

Pairwise Sequence Alignment: Dynamic Programming Algorithms

COMP 571 - Fall 2010
Luay Nakhleh, Rice University

DP Algorithms for Pairwise Alignment

- * The number of all possible pairwise alignments (if gaps are allowed) is exponential in the length of the sequences
- * Therefore, the approach of “score every possible alignment and choose the best” is infeasible in practice
- * Efficient algorithms for pairwise alignment have been devised using **dynamic programming (DP)**

DP Algorithms for Pairwise Alignment

- * The key property of DP is that the problem can be divided into many smaller parts and the solution can be obtained from the solutions to these smaller parts

The Needleman-Wunch Algorithm for Global Pairwise Alignment

- * The problem is to align two sequences x ($x_1x_2\dots x_m$) and y ($y_1y_2\dots y_n$) finding the best scoring alignment in which all residues of both sequences are included
- * The score is assumed to be a measure of similarity, so the highest score is desired
- * Alternatively, the score could be an evolutionary distance, in which case the smallest score would be sought, and all uses of "max" in the algorithm would be replaced by "min"

The Needleman-Wunch Algorithm for Global Pairwise Alignment

- * The key concept in all these algorithms is the matrix S of optimal scores of subsequence alignments
- * The matrix has $(m+1)$ rows labeled $0 \rightarrow m$ and $(n+1)$ columns labeled $0 \rightarrow n$
- * The rows correspond to the residues of sequence x , and the columns correspond to the residues of sequence y

The Needleman-Wunch Algorithm for Global Pairwise Alignment

- * We'll use as a working example the two sequences $x=THISLINE$ and $y=ISALIGNED$ with BLOSUM-62 substitution matrix as the scoring matrix and linear gap penalty $g=E$
- * The optimal alignment of these two sequences is

```
T H I S - L I - N E -  
- - I S A L I G N E D
```

The Matrix S

$$E = -8$$

		y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	
		I	S	A	L	I	G	N	E	D	
		0	-8	-16	-24	-32	-40	-48	-56	-64	-72
x_1	T	-8									
x_2	H	-16									
x_3	I	-24									
x_4	S	-32									
x_5	L	-40									
x_6	I	-48									
x_7	N	-56									
x_8	E	-64									

$S_{i,0}$ (pointing to the first column)

$S_{0,j}$ (pointing to the first row)

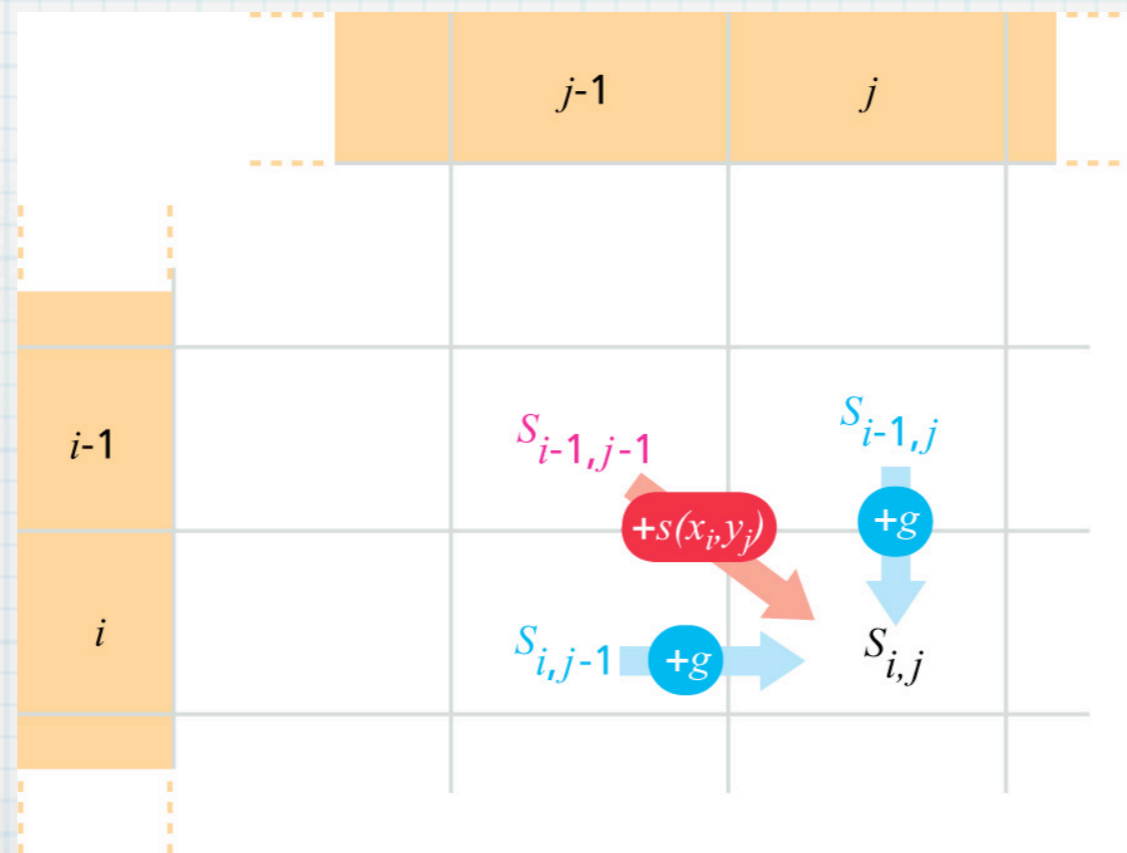
$S_{i,j}$ stores the score for the optimal alignment of all residues up to x_i of sequence x will all residues up to y_j of sequence y

$$S_{i,0} = S_{0,i} = ig$$

The Matrix S

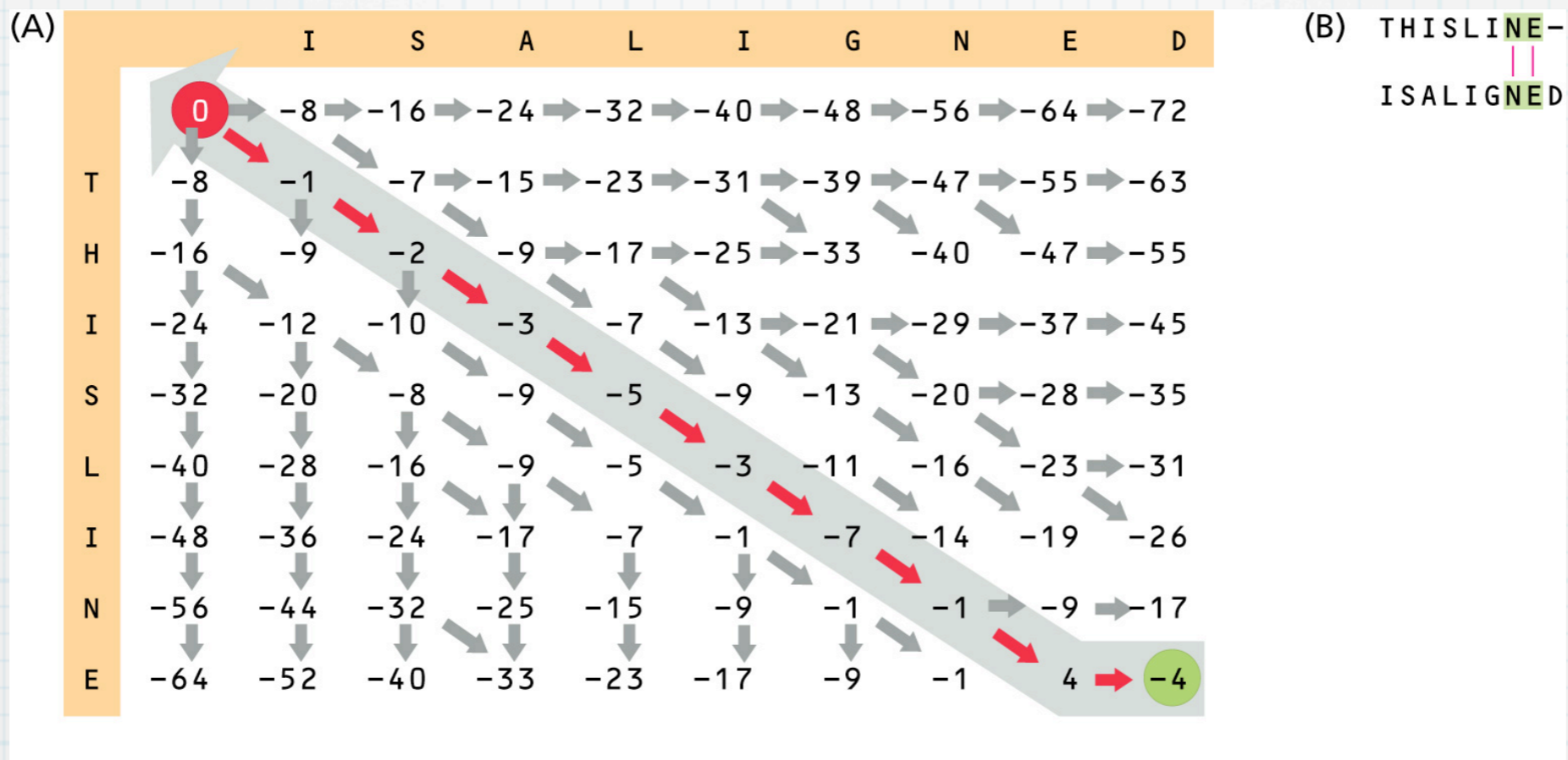
- * To complete the matrix, we use the formula

$$S_{i,j} = \max \begin{cases} S_{i-1,j-1} + s(x_i, y_j) \\ S_{i-1,j} + g \\ S_{i,j-1} + g \end{cases}$$



The Matrix S

E=-8



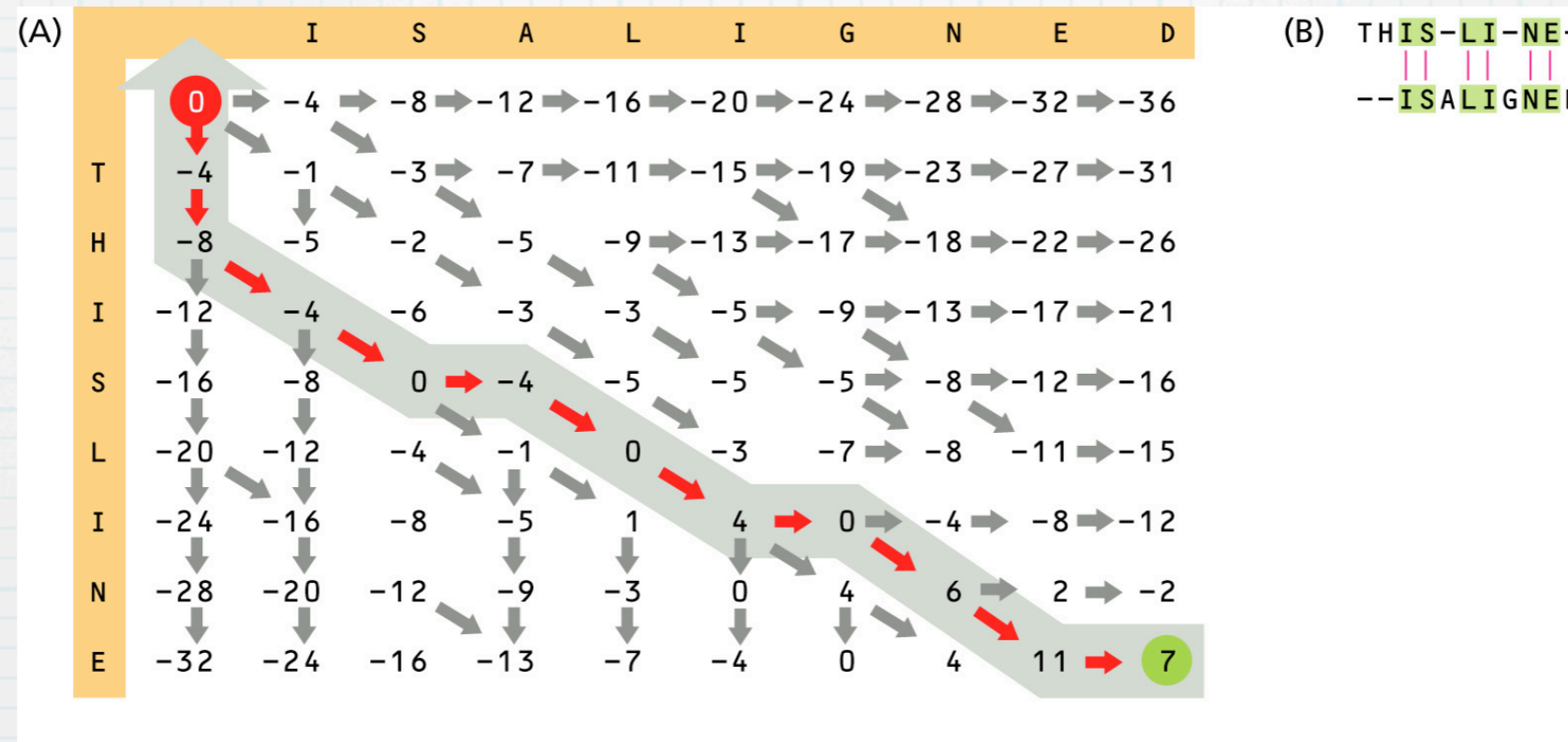
$$S_{i,j} = \max \begin{cases} S_{i-1,j-1} + s(x_i, y_j) \\ S_{i-1,j} + g \\ S_{i,j-1} + g \end{cases}$$

The global alignment can be obtained by traceback

- * If instead we use a linear gap penalty of -4 , inserting a gap becomes less severe, and a gapped alignment is more likely to be obtained
- * Therefore it is very important to match the gap penalty to the substitution matrix used

The Matrix S

$$E = -4$$



$$S_{i,j} = \max \begin{cases} S_{i-1,j-1} + s(x_i, y_j) \\ S_{i-1,j} + g \\ S_{i,j-1} + g \end{cases}$$

The global alignment can be obtained by traceback

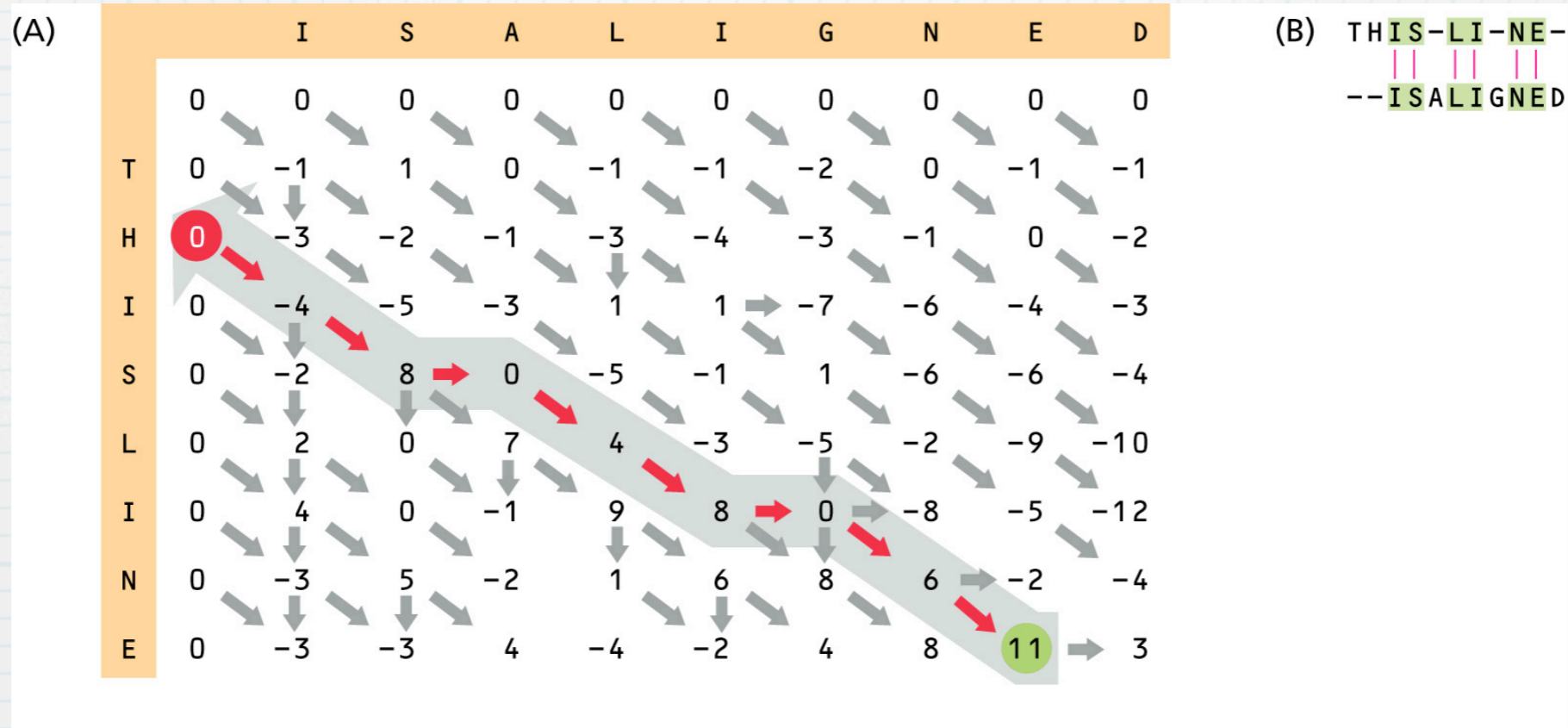
Multiple Optimal Alignments

- * There may be more than one optimal alignment
- * During traceback this is indicated by encountering an element that was derived from more than one of the three possible alternatives
- * The algorithm does not distinguish between these possible alignments, although there may be reasons (such as knowledge of molecular structure or function) for preferring one to the others
- * Most programs will arbitrarily report just one single alignment

Semiglobal Alignments

- * In certain applications, we don't want to penalize gaps at the beginning or end of either of the two sequences in the alignment
- * The resulting alignment is called **semiglobal alignment**
- * To obtain it, we have two modifications to the Needleman-Wunch algorithm: (1) we initialize $S_{i,0}$ and $S_{0,j}$ to 0 for all i and j , and (2) the traceback starts at the highest-scoring element in the bottom row or right-most column

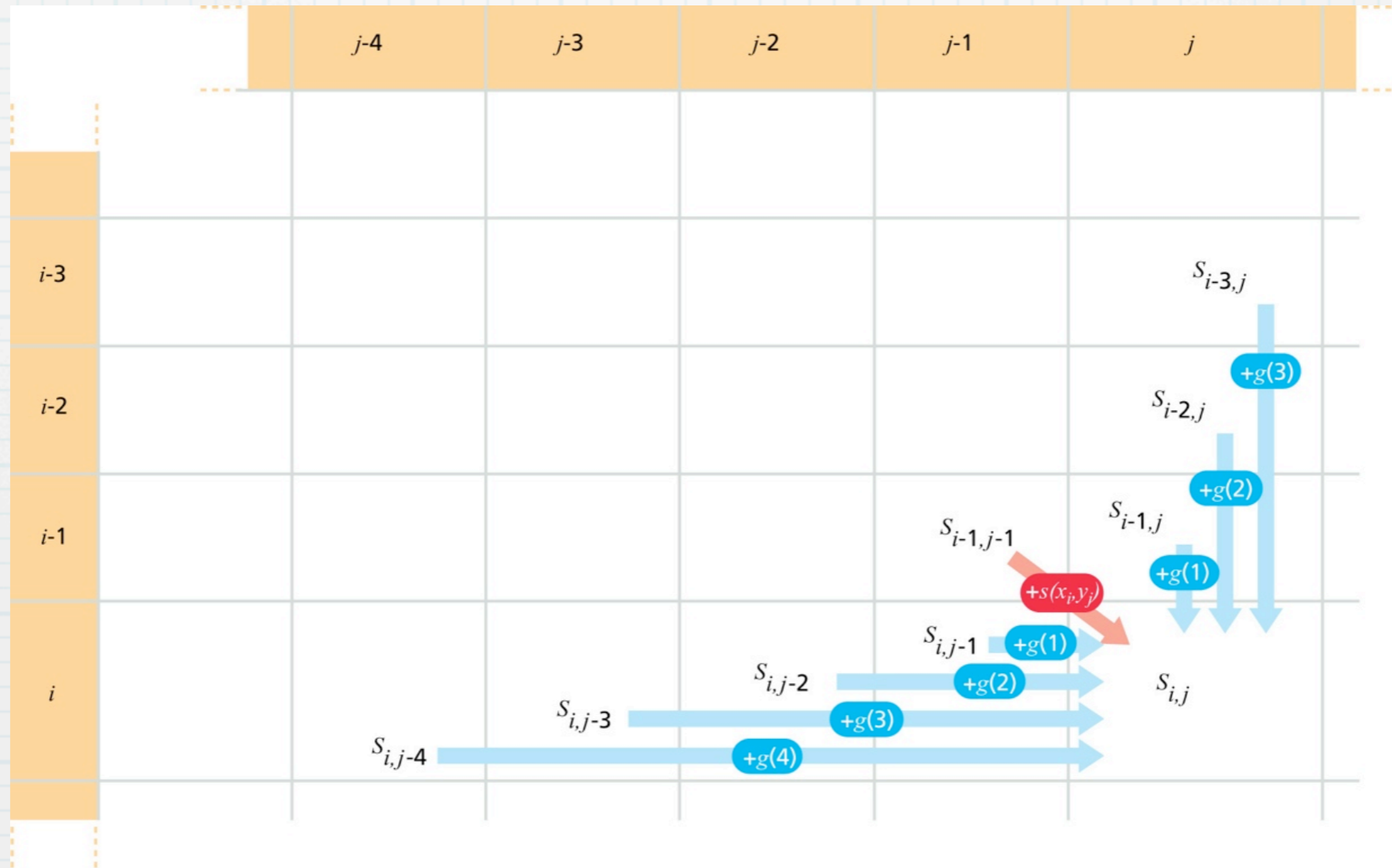
Semiglobal Alignment



General Gap Penalty

- * The algorithm we presented works with a linear gap penalty of the form $g(n_{\text{gap}}) = -n_{\text{gap}}\epsilon$
- * For a general gap penalty model $g(n_{\text{gap}})$, one must consider the possibility of arriving at $S_{i,j}$ directly via insertion of a gap of length up to i in sequence x or j in sequence y

General Gap Penalty



General Gap Penalty

- * The algorithm now has to be modified to

$$S_{i,j} = \max \left\{ \begin{array}{l} S_{i-1,j-1} + s(x_i, y_j) \\ (S_{i-n_{gap1},j} + g(n_{gap1}))_{1 \leq n_{gap1} \leq i} \\ (S_{i,j-n_{gap2}} + g(n_{gap2}))_{1 \leq n_{gap2} \leq j} \end{array} \right.$$

- This algorithm takes time that is proportional to mn^2 , where m and n are the sequence lengths with $n \geq m$

Affine Gap Penalty

- * For an affine gap penalty $g(n_{gap}) = -I - (n_{gap} - 1)E$, we can refine this algorithm to obtain an $O(mn)$ algorithm
- * Define two matrices

$$V_{i,j} = \max\{S_{i-n_{gap1},j} + g(n_{gap1})\}_{1 \leq n_{gap1} \leq i}$$

$$W_{i,j} = \max\{S_{i,j-n_{gap2}} + g(n_{gap2})\}_{1 \leq n_{gap2} \leq j}$$

Affine Gap Penalty

* These matrices can be defined recursively as

$$V_{i,j} = \max \begin{cases} S_{i-1,j} - I \\ V_{i-1,j} - E \end{cases}$$

$$W_{i,j} = \max \begin{cases} S_{i,j-1} - I \\ W_{i,j-1} - E \end{cases}$$

● Now, the matrix **S** can be written as

$$S_{i,j} = \max \begin{cases} S_{i-1,j-1} + s(x_i, y_j) \\ V_{i,j} \\ W_{i,j} \end{cases}$$

Local Pairwise Alignment

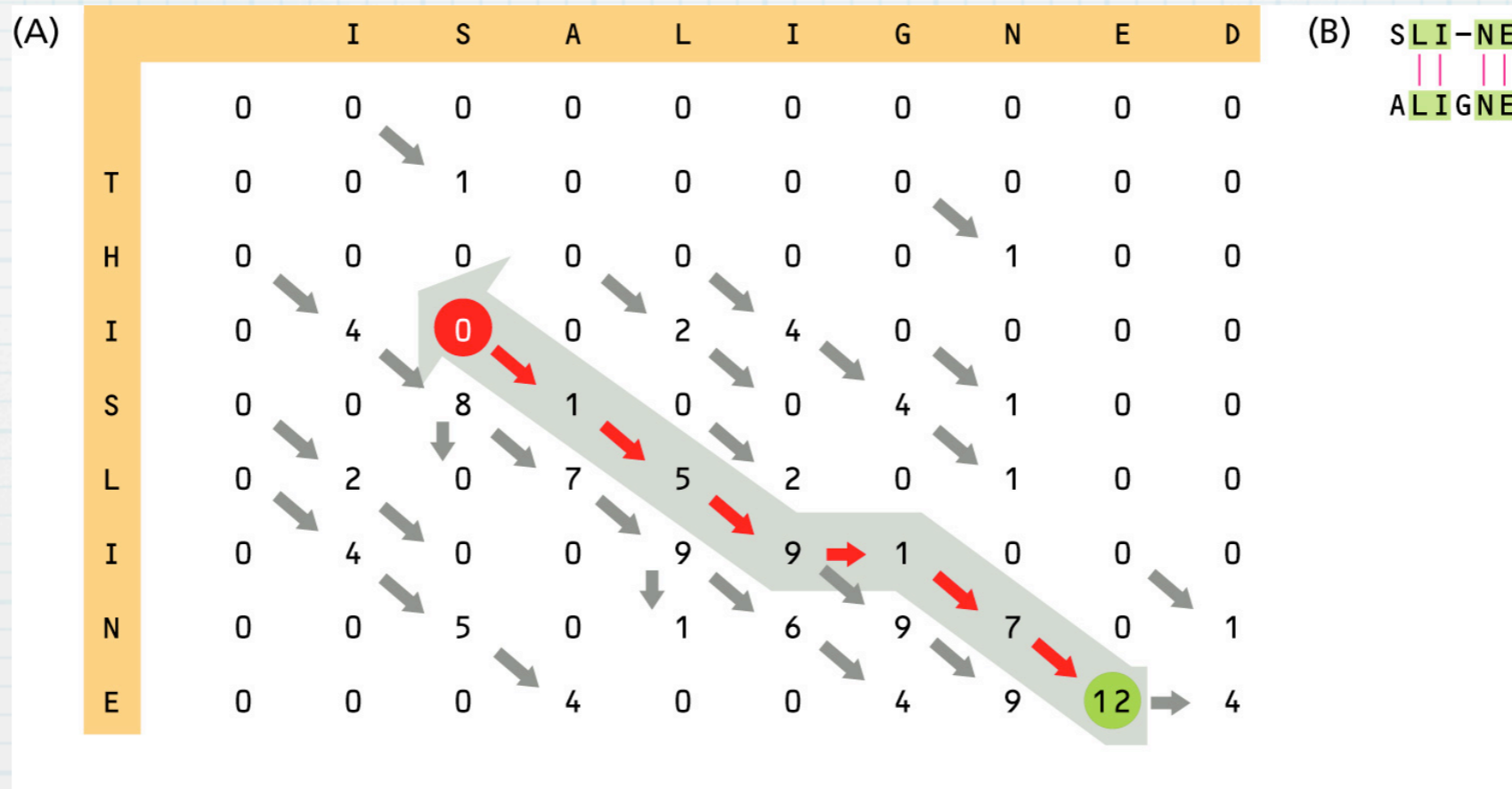
- * As mentioned before, sometimes local alignment is more appropriate (e.g., aligning two proteins that have just one domain in common)
- * The algorithmic differences between the algorithm for local alignment (Smith-Waterman algorithm) and the one for global alignment:
 - * Whenever the score of the optimal sub-alignment is less than zero, it is rejected (the matrix element is set to 0)
 - * Traceback starts from the highest-scoring matrix element

Local Pairwise Alignment

$$S_{i,j} = \max \left\{ \begin{array}{ll} S_{i-1,j-1} & + s(x_i, y_j) \\ (S_{i-n_{gap1},j} & + g(n_{gap1}))_{1 \leq n_{gap1} \leq i} \\ (S_{i,j-n_{gap2}} & + g(n_{gap2}))_{1 \leq n_{gap2} \leq j} \\ 0 & \end{array} \right.$$

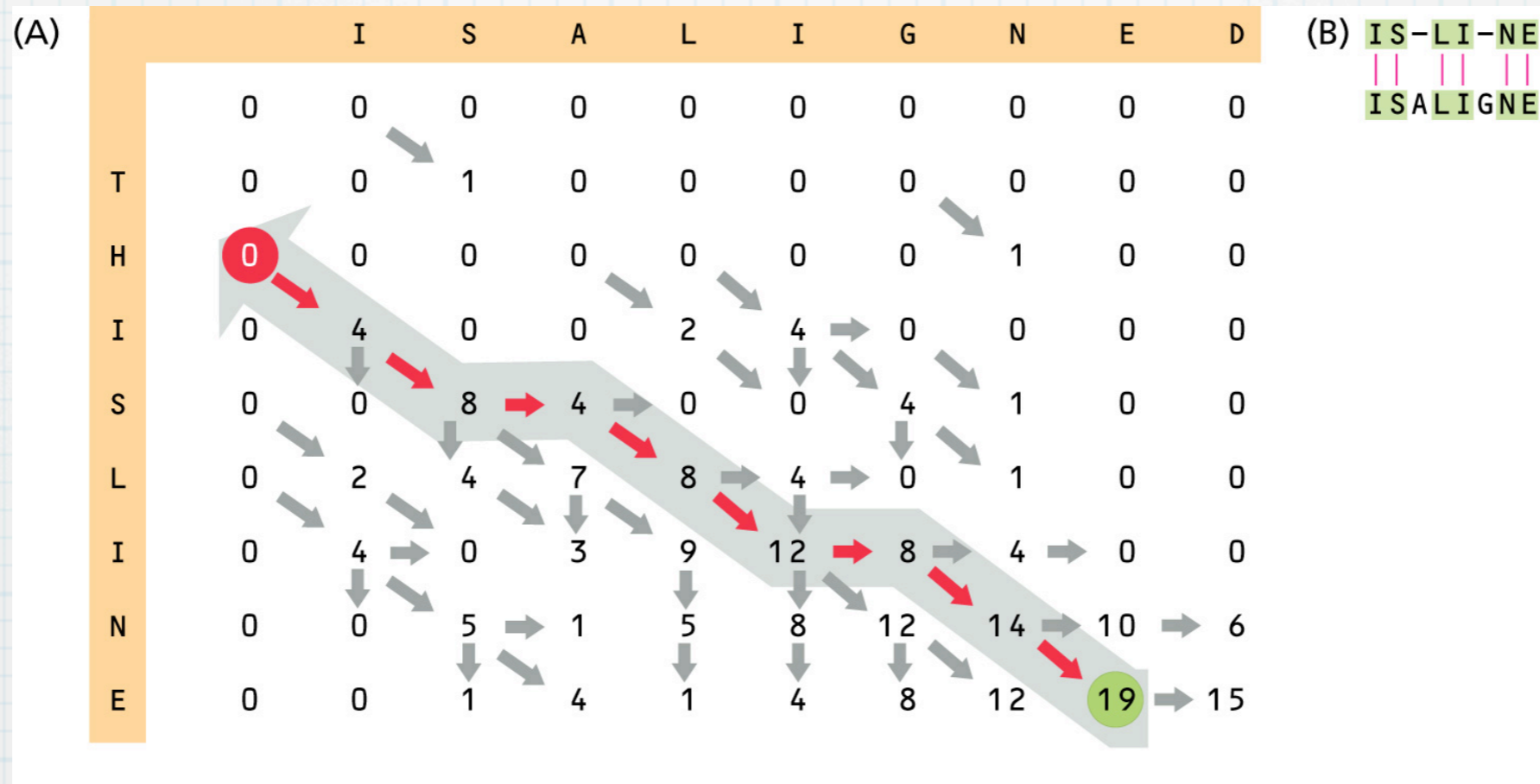
Local Pairwise Alignment

$E = -8$



Local Pairwise Alignment

$$E = -4$$



Linear Space Alignment

- * Is there a linear-space algorithm for the problem?
- * If only the maximal score is needed, the problem is simple
- * But even if the alignment itself is needed, there is a linear-space algorithm (originally due to Hirschberg 1975, and introduced into computational biology by Myers and Miller 1988)

Linear Space Alignment

- * Main observation

$$S_{i,j} = \max_{0 \leq k \leq n} \{ S_{\frac{i}{2},k} + S_{\frac{i}{2},n-k}^r \}$$

$S_{i,j}^r$

score of the best alignment of last
i residues of x with last j residues
of y

Linear Space Alignment

- * Compute $S_{m,n}$ and save row $m/2$
- * Compute $S_{m,n}^r$ and save row $m/2$
- * Find k^* that satisfies

$$S_{\frac{m}{2},k^*} + S_{\frac{m}{2},n-k^*}^r = S_{m,n}$$

- Recurse