Practical Bioinformatics

Mark Voorhies

6/23/2010

Exercise: Scoring an ungapped alignment

```
s = {"A": {"A":  1.0, "T": -1.0, "G": -1.0, "C": -1.0},
     "T": {"A": -1.0, "T":  1.0, "G": -1.0, "C": -1.0},
     "G": {"A": -1.0, "T": -1.0, "G":  1.0, "C": -1.0},
     "C": {"A": -1.0, "T": -1.0, "G": -1.0, "C":  1.0}}
```

$$S(x, y) = \sum_{i}^{N} s(x_i, y_i)$$

1. Given two equal length sequences and a scoring matrix, return the alignment score for a full length, ungapped alignment.

Exercise: Scoring an ungapped alignment

```
s = {"A": {"A":  1.0, "T": -1.0, "G": -1.0, "C": -1.0},
     "T": {"A": -1.0, "T":  1.0, "G": -1.0, "C": -1.0},
     "G": {"A": -1.0, "T": -1.0, "G":  1.0, "C": -1.0},
     "C": {"A": -1.0, "T": -1.0, "G": -1.0, "C":  1.0}}
```

$$S(x, y) = \sum_{i}^{N} s(x_i, y_i)$$

1. Given two equal length sequences and a scoring matrix, return the alignment score for a full length, ungapped alignment.
2. Given two sequences and a scoring matrix, find the offset that yields the best scoring ungapped alignment.

Exercise: Scoring a gapped alignment

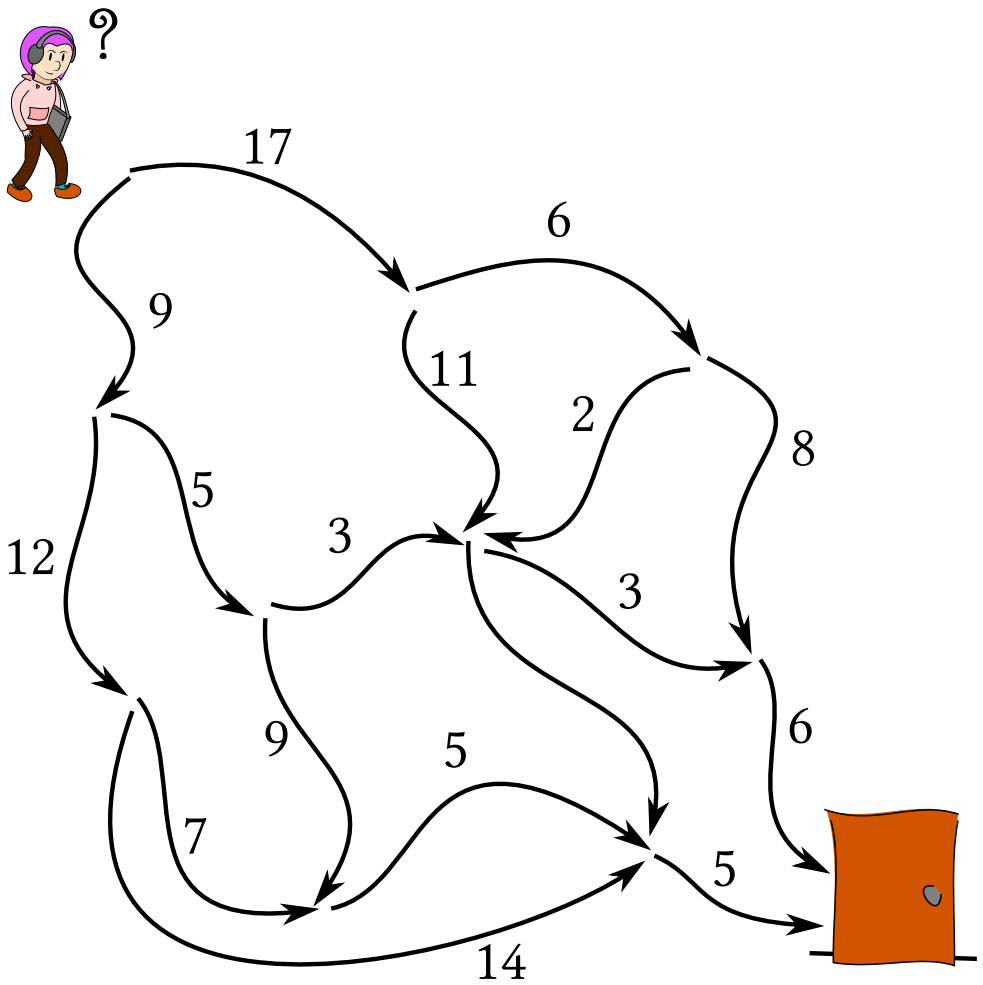
1. Given two equal length gapped sequences (where "-" represents a gap) and a scoring matrix, calculate an alignment score with a -1 penalty for each base aligned to a gap.

Exercise: Scoring a gapped alignment

1. Given two equal length gapped sequences (where "-" represents a gap) and a scoring matrix, calculate an alignment score with a -1 penalty for each base aligned to a gap.
2. Write a new scoring function with separate penalties for opening a zero length gap (*e.g.*, G = -11) and extending an open gap by one base (*e.g.*, E = -1).

$$S_{gapped}(x, y) = S(x, y) + \sum_{i}^{gaps} (G + E * len(i))$$

Dynamic Programming



max and min

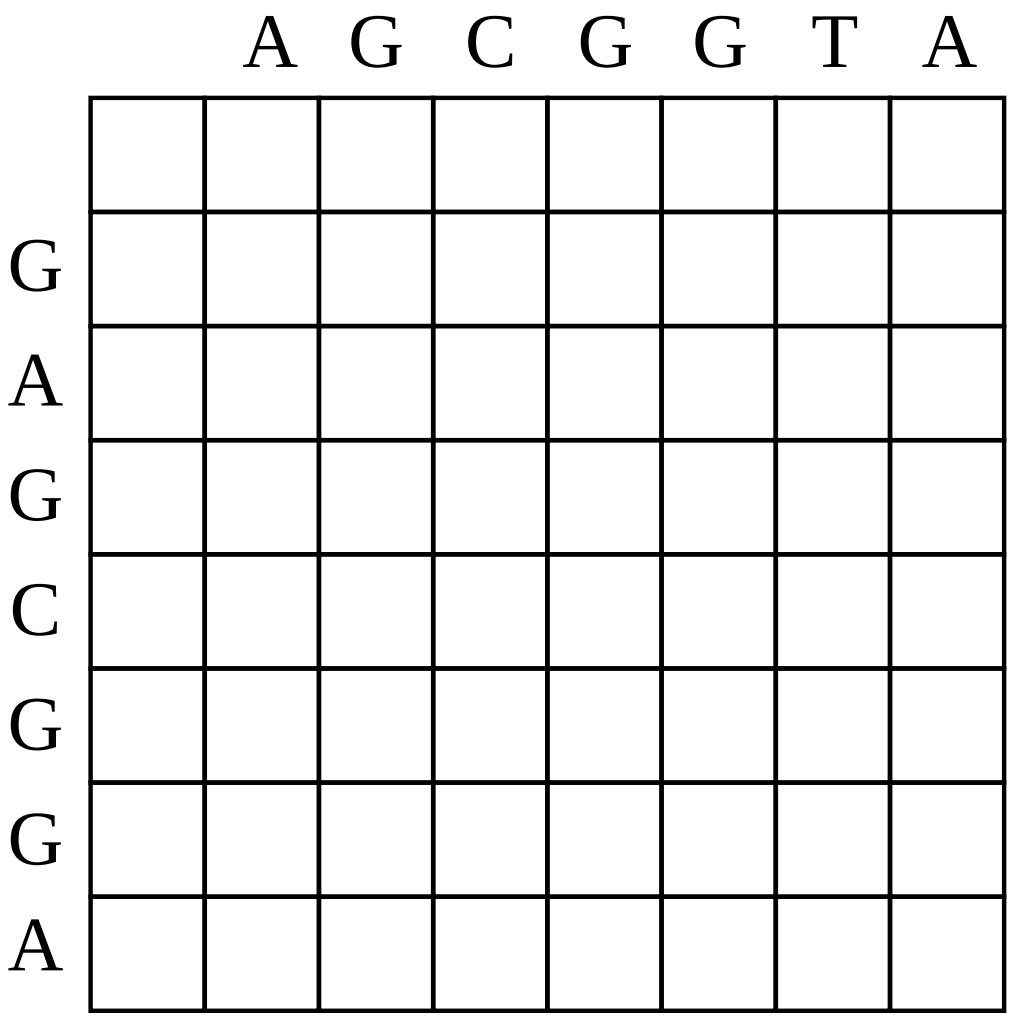
$$\min((1,2,3,4,5)) == 1$$
$$\max((1,2,3,4,5)) == 5$$

Types of alignments

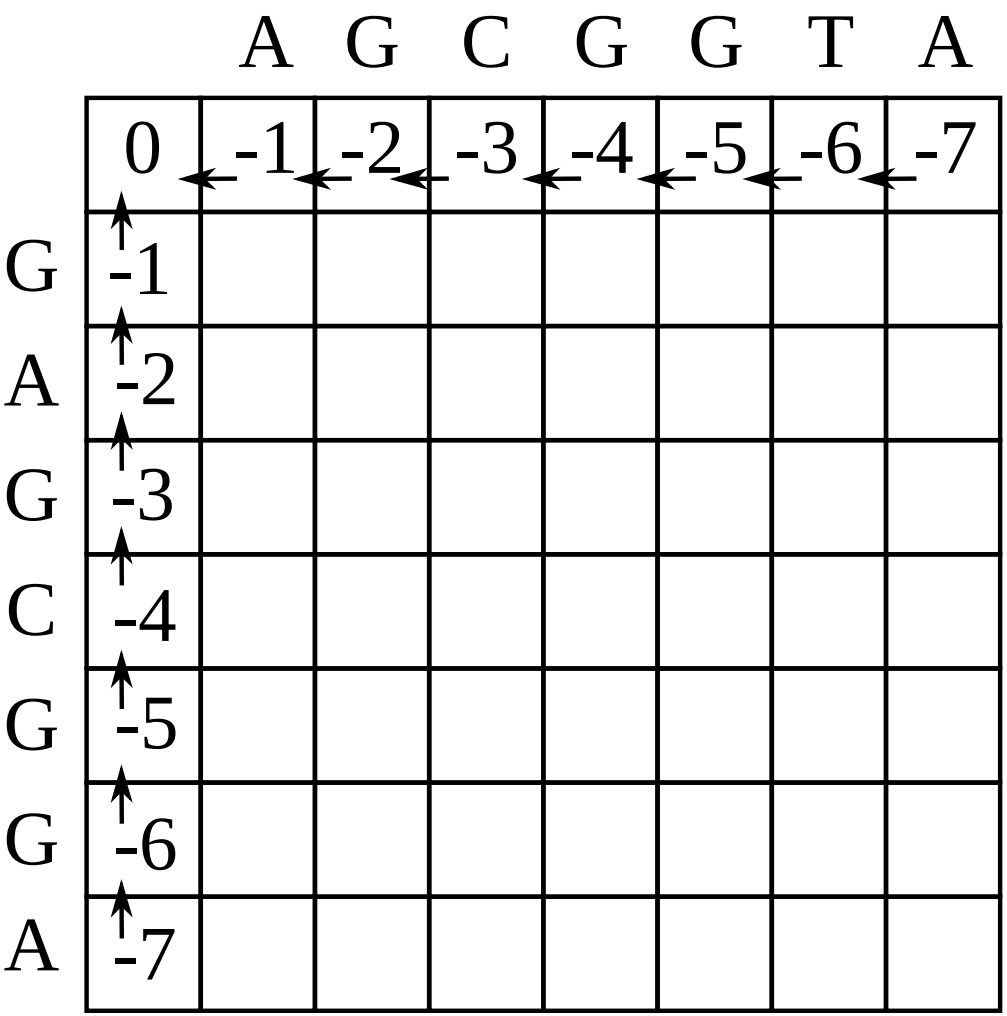
Global Alignment Each letter of each sequence is aligned to a letter or a gap (*e.g.*, Needleman-Wunsch)

Local Alignment An optimal pair of subsequences is taken from the two sequences and globally aligned (*e.g.*, Smith-Waterman)

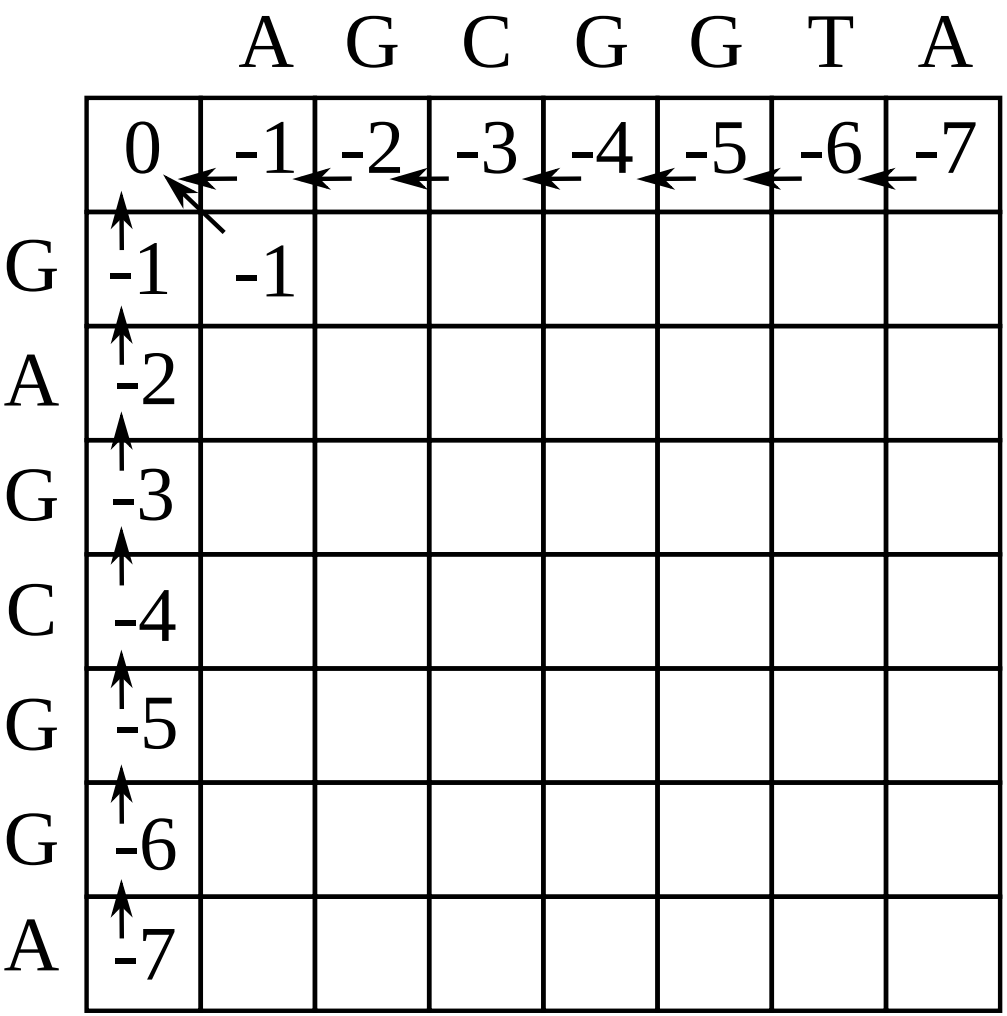
Needleman-Wunsch



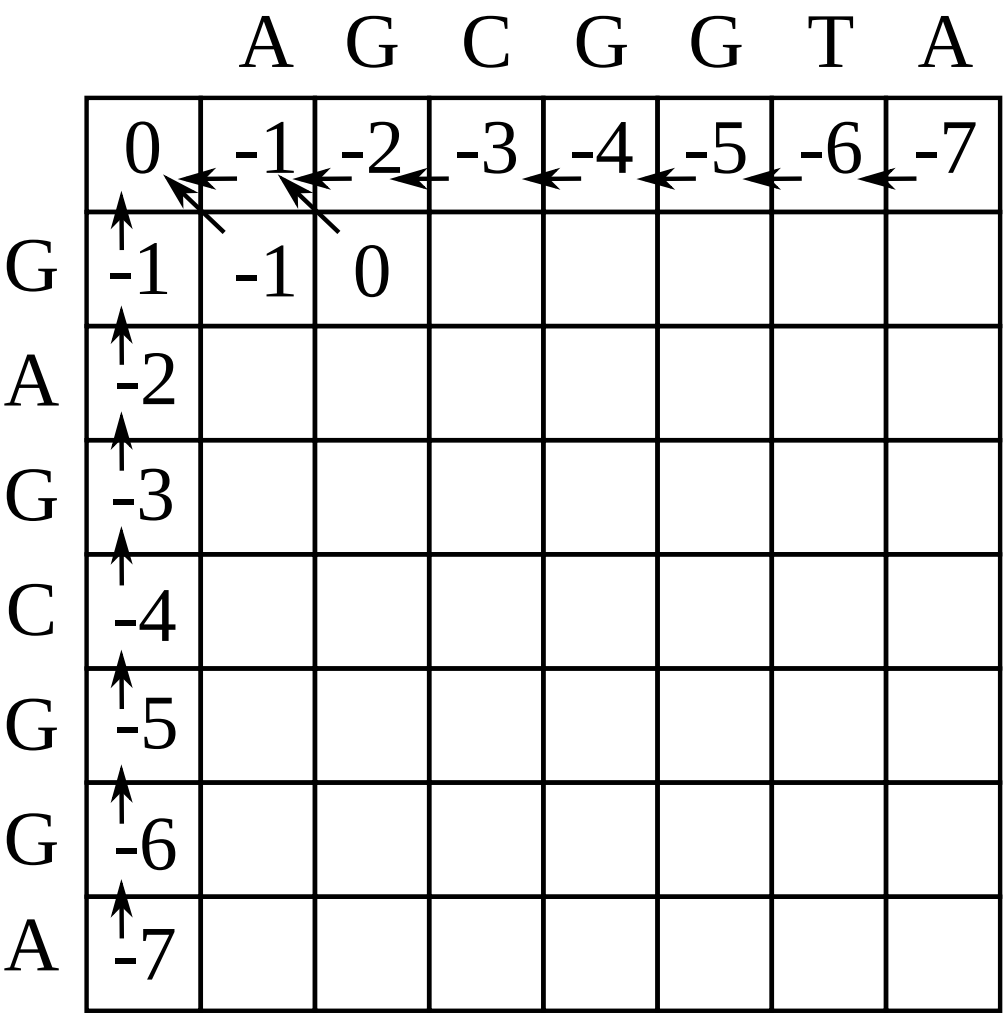
Needleman-Wunsch



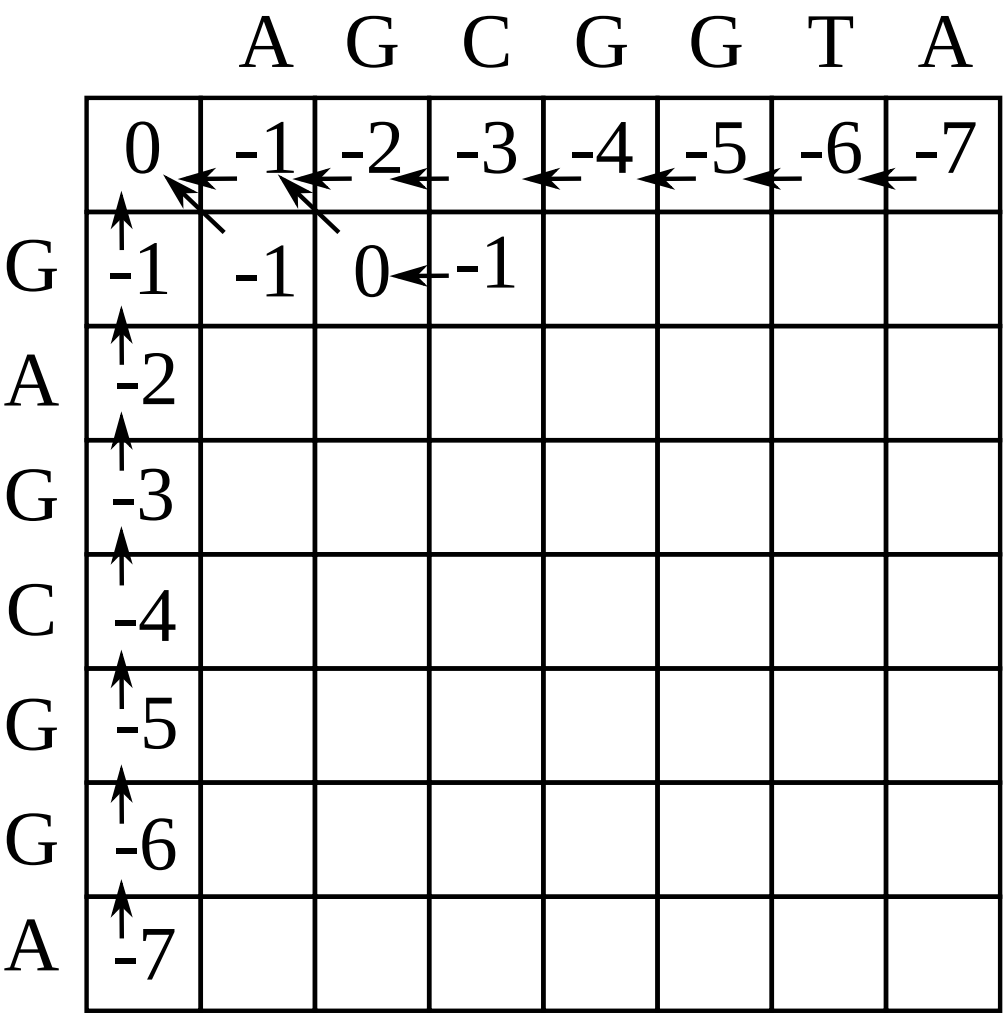
Needleman-Wunsch



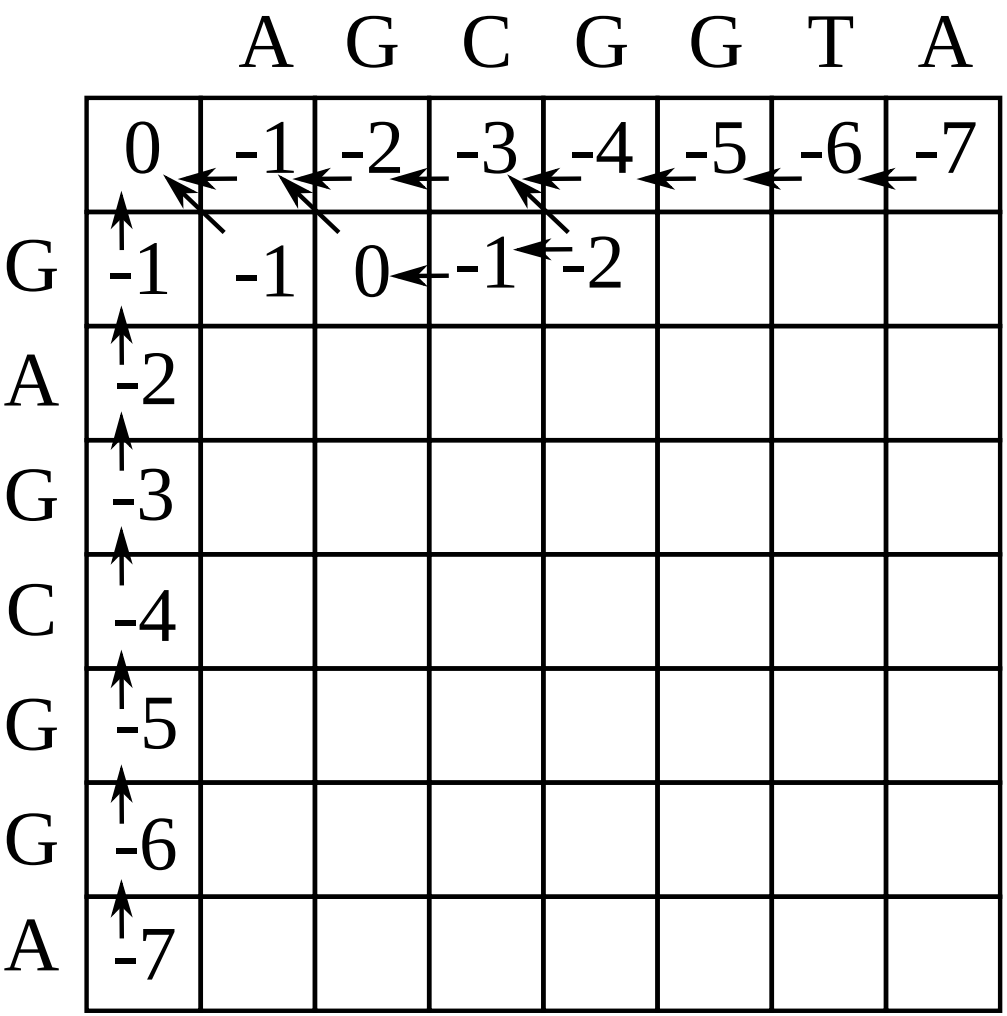
Needleman-Wunsch



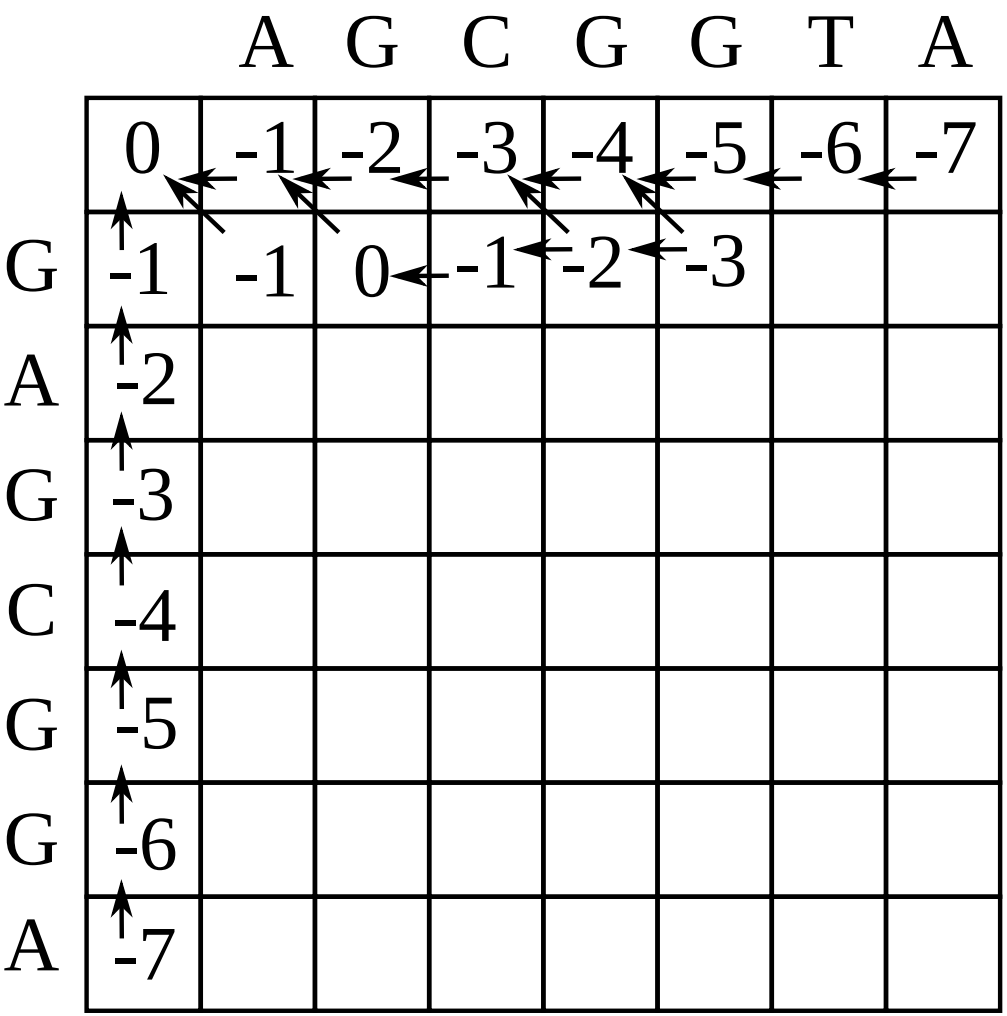
Needleman-Wunsch



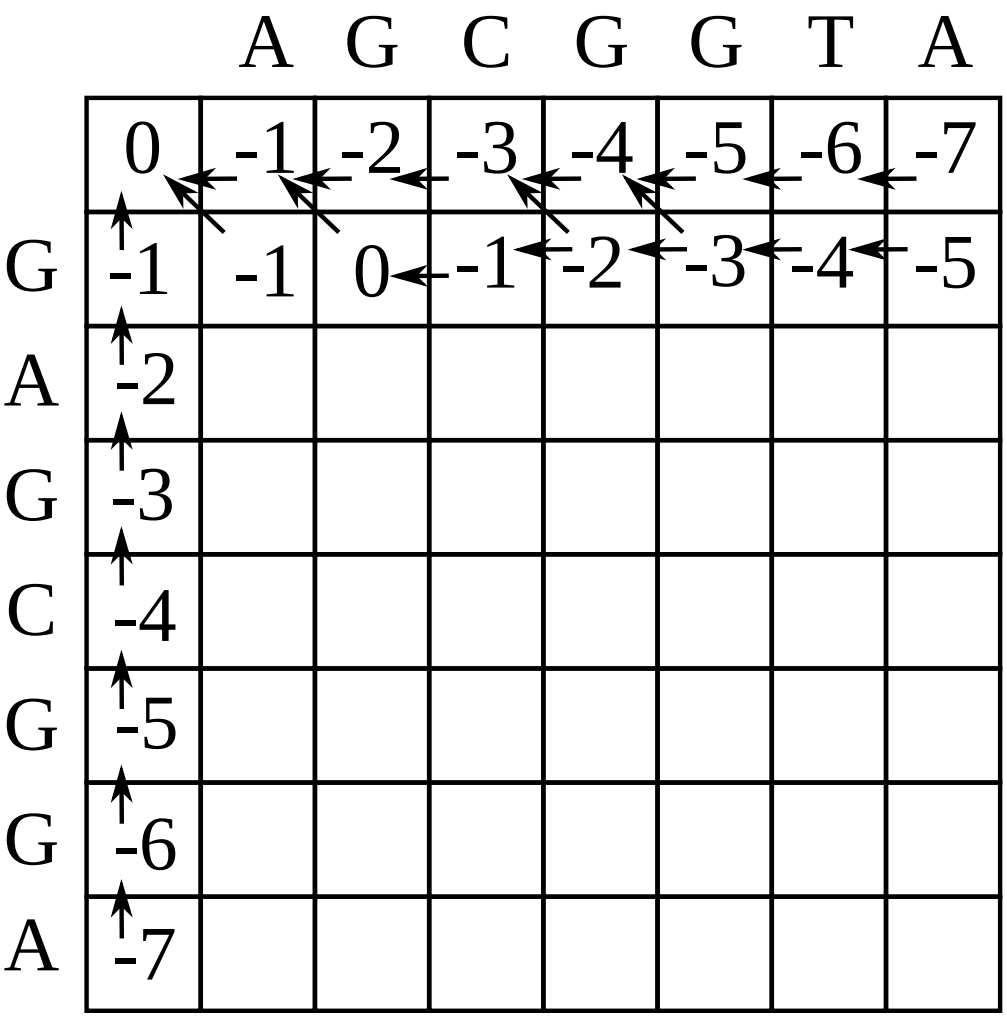
Needleman-Wunsch



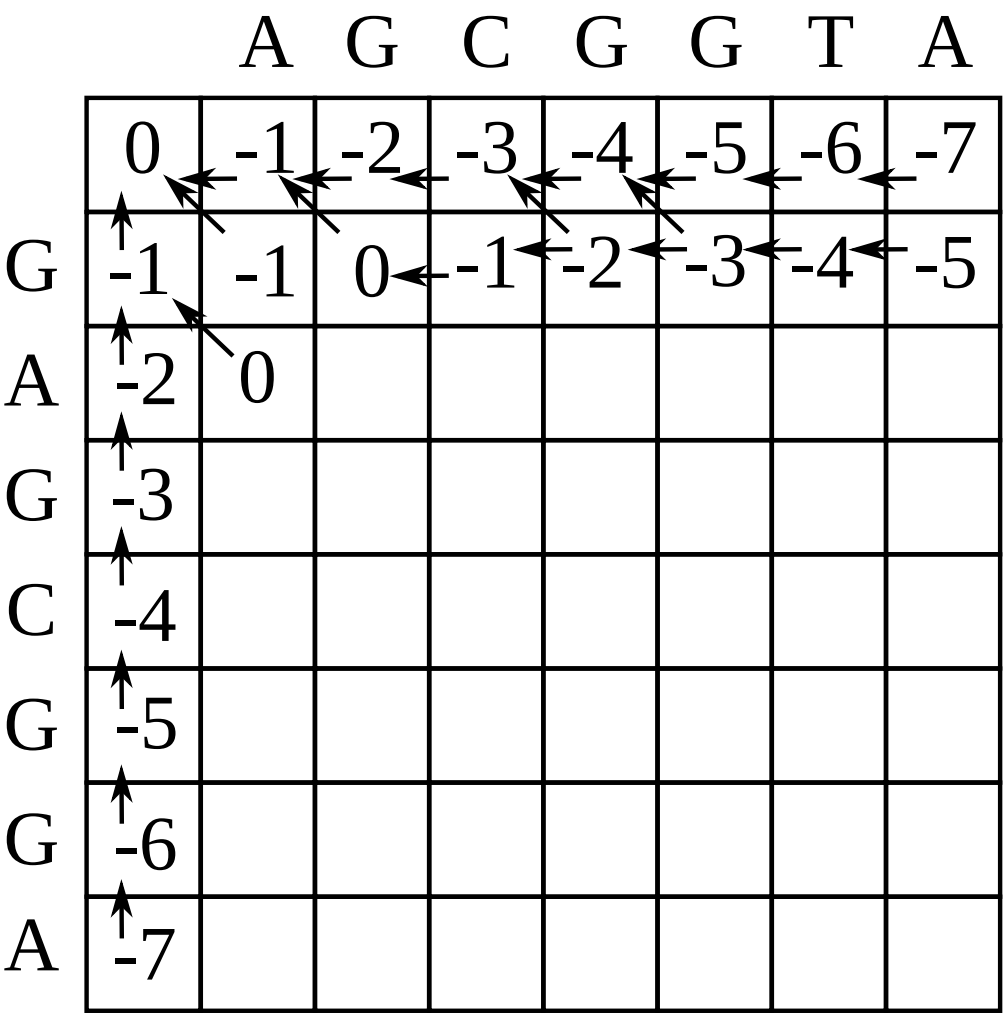
Needleman-Wunsch



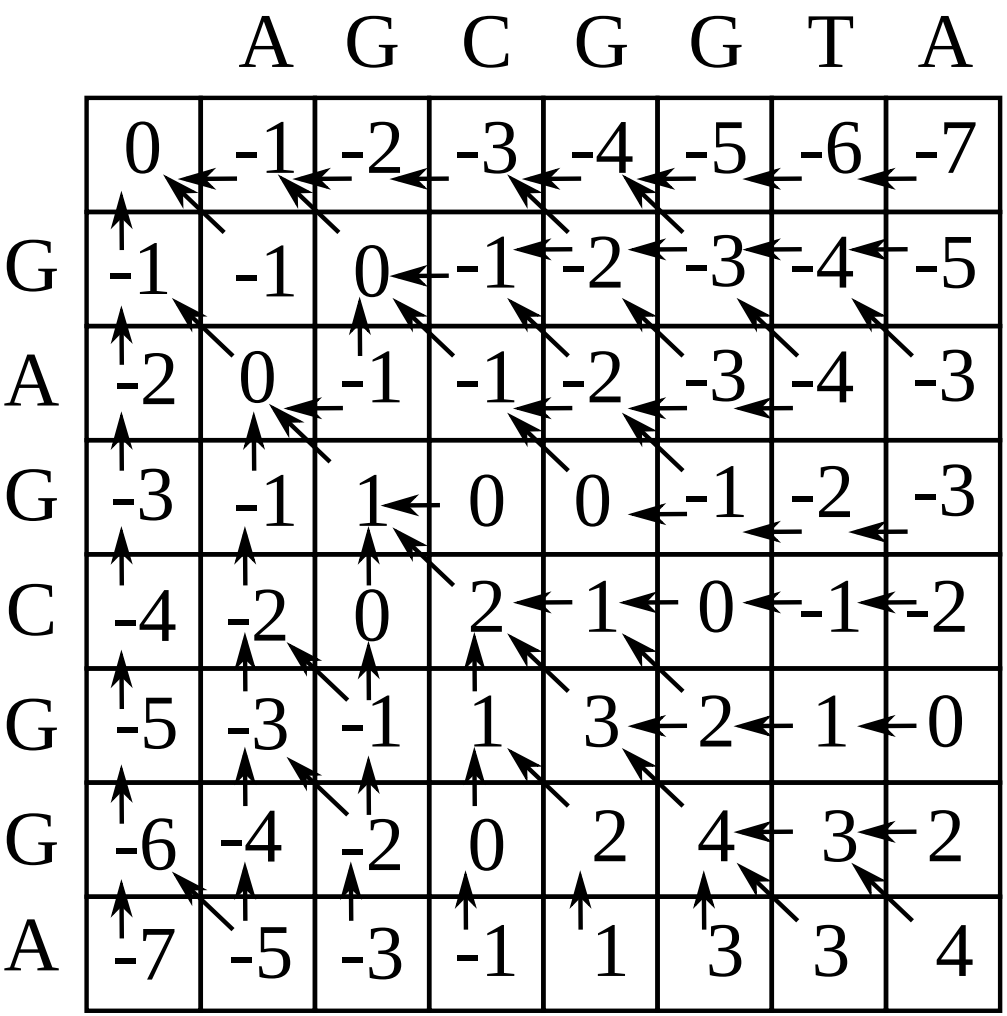
Needleman-Wunsch



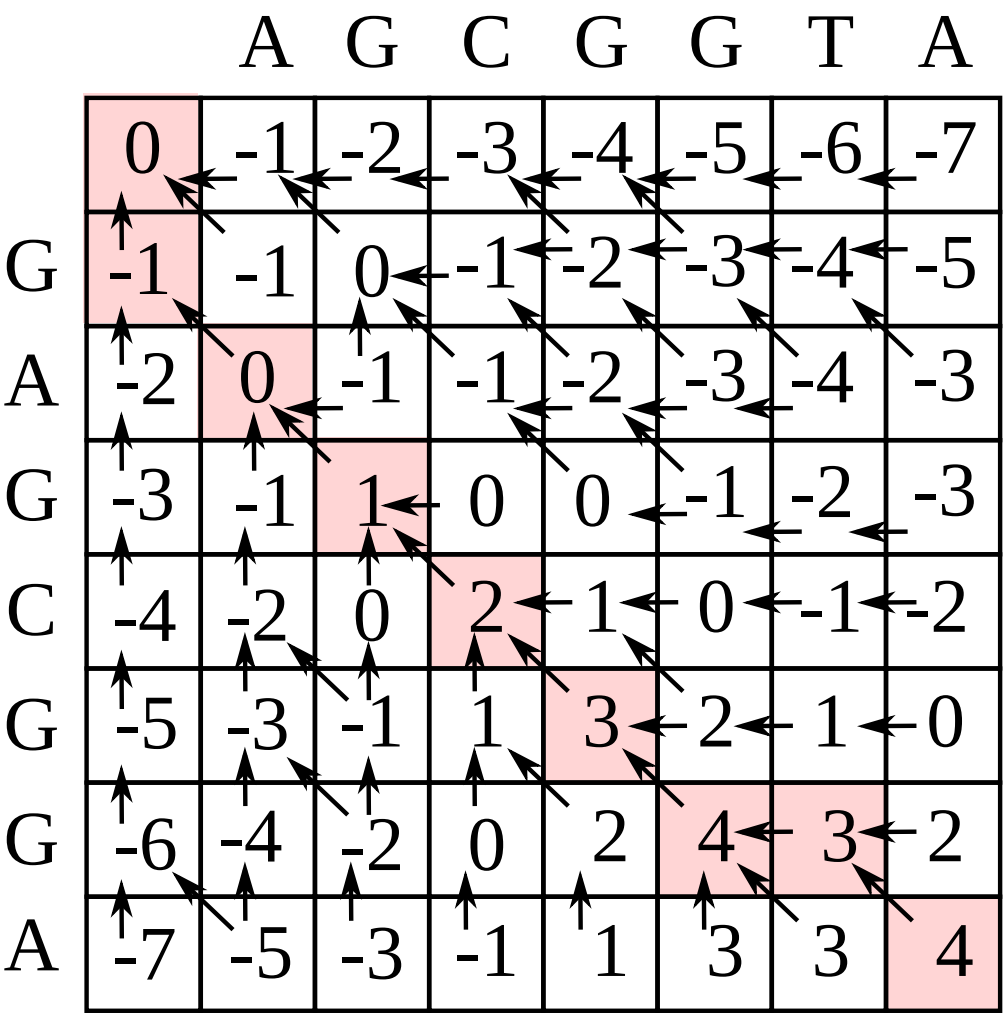
Needleman-Wunsch



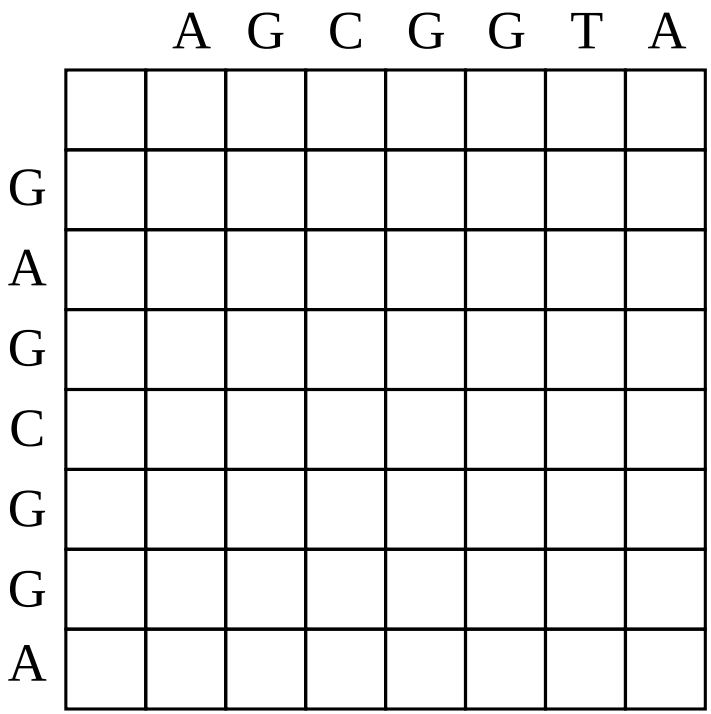
Needleman-Wunsch



Needleman-Wunsch

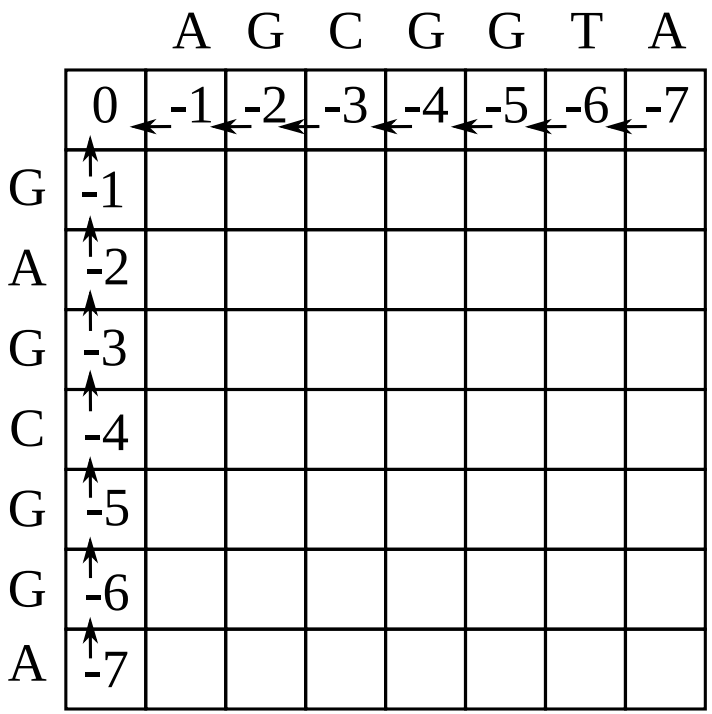


dp.nw_fill: Needleman-Wunsch with g=0



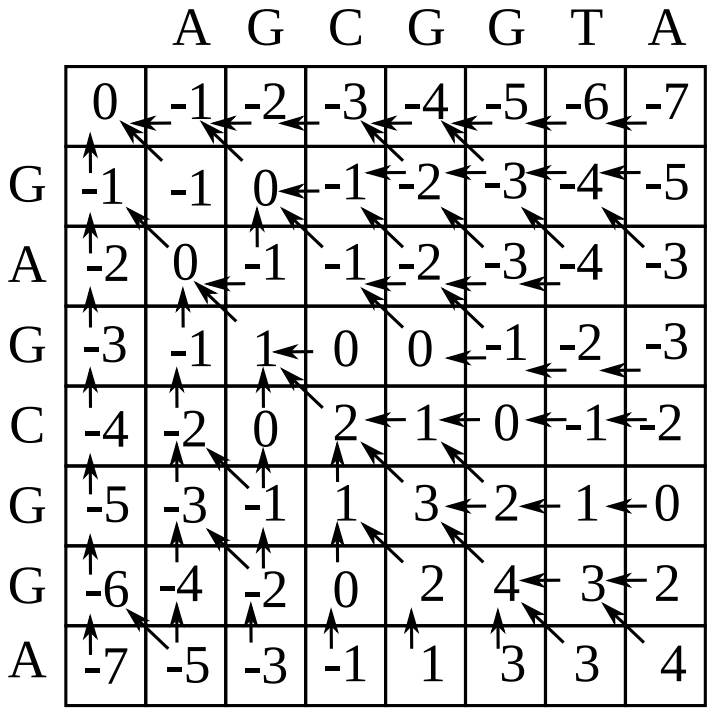

```
def nw_fill(seq1, seq2, s, e):
    # m[i][j] = best score for subalignment
    #   of seq1[:i+1],seq2[:j+1]
    m = [[0]]

    # p[i][j] = list of best paths from
    #   m[i][j], or None if the alignment
    #   stops here.
    p = [[None]]
```

dp.nw_fill: Needleman-Wunsch with g=0



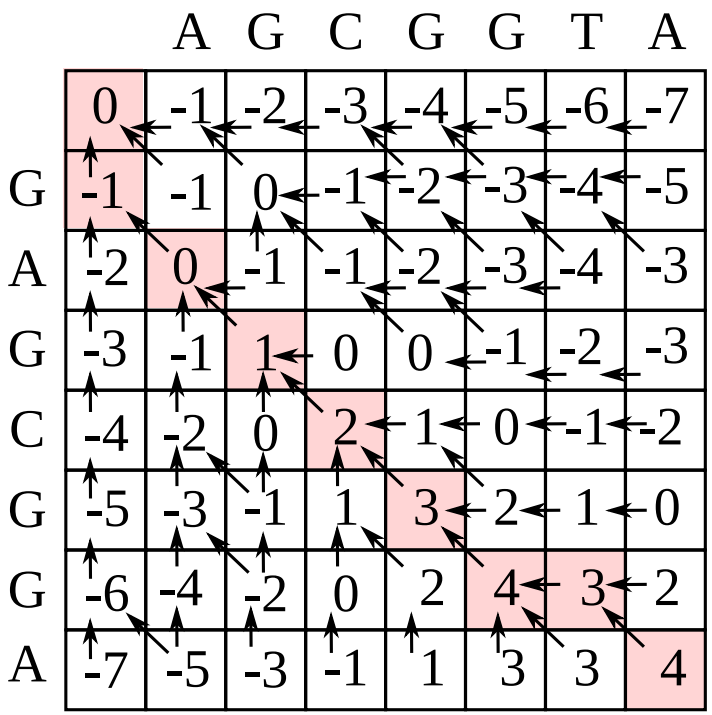

```
# Fill first row as leading gaps
for j in range(len(seq2)):
    m[-1].append(m[0][j]+e)
    p[-1].append([(0,j)])
```

dp.nw_fill: Needleman-Wunsch with g=0




```
for i in range(len(seq1)):
    # First column is leading gaps
    m.append([m[i][0]+e])
    p.append([[(i,0)]])
    for j in range(len(seq2)):
        # Score for aligning seq1[i] with seq2[j]
        match = m[i][j]+s[seq1[i]][seq2[j]]
        # Score for aligning seq1[i] with a gap
        hgap = m[i+1][j]+e
        # Score for aligning seq2[i] with a gap
        vgap = m[i][j+1]+e

        best = max(match, vgap, hgap)
        m[-1].append(best)
        p[-1].append([])

        if(match == best):
            p[-1][-1].append((i,j))
        if(hgap == best):
            p[-1][-1].append((i+1,j))
        if(vgap == best):
            p[-1][-1].append((i,j+1))
```

dp.nw_traceback: Needleman-Wunsch with g=0




```
# Start at bottom right corner
curpos = (len(seq1),len(seq2))
aligned1 = ""
aligned2 = ""

plist = p[curpos[0]][curpos[1]]
while(plist is not None):
    nextpos = plist[0]
    # Check for vgap
    if(nextpos[0] == curpos[0]):
        aligned1 = "-"+aligned1
    else:
        aligned1 = seq1[nextpos[0]]+aligned1
    # Check for hgap
    if(nextpos[1] == curpos[1]):
        aligned2 = "-"+aligned2
    else:
        aligned2 = seq2[nextpos[1]]+aligned2

    curpos = nextpos
    plist = p[curpos[0]][curpos[1]]
```

Smith-Waterman

Paraphrasing from the BLAST book

- Initialize edges to 0 *(no penalty for starting in the middle of a sequence)*
- The maximum score is never less than 0, and no pointer is recorded unless the score is greater than 0
- The trace-back starts from the highest score in the matrix and ends at a score of 0 *(local, rather than global, alignment)*

Real world dynamic programming implementations

Program	files	algorithms
clustalw	pairalign.c	Myers and Miller pairwise alignment
NCBI blast	blast_gapalign.c	Gapped extension
hmmer2	core_algorithms.c, fast_algorithms.c	Forward, Backward, Vterbi

Homework

- Read chapter 3 of the BLAST book (Sequence Alignment).
- Try initializing and filling in a dynamic programming matrix by hand (*e..g*, try reproducing one of the examples from the BLAST book on paper).
- Try extending the Needleman-Wunsch implementation to handle non-zero gap opening penalties.
- Try implementing Smith-Waterman local alignment