Outline

- Recursion
- More Dynamic Programming
- Scoring Matrices
- BLAST

Recursion

General Idea: Solve a problem by solving related sub-problems and combining the results.
Advantage: Sub-problems are easier to solve and their total run time can be much less than original problem.
Example: Merge Sort – To sort a large list, split the list in half, sort each half, and merge the two sorted lists.
Example: Factorial(x)

\[ x! = x \times (x-1) \times (x-2) \times \ldots \times 1 \]

5! = 5 \times 4 \times 3 \times 2 \times 1 = 120

Recursion - Factorial

Iterative Algorithm:

```
int factorial(int n) {
    int x = 1;
    for (int i = 2; i <= n; i++) {
        x *= i;
    }
    return x;
}
```

Recursive Algorithm:

```
int factorial(int n) {
    if (n == 1) {
        return 1;
    } else {
        return n * factorial(n-1);
    }
}
```

Dynamic Programming

General Idea: Solve a problem by solving related sub-problems and combining the results.
Key Point: In recursive algorithms (e.g., Merge Sort), the sub-problems are independent. In dynamic programming the sub-problems share sub-sub-problems.
Implementation: Solve each sub-sub-problem only once and store the answers in an array.

Dynamic Programming – Step 1

Characterize the structure of the optimal solution.
Example: Sequence Alignment

Three choices at each step:
1. Align a base in X with a base in Y
2. Align a base in X with a gap
3. Align a base in Y with a gap
Goal:
Maximize the total score

Dynamic Programming

Scoring matrices
BLAST

Section 3

Griffin Weber
Oct. 7th, 2003
Dynamic Programming – Step 2
Recursively define the value of an optimal solution.
Example: Needleman-Wunsch (Global Alignment)

\[ F(i,j) = \max \{ \]
\[ F(i-1, j-1) + s(x_i, y_j) , \]
\[ F(i-1, j) - d , \]
\[ F(i, j-1) - d \] \}

Dynamic Programming – Step 3
Compute the optimal solution, starting from the easiest sub-sub-problems and saving the results.

Dynamic Programming – Step 4
Construct an optimal solution from the computed information. (Perform a trace-back.)

Multiple Sequence Alignment
Dynamic programming in many dimensions!!!

Scoring Matrices
Definition: A table (matrix) of values that describe the probability of a residue (amino acid or base) pair occurring in an alignment of related sequences.

Why a matrix? Evolution tends to favor different sets of substitutions. Thus, a single probability does not accurately describe all residue pairs.
Example: PAM

PAM = Percent Accepted Mutation

- Ranks a.a. substitutions on how well they're "accepted" by natural selection, without changing the function of the protein.
- Derived from global alignments of closely related sequences (85% identical).
- PAM1 models the mutation rate in 1 "PAM Unit" of evolutionary time.
- \( PAM^n = (PAM1)^n \) = mutation rate in \( n \) PAM Units
- Examples: PAM40, PAM100

Example: BLOSUM

BLOSUM = Blocks Amino Acid Substitution Matrices

- Derived from large data sets of local, ungapped alignments of biochemically related sequences.
- Each matrix is derived from different data sets.
- BLOSUMn means no pair of sequences in the data set used to derive the matrix had more than n% similarity.
- BLOSUM62 indicates the substitution frequencies for sequences that are up to 62% similar

PAM vs BLOSUM

- BLOSUM better than PAM for local similarity
- BLOSUM better in similarity searches in databases
- PAM incorporates an evolutionary model representing species divergence
- BLOSUM tolerant of hydrophobic changes and of cysteine and tryptophan mismatches
- PAM tolerant of change to/from hydrophilic a.a.

BLAST

BLAST = Basic Local Alignment Search Tool

- At least 50 times faster than dynamic programming
- Not guaranteed to find the optimal solution, but…
  - Individual alignments need not be perfect, you can always fine-tune later.
  - Most sequences will be completely unrelated to the query.
  - Will always find related sequences in a database that meet some similarity criteria.
- BLAST is a heuristic algorithm (usually works well)
- Always translate DNA into protein sequences (if possible) before running BLAST.

BLAST : K-mer Words

- Create a list of all "words" of length K (default is 3 amino acids or 11 nucleotides)
- Example: TPQGQR \( \rightarrow \) TPQ, PQG, QGQ, GQR
**BLAST: K-mer Hashing**
- For each k-mer, list all locations where it occurs in the database.
- Match the query to k-mers (or “neighborhood words” above a threshold) in the list. This forms seeds.

**BLAST: Extending Seeds**
- Connect “close” seeds into single long sequences.
- Extend seeds until the cumulative score drops.
- High-scoring Segment Pair = locally maximal segment.

**BLAST: Evaluating HSPs**
- Bit Score: independent of the scoring matrix used.
- E-value: “Expected number of High-Scoring Segment Pairs”. The smaller the number, the better the alignment.
- E-values of 0.1 or 0.05 are typically used as cutoffs.

**Next Week**
- Microarrays
- Sequencing

**Acknowledgement / References**
This handout includes material written by Suzanne Komili, Yonatan Grad, Doug Selinger, and Zhou Zhu.

*Mount, Bioinformatics – Sequence and Genome Analysis; Durbin et al., Biological Sequence Analysis; Cormen et al., Introduction to Algorithms*