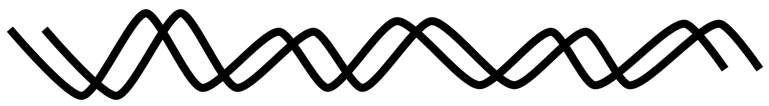
Genome Sequencing & Assembly

Slides by Carl Kingsford

Genome Sequencing

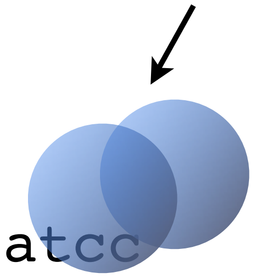


```
ACCGTCCAATTGG...
TGGCAGGTTAACC...
```

E.g. human: 3 billion bases split into 23 chromosomes

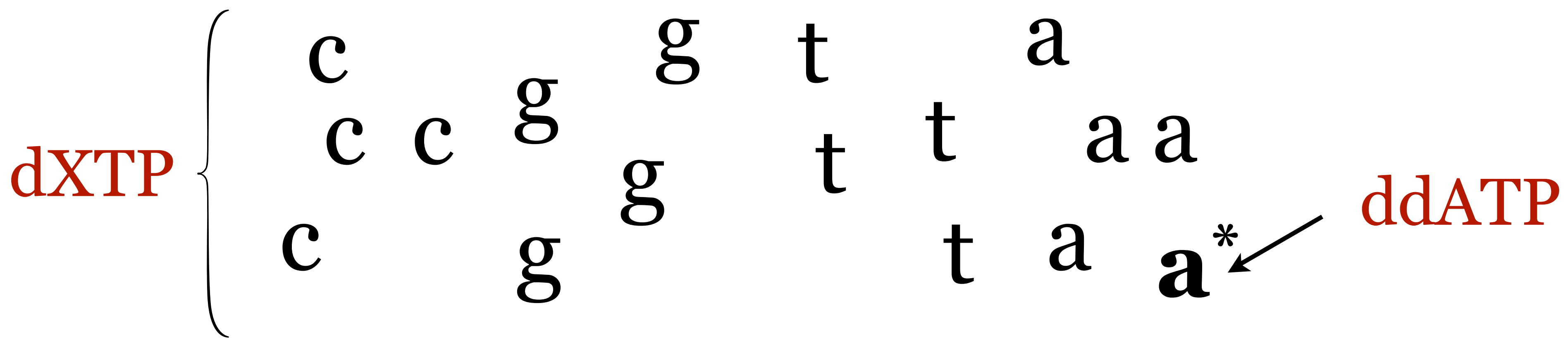
Main tool of traditional sequencing: DNA Synthesis

DNA polymerase: enzyme that will grow a complementary DNA strand.



```
gacgatcggtttatcc
ctgctagccaaataggctaatactacgga
```

Sanger Sequencing: Finding the As



```
gacgatcgg tttA*                        gacgatcgg tttAtccgAttAtgA*
ctgctagcc aaaTaggcTaaTacTacgga         ctgctagcc aaaTaggcTaaTacTacgga


gacgatcgg tttAtccgA*                   gacgatcgg tttAtccgAttA*
ctgctagcc aaaTaggcTaaTacTacgga         ctgctagcc aaaTaggcTaaTacTacgga


gacgatcgg tttA*                        gacgatcgg tttAtccgAttAtgA*
ctgctagcc aaaTaggcTaaTacTacgga         ctgctagcc aaaTaggcTaaTacTacgga


gacgatcgg tttAtccgAttA*                gacgatcgg tttAtccgA*
ctgctagcc aaaTaggcTaaTacTacgga         ctgctagcc aaaTaggcTaaTacTacgga
```

Size → Sequence

gacgatcggttt**A***

gacgatcggttt**A***

gacgatcggttt**A**tccg**A***

gacgatcggttt**A**tccg**A***

gacgatcggttt**A**tccg**A**tt**A***

gacgatcggttt**A**tccg**A**tt**A***

gacgatcggttt**A**tccg**A**tt**A**tg**A***

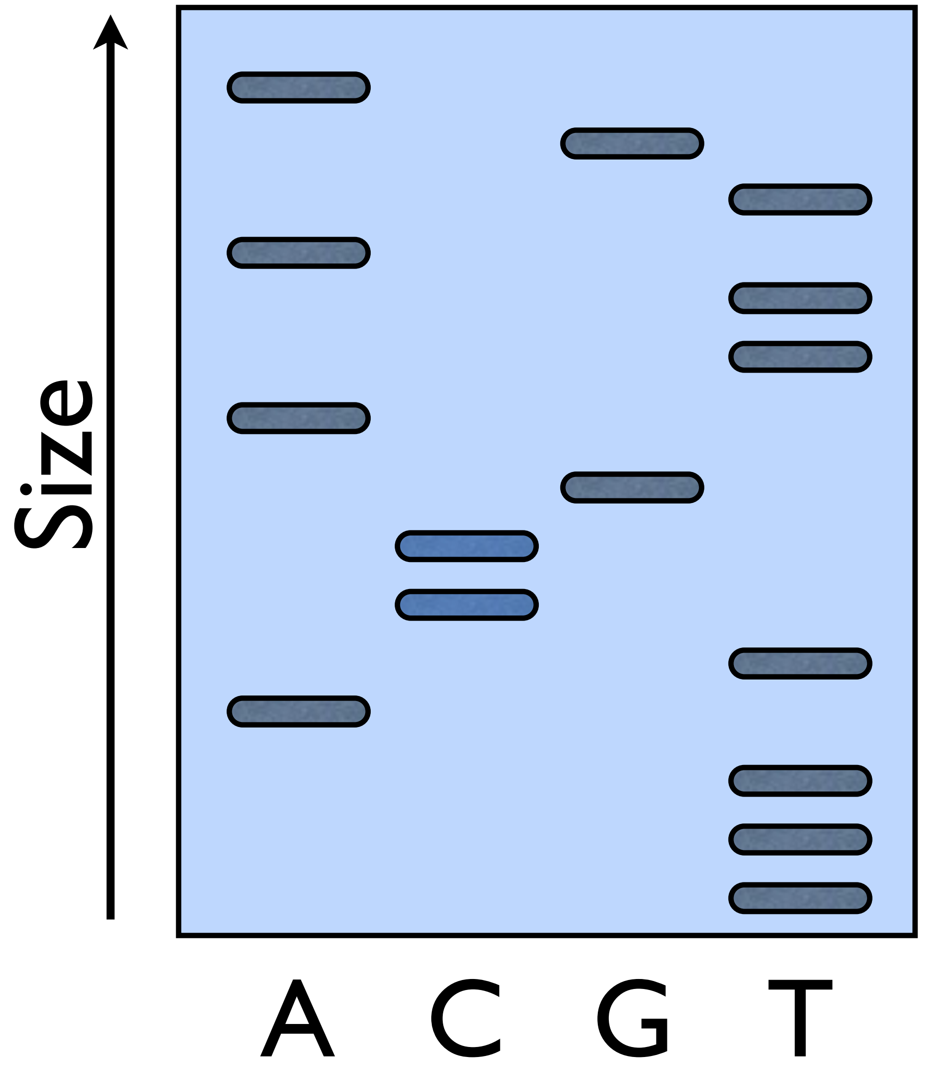
gacgatcggttt**A**tccg**A**tt**A**tg**A***

gacgatcggtttatccgattat**G***

gacgatcggtttatcc**G***

gacgatcggtttat**C***

gacgatcggtttatc**C***



Size → Sequence

gacgatcggtttA*

gacgatcggtttA*

gacgatcggtttAtccgA*

gacgatcggtttAtccgA*

gacgatcggtttAtccgAttA*

gacgatcggtttAtccgAttA*

gacgatcggtttAtccgAttAtgA*

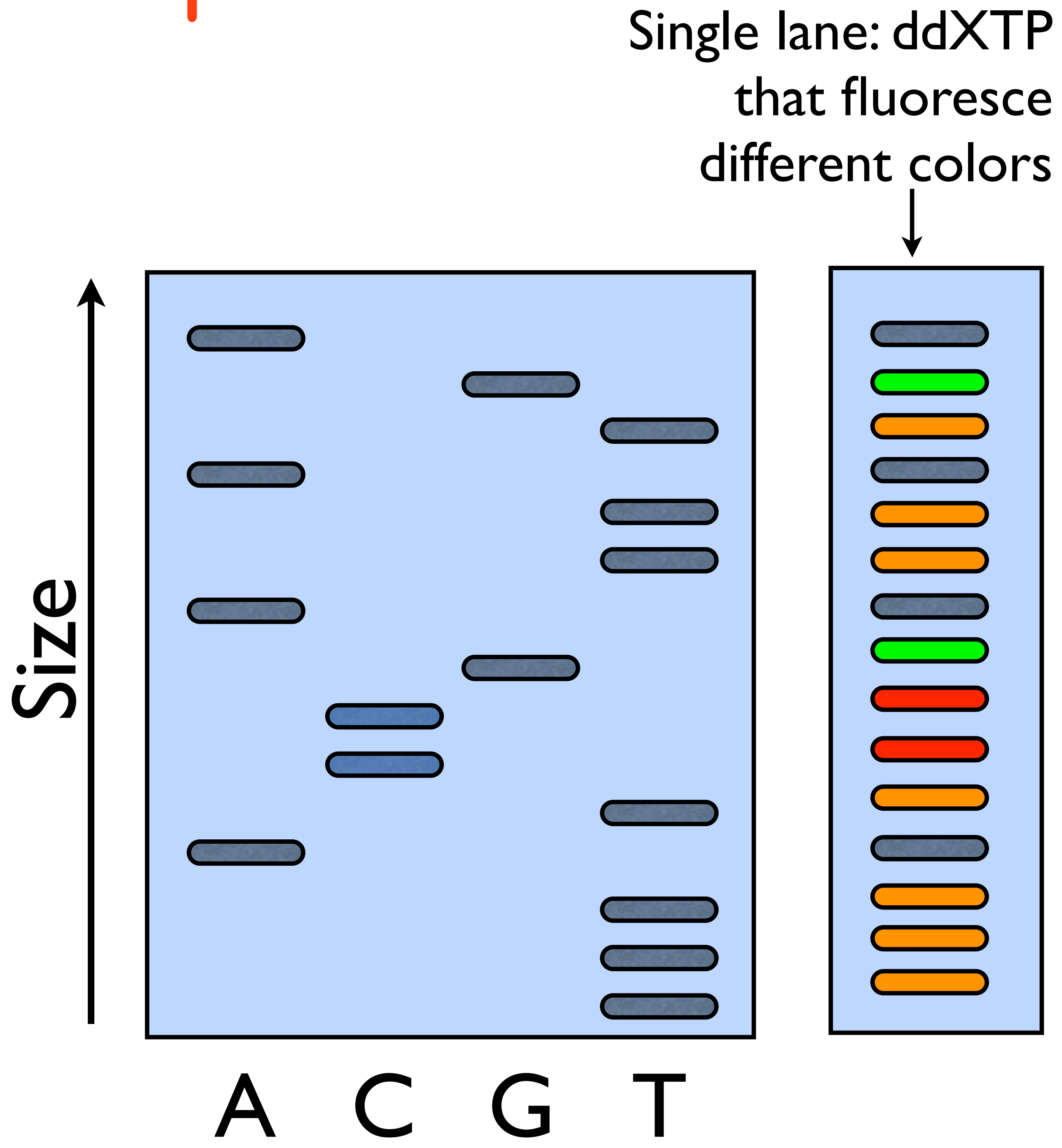
gacgatcggtttAtccgAttAtgA*

gacgatcggtttatccgattatG*

gacgatcggtttatccG*

gacgatcggtttatC*

gacgatcggtttatcC*



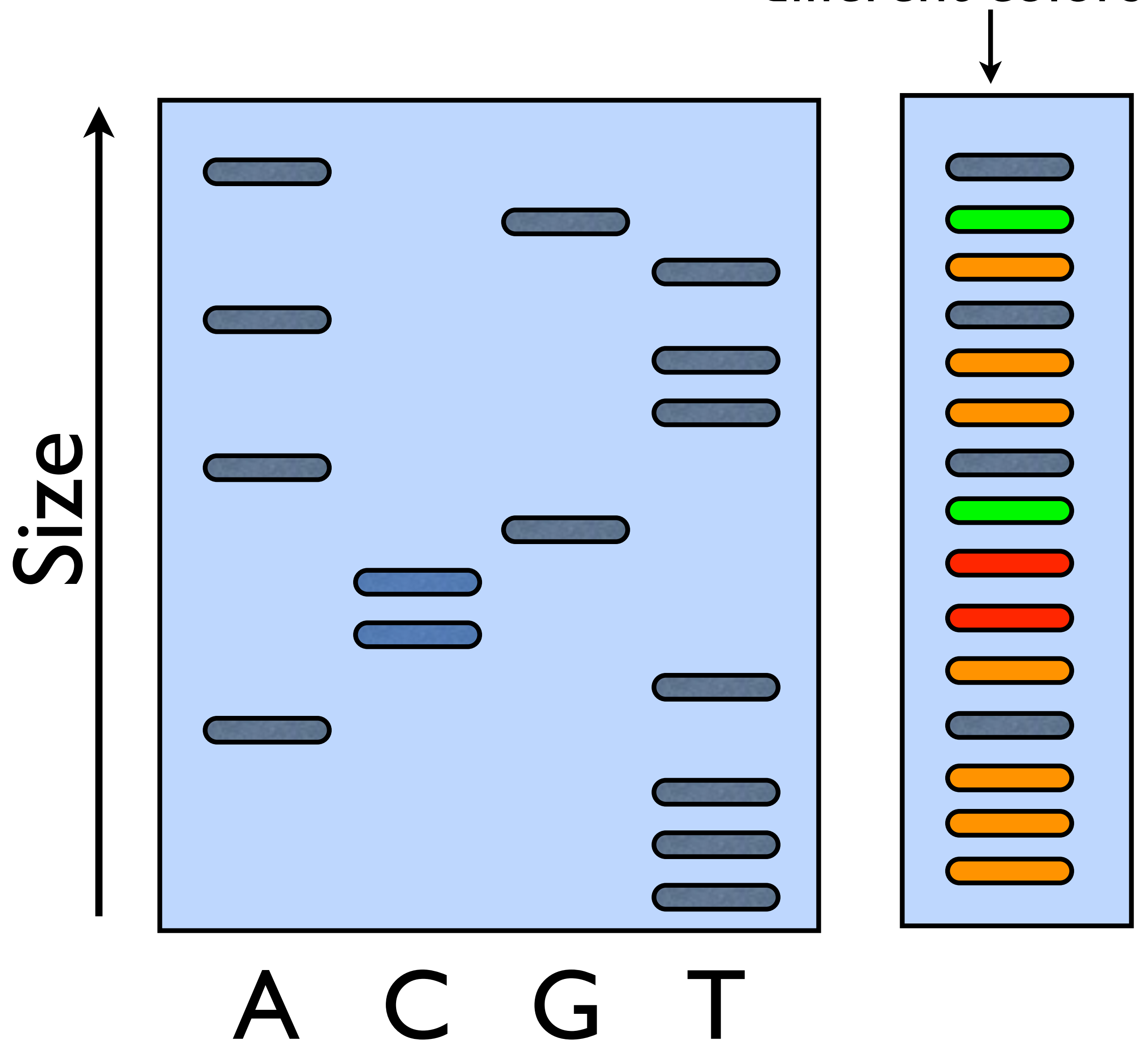
Size → Sequence

Single lane: ddXTP that fluoresce different colors

gacgatcggttt**A***
gacgatcggttt**A***
gacgatcggttt**A**tccg**A***
gacgatcggttt**A**tccg**A***
gacgatcggttt**A**tccg**A**tt**A***
gacgatcggttt**A**tccg**A**tt**A***
gacgatcggttt**A**tccg**A**tt**A**tg**A***
gacgatcggttt**A**tccg**A**tt**A**tg**A***

gacgatcggtttatccgattat**G***
gacgatcggtttatcc**G***

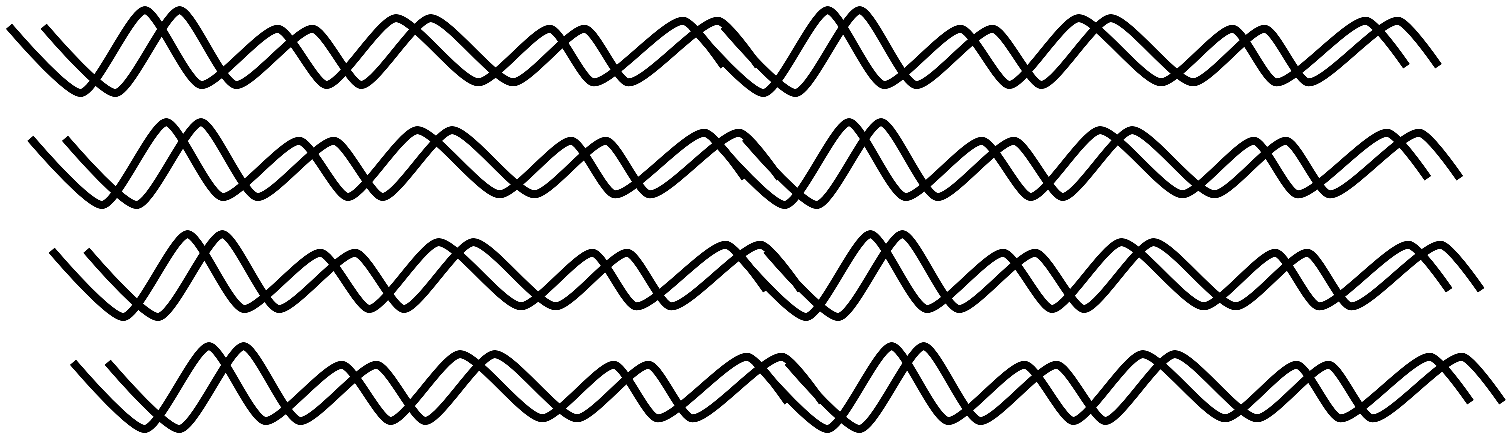
gacgatcggtttat**C***
gacgatcggtttatc**C***



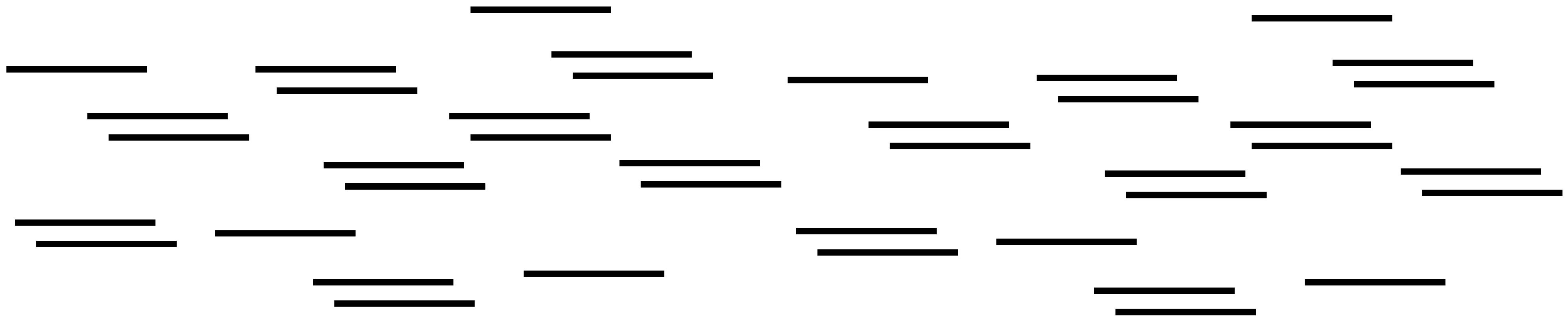
Main problem: larger fragments take a long time to be sorted correctly (or don't sort correctly ever) → 800-1000 letter maximum

Shotgun Sequencing

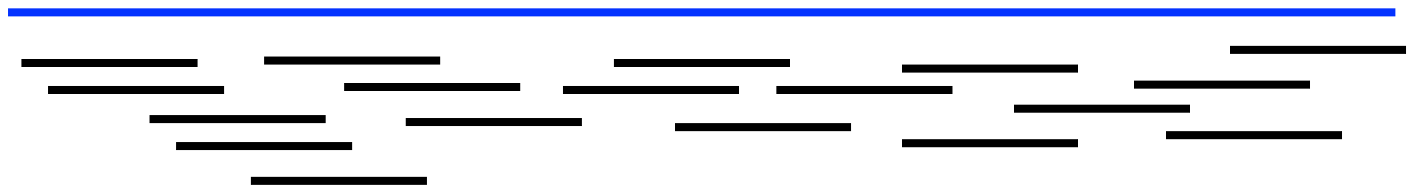
Many copies of the DNA



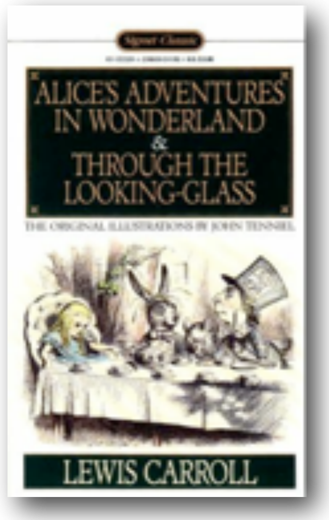
Shear it, randomly breaking them into many small pieces, read ends of each:



Assemble into original genome:



We can only read ~ 1000 characters at a time from a random place:


ALICE'S ADVENTURES IN WONDERLAND & THROUGH THE LOOKING-GLASS
LEWIS CARROLL

```
good-natured, she thought: still
                         when it saw Alice. It looked
          ought to be treated
                            good-natured, she thought, still
                          Cat only
                        a greet many
                                                  It looked good-
 The Cat only grinned when it saw Alice.
     be treated with respect.
                              still it had very long claws
          claws and a great many teeth, so she

                                          so she felt that it ought
```

Algorithms are needed to piece the story together.

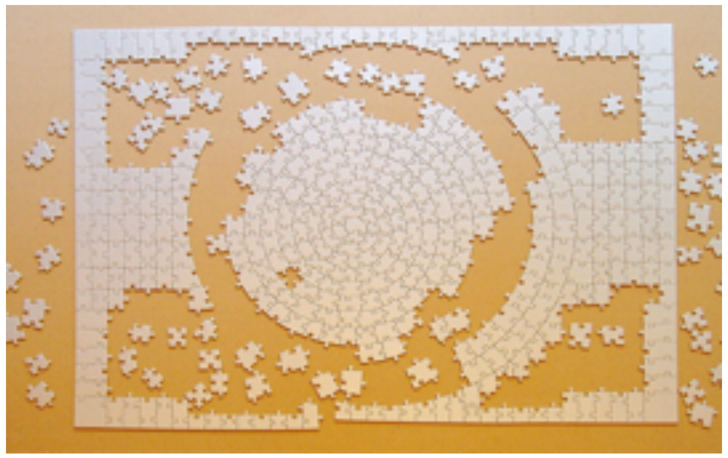
```
The Cat only grinned when it saw Alice.
    Cat only             when it saw Alice. It looked
                                            It looked good-

good-natured, she thought: still
good-natured, she thought, still
                           still it had very long claws

claws and a great many teeth, so she
          a greet many        so she felt that it ought

ought to be treated
         be treated with respect.
```

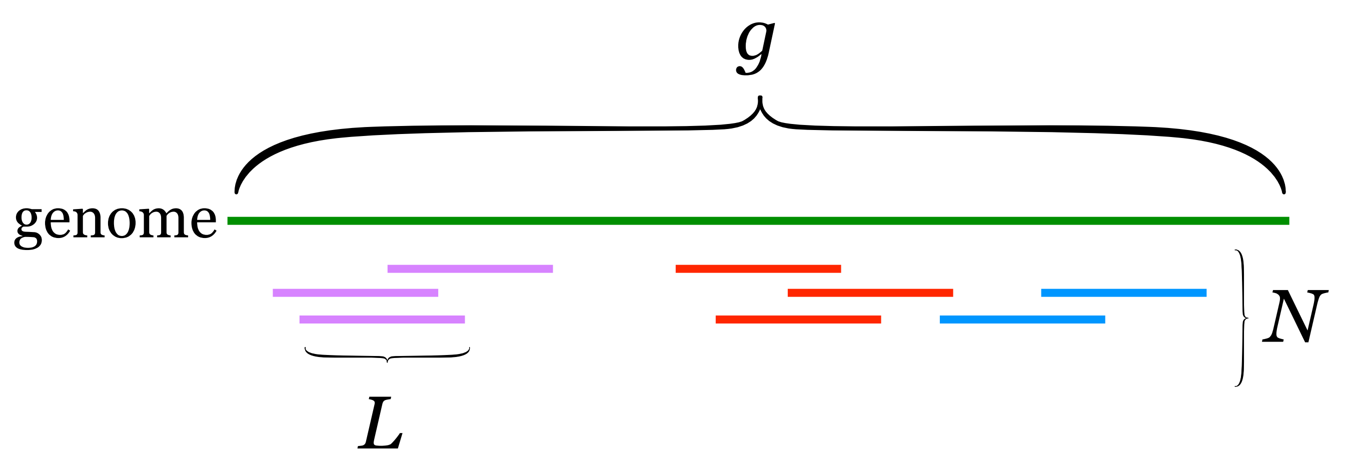
It's a jigsaw puzzle ...

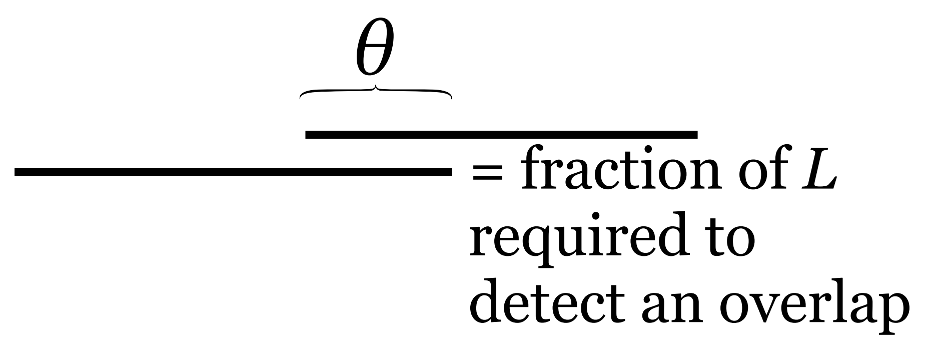


...except with 35 million pieces

Lander-Waterman Statistics

How many reads to we need to be sure we cover the whole genome?





An ***island*** is a contiguous group of reads that are connected by overlaps of length ≥ θL.
(Various colors above)

Want: Expression for expected # of islands given N, g, L, θ.

Expected # of Islands

$\lambda := N/g$ = probability a read starts at a given position (assuming random sampling)

Pr(k reads start in an interval of length x)

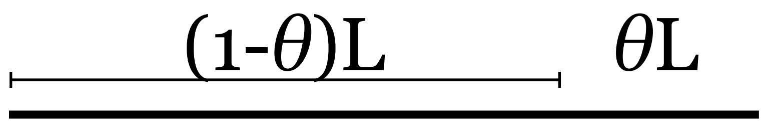
x trials, want k "successes," small probability λ of success

Expected # of successes = λx

Poisson approximation to binomial distribution:

$$\Pr(k \text{ reads in length } x) = e^{-\lambda x}\frac{(\lambda x)^k}{k!}$$

Expected # of islands = $N \times$ Pr(read is at rightmost end of island)



$$= N \times \Pr(0 \text{ reads start in } (1-\theta)L)$$

$$= Ne^{-\lambda(1-\theta)L}\frac{\lambda^0}{0!} \quad \text{(from above)}$$

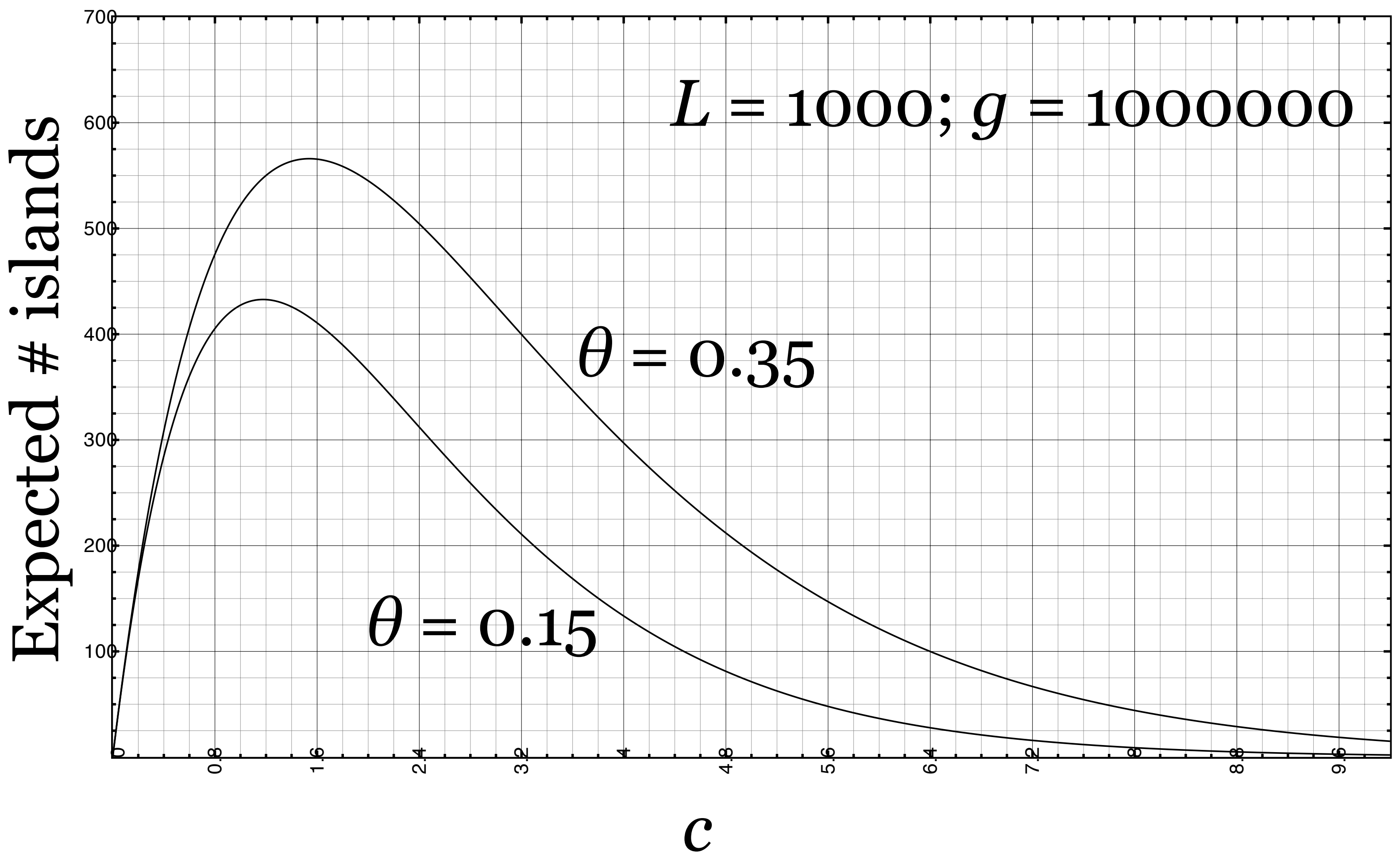
$$= Ne^{-\lambda(1-\theta)L}$$

$$= Ne^{-(1-\theta)LN/g}$$ $\leftarrow LN/g$ is called the **coverage** c.

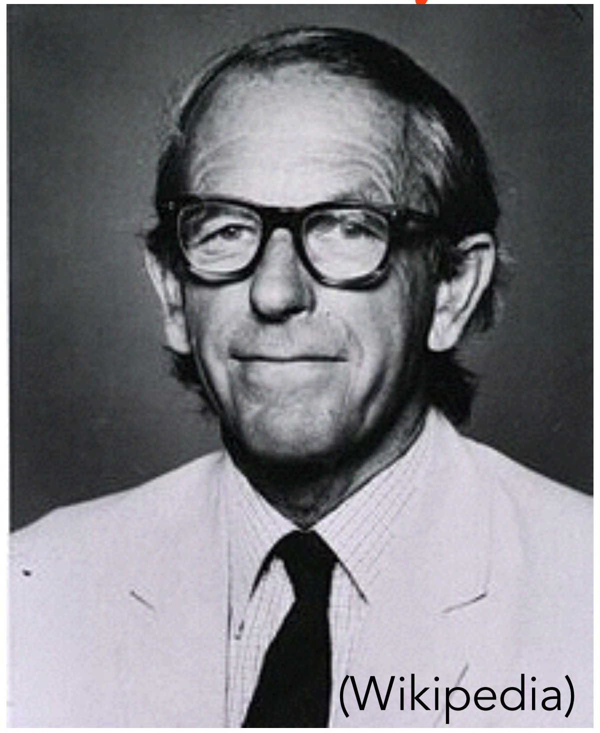
Expected # of Islands, 2

Rewrite to depend more directly on the things we can control: c and θ

$$\text{Expected \# of islands} = Ne^{-(1-\theta)LN/g}$$
$$= Ne^{-(1-\theta)c}$$
$$= \frac{L/g}{L/g} Ne^{-(1-\theta)c}$$
$$= \frac{g}{L} ce^{-(1-\theta)c}$$



Shotgun Sequencing Summary



- "Sanger" sequencing widely used up through 2006 or 2007, including for the human genome project.
- Won Sanger his second Nobel prize.
- Lander-Waterman statistics estimate the number of islands you will get for a given coverage.
    - Used as a way to guess how much sequencing you need to do for a given technology and genome size.
    - Often hard in practice to guess the genome size g before you've sequenced it.

Genome Assembly Paradigms

Shortest Common Superstring

Def. Given strings $s_1, \ldots, s_n$, find the shortest string T such that each s_i is a sub**string** of T.

- NP-hard (contrast with case when requiring s_i to be sub**sequences** of T)

- Approximation algorithms exist with factors: 4, 3, 2.89, 2.75, 2.67, 2.596, 2.5, ...

- Basic greedy method: find pair of strings that overlap the best, merge them, repeat (4 approximation):

Given match, mismatch, gap costs, how can we compute the score of the best overlap?

Overlap Alignment

Score of an optimal alignment between a suffix of Y and a prefix of X

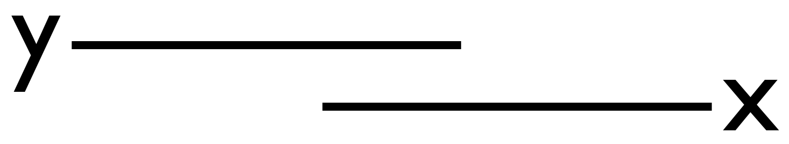
y

C	9	0												
A	8	0												
G	7	0												
T	6	0												
T	5	0												
G	4	0												
C	3	0												
A	2	0												
A	1	0												
	0	0	1g	2g	3g	4g	5g	6g	7g	8g	9g	10g	11g	12g
		0	1	2	3	4	5	6	7	8	9	10	11	12
			A	A	G	G	T	A	T	G	A	A	T	C

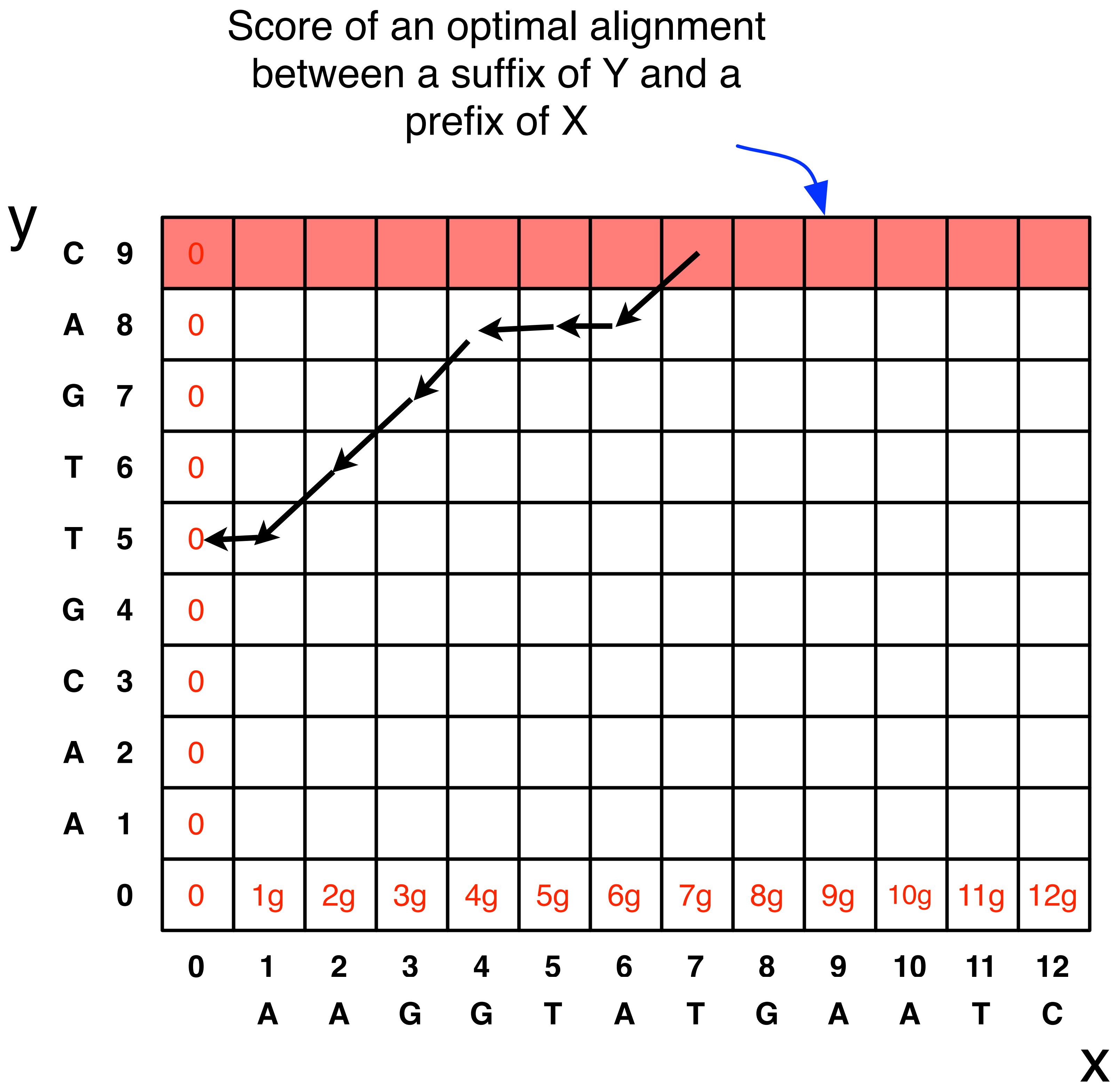
x

- Initialize first column to 0s

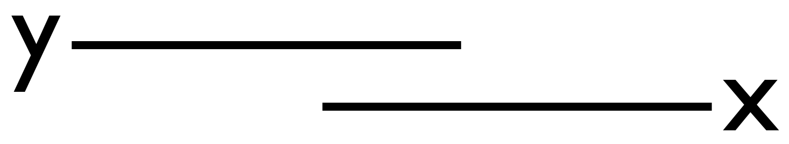
- Answer is maximum score in top row (traceback starts from there until it falls off left side)



Overlap Alignment

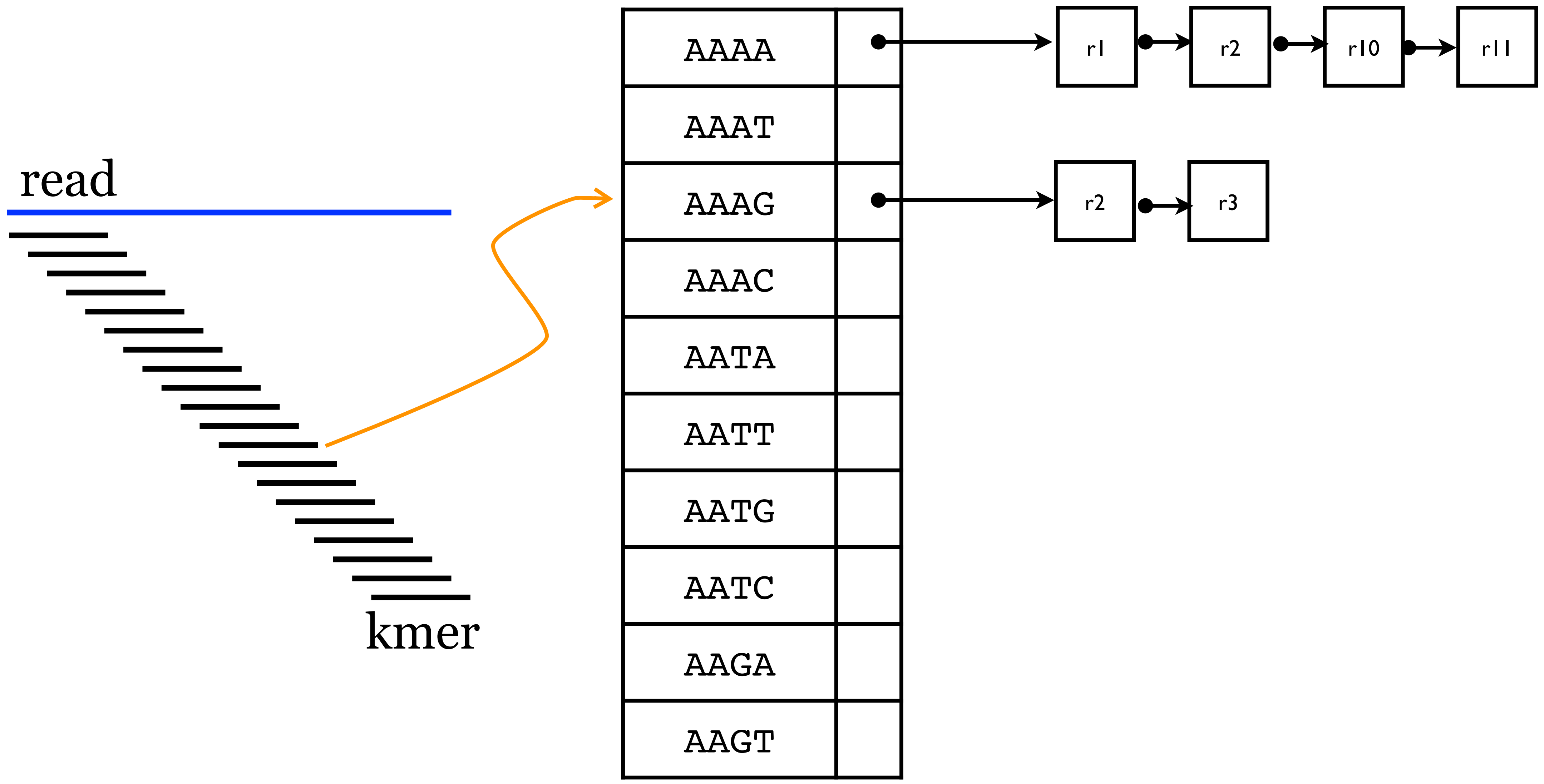

Score of an optimal alignment between a suffix of Y and a prefix of X
y
C 9
A 8
G 7
T 6
T 5
G 4
C 3
A 2
A 1
0
0
0 1g 2g 3g 4g 5g 6g 7g 8g 9g 10g 11g 12g
0 1 2 3 4 5 6 7 8 9 10 11 12
A A G G T A T G A A T C
x

- Initialize first column to 0s
- Answer is maximum score in top row (traceback starts from there until it falls off left side)



K-mer Hashing

Only compute overlap alignment between reads that share a kmer:



The problem with Shortest Common Superstring (SCS): Repeats

Truth:

```
AAAAAAAAAAAAAAAAAAAA
AAAAA
 AAAAA
  AAAAA
   AAAAA
    AAAAA
     AAAAA
        .
         .
          .
```

SCS:

```
AAAAA
AAAAA
AAAAA
AAAAA
AAAAA
```

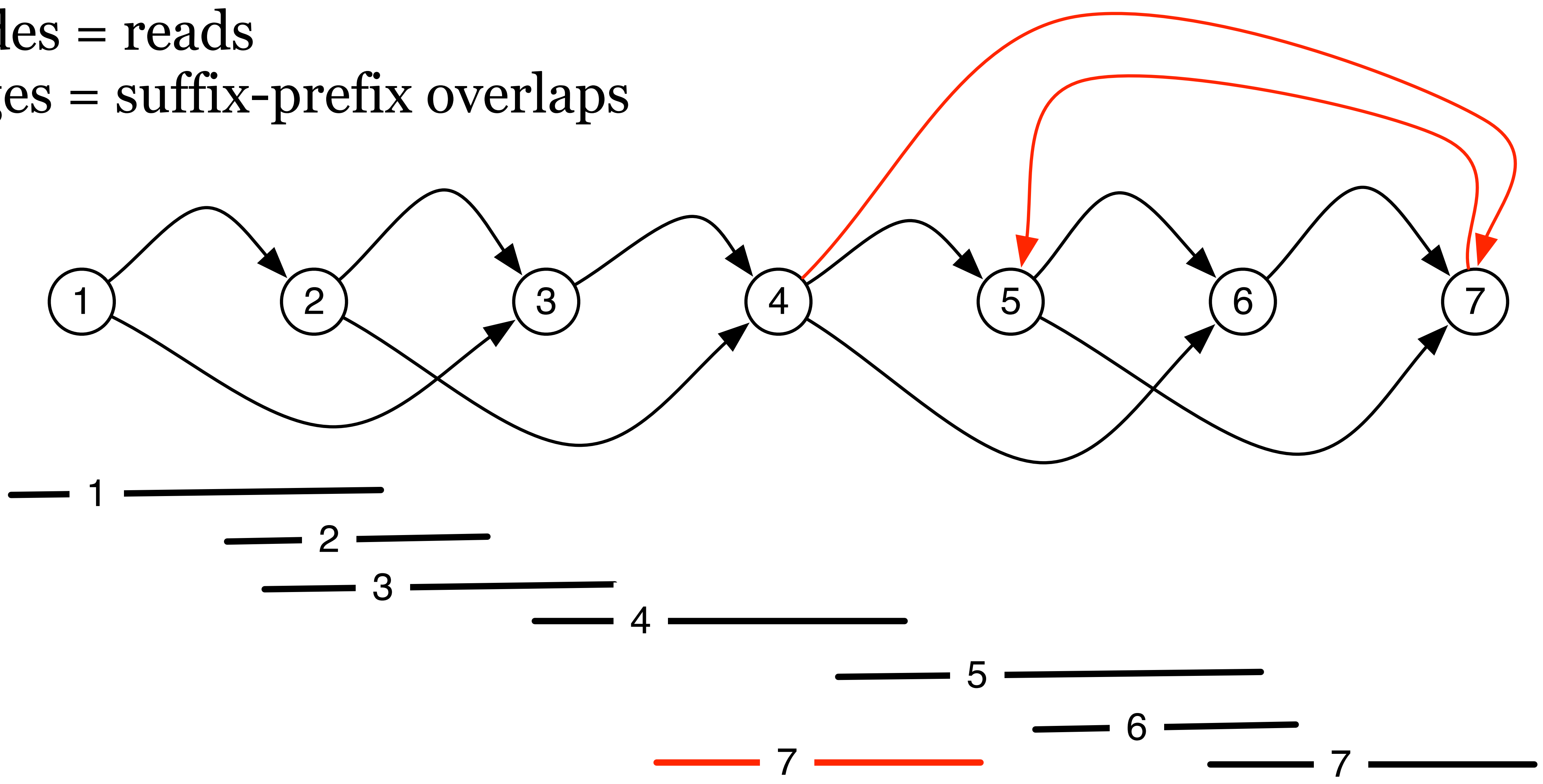
More complex example:

```
ACCGCCT ACCGCCT ACCGCCT
```

2 or 3 copies?

Overlap Consensus

Overlap graph:
- Nodes = reads
- Edges = suffix-prefix overlaps



Given overlap graph, how can we find a good candidate assembly?

Idea: Every read must be used in exactly one place in the genome.

Assembly by Traveling Salesman

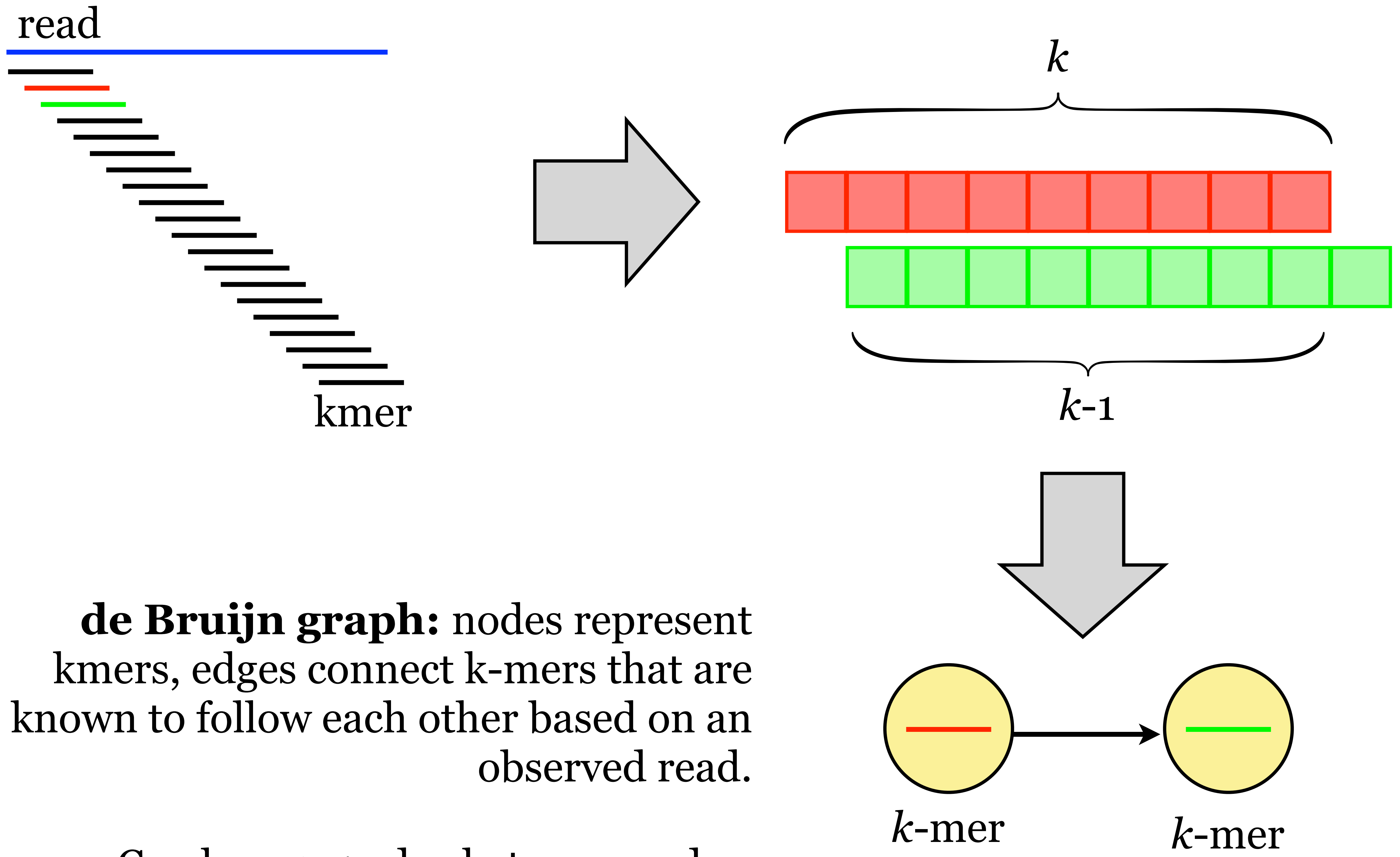
Traveling Salesman Problem: Find a path that visits every node in the graph exactly once.

Optimal Traveling Salesman path of 24,978 cities in Sweden

(Applegate et al, 2004, www.tsp.gatech.edu/sweden/index.html).

Assembly via <ins>Eulerian</ins> Path

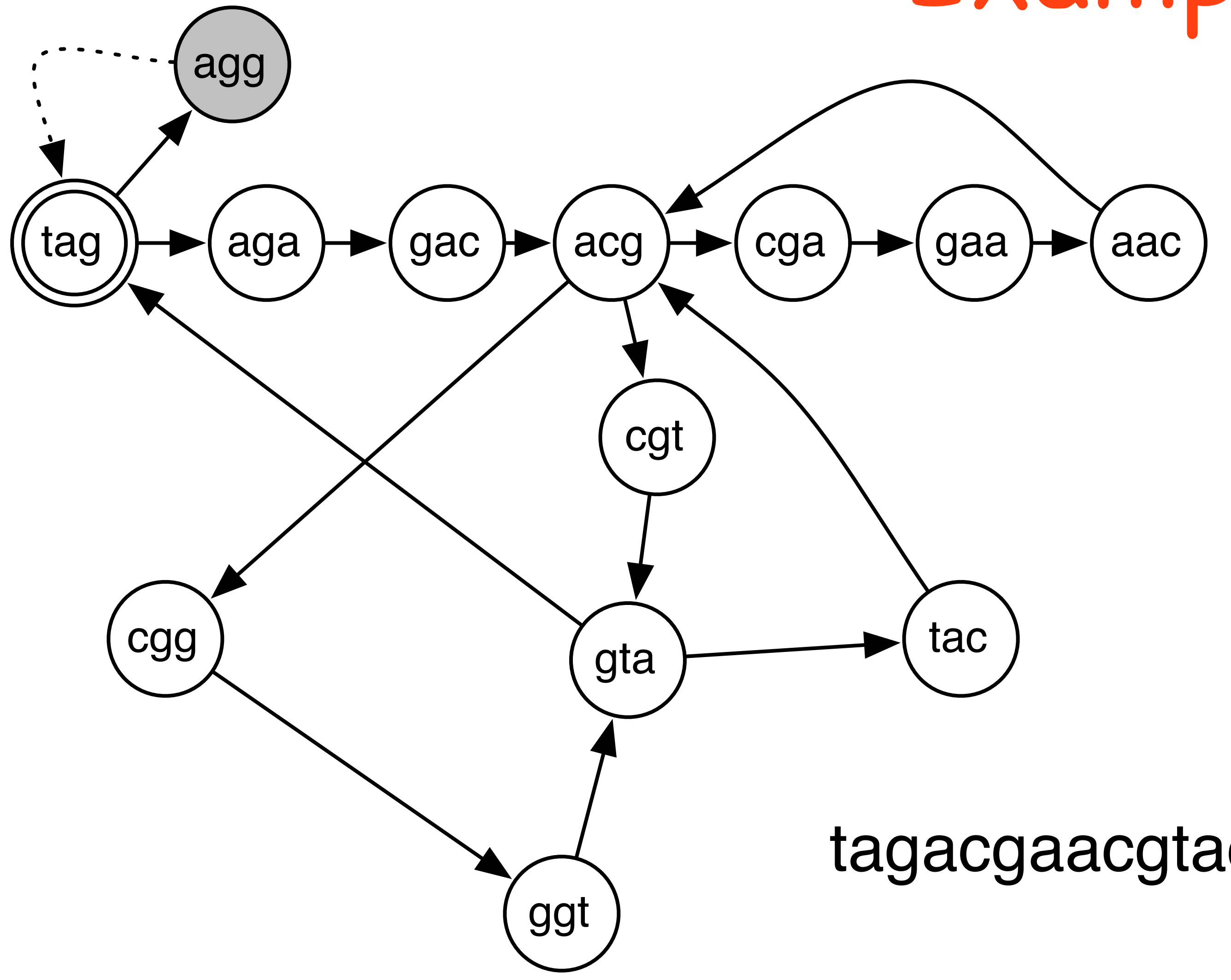
de Bruijn graph



de Bruijn graph: nodes represent kmers, edges connect k-mers that are known to follow each other based on an observed read.

Can have > 1 edge between nodes.

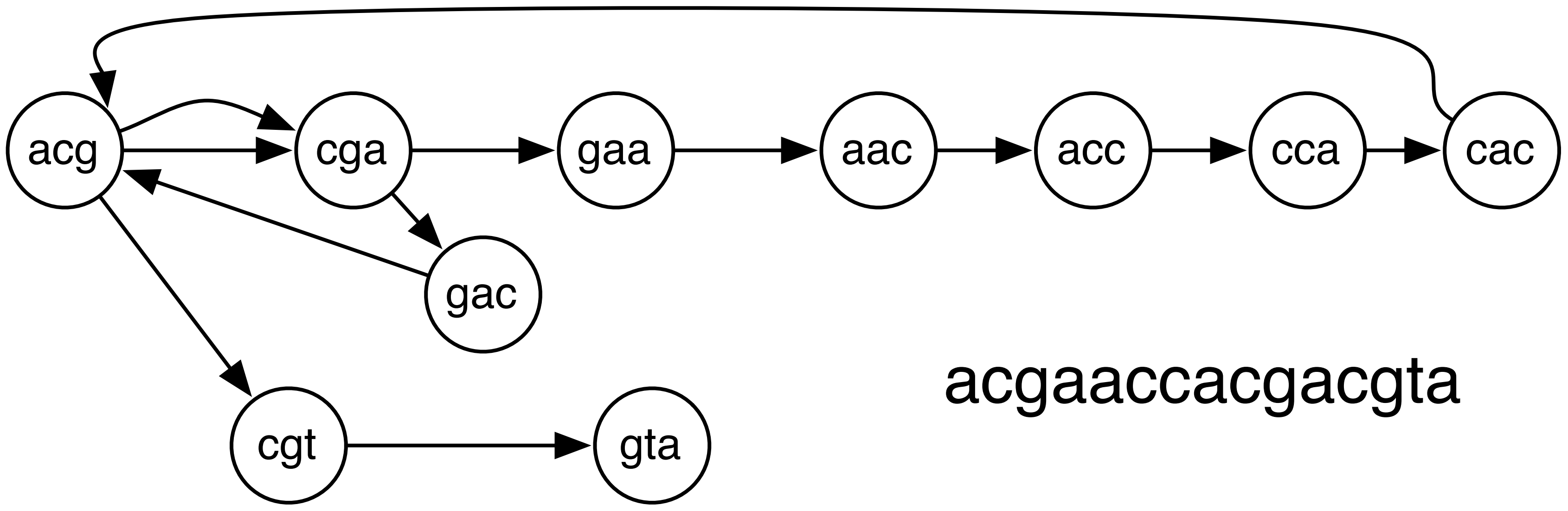
Examples



A directed graph has an Eulerian **cycle** if and only if:

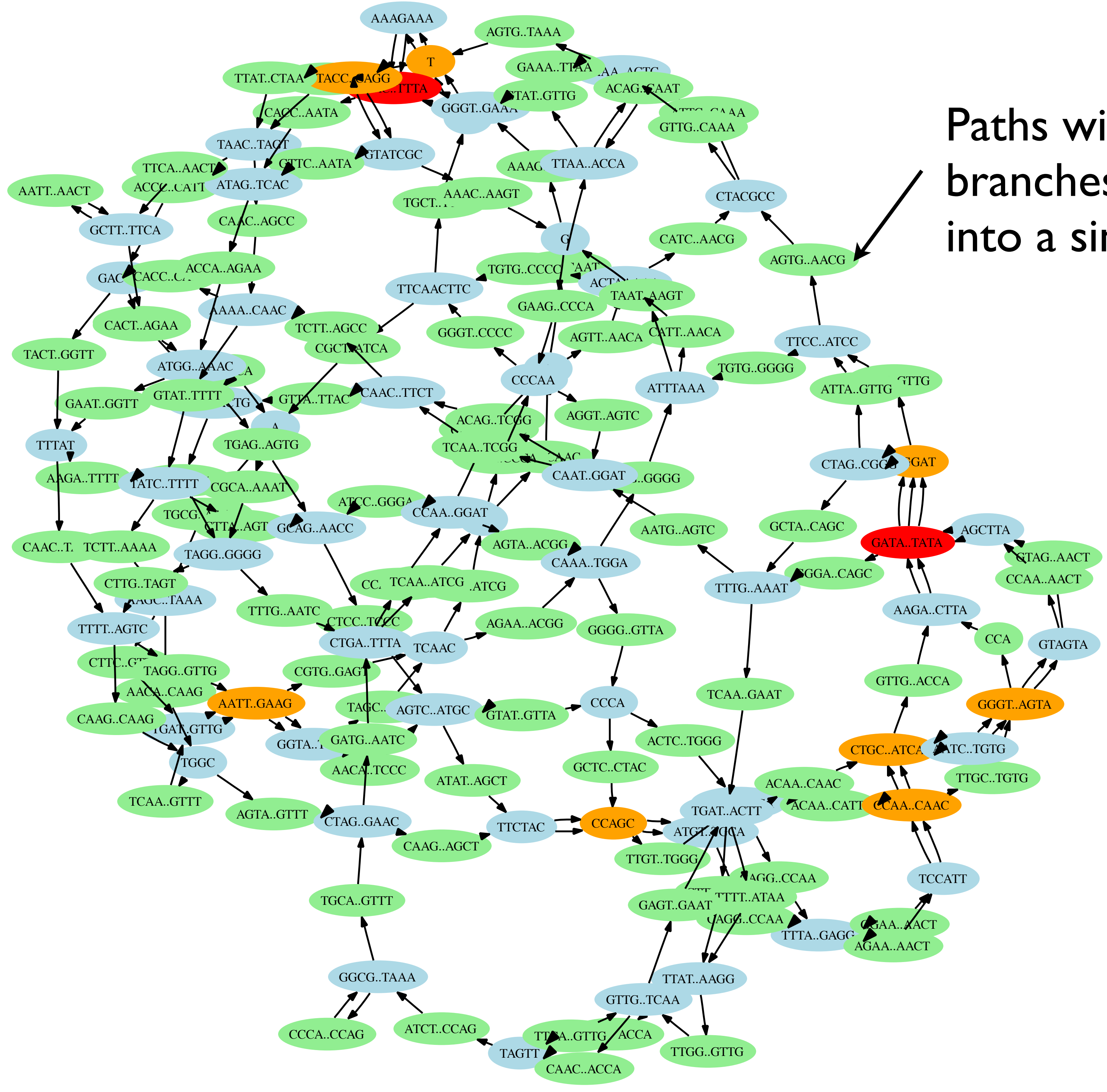
- All nodes have the same number of edges entering and leaving

tagacgaacgtacggtagg



acgaaccacgacgta

Example bacterial de Bruijn graph

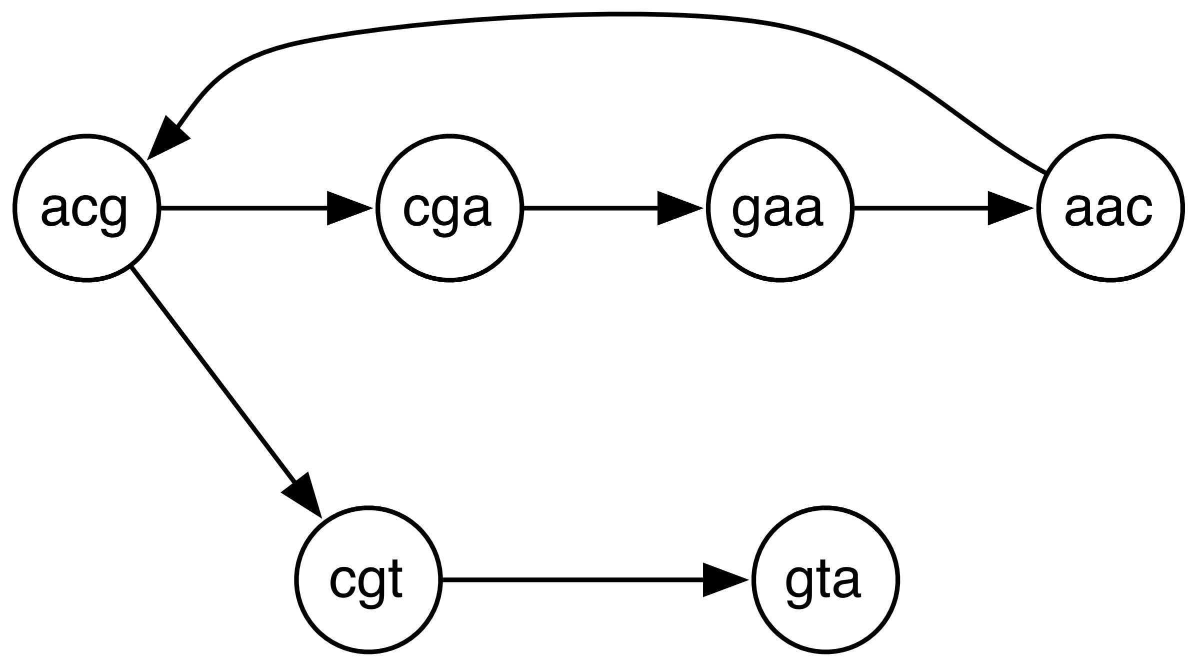


Paths with no branches compressed into a single node

Eulerian path = use every edge exactly once.

With perfect data, the genome can be reconstructed by some Eulerian path through this graph

Assembly via Eulerian Path



acgaacgta

Let dG(*s*) be the de Bruijn graph of string *s*. Then *s* corresponds to some Eulerian path in dG(*s*).

A directed graph has an Eulerian path if and only if:

- One node has one more edge leaving it than entering
- One node has one more edge entering than leaving
- All other nodes have the same number of edges entering and leaving

How can we find such a path?

Eulerian Path Algorithm

Connect node with out-degree < in-degree to node with out-degree < in-degree. So that we will have an Eulerian cycle.

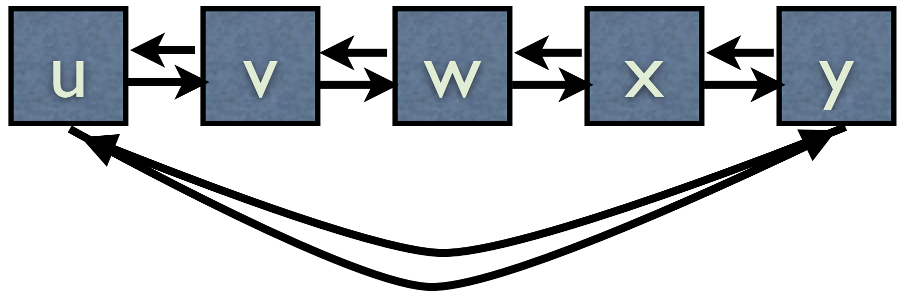
Why will you return to *u*?

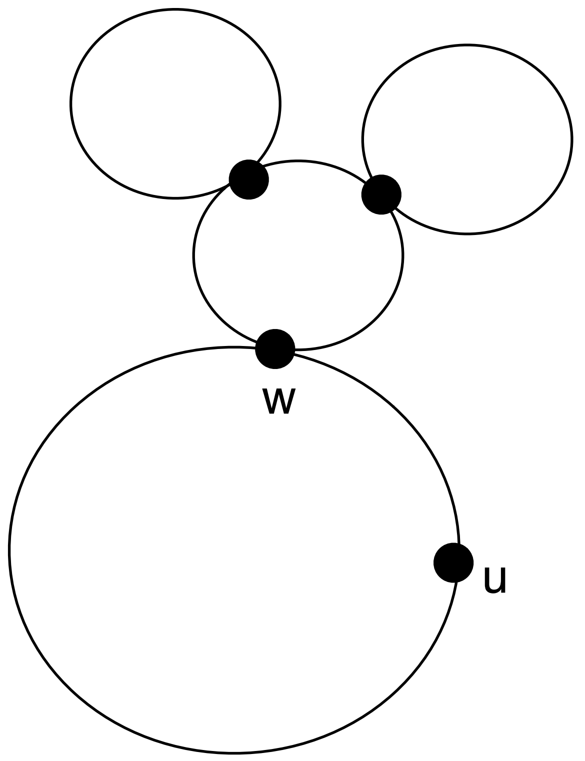
Walk from some arbitrary node *u* until you return to *u*, creating a doubly liked list of the path you visit.

*How can find such a node quickly?

Repeat until all edges used:

- Start from some node *w* on the current tour with unused edges*.
- Walk along unused edges until you return to *w*, inserting the visited nodes after *w* into the current tour list.





Eulerian Path Algorithm

Connect node with out-degree < in-degree to node with out-degree < in-degree. So that we will have an Eulerian cycle.

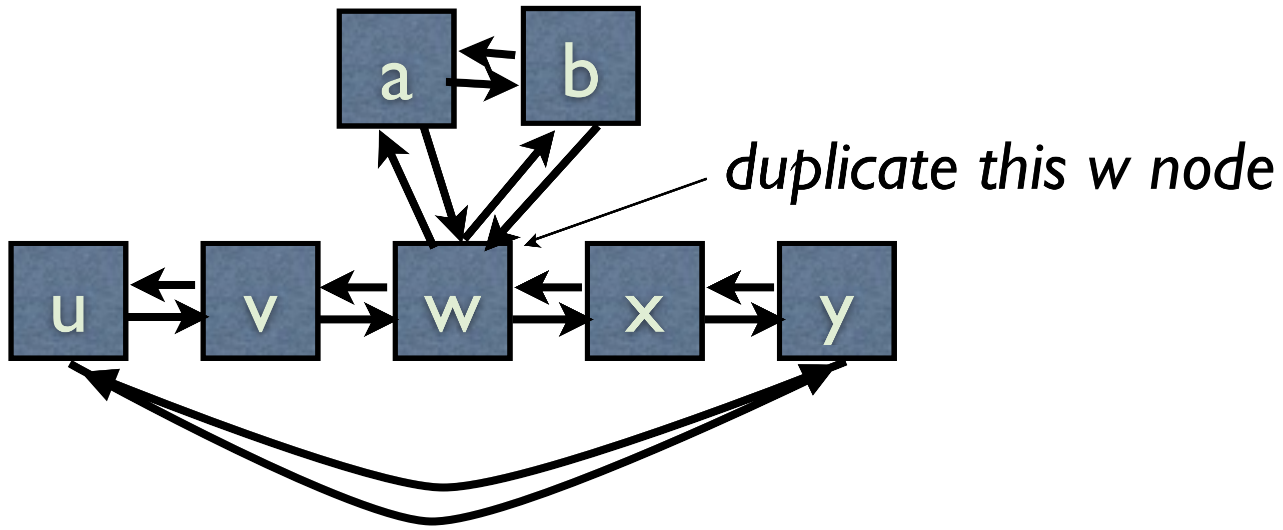
Why will you return to *u*?

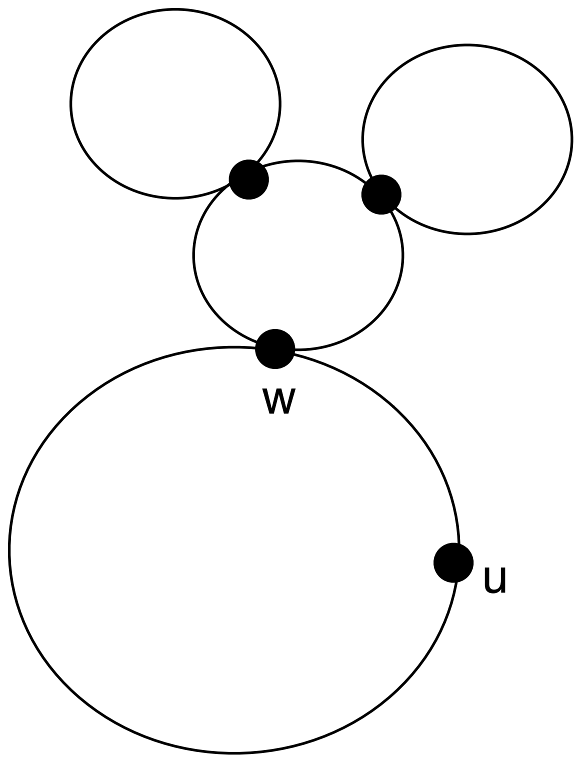
Walk from some arbitrary node *u* until you return to *u*, creating a doubly liked list of the path you visit.

*How can find such a node quickly?

Repeat until all edges used:

- Start from some node *w* on the current tour with unused edges*.
- Walk along unused edges until you return to *w*, inserting the visited nodes after *w* into the current tour list.





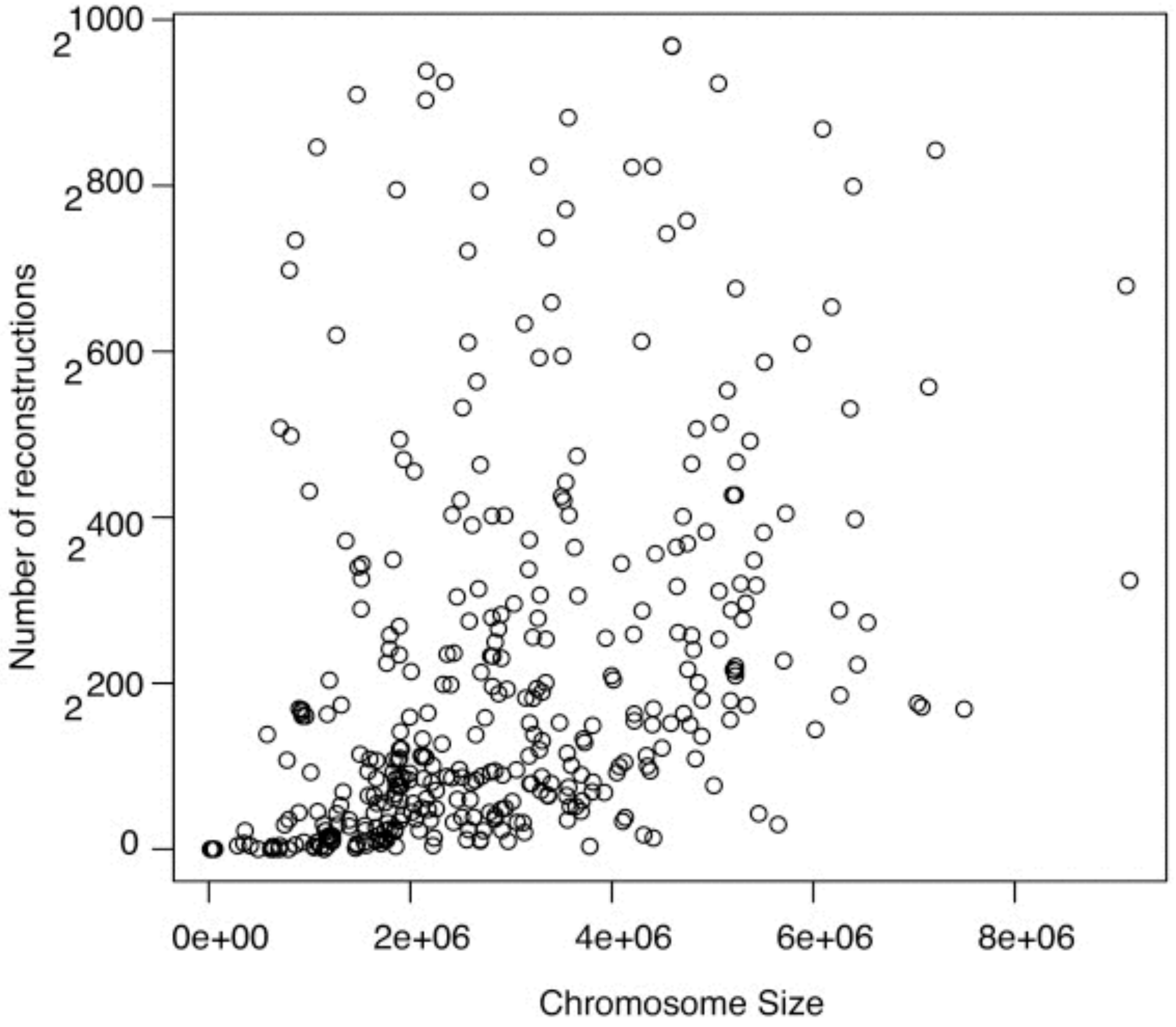
The Problem with Eulerian Paths

There are typically an astronomical number of possible Eulerian tours with perfect data.

Adding back constraints to limit # of tours leads to a NP-hard problem.

With imperfect data, there are usually NO Eulerian tours

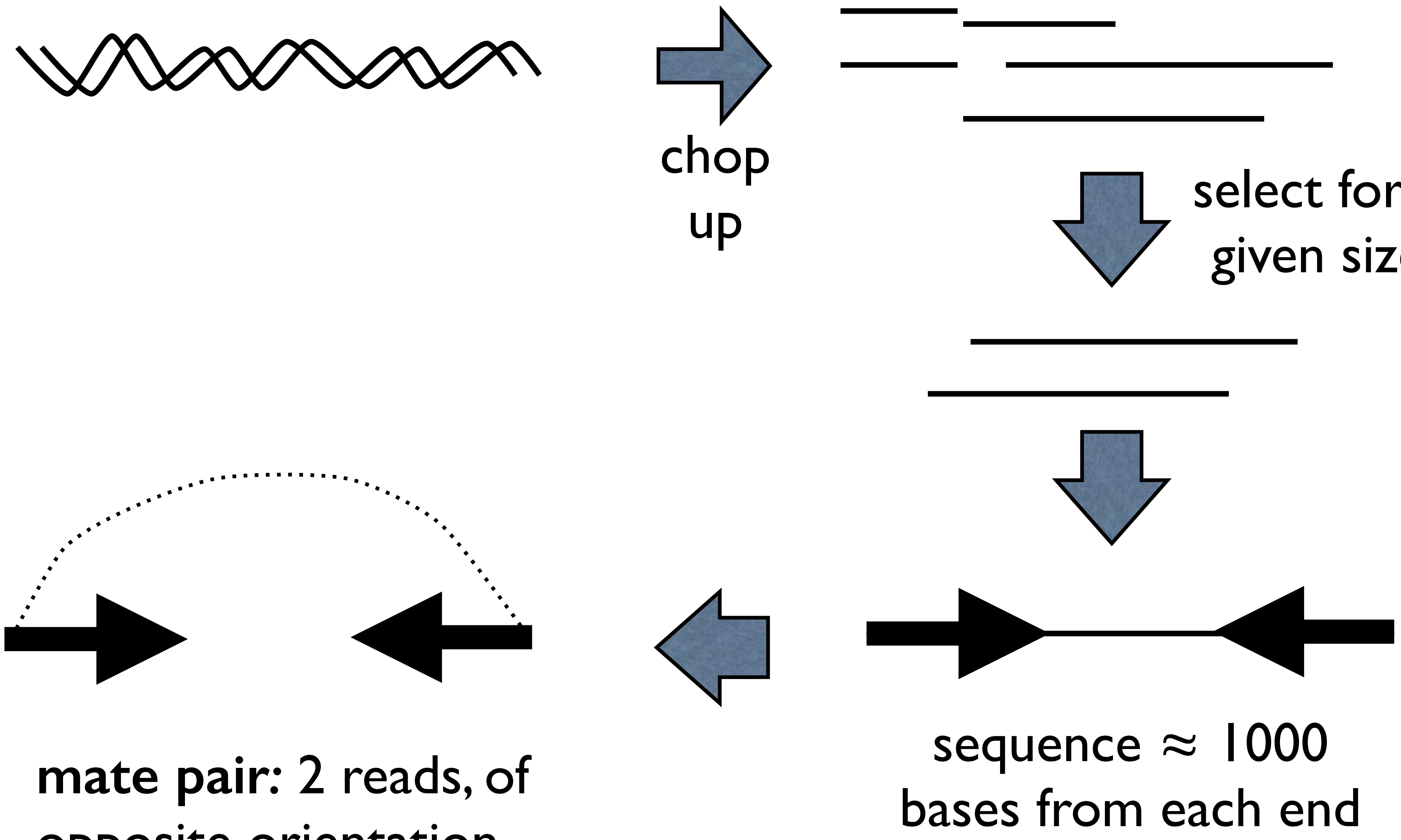
Estimating # of parallel edges is usually tricky.



(Kingsford, Schatz, Pop, 2010)

Aside: counting # of Eulerian tours in a directed graph is easy, but in an undirected graph is #P-complete (hard).

Mate Pairs

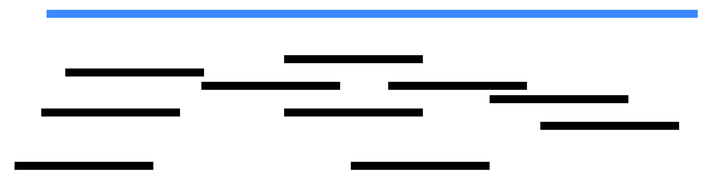
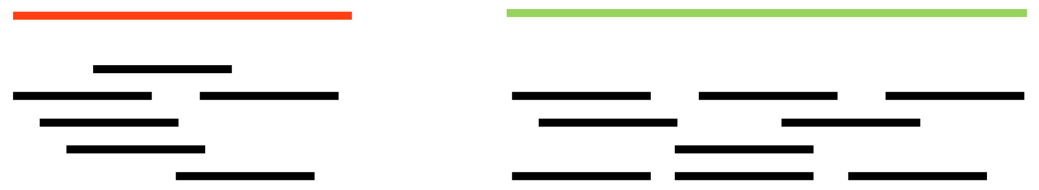


mate pair: 2 reads, of opposite orientation, separated by an approximately known distance

⇒ long range information

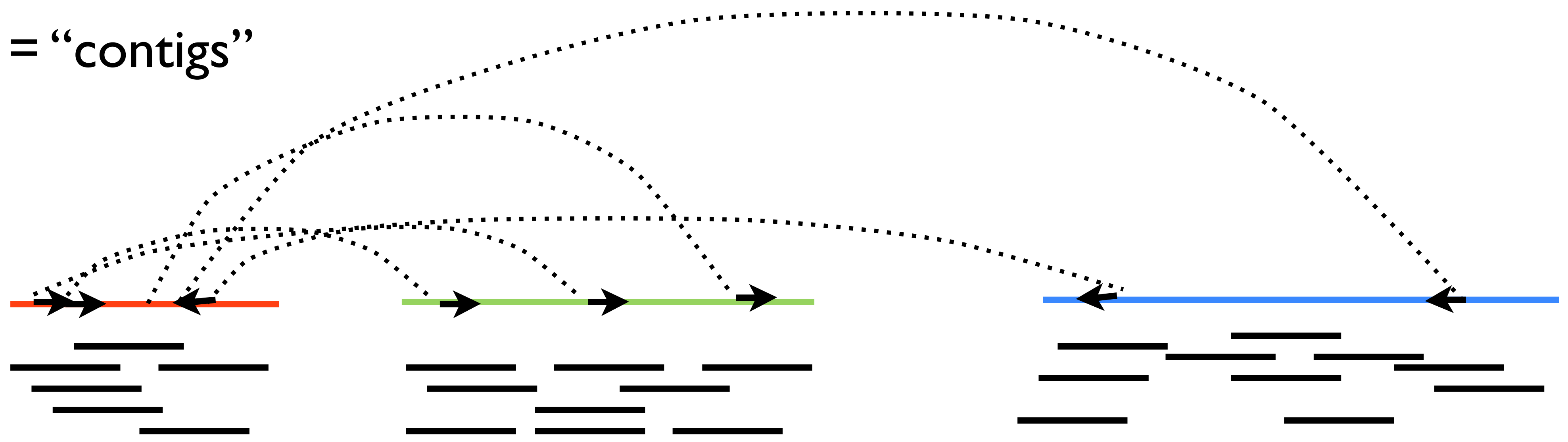
Scaffolding

Islands = "contigs"

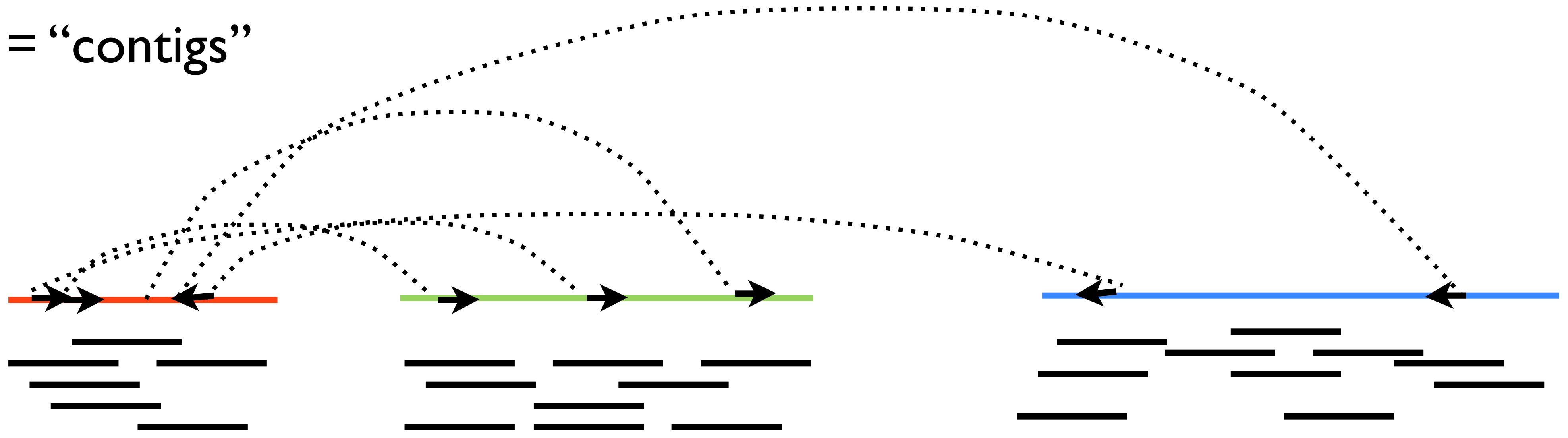
Scaffolding

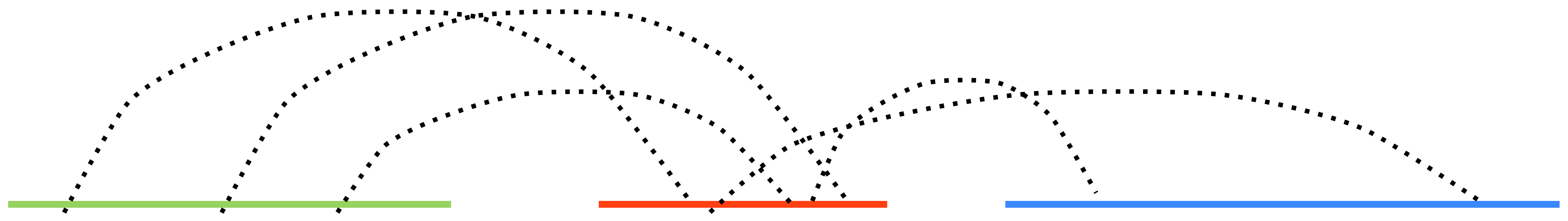
Islands = “contigs”



Scaffolding

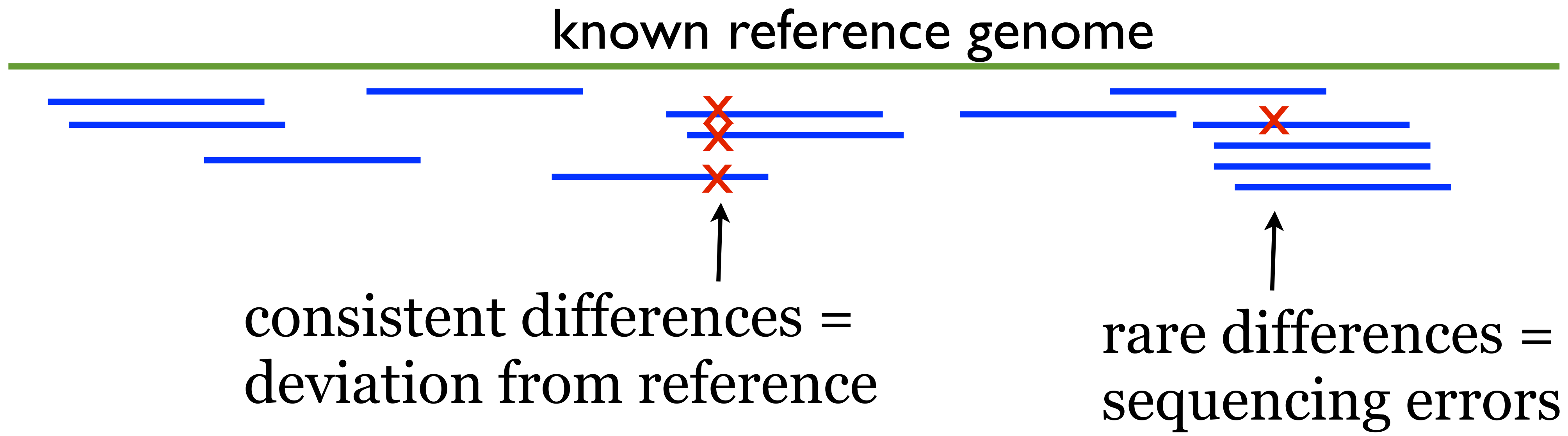
Islands = “contigs”





Comparative Assembly (Read Mapping)

Align reads to known genome:


known reference genome
consistent differences = deviation from reference
rare differences = sequencing errors

Can use much lower coverage
(e.g. 4X coverage instead of 20-30X for *de novo* assembly).

Aligning a large # of short sequences to one large sequence is an important special case of sequence alignment.

Summary

- Sanger sequencing reads DNA via synthesis; 800-1000bp.
- Assembly Paradigms:
    - Shortest Common Superstring (NP-hard; sensitive to repeats)
    - Hamiltonian cycle in overlap graph (NP-hard)
    - Eulerian cycle in de Bruijn graph (polynomial in basic form, but large # of solutions)
- Overlap alignment can be computed with slight variant of sequence alignment DP.
    - K-mer hashing technique avoids all pairs overlap alignment

Hard vs. Easy

- Eulerian path – visit every edge once (easy)
- Hamiltonian path – visit every node once (hard)

- Shortest common supersequence (easy)
- Shortest common superstring (hard)

- Counting Eulerian tours in directed graphs (easy)
- Counting Eulerian tours in undirected graphs (hard)

- Aligning 2 sequences (easy)
- Aligning $k > 2$ sequences (hard)

- Shortest path (easy)
- Longest path (hard)

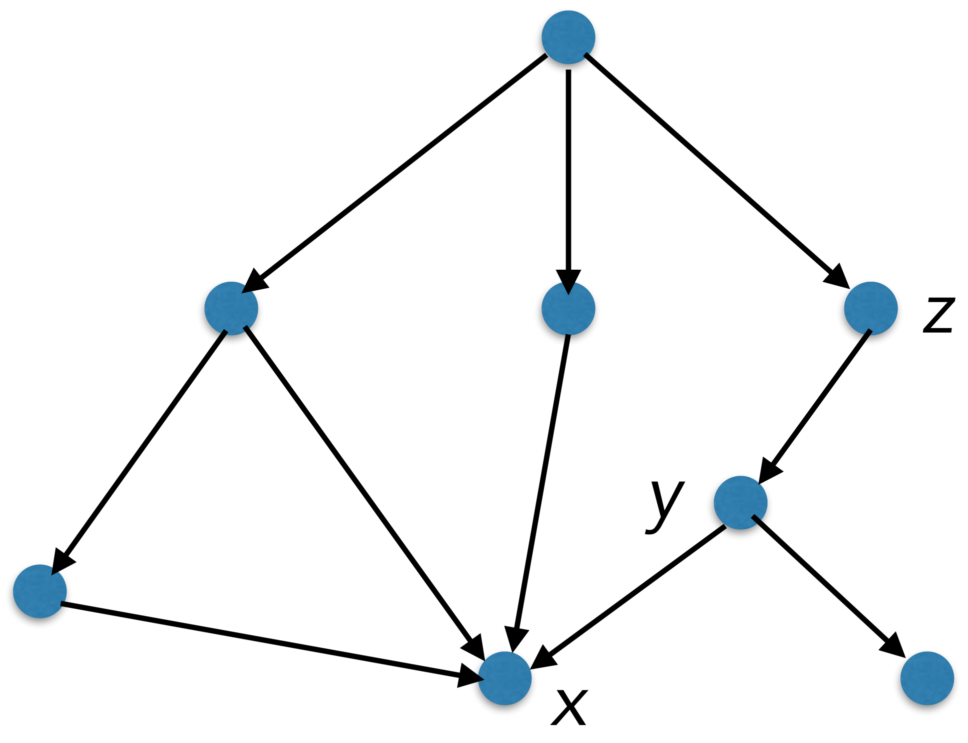
Cufflinks Transcript Assembly

Cole Trapnell, Brian A. Williams, Geo Pertea, Ali Mortazavi, Gordon Kwan, Marijke J. van Baren, Steven L. Salzberg, Barbara J. Wold, and Lior Pachter.· Transcript assembly and abundance estimation from RNA-Seq reveals thousands of new transcripts and switching among isoforms. *Nat Biotechnol,* 28(5): 511–515 (2010)

Partially Ordered Sets

Def. A pair (S, ≤) is a partial order if, for all x, y ∈ S:

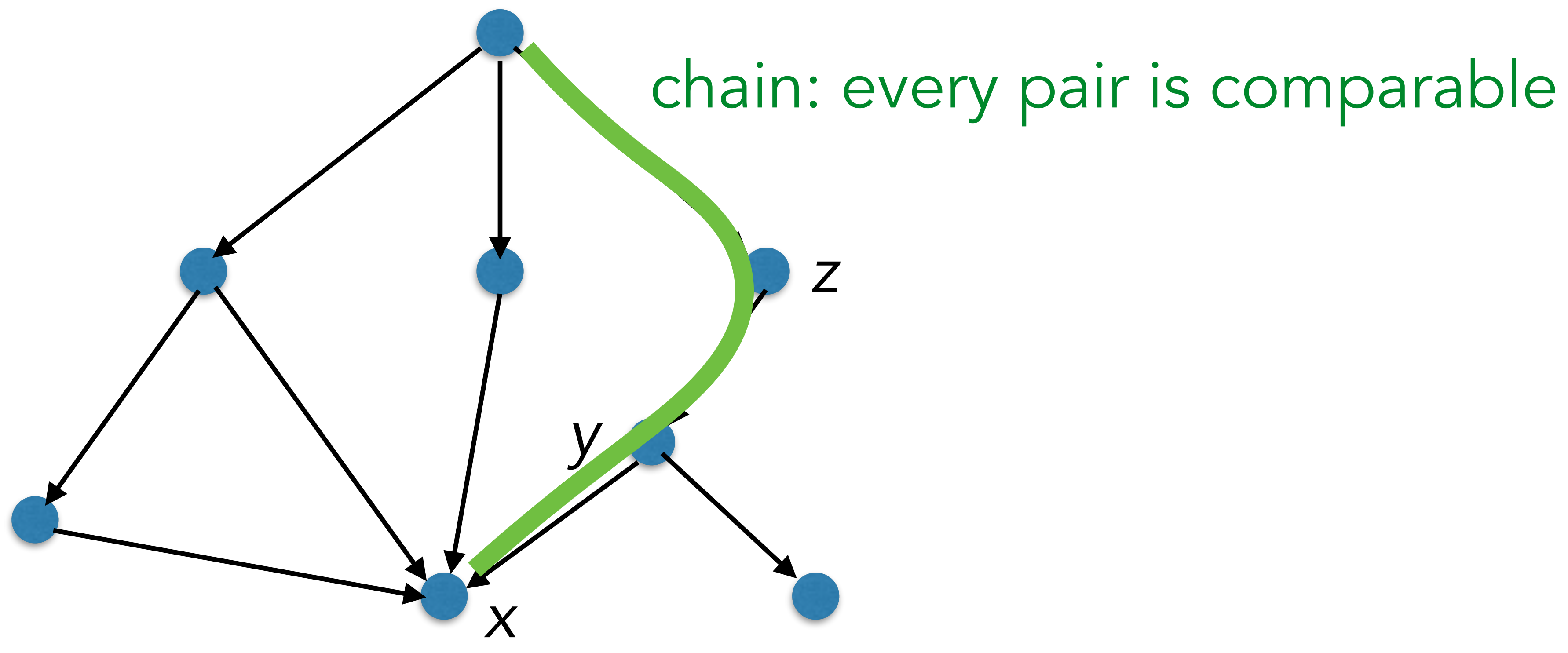
(transitivity)	$x \le y$ and $y \le z \Rightarrow x \le z$
(reflexivity)	$x \le x$
(antisymmetry)	$x \le y$ and $y \le x \Rightarrow x = y$



Partially Ordered Sets

Def. A pair (S, ≤) is a partial order if, for all x, y ∈ S:

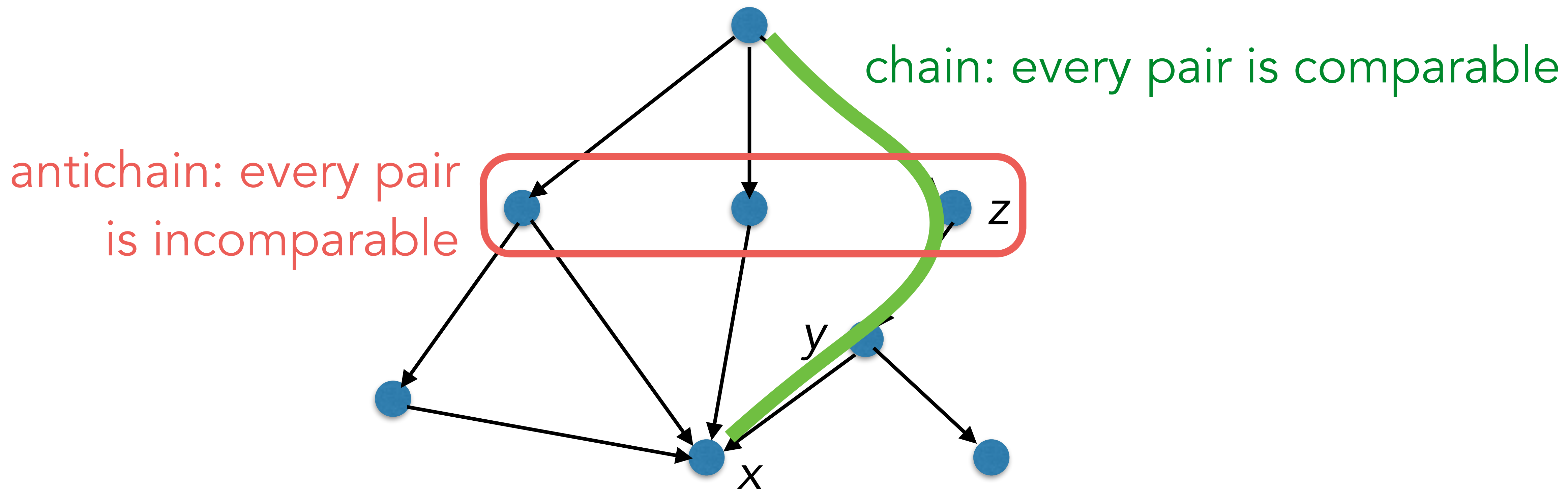
(transitivity)	$x \le y$ and $y \le z \Rightarrow x \le z$
(reflexivity)	$x \le x$
(antisymmetry)	$x \le y$ and $y \le x \Rightarrow x = y$



Partially Ordered Sets

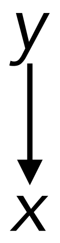
Def. A pair (S, ≤) is a partial order if, for all x, y ∈ S:

(transitivity)	$x \le y$ and $y \le z \Rightarrow x \le z$
(reflexivity)	$x \le x$
(antisymmetry)	$x \le y$ and $y \le x \Rightarrow x = y$

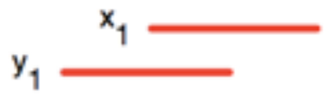

chain: every pair is comparable
antichain: every pair is incomparable
z
y
x

Cufflink's Partial Order

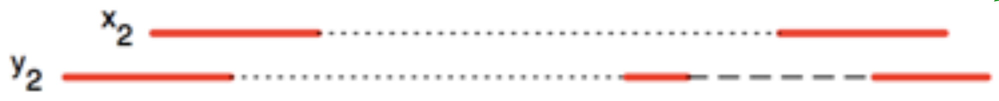
● = sequenced fragment:



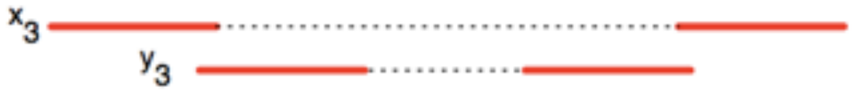
= x aligns to the left of y and x and y have compatible intron structure



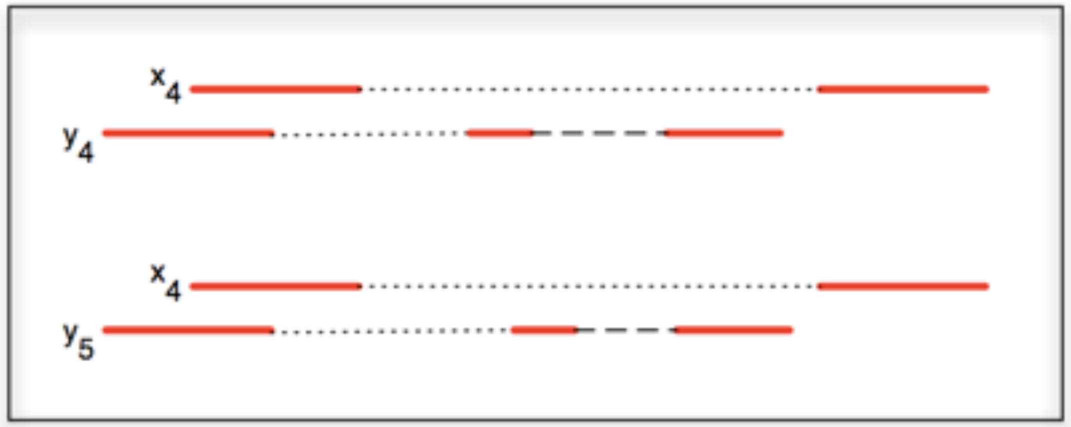
$y_1 \leq x_1$



incompatible b/c the right end of y_2 is split-mapped, implying an intron where there is no intron in x_2.



x_3 and y_3 are nested, and so are merged into a single fragment.

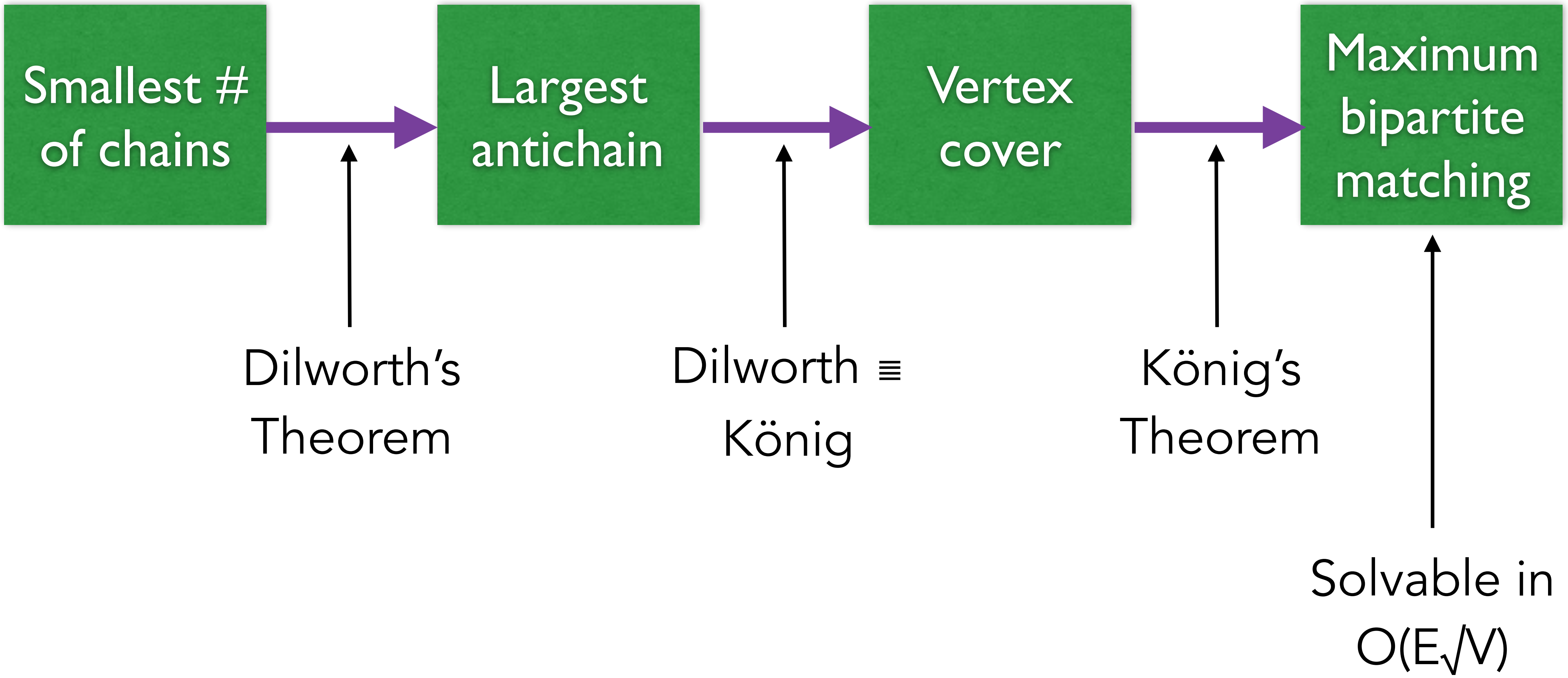


x_4 is uncertain because it could be compatible with either y_4 or y_5; x_4 is therefore thrown away.

(Trapnell et al., 2010)

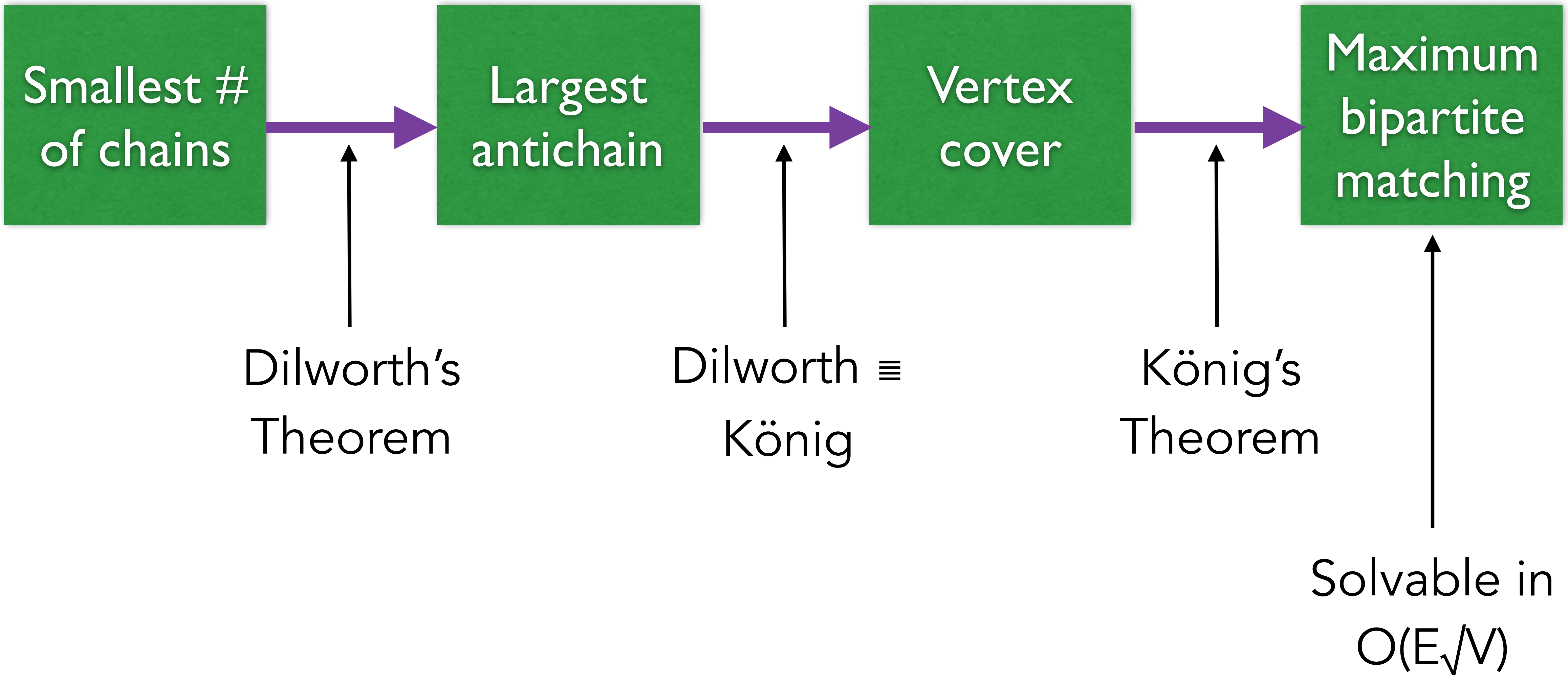
Cufflinks' Assembly Algorithm

Partitioning partial order into smallest # of chains →
"parsimonious" set of transcripts that explains the observed reads



Cufflinks' Assembly Algorithm

(covering)
Partitioning partial order into smallest # of chains →
"parsimonious" set of transcripts that explains the observed reads



Dilworth's Theorem

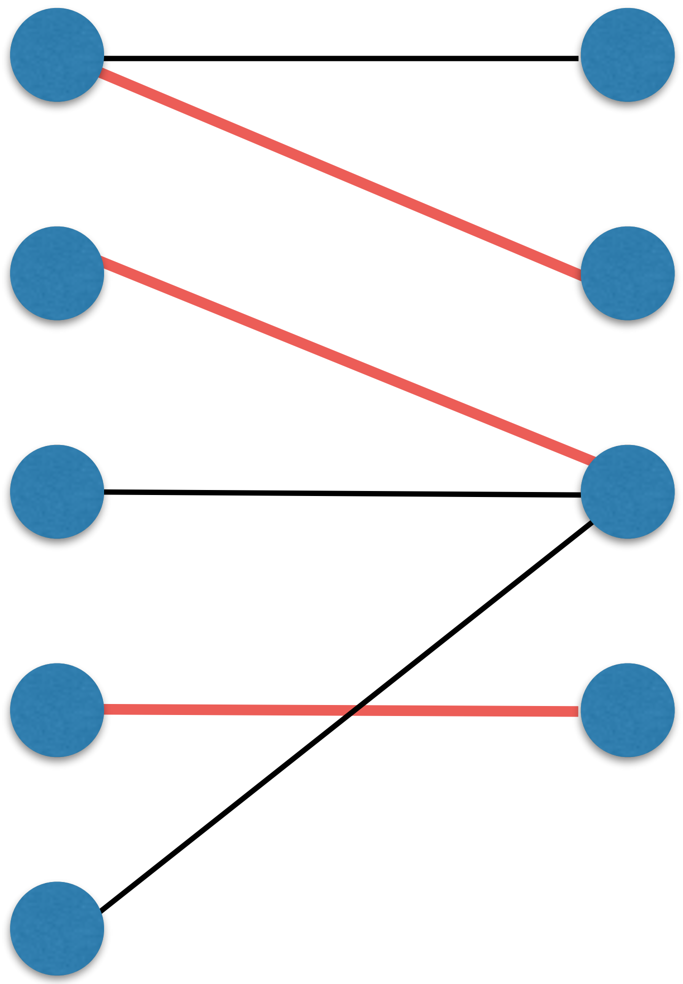
Thm (Dilworth). In a poset, the size of the largest antichain = the size of the minimum cover by chains.

Proof intuition.

- The largest antichain must hit every chain (otherwise it could be made larger).
- It can't hit any chain twice, otherwise it would contain two comparable items.

König's Theorem

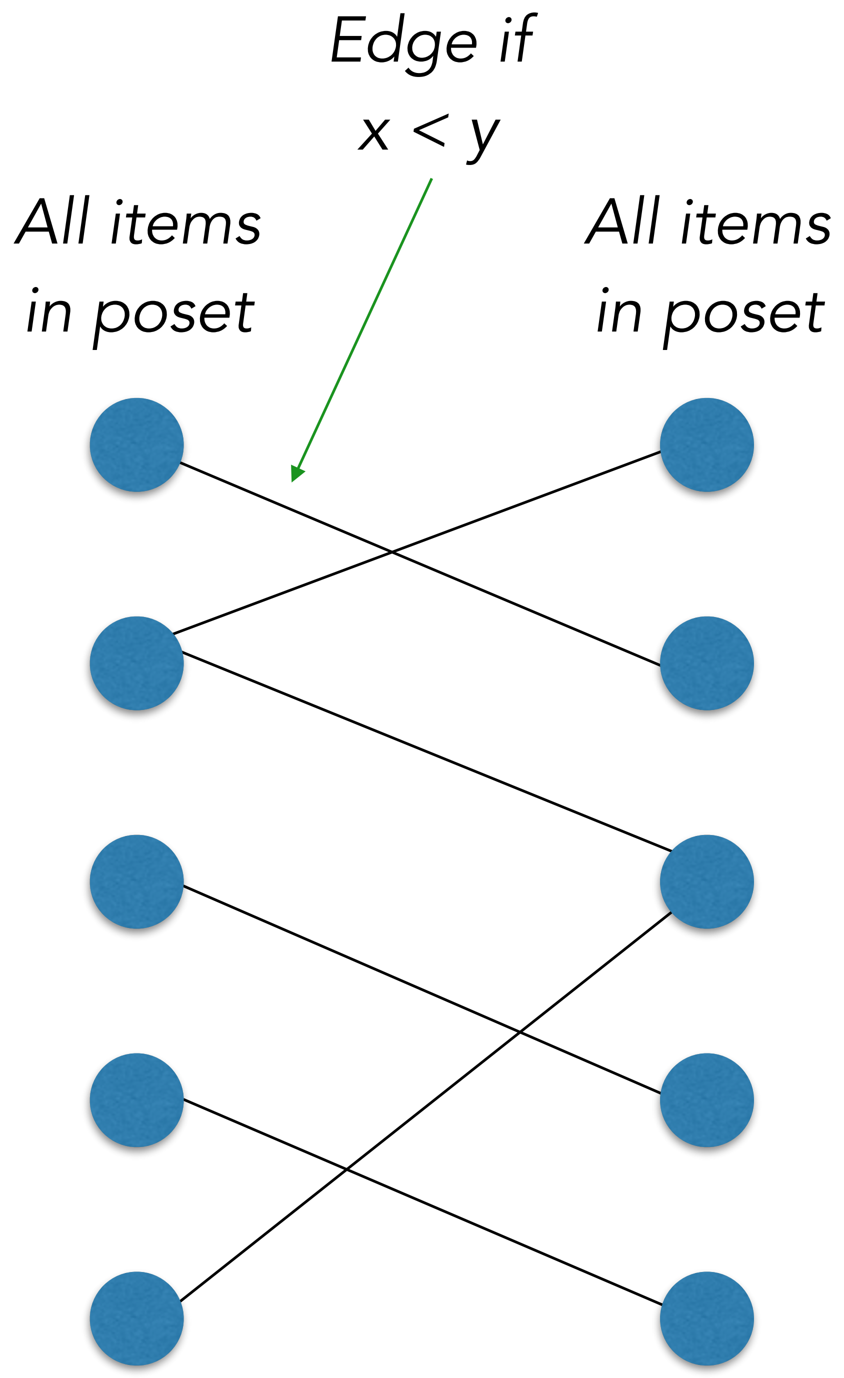
Thm (König). In a bipartite graph, the # of edges in a maximum matching = # of vertices in the smalelst vertex cover.



Proof intuition.

- In a maximum matching, every edge must be covered.
- Otherwise, if both endpoints are not matched, we could add that edge to the matching and increase its size.

Using Matching to Find a Minimal Chain Cover



Let M be the maximal matching.

By König's theorem, there is a (minimal) vertex cover C of the same size as M.

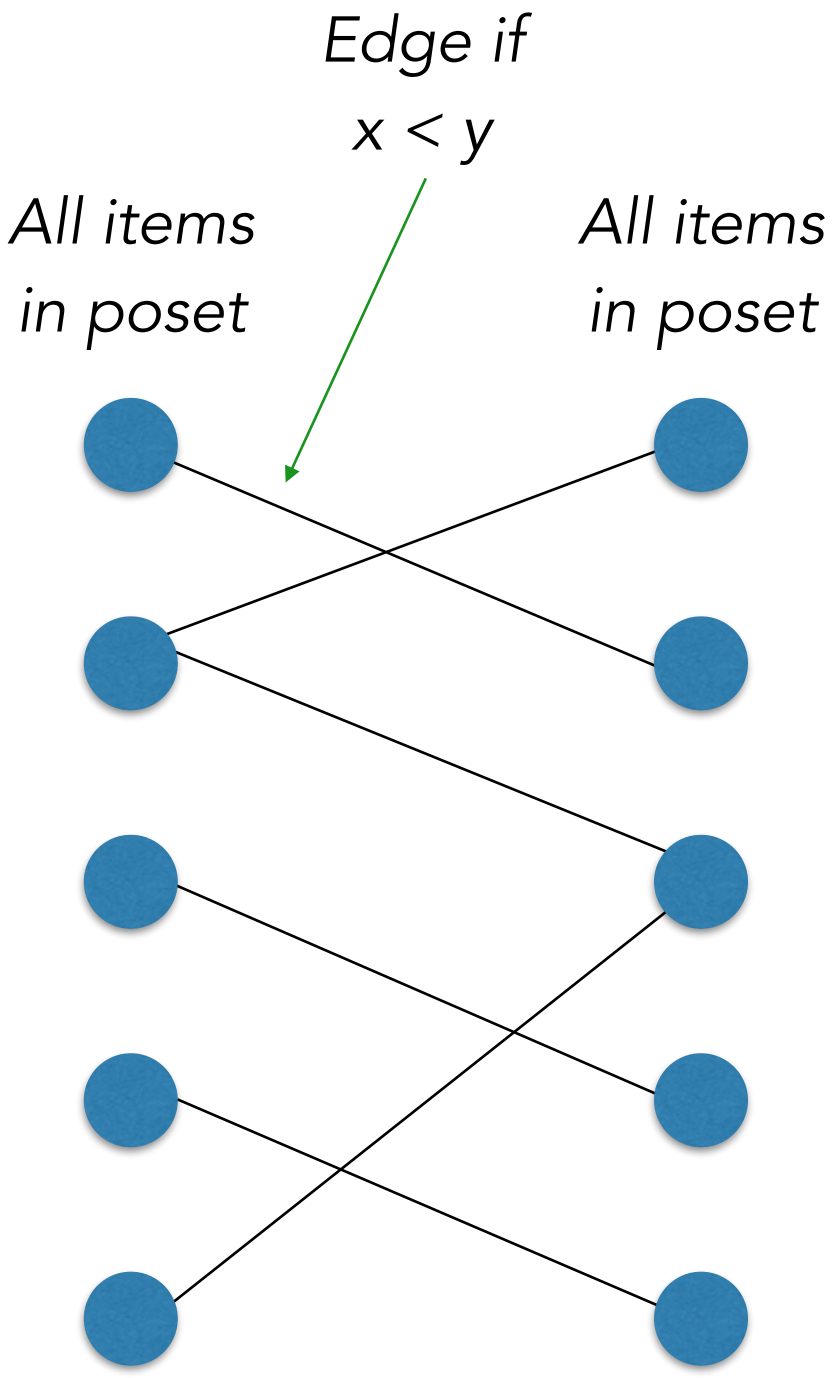
Let T be the elements of the poset that are not in C.

T is an antichain. Why?

Make a set W of chains by u ≡ v if (u,v) ∈ M.

These equivalence classes are chains. Why?

Using Matching to Find a Minimal Chain Cover



Let M be the maximal matching.

By König's theorem, there is a (minimal) vertex cover C of the same size as M.

Let T be the elements of the poset that are not in C.

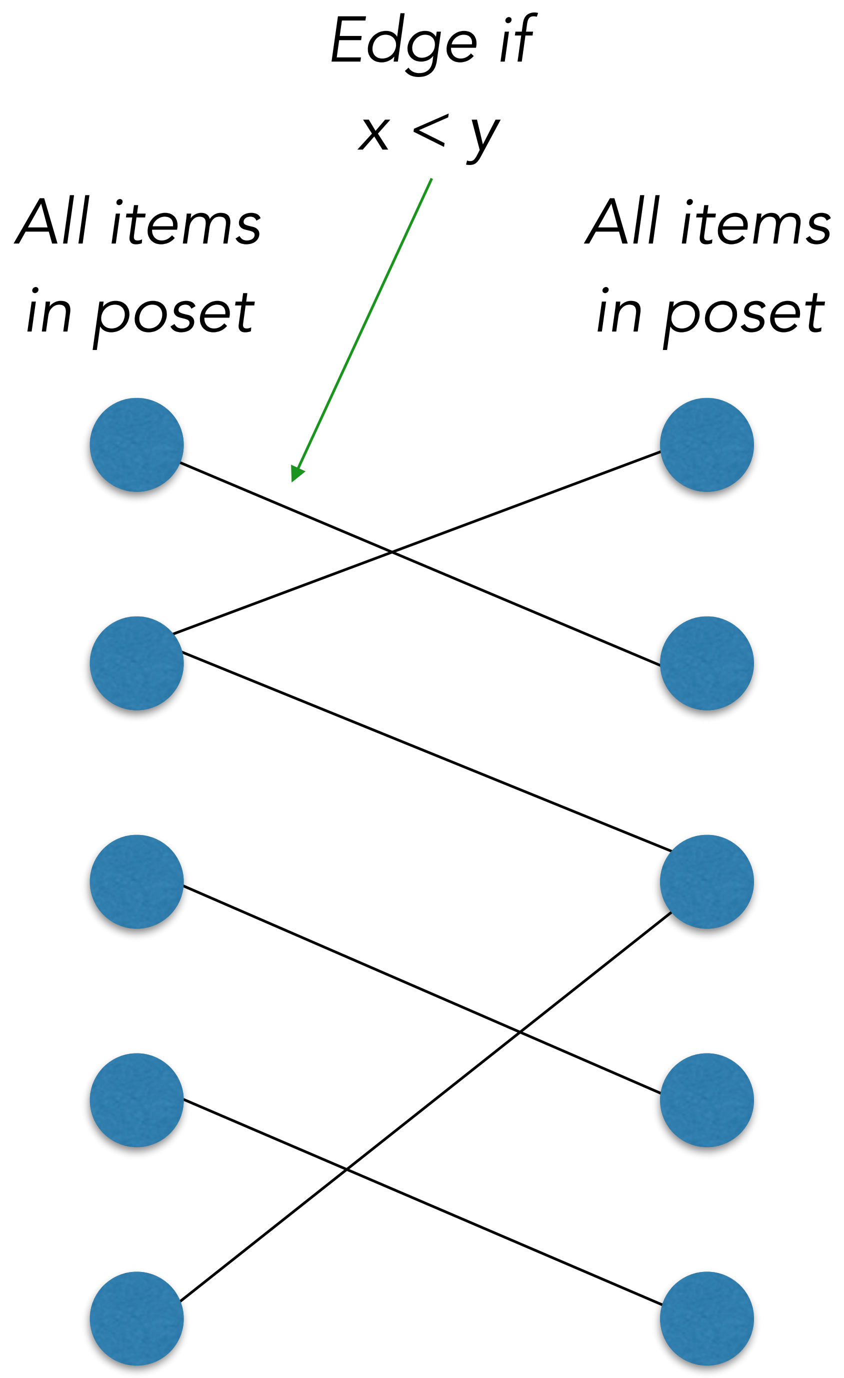
T is an antichain. Why?

If u and v were comparable, there would be an edge between them, and since neither u or v used in M, we could add that edge to M.

Make a set W of chains by u ≡ v if (u,v) ∈ M.

These equivalence classes are chains.
Why?

Using Matching to Find a Minimal Chain Cover



Let M be the maximal matching.

By König's theorem, there is a (minimal) vertex cover C of the same size as M.

Let T be the elements of the poset that are not in C.

T is an antichain. Why?

If u and v were comparable, there would be an edge between them, and since neither u or v used in M, we could add that edge to M.

Make a set W of chains by u ≡ v if (u,v) ∈ M.

These equivalence classes are chains. Why?

Every pair of items in each equivalence class had an edge between them, meaning they were comparable.

|W| = |T|

M = maximal matching.

C = vertex cover of the same size as M.

T = antichain elements of poset that are not in C.

W = set of chains formed from edges of M.

n = # elements in poset

m = # of edges in matching

Size of *T* is n - m. Why?

Size of *W* is n - m. Why?

|W| = |T|

M = maximal matching.

C = vertex cover of the same size as M.

T = antichain elements of poset that are not in C.

W = set of chains formed from edges of M.

n = # elements in poset

m = # of edges in matching

Size of *T* is n - m. Why?

Every edge uses up exactly one element on the LHS of the bipartite graph.

Size of *W* is n - m. Why?

|W| = |T|

M = maximal matching.

C = vertex cover of the same size as M.

T = antichain elements of poset that are not in C.

W = set of chains formed from edges of M.

n = # elements in poset

m = # of edges in matching

Size of *T* is n - m. Why?

Every edge uses up exactly one element on the LHS of the bipartite graph.

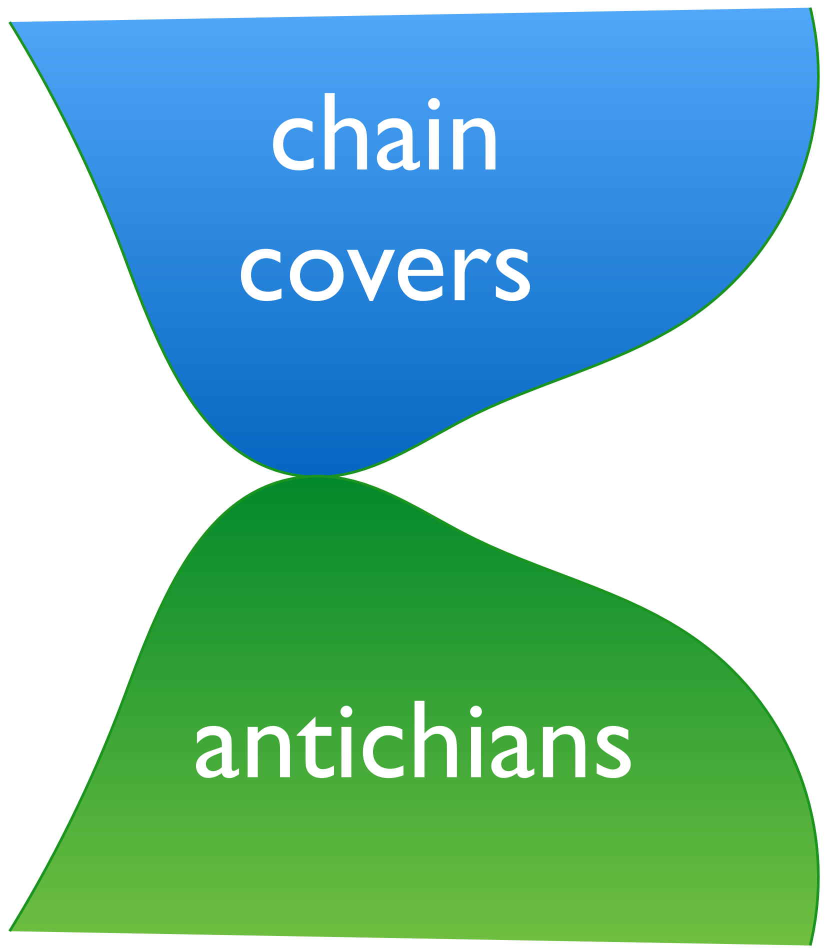
Size of *W* is n - m. Why?

Consider set of n "chains" each consisting of a single element of poset.

Each edge (u,v) that we use to put v into the same poset as u reduces the number of chains by 1.

⇒ Number of equivalence-class chains = n - m

Why is W Minimum Size?



All antichains must be of size ≤ all chain covers.

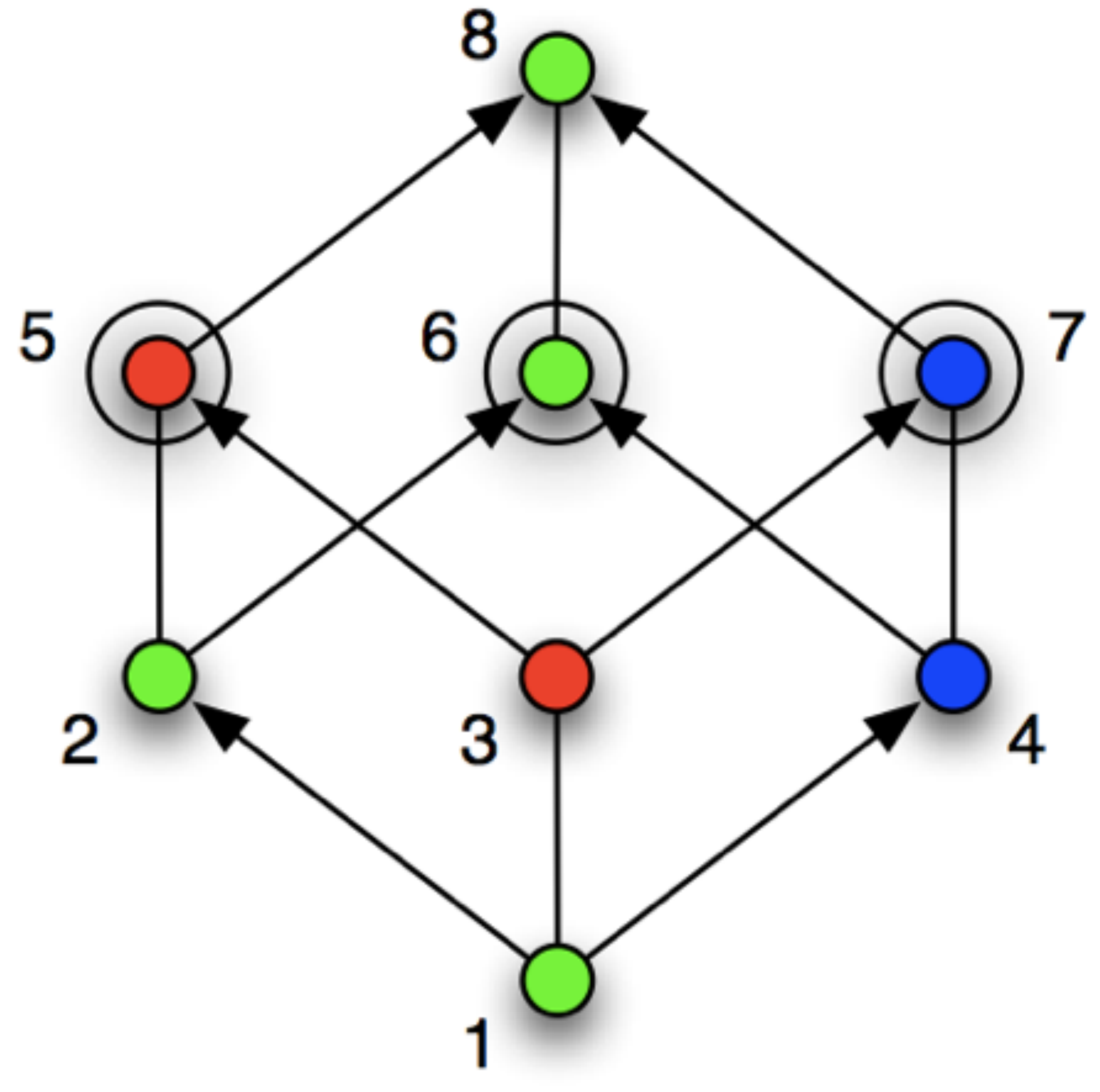
Suppose not, and let A be an antichain bigger than cover Q.

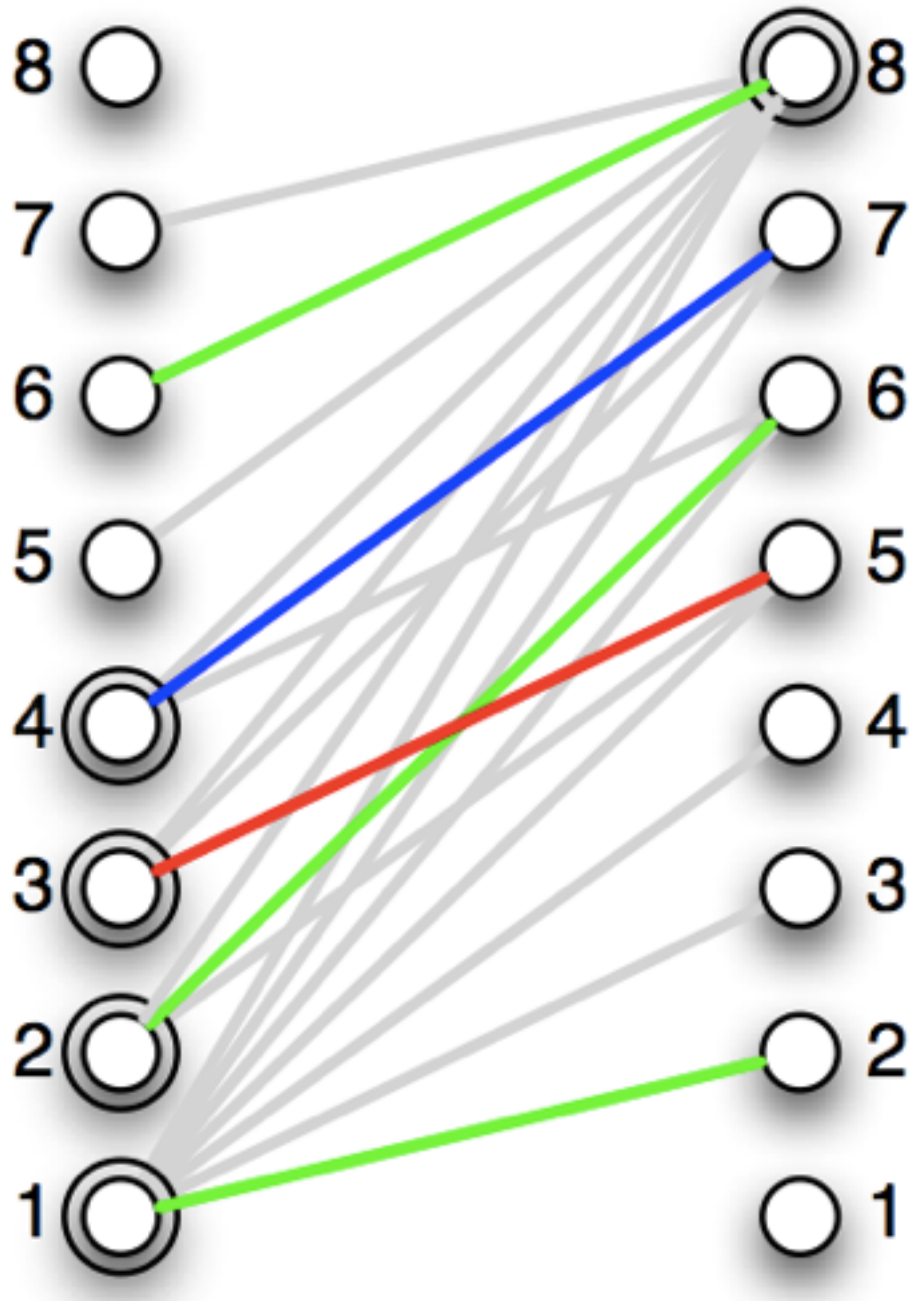
Then, by pigeonhole, A must contain at least 2 elements x, y from the same chain in Q.

But x, y are comparable because they are in the same chain.

⇒ the pair (T,W) must be a largest antichain and a smallest W because they are the same size.

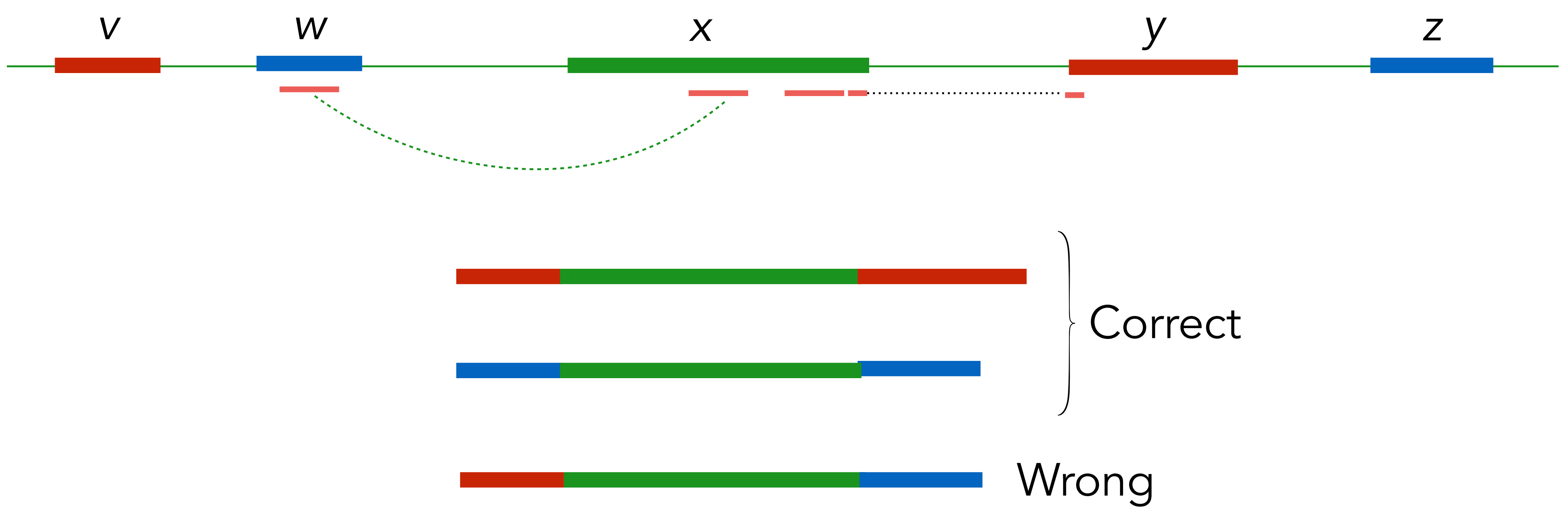
A Matching-Covering Example





(Trapnell et al., 2010)

Selecting From Among Many Minimum Solutions

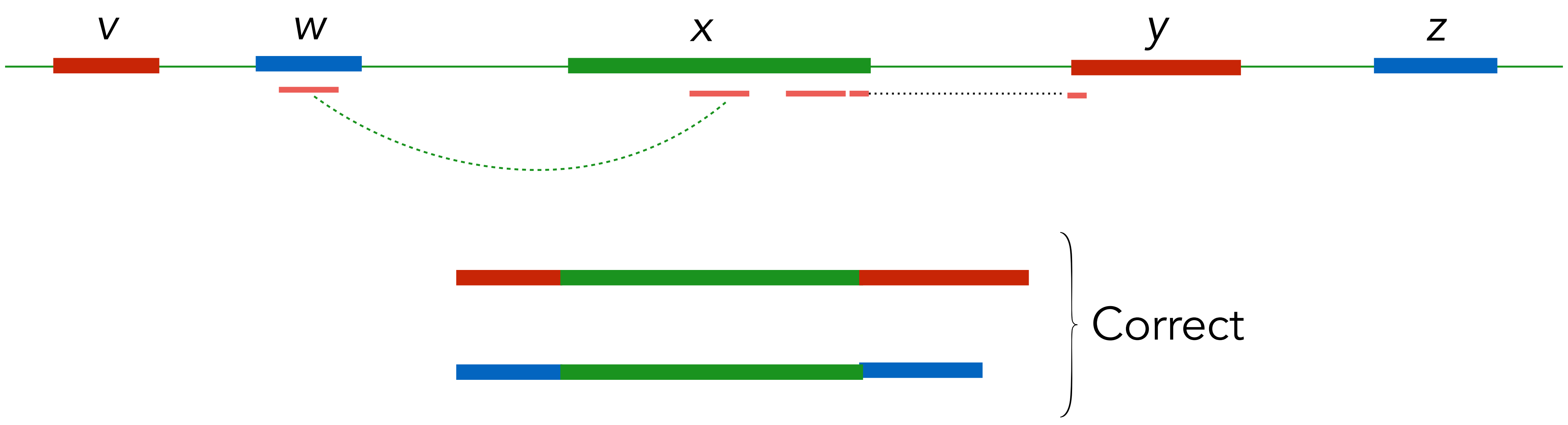


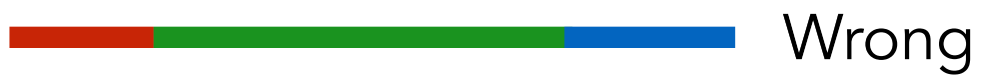
Idea: exons included in same transcript should have similar expression

Estimate Percent Spliced In (PSI, ψ): # of reads crossing exon x that are compatible with x divided by # of reads overlapping x (divided by length of x).

$$\mathrm{weight}(x, y) = -\log(1 - |\psi_x - \psi_y|)$$

Selecting From Among Many Minimum Solutions



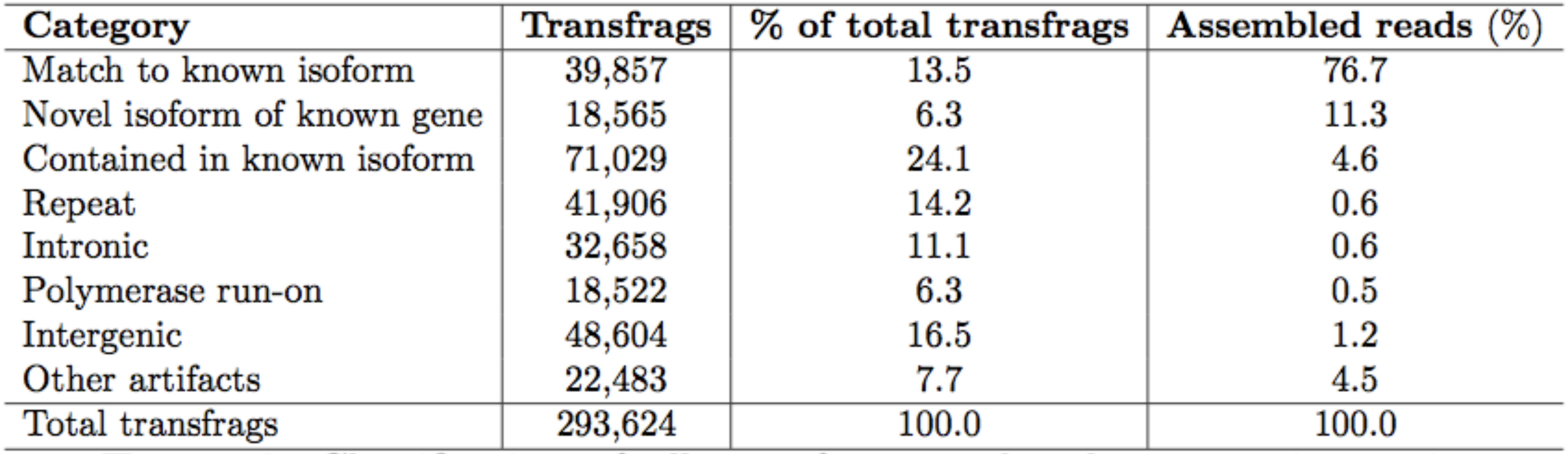


Idea: exons included in same transcript should have similar expression

Estimate measures how similar the exons' PSI values are that are compatib d by length of x).

$$\text{weight}(x, y) = -\log(1 - |\psi_x - \psi_y|)$$

Discovery of Novel Isoforms



Category	Transfrags	% of total transfrags	Assembled reads (%)
Match to known isoform	39,857	13.5	76.7
Novel isoform of known gene	18,565	6.3	11.3
Contained in known isoform	71,029	24.1	4.6
Repeat	41,906	14.2	0.6
Intronic	32,658	11.1	0.6
Polymerase run-on	18,522	6.3	0.5
Intergenic	48,604	16.5	1.2
Other artifacts	22,483	7.7	4.5
Total transfrags	293,624	100.0	100.0

TABLE 2. Classification of all transfrags produced at any time point with respect to annotated gene models and masked repeats in the mouse genome. Transfrags that are present in multiple time point assemblies are multiply counted to preserve the relative distribution of transfrags among the categories across the full experiment.

(Trapnell et al., 2010)