-1}, how can we compute δ_i?

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



How many points in P_{i-1} can lie in the same grid cell?

At most 4 (in the corners).

After finding p_i's cell, need to check only $O(1)$ points in vicinity.

Cases:
- $\delta_i < \delta_{i-1}$: Need to recompute grid in $O(i)$ time.
- $\delta_i = \delta_{i-1}$: Need to store p_i in its cell in $O(1)$ time.

Backwards Analysis

What is the w-c running time of the algorithm?

Backwards Analysis

What is the w-c running time of the algorithm? $\Theta(n^2)$

Backwards Analysis

What is the w-c running time of the algorithm? $\Theta(n^2)$

How do we randomize?

Backwards Analysis

What is the w-c running time of the algorithm? $\Theta(n^2)$

How do we randomize? Randomly permute points at beginning.

Backwards Analysis

What is the w-c running time of the algorithm? $\Theta(n^2)$
How do we randomize? Randomly permute points at beginning.

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?

- The closest distance in P_i is unique:
- One point has the same smallest distance to several points:
- There are at least two disjoint closest pairs:

Backwards Analysis

What is the w-c running time of the algorithm? $\Theta(n^2)$
How do we randomize? Randomly permute points at beginning.

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?

- The closest distance in P_i is unique: 2 points.
- One point has the same smallest distance to several points:
- There are at least two disjoint closest pairs:

Backwards Analysis

What is the w-c running time of the algorithm? $\Theta(n^2)$
How do we randomize? Randomly permute points at beginning.

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?

- The closest distance in P_i is unique: 2 points.
- One point has the same smallest distance to several points: 1 point.
- There are at least two disjoint closest pairs:

Backwards Analysis

What is the w-c running time of the algorithm? $\Theta(n^2)$
How do we randomize? Randomly permute points at beginning.

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?

- The closest distance in P_i is unique: 2 points.
- One point has the same smallest distance to several points: 1 point.
- There are at least two disjoint closest pairs: 0 points.

Backwards Analysis

What is the w-c running time of the algorithm? $\Theta(n^2)$
How do we randomize? Randomly permute points at beginning.

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?

- The closest distance in P_i is unique: 2 points.
- One point has the same smallest distance to several points: 1 point.
- There are at least two disjoint closest pairs: 0 points.

Let X_i be the work for adding p_i.
$\Rightarrow \mathbf{E}[X_i] \leq$

Backwards Analysis

What is the w-c running time of the algorithm? $\Theta(n^2)$
How do we randomize? Randomly permute points at beginning.

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?

- The closest distance in P_i is unique: 2 points.
- One point has the same smallest distance to several points: 1 point.
- There are at least two disjoint closest pairs: 0 points.

Let X_i be the work for adding p_i.
$\Rightarrow \mathbf{E}[X_i] \leq 2/i \cdot O(i) + (i-2)/i \cdot O(1) = O(1)$

Backwards Analysis

What is the w-c running time of the algorithm? $\Theta(n^2)$
How do we randomize? Randomly permute points at beginning.

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?

- The closest distance in P_i is unique: 2 points.
- One point has the same smallest distance to several points: 1 point.
- There are at least two disjoint closest pairs: 0 points.

Let X_i be the work for adding p_i.
$\Rightarrow \mathbf{E}[X_i] \leq 2/i \cdot O(i) + (i-2)/i \cdot O(1) = O(1)$

Let X be the total work done by the algorithm.

$\Rightarrow \mathbf{E}[X] =$

Backwards Analysis

What is the w-c running time of the algorithm? $\Theta(n^2)$
How do we randomize? Randomly permute points at beginning.

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?

- The closest distance in P_i is unique: 2 points.
- One point has the same smallest distance to several points: 1 point.
- There are at least two disjoint closest pairs: 0 points.

Let X_i be the work for adding p_i.
$\Rightarrow \mathbf{E}[X_i] \leq 2/i \cdot O(i) + (i-2)/i \cdot O(1) = O(1)$

Let X be the total work done by the algorithm.

$\Rightarrow \mathbf{E}[X] = \mathbf{E}[X_1 + \cdots + X_n]$

Backwards Analysis

What is the w-c running time of the algorithm? $\Theta(n^2)$
How do we randomize? Randomly permute points at beginning.

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?

- The closest distance in P_i is unique: 2 points.
- One point has the same smallest distance to several points: 1 point.
- There are at least two disjoint closest pairs: 0 points.

Let X_i be the work for adding p_i.
$\Rightarrow \mathbf{E}[X_i] \leq 2/i \cdot O(i) + (i-2)/i \cdot O(1) = O(1)$

Let X be the total work done by the algorithm.
$\Rightarrow \mathbf{E}[X] = \mathbf{E}[X_1 + \cdots + X_n] \in O(n)$ $\square$