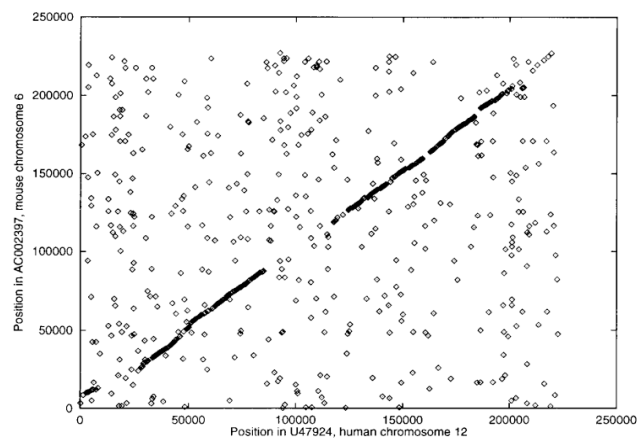


Potentially fun link

- <http://baba.sourceforge.net/>

Example Human-Mouse Dot Plot



Simple dotplots

- Lets construct simple dot plots for the following strings:
 - s = xxixxxixixxgngab
 - t = bagngxxixixxxixx
- To make a dotpot:
 - Fill in a box/pixel if $x_i = x_j$, leave it empty if $x_i \neq y_j$
- See anything interesting?

Less simple dotplots

- We will construct dot plots for the same strings as before:
 - s = xxixxxixixxgngab
 - t = bagngxxixixxxixx
- To make the dotplot
 - Fill in a box/pixel at i,j if $x_i x_{i+1} x_{i+2} = x_j x_{j+1} x_{j+2}$
 - leave it empty otherwise
- See anything different?

Faster implementation

Let:

A = 0 (00) C = 1 (01) G = 2 (10) T = 3 (11)

Strings can be converted based on the binary string they represent

String Binary Integer

AAA = 000000 = 0

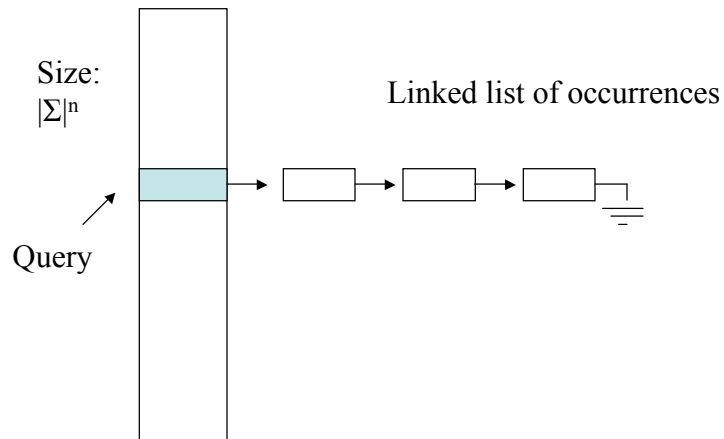
ATA = 001100 = 12

AAC = 000001 = 1

Look-up tables

- Strings of size k over Σ can be represented by an integer index i , $0 \leq i \leq |\Sigma|^k - 1$.
- DNA is composed of four characters.
 - $\Sigma = \{A, G, C, T\}$ $|\Sigma| = 4$
- We can preprocess a database into a lookup table to locate all occurrences of a query index.
 - Linear time and linear space

Search using an index



Indexing discussion

- A database can be preprocessed in linear time to allow locating all instances of a short string.
- Major limitation is it limits searching to fixed length strings.

Applications of indexing

- Seeds for searching sequence databases
 - BLAST
- Pair generation for fragment assembly
 - CAP3 sequence assembly program

Exclusion methods

- Assume s must match t with at most k total errors.
- Split s into $\text{floor}(n/(k+1))$ pieces
- Because one such piece must match t , we use fast approaches (read: linear time) to find these regions
- Running time is $O(m)$ on average for most data

Applications for later classes

- Comparing different genome assemblies
- Locating genome duplications and conserved segments
- Gene finding through comparative genomics
- Analyzing pathogenic bacteria against their harmless close relatives

Affine gap penalties

- To date, we have considered constant gap penalties.
- However, nature doesn't necessarily work this way. For example which do you think is more likely?

ATA--GC

ATATTGC

ATAG-GC

AT-GTGC

Gap scoring

- Ideally, we would like a gap of length l to be penalized by $-(a + bl)$
- a is called the gap open penalty
- b is the gap extension penalty

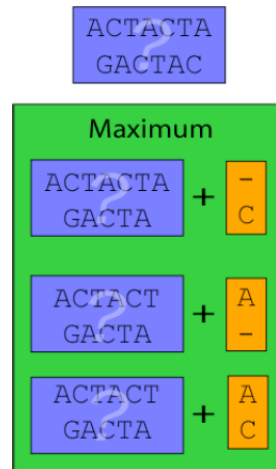
Runtime

- Implementing an affine gap penalty in the way we understand requires $O(n)$ work per cell to work.
- Given we have $O(n^2)$ cells, this is an $O(n^3)$ algorithm. Can we do better?

Flashback:

We can solve this recursively based on looking at three smaller problems

$$T[i,j] = \max \begin{cases} T[i-1,j-1] + \text{score}(s[i],t[j]) \\ T[i-1,j] + g \\ T[i,j-1] + g \end{cases}$$



Solution

- We will use 4 tables instead of 1:
 - V - stores best alignment between $s[1..i]$ and $t[1..j]$
 - G - stores best alignment between $s[1..i]$, $t[1..j]$, i.e., $s[i]$ aligned to $t[j]$
 - E - best alignment between $s[1..i]$, $t[1..j]$ ending with a gap in s
 - F - best alignment between $s[1..i]$, $t[1..j]$, ending with a gap in t
- As before, best global alignment is $V[m,n]$ if $|s|$ is m and $|t| = n$

Updated recurrences

- $V[i,j] = \max\{E[i,j], F[i,j], G[i,j]\}$
- $G[i,j] = V[i-1,j-1] + \text{score}(s[i],t[j])$
- $E[i,j] = \max\{E[i,j-1], V[i,j-1] - \text{gap_open}\} - \text{gap_extend}$
- $F[i,j] = \max\{F[i-1, j], V[i-1, j] - \text{gap_open}\} - \text{gap_extend}$

Gusfield's notation

Table G

- $G[i,j] = V[i-1,j-1] + \text{score}(s[i],t[j])$
 - V is the best one from somewhere
 - We match $\text{score}(s[i],t[j])$ to diagonal value
 - By design, G is the best alignment that ends on a match

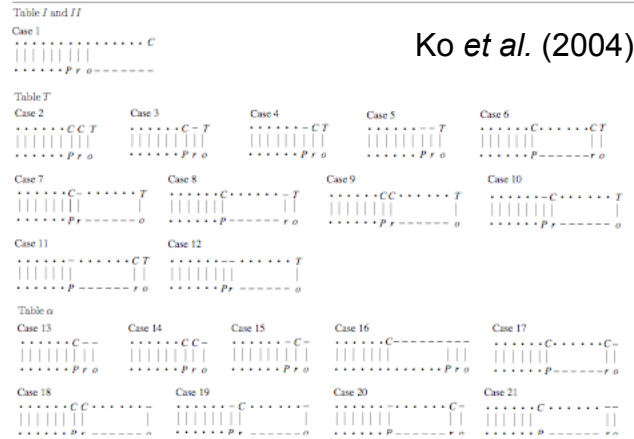
Table E

- $E[i,j] = \max\{E[i,j-1], V[i,j-1] - \text{gap_open}\} - \text{gap_extend}$
 - No matter what, we are extending a gap
 - Two options
 - Add a gap to an existing gap in s (thus use of E in part 1)
 - Open a new gap in s
 - This gives us the best alignment ending in a gap in s

Table F

- $F[i,j] = \max\{F[i-1, j], V[i-1, j] - \text{gap_open}\} - \text{gap_extend}$
 - No matter what, we are extending a gap here too
 - Two options
 - Add a gap to an existing gap in t (thus use of F in part 1)
 - Open a new gap in t
 - This gives us the best alignment ending in a gap in t per the original idea

WhaT? Only four tables?



Ko et al. (2004)

Figure 2. An example for each case considered by our formulation. Case 1 corresponds to table I and table II. Cases 2 through 12 correspond to table T, and the remaining correspond to table α . The alignment of an amino acid to each of the three characters in the nucleotide sequence is denoted by '|'. Pro denotes the amino acid Proline.

WhaT? Four tables?

- It is possible to derive a recurrence that uses only two tables, denoted M and I per Durbin's terminology:
 - M = match
 - I = insertion
- This approach is only guaranteed to find the optimal alignment when lowest mismatch score is $\geq -2 \text{ gap_extend}$

One half the recurrences

<p>Case 2: $\alpha^i[i-3, j-1] + \sigma(a_{i-2}a_{i-1}a_i, b_j)$ $T^i[i-3, j-1] + \sigma(a_{i-2}a_{i-1}a_i, b_j)$ $I^i[i-3, j-1] + \sigma(a_{i-2}a_{i-1}a_i, b_j)$ $II^i[i-3, j-1] + \sigma(a_{i-2}a_{i-1}a_i, b_j)$</p>	<p>Case 6: $M_A^i[i] + \sigma(Aa_{i-1}a_i, b_j)$ \dots $M_T^i[i] + \sigma(Ta_{i-1}a_i, b_j)$ $IM_A^i[i] + \sigma(Aa_{i-1}a_i, b_j)$ \dots $IM_T^i[i] + \sigma(Ta_{i-1}a_i, b_j)$</p>	<p>Case 9: $M_{AA}^i[i] + \sigma(AAa_i, b_j)$ \dots $M_{TT}^i[i] + \sigma(TTa_i, b_j)$ $IM_{AA}^i[i] + \sigma(AAa_i, b_j)$ \dots $IM_{TT}^i[i] + \sigma(TTa_i, b_j)$</p>
<p>Case 3: $\alpha^i[i-2, j-1] + \sigma(a_{i-1} - a_i, b_j) - q - r$ $T^i[i-2, j-1] + \sigma(a_{i-1} - a_i, b_j) - q - r$ $I^i[i-2, j-1] + \sigma(a_{i-1} - a_i, b_j) - q - r$ $II^i[i-2, j-1] + \sigma(a_{i-1} - a_i, b_j) - q - r$</p>	<p>Cases 7, Case 8: $M_A^i[i+1] + \sigma(A - a_i, b_j) - q - r$ \dots $M_T^i[i+1] + \sigma(T - a_i, b_j) - q - r$ $IM_A^i[i+1] + \sigma(A - a_i, b_j) - q - r$ \dots $IM_T^i[i+1] + \sigma(T - a_i, b_j) - q - r$</p>	<p>Case 10: $M_{-A}^i[i] + \sigma(-Aa_i, b_j)$ \dots $M_{-T}^i[i] + \sigma(-Ta_i, b_j)$ $IM_{-A}^i[i] + \sigma(-Aa_i, b_j)$ \dots $IM_{-T}^i[i] + \sigma(-Ta_i, b_j)$</p>
<p>Case 4: $\alpha^i[i-2, j-1] + \sigma(-a_{i-1}a_i, b_j) - r$ $T^i[i-2, j-1] + \sigma(-a_{i-1}a_i, b_j) - q - r$ $I^i[i-2, j-1] + \sigma(-a_{i-1}a_i, b_j) - q - r$ $II^i[i-2, j-1] + \sigma(-a_{i-1}a_i, b_j) - q - r$</p>	<p>Case 11: $M_A^i[i] + \sigma(-a_{i-1}a_i, b_j) - q - r$ $IM_A^i[i] + \sigma(-a_{i-1}a_i, b_j) - q - r$</p>	<p>Case 12: $M_{-}^i[i+1] + \sigma(- - a_i, b_j) - q - 2r$ $IM_{-}^i[i+1] + \sigma(- - a_i, b_j) - q - 2r$</p>
<p>Case 5: $\alpha^i[i-1, j-1] + \sigma(- a_i, b_j) - 2r$ $T^i[i-1, j-1] + \sigma(- a_i, b_j) - q - 2r$ $I^i[i-1, j-1] + \sigma(- a_i, b_j) - q - 2r$ $II^i[i-1, j-1] + \sigma(- a_i, b_j) - q - 2r$</p>		

Figure 3. Recurrences for computing $T^i[i, j]$. The value chosen is the maximum of all the entries shown.

Ko *et al.* (2004)

Next week

- There are two distinct ways to look at sequence alignment from an algorithmic perspective.
- Next week we will use the finite state automation approach, exemplified by hidden markov models (HMMs)