Restriction Mapping

Bioinformatics: Issues and Algorithms

CSE 308-408 • Fall 2007 • Lecture 7
Outline

• Restriction Enzymes
• Gel Electrophoresis
• Partial Digest Problem
• Brute Force Algorithm for Partial Digest Problem
• Branch and Bound Algorithm for Partial Digest Problem
• Double Digest Problem
Recall concept of a restriction enzyme...

A *restriction enzyme* surrounds DNA molecule at specific point, called *restriction site* (sequence GAATTC in this case). It cuts one strand of DNA double helix at one point and second strand at a different, complementary point (between G and A base). The separated pieces have single-stranded *sticky ends*, which allow complementary pieces to combine.

Note that GAATTC $\rightarrow$ CTTAAG $\rightarrow$ GAATTC (i.e., palindrome).

“Molecular scissors”

Molecular Cell Biology, 4th edition
History of restriction enzymes

In 1970, Hamilton Smith discovered that the restriction enzyme \textit{Hind}II cleaves DNA at every occurrence of the sequence GTGCAC or GTTAAC.

It is important to note, however, that this discovery occurred decades before we were able to sequence DNA.
Discovering restriction enzymes

My father has discovered a servant who serves as a pair of scissors. If a foreign king invades a bacterium, this servant can cut him in small fragments, but he does not do any harm to his own king. Clever people use the servant with the scissors to find out the secrets of the kings. For this reason my father received the Nobel Prize for the discovery of the servant with the scissors".

Daniel Nathans’ daughter (from Nobel lecture)

Werner Arber
discovered restriction enzymes

Daniel Nathans
pioneered application of restriction for construction of genetic maps

Hamilton Smith
showed that restriction enzyme cuts DNA in middle of specific sequence

http://www.bioalgorithms.info
Uses of restriction enzymes

- Recombinant DNA technology
- Cloning
- cDNA / genomic library construction
- DNA mapping

http://www.bioalgorithms.info
Restriction maps

- A map showing positions of restriction sites in a DNA sequence.
- If DNA sequence is known, then construction of restriction map is a trivial exercise.
- In early days of molecular biology, DNA sequences were often unknown.
- Biologists had to solve problem of constructing restriction maps **without knowing DNA sequences**.

http://www.bioalgorithms.info
Measuring length of restriction fragments

- Restriction enzymes break DNA into restriction fragments.
- Gel electrophoresis is process for separating DNA by size and measuring sizes of restriction fragments.
- Can separate DNA fragments that differ in length by only 1 nucleotide for fragments up to 500 nucleotides long.

Gel electrophoresis:

- DNA fragments injected into gel positioned in electric field.
- DNA is negatively charged near neutral pH: ribose phosphate backbone of each nucleotide is acidic.
- DNA molecules move towards positive electrode.

http://www.bioalgorithms.info
Gel electrophoresis (continued)

- DNA fragments of different lengths separated based on size.
- Smaller molecules move through gel matrix more readily than larger molecules.
- Gel matrix restricts random diffusion, so molecules of different lengths separate into different bands.

http://www.bioalgorithms.info
Detecting DNA bands

Autoradiography:
- DNA is radioactively labeled
- Gel is laid against sheet of photographic film in the dark, exposing film at positions where DNA is present.

Fluorescence:
- Gel is incubated with solution containing fluorescent dye ethidium.
- Ethidium binds to DNA.
- DNA lights up when gel is exposed to ultraviolet light.

http://www.bioalgorithms.info
Restriction mapping

Using gel electrophoresis, it's possible to measure size of resulting restriction fragments.

Knowing sequence, it is easy to determine size of fragments.

In Perl, it's natural to think of substitution operator:

```
$dna =~ s/GTGCAC/GTG CAC/g;
```

Inserts spaces at all restriction sites

The problem is, we don't know the sequence!
Normally, we might think of a restriction enzyme cutting at all possible restriction sites. This is known as a complete digest:

**What we want**

\[0 \quad 2 \quad 4 \quad 10 \quad 7 \quad 10\]

**What we're given**

\[2 \quad 3 \quad 2 \quad 3 \quad 2 \quad 3\]

**In reality, it's even worse ...**

\[2 \quad 3 \quad 2 \quad 3\]
Complete digest

The situation here is essentially hopeless ...

What we want

Even if we somehow knew there were two different fragments of lengths 2 and 3, there are still a large number of solutions:
Solving a complete digest is essentially impossible – there's too much ambiguity. Fortunately, if we don't let the enzyme digest the DNA completely, we obtain much more information:

We have fragment lengths corresponding to all possible distances between sites. This is known as a **partial digest**.
Some definitions

A **multiset** is a set that allows duplicate elements. E.g.,

$$\{2, 2, 3, 3, 4, 5\}$$

Let $$X = \{x_1 = 0, x_2, ..., x_n\}$$ be a set of $$n$$ points along a line, and define $$\Delta X$$ to be the multiset of all pairwise distances:

$$\Delta X = \{x_j - x_i : 1 \leq i < j \leq n\}$$

For example, if $$X = \{0, 2, 4, 7, 10\}$$, then $$\Delta X$$ is:

$$\{2, 2, 3, 3, 4, 5, 6, 7, 8, 10\}$$

In general, if $$X$$ contains $$n$$ points, how big will $$\Delta X$$ be?

$$\binom{n}{2} = \frac{n(n-1)}{2}$$

“n choose 2”
Some definitions

We can build a table of $\Delta X$ values given $X$ as input.

\[
\begin{array}{c|c|c|c|c|c}
X & |X| & |\Delta X| \\
 \hline
0 & 2 & 4 & 7 & 10 \\
 \hline
0 & 2 & 4 & 7 & 10 \\
 \hline
2 & 2 & 5 & 8 \\
 \hline
4 & 3 & 6 \\
 \hline
7 & 3 \\
 \hline
10 & \\
 \hline
100 & 4,950 \\
 \hline
\end{array}
\]

$\Delta X = \{x_j - x_i : 1 \leq i < j \leq n\}$

= $x_4 - x_2$

What we want

What we're given

The input gets big fast
Formal definition of the problem

The Partial Digest Problem.

Given all pairwise distances between points on a line, reconstruct the positions of those points.

**Input:** The multiset of pairwise distances $L$.

**Output:** A set $X$ of integers such that $\Delta X = L$.

**Note:** If there are $\binom{n}{2}$ integers in $L$, there are $n$ integers in $X$.

Interesting sidenote: this problem is also known as the “Turnpike problem”: given all of the distances between exits on a turnpike, determine the layout of the exits.
Homometric sets

A less trivial example is the following:

\{0, 1, 3, 4, 5, 7, 12, 13, 15\}

\{0, 1, 3, 8, 9, 11, 12, 13, 15\}

Two sets \(A\) and \(B\) are homometric if \(\Delta A = \Delta B\)

Multisets are identical
Homometric sets

As we saw earlier, the complete digest problem is hopeless because there's too much ambiguity. Unfortunately, we can't completely eliminate this here either.

For any integer \( v \) and set \( A \), note \( \Delta A \) is equal to \( \Delta (A \oplus \{v\}) \):

\[
\Delta (A \oplus \{v\}) = \{a + v : a \in A\}
\]

Also, \( \Delta A = \Delta (-A) \).

E.g., if \( A = \{0, 2, 4, 7, 10\} \), then the same partial digest is also produced by:

\[
A \oplus \{100\} = \{100, 102, 104, 107, 110\}
\]

and

\[
-A = \{-10, -7, -4, -2, 0\}
\]

These cases are utterly indistinguishable.
A brute force solution ...

Given all pairwise distances between points on a line, reconstruct the positions of those points.

**Input:** The multiset of pairwise distances \(L\).

**Output:** A set \(X\) of integers such that \(\Delta X = L\).

BruteForcePDP (L, n)

\[ M \leftarrow \text{maximum element in } L \]

for every set of \(n-2\) integers \(0 < x_2 < \ldots < x_{n-1} < M\)

\[ X \leftarrow \{0, x_2, \ldots, x_{n-1}, M\} \]

Form \(\Delta X\) from \(X\)

if \(\Delta X = L\)

return \(L\)

output “No Solution”

How many such sets are there? \(\binom{M-1}{n-2}\)

Time complexity is \(O(M^{n-2})\)
A better brute force solution ...

It seems like overkill to try all possible subsets of integers. Some integers certainly cannot occur in a given problem. If

\[ L = \{2, 2, 3, 3, 4, 5, 6, 7, 8, 10\} \]

Then we shouldn't bother trying \( x_i = 9 \) because it can't occur.

AnotherBruteForcePDP ( L, n )
M ← maximum element in L
for every set of n – 2 integers 0 < \( x_2 < \ldots < x_{n-1} < M \) from L
\[ X ← \{0, x_2, \ldots, x_{n-1}, M\} \]
Form \( \Delta X \) from \( X \)
if \( \Delta X = L \)
return L
output “No Solution”

How many such sets are there? \( \binom{|L|}{n-2} \)

Since \[ |L| = \frac{n(n-1)}{2} \]

Time complexity is \( O(n^{2n-4}) \)
Brute force solutions

Both of these approaches require exponential time, so neither is an effective solution to the general partial digest problem.

For BruteForcePDP, time complexity is $O(M^{n-2})$.
For AnotherBruteForcePDP, time complexity is $O(n^{2n-4})$.

Still, is it possible that one is better than the other?

Consider this example:

$$L = \{2, 998, 1000\}$$

Here, $M = 1000$, so BruteForcePDP takes a very long time. On the other hand, $n = 3$, so AnotherBruteForcePDP is fast. Ratio should be about 100:1.
A more intelligent solution

Developed by Skiena in 1990, this approach works from the largest possible distances down to smaller ones.

(1) First, find largest distance in L. This must determine two outermost points. Delete this distance from L.

(2) Select next largest distance in L, call it $\delta$. One of the points responsible for $\delta$ must be an outermost point.

(3) Evaluate whether remaining distances are consistent with either choice. Repeat recursively, backtracking if necessary.
A more intelligent solution

Consider this example:

\[ L = \{2, 2, 3, 3, 4, 5, 6, 7, 8, 10\} \]

Since \[ |L| = \frac{n(n-1)}{2} = 10 \] we know that \( n \) must be 5.

Hence, solution will consist of five points \( x_1 = 0, x_2, x_3, x_4, x_5 \).

Because 10 is largest distance in \( L \), we set \( x_5 = 10 \), giving us:

\[ X = \{0, 10\} \quad \text{L} = \{2, 2, 3, 3, 4, 5, 6, 7, 8\} \]

Now largest distance is 8. Either \( x_4 = 8 \) or \( x_2 = 2 \). For latter:

\[ X = \{0, 2, 10\} \quad \text{L} = \{2, 3, 3, 4, 5, 6, 7\} \]
A more intelligent solution

\[
X = \{0, 2, 10\} \quad L = \{2, 3, 3, 4, 5, 6, 7\}
\]

Now largest distance is 7. Either \( x_4 = 7 \) or \( x_3 = 3 \).

In latter case, \( x_3 - x_2 = 1 \) should be in set \( L \), but it isn't. Hence, the former case must be the correct one and \( x_4 = 7 \). We must remove following distances from \( L \):

\[
x_5 - x_4 = 3 \quad x_4 - x_2 = 5 \quad x_4 - x_1 = 7
\]

\[
X = \{0, 2, 7, 10\} \quad L = \{2, 3, 4, 6\}
\]

Now 6 is largest distance. Either \( x_3 = 4 \) or \( x_3 = 6 \).

In latter case, \( x_4 - x_3 = 1 \) should be in set \( L \), but it isn't. Hence:

\[
X = \{0, 2, 4, 7, 10\} \quad L = \{\}
\]

Done!
Here's the complete branch-and-bound tree we explored:

```
0  2  3  6  7
0  2  4  7
0  2
{2, 3, 4, 6}  Impossible!
{2, 3, 4}  Impossible!
{2, 2, 3, 3, 4, 5, 6, 7, 8}
{2, 3, 4, 5, 6, 7}
{2, 3, 4, 6}
{2, 3, 4}
{2, 3}
{2}
{ }
```
Branch-and-bound

Or, if you prefer, drawn like a tree:

_redundant due to symmetry_

_impossible_

impossible
Partial Digest Algorithm

Place \((L, X)\)

\textbf{if} \(L\) is empty

\textbf{output} \(X\); \textbf{return}

\(y \leftarrow \text{maximum element in } L\)

\textbf{if} \(\Delta(y, X) \subseteq L\)

Add \(y\) to \(X\); remove lengths \(\Delta(y, X)\) from \(L\)

Place\((L, X)\)

Remove \(y\) from \(X\); add lengths \(\Delta(y, X)\) to \(L\)

\textbf{if} \(\Delta(\text{width} – y, X) \subseteq L\)

Add \(\text{width} – y\) to \(X\); remove lengths \(\Delta(\text{width} – y, X)\) from \(L\)

Place\((L, X)\)

Remove \(y\) from \(X\); add lengths \(\Delta(y, X)\) to \(L\)

\textbf{return}

\textbf{PartialDigest} \((L)\)

width \(\leftarrow \text{maximum element in } L\)

Delete\((\text{width}, L)\)

\(X \leftarrow \{0, \text{width}\}\)

Place\((L, X)\)

Removes \(\text{width}\) from \(L\)

Multiset of distances between point & set

Recursively generates all possible solutions
Let $T(n)$ be maximum number of steps taken by PartialDigest on an input of size $n$.

When only one alternative viable:
\[ T(n) = T(n - 1) + O(n) \]

Time complexity is $O(n^2)$

When both alternatives viable:
\[ T(n) = 2T(n - 1) + O(n) \]

Time complexity is $O(2^n)$

Big difference here!
**Double digest mapping**

Double Digest is yet another experimental method to construct restriction maps.

Use two restriction enzymes to create three full digests:
- One using only first enzyme.
- One using only second enzyme.
- One using both enzymes.

Computationally, Double Digest problem is more complex than Partial Digest problem.

http://www.bioalgorithms.info
Double digest example

Enzyme A yields fragments \{2, 3, 4\}

Enzyme B yields fragments \{1, 3, 5\}

Combined, they yield fragments \{1, 2, 3, 1, 2\}

http://www.bioalgorithms.info
Double digest example

Both solutions equally plausible if don't know $\Delta X$

Without information about $\Delta X$ (i.e., fragment lengths when both Enzymes A and B are used), it is impossible to solve double digest problem, as this diagram illustrates

http://www.bioalgorithms.info
The Double Digest Problem.

Given fragment lengths from application of two enzymes, determine corresponding restriction map.

**Input:** \( \Delta A \) = fragment lengths from digest with enzyme A. 
\( \Delta B \) = fragment lengths from digest with enzyme B. 
\( \Delta X \) = fragment lengths from digest with both A and B.

**Output:** \( A \) = location of cuts in restriction map for enzyme A. 
\( B \) = location of cuts in restriction map for enzyme B.
Even when well-specified, problem still may have multiple solutions:

Solution 1

Solution 2
Wrap-up

Readings for next time:
- IBA Sections 4.4 – 4.9 (regulatory motifs, profiles, etc.).

Remember:
- Come to class having done the readings.
- Check Blackboard regularly for updates.