16 Sequence Alignment

- Sequence alignment
- Next: shortest paths in graphs with negative weights

16.1 Sequence Alignment

A way of comparing strings: line them up, possibly with gaps, to minimize the mismatches.

Example 16.1
Suppose we have strings $GACGTTA$ and $GAACGCTA$.

We could align them as follows:
GA_CGTTA
$GAACGCTA$

But this is not a perfect alignment, since there are mismatches and gaps.

To quantify the quality of an alignment, we can define a **gap penalty** δ and a **mismatch penalty** α_{xy} for symbols x and$y$, with the total cost being the sum of the penalties.

Our goal is to find the alignment of lowest cost.

Example 16.2
GA_CG_TTA
$GAACGCT\_A$
Here, the cost is 3δ.

Definition 16.3
Given strings $x \in \Sigma^n$ and $y \in \Sigma^m$ (Σ being an alphabet), a **matching** of sets I and J is a set of ordered pairs (i, j) with $i \in I$, $j \in J$ such that each $i \in I$ and $j \in J$ appears at most once (some can be unpaired).

Definition 16.4
A matching M of $\{1, \cdots, n\}$ and $\{1, \dots, m\}$ is an **alignment** if there are no "crossings", i.e., if $(i, j) \in M$ and $(i', j') \in M$ with $i < i'$, then $j < j'$.

Example 16.5
$M = \{(1, 3), (2, 2)\}$ is a matching but not an alignment.

16.1.1 Dichotomy for defining subproblems

Are the last two symbols matched? i.e., is $(n, m) \in M$?

If not, then we know something more: either the last symbol of x or the last symbol of y must be unpaired. This is because if both x_n and y_m are matched, but not to each other, then there must be a crossing.

Subproblem: sequence alignment of $x_1 x_2 \cdots x_i$ with $y_1 y_2 \cdots y_j$.

Let $\text{opt}(i, j)$ be the minimum cost of an alignment between these substrings.

Is x_i aligned with y_j?

- If yes, then $\text{opt}(i, j) = \alpha_{x_i y_j} + \text{opt}(i - 1)(j - 1)$.
- If no, then is x_i unmatched?

  - If yes, then $\text{opt}(i, j) = \delta + \text{opt}(i - 1, j)$
  - If no, then y_j must be unmatched, so $\text{opt}(i, j) = \delta + \text{opt}(i, j - 1)$.

In general, we have

$$\text{opt}(i, j) = \min\{\alpha_{x_i y_j} + \text{opt}(i - 1, j - 1), \delta + \text{opt}(i - 1, j), \delta + \text{opt}(i, j - 1)\}$$

We define $\text{opt}(i, 0) = i \cdot \delta$, and $\text{opt}(0, j) = j \cdot \delta$.

```
Alignment(x, y):
    let A be an (n+1)x(m+1) array indexed by
        i in {0, 1, ..., n} and j in {0, 1, ..., m}
    let A[i, 0] = i * delta for all i in {0, 1, ... n}
    let A[0, j] = j * delta for all j in {0, 1, ... m}
    for i=1, ..., n
        for j=1, ..., m
            let A[i, j] = min{alpha_xiyj + A[i-1, j-1],
              delta + A[i-1, j], delta + A[i, j-1]}
        endfor
    endfor
    return A[n, m]
```