BioE/MCB/PMB C146/C246: Topics in Computational Biology and Genomics

Lecture 1) Introduction to Sequence Evolution

21 Jan 2003

Scribe: Derek Chiang *Reader*:

INTRODUCTION TO SEQUENCE ALIGNMENT

Finding the best alignment between two sequences is a fundamental problem in computational biology. We briefly consider this problem to illustrate the importance of efficient algorithms in this field.

How many possible alignments exist between any two sequences of length m and n (assume $n > m$)? For example, consider the two sequences: ACGCAT
AGCGCA

If no gaps are allowed, there can be n possible alignments, so time complexity is $O(n)$[1].

If any number of gaps is allowed, as long a gap is not aligned to another gap, there number of possible alignments is given below. The factorial terms come from the number of ways to choose columns in which to insert k gaps in the two sequences.

$$\sum_{k=0}^{m}\left(\frac{m!}{k!(m-k)!}\right)\left(\frac{n!}{n!(n-k)!}\right)\approx\frac{2}{\pi}4^{n}$$

The time complexity of searching through every possible alignment would make this method infeasible. In future lectures, we will discuss an important technique, called dynamic programming, to reduce this time complexity to $O(n^2)$. Other heuristic methods, such as BLAST, can accomplish even faster searches.

Dynamic Programming applied to Fibonacci numbers

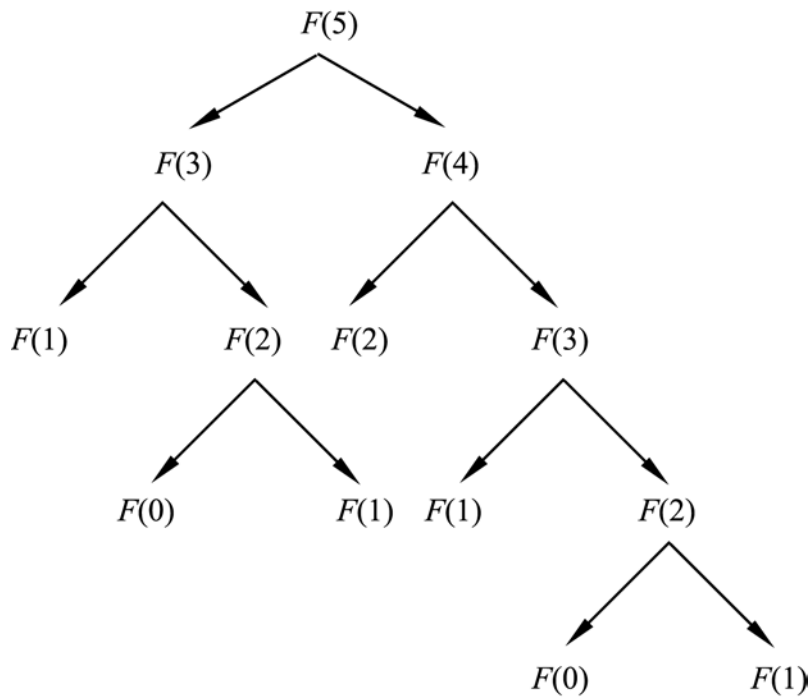
Fibonacci numbers are defined recursively by the following formulae:

$$F(n) = F(n-1) + F(n-2)$$
$$F(1) = 1$$
$$F(0) = 1$$

[1] Big-oh notation summarizes the worst-case time or memory complexity of an algorithm, as a function of the input data size. Only the highest exponential term matters, *e.g.,* $O(3n^2 + 5n)$ should always be re-written as $O(n^2)$. The following list is sorted in ascending order of time complexity (algorithms closer to the top are better).

$O(1)$	Constant	Exact formula for Fibonacci numbers
$O(\log n)$	Logarithmic	Binary search through a sorted list of numbers
$O(n)$	Linear	Reading through a list of numbers
$O(n \log n)$		Merge sort
$O(n^2)$	Quadratic	Insertion sort; dynamic programming alignments
$O(k^n)$	Exponential	Enumerating all possible words of length n with an alphabet of size k
$O(n!)$	Factorial	Naïve method to search all alignments of length n

A naïve way to write a program to compute the 5th Fibonacci number would be to solve for $F(5)$ recursively. The program execution can be represented by a tree, with branches representing nested levels of recursion (note that branching from one of the $F(2)$ terms has been omitted). The branching terminates when either of the base cases, $F(0)$ or $F(1)$, is reached.



The time complexity of a recursive implementation is approximately $O(n^{1.6})$. Of course, this procedure is not the intuitive way (for humans; computers do what they're told) to compute Fibonacci numbers. Instead, we would simply store the intermediate values by writing down the Fibonacci sequence. This process of storing intermediate values, called memoization or caching, turns out to be the dynamic programming solution to the Fibonacci problem, which has time complexity $O(n)$.

n	0	1	2	3	4	5
$F(n)$	1	1	2	3	5	8

There is an even faster way, *i.e.* $O(1)$, to obtain the nth Fibonacci number, using Binet's formula (but actually attributable to Euler):

$$F(n-1) = \frac{\Phi^n - \phi^n}{\Phi - \phi}$$

where $\Phi = (1 + \sqrt{5}) / 2$ also known as the golden mean
and $\phi = (1 - \sqrt{5}) / 2$

SEQUENCE EVOLUTION

Effects of Sequence Mutations

Causes of mutations	*Possible effects on proteins*	*Possible effects on organism*
Errors in DNA replication DNA damage Errors in crossover	Silent: codon degeneracy Conservative mutation Changes to protein structure Change in active site specificity	Lethal Masked by duplication or other redundancy Superior fitness

In summary, mutations can be grouped under 3 categories. Silent mutations to the same or similar amino acids may not affect the protein's function. Negative mutations may interfere with protein activity by affecting its structure or specificity. In rare cases, positive mutations may introduce improvements by changing the protein's structure or specificity.

If an organism dies due to a mutation causing a malfunction in the protein, the mutation should disappear. However, if the mutation is retained (and observed in extant sequences), then we assume that it had been accepted at some point in evolutionary history. Although selection occurs after every round of DNA replication, it occurs on the level of protein function.

Consider a hypothetical example of sequence evolution for an essential gene:

Years ago	*DNA sequence similarity*	*Protein sequence similarity*	*Protein structure similarity*	*Functional similarity*
800 million	*D* %	< *D* %	High	Pretty well conserved
400 million	Very low	Moderately	Moderate	Pretty well conserved
Today	Very low	Very low	Quite similar	Pretty well conserved

Imagine that a gene duplication occurred about 200 million years ago. There are several possible scenarios for the other gene. One possibility is that one copy's sequence changed even more rapidly, and perhaps even gained a new function, than the other.

Detecting Evolutionary Relationships

As a rule, related sequences become more and more divergent, rather than more similar. In general, protein structure similarity > protein sequence similarity > DNA sequence similarity. Therefore, most powerful tools for discovering evolutionary relationships between proteins involve structural comparisons. But even in the absence of protein structure, comparisons between protein sequences perform much better than comparisons between DNA sequences.

Since function involves some complicated relationship among structural similarity, protein sequence similarity and DNA sequence similarity, the inference of functional similarity can be difficult. In particular, similarity at a sequence or structural level does not necessarily guarantee functional similarity.

Mutation rates are not constant among different classes of protein sequences. Examples of protein families that evolve quickly include host immune response proteins, as well as cell

surface receptors of pathogens. Examples of proteins that evolve slowly include polymerases, histones and cytochrome c (which is part of the electron transport chain for ATP synthesis). In general, there is a high correlation between the number of interactions at protein surfaces and mutation rate.

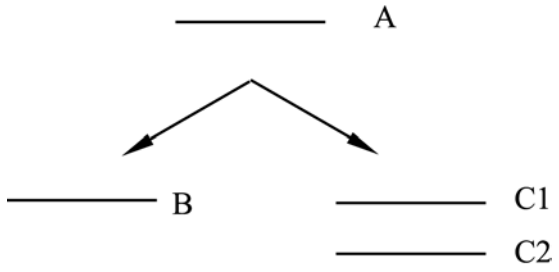
Mutation rates may also vary within proteins, due to particular features within proteins (such as active site residues of enzymes, residues important for the protein fold, etc.)

For both of the cases mentioned above, note that the mutation rate variations are primarily due to variations in selective pressures on the protein sequence, not positional variations in the mutability of DNA.

Vocabulary for Evolutionary Relationships

When we align sequences, we do not observe the ancestral sequences, only those present in the extant organisms.

Two sequences that are evolutionarily related are **homologous**. Note that this term is NOT tautologous to *similarity*; in particular, two sequences that are similar are NOT necessarily homologous. In addition, the concept of homology (just like pregnancy) is indivisible: either a pair of sequences is homologous or not (see the Fitch articles on the course website).

The terms **orthologs** and **paralogs** are applied based on the evolutionary history of sequences. Orthologous genes are related to each other via species divergence. Paralogous genes are related to each other via gene duplication.

Think carefully about the application of these terms to the example below, which will help you with your first problem set.