Bioinformatics Algorithms

(Fundamental Algorithms, module 2)

Zsuzsanna Lipták

Masters in Medical Bioinformatics
academic year 2017/18, spring term

$\boxed{\text{Pairwise Alignment 3}}$

Optimal pairwise alignment in linear space

Given two sequences s, t of length n:

- DP algorithm for global alignment: $O(n^2)$ time and space
- if we only want the score of an optimal alignment $sim(s, t)$ (problem variant 1), then we can do this in $O(n^2)$ time and $O(n)$ space (space-saving variant)
- But that algo does not give us the optimal alignment itself (problem variant 2)
- Now: algorithm for computing an optimal alignment itself in time $O(n^2)$ but space $O(n)$

There are several algorithms achieving this, e.g. Hirschberg (1975) aka Myers-Miller (1988). Here we present the divide-and-conquer algorithm from the book by Durbin, Eddy, Krogh, Mitchison (ch. 2.6).

$s = \texttt{GAAGA}$, $t = \texttt{CACA}$

match: 2, mismatch: -1, gap: -1

$D(i,j)$			C	A	C	A
		0	1	2	3	4
	0	0	-1	-2	-3	-4
G	1	-1	-1	-2	-3	-4
A	2	-2	-2	1	0	-1
A	3	-3	-3	0	0	2
G	4	-4	-4	-1	-1	1
A	5	-5	-5	-2	-2	1

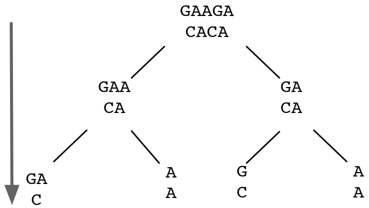
The optimal alignments are:

1. $\binom{\texttt{GAAGA}}{\texttt{-CACA}}$
2. $\binom{\texttt{GAAGA}}{\texttt{CA-CA}}$
3. $\binom{\texttt{GAAGA}}{\texttt{C-ACA}}$
4. $\binom{\texttt{GAAGA}}{\texttt{CAC-A}}$

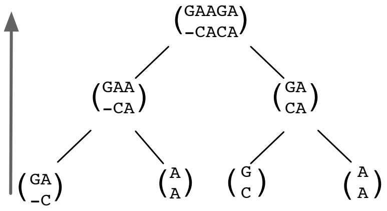
Consider the first optimal alignment $\binom{\texttt{GAAGA}}{\texttt{-CACA}}$:

Idea: Divide-and-conquer

We divide the two sequences s, t in two parts, left and right, align left with left, right with right, and then concatenate the two alignments:



top-down: split sequences into two

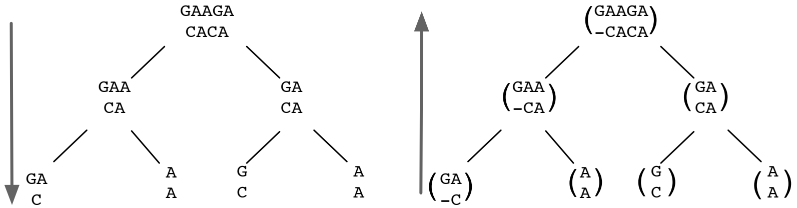


bottom-up: concatenate alignments

Consider the first optimal alignment $\binom{\texttt{GAAGA}}{\texttt{-CACA}}$:

Idea: Divide-and-conquer

We divide the two sequences s, t in two parts, left and right, align left with left, right with right, and then concatenate the two alignments:



top-down: split sequences into two *bottom-up: concatenate alignments*

Question

But how do we know where to divide them?

Definition

A cut is a pair of positions (n', m'), where $1 \leq n' \leq n$, and $1 \leq m' \leq m$ (with $|s| = n, |t| = m$).

1. In sequence 1, we will always take the middle cut position $n' = \lceil n/2 \rceil$.
2. In sequence 2, we will remember where the middle row $n' = \lceil n/2 \rceil$ was crossed.
3. For this, we will need to compute another matrix M (again, in space-saving manner!).

Matrix M

- Cell $M(i,j)$ contains, where $i \geq n'$, an index r s.t. there is an optimal alignment with score $D(i,j)$ passing through cell (n', r).
- Computation:
  $M(n', j) = j$ for all $j = 1, \ldots, m$;
  for $i > n', 0 \leq j \leq m$: $M(i,j) = M(i',j')$, where $D(i,j)$ derives from cell (i',j') (therefore $(i',j') \in \{(i-1,j),(i-1,j-1),(i,j-1)\}$) – if there is more than one, then choose one acc. to priority (e.g. *left-diag-top*)
- Then $M(n,m) = r$ s.t. there is an optimal alignment of s and t which passes through cell $(\lceil n/2 \rceil, r)$.
- Thus, we can use the cut $(n', r) = (\lceil n/2 \rceil, M(n,m))$ in the divide-step and recurse with $s_1 \ldots s_{n'}$ and $t_1 \ldots t_r$ on the left, and $s_{n'+1} \ldots s_n$ and $t_{r+1} \ldots t_m$ on the right.

Algorithm PWA(s,t)

```
1. if $\max(|s|, |t|) \leq 2$, then return an optimal alignment computed with
   N-W-algorithm
2. else
3.         compute DP-table $D$ row-wise, and
4.         from $i = \lceil n/2 \rceil$ on, compute also matrix $M$ (row-wise)
5.         return $PWA(s_1 \ldots s_{\lceil n/2 \rceil}, t_1 \ldots t_r)$ concatenated
              with $PWA(s_{\lceil n/2 \rceil + 1} \ldots s_n, t_{r+1} \ldots t_m)$.
```

(for a detailed example, see class notes)

Analysis

- Space: Since all matrix computations are row-wise, they all need linear space in m, and none need to be stored, thus $O(m)$; we need to store the partial alignments, whose total length is the length of the final alignment, thus $O(n+m)$: altogether space $O(n+m)$
- Time: In each iteration, we are exactly halving the problem size (wherever we cut t, string s is always cut in the middle), thus we get:

$$nm + \frac{1}{2}nm + \frac{1}{4}nm + \ldots \leq nm \sum_{k=1}^{\infty} \frac{1}{2^k} = 2nm \in O(nm).$$

Thus we doubled the time (asymptotically the same, both $O(nm)$), but reduced the space from quadratic to linear.