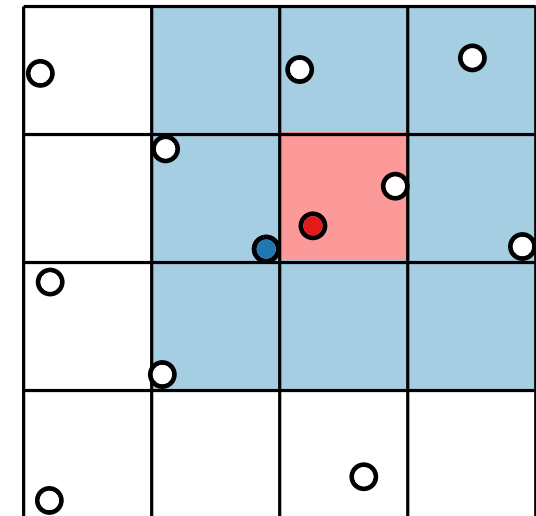
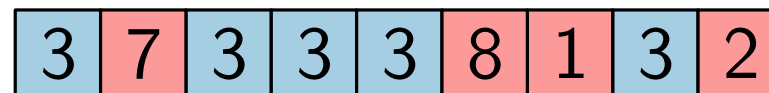
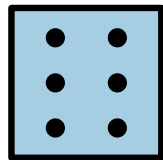
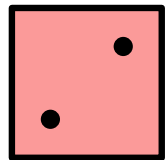


Advanced Algorithms

Randomized Algorithms

An introduction



Basic Definitions

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

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

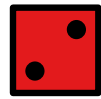
$P: \Omega \rightarrow [0, 1]$ assigns a **probability** $P(\omega)$ to each $\omega \in \Omega$ s.t. $\sum_{\omega \in \Omega} P(\omega) = 1$.

Basic Definitions

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

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

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



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

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

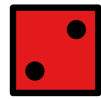
Basic Definitions

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

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

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

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



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

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

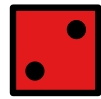
Basic Definitions

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

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

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

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



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

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

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

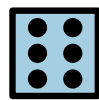
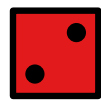
Basic Definitions

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

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

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

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



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

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

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

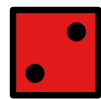
$$P(A) = 6 \cdot \frac{1}{36} = \frac{1}{6}$$

Basic Definitions

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

Basic Definitions

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

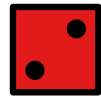


Example. Rolling a **red** and a **blue** fair six-sided die.

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

Basic Definitions

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



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

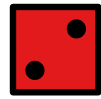
$$\Omega = \{(1, 1), (1, 2), (1, 3), \dots, (6, 6)\}, \quad P((i, j)) = \frac{1}{6} \cdot \frac{1}{6} = \frac{1}{36} \text{ for each } (i, j) \in \Omega$$

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

Basic Definitions

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

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



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

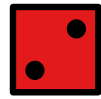
$$\Omega = \{(1, 1), (1, 2), (1, 3), \dots, (6, 6)\}, \quad P((i, j)) = \frac{1}{6} \cdot \frac{1}{6} = \frac{1}{36} \text{ for each } (i, j) \in \Omega$$

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

Basic Definitions

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

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



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

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

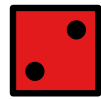
$$X((i, j)) = \max\{i, j\}$$

$$P[X = 1] =$$

Basic Definitions

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

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



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

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

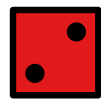
$$X((i, j)) = \max\{i, j\}$$

$$P[X = 1] = \frac{1}{36}, \quad P[X = 2] =$$

Basic Definitions

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

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



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

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

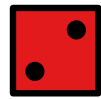
$$X((i, j)) = \max\{i, j\}$$

$$P[X = 1] = \frac{1}{36}, \quad P[X = 2] = \frac{3}{36}, \quad P[X = 3] =$$

Basic Definitions

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

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



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

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

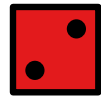
$$X((i, j)) = \max\{i, j\}$$

$$P[X = 1] = \frac{1}{36}, \quad P[X = 2] = \frac{3}{36}, \quad P[X = 3] = \frac{5}{36}, \quad \dots, \quad P[X = 6] =$$

Basic Definitions

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

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



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

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

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

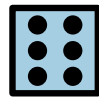
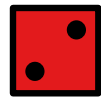
$$P[X = 1] = \frac{1}{36}, \quad P[X = 2] = \frac{3}{36}, \quad P[X = 3] = \frac{5}{36}, \quad \dots, \quad P[X = 6] = \frac{11}{36}$$

Basic Definitions

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

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

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



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

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

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

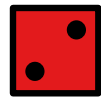
$$P[X = 1] = \frac{1}{36}, \quad P[X = 2] = \frac{3}{36}, \quad P[X = 3] = \frac{5}{36}, \quad \dots, \quad P[X = 6] = \frac{11}{36}$$

Basic Definitions

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

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

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



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

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

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

$$P[X = 1] = \frac{1}{36}, \quad P[X = 2] = \frac{3}{36}, \quad P[X = 3] = \frac{5}{36}, \quad \dots, \quad P[X = 6] = \frac{11}{36}$$

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

Linearity of Expectation

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

Linearity of Expectation

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

Linearity of expectation: $E[X_1 + X_2 + \dots + X_n] = E[X_1] + E[X_2] + \dots + E[X_n]$

Linearity of Expectation

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

Linearity of expectation: $E[X_1 + X_2 + \dots + X_n] = E[X_1] + E[X_2] + \dots + E[X_n]$

Proof of correctness:

$$E[X_1 + X_2 + \dots + X_n] =$$

Linearity of Expectation

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

Linearity of expectation: $E[X_1 + X_2 + \dots + X_n] = E[X_1] + E[X_2] + \dots + E[X_n]$

Proof of correctness:

$$E[X_1 + X_2 + \dots + X_n] = \sum_{\omega \in \Omega} P[\omega] \cdot \sum_{i=1}^n X_i(\omega) =$$

Linearity of Expectation

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

Linearity of expectation: $E[X_1 + X_2 + \dots + X_n] = E[X_1] + E[X_2] + \dots + E[X_n]$

Proof of correctness:

$$E[X_1 + X_2 + \dots + X_n] = \sum_{\omega \in \Omega} P[\omega] \cdot \sum_{i=1}^n X_i(\omega) = \sum_{\omega \in \Omega} \sum_{i=1}^n P[\omega] \cdot X_i(\omega)$$

Linearity of Expectation

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

Linearity of expectation: $E[X_1 + X_2 + \dots + X_n] = E[X_1] + E[X_2] + \dots + E[X_n]$

Proof of correctness:

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

Linearity of Expectation

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

Linearity of expectation: $E[X_1 + X_2 + \dots + X_n] = E[X_1] + E[X_2] + \dots + E[X_n]$

Proof of correctness:

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

Linearity of Expectation

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

Linearity of expectation: $E[X_1 + X_2 + \dots + X_n] = E[X_1] + E[X_2] + \dots + E[X_n]$

Proof of correctness:

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

Using Indicator Random Variables (I)

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

```
FINDMAX(int[] A)
   $m := A[1]$ 
  for  $i = 2$  to  $n$  do
    if  $A[i] > m$  then
       $m := A[i]$ 
  return  $m$ 
```

Using Indicator Random Variables (I)

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

How often is the maximum m updated?

```
FINDMAX(int[] A)
   $m := A[1]$ 
  for  $i = 2$  to  $n$  do
    if  $A[i] > m$  then
       $m := A[i]$ 
  return  $m$ 
```

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

```
FINDMAX(int[] A)
   $m := A[1]$ 
  for  $i = 2$  to  $n$  do
    if  $A[i] > m$  then
       $m := A[i]$ 
  return  $m$ 
```

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

Assume that the integers in A are randomly permuted.

```
FINDMAX(int[] A)
   $m := A[1]$ 
  for  $i = 2$  to  $n$  do
    if  $A[i] > m$  then
       $m := A[i]$ 
  return  $m$ 
```

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

Assume that the integers in A are randomly permuted.

Let X denote the random variable that counts the number of times m is updated. For $i \in \{1, \dots, n\}$, let

```
FINDMAX(int[] A)
  m := A[1]
  for i = 2 to n do
    if A[i] > m then
      m := A[i]
  return m
```

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

Assume that the integers in A are randomly permuted.

Let X denote the random variable that counts the number of times m is updated. For $i \in \{1, \dots, n\}$, let

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

```

FINDMAX(int[]  $A$ )
   $m := A[1]$ 
  for  $i = 2$  to  $n$  do
    if  $A[i] > m$  then
       $m := A[i]$ 
  return  $m$ 

```

indicator random variable

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

Assume that the integers in A are randomly permuted.

Let X denote the random variable that counts the number of times m is updated. For $i \in \{1, \dots, n\}$, let

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

indicator random variable

Observation. $X = X_1 + X_2 + \dots + X_n$

```

FINDMAX(int[] A)
  m := A[1]
  for i = 2 to n do
    if A[i] > m then
      m := A[i]
  return m
  
```

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

Assume that the integers in A are randomly permuted.

Let X denote the random variable that counts the number of times m is updated. For $i \in \{1, \dots, n\}$, let

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

indicator random variable

Observation. $X = X_1 + X_2 + \dots + X_n$

$$P[X_i = 1] = \frac{1}{i}$$

```

FINDMAX(int[] A)
  m := A[1]
  for i = 2 to n do
    if A[i] > m then
      m := A[i]
  return m

```

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

Assume that the integers in A are randomly permuted.

Let X denote the random variable that counts the number of times m is updated. For $i \in \{1, \dots, n\}$, let

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

indicator random variable

Observation. $X = X_1 + X_2 + \dots + X_n$

$$P[X_i = 1] = \frac{1}{i} \quad \Rightarrow \quad E[X_i] = 0 + 1 \cdot \frac{1}{i} = \frac{1}{i}$$

```

FINDMAX(int[] A)
  m := A[1]
  for i = 2 to n do
    if A[i] > m then
      m := A[i]
  return m
  
```

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

Assume that the integers in A are randomly permuted.

Let X denote the random variable that counts the number of times m is updated. For $i \in \{1, \dots, n\}$, let

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

indicator random variable

```
FINDMAX(int[] A)
  m := A[1]
  for i = 2 to n do
    if A[i] > m then
      m := A[i]
  return m
```

Observation. $X = X_1 + X_2 + \dots + X_n$

$$P[X_i = 1] = \frac{1}{i} \Rightarrow E[X_i] = 0 + 1 \cdot \frac{1}{i} = \frac{1}{i}$$

$$E[X] = E[X_1] + E[X_2] + \dots + E[X_n] = 1 + \frac{1}{2} + \dots + \frac{1}{n}$$

←
linearity of expectation

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

Assume that the integers in A are randomly permuted.

Let X denote the random variable that counts the number of times m is updated. For $i \in \{1, \dots, n\}$, let

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

Observation. $X = X_1 + X_2 + \dots + X_n$

$$P[X_i = 1] = \frac{1}{i} \Rightarrow E[X_i] = 0 + 1 \cdot \frac{1}{i} = \frac{1}{i}$$

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

← linearity of expectation

```

FINDMAX(int[] A)
  m := A[1]
  for i = 2 to n do
    if A[i] > m then
      m := A[i]
  return m

```

indicator random variable

H_n is the n -th harmonic number;

$$\ln(n+1) \leq H_n \leq \ln(n) + 1.$$



Playing until You Win

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

Playing until You Win

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

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

Playing until You Win

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

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

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

Playing until You Win

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

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

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

We repeat such an experiment multiple times and assume that the outcomes are independent from each other.

Playing until You Win

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

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

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

We repeat such an experiment multiple times and assume that the outcomes are independent from each other.

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

Playing until You Win

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

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

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

We repeat such an experiment multiple times and assume that the outcomes are independent from each other.

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

$$P[X = j] =$$

Playing until You Win

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

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

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

We repeat such an experiment multiple times and assume that the outcomes are independent from each other.

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

$$P[X = j] = q^{j-1}p$$

Playing until You Win

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

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

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

We repeat such an experiment multiple times and assume that the outcomes are independent from each other.

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

$$P[X = j] = q^{j-1}p$$

$$\Rightarrow E[X] =$$

Playing until You Win

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

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

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

We repeat such an experiment multiple times and assume that the outcomes are independent from each other.

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

$$P[X = j] = q^{j-1}p$$

$$\Rightarrow E[X] = \sum_{j=0}^{\infty} j \cdot q^{j-1}p =$$

Playing until You Win

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

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

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

We repeat such an experiment multiple times and assume that the outcomes are independent from each other.

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

$$P[X = j] = q^{j-1}p$$

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

Playing until You Win

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

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

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

We repeat such an experiment multiple times and assume that the outcomes are independent from each other.

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

$$P[X = j] = q^{j-1} p$$

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

Playing until You Win

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

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

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

We repeat such an experiment multiple times and assume that the outcomes are independent from each other.

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

$$P[X = j] = q^{j-1} p$$

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

Playing until You Win

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

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

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

We repeat such an experiment multiple times and assume that the outcomes are independent from each other.

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

$$P[X = j] = q^{j-1} p$$

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

Playing until You Win

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

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

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

We repeat such an experiment multiple times and assume that the outcomes are independent from each other.

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

$$P[X = j] = q^{j-1} p$$

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

Playing until You Win

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

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

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

We repeat such an experiment multiple times and assume that the outcomes are independent from each other.

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.

$$P[X = j] = q^{j-1} p$$

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

Using Indicator Random Variables (II)

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

Using Indicator Random Variables (II)

Each time you buy groceries at your local supermarket for more than 10 €, 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

Using Indicator Random Variables (II)

Each time you buy groceries at your local supermarket for more than 10 €, 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 $p_i =$

Using Indicator Random Variables (II)

Each time you buy groceries at your local supermarket for more than 10 €, 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 $p_i = \frac{n - (i - 1)}{n}$.

Using Indicator Random Variables (II)

Each time you buy groceries at your local supermarket for more than 10 €, 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

Using Indicator Random Variables (II)

Each time you buy groceries at your local supermarket for more than 10 €, 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

$$\Rightarrow E[X_i] =$$

Using Indicator Random Variables (II)

Each time you buy groceries at your local supermarket for more than 10 €, 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

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

Using Indicator Random Variables (II)

Each time you buy groceries at your local supermarket for more than 10 €, 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

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

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

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.

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:

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.

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.)

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:
 $\Theta(n)$

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:

```
FINDLARGE(int[] A, int k)
   $\ell := 0$ 
  for  $i = 1$  to  $k$  do
    randomly choose  $r \in \{1, 2, \dots, n\}$ 
    if  $A[r] > \ell$  then
       $\ell := A[r]$ 
  return  $\ell$ 
```

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:

```

FINDLARGE(int[] A, int k)
  ℓ := 0
  for i = 1 to k do
    randomly choose  $r \in \{1, 2, \dots, n\}$ 
    if  $A[r] > \ell$  then
      ℓ :=  $A[r]$ 
  return ℓ

```

FINDLARGE has error probability \leq

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:

```

FINDLARGE(int[] A, int k)
  ℓ := 0
  for i = 1 to k do
    randomly choose  $r \in \{1, 2, \dots, n\}$ 
    if  $A[r] > \ell$  then
      ℓ :=  $A[r]$ 
  return ℓ

```

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

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:

```

FINDLARGE(int[] A, int k)
  ℓ := 0
  for i = 1 to k do
    randomly choose  $r \in \{1, 2, \dots, n\}$ 
    if  $A[r] > \ell$  then
      ℓ :=  $A[r]$ 
  return ℓ

```

FINDLARGE has error probability $\leq \frac{1}{2^k}$.
Set $k = c \log_2 n$ for some constant $c > 1$.

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:

```

FINDLARGE(int[] A, int k)
  ℓ := 0
  for i = 1 to k do
    randomly choose  $r \in \{1, 2, \dots, n\}$ 
    if  $A[r] > \ell$  then
      ℓ :=  $A[r]$ 
  return ℓ

```

FINDLARGE has error probability $\leq \frac{1}{2^k}$.
Set $k = c \log_2 n$ for some constant $c > 1$.
 \Rightarrow Error probability \leq

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:

```

FINDLARGE(int[] A, int k)
  ℓ := 0
  for i = 1 to k do
    randomly choose  $r \in \{1, 2, \dots, n\}$ 
    if  $A[r] > \ell$  then
      ℓ :=  $A[r]$ 
  return ℓ

```

FINDLARGE has error probability $\leq \frac{1}{2^k}$.
 Set $k = c \log_2 n$ for some constant $c > 1$.
 \Rightarrow Error probability $\leq \frac{1}{n^c}$

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:

```

FINDLARGE(int[] A, int k)
  ℓ := 0
  for i = 1 to k do
    randomly choose  $r \in \{1, 2, \dots, n\}$ 
    if  $A[r] > \ell$  then
      ℓ :=  $A[r]$ 
  return ℓ

```

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

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:

```

FINDLARGE(int[] A, int k)
  ℓ := 0
  for i = 1 to k do
    randomly choose  $r \in \{1, 2, \dots, n\}$ 
    if  $A[r] > \ell$  then
      ℓ :=  $A[r]$ 
  return ℓ

```

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 runtime.

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
---	---	---	---	---	---	---	---	---

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:

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

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

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

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

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:

Success probability in each step

```

FINDREPEATED(int[] A)
  while true do
    randomly choose  $i \in \{1, \dots, n\}$ 
    randomly choose  $j \in \{1, \dots, n\} \setminus \{i\}$ 
    if  $A[i] = A[j]$  then return  $A[i]$ 
  
```

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

Success probability in each step

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

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

Success probability in each step

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

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(int[] A)
  while true do
    randomly choose  $i \in \{1, \dots, n\}$ 
    randomly choose  $j \in \{1, \dots, n\} \setminus \{i\}$ 
    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

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(int[] A)
  while true do
    randomly choose  $i \in \{1, \dots, n\}$ 
    randomly choose  $j \in \{1, \dots, n\} \setminus \{i\}$ 
    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

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

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.

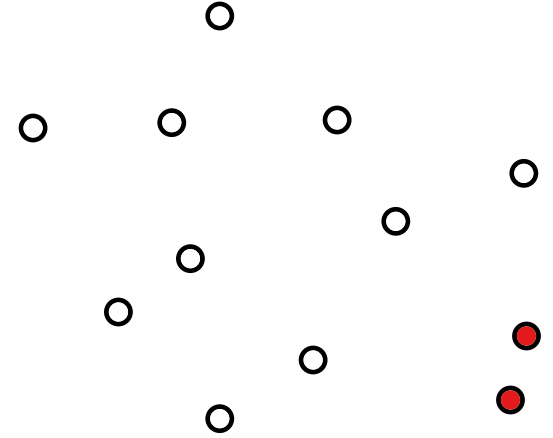
Examples. FINDLARGE, Karger's randomized MinCut algorithm

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

CLOSEST PAIR

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

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



CLOSEST PAIR

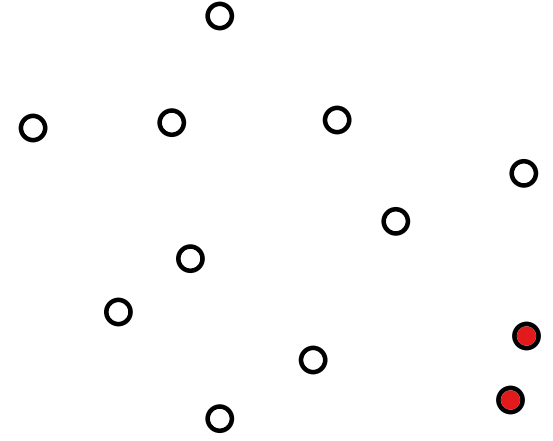
Given: (multi-)set of points $P = \{p_1, p_2, \dots, p_n\} \subseteq \mathbb{R}^2$.

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

Deterministic approaches:

Brute-force:

$\Theta(n^2)$ time



CLOSEST PAIR

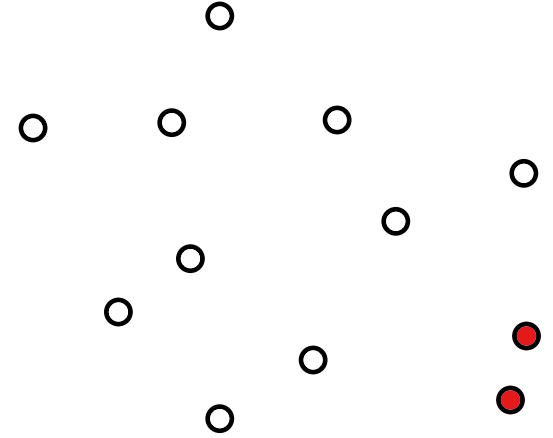
Given: (multi-)set of points $P = \{p_1, p_2, \dots, p_n\} \subseteq \mathbb{R}^2$.

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

Deterministic approaches:

Brute-force: $\Theta(n^2)$ time

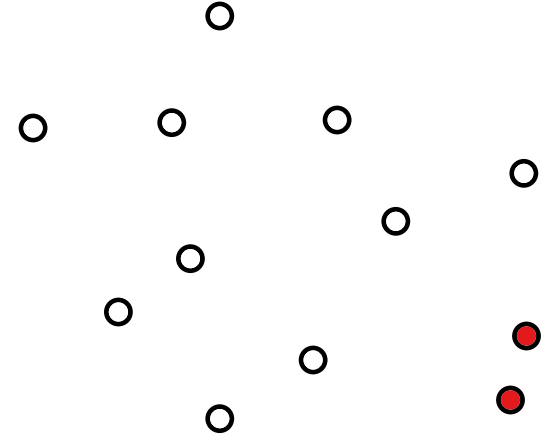
Divide and conquer (recall from ADS): $\Theta(n \log n)$ time



CLOSEST PAIR

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

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



Deterministic approaches:

Brute-force: $\Theta(n^2)$ time

Divide and conquer (recall from ADS): $\Theta(n \log n)$ time

Lower bound:

ELEMENT UNIQUENESS: Given numbers a_1, a_2, \dots, a_n . Are they pairwise distinct?

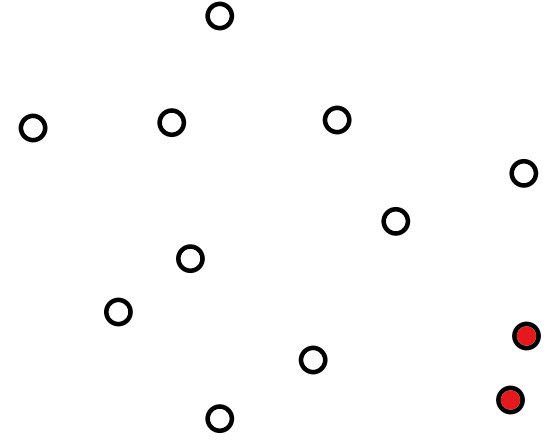
There is no $o(n \log n)$ -time algorithm for ELEMENT UNIQUENESS.

(under some assumption concerning the arithmetic model)

CLOSEST PAIR

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

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



Deterministic approaches:

Brute-force: $\Theta(n^2)$ time

Divide and conquer (recall from ADS): $\Theta(n \log n)$ time

Lower bound:

ELEMENT UNIQUENESS: Given numbers a_1, a_2, \dots, 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)

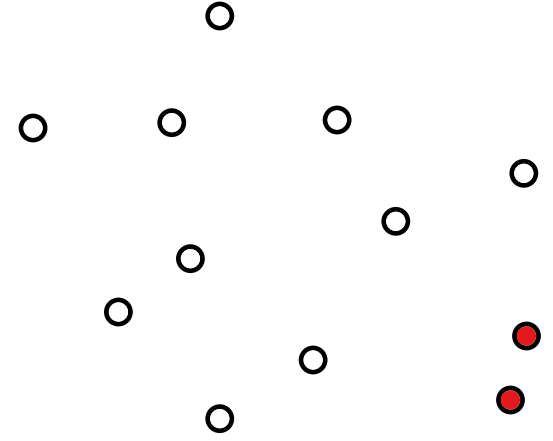
\Rightarrow There is no $o(n \log n)$ -time algorithm for CLOSEST PAIR.

(under the same assumption concerning the arithmetic model)

CLOSEST PAIR

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

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



Deterministic approaches:

Brute-force: $\Theta(n^2)$ time

Divide and conquer (recall from ADS): $\Theta(n \log n)$ time

Lower bound:

ELEMENT UNIQUENESS: Given numbers a_1, a_2, \dots, 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: For each i , map a_i to point $(a_i, 0)$ and test if the minimum distance is 0.

A Randomized Incremental Algorithm for CLOSEST PAIR

For $i \in \{1, \dots, n\}$, let $P_i = \{p_1, \dots, p_i\}$ and let $\delta_i =$ distance of a closest pair in P_i .

A Randomized Incremental Algorithm for CLOSEST PAIR

For $i \in \{1, \dots, n\}$, let $P_i = \{p_1, \dots, p_i\}$ and let $\delta_i =$ distance of a closest pair in P_i .

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

A Randomized Incremental Algorithm for CLOSEST PAIR

For $i \in \{1, \dots, n\}$, let $P_i = \{p_1, \dots, p_i\}$ and let $\delta_i =$ distance of a closest pair in P_i .

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

Assume that we have already determined δ_{i-1} .

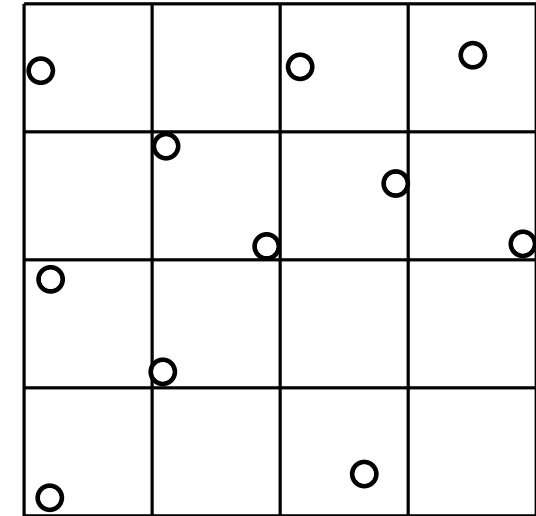
A Randomized Incremental Algorithm for CLOSEST PAIR

For $i \in \{1, \dots, n\}$, let $P_i = \{p_1, \dots, p_i\}$ and let $\delta_i =$ distance of a closest pair in P_i .

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

Assume that we have already determined δ_{i-1} .

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



A Randomized Incremental Algorithm for CLOSEST PAIR

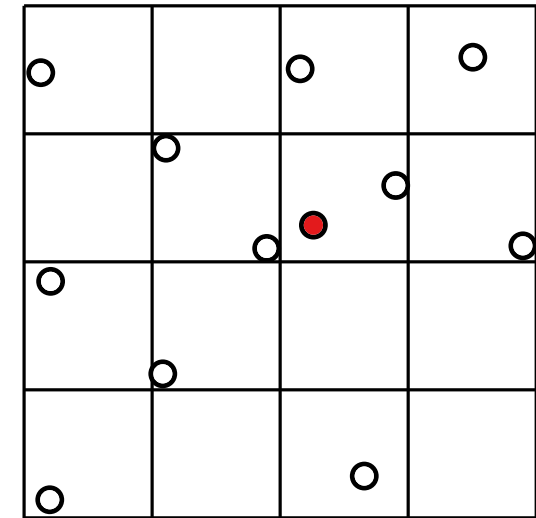
For $i \in \{1, \dots, n\}$, let $P_i = \{p_1, \dots, p_i\}$ and let $\delta_i =$ distance of a closest pair in P_i .

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

Assume that 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) .



A Randomized Incremental Algorithm for CLOSEST PAIR

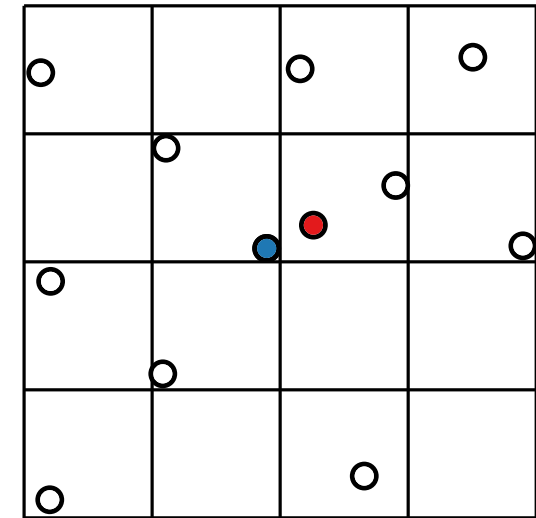
For $i \in \{1, \dots, n\}$, let $P_i = \{p_1, \dots, p_i\}$ and let $\delta_i =$ distance of a closest pair in P_i .

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

Assume that 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) .



A Randomized Incremental Algorithm for CLOSEST PAIR

For $i \in \{1, \dots, n\}$, let $P_i = \{p_1, \dots, p_i\}$ and let $\delta_i =$ distance of a closest pair in P_i .

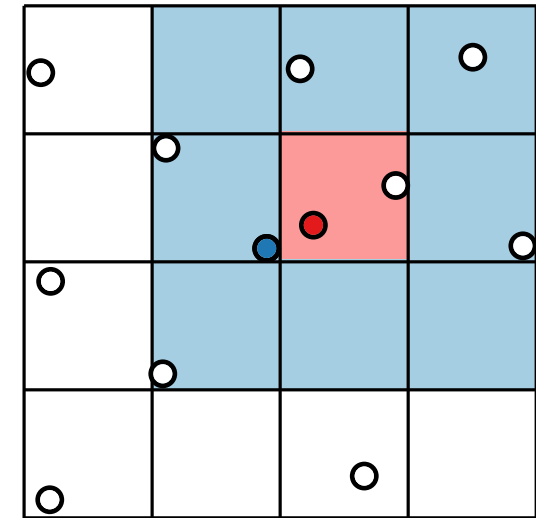
Idea: $\delta_2 = \|p_1 - p_2\|$. Compute $\delta_3, \delta_4, \dots, \delta_n$ by adding the points iteratively.

Assume that 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**.



A Randomized Incremental Algorithm for CLOSEST PAIR

For $i \in \{1, \dots, n\}$, let $P_i = \{p_1, \dots, p_i\}$ and let $\delta_i =$ distance of a closest pair in P_i .

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

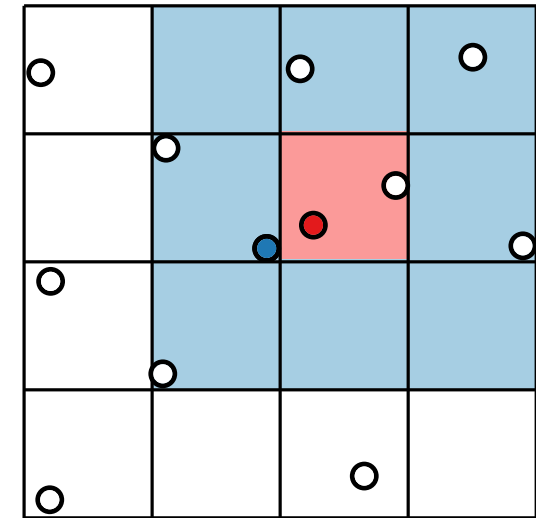
Assume that we have already determined δ_{i-1} .

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**.

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



(simple)
exercise
↙

A Randomized Incremental Algorithm for CLOSEST PAIR

For $i \in \{1, \dots, n\}$, let $P_i = \{p_1, \dots, p_i\}$ and let $\delta_i =$ distance of a closest pair in P_i .

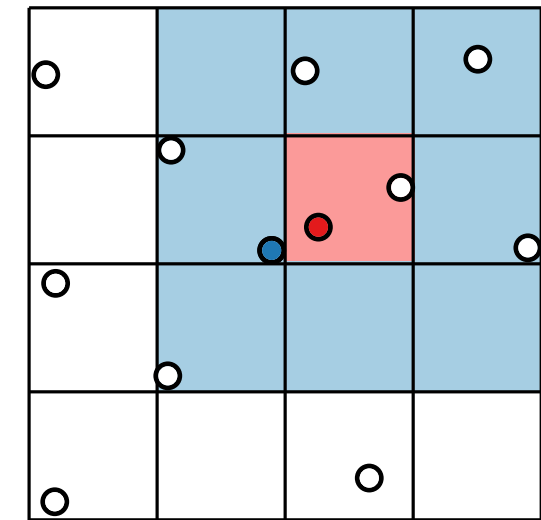
Idea: $\delta_2 = \|p_1 - p_2\|$. Compute $\delta_3, \delta_4, \dots, \delta_n$ by adding the points iteratively.

Assume that 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 that the floor function can be computed in $\mathcal{O}(1)$ time.

A Randomized Incremental Algorithm for CLOSEST PAIR

For $i \in \{1, \dots, n\}$, let $P_i = \{p_1, \dots, p_i\}$ and let $\delta_i =$ distance of a closest pair in P_i .

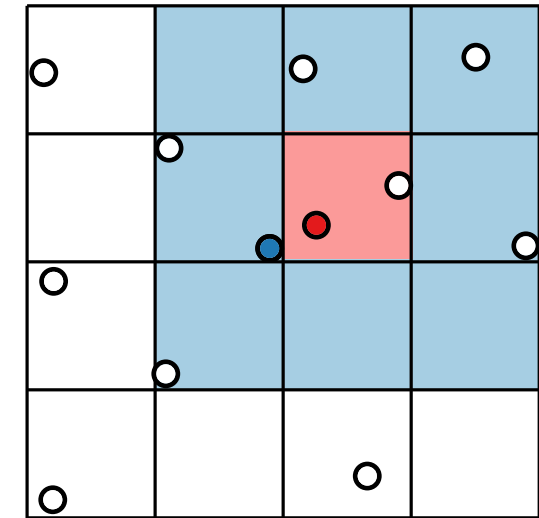
Idea: $\delta_2 = \|p_1 - p_2\|$. Compute $\delta_3, \delta_4, \dots, \delta_n$ by adding the points iteratively.

Assume that 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 that 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 that P_{i-1} is stored in a suitable dictionary for the nonempty cells (e.g., 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.

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 $\mathcal{O}(n^2)$.

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 .

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

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
= ???

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
= ???

Use backwards analysis!

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

Use backwards analysis!

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 ?

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 ?

■ If the closest distance in P_i is unique:

2 points

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 ?

- If the closest distance in P_i is unique: 2 points
- If one point has the same smallest distance to several points: 1 point

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 ?

- If the closest distance in P_i is unique: 2 points
- If one point has the same smallest distance to several points: 1 point
- If there are at least two disjoint closest pairs: 0 points

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

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 .

$\Rightarrow \mathbf{E}[X_i] \leq$

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 .

$\Rightarrow \mathbb{E}[X_i] \leq \frac{2}{i} \cdot \mathcal{O}(i) + \frac{i-2}{i} \cdot \mathcal{O}(1) =$

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 .

$\Rightarrow \mathbb{E}[X_i] \leq \frac{2}{i} \cdot \mathcal{O}(i) + \frac{i-2}{i} \cdot \mathcal{O}(1) = \mathcal{O}(1)$

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 .

$$\Rightarrow \mathbb{E}[X_i] \leq \frac{2}{i} \cdot \mathcal{O}(i) + \frac{i-2}{i} \cdot \mathcal{O}(1) = \mathcal{O}(1)$$

Let $X = X_1 + \dots + X_n$ be the total running time of the algorithm.

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

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 .

$$\Rightarrow \mathbf{E}[X_i] \leq \frac{2}{i} \cdot \mathcal{O}(i) + \frac{i-2}{i} \cdot \mathcal{O}(1) = \mathcal{O}(1)$$

Let $X = X_1 + \dots + X_n$ be the total running time of the algorithm.

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

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 .

$$\Rightarrow \mathbf{E}[X_i] \leq \frac{2}{i} \cdot \mathcal{O}(i) + \frac{i-2}{i} \cdot \mathcal{O}(1) = \mathcal{O}(1)$$

Let $X = X_1 + \dots + X_n$ be the total running time of the algorithm.

$$\Rightarrow \mathbf{E}[X] = \mathbf{E}[X_1] + \dots + \mathbf{E}[X_n] \in \mathcal{O}(n)$$

Discussion

Randomized algorithms (often)

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

Discussion

Randomized algorithms (often)

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

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,

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,

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.