String Alignment Example

Suppose we have two strings x and y

$x =$ `LITTLE SPECK`
$y =$ `MISTER SPOCK`

Hypothetically, you could have three different alignments, as described in **1.2** of *Lecture 7*:

Alignment 1: (substitute)
```
LITTLE SPECK
s|s|ss|||s||
MISTER SPOCK
```

Alignment 2: (insert & delete; *indel*)
```
L-IT-T-LE SPE-CK
di|di|i|d|||di||
-MI-STER- SP-OCK
```

Alignment 3: (delete all & replace)
```
LITTLE SPECK------------
ddddddddddddiiiiiiiiiiii
------------MISTER SPOCK
```

Now, looking at these three alignments, we can see that, for *Alignment 1*, there are **5** operations performed. Assuming any operation (a substitute, a delete, or an insert) has a cost of **1**, this means that *Alignment 1* should have a total cost of **5** *sub* operations.

For *Alignment 2*, we see that there are **7** operations performed, which would give a total cost of **7** *indel* operations.

Conversely, *Alignment 3*, which deletes everything and then inserts the new string, requires **24** *indel* operations, giving it a total cost of **24**.

This is all fine, but what if we want to find the ideal-cost solution without trying all of these operations by hand???

Finding the Ideal Cost Solution

You can find the ideal cost solution for aligning two strings by generating a cost matrix.

In this matrix, the first column and row each represent a NULL, empty string. In each of these cases, we would have to insert every character (11 letters and 1 space) into the string. In the cost matrix, we would represent this in **column 0** and **row 0** by adding a cost of **1** to each insertion operation we perform. (*i.e.*, if we're aligning the empty string to MISTER SPOCK, we will need 1 insert for M, 2 inserts for MI, 3 for MIS, and so-on.)

Now, for the rest of the matrix, it is easiest to populate it diagonally, which will represent if we were to substitute a letter in row *i* with a letter in column *j*. Beginning at position (1,1) *(comparing M and L)*, we see that the cost of the previous operation (0,0) was **0**, we will have to perform a substitution, since M $\neq$ L, which will require us to have a cost of **0 + 1 = 1**. Then, for position (2,2), we observe that I = I, which means we do not need to perform any operation, leaving us a cost of **1**. We proceed down this central diagonal until we find that the total cost at position (12,12) is **5**.

We can then fill-in the remainder of the matrix in a similar fashion, comparing a letter in any row *i* with a letter in column *j*, and adding a cost of **1** to the cost in the position to the left and above it if *i* and *j* are not equivalent.

x/y	–	M	I	S	T	E	R	_	S	P	O	C	K
–	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
L	**1**	**1**	2	3	4	5	6	7	8	9	10	11	12
I	**2**	2	**1**	3	4	5	6	7	8	9	10	11	12
T	**3**	3	3	**2**	3	5	6	7	8	9	10	11	12
T	**4**	4	4	4	**2**	4	6	7	8	9	10	11	12
L	**5**	5	5	5	5	**3**	5	7	8	9	10	11	12
E	**6**	6	6	6	6	5	**4**	6	8	9	10	11	12
_	**7**	7	7	7	7	7	6	**4**	7	9	10	11	12
S	**8**	8	8	7	8	8	8	7	**4**	8	10	11	12
P	**9**	9	9	9	8	9	9	9	8	**4**	9	11	12
E	**10**	10	10	10	10	9	10	10	10	9	**5**	10	12
C	**11**	11	11	11	11	11	10	11	11	11	10	**5**	11
K	**12**	12	12	12	12	12	12	11	12	12	12	11	**5**

We can then determine the set of operations—*subs* and *indels*—needed to generate an ideal alignment cost by beginning at the bottom-right corner of the matrix (in this case **(12,12)**), and moving backwards. Every move back will be determined by finding which of the three cells preceding the current cell (left, up, and diagonal) has the lowest value. *(In the event of a tie, an arbitrary decision must be made.)* This process continues until you reach cell (0,0), at which point you have found the ideal path. Then, you can look back and find that, moving forward, a move to the *right* or *down* constitutes an *indel* operation, while a move *diagonally* is a *sub* operation. In the case of the above problem, we find that the ideal solution is entirely *substitutions*, which gives us a total alignment cost of **5**, as we saw in the example alignments at the beginning.

In other string alignment problems, of course, there can be multiple ideal paths, or ideal paths consisting of *indel* operations as well as, or instead of, *sub* operations. An example of a cost matrix that could give multiple solutions, an also includes solutions involving *indels* can be found on Page 9 of Professor Clauset's *Lecture 7* notes.