CSE 527 10.2.06
Lecture 2 Justin Shaffer

Sequence Alignment – Part 1

Administrative Notes
- Homework 1 is due 10.9.06
- Possible programming languages to use for class
  - R, Ruby, Python, C, C++, Java, Perl, MATLAB, Octave,…

People have been getting sequences since the 1950s
- Essential to computational biology
- Need to be searchable for comparison purposes

What is sequence similarity?
- Aligning two sequences by common nucleotides
- Can include spacers to make a better fit

Why is sequence alignment important?
- Can compare sequences to databases of sequences
  - Similar sequences often have similar origin or function
- Selection and survival occurs at the system level, but mutations occur at the sequence level
  - Mutations in DNA can occur through chemical, radiation, or transcription errors
- Recognizable similarity is noticeable after 10^8 to 10^9 years

Can use Genbank search to check sequences
- BLAST is a sequence comparison tool
- Can compare nucleotide or protein sequences or entire genomes for similarity
- http://www.ncbi.nlm.nih.gov/blast/
  - lower case nucleotides may indicate uncertainty in sequence
  - predicted sequences are determined by some algorithm
  - taxonomic score gives the number of hits by taxonomy
  - E-values start at 0.0 for near perfect matches
    - For a given match, the E-value describes the expected number of matches that you will find that are as good or better than the current one, in a random data base of the same size.
- BLAST is useful because…
  - Webserver
  - Fast
  - E-values give statistical significance of match

Sequence Terminology
- **String** – ordered list of letters
- **Prefix** – consecutive letters from front of string
- **Suffix** – consecutive letters from end of string

- **Substring** – letters from end or middle
- **Subsequence** – ordered, nonaligned letters
- **Alignment** – of strings S and T is a pair of strings (with spaces) S' and T'
    - |S'| = |T'| = length of S

Alignment Scoring

- Mismatch (-1), Match (+2) [For examples on slides, only.]
- The score of aligning two sequences S and T is σ(S, T)
- The value of an alignment is the sum of all of the scores of the strings S' and T' from one to |S'| -- BIG assumption, e.g. assumes adjacent positions independent
- The optimal alignment is the one that results in the maximum alignment score
    - Bonuses for correct alignment, penalties for mistakes
- Scoring amino acid sequence alignment can be difficult
    - Scores can be based on side chains
    - Reflects chemical/physical properties of amino acids

Where do scores come from?

- Develop an algorithm to compare sequences and tabulate maximal score
- Simple method
    - For all subsequences A of S and B of T, set |A| = |B|
    - Align A(i) = B(i) for 1 <= I <= |A|
    - Align all other characters to spaces
    - Compute values
    - Retain the max alignment
- Assume n = |S| = |T|
    - Cost of evaluation of one alignment is 2n
    - Polynomial versus exponential growth
        - 2^{2n} hits wall really fast
        - run time grows with stiffness

Example: Fibonacci Numbers

- Uses a simple recursion loop, but results in a huge number of cycles (subproblems)
    - Values at n – 1 and n – 2 is calculated for every cycle
    - Time = $\Omega(1.61^n)$
- Can use dynamic programming to greatly speed up run time
    - By using a table or array, values from each iteration can be stored into memory and thus do not need to be calculated every cycle
        - Time = O(n)

What is the optimal substructure to use for determining alignment?

- The optimal alignment ends in one of three ways…
    - Last character of S and T are aligned to each other
    - Last character of S is aligned with a spacer in T
    - Last character of T is aligned with a spacer in S
    - Never align spacer with spacer (σ(--,--) < 0)

- In each case, the remainder of S and T should be optimally aligned to each other
- The optimal alignment can be accomplished in $O(n^2)$ time by using dynamic programming
    - Input: S and T, |S| = n and |T| = m
    - Output: value of optimal alignment
- It is easier to solve a "harder" problem
    - V(i,j) = value of optimal alignment of S[1], S[2], … , S[i] with T[1]…T[j]
    - Etc, etc…

Recursion

- See powerpoint notes for example of how to use the following algorithm
    - $V(i,j) = \max \begin{cases} V(i-1,j-1) + \sigma(S[i],T[j]) \\ V(i-1,j) + \sigma(S[i],--) \\ V(i,j-1) + \sigma(--,T[j]) \end{cases}$ for all $1 <= I <= n$, $1 <= j <= m$
- fill in the entries row by row or column by column in order to fill in the entire table
    - S is for rows
    - T is for columns
- The time to run this algorithm will be O(m*n)
- The goal is to find the n x m entry of the table
    - This will tell you the score of the overall best match, but not what the match is!
    - To find out what the best match is, trace back in the table to the 1 x 1 entry

Complexity Notes

- Time = O(m*n)
- Physical space = O(m*n)
- Practical to use this algorithm for small values of m and n
    - Space can be more of a limitation than time (there's a more complex algorithm that reduces space to O(max(m,n)), still in O(mn) time).

Part II – Variations in Sequence Alignment

- Local alignment
    - Preceding algorithm gives global alignment (uses the full length of both strings)
    - This method might well miss strong similarity of the middle of the strings
- Gap penalties
    - Some worth more than others
    - Gaps are correlated
        - Better to lose 3n nucleotides than any other number
            - 3 nucleotides per codon

More on these variants next lecture.