Local Alignment

CMSC 423

Representing edits as alignments

```
prin-ciple
|||| |||xx
prinncipal
(1 gap, 2 mm)
```

```
prin-cip-le
|||| |||  |
prinncipal-
(3 gaps, 0 mm)
```

```
misspell
||| ||||
mis-pell
(1 gap)
```

```
prehistoric
   ||||||||
---historic
(3 gaps)
```

```
aa-bb-ccaabb
|x || | | |
ababbbc-a-b-
(5 gaps, 1 mm)
```

```
al-go-rithm-
|| xx ||x |
alKhwariz-mi
(4 gaps, 3 mm)
```

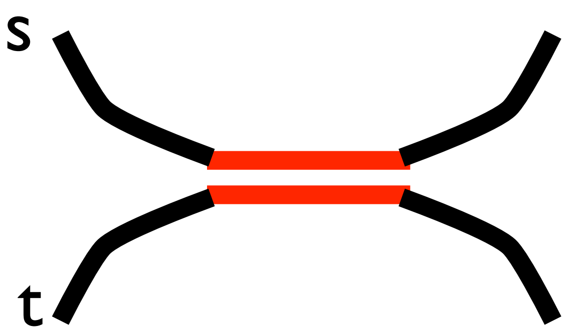
Maximization vs. Minimization

Edit distance:

$$OPT(i,j) = \min \begin{cases} \mathrm{cost}(a_i, b_j) + OPT(i-1, j-1) & \text{match } a_i, b_j \\ \mathrm{gap} + OPT(i-1, j) & a_i \text{ is not matched} \\ \mathrm{gap} + OPT(i, j-1) & b_j \text{ is not matched} \end{cases}$$

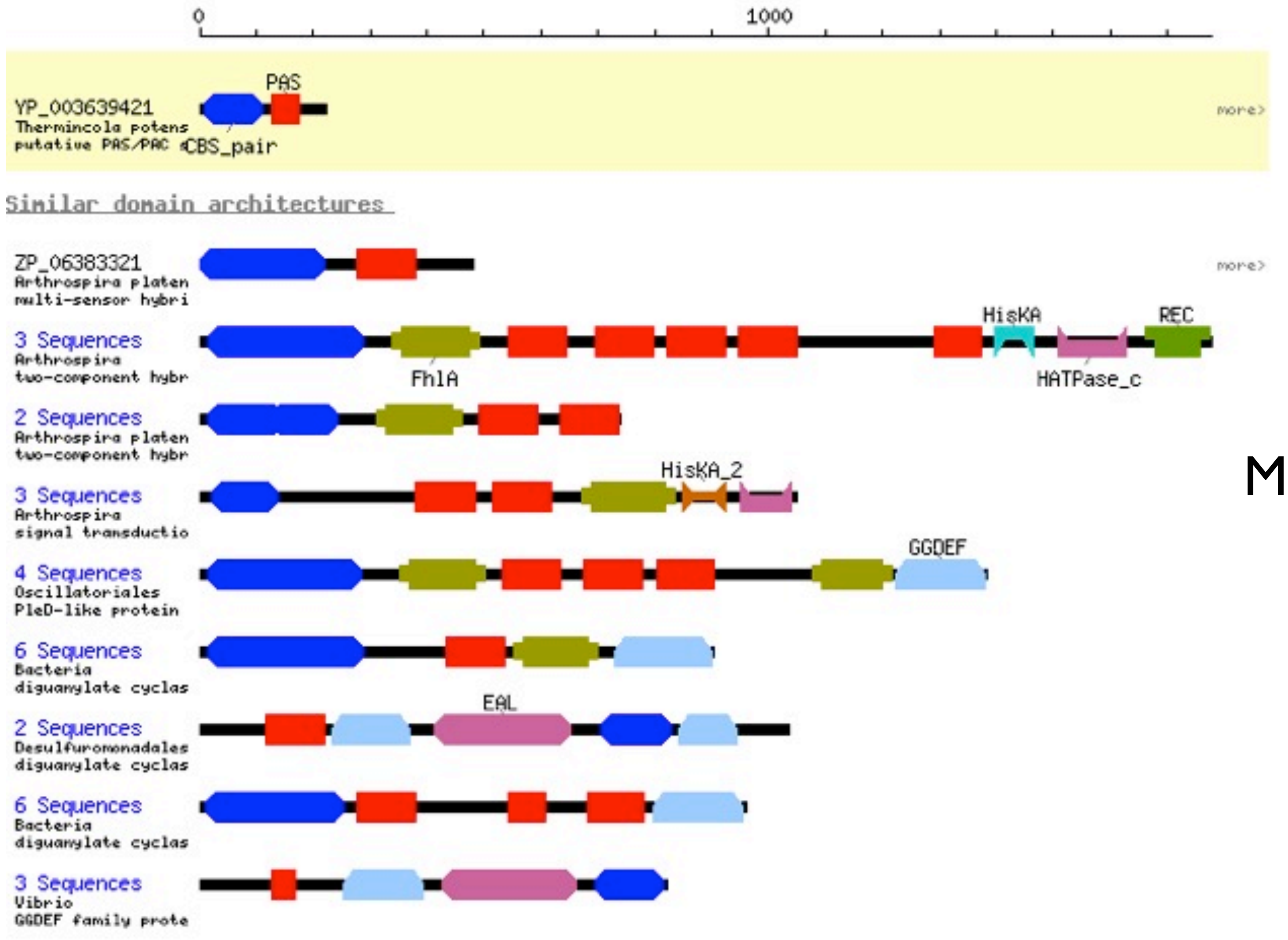
Sequence Similarity: replace min with a max and negate the parameters.

gap penalty → gap benefit (probably negative)
cost → score



Local Alignment

Local alignment between s and t: Best alignment between a subsequence of s and a subsequence of t.



Motivation:

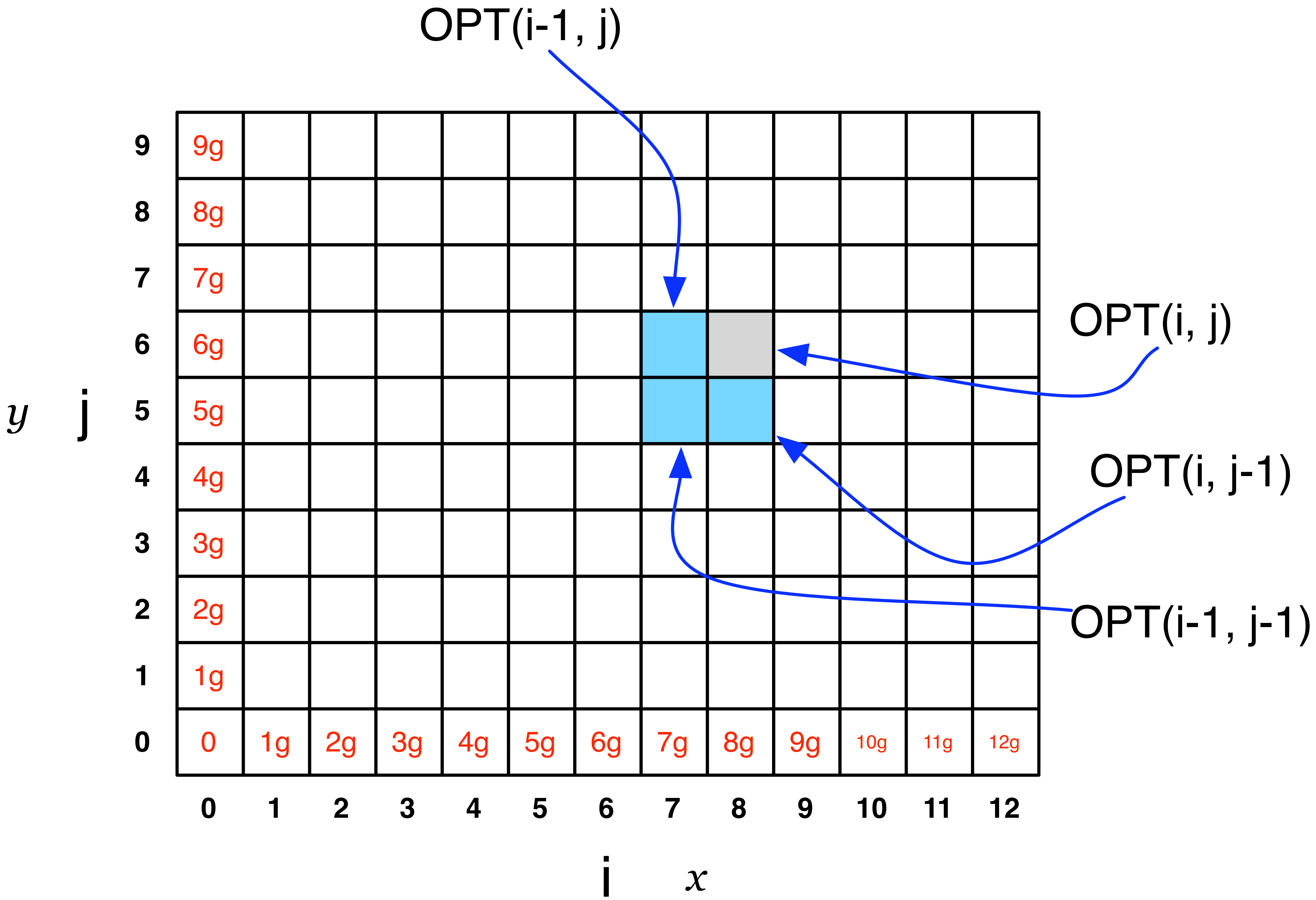
Many genes are composed of *domains*, which are subsequences that perform a particular function.

Recall: Global Alignment Matrix

OPT(*i*,*j*) contains the score for the best alignment between:

the first *i* characters of string *x* [prefix *i* of *x*]

the first *j* character of string *y* [prefix *j* of *y*]



9	9g												
8	8g												
7	7g												
6	6g												
5	5g												
4	4g												
3	3g												
2	2g												
1	1g												
0	0	1g	2g	3g	4g	5g	6g	7g	8g	9g	10g	11g	12g
	0	1	2	3	4	5	6	7	8	9	10	11	12

OPT(i-1, j)

OPT(i, j)

OPT(i, j-1)

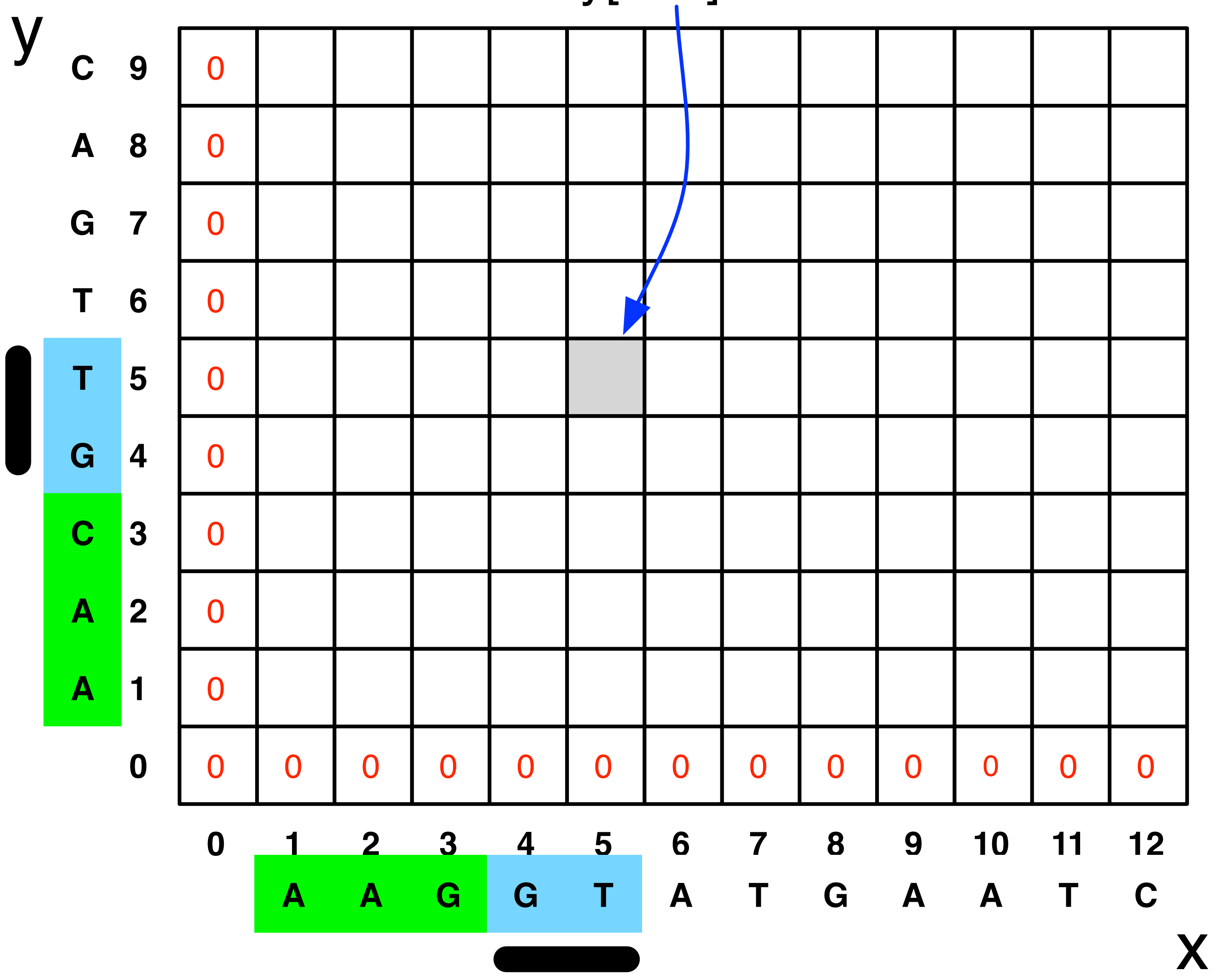
OPT(i-1, j-1)

Local Alignment

New meaning of entry of matrix entry:

A[i, j] = best score between:
some suffix of *x*[1...*i*]
some suffix of *y*[1...*j*]

Best alignment between a suffix of x[1..5] and a suffix of y[1..5]



y														
C	9	0												
A	8	0												
G	7	0												
T	6	0												
T	5	0												
G	4	0												
C	3	0												
A	2	0												
A	1	0												
	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	1	2	3	4	5	6	7	8	9	10	11	12
			A	A	G	G	T	A	T	G	A	A	T	C

x

How do we fill in the local alignment matrix?

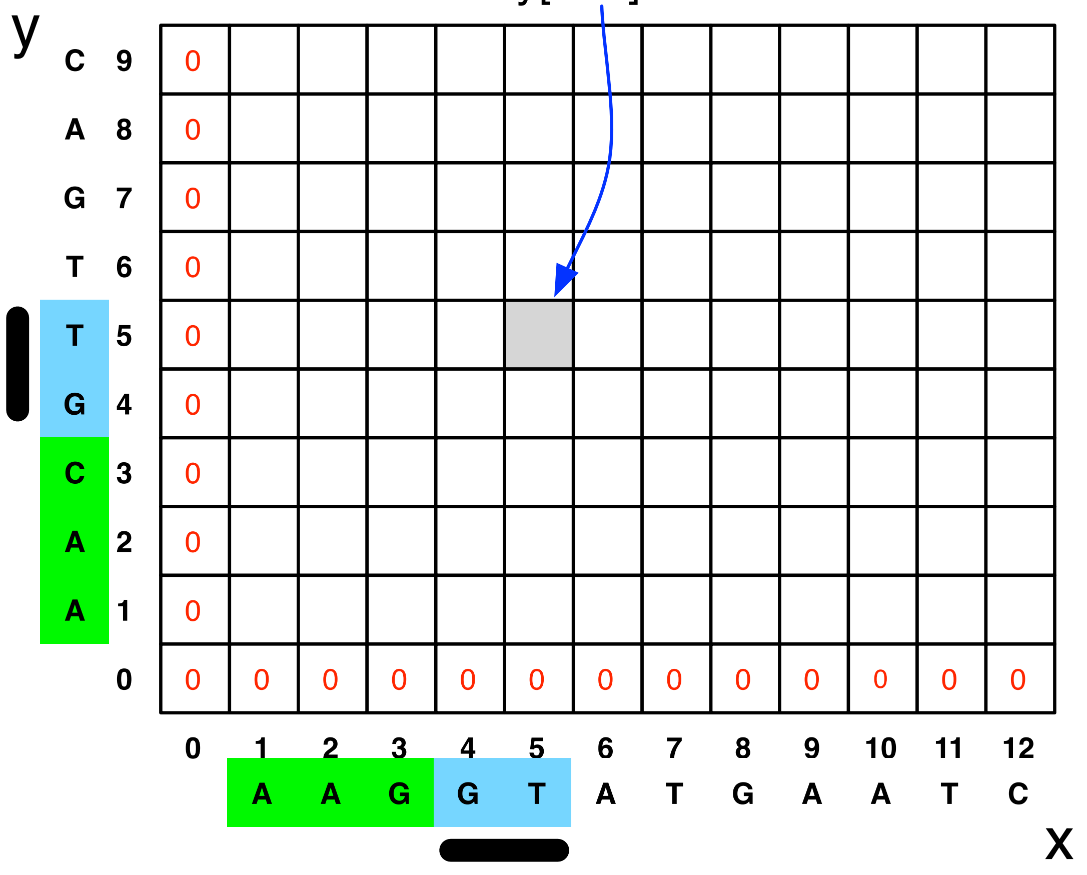
$$A[i,j] = \max \begin{cases} A[i, j-1] + \text{gap} & \text{(1)} \\ A[i-1, j] + \text{gap} & \text{(2)} \\ A[i-1, j-1] + \text{match}(i,j) & \text{(3)} \\ 0 \end{cases}$$

(1), (2), and (3): same cases as before:

gap in x, gap in y, match x and y

New case: 0 allows you to say the best alignment between a suffix of *x* and a suffix of *y* is the empty alignment.

Lets us "start over"



Best alignment between a suffix of x[1..5] and a suffix of y[1..5]

C	9	0												
A	8	0												
G	7	0												
T	6	0												
T	5	0												
G	4	0												
C	3	0												
A	2	0												
A	1	0												
	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	1	2	3	4	5	6	7	8	9	10	11	12
			A	A	G	G	T	A	T	G	A	A	T	C

Local Alignment

- Initialize first row and first column to be 0.

- The score of the best local alignment is the largest value in the entire array.

- To find the actual local alignment:
  - start at an entry with the maximum score
  - traceback as usual
  - stop when we reach an entry with a score of 0

Local Alignment Python Code

```
def local_align(x, y, score=ScoreParam(-7, 10, -5)):
    """Do a local alignment between x and y"""
    # create a zero-filled matrix
    A = make_matrix(len(x) + 1, len(y) + 1)

    best = 0
    optloc = (0,0)

    # fill in A in the right order
    for i in xrange(1, len(y)):
        for j in xrange(1, len(x)):

            # the local alignment recurrance rule:
            A[i][j] = max(
               A[i][j-1] + score.gap,
               A[i-1][j] + score.gap,
               A[i-1][j-1] + (score.match if x[i] == y[j] else score.mismatch),
               0
            )

            # track the cell with the largest score
            if A[i][j] >= best:
                best = A[i][j]
                optloc = (i,j)

    # return the opt score and the best location
    return best, optloc
```

Local Alignment Python Code

```python
def make_matrix(sizex, sizey):
    """Creates a sizex by sizey matrix filled with zeros."""
    return [[0]*sizey for i in xrange(sizex)]


class ScoreParam:
    """The parameters for an alignment scoring function"""
    def __init__(self, gap, match, mismatch):
        self.gap = gap
        self.match = match
        self.mismatch = mismatch
```

Local Alignment Example #1

```
local_align("AGCGTAG", "CTCGTC")
```

	*	A	G	C	G	T	A	G
*	0	0	0	0	0	0	0	0
C	0	0	0	10	3	0	0	0
T	0	0	0	3	5	13	6	0
C	0	0	0	↖10	3	6	8	1
G	0	0	10	3	↖20	13	6	18
T	0	0	3	5	13	↖**30**	23	16
C	0	0	0	13	6	23	25	18

Score(match) = 10
Score(mismatch) = -5
Score(gap) = -7

Note: this table written top-to-bottom instead of bottom-to-top

Local Alignment Example #2

```
local_align("bestoftimes", "soften")
```

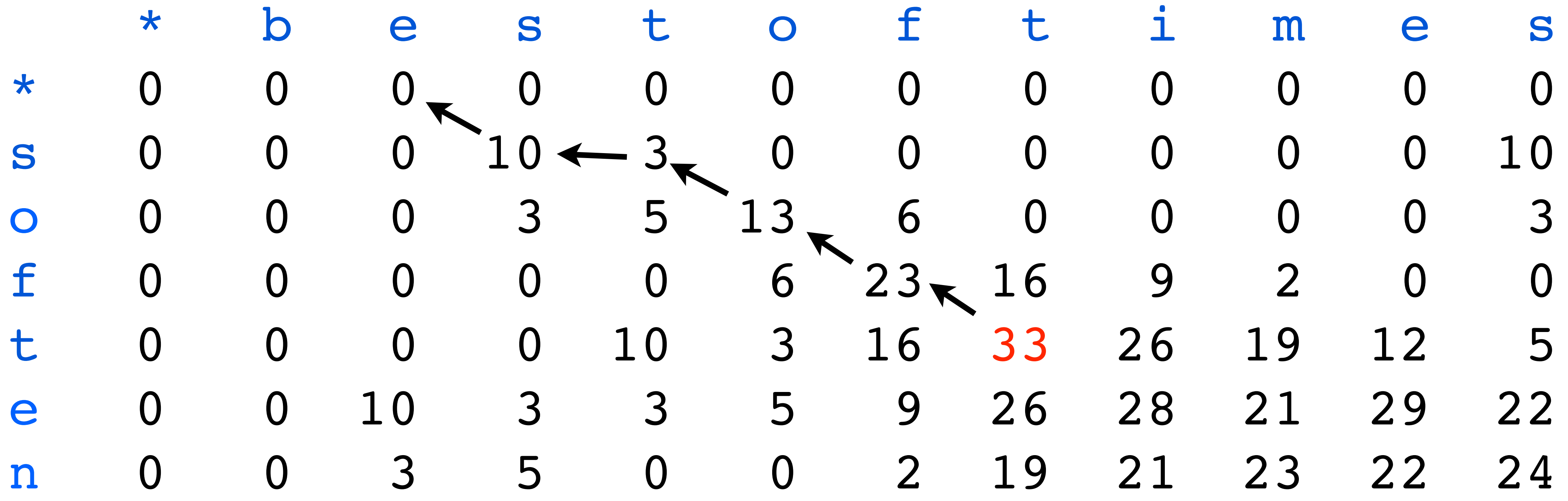
	*	b	e	s	t	o	f	t	i	m	e	s
*	0	0	0	0	0	0	0	0	0	0	0	0
s	0	0	0	10	3	0	0	0	0	0	0	10
o	0	0	0	3	5	13	6	0	0	0	0	3
f	0	0	0	0	0	6	23	16	9	2	0	0
t	0	0	0	0	10	3	16	33	26	19	12	5
e	0	0	10	3	3	5	9					
n	0	0	3	5	0	0	2					

Score(match) = 10
Score(mismatch) = -5
Score(gap) = -7

Note: this table written top-to-bottom instead of bottom-to-top

Local Alignment Example #2

```
local_align("bestoftimes", "soften")
```



	*	b	e	s	t	o	f	t	i	m	e	s
*	0	0	0	0	0	0	0	0	0	0	0	0
s	0	0	0	10	3	0	0	0	0	0	0	10
o	0	0	0	3	5	13	6	0	0	0	0	3
f	0	0	0	0	0	6	23	16	9	2	0	0
t	0	0	0	0	10	3	16	33	26	19	12	5
e	0	0	10	3	3	5	9	26	28	21	29	22
n	0	0	3	5	0	0	2	19	21	23	22	24

```
Score(match) = 10
Score(mismatch) = -5
Score(gap) = -7
```

Note: this table written top-to-bottom instead of bottom-to-top

More Local Alignment Examples

```
Score(match) = 10
Score(mismatch) = -5
Score(gap) = -7
```

```
local_align("catdogfish", "dog")
```

	*	c	a	t	d	o	g	f	i	s	h
*	0	0	0	0	0	0	0	0	0	0	0
d	0	0	0	0	10	3	0	0	0	0	0
o	0	0	0	0	3	20	13	6	0	0	0
g	0	0	0	0	0	13	**30**	23	16	9	2

```
local_align("mississippi", "issp")
```

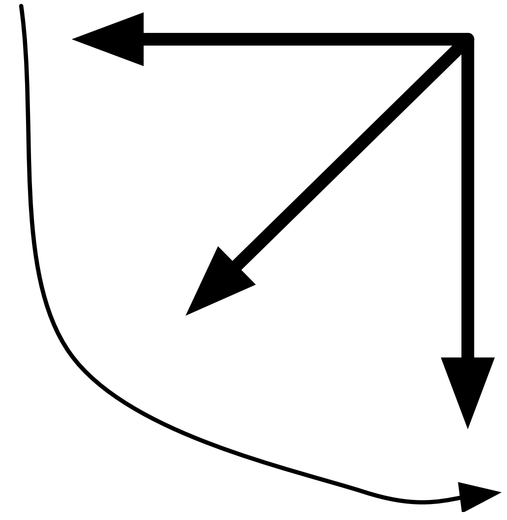
	*	m	i	s	s	i	s	s	i	p	p	i
*	0	0	0	0	0	0	0	0	0	0	0	0
i	0	0	10	3	0	10	3	0	10	3	0	10
s	0	0	3	20	13	6	20	13	6	5	0	3
s	0	0	0	13	30	23	16	30	23	16	9	2
p	0	0	0	6	23	25	18	23	25	**33**	26	19

```
local_align("aaaa", "aa")
```

	*	a	a	a	a
*	0	0	0	0	0
a	0	10	10	10	10
a	0	10	**20**	**20**	**20**

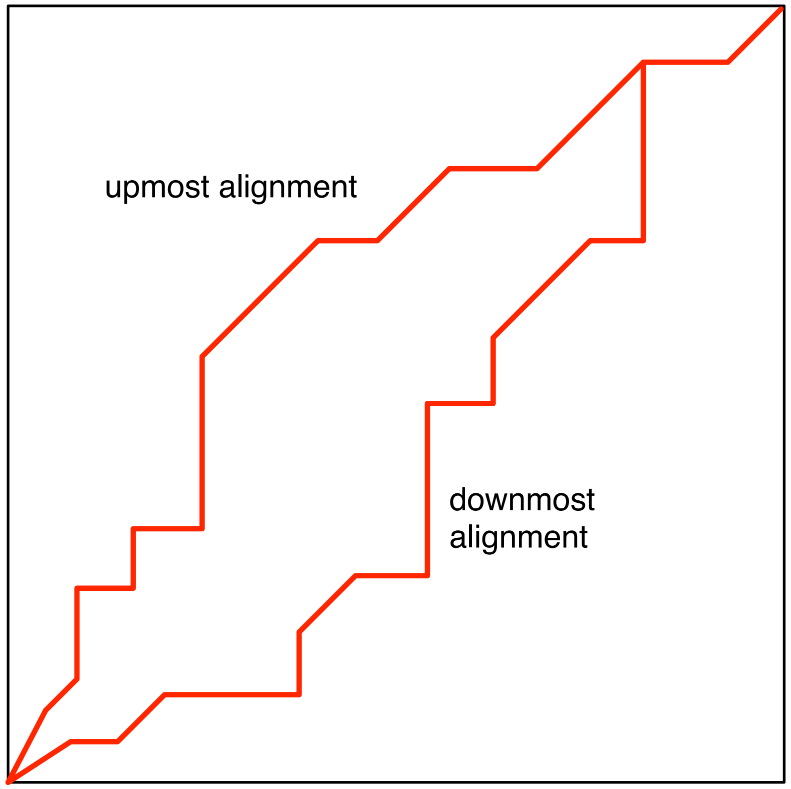
Upmost and Downmost Alignments

When there are ties in the max{}, we have a choice about which arrow to follow.



If we prefer arrows higher in the matrix, we get the *upmost* alignment.

If we prefer arrows lower in the matrix, we get the *downmost* alignment.



Local / Global Recap

- Alignment score sometimes called the "edit distance" between two strings.
- Edit distance is sometimes called Levenshtein distance.

- Algorithm for local alignment is sometimes called "Smith-Waterman"
- Algorithm for global alignment is sometimes called "Needleman-Wunsch"

- Same basic algorithm, however.
- Underlies BLAST