But: Aho-Corasick creates search term tree of size $\mathcal{O}(m)$, $m := \sum_{1 \leq i \leq N} |P_i|$, in time $\mathcal{O}(m)$, and searches in time $\mathcal{O}(n)$, $n := |T|$.
Suffix tree has size $\mathcal{O}(n)$, construction time $\mathcal{O}(n)$ and search time $\mathcal{O}(m)$.
$\Rightarrow$ If all P_i together are longer than the text, the suffix tree solution needs less space but more time (preprocessing ignored). If the set of the strings is shorter than the text, the Aho-Crasick needs less space but more time.
$\Rightarrow$ We observe a place/time-*trade-off*, as no solution is superior in place and time consumption at the same time.

Substring Problem for a Set of Texts:

Look for string P in a set of texts $\mathcal{T} := \{T_i \mid 1 \leq i \leq N\}$.
Example: Newly sequenced DNA fragment (P) is to be searched amongst DNA sequences in the database ($\mathcal{T}$) (Identification by mitochondrial DNA).

Generalized compact suffix tree: compact suffix tree for text $T' = T_1\$_1 T_2\$_2 \cdots T_N\$_N$ (all $\$_i$ different).
Leaves are labeled with pairs (text, position), symbols following an end mark are removed (as the $\$_i$ are different, this only happens at leaf edges).

So: Create generalized suffix tree for T' in time and space $\mathcal{O}(|T'|)$ and traverse along P to solve the substring problem for the given set of texts.

Dead end $\Leftrightarrow$ P not contained.

Leaf $\Leftrightarrow$ P contained exactly once (label identifies text and position).

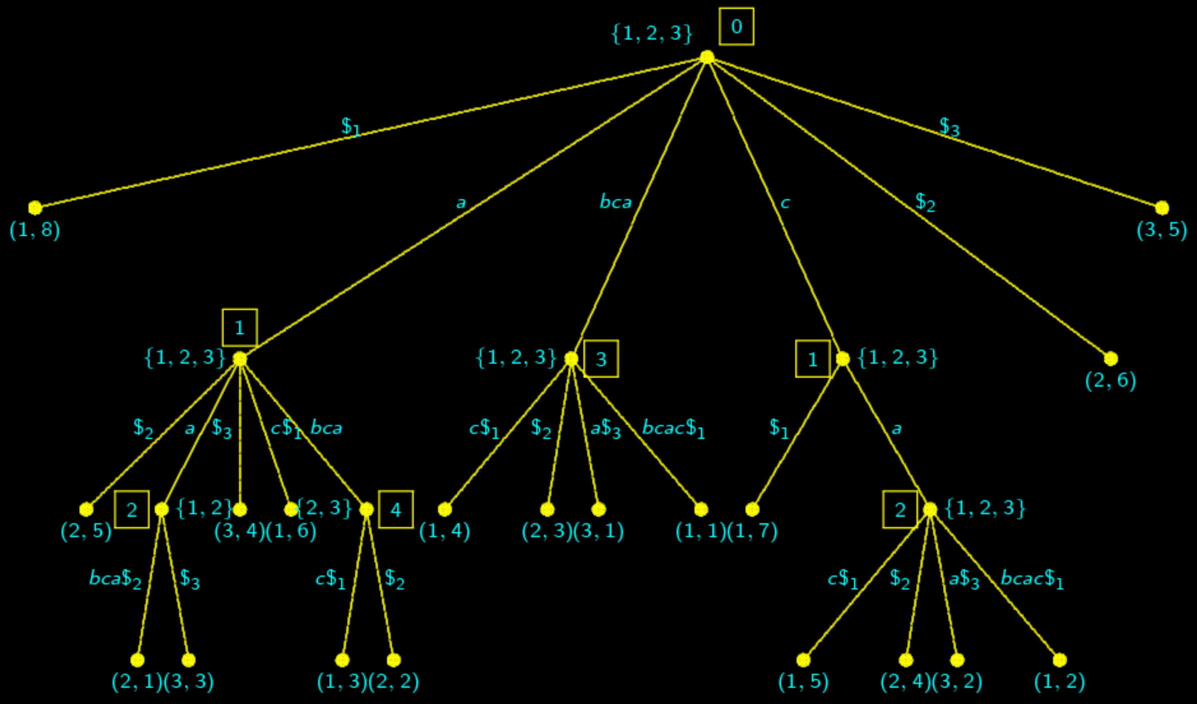
Internal node $\Leftrightarrow$ P contained multiple times (visit all leaves in time proportional to number of occurrences).

Longest Common Substring:

Search a longest substring, common to all words in $\mathcal{T} := \{T_i \mid T_i \in \Sigma^\star, 1 \leq i \leq N\}$.
Example: Identify important and thus in related organisms *mutationless* regions of DNA.
Solution: construct generalized suffix tree for $\mathcal{T}$.
Inner node x corresponds to

- prefix of a suffix of only one of the T_i (all leaves in the subtree with root x belong to the same T_i), or
- prefix of a suffix of multiple texts (leaves in the subtree with root x belong to different T_i).

Thus we label each internal node x with the set of text indices appearing in the subtree with root x (This can e.g. be achieved by traversing the tree in postorder and labeling each node with the union of the labels of its children).
Now only such nodes are solution candidates that are labeled with the complete set $\{1, 2, \ldots, N\}$ (the solution has to appear in each of the T_i).
From these nodes we chose one with maximum *string depth*, i.e. a node y which has a path-label α of maximum length. This can be done with a tree traversal.
Now the searched substring is given by α.



Repeats in Words:

Definition

Let $T \in \Sigma^n$ and $P \in \Sigma^m$, $0 < m \leq n$ and let $T_0 := \$ =: T_{n+1}$, $\$ \notin \Sigma$. P is called an exact repeat *of T, if and only if*

$$(\exists i, j \in [0 : n-1])(i \neq j \wedge P = T_{i+1,i+m} = T_{j+1,j+m}).$$

If additionally $T_i \neq T_j$ and $T_{i+m+1} \neq T_{j+m+1}$, P is called a maximum repeat *in T.*

This definition does not rule out the possibility of several different maximum repeats starting at the same position of T.

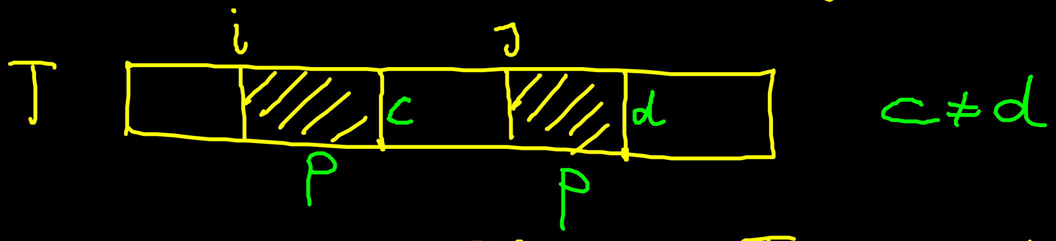
Example: In $T = aabcbabacabcc$, ab (because of $a\mathbf{ab}cb\mathbf{ab}acabcc$) and abc (because of $a\mathbf{abc}babac\mathbf{abc}c$) are maximum repeats, one of which starts at T_2.

Lemma

Let T be a word given as compact suffix tree and P a maximum repeat in T. Then there exists an internal node x with path label P in the tree.

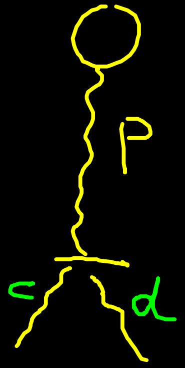
Note that this lemma implies a maximum number of $n-1$ maximum repeats in $T \in \Sigma^n$.

Beweis: P max. repeat ⇝ $\exists$ mind. zwei verschiedene Positionen in T, an denen P beginnt.



$\Rightarrow \exists$ zwei verschiedene Suffixe von T die mit P beginnen

Im Baum:



□

Definition

Given a compact suffix tree t for text T an internal node x of t is called left divers*, if the subtree with root x contains two leaves with labels i and j for which $T_{i-1} \neq T_{j-1}$ holds (Remember that the label of the leaves denotes the position in the text where the respective suffix starts).*

Theorem

Let $T \in \Sigma^n$ be given as compact suffix tree. Then $P \in \Sigma^+$ is a maximum repeat in T, if and only if traversal along P leads to a left divers internal node.

As a node is left divers if and only if it has at least one left divers child the left divers nodes can be determined in linear time starting at the fathers of the leaves.
Thus we have found an efficient procedure to find the maximum repeats of a text.

Alignments

Motivation: Data in computational biology is gained from experiments $\Rightarrow$ flawed data.

To e.g. search for genes we must not look for exact repeats but multiple occurences of similar words.

How can the similarity of words be formalized?

We start with the case of two words.

Pairwise alignments

Consider words *Monkey* and *Money*. These are obviously similar as

$$\begin{array}{cccccc} M & o & n & k & e & y \\ M & o & n & - & e & y \end{array}$$

Consider *Money* and *Honey*:

$$\begin{array}{ccccc} M & o & n & e & y \\ H & o & n & e & y \end{array}$$

We call these constructions an alignment of the words involved.

Definition

Let $S \in \Sigma^m$ and $T \in \Sigma^n$ and let $- \notin \Sigma$ be a so called gap symbol. Furthermore let $\Sigma' := \Sigma \cup \{-\}$ and $h : (\Sigma')^\star \to \Sigma^\star$ the homomorphism induced by $h(a) = a$ for $a \in \Sigma$ and $h(-) = \varepsilon$. An alignment of S and T is a pair (S', T') of words over Σ'satisfying the following conditions:

1. $|S'| = |T'| := l \geq \max(m, n)$,
2. $h(S') = S \wedge h(T') = T$,
3. $(\nexists i \in [1 : l])(T'_i = S'_i = -)$.

We take our descriptive representation as $2 \times l$-matrix over $\Sigma \cup \{-\}$.
In this matrix the following kinds of columns may occur:

Insertion: The upper word has a gap $-$ in the column.
Deletion: The lower word has a gap $-$ in the column.
Match: Both symbols in the column are the same
Substitution: None of the words has a gap in the column and the symbols do not match.

$\Rightarrow$ We may consider an alignment as generating process creating the lower word from the upper one.

Definition

Let S and T two words over Σ, let $p : \Sigma \times \Sigma \to \mathbb{Q}$ and $g \in \mathbb{Q}$. The scoring δ of an alignment (S', T') of length l is defined by column at first: For $x, y \in \Sigma$ let $\delta(x, y) := p(x, y)$ and $\delta(-, y) = \delta(x, -) := g$. The scoring of the complete alignment is then given by the sum of the scorings of its columns

$$\delta(S', T') := \sum_{1 \leq i \leq l} \delta(S'_i, T'_i).$$

Furthermore a scoring δ is always provided with an optimisation goal $goal_\delta \in \{min, max\}$.

For the function p usually $p(a, b) = p(b, a)$ for all $(a, b) \in \Sigma^2$ is required.

Definition

Let S and T two words over Σ and δ an alignment scoring. The similarity $sim_\delta(S, T)$ of S and T wrt. δ is the scoring of an optimal alignment of S and T, i.e.

$$sim_\delta(S, T) := goal_\delta\{\delta(S', T') \mid (S', T') \text{ is alignment of } S \text{ and } T\}.$$

If the choice of δ is obvious from context we will omit the index.

How should p, g and the optimization goal be chosen?

Edit distance: Counts the number of insertions, deletions and substitutions needed at minimum to transfer one word into the other.
$\Rightarrow$ $p(a, b) = 1$ for $a \neq b$, $p(a, a) = 0$ and $g = 1$ with optimization goal min.

Alternative: Let $p(a, a) = 1$, $p(a, b) = -1$ for $a \neq b$, and $g = -2$, maximizing.

Global alignments

We talk of *global alignments* if we consider the similarity of two words. On the opposite *local alignments* are used to find similar substrings.

Here: Solution by dynamic programming from Needleman and Wunsch (1970). We assume optimization goal *max*.

If $S \in \Sigma^m$ and $T \in \Sigma^n$ are given and ε is counted as a prefix, there are $m+1$ possible prefixes of S and $n+1$ possible prefixes of T.

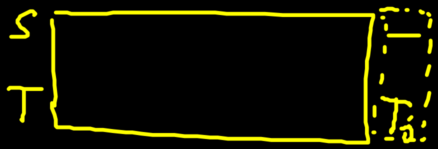
$\Rightarrow$ Create $(m+1) \times (n+1)$ matrix M of similarities of all pairs of prefixes.
Entry (we assume line numbers and column numbers starting at 0) $M_{i,j}$ represents the scoring of an optimal alignment of $S_{0,i}$ and $T_{0,j}$.
$M_{m,n}$ is the scoring of the global alignment.

The scoring of an optimal alignment of $S_{0,i}$ and the prefix ε of T is obviously $sim_\delta(S_{0,i}, \varepsilon) = g \cdot i$, $0 \leq i \leq m$, (i deletions for $S_{0,i} \rightarrow \varepsilon$).
Analogous the optimal alignment of ε and $T_{0,j}$ is rated $sim_\delta(\varepsilon, T_{0,j}) = g \cdot j$, $0 \leq j \leq n$.

Dynamic programming: Find the scoring of an optimal alignment of $S_{0,i}$ and $T_{0,j}$ assuming the scorings of optimal alignments of all pairs of shorter prefixes are known.

Two possibilities for the last column of the alignment of $S_{1,i}$ and $T_{1,j}$:

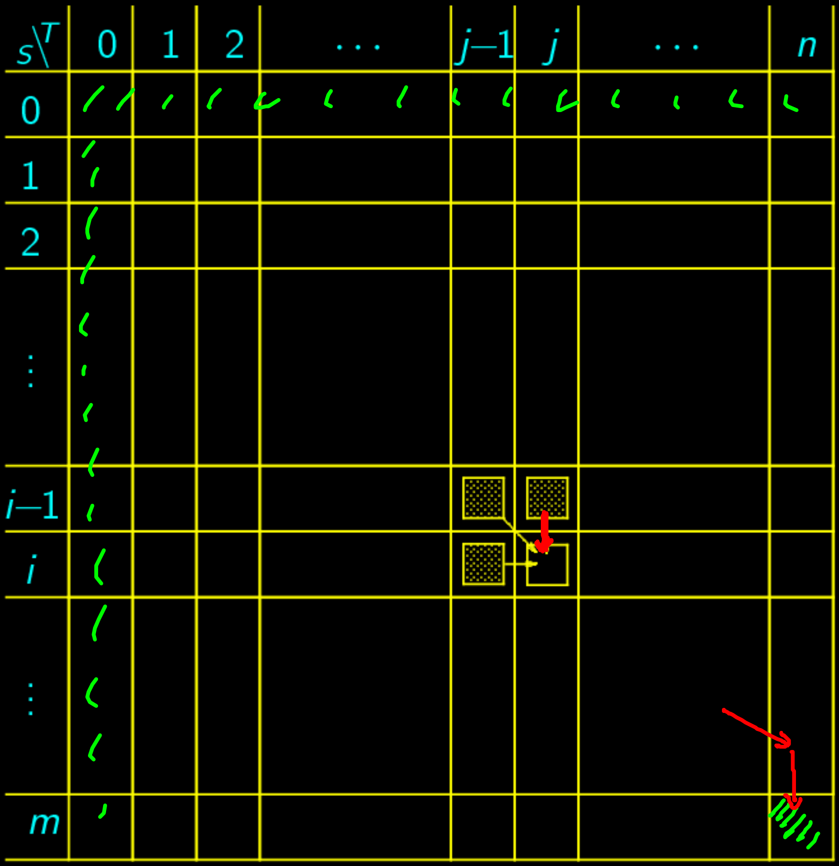
- It consists of symbols S_i and T_j or
- exactly one of the rows ends with the gap symbol $-$.



In each of the cases the scoring results from the scoring of an alignment of shorter prefixes plus the scoring of the last column:

$$sim_\delta(S_{1,i},\, T_{1,j}) = \max \begin{cases} \underbrace{sim_\delta(S_{1,i-1},\, T_{1,j}) + g}_{\text{deletion}}, \\ \underbrace{sim_\delta(S_{1,i},\, T_{1,j-1}) + g}_{\text{insertion}}, \\ \underbrace{sim_\delta(S_{1,i-1},\, T_{1,j-1}) + p(S_i,\, T_j)}_{\text{match/substitution}}. \end{cases} \tag{1}$$

First row and first column already initialized.
$\Rightarrow$ M can be filled by row or by column since we only need $M_{i,j-1}$, $M_{i-1,j}$ and $M_{i-1,j-1}$ to determine $M_{i,j}$.



Each path through M starting at $M_{0,0}$ and ending in $M_{m,n}$ and only stepping to the neighbors to the right, below or to the right and below corresponds to an alignment of S and T.

A step

- to the right corresponds to an insertion,
- down corresponds to a deletion,
- down and to the right corresponds to a match or substitution.

This observation can be used to determine the optimal alignment computed in $M_{m,n}$:
In each cell of M store, which of the alternatives from equation (1) contributed the maximum. If there is no unique source of the maximum any of the sources can be chosen.

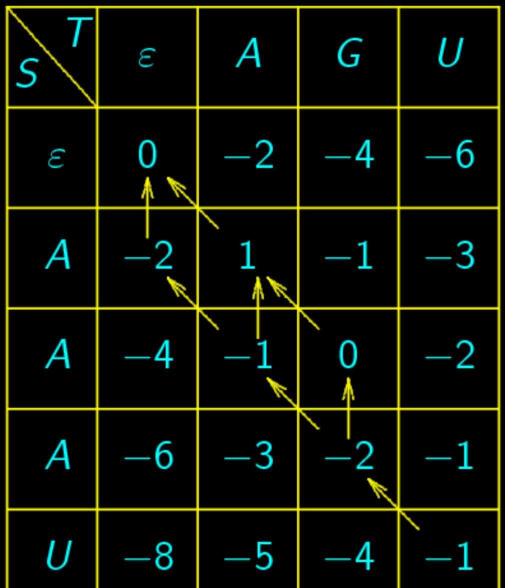
Starting in $M_{m,n}$ we traverse M along the path saved this way.

The alignment is generated from right to left, a step

- to the left corresponding to an insertion,
- up corresponding to a deletion,
- up and to the left corresponding to a match or substitution depending on if the symbols match.

Running time: $\Theta(m \cdot n)$.

Example: $S = AAAU$ and $T = AGU$, $p(a, a) = 1$, $p(a, b) = -1$ for $a \neq b$ and $g = -2$.



$S \backslash T$	ε	A	G	U
ε	0	-2	-4	-6
A	-2	1	-1	-3
A	-4	-1	0	-2
A	-6	-3	-2	-1
U	-8	-5	-4	-1

Similarity: $M_{4,3} = -1$, three alignments with similarity -1:

$$\begin{matrix} A & A & A & U \\ - & A & G & U \end{matrix}$$

$$\begin{matrix} A & A & A & U \\ A & - & G & U \end{matrix}$$

$$\begin{matrix} A & A & A & U \\ A & G & - & U \end{matrix}.$$

Note that there may be exponentially many optimal alignments (e.g. for $S = A^{2n}$ and $T = A^n$ there are $\binom{2n}{n}$ optimal alignments), so it is not recommended to output all solutions in an algorithm.

Remark: M may also be represented as a graph, the so-called *edit graph* $G = (V, E)$:

- Vertex = Entry in M ($V = \{0, \dots, m\} \times \{0, \dots, n\}$);
- Edges = Dependencies of the nodes according to equation (1) (Edge $u \longrightarrow v \Leftrightarrow v$ needs u to be computed).

Labeling: Edges $(i, j) \longrightarrow (i + 1, j)$ and $(i, j) \longrightarrow (i, j + 1)$ with g, Edges $(i, j) \longrightarrow (i + 1, j + 1)$ with $p(S_{i+1}, T_{j+1})$.

$\Rightarrow$ Construction of an optimal alignment $\Leftrightarrow$ Searching a path with maximum weight.

Local and semiglobal alignments

Definition

Let $S \in \Sigma^m$ and $T \in \Sigma^n$ and an alignment scoring δ with optimization goal maximization be given. A local alignment of S and T is a (global) alignment of substrings $\bar{S} := S_{i_1,i_2}$ and $\bar{T} := T_{j_1,j_2}$. An alignment $A := (\bar{S}', \bar{T}')$ of substrings $\bar{S}$ and $\bar{T}$ is an optimal local alignment of S and T wrt. δ, if

$$\delta(A) = \max\{sim_\delta(\bar{S}, \bar{T}) \mid \bar{S} \textit{ is substring of } S \wedge \bar{T} \textit{ is substring of } T\}.$$

Application: Comparison of unknown DNA or protein sequences. (Often in such sequences only substrings are similar.)
No minimizing!

Example: $S = AAAAACUCUCUCU$, $T = GCGCGCGCAAAAA$, matches rated $+1$, substitutions rated -1 and gaps rated -1. In this case the optimal local alignment is

$$\begin{array}{cccccccccccccccccccccc} & & & & & & & & A & A & A & A & A & (C & U & C & U & C & U & C & U) \\ (G & C & G & C & G & C & G & C) & A & A & A & A & A & & & & & & & & \end{array},$$

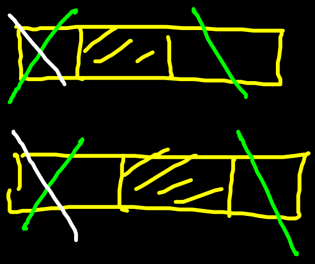
with scoring 5. (Strong emphasis of the parts in which both words match.) The optimal global alignment

$$\begin{array}{ccccccccccccc} A & A & A & A & A & C & U & C & U & C & U & C & U \\ G & C & G & C & G & C & G & C & A & A & A & A & A \end{array}$$

on the other hand is not very significant.

Smith and Waterman 1981: Compute an $(m+1) \times (n+1)$ matrix M, where $M_{i,j}$ gives the maximal scoring of an alignment of a suffix of $S_{1,i}$ and a suffix of $T_{1,j}$:

$$M_{i,j} = \max \begin{cases} M_{i-1,j} + g, \\ M_{i,j-1} + g, \\ M_{i-1,j-1} + p(S_i, T_j), \\ 0, \quad \longleftarrow \end{cases}$$



The first line and first column of M (corresponding to the empty word) can obviously be initialized with 0 as an alignment of the empty word with the empty word with scoring 0 is always possible.

The scoring of an optimal local alignment is now given by the maximum entry of M. The indices of this entry give the end positions of the alignment in S and T.

Edit graph: Add edges with weight 0 from vertex $(0,0)$ to every other vertex and from every other vertex to (m,n). Again we search a maximal path from $(0,0)$ to (m,n).

Semiglobal alignment: Global alignment with *free* gaps at beginning or end.

There are several variants depending on the location where gaps are free (beginning, end or both). If e.g. gaps at the end of one word and at the beginning of the other word are free we can find an approximative maximal overlapping of the words.
From an algorithmic point of view all variants can be reduced to global alignments.
We will now discuss the different cases for $S \in \Sigma^m$ and $T \in \Sigma^n$.

Free gaps at the end of S: If an alignment (S', T') of S and T of length l contains gaps to the right of symbols S_m, there exists $1 \leq j < l$ satisfying $T'_j = S_m$ and $S'_{j+1,l}$ contains only gaps.
If gaps at the end of S are rated 0 their consideration leaves the scoring of the alignment unchanged. Thus it is sufficient to find the best alignment of S and a prefix of T. Such alignments are rated in the last row of M in our algorithm.
Hence the scoring of the best semiglobal alignment of S and T is given by the maximum entry in the last row of M.

Free gaps at the end of T: Analogous to the above considerations we find the maximum entry in the last column of M to give the scoring searched for.

Free gaps at the beginning of S: As gaps at the beginning of S do not affect the scoring this case corresponds with an optimal alignment of S and a suffix of T.
By initializing the first row of M with 0 our method finds the searched scoring since instead of scoring the alignment of the prefix ε of S with $T_{0,j}$ with $g \cdot j$ we can now ignore the first symbols of T free of charge.
The algorithm will then choose the 0 in $M_{0,j}$ if $T_{j+1,n}$ matches S best.

Free gaps at the beginning of T: Analogous to the previous case we initialize the first column of M with 0.

Generalized scoring functions

Scoring of gaps: When considering biological sequences an alignment having several gaps in one block should be rated better than one having the same number of gaps spread around.

Definition

Let S and T words and let (S', T') an alignment of S and T. A substring $S'_{i+1,i+k} = -^k$ with $S'_i \neq - \neq S'_{i+k+1}$ (resp. a substring $T'_{j+1,j+k} = -^k$ with $T'_j \neq - \neq T'_{j+k+1}$) is called gap of length k.

Affine gap scoring: Rate gaps of length k by $-(\rho + \sigma k)$ instead of $k \cdot g$, $\rho, \sigma > 0$ chosen appropriate.
Here ρ penelizes general existence of a gap, σ gives a contribution proportional to gap length.

Affine gap scoring can be evaluated using dynamic programming, the recursions get however much more complicated.