Biologists use genetic similarity to determine evolutionary relationships. But how do we say if two gene sequences are similar or not? We align them. Also used in spell-checkers and search engines.

Example. TAIL vs TALE

For two strings $X = x_1x_2...x_m, Y = y_1y_2...y_n$, an alignment $M$ is a matching between $\{1, ..., m\}$ and $\{1, ..., n\}$.

$M$ is valid if

- Matching. Each element appears in at most one pair in $M$.
- No crossings. If $(i, j), (k, \ell) \in M$ and $i < k$, then $j < \ell$.
- Cost of $M$:
  - Gap penalty. For each unmatched character, you pay $\delta$.
  - Alignment cost. For a match $(i, j)$, you pay $C(x_i, y_j)$.

$$\text{cost}(M) = \delta(m + n - 2|M|) + \sum_{(i, j) \in M} C(x_i, y_j).$$

Problem. Given strings $X, Y$ gap-penalty $\delta$ and cost matrix $C$, find valid alignment of minimal cost.

Example 1. TAIL vs TALE, $\delta = 0.5$, $C(x, y) = 1[x \neq y]$. Example 2. TAIL vs TALE, $\delta = 10$, $C(x, y) = 1[x \neq y]$.

Toward an algorithm

Try what we did before: Let $O$ be optimal alignment.

- If $(m, n) \in O$ we can align $x_1x_2...x_{m-1}$ with $y_1y_2...y_{n-1}$.
- If $(m, n) \notin O$ then either $x_m$ or $y_n$ must be unmatched (by no crossing).

Value $\text{OPT}(m, n)$ of optimal alignment is either:

- $C(x_m, y_n) + \text{OPT}(m - 1, n - 1)$, if $(m, n)$ matched
- $\delta + \text{OPT}(m - 1, n)$, if $m$ unmatched
- $\delta + \text{OPT}(m, n - 1)$, if $n$ unmatched

Let $\text{OPT}(i, j)$ be cost of optimal alignment of $x_1x_2...x_i$ and $y_1y_2...y_j$.

$$\text{OPT}(i, j) = \min \left\{ \begin{array}{ll} C(x_i, y_j) + \text{OPT}(i - 1, j - 1) \\ \delta + \text{OPT}(i - 1, j) \\ \delta + \text{OPT}(i, j - 1) \end{array} \right\}$$

And, $(i, j)$ is in optimal alignment iff first term is the minimum.
Sequence Alignment pseudocode

```plaintext
align(X,Y)
    Initialize M[0..m,0..n] = null.
    M[0,0] = iδ, M[0,j] = jδ for all i, j.
    for j = 1, ..., n do
        for i = 1, ..., m do
            v1 = $C(x_i, y_j) + M[i-1, j-1]$.
            v2 = $\delta + M[i-1, j]$.
            v3 = $\delta + M[i, j-1]$.
            M[i,j] ← min{v1, v2, v3}.
        end for
    end for
Example. TALE and TAIL, $\delta = 1, C(x,y) = 2 \cdot 1[x \neq y]$.
```

Sequence Alignment

- Running time is $O(mn)$.
- Computing optimal matching is easy.
- Related to shortest path in weighted directed graph.

![Shortest Path Diagram]