Lovász Local Lemma as Double Counting

nonrepetitive sequences

Jakub Kozik
(based on joint work with Piotr Micek)

Theoretical Computer Science
Jagiellonian University

AofA 2011 in Będlewo, Poland

It would be hopeless to replace the applications of many of the tools appearing in this book, including, for example, the second moment method, the Lovász Local Lemma and the concentration via martingales by counting arguments, even when these are applied to finite probability spaces.

N. Alon, J. Spencer, The Probabilistic Method

It would be hopeless to replace the applications of many of the tools appearing in this book, including, for example, the second moment method, the Lovász Local Lemma and the concentration via martingales by counting arguments, even when these are applied to finite probability spaces.

N. Alon, J. Spencer, The Probabilistic Method

It would be hopeless to replace the applications of many of the tools appearing in this book, including, for example, the second moment method, the Lovász Local Lemma and the concentration via martingales by counting arguments, even when these are applied to finite probability spaces.

N. Alon, J. Spencer, The Probabilistic Method

Outline

- Lovász Local Lemma (LLL)
- application to nonrepetitive sequences
- constructive proof of LLL (by Moser and Tardos)
- application of double counting to nonrepetitive sequences

Lovász Local Lemma (Lovász, Erdös 1975)

Let $\mathcal{A}$ be a finite set of events in a probability space. For $A \in \mathcal{A}$ let $\Gamma(A)$ be a subset of $\mathcal{A}$ satisfying that A is independent from the collection of events $\mathcal{A} - (\{A\} \cup \Gamma(A))$. If there exists an assignment of reals $x : \mathcal{A} \to (0, 1)$ such that

$$\textit{for all } A \in \mathcal{A} \ : \ Pr(A) \leq x(A) \prod_{B \in \Gamma(A)} (1 - x(B)),$$

then the probability of avoiding all events in $\mathcal{A}$ is at least

$$\prod_{A \in \mathcal{A}} (1 - x(A)).$$

Lovász Local Lemma (Lovász, Erdös 1975)

Let $\mathcal{A}$ be a finite set of events in a probability space. For $A \in \mathcal{A}$ let $\Gamma(A)$ be a subset of $\mathcal{A}$ satisfying that A is independent from the collection of events $\mathcal{A} - (\{A\} \cup \Gamma(A))$. If there exists an assignment of reals $x : \mathcal{A} \to (0, 1)$ such that

$$\textit{for all } A \in \mathcal{A} \ : \ Pr(A) \leq x(A) \prod_{B \in \Gamma(A)} (1 - x(B)),$$

then the probability of avoiding all events in $\mathcal{A}$ is at least

$$\prod_{A \in \mathcal{A}} (1 - x(A)).$$

Nonrepetitive sequences

A repetition of size h in a sequence is a substring of the form:

$$xx = x_1 \dots x_h x_1 \dots x_h.$$

Examples:

Nonrepetitive sequences

A repetition of size h in a sequence is a substring of the form:

$$xx = x_1 \dots x_h x_1 \dots x_h.$$

Examples:

- `lemma`

Nonrepetitive sequences

A repetition of size h in a sequence is a substring of the form:

$$xx = x_1 \dots x_h x_1 \dots x_h.$$

Examples:

- le**mm**a
- repe**titi**ve

Nonrepetitive sequences

A repetition of size h in a sequence is a substring of the form:

$$xx = x_1 \dots x_h x_1 \dots x_h.$$

Examples:

- le<u>mm</u>a
- repe<u>titi</u>ve
- no<u>nsense</u>

Nonrepetitive sequences

A repetition of size h in a sequence is a substring of the form:

$$xx = x_1 \dots x_h x_1 \dots x_h.$$

Examples:

- lemma
- repetitive
- nonsense

A sequence is nonrepetitive if it does not contain a repetition of any size.

Nonrepetitive sequences

A repetition of size h in a sequence is a substring of the form:

$$xx = x_1 \dots x_h x_1 \dots x_h.$$

Examples:

- lemma
- repetitive
- nonsense

A sequence is nonrepetitive if it does not contain a repetition of any size.

Question

How many symbols do we need to build arbitrarily long nonrepetitive sequence?

Nonrepetitive sequences - LLL

Fix $n \in \mathbb{N}$. Consider random word $\omega = \omega_1 \dots \omega_n$. (each letter chosen independently from the set of k symbols with uniform distribution)

Nonrepetitive sequences - LLL

Fix $n \in \mathbb{N}$. Consider random word $\omega = \omega_1 \dots \omega_n$. (each letter chosen independently from the set of k symbols with uniform distribution)

Events

$A_{i,m}$ - substring $\omega_i \omega_{i+1} \dots \omega_{i+2m-1}$ is a repetition

$$Pr(A_{i,m}) = k^{-m}$$

Nonrepetitive sequences - LLL

Fix $n \in \mathbb{N}$. Consider random word $\omega = \omega_1 \dots \omega_n$. (each letter chosen independently from the set of k symbols with uniform distribution)

Events

$A_{i,m}$ - substring $\omega_i \omega_{i+1} \dots \omega_{i+2m-1}$ is a repetition

$$Pr(A_{i,m}) = k^{-m}$$

Independencies

$A_{i,m}$ independent from all events $A_{j,p}$ for which

$$\{i, i+1, ..., i+2m-1\} \cap \{j, j+1, \dots, j+2p-1\} = \emptyset$$

Nonrepetitive sequences - LLL

Fix $n \in \mathbb{N}$. Consider random word $\omega = \omega_1 \dots \omega_n$. (each letter chosen independently from the set of k symbols with uniform distribution)

Events

$A_{i,m}$ - substring $\omega_i \omega_{i+1} \dots \omega_{i+2m-1}$ is a repetition

$$Pr(A_{i,m}) = k^{-m}$$

Independencies

$A_{i,m}$ independent from all events $A_{j,p}$ for which

$$\{i, i+1, ..., i+2m-1\} \cap \{j, j+1, \dots, j+2p-1\} = \emptyset$$

Assignment

$$x(A_{i,m}) := \frac{1}{6^m + 1}$$

Nonrepetitive sequences - LLL

Fix $n \in \mathbb{N}$. Consider random word $\omega = \omega_1 \dots \omega_n$. (each letter chosen independently from the set of k symbols with uniform distribution)

Events

$A_{i,m}$ - substring $\omega_i \omega_{i+1} \dots \omega_{i+2m-1}$ is a repetition

$$Pr(A_{i,m}) = k^{-m}$$

Independencies

$A_{i,m}$ independent from all events $A_{j,p}$ for which

$$\{i, i+1, ..., i+2m-1\} \cap \{j, j+1, \dots, j+2p-1\} = \emptyset$$

Assignment

$$x(A_{i,m}) := \frac{1}{6^m + 1}$$

Result

For $k \geq 14$ the assumptions of LLL hold. Therefore with positive probability the word ω is nonrepetitive.

Thue's theorem

Theorem (Thue 1906)

There exist arbitrarily long nonrepetitive sequences over 3 symbols.

Thue's theorem

Theorem (Thue 1906)

There exist arbitrarily long nonrepetitive sequences over 3 symbols.

Thue's substitution:

$$a \to abcab, \quad b \to acabcb, \quad c \to acbcacb$$

sequence chosen from lists

$L_1, \ldots, L_n$ – sets of symbols (called lists)

L_1	L_2	L_3	L_4	L_5	L_6	L_7
a	b	a	c	f	b	c
b	d	b	h	g	c	e
c	e	g			h	g
d	g	h				h
e		i				

A sequence $s_1, \ldots, s_n$ is chosen from $L_1, \ldots, L_n$ if $s_i \in L_i$ for all i.

sequence chosen from lists

$L_1, \ldots, L_n$ – sets of symbols (called lists)

L_1	L_2	L_3	L_4	L_5	L_6	L_7
a	b	a	c	f	b	c
b	d	b	h	g	c	e
c	e	g			h	g
d	g	h				h
e		i				

A sequence $s_1, \ldots, s_n$ is chosen from $L_1, \ldots, L_n$ if $s_i \in L_i$ for all i.

Real Question

What is the minimum T such that from any sequence of lists of size T one can always choose a nonrepetitive sequence?

nonrepetitive sequences from lists

Real Question

What is the minimum T such that from any sequence of lists of size T one can always choose a nonrepetitive sequence?

nonrepetitive sequences from lists

Real Question

What is the minimum T such that from any sequence of lists of size T one can always choose a nonrepetitive sequence?

$$3 \leqslant T \leqslant$$

nonrepetitive sequences from lists

Real Question

What is the minimum T such that from any sequence of lists of size T one can always choose a nonrepetitive sequence?

64 Alon, Grytczuk, Hałuszczak, Riordan 2002

$3 \leqslant T \leqslant$

special case of the result on nonrepetitive colorings

nonrepetitive sequences from lists

Real Question

What is the minimum T such that from any sequence of lists of size T one can always choose a nonrepetitive sequence?

	64	Alon, Grytczuk, Hałuszczak, Riordan 2002
$3 \leqslant T \leqslant$	7	Lovász Local Lemma

nonrepetitive sequences from lists

Real Question

What is the minimum T such that from any sequence of lists of size T one can always choose a nonrepetitive sequence?

	64	Alon, Grytczuk, Hałuszczak, Riordan 2002
$3 \leqslant T \leqslant$	7	Lovász Local Lemma
	4	Grytczuk, Przybyło, Zhu 2010

using the lefthanded local lemma by Pegden (2010)

nonrepetitive sequences from lists

Real Question

What is the minimum T such that from any sequence of lists of size T one can always choose a nonrepetitive sequence?

	64	Alon, Grytczuk, Hałuszczak, Riordan 2002
$3 \leqslant T \leqslant$	7	Lovász Local Lemma
	4	Grytczuk, Przybyło, Zhu 2010

Answer: 3 or 4.

Moser Tardos proof of LLL

- $v_1, \ldots, v_m$ set of independent random variables,

Moser Tardos proof of LLL

- $v_1, \ldots, v_m$ set of independent random variables,
- Events from $\mathcal{A}$ are determined by these variables.

Moser Tardos proof of LLL

- $v_1, \ldots, v_m$ set of independent random variables,
- Events from $\mathcal{A}$ are determined by these variables.
- $vbl(A)$ - set of variables on which A depends.

Moser Tardos proof of LLL

- $v_1, \ldots, v_m$ set of independent random variables,
- Events from $\mathcal{A}$ are determined by these variables.
- $vbl(A)$ - set of variables on which A depends.

Algorithm

for $i := 1$ **to** m
    $v_i :=$ random evaluation
while some $A \in \mathcal{A}$ holds
    for each $v \in vbl(A)$
        $v :=$ new random evaluation

Moser Tardos proof of LLL

- $v_1, \ldots, v_m$ set of independent random variables,
- Events from $\mathcal{A}$ are determined by these variables.
- $vbl(A)$ - set of variables on which A depends.

Algorithm

for $i := 1$ **to** m
    $v_i :=$ random evaluation
while some $A \in \mathcal{A}$ holds
    for each $v \in vbl(A)$
        $v :=$ new random evaluation

Theorem (Moser, Tardos, 2010)

The expected number of resamples after which the algorithm finds an instance avoiding all events from $\mathcal{A}$ is at most

$$\sum_{A \in \mathcal{A}} \frac{x(A)}{1 - x(A)}$$

Nonrepetitive sequences

Nonrepetitive sequences

Algorithm (Moser Tardos style)

$i := 1$
while $i \leqslant n$ **do**
  $s_i :=$ random symbol
  if $s_1, \ldots, s_i$ is nonrepetitive **then**
    $i := i + 1$
  else
    there is exactly one repetition, say of size h
    $i := i - h + 1$

Nonrepetitive sequences

Algorithm (Moser Tardos style)

$i := 1$
while $i \leqslant n$ **do**
  $s_i :=$ random symbol
  if $s_1, \ldots, s_i$ is nonrepetitive **then**
    $i := i + 1$
  else
    there is exactly one repetition, say of size h
    $i := i - h + 1$

Example of first few iterations (alphabet $\{a, b, c\}$)

ab

Nonrepetitive sequences

Algorithm (Moser Tardos style)

$i := 1$
while $i \leqslant n$ **do**
  $s_i :=$ random symbol
  if $s_1, \ldots, s_i$ is nonrepetitive **then**
    $i := i + 1$
  else
    there is exactly one repetition, say of size h
    $i := i - h + 1$

Example of first few iterations (alphabet $\{a, b, c\}$)

aba

Nonrepetitive sequences

Algorithm (Moser Tardos style)

$i := 1$
while $i \leqslant n$ **do**
  $s_i :=$ random symbol
  if $s_1, \ldots, s_i$ is nonrepetitive **then**
    $i := i + 1$
  else
    there is exactly one repetition, say of size h
    $i := i - h + 1$

Example of first few iterations (alphabet $\{a, b, c\}$)
abab

Nonrepetitive sequences

Algorithm (Moser Tardos style)

$i := 1$
while $i \leqslant n$ **do**
  $s_i :=$ random symbol
  if $s_1, \ldots, s_i$ is nonrepetitive **then**
    $i := i + 1$
  else
    there is exactly one repetition, say of size h
    $i := i - h + 1$

Example of first few iterations (alphabet $\{a, b, c\}$)

ab

Nonrepetitive sequences

Algorithm (Moser Tardos style)

$i := 1$
while $i \leqslant n$ **do**
  $s_i :=$ random symbol
  if $s_1, \ldots, s_i$ is nonrepetitive **then**
    $i := i + 1$
  else
    there is exactly one repetition, say of size h
    $i := i - h + 1$

Example of first few iterations (alphabet $\{a, b, c\}$)

aba

Nonrepetitive sequences

Algorithm (Moser Tardos style)

$i := 1$
while $i \leqslant n$ **do**
  $s_i :=$ random symbol
  if $s_1, \ldots, s_i$ is nonrepetitive **then**
    $i := i + 1$
  else
    there is exactly one repetition, say of size h
    $i := i - h + 1$

Example of first few iterations (alphabet $\{a, b, c\}$)

abac

Nonrepetitive sequences

Algorithm (Moser Tardos style)

$i := 1$
while $i \leqslant n$ **do**
  $s_i :=$ random symbol
  if $s_1, \ldots, s_i$ is nonrepetitive **then**
    $i := i + 1$
  else
    there is exactly one repetition, say of size h
    $i := i - h + 1$

Example of first few iterations (alphabet $\{a, b, c\}$)

abacc

Nonrepetitive sequences

Algorithm (Moser Tardos style)

$i := 1$
while $i \leqslant n$ **do**
  $s_i :=$ random symbol
  if $s_1, \ldots, s_i$ is nonrepetitive **then**
    $i := i + 1$
  else
    there is exactly one repetition, say of size h
    $i := i - h + 1$

Example of first few iterations (alphabet $\{a, b, c\}$)

abac

Nonrepetitive sequences

Algorithm (Moser Tardos style)

$i := 1$
while $i \leqslant n$ **do**
  $s_i :=$ random symbol
  if $s_1, \ldots, s_i$ is nonrepetitive **then**
    $i := i + 1$
  else
    there is exactly one repetition, say of size h
    $i := i - h + 1$

Example of first few iterations (alphabet $\{a, b, c\}$)

abacb

Nonrepetitive sequences

Algorithm (Moser Tardos style)

$i := 1$
while $i \leqslant n$ **do**
  $s_i :=$ random symbol
  if $s_1, \ldots, s_i$ is nonrepetitive **then**
    $i := i + 1$
  else
    there is exactly one repetition, say of size h
    $i := i - h + 1$

Example of first few iterations (alphabet $\{a, b, c\}$)

abacba

Nonrepetitive sequences

Algorithm (Moser Tardos style)

$i := 1$
while $i \leqslant n$ **do**
  $s_i :=$ random symbol
  if $s_1, \ldots, s_i$ is nonrepetitive **then**
    $i := i + 1$
  else
    there is exactly one repetition, say of size h
    $i := i - h + 1$

Example of first few iterations (alphabet $\{a, b, c\}$)

abacbac

Nonrepetitive sequences

Algorithm (Moser Tardos style)

$i := 1$
while $i \leqslant n$ **do**
  $s_i :=$ random symbol
  if $s_1, \ldots, s_i$ is nonrepetitive **then**
    $i := i + 1$
  else
    there is exactly one repetition, say of size h
    $i := i - h + 1$

Example of first few iterations (alphabet $\{a, b, c\}$)

abac

Nonrepetitive sequences

Algorithm (Moser Tardos style)

$i := 1$
while $i \leqslant n$ **do**
  $s_i :=$ random symbol
  if $s_1, \ldots, s_i$ is nonrepetitive **then**
    $i := i + 1$
  else
    there is exactly one repetition, say of size h
    $i := i - h + 1$

Example of first few iterations (alphabet $\{a, b, c\}$)

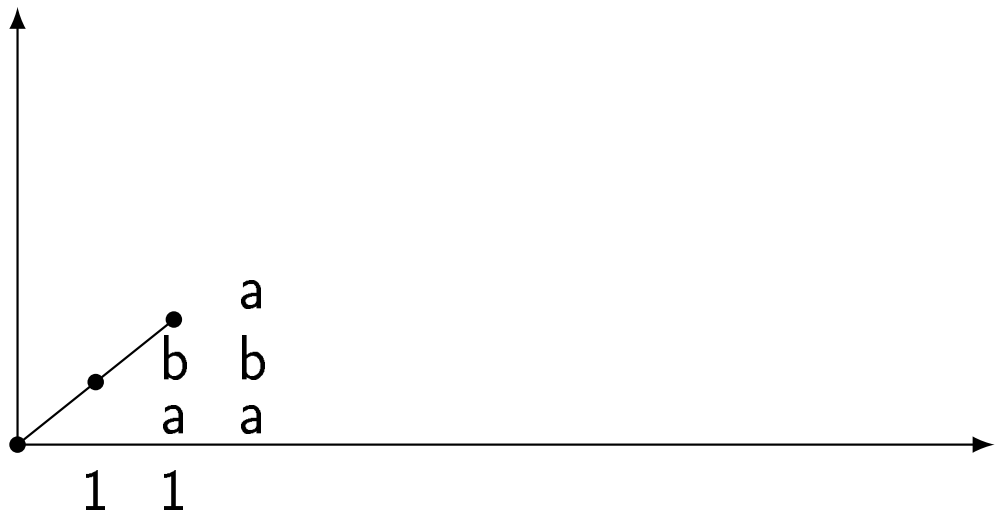
abacb...

log of Algorithm run (like ship's log)

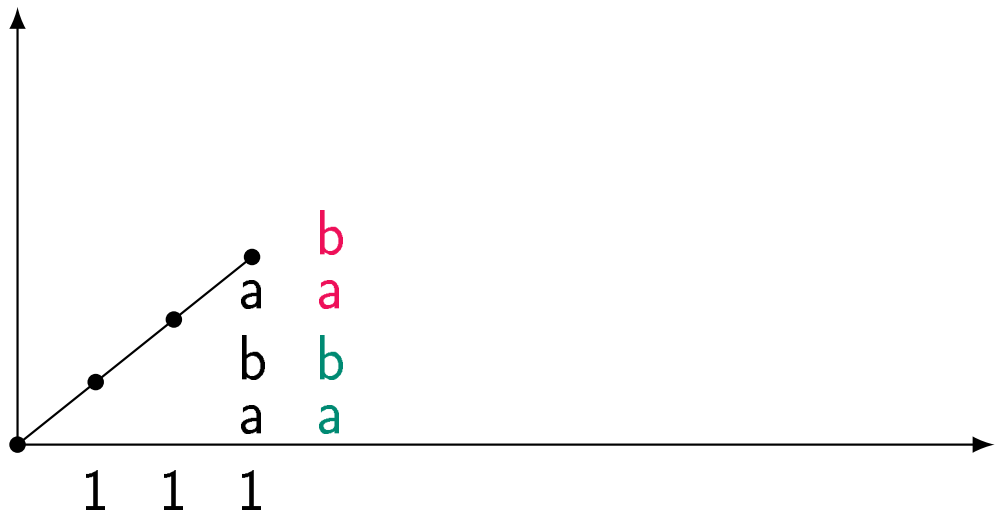
log of Algorithm run (like ship's log)

log of Algorithm run (like ship's log)

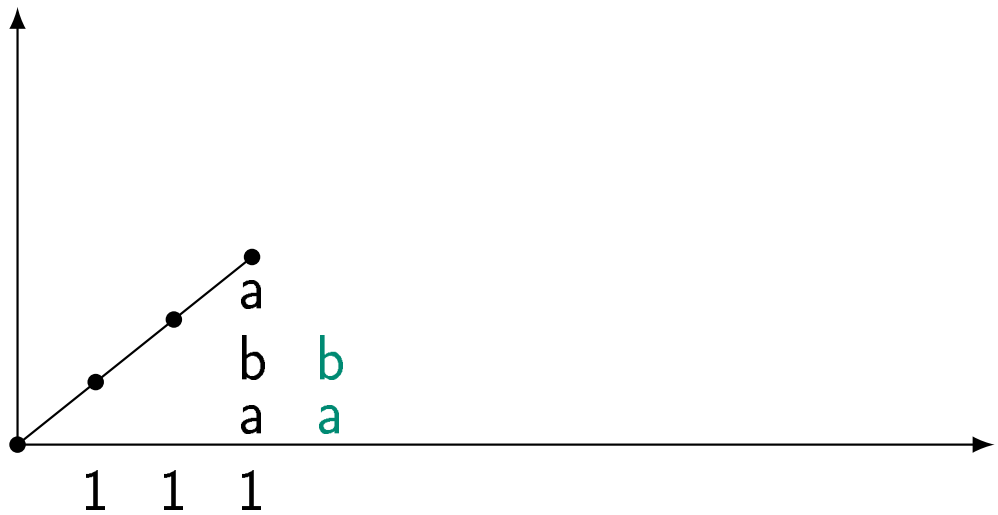
log of Algorithm run (like ship's log)



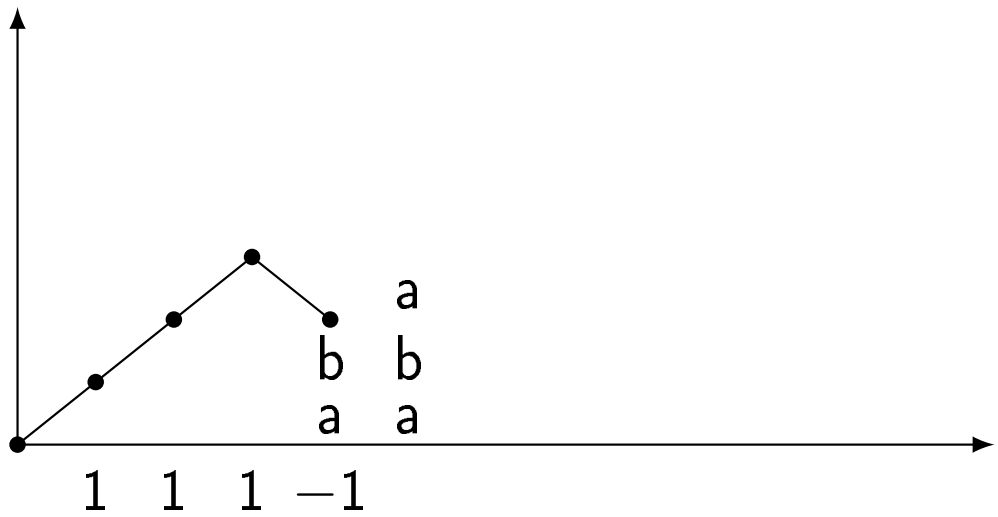
log of Algorithm run (like ship's log)



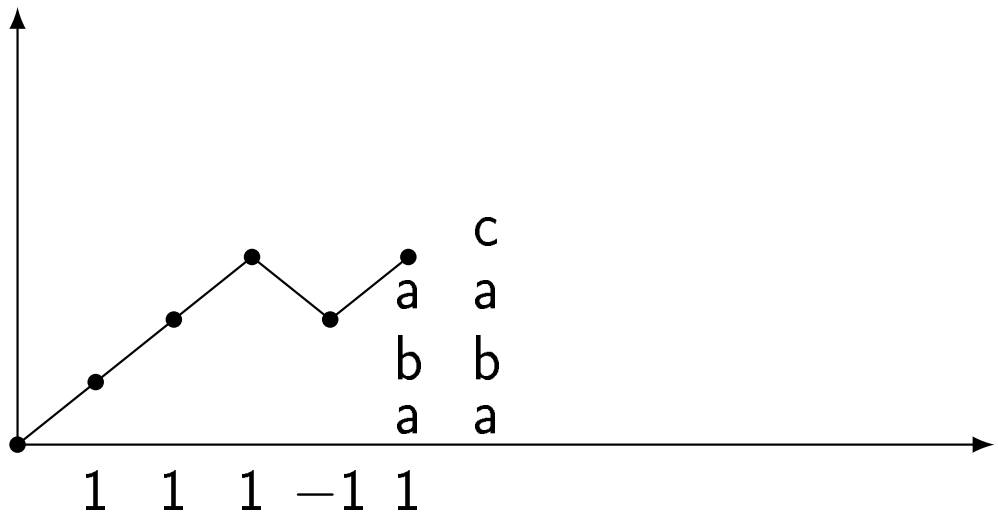
log of Algorithm run (like ship's log)



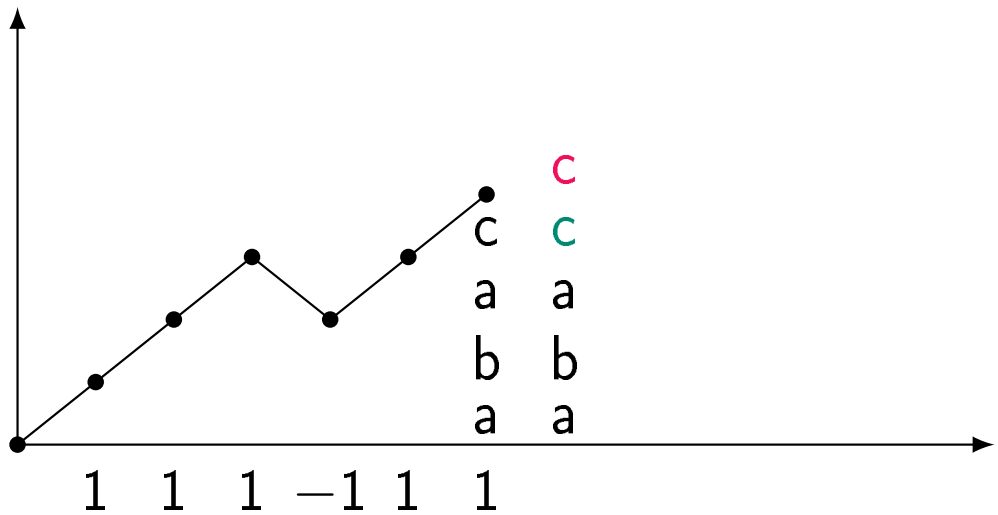
log of Algorithm run (like ship's log)



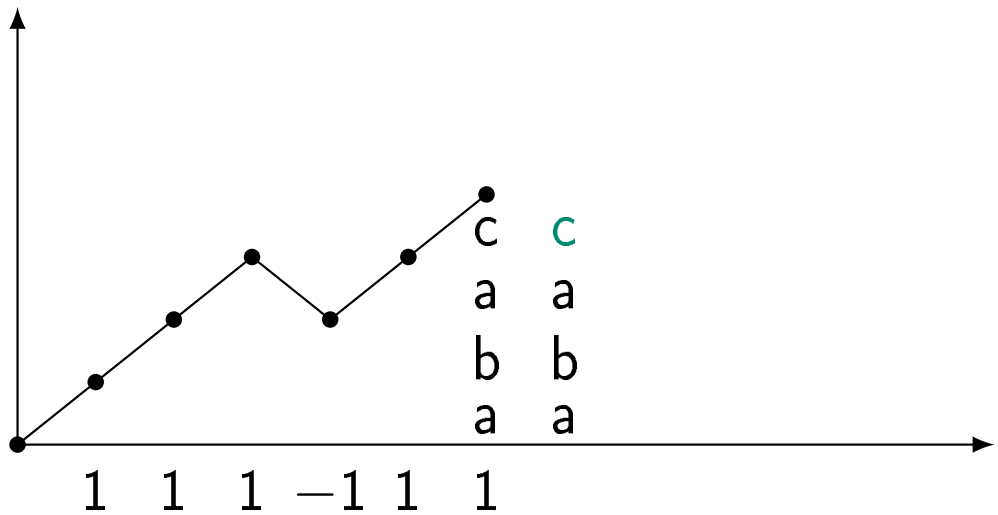
log of Algorithm run (like ship's log)



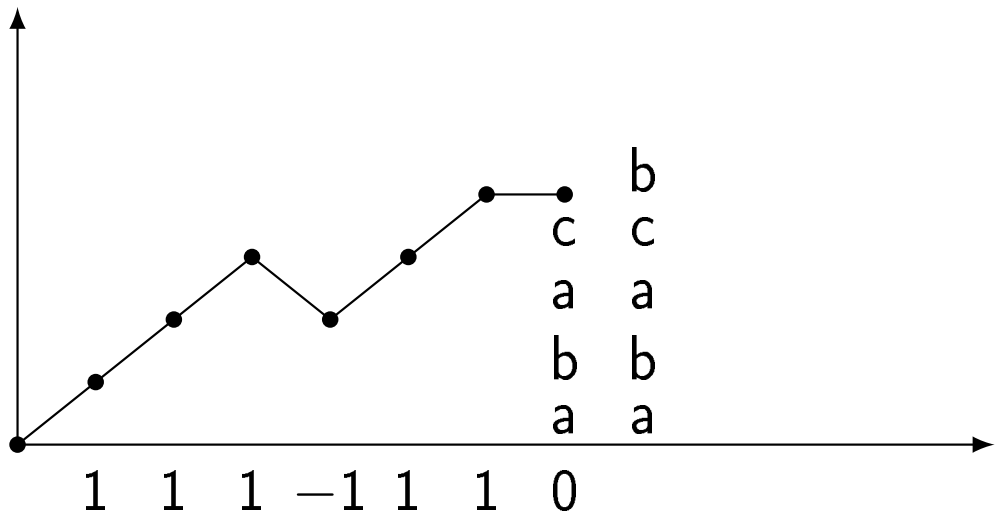
log of Algorithm run (like ship's log)



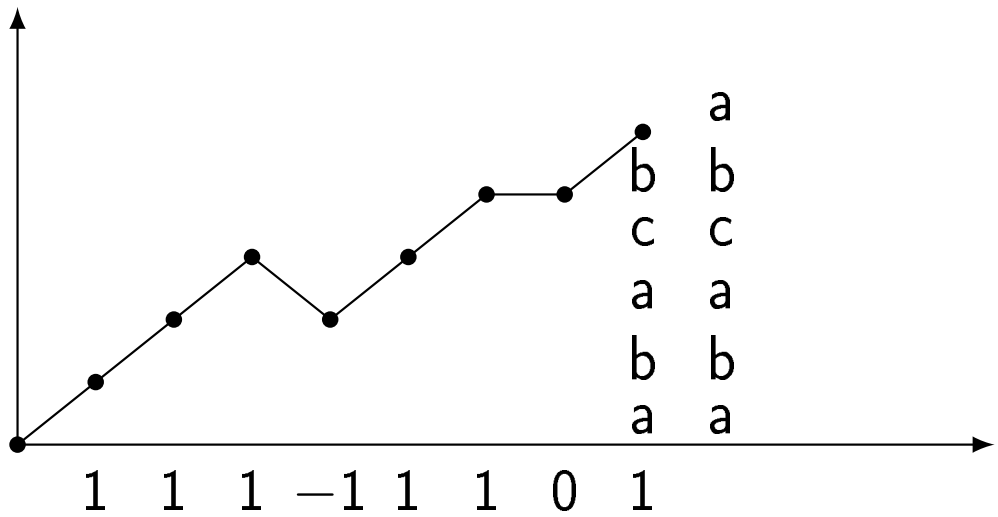
log of Algorithm run (like ship's log)



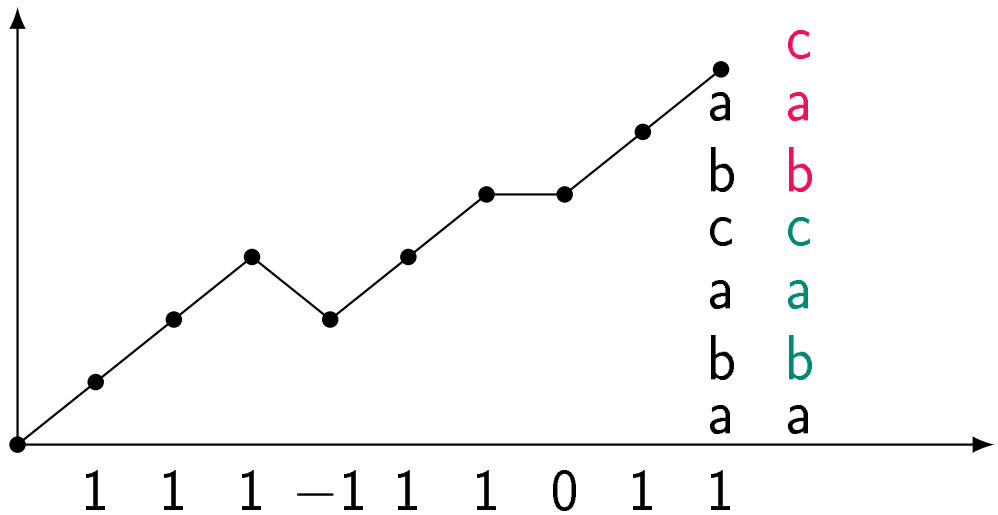
log of Algorithm run (like ship's log)



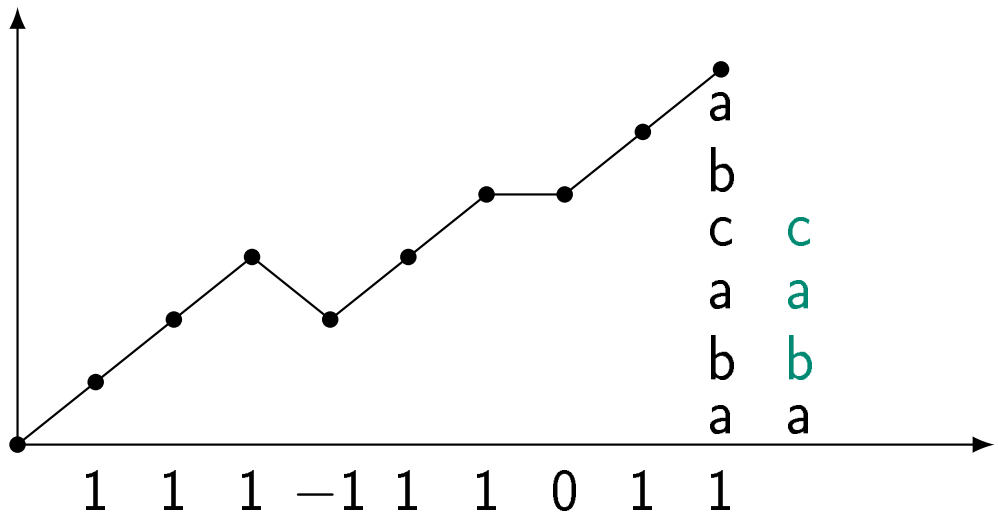
log of Algorithm run (like ship's log)



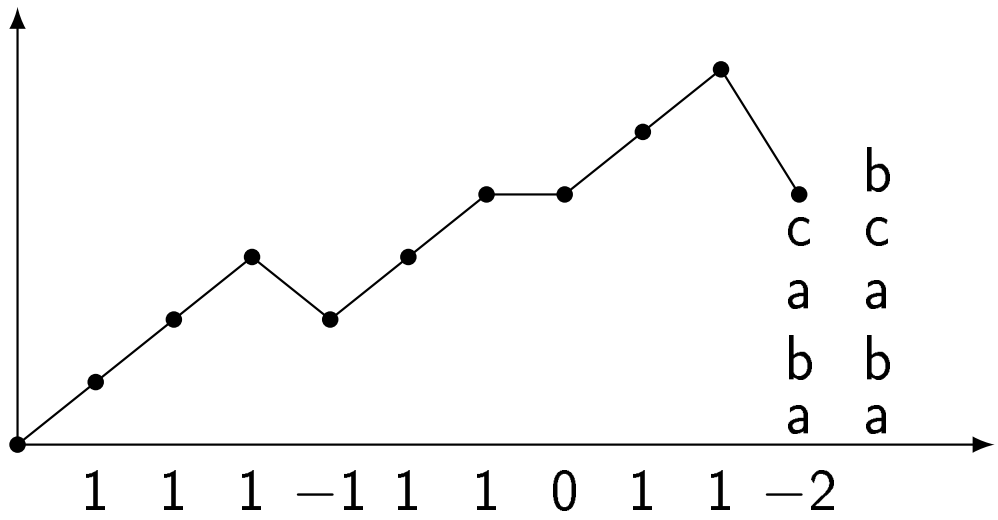
log of Algorithm run (like ship's log)



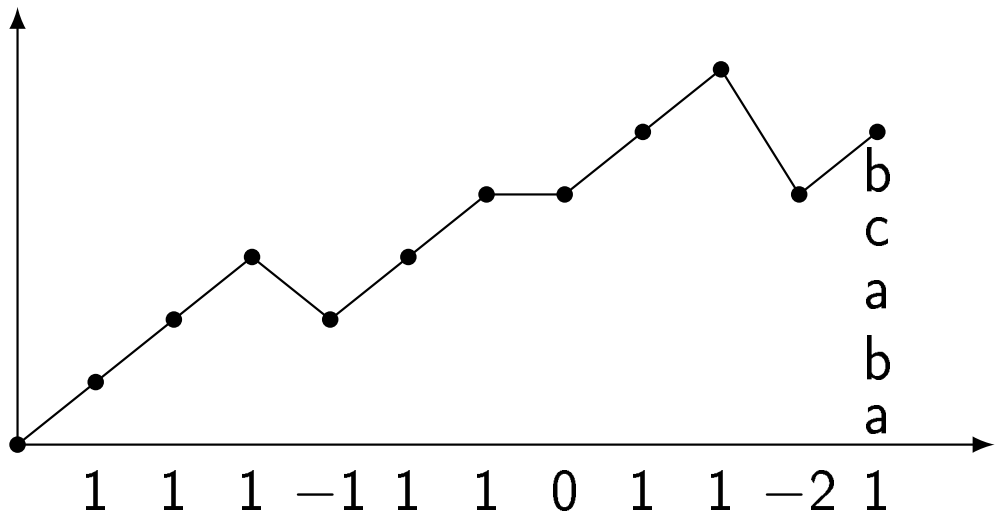
log of Algorithm run (like ship's log)



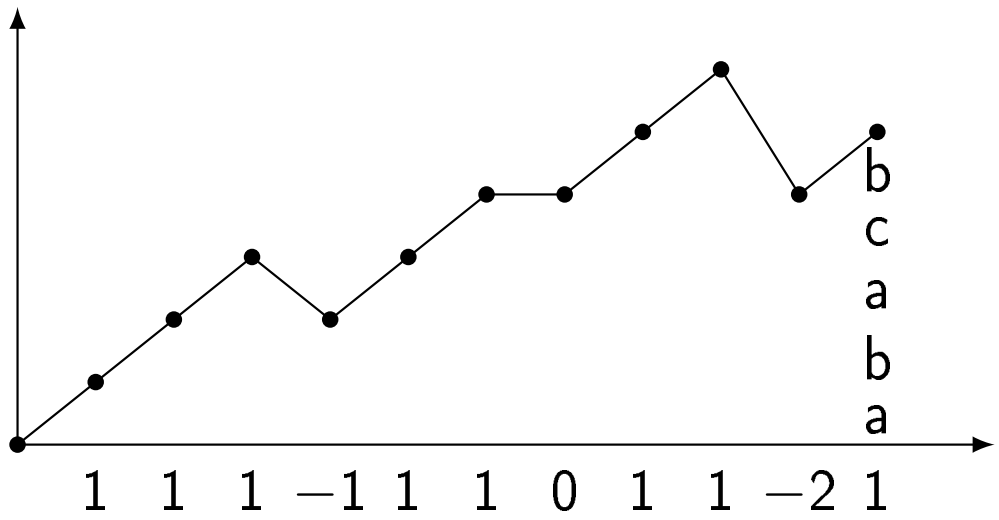
log of Algorithm run (like ship's log)



log of Algorithm run (like ship's log)



log of Algorithm run (like ship's log)



log

- $(1, 1, 1, -1, 1, 1, 0, 1, 1, -2, 1)$ – sequence of differences
- abacb – final word

logs vs random strings

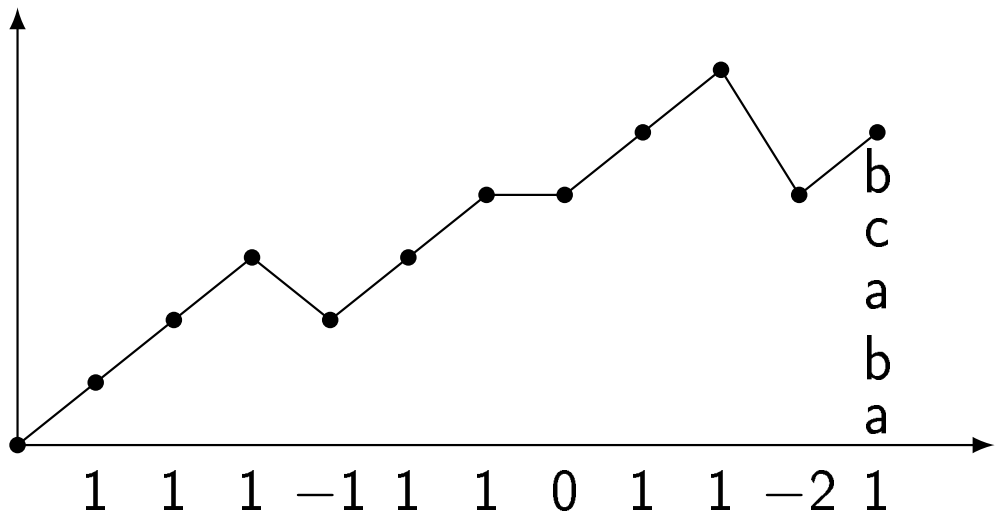
Observation

Log encodes sequence of results of random experiments $r_1, \ldots, r_n$ in a lossless fashion.

logs vs random strings

Observation

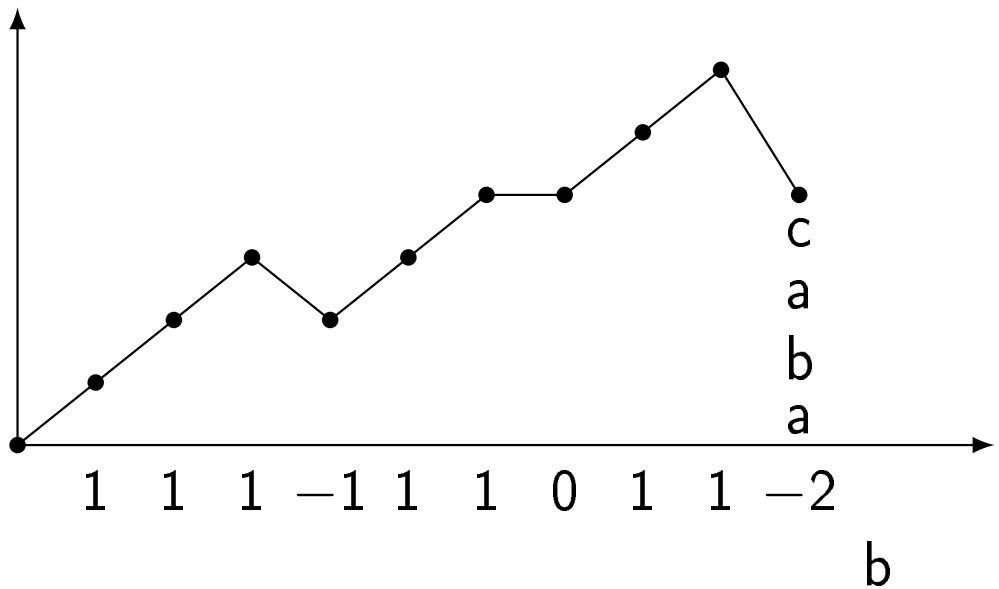
Log encodes sequence of results of random experiments $r_1, \ldots, r_n$ in a lossless fashion.



logs vs random strings

Observation

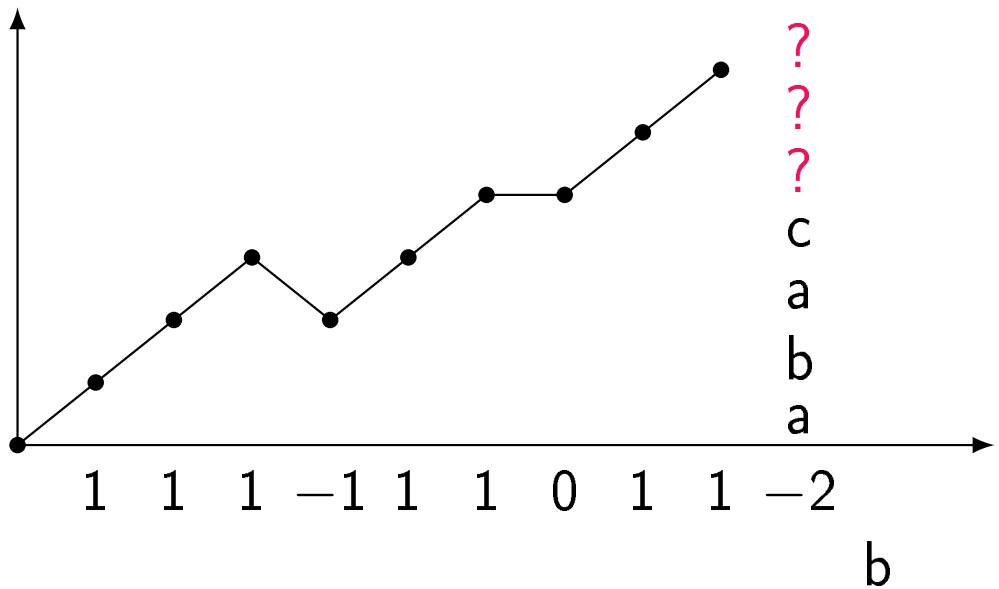
Log encodes sequence of results of random experiments $r_1, \ldots, r_n$ in a lossless fashion.



logs vs random strings

Observation

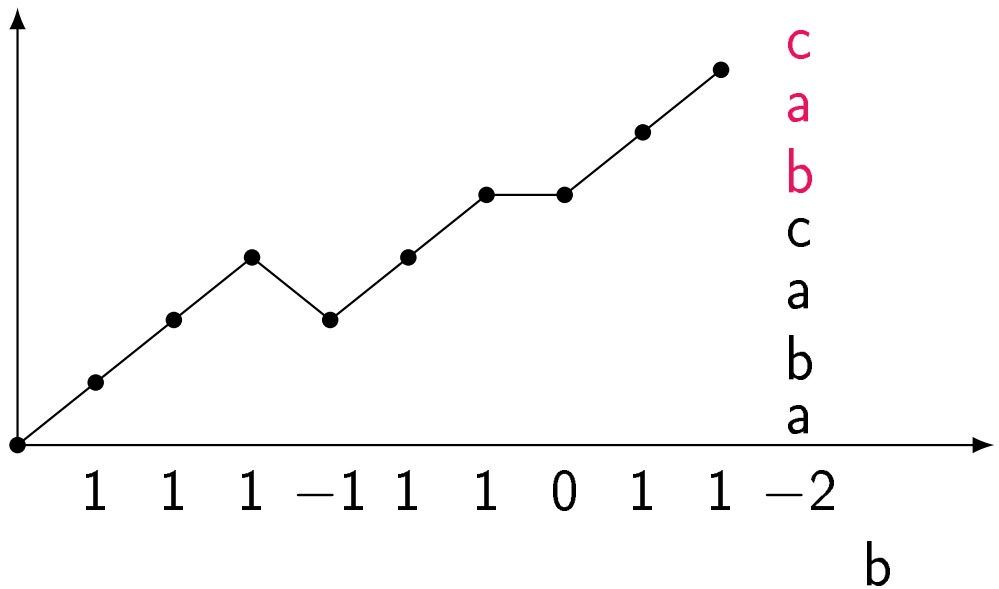
Log encodes sequence of results of random experiments $r_1, \ldots, r_n$ in a lossless fashion.



logs vs random strings

Observation

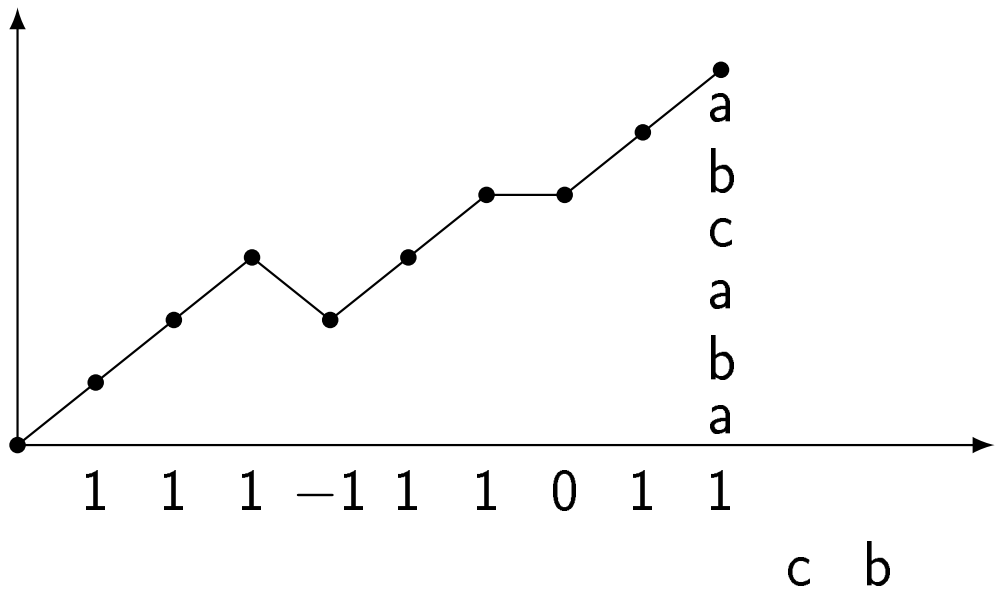
Log encodes sequence of results of random experiments $r_1, \ldots, r_n$ in a lossless fashion.



logs vs random strings

Observation

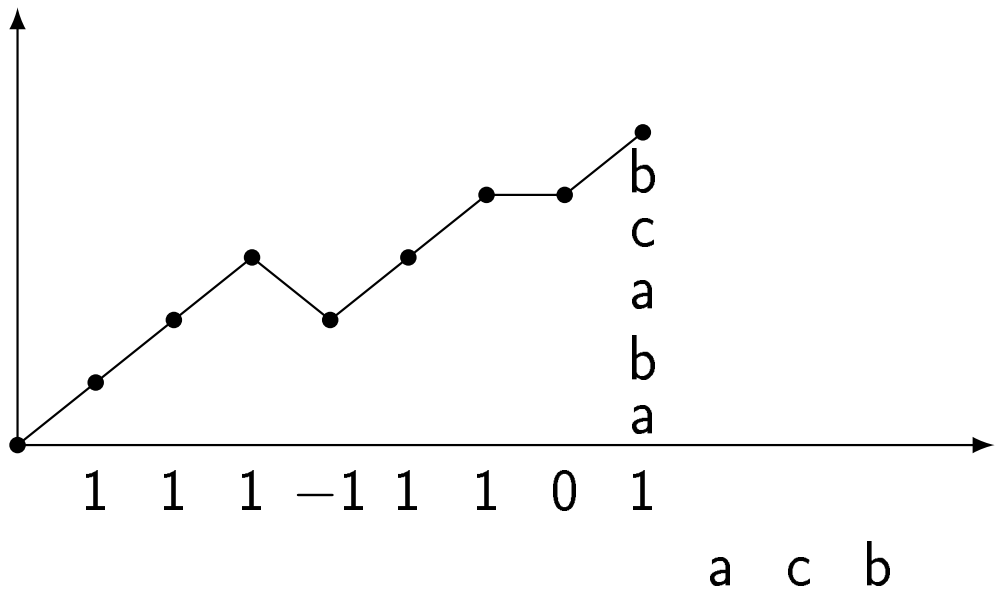
Log encodes sequence of results of random experiments $r_1, \ldots, r_n$ in a lossless fashion.



logs vs random strings

Observation

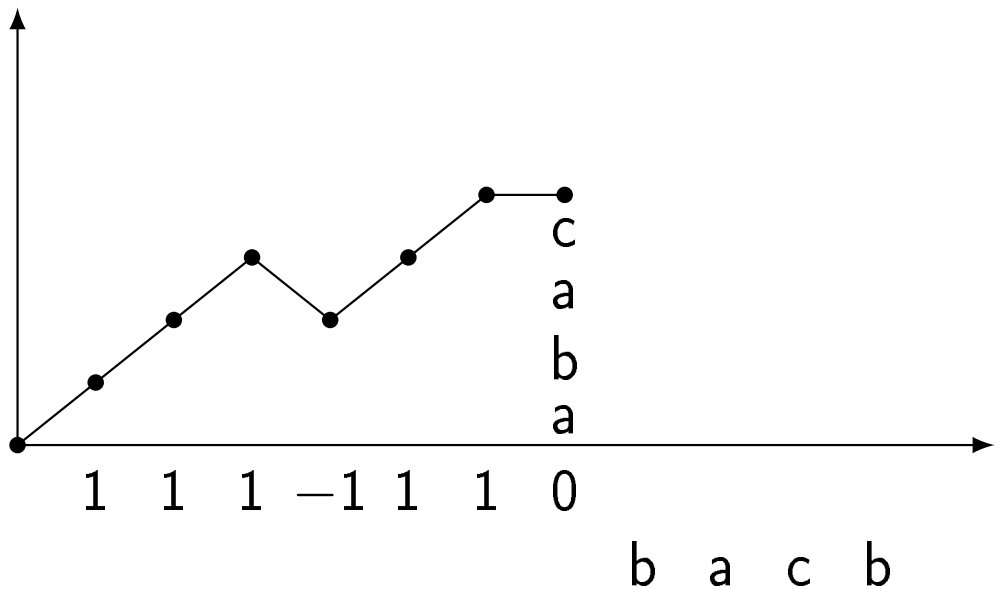
Log encodes sequence of results of random experiments $r_1, \ldots, r_n$ in a lossless fashion.



logs vs random strings

Observation

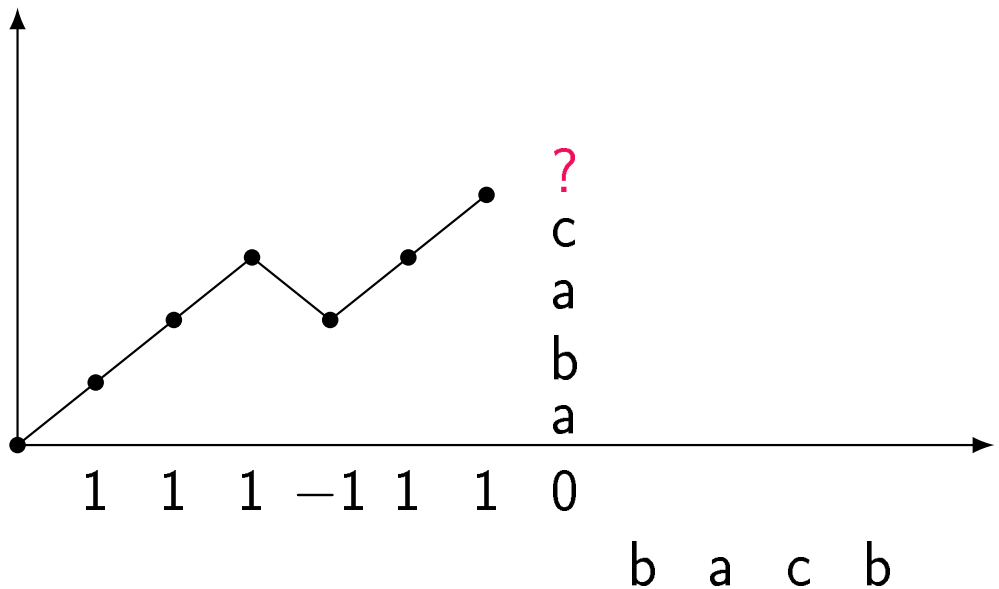
Log encodes sequence of results of random experiments $r_1, \ldots, r_n$ in a lossless fashion.



logs vs random strings

Observation

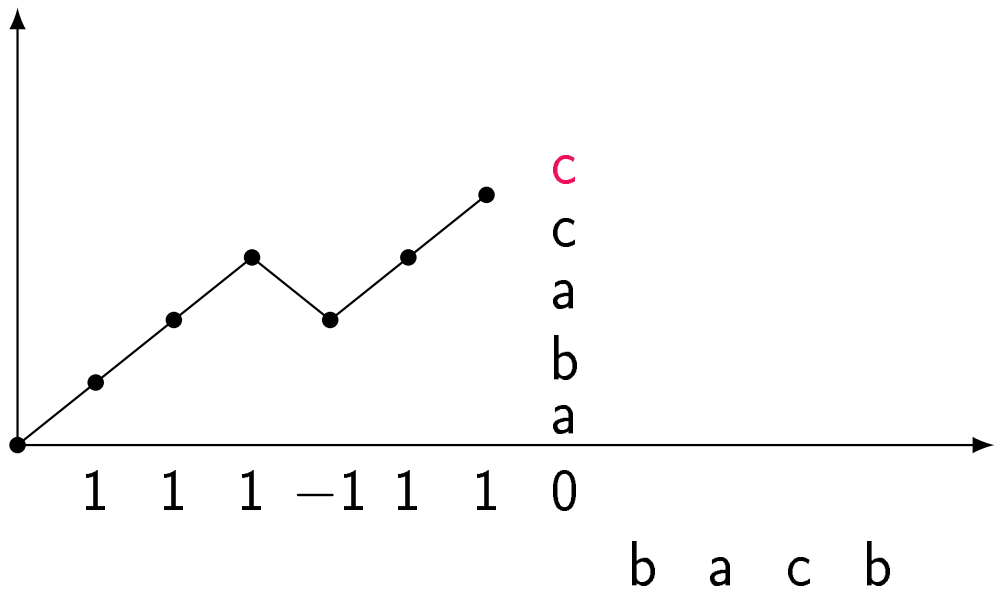
Log encodes sequence of results of random experiments $r_1, \ldots, r_n$ in a lossless fashion.



logs vs random strings

Observation

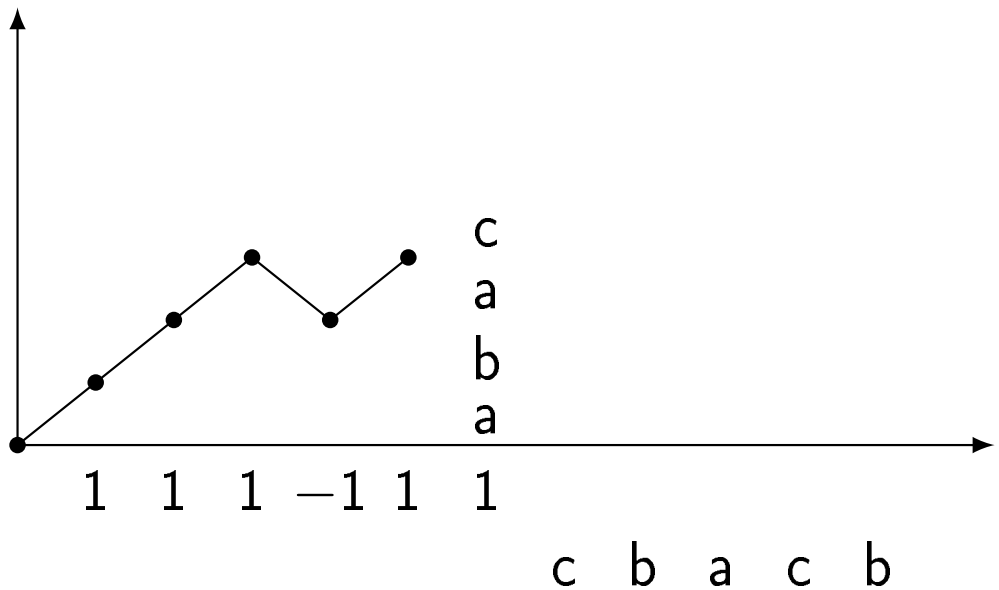
Log encodes sequence of results of random experiments $r_1, \ldots, r_n$ in a lossless fashion.



logs vs random strings

Observation

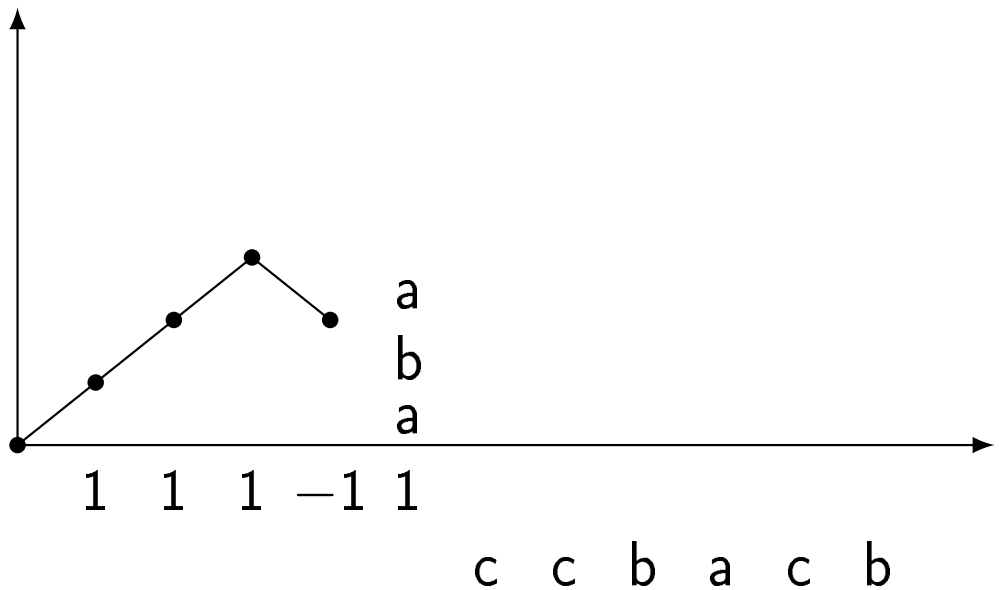
Log encodes sequence of results of random experiments $r_1, \ldots, r_n$ in a lossless fashion.



logs vs random strings

Observation

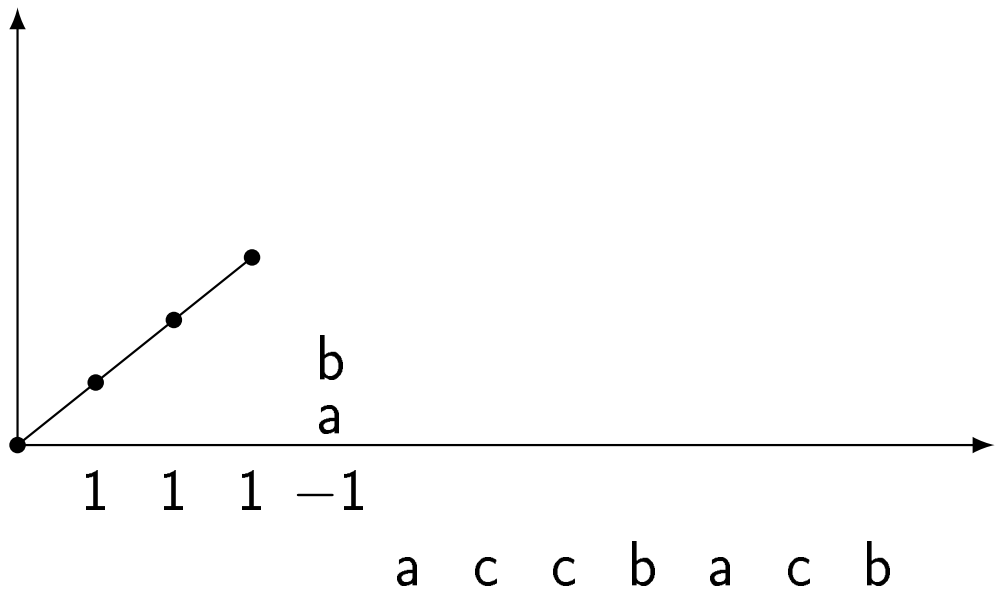
Log encodes sequence of results of random experiments $r_1, \ldots, r_n$ in a lossless fashion.



logs vs random strings

Observation

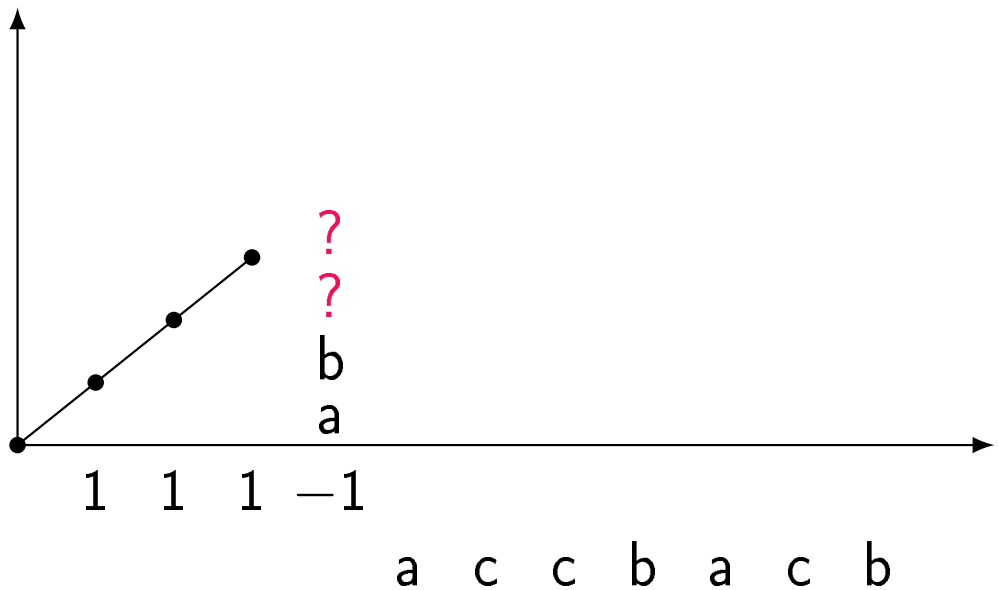
Log encodes sequence of results of random experiments $r_1, \ldots, r_n$ in a lossless fashion.



logs vs random strings

Observation

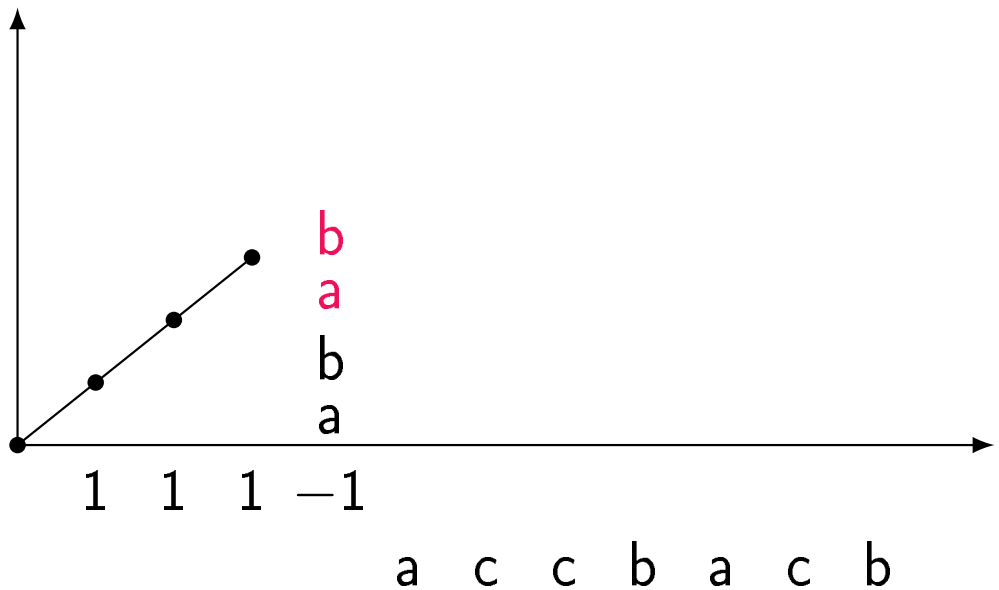
Log encodes sequence of results of random experiments $r_1, \ldots, r_n$ in a lossless fashion.



logs vs random strings

Observation

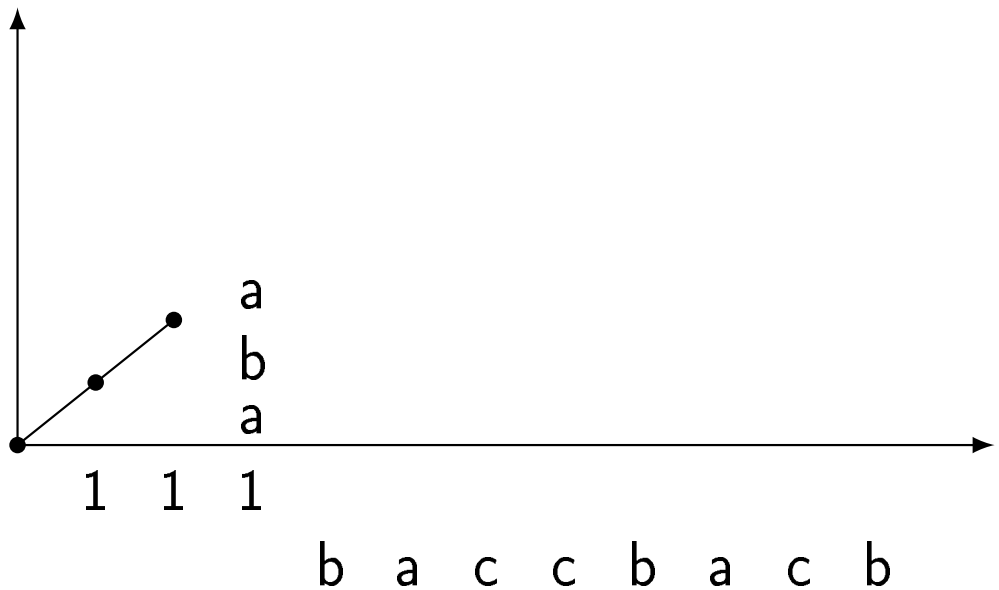
Log encodes sequence of results of random experiments $r_1, \ldots, r_n$ in a lossless fashion.



logs vs random strings

Observation

Log encodes sequence of results of random experiments $r_1, \ldots, r_n$ in a lossless fashion.



logs vs random strings

Observation

Log encodes sequence of results of random experiments $r_1, \ldots, r_n$ in a lossless fashion.

logs vs random strings

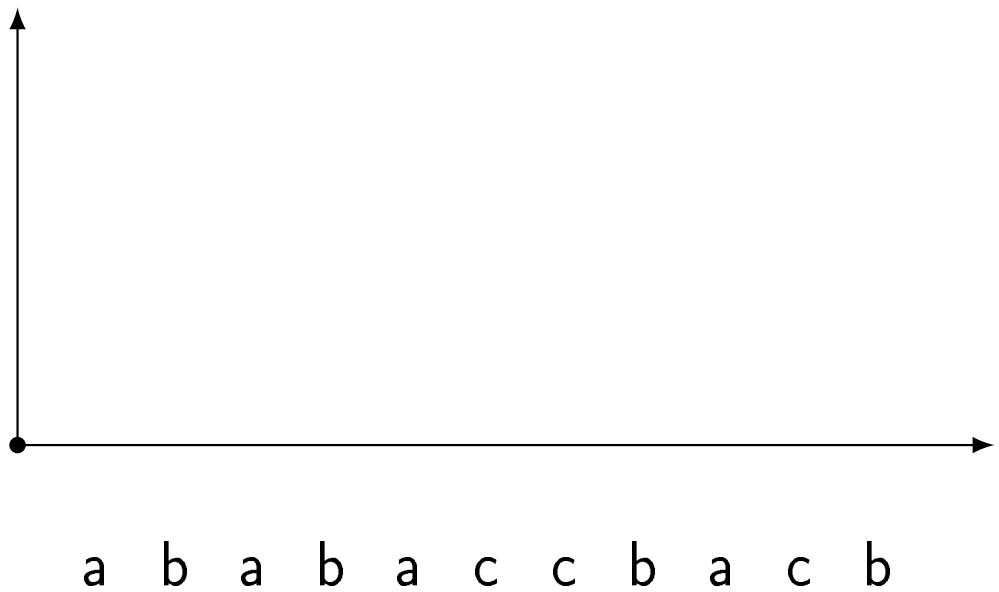
Observation

Log encodes sequence of results of random experiments $r_1, \ldots, r_n$ in a lossless fashion.

logs vs random strings

Observation

Log encodes sequence of results of random experiments $r_1, \ldots, r_n$ in a lossless fashion.



Counting logs.

We let the Algorithm work for n steps.

Counting logs.

We let the Algorithm work for n steps.

Logs which end with a word of length $< M$:

Counting logs.

We let the Algorithm work for n steps.

Logs which end with a word of length $< M$:

- number of lattice paths $= O(4^n/\sqrt{n})$

Counting logs.

We let the Algorithm work for n steps.

Logs which end with a word of length $< M$:

- number of lattice paths $= O(4^n/\sqrt{n})$
- number of final words $< k^M$

Counting logs.

We let the Algorithm work for n steps.

Logs which end with a word of length $< M$:

- number of lattice paths $= O(4^n/\sqrt{n}) = o(4^n)$
- number of final words $< k^M$

Counting logs.

We let the Algorithm work for n steps.

Logs which end with a word of length $< M$:

- number of lattice paths $= O(4^n/\sqrt{n}) = o(4^n)$
- number of final words $< k^M = O(1)$

Counting logs.

We let the Algorithm work for n steps.

Logs which end with a word of length $< M$:

- number of lattice paths $= O(4^n/\sqrt{n}) = o(4^n)$
- number of final words $< k^M = O(1)$

Number of such logs is $o(4^n)$.

Counting logs.

We let the Algorithm work for n steps.

Logs which end with a word of length $< M$:

- number of lattice paths $= O(4^n/\sqrt{n}) = o(4^n)$
- number of final words $< k^M = O(1)$

Number of such logs is $o(4^n)$.
Number of all sequences of results of random experiments is k^n.

Counting logs.

We let the Algorithm work for n steps.

Logs which end with a word of length $< M$:

- number of lattice paths $= O(4^n/\sqrt{n}) = o(4^n)$
- number of final words $< k^M = O(1)$

Number of such logs is $o(4^n)$.
Number of all sequences of results of random experiments is k^n.

Conclusion

For $k \geq 4$ and large n most logs end with word of length greater than M.

Avoiding cubes etc.

What is the minimum T such that from any sequence of lists of size T one can always choose a sequence avoiding pattern $\alpha^3 = \alpha\alpha\alpha$?

Avoiding cubes etc.

What is the minimum T such that from any sequence of lists of size T one can always choose a sequence avoiding pattern $\alpha^3 = \alpha\alpha\alpha$?

Same method with constrained set of allowed steps in the path.

Avoiding cubes etc.

What is the minimum T such that from any sequence of lists of size T one can always choose a sequence avoiding pattern $\alpha^3 = \alpha\alpha\alpha$?

Same method with constrained set of allowed steps in the path.

Pattern	steps	radius of conv.	T (upperbound)
α^2	$\{1, 0, -1, -2, -3, \ldots\}$	0.25	4

Avoiding cubes etc.

What is the minimum T such that from any sequence of lists of size T one can always choose a sequence avoiding pattern $\alpha^3 = \alpha\alpha\alpha$?

Same method with constrained set of allowed steps in the path.

Pattern	steps	radius of conv.	T (upperbound)
α^2	$\{1, 0, -1, -2, -3, \ldots\}$	0.25	4
α^3	$\{1, -1, -3, -5, \ldots\}$	0.385	3

Avoiding cubes etc.

What is the minimum T such that from any sequence of lists of size T one can always choose a sequence avoiding pattern $\alpha^3 = \alpha\alpha\alpha$?

Same method with constrained set of allowed steps in the path.

Pattern	steps	radius of conv.	T (upperbound)
α^2	$\{1, 0, -1, -2, -3, \ldots\}$	0.25	4
α^3	$\{1, -1, -3, -5, \ldots\}$	0.385	3
α^4	$\{1, -2, -5, -8, \ldots\}$	0.427	3

Avoiding cubes etc.

What is the minimum T such that from any sequence of lists of size T one can always choose a sequence avoiding pattern $\alpha^3 = \alpha\alpha\alpha$?

Same method with constrained set of allowed steps in the path.

Pattern	steps	radius of conv.	T (upperbound)
α^2	$\{1, 0, -1, -2, -3, \ldots\}$	0.25	4
α^3	$\{1, -1, -3, -5, \ldots\}$	0.385	3
α^4	$\{1, -2, -5, -8, \ldots\}$	0.427	3
α^5	$\{1, -3, -7, -11, \ldots\}$	0.535	2

Nonrepetitive game

- Ann and Ben build a sequence picking in turns consecutive symbols from k-element alphabet.
- Ben wins if repetition of length at least 2 is generated.

Nonrepetitive game

- Ann and Ben build a sequence picking in turns consecutive symbols from k-element alphabet.
- Ben wins if repetition of length at least 2 is generated.

What is the minimum value of k for which Ann can play indefinitely?

Nonrepetitive game

- Ann and Ben build a sequence picking in turns consecutive symbols from k-element alphabet.
- Ben wins if repetition of length at least 2 is generated.

What is the minimum value of k for which Ann can play indefinitely?

- LLL - ?

Nonrepetitive game

- Ann and Ben build a sequence picking in turns consecutive symbols from k-element alphabet.
- Ben wins if repetition of length at least 2 is generated.

What is the minimum value of k for which Ann can play indefinitely?

- LLL - ?
- lefthanded LLL - 37 (Pegden 2010)

Nonrepetitive game

- Ann and Ben build a sequence picking in turns consecutive symbols from k-element alphabet.
- Ben wins if repetition of length at least 2 is generated.

What is the minimum value of k for which Ann can play indefinitely?

- LLL - ?
- lefthanded LLL - 37 (Pegden 2010)
- double counting - 6 (Grytczuk, JK, Micek 2011)

Nonrepetitive game

- Ann and Ben build a sequence picking in turns consecutive symbols from k-element alphabet.
- Ben wins if repetition of length at least 2 is generated.

What is the minimum value of k for which Ann can play indefinitely?

- LLL - ?
- lefthanded LLL - 37 (Pegden 2010)
- double counting - 6 (Grytczuk, JK, Micek 2011)

(nonconstructive)

Nonrepetitive game

- Ann and Ben build a sequence picking in turns consecutive symbols from k-element alphabet.
- Ben wins if repetition of length at least 2 is generated.

What is the minimum value of k for which Ann can play indefinitely?

- LLL - ?
- lefthanded LLL - 37 (Pegden 2010)
- double counting - 6 (Grytczuk, JK, Micek 2011)

(nonconstructive)

Thank you for your attention.