CSCI 270 Lecture 8: Sequence Alignment

We are given two strings $X = x_1x_2...x_n$ and $Y = y_1y_2...y_m$, and we want to determine how similar these strings are.

X=ocurrance
Y=occurrence

- If we simply check how many positions i satisfy $x_i = y_i$, what will we determine is the difference between these two strings?
- The above metric isn't very intelligent. What number should we really return?

The **edit distance** between two strings is the minimal distance possible between two strings after inserting your choice of spaces. Our goal is to efficiently calculate the edit distance between X and Y.

X=oc-urrance
Y=occurrence

Edit distance = 2

- We need to break the problem into bite-size decisions. What should be our first question?
- The top level of recursion will handle the first decision only. What information do we need to pass down to the next level of recursion?

Define: $SA[i, j]$ is the min cost of aligning strings $x_ix_{i+1}...x_n$ and $y_jy_{j+1}y_m$.

- If $x_i = y_j$, what recursive call should I make?
- What cost is incurred if $x_i \neq y_j$?
- If I accept the mismatch between x_i and y_j, what recursive call should I make?
- If I insert a gap in the X string, what recursive call should I make?
- Under what conditions should I stop recursing?

$SA[i, j] = SA[i + 1, j + 1]$, if $x_i = y_j$
$SA[i, j] = 1 + \min(SA[i + 1, j], SA[i, j + 1], SA[i + 1, j + 1])$, if $x_i \neq y_j$.
$SA[i, m + 1] = n - i + 1$
$SA[n + 1, j] = m - j + 1$

- What order do I fill the array in?

- Where is the answer stored?

- What is the runtime?

- What are the space requirements?

	A	C	A	C	A	C	T	A	
A	2	3	3	4	5	6	7	7	8
G	2	2	3	3	4	5	6	6	7
C	2	1	2	2	3	4	5	5	6
A	3	2	1	2	2	3	4	4	5
C	4	3	2	1	2	2	3	3	4
A	5	4	3	2	1	2	2	2	3
C	6	5	4	3	2	1	1	1	2
A	7	6	5	4	3	2	1	0	1
	8	7	6	5	4	3	2	1	0

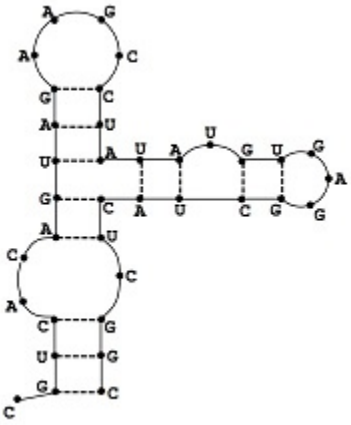
This algorithm is used in computational genetics, which has absolutely huge strings.

- How could one reduce the space requirements?

- Are there any drawbacks to this solution?

RNA Secondary Structure

RNA is a string $B = b_1 b_2 ... b_n$ over the alphabet $\{A, C, G, U\}$. RNA is single-stranded (as opposed to DNA which is double-stranded), and loops back and forms pairs with itself. The "secondary structure" is determined by figuring out what the pairings are.



In reality, how RNA bonds with itself is very complicated. We will simplify it a bit in order to get an approximation of how the structure will look. We will assume the following 3 rules are always followed:

- Each pairing is AU or CG.
- The ends of each pair are separated by at least 4 bases: if $(b_i, b_j) \in S$, then $i < j - 4$.
- No pairs cross each other: if $(b_i, b_j), (b_k, b_l) \in S$, it is not the case that $i < k < j < l$

We will assume that the RNA strand forms the maximum possible number of pairs according to the above rules.

- What should our first decision be?

- What information do we need to pass down to the next level of recursion?

$RNA[i,j]$ = max number of pairings for string $b_i b_{i+1} ... b_j$.

- If we do not pair b_i with anything, what recursive call should I make?
- If we pair b_i with something, how does the number of pairs found change?
- What subproblems (plural!) do we need to consider if we pair b_i with b_j?
- Under what conditions should we stop recursing?

$RNA[i,j] = \max(RNA[i+1,j], 1 + \max_t(RNA[i+1,t-1] + RNA[t+1,j]))$
$RNA[i,j] = 0$, if $i \geq j - 4$

- What choices of t are valid?
- What order do I fill the array in?
- Where is the answer stored?
- What is the runtime?