Sequence alignment

TDT4187 – Exercise 1 September 6, 2017

String terms

Given string $\omega = ABBACBDACC$,
and a set of strings $\mathbb{S} = \{ACCB, AB, BCDA, CBDA, ACC, BACA, ABBA, \omega\}$:

a) Which of the strings from $\mathbb{S}$ are substrings of ω?

b) Which of the strings from $\mathbb{S}$ are prefixes of ω?

c) Which of the strings from $\mathbb{S}$ are suffixes of ω?

d) Which of the strings from $\mathbb{S}$ are subsequences of ω?

1 Longest common subsequence (LCS)

1.1 Alignments

a) Compute the LCS for the following pair of strings:

$\omega_1 = ACGGTAC$ $\omega_2 = CTCGACT$

b) Determine the alignments, given the following dynamic programming (DP) tables:

	ε	C	T	A	A	G	C	C
ε	0	0	0	0	0	0	0	0
C	0	1	1	1	1	1	1	1
T	0	1	2	2	2	2	2	2
C	0	1	2	2	2	2	3	3
G	0	1	2	2	2	3	3	3
A	0	1	2	3	3	3	3	3
C	0	1	2	3	3	3	4	4
T	0	1	2	3	3	3	4	4

	ε	C	T	T	C	G	T	C
ε	0	0	0	0	0	0	0	0
C	0	1	1	1	1	1	1	1
T	0	1	2	2	2	2	2	2
C	0	1	2	2	3	3	3	3
G	0	1	2	2	3	4	4	4
A	0	1	2	2	3	4	4	4
C	0	1	2	2	3	4	4	5
T	0	1	2	3	3	4	5	5

	ε	C	T	A	A	C	T	C
ε	0	0	0	0	0	0	0	0
C	0	1	1	1	1	1	1	1
T	0	1	2	2	2	2	2	2
C	0	1	2	2	2	3	3	3
G	0	1	2	2	2	3	3	3
A	0	1	2	3	3	3	3	3
C	0	1	2	3	3	4	4	4
T	0	1	2	3	3	4	5	5

	ε	T	C	C	G	G	A	T
ε	0	0	0	0	0	0	0	0
C	0	0	1	1	1	1	1	1
T	0	1	1	1	1	1	1	2
C	0	1	2	2	2	2	2	2
G	0	1	2	2	3	3	3	3
A	0	1	2	2	3	3	4	4
C	0	1	2	3	3	3	4	4
T	0	1	2	3	3	3	4	5

c) How do we represent an alignment?

antonink@idi.ntnu.no

1.2 LCS as a graph problem

Lemma 1. Let ω_1, ω_2 be two strings for which we are to compute LCS, where $|\omega_1| = n$, $|\omega_2| = m$. Let $M[0..m, 0..n]$ be the DP table (matrix) we would compute using the standard algorithm.
Consider graph $G(V, E)$, where each vertex $v \in V$ corresponds to an entry in M. The edges E then correspond to _____
The LCS solution is than corresponds to the _____ path from vertex M_{00} to vertex M_{mn}.

a) Specify the edge set E. Determine $|E|$.

b) Is it shortest or longest path that we seek as solution?

c) What does the number at table entry $M[i, j]$ resemble for the aforementioned graph problem?

1.3 Backtracking the LCS

Consider the following algorithm for backtracking matrix M_{mn}:

```
Data: Dynamic programming matrix M[0..m, 0..n] for sequences $\omega_1$, $\omega_2$.
Result: Alignment [alignment] representing LCS for $\omega_1$, $\omega_2$.
1  (i,j) = (m,n);
2  alignment = emptyMatrix();
3  col = 0 while not (i,j) == (0,0) do
4      alignment[0..1, col] = (i,j);
5      if M[i,j] == M[i-1, j] then
6          i = i-1;
7      else if M[i,j] == M[i, j-1] then
8          j = j-1;
9      else
10         .... ;
11     end
12 end
13 alignment = reverseCols(alignment)
```

a) Fill in the code for line 10, so that the algorithm works correctly.

*b) Prove that the algorithm always outputs the correct alignment.

Note that the algorithm does needn't check the actual sequence characters. This makes it highly suitable for quick manual backtracking.

1.4 *Alternative recurrence relation

a) Could the following recurrence relation be used as basis for the LCS algorithm? Provide reasoning.

$$\text{LCS}(S_i, R_j) = \begin{cases} \max\{LCS(S_{i-1}, R_j), LCS(S_i, R_{j-1})\} & \text{if } s_i \neq r_j \\ \min\{LCS(S_{i-1}, R_j), LCS(S_i, R_{j-1})\} + 1 & \text{if } s_i = r_j \end{cases}$$

2 Recursion vs Dynamic Programming

Both the recursive and dynamic programming algorithm for LCS use the same recursive formula:

$$\text{LCS}(S_i, R_j) = \max \begin{cases} LCS(S_{i-1}, R_j) \\ LCS(S_i, R_{j-1}) \\ LCS(S_{i-1}, R_{j-1}) \end{cases}$$

a) What is the difference between the recurrent and dynamic programming algorithm?

b) Illustrate the difference by quantifying the number of comparisons made by recurrent and DP algorithm for strings of length 7. (Calculating max of three numbers takes 2 comparisons.)

c) Is the recursive algorithm polynomial in time/space?

2.1 Dynamic programming schema

The schema hereunder is a general description of a dynamic programming algorithm for problem $\mathcal{P}$.

I) Subproblems $\mathcal{S} = \{\mathcal{L}(i)\}$

II) Direct computation of base cases $\Omega(\mathcal{S}) \subseteq \mathcal{S}$

III) Recurrent computation of $\mathcal{L}(i) \in S \setminus \Omega(S)$

IV) Order in which $\mathcal{S}$ can be solved.

V) Solve $\mathcal{P}$ using solutions of subproblems

a) Identify phases I–V for the LCS algorithm.

*b) The order in IV has to be topological ordering w.r.t. the graph described in **Lemma 1**. Show why the ordering used in LCS is topological, and show an example of an ordering that is not topological.

3 Edit Distance

Hamming distance is defined as $d_{\mathrm{H}}(\omega_1, \omega_2) = |\{i \mid \omega_1[i] \neq \omega_2[i]\}|$.
Recall from lecture 2, that we defined Edit Distance (ED, or d_{ED}) as follows:

$$\mathrm{ED}(S_i, R_j) = \min \begin{cases} ED(S_{i-1}, R_j) + 1 \\ ED(S_i, R_{j-1}) + 1 \\ ED(S_{i-1}, R_{j-1}) + f(s_i, r_j) \end{cases} \qquad \text{f(s, r)} = \begin{cases} 0 & r = s \\ 1 & \text{otherwise} \end{cases}$$

a) Fill in the correct inequality sign to the following expression:

$d_{\mathrm{H}}(\omega_1, \omega_2) \quad ? \quad d_{\mathrm{ED}}(\omega_1, \omega_2)$

b) Can you reformulate the problem as a graph problem, just as we did for LCS in **Lemma 1**?

*c) Show that d_{ED} is a distance (metric) in mathematical sense. That is, show that the following conditions are each satisfied for any x, y :

i) $d_{\mathrm{ED}}(x, y) \geq 0$

ii) $d_{\mathrm{ED}}(x, y) = 0 \Leftrightarrow x = y$

iii) $d_{\mathrm{ED}}(x, y) = d_{\mathrm{ED}}(y, x)$

iv) $d_{\mathrm{ED}}(x, z) \leq d_{\mathrm{ED}}(x, y) + d_{\mathrm{ED}}(y, z)$

*d) Is $LCS : \Sigma * \times \Sigma * \to \mathbb{R}$ also a distance (metric)?

4 Local and Global alignment

Recall the recurrent relations for Global Alignment (GA) and Local Alignment (LA).

$$\mathrm{GA}(S_i, R_j) = \max \begin{cases} LA(S_{i-1}, R_j) + I \\ LA(S_i, R_{j-1}) + I \\ LA(S_{i-1}, R_{j-1}) + \delta(s_i, r_j) \end{cases}$$

$$\mathrm{LA}(S_i, R_j) = \max \begin{cases} LA(S_{i-1}, R_j) + I \\ LA(S_i, R_{j-1}) + I \\ LA(S_{i-1}, R_{j-1}) + \delta(s_i, r_j) \\ 0 \end{cases}$$

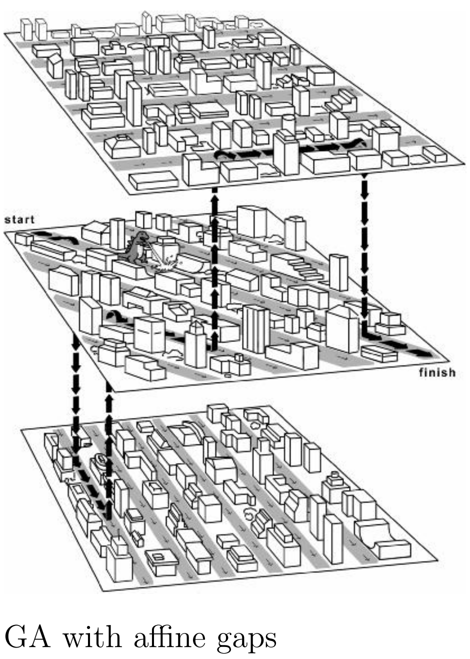
a) Compute GA and LA for sequences $\omega_1 = CTCTAGC$, $\omega_2 = CGGATAC$, with match score 1, mismatch -2, and indel penalty -2.

b) Compute GA with affine gaps for ω_1, ω_2, with match score 1, mismatch -2, gap open penalty -3, and gap extension penalty -1.

c) Fill in the correct inequality: $\mathrm{GA}(S_i, R_j) \quad ? \quad \mathrm{LA}(S_i, R_j)$

d) Interpret the GA, and LA problems as graph problems again, alike in **Lemma 1**.

*e) Create a backtracking algorithm for GA with affine gaps, that derives the correct alignment from the filled DP matrices. We require that, alike the algorithm in 1.3, the algorithm does not compare sequence letters (and thus only reads the matrix values).



GA with affine gaps
[Jones, Pevzer]