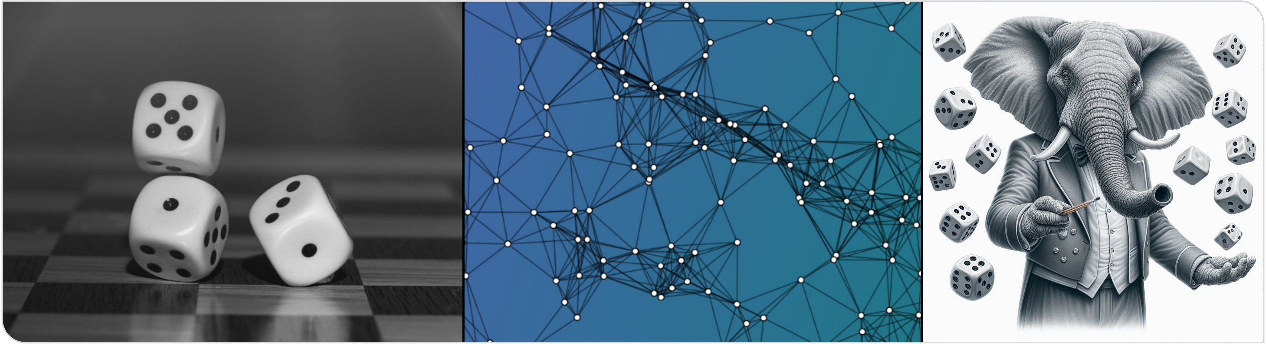
Karlsruhe Institute of Technology

Probability and Computing –
Important Random Variables and How to Sample Them

Stefan Walzer | WS 2024/2025



Content

1. What is Probability?
2. Bernoulli Distribution
3. Uniform Distribution
4. Rejection Sampling
5. Inverse Transform Sampling
6. Geometric Distribution
7. Sampling Without Replacement
8. Reservoir Sampling

What is a Probability?

Physical Accounts

Probabilities are persistent rates of outcomes when observing the same (random) process over and over again.

What is a Probability?

Physical Accounts

Probabilities are persistent rates of outcomes when observing the same (random) process over and over again.

It's about **objective stuff**:

"The probability that the coin comes up heads is 50%*."*

What is a Probability?

Physical Accounts

Probabilities are persistent rates of outcomes when observing the same (random) process over and over again.

Evidential / Bayesian Accounts

Probabilities reflect how much a rational agent believes in a proposition and about how much they are willing to bet on it.

It's about **objective stuff**:

"The probability that the coin comes up heads is 50%*."*

What is a Probability?

Physical Accounts

Probabilities are persistent rates of outcomes when observing the same (random) process over and over again.

It's about **objective stuff**:

"The probability that the coin comes up heads is 50%*."*

Evidential / Bayesian Accounts

Probabilities reflect how much a rational agent believes in a proposition and about how much they are willing to bet on it.

It's about **what I subjectively know**:

"The probability that it is going to rain tomorrow is 33%*."*

What is a Probability?

Physical Accounts

Probabilities are persistent rates of outcomes when observing the same (random) process over and over again.

It's about **objective stuff**:

"The probability that the coin comes up heads is 50%*."*

Evidential / Bayesian Accounts

Probabilities reflect how much a rational agent believes in a proposition and about how much they are willing to bet on it.

It's about **what I subjectively know**:

"The probability that it is going to rain tomorrow is 33%*."*

See `https://en.wikipedia.org/wiki/Probability_interpretations`.

In this lecture, we use a naive notion.

Bernoulli Distribution

Definition: $Ber(p)$ for $p \in [0, 1]$

$B \sim Ber(p)$ is a random variable with

$$\Pr[B = 1] = p \text{ and } \Pr[B = 0] = 1 - p.$$

Bernoulli Distribution

Definition: $Ber(p)$ for $p \in [0, 1]$

$B \sim Ber(p)$ is a random variable with

$$\Pr[B = 1] = p \text{ and } \Pr[B = 0] = 1 - p.$$

Standard Assumption: Access to Coin Flips

Algorithms have access to a sequence $B_1, B_2, \ldots \sim Ber(1/2)$ in independent uniformly random bits.

Bernoulli Distribution

Definition: $Ber(p)$ for $p \in [0, 1]$

$B \sim Ber(p)$ is a random variable with

$$\Pr[B = 1] = p \text{ and } \Pr[B = 0] = 1 - p.$$

Standard Assumption: Access to Coin Flips

Algorithms have access to a sequence $B_1, B_2, \ldots \sim Ber(1/2)$ in independent uniformly random bits.

Exercise: $Ber(1/3)$ from $Ber(1/2)$

Design an algorithm that outputs B such that $B \sim Ber(1/3)$.

Uniform Distribution

Definition: $\mathcal{U}(D)$ on finite D

If $|D| < \infty$, then $X \sim \mathcal{U}(D)$ is a random variable with

$$\Pr[X = x] = \frac{1}{|D|} \text{ for all } x \in D.$$

Uniform Distribution

Definition: $\mathcal{U}(D)$ on finite D

If $|D| < \infty$, then $X \sim \mathcal{U}(D)$ is a random variable with

$$\Pr[X = x] = \frac{1}{|D|} \text{ for all } x \in D.$$

Definition: $\mathcal{U}(D)$ on infinite D

If D is infinite but has finite measure[a] then $X \sim \mathcal{U}(D)$ is a random variable with uniform density function on D.
Important example:

$$X \sim \mathcal{U}([0,1]) \Leftrightarrow \forall x \in [0,1] : \Pr[X < x] = x.$$

[a] Formal details: Not in this lecture.

Uniform Distribution

Definition: $\mathcal{U}(D)$ on finite D

If $|D| < \infty$, then $X \sim \mathcal{U}(D)$ is a random variable with

$$\Pr[X = x] = \frac{1}{|D|} \text{ for all } x \in D.$$

Definition: $\mathcal{U}(D)$ on infinite D

If D is infinite but has finite measure[a] then $X \sim \mathcal{U}(D)$ is a random variable with uniform density function on D. Important example:

$$X \sim \mathcal{U}([0,1]) \Leftrightarrow \forall x \in [0,1] : \Pr[X < x] = x.$$

[a] Formal details: Not in this lecture.

Standard Assumption

Algorithms have access to $X_1, X_2, \ldots \sim \mathcal{U}([0,1])$. In practice: Initialise the significand[a] of floating point number with random bits.

[a] Deutsch: Mantisse.

Uniform Distribution

Karlsruhe Institute of Technology

Definition: $\mathcal{U}(D)$ on finite D

If $|D| < \infty$, then $X \sim \mathcal{U}(D)$ is a random variable with

$$\Pr[X = x] = \frac{1}{|D|} \text{ for all } x \in D.$$

Definition: $\mathcal{U}(D)$ on infinite D

If D is infinite but has finite measure[a] then $X \sim \mathcal{U}(D)$ is a random variable with uniform density function on D. Important example:

$$X \sim \mathcal{U}([0,1]) \Leftrightarrow \forall x \in [0,1] : \Pr[X < x] = x.$$

[a] Formal details: Not in this lecture.

Standard Assumption

Algorithms have access to $X_1, X_2, \ldots \sim \mathcal{U}([0,1])$. In practice: Initialise the significand[a] of floating point number with random bits.

[a] Deutsch: Mantisse.

Exercise: $\mathcal{U}(\{1,\ldots,n\})$ from $\mathcal{U}([0,1])$

Design an algorithm that outputs X such that $X \sim \mathcal{U}(\{1,\ldots,n\})$.

Uniform Distribution on a Disc

Task

Sample $P \sim \mathcal{U}(D)$ for $D = \{(x, y) \in \mathbb{R}^2 \mid x^2 + y^2 \leq 1\}$.

Uniform Distribution on a Disc

Task

Sample $P \sim \mathcal{U}(D)$ for $D = \{(x, y) \in \mathbb{R}^2 \mid x^2 + y^2 \leq 1\}$.

Flawed Attempt

```
sample $\Phi \sim \mathcal{U}([0, 2\pi])$
sample $R \sim \mathcal{U}([0, 1])$
**return** $(R \cdot \cos\Phi, R \cdot \sin\Phi)$
```

Uniform Distribution on a Disc

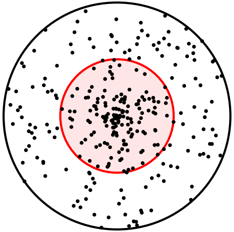
Karlsruhe Institute of Technology

Task

Sample $P \sim \mathcal{U}(D)$ for $D = \{(x, y) \in \mathbb{R}^2 \mid x^2 + y^2 \leq 1\}$.

Flawed Attempt

```
sample $\Phi \sim \mathcal{U}([0, 2\pi])$
sample $R \sim \mathcal{U}([0, 1])$
**return** $(R \cdot \cos\Phi, R \cdot \sin\Phi)$
```



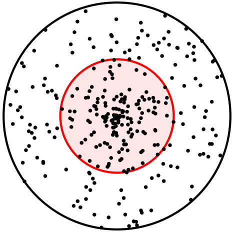
Uniform Distribution on a Disc

Task

Sample $P \sim \mathcal{U}(D)$ for $D = \{(x, y) \in \mathbb{R}^2 \mid x^2 + y^2 \leq 1\}$.

Flawed Attempt

```
sample $\Phi \sim \mathcal{U}([0, 2\pi])$
sample $R \sim \mathcal{U}([0, 1])$
**return** $(R \cdot \cos\Phi, R \cdot \sin\Phi)$
```



Issue

Disc of half the radius is hit 50% of the time but makes up only $1/4$ of the area!

Uniform Distribution on a Disc with Rejection Sampling

Task

Sample $P \sim \mathcal{U}(D)$ for $D = \{(x, y) \in \mathbb{R}^2 \mid x^2 + y^2 \leq 1\}$.

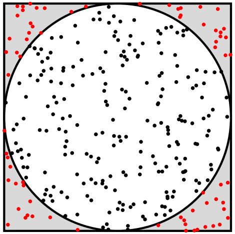
Uniform Distribution on a Disc with Rejection Sampling

Task

Sample $P \sim \mathcal{U}(D)$ for $D = \{(x, y) \in \mathbb{R}^2 \mid x^2 + y^2 \leq 1\}$.

Solution with Rejection Sampling

```
repeat
    sample $X \sim \mathcal{U}([-1, 1])$
    sample $Y \sim \mathcal{U}([-1, 1])$
until $X^2 + Y^2 \leq 1$
return $(X, Y)$
```



Uniform Distribution on a Disc with Rejection Sampling

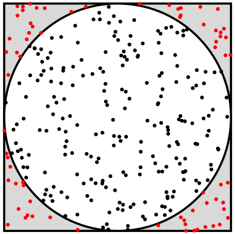
Karlsruhe Institute of Technology

Task

Sample $P \sim \mathcal{U}(D)$ for $D = \{(x, y) \in \mathbb{R}^2 \mid x^2 + y^2 \leq 1\}$.

Solution with Rejection Sampling

```
**repeat**
    sample $X \sim \mathcal{U}([-1, 1])$
    sample $Y \sim \mathcal{U}([-1, 1])$
**until** $X^2 + Y^2 \leq 1$
**return** $(X, Y)$
```



- Idea: $P \sim \mathcal{U}([-1, 1]^2)$ *conditioned* on $P \in D$ is uniform on D.

Uniform Distribution on a Disc with Rejection Sampling

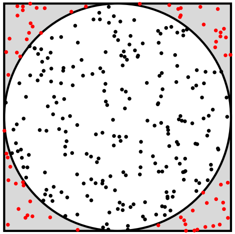
Karlsruhe Institute of Technology

Task

Sample $P \sim \mathcal{U}(D)$ for $D = \{(x, y) \in \mathbb{R}^2 \mid x^2 + y^2 \leq 1\}$.

Solution with Rejection Sampling

```
repeat
    sample $X \sim \mathcal{U}([-1, 1])$
    sample $Y \sim \mathcal{U}([-1, 1])$
until $X^2 + Y^2 \leq 1$
return $(X, Y)$
```



- Idea: $P \sim \mathcal{U}([-1, 1]^2)$ *conditioned* on $P \in D$ is uniform on D.
- Each sample is accepted with probability $\pi/4$.
- Expected number of rounds is $1/(\pi/4) = \mathcal{O}(1)$.

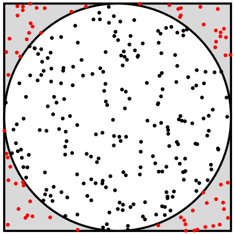
Uniform Distribution on a Disc with Rejection Sampling

Task

Sample $P \sim \mathcal{U}(D)$ for $D = \{(x, y) \in \mathbb{R}^2 \mid x^2 + y^2 \leq 1\}$.

Solution with Rejection Sampling

```
repeat
    sample $X \sim \mathcal{U}([-1, 1])$
    sample $Y \sim \mathcal{U}([-1, 1])$
until $X^2 + Y^2 \leq 1$
return $(X, Y)$
```



- Idea: $P \sim \mathcal{U}([-1, 1]^2)$ *conditioned* on $P \in D$ is uniform on D.
- Each sample is accepted with probability $\pi/4$.
- Expected number of rounds is $1/(\pi/4) = \mathcal{O}(1)$.

Spoiler alert: We'll get worst-case constant time with inverse transform sampling later.

Probability? ○ Bernoulli ○ Uniform ○○ Rejection ●○ Inverse Transform ○○ Geometric ○ No Replacement ○ Reservoir ○ Appendix ○○

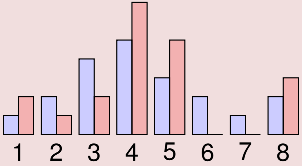
Rejection Sampling in General Discrete Distributions

Exercise

Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be distributions on a finite[a] set D. Assume

1. We can sample in constant time from $\mathcal{D}_1$.
2. There exists $C > 0$ such that for any $x \in D$ we have

$$\Pr_{X \sim \mathcal{D}_2}[X = x] \leq C \cdot \Pr_{X \sim \mathcal{D}_1}[X = x].$$



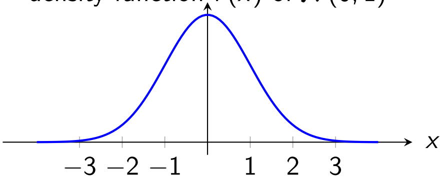
Design an algorithm that generates a sample from $\mathcal{D}_2$ in expected time $\mathcal{O}(C)$.

[a] This can be generalised.

Inverse Transform Sampling

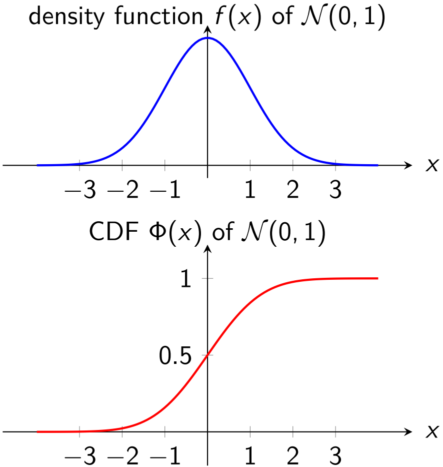
- Let $\mathcal{D}$ be a distribution on $\mathbb{R}$.
  $\hookrightarrow$ e.g. $\mathcal{D} = \mathcal{N}(0,1)$

density function $f(x)$ of $\mathcal{N}(0,1)$



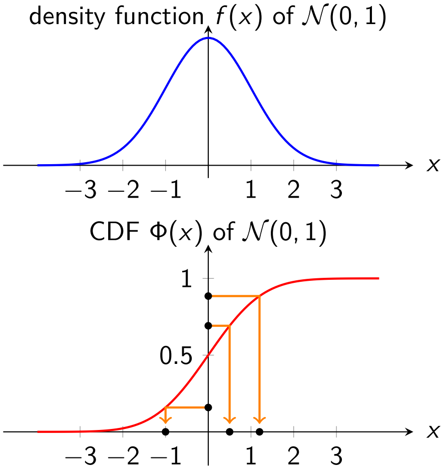
Inverse Transform Sampling

- Let $\mathcal{D}$ be a distribution on $\mathbb{R}$.
  $\hookrightarrow$ e.g. $\mathcal{D} = \mathcal{N}(0,1)$
- Let $X \sim \mathcal{D}$ and $F_X(x) = \Pr[X \leq x]$.
  $\hookrightarrow$ F_X is the *cumulative distribution function* of X
  $\hookrightarrow$ the CDF of the normal distribution is called Φ



Inverse Transform Sampling

- Let $\mathcal{D}$ be a distribution on $\mathbb{R}$.
  $\hookrightarrow$ e.g. $\mathcal{D} = \mathcal{N}(0,1)$
- Let $X \sim \mathcal{D}$ and $F_X(x) = \Pr[X \leq x]$.
  $\hookrightarrow$ F_X is the *cumulative distribution function* of X
  $\hookrightarrow$ the CDF of the normal distribution is called Φ
- Let $F_X^{-1}(u) := \inf\{x \in \mathbb{R} \mid F_X(x) \geq u\}$.
  $\hookrightarrow$ ordinary inverse for strictly monotone F_X



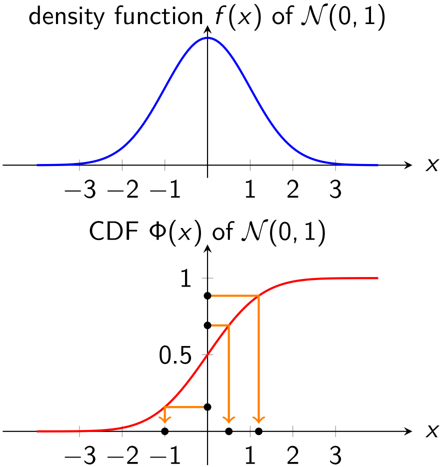
Inverse Transform Sampling

- Let $\mathcal{D}$ be a distribution on $\mathbb{R}$.
  $\hookrightarrow$ e.g. $\mathcal{D} = \mathcal{N}(0,1)$
- Let $X \sim \mathcal{D}$ and $F_X(x) = \Pr[X \le x]$.
  $\hookrightarrow$ F_X is the *cumulative distribution function* of X
  $\hookrightarrow$ the CDF of the normal distribution is called Φ
- Let $F_X^{-1}(u) := \inf\{x \in \mathbb{R} \mid F_X(x) \ge u\}$.
  $\hookrightarrow$ ordinary inverse for strictly monotone F_X

Theorem (Inverse Transform Sampling)

If $U \sim \mathcal{U}([0,1])$ then $F_X^{-1}(U) \overset{\text{d}}{=} X$, i.e. $F_X^{-1}(U) \sim \mathcal{D}$.

("$\overset{\text{d}}{=}$" means: "has the same distribution as")



Inverse Transform Sampling

- Let $\mathcal{D}$ be a distribution on $\mathbb{R}$.
  $\hookrightarrow$ e.g. $\mathcal{D} = \mathcal{N}(0, 1)$
- Let $X \sim \mathcal{D}$ and $F_X(x) = \Pr[X \leq x]$.
  $\hookrightarrow$ F_X is the *cumulative distribution function* of X
  $\hookrightarrow$ the CDF of the normal distribution is called Φ
- Let $F_X^{-1}(u) := \inf\{x \in \mathbb{R} \mid F_X(x) \geq u\}$.
  $\hookrightarrow$ ordinary inverse for strictly monotone F_X

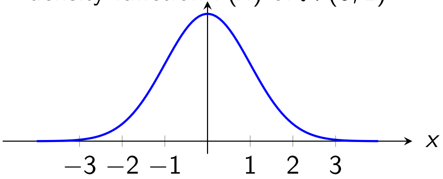
Theorem (Inverse Transform Sampling)

If $U \sim \mathcal{U}([0, 1])$ then $F_X^{-1}(U) \overset{\mathrm{d}}{=} X$, i.e. $F_X^{-1}(U) \sim \mathcal{D}$.

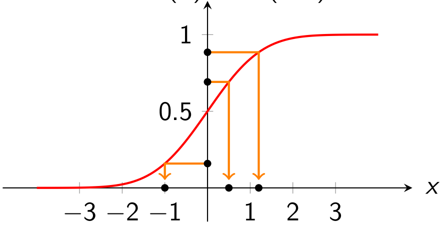
("$\overset{\mathrm{d}}{=}$" means: "has the same distribution as")

Reason: $\Pr[F_X^{-1}(U) \leq x] = \Pr[U \leq F_X(x)] = F_X(x)$.

density function $f(x)$ of $\mathcal{N}(0, 1)$



CDF $\Phi(x)$ of $\mathcal{N}(0, 1)$



Uniform Distribution on a Disc with Inverse Transform Sampling

Task

Sample $P \sim \mathcal{U}(D)$ for $D = \{(x, y) \in \mathbb{R}^2 \mid x^2 + y^2 \leq 1\}$.

Uniform Distribution on a Disc with Inverse Transform Sampling

Task

Sample $P \sim \mathcal{U}(D)$ for $D = \{(x, y) \in \mathbb{R}^2 \mid x^2 + y^2 \leq 1\}$.

Preparation

If $(x, y) \sim \mathcal{U}(D)$ then $R = \sqrt{x^2 + y^2}$ satisfies

$$F_R(r) = \Pr[R \leq r] = r^2\pi/\pi = r^2 \text{ hence } F_R^{-1}(u) = \sqrt{u}.$$

Uniform Distribution on a Disc with Inverse Transform Sampling

Task

Sample $P \sim \mathcal{U}(D)$ for $D = \{(x, y) \in \mathbb{R}^2 \mid x^2 + y^2 \leq 1\}$.

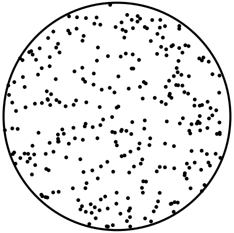
Preparation

If $(x, y) \sim \mathcal{U}(D)$ then $R = \sqrt{x^2 + y^2}$ satisfies

$$F_R(r) = \Pr[R \leq r] = r^2\pi/\pi = r^2 \text{ hence } F_R^{-1}(u) = \sqrt{u}.$$

Solution with Inverse Transform Sampling

```
sample $\Phi \sim \mathcal{U}([0, 2\pi])$
sample $U \sim \mathcal{U}([0, 1])$
$R \leftarrow \sqrt{U}$
**return** $(R \cdot \cos\Phi, R \cdot \sin\Phi)$
```



Geometric Distribution

Definition: $G \sim \mathit{Geom}_1(p)$ and $G' \sim \mathit{Geom}_0(p)$

Let $p \in (0,1]$ and $B_1, B_2, \ldots \sim \mathit{Ber}(p)$.
Then we define the geometric random variables

$$G := \min\{i \in \mathbb{N} \mid B_i = 1\}$$

↪ number of $\mathit{Ber}(p)$ trials until (and including) the first success

$$G' := G - 1$$

↪ number of $\mathit{Ber}(p)$ failures before the first success

We write $G \sim \mathit{Geom}_1(p)$ and $G' \sim \mathit{Geom}_0(p)$.[a]

[a] In the literature *Geom* is used inconsistently.

Geometric Distribution

Definition: $G \sim Geom_1(p)$ and $G' \sim Geom_0(p)$

Let $p \in (0, 1]$ and $B_1, B_2, \ldots \sim Ber(p)$.
Then we define the geometric random variables

$$G := \min\{i \in \mathbb{N} \mid B_i = 1\}$$

↪ number of $Ber(p)$ trials until (and including) the first success

$$G' := G - 1$$

↪ number of $Ber(p)$ failures before the first success

We write $G \sim Geom_1(p)$ and $G' \sim Geom_0(p)$.[a]

[a] In the literature *Geom* is used inconsistently.

Sampling $G \sim Geom_1(p)$ in time $\mathcal{O}(G)$

```
$i \leftarrow 0$
repeat
    $i \leftarrow i + 1$
    sample $X \sim Ber(p)$
until $X = 1$
return $i$
```

Quite bad: $\mathbb{E}[G] = 1/p$ might be large.

Geometric Distribution

Definition: $G \sim Geom_1(p)$ and $G' \sim Geom_0(p)$

Let $p \in (0,1]$ and $B_1, B_2, \ldots \sim Ber(p)$.
Then we define the geometric random variables

$$G := \min\{i \in \mathbb{N} \mid B_i = 1\}$$

↪ number of $Ber(p)$ trials until (and including) the first success

$$G' := G - 1$$

↪ number of $Ber(p)$ failures before the first success

We write $G \sim Geom_1(p)$ and $G' \sim Geom_0(p)$.[a]

[a] In the literature *Geom* is used inconsistently.

Sampling $G \sim Geom_1(p)$ in time $\mathcal{O}(G)$

```
$i \leftarrow 0$
repeat
    $i \leftarrow i + 1$
    sample $X \sim Ber(p)$
until $X = 1$
return $i$
```

Quite bad: $\mathbb{E}[G] = 1/p$ might be large.

Exercise

Use inverse transform sampling to sample $G \sim Geom_1(p)$ in time $\mathcal{O}(1)$.

Sampling Without Replacement

Exercise

Design an algorithm that, given $k, n \in \mathbb{N}$ with $0 \leq k \leq n$ outputs a set $S \subseteq [n]$ of size $|S| = k$ uniformly at random.

Reservoir Sampling

Task: Maintain a fair sample of k items while reading a (possibly infinite) stream.

Reservoir Sampling

Task: Maintain a fair sample of k items while reading a (possibly infinite) stream.

```
Algorithm init($k$):
    allocate reservoir$[1..k]$
    $n \leftarrow 0$
Algorithm observeItem($x$):
    $n \leftarrow n + 1$
    if $n \leq k$ then
        reservoir$[n] \leftarrow x$
    else
        sample $I \sim \mathcal{U}(\{1, \ldots, n\})$
        if $I \leq k$ then
            reservoir$[I] \leftarrow x$
```

Theorem

Assume we call init(k) and then observeItem(x) for $x \in \{x_1, \ldots, x_n\}$ with $n \geq k$. Afterwards `reservoir` contains every subset of $\{x_1, \ldots, x_n\}$ of size k with equal probability.

Proof by induction (not here).

Reservoir Sampling

Task: Maintain a fair sample of k items while reading a (possibly infinite) stream.

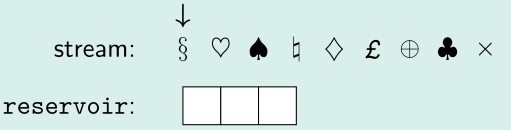
```
Algorithm init(k):
    allocate reservoir[1..k]
    $n \leftarrow 0$

Algorithm observeItem(x):
    $n \leftarrow n + 1$
    if $n \leq k$ then
        reservoir[n] $\leftarrow x$
    else
        sample $I \sim \mathcal{U}(\{1, \dots, n\})$
        if $I \leq k$ then
            reservoir[I] $\leftarrow x$
```

Theorem

Assume we call init(k) and then observeItem(x) for $x \in \{x_1, \dots, x_n\}$ with $n \geq k$. Afterwards `reservoir` contains every subset of $\{x_1, \dots, x_n\}$ of size k with equal probability.

Proof by induction (not here).

Example ($k = 3$)



Reservoir Sampling

Task: Maintain a fair sample of k items while reading a (possibly infinite) stream.

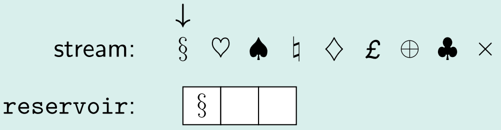
```
Algorithm init($k$):
    allocate reservoir[1..$k$]
    $n \leftarrow 0$
Algorithm observeItem($x$):
    $n \leftarrow n + 1$
    if $n \leq k$ then
        reservoir[$n$] $\leftarrow x$
    else
        sample $I \sim \mathcal{U}(\{1, \ldots, n\})$
        if $I \leq k$ then
            reservoir[$I$] $\leftarrow x$
```

Theorem

Assume we call init(k) and then observeItem(x) for $x \in \{x_1, \ldots, x_n\}$ with $n \geq k$. Afterwards `reservoir` contains every subset of $\{x_1, \ldots, x_n\}$ of size k with equal probability.

Proof by induction (not here).

Example ($k = 3$)



Reservoir Sampling

Task: Maintain a fair sample of k items while reading a (possibly infinite) stream.

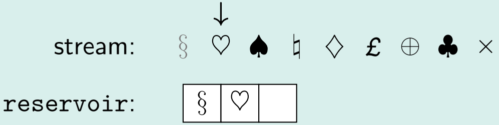
```
**Algorithm** init($k$):
    allocate reservoir[1..$k$]
    $n \leftarrow 0$

**Algorithm** observeItem($x$):
    $n \leftarrow n + 1$
    **if** $n \leq k$ **then**
        reservoir[$n$] $\leftarrow x$
    **else**
        sample $I \sim \mathcal{U}(\{1, \dots, n\})$
        **if** $I \leq k$ **then**
            reservoir[$I$] $\leftarrow x$
```

Theorem

Assume we call init(k) and then observeItem(x) for $x \in \{x_1, \dots, x_n\}$ with $n \geq k$. Afterwards `reservoir` contains every subset of $\{x_1, \dots, x_n\}$ of size k with equal probability.

Proof by induction (not here).

Example ($k = 3$)



stream: § ♡ ♠ ♮ ♢ £ ⊕ ♣ ×

reservoir: | § | ♡ | |

Reservoir Sampling

Task: Maintain a fair sample of k items while reading a (possibly infinite) stream.

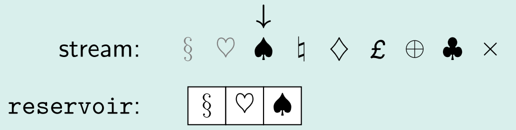
```
Algorithm init($k$):
    allocate reservoir[1..$k$]
    $n \leftarrow 0$
Algorithm observeItem($x$):
    $n \leftarrow n + 1$
    if $n \leq k$ then
        reservoir[$n$] $\leftarrow x$
    else
        sample $I \sim \mathcal{U}(\{1, \ldots, n\})$
        if $I \leq k$ then
            reservoir[$I$] $\leftarrow x$
```

Theorem

Assume we call init(k) and then observeItem(x) for $x \in \{x_1, \ldots, x_n\}$ with $n \geq k$. Afterwards `reservoir` contains every subset of $\{x_1, \ldots, x_n\}$ of size k with equal probability.

Proof by induction (not here).

Example ($k = 3$)



Reservoir Sampling

Task: Maintain a fair sample of k items while reading a (possibly infinite) stream.

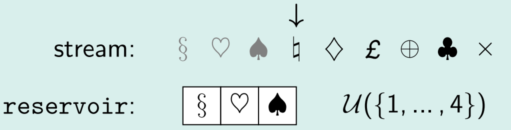
```
**Algorithm** init($k$):
    allocate reservoir[1..$k$]
    $n \leftarrow 0$

**Algorithm** observeItem($x$):
    $n \leftarrow n + 1$
    **if** $n \leq k$ **then**
        reservoir[$n$] $\leftarrow x$
    **else**
        sample $I \sim \mathcal{U}(\{1, \dots, n\})$
        **if** $I \leq k$ **then**
            reservoir[$I$] $\leftarrow x$
```

Theorem

Assume we call init(k) and then observeItem(x) for $x \in \{x_1, \dots, x_n\}$ with $n \geq k$. Afterwards `reservoir` contains every subset of $\{x_1, \dots, x_n\}$ of size k with equal probability.

Proof by induction (not here).

Example ($k = 3$)



Reservoir Sampling

Task: Maintain a fair sample of k items while reading a (possibly infinite) stream.

```
**Algorithm** init($k$):
    allocate reservoir[$1..k$]
    $n \leftarrow 0$
**Algorithm** observeItem($x$):
    $n \leftarrow n + 1$
    **if** $n \leq k$ **then**
        reservoir[$n$] $\leftarrow x$
    **else**
        sample $I \sim \mathcal{U}(\{1,...,n\})$
        **if** $I \leq k$ **then**
            reservoir[$I$] $\leftarrow x$
```

Theorem

Assume we call init(k) and then observeItem(x) for $x \in \{x_1, \dots, x_n\}$ with $n \geq k$. Afterwards `reservoir` contains every subset of $\{x_1, \dots, x_n\}$ of size k with equal probability.

Proof by induction (not here).

Example ($k = 3$)

↓

stream: § ♡ ♠ ♮ ♢ £ ⊕ ♣ ×

`reservoir`: | § | ♡ | ♠ | $\mathcal{U}(\{1, \dots, 4\}) \leadsto I = 2$

Reservoir Sampling

Task: Maintain a fair sample of k items while reading a (possibly infinite) stream.

```
Algorithm init(k):
    allocate reservoir[1..k]
    $n \leftarrow 0$
Algorithm observeItem(x):
    $n \leftarrow n + 1$
    if $n \leq k$ then
        reservoir[n] $\leftarrow x$
    else
        sample $I \sim \mathcal{U}(\{1, \dots, n\})$
        if $I \leq k$ then
            reservoir[I] $\leftarrow x$
```

Theorem

Assume we call init(k) and then observeItem(x) for $x \in \{x_1, \dots, x_n\}$ with $n \geq k$. Afterwards `reservoir` contains every subset of $\{x_1, \dots, x_n\}$ of size k with equal probability.

Proof by induction (not here).

Example ($k = 3$)

stream: § ♡ ♠ $\overset{\downarrow}{\natural}$ ◇ £ ⊕ ♣ ×

`reservoir`: | § | ♮ | ♠ | $\mathcal{U}(\{1, \dots, 4\}) \rightsquigarrow I = 2$ ✓

Reservoir Sampling

Task: Maintain a fair sample of k items while reading a (possibly infinite) stream.

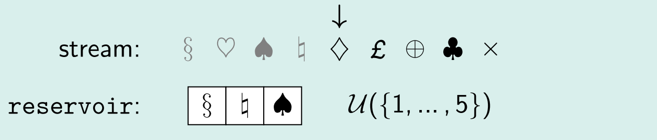
```
**Algorithm** init($k$):
    allocate reservoir[1..$k$]
    $n \leftarrow 0$
**Algorithm** observeItem($x$):
    $n \leftarrow n + 1$
    if $n \leq k$ then
        reservoir[$n$] $\leftarrow x$
    else
        sample $l \sim \mathcal{U}(\{1, \dots, n\})$
        if $l \leq k$ then
            reservoir[$l$] $\leftarrow x$
```

Theorem

Assume we call init(k) and then observeItem(x) for $x \in \{x_1, \dots, x_n\}$ with $n \geq k$. Afterwards `reservoir` contains every subset of $\{x_1, \dots, x_n\}$ of size k with equal probability.

Proof by induction (not here).

Example ($k = 3$)



Reservoir Sampling

Task: Maintain a fair sample of k items while reading a (possibly infinite) stream.

```
Algorithm init(k):
    allocate reservoir[1..k]
    $n \leftarrow 0$

Algorithm observeItem(x):
    $n \leftarrow n + 1$
    if $n \leq k$ then
        reservoir[n] $\leftarrow x$
    else
        sample $I \sim \mathcal{U}(\{1, \dots, n\})$
        if $I \leq k$ then
            reservoir[I] $\leftarrow x$
```

Theorem

Assume we call init(k) and then observeItem(x) for $x \in \{x_1, \dots, x_n\}$ with $n \geq k$. Afterwards `reservoir` contains every subset of $\{x_1, \dots, x_n\}$ of size k with equal probability.

Proof by induction (not here).

Example ($k = 3$)

stream: § ♡ ♠ ♮ $\overset{\downarrow}{\diamondsuit}$ £ ⊕ ♣ ×

`reservoir`: | § | ♮ | ♠ | $\quad \mathcal{U}(\{1, \dots, 5\}) \rightsquigarrow I = 5$

Reservoir Sampling

Task: Maintain a fair sample of k items while reading a (possibly infinite) stream.

```
Algorithm init($k$):
    allocate reservoir[1..$k$]
    $n \leftarrow 0$
Algorithm observeItem($x$):
    $n \leftarrow n + 1$
    if $n \leq k$ then
        reservoir[$n$] $\leftarrow x$
    else
        sample $I \sim \mathcal{U}(\{1, \dots, n\})$
        if $I \leq k$ then
            reservoir[$I$] $\leftarrow x$
```

Theorem

Assume we call init(k) and then observeItem(x) for $x \in \{x_1, \dots, x_n\}$ with $n \geq k$. Afterwards `reservoir` contains every subset of $\{x_1, \dots, x_n\}$ of size k with equal probability.

Proof by induction (not here).

Example ($k = 3$)

stream: $\S \quad \heartsuit \quad \spadesuit \quad \natural \quad \overset{\downarrow}{\diamondsuit} \quad \pounds \quad \oplus \quad \clubsuit \quad \times$

`reservoir`: $\boxed{\S}\boxed{\natural}\boxed{\spadesuit}$ $\quad \mathcal{U}(\{1, \dots, 5\}) \rightsquigarrow I = 5$ ✗

Reservoir Sampling

Task: Maintain a fair sample of k items while reading a (possibly infinite) stream.

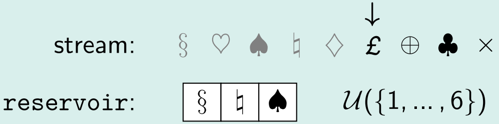
```
Algorithm init($k$):
    allocate reservoir[$1..k$]
    $n \leftarrow 0$

Algorithm observeItem($x$):
    $n \leftarrow n + 1$
    if $n \leq k$ then
        reservoir[$n$] $\leftarrow x$
    else
        sample $I \sim \mathcal{U}(\{1, \dots, n\})$
        if $I \leq k$ then
            reservoir[$I$] $\leftarrow x$
```

Theorem

Assume we call init(k) and then observeItem(x) for $x \in \{x_1, \dots, x_n\}$ with $n \geq k$. Afterwards `reservoir` contains every subset of $\{x_1, \dots, x_n\}$ of size k with equal probability.

Proof by induction (not here).

Example ($k = 3$)



stream: § ♡ ♠ ♮ ♢ £ ⊕ ♣ ×

reservoir: | § | ♮ | ♠ | $\mathcal{U}(\{1, \dots, 6\})$

Reservoir Sampling

Task: Maintain a fair sample of k items while reading a (possibly infinite) stream.

```
**Algorithm** init($k$):
    allocate reservoir[1..$k$]
    $n \leftarrow 0$
**Algorithm** observeItem($x$):
    $n \leftarrow n + 1$
    **if** $n \leq k$ **then**
        reservoir[$n$] $\leftarrow x$
    **else**
        sample $I \sim \mathcal{U}(\{1, \dots, n\})$
        **if** $I \leq k$ **then**
            reservoir[$I$] $\leftarrow x$
```

Theorem

Assume we call init(k) and then observeItem(x) for $x \in \{x_1, \dots, x_n\}$ with $n \geq k$. Afterwards `reservoir` contains every subset of $\{x_1, \dots, x_n\}$ of size k with equal probability.

Proof by induction (not here).

Example ($k = 3$)

↓

stream: § ♡ ♠ ♮ ♢ £ ⊕ ♣ ×

reservoir: | § | ♮ | ♠ | $\mathcal{U}(\{1, \dots, 6\}) \rightsquigarrow I = 1$

Reservoir Sampling

Task: Maintain a fair sample of k items while reading a (possibly infinite) stream.

```
Algorithm init($k$):
    allocate reservoir[1..$k$]
    $n \leftarrow 0$

Algorithm observeItem($x$):
    $n \leftarrow n + 1$
    if $n \leq k$ then
        reservoir[$n$] $\leftarrow x$
    else
        sample $I \sim \mathcal{U}(\{1, \dots, n\})$
        if $I \leq k$ then
            reservoir[$I$] $\leftarrow x$
```

Theorem

Assume we call init(k) and then observeItem(x) for $x \in \{x_1, \dots, x_n\}$ with $n \geq k$. Afterwards `reservoir` contains every subset of $\{x_1, \dots, x_n\}$ of size k with equal probability.

Proof by induction (not here).

Example ($k = 3$)

↓ (over £)

stream: § ♡ ♠ ♮ ♢ £ ⊕ ♣ ×

reservoir: | £ | ♮ | ♠ | $\mathcal{U}(\{1, \dots, 6\}) \rightsquigarrow I = 1$ ✓

Reservoir Sampling

Task: Maintain a fair sample of k items while reading a (possibly infinite) stream.

```
Algorithm init($k$):
    allocate reservoir[1..$k$]
    $n \leftarrow 0$

Algorithm observeItem($x$):
    $n \leftarrow n + 1$
    if $n \leq k$ then
        reservoir[$n$] $\leftarrow x$
    else
        sample $I \sim \mathcal{U}(\{1, \dots, n\})$
        if $I \leq k$ then
            reservoir[$I$] $\leftarrow x$
```

Theorem

Assume we call init(k) and then observeItem(x) for $x \in \{x_1, \dots, x_n\}$ with $n \geq k$. Afterwards `reservoir` contains every subset of $\{x_1, \dots, x_n\}$ of size k with equal probability.

Proof by induction (not here).

Example ($k = 3$)

stream: $\S \quad \heartsuit \quad \spadesuit \quad \natural \quad \diamondsuit \quad \pounds \quad \overset{\downarrow}{\oplus} \quad \clubsuit \quad \times$

`reservoir`: $\boxed{\pounds}\boxed{\natural}\boxed{\spadesuit}$ $\quad \mathcal{U}(\{1, \dots, 7\})$

Reservoir Sampling

Task: Maintain a fair sample of k items while reading a (possibly infinite) stream.

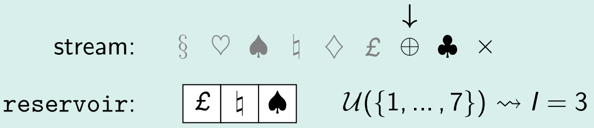
```
Algorithm init($k$):
    allocate reservoir[1..$k$]
    $n \leftarrow 0$
Algorithm observeItem($x$):
    $n \leftarrow n + 1$
    if $n \leq k$ then
        reservoir[$n$] $\leftarrow x$
    else
        sample $I \sim \mathcal{U}(\{1, \dots, n\})$
        if $I \leq k$ then
            reservoir[$I$] $\leftarrow x$
```

Theorem

Assume we call init(k) and then observeItem(x) for $x \in \{x_1, \dots, x_n\}$ with $n \geq k$. Afterwards `reservoir` contains every subset of $\{x_1, \dots, x_n\}$ of size k with equal probability.

Proof by induction (not here).

Example ($k = 3$)



stream: § ♡ ♠ ♮ ♢ £ ⊕ ♣ ×

`reservoir:` | £ | ♮ | ♠ |

$\mathcal{U}(\{1, \dots, 7\}) \rightsquigarrow I = 3$

Reservoir Sampling

Task: Maintain a fair sample of k items while reading a (possibly infinite) stream.

```
Algorithm init(k):
    allocate reservoir[1..k]
    $n \leftarrow 0$
Algorithm observeItem(x):
    $n \leftarrow n + 1$
    if $n \leq k$ then
        reservoir[$n$] $\leftarrow x$
    else
        sample $I \sim \mathcal{U}(\{1, \dots, n\})$
        if $I \leq k$ then
            reservoir[$I$] $\leftarrow x$
```

Theorem

Assume we call init(k) and then observeItem(x) for $x \in \{x_1, \dots, x_n\}$ with $n \geq k$. Afterwards `reservoir` contains every subset of $\{x_1, \dots, x_n\}$ of size k with equal probability.

Proof by induction (not here).

Example ($k = 3$)

stream: $\S \quad \heartsuit \quad \spadesuit \quad \natural \quad \diamondsuit \quad \pounds \quad \overset{\downarrow}{\oplus} \quad \clubsuit \quad \times$

`reservoir:` $\boxed{\pounds}\boxed{\natural}\boxed{\oplus}$ $\quad \mathcal{U}(\{1, \dots, 7\}) \rightsquigarrow I = 3$ ✓

Reservoir Sampling

Task: Maintain a fair sample of k items while reading a (possibly infinite) stream.

```
Algorithm init(k):
    allocate reservoir[1..k]
    $n \leftarrow 0$

Algorithm observeItem(x):
    $n \leftarrow n + 1$
    if $n \leq k$ then
        reservoir[n] ← x
    else
        sample $I \sim \mathcal{U}(\{1, \dots, n\})$
        if $I \leq k$ then
            reservoir[I] ← x
```

Theorem

Assume we call init(k) and then observeItem(x) for $x \in \{x_1, \dots, x_n\}$ with $n \geq k$. Afterwards `reservoir` contains every subset of $\{x_1, \dots, x_n\}$ of size k with equal probability.

Proof by induction (not here).

Example ($k = 3$)

↓

stream: § ♡ ♠ ♮ ♢ £ ⊕ ♣ ×

`reservoir`: | £ | ♮ | ⊕ | $\mathcal{U}(\{1, \dots, 8\})$

Reservoir Sampling

Task: Maintain a fair sample of k items while reading a (possibly infinite) stream.

```
Algorithm init(k):
    allocate reservoir[1..k]
    $n \leftarrow 0$
Algorithm observeItem(x):
    $n \leftarrow n + 1$
    if $n \leq k$ then
        reservoir[n] ← x
    else
        sample $I \sim \mathcal{U}(\{1, \dots, n\})$
        if $I \leq k$ then
            reservoir[I] ← x
```

Theorem

Assume we call init(k) and then observeItem(x) for $x \in \{x_1, \dots, x_n\}$ with $n \geq k$. Afterwards `reservoir` contains every subset of $\{x_1, \dots, x_n\}$ of size k with equal probability.

Proof by induction (not here).

Example ($k = 3$)

stream: $\S \quad \heartsuit \quad \spadesuit \quad \natural \quad \diamondsuit \quad \pounds \quad \oplus \quad \overset{\downarrow}{\clubsuit} \quad \times$

`reservoir:` $\boxed{\pounds}\boxed{\natural}\boxed{\oplus}$ $\quad \mathcal{U}(\{1, \dots, 8\}) \rightsquigarrow I = 3$

Reservoir Sampling

Task: Maintain a fair sample of k items while reading a (possibly infinite) stream.

```
**Algorithm** init($k$):
    allocate reservoir[1..$k$]
    $n \leftarrow 0$

**Algorithm** observeItem($x$):
    $n \leftarrow n + 1$
    **if** $n \leq k$ **then**
        reservoir[$n$] $\leftarrow x$
    **else**
        sample $I \sim \mathcal{U}(\{1, \dots, n\})$
        **if** $I \leq k$ **then**
            reservoir[$I$] $\leftarrow x$
```

Theorem

Assume we call init(k) and then observeItem(x) for $x \in \{x_1, \dots, x_n\}$ with $n \geq k$. Afterwards `reservoir` contains every subset of $\{x_1, \dots, x_n\}$ of size k with equal probability.

Proof by induction (not here).

Example ($k = 3$)

↓

stream: § ♡ ♠ ♮ ♢ £ ⊕ ♣ ×

`reservoir`: | £ | ♮ | ♣ | $\mathcal{U}(\{1, \dots, 8\}) \rightsquigarrow I = 3$ ✓

Reservoir Sampling

Task: Maintain a fair sample of k items while reading a (possibly infinite) stream.

```
Algorithm init(k):
    allocate reservoir[1..k]
    $n \leftarrow 0$

Algorithm observeItem(x):
    $n \leftarrow n + 1$
    if $n \leq k$ then
        reservoir[n] $\leftarrow x$
    else
        sample $I \sim \mathcal{U}(\{1, \ldots, n\})$
        if $I \leq k$ then
            reservoir[I] $\leftarrow x$
```

Theorem

Assume we call init(k) and then observeItem(x) for $x \in \{x_1, \ldots, x_n\}$ with $n \geq k$. Afterwards `reservoir` contains every subset of $\{x_1, \ldots, x_n\}$ of size k with equal probability.

Proof by induction (not here).

Example ($k = 3$)

stream: $\S$ $\heartsuit$ $\spadesuit$ $\natural$ $\diamondsuit$ £ $\oplus$ $\clubsuit$ $\overset{\downarrow}{\times}$

reservoir: $\boxed{£}\boxed{\natural}\boxed{\clubsuit}$ $\mathcal{U}(\{1, \ldots, 9\})$

Reservoir Sampling

Task: Maintain a fair sample of k items while reading a (possibly infinite) stream.

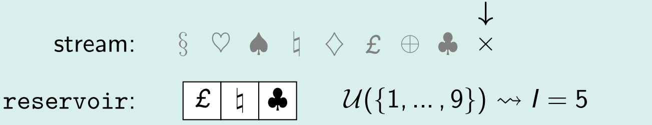
```
**Algorithm** init($k$):
    allocate reservoir[1..$k$]
    $n \leftarrow 0$

**Algorithm** observeItem($x$):
    $n \leftarrow n + 1$
    **if** $n \leq k$ **then**
        reservoir[$n$] $\leftarrow x$
    **else**
        sample $I \sim \mathcal{U}(\{1, \dots, n\})$
        **if** $I \leq k$ **then**
            reservoir[$I$] $\leftarrow x$
```

Theorem

Assume we call init(k) and then observeItem(x) for $x \in \{x_1, \dots, x_n\}$ with $n \geq k$. Afterwards `reservoir` contains every subset of $\{x_1, \dots, x_n\}$ of size k with equal probability.

Proof by induction (not here).

Example ($k = 3$)



stream: § ♡ ♠ ♮ ♢ £ ⊕ ♣ ×

reservoir: | £ | ♮ | ♣ |

$\mathcal{U}(\{1, \dots, 9\}) \rightsquigarrow I = 5$

Reservoir Sampling

Task: Maintain a fair sample of k items while reading a (possibly infinite) stream.

```
Algorithm init(k):
    allocate reservoir[1..k]
    $n \leftarrow 0$

Algorithm observeItem(x):
    $n \leftarrow n + 1$
    if $n \le k$ then
        reservoir[n] $\leftarrow x$
    else
        sample $I \sim \mathcal{U}(\{1, \dots, n\})$
        if $I \le k$ then
            reservoir[I] $\leftarrow x$
```

Theorem

Assume we call init(k) and then observeItem(x) for $x \in \{x_1, \dots, x_n\}$ with $n \ge k$. Afterwards `reservoir` contains every subset of $\{x_1, \dots, x_n\}$ of size k with equal probability.

Proof by induction (not here).

Example ($k = 3$)

stream: § ♡ ♠ ♮ ♢ £ ⊕ ♣ $\overset{\downarrow}{\times}$

reservoir: $\boxed{£}\boxed{\natural}\boxed{\clubsuit}$ $\mathcal{U}(\{1, \dots, 9\}) \rightsquigarrow I = 5$ ✗

Reservoir Sampling

Task: Maintain a fair sample of k items while reading a (possibly infinite) stream.

```
Algorithm init($k$):
    allocate reservoir[$1..k$]
    $n \leftarrow 0$
Algorithm observeItem($x$):
    $n \leftarrow n + 1$
    if $n \leq k$ then
        reservoir[$n$] $\leftarrow x$
    else
        sample $I \sim \mathcal{U}(\{1, \ldots, n\})$
        if $I \leq k$ then
            reservoir[$I$] $\leftarrow x$
```

Theorem

Assume we call init(k) and then observeItem(x) for $x \in \{x_1, \ldots, x_n\}$ with $n \geq k$. Afterwards `reservoir` contains every subset of $\{x_1, \ldots, x_n\}$ of size k with equal probability.

Proof by induction (not here).

Example ($k = 3$)

stream: § ♡ ♠ ♮ ♢ £ ⊕ ♣ ×

`reservoir`: | £ | ♮ | ♣ |

Conclusion

General Techniques

- rejection sampling
- inverse transform sampling

Distributions

- Bernoulli distribution
- uniform distribution
- geometric distribution

Other Stuff

- sampling from a set without replacement
- reservoir sampling

Anhang: Mögliche Prüfungsfragen I

- Wie kann man $B \sim Ber(p)$ sampeln? Wie $X \sim \mathcal{U}(\{1, \dots, n\})$? Unter welchen Annahmen?
- Wie funktioniert Rejection Sampling allgemein? Unter welchen Voraussetzungen führt Rejection Sampling zu einem effizienten Algorithmus?
- Wie funktioniert Inverse Transform Sampling allgemein? Unter welchen Voraussetzungen führt Inverse Transform Sampling zu einem effizienten Algorithmus?
- Wie kann man einen zufälligen Punkt einer Kreisscheibe sampeln? Nenne zwei Techniken und nenne Vor- bzw. Nachteile.
- Gegeben eine Menge der Größe n. Wie kann ich eine zufällige Teilmenge der Größe $k \leq n$ bestimmen und wie lange dauert das?
- Erkläre Reservoir Sampling. Ist das nicht einfach ein langsamerer Algorithmus für „Sampling without Replacement“?