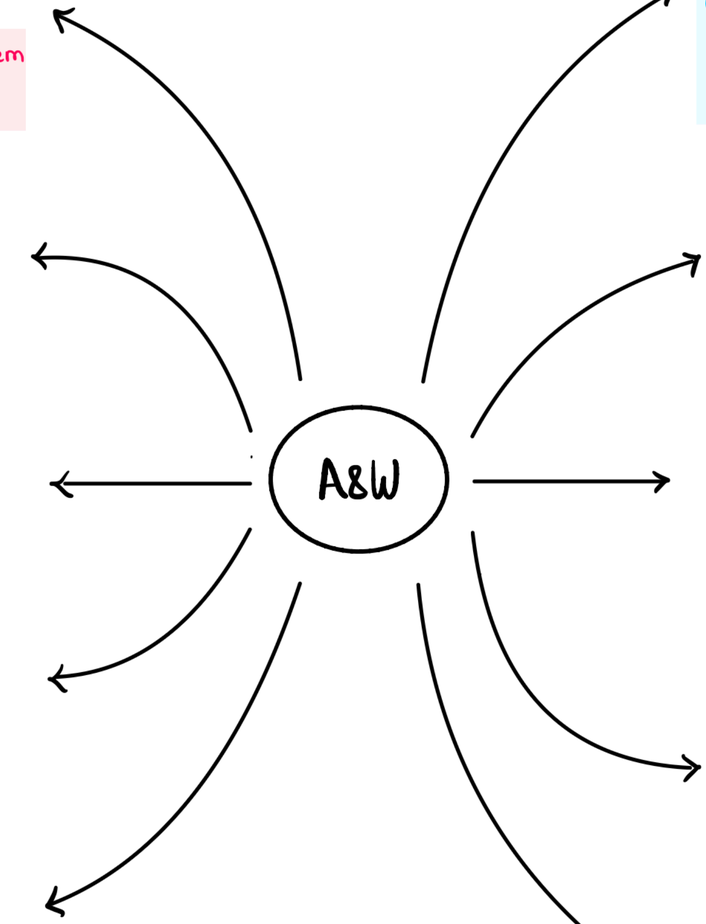
A&W

Exercise Session 9

Randomized Algorithms

Nil Ozer

A&W Overview



Connectivity

- Articulation Points
- Bridges
- Block - Decomposition
- Menger's Theorem

Cycles

- Eulerian Cycle
- Hamiltonian Cycle
- TSP

Matchings

- Definition
- Algorithms
- Hall's Theorem

Colorings

- Definition
- Algorithm
- Brooks's Theorem

Wahrscheinlichkeit

- Grundbegriffe und Notationen
- Bedingte Wahrscheinlichkeiten
- Unabhängigkeiten
- Zufallsvariablen
- Wichtige Diskrete Verteilungen
- Abschätzen von Wahrscheinlichkeiten

Randomized Algorithms

- Las-Vegas
- Monte-Carlo
- Longest Path Problem
- Primality Test
- Target-Shooting
- Finding Duplicates

Flow

- Definition
- Maxflow - Mincut
- Ford - Fulkerson
- Matching w. Flow
- Edge-disjoint paths w. Flow

Minimum Cut

- Definition
- Cut(G) Algorithm
- Bootstrapping

Graph Algorithms

Convex Hull

- Definition
- Jarvis Wrap
- Local optimization

Smallest Enclosing Circle

- Definition
- First Algorithm
- Final Algorithm

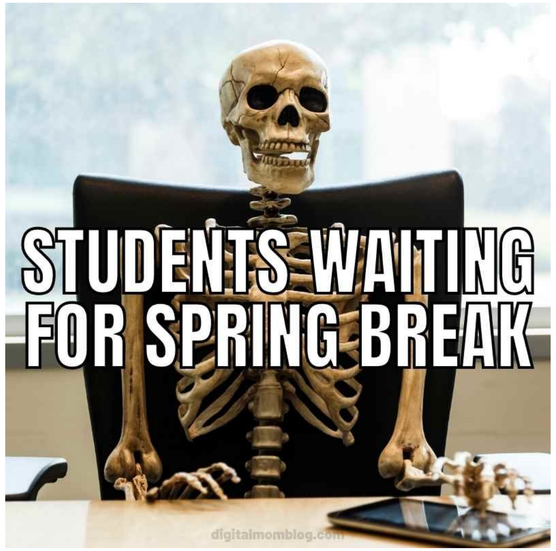
Geometric Algorithms

Outline

- Probability Kahoot

- Randomized Algorithms I

Spring Break



Probability Kahoot

Let’s take a break

Randomized Algorithms

Randomized Algorithms

Classic vs. Randomized

classic

Input I

↓

Algorithm A

↓

Output $A(I)$

randomized

Input I

↓

Algorithm A ⟷ Random Variable R

random bits, random numbers

↓

Output $A(I, R)$

Randomized Algorithms

Classic vs. Randomized

classic

Input I

Algorithm A

Output $A(I)$

randomized

Input I

Algorithm A RNG R

random bits, random numbers

Output $A(I, R)$

Randomized Algorithms

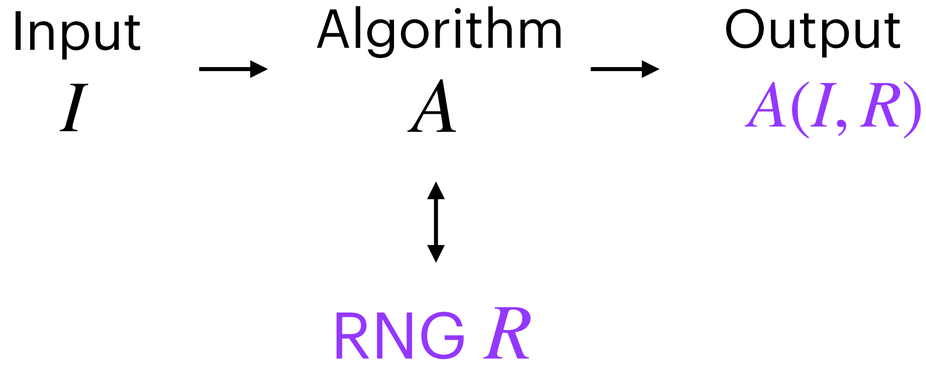
Classic vs. Randomized

classic

Input I → Algorithm A → Output $A(I)$

- $A(I)$ is correct for all I
- The runtime is O(f(n)) for all I with $|I| = n$

randomized



- $A(I, R)$ is correct with $Pr_R[A(I, R)\ is\ correct] \geq$ ... for all I
- The runtime is O(f(n)) and/or $Pr_R[\text{Runtime} \leq \text{O(f(n))}\ ] \geq$ ... for all I with $|I| = n$

Randomized Algorithms

Classic vs. Randomized

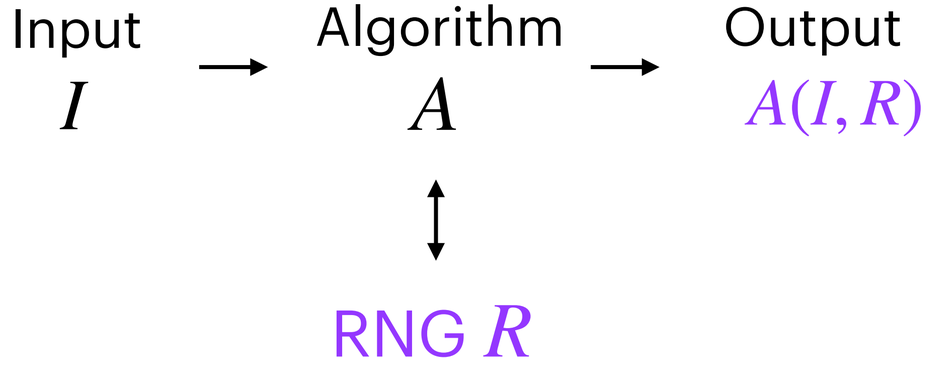
$A(I, R)$ can't be reproduced

classic

Input I → Algorithm A → Output $A(I)$

- $A(I)$ is correct and definite for all I
- The runtime is O(f(n)) for all I with $|I| = n$

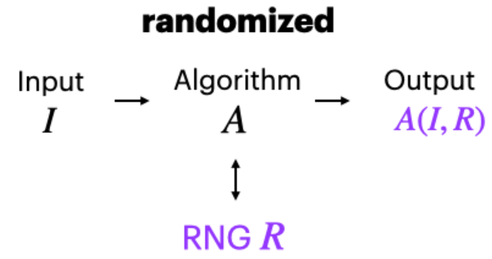
randomized



- $A(I, R)$ is correct with $Pr_R[A(I, R) \ is \ correct] \geq \ldots$ for all I
- The runtime is O(f(n)) and/or $Pr_R[\text{Runtime} \leq O(f(n))] \geq \ldots$ for all I with $|I| = n$

Randomized Algorithms

Las-Vegas vs. Monte-Carlo



Las-Vegas

- can output true answer
- cannot output false answer
- can run forever/ can output no answer (???)

Runtime is the RV

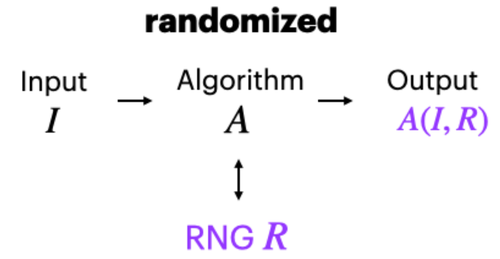
Monte-Carlo

- can output true answer
- can output false answer
- always outputs an answer

Correctness/Quality is the RV

Randomized Algorithms

Las-Vegas vs. Monte-Carlo



Las-Vegas

Runtime is the RV

- can output true answer
- cannot output false answer
- can run forever/ can output no answer (???)

Monte-Carlo

Correctness/Quality is the RV

- can output true answer
- can output false answer
- always outputs an answer

Input: An array of $n \geq 2$ elements, in which half are '*a*'s and the other half are '*b*'s.

Output: Find an '*a*' in the array.

```
findingA_LV(array A, n)
begin
    repeat
        Randomly select one element out of n elements.
    until 'a' is found
end
```

```
findingA_MC(array A, n, k)
begin
    i := 0
    repeat
        Randomly select one element out of n elements.
        i := i + 1
    until i = k or 'a' is found
end
```

Randomized Algorithms

Reducing the error probability - Las Vegas

Las-Vegas Runtime is the RV

- can output true answer
- cannot output false answer
- can run forever/ can output no answer (???)

To reduce the error probability by a constant factor, only a constant number of additional iterations are necessary

Let A be a randomized algorithm that never returns an incorrect answer but sometimes outputs '???' where $\Pr[A(I) \text{ correct}] \geq \varepsilon$

A_δ : invokes A until either a value different from '???' is returned or '???' is returned $N = \lceil \varepsilon^{-1} \ln \delta^{-1} \rceil$ times

Then it holds that $\Pr[A_\delta(I) \text{ correct}] \geq 1 - \delta$

$\varepsilon = 0.25$

δ	$N = \lceil \varepsilon^{-1} \ln \delta^{-1} \rceil$
0.1	10
0.01	19
0.001	28
0.0001	37
0.00001	47
0.000001	56

Randomized Algorithms

Reducing the error probability - Monte-Carlo

Fehlerreduktionen:

- **Wiederholung MC:** Eine N-fache Wiederholung mit $N = 4\varepsilon^{-2} \ln \delta^{-1}$ steigert die Erfolgswahrscheinlichkeit eines Monte-Carlo-Algorithmus von $\frac{1}{2} + \varepsilon$ auf $\geq 1 - \delta$.
- **Wiederholung MC mit einseitigem Fehler:** Eine N-fache Wiederholung mit $N = \varepsilon^{-1} \ln \delta^{-1}$ steigert fÃ¼r einen Monte-Carlo-Algorithmus mit einseitigem Fehler die Erfolgswahrscheinlichkeit von ε auf $\geq 1 - \delta$.
- **Target Shooting:** Bestimmt der Target-Shooting-Algorithmus eine Menge $S \subseteq U$ mit $N \geq 3\frac{|U|}{|S|}\varepsilon^{-2} \ln(2/\delta)$ Versuchen, so ist die Ausgabe mit Wahrscheinlichkeit $\geq 1 - \delta$ im Intervall $\left[(1-\varepsilon)\frac{|S|}{|U|}, (1+\varepsilon)\frac{|S|}{|U|}\right]$.

Target-Shooting

Problem Description

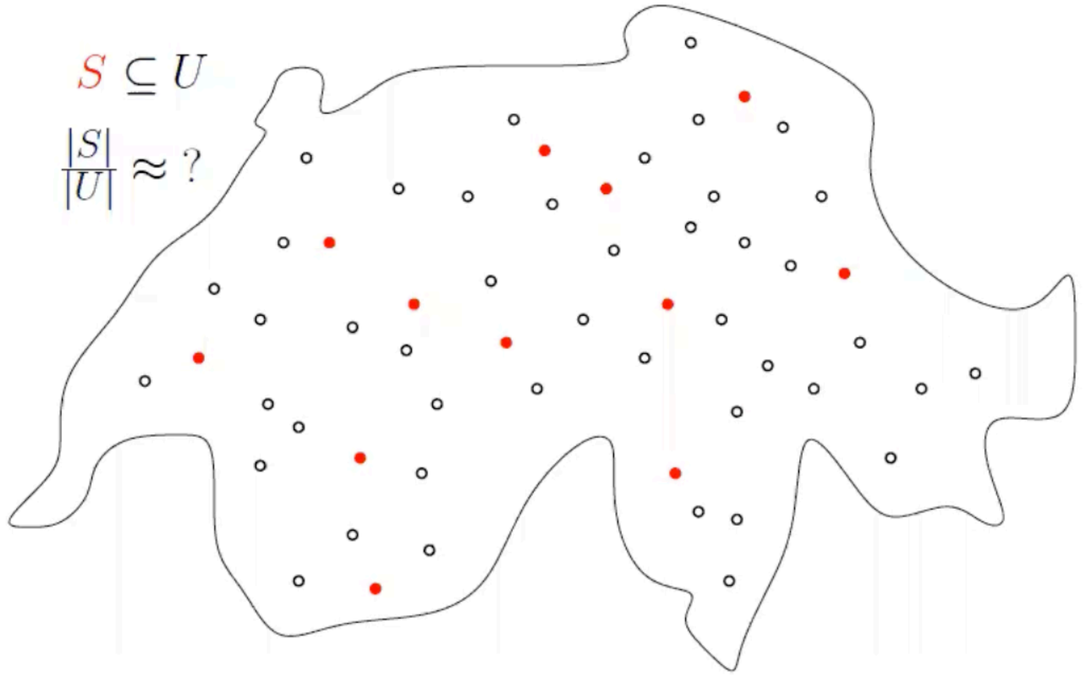
given : finite sets S and U with $S \subseteq U$

to find : $\dfrac{|S|}{|U|}$

We can generate elements u in U uniformly distributed

$$I_S : U \to \{0,1\}$$

$$I_S(u) = 1 \iff u \in S$$



Target-Shooting

Problem Description

given : finite sets S and U with $S \subseteq U$

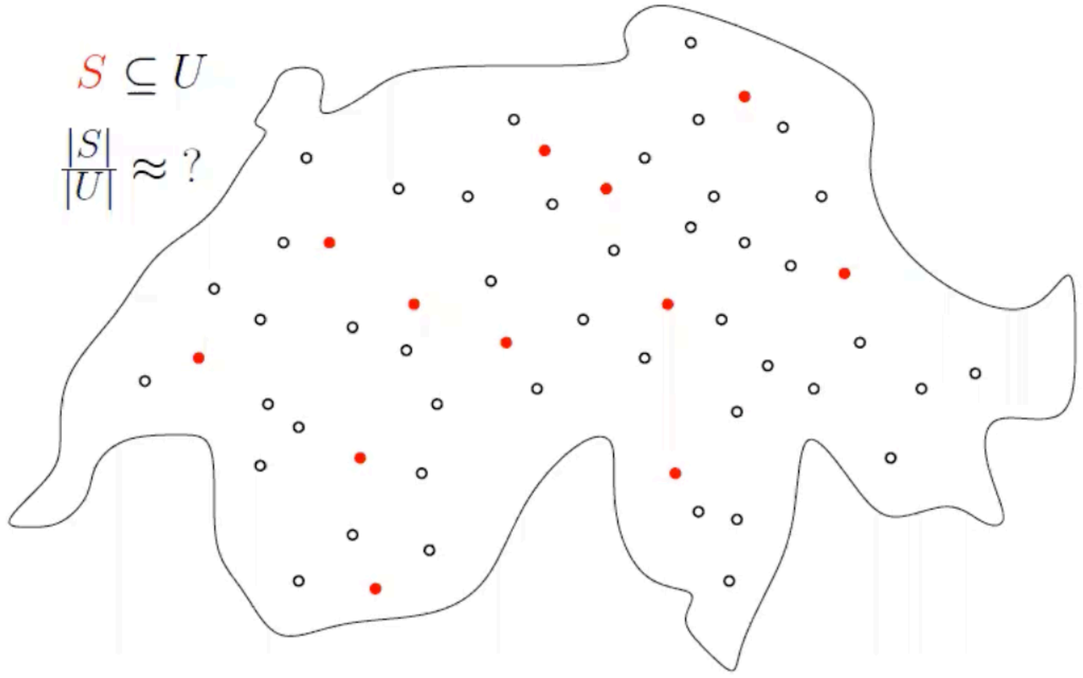
to find : $\approx \frac{|S|}{|U|}$

U is very large. We cannot afford to iterate through U

We can generate elements u in U uniformly distributed

$I_S : U \to \{0,1\}$

$I_S(u) = 1 \iff u \in S$



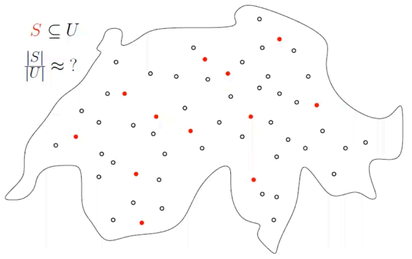
Target-Shooting

Algorithm

given : finite sets S and U with $S \subseteq U$

to find : $\approx \frac{|S|}{|U|}$

$$I_S(u) = 1 \iff u \in S$$



1 : Pick $u_1, \ldots, u_N$ from U randomly, uniformly and independently

2 : Return $\frac{1}{N} \cdot \sum_{i=1}^{N} I_S(u_i)$

Target-Shooting

Algorithm

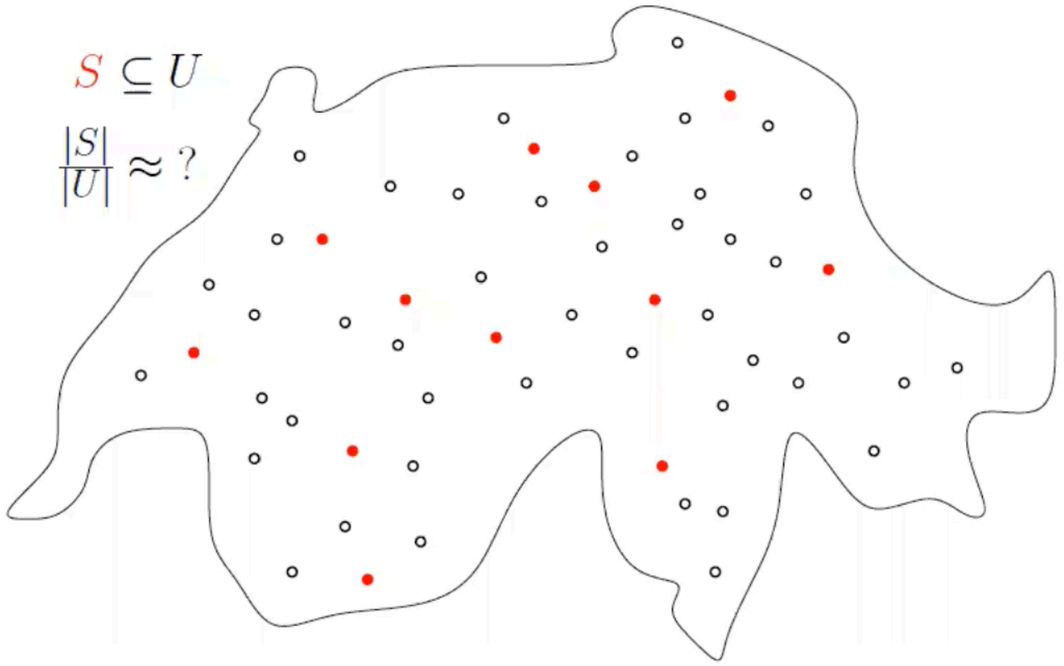
given : finite sets S and U with $S \subseteq U$

to find : $\approx \frac{|S|}{|U|}$

$$I_S(u) = 1 \iff u \in S$$

1 : Pick $u_1, \ldots, u_N$ from U randomly, uniformly and independently

2 : Return $\frac{1}{N} \cdot \sum_{i=1}^{N} I_S(u_i)$



Target-Shooting

Algorithm

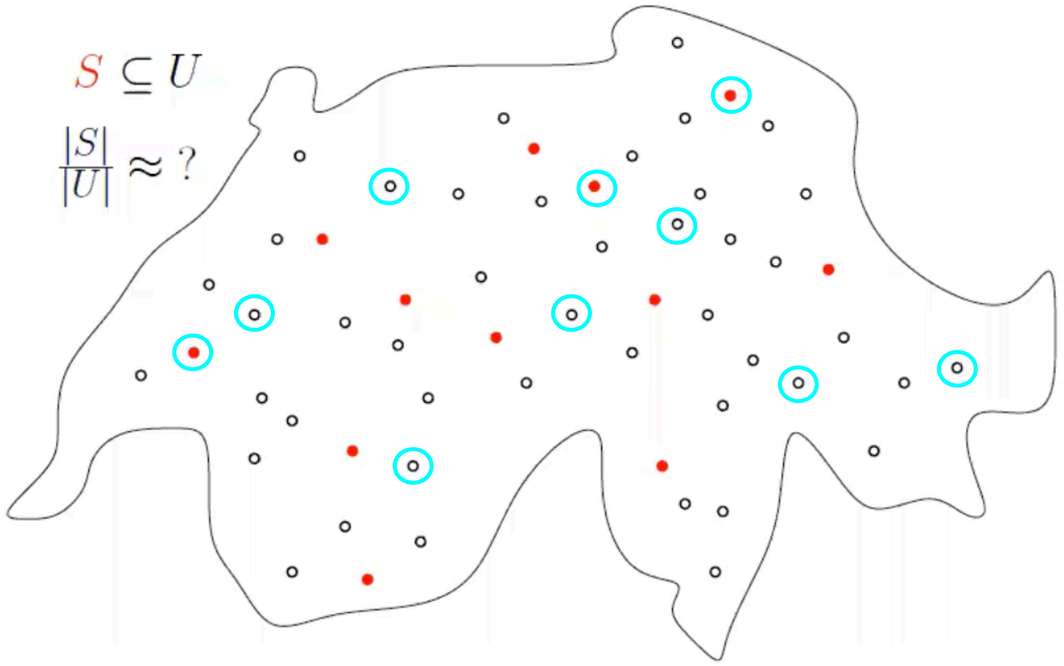
given : finite sets S and U with $S \subseteq U$

to find : $\approx \frac{|S|}{|U|}$

$$I_S(u) = 1 \iff u \in S$$

1 : Pick $u_1, \ldots, u_N$ from U randomly, uniformly and independently

2 : Return $\frac{1}{N} \cdot \sum_{i=1}^{N} I_S(u_i)$



Target-Shooting

Algorithm

given : finite sets S and U with $S \subseteq U$

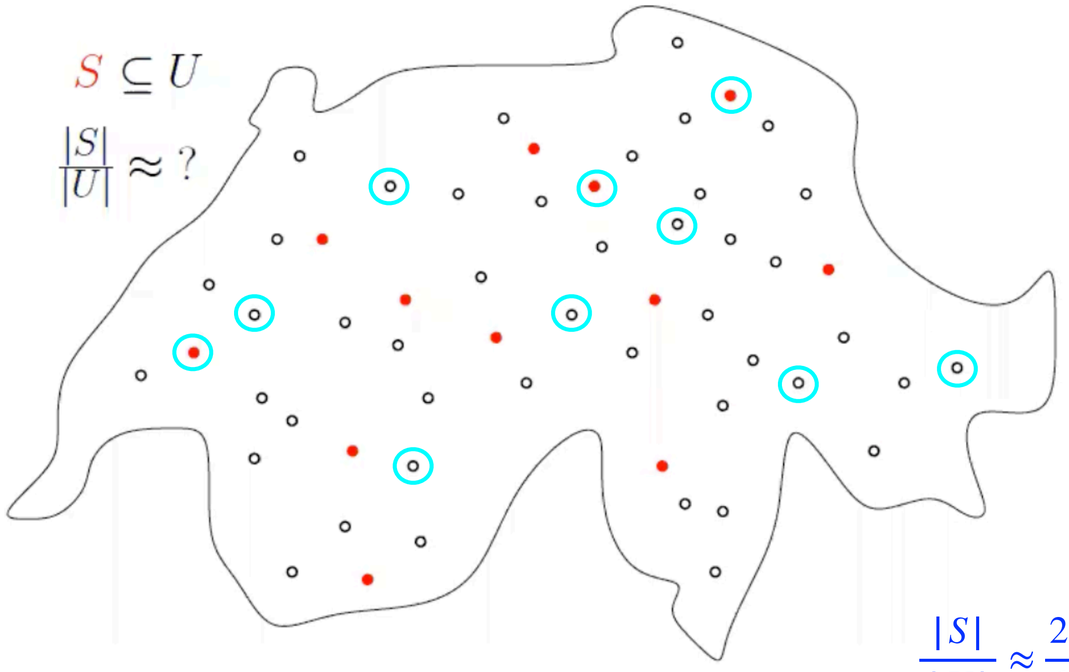
to find : $\approx \frac{|S|}{|U|}$

$$I_S(u) = 1 \iff u \in S$$

1 : Pick $u_1, \ldots, u_N$ from U randomly, uniformly and independently

2 : Return $\frac{1}{N} \cdot \sum_{i=1}^{N} I_S(u_i)$

$$\frac{1}{10} \cdot \sum_{i=1}^{10} I_S(u_i) = \frac{3}{10}$$



$$\frac{|S|}{|U|} \approx \frac{20}{64} = 0.3125$$

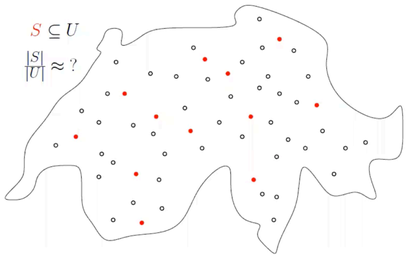
Target-Shooting

Algorithm

given : finite sets S and U with $S \subseteq U$

to find : $\approx \frac{|S|}{|U|}$

$$I_S(u) = 1 \iff u \in S$$



1 : Pick $u_1, \ldots, u_N$ from U randomly, uniformly and independently

2 : Return $\frac{1}{N} \cdot \sum_{i=1}^{N} I_S(u_i)$

Fehlerreduktionen:

- **Wiederholung MC:** Eine N-fache Wiederholung mit $N = 4\varepsilon^{-2} \ln \delta^{-1}$ steigert die Erfolgswahrscheinlichkeit eines Monte-Carlo-Algorithmus von $\frac{1}{2} + \varepsilon$ auf $\geq 1 - \delta$.
- **Wiederholung MC mit einseitigem Fehler:** Eine N-fache Wiederholung mit $N = \varepsilon^{-1} \ln \delta^{-1}$ steigert fÃ¼r einen Monte-Carlo-Algorithmus mit einseitigem Fehler die Erfolgswahrscheinlichkeit von ε auf $\geq 1 - \delta$.
- **Target Shooting:** Bestimmt der Target-Shooting-Algorithmus eine Menge $S \subseteq U$ mit $N \geq 3\frac{|U|}{|S|}\varepsilon^{-2} \ln(2/\delta)$ Versuchen, so ist die Ausgabe mit Wahrscheinlichkeit $\geq 1 - \delta$ im Intervall $\left[(1-\varepsilon)\frac{|S|}{|U|}, (1+\varepsilon)\frac{|S|}{|U|}\right]$.

Finding Duplicates

Problem Description

given : A dataset $D = (s_1, s_2, \ldots, s_n)$, is a sequence of n elements

to find : find all duplicates in D (i, j) with $1 \leq i < j \leq n$ is a duplicate in D if $s_i = s_j$

$$\begin{array}{lccccccc} \mathcal{D} = (& A, & C, & B, & Z, & C, & B, & C \quad) \\ & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array}$$

$$\text{Dupl}(\mathcal{D}) = \{(2,5), (2,7), (5,7), (3,6)\}$$

Finding Duplicates

Problem Description

Elements in D are very large.

Storing and comparing is expensive

given : A dataset $D = (s_1, s_2, \ldots, s_n)$, is a sequence of n elements

to find : find all duplicates in D (i, j) with $1 \leq i < j \leq n$ is a duplicate in D if $s_i = s_j$

Hashfunction h :

$h : U \rightarrow [m]$ $[m] = \{1,2,...,m\}$

h is efficiently computable

h behaves like a random variable

$\forall u \in U \ \forall i \in [m] : \quad \Pr[h(u) = i] = \frac{1}{m}$ (independent for different u)

Finding Duplicates

Problem Description

Elements in D are very large.

Storing and comparing is expensive

given : A dataset $D = (s_1, s_2, \ldots, s_n)$, is a sequence of n elements

to find : find all duplicates in D (i, j) with $1 \leq i < j \leq n$ is a duplicate in D if $s_i = s_j$

Hashfunction h :

$$h : U \to [m] \quad [m] = \{1,2,...,m\} \quad \forall u \in U \ \forall i \in [m] : \quad \Pr[h(u) = i] = \frac{1}{m}$$

Each $h(s_i)$ is uniformly randomly distributed in $[m]$ BUT

$$s_i = s_j \implies h(s_i) = h(s_j)$$

Our m is much smaller than $|U|$ (compression)

Finding Duplicates

Algorithm

given : A dataset $D = (s_1, s_2, \ldots, s_n)$, is a sequence of n elements

to find : find all duplicates in D (i, j) with $1 \leq i < j \leq n$ is a duplicate in D if $s_i = s_j$

Hashfunction h :

$$h : U \to [m] \quad [m] = \{1,2,\ldots,m\} \quad \forall u \in U \; \forall i \in [m] : \quad \Pr[h(u) = i] = \frac{1}{m}$$

Each $h(s_i)$ is uniformly randomly distributed in $[m]$ BUT

$$s_i = s_i \implies h(s_i) = h(s_j)$$

Our m is much smaller than $|U|$ (compression)

	A	C	B	Z	C	B	C
hashing:	$(\underbrace{31}_{h(A)}, 1)$	$(\underbrace{27}_{h(C)}, 2)$	$(\underbrace{12}_{h(B)}, 3)$	$(\underbrace{12}_{h(Z)}, 4)$	$(\underbrace{27}_{h(C)}, 5)$	$(\underbrace{12}_{h(B)}, 6)$	$(\underbrace{27}_{h(C)}, 7)$
sorting:	$(12, 3)$	$(12, 4)$	$(12, 6)$	$(27, 2)$	$(27, 5)$	$(27, 7)$	$(31, 1)$
duplicates:		$(3, 6)$,			$(2, 5), (2, 7), (5, 7)$		

Finding Duplicates

Challenge : Collisions

given : A dataset $D = (s_1, s_2, \ldots, s_n)$, is a sequence of n elements

to find : find all duplicates in D (i, j) with $1 \leq i < j \leq n$ is a duplicate in D if $s_i = s_j$

Hashfunction h :

$h : U \to [m]$ $[m] = \{1,2,\ldots,m\}$ $\forall u \in U \ \forall i \in [m] :$ $\Pr[h(u) = i] = \frac{1}{m}$

Each $h(s_i)$ is uniformly randomly distributed in $[m]$ BUT

$$s_i = s_i \implies h(s_i) = h(s_j)$$

Our m is much smaller than $|U|$ (compression)

	A	C	B	Z	C	B	C
hashing:	$(\underbrace{31}_{h(A)}, 1)$	$(\underbrace{27}_{h(C)}, 2)$	$(\underbrace{12}_{h(B)}, 3)$	$(\underbrace{12}_{h(Z)}, 4)$	$(\underbrace{27}_{h(C)}, 5)$	$(\underbrace{12}_{h(B)}, 6)$	$(\underbrace{27}_{h(C)}, 7)$
sorting:	$(12, 3)$	$(12, 4)$	$(12, 6)$	$(27, 2)$	$(27, 5)$	$(27, 7)$	$(31, 1)$
duplicates:		$(3, 6)$,		$(2, 5), (2, 7), (5, 7)$			

collision : $h(B) = h(Z)$

Finding Duplicates

Challenge : Collisions

given : A dataset $D = (s_1, s_2, \ldots, s_n)$, is a sequence of n elements

to find : find all duplicates in D (i, j) with $1 \leq i < j \leq n$ is a duplicate in D if $s_i = s_j$

Hashfunction h :

$h : U \to [m] \quad [m] = \{1,2,\ldots,m\} \quad \forall u \in U \; \forall i \in [m] : \quad \Pr[h(u) = i] = \frac{1}{m}$

Each $h(s_i)$ is uniformly randomly distributed in $[m]$ BUT

$$s_i = s_i \implies h(s_i) = h(s_j)$$

Our m is much smaller than $|U|$ (compression)

Collision :

The new, undesired duplicates in the hashmap

the pairs (i, j) , $1 \leq i < j \leq n,$ with $s_i \neq s_j$ and $h(s_i) = h(s_j)$

Finding Duplicates

Challenge : Collisions

given : A dataset $D = (s_1, s_2, \ldots, s_n)$, is a sequence of n elements

to find : find all duplicates in D (i,j) with $1 \leq i < j \leq n$ is a duplicate in D if $s_i = s_j$

Hashfunction h :

$h : U \to [m] \quad [m] = \{1,2,\ldots,m\} \quad \forall u \in U \; \forall i \in [m] : \quad \Pr[h(u) = i] = \frac{1}{m}$

Each $h(s_i)$ is uniformly randomly distributed in $[m]$ BUT

$s_i = s_i \implies h(s_i) = h(s_j)$

Our m is much smaller than $|U|$ (compression)

Collision :

The new, undesired duplicates in the hashmap

the pairs (i,j) , $1 \leq i < j \leq n,$ with $s_i \neq s_j$ and $h(s_i) = h(s_j)$

$\mathbb{E}$[#Collisions] :

$K_{i,j}$ bernoulli RV. with : $K_{i,j} = 1 \iff (i,j)$ is a collision

$$\Pr[K_{i,j} = 1] = \begin{cases} \frac{1}{m} & \text{if } s_i \neq s_j \\ 0 & \text{otherwise} \end{cases} \qquad \mathbb{E}[K_{i,j}] \leq \frac{1}{m}$$

$$\mathbb{E}[\#\text{Collisions}] = \sum_{1 \leq i < j \leq n} \mathbb{E}[K_{i,j}] \leq \binom{n}{2} \cdot \frac{1}{m}$$

Finding Duplicates

Challenge : Collisions

given : A dataset $D = (s_1, s_2, \ldots, s_n)$, is a sequence of n elements

to find : find all duplicates in D (i,j) with $1 \leq i < j \leq n$ is a duplicate in D if $s_i = s_j$

Hashfunction h :

$h : U \to [m] \quad [m] = \{1,2,\ldots,m\} \quad \forall u \in U \; \forall i \in [m] : \quad \Pr[h(u) = i] = \frac{1}{m}$

Each $h(s_i)$ is uniformly randomly distributed in $[m]$ BUT

$s_i = s_i \implies h(s_i) = h(s_j)$

Our m is much smaller than $|U|$ (compression)

Collision :

The new, undesired duplicates in the hashmap

the pairs (i,j) , $1 \leq i < j \leq n,$ with $s_i \neq s_j$ and $h(s_i) = h(s_j)$

$\mathbb{E}$[#Collisions] :

$K_{i,j}$ bernoulli RV. with : $K_{i,j} = 1 \iff (i,j)$ is a collision

$$\Pr[K_{i,j} = 1] = \begin{cases} \frac{1}{m} & \text{if } s_i \neq s_j \\ 0 & \text{otherwise} \end{cases} \qquad \mathbb{E}[K_{i,j}] \leq \frac{1}{m}$$

$$\mathbb{E}[\#\text{Collisions}] \leq \binom{n}{2} \cdot \frac{1}{m} < 1 \quad \text{for } m = n^2$$

Finding Duplicates

Runtime

given : A dataset $D = (s_1, s_2, \ldots, s_n)$, is a sequence of n elements

to find : find all duplicates in D (i, j) with $1 \leq i < j \leq n$ is a duplicate in D if $s_i = s_j$

Hashfunction h :

$h : U \to [m] \quad [m] = \{1,2,\ldots,m\} \quad \forall u \in U \; \forall i \in [m] : \quad \Pr[h(u) = i] = \frac{1}{m}$

Each $h(s_i)$ is uniformly randomly distributed in $[m]$ BUT

$s_i = s_i \implies h(s_i) = h(s_j)$

Our m is much smaller than $|U|$ (compression)

Collision :

The new, undesired duplicates in the hashmap

the pairs (i, j) , $1 \leq i < j \leq n,$ with $s_i \neq s_j$ and $h(s_i) = h(s_j)$

$$\mathbb{E}[\#\text{Collisions}] \leq \binom{n}{2} \cdot \frac{1}{m} < 1 \quad \text{for } m = n^2$$

Runtime :

- n hash computations
- sorting in O(n log n)
- duplicate check comparisons (|Dupl(D)|+#Kollisionen) ≈ O(n)

additional memory

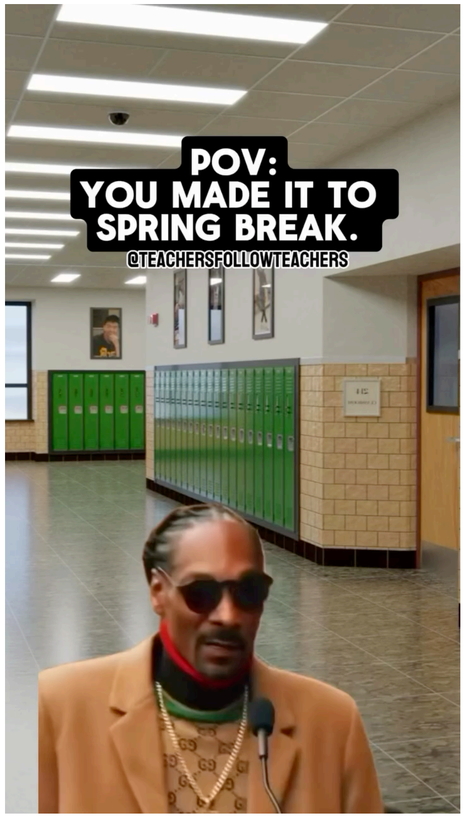
$$\underbrace{O(n \log n)}_{\text{indices}} + \underbrace{O(n \log m)}_{\text{hash values}} \overset{m=n^2}{=} O(n \log n)$$

Overall : O(n log n)

Final Announcements

- Easter break is the best !!

- For us :
  - Revisiting is possible .
  - Give me feedback with the form
  - I’m one text away ! (can take some time)



Questions

Feedbacks , Recommendations

Nil Ozer