

Subsequence Combinatorics and Applications to Microarray Production, DNA Sequencing and Chaining Algorithms

Sven Rahmann¹

Algorithms and Statistics for Systems Biology Group,
Genome Informatics, Faculty of Technology, Bielefeld University,
D-33594 Bielefeld, Germany
Sven.Rahmann@cebitec.uni-bielefeld.de

Abstract. We investigate combinatorial enumeration problems related to subsequences of strings; in contrast to substrings, subsequences need not be contiguous. For a finite alphabet Σ , the following three problems are solved. **(1) Number of distinct subsequences:** Given a sequence $s \in \Sigma^n$ and a nonnegative integer $k \leq n$, how many distinct subsequences of length k does s contain? A previous result by Chase states that this number is maximized by choosing s as a repeated permutation of the alphabet. This has applications in DNA microarray production. **(2) Number of ρ -restricted ρ -generated sequences:** Given $s \in \Sigma^n$ and integers $k \geq 1$ and $\rho \geq 1$, how many distinct sequences in Σ^k contain no single nucleotide repeat longer than ρ and can be written as $s_1^{r_1} \dots s_n^{r_n}$ with $0 \leq r_i \leq \rho$ for all i ? For $\rho = \infty$, the question becomes how many length- k sequences match the regular expression $s_1^* s_2^* \dots s_n^*$. These considerations allow a detailed analysis of a new DNA sequencing technology (“454 sequencing”). **(3) Exact length distribution of the longest increasing subsequence:** Given $\Sigma = \{1, \dots, K\}$ and an integer $n \geq 1$, determine the number of sequences in Σ^n whose longest strictly increasing subsequence has length k , where $0 \leq k \leq K$. This has applications to significance computations for chaining algorithms.

1 Introduction

In contrast to substrings, subsequences have received less attention as objects in pattern matching; yet certain aspects of recent technologies emerging in the life sciences, such as short oligonucleotide microarrays or massive short range DNA sequencing by the so-called 454 approach, directly lead to subsequence enumeration problems. The present paper studies a selection of them and presents applications in molecular biology.

A string of length n over a finite alphabet Σ contains $(n+1)n/2 = \Theta(n^2)$ (nonempty) substrings, but 2^n subsequences (including the empty string), making enumerative combinatorics on subsequences potentially more difficult. For a fixed length $1 \leq k \leq n$, there are $n-k+1$ substrings and $\binom{n}{k}$ subsequences of length k . Note that not all of these need to be different.

In Section 2 we present an algorithm that needs $O(k(n + |\Sigma|))$ arithmetic operations to count the number $C_k(s)$ of distinct length- k subsequences in s . When we compute $C_k(s)$ exactly, the size of these numbers is $O(k \log |\Sigma|)$ bits, so arithmetic operations cannot be assumed to take constant time in the RAM model of computation; however, if we are satisfied with computing them with constant precision, we can make this assumption. Therefore we specify running times in numbers of arithmetic operations. The ability to count the number of distinct subsequences contained in a sequence has applications to DNA microarray production, which we outline also in Section 2.

For the next problem, we generalize the notion of subsequence and say that t is *generated* by s if it consists of a concatenation of runs (repetitions) of selected characters from s . This allows, for example, to determine the number of length- k sequences that match the regular expression $s_1^*s_2^*\dots s_n^*$, where a^* matches an arbitrary number of (including zero) occurrences of $a \in \Sigma$. If we additionally restrict the run lengths to be bounded by a constant $\rho \geq 1$, the question of determining how many length- k strings are generated by a given string s is of interest for evaluating a new massively parallel DNA sequencing technology (“454 sequencing”, [1]). Section 3 presents an efficient counting algorithm and computational results. The results of Sections 2 and 3 can be summarized as

Theorem 1. *The number of distinct subsequences and the number of ρ -restricted ρ -generated length- k sequences from a sequence of length n over an alphabet Σ can be computed with $O(k(n + |\Sigma|))$ arithmetic operations.*

Finally, we are interested in the distribution of the longest (strictly) increasing subsequence of $s \in \Sigma^n$ over an ordered alphabet $\Sigma := \{1, \dots, K\}$. We say that t is an increasing subsequence of s if there exists an integer $1 \leq k \leq n$ and indices $1 \leq j_1 < j_2 < \dots < j_k \leq n$ such that $t = s_{j_1}s_{j_2}\dots s_{j_k}$ and $s_{j_1} < s_{j_2} < \dots < s_{j_k}$. Let $I(s)$ be the set of all increasing subsequences of s , and let $LIS(s) := \max_{t \in I(s)} |t|$ be the length of the longest increasing subsequence.

Our goal is to determine the distribution of $L_n := LIS(S)$, where S is a random length- n sequence. Recently, the analogous problem has been completely solved on uniform random permutations; there are exact results for finite n and asymptotic results for $n \rightarrow \infty$ provided by the Baik-Deift-Johansson Theorem, e.g., the expected length is $2\sqrt{n} + \Theta(n^{1/6})$, the standard deviation is $\Theta(n^{1/6})$, and the limiting distribution of $(L_n - 2\sqrt{n})/n^{1/6}$ is completely known. A review of these results on permutations and additional results on weakly increasing subsequences on words appears in [2]. So far there seem to exist no exact nor asymptotic results on strictly increasing subsequences in words. Our contribution is a method that needs $O(nK2^K)$ arithmetic operations on $O(n \log K)$ -bit numbers to compute the exact distribution (in terms of absolute numbers). We thus have the following fixed-parameter tractability (FPT) result (see [3] for an introduction to the terminology).

Theorem 2. *For given string length n and parameter alphabet size K , the decision problem whether there are at least $T \geq 0$ sequences in $s \in \{1, \dots, K\}^n$ with $L(s) = k$ for any $1 \leq k \leq K$ is FPT.*

2 The Number of Distinct Subsequences

Let Σ be a finite alphabet of size σ ; w.l.o.g. we assume $\Sigma = \{1, \dots, \sigma\}$. Further, let $s \in \Sigma^n$ and an integer $1 \leq k \leq n$ be given. We write $t \triangleleft s$ to indicate that t is a subsequence of s , i.e., there exist indices $1 \leq i_1 < i_2 < \dots < i_{|t|} \leq n$ such that $s_{i_1} s_{i_2} \dots s_{i_{|t|}} = t$. Our goal is to determine the cardinality $C_k(s)$ of $S_k(s) := \{t \in \Sigma^k : t \triangleleft s\}$, i.e., the number of distinct length- k subsequences in s . To compute $C_k(s)$ efficiently, we derive a recurrence on the number of distinct subsequences of given length in a given prefix of s that end with a specified character. We drop the dependence on Σ and s in the notation and define

$$S_{m,j} := \{t \in \Sigma^m : t \triangleleft s_1 \dots s_j\}, \quad C_{m,j} := |S_{m,j}|.$$

We refine this definition by conditioning on the last character $a \in \Sigma$:

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

Note that $S_{0,j} = \{\epsilon\}$ (the set consisting of the empty string) for all j , but $S_{0,j}[a] = \{\}$ for all $a \in \Sigma$, so $1 = C_{0,j} \neq \sum_{a \in \Sigma} C_{0,j}[a] = 0$. However, for $m > 0$, we do have $C_{m,j} = \sum_{a \in \Sigma} C_{m,j}[a]$ for all j .

For two sets S and T of strings over Σ , let $S \circ T := \{st : s \in S, t \in T\}$.

The goal is thus to compute $C_k(s) = C_{k,n} = \sum_{a \in \Sigma} C_{k,n}[a]$. The following lemma presents a structural equation for $S_{m,j}[a]$, which leads to a recurrence on $C_{m,j}[a]$ in Lemma 2.

Lemma 1. *Let $1 \leq m \leq j$. Then*

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

Proof. Assume first that $s_j \neq a$. The inclusion $S_{m,j-1}[a] \subset S_{m,j}[a]$ is trivial. We prove that $S_{m,j}[a] \subset S_{m,j-1}[a]$: Take $t \in S_{m,j}[a]$. Since $s_j \neq a$, it follows that t is already a subsequence of a shorter prefix of s , i.e., $t \in S_{m,j-1}[a]$.

Now assume that $s_j = a$. By appending an a to each $t \in S_{m-1,j-1}$ (regardless of its last character), we obtain a distinct string $ta \in S_{m,j}[a]$, thus $S_{m-1,j-1} \circ \{a\} \subset S_{m,j}[a]$. Conversely, every string in $S_{m,j}[a]$ can be written as ta with some $t \in S_{m-1,j-1}$. \square

Lemma 2. *We have $C_{0,0} = 1$ and $C_{0,0}[a] = 0$ for all $a \in \Sigma$. Further, $C_{m,j} = 0$ if $m > j$. For $1 \leq m \leq j$, we have*

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

Proof. Immediate by taking cardinalities in Lemma 1 and noting that concatenation translates to multiplication of set cardinalities. \square

Bernoulli String Model. The fraction of length- k sequences contained in s (or covered by s) is thus $C_{k,n}/\sigma^k$. We can generalize Lemma 2 to a Bernoulli or i.i.d. random string model, where the probability or weight of each length- k string is equal to the product of its (possibly unequal) character frequencies. Hence, let $\pi := (\pi_a)_{a \in \Sigma}$ be a non-degenerate probability distribution on Σ , i.e., $\pi_a > 0$ for all $a \in \Sigma$ and $\sum_{a \in \Sigma} \pi_a = 1$. Let $\mathbb{P}_k(t_1 \dots t_k) := \prod_{j=1}^k \pi_{t_j}$ be the probability of generating $t_1 \dots t_k$ in k steps. It follows that $\sum_{t \in \Sigma^k} \mathbb{P}_k(t) = 1$ for all $k \geq 1$ and $\mathbb{P}_{k+1}(ta) = \mathbb{P}_k(t) \cdot \pi_a$ for $t \in \Sigma^k$ and $a \in \Sigma$.

Let us define $W_k(s) := \mathbb{P}_k(S_k(s))$ as the weighted fraction of length- k sequence space covered by s . For $m \leq k$ and $j \leq |s| = n$, define

$$W_{m,j} := \mathbb{P}_m(S_{m,j}) = \sum_{t \in S_{m,j}[a]} \mathbb{P}_m(t), \quad W_{m,j}[a] := \mathbb{P}_m(S_{m,j}[a]).$$

Lemma 3. $W_{0,0} = 1$ and $W_{0,0}[a] = 0$ for all $a \in \Sigma$. Further, $W_{m,j} = 0$ if $m > j$. For $1 \leq m \leq j$, we have

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

Proof. Immediate by applying $\mathbb{P}_m(\cdot)$ resp. $\mathbb{P}_{m-1}(\cdot)$ to Lemma 1. \square

A straightforward implementation of the recurrence would need $O(nk|\Sigma|)$ arithmetic operations. It is possible to remove the factor $|\Sigma|$ in a careful implementation: Figure 1 presents an algorithm to compute $W_k(s)$ in $O(k(n + |\Sigma|))$ operations. The memory requirements are $O(k|\Sigma|)$ if only $W_k(s)$ is desired or $O(k(n + |\Sigma|))$ if the whole array $W_m(s_1 \dots s_j)$, $1 \leq m \leq k$, $1 \leq j \leq n$, is desired.

Application to DNA microarray production. DNA oligonucleotide microarrays (“DNA chips”) are a tool to monitor the activity level of many genes in cells of living organisms. A DNA chip is a plastic or glass slide containing many *spots*, each consisting of many copies of a known oligomer (a 25-mer for Affymetrix GeneChips[®], which we consider here), also called *probe*, attached to the chip. During production, the probes are synthesized on the chip in parallel on a nucleotide-by-nucleotide-basis. In each synthesis step, the same nucleotide is appended to all probes that have been selectively activated to receive it. Activation occurs by exposure to light, enabling the chemical synthesis reaction. Thus each synthesis step is specified by (1) a nucleotide (a character from the DNA alphabet $\{\mathbf{A}, \mathbf{C}, \mathbf{G}, \mathbf{T}\}$) and (2) a mask, i.e., an index set of the probes to which the nucleotide is appended. The sequence of nucleotides used in the synthesis process is called the *deposition sequence*. Each probe is a subsequence of the deposition sequence, so the deposition sequence is a common supersequence of all probes.

Given a set of probe sequences (in practice up to 10^6 probes can fit on a single chip), one can try to find the shortest deposition sequence, i.e., the shortest common supersequence of all probes (see [4] for bounds on its length and heuristic algorithms). In practice, good deposition sequences can be found but

Input: Alphabet Σ with probability distribution π , string $s \in \Sigma^n$, integer $1 \leq k \leq n$
Output: $W_k(s)$ or the whole array $W_m(s_1 \dots s_j)$ for $m = 1, \dots, k$, $j = 1, \dots, n$

```

// Initialize arrays W, V and Vsum
W[m, j] ← 0 for m ← 1, ..., k, j ← 1, ..., n // optional: stores Wm,j
V[m, a] ← 0 for m ← 1, ..., k, a ∈ Σ // stores Wm,j[a] for current value of j
Vsum[m] ← 0 for m ← 1, ..., k // stores Wm,j for current value of j

for j ← 1, ..., n
  c ← sj // the current character
  for m ← min{j, k}, ..., 3, 2
    // Update V and Vsum s.th. V[m, a] = Wm,j[a] (a ∈ Σ); Vsum[m] = Wm,j:
    // (only the c-entry needs to be updated, saving a factor of |Σ|)
    Vsum[m] ← Vsum[m] - V[m, c]
    V[m, c] ← Vsum[m - 1] · πc
    Vsum[m] ← Vsum[m] + V[m, c]
  end for m
  // Finally, treat the case m = 1 specially:
  if V[1, c] = 0 then V[1, c] ← πc; Vsum[1] ← Vsum[1] + πc; end if
  // Invariant: Here Vsum[m] = Wm(s1 ... sj) = Wm,j for m = 1, ..., k
  W[m, j] ← Vsum[m] for m ← 1, ..., k // optional: set j-th column of W:
end for j
return Vsum[k] // optional: return array W

```

Fig. 1. An algorithm with $O(k(n + |\Sigma|))$ operations to compute the π -weighted fraction $W_k(s)$ of length- k strings that are subsequences of s . The array W is not needed when only $W_k(s)$ is required: after step j , the j -th column of W is equal to $Vsum$.

not proved optimal in a reasonable amount of time. Therefore one can approach the question differently and ask for a deposition sequence that is as “universal” as possible, i.e., that contains the largest number of distinct subsequences. We thus ask for

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

A result due to P.J. Chase [5] from 1976 (long before the invention of microarrays) states that precisely the *repeated permutations* of the alphabet form the set $Best_k^*(n, |\Sigma|)$ with the consequence that this set does not depend on k .

Definition 1. For a finite alphabet Σ of size σ , a string s of length n is called a repeated permutation of Σ if there exists a permutation $\pi = \pi_1 \dots \pi_\sigma$ of the characters in Σ such that $s = \pi^c \pi_1 \dots \pi_m$, where the number of full cycles is $c := \lfloor n/\sigma \rfloor$ and the number of remaining characters $m := n \bmod \sigma$ satisfies $0 \leq m < \sigma$.

In fact, any sequence that is not a repeated permutation contains strictly fewer subsequences of (some) smaller length. Even though this result appears intuitive, it is nontrivial to prove and apparently does not follow directly from the recurrence in Lemma 2; Chase used induction on the longest sequence prefix that is a repeated permutation to prove optimality.

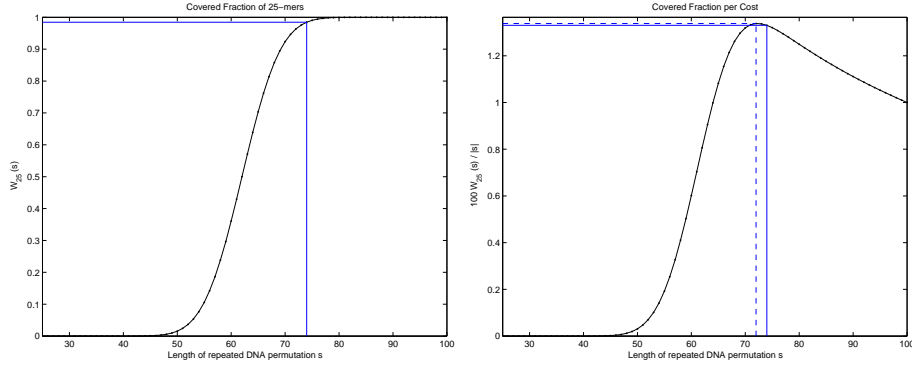


Fig. 2. Left: Fraction of 25-mers covered by a repeated permutation of varying length from 25 to 100: From the deposition sequence s^* of length 74 used for GeneChip[®] production, 98.45% of all 25-mers can be synthesized. Right: Assuming that each synthesis step costs 1/100 (such that using 100 steps implies a cost of 1), the graph shows the covered fraction per cost. The “best value” is obtained for a repeated permutation $s^{\$}$ with 72 steps or 18 full cycles ($W_{25}(s^{\$}) = 96.34\%$, $100 W_{25}(s^{\$})/72 = 1.338$), but s^* is almost as cost-effective ($100 W_{25}(s^*)/74 = 1.3304$) and has higher coverage 98.45%.

The Affymetrix GeneChip[®] technology uses a repeated permutation of length 74, such as $s^* := (\text{ACGT})^{18}\text{AC}$, to synthesize 25-mers. Figure 2 (left) shows the fraction of 25-mers contained in repeated permutations of increasing length: s^* covers a fraction of 98.45% of all 25-mers. Elongating s^* further quickly results in diminishing returns; for example, adding one additional nucleotide would result in 99.04% of the 25-mers being covered. It is unknown to the author why the length of 74 was chosen, but we offer the following hypotheses: The sequences not covered by s^* have somewhat extremal properties. For example, many of them contain runs of a repeated nucleotide. We may assume that such oligos are rarely used on microarrays because of undesirable thermodynamic properties, so s^* may cover in fact all oligos that are ever chosen to be placed on a chip. For another argument consider Figure 2 (right): In practice, each synthesis step has a certain cost (mask production, chemicals, time, etc.). Assuming that the production cost of a chip is proportional to the number of synthesis steps, we see that using a deposition sequence of length 74 offers both high coverage in absolute terms and close to optimal coverage per money.

3 The Number of ρ -Restricted ρ -Generated Sequences

We consider a variation of the previous problem, where we modify the notion of subsequence: We allow that each character from s , which we call the *generating sequence*, may produce a whole run (up to a specified length ρ) of this character. Thus we write $t \triangleleft_{\rho} s$ if there exist n numbers $0 \leq r_i \leq \rho$ for $i = 1, \dots, n$, with $|t| = \sum_i r_i$, such that $t = s_1^{r_1} s_2^{r_2} \dots s_n^{r_n}$. We say that t is ρ -generated by s . For $\rho = 1$, we get the usual notion of subsequence. Note that $t \triangleleft_{\rho} s$ implies $|t| \leq \rho|s|$.

Motivated by the 454 DNA sequencing technology (see below), we are only interested in counting sequences that do not contain a single character run longer than ρ ; so we define Σ_ρ^k as the set of all length- k strings over Σ that do not contain $a^{\rho+1}$ as a substring for any $a \in \Sigma$ and call them the ρ -restricted strings.

The set of ρ -restricted length- k strings ρ -generated by s is denoted by

$$S_k(s; \rho) := \{t \in \Sigma_\rho^k : t \triangleleft_\rho s\}.$$

It is important to note that $a^{2\rho} \triangleleft_\rho aba$, but $a^{2\rho} \notin \Sigma_\rho^{2\rho}$, so $a^{2\rho} \notin S_{2\rho}(aba; \rho)$. Therefore $S_k(s; 1)$ is different from $S_k(s)$ as defined in the previous section.

For the generating sequence $s = (s_1, \dots, s_n)$, we may assume that $s_i \neq s_{i+1}$ for all $i = 1, \dots, n-1$, i.e., $s \in \Sigma_1^n$, since repetitions in the generating sequence do not allow to generate additional ρ -restricted sequences.

We set $C_k(s; \rho) := |S_k(s; \rho)|$ and $W_k(s; \rho) := \mathbb{P}_k(S_k(s; \rho))$. Assuming s and ρ fixed, we define for $1 \leq m \leq k$, $0 \leq j \leq n$ and $a \in \Sigma$ the auxiliary quantities

$$\begin{aligned} S_{m,j}[a] &:= \{t \in \Sigma_\rho^m : t \triangleleft_\rho s_1 \dots s_j \text{ and } t_m = a\}, & S_{m,j}[\bar{a}] &:= \bigcup_{b \neq a} S_{m,j}[b], \\ C_{m,j}[a] &:= |S_{m,j}[a]|, & C_{m,j}[\bar{a}] &:= |S_{m,j}[\bar{a}]|, \\ W_{m,j}[a] &:= \mathbb{P}_m(S_{m,j}[a]), & W_{m,j}[\bar{a}] &:= \mathbb{P}_m(S_{m,j}[\bar{a}]), \end{aligned}$$

with the boundary cases $S_{0,j}[a] = \{\}$ and $S_{0,j}[\bar{a}] = \{\epsilon\}$. The structural recurrence for $S_{m,j}[a]$ is slightly more complicated than in the previous section, since we need to express $S_{m,j}[a]$ as a *disjoint* union to determine its cardinality.

Lemma 4. *Let $1 \leq m \leq j$. Then*

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

where the union is disjoint.

Proof. The case $s_j \neq a$ is proved as in Lemma 1.

For $s_j = a$, appending a^r to any $t \in S_{m-r,j-1}[\bar{a}]$ for any ‘‘run length’’ $1 \leq r \leq \min\{\rho, m\}$ clearly results in a distinct string in $S_{m,j}[a]$. Note that any run length in t is bounded by r by assumption, and in ta^r by construction since t does not end with a . This shows $\bigcup_{r=1}^{\min\{\rho, m\}} S_{m-r,j-1}[\bar{a}] \circ \{a^r\} \subset S_{m,j}[a]$. Conversely, every string in $S_{m,j}[a]$ can be written uniquely as ta^r , where $r \leq \rho$ and $r \leq m$ and $t \in S_{m-r,j-1}[\bar{a}]$ (possibly the empty string). Because of the uniqueness of the above decomposition, the union is disjoint. \square

Lemma 4 immediately allows us to count $C_k(s; \rho)$ and to determine $W_k(s; \rho)$. We only give the Bernoulli string model version for $W_k(s; \rho)$ here.

Lemma 5. *We have $W_{0,j}[\bar{a}] = 1$ and $W_{0,j}[a] = 0$ for all $a \in \Sigma$, $j \geq 0$. For $m \geq 1$ and $j \geq 1$, we have*

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

The desired result is $W_k(s) = W_{k,n}[\bar{a}] + W_{k,n}[a]$ for any $a \in \Sigma$.

Remarks:

1. The recurrence in Lemma 5 can be implemented to run in $O(k(n + |\Sigma|))$ arithmetic operations by remembering appropriate partial sums.
2. Using $\rho = \infty$ answers the question how many strings of length k match the regular expression $s_1^*s_2^*\dots s_n^*$, where a^* matches zero or an arbitrary number of occurrences of $a \in \Sigma$. In Section 2, we effectively determined how many strings of length k match the regular expression $s_1?s_2?\dots s_n?$, where $a?$ matches zero or one occurrence(s) of $a \in \Sigma$.
3. It is reasonable to conjecture that again a repeated permutation s^* maximizes $C_k(s; \rho)$ over all $s \in \Sigma^n$, but this is so far not rigorously proved.
4. Even for arbitrarily large n and optimal $s^* \in \Sigma^n$, we have $C_k(s; \rho)/|\Sigma^k| \leq |\Sigma_\rho^k|/|\Sigma^k| \rightarrow 0$ as $k \rightarrow \infty$, because the probability that a length- k sequence contains a run longer than ρ approaches 1 as $k \rightarrow \infty$.

Analysis of 454 Sequencing. Recently, the company “454 Life Sciences” has developed a massively parallel DNA sequencing technology (simply called “454 sequencing”). We refer the reader to [1] and www.454.com for more detailed information. Several copies of an organism’s genome are randomly cut into DNA fragments; a part of the sequence of each fragment is determined in parallel, and finally the fragment sequences can be assembled to retrieve the whole genomic sequence if each position of the genome is covered by enough fragments. Many copies of one single fragment type are attached to a microscopic bead; each bead is held in place in a different well of the reaction carrier (70 mm \times 75 mm). A typical reaction carrier has 1.6 million wells, from which typically 200,000 different high-quality fragment reads can be obtained.

The fragments are sequenced by synthesizing the complementary (A \leftrightarrow T, C \leftrightarrow G) DNA strand to each fragment in several steps. Initially, the complementary strand of each fragment is empty but ready for extension at its starting point. Then, e.g., in an A-step, T-nucleotides are flooded over the reaction carrier, and Ts are incorporated into complementary strands in those wells where the next character in the fragment sequence is A. Successful elongation of the complementary strand results in a flash of light from the corresponding wells. The light emission pattern is detected with a CCD camera for all wells in parallel. If a fragment contains a consecutive run (homopolymer) of As, all of their counterpart Ts are incorporated in a single step and the light intensity is proportional to the run length. This works reliably only up to a certain length $\rho = 8$, which was the reason for introducing ρ -restricted strings above. Sequences that contain longer homopolymers cannot be reliably sequenced.

Sequencing steps for different nucleotides are repeated in a cyclic pattern for c cycles, e.g., (ACGT) ^{c} . This process cannot go on forever because the signal/noise ratio deteriorated over time. Public information (as of February 2005) at www.454.com states that high-quality sequencing of on average 100-base reads is achieved in 42 cycles of TACG. It has also been attempted to use 84 and 168 cycles for high-quality reads of 200 and 400 bases, respectively.

The key issue is that the fraction of length- k DNA sequences that can be reliably sequenced by this technology in n steps is precisely given by $W_k(s; \rho)$,

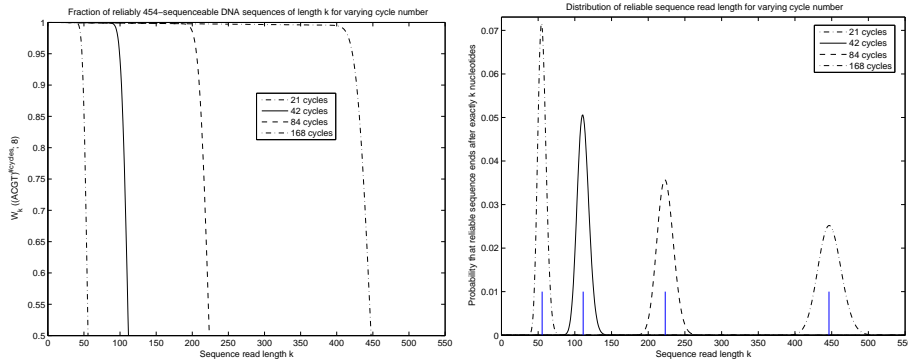


Fig. 3. Left: Fraction W_k of 454-sequenceable length- k DNA sequences by using a repeated permutation of the alphabet for $c \in \{21, 42, 84, 168\}$ cycles, $\rho = 8$. Right: Length distribution of the reliably sequenceable initial fragment of a random DNA sequence, for c and ρ as before. Vertical lines mark the expected lengths.

where $\rho = 8$ and s is a repeated permutation of the DNA alphabet. Assuming a uniform distribution $\pi_a = 1/4$ for each $a \in \Sigma$, we thus determine which fraction $W_k((ACGT)^c; 8)$ of length- k DNA sequences for $1 \leq k \leq 550$ can be reliably sequenced in $c \in \{21, 42, 84, 168\}$ full cycles.

The results are visualized in Figure 3 (left). The longest sequence lengths for which the sequenceable fraction exceeds 99% are $k_{\max} = 48, 101, 209,$ and 427 for $c = 21, 42, 84,$ and 168 cycles. 85.8% of length-50 sequences are sequenceable in 21 cycles, 94.0% of length-100 sequences in 42 cycles, 98.55% of length-200 sequences in 84 cycles, and 99.48% of length-400 sequences in 168 cycles.

A different perspective is shown in Figure 3 (right): If T is any (potentially infinite) random sequence according to the uniform distribution, a certain finite prefix will be reliably sequenced by the generating sequence $s = (ACGT)^c$. Let $L_c(T)$ denote the length of this prefix for c cycles. The figure shows the distribution of L_c for $c \in \{21, 42, 84, 168\}$ cycles, which is obtained as follows. The probability that sequencing ends after k steps or later is $W_k \equiv W_k(s; \rho)$. Therefore, the probability that the read ends *exactly* after k steps is $\mathbb{P}(L_c = k) = W_k - W_{k+1}$. The figure also shows that the expected sequence read length $\mathbb{E}[L_c]$ for 21 (42, 84, 168) cycles is 55.4 (111.4, 223.1, 446.3), which exceeds the company-guaranteed values of 50 (100, 200, 400) by more than 10%. To guarantee these expected read lengths, only 19 (37.75, 75.5, 150.75) cycles, i.e., 76 (151, 302, 603) steps would in fact be necessary on random sequences.

4 Longest Increasing Subsequence Length Distribution

We consider an ordered alphabet $\Sigma := \{1, \dots, K\}$ and a string $s \in \Sigma^n$, and equip Σ^n with a Bernoulli probability measure \mathbb{P}_n given by a probability vector $\pi = (\pi_1, \dots, \pi_K)$, such that $\mathbb{P}_n(s) = \prod_{j=1}^n \pi_{s_j}$. Several algorithms (e.g., [6, 2]) compute the length $LIS(s)$ of the longest increasing subsequence in s .

Our counting method is based on the *patience sorting* algorithm, which scans s from left to right and keeps track of a subset $\kappa \subset [K] := \{1, \dots, K\}$ whose cardinality after j steps is equal to $LIS(s_1 \dots s_j)$. We write $2^{[K]}$ for the power set of $\{1, \dots, K\}$. Initially, we set $\kappa_0 = \{\}$ and in step $j = 1, \dots, n$, κ_j is computed in $O(\log K)$ operations as $\kappa_j := u(\kappa_{j-1}, s_j)$ from the *update function* $u : 2^{[K]} \times [K] \rightarrow 2^{[K]}$; $(\kappa, c) \mapsto \kappa^+$, defined as follows:

- 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 $\kappa^+ := \kappa \setminus \{k\} \cup \{c\}$.

A proof that $|\kappa_j| = LIS(s_1, \dots, s_j)$ and an explanation in terms of stacks of cards is found in [2]. The running time is seen to be $O(n \log K)$. To avoid running patience sorting for all K^n sequences separately, we condition on κ : Let $\kappa_j(t)$ be the final set κ_j in patience sorting when it is applied to $t \in \Sigma^j$. We set

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

It follows that for $0 \leq k \leq K$,

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

are the set, number, and weighted fraction of length- n sequences with $LIS = k$, respectively. The following lemma presents a structural equation between $S_j(\kappa)$ and $S_{j-1}(\kappa')$, where κ' is an update-preimage under u .

Lemma 6. *For $j = 0$, we have $S_0(\{\}) = \{\epsilon\}$, $C_0(\{\}) = 1$, $W_0(\{\}) = 1$, and for $\kappa \in 2^{[K]}$, $\kappa \neq \{\}$, we have $S_0(\kappa) = \{\}$, $C_0(\kappa) = 0$, $W_0(\kappa) = 0$. For $1 \leq j \leq n$ and $\kappa \in 2^{[K]}$,*

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

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

Proof. The equations for C_j and W_j follow immediately from the one for S_j (obviously the union is disjoint), which in turn is a trivial consequence of the correctness of the patience sorting algorithm (i.e., of the update function). \square

Lemma 6 implies a “pull”-type dynamic programming algorithm for computing $W_n(k)$, which has the disadvantage that the update rules must be read “backwards”, i.e., for given κ , we need to determine the pairs (κ', c) with $\kappa = u(\kappa', c)$. It is easier to implement a “push”-type algorithm that pushes the information for all (κ, c) forward to the corresponding $\kappa^+ = u(\kappa, c)$. This is shown in Figure 4.

Application: Significance Computations for Chaining Algorithms. In biological sequence analysis, the following problem arises in several situations (e.g., when attempting to classify proteins or to detect cis-regulatory modules): Certain biological sequences (the *family members*) are characterized by the appearance

Input: Alphabet size K , distribution $\pi = (\pi_1, \dots, \pi_K)$, sequence length n
Output: $W_n(k)$ for $0 \leq k \leq K$ as array $\mathbf{w}[0..K]$

```

 $\mathbb{W}'[\{\}] \leftarrow 1$  and  $\mathbb{W}'[\kappa] \leftarrow 0$  for  $\kappa \in 2^{[K]}$  with  $|\kappa| \geq 1$  // Initialize array  $\mathbb{W}'[\kappa]$  to  $W_0(\kappa)$ 
for  $j \leftarrow 1, \dots, n$ 
   $\mathbb{W}[\kappa] \leftarrow 0$  for  $\kappa \in 2^{[K]}$  // reset array  $\mathbb{W}$  to zero
  // Invariant here:  $\mathbb{W}'[\kappa] = W_{j-1}(\kappa)$  and  $\mathbb{W}[\kappa] \equiv 0$ 
  for  $\kappa \in 2^{[K]}$ ; for  $c \in \Sigma$ 
     $\kappa^+ \leftarrow u(\kappa, c)$ 
     $\mathbb{W}[\kappa^+] \leftarrow \mathbb{W}[\kappa^+] + \mathbb{W}'[\kappa] \cdot \pi_c$ 
  end for  $c$ ; end for  $\kappa$ 
   $\mathbb{W}' \leftarrow \mathbb{W}$  // Invariant:  $\mathbb{W}[\kappa] = \mathbb{W}'[\kappa] = W_j(\kappa)$ 
end for  $j$ 
 $\mathbf{w}[k] \leftarrow 0$  for  $k \leftarrow 0, \dots, K$ 
 $\mathbf{w}[|\kappa|] \leftarrow \mathbf{w}[|\kappa|] + \mathbb{W}[\kappa]$  for all  $\kappa \in 2^{[K]}$ 
return  $\mathbf{w}$ 

```

Fig. 4. Push-type dynamic programming algorithm to compute the length distribution of the longest increasing subsequence for alphabet size K with character distribution $\pi = (\pi_1, \dots, \pi_K)$ and sequence length n . Subsets κ can be encoded as bit-vectors and represented as integers in the range from 0 to $2^K - 1$.

of sequence motifs (e.g., substrings, regular expressions, or sequence profiles) in a certain order. Let there be K distinct motifs and assume that true family members usually contain all of them in the correct order $1, \dots, K$. However, in some family members some motifs may not be present or detected. To decide whether a sequence should be classified as a family member, in a first step, all motif occurrences are tabulated. Then the best chain of motifs is found in a second “chaining” step. We assume that the quality of a chain is its length, so we classify a sequence as a family member if the longest increasing sequence of motif indices reaches a threshold t . To find a statistically significant value of t , we determine the frequency p_t of length- t chains in random sequences.

We assume that the motifs are chosen in such a way that each one occurs with low frequency $0 < f_k \ll 1$ in random sequences. If also $f := \sum_{k=1}^K f_i \ll 1$, motif occurrences can be treated as a Poisson process along a random sequence of length m : If N is the total number of motif occurrences, then $\mathbb{E}[N] = \lambda := m \cdot f$, and the distribution of N can be well approximated as Poisson(λ) with $\mathbb{P}(N = n) = \exp(-\lambda) \cdot \lambda^n / n!$. Given that a motif occurs at some position, it is motif k with probability $\pi_k := f_k / f$.

It follows that the probability of observing an increasing motif sequence of length k in such a random sequence is given by $W_{\text{Poisson}(\lambda)}(k) := \sum_{n=0}^{\infty} \exp(-\lambda) \cdot \lambda^n / n! \cdot W_n(k)$. The p-value associated to a threshold length t is then $p_t = \sum_{k=t}^K W_{\text{Poisson}(\lambda)}(k)$. Now t can be chosen such that p_t is reasonably small.

For example, for $K = 6$ distinct motifs that each appear once in 100 positions on average and sequence length $m = 100$, we have $\lambda = 6$ motif occurrences on average. The Poisson mixture distribution $W_{\text{Poisson}(\lambda)}(k) := \sum_{n=0}^{\infty} \exp(-\lambda) \lambda^n / n! \cdot W_n(k)$ is shown on the left side of Figure 5, the p_t -values on the right side: Thresholds of 5 and 6 imply $p_5 = 0.0165$ and $p_6 = 0.0006$, respectively.

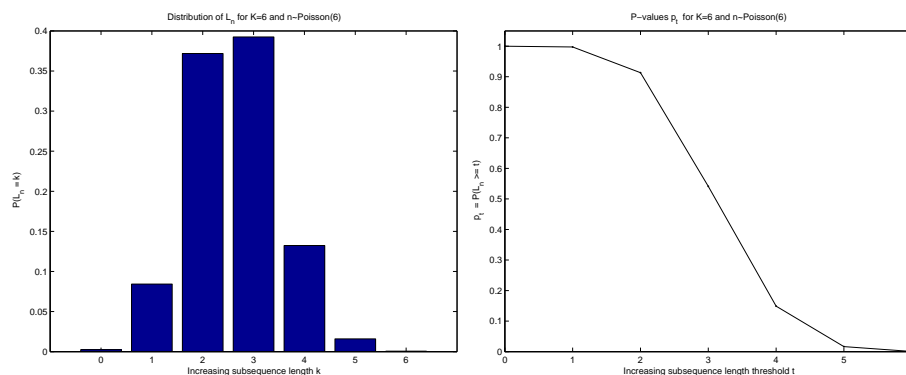


Fig. 5. Left: Length distribution of longest increasing subsequences for alphabet size $K = 6$ and random sequence length $N \sim \text{Poisson}(6)$. Right: Associated p-values.

Concluding Remarks. There is considerable literature about subsequence combinatorics (exact and asymptotic counting) on permutations, but there are few results on words, despite the fact that these have interesting practical consequences, as we have shown. Subsequence combinatorics contains a number of interesting problems., e.g., it remains open to prove that indeed the repeated permutations maximize the number of distinct ρ -restricted ρ -generated sequences.

Acknowledgments. I thank Marc Rehmsmeier, Sergio A. de Carvalho Jr., Michael Beckstette, Robert Homann, Jens Stoye, and Dirk Evers for stimulating discussions, and especially Lea Sasaki for her support.

Terms marked $\text{\textcircled{R}}$ are registered trademarks of their respective owners. The author is not affiliated with Affymetrix or 454 Life Sciences and has no financial interests competing with this research.

References

1. Margulies, M., et al.: Genome sequencing in microfabricated high-density picolitre reactors. *Nature* **437**(7057) (2005) 376–380 / Corrigendum in *Nature* **439**(7075) (2006) p.502.
2. Aldous, D., Diaconis, P.: Longest increasing subsequences: From patience sorting to the Baik-Deift-Johansson theorem. *Bulletin of the American Mathematical Society* **36**(4) (1999) 413–432
3. Niedermeier, R.: *Invitation to Fixed Parameter Algorithms*. Oxford University Press (2006)
4. Rahmann, S.: The shortest common supersequence problem in a microarray production setting. In: *Proceedings of the 2nd European Conference in Computational Biology (ECCB 2003)*. Volume 19 Suppl. 2 of *Bioinformatics*. (2003) ii156–ii161
5. Chase, P.: Subsequence numbers and logarithmic concavity. *Discrete Math.* **16** (1976) 123–140
6. Skiena, S.S.: *The Algorithm Design Manual*. Springer (1997)