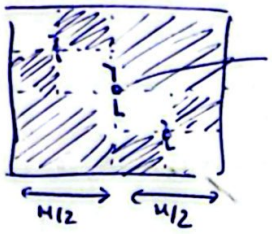
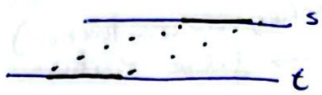
Lecture 3 Database search

Continuation of traceback from L2: Finding optimal path using linear space



Local Alignment

A local alignment of string s and t is an alignment of a substring of s to a substring of t.



Applications:
- small domains may be the only conserved
- search
- rearrangements

Global Algn. (Needleman-Wunch)

Init: $F(0,0) = 0$

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

Ter: ↘

Local Algn. (Smith-Waterman)

Init: $F(0,j) = F(i,0) = 0$

It.
$$F(i,j) = \max \begin{cases} 0 \\ F(i-1,j) - d \\ F(i,j-1) - d \\ F(i-1,j-1) + s(x_i, y_j) \end{cases}$$

Ter: anywhere

Semi global

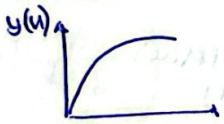
- No penalization for terminations

Init: $F(0,j) = 0$
$F(i,0)$

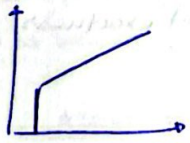
It. $\max \begin{cases} F(i-1,j) - d \\ F(i,j-1) - d \\ F(i-1,j-1) + s(\end{cases}$

Ter: ↓ →

- **Gap penalty functions**
  - ~~Generalized~~ Linear gap penalty: pay same for each gap.
  - Quadratic: encodes the length of the gap ($O(n)$) and pays less as it grows!



  - Affine gap: start gap (price = p) ne (price = q) $w(k) = p + q^* k$



Need another matrix to see whether you are in a gap or not.

  - Lengths (mod 3) gap penalty for protein-coding regions
    ↳ prefer losing 1aa than displacing the reading frame

A A A G A A T T C A
A - A - A - T - C A

or

A A A - - - T C A is more likely

Linear-time string matching

- Looking for exact matches (no gaps)
- Karp-Rabin algorithm (probabilistic linear time)
  - → interpret string numerically

Given $a, b \in \mathbb{R}$:
$a \bmod b$ is the remainder of $\frac{a}{b}$

T: 2 3 5 9 0 2 | 3 1 4 1 5 | 2 6 7 3 9 9

$y_1 = 23,590$ ↓ $y_2 = 35,902$

$y_7 = 31,415$

$x == y_7$

P: 31415 → $x = 31,415$

compute x
for i in [1...n]:
  compute y_i ← not costant, linear!
  if $x == y_i$:
    print "matched S[i]"

we need to compute y_i based on y_{i-1}

- The n^c are too big → take (reduce) to make computation costant. (hashing: one way function)
- Taking the mod → collisions → false positives.
- So when I have a hit → go and check!

Computing t_{s+1} from t_s in constant time

old digit ↓ new digit

|3|1|4|1|5|2|

$y_1 = 31,415$ → $y_2 = 14152$

$14152 = (31,415 - 3 * 10,000) * 10 + 2$

$14152 \bmod 13 = 8$

10 = 1 left shift

The BLAST algorithm and inexact matching

Search algorithm
Assumes common ancestry
Finds alignment
Evo interpretation → $\min(Cost(\pi))$
Query must be very fast! → most seq. will be unrelated → then analized.

1. reject idper <= 90 why bother?
2. Speed in preprocessing (offline) Once query arrives → fast!
3. Extent-based matching index 10-mers
   ↳ only $1/4^{10}$ will match
   ↳ 1 in a million

- Hashing
- Neighborhood search → find words regardless of exactness

Overview

1. Query into overlapping words of length W
2. Neighborhood words for each word until threshold T
3. Look in table when neighbor words occur: seeds S
4. Extend seeds S until score < X

W=3

Query: . . . . PQG . . . .

Neighbor: PQG 16, PEG 15, ..., PHG 13

DATABASE

Q: PQG
G: PAG

extend bidirectionally!

BLAST works because → Pigeonhole principle  at least one hole with two pigeons.

1) Filter out low complexity regions (NNN...) or very common sub sequences (uninformative)
2) Two-hit BLAST → two smaller w-mers more likely than one larger
3) Non consecutive k-mers (seeds)

RG**KW → R**K**, RG**KW

↳ Random projections → n dimensions → n−k dimensions

Probabilistic foundations of seq. alignment

Varying scores / penalties for matches / mismatches

Nucleotide space:

	A	G	T	C
A	1	-½	-1	-1
G	-½	1	-1	-1
T	-1	-1	1	-½
C	-1	-1	-½	1

Protein space: BLOSUM matrix of bin scores

- The score should represent:

P(homology) likelihood ratio { H_1: reader alignment → chance; H_2: common ancestry

$$\left.\begin{matrix} P(x,y|H_1) \\ P(x,y|H_2) \end{matrix}\right\} P(\text{homology}) = \frac{P(x,y|H_2)}{P(x,y|H_1)}$$

$$\text{Score} = \log(P(\text{homology}))$$

☆ additive metrices are exactly that:
- take logs for each aa and add.

- How can we decide U or R?
→ odds ratio

$$\frac{Pr(x,y|R)}{Pr(x,y|U)} = \frac{\prod_i p_{x_i y_i}}{\prod_i q_{x_i} \prod_i q_{y_i}} = \frac{\prod p_{x_i y_i}}{\prod q_{x_i} q_{y_i}}$$

$$Pr(x,y|U) = \prod_{i=1}^{n} q_{x_i} \prod_{i=1}^{n} q_{y_i}$$

- uncorrelated

$$Pr(x,y|R) = \prod_{i=1}^{n} p_{x_i y_i}$$

- recorded

$$\log\left(\frac{P(x,y|R)}{P(x,y|U)}\right) = \sum_i \log\left(\frac{p_{x_i y_i}}{q_{x_i} q_{y_i}}\right) = S$$

Bayes' Theorem

$$P(R|x,y) = \frac{P(x,y|R)\,P(R)}{P(x,y|R)\cdot P(R) + P(x,y|U)\cdot P(U)}$$

↳ $= \sigma(S')$ $\quad \sigma(x) = \frac{e^x}{1+e^x}$; $S' = S + \log\frac{P(R)}{P(U)}$

Deterministic linear-time exact string matching

Input: P: pattern, T: text → analyze int. redundancies → as soon as mismatch → shift

$O \sim O(n) \rightarrow O(n^2)$

eg.

P = a b a

T = b a a b a c a b a b a d

* Make bigger shifts!