Sequence Alignment using Dynamic Programming

- Underlying sequences can be: DNA, RNA, or amino acids
- We will focus on *pairwise* sequence alignment in this lecture

- Problem: Given two sequences, find an alignment that indicates conserved regions

  e.g.

```
_ A T G A _ A T C A
G A _ G C C A T _ A
```

Global Alignment

- Scoring matchs/mismatches and insertions/deletions:

s	A	C	G	T	-
A	1	-1	-2	0	-1
C		3	-2	-1	0
G			3	-4	-2
T				3	-1
-					0

Calculating the Value of an Alignment

- Given: alignment A of sequences S_1 and S_2

```
S1      C A C - G T A - A A G
S2      - A T G - T A T A - G
```

- Value of the alignment: V(A) = sum over column scores:

```
0+1-1-1-1+0+1-1+1-1+3 = 1
```

Sequence Similarity

- We want to find the alignment A of S_1 and S_2 that maximizes V(A).
- The optimal value of V(A) is called the *similarity* of S_1 and S_2 .

- Is there a fast way to calculate similarity?

Yes…

The Smith-Waterman Algorithm

- **Given**: S_1 of length n, S_2 of length m
  Problem: Compute the optimal alignment of S_1 and S_2

- Let V(i,j) be the value of the optimal alignment of $S_1[1..i]$ and $S_2[1..j]$. *(these are small problems)*

Base Conditions

- Aligning the empty prefix of S_1 to prefixes of S_2 :

$$V(0, j) = \sum_{1 \le k \le j} s(-, S_2(k))$$

- Aligning the empty prefix of S_2 to prefixes of S_1 :

$$V(i,0) = \sum_{1 \le k \le i} s(S_1(k),-)$$

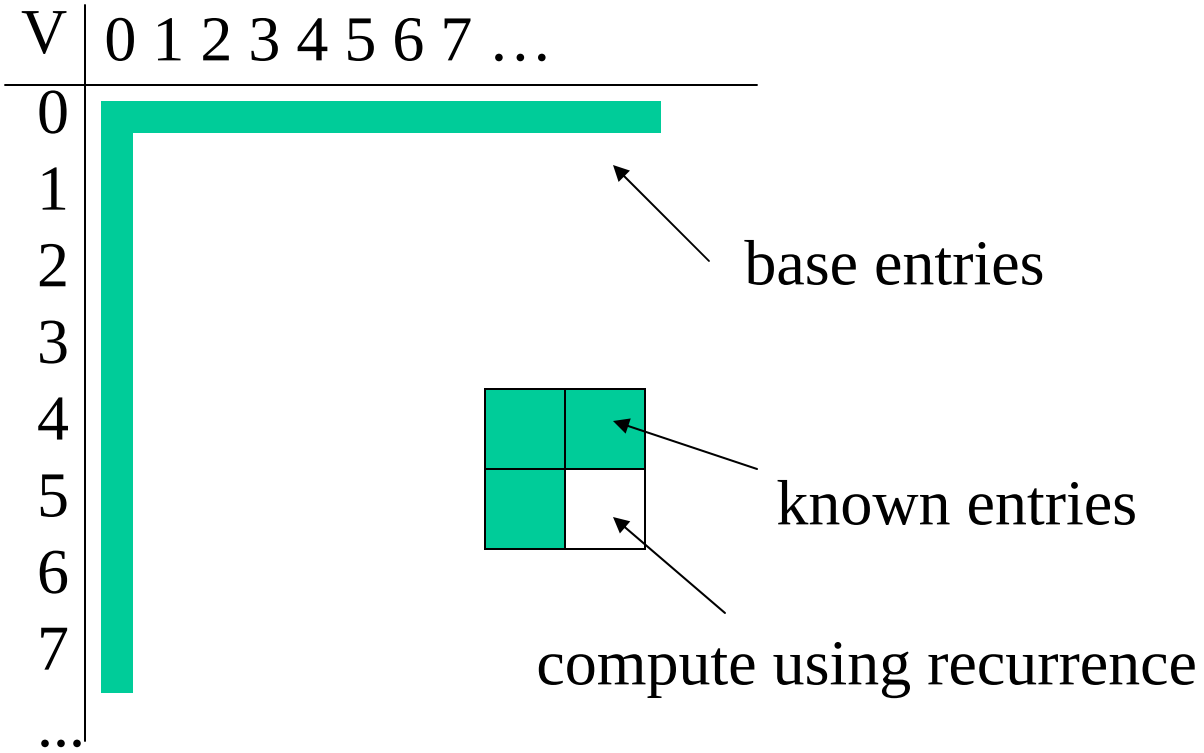
The Recurrence Relation

- For i and j positive we have:

$$V(i,j) = \max \begin{bmatrix} V(i-1, j-1) + s(S_1(i), S_2(j)), & \text{(mis)match} \\ V(i-1, j) + s(S_1(i), -), & \text{deletion} \\ V(i, j-1) + s(-, S_2(j)) & \text{insertion} \end{bmatrix}$$

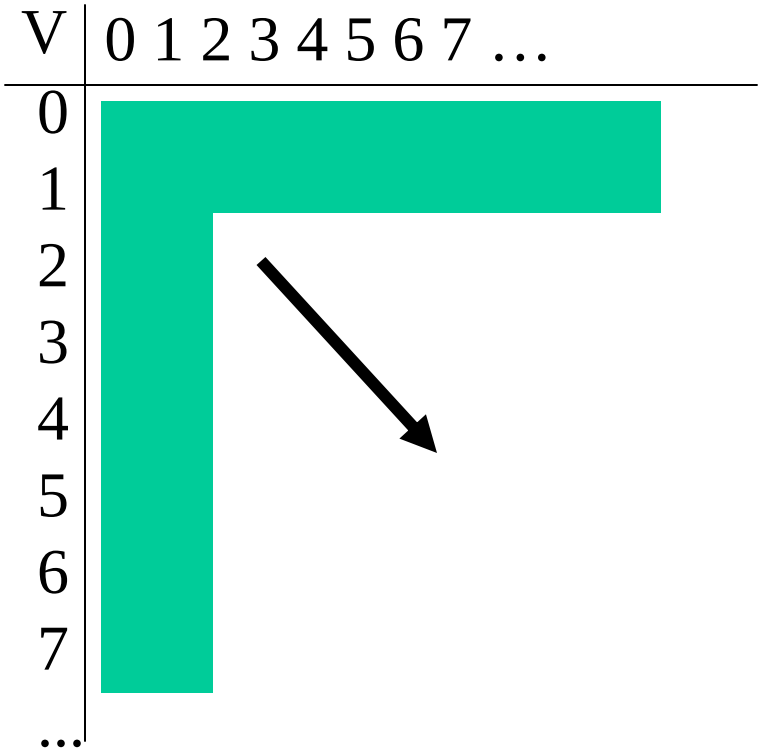
Solving…

- We can efficiently find V(i,j) for all values of i and j using a table:



Filling in the table

- We can fill in the table by working progressively down and right:



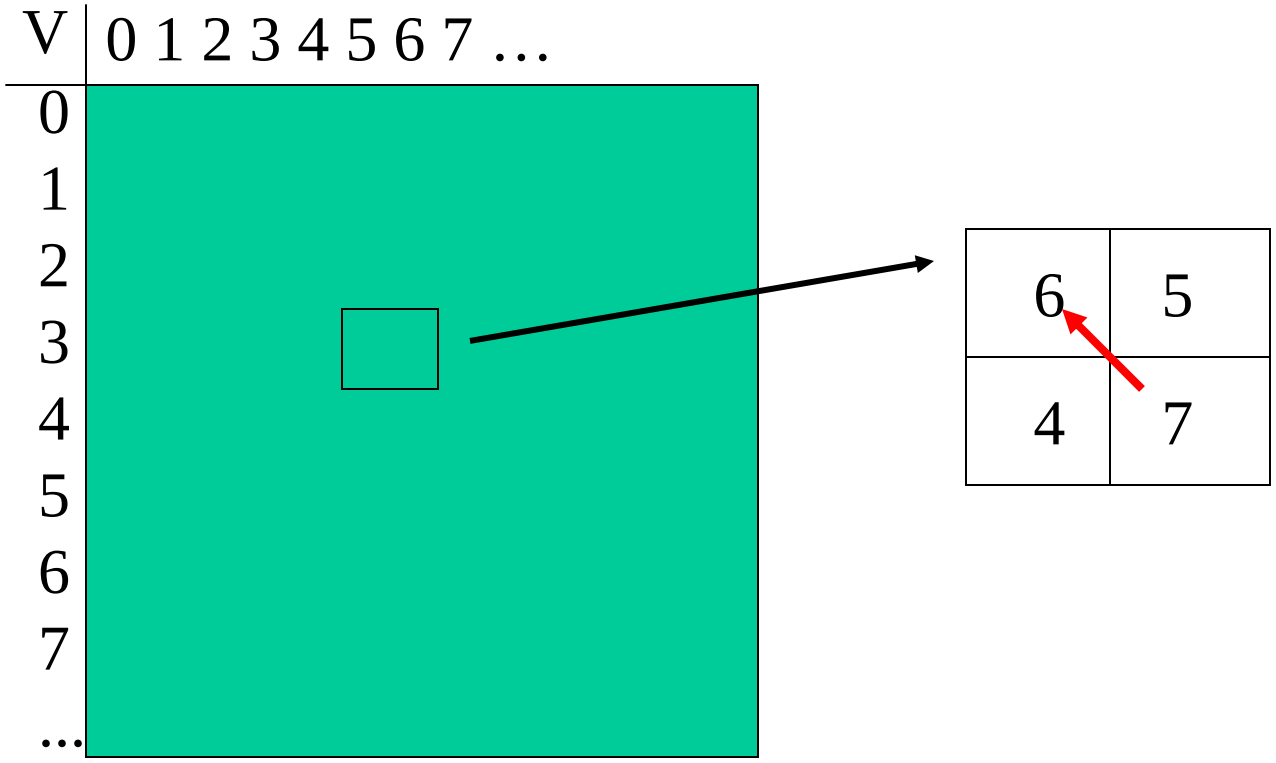
Finding the alignment

- When the table entry V(n,m) has been found we know the value of the optimal alignment
- How do we find the actual alignment itself?

…using traceback

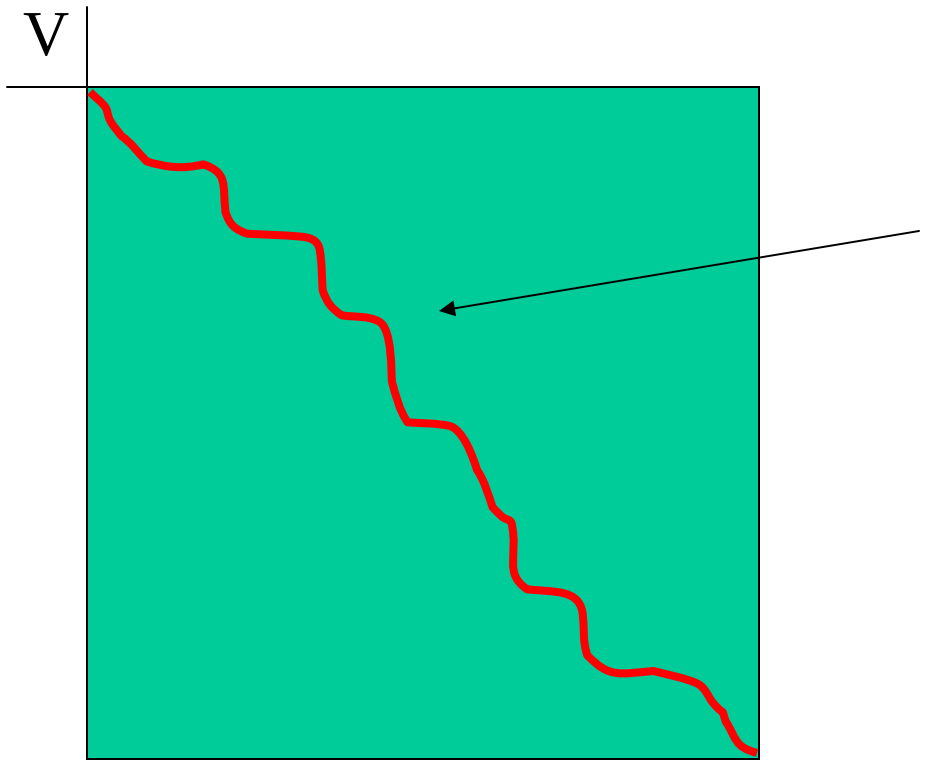
Traceback

- Keep track of optimal direction choice for each table entry:



Traceback Continued

- Trace a red path from V(n,m) back to V(0,0):



This path indicates the optimal alignment A

Example

- Let’s work through an example:

```
S1 = A C T A G C
S2 = C T A C A
```

Example Cont.

V	-	C	T	A	C	A
-	0	0	-1	-2	-2	-3
A	-1	-1	0	0	0	-1
C	-1	2	1	0	3	2
T	-2	1	5	4	4	3
A	-3	0	4	6	6	5
G	-5	-2	2	4	4	4
C	-5	-2	2	4	7	6

Optimal alignment:

A C T A G C -

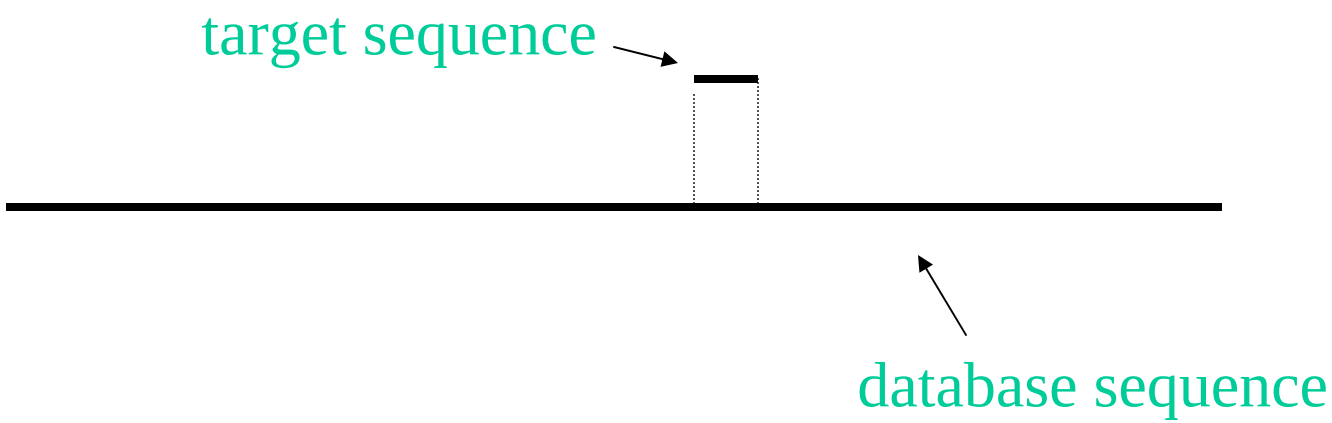
- C T A - C A

Complexity Issues

- Time used: O(nm)
- Space: O(nm) -- *can be reduced to O(n+m)*

Local Alignments and Beyond

- *Ubiquitous task:* Searching for related sequence in databases…



Some Gap Models

- **Constant**: each insertion/deletion is free and there is a fixed cost per gap O(nm) time
- **Affine**: there is a fixed cost per gap, plus a cost per gap space O(nm) time
- **Convex**: here the cost per gap is a convex (but not affine) function of gap length O(nm log(m+n)) time