Subsequence Combinatorics and Applications

Sven Rahmann

Algorithms and Statistics for Systems Biology Group
Genome Informatics, Department of Technology, Bielefeld University, Germany

CPM'06, Barcelona, 05.07.2006

Subsequences

Fundamentals

- Σ: finite alphabet
- $s = (s_1, \ldots, s_n) \in \Sigma^n$: a string (sequence) of length n

Subsequences

Fundamentals

- Σ: finite alphabet
- $s = (s_1, \ldots, s_n) \in \Sigma^n$: a string (sequence) of length n

Subsequences

- Each $I \subset \{1, \ldots, n\}$ defines a *subsequence* of s:
  $I = \{i_1, \ldots, i_k\}$ with $i_1 < \cdots < i_k \implies s_I := s_{i_1} s_{i_2} \ldots s_{i_k}$.
- Notation: $t \lhd s \iff t$ is a subsequence of s.

Subsequences

Fundamentals

- Σ: finite alphabet
- $s = (s_1, \ldots, s_n) \in \Sigma^n$: a string (sequence) of length n

Subsequences

- Each $I \subset \{1, \ldots, n\}$ defines a *subsequence* of s:
  $I = \{i_1, \ldots, i_k\}$ with $i_1 < \cdots < i_k \implies s_I := s_{i_1} s_{i_2} \ldots s_{i_k}$.
- Notation: $t \lhd s \iff t$ is a subsequence of s.
- There are up to 2^n subsequences, $\binom{n}{k}$ of length k.
- Not all of these need to be different!

Subsequences

Fundamentals

- Σ: finite alphabet
- $s = (s_1, \ldots, s_n) \in \Sigma^n$: a string (sequence) of length n

Subsequences

- Each $I \subset \{1, \ldots, n\}$ defines a *subsequence* of s:
  $I = \{i_1, \ldots, i_k\}$ with $i_1 < \cdots < i_k \implies s_I := s_{i_1} s_{i_2} \ldots s_{i_k}$.
- Notation: $t \lhd s \iff t$ is a subsequence of s.
- There are up to 2^n subsequences, $\binom{n}{k}$ of length k.
- Not all of these need to be different!

First Question

How many distinct subsequences of length k exist in s?

Aside: Number of distinct substrings

Side Question

Number of distinct (contiguous) length-k substrings of s?

Aside: Number of distinct substrings

Side Question

Number of distinct (contiguous) length-k substrings of s?

Easy, given the lcp table of the suffix array of s:
Number is $n - k + 1 - |\{i : lcp[i] \geq k\}|$.

Aside: Number of distinct substrings

Side Question

Number of distinct (contiguous) length-k substrings of s?

Easy, given the lcp table of the suffix array of s:
Number is $n - k + 1 - |\{i : lcp[i] \geq k\}|$.

Can be computed in $O(n)$ operations.

Counting Distinct Subsequences Efficiently

General Notational Conventions

- S: a set of sequences
- C: cardinality of S
- W: weight of S,
  e.g., $W = C/|\Sigma|^m$ if all sequences in S have length m

Counting Distinct Subsequences Efficiently

General Notational Conventions

- S: a set of sequences
- C: cardinality of S
- W: weight of S,
  e.g., $W = C/|\Sigma|^m$ if all sequences in S have length m

Definitions

$$S_{m,j} := \{t \in \Sigma^m : t \lhd s_1 \dots s_j\}, \qquad C_{m,j} := |S_{m,j}|,$$

Counting Distinct Subsequences Efficiently

General Notational Conventions

- S: a set of sequences
- C: cardinality of S
- W: weight of S,
  e.g., $W = C/|\Sigma|^m$ if all sequences in S have length m

Definitions

$S_{m,j} := \{t \in \Sigma^m : t \lhd s_1 \dots s_j\}$, $C_{m,j} := |S_{m,j}|$,
$S_{m,j}[a] := \{t \in \Sigma^m : t \lhd s_1 \dots s_j \text{ and } t_m = a\}$, $C_{m,j}[a] := |S_{m,j}[a]|$.

Counting Distinct Subsequences Efficiently

General Notational Conventions

- S: a set of sequences
- C: cardinality of S
- W: weight of S,
  e.g., $W = C/|\Sigma|^m$ if all sequences in S have length m

Definitions

$S_{m,j} := \{t \in \Sigma^m : t \lhd s_1 \dots s_j\}$, $C_{m,j} := |S_{m,j}|$,
$S_{m,j}[a] := \{t \in \Sigma^m : t \lhd s_1 \dots s_j \text{ and } t_m = a\}$, $C_{m,j}[a] := |S_{m,j}[a]|$.

Goal

Compute $C_k(s) = C_{k,n} = \sum_{a \in \Sigma} C_{k,n}[a]$.

A Recurrence on Sets

- $S_{0,j} = \{\epsilon\}$ for all j.
- $S_{0,j}[a] = \{\}$ for all j and all $a \in \Sigma$.
- For $1 \leq m \leq j$,

$$S_{m,j}[a] = \begin{cases} S_{m,j-1}[a] & \text{if } s_j \neq a, \\ S_{m-1,j-1} \circ \{a\} & \text{if } s_j = a. \end{cases}$$

A Recurrence on Sets

- $S_{0,j} = \{\epsilon\}$ for all j.
- $S_{0,j}[a] = \{\}$ for all j and all $a \in \Sigma$.
- For $1 \leq m \leq j$,

$$S_{m,j}[a] = \begin{cases} S_{m,j-1}[a] & \text{if } s_j \neq a, \\ S_{m-1,j-1} \circ \{a\} & \text{if } s_j = a. \end{cases}$$

Proof

Case $s_j \neq a$: $S_{m,j-1}[a] \subset S_{m,j}[a]$ is trivial; thus prove $S_{m,j}[a] \subset S_{m,j-1}[a]$: Take $t \in S_{m,j}[a]$. Since $s_j \neq a$, already $t \in S_{m,j-1}[a]$.

s_1	$\cdots$	s_{j-1}	s_j
	t		b

A Recurrence on Sets

- $S_{0,j} = \{\epsilon\}$ for all j.
- $S_{0,j}[a] = \{\}$ for all j and all $a \in \Sigma$.
- For $1 \leq m \leq j$,

$$S_{m,j}[a] = \begin{cases} S_{m,j-1}[a] & \text{if } s_j \neq a, \\ S_{m-1,j-1} \circ \{a\} & \text{if } s_j = a. \end{cases}$$

Proof

Case $s_j = a$: Appending a to each $t \in S_{m-1,j-1}$ yields a distinct string $ta \in S_{m,j}[a]$, thus $S_{m-1,j-1} \circ \{a\} \subset S_{m,j}[a]$. Conversely, every $s \in S_{m,j}[a]$ can be written as $s = ta$ with $t \in S_{m-1,j-1}$.

s_1	$\cdots$	s_{j-1}	s_j
	t		a

A Recurrence on Cardinalities

- $C_{0,0} = 1$ and $C_{0,0}[a] = 0$ for all $a \in \Sigma$.
- $C_{m,j} = 0$ if $m > j$.
- For $1 \leq m \leq j$,

$$C_{m,j}[a] = \begin{cases} C_{m,j-1}[a] & \text{if } s_j \neq a, \\ C_{m-1,j-1} & \text{if } s_j = a. \end{cases}$$

Bernoulli (i.i.d.) String Model

Consider an i.i.d. model:

- $\pi := (\pi_a)_{a \in \Sigma}$: given probability distribution on Σ
- $\mathbb{P}_k(t_1 \dots t_k) := \prod_{j=1}^{k} \pi_{t_j}$
- Then $\mathbb{P}_{k+1}(ta) = \mathbb{P}_k(t) \cdot \pi_a$ for $t \in \Sigma^k$ and $a \in \Sigma$.

Bernoulli (i.i.d.) String Model

Consider an i.i.d. model:

- $\pi := (\pi_a)_{a\in\Sigma}$: given probability distribution on Σ
- $\mathbb{P}_k(t_1 \dots t_k) := \prod_{j=1}^{k} \pi_{t_j}$
- Then $\mathbb{P}_{k+1}(ta) = \mathbb{P}_k(t) \cdot \pi_a$ for $t \in \Sigma^k$ and $a \in \Sigma$.

Define

- $W_k(s) := \mathbb{P}_k(S_k(s))$, covered fraction of Σ^k
- $W_{m,j} := \mathbb{P}_m(S_{m,j}) = \sum_{t\in S_{m,j}[a]} \mathbb{P}_m(t)$ $(m \le k,\, j \le |s| = n)$
- $W_{m,j}[a] := \mathbb{P}_m(S_{m,j}[a])$

Recurrence on Weights

- $W_{0,0} = 1$ and $W_{0,0}[a] = 0$ for all $a \in \Sigma$.
- $W_{m,j} = 0$ if $m > j$.
- For $1 \leq m \leq j$,

$$W_{m,j}[a] = \begin{cases} W_{m,j-1}[a] & \text{if } s_j \neq a, \\ W_{m-1,j-1} \cdot \pi_a & \text{if } s_j = a. \end{cases}$$

Recurrence on Weights

- $W_{0,0} = 1$ and $W_{0,0}[a] = 0$ for all $a \in \Sigma$.
- $W_{m,j} = 0$ if $m > j$.
- For $1 \leq m \leq j$,

$$W_{m,j}[a] = \begin{cases} W_{m,j-1}[a] & \text{if } s_j \neq a, \\ W_{m-1,j-1} \cdot \pi_a & \text{if } s_j = a. \end{cases}$$

Straightforward implementation: $O(kn|\Sigma|)$ operations.
Can be reduced to $O(k(n + |\Sigma|))$ operations.

Application: DNA Microarray Production

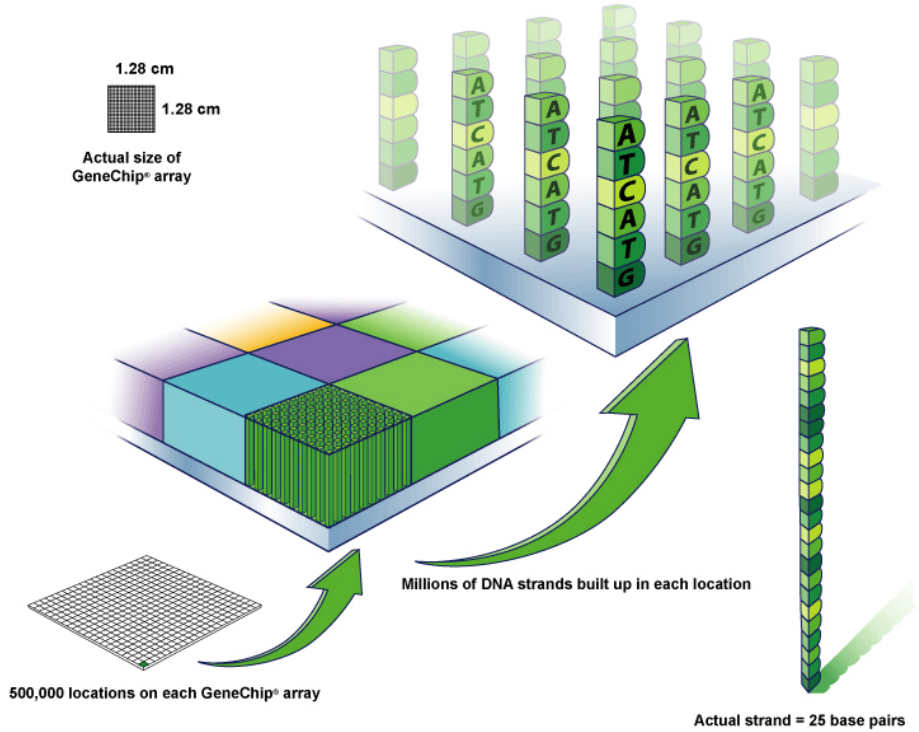
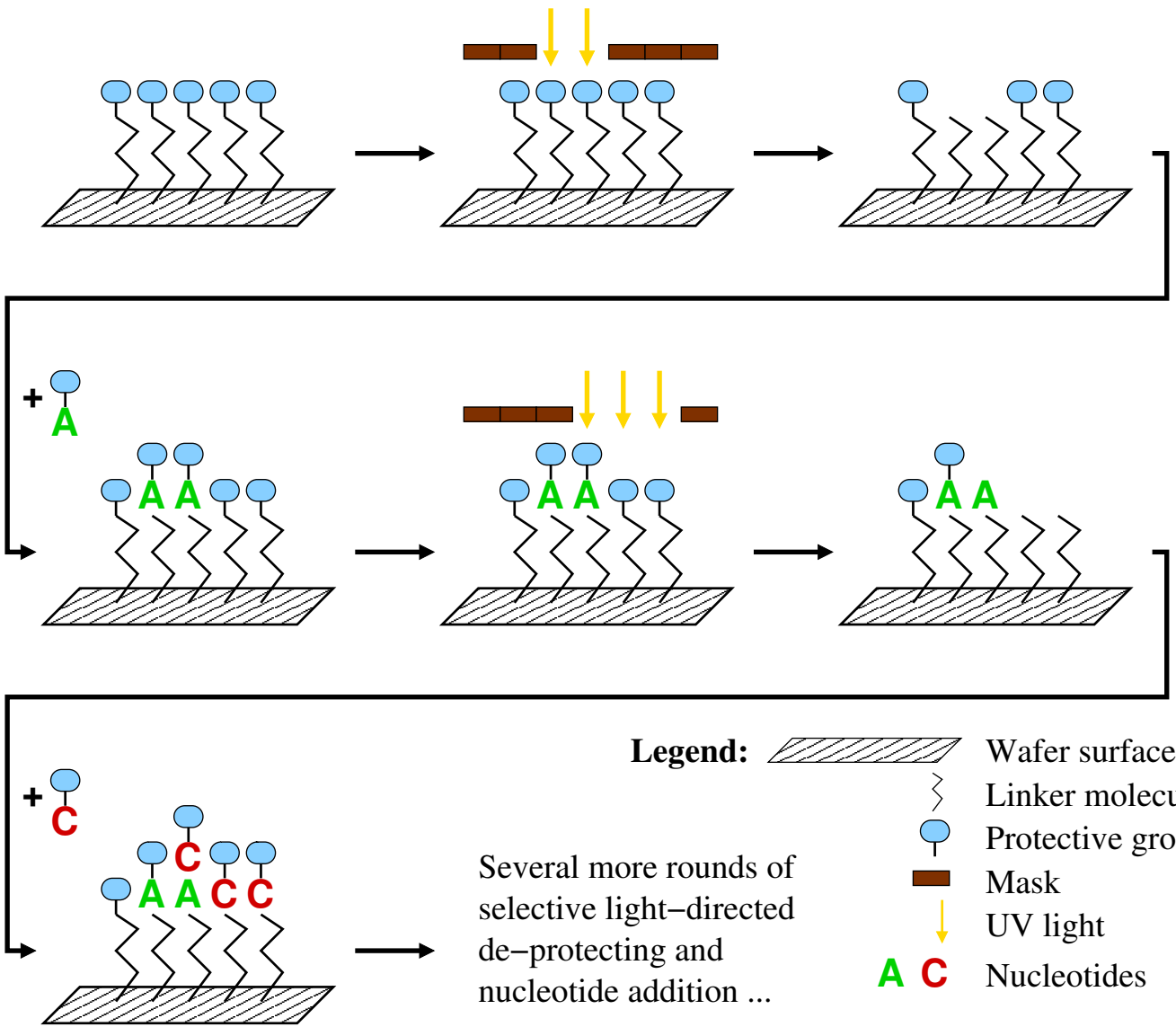


Image: Affymetrix, Inc.

- Each spot contains many copies of a probe
- 10 K to 1 M spots
- Probes are oligonucleotides (e.g. DNA 25-mers)
- One probe type, like `ACGTTAG...`, per spot

Chip Production: Probe Synthesis



Two Optimization Problems

Situation

- We know the probe sequences.
- They must be synthezised on the chip.
- Probes are subsequences of the deposition sequence.

Problem 1

Shortest deposition sequence for given probes?

Two Optimization Problems

Situation

- We know the probe sequences.
- They must be synthezised on the chip.
- Probes are subsequences of the deposition sequence.

Problem 1

Shortest deposition sequence for given probes?
Hard to solve to optimality in practice, e.g. SR, ECCB 2003.

Two Optimization Problems

Situation

- We know the probe sequences.
- They must be synthezised on the chip.
- Probes are subsequences of the deposition sequence.

Problem 1

Shortest deposition sequence for given probes?
Hard to solve to optimality in practice, e.g. SR, ECCB 2003.

Problem 2

Most universal deposition sequence of given length? I.e., find

$$C_k^*(n, |\Sigma|) = \max_{s \in \Sigma^n} C_k(s),$$

$$Best_k^*(n, |\Sigma|) = \{s \in \Sigma^n : C_k(s) = C_k^*(n, \Sigma)\}.$$

Already solved when I was born

Theorem (Chase, 1976)

The sequences of length N that contain the largest number of distinct subsequences of length n, uniformly of each $n \leq N$, are precisely the repeated permutations of the alphabet.

Already solved when I was born

Theorem (Chase, 1976)

The sequences of length N that contain the largest number of distinct subsequences of length n, uniformly of each $n \leq N$, are precisely the repeated permutations of the alphabet.

Caveats:

- No straightforward way to prove this from recurrence.
- Does not apply to general Bernoulli model.

Already solved when I was born

Theorem (Chase, 1976)

The sequences of length N that contain the largest number of distinct subsequences of length n, uniformly of each $n \leq N$, are precisely the repeated permutations of the alphabet.

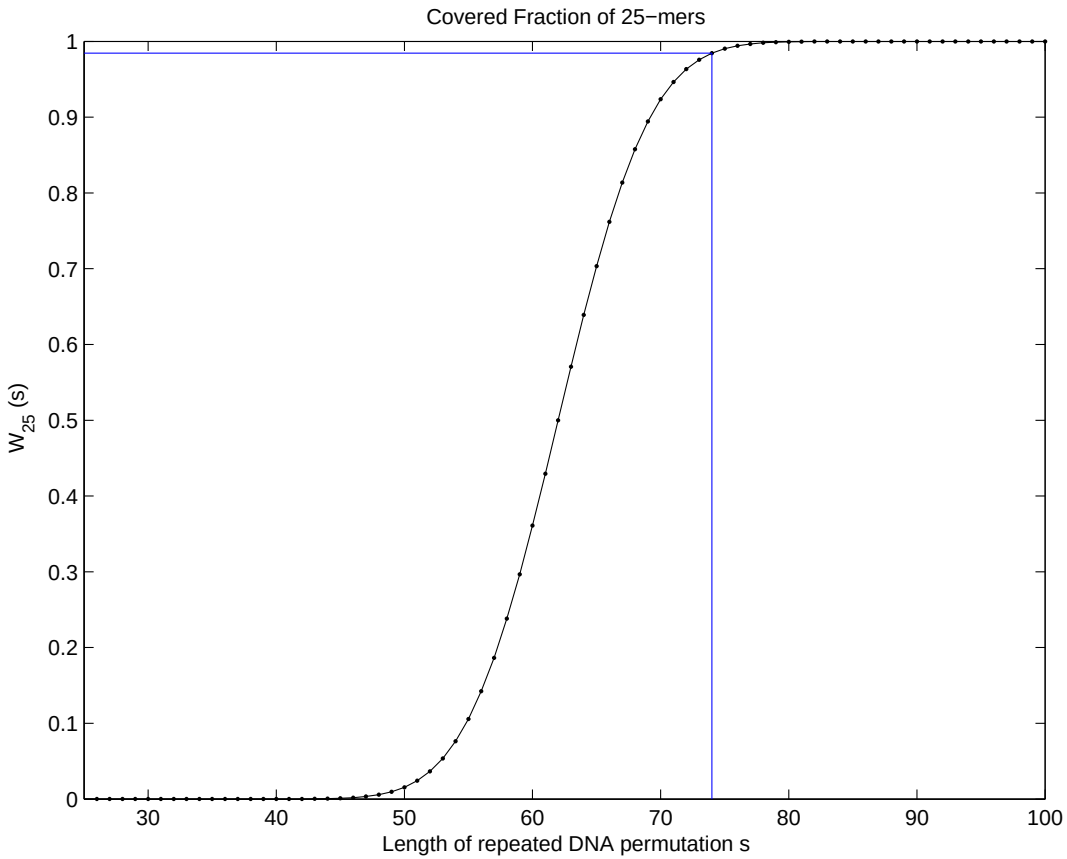
Caveats:

- No straightforward way to prove this from recurrence.
- Does not apply to general Bernoulli model.

In practice:

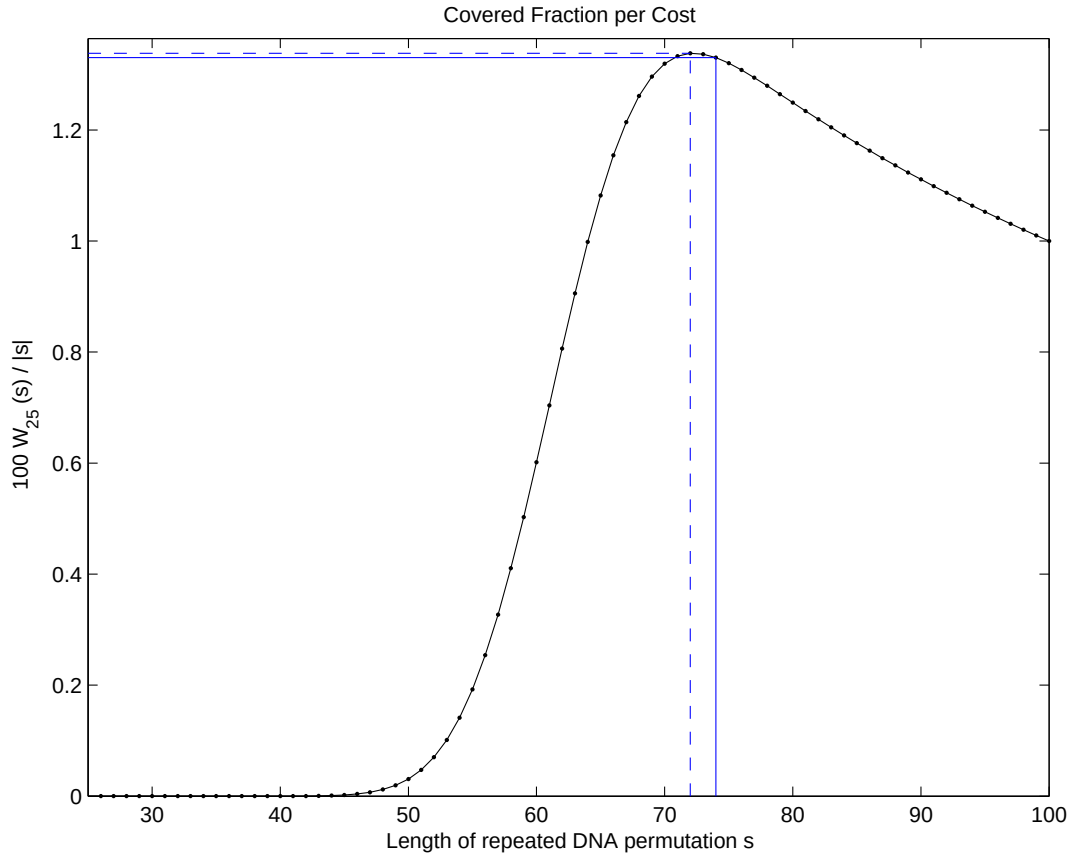
Deposition sequence $\mathrm{ACGT}^{18.5}$ (length 74) for 25-mers.

Covered Fraction of Probes



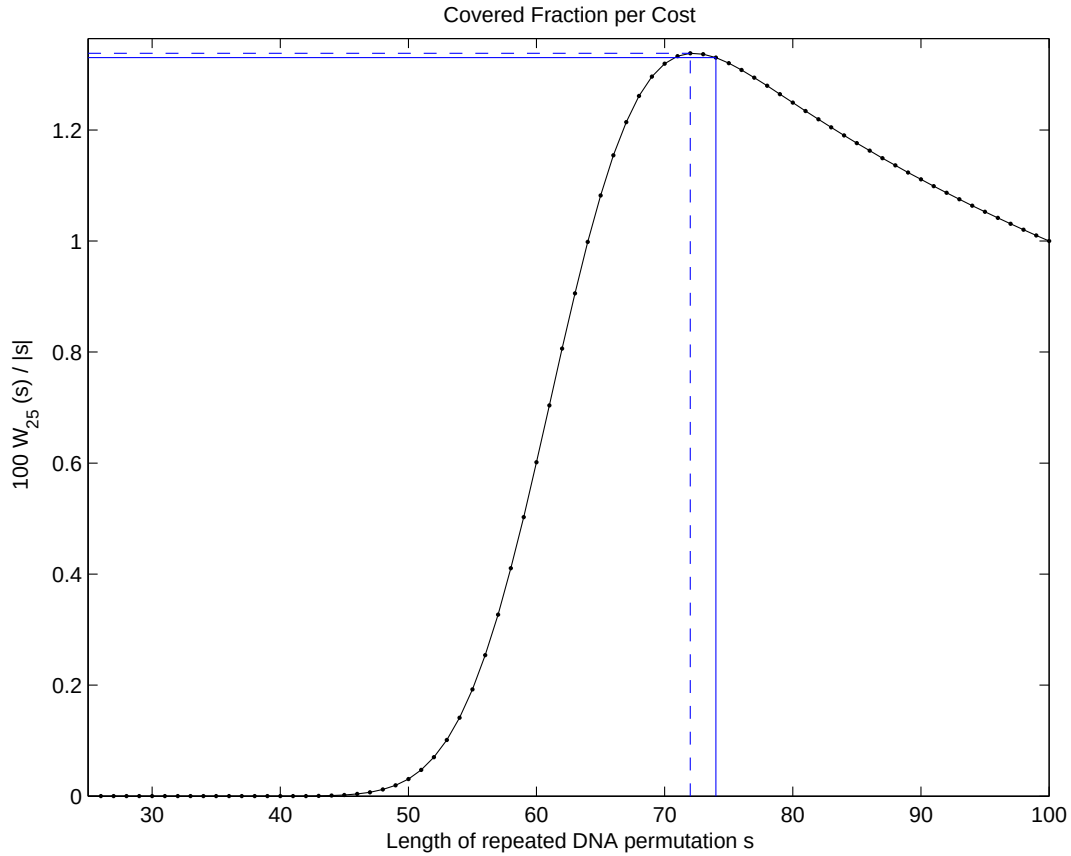
Fraction of 25-mers covered by a repeated permutation of varying length. Highlighted: s^* (length 74), 98.45% of 25-mers covered.

Value-based View



Covered fraction per number of steps. Best value:
$s^{\$}$, 72 steps, $W_{25}(s^{\$}) = 96.34\%$, $100\, W_{25}(s^{\$})/72 = 1.338$.

Value-based View



Covered fraction per number of steps. Best value:

$s^{\$}$, 72 steps, $W_{25}(s^{\$}) = 96.34\%$, $100\, W_{25}(s^{\$})/72 = 1.338$.

s^{*}, 74 steps, $W_{25}(s^{*}) = 98.45\%$, $100\, W_{25}(s^{*})/74 = 1.3304$.

Generalizing Subsequences

- Now: Call s the generating sequence.
- Each s_j may generate a run of up to ρ copies of itself.

Generalizing Subsequences

- Now: Call s the generating sequence.
- Each s_j may generate a run of up to ρ copies of itself.
- Write $t \lhd_\rho s$ ("t is ρ-genrated by s")
  if $\exists\ 0 \leq r_i \leq \rho\ (i = 1, \ldots, n)$ with $|t| = \sum_i r_i$, such that

$$t = s_1^{r_1} s_2^{r_2} \ldots s_n^{r_n}.$$

Generalizing Subsequences

- Now: Call s the generating sequence.
- Each s_j may generate a run of up to ρ copies of itself.
- Write $t \lhd_\rho s$ ("t is ρ-genrated by s")
  if $\exists\ 0 \leq r_i \leq \rho\ (i = 1, \ldots, n)$ with $|t| = \sum_i r_i$, such that

$$t = s_1^{r_1} s_2^{r_2} \ldots s_n^{r_n}.$$

Additional Restriction:

- t may not contain runs longer than ρ.
- Example ($\rho = 2$): Forbidden to generate AAA from BABA.
- Define Σ_ρ^k as ρ-restricted strings of length k.

Combinatorial Problems

Problem

Determine the number (weight) of ρ-restricted length-k strings ρ-generated by s.

Combinatorial Problems

Problem

Determine the number (weight) of ρ-restricted length-k strings ρ-generated by s.

Assumption for s: $s_j \neq s_{j+1}$ for all j.

Combinatorial Problems

Problem

Determine the number (weight) of ρ-restricted length-k strings ρ-generated by s.

Assumption for s: $s_j \neq s_{j+1}$ for all j.

Definitions:

- $S_k(s;\rho) := \{t \in \Sigma_\rho^k : t \lhd_\rho s\}$
- $C_k(s;\rho) := |S_k(s;\rho)|$
- $W_k(s;\rho) := \mathbb{P}_k(S_k(s;\rho))$.

Combinatorial Problems

Problem

Determine the number (weight) of ρ-restricted length-k strings ρ-generated by s.

Assumption for s: $s_j \neq s_{j+1}$ for all j.

Definitions:

- $S_k(s;\rho) := \{t \in \Sigma_\rho^k : t \lhd_\rho s\}$
- $C_k(s;\rho) := |S_k(s;\rho)|$
- $W_k(s;\rho) := \mathbb{P}_k(S_k(s;\rho))$.

From now on, assume s and ρ as fixed.

Auxiliary Quantities

- $S_{m,j}[a] := \{t \in \Sigma_\rho^m : t \lhd_\rho s_1 \dots s_j \text{ and } t_m = a\}$
- $S_{m,j}[\overline{a}] := \bigcup_{b \neq a} S_{m,j}[b]$ $(m > 0)$, or $:= \{\epsilon\}$ $(m = 0)$

Auxiliary Quantities

- $S_{m,j}[a] := \{t \in \Sigma_\rho^m : t \lhd_\rho s_1 \dots s_j \text{ and } t_m = a\}$
- $S_{m,j}[\overline{a}] := \bigcup_{b \neq a} S_{m,j}[b]$ $(m > 0)$, or $:= \{\epsilon\}$ $(m = 0)$
- $C_{m,j}[a] := |S_{m,j}[a]|$
- $C_{m,j}[\overline{a}] := |S_{m,j}[\overline{a}]|$

Auxiliary Quantities

- $S_{m,j}[a] := \{t \in \Sigma_\rho^m : t \lhd_\rho s_1 \dots s_j \text{ and } t_m = a\}$
- $S_{m,j}[\overline{a}] := \bigcup_{b \neq a} S_{m,j}[b]$ $(m > 0)$, or $:= \{\epsilon\}$ $(m = 0)$
- $C_{m,j}[a] := |S_{m,j}[a]|$
- $C_{m,j}[\overline{a}] := |S_{m,j}[\overline{a}]|$
- $W_{m,j}[a] := \mathbb{P}_m(S_{m,j}[a])$
- $W_{m,j}[\overline{a}] := \mathbb{P}_m(S_{m,j}[\overline{a}])$

Auxiliary Quantities

- $S_{m,j}[a] := \{t \in \Sigma_\rho^m : t \lhd_\rho s_1 \ldots s_j \text{ and } t_m = a\}$
- $S_{m,j}[\overline{a}] := \bigcup_{b \neq a} S_{m,j}[b]$ $(m > 0)$, or $:= \{\epsilon\}$ $(m = 0)$
- $C_{m,j}[a] := |S_{m,j}[a]|$
- $C_{m,j}[\overline{a}] := |S_{m,j}[\overline{a}]|$
- $W_{m,j}[a] := \mathbb{P}_m(S_{m,j}[a])$
- $W_{m,j}[\overline{a}] := \mathbb{P}_m(S_{m,j}[\overline{a}])$

Obviously,

- $S_{0,j}[a] = \{\}$
- $S_{0,j}[\overline{a}] = \{\epsilon\}$

Recurrences

Lemma (Structure)

$$\text{For } 1 \leq m \leq j, \quad S_{m,j}[a] = \begin{cases} S_{m,j-1}[a] & \text{if } s_j \neq a, \\ \bigcup_{r=1}^{\min\{\rho,m\}} (S_{m-r,j-1}[\bar{a}] \circ \{a^r\}) & \text{if } s_j = a, \end{cases}$$

where the union is disjoint.

Recurrences

Lemma (Structure)

$$\text{For } 1 \le m \le j, \quad S_{m,j}[a] = \begin{cases} S_{m,j-1}[a] & \text{if } s_j \neq a, \\ \bigcup_{r=1}^{\min\{\rho,m\}} (S_{m-r,j-1}[\overline{a}] \circ \{a^r\}) & \text{if } s_j = a, \end{cases}$$

where the union is disjoint.

Lemma (Weights)

$W_{0,j}[\overline{a}] = 1$, $W_{0,j}[a] = 0$ *for all* $a \in \Sigma$, $j \ge 0$. *For* $m \ge 1$, $j \ge 1$,

$$W_{m,j}[a] = \begin{cases} W_{m,j-1}[a] & \text{if } s_j \neq a, \\ \sum_{r=1}^{\min\{m,\rho\}} W_{m-r,j-1}[\overline{a}] \cdot \pi_a^r & \text{if } s_j = a. \end{cases}$$

Result is $W_k(s) = W_{k,n}[\overline{a}] + W_{k,n}[a]$ *for any* $a \in \Sigma$.

Remarks

1. W-recurrence can be implemented in $O(k(n + |\Sigma|))$ operations.

Remarks

1. W-recurrence can be implemented in $O(k(n + |\Sigma|))$ operations.
2. $\rho = \infty$: How many Σ^k-strings match $s_1 * s_2 * \ldots s_n *$?

Remarks

1. W-recurrence can be implemented in $O(k(n + |\Sigma|))$ operations.
2. $\rho = \infty$: How many Σ^k-strings match $s_1 * s_2 * \ldots s_n *$?
3. Conjecture:
   Repeated permutations maximize $C_k(s; \rho)$ over $s \in \Sigma^n$ for all k

454 Sequencing

- Large-scale sequencing technology by synthesis & light emission.
- Up to 200K (short) fragments in parallel.
- Each nucleotide run is determined in one step
- Works only reliably up to run length $\rho = 8$ in practice.

Example ($\rho = 4$): Try to sequence

1. ACCTG
2. CTTTT
3. GGGGG

by using the probing sequence

- ACGTACG...

454 Sequencing

- Large-scale sequencing technology by synthesis & light emission.
- Up to 200K (short) fragments in parallel.
- Each nucleotide run is determined in one step
- Works only reliably up to run length $\rho = 8$ in practice.

Example ($\rho = 4$): Try to sequence

1. ACCTG
   A
2. CTTTT
3. GGGGG

by using the probing sequence

- ACGTACG...
  A

454 Sequencing

- Large-scale sequencing technology by synthesis & light emission.
- Up to 200K (short) fragments in parallel.
- Each nucleotide run is determined in one step
- Works only reliably up to run length $\rho = 8$ in practice.

Example ($\rho = 4$): Try to sequence

1. ACCTG
   ACC
2. CTTTT
   C
3. GGGGG

by using the probing sequence

- ACGTACG...
  AC

454 Sequencing

- Large-scale sequencing technology by synthesis & light emission.
- Up to 200K (short) fragments in parallel.
- Each nucleotide run is determined in one step
- Works only reliably up to run length $\rho = 8$ in practice.

Example ($\rho = 4$): Try to sequence

1. ACCTG
   ACC
2. CTTTT
   C
3. GGGGG
   GGGG?

by using the probing sequence

- ACGTACG...
  ACG

454 Sequencing

- Large-scale sequencing technology by synthesis & light emission.
- Up to 200K (short) fragments in parallel.
- Each nucleotide run is determined in one step
- Works only reliably up to run length $\rho = 8$ in practice.

Example ($\rho = 4$): Try to sequence

1. ACCTG
   ACCT
2. CTTTT
   CTTTT
3. GGGGG
   GGGG?

by using the probing sequence

- ACGTACG...
  ACGT

454 Sequencing

- Large-scale sequencing technology by synthesis & light emission.
- Up to 200K (short) fragments in parallel.
- Each nucleotide run is determined in one step
- Works only reliably up to run length $\rho = 8$ in practice.

Example ($\rho = 4$): Try to sequence

1. ACCTG
   ACCT
2. CTTTT
   CTTTT
3. GGGGG
   GGGG?

by using the probing sequence

- ACGTACG...
  ACGTA

454 Sequencing

- Large-scale sequencing technology by synthesis & light emission.
- Up to 200K (short) fragments in parallel.
- Each nucleotide run is determined in one step
- Works only reliably up to run length $\rho = 8$ in practice.

Example ($\rho = 4$): Try to sequence

1. ACCTG
   ACCT
2. CTTTT
   CTTTT
3. GGGGG
   GGGG?

by using the probing sequence

- ACGTACG...
  ACGTAC

454 Sequencing

- Large-scale sequencing technology by synthesis & light emission.
- Up to 200K (short) fragments in parallel.
- Each nucleotide run is determined in one step
- Works only reliably up to run length $\rho = 8$ in practice.

Example ($\rho = 4$): Try to sequence

1. ACCTG
   ACCTG
2. CTTTT
   CTTTT
3. GGGGG
   GGGG?

by using the probing sequence

- ACGTACG...
  ACGTACG

454 Sequencing

- Large-scale sequencing technology by synthesis & light emission.
- Up to 200K (short) fragments in parallel.
- Each nucleotide run is determined in one step
- Works only reliably up to run length $\rho = 8$ in practice.

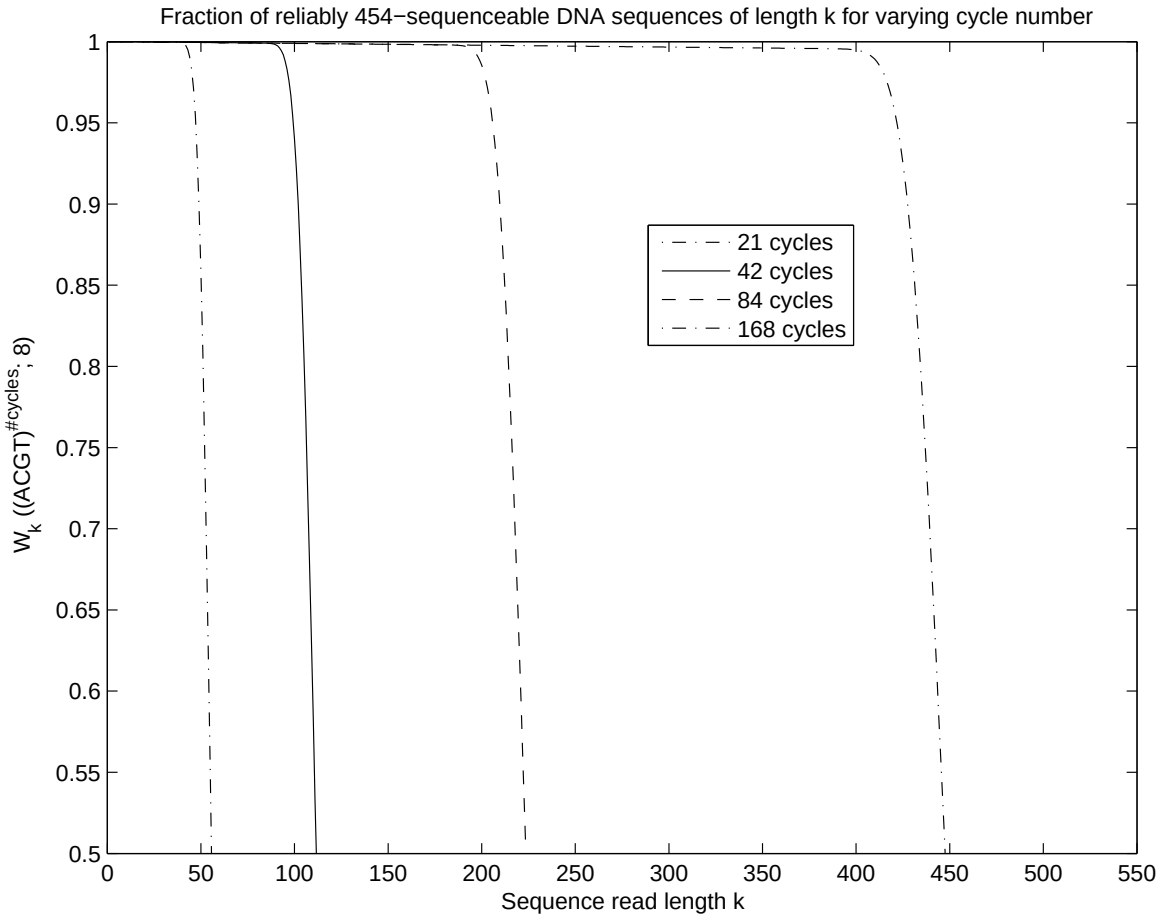
Example ($\rho = 4$): Try to sequence

1. ACCTG
   ACCTG
2. CTTTT
   CTTTT
3. GGGGG
   GGGG?

by using the probing sequence

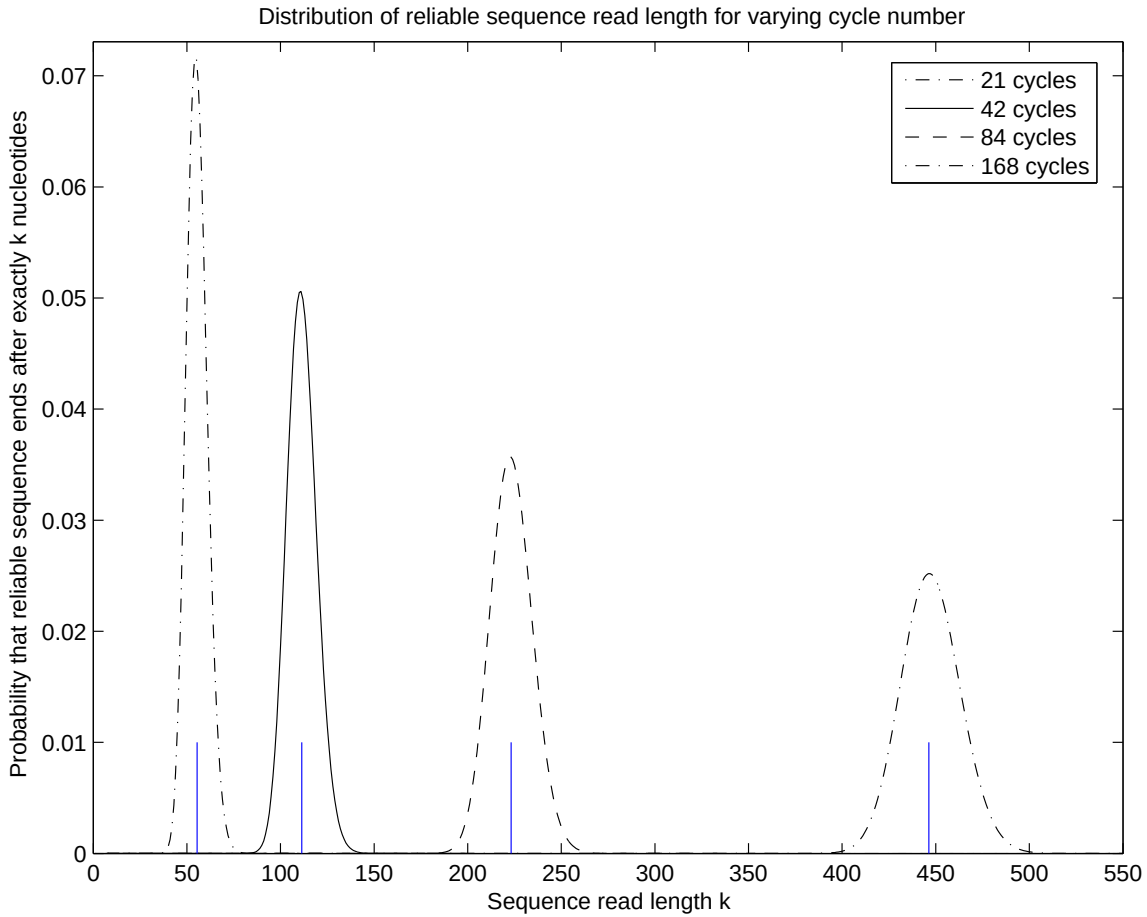
- ACGTACG...
  ACGTACG...

Fraction of 454-sequenceable Sequences



Fraction W_k of 454-sequenceable length-k DNA sequences by using a repeated permutation for $c \in \{21, 42, 84, 168\}$ cycles, $\rho = 8$.

Distribution of Read Lengths



Length distribution of sequenceable initial fragment of a random DNA sequence, $c \in \{21, 42, 84, 168\}$, $\rho = 8$. Vertical lines: Expectations.

Longest Increasing Subsequences

- $\Sigma := \{1, \ldots, K\} =: [K]$
- S: random sequence from Σ^K (Bernoulli model)
- $L_n := LIS(S) :=$ length of longest increasing subsequence of S

Longest Increasing Subsequences

- $\Sigma := \{1, \ldots, K\} =: [K]$
- S: random sequence from Σ^K (Bernoulli model)
- $L_n := LIS(S) :=$ length of longest increasing subsequence of S

Problem

Determine exact distribution of random variable L_n.
(How many sequences s exist with $LIS(s) = k$?)

Patience Sorting

Algorithm for computing $LIS(s)$ for given s:

- Scan s from left to right
- Track $\kappa \subset [K]$ with $|\kappa| = LIS(s_1 \dots s_j)$ after step j
- Update in $O(\log K)$ operations per step:

Patience Sorting

Algorithm for computing $LIS(s)$ for given s:

- Scan s from left to right
- Track $\kappa \subset [K]$ with $|\kappa| = LIS(s_1 \dots s_j)$ after step j
- Update in $O(\log K)$ operations per step:
  $\kappa_j := u(\kappa_{j-1}, s_j)$, where
  $u : 2^{[K]} \times [K] \to 2^{[K]}; (\kappa, c) \mapsto \kappa^+$:
  - If $c \in \kappa$, do nothing, i.e., set $\kappa^+ := \kappa$.
  - If $c \notin \kappa$ and κ contains no element $> c$, add c, i.e., set $\kappa^+ := \kappa \cup \{c\}$.
  - If $c \notin \kappa$ and there exists $k \in \kappa$ with $k > c$, find the smallest such k and decrease it to c, i.e., set $\kappa^+ := \kappa \setminus \{k\} \cup \{c\}$.

Patience Sorting

Algorithm for computing $LIS(s)$ for given s:

- Scan s from left to right
- Track $\kappa \subset [K]$ with $|\kappa| = LIS(s_1 \dots s_j)$ after step j
- Update in $O(\log K)$ operations per step:
  $\kappa_j := u(\kappa_{j-1}, s_j)$, where
  $u : 2^{[K]} \times [K] \to 2^{[K]}; (\kappa, c) \mapsto \kappa^+$:
  - If $c \in \kappa$, do nothing, i.e., set $\kappa^+ := \kappa$.
  - If $c \notin \kappa$ and κ contains no element $> c$, add c, i.e., set $\kappa^+ := \kappa \cup \{c\}$.
  - If $c \notin \kappa$ and there exists $k \in \kappa$ with $k > c$, find the smallest such k and decrease it to c, i.e., set $\kappa^+ := \kappa \setminus \{k\} \cup \{c\}$.
- Intuition:
  - s: sequence of cards
  - place card s_j on top of leftmost stack with higher/equal value
  - κ: set of cards on top
  - $|\kappa|$: number of stacks $= LIS$

Efficent Counting by Conditioning on κ

Avoid running patience sorting for all K^n sequences separately: Let

- $S_j(\kappa) := \{t \in \Sigma^j : \kappa_j(t) = \kappa\}$,
- $C_j(\kappa) := |S_j(\kappa)|$,
- $W_j(\kappa) := \mathbb{P}_j(S_j(\kappa))$.

Then

- $S_n(k) := \bigcup_{\substack{\kappa \subset [K], \\ |\kappa|=k}} S_j(\kappa)$,
- $C_n(k) := \sum_{\substack{\kappa \subset [K], \\ |\kappa|=k}} C_j(\kappa)$,
- $W_j(k) := \sum_{\substack{\kappa \subset [K], \\ |\kappa|=k}} W_j(\kappa)$

are set, number, weight of length-n sequences with $LIS = k$, respectively.

Efficent Counting by Conditioning on κ

- For $j = 0$:
  - $\kappa = \{\}$: $S_0(\kappa) = \{\epsilon\}$, $C_0(\kappa) = 1$, $W_0(\kappa) = 1$.
  - $\kappa \neq \{\}$: $S_0(\kappa) = \{\}$, $C_0(\kappa) = 0$, $W_0(\kappa) = 0$.
- For $1 \leq j \leq n$ and $\kappa \in 2^{[K]}$,

$$S_j(\kappa) = \bigcup_{(\kappa',c)\in u^{-1}(\kappa)} S_{j-1}(\kappa') \circ \{c\},$$

$$C_j(\kappa) = \sum_{(\kappa',c)\in u^{-1}(\kappa)} C_{j-1}(\kappa'),$$

$$W_j(\kappa) = \sum_{(\kappa',c)\in u^{-1}(\kappa)} W_{j-1}(\kappa') \cdot \pi_c.$$

Efficent Counting by Conditioning on κ

- For $j = 0$:
  - $\kappa = \{\}$: $S_0(\kappa) = \{\epsilon\}$, $C_0(\kappa) = 1$, $W_0(\kappa) = 1$.
  - $\kappa \neq \{\}$: $S_0(\kappa) = \{\}$, $C_0(\kappa) = 0$, $W_0(\kappa) = 0$.
- For $1 \leq j \leq n$ and $\kappa \in 2^{[K]}$,

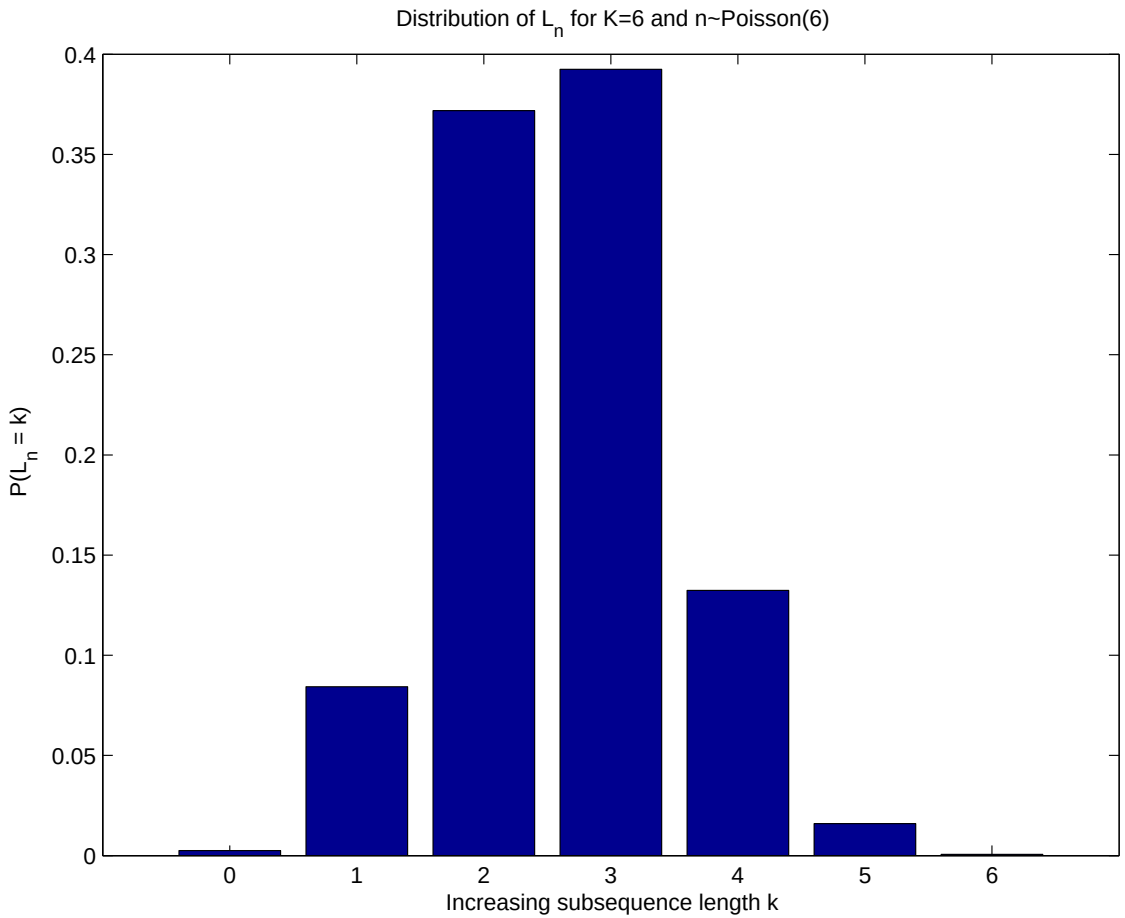
$$S_j(\kappa) = \bigcup_{(\kappa',c)\in u^{-1}(\kappa)} S_{j-1}(\kappa') \circ \{c\},$$

$$C_j(\kappa) = \sum_{(\kappa',c)\in u^{-1}(\kappa)} C_{j-1}(\kappa'),$$

$$W_j(\kappa) = \sum_{(\kappa',c)\in u^{-1}(\kappa)} W_{j-1}(\kappa') \cdot \pi_c.$$

- Pull-type recurrence.
- Easier to implement: Push-type algorithm
- Number of operations: $O(nK2^K)$, FPT in K.

Application: Significance of long chains of rare motifs



LIS-length distribution, $K = 6$.
Random sequence length $N \sim \text{Poisson}(6)$.

Summary

1. Number of distinct subsequences of s, related to microarray production.

Summary

1. Number of distinct subsequences of s,
   related to microarray production.
2. Number of ρ-generated ρ-restricted subsequences of s,
   related to 454 sequencing.
   First analysis of this new high-throughput technology.

Summary

1. Number of distinct subsequences of s,
   related to microarray production.
2. Number of ρ-generated ρ-restricted subsequences of s,
   related to 454 sequencing.
   First analysis of this new high-throughput technology.
3. Distribution of the length of the LIS of s,
   applicable to analysis to chaining algorithms.

Summary

1. Number of distinct subsequences of s,
   related to microarray production.
2. Number of ρ-generated ρ-restricted subsequences of s,
   related to 454 sequencing.
   First analysis of this new high-throughput technology.
3. Distribution of the length of the LIS of s,
   applicable to analysis to chaining algorithms.

Properties of subsequences not widely analyzed yet.
Many more questions to ask and problems to solve!

Acknowledgments

Practical and Genome Informatics, Bielefeld University:

- Marc Rehmsmeier
- Sérgio A. de Carvalho Jr.
- Michael Beckstette
- Robert Homann
- Jens Stoye

Graduate School for Bioinformatics and Genome Research:

- Dirk Evers

Especially:

- Lea Sasaki

Acknowledgments

Practical and Genome Informatics, Bielefeld University:

- Marc Rehmsmeier
- Sérgio A. de Carvalho Jr.
- Michael Beckstette
- Robert Homann
- Jens Stoye

Graduate School for Bioinformatics and Genome Research:

- Dirk Evers

Especially:

- Lea Sasaki

Thank you for listening

Questions?