COMP 5970/6970-004
Computational Biology: Genomics and Transcriptomics
Lecture notes 1: 1/13/2022

Haynes Heaton M.D., Ph.D.

Spring, 2022

Lecture Objectives

- Understand what makes problems amenable to Dynamic Programming
- Define types of sequence differences
- Global alignment algorithm
  - Scoring system
  - Score matrix and backtrace annotations
  - Backtrace/Viterbi algorithm
  - Generate resulting alignment from backtrace
  - Runtime analysis

DNA

For now, DNA is just a string on the alphabet {A,C,G,T}.

Sequence Comparison

But this string does represent sequence from a genome. Different genomes have similarities and differences. These similarities may be due to

1. Evolutionary relatedness (distant common ancestor to familial related individuals)
2. Functional similarity (if functionally related and not evolutionary related, this can be due to convergent evolution)

These sequences may also have differences due to mutations if evolutionarily related or other differences if functionally similar. These differences can be thought of as

- Mismatches (A $\rightarrow$ G for example)
- Insertions (eg AGT $\rightarrow$ AGGT)
- Deletions (eg AGGT $\rightarrow$ AGT)

And when talking about pairwise alignment, since insertions with respect to one sequence are deletions with respect to the other sequence, these are termed indels. Single base mismatches may be due to sequencing errors or, if true mismatches, are called single nucleotide polymorphisms, or SNPs.

Pairwise Sequence Alignment

An alignment of two sequences can be represented by which bases from one sequence align with which bases in another sequence and in the case of an indel, which bases align with no base in the other sequence. This is usually represented by each sequence on a separate line with bases in the same column as aligned to one another and dashes for indels. For example:
GTCGTAGAATA
GTAGTAGA-TA
where the third base is a mismatch and the third to last column is an indel.

Scoring

If we wish to find an optimal alignment, we must have something to optimize. So we assign values to matches, mismatches, and indels. The sum of these values represents a score. The goal of an alignment algorithm is to find the alignment between two sequences that maximizes this score. For simplicity in this lecture we use the following scoring system:

- match: +1
- mismatch: -1
- indel: -1

Dynamic Programming

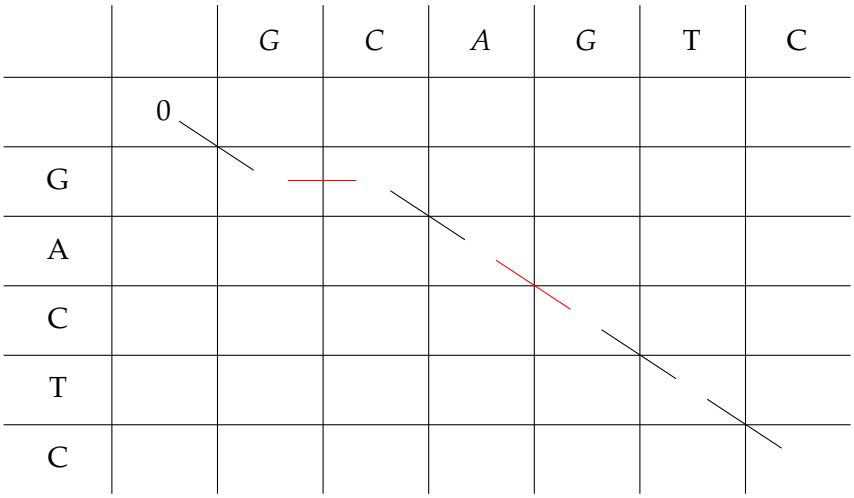
Dynamic programming is a class of algorithms. Much like other classes of algorithms, there is a property that some problems have that make them amenable to solutions using dynamic programming. In divide and conquer algorithms, for example, the problems must have some relatively easy way to combine sub-solutions into a solution of a larger problem. Dynamic programming is a poor name for this algorithmic strategy. I prefer inductive optimization. The word programming in this name comes from a time before computers were common and didn't mean coding but instead meant optimization. And dynamic is just an example of lazy naming as it is nonspecific. I instead use the word inductive after the idea of proof by induction. The

properties one needs for a proof by induction is a base case and the ability to prove the veracity of the statement at step $i+1$ given correctness at step i. Similarly, in dynamic programming, we start with some trivial base case solution and the property needed for dynamic programming to be effective is given an optimal solution at step i, we can easily compute the optimal solution at step $i+1$. But in our case, we will index one sequence with i and the other sequence with j and we will need solutions at $i-1, j-1$, $i-1, j$, and $i, j-1$ in order to generate a solution at i, j.

Global Alignment / Needleman-Wunsch algorithm

		G	*C*	*A*	*G*	T	C
	0						
G							
A							
C							
T							
C							

This is our score matrix with the base case that an empty alignment has a score of 0.



As you can see, a path through this matrix represents an alignment. Diagonal arrows represent matches or mismatchs while horizontal or vertical paths represent indels. The alignment represented by this path is

```
GCAGTC
G-ACTC
```

Because horizontal and vertical paths represent indels, we can initialize the border from the base case as follows.

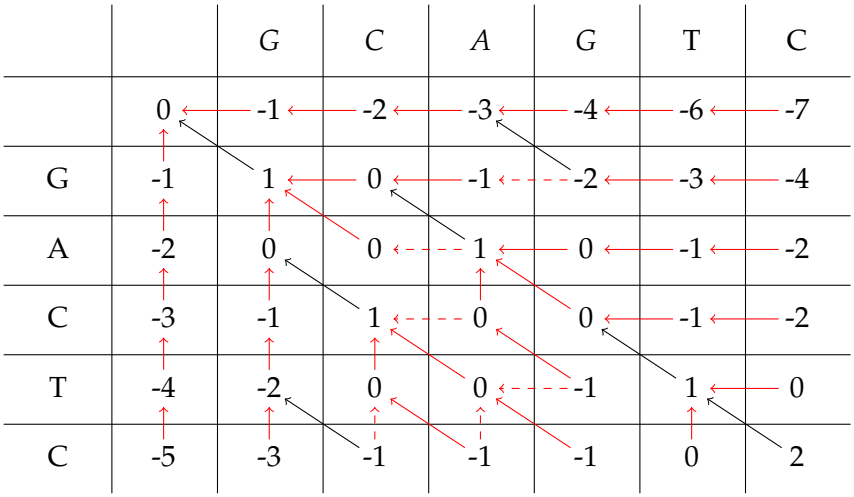
		G	*C*	*A*	*G*	T	C
	0 ←	-1 ←	-2 ←	-3 ←	-4 ←	-6 ←	-7
G	↑ -1						
A	↑ -2						
C	↑ -3						
T	↑ -4						
C	↑ -5						

We keep track of where we came from for each cell with a backtrace pointer.

Now we fill out the score matrix with the following rules where m is the scoring matrix and s_1 is the string/sequence 1 and s_2 is the string/sequence 2.

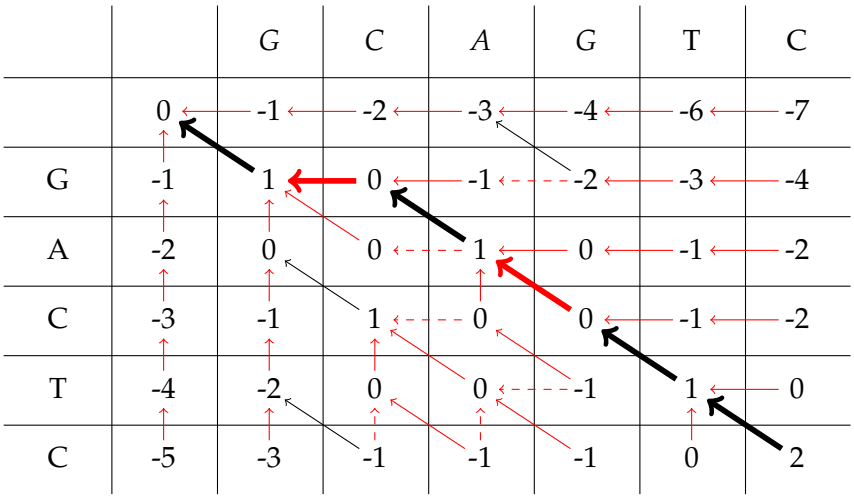
$$m[i,j] = max \begin{cases} m[i-1,j-1]+1 & \text{if} \quad s_1[i] = s_2[j] \text{ (match)} \\ m[i-1,j-1]-1 & \text{if} \quad s_1[i] \neq s_2[j] \text{ (mismatch)} \\ m[i-1,j]-1 \text{ (indel)} & \\ m[i,j-1]-1 \text{ (indel)} & \end{cases}$$

And keep an annotation of which cell we came from. Now we fill out all of the cells with these rules. Red arrows represent mismatches or indels. Dashed arrows represent equal scores which are ignored. If scores are equal, we favor matches or mismatches arbitrarily because we aren't looking for all optimal alignments, just one optimal alignment.



Backtrace / Viterbi Algorithm

Now to find the optimal global alignment, we start at the bottom righthand cell and follow our backtrace arrows to find the optimal path through the matrix.



which creates the following alignment

```
GCAGTC
G-ACTC
```