Introduction to Algorithms

Dynamic Programming

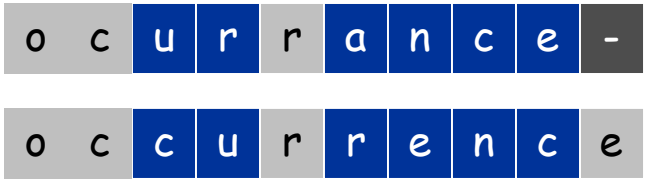
Sequence Alignment

Word Alignment

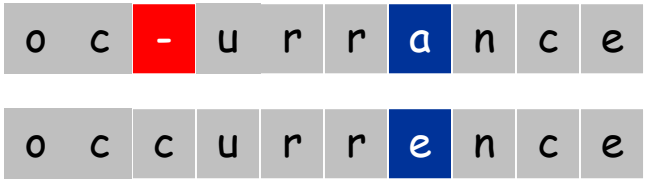
How similar are two strings?

ocurrance

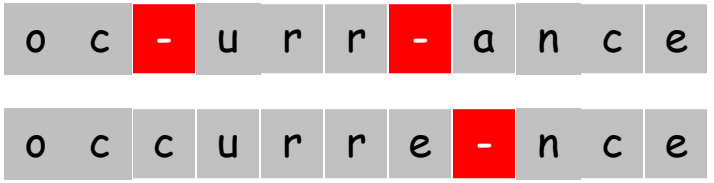
occurrence



5 mismatches, 1 gap



1 mismatch, 1 gap



0 mismatches, 3 gaps

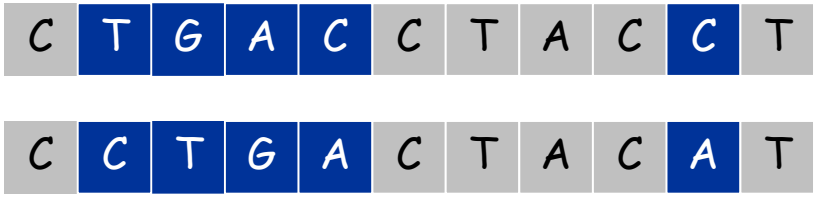
Edit Distance

Edit distance. [Levenshtein 1966, Needleman-Wunsch 1970]

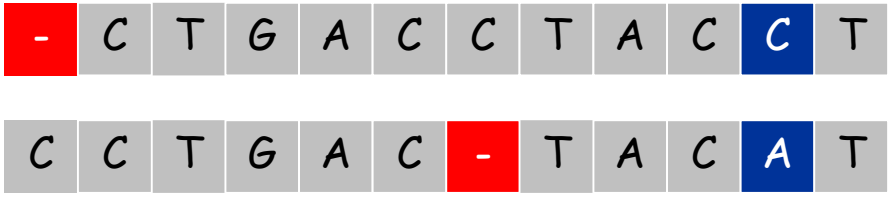
Cost = # of gaps + #mismatches.

Applications.

- Basis for Unix diff and Word correct in editors.
- Speech recognition.
- Computational biology.



Cost: 5



Cost: 3

Sequence Alignment

Given two strings $x_1, \cdots, x_m$ and $y_1, \cdots, y_n$ find an alignment with minimum number of mismatch and gaps.

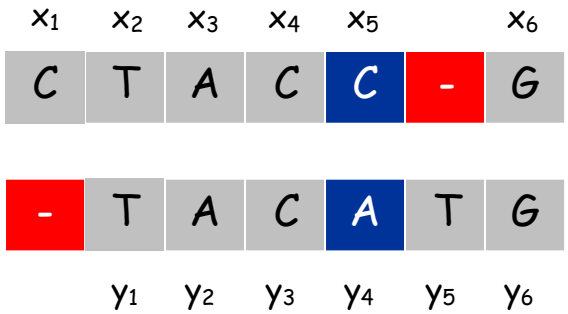
An alignment is a set of ordered pairs $(x_{i_1}, y_{j_1}), (x_{i_2}, y_{j_2}), \cdots$ such that $i_1 < i_2 < \cdots$ and $j_1 < j_2 < \cdots$

Example: CTACCG vs. TACATG.

Sol: We aligned

x_2-y_1, x_3-y_2, x_4-y_3, x_5-y_4, x_6-y_6.

So, the cost is 3.



DP for Sequence Alignment

Let $OPT(i, j)$ be min cost of aligning $x_1, \cdots, x_i$ and $y_1, \cdots, y_j$

Case 1: OPT matches x_i , y_j

- Then, pay mis-match cost if $x_i \neq y_j$ + min cost of aligning x_1 , $\cdots$, x_{i-1} and $y_1, \cdots, y_{j-1}$ i.e., $OPT(i-1, j-1)$

Case 2: OPT leaves x_i unmatched

- Then, pay gap cost for x_i + $OPT(i-1, j)$

Case 3: OPT leaves y_j unmatched

- Then, pay gap cost for y_j + $OPT(i, j-1)$

Bottom-up DP

NEEDLEMAN-WUNSCH(T, S)

```
1:  for i = 0 to m do
2:      OPT[i, 0] = −3 ∗ i;
3:  end for
4:  for j = 0 to n do
5:      OPT[0, j] = −3 ∗ j;
6:  end for
7:  for j = 1 to n do
8:      for i = 1 to m do
9:          OPT[i, j] = max{OPT[i − 1, j − 1] + s(T_i, S_j), OPT[i −
            1, j] − 3, OPT[i, j − 1] − 3};
10:     end for
11: end for
12: return OPT[m, n];
```

初始化与空字符串的匹配

Analysis: $\Theta(mn)$ time and space.

English words or sentences: m, n $\leq$ 10,..,20.

Computational biology: m = n = 100,000. 10 billions ops OK, but 40GB array?

T: \ S:	''	O	C	U	R	R	A	N	C	E
''	0	−3	−6	−9	−12	−15	−18	−21	−24	−27
O	−3	1	−2	−5	−8	−11	−14	−17	−20	−23
C	−6	−2	2	−1	−4	−7	−10	−13	−16	−19
C	−9	−5	−1	1	−2	−5	−8	−11	−12	−15
U	−12	−8	−4	0	0	−3	−6	−9	−12	−13
R	−15	−11	−7	−3	1	1	−2	−5	−8	−11
R	−18	−14	−10	−6	−2	2	0	−3	−6	−9
E	−21	−17	−13	−9	−5	−1	1	−1	−4	−5
N	−24	−20	−16	−12	−8	−4	−2	2	−1	−4
C	−27	−23	−19	−15	−11	−7	−5	−1	3	0
E	−30	−26	−22	−18	−14	−10	−8	−4	0	4

OCC
OC_

Score: OPT("OCCURRENCE", "OCURRANCE") = max { OPT("OCCURRENC", "OCURRANCE") −3 (=-3); OPT("OCCURRENC", "OCURRANC") +1 (=4); OPT("OCCURRENCE", "OCURRANC") −3 (=-3) }

Alignment:
S'= O-CURRANCE
T'= OCCURRENCE