2003

Physical Mapping of DNA

Dan E. Krane
Wright State University - Main Campus, dan.krane@wright.edu

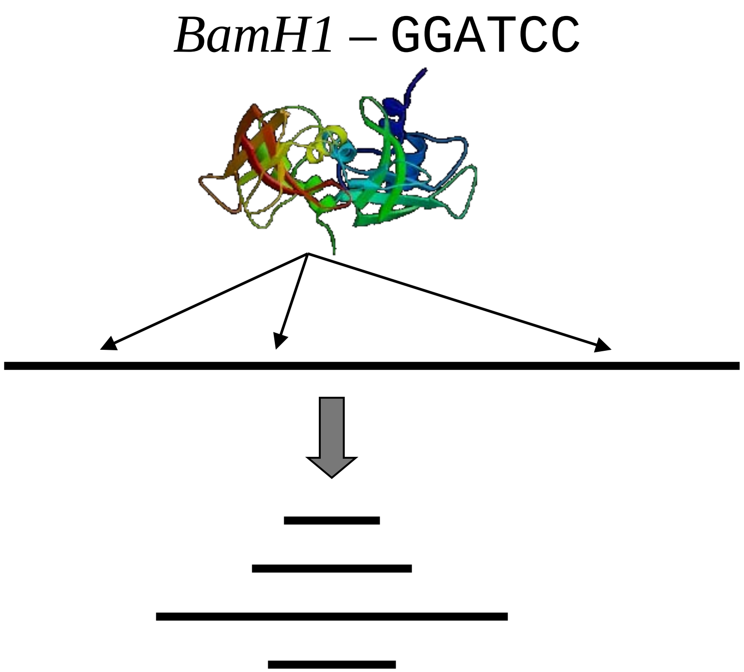
Michael L. Raymer
Wright State University - Main Campus, michael.raymer@wright.edu

Physical Mapping of DNA

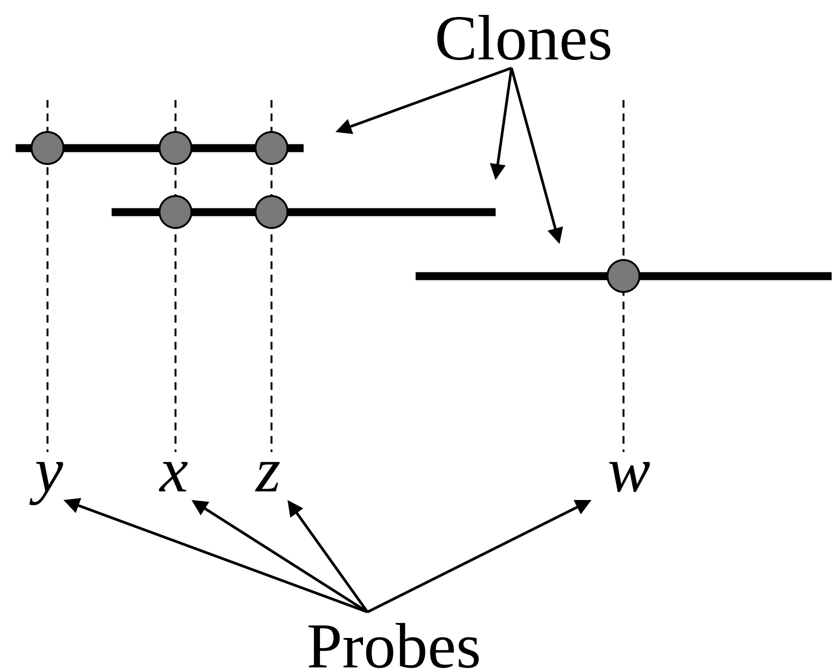
Landmarks on the genome

- Identify the order and/or location of sequence landmarks on the DNA

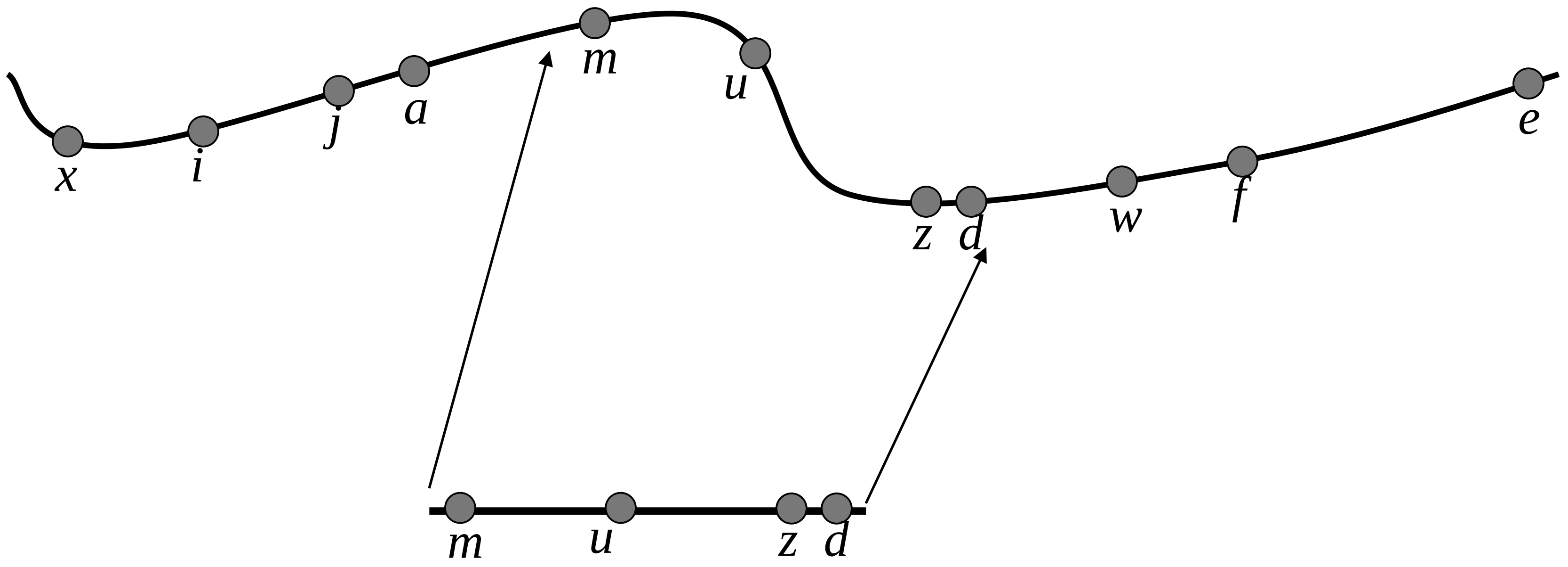
Restriction Enzyme Digests



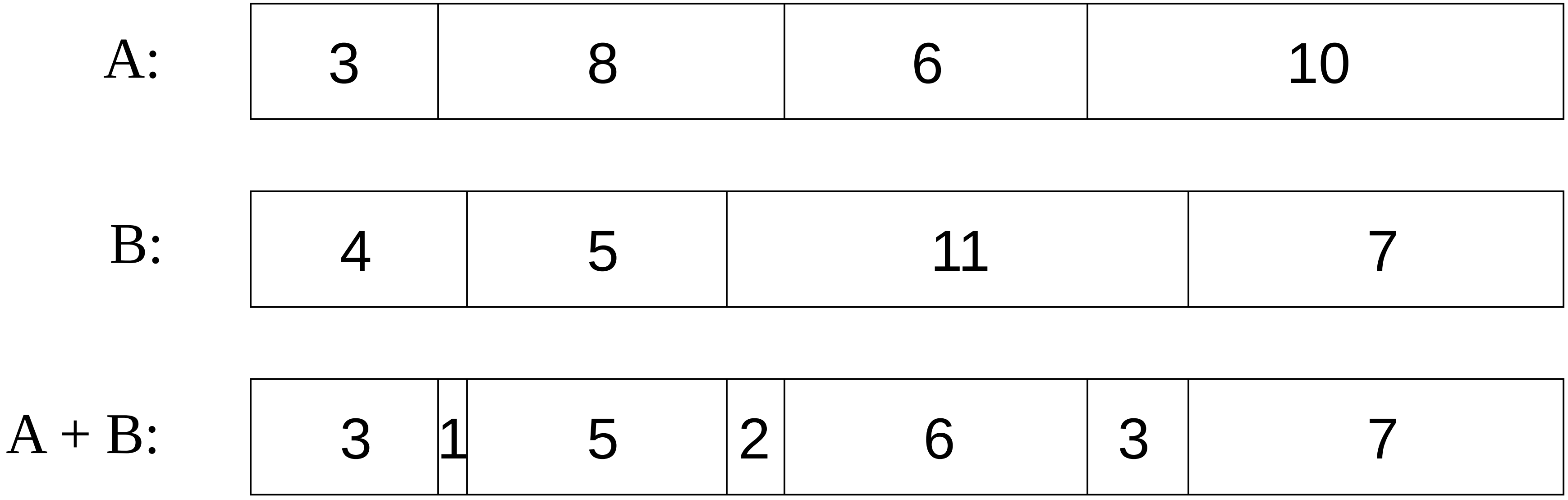
Hybridization Mapping



Producing a map of the genome



Restriction Fragment Mapping

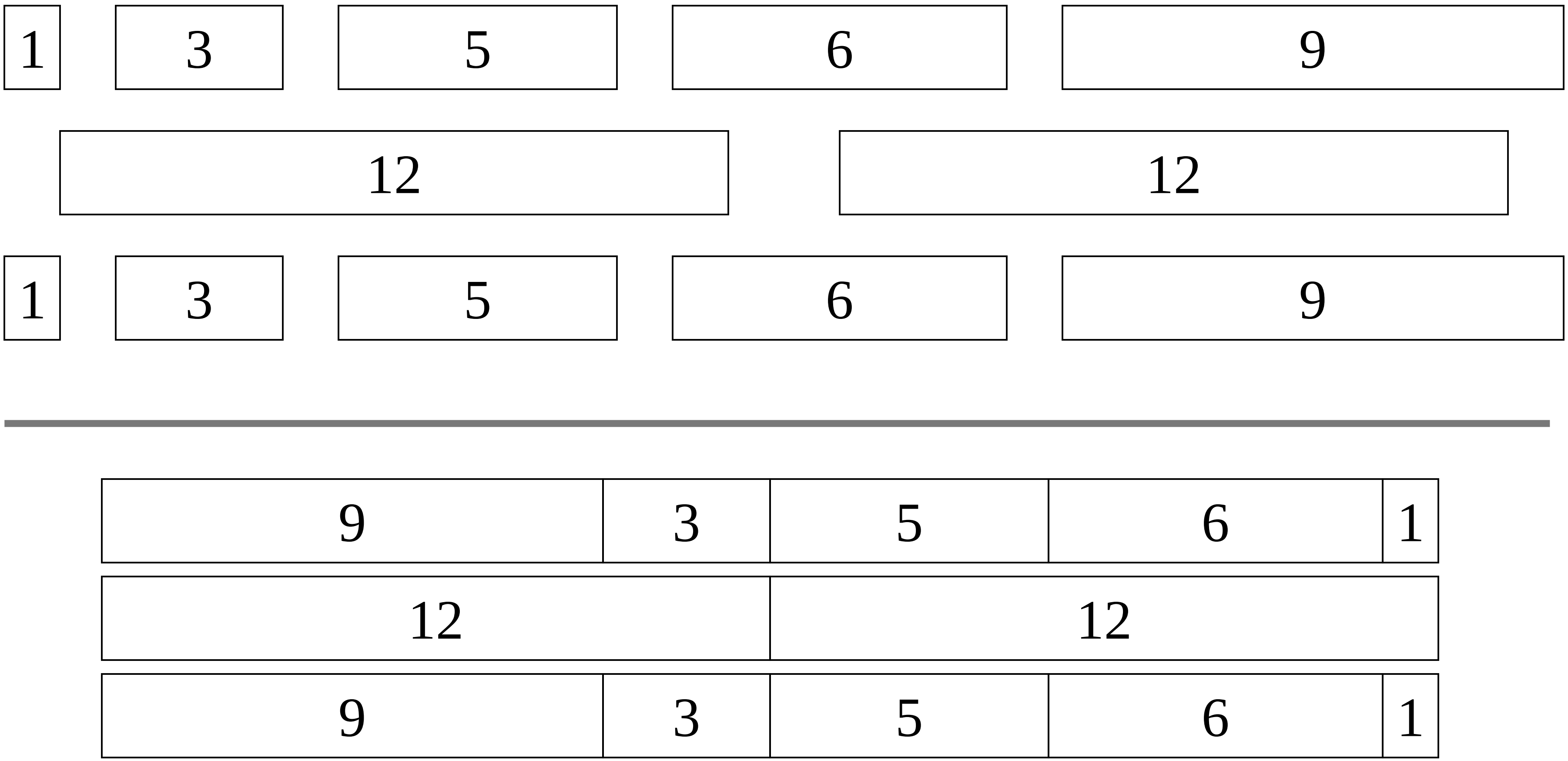


Set Partitioning

- The *Set Partition Problem*
  - Input: $X = \{x_1, x_2, x_3, \dots x_n\}$
  - Output: Partition of X into Y and Z such that $\sum Y = \sum Z$
- This problem is NP Complete
- Suppose we have $X = \{3, 9, 6, 5, 1\}$
  - Can we recast the problem as a double digest problem?

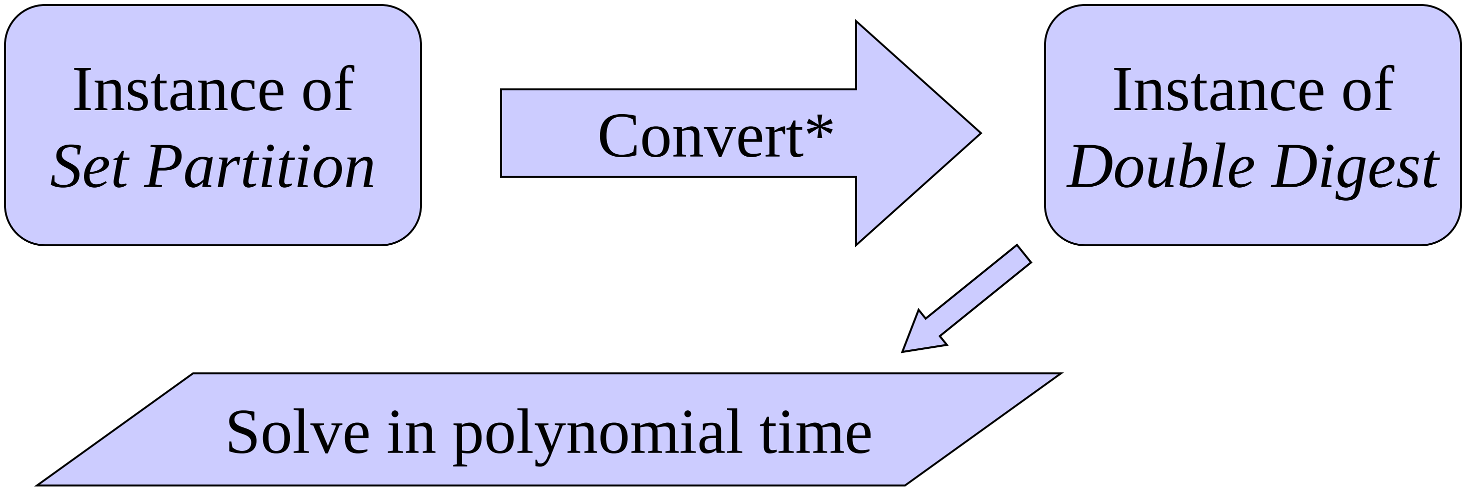
Reduction from Set Partition

- $X = \{3, 9, 6, 5, 1\}$



NP-Completeness

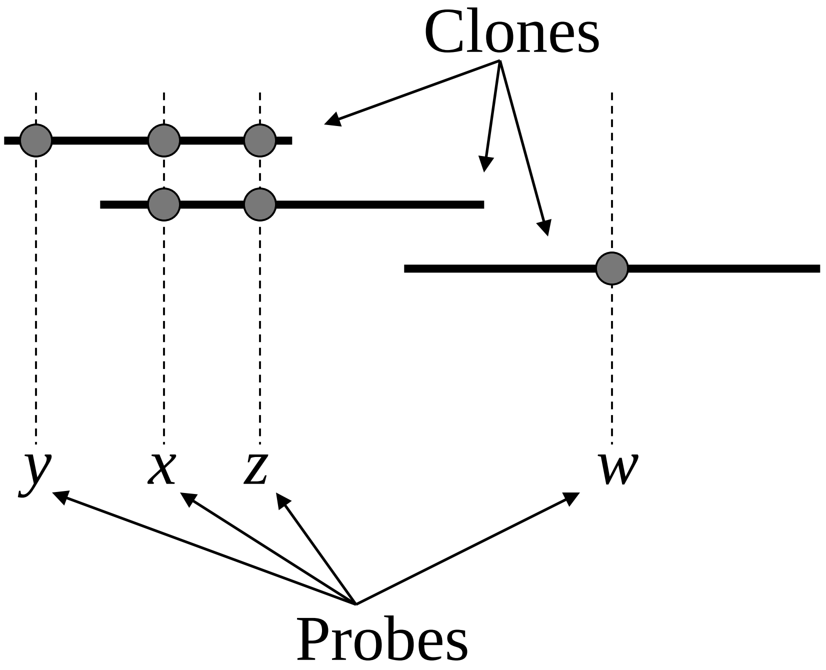
- Suppose we could solve the Double Digest Problem in polynomial time…



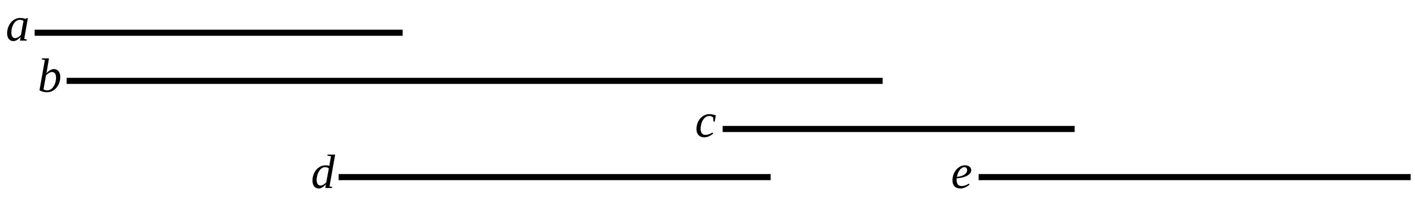
**polynomial time conversion*

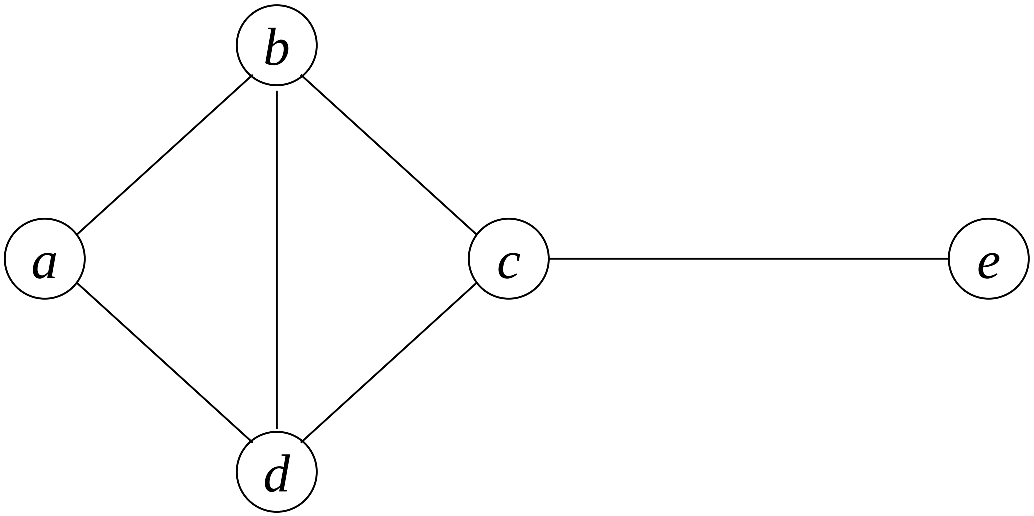
Hybridization Mapping

- The sequence of the clones remains unknown
- The relative order of the probes is identified
- The sequence of the probes is known in advance



Interval graph representation





Becomes a *graph coloring* problem, which is (you guessed it) NP-Complete

Simplifying Assumptions

- Probes are unique
  - Hybridize only once along the target DNA
- There are no errors
- Every probe hybridizes at every possible position on every possible clone

Consecutive Ones Problem (C1P)

Rearrange the columns such that all the ones in every row are together:

Probes

Clones:

	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9
l_1	1	1	0	1	1	0	1	0	1
l_2	0	1	1	1	1	1	1	1	1
l_3	0	1	0	1	1	0	1	0	1
l_4	0	0	1	0	0	0	0	1	0
l_5	0	0	1	0	0	1	0	0	0
l_6	0	0	0	1	0	0	1	0	0
l_7	0	1	0	0	0	0	1	0	0
l_8	0	0	0	1	1	0	0	0	1

An algorithm for *C1P*

1. Separate the rows (clones) into *components*
2. Permute the components
3. Merge the permuted components

	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9
l_1	1	1	0	1	1	0	1	0	1
l_2	0	1	1	1	1	1	1	1	1

$S_1 = \{1, 2, 4, 5, 7, 9\}$
$S_2 = \{2, 3, 4, 5, 6, 7, 8, 9\}$

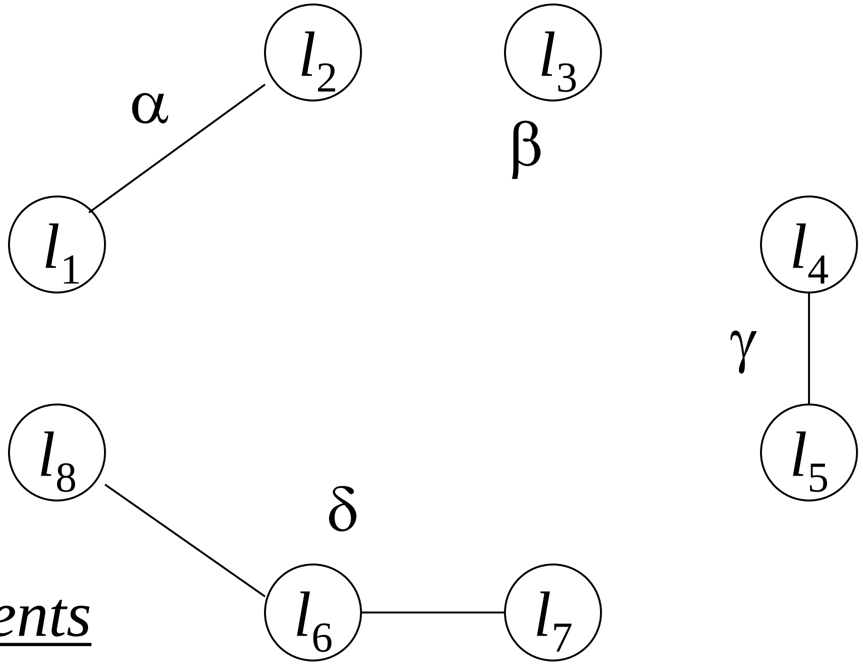
Partitioning clones into components

- Component graph G_c
  - Nodes correspond to clones
  - Connect l_i and l_j iff:

1. $S_i \cap S_j \neq \emptyset$
2. $S_i \not\subset S_j$
3. $S_j \not\subset S_i$

Component Graph

	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9
l_1	1	1	0	1	1	0	1	0	1
l_2	0	1	1	1	1	1	1	1	1
l_3	0	1	0	1	1	0	1	0	1
l_4	0	0	1	0	0	0	0	1	0
l_5	0	0	1	0	0	1	0	0	0
l_6	0	0	0	1	0	0	1	0	0
l_7	0	1	0	0	0	0	1	0	0
l_8	0	0	0	1	1	0	0	0	1

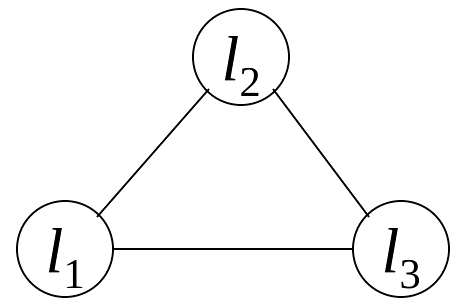


Here, connected components are labeled with greek letters.

Assembling a component

- Consider only row 1 of the following:

	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
l_1	0	1	0	0	0	0	1	1
l_2	0	1	0	0	1	0	1	0
l_3	1	0	0	1	0	0	1	1



- Placing all of the ones together, we can place columns 2, 7, and 8 in any order

{2, 7, 8} {2, 7, 8} {2, 7, 8}

$l_1 \rightarrow$ … 0 1 1 1 0 …

Row 2

- Because of the way we have constructed the component, l_2 will have some columns with 1's where l_1 has 1's, and some where l_2 does not.
- Shall we place the new 1's to the right or left?
- Doesn't matter because the reverse permutation is the same answer.

Adding row 2

	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
l_1	0	1	0	0	0	0	1	1
l_2	0	1	0	0	1	0	1	0
l_3	1	0	0	1	0	0	1	1

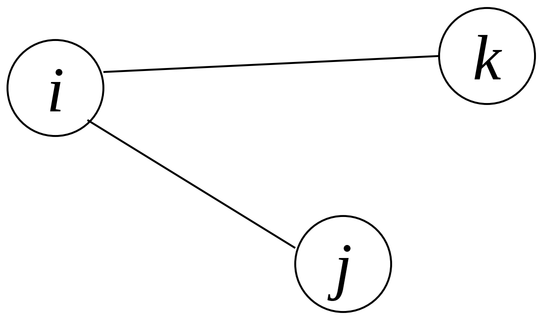
$S_1 = \{2, 7, 8\}$
$S_2 = \{2, 5, 7\}$

			{5}	{2, 7}	{2, 7}	{8}		
$l_1 \rightarrow$	…	0	0	1	1	1	0	…
$l_2 \rightarrow$	…	0	1	1	1	0	0	…

- Placing column 5 to the left partially resolves the {2, 7, 8} columns

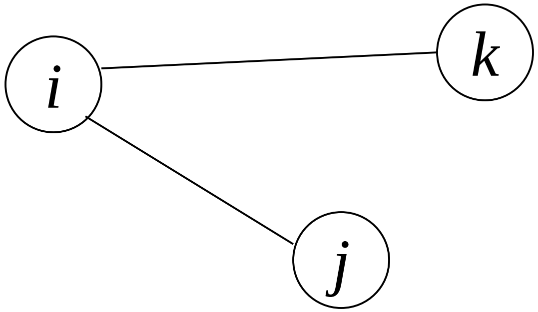
Additional Rows

- Select a new row *k* from the component such that edges (*i*, *j*) and (*i*, *k*) exist for two already added rows *i*, and *k*.



- Look at the relationship between *i* and *k*, and between *i* and *j* to determine if *k* goes on the **same side** or the **opposite side** of *i* as *j*.

Definitions



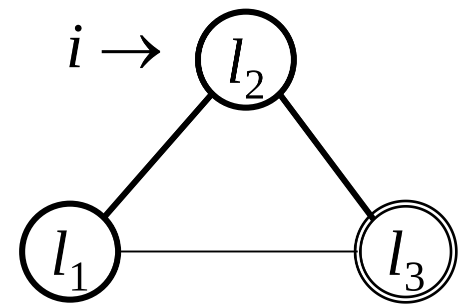
- Let $x \cdot y = \left| S_x \cap S_y \right|$
- Place *i* on the **same** side as *j* if

$$j \cdot k < \min\left(j \cdot i, i \cdot k \right)$$

- Else, place on the **opposite** side

Placing Rows

		c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
$j \rightarrow$	l_1	0	1	0	0	0	0	1	1
$i \rightarrow$	l_2	0	1	0	0	1	0	1	0
$k \rightarrow$	l_3	1	0	0	1	0	0	1	1



Place k on the **same** side as j if

$$l_1 \cdot l_3 < \min(l_1 \cdot l_2, l_2 \cdot l_3)$$

- Or:

$$2 < \min(2,1)$$

So we place l_3 on the **same** side of l_2 as l_1, which is the right.

Placing rows

	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
l_1	0	1	0	0	0	0	1	1
l_2	0	1	0	0	1	0	1	0
l_3	1	0	0	1	0	0	1	1

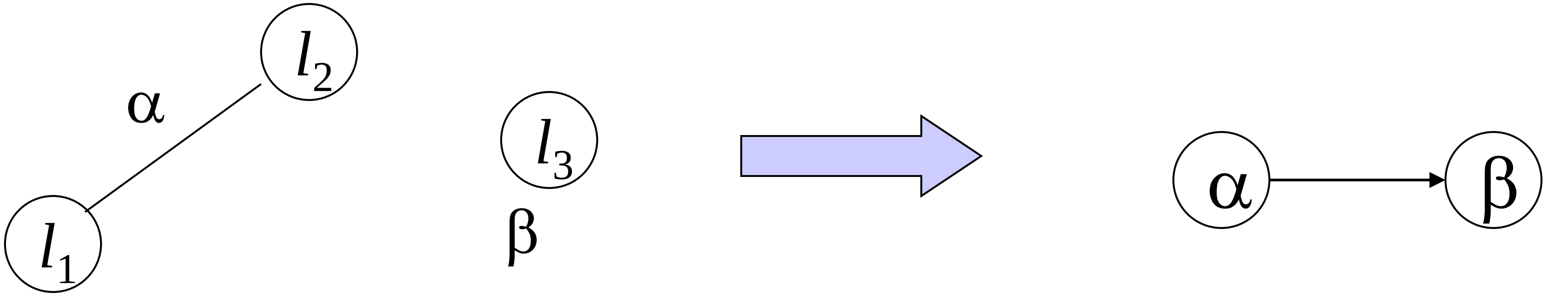
			{5}	{2}	{7}	{8}	{1,4}	{1,4}		
$l_1 \rightarrow$	…	0	0	1	1	1	0	0	0	…
$l_2 \rightarrow$	…	0	1	1	1	0	0	0	0	…
$l_3 \rightarrow$	…	0	0	0	1	1	1	1	0	…

- Repeat for every row in the component

Joining Components

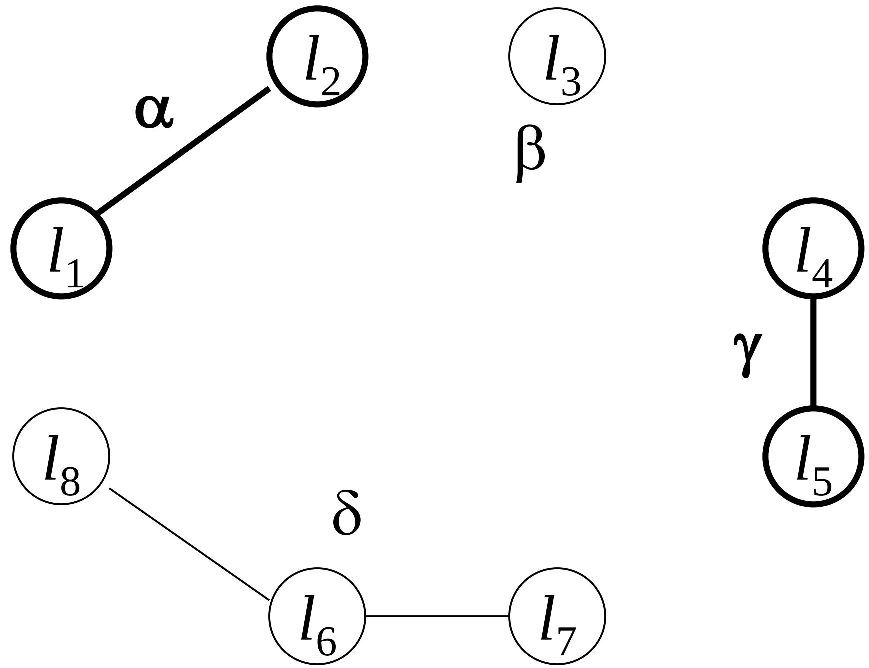
- New graph: G_m – the merge graph *(directed)*
  - Nodes are connected components of G_c
  - Edge (α, β) iff every set in β is a subset of a set in α

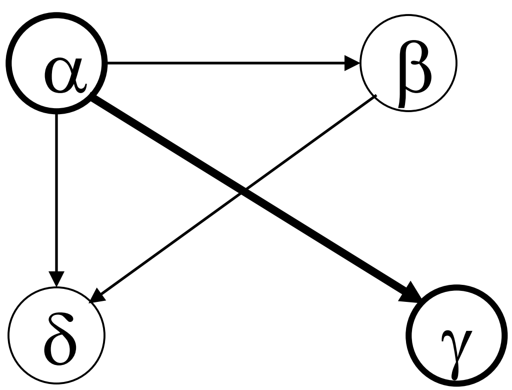
	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9
l_1	1	1	0	1	1	0	1	0	1
l_2	0	1	1	1	1	1	1	1	1
l_3	0	1	0	1	1	0	1	0	1



Constructing G_m

	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9
l_1	1	1	0	1	1	0	1	0	1
l_2	0	1	1	1	1	1	1	1	1
l_3	0	1	0	1	1	0	1	0	1
l_4	0	0	1	0	0	0	0	1	0
l_5	0	0	1	0	0	1	0	0	0
l_6	0	0	0	1	0	0	1	0	0
l_7	0	1	0	0	0	0	1	0	0
l_8	0	0	0	1	1	0	0	0	1

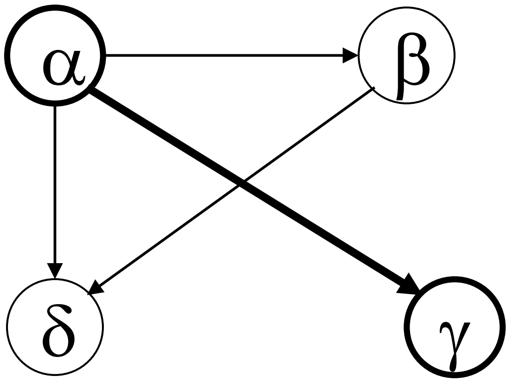




Properties of G_m

- All rows in γ will share the same disjoint/subset relationship with each row of α
  - Different compenents → disjoint *or* subset
  - Same component → shares a 1 column
  - That column matches in a row of α, then subset, else disjoint

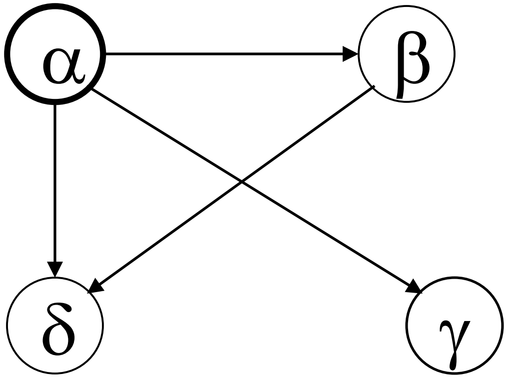
	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9
l_1	1	1	0	1	1	0	1	0	1
l_2	0	1	1	1	1	1	1	1	1
l_3	0	1	0	1	1	0	1	0	1
l_4	0	0	1	0	0	0	0	1	0
l_5	0	0	1	0	0	1	0	0	0
l_6	0	0	0	1	0	0	1	0	0
l_7	0	1	0	0	0	0	1	0	0
l_8	0	0	0	1	1	0	0	0	1



Ordering components

- Vertices without incoming edges: freeze their columns

		{1}	{2, 4, 5, 7, 9}					{3,6,8}			
$l_1 \rightarrow$	…	1	1	1	1	1	1	0	0	0	…
$l_2 \rightarrow$	…	0	1	1	1	1	1	1	1	1	…



- Process the rest in topological order.
  - β is a singleton and a subset

		{1}	{2, 4, 5, 7, 9}					{3,6,8}			
$l_1 \rightarrow$	…	1	1	1	1	1	1	0	0	0	…
$l_2 \rightarrow$	…	0	1	1	1	1	1	1	1	1	…
$l_3 \rightarrow$	…	0	1	1	1	1	1	0	0	0	…

Ordering Components (2)

- Find the leftmost column with a 1
- In the current assembly, find the rows that contain all ones for that column
- Merge the columns

δ