EECS 4425: Introductory Computational Bioinformatics

Fall 2018

Suprakash Datta

datta [at] cse.yorku.ca

Office: CSEB 3043

Phone: 416-736-2100 ext 77875

Course page: http://www.cse.yorku.ca/course/4425

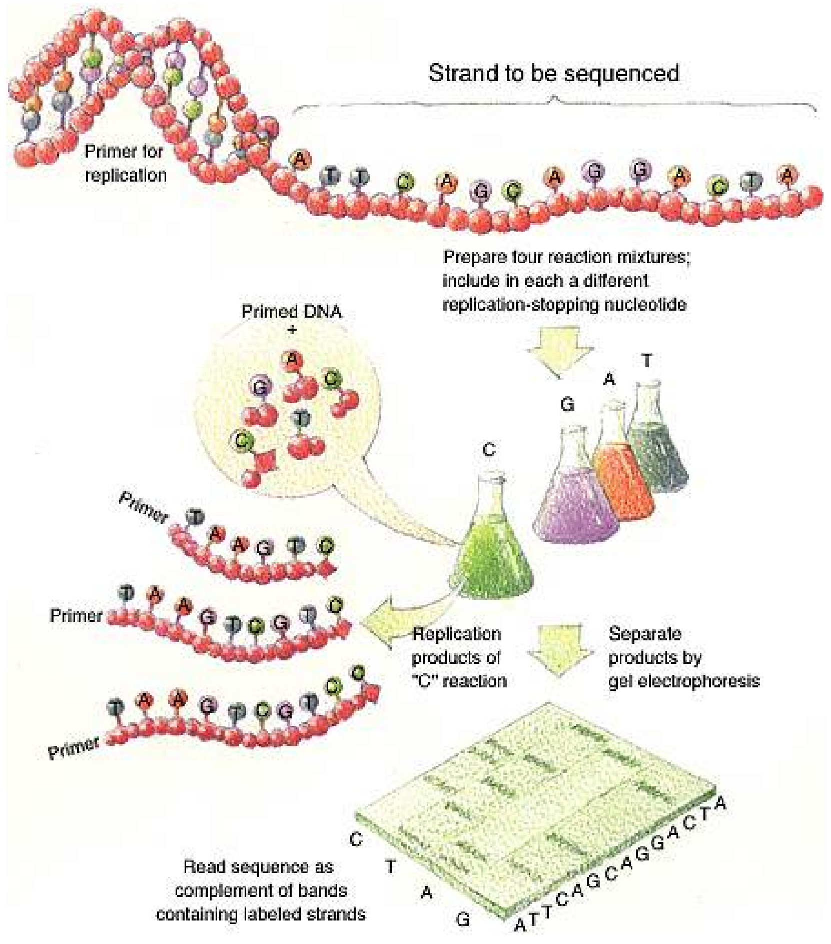
Next

Graph algorithms

Some of the following slides are based on slides from www.bioalgorithms.info

DNA Sequencing

- Shear DNA into millions of small fragments
- Read 500 – 700 nucleotides at a time from the small fragments (Sanger method)


Strand to be sequenced
Primer for replication
A T T C A G G A G G A C T A
Prepare four reaction mixtures; include in each a different replication-stopping nucleotide
Primed DNA +
A G C T C
T A G C
Primer T A G T C
Primer T A A G T C G T C
Primer T A A G T C G T C C
Replication products of "C" reaction
Separate products by gel electrophoresis
Read sequence as complement of bands containing labeled strands
C T A G
ATTCAGCAGGACTA

Fragment Assembly

- **Computational Challenge:** assemble individual short fragments (reads) into a single genomic sequence (“superstring”)

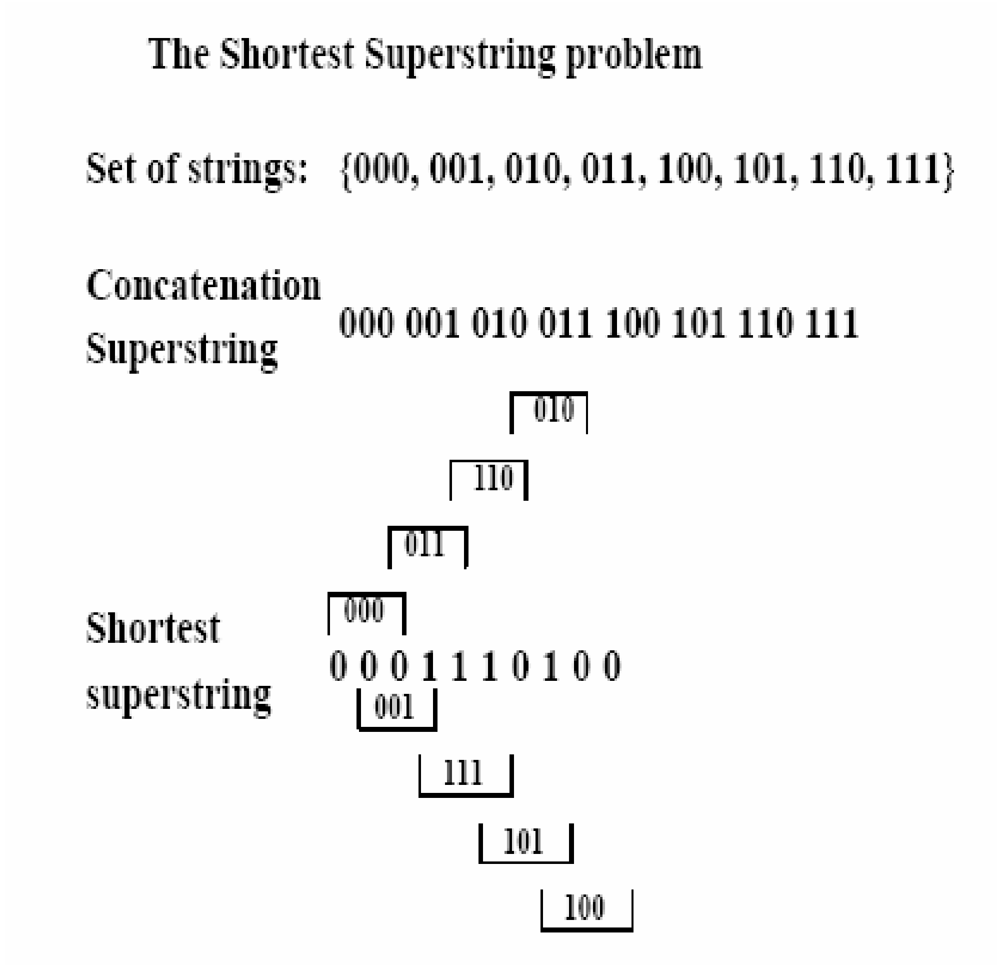
- Until late 1990s the shotgun fragment assembly of human genome was viewed as intractable problem

Shortest Superstring Problem

- Problem: Given a set of strings, find a shortest string that contains all of them
- Input: Strings $s_1, s_2, \ldots, s_n$
- Output: A string s that contains all strings $s_1, s_2, \ldots, s_n$ as substrings, such that the length of s is minimized

- **Complexity:** NP – complete
- **Note:** this formulation does not take into account sequencing errors

Shortest Superstring Problem: Example



The Shortest Superstring problem

Set of strings: {000, 001, 010, 011, 100, 101, 110, 111}

Concatenation Superstring 000 001 010 011 100 101 110 111

Shortest superstring 0 0 0 1 1 1 0 1 0 0

Reducing SSP to TSP

- Define *overlap* (s_i, s_j) as the length of the longest prefix of s_j that matches a suffix of s_i.

  aaaggcatcaaatctaaaggcatcaaa

  aaaggcatcaaatctaaaggcatcaaa

What is** overlap* (s_i, s_j) ***for these strings?

Reducing SSP to TSP

- Define *overlap* (s_i, s_j) as the length of the longest prefix of s_j that matches a suffix of s_i.

aaaggcatcaaatctaaaggcatcaaa

aaaggcatcaaatctaaaggcatcaaa

aaaggcatcaaatctaaaggcatcaaa

overlap=12

Reducing SSP to TSP

- Define *overlap (s_i, s_j)* as the length of the longest prefix of s_j that matches a suffix of s_i.

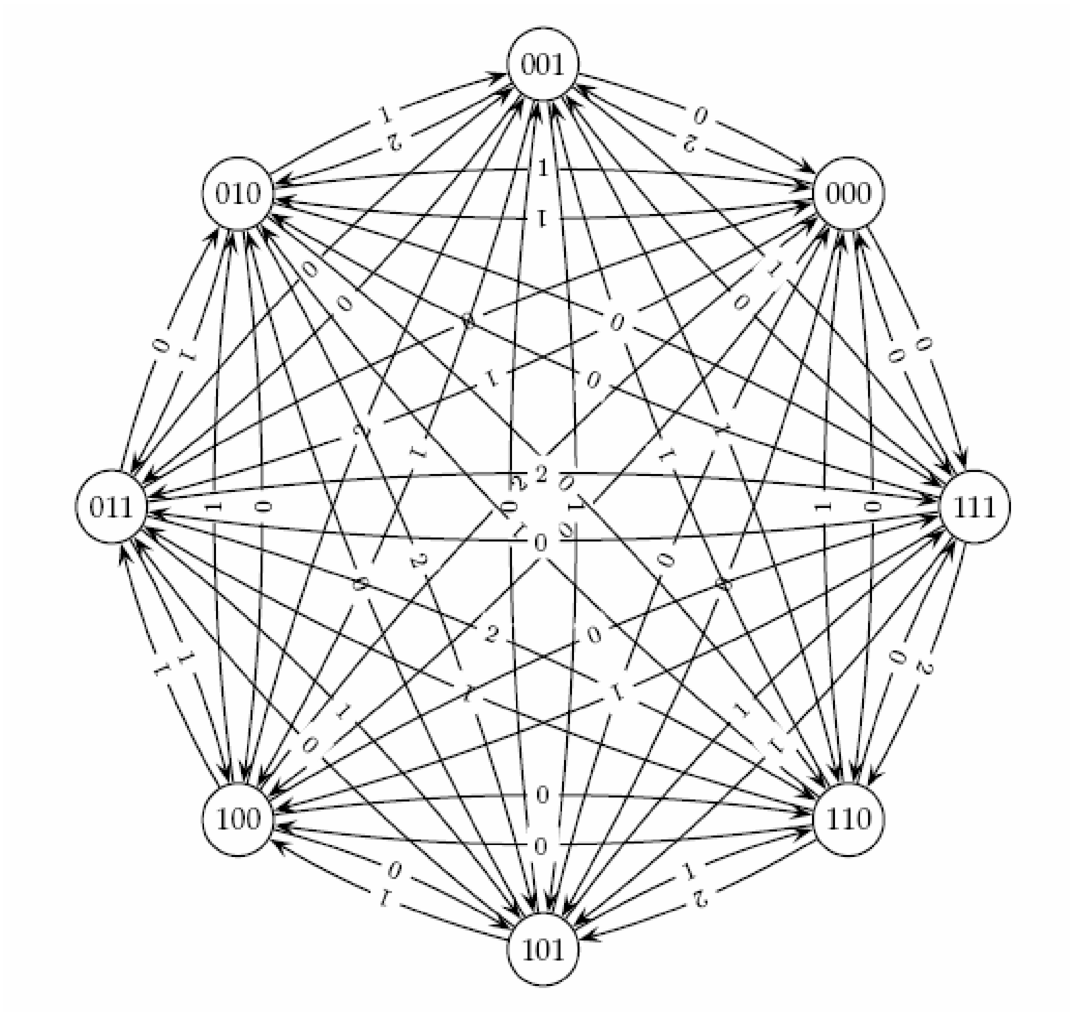
  aaaggcatcaaatctaaaggcatcaaa

  aaaggcatcaaatctaaaggcatcaaa

  aaaggcatcaaatctaaaggcatcaaa

- Construct a graph with *n* vertices representing the *n* strings $s_1, s_2, \ldots, s_n$.
- Insert edges of length *overlap (s_i, s_j)* between vertices s_i and s_j.
- Find the shortest path which visits every vertex exactly once. This is the **Traveling Salesman Problem** (TSP), which is also NP – complete.

Reducing SSP to TSP (cont'd)



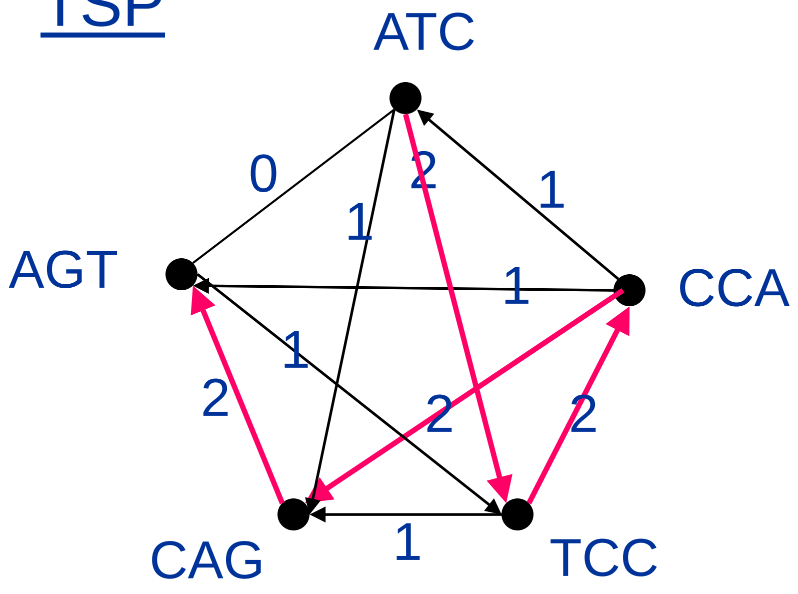
SSP to TSP: An Example

S = { ATC, CCA, CAG, TCC, AGT }

SSP

```
                AGT
             CCA
          ATC
          ATCCAGT
            TCC
               CAG
```

TSP



ATCCAGT

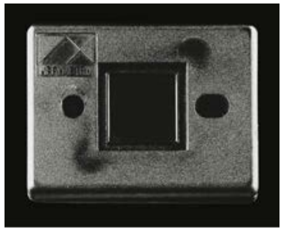
Sequencing by Hybridization (SBH): History

- **1988:** SBH suggested as an an alternative sequencing method. Nobody believed it will ever work

- **1991:** Light directed polymer synthesis developed by Steve Fodor and colleagues.

- **1994:** Affymetrix develops first 64-kb DNA microarray

*First microarray prototype **(1989)***


*First commercial DNA microarray prototype w/16,000 features **(1994)***

*500,000 features per chip **(2002)***

How SBH Works

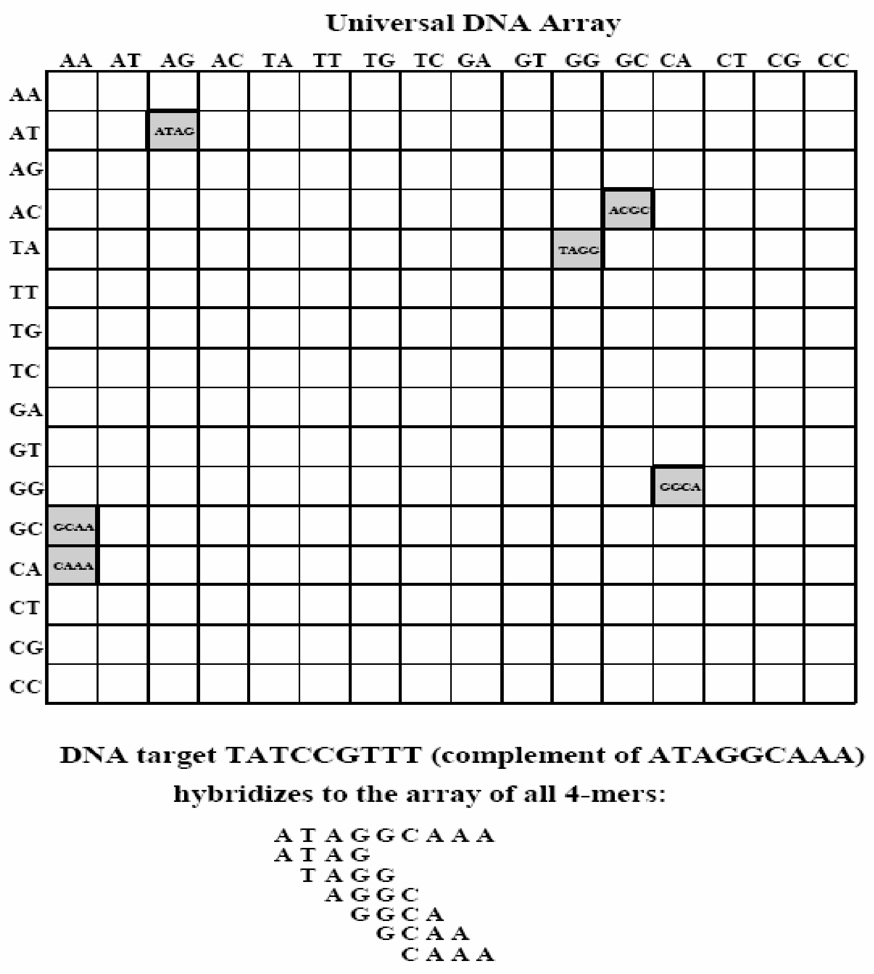
- Attach all possible DNA probes of length *l* to a flat surface, each probe at a distinct and known location. This set of probes is called the DNA array.
- Apply a solution containing fluorescently labeled DNA fragment to the array.
- The DNA fragment hybridizes with those probes that are complementary to substrings of length *l* of the fragment.

How SBH Works (cont'd)

- Using a spectroscopic detector, determine which probes hybridize to the DNA fragment to obtain the *l*–mer composition of the target DNA fragment.

- Apply the combinatorial algorithm (below) to reconstruct the sequence of the target DNA fragment from the *l* – mer composition.

Hybridization on DNA Array



Universal DNA Array

	AA	AT	AG	AC	TA	TT	TG	TC	GA	GT	GG	GC	CA	CT	CG	CC
AA																
AT			ATAG													
AG																
AC												ACGC				
TA											TAGG					
TT																
TG																
TC																
GA																
GT																
GG													GGCA			
GC	GCAA															
CA	CAAA															
CT																
CG																
CC																

DNA target TATCCGTTT (complement of ATAGGCAAA)
hybridizes to the array of all 4-mers:

```
A T A G G C A A A
A T A G
  T A G G
    A G G C
      G G C A
        G C A A
          C A A A
```

l-mer composition

- ***Spectrum (s, l)*** - *unordered* multiset of all possible *(n – l + 1)* *l*-mers in a string *s* of length *n*
- The order of individual elements in *Spectrum (s, l)* does not matter
- For *s* = TATGGTGC all of the following are equivalent representations of *Spectrum (s, 3):*

  {TAT, ATG, TGG, GGT, GTG, TGC}

  {ATG, GGT, GTG, TAT, TGC, TGG}

  {TGG, TGC, TAT, GTG, GGT, ATG}

l-mer composition

- ***Spectrum (s, l)*** - *unordered* multiset of all possible *(n – l + 1)* *l*-mers in a string *s* of length *n*
- The order of individual elements in *Spectrum (s, l)* does not matter
- For *s* = TATGGTGC all of the following are equivalent representations of *Spectrum (s, 3):*

  {TAT, ATG, TGG, GGT, GTG, TGC}

  {ATG, GGT, GTG, TAT, TGC, TGG}

  {TGG, TGC, TAT, GTG, GGT, ATG}
- We usually choose the lexicographically maximal representation as the canonical one.

Different sequences – the same spectrum

- Different sequences may have the same spectrum:

  Spectrum(GTATCT,2)=

  Spectrum(GTCTAT,2)=

  {AT, CT, GT, TA, TC}

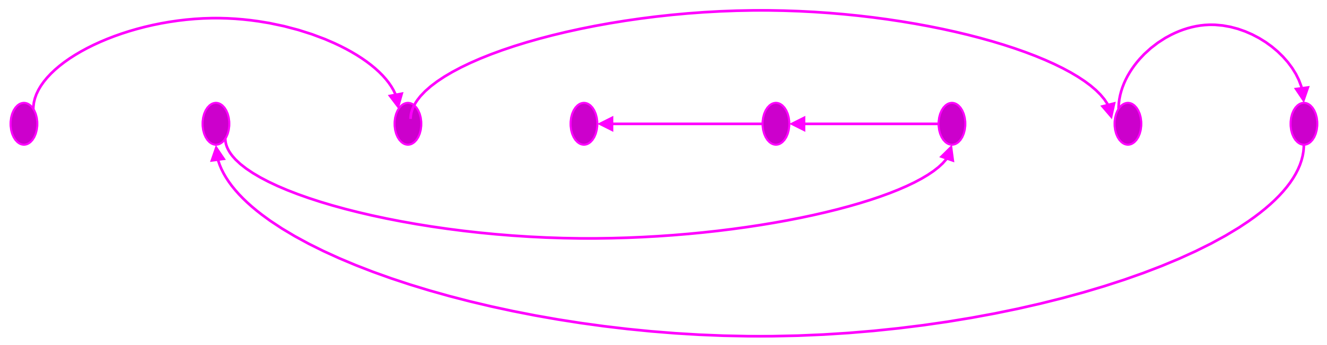
The SBH Problem

- Goal: Reconstruct a string from its *l*-mer composition
- Input: A set *S*, representing all *l*-mers from an (unknown) string *s*
- Output: String *s* such that *Spectrum (s,l)* = *S*

SBH: Hamiltonian Path Approach

S = { ATG AGG TGC TCC GTC GGT GCA CAG }

H ATG AGG TGC TCC GTC GGT GCA CAG



ATGCAGGTCC

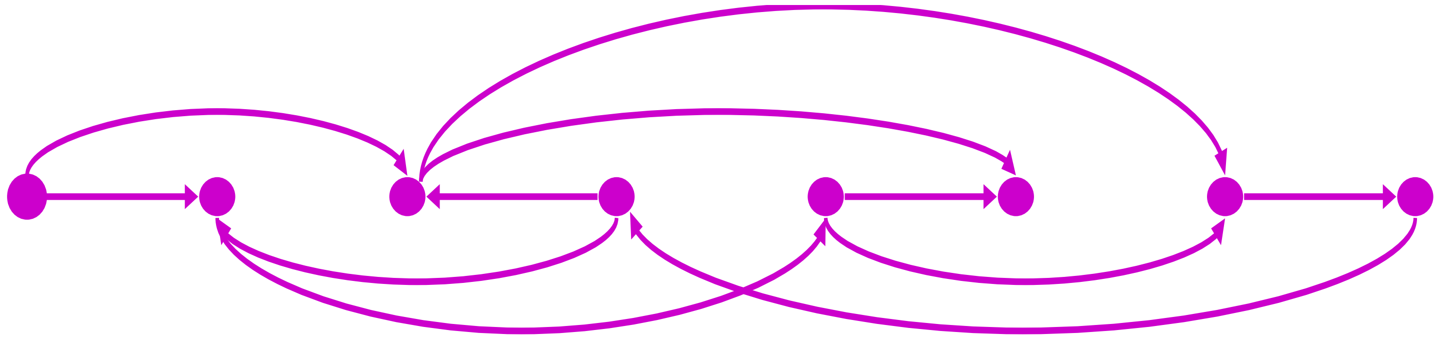
Path visited every VERTEX once

SBH: Hamiltonian Path Approach

A more complicated graph:

S = { ATG TGG TGC GTG GGC GCA GCG CGT }

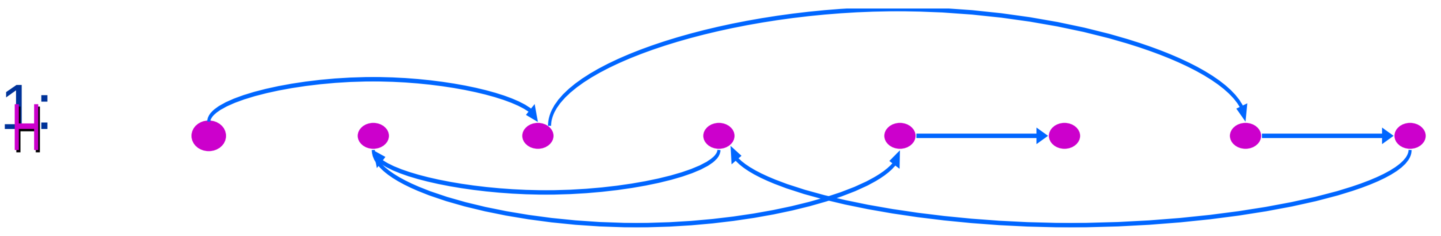
H



SBH: Hamiltonian Path Approach

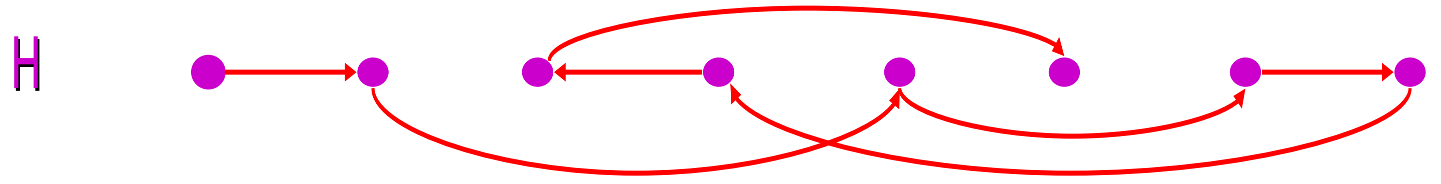
S = { ATG TGG TGC GTG GGC GCA GCG CGT }

Path 1:



ATGCGTGGCA

Path 2:



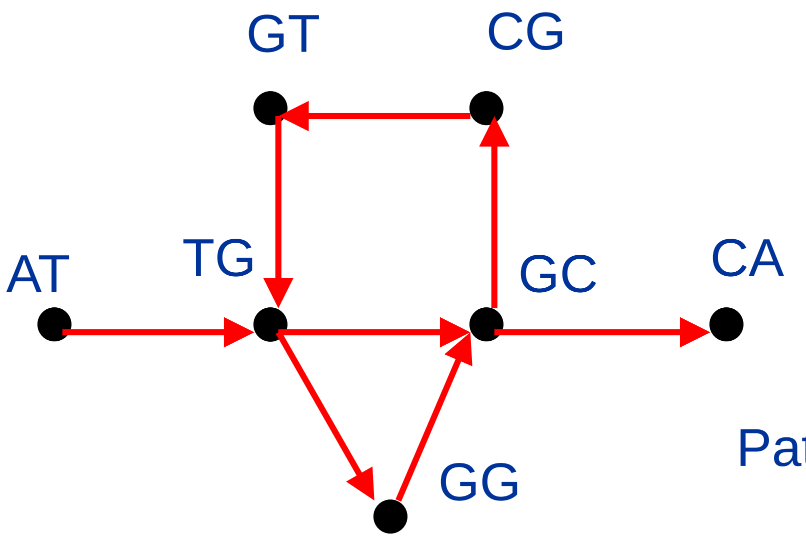
ATGGCGTGCA

SBH: Eulerian Path Approach

S = { ATG, TGC, GTG, GGC, GCA, GCG, CGT }

Vertices correspond to (*l* – 1) – mers : { AT, TG, GC, GG, GT, CA, CG }

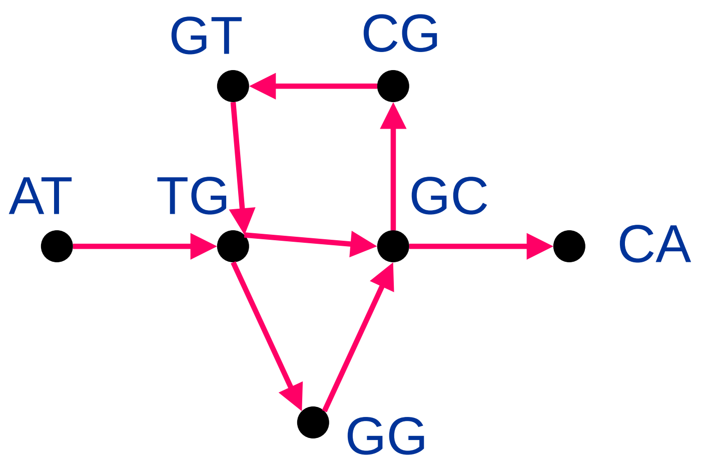
Edges correspond to *l* – mers from *S*



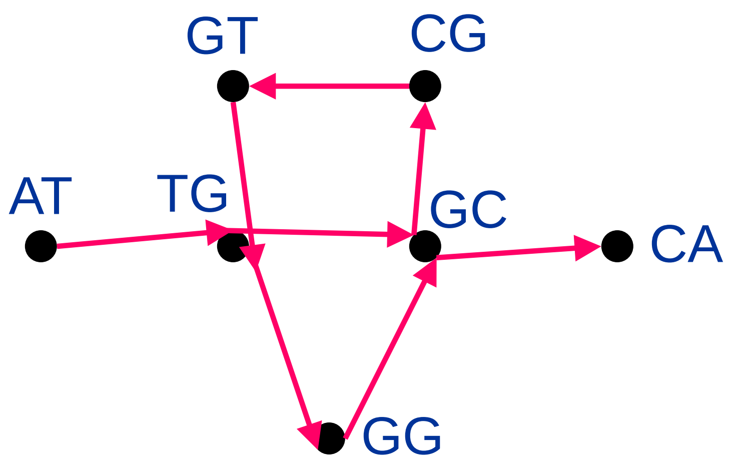
Path visited every EDGE once

SBH: Eulerian Path Approach

S = { AT, TG, GC, GG, GT, CA, CG } corresponds to two different paths:



ATGGCGTGCA



ATGCGTGGCA

Euler Theorem

- A graph is balanced if for every vertex the number of incoming edges equals to the number of outgoing edges:

$$in(v)=out(v)$$

- **Theorem**: *A connected graph is Eulerian if and only if each of its vertices is balanced.*

Euler Theorem: Proof

- Eulerian → balanced

  for every edge entering *v* (incoming edge) there exists an edge leaving *v* (outgoing edge). Therefore

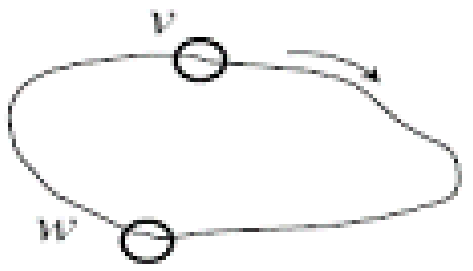
  $$in(v)=out(v)$$

- Balanced → Eulerian

  ???

Algorithm for Constructing an Eulerian Cycle

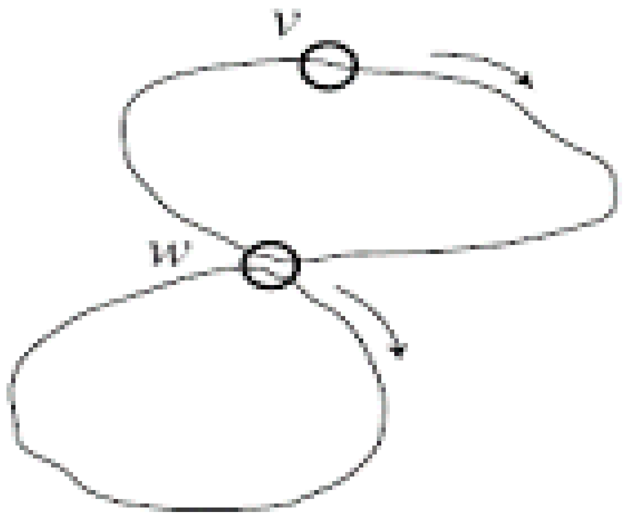
a. Start with an arbitrary vertex *v* and form an arbitrary cycle with unused edges until a dead end is reached. Since the graph is Eulerian this dead end is necessarily the starting point, i.e., vertex *v*.



(a)

Algorithm for Constructing an Eulerian Cycle (cont'd)

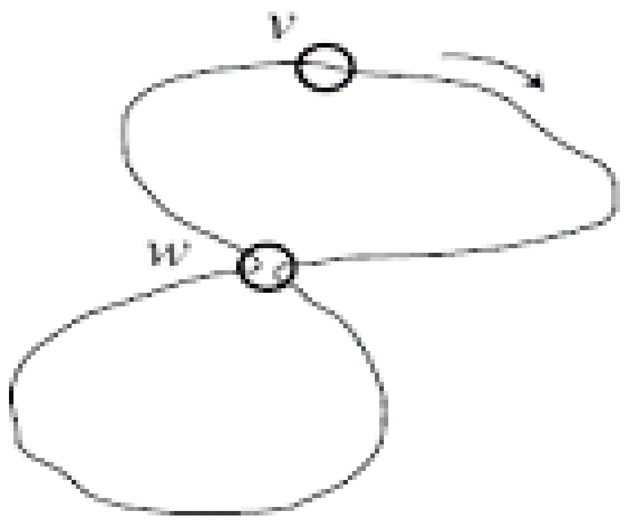
b. If cycle from (a) above is not an Eulerian cycle, it must contain a vertex *w*, which has untraversed edges. Perform step (a) again, using vertex *w* as the starting point. Once again, we will end up in the starting vertex *w.*



(b)

Algorithm for Constructing an Eulerian Cycle (cont'd)

c. Combine the cycles from (a) and (b) into a single cycle and iterate step (b).



(c)

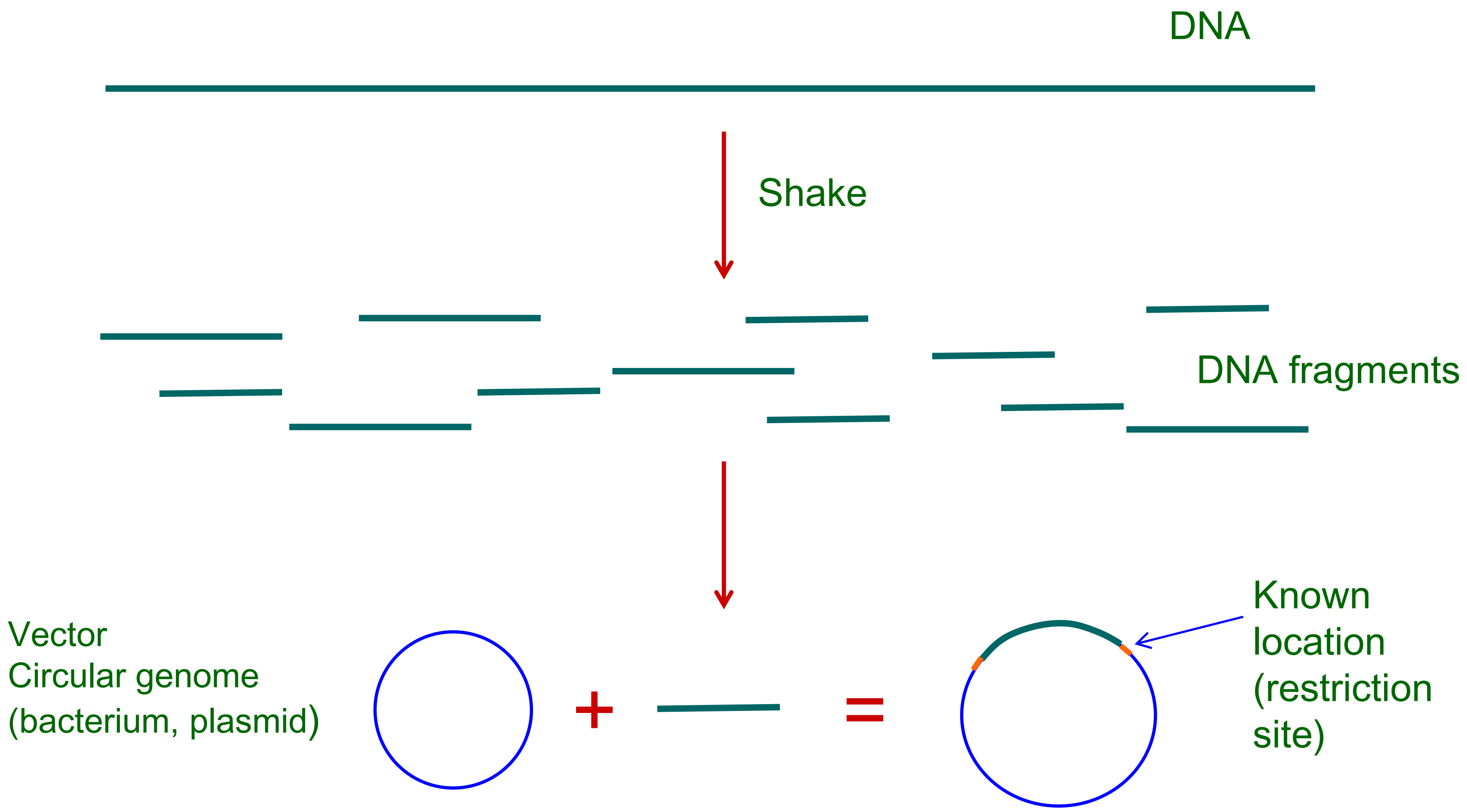
Euler Theorem: Extension

- **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.*

Some Difficulties with SBH

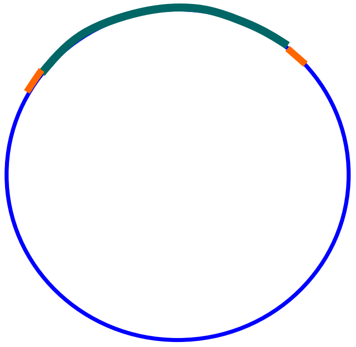
- **Fidelity of Hybridization:** difficult to detect differences between probes hybridized with perfect matches and 1 or 2 mismatches
- **Array Size:** Effect of low fidelity can be decreased with longer *l*-mers, but array size increases exponentially in *l*. Array size is limited with current technology.
- **Practicality:** SBH is still impractical. As DNA microarray technology improves, SBH may become practical in the future
- **Practicality again**: Although SBH is still impractical, it spearheaded expression analysis and SNP analysis techniques

Traditional DNA Sequencing

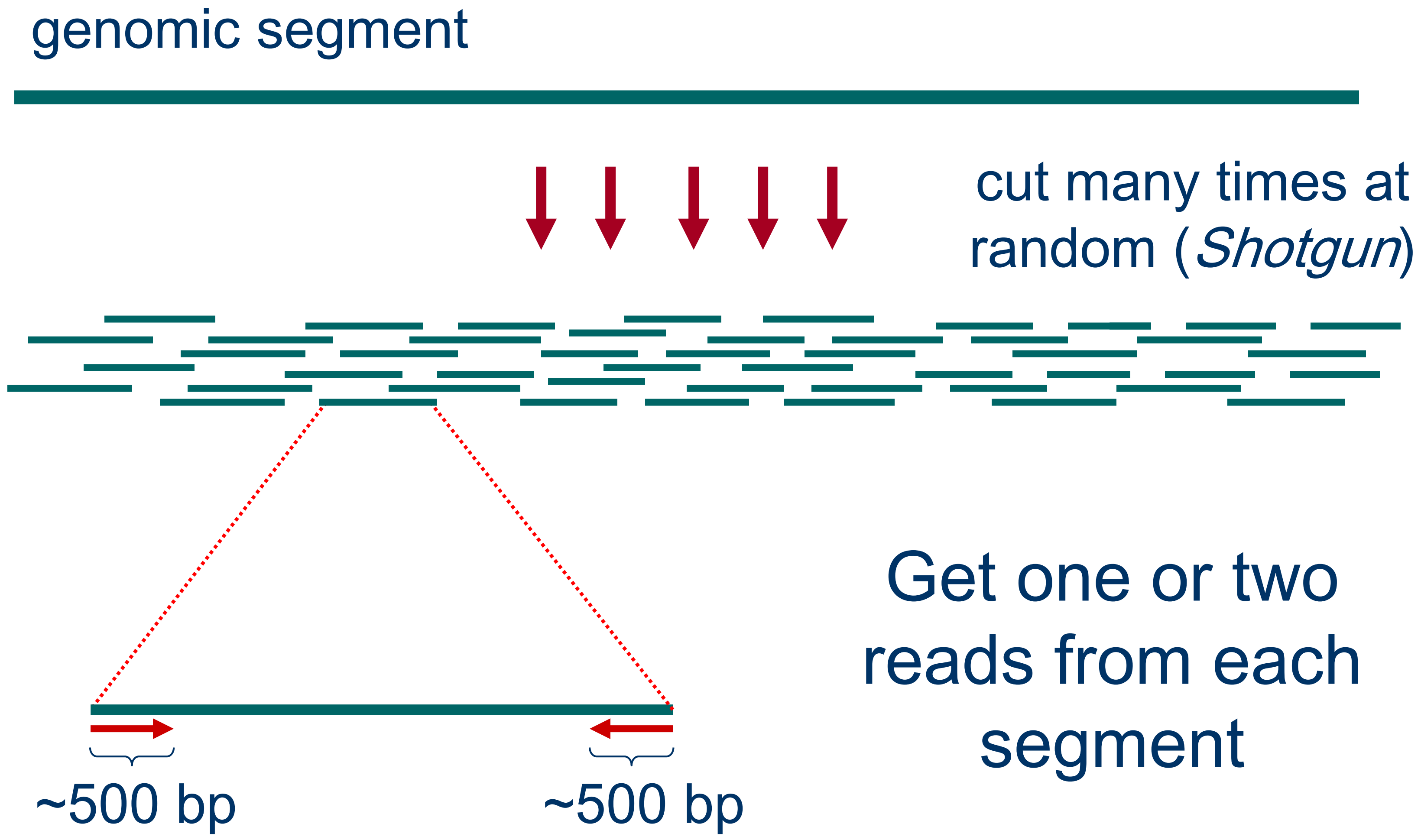


Different Types of Vectors

VECTOR	**Size of insert (bp)**
Plasmid	2,000 - 10,000
Cosmid	40,000
BAC (Bacterial Artificial Chromosome)	70,000 - 300,000
YAC (Yeast Artificial Chromosome)	> 300,000 Not used much recently

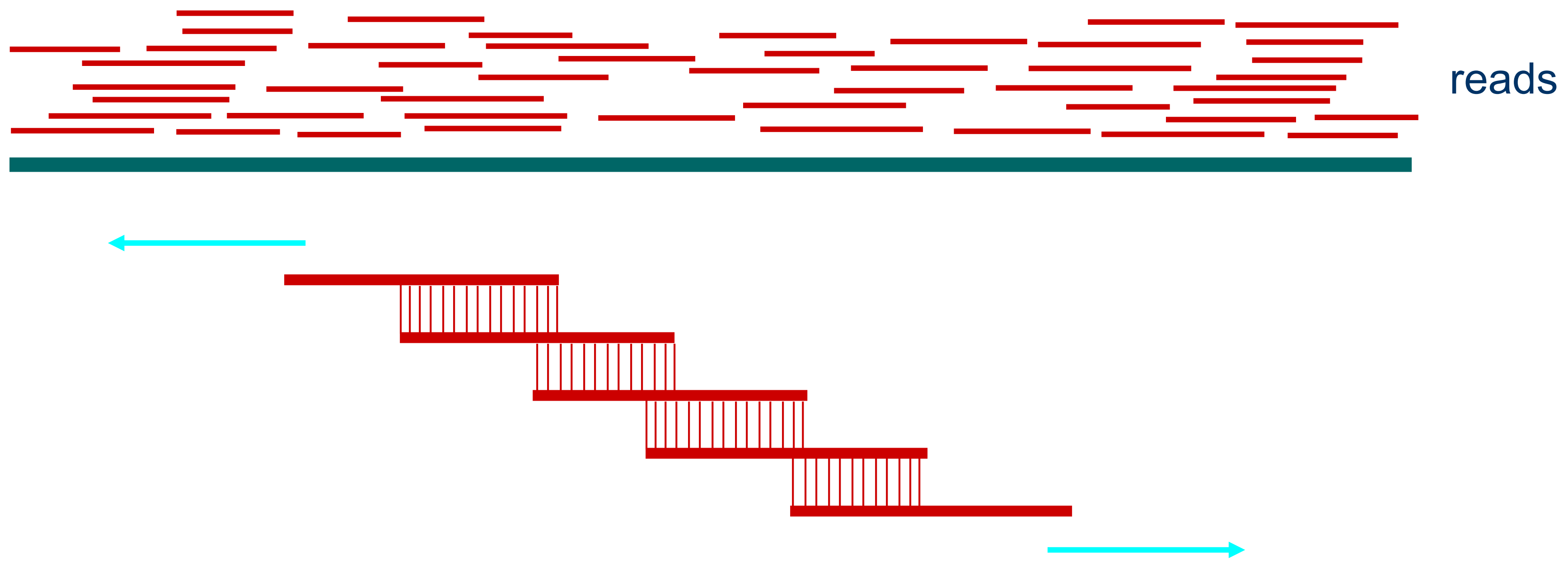


Shotgun Sequencing



Get one or two reads from each segment

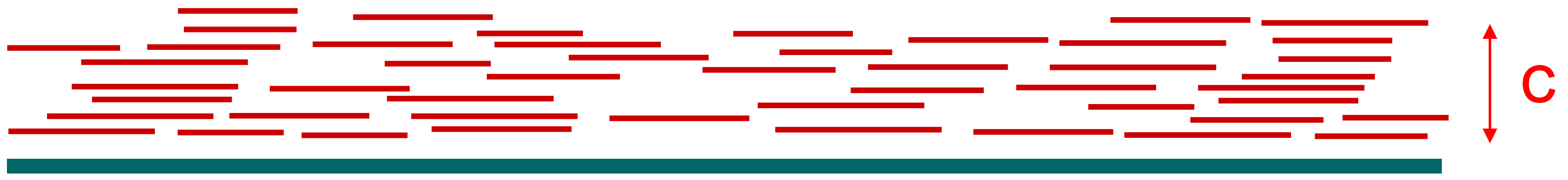
Fragment Assembly



Cover region with ~7-fold redundancy

Overlap reads and extend to reconstruct the original genomic region

Read Coverage



Length of genomic segment: ***L***

Number of reads: ***n***

Length of each read: ***l***

Coverage $\boldsymbol{C = n\, l\, /\, L}$

How much coverage is enough?

Lander-Waterman model:

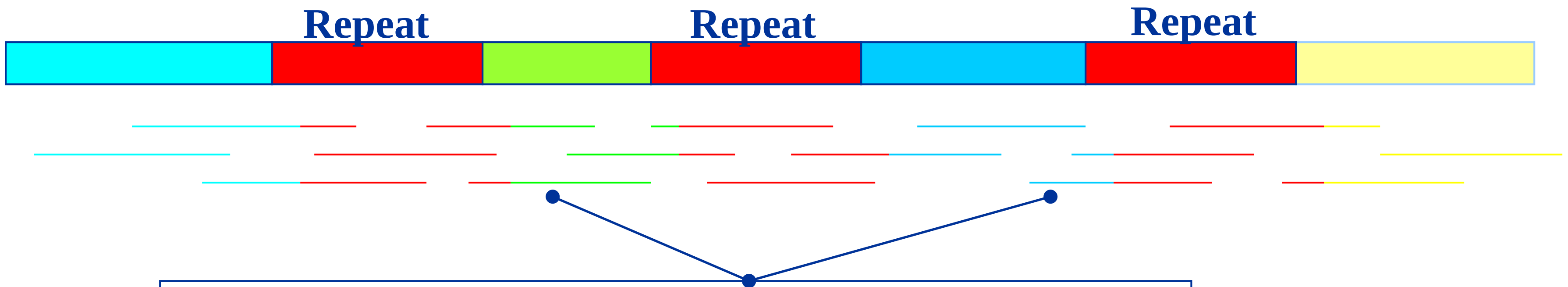
Assuming uniform distribution of reads, *C*=10 results in 1 gapped region per 1,000,000 nucleotides

Challenges in Fragment Assembly

- Repeats: A **major** problem for fragment assembly
- > 50% of human genome are repeats:

  - over 1 million *Alu* repeats (about 300 bp)

  - about 200,000 LINE repeats (1000 bp and longer)



Green and blue fragments are interchangeable when assembling repetitive DNA

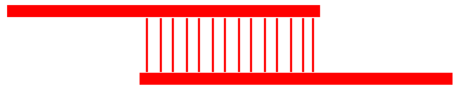
Repeat Types

• Low-Complexity DNA	(e.g. ATATATATACATA…)
• Microsatellite repeats	$(a_1 \ldots a_k)^N$ where k ~ 3-6 (e.g. CAGCAGTAGCAGCACCAG)
• Transposons/retrotransposons	
– SINE	Short Interspersed Nuclear Elements (e.g., *Alu*: ~300 bp long, 10^6 copies)
– LINE	Long Interspersed Nuclear Elements ~500 - 5,000 bp long, 200,000 copies
– LTR retrotransposons	Long Terminal Repeats (~700 bp) at each end
• Gene Families	genes duplicate & then diverge
• Segmental duplications	~very long, very similar copies

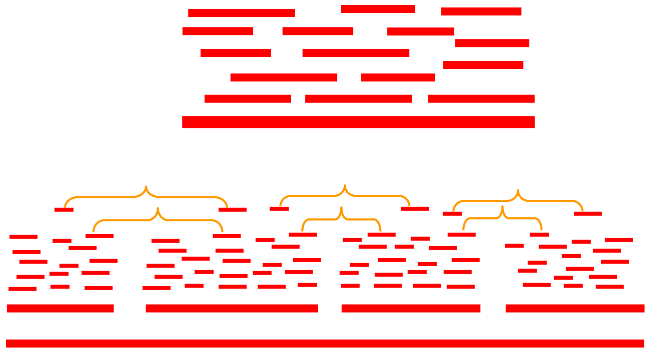
Overlap-Layout-Consensus

Assemblers: ARACHNE, PHRAP, CAP, TIGR, CELERA

Overlap: find potentially overlapping reads



Layout: merge reads into contigs and contigs into supercontigs



Consensus: derive the DNA sequence and correct read errors

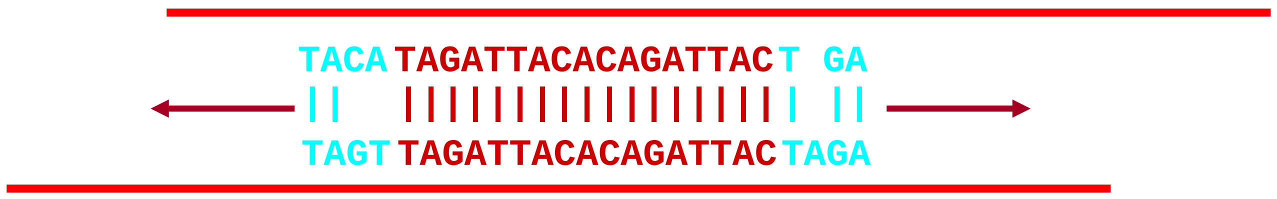
..ACGATTACAATAGGTT..

Overlap

- Find the best match between the suffix of one read and the prefix of another
- Due to sequencing errors, need to use dynamic programming to find the optimal *overlap alignment*
- Apply a filtration method to filter out pairs of fragments that do not share a significantly long common substring

Overlapping Reads

- Sort all *k*-mers in reads (*k* ~ 24)
- Find pairs of reads sharing a k-mer
- Extend to full alignment – throw away if not >95% similar



Overlapping Reads and Repeats

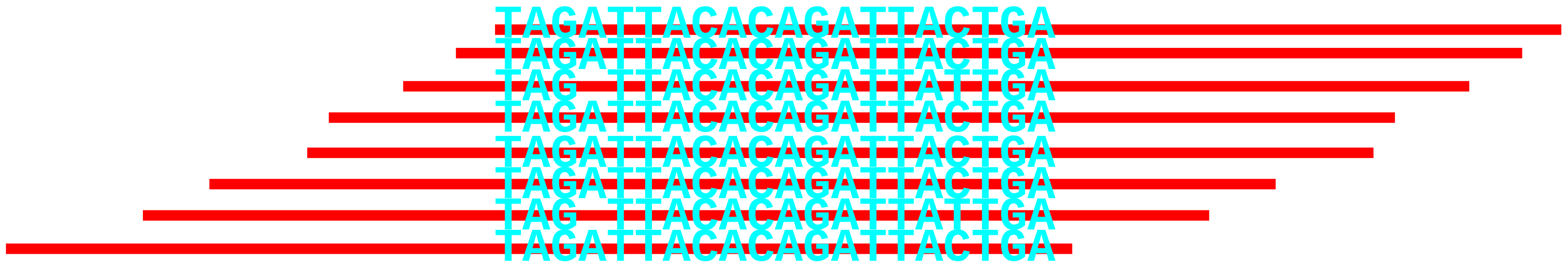
- A *k*-mer that appears N times, initiates N^2 comparisons
- For an *Alu* that appears 10^6 times → 10^{12} comparisons – too much
- **Solution:**

  Discard all *k*-mers that appear more than

  $t \times$ Coverage, ($t \sim 10$)

Finding Overlapping Reads

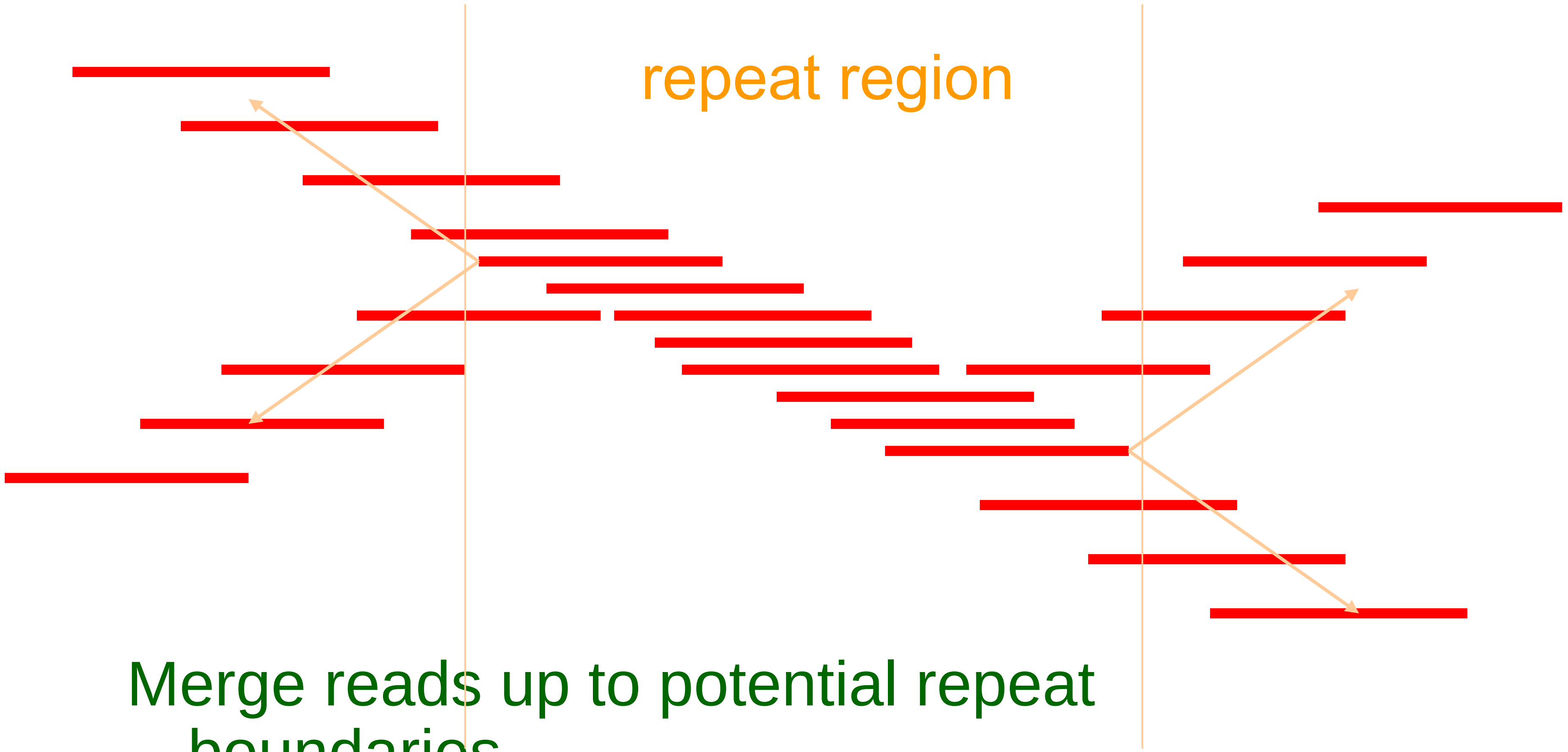
Create local multiple alignments from the overlapping reads



Layout

- Repeats are a major challenge
- Do two aligned fragments really overlap, or are they from two copies of a repeat?
- Solution: repeat masking – hide the repeats!!!
- Masking results in high rate of misassembly (up to 20%)
- Misassembly means a lot more work at the finishing step

Merge Reads into Contigs



Merge reads up to potential repeat boundaries

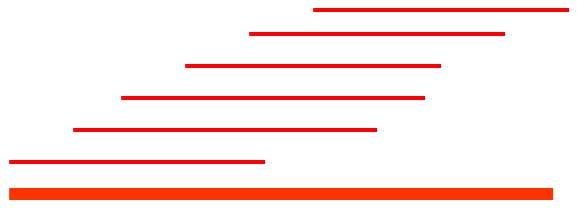
Repeats, Errors, and Contig Lengths

- Repeats shorter than read length are OK

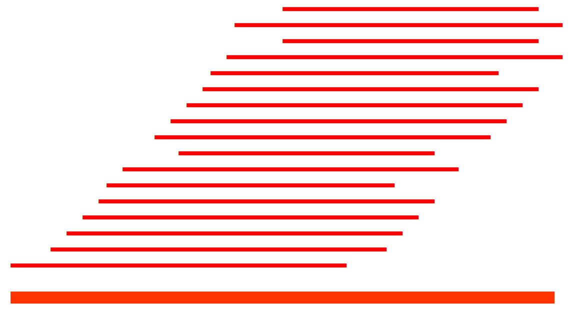
- Repeats with more base pair differencess than sequencing error rate are OK

- To make a smaller portion of the genome **appear** repetitive, try to:
  - Increase read length
  - Decrease sequencing error rate

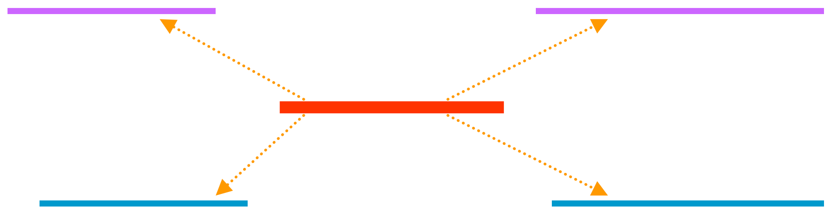
Link Contigs into Supercontigs



Normal density



Too dense:
Overcollapsed?

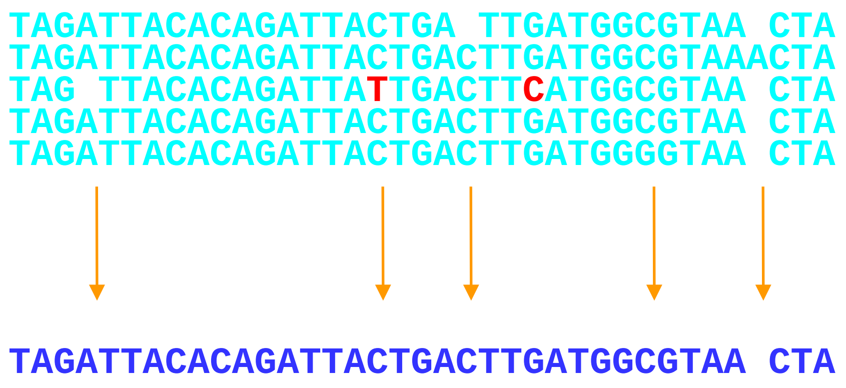


Inconsistent links:
Overcollapsed?

Consensus

- A consensus sequence is derived from a profile of the assembled fragments
- A sufficient number of reads is required to ensure a statistically significant consensus
- Reading errors are corrected

Derive Consensus Sequence



```
TAGATTACACAGATTACTGA TTGATGGCGTAA CTA
TAGATTACACAGATTACTGACTTGATGGCGTAAACTA
TAG TTACACAGATTATTGACTTCATGGCGTAA CTA
TAGATTACACAGATTACTGACTTGATGGCGTAA CTA
TAGATTACACAGATTACTGACTTGATGGGGTAA CTA

TAGATTACACAGATTACTGACTTGATGGCGTAA CTA
```

Derive multiple alignment from pairwise read alignments

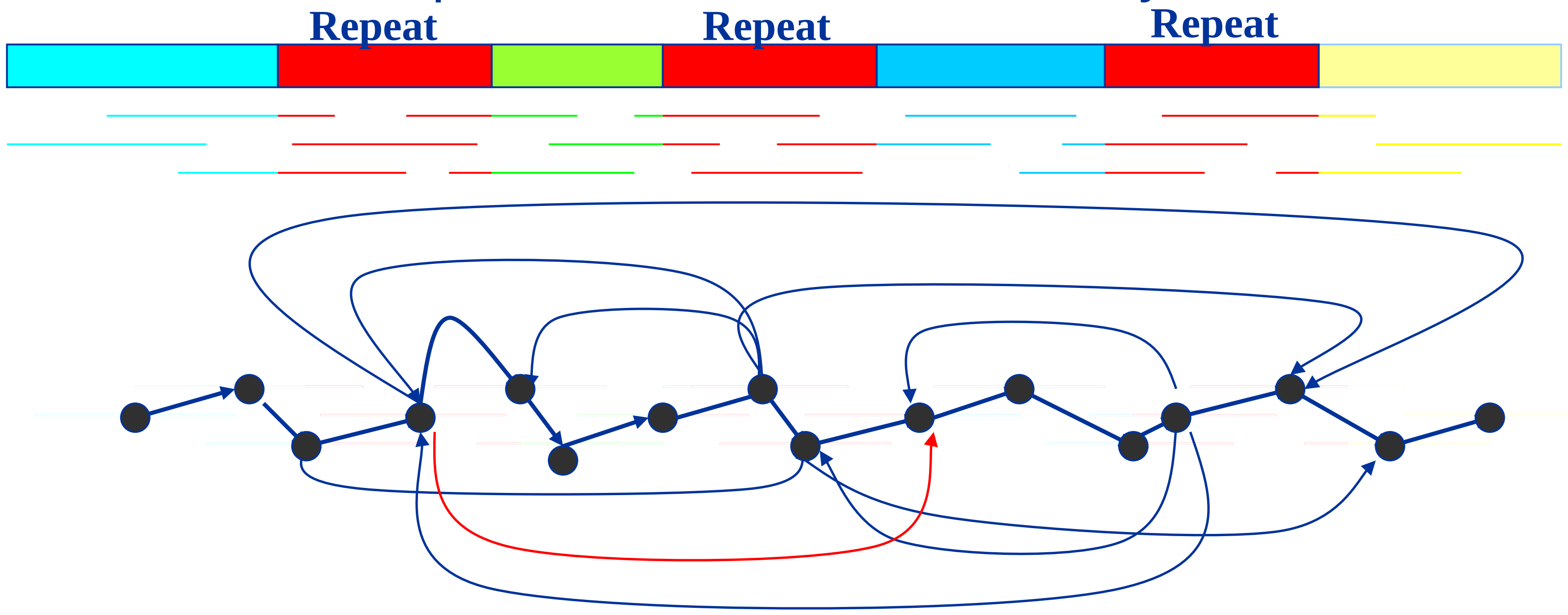
Derive each consensus base by weighted voting

EULER - A New Approach to Fragment Assembly

- Traditional “overlap-layout-consensus” technique has a high rate of mis-assembly
- EULER uses the Eulerian Path approach borrowed from the SBH problem
- Fragment assembly without repeat masking can be done in linear time with greater accuracy

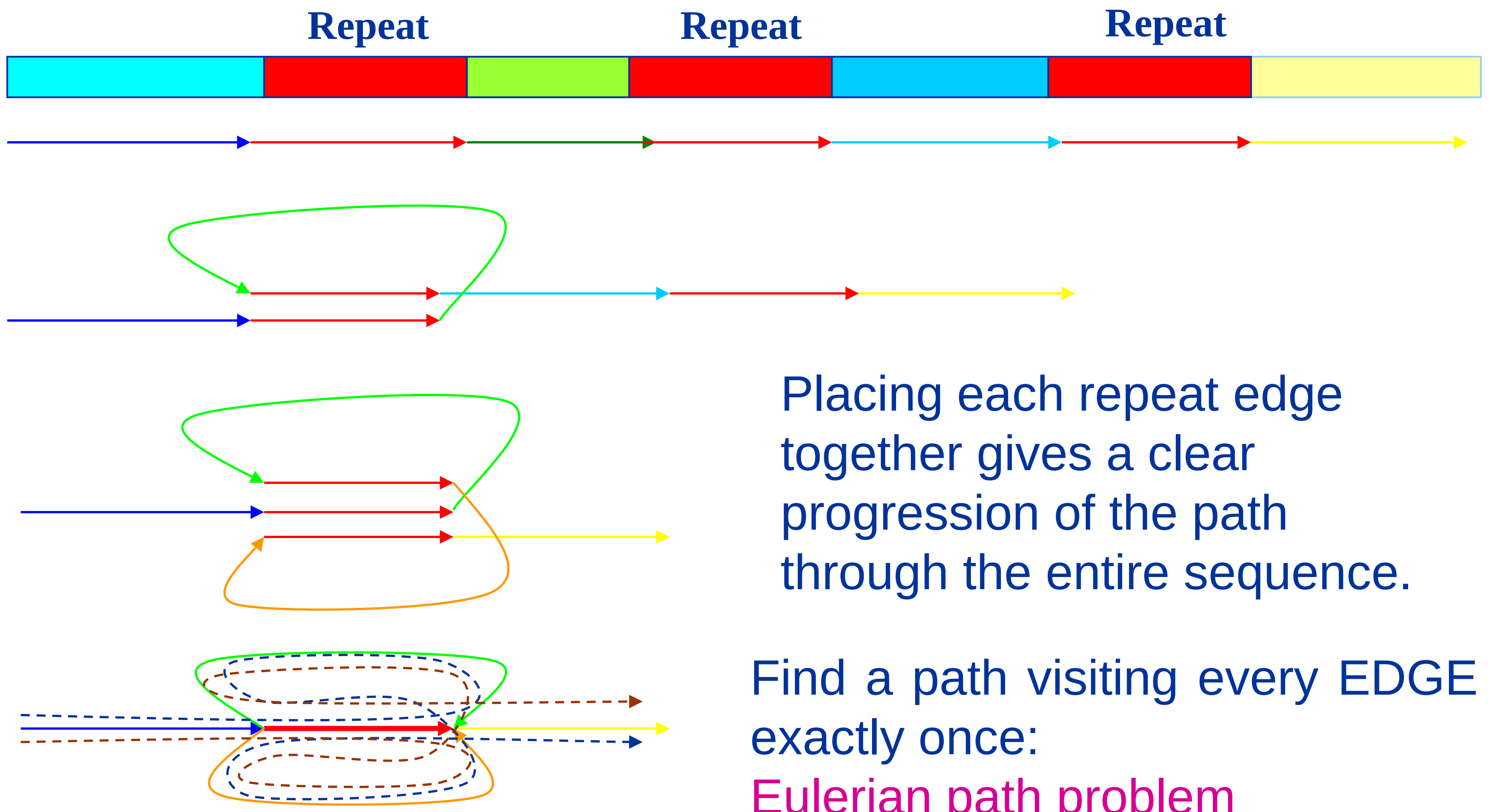
Overlap Graph: Hamiltonian Approach

Each vertex represents a read from the original sequence.
Vertices from repeats are connected to many others.



Find a path visiting every VERTEX exactly once: Hamiltonian path problem

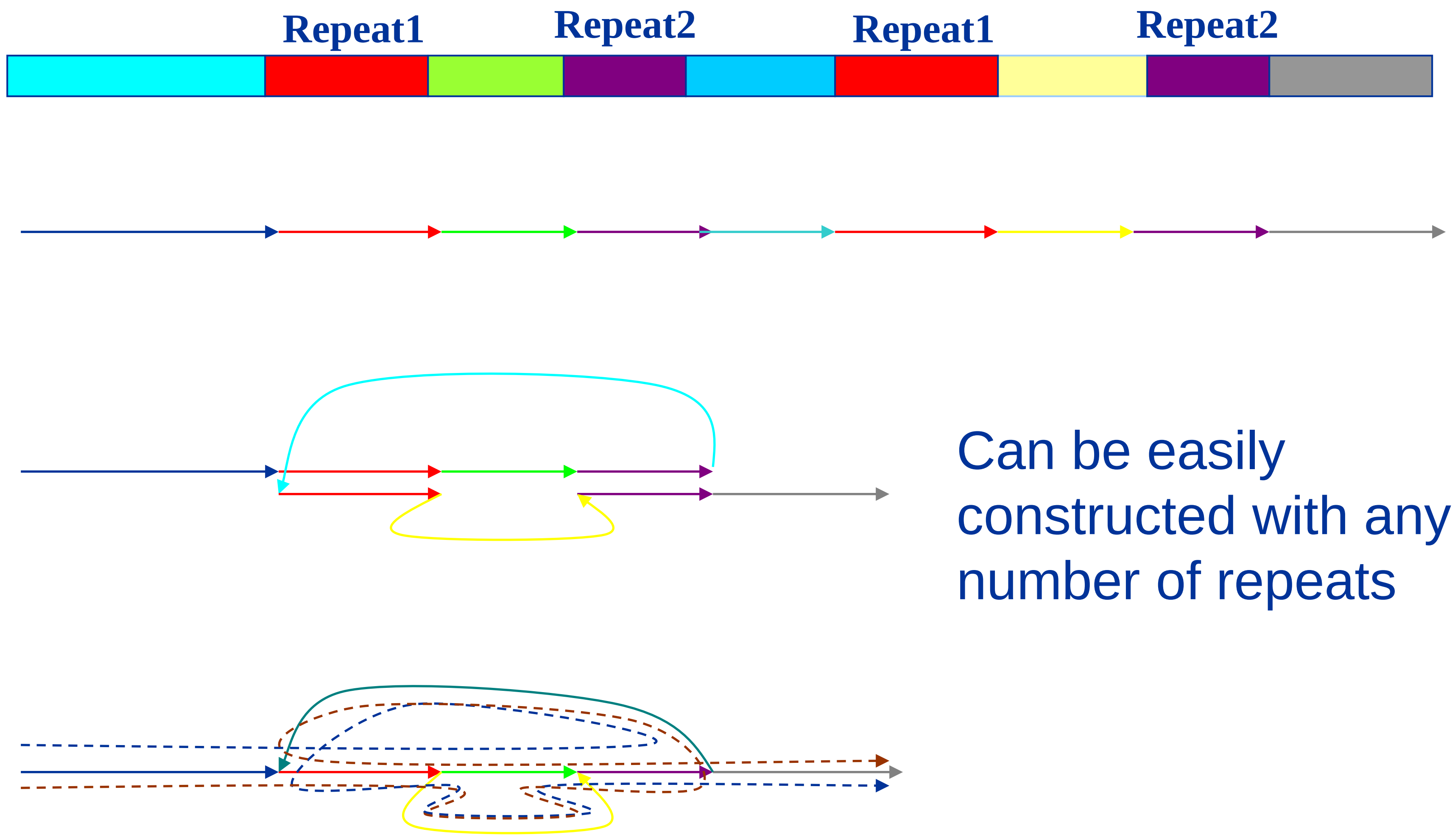
Overlap Graph: Eulerian Approach



Placing each repeat edge together gives a clear progression of the path through the entire sequence.

Find a path visiting every EDGE exactly once:
Eulerian path problem

Multiple Repeats



Can be easily constructed with any number of repeats

Construction of Repeat Graph

- Construction of repeat graph from *k* – mers: emulates an SBH experiment with a huge (virtual) DNA chip.

- Breaking reads into *k* – mers: Transform sequencing data into virtual DNA chip data.

Construction of Repeat Graph (cont'd)

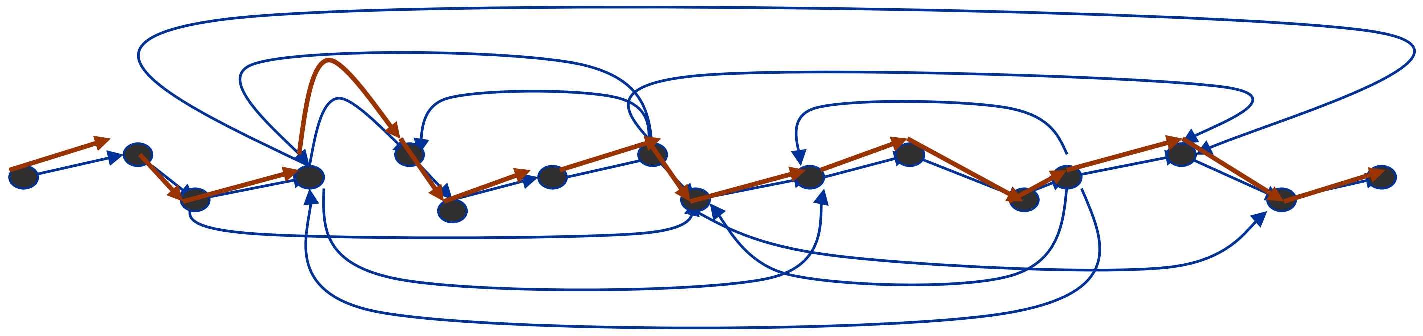
- Error correction in reads: "consensus first" approach to fragment assembly. Makes reads (almost) error-free BEFORE the assembly even starts.

- Using reads and mate-pairs to simplify the repeat graph (Eulerian Superpath Problem).

Approaches to Fragment Assembly

Find a path visiting every VERTEX exactly once in the OVERLAP graph:

Hamiltonian path problem

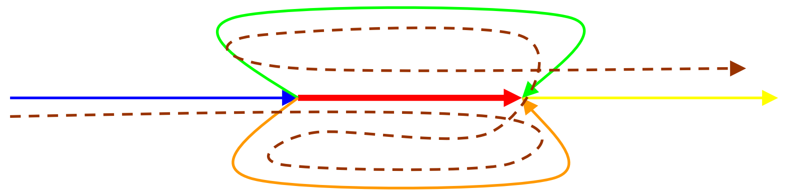


NP-complete: algorithms unknown

Approaches to Fragment Assembly (cont'd)

Find a path visiting every EDGE exactly once in the REPEAT graph:

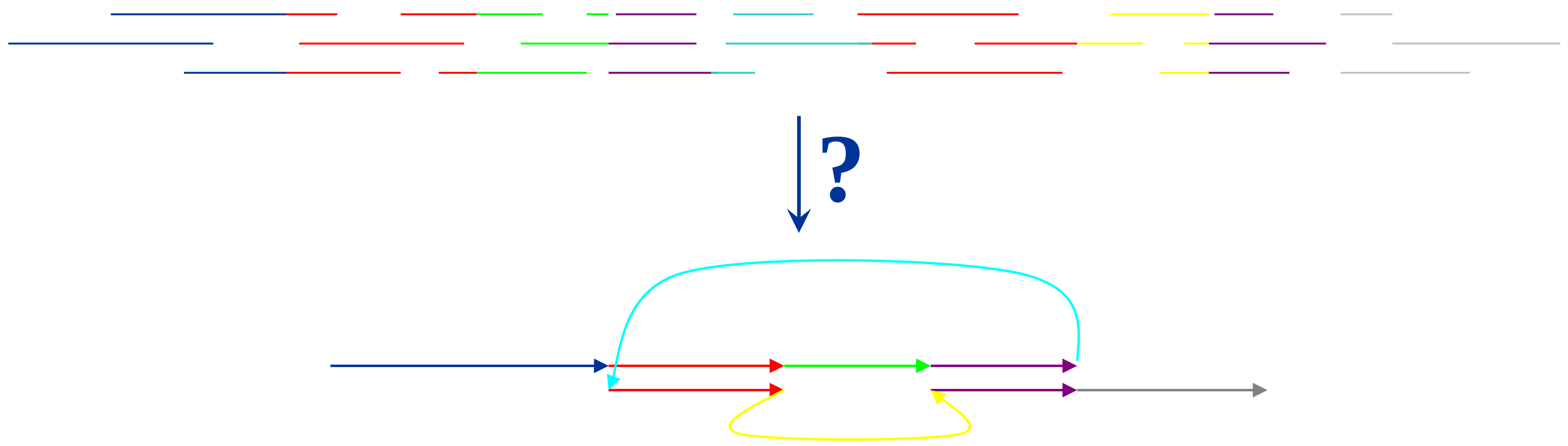
Eulerian path problem



Linear time algorithms are known

Making Repeat Graph Without DNA

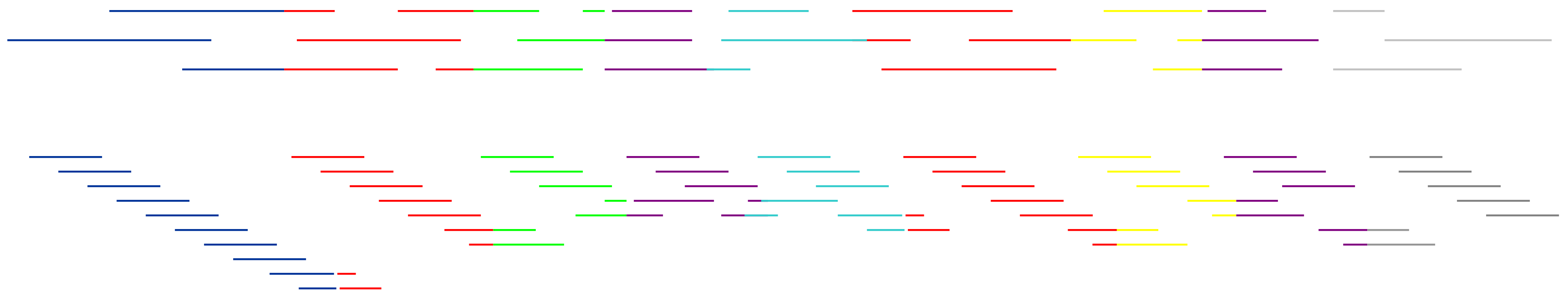
- Problem: Construct the repeat graph from a collection of reads.



- Solution: Break the reads into smaller pieces.

Repeat Sequences: Emulating a DNA Chip

- Virtual DNA chip allows the biological problem to be solved within the technological constraints.

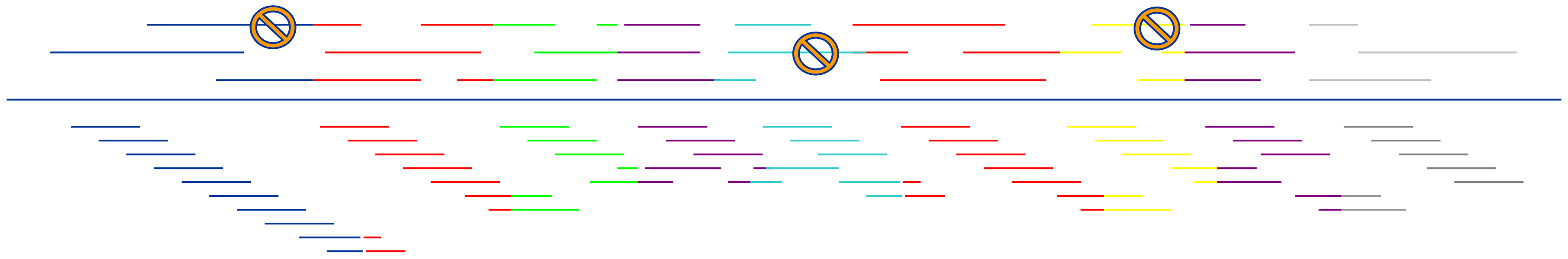


Repeat Sequences: Emulating a DNA Chip (cont'd)

- Reads are constructed from an original sequence in lengths that allow biologists a high level of certainty.
- They are then broken again to allow the technology to sequence each within a reasonable array.

Minimizing Errors

- If an error exists in one of the 20-mer reads, the error will be perpetuated among all of the smaller pieces broken from that read.



Minimizing Errors (cont’d)

- However, that error will not be present in the other instances of the 20-mer read.
- So it is possible to eliminate most point mutation errors before reconstructing the original sequence.

Conclusions

- Graph theory is a vital tool for solving biological problems
- Wide range of applications, including sequencing, motif finding, protein networks, and many more