1 Practical Bioinformatics – Day 8

1.1 Data structures for dynamic programming

This is from our in-class discussion re: how to manage the arrows needed for the dynamic-programming traceback step.

Make a 5x6 array lled in with **None** \sentinel" values:

```
arrows = []
for i in xrange(5):
    arrows.append([])
    for j in xrange(6):
        arrows[-1].append(None)
```

Default formatting of the arrays:

```
print arrows
```

```
[[None, None, None, None, None, None], [None, None, None, None, None, None], [None, None, None, None, No
```

A \pretty" printing function { useful for debugging:

```
def dump(M):
    for i in M:
        for j in i:
            print j,
        print
```

Here's the array as initialized:

```
dump(arrows)
```

```
None None None None None None
None None None None None None
None None None None None None
None None None None None None
None None None None None None
```

A diagonal (\align") arrow pointing from the 4th row, 3rd column to the 3rd row, 2nd column:

```
arrows[3][2] = (2,1)
```

```
dump(arrows)
```

```
None None None None None None
None None None None None None
None None None None None None
None None (2, 1) None None None
None None None None None None
```

A vertical (\gap") arrow:

```
arrows[2][5] = (0,5)
```

```
dump(arrows)
```

```
None None None None None None
None None None None None None
None None None None None (0, 5)
None None (2, 1) None None None
None None None None None None
```

Note that it's easy to use the arrows directly as indices:

```
(i,j) = arrows[2][5]
arrows[i][j]
```

A parallel array for the subalignment scores:

```
scores = [range(6) for i in xrange(5)]
```

```
dump(scores)
```

```
0 1 2 3 4 5
0 1 2 3 4 5
0 1 2 3 4 5
0 1 2 3 4 5
0 1 2 3 4 5
```

```
dump(arrows)
```

```
None None None None None None
None None None None None None
None None None None None (0, 5)
None None (2, 1) None None None
None None None None None None
```

Since the arrays are parallel, I can index them in the same way:

```
(i,j) = arrows[2][5]
arrows[i][j]
scores[i][j]
```

```
5
```

Let's make another 5x6 arrows array, this time initializing with empty lists rather than **None** values { this gives us room to store more than one arrow per location:

```
arrows2 = []
for i in xrange(5):
    arrows2.append([])
    for j in xrange(6):
```

```
        arrows2[-1].append([])
```

```
dump(arrows2)
```

```
[] [] [] [] [] []
[] [] [] [] [] []
[] [] [] [] [] []
[] [] [] [] [] []
[] [] [] [] [] []
```

Adding one arrow:

```
arrows2[2][5].append((0,5))
```

```
dump(arrows2)
```

```
[] [] [] [] [] []
[] [] [] [] [] []
[] [] [] [] [] [(0, 5)]
[] [] [] [] [] []
[] [] [] [] [] []
```

Adding a second arrow to the same location:

```
arrows2[2][5].append((1,4))
```

```
dump(arrows2)
```

```
[] [] [] [] [] []
[] [] [] [] [] []
[] [] [] [] [] [(0, 5), (1, 4)]
[] [] [] [] [] []
[] [] [] [] [] []
```

```
%logstart -o BMS270b.2013.08.log
```

```
Activating auto-logging. Current session state plus future input saved.
Filename       : BMS270b.2013.08.log
Mode           : backup
Output logging : True
Raw input log  : False
Timestamping   : False
State          : active
```