Local Sequence Alignment & Heuristic Local Aligners

Lectures 18 – Nov 28, 2011
CSE 527 Computational Biology, Fall 2011

Instructor: Su-In Lee
TA: Christopher Miles

Monday & Wednesday 12:00-1:20
Johnson Hall (JHN) 022

1

Review: Probabilistic Interpretation

```
X: TCCAGGTG-GAT
   |  ||  |||  |  |
Y: TGCAAGTGCG-T
```

Chance or true homology?

Sharing a common ancestor

2

Review: Likelihood Ratio

X: **TCCAGGTG-GAT**

Y: **TGCAAGTGCG-T**

$$\frac{\Pr(\text{Data}\mid\text{Homology})}{\Pr(\text{Data}\mid\text{Chance})}$$

3

Review: Log Likelihood Ratio Score

- The most commonly used alignment score of aligning two sequences is the log likelihood ratio of the alignment under two models
  - Common ancestry
  - By chance

$$Score = \log\left(\prod_i \frac{\Pr(x_i y_i)}{\Pr(x_i)\Pr(y_i)}\right) =$$

$$= \sum_i \log\left(\frac{\Pr(x_i y_i)}{\Pr(x_i)\Pr(y_i)}\right) = \sum_i s(x_i, y_i)$$

4

Outline: Scoring Alignments

- Scoring alignments
  - Probabilistic meaning
  - Scoring matrices
    - PAM: scoring based on evolutionary statistics
    - BLOSUM: tuning to evolutionary conservation
  - Gaps revisited
- Local vs global alignment
- Database search
  - FASTA
  - BLAST

5

Gap Initiation and Extension

```
TCCACCGTG-GA
|  ||   ||| |  ||
TGCA--GTGCGA
```

6

Gap Initiation and Extension

Insertion / deletion (indel)

7

Scoring Indels: Naive Approach

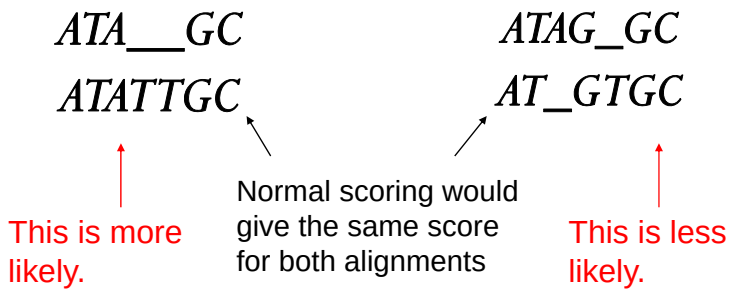
- A fixed penalty d is given to every indel:
  - $-d$ for 1 indel,
  - $-2d$ for 2 consecutive indels
  - $-3d$ for 3 consecutive indels, etc.

Can be too severe penalty for a series of 100 consecutive indels!

8

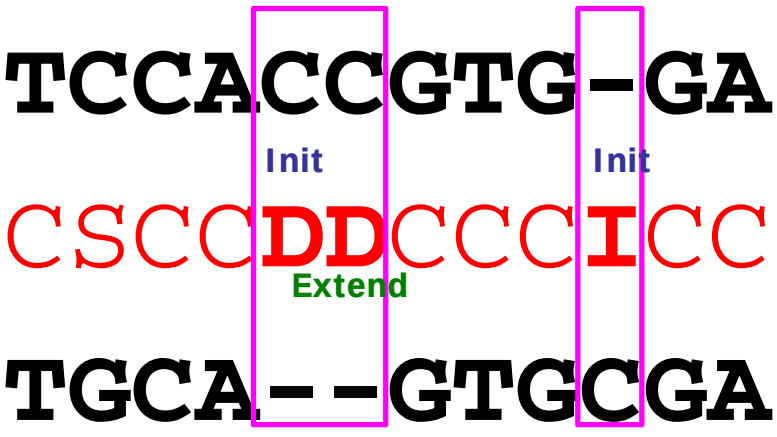
Affine Gap Penalties

- In nature, a series of *k* indels often come as a single event rather than a series of *k* single nucleotide events:



9

Gap Initiation and Extension



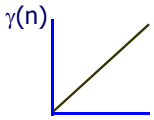
Insertion / deletion (indel)

10

Scoring the Gaps More Accurately

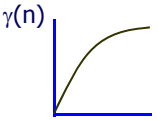
- Current model:

  Gap of length n
  incurs penalty n×d



- However, gaps usually occur in bunches

- Convex gap penalty function:
  γ(n):
  for all n, $\gamma(n+1) - \gamma(n) \leq \gamma(n) - \gamma(n-1)$



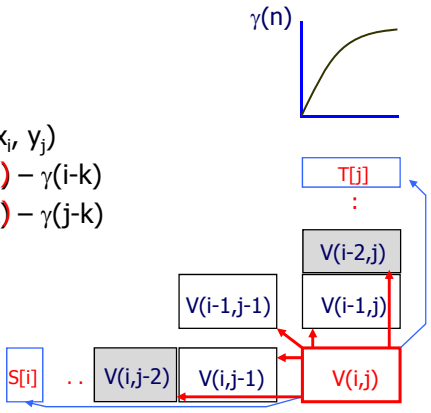
General Gap Dynamic Programming

Initialization: same

Iteration:

$$V(i,j) = \max \begin{cases} V(i-1, j-1) + s(x_i, y_j) \\ \max_{k=0\ldots i-1} V(k,j) - \gamma(i-k) \\ \max_{k=0\ldots j-1} V(i,k) - \gamma(j-k) \end{cases}$$

Previously…



Termination: same

Running Time: $O(N^2M)$ (assume N>M)
Space: O(NM)

Accounting for Gaps

- *Gaps*- contiguous sequence of spaces in one of the rows

- Score for a gap of length x is:

  $-(d + ex)$

  where $d > 0$ is the penalty for introducing a gap:

  gap opening penalty

  d will be large relative to e:

  gap extension penalty

  because you do not want to add too much of a penalty for extending the gap.

13

Affine Gap Penalties

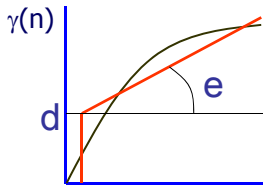
- Gap penalties:
  - $-d - e$ when there is 2 indel
  - $-d - 2e$ when there are 3 indels
  - $-d - 3e$ when there are 4 indels, etc.
  - $-d - (n-1) \cdot e$ when there are n indels

- Somehow reduced penalties (as compared to naïve scoring) are given to runs of horizontal and vertical arrows in the V matrix

14

Needleman-Wunsch With Affine Gaps

- $\gamma(n) = d + (n-1) \times e$
  - d: gap open
  - e: gap extend



- To compute optimal alignment,
- At position i,j, need to "remember" best score if gap is open
  best score if gap is not open
- F(i, j): score of alignment $x_1 \ldots x_i$ to $y_1 \ldots y_j$ **if** x_i aligns to y_j
- G(i, j): score **if** x_i or y_j aligns to a gap

15

Needleman-Wunsch With Affine Gaps

Initialization:

$$F(i, 0) = d + (i-1) \times e$$
$$F(0, j) = d + (j-1) \times e$$

Iteration:

$$F(i, j) = \max \begin{cases} F(i-1, j-1) + s(x_i, y_j) \\ G(i-1, j-1) + s(x_i, y_j) \end{cases}$$

$$G(i, j) = \max \begin{cases} F(i-1, j) - d \\ F(i, j-1) - d \\ G(i, j-1) - e \\ G(i-1, j) - e \end{cases}$$

Termination: same

16

Outline: Scoring Alignments

- Scoring alignments
  - Probabilistic meaning
  - Scoring matrices
    - PAM: scoring based on evolutionary statistics
    - BLOSUM: tuning to evolutionary conservation
  - Gaps revisited
- Local vs global alignment
- Database search
  - FASTA
  - BLAST

17

Local vs. Global Alignment

- The Global Alignment Problem tries to find the highest scoring alignment between input sequences S (of length n) and T (of length m) – S[1-n] and T[1-m].
- The Local Alignment Problem tries to find the highest scoring alignment between the substrings S[i-i'] and T[j-j'], where i,j>0, i'<n+1 and j'<m+1.
  - In the "V matrix" (alignment scores of substrings) with negatively-scored arrows, Local Alignment may score higher than Global Alignment

18

Local vs. Global Alignment (cont'd)

- Global alignment

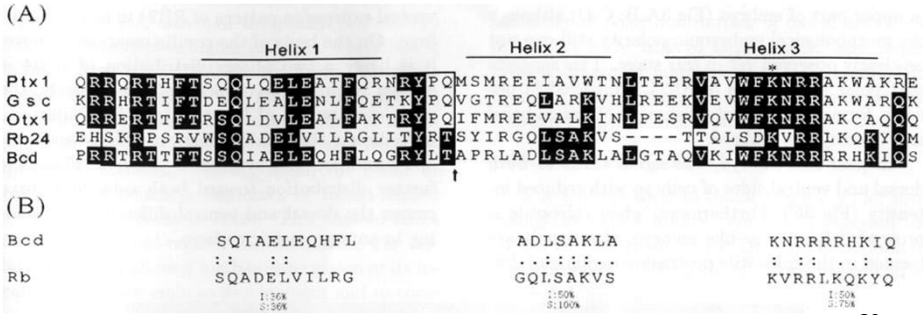
```
--T--CC-C-AGT--TATGT-CAGGGGACACG-A-GCATGCAGA-GAC
  |  || |  ||  | | | |||    || | | |  | ||||   |
AATTGCCGCC-GTCGT-T-TTCAG----CA-GTTATG-T-CAGAT--C
```

- Local alignment: better alignment to find conserved segment

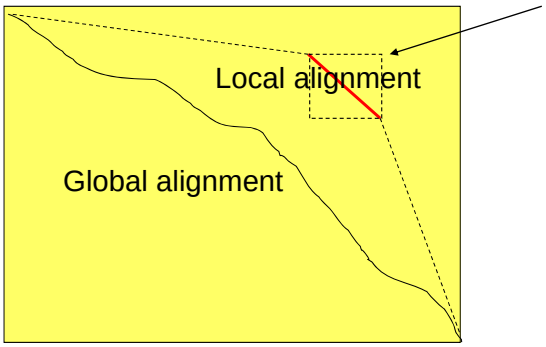
```
                  tccCAGTTATGTCAGgggacacgagcatgcagagac
                     ||||||||||||
aattgccgccgtcgttttcagCAGTTATGTCAGatc
```


Local Alignments: Why?

- Genes are shuffled between genomes
  - Two genes in different species may be similar over short conserved regions and dissimilar over remaining regions.
- Portions of proteins (domains) are often conserved



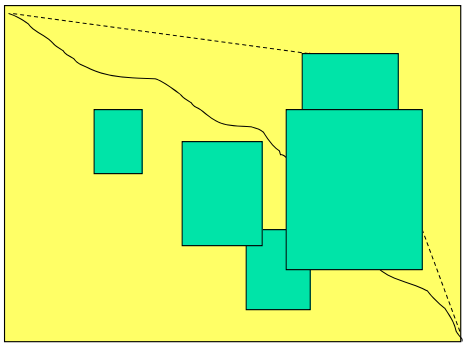
Local Alignment: Example



Compute a “mini” Global Alignment to get Local

21

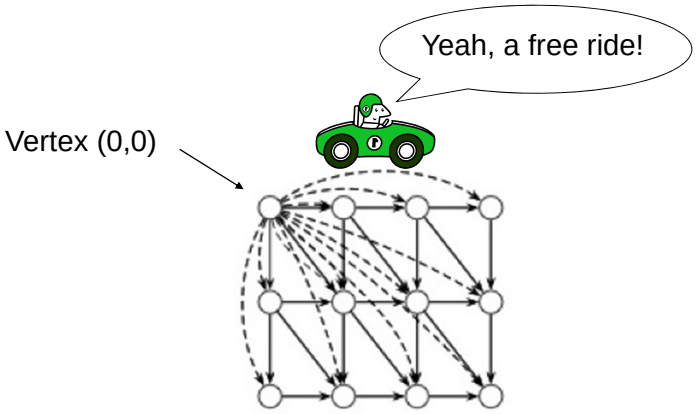
Local Alignment: Example



- Local run time $O(n^4)$:
  - In the grid of size n x n, there are ~n^2 vertices (i,j) that may serve as a source.
  - For each such vertex computing alignments from (i,j) to (i’,j’) takes $O(n^2)$ time.
- This can be remedied by giving free rides.

22

Local Alignment: Free Rides



The dashed arrows represent the free rides from (0,0) to every other entry in the V matrix.

23

The Local Alignment Problem

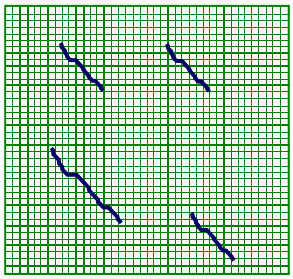
- **Goal:** Find the best local alignment between two sequences
- **Input :** Sequences S, T and scoring matrix σ
- **Output :** Alignment of sequences S and T whose alignment score is maximum among all possible alignment of all possible substrings

24

The Smith-Waterman Algorithm

Idea: Ignore badly aligning regions

Modifications to Needleman-Wunsch:



Initialization: $V(0, j) = V(i, 0) = 0$

Iteration: $V(i, j) = \max \begin{cases} 0 \\ V(i-1, j) - d \\ V(i, j-1) - d \\ V(i-1, j-1) + s(x_i, y_j) \end{cases}$

Power of ZERO: there is only this change from the original recurrence of a Global Alignment - since there is only one "free ride" arrow entering into every vertex

25

The Smith-Waterman Algorithm

Termination:

1. If we want the best local alignment…

$$V_{OPT} = \max_{i,j} V(i, j)$$

2. If we want all local alignments scoring > t

For all i, j find V(i, j) > t, and trace back

26

Outline: Scoring Alignments

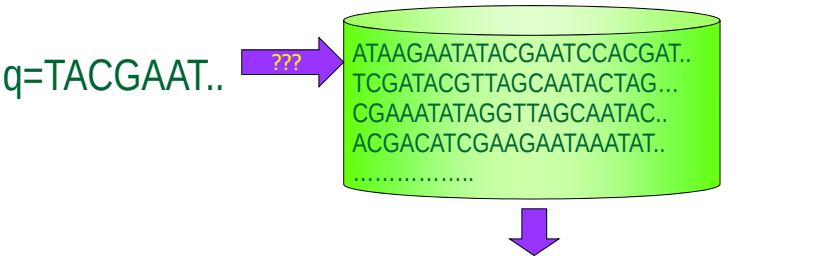
- Scoring alignments
  - Probabilistic meaning
  - Scoring matrices
    - PAM: scoring based on evolutionary statistics
    - BLOSUM: tuning to evolutionary conservation
  - Gaps revisited
- Local vs global alignment
- Database search
  - FASTA
  - BLAST

27

Database Search

The problems:

- Dynamic programming: prohibitively complex
- Exact matching: prohibitively mismatch-sensitive



28

State of Biological Databases

Sequenced Genomes:

Human	3×10^9
Mouse	2.7×10^9
Rat	2.6×10^9
Fugu fish	3.3×10^8
Tetraodon	3×10^8
Mosquito	2.8×10^8
Drosophila	1.2×10^8
Worm	1.0×10^8
2 sea squirts ×	1.6×10^8
Rice	1.0×10^9
Arabidopsis	1.2×10^8

Yeast	1.2×10^7
	× 12 different strains
Neurospora	4×10^7
	14 more fungi within next year

~250 bacteria/viruses

Current rate of sequencing:

4 big labs × 3 $\times10^9$ bp /year/lab
10s small labs

29

State of Biological Databases

- Number of genes

  Vertebrate: ~30,000
  Insects: ~14,000
  Worm: ~17,000
  Fungi: ~6,000-10,000

  Small organisms: 100s-1,000s

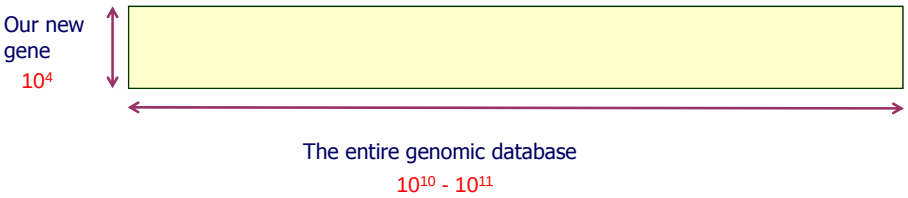
- Each known or predicted gene has an associated protein sequence

- >1,000,000 known / predicted protein sequences

30

Some Useful Applications of Alignments

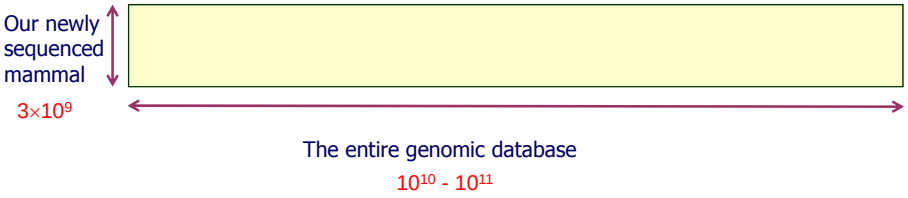
- Given a newly discovered gene,
  - Does it occur in other species?
  - How fast does it evolve?
- Assume we try Smith-Waterman:



31

Some Useful Applications of Alignments

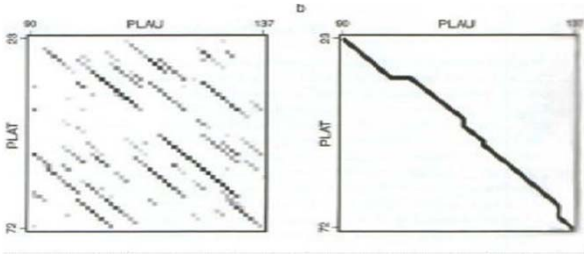
- Given a newly sequenced organism,
  - Which subregions align with other organisms?
    - Potential genes
    - Other biological characteristics
- Assume we try Smith-Waterman:



32

Reconsider DP Geometry

- Diagonal matching segments: Basis for alignment
- Alignment: Connecting matching diagonals
  - With mismatched diagonals or horizontal/vertical gaps



- Score: Additive contributions of diagonals and connectors
  - Connectors may reduce the score
  - Focus: high score diagonals, positive score connectors

33

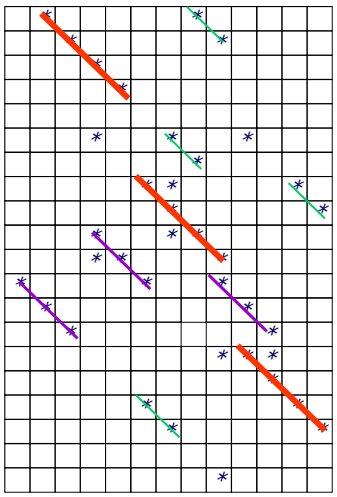
Dot Matrix Heuristics

Rule 1:
Find high-scoring diagonals

- Search small diagonal segments
- Extend to max diagonal matches
- Connect diagonals to max score

Rule 2:
Focus on meaningful alignments

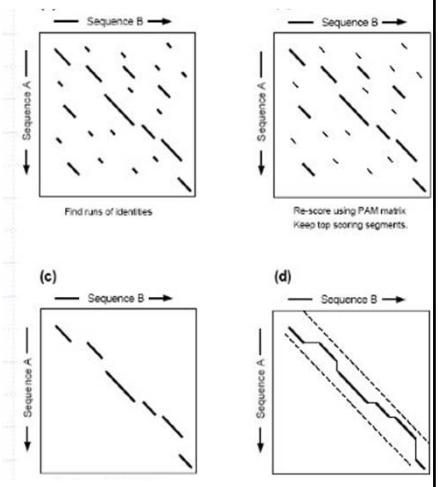
- Filter out low-scoring diagonals



34

FASTA

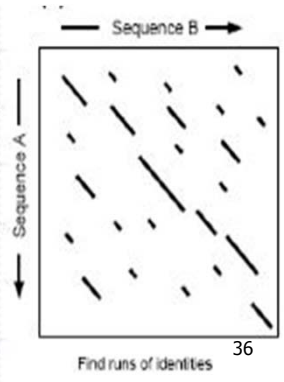
- Key idea (Pearson & Lipman 88):
  - Find short diagonals by indexing the DB
  - Extend these to high scoring diagonals
  - Use DP to connect them
- A 4 steps process



35

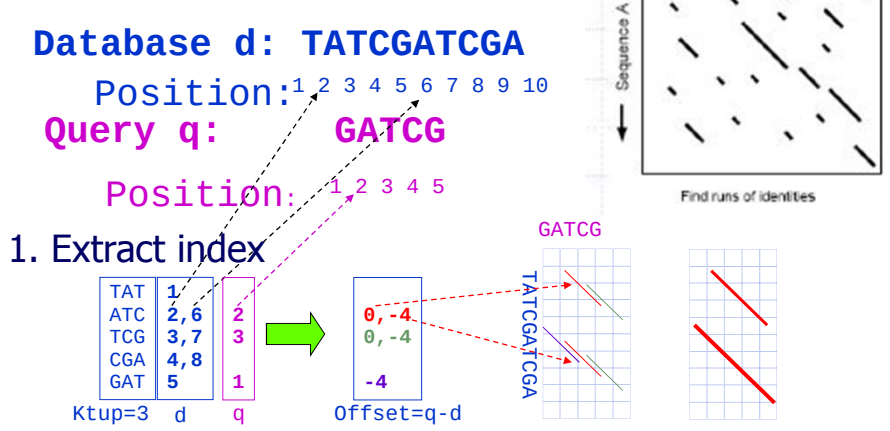
Step (a): Find diagonal matches by indexing

- Key idea: k-mers index of of the DB
- Preprocess:
  - Scan database to index words of size k (k-mers) [k=1..5] (
- Query:
  - Scan query to index k-mers
  - Compare hashes to find all diagonal matches of length k
  - Merge short diagonals into maximal diagonal matches



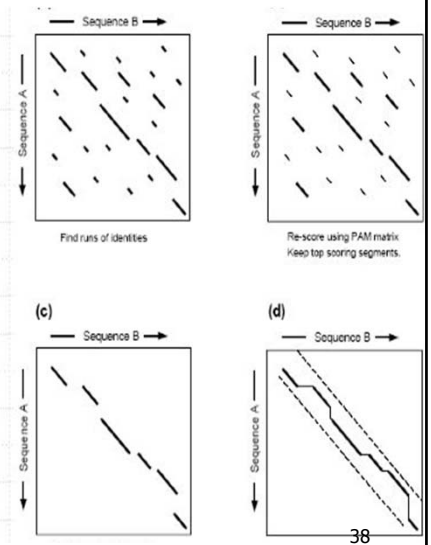
36

Find Diagonal Matches by Indexing Example:



FASTA Steps (b-d): Optimize score

1. Filter low-score diagonals
2. Extend diagonals to max score; keep high-scoring segments
3. Use DP in a narrow band around the high scoring segments



38

BLAST:
Basic Local Alignment Search Tool

- Altschul & Karlin [1990]; a family of algorithms
  - BLAST, WU-BLAST, BlastZ, MegaBLAST, BLAT
- Idea: find matches with significant score statistics
  - Find maximal segment pairs (MSP):
    segments with significant score

39

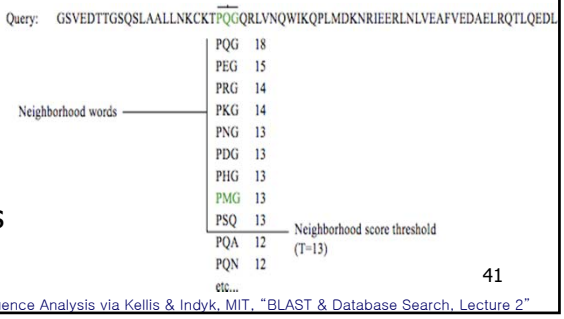
BLAST Algorithm

- Step 1: index DB for words of size W (W-mers);
  index query sequence for W-mers with
  score >Threshold
  - W= 3 for protein, 11 for nucleotides
- Step 2: search for matches with high score (HSP=high scoring pairs)
- Step 3: extend hits to maximal score segments
- Step 4: report matches with score above S

40

BLAST Step 1-3: Finding Short High-Scoring Pairs (HSP)

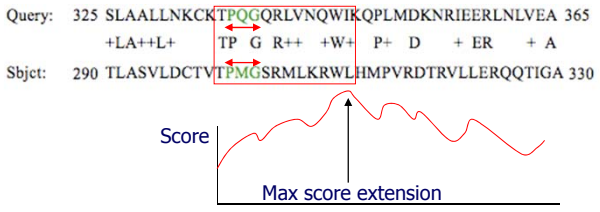
- Create an index of W-mers for database & query
  - For proteins W=3 ⇒ a dictionary of 20^3=8000 words
- Match W-mers that score above a threshold T
  - FASTA searches for exact matches of k-mers
  - BLAST searches for high scoring pairs (HSP)
  - Key idea: exploit fast part of the search to max the score rather than push maximization for later, slower, phases



From A. Baxevanis: "Nucleotide and Protein Sequence Analysis via Kellis & Indyk, MIT, "BLAST & Database Search, Lecture 2"

Blast Steps 3-4: Extending Short HSPs

- The short HSPs are extended to increase the score



- Report above threshold HSPs and their scores