Sequence Alignment using *Align*

Part II: Dynamic Programming Algorithms

Richard M. Salter
Oberlin College

Supported by NSF CCLI DUE 0618252

Making alignment doable

- In Part I, we learned that aligning a pair of sequences involves a 3 step process
  - Consider all possible matches between the pair when gaps are inserted in various places
  - Score each match according to a stated criterion
  - Select a match with highest score

- Why is this hard?
  - The number of possible matches becomes very large, even for sequences of modest length.
  - For example, recall that matching sequences of length 50 involves

    matches.

$$\binom{100}{50} = 100891344545564193334812497256$$

How can we make it simpler?

- Fortunately, there is a shortcut built into the structure of this problem.
- Consider the following 2 matches from our original example:

```
a)  aatctata        b)  aatctata
    -aaga-ta            -aag-ata
```

- Here they are again, each split into 2 subsequences of 4 letters. Notice that they differ only in the second half.

```
a)  aatc tata       b)  aatc tata
    -aag a-ta           -aag -ata
```

  - If we score only the last 4 letter pairs of (a) and (b), we get scores of 1 and 2 respectively. Thus, no matter how we rearrange and score the first 4 letter pairs (keeping the last 4 unchanged), the match in (b) will always beat the corresponding match in (a) (see box)
  - We can therefore eliminate from consideration all sequence patterns that end with the 4 letter subsequences shown in (a).

```
aatc tata  1   aatc tata  2
-aag a-ta      -aag -ata

aatc tata  1   aatc tata  2
a-ag a-ta      a-ag -ata

aatc tata  2   aatc tata  3
aa-g a-ta      aa-g -ata

aatc tata  2   aatc tata  3
aag- a-ta      aag- -ata
```

Optimal Substructure

- Sequence alignment is an example of a problem exhibiting optimal substructure.
- Optimal substructure means that optimal solutions of subproblems can be used to find the optimal solutions of the overall problem.
  - When optimal substructure is present, an optimal solution to some subproblem is always present in any optimal solution to a larger problem containing that subproblem.

- When optimal substructure exists, the procedure to find an optimal solution is called **divide and conquer:**
  - If the problem has no subproblems, solve it directly by brute force; otherwise
    - Break the problem into smaller subproblems
    - Solve these problems optimally, using divide and conquer recursively.
    - Use these optimal subproblem solutions to construct an optimal solution for the original problem.

- For sequence alignment, the subproblems are the alignment of subsequences of the two original sequences.
  - In our example, we saw that optimally placing gaps in a pair of subsequences will always lead to an optimal solution in a sequence pair extending that pair of subsequences.
- These subproblems exhibit a second important property, namely that they are overlapping.
  - Example: In the sequence `aatctata`, the subsequence `tat` is also a subsequence of `tata`, `ctat`, `ctata`, etc.
  - That is, the same subproblems are used to solve many different larger problems.

aatc**tat**a
tata
c**tat**
c**tat**a
tc**tat**a
. . .

Dynamic Programming

- Whenever optimal substructure and overlapping subsequences exist in an optimization problem, the technique known as dynamic programming can be applied.
  - Note: The word "programming" in "dynamic programming" has no particular connection to computer programming at all, and instead comes from the term "mathematical programming", a synonym for optimization.
- Dynamic programming takes much less effort than naive brute force methods.
- Dynamic programming can be implemented in several ways, but they all involve the same idea:
  - Compute the solution to each subproblem only once, even if that subproblem may be needed in several larger problems due to overlapping.
  - Record the solutions you compute so that they may be used in the computation of the larger subproblems.
  - Perform your computation systematically so that the solution to subproblems is available when constructing the solution to larger problems.

Dynamic Programming and Sequence Alignment

- For sequence alignment, dynamic programming is implemented using a matrix to hold the intermediate results, as shown below. This matrix can also be found in the main window of *Align.*
  - If the sequences being aligned have length *m* and *n* (with $m \geq n$), then the matrix has dimension $(m + 1)$ by $(n + 1)$.
  - Starting with the second row and column, the rows and columns are labeled by the letters in the sequence

		a	a	t	c	t	a	t	a
a									
a									
g									
a									
t									
a									

Dynamic Programming Procedure

- The object of the procedure is to fill in the matrix.
  We start by initializing the top row and first column, then proceed left to right and top to bottom, filling in successive squares.

- The squares first row and left-most column of the matrix are initialized in as follows:
  - Place 0.0 in the upper left hand corner.
  - Along the top row, from left to right, add the gap penalty to each successive square and write the result in the next square.
    - Recall that in our example we use a gap penalty of -1
    - The reason for doing this will become apparent later.
  - Do the same thing down the first column, going from top to bottom.

		a	a	t	c	t	a	t	a
	0.0	-1.0	-2.0	-3.0	-4.0	-5.0	-6.0	-7.0	-8.0
a	-1.0								
a	-2.0								
g	-3.0								
a	-4.0								
t	-5.0								
a	-6.0								

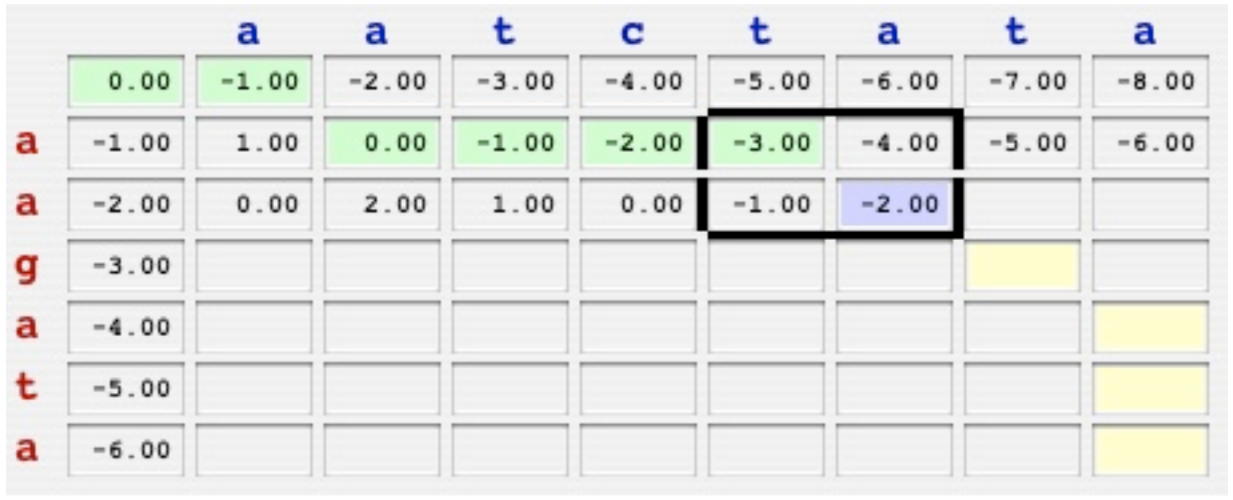
Dynamic Programming Procedure using *Align*

- We are now ready for the main part of the procedure. Let's use *Align* to help us figure it out.

- Launch *Align* as before and click the **Start** button. Notice that the top row and leftmost column are filled as described above.

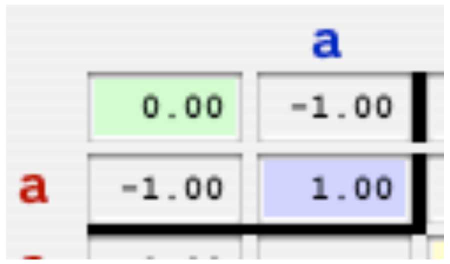
- Now click the **Auto** button in the Control panel and notice that buttons labeled **Step** and **Solve** appear.

- Click the Step button several times. Note the following:
  - The cells of the matrix are filled in running from left to right and top to bottom.
  - There is a box outlining the three cells immediately above and to the left of the current cell (i.e. the one being filled in).
  - There is a green path through the matrix from the current cell being filled in back to the leftmost-uppermost cell.

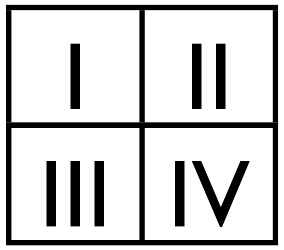


		a	a	t	c	t	a	t	a
	0.00	-1.00	-2.00	-3.00	-4.00	-5.00	-6.00	-7.00	-8.00
a	-1.00	1.00	0.00	-1.00	-2.00	-3.00	-4.00	-5.00	-6.00
a	-2.00	0.00	2.00	1.00	0.00	-1.00	-2.00		
g	-3.00								
a	-4.00								
t	-5.00								
a	-6.00								

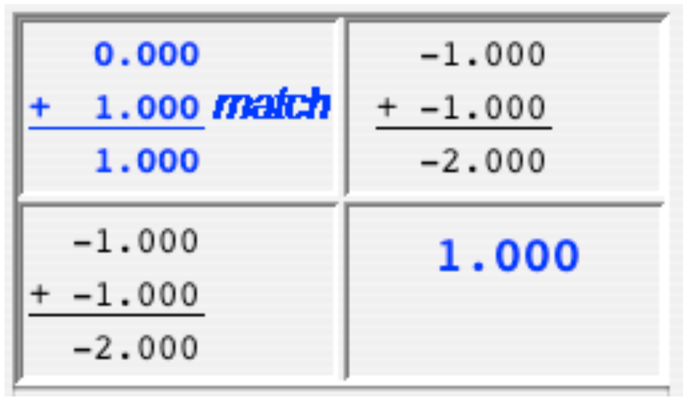
What goes in a cell?



- Let us return to the first cell to see how it was filled in.
  - Click Restart, Auto and Step.
- Let's refer to the four cells within the black box as Cells I, II, III and IV, as shown.
- To determine the value to enter into Cell IV, we first have to compute 3 numbers. Call them x, y, and z. The first two are easy to compute.
  - Let *y* be the value in Cell II plus the gap penalty.
  - Similarly, let *z* be the value in Cell III plus the gap penalty.



- To compute *x*, consider the letters labeling the row and column of Cell IV. If they match, add the match score to the value in Cell I. Otherwise, if they don't match, add the mismatch score to the value in Cell I.
  - In this case they match, so the *z* is the match score plus the value in Cell I
- Finally, select the largest of *x*, *y* and *z*, and enter it into Cell IV.
- Notice that the selection process is shown in the scoring panel in the upper-right section of the Align window.
- Now, click Step several times and observe the scoring as the cells are filled in.

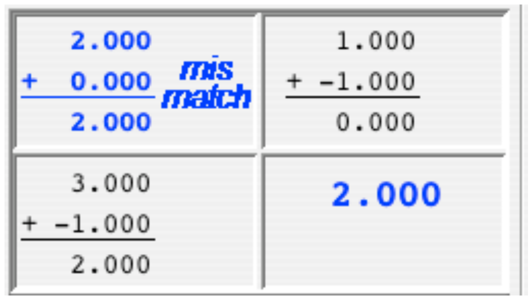


0.000 + 1.000 match 1.000	-1.000 + -1.000 -2.000
-1.000 + -1.000 -2.000	1.000

Exercise 2.1

- Fill in Cell IV given the values for Cells I-III and element pair being matched, as shown. Be sure you can justify your choice
  - Example:

$$\begin{array}{c|c|c} & & a \\ \hline & 2.00 & 1.00 \\ \hline c & 3.00 & 2.00 \end{array}$$



2.000 + 0.000 2.000 mis match	1.000 + -1.000 0.000
3.000 + -1.000 2.000	2.000

a)

$$\begin{array}{c|c|c} & & t \\ \hline & 2.00 & 3.00 \\ \hline t & 1.00 & \end{array}$$

b)

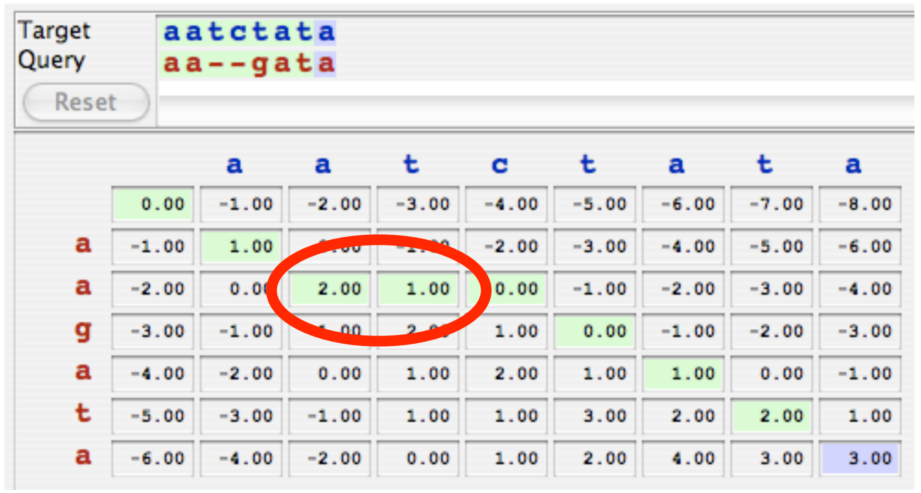
$$\begin{array}{c|c|c} & & t \\ \hline & -3.00 & -4.00 \\ \hline t & -1.00 & \end{array}$$

c)

$$\begin{array}{c|c|c} & & t \\ \hline & 0.00 & 2.00 \\ \hline g & -1.00 & \end{array}$$

Exercise 2.1 Solution

What does it all mean?



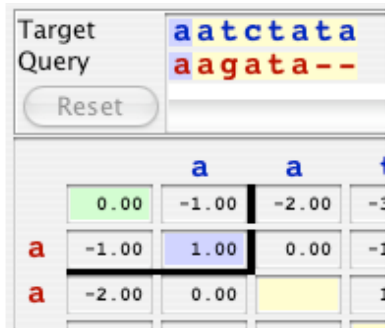
Target aatctata
Query aa--gata
Reset

		a	a	t	c	t	a	t	a
	0.00	-1.00	-2.00	-3.00	-4.00	-5.00	-6.00	-7.00	-8.00
a	-1.00	1.00	[illegible]	[illegible]	-2.00	-3.00	-4.00	-5.00	-6.00
a	-2.00	0.00	2.00	1.00	0.00	-1.00	-2.00	-3.00	-4.00
g	-3.00	-1.00	[illegible]	[illegible]	1.00	0.00	-1.00	-2.00	-3.00
a	-4.00	-2.00	0.00	1.00	2.00	1.00	1.00	0.00	-1.00
t	-5.00	-3.00	-1.00	1.00	1.00	3.00	2.00	2.00	1.00
a	-6.00	-4.00	-2.00	0.00	1.00	2.00	4.00	3.00	3.00

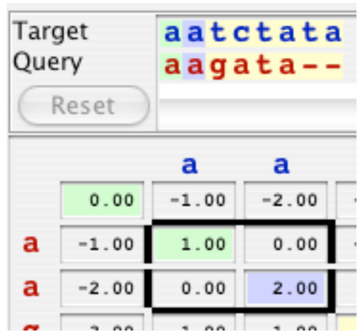
- The position of each cell determines a partial alignment between initial segments of the sequence pair.
  - Note that we refer to the top sequence as the target, and the bottom as the query.
- In *Align*, that initial segment (i.e. subsequence) is determined by the green path leading to the current cell. The green path is based on which of the three values (*x*, *y*, or *z*) was selected when the value contained in a cell was computed.
- The correspondence between path and subsequence is as follows:
  - Each horizontal path step introduces a gap in the query sequence.
    - In the example, you see this in row 3, columns 3 and 4, corresponding to the 2 successive gaps appearing between the second `a` and the first `g`.
  - Similarly, each vertical path step introduces a gap in the target sequence. None are shown in the example, but try clicking on the boxes in a filled matrix, and notice the green path and corresponding sequences.
  - A diagonal path step represents a match or mismatch between corresponding sequence letters. In the example this is evident everywhere except row 3, columns 3 and 4.

What is the green path?

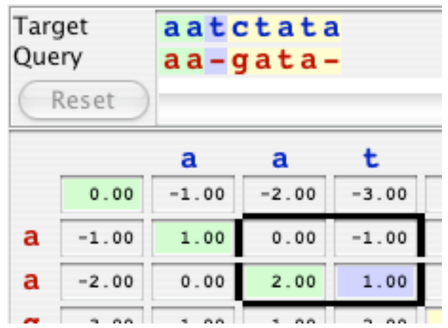
- The green path represents the pattern for placing gaps so that the subsequences have optimal scores.
  - We now realize that the potential values for a cell (*x*, *y* or *z*) respectively represent a choice of matching or mismatching the corresponding sequence letters, or introducing gaps into the query or target sequence.
  - Recall that in choosing the value for a cell, we always chose the largest of the three possibilities. Thus, we always made the optimal choice for that situation.
- Following the green path is a pair of elements in blue, and a yellow path to the end of the sequences. Note that these colors highlight both the matrix and the corresponding positions in the aligned sequences.
  - The yellow path is the part yet to be scored.
  - The blue pair is the value chosen for the current Cell IV. Here we are making our choice for optimally extending the initial segments.
  - We know our selection will be optimal because we are optimally extending an optimal value previously recorded in Cells I - III.


Target aatctata
Query aagata--

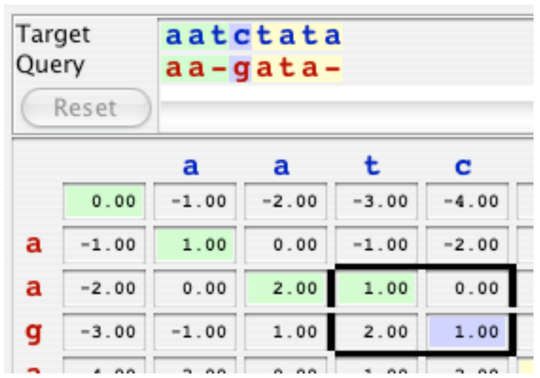
		a	a	t
	0.00	-1.00	-2.00	-3
a	-1.00	1.00	0.00	-1
a	-2.00	0.00		1


Target aatctata
Query aagata--

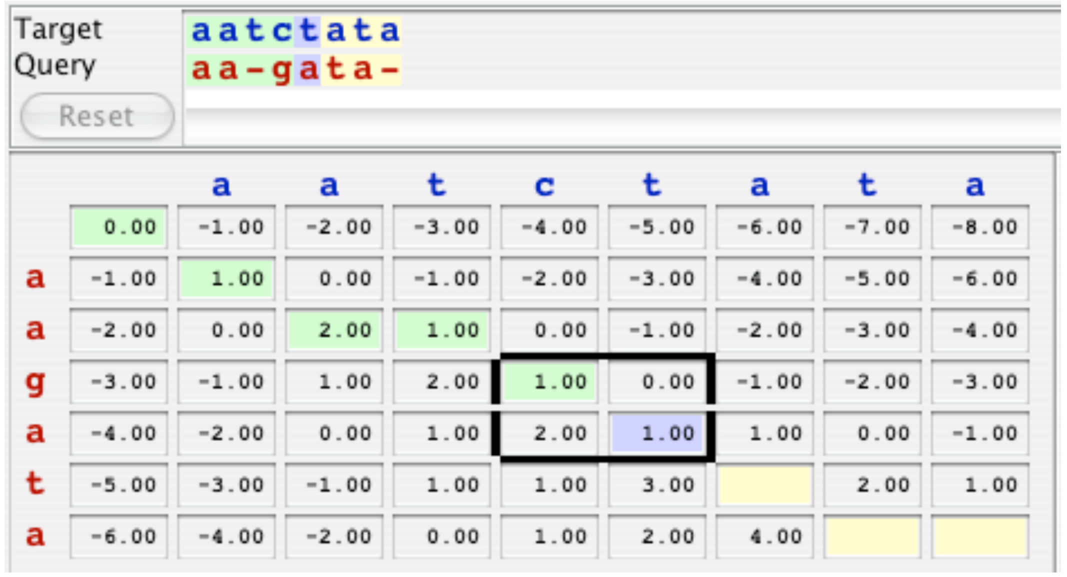
		a	a
	0.00	-1.00	-2.00
a	-1.00	1.00	0.00
a	-2.00	0.00	2.00


Target aatctata
Query aa-gata-

		a	a	t
	0.00	-1.00	-2.00	-3.00
a	-1.00	1.00	0.00	-1.00
a	-2.00	0.00	2.00	1.00


Target aatctata
Query aa-gata-

		a	a	t	c
	0.00	-1.00	-2.00	-3.00	-4.00
a	-1.00	1.00	0.00	-1.00	-2.00
a	-2.00	0.00	2.00	1.00	0.00
g	-3.00	-1.00	1.00	2.00	1.00

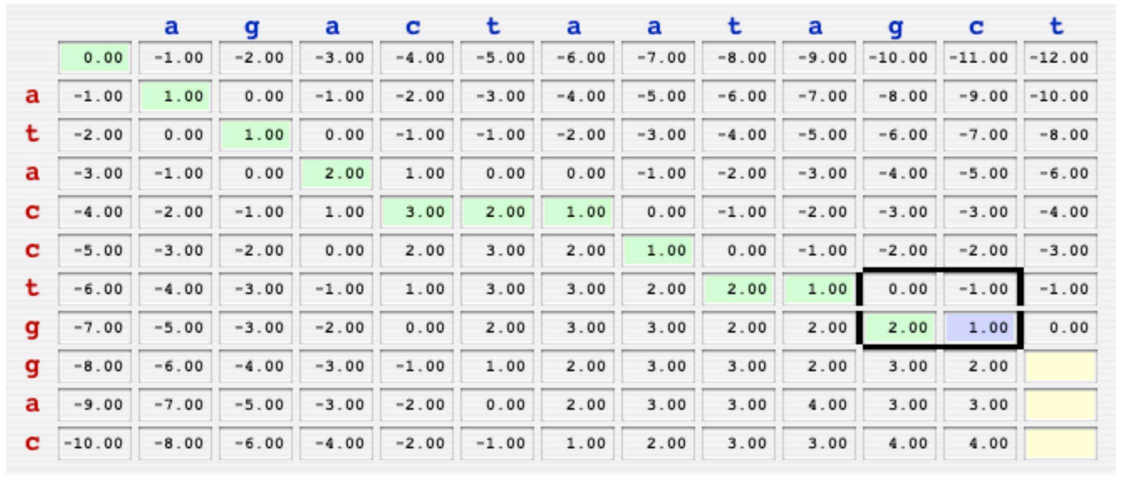

Target aatctata
Query aa-gata-

		a	a	t	c	t	a	t	a
	0.00	-1.00	-2.00	-3.00	-4.00	-5.00	-6.00	-7.00	-8.00
a	-1.00	1.00	0.00	-1.00	-2.00	-3.00	-4.00	-5.00	-6.00
a	-2.00	0.00	2.00	1.00	0.00	-1.00	-2.00	-3.00	-4.00
g	-3.00	-1.00	1.00	2.00	1.00	0.00	-1.00	-2.00	-3.00
a	-4.00	-2.00	0.00	1.00	2.00	1.00	1.00	0.00	-1.00
t	-5.00	-3.00	-1.00	1.00	1.00	3.00		2.00	1.00
a	-6.00	-4.00	-2.00	0.00	1.00	2.00	4.00		

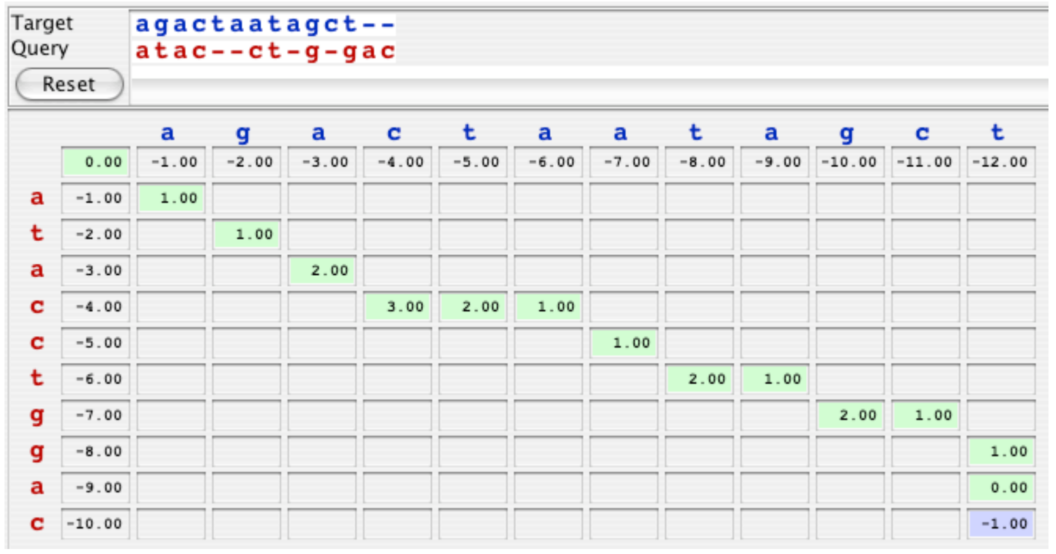
Exercise 2.2

- Load the following sequence pair into *Align* : agactaatagct / atacctggac
  - Recall that to load a sequence pair select **File | Load Sequences ...** and type the sequences into the Load Sequences dialog window.
- On the next page you will find several matrices corresponding to several possible alignments of the target and query sequences. Each possible alignment generates different values in the matrix and a different green pathway. For each matrix shown, determine how the target-query sequences were aligned to generate the green pathway shown.
- Once you have determined the alignment, verify it using ALIGN. To do this, enter gaps in the target or query sequence as needed. Note that ALIGN must be in Manual mode in order to introduce gaps into the target or query sequences. Then click *Evaluate* to check if your green/blue path is the same as the **Auto** mode green/blue/yellow path.
  - You can flip back and forth between Manual and Auto mode to compare results.
- Example (**Auto** mode)

Solution (**Manual** mode)



		a	g	a	c	t	a	a	t	a	g	c	t
	0.00	-1.00	-2.00	-3.00	-4.00	-5.00	-6.00	-7.00	-8.00	-9.00	-10.00	-11.00	-12.00
a	-1.00	1.00	0.00	-1.00	-2.00	-3.00	-4.00	-5.00	-6.00	-7.00	-8.00	-9.00	-10.00
t	-2.00	0.00	1.00	0.00	-1.00	-1.00	-2.00	-3.00	-4.00	-5.00	-6.00	-7.00	-8.00
a	-3.00	-1.00	0.00	2.00	1.00	0.00	0.00	-1.00	-2.00	-3.00	-4.00	-5.00	-6.00
c	-4.00	-2.00	-1.00	1.00	3.00	2.00	1.00	0.00	-1.00	-2.00	-3.00	-3.00	-4.00
c	-5.00	-3.00	-2.00	0.00	2.00	3.00	2.00	1.00	0.00	-1.00	-2.00	-2.00	-3.00
t	-6.00	-4.00	-3.00	-1.00	1.00	3.00	3.00	2.00	2.00	1.00	0.00	-1.00	-1.00
g	-7.00	-5.00	-3.00	-2.00	0.00	2.00	3.00	3.00	2.00	2.00	2.00	1.00	0.00
g	-8.00	-6.00	-4.00	-3.00	-1.00	1.00	2.00	3.00	3.00	2.00	3.00	2.00	
a	-9.00	-7.00	-5.00	-3.00	-2.00	0.00	2.00	3.00	3.00	4.00	3.00	3.00	
c	-10.00	-8.00	-6.00	-4.00	-2.00	-1.00	1.00	2.00	3.00	3.00	4.00	4.00	

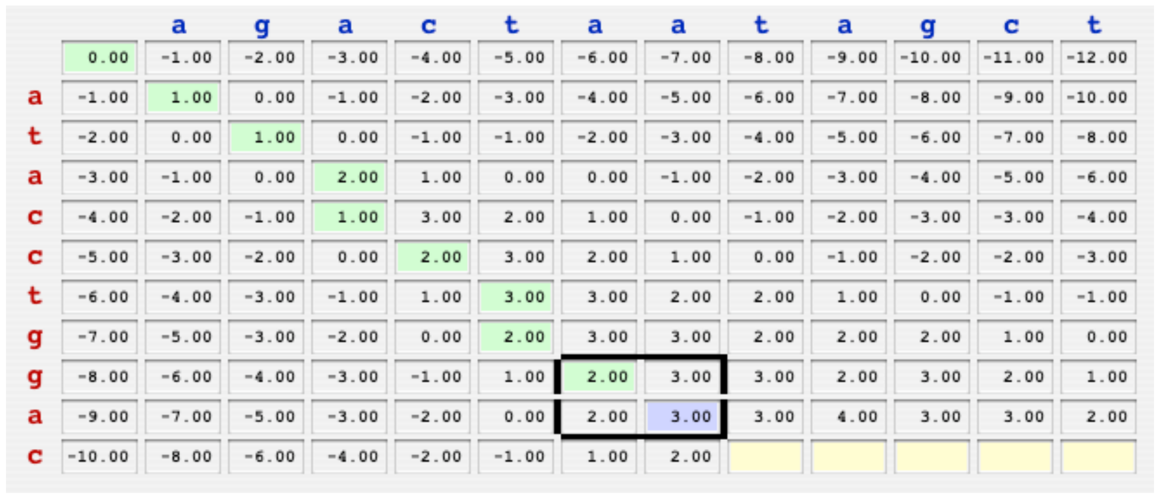


Target agactaatagct--
Query atac--ct-g-gac
Reset

		a	g	a	c	t	a	a	t	a	g	c	t
	0.00	-1.00	-2.00	-3.00	-4.00	-5.00	-6.00	-7.00	-8.00	-9.00	-10.00	-11.00	-12.00
a	-1.00	1.00											
t	-2.00		1.00										
a	-3.00			2.00									
c	-4.00				3.00	2.00	1.00						
c	-5.00							1.00					
t	-6.00								2.00	1.00			
g	-7.00										2.00	1.00	
g	-8.00												1.00
a	-9.00												0.00
c	-10.00												-1.00

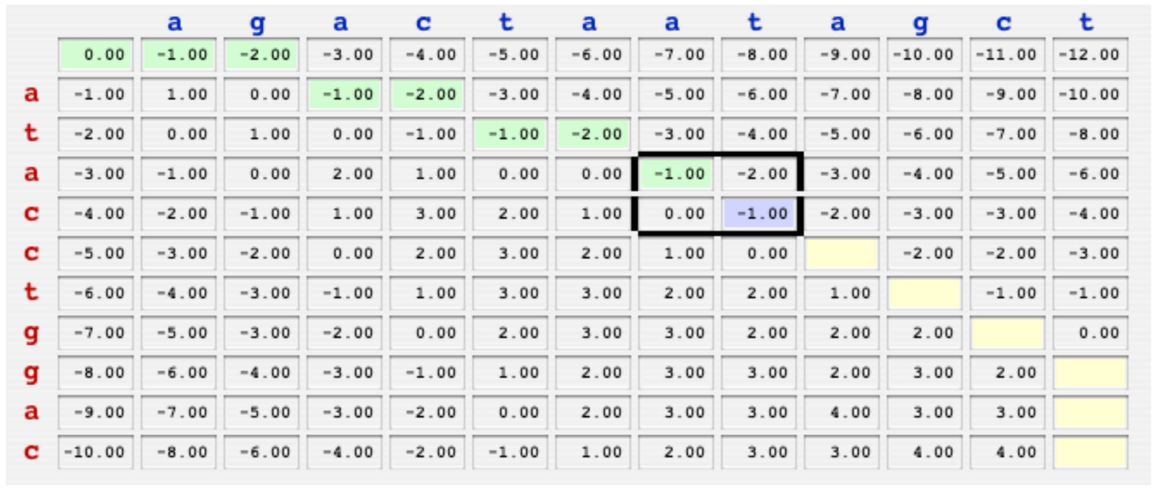
Exercise 2.2 (continued)

a)



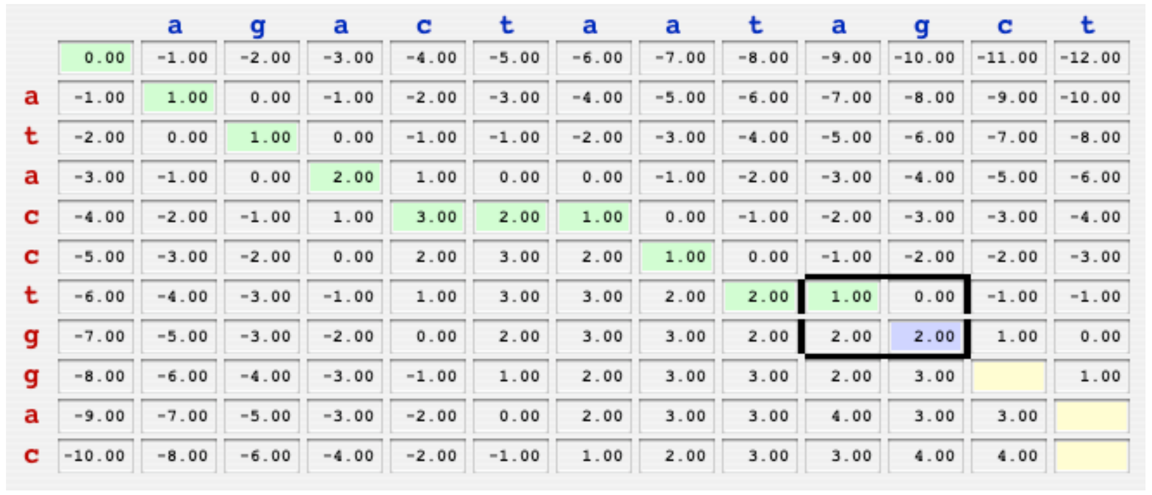
		a	g	a	c	t	a	a	t	a	g	c	t
	0.00	-1.00	-2.00	-3.00	-4.00	-5.00	-6.00	-7.00	-8.00	-9.00	-10.00	-11.00	-12.00
a	-1.00	1.00	0.00	-1.00	-2.00	-3.00	-4.00	-5.00	-6.00	-7.00	-8.00	-9.00	-10.00
t	-2.00	0.00	1.00	0.00	-1.00	-1.00	-2.00	-3.00	-4.00	-5.00	-6.00	-7.00	-8.00
a	-3.00	-1.00	0.00	2.00	1.00	0.00	0.00	-1.00	-2.00	-3.00	-4.00	-5.00	-6.00
c	-4.00	-2.00	-1.00	1.00	3.00	2.00	1.00	0.00	-1.00	-2.00	-3.00	-3.00	-4.00
c	-5.00	-3.00	-2.00	0.00	2.00	3.00	2.00	1.00	0.00	-1.00	-2.00	-2.00	-3.00
t	-6.00	-4.00	-3.00	-1.00	1.00	3.00	3.00	2.00	2.00	1.00	0.00	-1.00	-1.00
g	-7.00	-5.00	-3.00	-2.00	0.00	2.00	3.00	3.00	2.00	2.00	2.00	1.00	0.00
g	-8.00	-6.00	-4.00	-3.00	-1.00	1.00	2.00	3.00	3.00	2.00	3.00	2.00	1.00
a	-9.00	-7.00	-5.00	-3.00	-2.00	0.00	2.00	3.00	3.00	4.00	3.00	3.00	2.00
c	-10.00	-8.00	-6.00	-4.00	-2.00	-1.00	1.00	2.00					

b)



		a	g	a	c	t	a	a	t	a	g	c	t
	0.00	-1.00	-2.00	-3.00	-4.00	-5.00	-6.00	-7.00	-8.00	-9.00	-10.00	-11.00	-12.00
a	-1.00	1.00	0.00	-1.00	-2.00	-3.00	-4.00	-5.00	-6.00	-7.00	-8.00	-9.00	-10.00
t	-2.00	0.00	1.00	0.00	-1.00	-1.00	-2.00	-3.00	-4.00	-5.00	-6.00	-7.00	-8.00
a	-3.00	-1.00	0.00	2.00	1.00	0.00	0.00	-1.00	-2.00	-3.00	-4.00	-5.00	-6.00
c	-4.00	-2.00	-1.00	1.00	3.00	2.00	1.00	0.00	-1.00	-2.00	-3.00	-3.00	-4.00
c	-5.00	-3.00	-2.00	0.00	2.00	3.00	2.00	1.00	0.00		-2.00	-2.00	-3.00
t	-6.00	-4.00	-3.00	-1.00	1.00	3.00	3.00	2.00	2.00	1.00		-1.00	-1.00
g	-7.00	-5.00	-3.00	-2.00	0.00	2.00	3.00	3.00	2.00	2.00	2.00		0.00
g	-8.00	-6.00	-4.00	-3.00	-1.00	1.00	2.00	3.00	3.00	2.00	3.00	2.00	
a	-9.00	-7.00	-5.00	-3.00	-2.00	0.00	2.00	3.00	3.00	4.00	3.00	3.00	
c	-10.00	-8.00	-6.00	-4.00	-2.00	-1.00	1.00	2.00	3.00	3.00	4.00	4.00	

c)

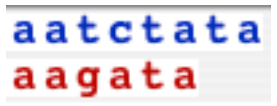


		a	g	a	c	t	a	a	t	a	g	c	t
	0.00	-1.00	-2.00	-3.00	-4.00	-5.00	-6.00	-7.00	-8.00	-9.00	-10.00	-11.00	-12.00
a	-1.00	1.00	0.00	-1.00	-2.00	-3.00	-4.00	-5.00	-6.00	-7.00	-8.00	-9.00	-10.00
t	-2.00	0.00	1.00	0.00	-1.00	-1.00	-2.00	-3.00	-4.00	-5.00	-6.00	-7.00	-8.00
a	-3.00	-1.00	0.00	2.00	1.00	0.00	0.00	-1.00	-2.00	-3.00	-4.00	-5.00	-6.00
c	-4.00	-2.00	-1.00	1.00	3.00	2.00	1.00	0.00	-1.00	-2.00	-3.00	-3.00	-4.00
c	-5.00	-3.00	-2.00	0.00	2.00	3.00	2.00	1.00	0.00	-1.00	-2.00	-2.00	-3.00
t	-6.00	-4.00	-3.00	-1.00	1.00	3.00	3.00	2.00	2.00	1.00	0.00	-1.00	-1.00
g	-7.00	-5.00	-3.00	-2.00	0.00	2.00	3.00	3.00	2.00	2.00	2.00	1.00	0.00
g	-8.00	-6.00	-4.00	-3.00	-1.00	1.00	2.00	3.00	3.00	2.00	3.00		1.00
a	-9.00	-7.00	-5.00	-3.00	-2.00	0.00	2.00	3.00	3.00	4.00	3.00	3.00	
c	-10.00	-8.00	-6.00	-4.00	-2.00	-1.00	1.00	2.00	3.00	3.00	4.00	4.00	

Exercise 2.2 Solution

Exercise 2.3

- If *Align* contains the sequences from Exercise 2.2, reload the original sequence pair:



```
aatctata
aagata
```

- Click **Restart**, then **Auto**.
- Starting with the upper left corner, determine by hand the value that belongs in Cell IV.
- Then click **Step** and check your answer. The scoring panel will help justify the right answer. Also note the green path and the sequence alignment corresponding to the cell being filled in.
- Do this several times in row 1, then click **Step** to advance to later rows and try again. Carefully observe the green path and the sequences in later rows.
- When you get tired, click Solve, and *Align* will complete the job.
- Notice the final green path and sequence alignment. It is a solution for the optimal match.
  - Are there other optimal solutions?
  - If so, why didn't *Align* find one of those?
- Once the entire matrix is filled in, you can click on different cells to see the computation for those cells.
- The final result gives 3 as the optimal score, yet there is a score of 4.0 in a cell on the bottom row. How do you explain this?

Exercise 2.3 Solution

Exercise 2.4

- For each of the following sequence pairs:
  - Load the pair into *Align*. To load the sequences, click **File | Load Sequences ...**, and type the sequences into the "Load Sequences" dialog window.
  - Click **Start**, then **Manual**. Try to find an optimal alignment by inserting gaps into the sequences, as in Exercises 1.1 and 1.2.
    - Recall that right-click inserts a gap, and left-click deletes a gap.
  - Now click **Auto**, and either by using **Step** or **Solve**, let Align find an optimal solution.
  - Now click **Manual** again. *Align* remembered your solution. You can compare your solution with the one found by *Align* by going back and forth between **Manual** and **Auto**. How did you do?

a)
```
tgtacggctata
tccgcctta
```

b)
```
tctgtacgcgatcatgt
tagcgttcgatata
```

c)
```
agatagaaactgatatata
accgtccg
```

Exercise 2.4 Solution

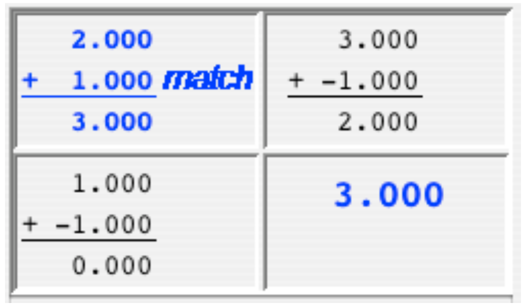
◁

End of Part II

Solution to Exercise 2.1

a)

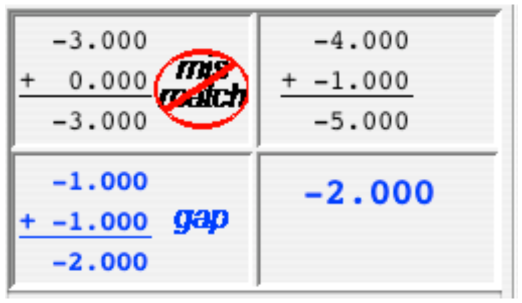
		t
	2.00	3.00
t	1.00	**3.00**



2.000 + 1.000 = 3.000 *match*	3.000 + -1.000 = 2.000
1.000 + -1.000 = 0.000	**3.000**

b)

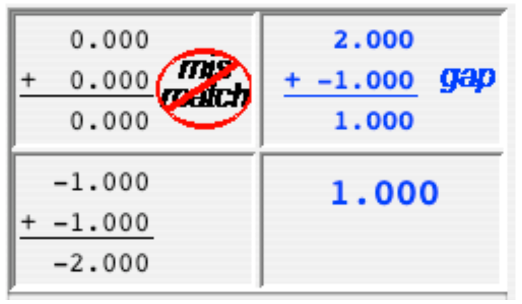
		t
	-3.00	-4.00
t	-1.00	**-2.00**



-3.000 + 0.000 = -3.000 *mis match*	-4.000 + -1.000 = -5.000
-1.000 + -1.000 = -2.000 *gap*	**-2.000**

c)

		t
	0.00	2.00
g	-1.00	**1.00**

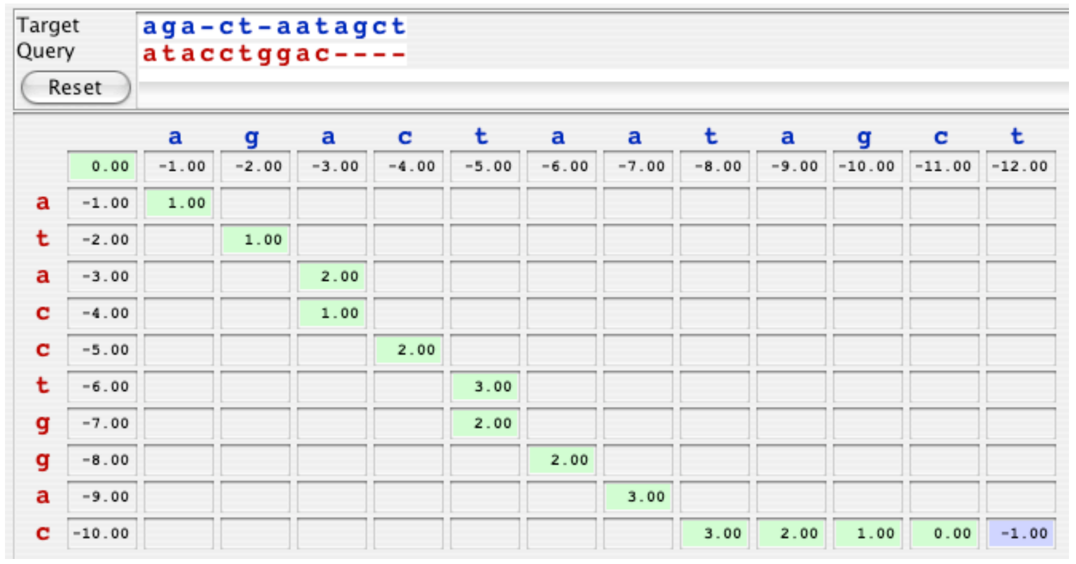


0.000 + 0.000 = 0.000 *mis match*	2.000 + -1.000 = 1.000 *gap*
-1.000 + -1.000 = -2.000	**1.000**

Return to Exercise 2.1

Solution to Exercise 2.2

a)

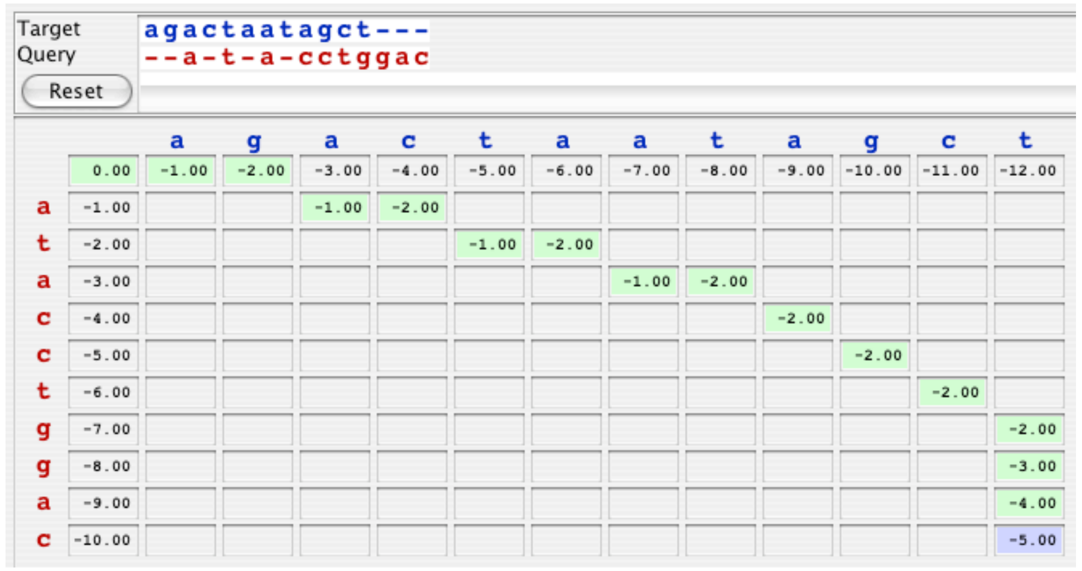


Target aga-ct-aatagct
Query atacctggac----

Reset

		a	g	a	c	t	a	a	t	a	g	c	t
	0.00	-1.00	-2.00	-3.00	-4.00	-5.00	-6.00	-7.00	-8.00	-9.00	-10.00	-11.00	-12.00
a	-1.00	1.00											
t	-2.00		1.00										
a	-3.00			2.00									
c	-4.00			1.00									
c	-5.00				2.00								
t	-6.00					3.00							
g	-7.00					2.00							
g	-8.00						2.00						
a	-9.00							3.00					
c	-10.00								3.00	2.00	1.00	0.00	-1.00

b)

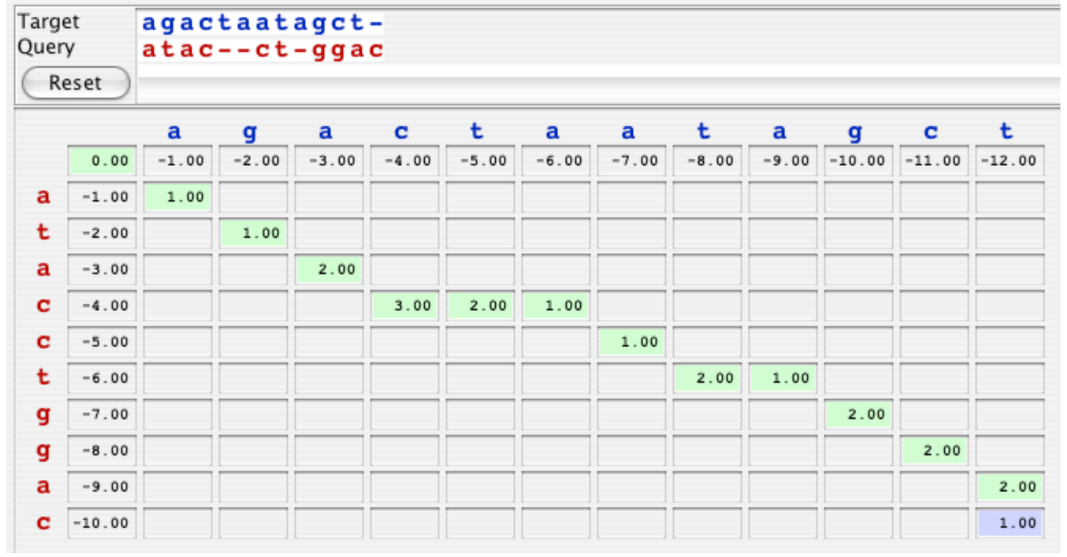


Target agactaatagct---
Query --a-t-a-cctggac

Reset

		a	g	a	c	t	a	a	t	a	g	c	t
	0.00	-1.00	-2.00	-3.00	-4.00	-5.00	-6.00	-7.00	-8.00	-9.00	-10.00	-11.00	-12.00
a	-1.00			-1.00	-2.00								
t	-2.00					-1.00	-2.00						
a	-3.00							-1.00	-2.00				
c	-4.00									-2.00			
c	-5.00										-2.00		
t	-6.00											-2.00	
g	-7.00												-2.00
g	-8.00												-3.00
a	-9.00												-4.00
c	-10.00												-5.00

c)



Target agactaatagct-
Query atac--ct-ggac

Reset

		a	g	a	c	t	a	a	t	a	g	c	t
	0.00	-1.00	-2.00	-3.00	-4.00	-5.00	-6.00	-7.00	-8.00	-9.00	-10.00	-11.00	-12.00
a	-1.00	1.00											
t	-2.00		1.00										
a	-3.00			2.00									
c	-4.00				3.00	2.00	1.00						
c	-5.00							1.00					
t	-6.00								2.00	1.00			
g	-7.00										2.00		
g	-8.00											2.00	
a	-9.00												2.00
c	-10.00												1.00

Return to Exercise 2.2

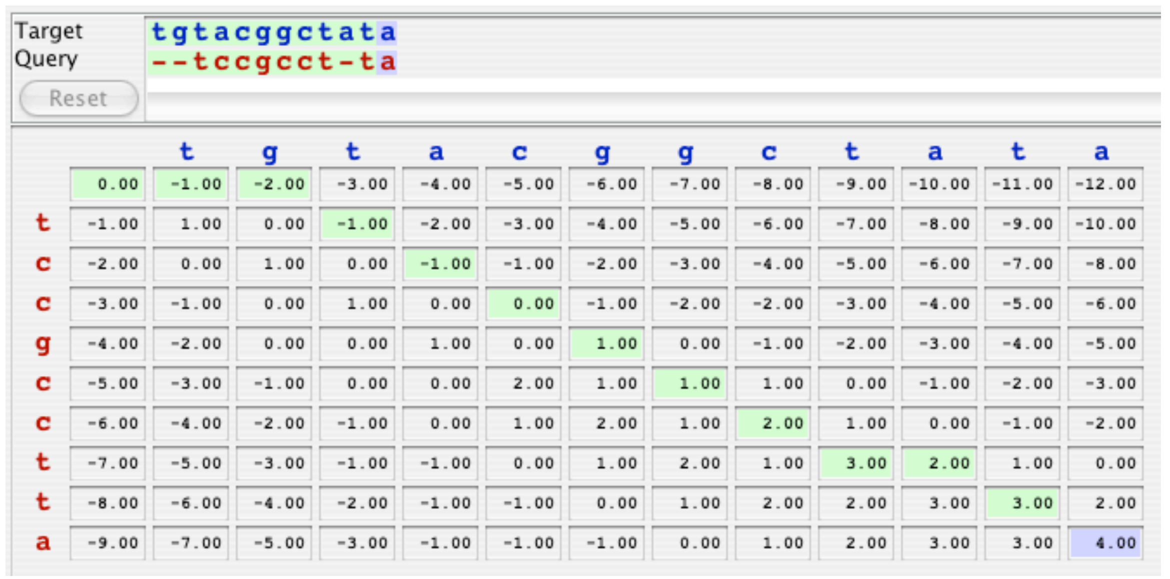
Solution to Exercise 2.3

- There are at least three optimal solutions. See the solution to Exercise 1.2.
- *Align* will always choose the optimal solution in which the match for each initial segment is also optimal. There can only be one of those.
- The 4.0 is scoring an incomplete problem. If you click on it you will find it is followed by 2 cells that have yet to be scored.
  - The yellow part of the sequence is the part yet to be evaluated, and so the final score may be less than the score for the initial segment that terminates at a particular cell.

Return to Exercise 2.3

Solution to Exercise 2.4

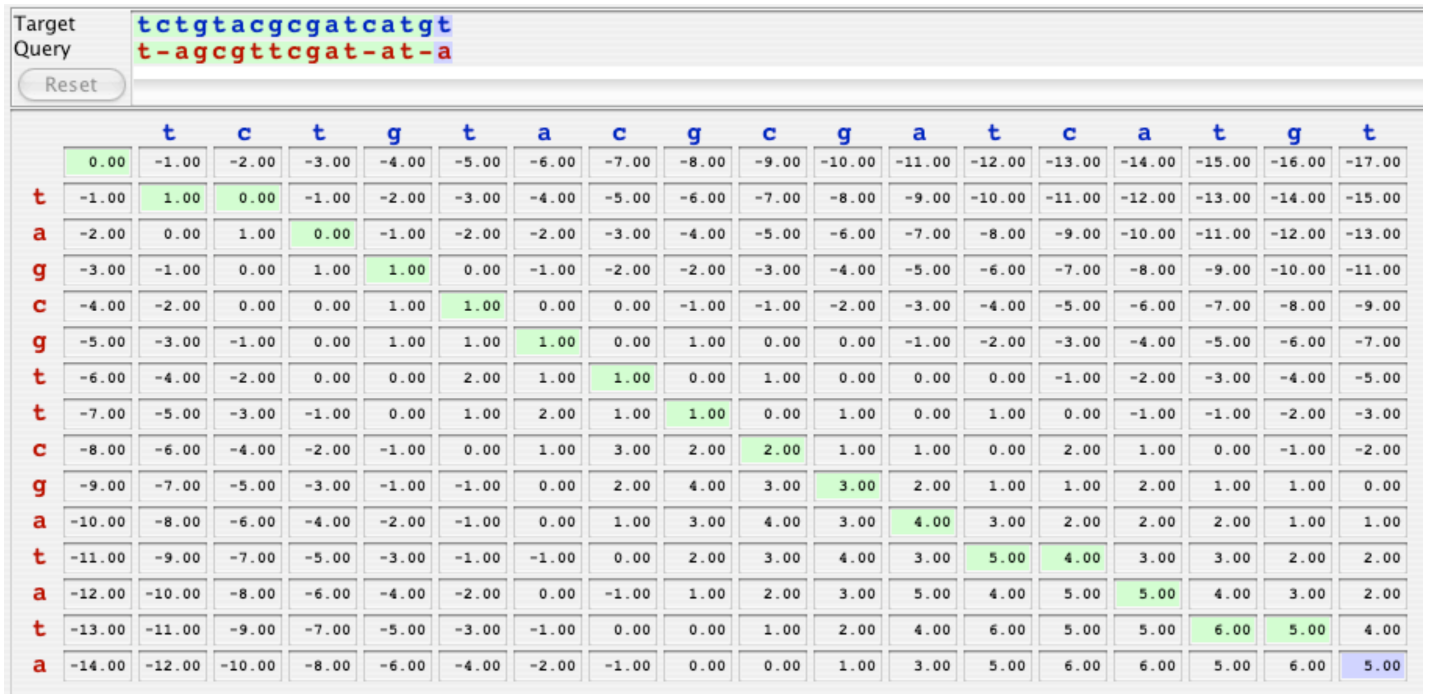
a)



Target `tgtacggctata`
Query `--tccgcct-ta`

		t	g	t	a	c	g	g	c	t	a	t	a
	0.00	-1.00	-2.00	-3.00	-4.00	-5.00	-6.00	-7.00	-8.00	-9.00	-10.00	-11.00	-12.00
t	-1.00	1.00	0.00	-1.00	-2.00	-3.00	-4.00	-5.00	-6.00	-7.00	-8.00	-9.00	-10.00
c	-2.00	0.00	1.00	0.00	-1.00	-1.00	-2.00	-3.00	-4.00	-5.00	-6.00	-7.00	-8.00
c	-3.00	-1.00	0.00	1.00	0.00	0.00	-1.00	-2.00	-2.00	-3.00	-4.00	-5.00	-6.00
g	-4.00	-2.00	0.00	0.00	1.00	0.00	1.00	0.00	-1.00	-2.00	-3.00	-4.00	-5.00
c	-5.00	-3.00	-1.00	0.00	0.00	2.00	1.00	1.00	1.00	0.00	-1.00	-2.00	-3.00
c	-6.00	-4.00	-2.00	-1.00	0.00	1.00	2.00	1.00	2.00	1.00	0.00	-1.00	-2.00
t	-7.00	-5.00	-3.00	-1.00	-1.00	0.00	1.00	2.00	1.00	3.00	2.00	1.00	0.00
t	-8.00	-6.00	-4.00	-2.00	-1.00	-1.00	0.00	1.00	2.00	2.00	3.00	3.00	2.00
a	-9.00	-7.00	-5.00	-3.00	-1.00	-1.00	-1.00	0.00	1.00	2.00	3.00	3.00	4.00

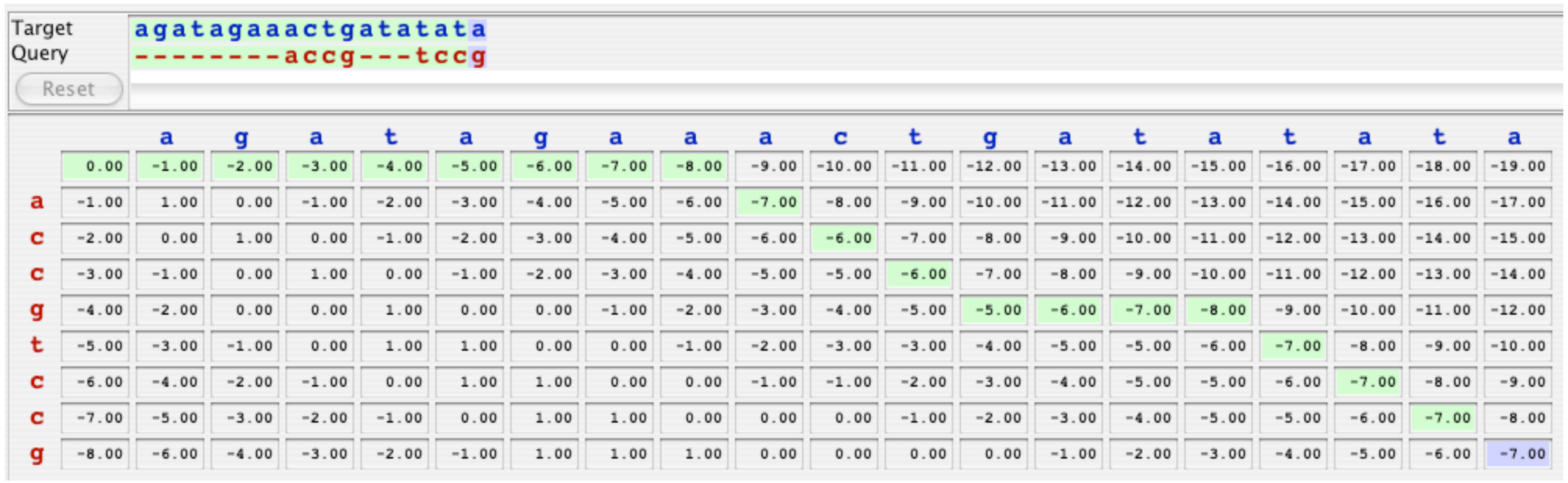
b)



Target `tctgtacgcgatcatgt`
Query `t-agcgttcgat-at-a`

		t	c	t	g	t	a	c	g	c	g	a	t	c	a	t	g	t
	0.00	-1.00	-2.00	-3.00	-4.00	-5.00	-6.00	-7.00	-8.00	-9.00	-10.00	-11.00	-12.00	-13.00	-14.00	-15.00	-16.00	-17.00
t	-1.00	1.00	0.00	-1.00	-2.00	-3.00	-4.00	-5.00	-6.00	-7.00	-8.00	-9.00	-10.00	-11.00	-12.00	-13.00	-14.00	-15.00
a	-2.00	0.00	1.00	0.00	-1.00	-2.00	-2.00	-3.00	-4.00	-5.00	-6.00	-7.00	-8.00	-9.00	-10.00	-11.00	-12.00	-13.00
g	-3.00	-1.00	0.00	1.00	1.00	0.00	-1.00	-2.00	-2.00	-3.00	-4.00	-5.00	-6.00	-7.00	-8.00	-9.00	-10.00	-11.00
c	-4.00	-2.00	0.00	0.00	1.00	1.00	0.00	0.00	-1.00	-1.00	-2.00	-3.00	-4.00	-5.00	-6.00	-7.00	-8.00	-9.00
g	-5.00	-3.00	-1.00	0.00	1.00	1.00	1.00	0.00	1.00	0.00	0.00	-1.00	-2.00	-3.00	-4.00	-5.00	-6.00	-7.00
t	-6.00	-4.00	-2.00	0.00	0.00	2.00	1.00	1.00	0.00	1.00	0.00	0.00	0.00	-1.00	-2.00	-3.00	-4.00	-5.00
t	-7.00	-5.00	-3.00	-1.00	0.00	1.00	2.00	1.00	1.00	0.00	1.00	0.00	1.00	0.00	-1.00	-1.00	-2.00	-3.00
c	-8.00	-6.00	-4.00	-2.00	-1.00	0.00	1.00	3.00	2.00	2.00	1.00	1.00	0.00	2.00	1.00	0.00	-1.00	-2.00
g	-9.00	-7.00	-5.00	-3.00	-1.00	-1.00	0.00	2.00	4.00	3.00	3.00	2.00	1.00	1.00	2.00	1.00	1.00	0.00
a	-10.00	-8.00	-6.00	-4.00	-2.00	-1.00	0.00	1.00	3.00	4.00	3.00	4.00	3.00	2.00	2.00	2.00	1.00	1.00
t	-11.00	-9.00	-7.00	-5.00	-3.00	-1.00	-1.00	0.00	2.00	3.00	4.00	3.00	5.00	4.00	3.00	3.00	2.00	2.00
a	-12.00	-10.00	-8.00	-6.00	-4.00	-2.00	0.00	-1.00	1.00	2.00	3.00	5.00	4.00	5.00	5.00	4.00	3.00	2.00
t	-13.00	-11.00	-9.00	-7.00	-5.00	-3.00	-1.00	0.00	0.00	1.00	2.00	4.00	6.00	5.00	5.00	6.00	5.00	4.00
a	-14.00	-12.00	-10.00	-8.00	-6.00	-4.00	-2.00	-1.00	0.00	0.00	1.00	3.00	5.00	6.00	6.00	5.00	6.00	5.00

c)



Target `agatagaaactgatatata`
Query `--------accg---tccg`

		a	g	a	t	a	g	a	a	a	c	t	g	a	t	a	t	a	t	a
	0.00	-1.00	-2.00	-3.00	-4.00	-5.00	-6.00	-7.00	-8.00	-9.00	-10.00	-11.00	-12.00	-13.00	-14.00	-15.00	-16.00	-17.00	-18.00	-19.00
a	-1.00	1.00	0.00	-1.00	-2.00	-3.00	-4.00	-5.00	-6.00	-7.00	-8.00	-9.00	-10.00	-11.00	-12.00	-13.00	-14.00	-15.00	-16.00	-17.00
c	-2.00	0.00	1.00	0.00	-1.00	-2.00	-3.00	-4.00	-5.00	-6.00	-6.00	-7.00	-8.00	-9.00	-10.00	-11.00	-12.00	-13.00	-14.00	-15.00
c	-3.00	-1.00	0.00	1.00	0.00	-1.00	-2.00	-3.00	-4.00	-5.00	-5.00	-6.00	-7.00	-8.00	-9.00	-10.00	-11.00	-12.00	-13.00	-14.00
g	-4.00	-2.00	0.00	0.00	1.00	0.00	0.00	-1.00	-2.00	-3.00	-4.00	-5.00	-5.00	-6.00	-7.00	-8.00	-9.00	-10.00	-11.00	-12.00
t	-5.00	-3.00	-1.00	0.00	1.00	1.00	0.00	0.00	-1.00	-2.00	-3.00	-3.00	-4.00	-5.00	-5.00	-6.00	-7.00	-8.00	-9.00	-10.00
c	-6.00	-4.00	-2.00	-1.00	0.00	1.00	1.00	0.00	0.00	-1.00	-1.00	-2.00	-3.00	-4.00	-5.00	-5.00	-6.00	-7.00	-8.00	-9.00
c	-7.00	-5.00	-3.00	-2.00	-1.00	0.00	1.00	1.00	0.00	0.00	0.00	-1.00	-2.00	-3.00	-4.00	-5.00	-5.00	-6.00	-7.00	-8.00
g	-8.00	-6.00	-4.00	-3.00	-2.00	-1.00	1.00	1.00	1.00	0.00	0.00	0.00	0.00	-1.00	-2.00	-3.00	-4.00	-5.00	-6.00	-7.00

Return to Exercise 2.4