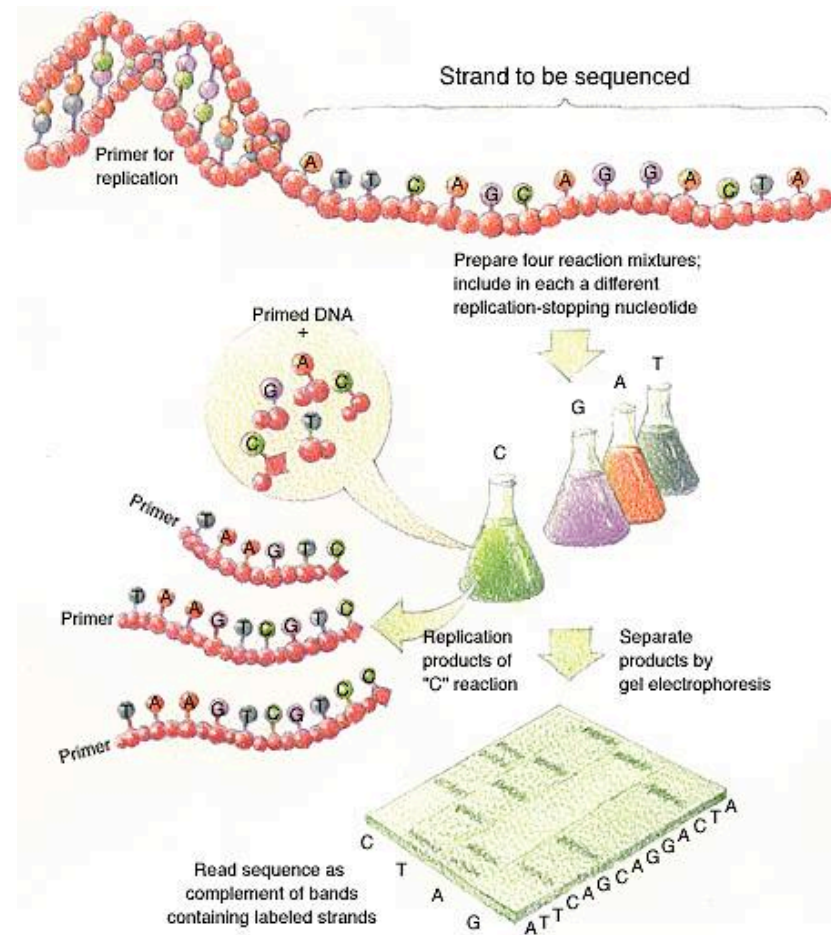


Eulerian graph

# DNA Sequencing

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



# 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  $s_1, s_2, \dots, s_n$
- Output: A string  $s$  that contains all strings  $s_1, s_2, \dots, s_n$  as substrings, such that the length of  $s$  is minimized
- **Complexity**: NP-hard
- **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

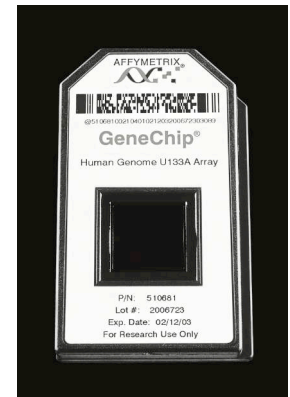
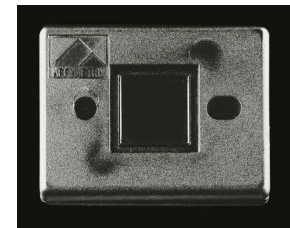
000 001 010 011 100 101 110 111

The diagram illustrates the shortest superstring 0001110100. The original strings are highlighted as substrings within this superstring:

- 000 (at the start)
- 001 (starting at index 3)
- 010 (starting at index 5)
- 011 (starting at index 4)
- 100 (starting at index 7)
- 101 (starting at index 6)
- 110 (starting at index 5)
- 111 (starting at index 4)

# Sequencing by Hybridization (SBH): History

- **1988:** SBH is suggested as an alternative sequencing method.  
*First microarray prototype (1989)*
- **1991:** Light directed polymer synthesis developed by Steve Fodor and colleagues.  
*First commercial DNA microarray prototype w/16,000 features (1994)*
- **1994:** Affymetrix develops first 64-kb DNA microarray  
*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

- 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										TAGC						
TT																
TG																
TC																
GA																
GT																
GG													GCCA			
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  $s = \text{TATGGTGC}$  all of the following are equivalent representations of *Spectrum* ( $s, 3$ ):
  - $\{\text{TAT}, \text{ATG}, \text{TGG}, \text{GGT}, \text{GTG}, \text{TGC}\}$
  - $\{\text{ATG}, \text{GGT}, \text{GTG}, \text{TAT}, \text{TGC}, \text{TGG}\}$
  - $\{\text{TGG}, \text{TGC}, \text{TAT}, \text{GTG}, \text{GGT}, \text{ATG}\}$
- We usually choose the lexicographically maximal representation as the canonical one.

Observations: different sequences may have the same spectrum

- Different sequences may have the same spectrum:

$\text{Spectrum}(\text{GTATCT}, 2) =$

$\text{Spectrum}(\text{GTCTAT}, 2) =$

$\{\text{AT}, \text{CT}, \text{GT}, \text{TA}, \text{TC}\}$

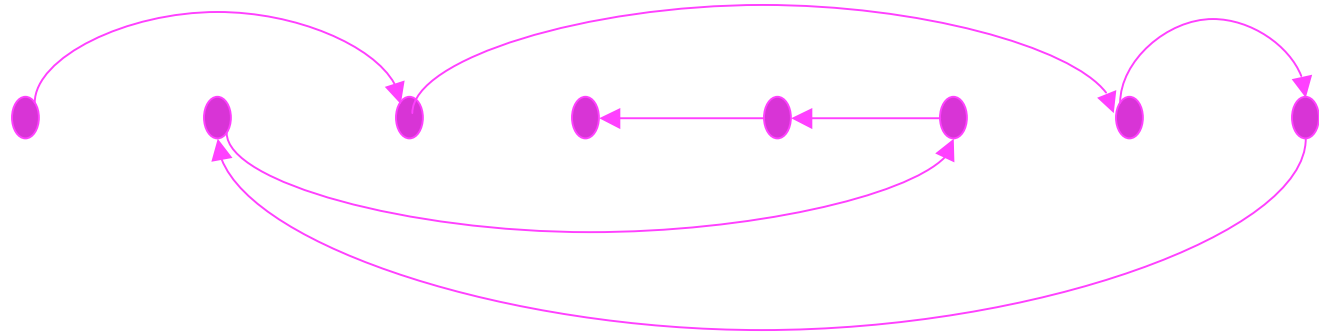
# The SBH problem

- 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  $\text{Spectrum}(s, l) = S$

# SBH: Hamiltonian path approach

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

**H**    ATG    AGG    TGC    TCC    GTC    GGT    GCA    CAG



**ATGCAGGTCC**

Path visited every VERTEX once

# SBH: Hamiltonian path approach

A more complicated graph:

$$S = \left\{ \begin{array}{ccccccc} \text{ATG} & \text{TGG} & \text{TGC} & \text{GTG} & \text{GGC} & \text{GCA} & \text{GCG} \\ \text{CGT} & & & & & & \end{array} \right\}$$

# SBH: Hamiltonian path approach

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

Path 1:

ATGCGTGGCA

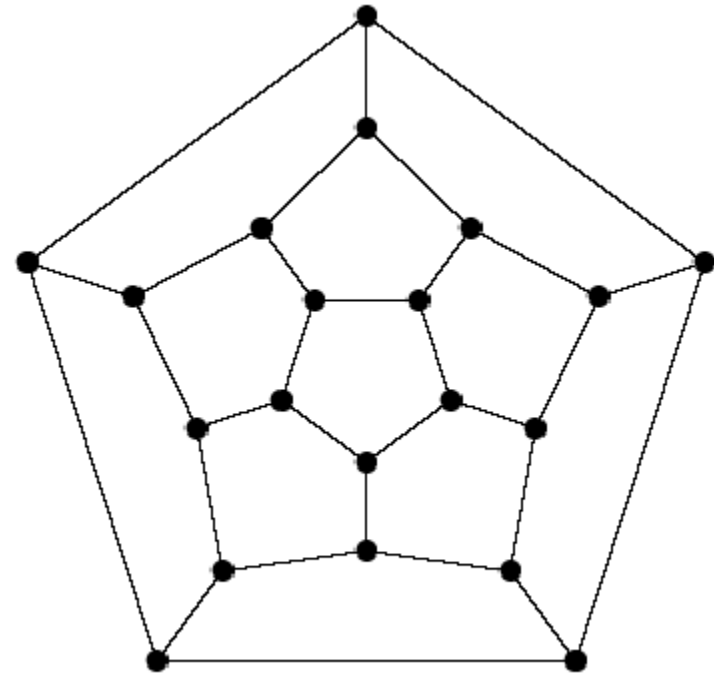
Path 2:

ATGGCGTGCA



# Hamiltonian cycle problem

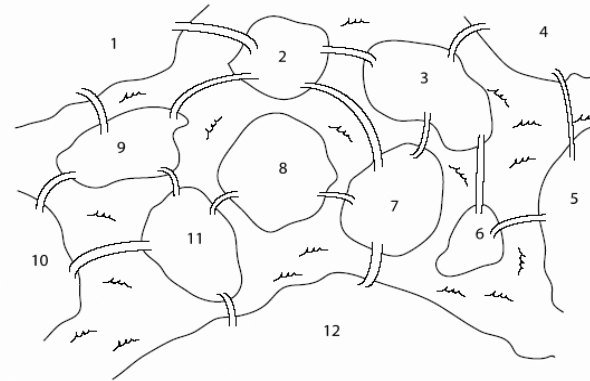
- Find a cycle that visits every ***vertex*** exactly once
- NP–complete



Game invented by Sir  
William Hamilton in 1857

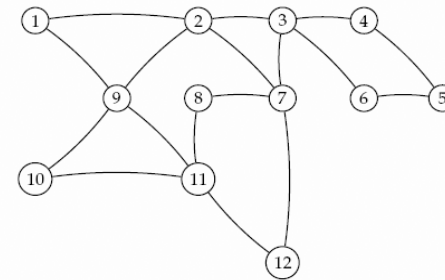
# Eulerian cycle problem

- Find a cycle that visits every **edge** exactly once



(a)

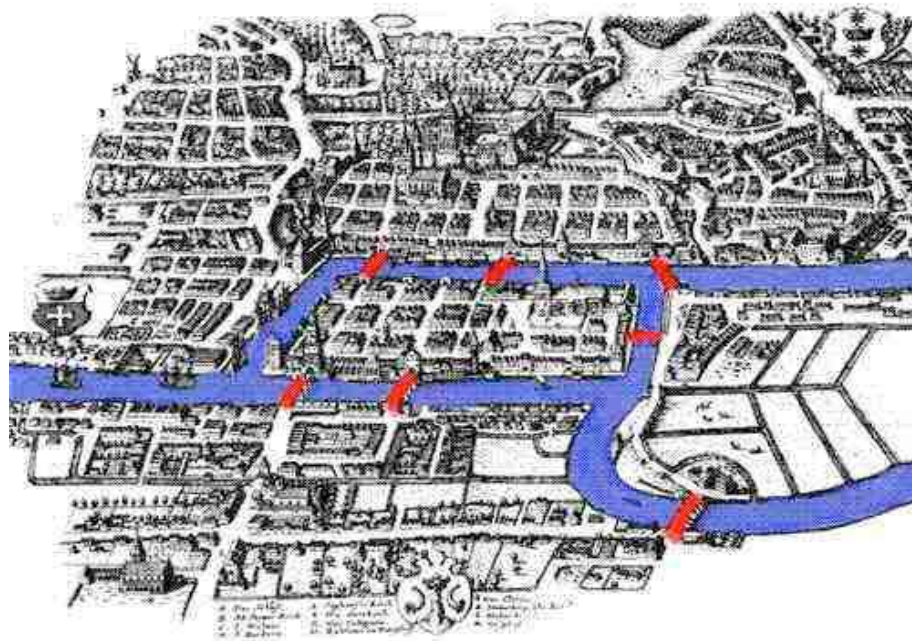
- Linear time



More complicated Königsberg

# The Bridge Obsession Problem

Find a tour crossing every bridge just once  
*Leonhard Euler, 1735*



*Bridges of Königsberg*

# 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  $\rightarrow$  balanced

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

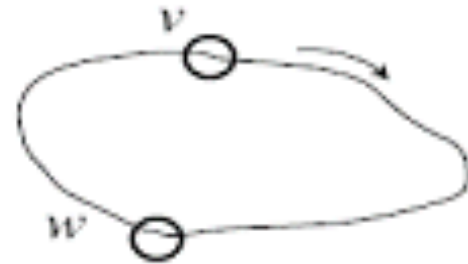
$$in(v)=out(v)$$

- Balanced  $\rightarrow$  Eulerian

???

# Algorithm for Constructing an Eulerian Cycle

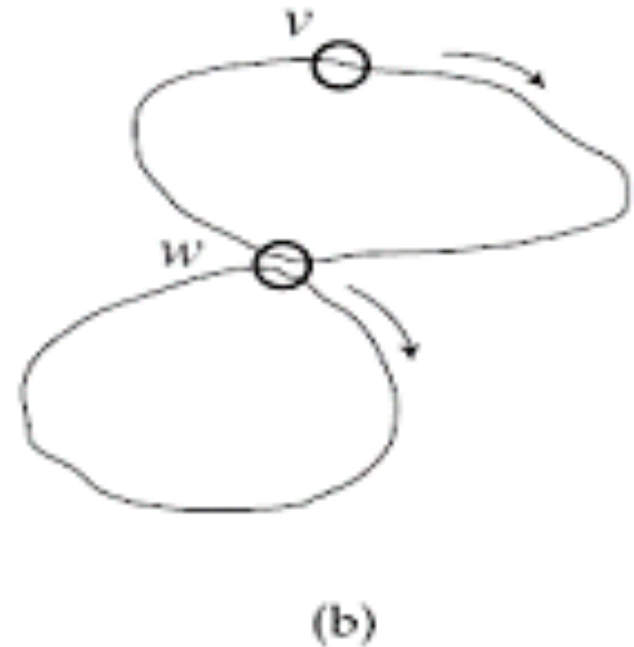
Start with an arbitrary vertex  $v$  and form an arbitrary cycle with unused edges until a dead end is reached. Since the graph is Eulerian this dead end is necessarily the starting point, i.e., vertex  $v$ .



(a)

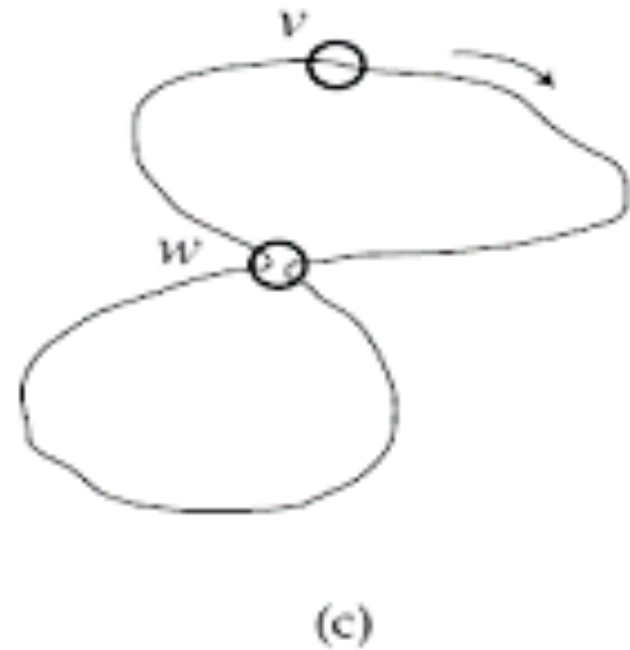
## Algorithm for Constructing an Eulerian Cycle (cont'd)

- 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  $w$ .



## Algorithm for Constructing an Eulerian Cycle

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





# Euler Theorem: extension

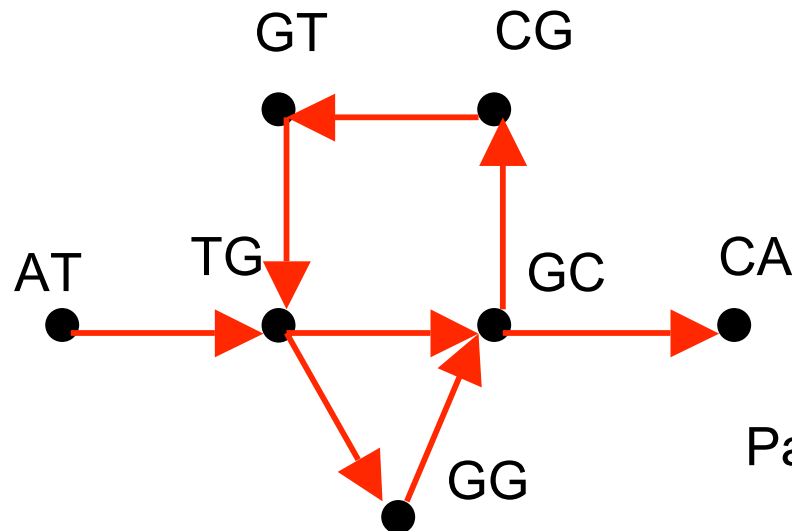
- **Theorem:** *A connected graph has an Eulerian path if and only if it contains at most two semi-balanced vertices and all other vertices are balanced.*

# SBH: Eulerian path approach

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

Vertices correspond to  $(l-1)$ -mers :  $\{ \text{AT}, \text{TG}, \text{GC}, \text{GG}, \text{GT}, \text{CA}, \text{CG} \}$

Edges correspond to  $l$ -mers from  $S$

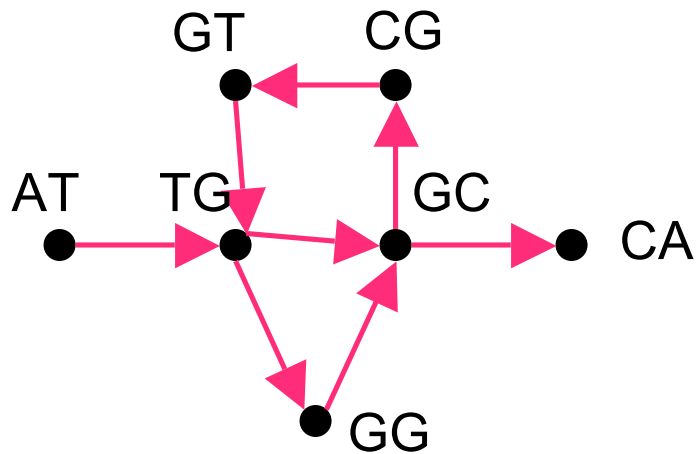


**De Bruijn graph**

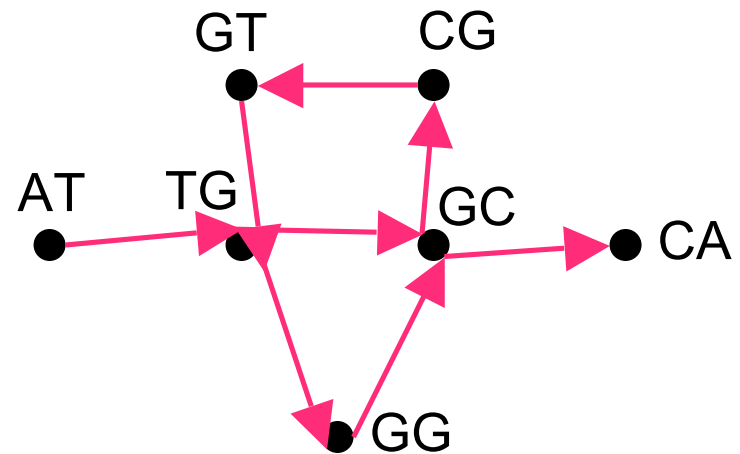
Path visited every EDGE once

# SBH: multiple solutions

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



ATGGCGTGCA



ATGCGTGGCA

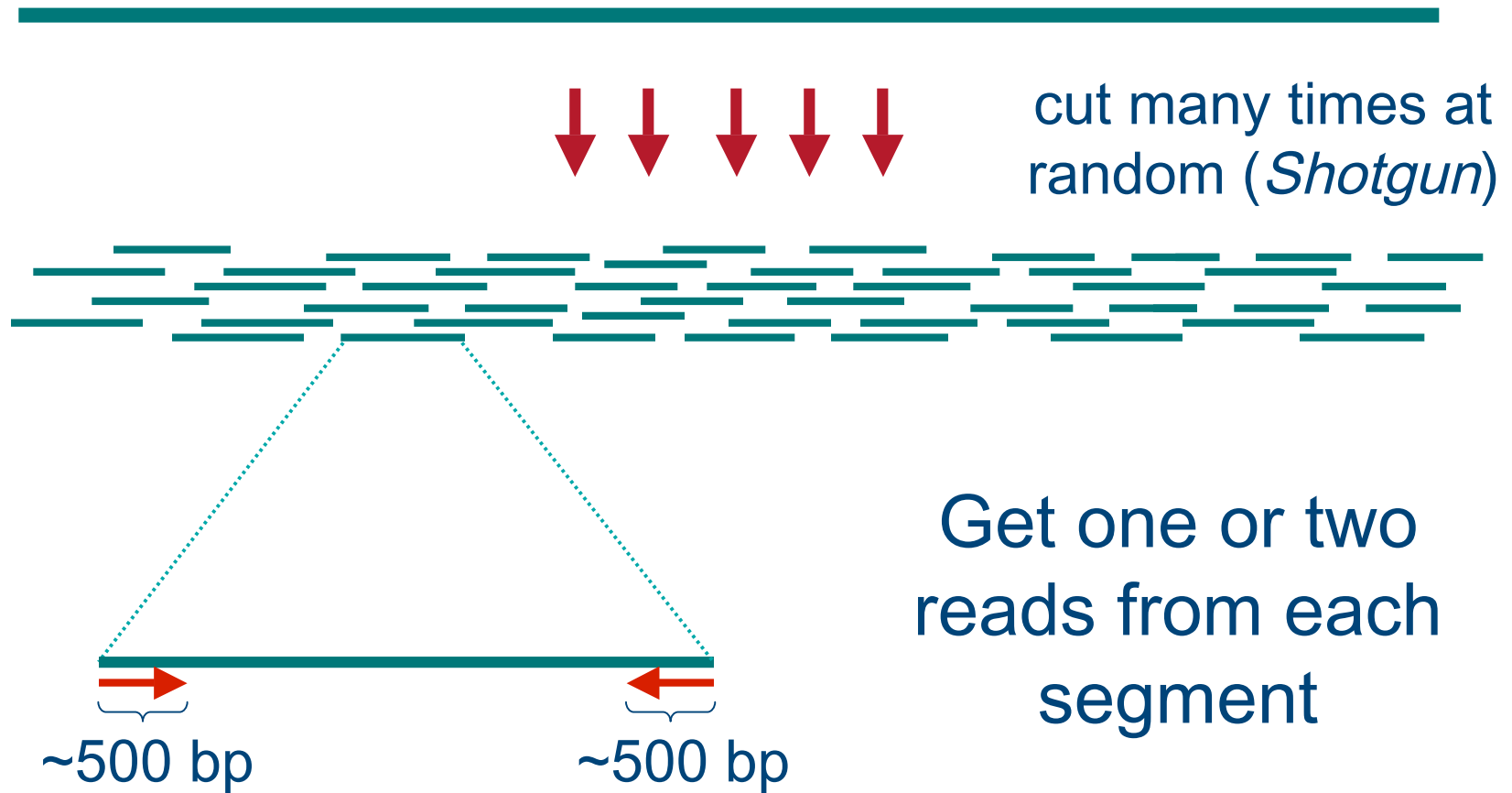
# Representing alternative SBH solutions by Eulerian graphs

- # of solutions  $\leftrightarrow$  # of Eulerian cycles
  - BEST theorem
- Obtains all solutions from one solution  $\rightarrow$  Cassette transformations

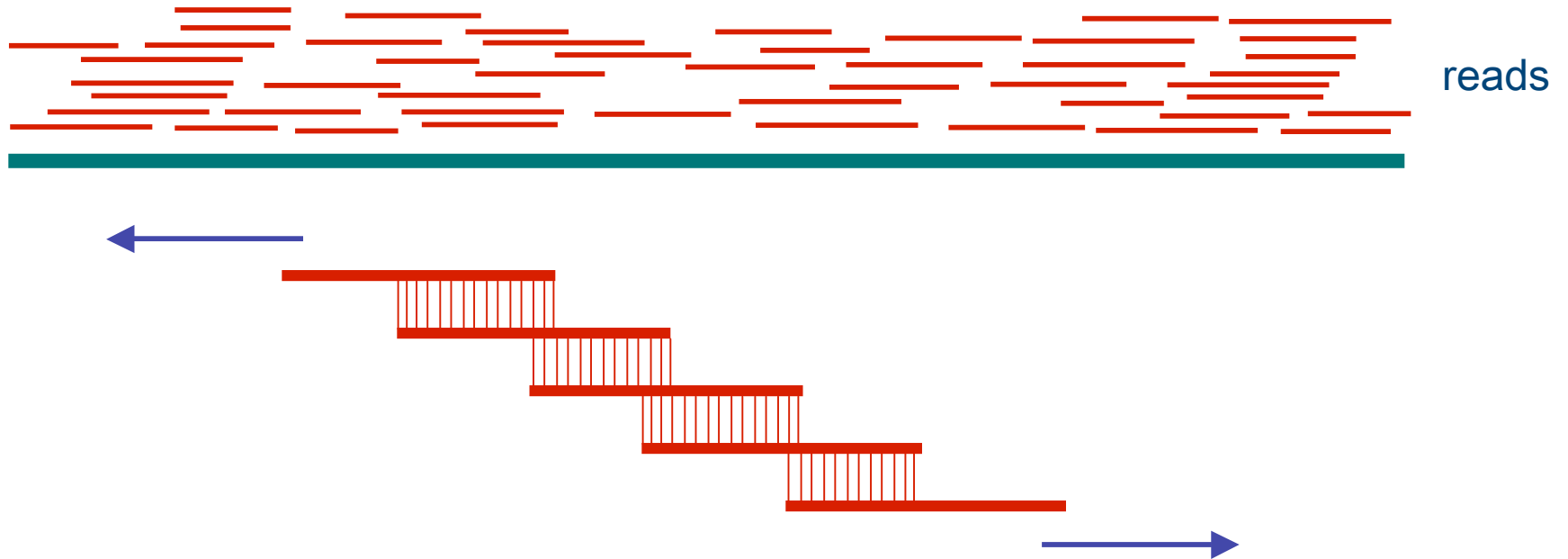
Reads assembly

# Shotgun Sequencing

genomic segment



# Fragment Assembly



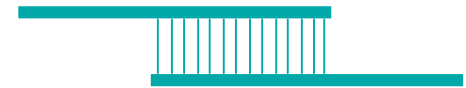
Cover region with ~7-fold redundancy

Overlap reads and extend to reconstruct the original genomic region

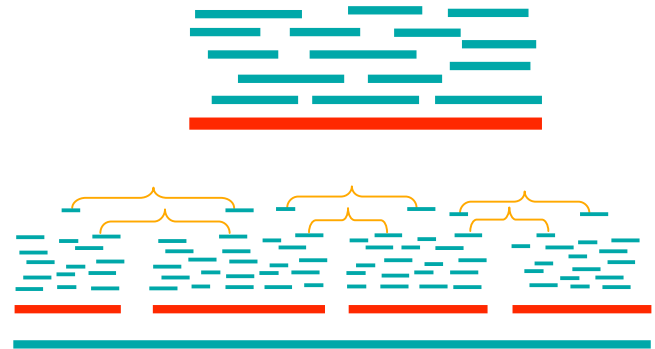
# Overlap-Layout-Consensus

**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..

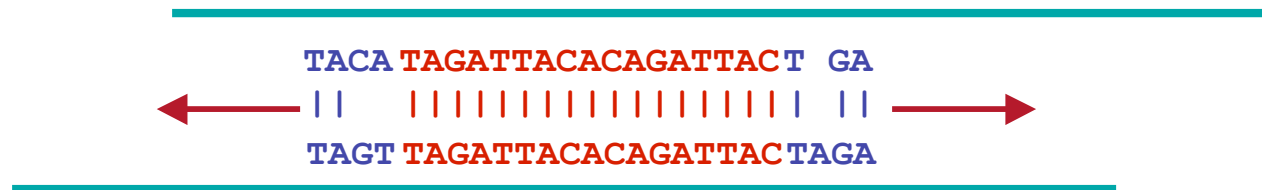


# Overlap

- 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 ( $k \sim 24$ )
- Find pairs of reads sharing a  $k$ -mer
- Extend to full alignment – throw away if not >95% similar

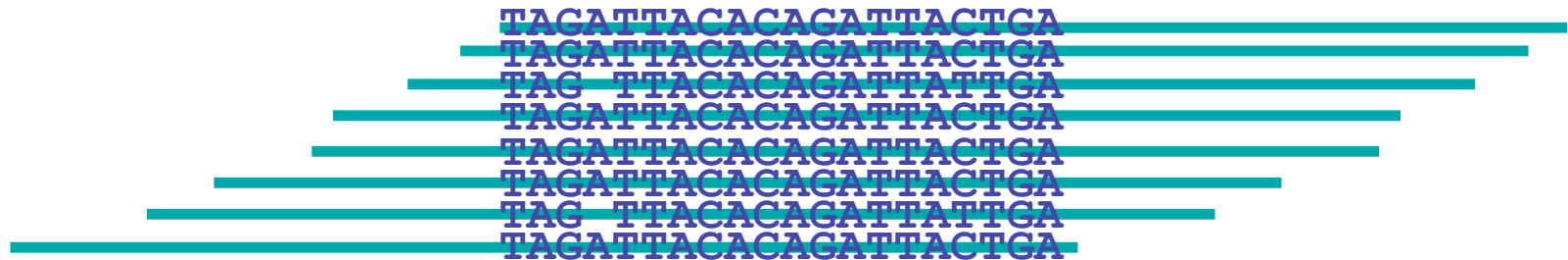


# Overlapping Reads and Repeats

- A  $k$ -mer that appears  $N$  times, initiates  $N^2$  comparisons
- For an *Alu* that appears  $10^6$  times  $\rightarrow 10^{12}$  comparisons – too much
- **Solution:**  
Discard all  $k$ -mers that appear more than  
 $t \times \text{Coverage}$ , ( $t \sim 10$ )

# Finding Overlapping Reads

Create local multiple alignments from the overlapping reads

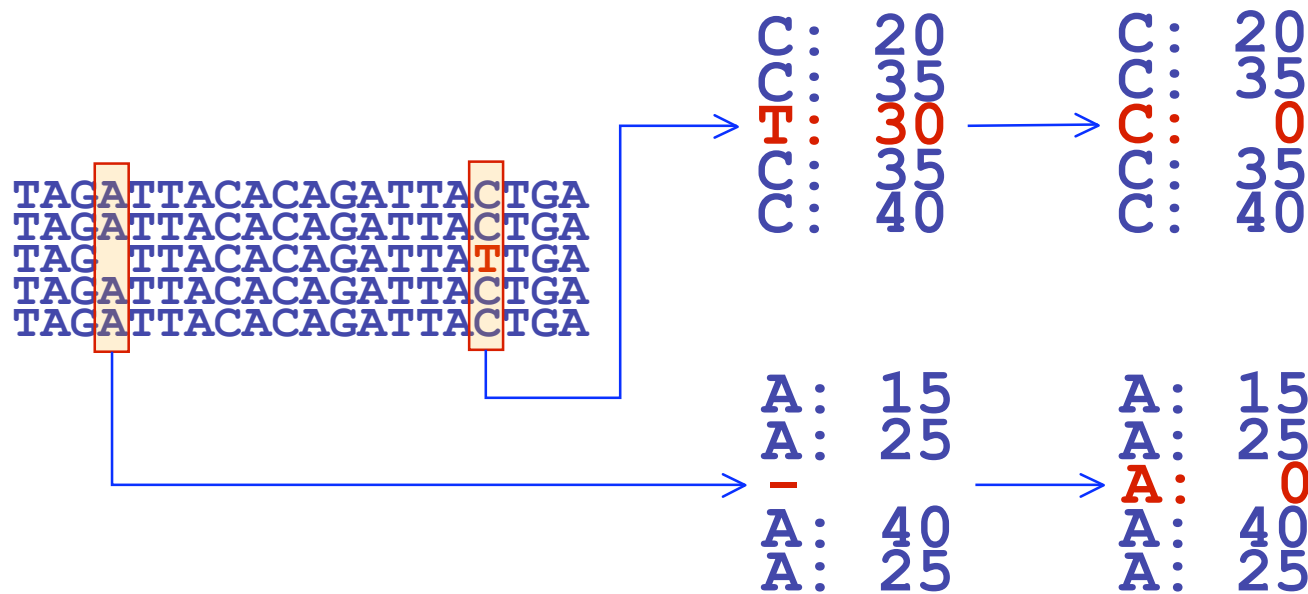


TAGATTACACAGATTACTGA  
TAGATTACACAGATTACTGA  
TAG TTACACAGATTATTGA  
TAGATTACACAGATTACTGA  
TAGATTACACAGATTACTGA  
TAGATTACACAGATTACTGA  
TAGATTACACAGATTACTGA  
TAG TTACACAGATTATTGA  
TAGATTACACAGATTACTGA

# Finding Overlapping Reads

(cont'd)

- Correct errors using multiple alignment

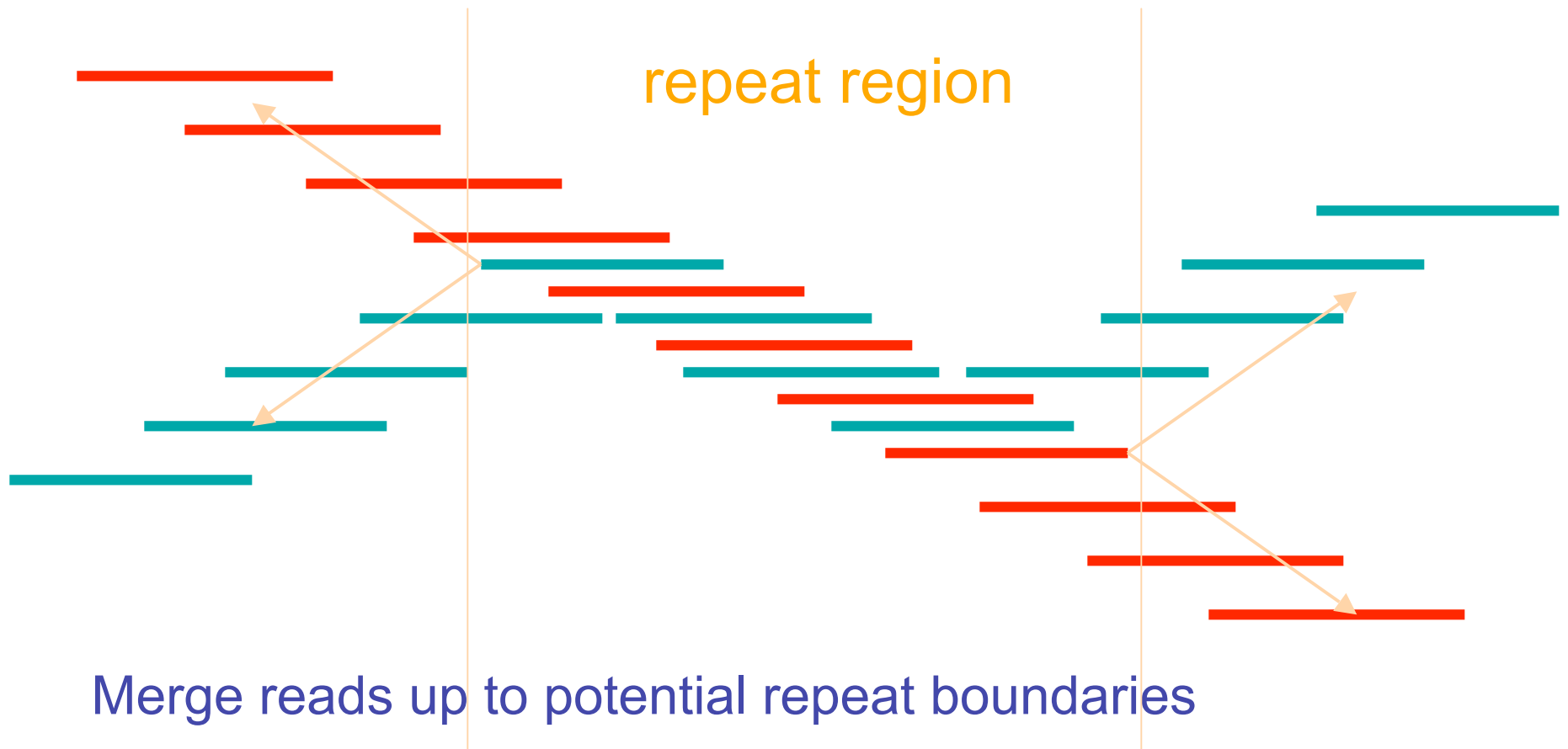


- Score alignments
- Accept alignments with good scores

# 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 alot more work at the finishing step

# Merge Reads into Contigs



# Repeats, Errors, and Contig Lengths

- Repeats shorter than read length are OK
- Repeats with more base pair differences 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



# Error Correction

## **Role of error correction:**

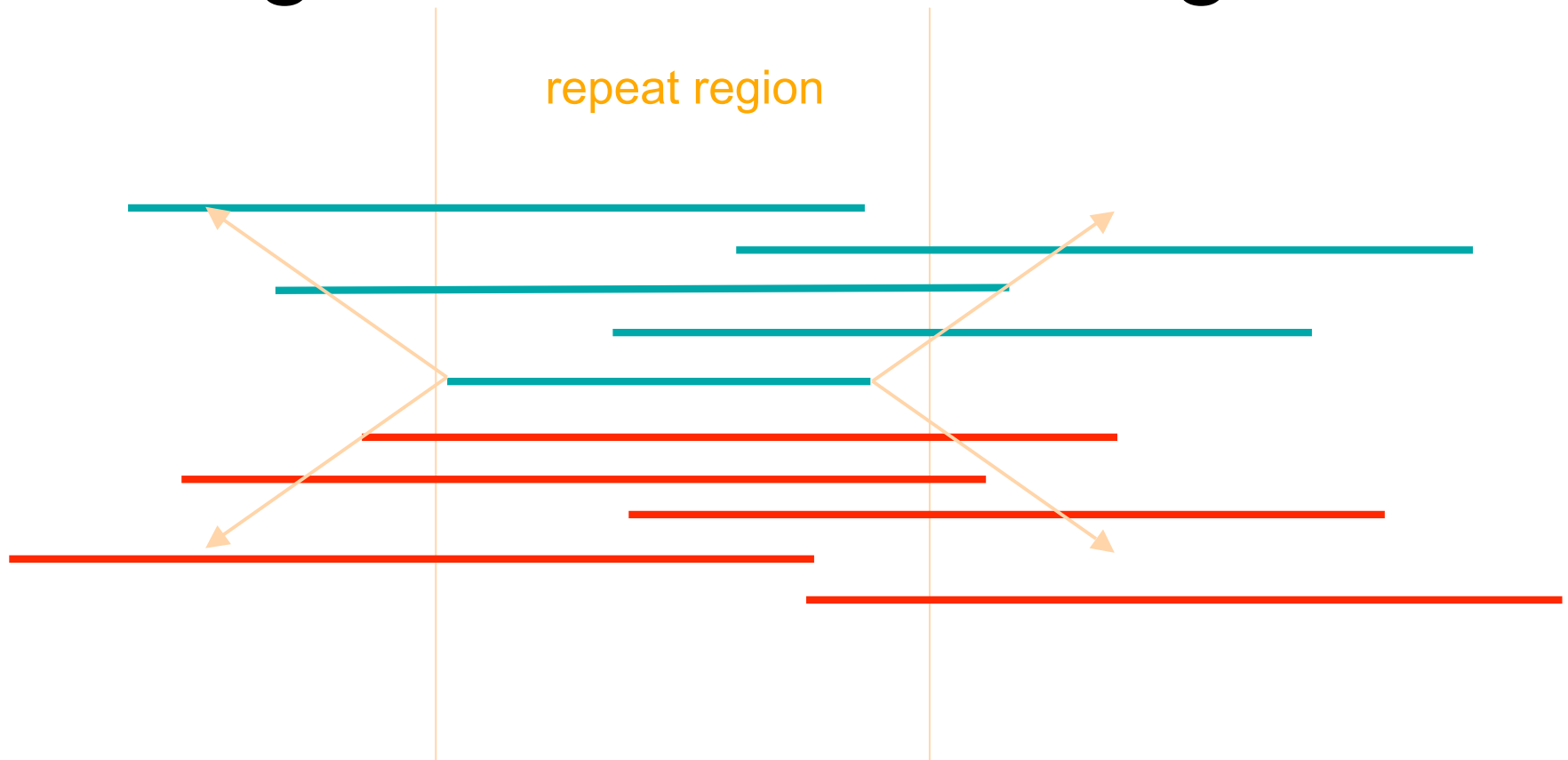
Discards ~90% of single-letter sequencing errors

decreases error rate

⇒ decreases effective repeat content

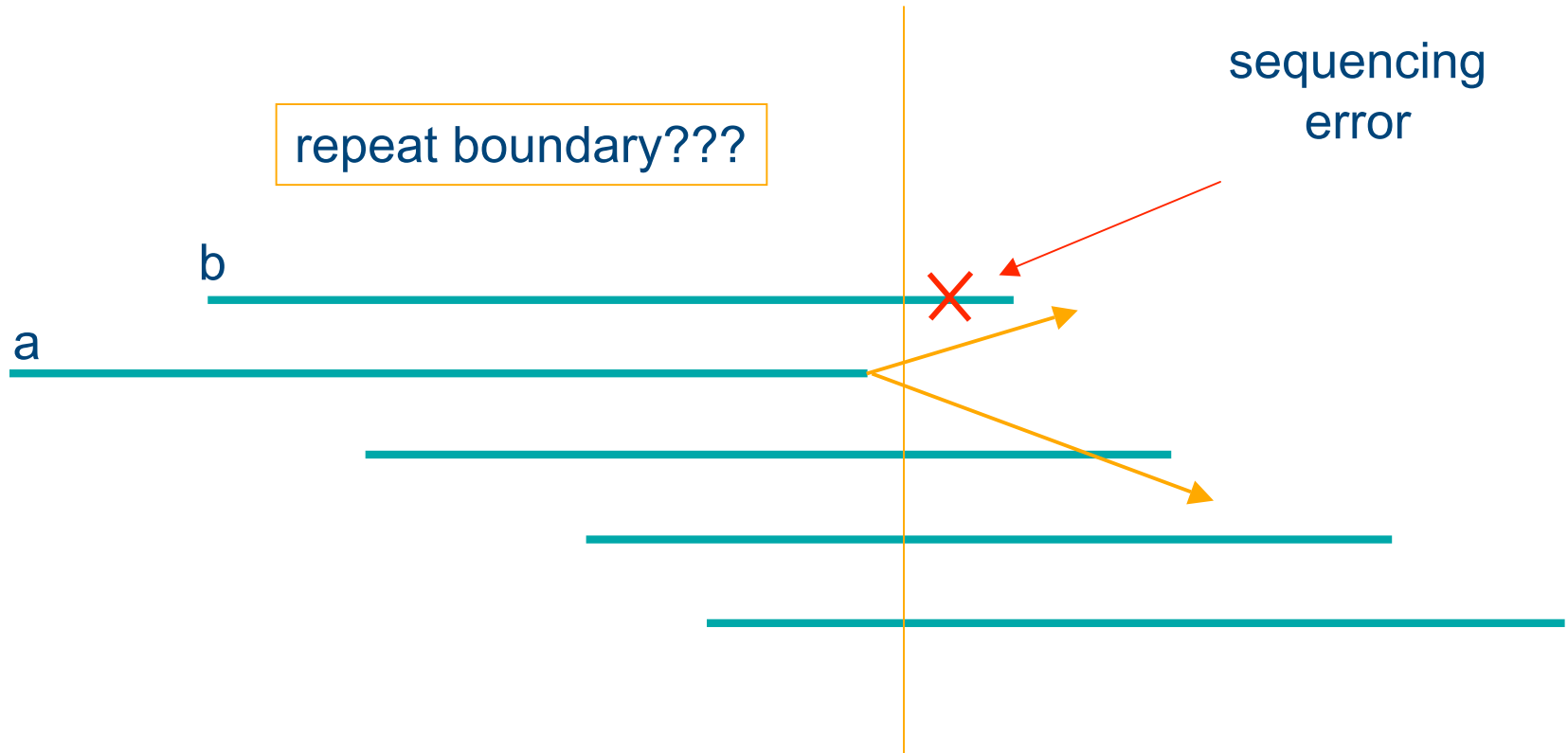
⇒ increases contig length

# Merge Reads into Contigs (cont'd)



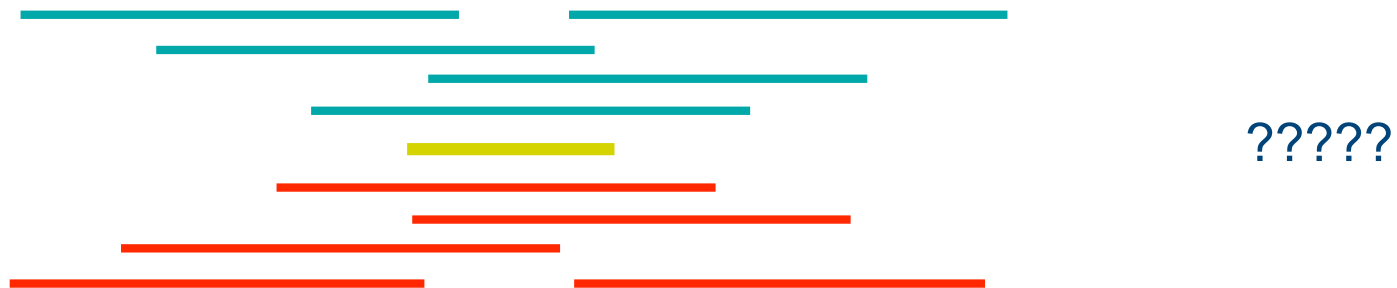
- Ignore non-maximal reads
- Merge only maximal reads into contigs

# Merge Reads into Contigs (cont'd)



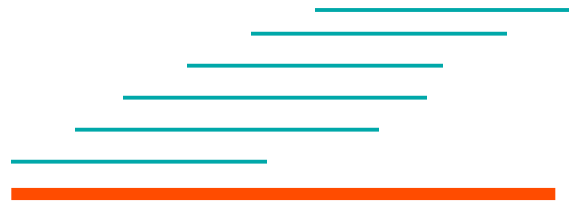
- Ignore “hanging” reads, when detecting repeat boundaries

# Merge Reads into Contigs (cont'd)



- Insert non-maximal reads whenever unambiguous

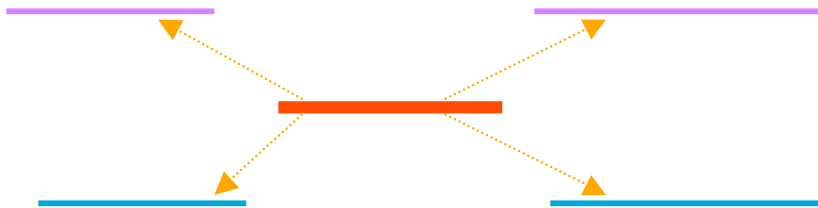
# Link Contigs into Supercontigs



Normal density



Too dense:  
Overcollapsed?



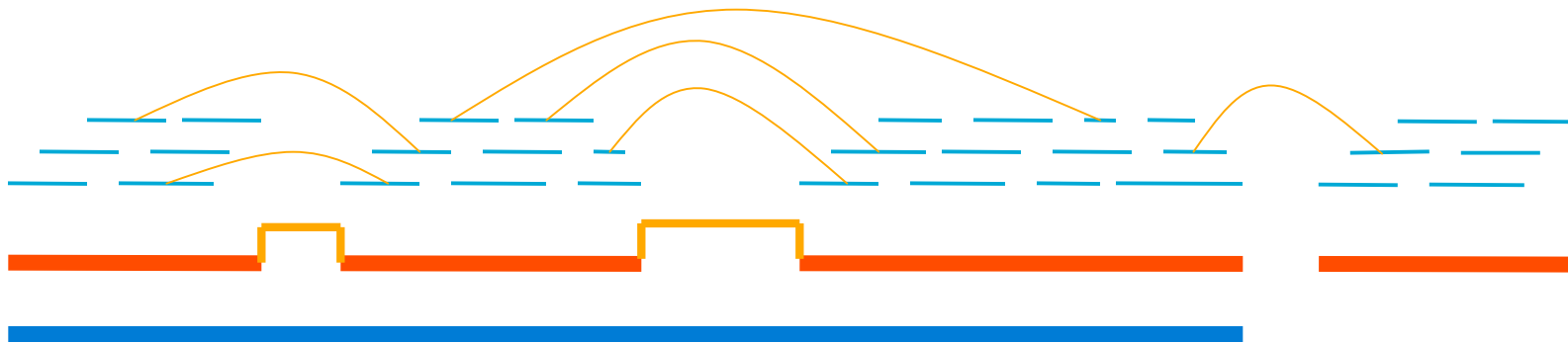
Inconsistent links:  
Overcollapsed?

# Link Contigs into Supercontigs

(cont'd)

Find all links between unique contigs

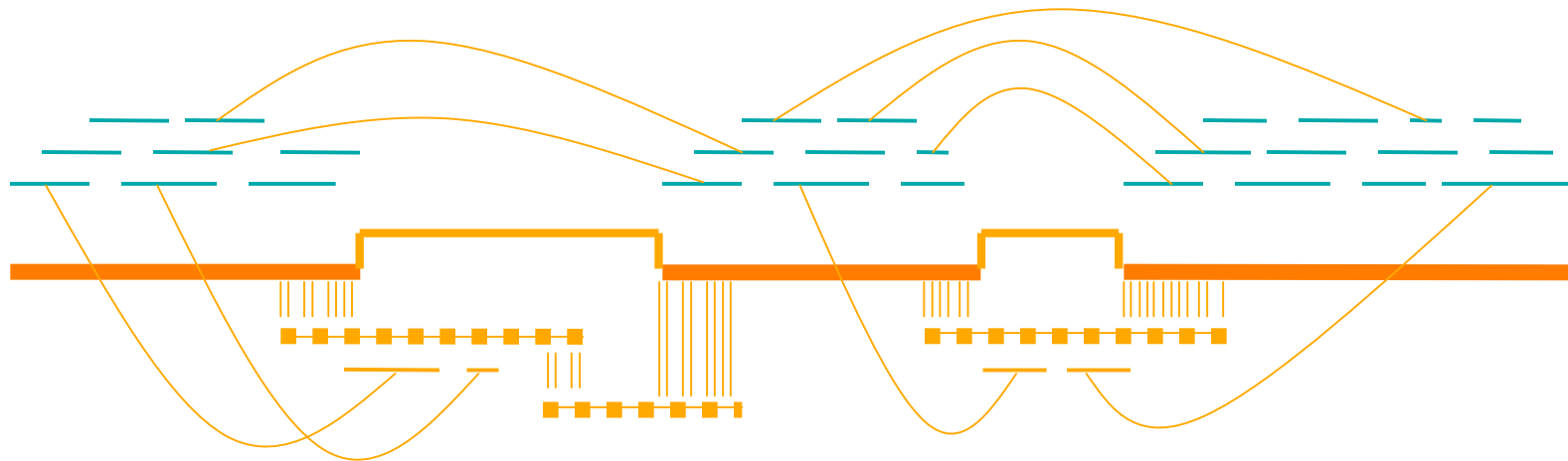
Connect contigs incrementally, if  $\geq 2$  links



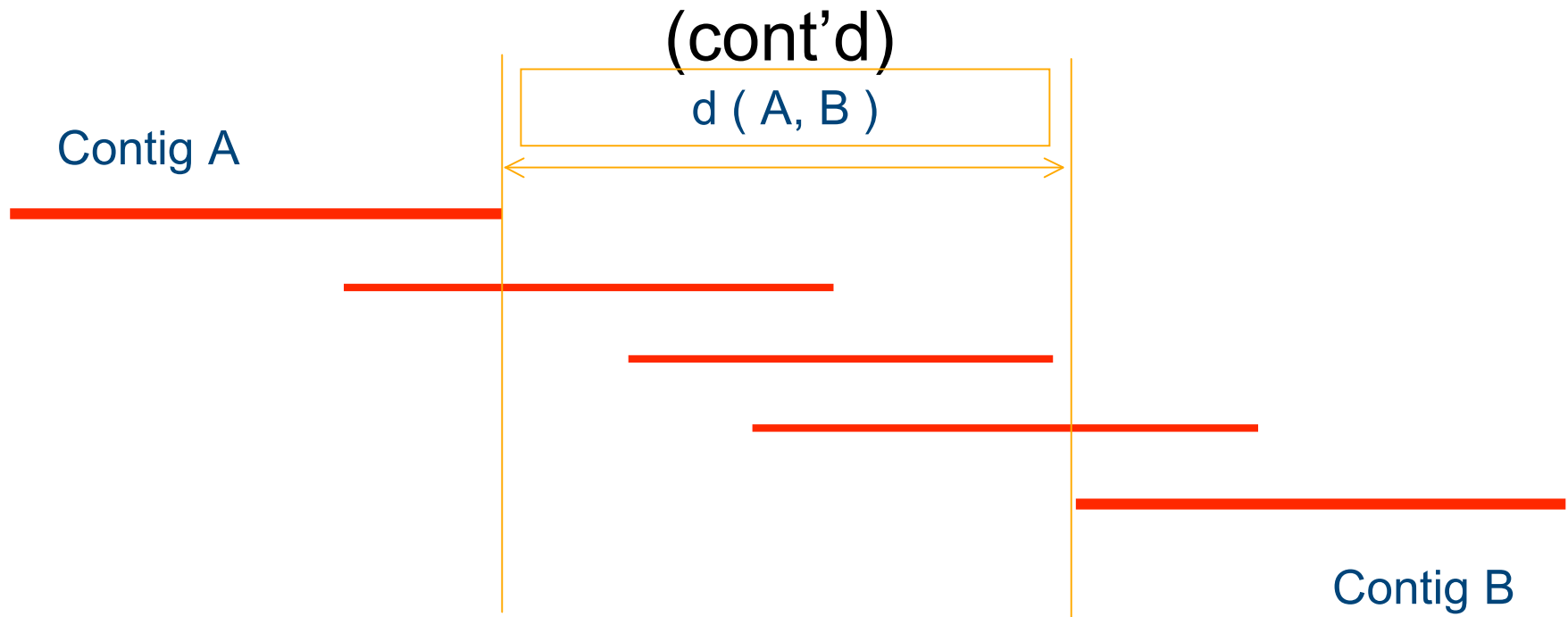
# Link Contigs into Supercontigs

(cont'd)

Fill gaps in supercontigs with paths of overcollapsed contigs



# Link Contigs into Supercontigs



Define  $G = ( V, E )$

$V := \text{contigs}$

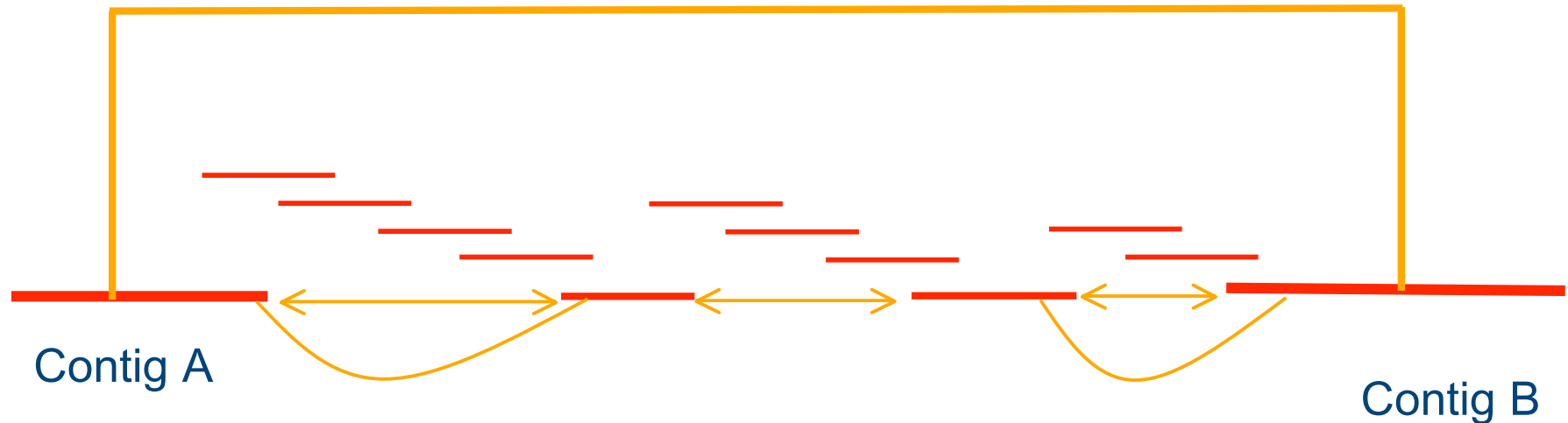
$E := ( A, B ) \text{ such that } d( A, B ) < C$

Reason to do so: Efficiency; full shortest paths cannot be computed



# Link Contigs into Supercontigs

(cont'd)



Define T: contigs linked to either A or B


Fill gap between A and B if there is a path in G passing only from contigs in T

# 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 TTACACAGATTATTTGACTTCATGGCGTAA CTA
TAGATTACACAGATTACTGACTTGATGGCGTAA CTA
TAGATTACACAGATTACTGACTTGATGGGGTAA CTA
```



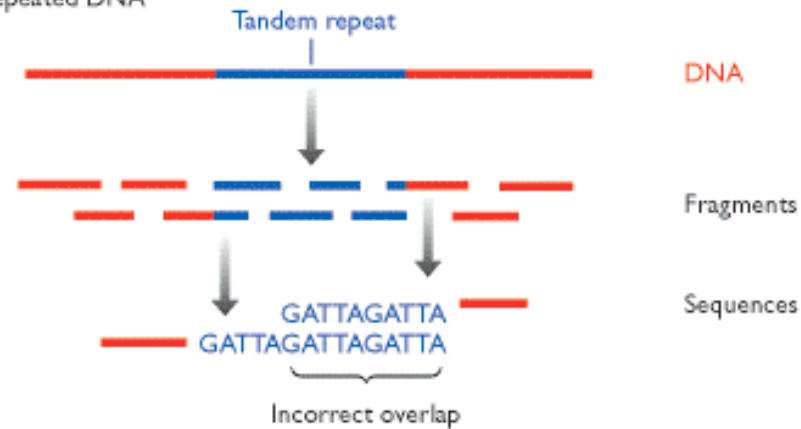
```
TAGATTACACAGATTACTGACTTGATGGCGTAA CTA
```

Derive **multiple alignment** from pairwise read alignments

Derive each consensus base by weighted voting

# Problems with the shotgun approach

(A) Problems with tandemly repeated DNA



(B) Problems with genome-wide repeats

