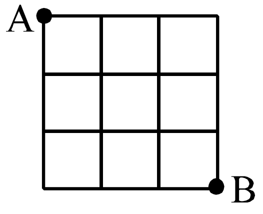
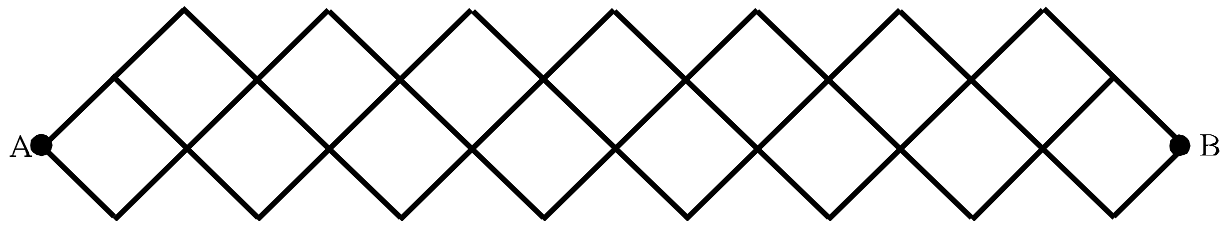
Exercises — Introduction to Dynamic Programming

Quick Concepts



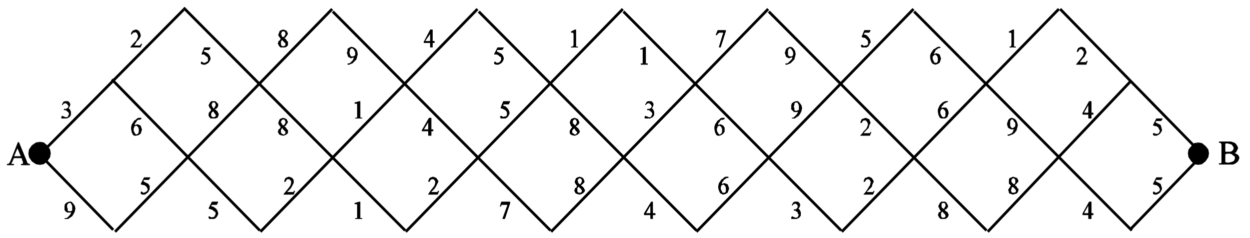
1. How many ways are there to walk from A to B on the grid to the right, without backtracking? *There are 20 ways.*

2. Same question on the grid below.



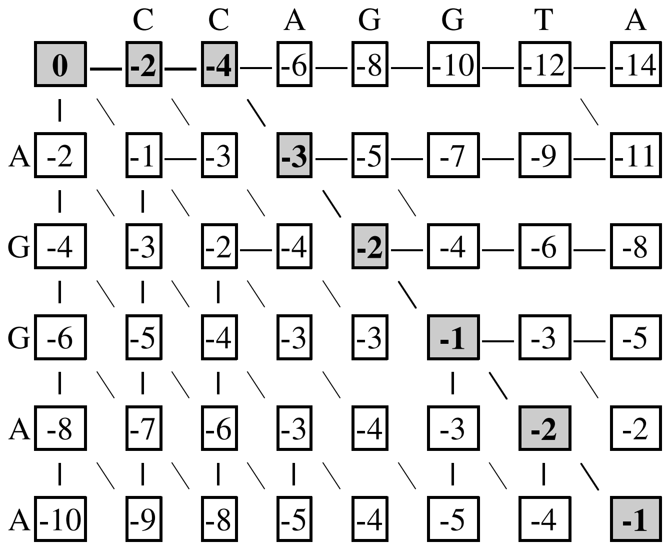
The numbers along the middle layer are the Fibonacci sequence: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597... this last being the number of ways to walk to B.

3. The numbers on the edges of the graph below represent distances. What is the length of the shortest path from A to B? How many routes achieve that length? (The take-away lesson is that even though there are *many* paths from A to B, dynamic programming finds the best one rather quickly!)



The length of the shortest path is 51, and there is only one. It goes: UDDUDUUUDDDUUUDD, where "U" = "up" and "D" = "down."

4. What optimal alignment corresponds to the following scoring matrix:

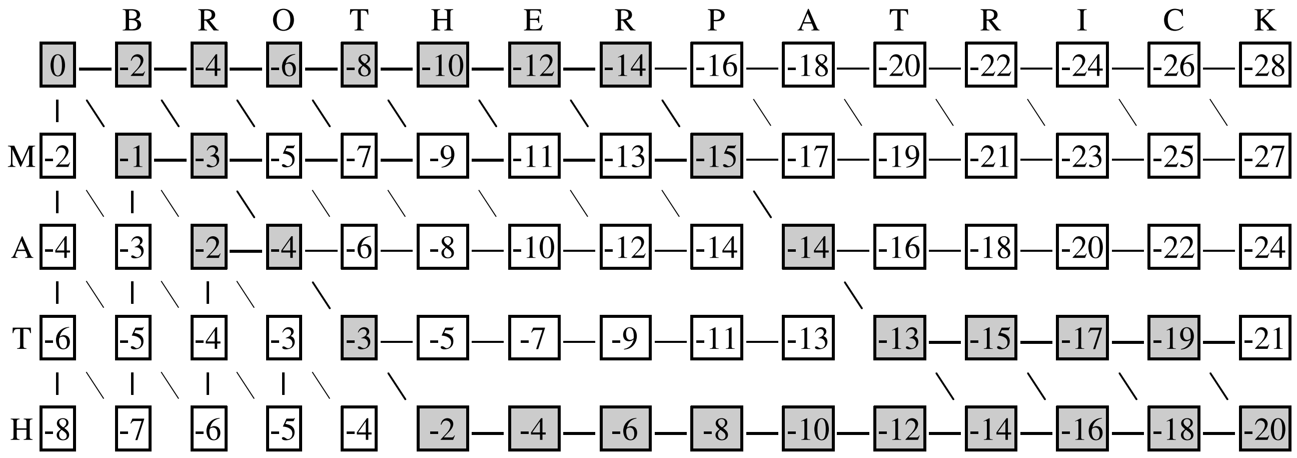


As suggested by the shaded cells, which were obtained by backtracking from the bottom-right cell to the top-left cell, the optimal alignment is

```
CCAGGTA
--AGGAA
```

Presentation Problems

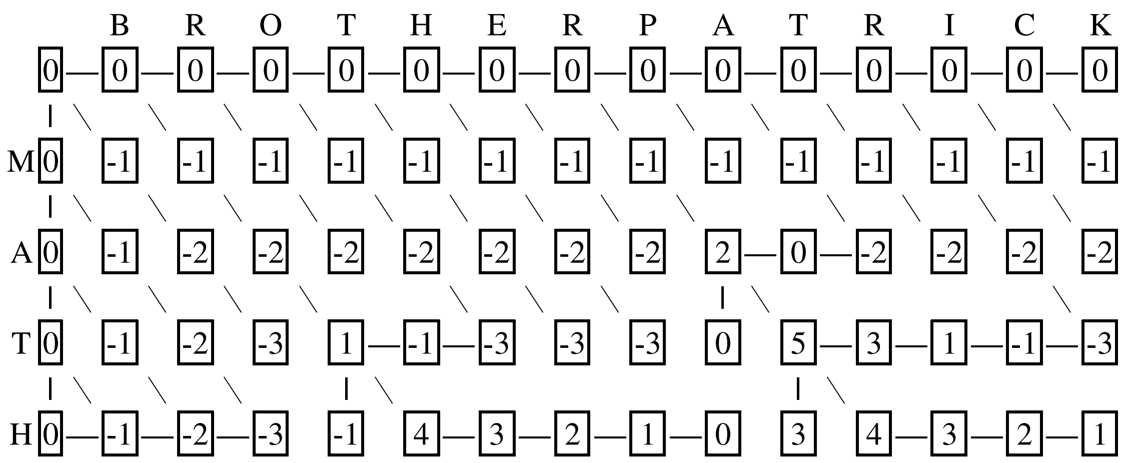
5. How many optimal alignments are indicated by the following scoring matrix?



There are 33 optimal paths, found by tracing the highlighted memos.

6. Tutorial...

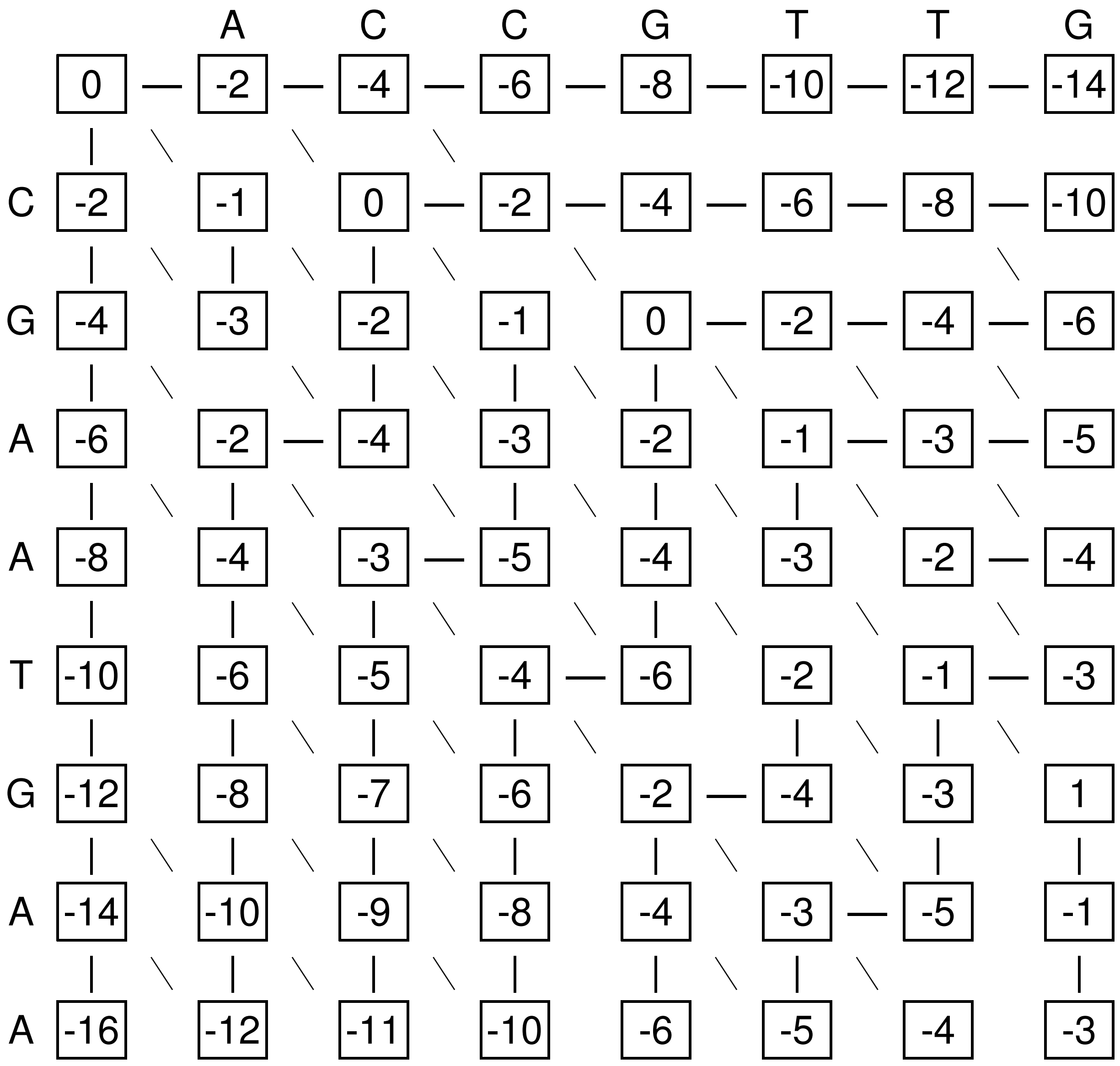
7. Suppose that in Problem 5 we wished to align the same pair of strings using the same scoring system, except that gaps at the end of "BROTHERPATRICK" cost "-2" and gaps at the end of "MATH" cost "-1." How would the scoring matrix be altered? In particular, show that the resulting matrix yields a unique optimal alignment. Here are two copies of the matrix to play with:



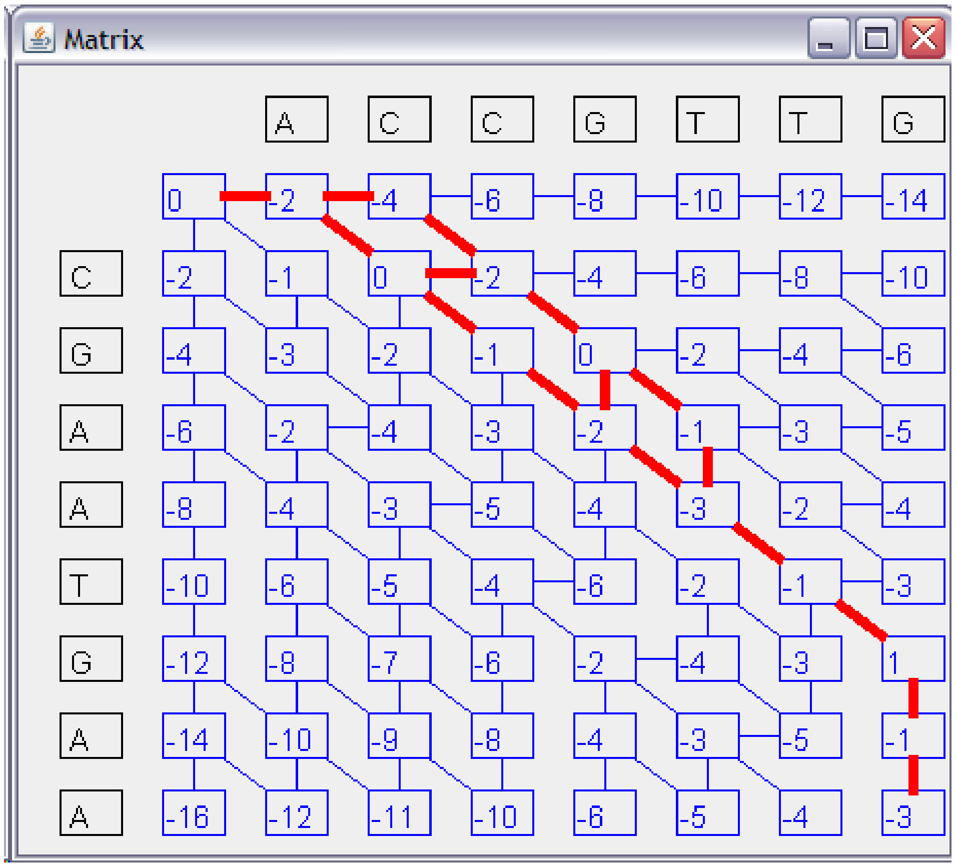
B R O T H E R P A T R I C K
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
M 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
A 0 -1 -2 -2 -2 -2 -2 -2 -2 2 0 -2 -2 -2 -2
T 0 -1 -2 -3 1 -1 -3 -3 -3 0 5 3 1 -1 -3
H 0 -1 -2 -3 -1 4 3 2 1 0 3 4 3 2 1

The resulting scoring matrix is shown above. Note that only the bottom row, and its associated memos, has been altered. Since there is only one way to backtrack from the bottom-right cell to the top-left cell, there is only one optimal alignment.

8. Here is the start of an alignment between "`ACCGTTG`" and "`CGAATGAA`" with match score 2, mismatch penalty -1 and gap penalty -2. Feel free to finish it.


A C C G T T G
0 -2 -4 -6 -8 -10 -12 -14
C -2 -1 0 -2 -4 -6 -8 -10
G -4 -3 -2 -1 0 -2 -4 -6
A -6 -2 -4 -3 -2 -1 -3 -5
A -8 -4 -3 -5 -4 -3 -2 -4
T -10 -6 -5 -4 -6 -2 -1 -3
G -12 -8 -7 -6 -2 -4 -3 1
A -14 -10 -9 -8 -4 -3 -5 -1
A -16 -12 -11 -10 -6 -5 -4 -3

9. Give all optimal alignments between "ACCGTTG" and "CGAATGAA" with match score 2, mismatch penalty -1 and gap penalty -2.

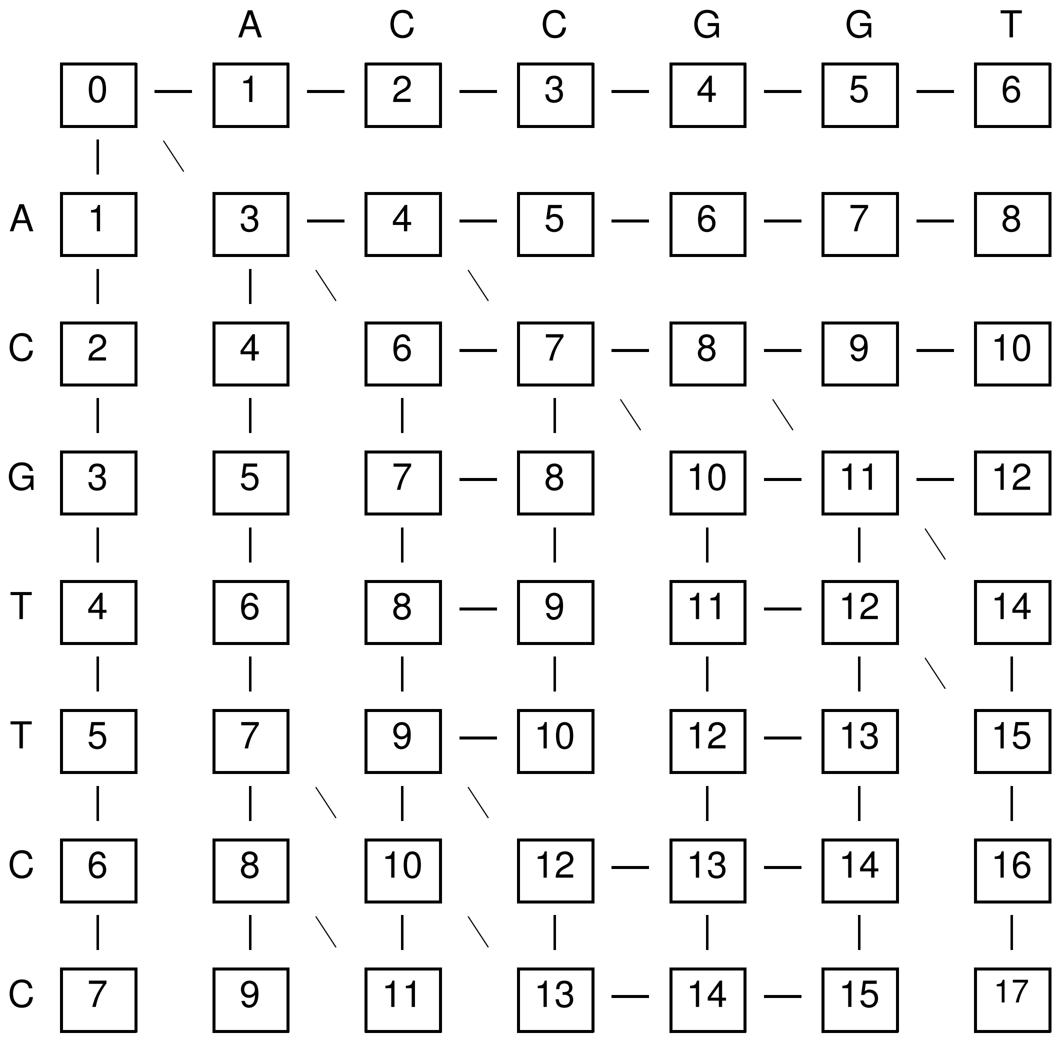


		A	C	C	G	T	T	G
	0	-2	-4	-6	-8	-10	-12	-14
C	-2	-1	0	-2	-4	-6	-8	-10
G	-4	-3	-2	-1	0	-2	-4	-6
A	-6	-2	-4	-3	-2	-1	-3	-5
A	-8	-4	-3	-5	-4	-3	-2	-4
T	-10	-6	-5	-4	-6	-2	-1	-3
G	-12	-8	-7	-6	-2	-4	-3	1
A	-14	-10	-9	-8	-4	-3	-5	-1
A	-16	-12	-11	-10	-6	-5	-4	-3

The figure above (generated by Jim Kupetz's "Align" program, available on the DIMACS website for free) shows the scoring matrix, with all optimal alignments highlighted. There are five optimal alignments, given in the table below:

`ACCGT-TG--` `--CGAATGAA`	`ACCG-TTG--` `--CGAATGAA`	`ACCGT-TG--` `-C-GAATGAA`	`ACCG-TTG--` `-C-GAATGAA`	`ACCGTTG--` `-CGAATGAA`

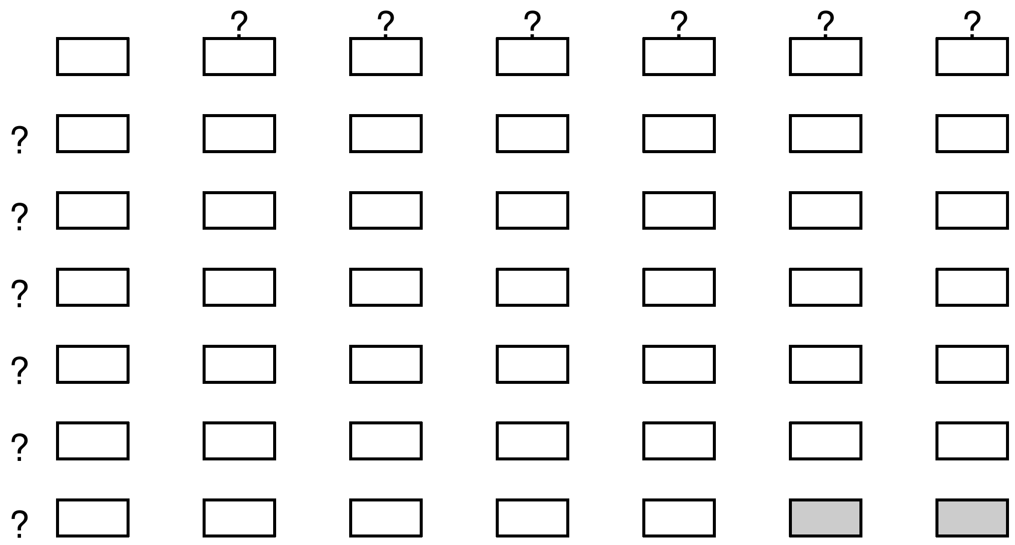
10. Two DNA sequences derived from a common ancestor in an environment in which deletions were much more likely than point mutations. To reflect this in an alignment, a researcher assigns a match score of +3, a mismatch score of -1 and a gap "penalty" of +1. Here is the resulting scoring matrix. You might enjoy finishing it.



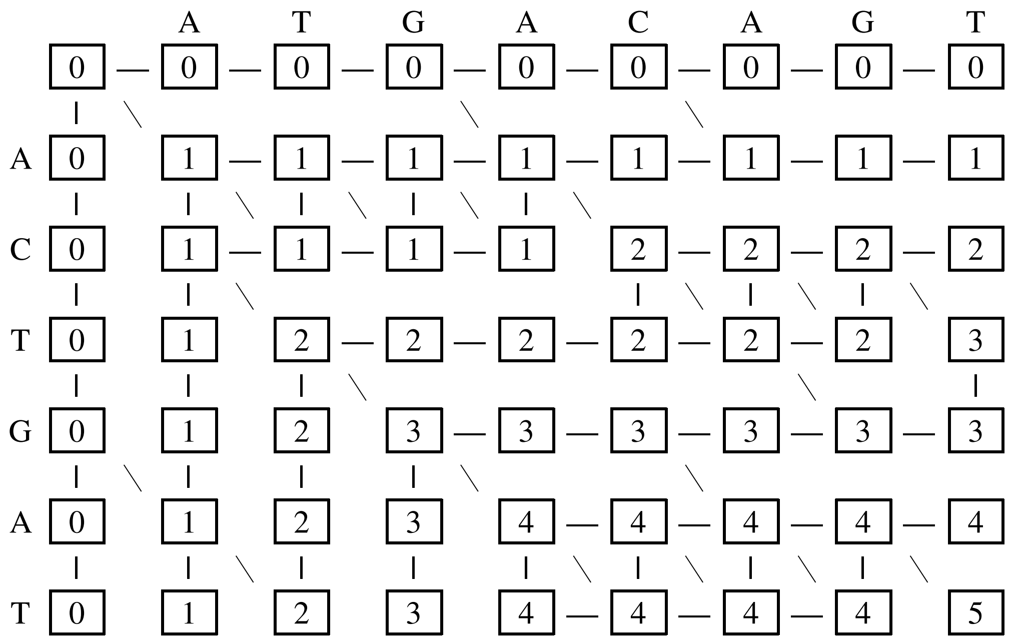
11. Prove that under the (very artificial) scoring system given in the previous problem, an optimal alignment of any two strings will never align two mismatched bases. In fact, what relationship between the gap penalty and the mismatch penalty will guarantee this behavior?
This is so because in any alignment with a mismatch, we may replace that mismatch with two gaps, and obtain a better score. For example, as shown to the right, if A is mismatched with C, that column will score –1, but if we insert two gaps and use two columns instead of one, we get a score of +1+1 = +2 for those two columns.

```
**A**   |   **-A**
**C**   |   **C-**
```

12. Under an alignment of two nucleotide strings with match score +1, mismatch penalty -1 and gap penalty -2, how large could the difference be between the values in the two boxes shaded below? Could they possibly be the same? *The largest the difference could be is 3,*



13. In the *Longest Common Subsequence* problem, two strings are given and one wishes to find the length of the longest (not-necessarily contiguous) common subsequence in those strings. For example, the strings ATGACAGT and ACTGAT have "ACGT" as a common substring, but there is a longer one. Devise a dynamic programming solution to the *Longest Common Subsequence* problem for these two strings. To help you get started, I've begun a matrix below.

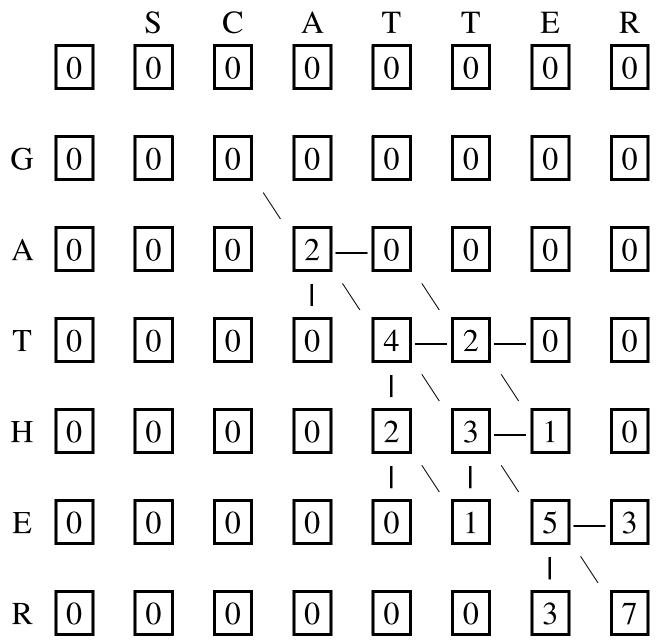


Simply score +1 for a match and 0 for gaps and mismatches. The result is shown above.

Exercises — Local Alignment

Quick Concepts:

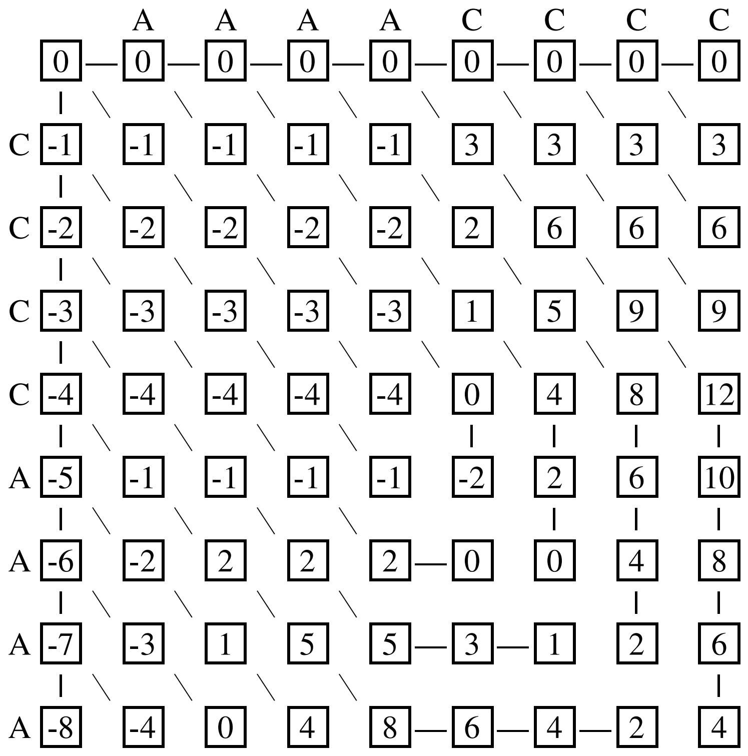
1. What optimal local alignment is suggested by the scoring matrix below?



$$\begin{array}{c|cccccccc}
 & & S & C & A & T & T & E & R \\
\hline
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
G & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
A & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\
T & 0 & 0 & 0 & 0 & 4 & 2 & 0 & 0 \\
H & 0 & 0 & 0 & 0 & 2 & 3 & 1 & 0 \\
E & 0 & 0 & 0 & 0 & 0 & 1 & 5 & 3 \\
R & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 7
\end{array}$$

The largest number we see is a "7." If we trace from the 7 back to, but not including, a "0," we find the subsequence "`ATTER`" aligned with "`ATHER`".

2. What are the various gap, match and mismatch scores in the following global alignment matrix? What are the four optimal alignments, depending on whether end gaps do or don't count for each string?



Initial gaps for string1 (the top string) are –1
Initial gaps for string2 (the left string) are 0
All other gaps are –2
Match is 3, and mismatch is –1

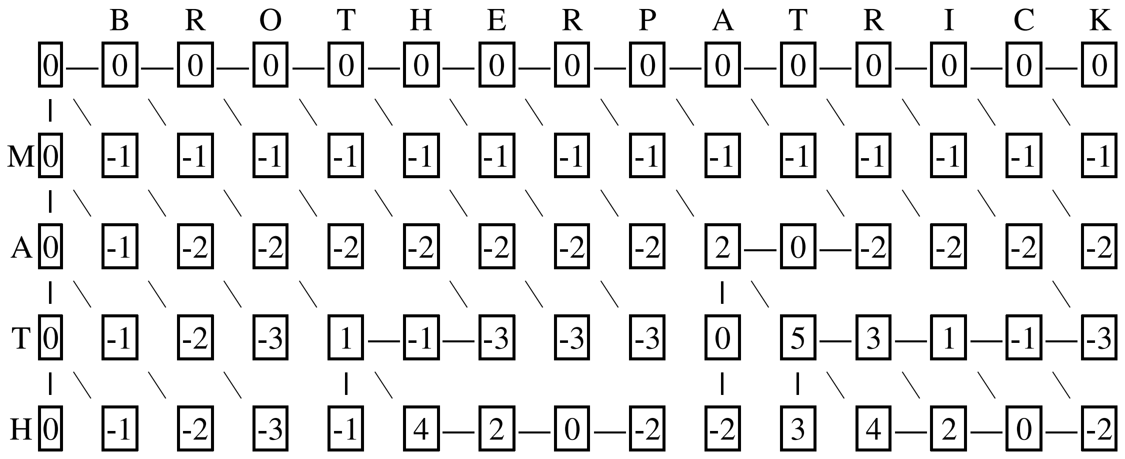
Here are the four optimal global alignments:

If all end gaps are –2	*If gaps don't count at the end of String1*	*If gaps don't count at the end of String2*	*If gaps don't count at the end of either string*
`AAAACCCC----` `----CCCCAAAA`	`AAAACCCC----` `----CCCCAAAA`	`----AAAACCCC` `CCCCAAAA----`	`AAAACCCC----` `----CCCCAAAA`

Presentation Problems:

3. This is a scoring matrix for a pair of strings from yesterday:
   a. Does it charge for initial gaps? *No*
   b. How many points are given for a matched column? *3*
   c. How many points are given for a mismatched column? *–1*
   d. Give all optimal alignments if there is no charge for initial gaps but there is a charge (of –2) for end gaps. *This matrix, as shown, does charge –2 for end gaps. So we find the optimal alignment straight from what is given by tracing back from the bottom-right corner to the top-left corner. There are four optimal alignments altogether. One is shown, and the other three may be obtained by moving the "H" in "MATH" to any of the last four positions:*

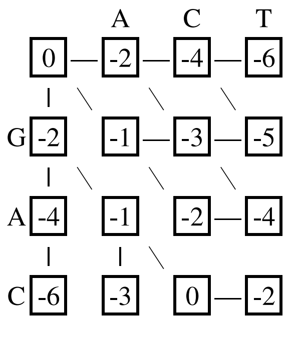
```
BROTHERPATRICK
-------MATH---
```

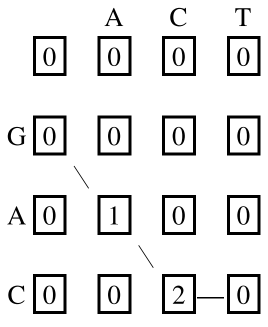

B R O T H E R P A T R I C K
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
M 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
A 0 -1 -2 -2 -2 -2 -2 -2 -2 2 0 -2 -2 -2 -2
T 0 -1 -2 -3 1 -1 -3 -3 -3 0 5 3 1 -1 -3
H 0 -1 -2 -3 -1 4 2 0 -2 -2 3 4 2 0 -2

   e. Give all optimal alignments if there is no charge for gaps at the start nor at the end. *Here we find two optimal alignments. The one shown above, and this one:*

```
BROTHERPATRICK
-MATH---------
```

4. Suppose two sequences are aligned twice, once globally and once locally, under the same match, mismatch and gap scoring. One such example is shown here. Is it the case that the set of edges in the Smith-Waterman matrix (on the right) must be a subset of the edges in the Needleman-Wunsch matrix (on the left)?

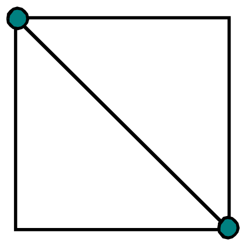




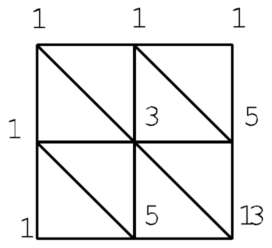
No, not always. For example, if you align GAA with AG, scoring +3 for matches and –2 for all mismatches and gaps.

5. There are three possible global alignments between the strings "A" and "T," shown to the right. These can be enumerated by counting paths from dot to dot on the graph shown below the alignments.

```
A  -A  A-
T  T-  -T
```



a. How many global alignments are there between the strings "AC" and "TG"? On what graph would you count paths to answer this question? *There are 13, as can be found by counting paths on the graph below:*



b. Show that there are 129 global alignments between a string of length 3 and a string of length 4. *These can be counted by extending the graph shown here one more columns to the right, and then two more rows down.*

c. There are 11 *local* alignments between a string of length 1 and a string of length 2. For example, if the strings were "AT" and "C," then we will have 3 ways to globally align the "A" with the "C," 3 ways to globally align the "T" with the "C," and 5 ways to globally align "AT" with "C." How many local alignments are possible between a string of length 2 and a string of length 3? *The table shown below shows the number of ways to select substrings from each of these given strings, and then the number of ways to globally align those (local) substrings. For the purposes of the solution, we'll assume that AB is the first string and XYZ is the second. The final answer is 141.*

Length from string AB	*Length from string XYZ*	*# of selections of a string of that length from AB*	*# of selections of a string of that length from XYZ*	*# of global alignments of those two strings*	*# of ways to align two strings of these given lengths*
1	*1*	*2*	*3*	*3*	*18*
2	*1*	*1*	*3*	*5*	*15*
1	*2*	*2*	*3*	*5*	*30*
2	*2*	*1*	*3*	*13*	*39*
1	*3*	*2*	*1*	*7*	*14*
2	*3*	*1*	*1*	*25*	*25*
				Total	***141***

6. Find an optimal local alignment using the scoring matrix below, with gap penalty -4.

$$\begin{tikzcd}
 & & W & R & A & P & S \\
 & \boxed{0} & \boxed{0} & \boxed{0} & \boxed{0} & \boxed{0} & \boxed{0} \\
S & \boxed{0} & \boxed{0} & \boxed{0} & \boxed{1} & \boxed{0} & \boxed{4} \arrow[d, dash] \\
W & \boxed{0} & \boxed{11} \arrow[r, dash] \arrow[d, dash] \arrow[dr, dash] & \boxed{7} \arrow[r, dash] \arrow[dr, dash] & \boxed{3} & \boxed{0} & \boxed{0} \\
I & \boxed{0} & \boxed{7} \arrow[d, dash] \arrow[dr, dash] & \boxed{8} \arrow[dr, dash] & \boxed{6} \arrow[r, dash] \arrow[dr, dash] & \boxed{2} & \boxed{0} \\
P & \boxed{0} & \boxed{3} \arrow[dr, dash] & \boxed{5} \arrow[dr, dash] & \boxed{7} & \boxed{13} \arrow[r, dash] \arrow[d, dash] \arrow[dr, dash] & \boxed{9} \\
E & \boxed{0} & \boxed{0} & \boxed{3} & \boxed{4} & \boxed{9} & \boxed{13}
\end{tikzcd}$$

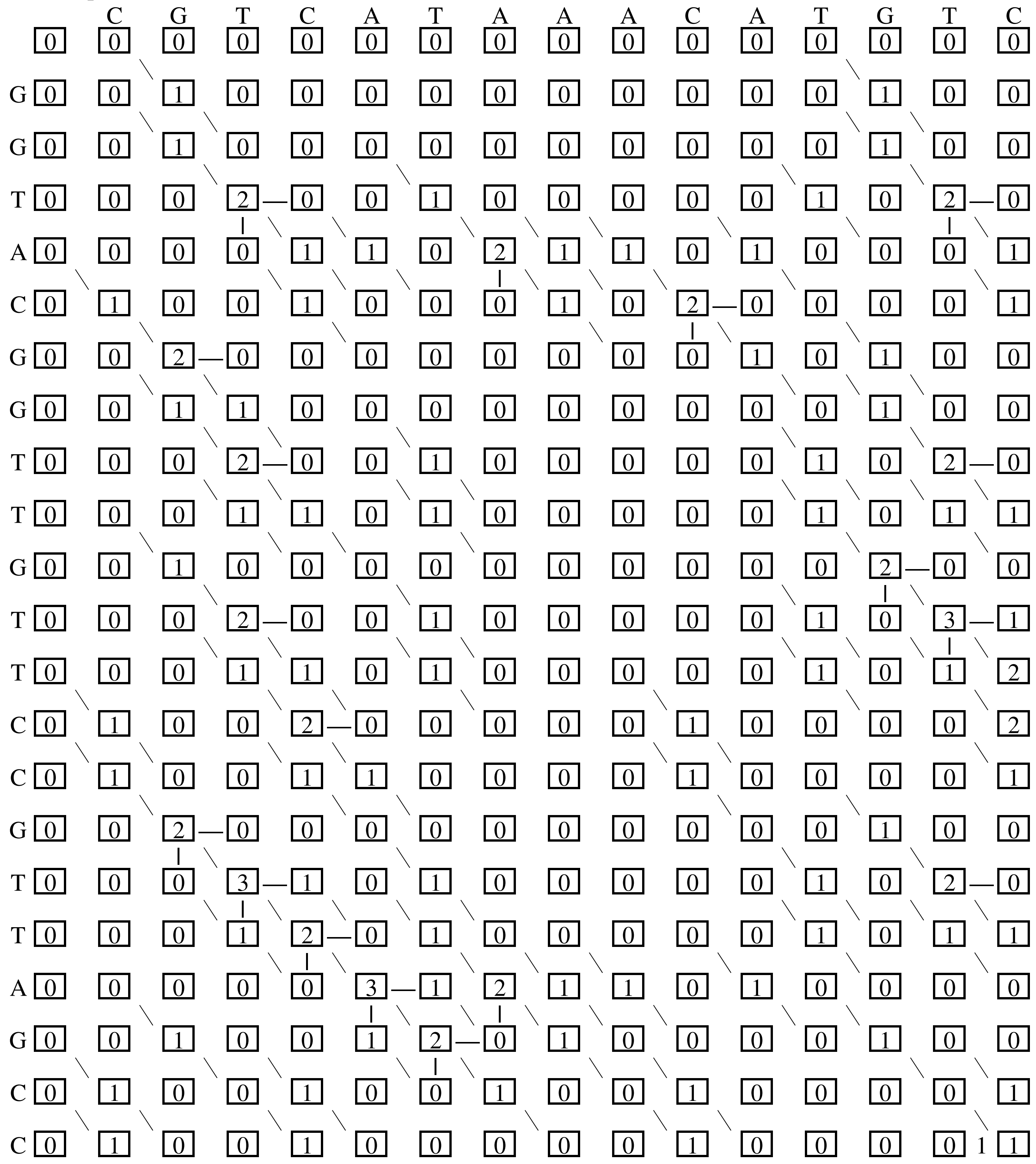
The completed scoring matrix is shown above. This corresponds to the alignment:

```
WRAPS
W-IPE
```

Note that the column aligning "S" with "E" may be left off for an equally optimal alignment.

	A	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	Y
A	**4**	0	-2	-1	-2	0	-2	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-2	-3	-2
C	0	**9**	-3	-4	-2	-3	-3	-1	-3	-1	-1	-3	-3	-3	-3	-1	-1	-1	-2	-2
D	-2	-3	**6**	2	-3	-1	-1	-3	-1	-4	-3	1	-1	0	-2	0	1	-3	-4	-3
E	-1	-4	2	**5**	-3	-2	0	-3	1	-3	-2	0	-1	2	0	0	0	-3	-3	-2
F	-2	-2	-3	-3	**6**	-3	-1	0	-3	0	0	-3	-4	-3	-3	-2	-2	-1	1	3
G	0	-3	-1	-2	-3	**6**	-2	-4	-2	-4	-3	-2	-2	-2	-2	0	1	0	-2	-3
H	-2	-3	1	0	-1	-2	**8**	-3	-1	-3	-2	1	-2	0	0	-1	0	-2	-2	2
I	-1	-1	-3	-3	0	-4	-3	**4**	-3	2	1	-3	-3	-3	-3	-2	-2	1	-3	-1
J	-1	-3	-1	1	-3	-2	-1	-3	**5**	-2	-1	0	-1	1	2	0	0	-3	-3	-2
L	-1	-1	-4	-3	0	-4	-3	2	-2	**4**	2	-3	-3	-2	-2	-2	-2	3	-2	-1
M	-1	-1	-3	-2	0	-3	-2	1	-1	2	**5**	-2	-2	0	-1	-1	-1	-2	-1	-1
N	-2	-3	1	0	-3	0	-1	-3	0	-3	-2	**6**	-2	0	0	1	0	-3	-4	-2
P	-1	-3	-1	-1	-4	-2	-2	-3	-1	-3	-2	-1	**7**	-1	-2	-1	1	-2	-4	-3
Q	-1	-3	0	2	-3	-2	0	-3	1	-2	0	0	-1	**5**	1	0	0	-2	-2	-1
R	-1	-3	-2	0	-3	-2	0	-3	2	-2	-1	0	-2	1	**5**	-1	-1	-3	-3	-2
S	1	-1	0	0	-2	0	-1	-2	0	-2	-1	1	-1	0	-1	**4**	1	-2	-3	-2
T	-1	-1	1	0	-2	1	0	-2	0	-2	-1	0	1	0	-1	1	**4**	-2	-3	-2
V	0	-1	-3	-2	-1	-3	-3	3	-2	1	1	-3	-2	-2	-3	-2	-2	**4**	-3	-1
W	-3	-2	-4	-3	1	-2	-2	-3	-3	-2	-1	-4	-4	-2	-3	-3	-3	-3	**11**	2
Y	-2	-2	-3	-2	3	-3	2	-1	-2	-1	-1	-2	-3	-1	-2	-2	-2	-1	2	**7**

7. Find all optimal local alignments for the Local Alignment matrix shown below. What is the optimal score?



		C	G	T	C	A	T	A	A	A	C	A	T	G	T	C
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0
G	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0
T	0	0	0	2	0	0	1	0	0	0	0	0	1	0	2	0
A	0	0	0	0	1	1	0	2	1	1	0	1	0	0	0	1
C	0	1	0	0	1	0	0	0	1	0	2	0	0	0	0	1
G	0	0	2	0	0	0	0	0	0	0	0	1	0	1	0	0
G	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0
T	0	0	0	2	0	0	1	0	0	0	0	0	1	0	2	0
T	0	0	0	1	1	0	1	0	0	0	0	0	1	0	1	1
G	0	0	1	0	0	0	0	0	0	0	0	0	0	2	0	0
T	0	0	0	2	0	0	1	0	0	0	0	0	1	0	3	1
T	0	0	0	1	1	0	1	0	0	0	0	0	1	0	1	2
C	0	1	0	0	2	0	0	0	0	0	1	0	0	0	0	2
C	0	1	0	0	1	1	0	0	0	0	1	0	0	0	0	1
G	0	0	2	0	0	0	0	0	0	0	0	0	0	1	0	0
T	0	0	0	3	1	0	1	0	0	0	0	0	1	0	2	0
T	0	0	0	1	2	0	1	0	0	0	0	0	1	0	1	1
A	0	0	0	0	0	3	1	2	1	1	0	1	0	0	0	0
G	0	0	1	0	0	1	2	0	1	0	0	0	0	1	0	0
C	0	1	0	0	1	0	0	1	0	0	1	0	0	0	0	1
C	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	1

There are three optimal alignments, corresponding to the three "3" entries in the matrix, shown below:

```
TGT      CGT      CGTTA
TGT      CGT      CGTTA
```

8. Here are some triangles full of numbers.

```
              1
           1     1
        1     2     1
     1     3     3     1
  1     4     6     4     1
1    5    10    10    5    1
1    6    15    20    15    6    1
1    7    21    35    35    21    7    1
```

```
              1
           1     1
        1     3     1
     1     5     5     1
  1     7    13     7     1
1    9    25    25    9    1
1    11   41   63   41   11   1
1    13   61   129  129  61   13   1
```

a. What is the rule for generating them?
b. What are the next two rows in each of them?
c. What does the one on the right have to do with string alignment?
d. What is the sum of the numbers in each row?
   1. In the left triangle, you can find an explicit formula. *2n for row n*
   2. In the right triangle, you should find a recursive formula. Can you guess what the ratio of successive row-sums approaches. *S(n) = 2S(n – 1) + S(n – 2). That is, each row sums to twice the row above it plus the row two above it.*
e. Just how far off-topic has this problem gotten? *Rather...*

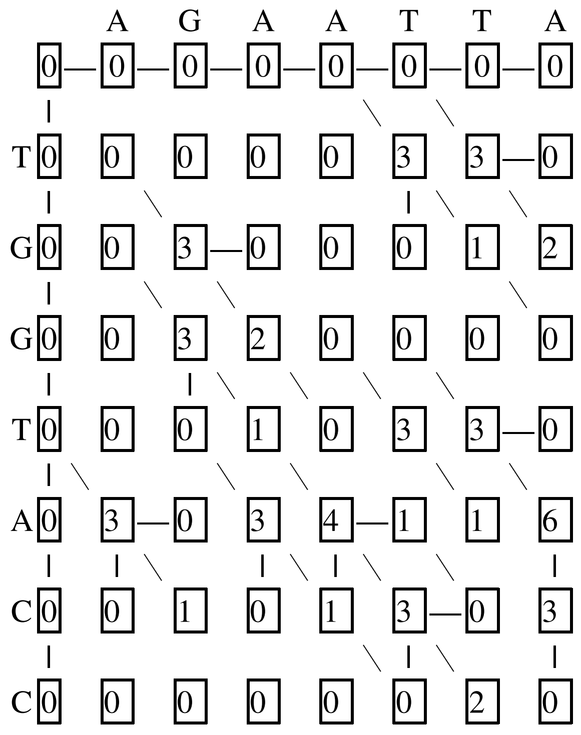
This problem is essentially the same as problem five, except that we haven't drawn the graph, and we've arranged the numbers in a Pascal's Triangle-like configuration. Pascal's triangle is shown on the left, and the "counting alignments" triangle is shown on the right.

9. Devise a way to find an optimal local alignment of "ACCACTT" and "TGGTACC" if:
a. Matches are worth +3
b. Mismatches of a purine with a purine, or a pyrimidine with a pyrimidine are worth –1
c. Mismatches of a purine with a pyrimidine are worth –2
d. Gaps are worth –3

Here is a scoring matrix to get you started:

The idea is to score the two types of mismatches differently when filling in the scoring matrix. The finished matrix is shown to the right. The optimal alignment has score 6, and is shown below:

TA
TA

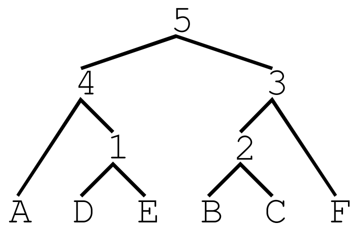


Exercises — Phylogeny

Quick Concepts:

1. Use the UPGMA method to discern the phylogeny tree on the following six species, where the distances have been scaled and rounded to be nice integers. Draw your answer as a rooted tree.

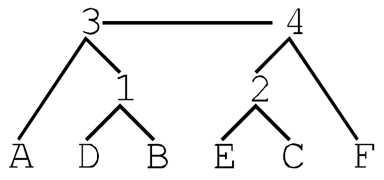
	A	B	C	D	E	F
A	0	4	11	7	9	10
B		0	3	12	4	6
C			0	9	7	7
D				0	2	13
E					0	8
F						0



The figure to the right shows the resulting tree. The numbers in the internal nodes show the order in which the nodes are constructed.

2. Suppose you are given a treelike matrix and you are told that all the weights on the edges are "1." Does that make it any easier to reconstruct the tree? Try it with the following treelike matrix, which does correspond to such an (unrooted) tree:

	A	B	C	D	E	F
A	0	3	4	3	4	3
B		0	5	2	5	4
C			0	5	2	3
D				0	5	4
E					0	3
F						0



It makes it much easier to create the tree. Start with the entries having distance 2, and after you join them, subtract "1" from the other distances to those nodes, and iterate. The resulting tree is shown on the right, with the internal numbers indicating the order in which I paired things up, though this order depends on how one finds the tree, not on any particular algorithm.

3. Please make sure you have an account on Biology Student Workbench. Create a new session called “Homework 5.”

4. Recall from the workshop that one type of similarity is that given by:

$$S = (S_{real} - S_{rand}) / (S_{ident} - S_{rand})$$

where

S_{real} = actual alignment score of two sequences
S_{rand} = average alignment of the two sequences after many random shuffles of the sequences
S_{ident} = average of the two scores obtained by aligning the sequences with themselves

Explain why S will usually be between 0 and 1, that S will be close to 1 when the sequences are very closely related, and S will be close to 0 when the sequences are not related.

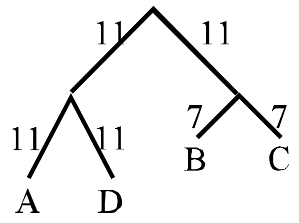
Informally, if the sequences are very similar, then S_{real} and S_{ident} will be close to one another, since aligning the sequences with one another will be nearly the same as aligning the sequence with itself. Thus the numerator will be very close to the denominator, and the fraction will be close to 1. If the sequences are very dissimilar, then aligning them with one another will give a score similar to aligning them with random sequences, making the numerator nearly 0.

Presentation Problems:

5. *<Tutorail>*

6. Below is the distance matrix for four species. Use UPGMA to draw the phylogeny tree of these species with edge lengths proportional to the distances between the joined pairs. To get the distances between the joined pairs, each time you add a new node, let it split the distance between the two joined species, as given in the matrix.

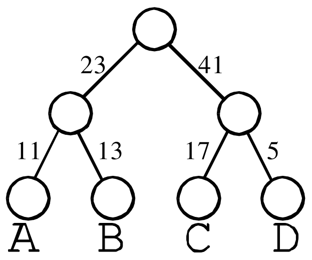
	A	B	C	D
A	0	32	14	22
B		0	14	19
C			0	25
D				0



The resulting tree is shown to the right. The lengths were rounded down for clarity.

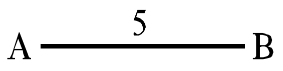
7. What treelike matrix arises from the weighted tree below?

	A	B	C	D
A	0	24	92	80
B		0	94	82
C			0	22
D				0

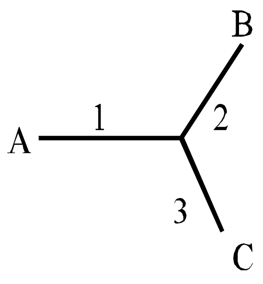


8. Recall that a "treelike" distance matrix is one which arose from a weighted binary tree by setting the distance between each pair of vertices to the sum of the weights on the edges on the path between them. The matrices below are treelike. In each case, obtain and draw the corresponding (unrooted) phylogeny tree using whatever method you wish. *Trees are shown.*

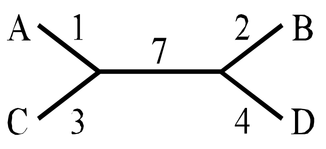
	A	B
A	0	5
B		0



	A	B	C
A	0	3	4
B		0	5
C			0



	A	B	C	D
A	0	10	4	12
B		0	12	6
C			0	14
D				0



9. Perform one (or more, if you really have nothing better to do) iteration of the algorithm of Studier and Keppler on the distance matrix below. That means you should find the closest pair, join them and replace their entries in the matrix with a new entry whose distances to the remaining entries are given by the formula on Handout #3.

	A	B	C	D	E
A	0	7	3	3	11
B		0	9	1	21
C			0	13	2
D				0	5
E					0

Here is the matrix after the first iteration. The marginal sums are shown.

	A	B	CE	D	
A	0	7	6	3	16
B	7	0	14	1	22
CE	6	14	0	8	28
D	3	1	8	0	12
	16	22	28	12	

10. The branch lengths connecting the new node to the two joined leaves are given by the formulas below. What are these lengths? Show that the sum of these lengths is equal to what it should be.

$$L_{i,u} = \frac{1}{2(N-2)}[(N-2)D_{i,j} + \sum_k D_{i,k} - \sum_k D_{j,k}]$$
$$L_{j,u} = \frac{1}{2(N-2)}[(N-2)D_{i,j} + \sum_k D_{j,k} - \sum_k D_{i,k}]$$

The sum of these lengths is equal to $D_{i,j}$, exactly the distance between the two nodes that the branches will be joining.