Bioinformatics Algorithms

Lecture Notes

a.a. 2020/2021

Alberto Mosconi
Politecnico di Milano

Contents

1 Definitions

1.1 Alphabet

Let Σ be a finite set of symbols (alternatively called characters), called the alphabet. No assumption is made about the nature of the symbols.

1.2 Strings

A string (or word) over Σ is any finite sequence of symbols from Σ. For example if $\Sigma = \{0, 1\}$ then 01011 is a string over Σ.

The **length of a string *S* is the number of symbols** in s (the length of the sequence) and can be any non-negative integer. It is often denoted as $|S|$.

The **empty string** is the unique string over Σ of length 0, and is denoted as ϵ.

The **set of all strings over $\boldsymbol{\Sigma}$ of length *n*** is denoted $\mathbf{\Sigma^n}$. For example , if $\Sigma = 0, 1$, then $\Sigma^2 = \{00, 01, 10, 11\}$. Note that $\Sigma^0 = \{\epsilon\}$ for any alphabet Σ.

The **set of all strings of any length over $\boldsymbol{\Sigma}$** is the *Kleene closure* of Σ and is denoted as $\boldsymbol{\Sigma}^*$. Also:

$$\Sigma^* = \bigcup_{n \in N \cup \{0\}} \Sigma^n$$

The **set of all non-empty strings over $\boldsymbol{\Sigma}$** is denoted by $\boldsymbol{\Sigma}^+$

1.3 Subsequences

A **subsequence** of S is a string that can be **obtained by deleting any number (from 0 to n) of non-consecutive characters** from S, including S and the empty string too.

1.3.1 Let us count: subsequences

A subsequence of S can be described by a binary string B of n elements, telling us for each position i if the corresponding character is kept in the subsequence (1) or deleted (0).

- S = apple, B = 11010 $\rightarrow$ subsequence = apl

Thus, the **number of subsequences** corresponds to the number of binary strings of length n, hence

$$subsequences = 2^n$$

Notice that if two subsequences produced by different choices are **identical, they will be counted twice**.

1.4 Substrings

A string s is said to be a **substring** (or *factor*) of t if there exist (possibly empty) strings u and v such that $t = usv$.

Given a string t, **suffixes** and **prefixes** are special substrings of t.

1.4.1 Let us count: substrings

Given a string of length n:

- We have 1 substring of length n (starting at 1)
- We have 2 substrings of length *n-1* (starting at 1 and 2)
- We have 3 substrings of length *n-2* (starting at 1,2 and 3)
- ...
- We have $n - i + 1$ substrings of length i (starting at all positions from 1 to $n - i + 1$)
- We have n substrings of length 1 (starting at every position)

Plus, 1 substring of length 0.
This brings the **total number of substrings** of a string of length n to

$$substrings = \sum_{i=1}^{n} i = \frac{n(n+1)}{2} + 1$$

1.5 Prefixes and suffixes

A string s is said to be a **prefix** of t if there exists a string u such that $t = su$. If u is nonempty, s is said to be a *proper* prefix of t.
Simmetrically, a string s is said to be a **suffix** of t if there exists a string u such that $t = us$. If u is nonempty, s is said to be a *proper* suffix of t.

1.5.1 Let us count: prefixes and suffixes

The number of all posssible prefixes/suffixes of a string of length n is

$$\textit{prefixes/suffixes} = n + 1$$

n are non-empty, +1 because we include also **the empty string** as prefix/suffix.

1.6 Reverse and palindrome

The **reverse** of a string is a string with the same symbols but in reverse order. For example, if $s = abc$ (where a, b, and c are symbols of the alphabet), then the reverse of s is cba.
A string that is the reverse of itself is called a **palindrome**, which also includes the empty string and all strings of length 1.

1.7 Rotations and permutations

A string $s = uv$ is said to be a **rotation** of t if $t = vu$. For example, if $\Sigma = \{0, 1\}$ the string *0011001* is a rotation of *0100110*, where u = *00110* and v = *01*.

1.7.1 Let us count: rotations

Given a string S, with $|S| = n$, we denote **S=AB**, where **A is a prefix** substring, and **B a suffix** substring.
For every possible pair A, B the string **R=BA is a rotation of S**, including S itself, that is, the case where $A = \epsilon$ and $B = S$.
Since we have n different non empty suffixes B, the number of rotations is

$$rotations = n$$

It is not *n + 1* because the two suffixes $B = S$ and $B = \epsilon$ produce exactly the same rotation, given by S itself.

1.7.2 Let us count: permutations

Given a string S, with $|S| = n$, the number of permutations of the characters in S is

$$permutations = n!$$

Notice that the formula does not consider the case of indentical permutations: that is, a string of n identical characters, will have n identical permutations.

2 Comparing strings

2.1 Hamming distance

The **Hamming distance** between **two strings of equal length** is the **number of positions at which the corresponding symbols are different**.

- “ka*rol*in” and “ka*thr*in” → 3
- 10*11*1*1*01 and 10*01*0*0*01 → 2

With Hamming distance we can formalize *substitutions* in biological sequences - or simply sequencing errors in which the wrong base pair is identified.

2.2 Edit distance

The **edit distance** is a way of quantifying how dissimilar two strings (e.g., words) are to one another by counting the **minimum number of operations required to transform on string into the other**.

In the *Levenshtein distance* (the most common), edit operations are: **removal**, **insertion**, and **substitution**.

The edit distance between “kitten” and “sitting” is 3. A minimal edit script that transforms the former into the latter is:

- **k**itten → **s**itten (substitute s for k)
- sitt**e**n → sitt**i**n (substitute i for e)
- sittin → sittin**g** (insert g at the end)

The number of solutions (sequences of operations) is infinite.