Introduction

Motivating Application: Sequence Alignment

Algorithms: Design and Analysis, Part II

Sequence Alignment

Sequence alignment: fundamental problem in computational genomics.

Input: two strings over the alphabet {A, C, G, T}.
[portions of one or more genomes]

Problem: figure out how "similar" the two strings are.

Example
AGGGCT
AGGCA

Example applications

1. extrapolate function of genome substrings
2. similar genomes can reflect proximity in evolutionary tree

Measuring Similarity

Question: what does "similar" mean?

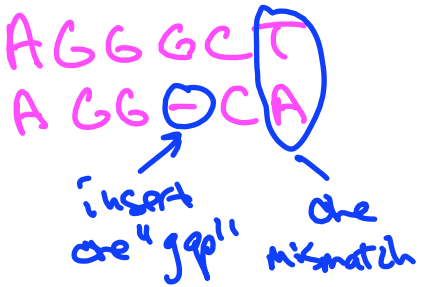
Intuition: AGGGCT, AGGCA are similar because they can be "nicely aligned".

Idea: measure similarity via quality of "best" alignment.

Assumption: have experimentally determined penalties for gaps and the possible matches.

Example

AGGGCT
AGGCA



Problem Statement

Input: 2 strings over $\{A, C, G, T\}$.
- Penalty $pen_{gap} \geq 0$ for each gap
- Penalty $pen_{AT} \geq 0$ for mismatching A & T
- etc.

Output: alignment of the strings that minimizes the total penalty.

$\Rightarrow$ called the Needleman-Wunsch score [1970]

Small NW score $\approx$ similar strings

Algorithms Are Fundamental

Note: this measure of genome similarity would be useless without an efficient algorithm to find the best alignment.

Brute-force search: try all possible alignments, remember the best one.

Question: Suppose each string has length 500. Roughly how many possible alignments are there?

(A) # of students in this class $\approx 10^4 - 10^5$

(B) # people on earth $\approx 10^9 - 10^{10}$

(C) # atoms in known universe $\approx 10^{80}$

(D) more than any of the above $\geq 2^{500} \geq 10^{125}$

Point: need a fast, clever algorithm

Solution: straightforward dynamic programming.