Sequencing: Physical Mapping and Assembly

Chin Lung Lu

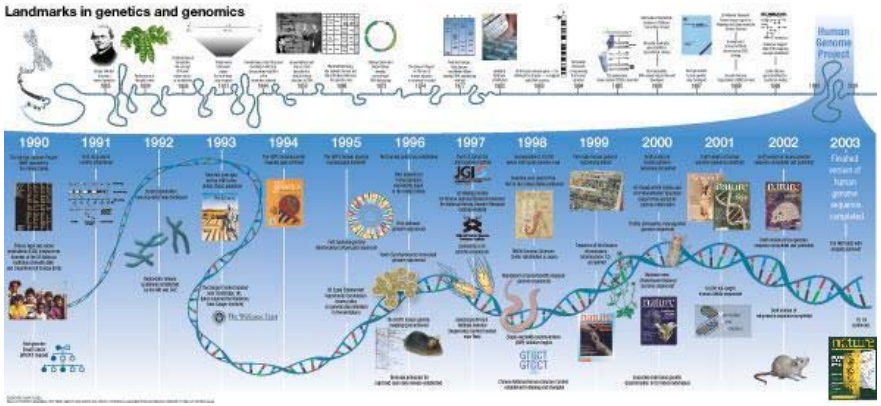
Computational Biology

Analyses and Applications of Sequences

Working draft of human genome

- "Without a doubt, this is the most important, most wondrous map ever produced by humankind", Clinton said at June 26, 2000

Completion of human genome



The International Human Genome Sequencing Consortium today (April 14, 2003) announced the successful completion of the HGP.

DNA Sequencing

- DNA sequencing: the process of determining the order of the individual nucleotide (A, T, G and C) in a DNA molecule
- The sequenced eukaryotic genomes:

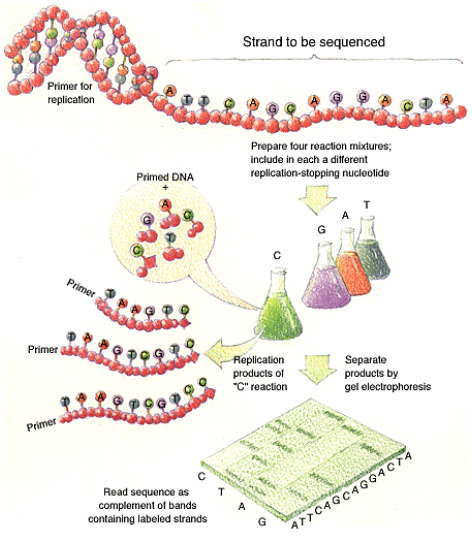
1995	1.8 Mbp	H. Influenzae
1998	100 Mbp	Nematode
1998	120 Mbp	Fruitfly Drosophila
1999	56 Mbp	Human Chromosome 22
2000	50 Mbp	Human Chromosome 21
2001	3 Gbp	Human (draft complete)
2003	3 Gbp	Human (complete)

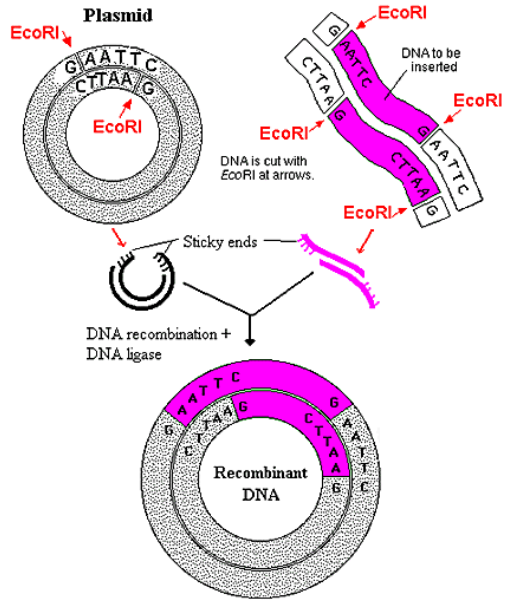
Biochemical sequencing

1. Produce all prefixes of the target DNA such that the last base of each prefix is known (by Sanger chain termination method)
2. Sort the prefixes in order of length (by gel electrophoresis)
3. Determine the target DNA (by scanning the end-bases of the sorted prefixes)

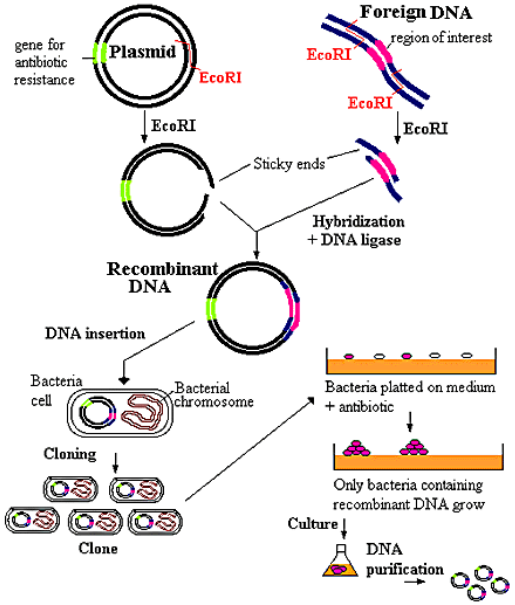
- On average, the target DNA's first 500–1000 bases (read) can be sequenced with today's technology.

Biochemical sequencing





Inserting a DNA Sample into a Plasmid



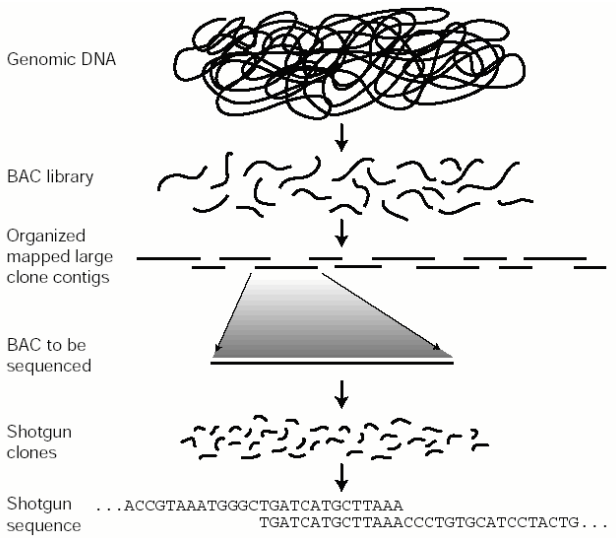
Cloning into a plasmid

Vectors

- Vector is a molecule capable of self-replication in a suitable host.

Cloning Vector	Insert Size
Bacteriophage M13	$\approx$ 1.5 kb
Plasmid	$\approx$ 5 kb
Bacteriophage λ	$\approx$ 25 kb
Cosmid	$\approx$ 40 kb
BAC (Bacterial Artificial Chromosome)	$\approx$ 150 kb
YAC (Yeast Artificial Chromosome)	$\approx$ 500 kb

Hierachical shotgun sequencing



Physical mapping

1. Break many copies of the targeted DNA into smaller pieces by restriction enzymes
   - The order of pieces is lost.
2. Fingerprint the pieces (clones) to distinguish if two clones is overlapping
   - Fragment lengths using restriction enzymes
   - Hybridization information using probes
3. Order the clones along the target DNA according to the overlapping information

Restriction map problem

- Restriction map: the locations of the restriction sites of a given enzyme on the target DNA
- Given a subset $E \subseteq \Delta X$, the restriction mapping problem is to construct X from E.
  - X: a set of points on the line (restriction map)
  - $\Delta X = \{|x_1 - x_2| : x_1 \text{ and } x_2 \in X\}$ (i.e., the multiset of all pairwise distances between two points in X)

1. Partial digest problem: one enzyme ($E = \Delta X$)
2. Double digest problem: two enzymes

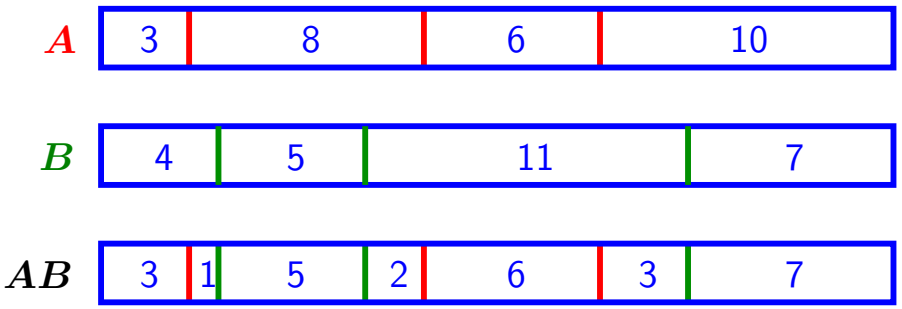
Partial digest porblem

- Given $E = \Delta X$, construct X from E.
- **Example**: $E = \{3, 11, 17, 19, 8, 14, 16, 6, 8, 2\}$

- No polynomial-time algorithm is yet known.
- Experimental errors:
  1. There is uncertainty in length measurement by gel electrophoresis (5% error)
  2. May lost some fragments in the digestion process (gaps occur)

Double digest porblem

- **Example**: $A = \{3, 6, 8, 10\}$, $B = \{4, 5, 7, 11\}$ and $AB = \{1, 2, 3, 3, 5, 6, 7\}$



- NP-compelte, by Goldstein and Waterman [GW87]

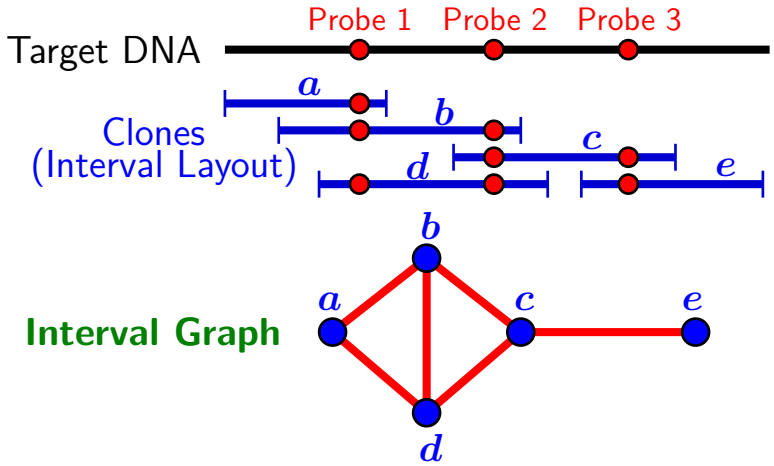
Hybridization data

- Probe p_j (Sequence Tag Site) hybridizes to clone C_i if C_i contains a substring complementary to p_j.
- Hybridization data: an $n \times m$ matrix $\mathcal{M}$, where $\mathcal{M}(i,j) = 1$ if probe p_j hybridizes to clone C_i and $\mathcal{M}(i,j) = 0$ otherwise.
- Fingerprint of a clone: the set of probes hybridizing to the clone
- Two clones perhaps overlap each other if they share part of fingerprints.

	A	B	C	D	E	F	G
1	1		1				
2	1	1			1		1
3		1	1	1	1	1	1
4		1	1	1		1	1
5			1	1		1	1
6	1			1		1	1
7	1	1		1	1		1
8	1	1		1			
9				1			

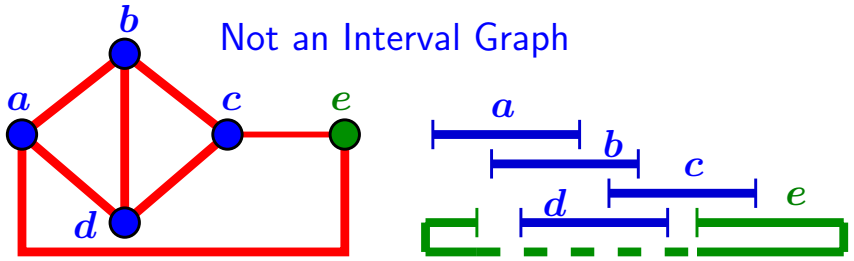
Interval graphs

- In the case of unique probes, every error-free hybridization matrix defines an interval graph.



Recognition of interval graphs

- Given a set of fragments and the information about if two fragments overlap (i.e., an arbitrary graph), determine if its corresponding graph is interval (linearly solvable, by Booth and Leuker [BL76])



Models of hybridization mapping

- Interval Sandwich Problem
  Instance: Two graphs $G_l = (V, E_l)$ and $G_r = (V, E_r)$ with $E_l \subseteq E_r$
  Question: Is there a graph $G' = (V, E')$ such that $E_l \subseteq E' \subseteq E_r$ and G' is an interval graph?
  - NP-complete [GS93]
- Intervalizing Colored Graphs Problem
  Instance: A graph $G = (V, E)$, a coloring $c : V \rightarrow \{1, 2, \ldots, k\}$
  Question: Is there a properly colored supergraph $G' = (V, E')$ of G which is an interval graph?
  - $k \in \{2, 3\}$: P and $k \geq 4$: NP-hard [BdF96]

Consecutive ones property (C1P)

- C1P: the columns can be permuted in such a way that 1s in each row occur in consecutive positions
- Error-free hybridization matrix has the C1P.
- Given an matrix, determine if it has the C1P.

	A	B	C	D	E	F	G
1	1						1
2	1	1			1		1
3			1	1		1	1
4			1	1		1	1
5			1	1		1	1
6	1			1		1	1
7	1	1		1	1		1
8	1	1		1			
9				1			

Column Permutation ⇨

	C	F	D	G	A	B	E
1				1	1		
2				1	1	1	1
3	1	1	1	1			
4	1	1	1	1			
5	1	1	1	1			
6		1	1	1	1		
7			1	1	1	1	1
8			1	1	1	1	
9			1				

The C1P problem

- The C1P problem: Given a binary matrix $\mathcal{M}_{n \times m}$, determine if $\mathcal{M}$ has the C1P
- Fulkerson and Gross proposed an $\mathcal{O}(mn)$ time algorithm for solving the C1P problem [FG65].
  1. Seperate rows into components
  2. Permute the columns of each component
  3. Join components together
- Booth and Leuker gave a linear-time algorithm for solving the C1P problem using the PQ tree [BL76].

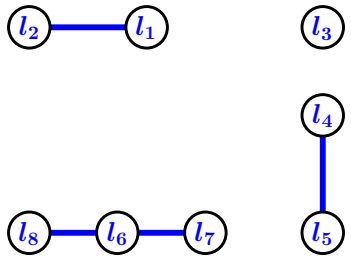
Assumptions and Properties

- **Assume 1**: all rows are different
  (no two clones have the same fingerprint)
- **Assume 2**: no row is all 0s
  (every clone is hybridized by at least a probe)
- Let $S_i = \{\text{column } k : \mathcal{M}_{i,k} = 1\}$ for each row i.
- Given two rows i and j, we have
  1. $S_i \cap S_j = \varnothing$
  2. $S_i \subseteq S_j$ or $S_j \subseteq S_i$
  3. $S_i \cap S_j \neq \varnothing$ and none of them is a subset of the other

1. Seperate rows into components

- Create $G_C = (V_C, E_C)$ from $\mathcal{M}$ such that
  - each vertex of V_C represents a row
  - $(i, j) \in E_C$ if $S_i \cap S_j \neq \varnothing$ and none of them is a subset of the other
- Component: a connected component of G_C

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



2. Permute component's columns

- **Example**: a component with three rows as follows

	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

- **Step 1**:

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

2. Permute component's columns

- **Step 2**:

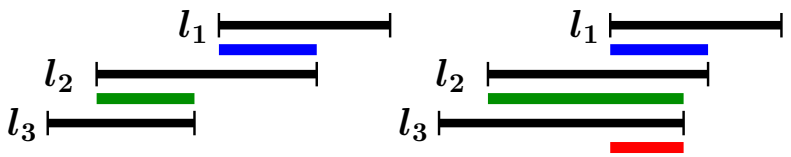
		$\{5\}$	$\{2,7\}$	$\{2,7\}$	$\{8\}$	
$l_1 \rightarrow$	0	0	1	1	1	0
$l_2 \rightarrow$	0	1	1	1	0	0

- **Step 3**:

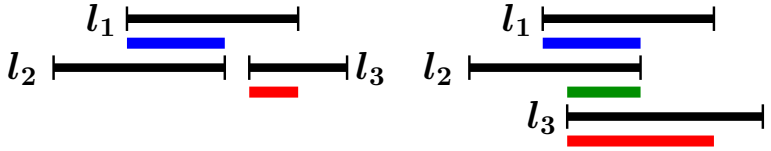
		$\{5\}$	$\{2\}$	$\{7\}$	$\{8\}$	$\{1,4\}$	$\{1,4\}$	
l_1	0	0	1	1	1	0	0	0
l_2	0	1	1	1	0	0	0	0
l_3	0	0	0	1	1	1	1	0

2. Permute component's columns

- Let $x \cdot y = |S_x \cap S_y|$ for any two rows x and y.
- If $l_1 \cdot l_3 < \min\{l_1 \cdot l_2, l_2 \cdot l_3\}$, then l_3 goes in the same direction that l_2 was placed w.r.t. l_1.

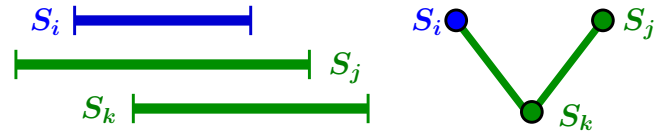


- If $l_1 \cdot l_3 > \min\{l_1 \cdot l_2, l_2 \cdot l_3\}$, then l_3 goes in the opposite direction that l_2 was placed w.r.t. l_1.

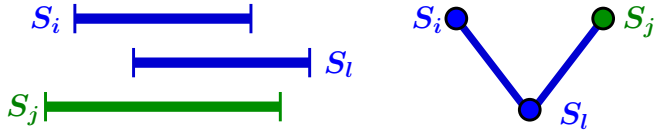


3. Join components together

- Let $S_i \in \beta$ be contained in $S_j \in \alpha$. Then there is no row $k \in \alpha$ such that S_i is not contained in S_k and $S_i \cap S_k \neq \varnothing$ (S_i is contained in some sets of α and is disjoint from the others).

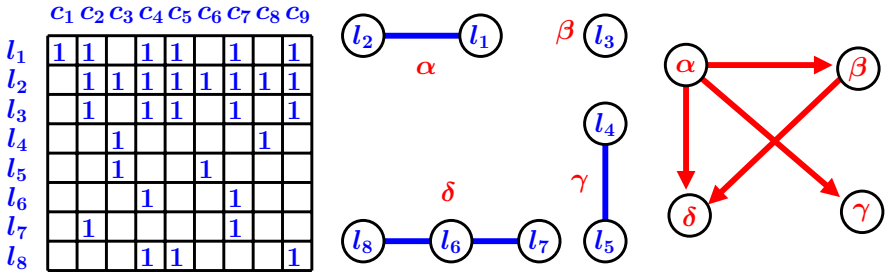


- The following case cannot be happened: $i, l \in \beta$, $S_i \cap S_l \neq \varnothing$, $j \in \alpha$ and $S_i \subseteq S_j$, but $S_l \not\subset S_j$.



3. Join components together

- $\alpha \rightarrow \beta$ if S_i for all $i \in \beta$ are contained in at least one set S_j of α.
- Join components together in topological ordering (ex. $\alpha \rightarrow \beta \rightarrow \delta \rightarrow \gamma$)



3. Join components together

- Compoent α:

	$\{1\}$	$\{2,4,5,7,9\}$	$\{3,6,8\}$
l_1	1	11111	000
l_2	0	11111	111

- Compoent β:

	$\{2,4,5,7,9\}$
l_3	11111

- $\alpha \rightarrow \beta$:

	$\{1\}$	$\{2,4,5,7,9\}$	$\{3,6,8\}$
l_1	1	11111	000
l_2	0	11111	111
l_3	0	11111	000

3. Join components together

- Compoent δ:

	$\{9,5\}$	$\{4\}$	$\{7\}$	$\{2\}$
l_6	00	1	1	0
l_7	00	0	1	1
l_8	11	1	1	0

- $\alpha \rightarrow \delta$:

	$\{1\}$	$\{9,5\}$	$\{4\}$	$\{7\}$	$\{2\}$	$\{3,6,8\}$
l_1	1	11	1	1	1	000
l_2	0	11	1	1	1	111
l_3	0	11	1	1	1	000
l_6	0	00	1	1	0	000
l_7	0	00	0	1	1	000
l_8	0	11	1	0	0	000

3. Join components together

- Compoent γ:

	$\{6\}$	$\{3\}$	$\{8\}$
l_4	0	1	1
l_5	1	1	0

- $\alpha \rightarrow \gamma$:

	$\{1\}$	$\{9,5\}$	$\{4\}$	$\{7\}$	$\{2\}$	$\{6\}$	$\{3\}$	$\{8\}$
l_1	1	11	1	1	1	0	0	0
l_2	0	11	1	1	1	1	1	1
l_3	0	11	1	1	1	0	0	0
l_6	0	00	1	1	0	0	0	0
l_7	0	00	0	1	1	0	0	0
l_8	0	11	1	0	0	0	0	0
l_4	0	00	0	0	0	0	1	1
l_5	0	00	0	0	0	1	1	0

Errors of hybridization data

1. Flase negative: a probe fails to bind to a site where it should (in hybridization process)
   - $11\underline{1}11 \Rightarrow 11\underline{0}11$
2. Flase positive: a probe binds to a site where it should not (in hybridization process)
   - $00\underline{0}00 \Rightarrow 00\underline{1}00$
3. Chimeric clone: a clone consisting of two distinct fragments (joined by an error in cloning process)
   - $\underline{111}0\cdots0\underline{111} \Rightarrow \underline{111111}$

Gaps created by errors

- Gap: a consecutive block of 0s bordered by 1s
  Example: $11100\cdots0111$

1. A chimeric error will create a gap.
   - $\underline{111111} \Rightarrow \underline{111}00\cdots0\underline{111}$ (create a gap)
2. A false positive may will create a gap.
   - $11100\underline{0}00 \Rightarrow 11100\underline{1}00$ (create a gap)
   - $111\underline{0}0000 \Rightarrow 111\underline{1}0000$ (create no gap)
3. A false negative may will create a gap.
   - $11\underline{1}11000 \Rightarrow 11\underline{0}11000$ (create a gap)
   - $1111\underline{1}000 \Rightarrow 1111\underline{0}000$ (create no gap)

Hybridization data with errors

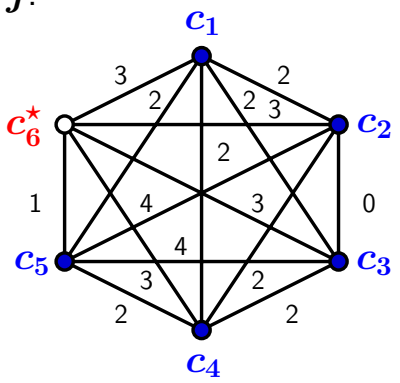
- Minimization of gaps $\approx$ minimization of the blocks of consecutive 1s

1. Given a matrix, find a column permutation such that the total number of blocks of consecutive 1's is minimized.
   - NP-complete [SM97]
2. Given a matrix, find a column permutation such that in each row, there are at most k blocks of consecutive 1's.
   - NP-complete [SM97]

Gap minimization vs. TSP

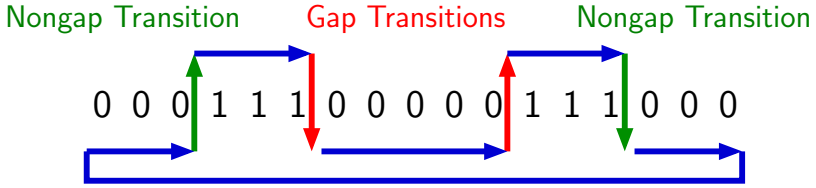
- Reduce the gap minimization problem to the traveling salesperson problem (TSP):
  - Weight of edge (c_i, c_j): the Hamming distance between columns i and j.

	c_1	c_2	c_3	c_4	c_5	$c_6^\star$
l_1	1	1	1	0	0	0
l_2	0	1	1	1	0	0
l_3	1	0	0	1	1	0
l_4	1	1	1	1	0	0



Gap minimization vs. TSP

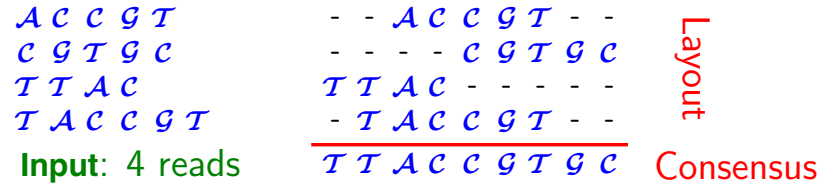
- A gap contribues exactly 2 to the weight of cycle.



- Purpose of extra column $c_6^\star$:
  - Ensure each row has a pair of nongap transitions
  - Avoid consecutive 1s from wrapping around in each row
- cycle weight = number of gap transitions + $2n$ where n is the number of rows

Fragment assembly problem

- Given the reads from a shotgun sequencing, the fragment assembly problem is to infer the target DNA sequence from these given reads.



- The typical phases of sequence assembly:
  1. Overlap: finding potentially overlapping reads
  2. Layout: finding the order of reads along DAN
  3. Consensus: deriving the DNA sequence from the layout

Sequencing errors

1. Base errors: substitutions, insertions, and deletions
   - Substitute $\mathcal{A}$ of the fourth read into $\mathcal{G}$:

```
A C C G T          - - A C C G T - -
C G T G C          - - - - C G T G C
T T A C            T T A C - - - - -
T G C C G T        - T G C C G T - -
                   T T A C C G T G C
Input: 4 reads
```

   - Insert $\mathcal{A}$ into the second read:

```
A C C G T          - - A C C - G T - -
C A G T G C        - - - - C A G T G C
T T A C            T T A C - - - - -
T A C C G T        - T A C C - G T - -
                   T T A C C A G T G C
Input: 4 reads
```


Sequencing errors

1. Base errors: substitutions, insertions, and deletions
   - Delete $\mathcal{C}$ from the fourth read:

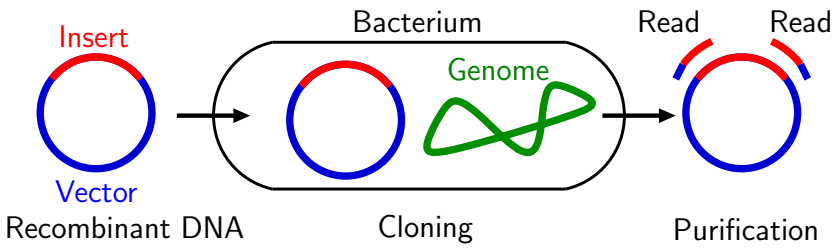
```
A C C G T          - - A C C G T - -
C G T G C          - - - - C G T G C
T T A C            T T A C - - - - -
T A C G T          - T A C - G T - -
                   T T A C C G T G C
Input: 4 reads
```

2. Chimeric fragment:

```
A C C G T          - - A C C G T - -
C G T G C          - - - - C G T G C
T T A C            T T A C - - - - -
T A C C G T        - T A C C G T - -
T T A T G C        T T A C C G T G C
                   T T A - - - T G C
Input: 5 reads
```


Sequencing errors

3. Contamination by host or vector DNA:



- The remedy for chimeras and contamination is to recognize and remove them from the input set before starting assembly process.

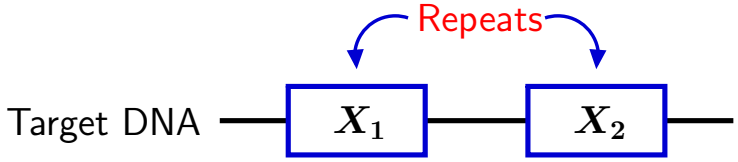
Unknown orientation

- Note that DNA is a double strands. Generally, we don't know which strand a read come from.

```
                   Orientation
C A C G T          →  C A C G T - - - - - - - - -
A C G T            →  - A C G T - - - - - - - - -
A C T A C G        ←  - - C G T A G T - - - - -
G T A C T          ←  - - - - - A G T A C - - -
A C T G A          →  - - - - - - - - - A C T G A
C T G A            →  - - - - - - - - - C T G A
Input: 6 reads        C A C G T A G T A C T G A
                      Consensus
```

Repeated regions

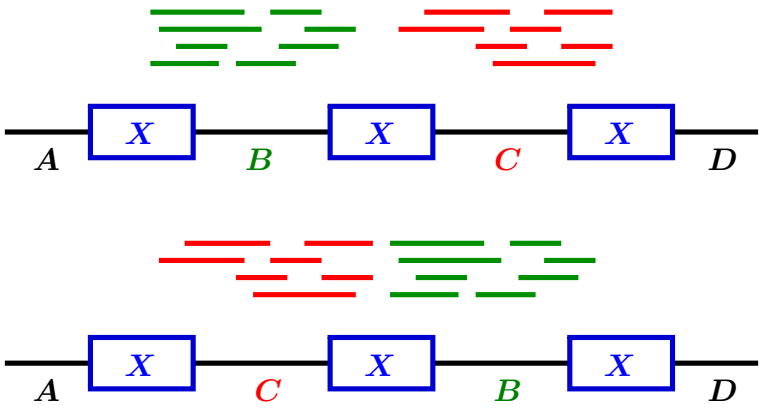
- Repeats: the sequences appear two or more times in the target DNA
  - **Example**: X_1 and X_2 are approximately the same sequences



- Inverted repeats: if X_2 is the reverse complement of X_1

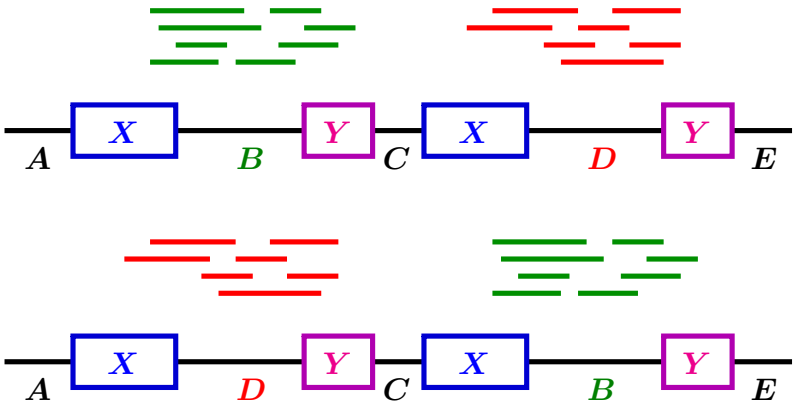
Problems of repeats

- Ambiguous assembly since repeats XXX



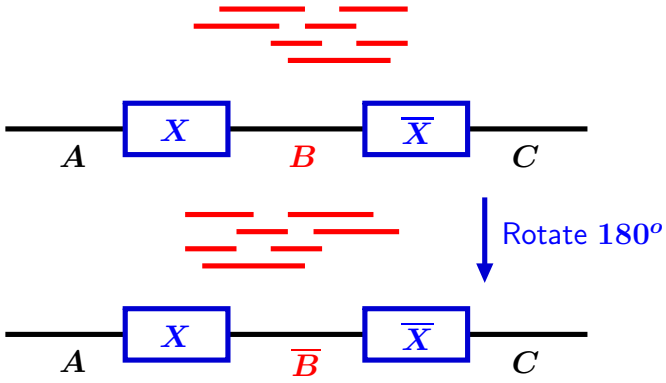
Problems of repeats

- Ambiguous assembly since repeats $XYXY$



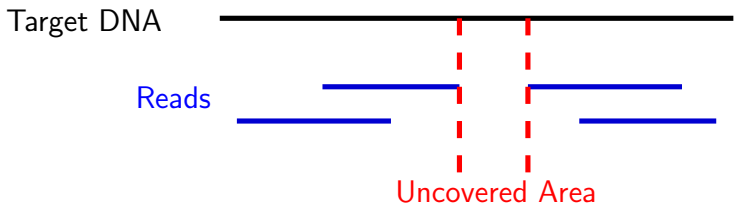
Problems of inverted repeats

- Ambiguous assembly since repeats $X\overline{X}$, where $\overline{X}$ is the reverse complement of X



Lack of coverage

- Mean coverage: the average number of reads covering each position of the target DNA (i.e., $\frac{\text{the total length of all reads}}{\text{the approximate target length}}$)
- Lack of coverage occurs since the sampling of reads is a random process



Shortest common superstring

- **Input**: $\mathcal{F} = \{f_i\}_{i=1}^{n}$ (a collection $\mathcal{F}$ of strings f_i)
- **Output**: a shortest string S such that every $f_i \in \mathcal{F}$ is a substring of S.
- **Example**: if $\mathcal{F} = \{\mathcal{ACT}, \mathcal{CTA}, \mathcal{AGT}\}$, then $S = \mathcal{ACTAGT}$ is a shortest common superstring of $\mathcal{F}$.

S	$\mathcal{A}$	$\mathcal{C}$	$\mathcal{T}$	$\mathcal{A}$	$\mathcal{G}$	$\mathcal{T}$
f_1	$\mathcal{A}$	$\mathcal{C}$	$\mathcal{T}$			
f_2		$\mathcal{C}$	$\mathcal{T}$	$\mathcal{A}$		
f_3				$\mathcal{A}$	$\mathcal{G}$	$\mathcal{T}$

Shortest common superstring

- NP-hard, by Gallant et al. [GMS80]
- MAX SNP-hard, 4-approx. (greedy), 3-approx. (modified greedy), by Blum et al. [BJL$^+$94]
- $2\frac{8}{9}$-approximation, by Teng & Yao [TY93]
- $2\frac{5}{6}$-approximation, by Czumaj et al. [CGPR94]
- $2\frac{3}{4}$-approximation, by Armen & Stein [AS95]
- $2\frac{2}{3}$-approximation, by Armen & Stein [AS96]
- 2.596-approx., by Breslauer et al. [BJJ97]
- $2\frac{1}{2}$-approximation, by Sweedyk [Swe99]

GREEDY algorithm

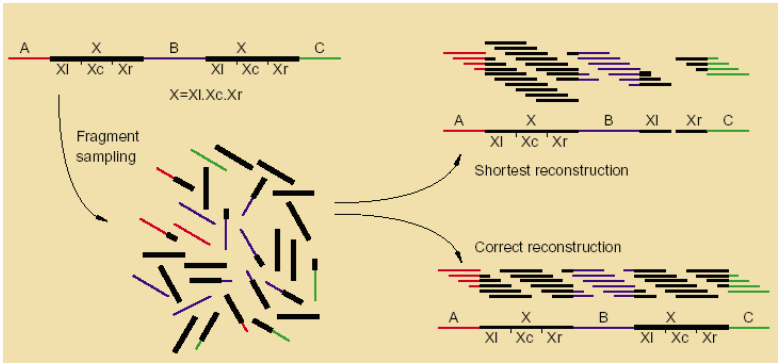
- Repeatedly merge the pair of distinct strings with maximum overlap until only one string remains
- **Example**: $\mathcal{F} = \{$ate, half, lethal, alpha, alfalfa$\}$
  1. $\mathcal{F} = \{$ate, lethal, alpha, halfalfa$\}$
  2. $\mathcal{F} = \{$ate, alpha, lethalfalfa$\}$
  3. $\mathcal{F} = \{$ate, lethalfalfalpha$\}$
  4. $\mathcal{F} = \{$lethalfalfalphate$\}$
- The performance ratio of GREEDY is shown to be 4 by Blum, Jiang, Li, Tromp, Yannakakis [BJL$^+$94].

Conjecture of GREEDY algorithm

- How well does GREEDY approximate a shortest common superstring?
- **Example**: $\mathcal{F} = \{c(ab)^k, (ba)^k, (ab)^k c\}$
  - GREEDY $= c(ab)^k c(ba)^k$
  - OPTIMAL $= ca(ba)^k bc$
  - $\frac{|\text{GREEDY}|}{|\text{OPTIMAL}|} = \frac{4k+2}{2k+4} \leq 2$
- Can you create an example such that GREEDY is more than twice as long as OPTIMAL?
- **Conjecture**: GREEDY is a 2-approx. algorithm

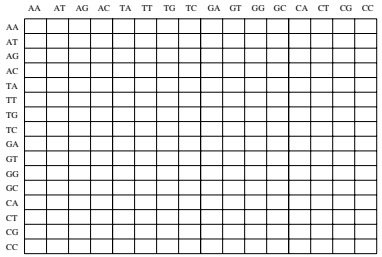
Shortest common superstring

- The shortest answer is not always the correct one.



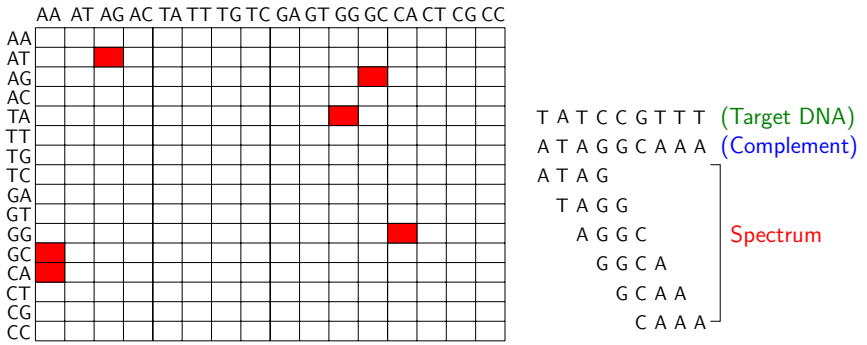
Probe and DNA array (DNA chip)

- Probe: a short single-stranded DNA of 8-30 letters which will hybridise (bind) to a single stranded target DNA if the substring complementary to the probe exists in the target
- DNA array: contains all 4^l probes of length l



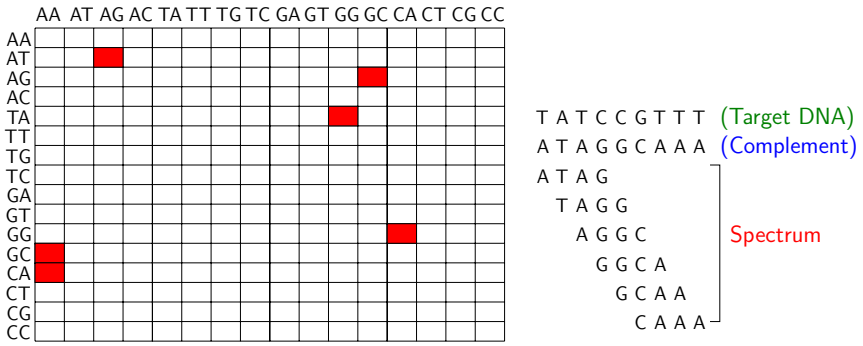
l-Tuples, spectrum

- l-tuples, spectrum: the information about all strings of length l contained in the target DNA
- **Example**: Target DNA = TATCCGTTT
  The complement of target DNA = ATAGGCAAA



SBH: Sequencing By Hybridization

- **Input**: all substrings of length l of a target DNA fragment (spectrum, l-tuple composition)
- **Output**: Reconstruction of the target DNA

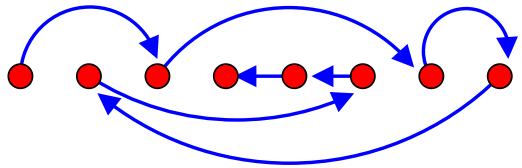


SBH & Hamiltonian path problems

- SBH: given a spectrum S of n l-tuples $s_1, \cdots, s_n$, find the shortest superstring of this spectrum
- $ov(s_i, s_j)$: the length of a maximal suffix of s_i that matches a prefix of s_j
- Overlap graph: a directed graph $H = (V, E)$ with $V = S$ and $E = \{(s_i, s_j) : ov(s_i, s_j) = l - 1\}$

S ATG AGG TGC TCC GTC GGT GCA CAG

H

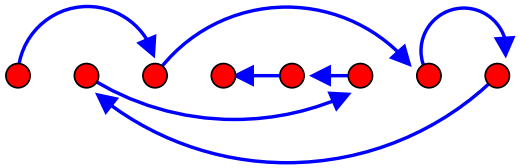


SBH & Hamiltonian path problems

- Hamiltonian path: a path visiting every vertex exactly once which corresponds to a DNA fragment with S

S ATG AGG TGC TCC GTC GGT GCA CAG

H



Complement of targe DNA = ATGCAGGTCC

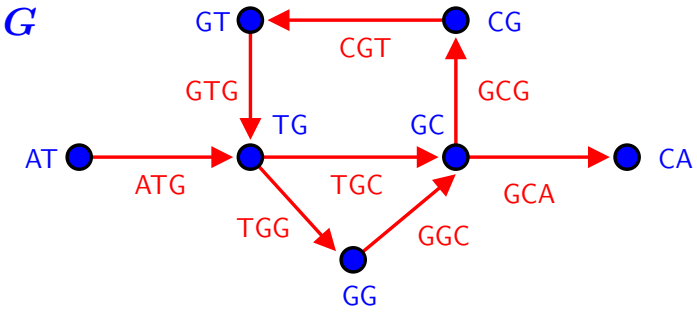
- Hmiltonian path problem is NP-complete [GJ79].

SBH & Eulerian path problems

- Construct a directed graph $G = (V, E)$ such that
  - edges correspond to l-tuples in spectrum S
  - vertices correspond to $l - 1$-tuples

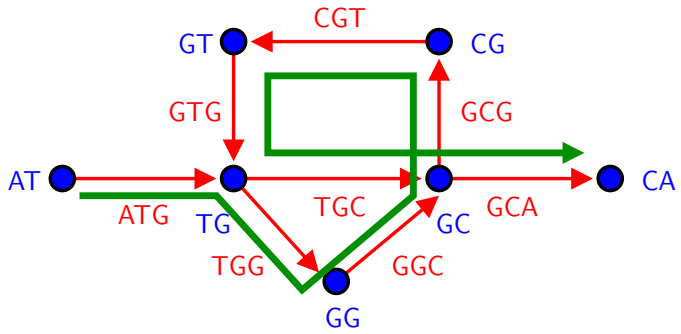
S ATG TGG TGC GTG GGC GCA GCG CGT

G



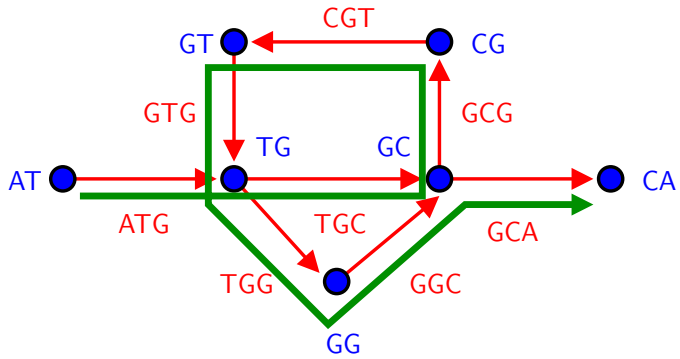
SBH & Eulerian path problems

- Finding a DNA fragment containing all probes from S corresponds to finding an Eulerian path (visiting all edges exactly once) in G.



Complement of target DNA = ATGCGTGGCA

SBH & Eulerian path problems



Complement of target DNA = ATGCGTGGCA

- The Eulerian path of a directed graph can be found in linear time [Fle90].

Eulerian cycle and path

- Eulerian cycle (path): a cycle (path) traversing every directed edge of G exactly once
- Balanced vertex v: the number of edge entering v ($indeg(v)$) equals the number of edges leaving v ($outdeg(v)$)
- Semi-balanced v: $|indeg(v) - outdeg(v)| = 1$
- **Theorem**: A connected graph has an Eulerian cycle if and only if each of its vertices is balanced.
- **Theorem**: A connected graph has an Eulerian path if and only if it contains at most two semi-balanced vertices and all other vertices are balanced.

References

[AS95] C. Armen and C. Stein. Improved length bounds for the shortest superstring problem. In S. G. Akl, F. K. H. A. Dehne, J.-R. Sack, and N. Santoro, editors, *Proc. 4th Workshop on Algorithms and Data Structures*, volume 955 of *Lecture Notes in Computer Science*, pages 494–505, Kingston, Ontario, Canada, 1995. Springer.

[AS96] C. Armen and C. Stein. A $2\frac{2}{3}$-approximation algorithm for the shortest superstring problem. In D. S. Hirschberg and E. W. Myers, editors, *Proc. 7th Annual Symposium on Combinatorial Pattern Matching*, volume 1075 of *Lecture Notes in Computer Science*, pages 87–101, Laguna Beach, California, 1996. Springer.

[BdF96] H. L. Bodlaender and B. de Fluiter. On intervalizing k-colored graphs for DNA physical mapping. *Discrete Applied Mathematics*, 71:55–77, 1996.

[BJJ97] D. Breslauer, T. Jiang, and Z. Jiang. Rotations of periodic strings and short superstring. *Journal of Algorithms*, 24:340–353, 1997.

[BJL+94] A. Blum, T. Jiang, M. Li, J. Tromp, and M. Yannakakis. Linear approximation of shortest superstrings. *Journal of Computer and System Sciences*, 41:630–647, 1994.

[BL76] K. S. Booth and G. S. Leuker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *Journal of Computer and System Sciences*, 13(3):335–379, 1976.

[CGPR94] A. Czumaj, L. Gąsieniec, M. Piotrów, and W. Rytter. Parallel and sequential approximations of shortest superstrings. In Erik M. Schmidt and Sven Skyum, editors, *Proc. 4th Scandinavian Workshop on Algorithm Theory*, volume 824 of *Lecture Notes in Computer Science*, pages 95–106, Aarhus, Denmark, 1994. Springer-Verlag.

[FG65] D. R. Fulkerson and O. A. Gross. Incidence matrices and interval graphs. *Pacific Journal of Mathematics*, 15:835–856, 1965.

[Fle90] H. Fleischner. *Eulerian Graphs and Related Topics*. Elsevier Science Publishers, 1990.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, Freeman, 1979.

[GMS80] J. Gallant, D. Maier, and J. Storer. On finding minimal length superstring. *Journal of Computer and System Sciences*, 20:50–58, 1980.

[GS93] M. C. Golumbic and R. Shamir. Complexity and algorithms for reasoning about time: A graph-theoretic approach. *Journal of the Association for Computing Machinery*, 40(5):1108–1133, 1993.

[GW87] L. Goldstein and M. S. Waterman. Mapping DNA by stochastic relaxation. *Advances in Applied Mathematics*, 8:194–207, 1987.

[SM97] J. Setubal and J. Meidanis. *Introduction to Computational Molecular Biology.* PWS Publishing Company, 1997.

[Swe99] E. Sweedyk. A $\mathbf{2\frac{1}{2}}$-approximation algorithm for shortest superstring. *SIAM Journal on Computing*, 29:954–986, 1999.

[TY93] S.H. Teng and F. Yao. Approximating shortest superstrings. In *Proc. 34th Annual Symposium on*

Foundations of Computer Science, pages 158–165. IEEE Computer Society Press, 1993.