Assembly in Practice: Part 1: OLC

Ben Langmead

JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

Department of Computer Science

Please sign guestbook (www.langmead-lab.org/teaching-materials) to tell me briefly how you are using the slides. For original Keynote files, email me (ben.langmead@gmail.com).

Assembly in the real world

Neither formulation (SCS, Eulerian walk) is practical

…but *graphs* discussed (overlap, De Bruijn) are useful; one or the other is at the core of all practical assembly methods

Assembly in the real world

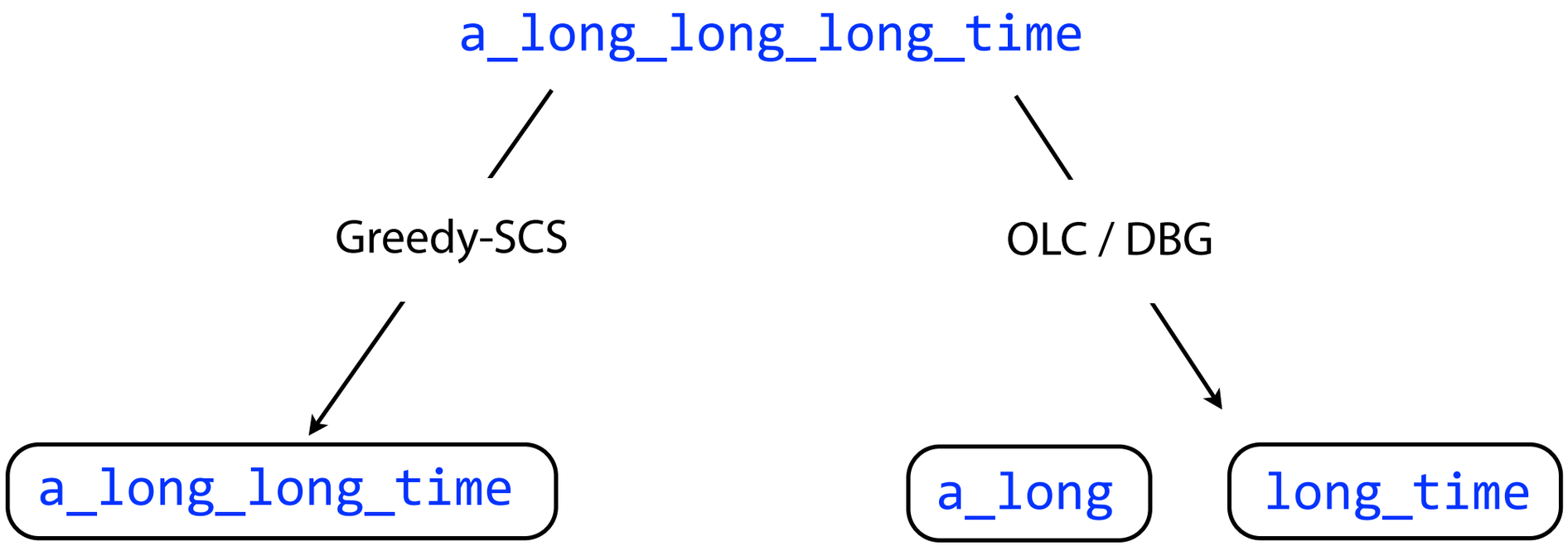
OLC: Overlap-Layout-Consensus assembly

DBG: De Bruijn graph assembly

Handle unresolvable repeats by *leaving them out*

This breaks the assembly into fragments

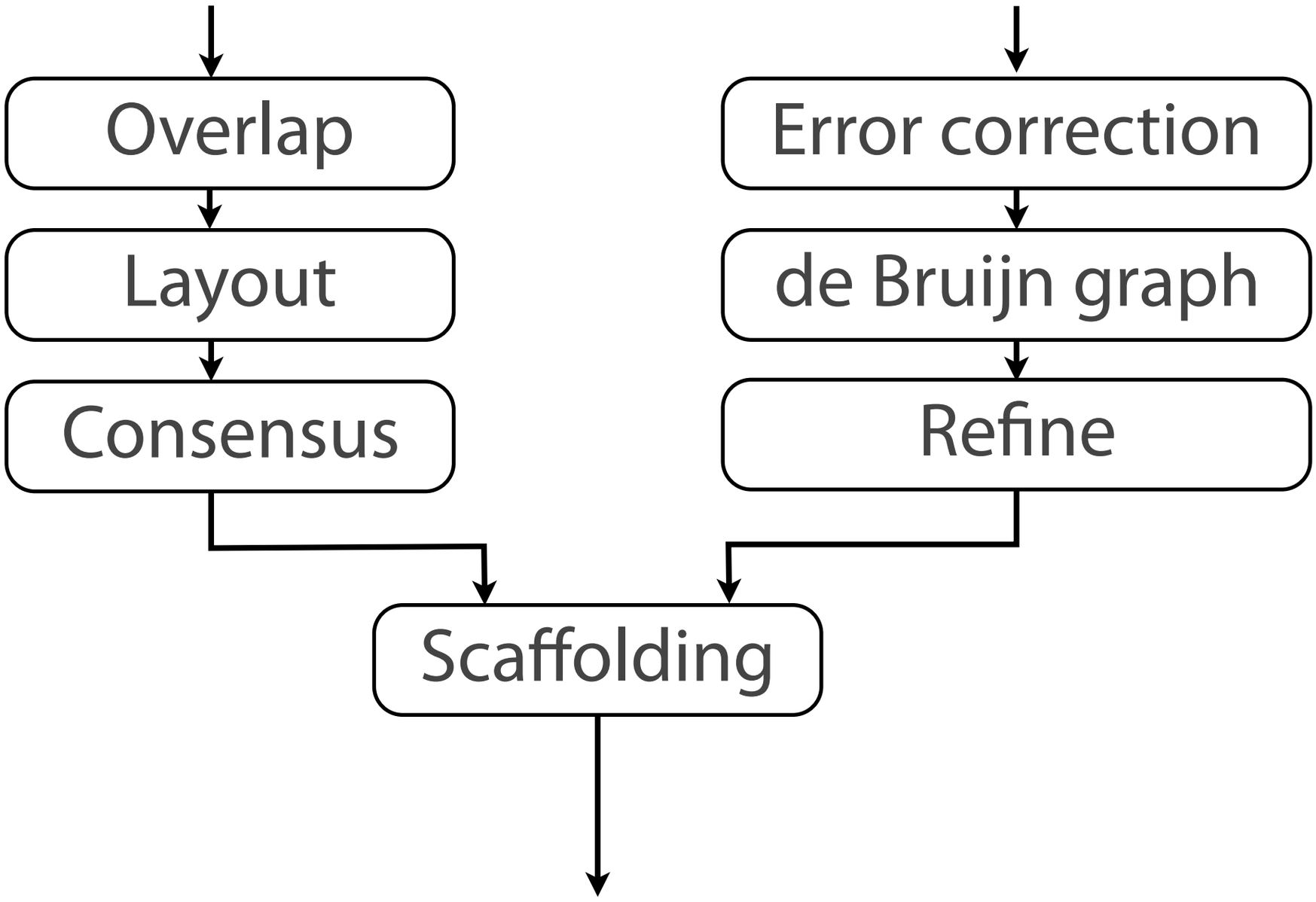
Fragments called *contigs* (short for *contiguous*)



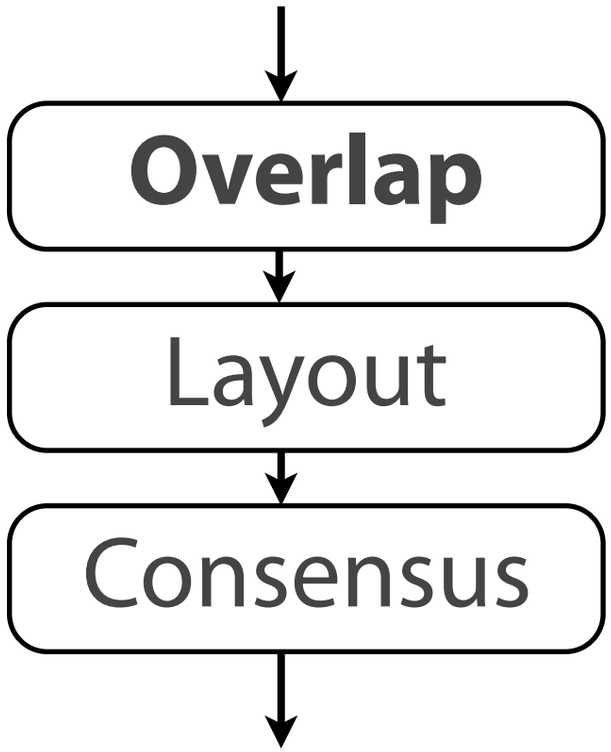
Assembly alternatives

Alternative 1: Overlap-Layout-Consensus (OLC) assembly

Alternative 2: De Bruijn graph (DBG) assembly



Overlap Layout Consensus



Build overlap graph

Coalesce paths into *contigs*

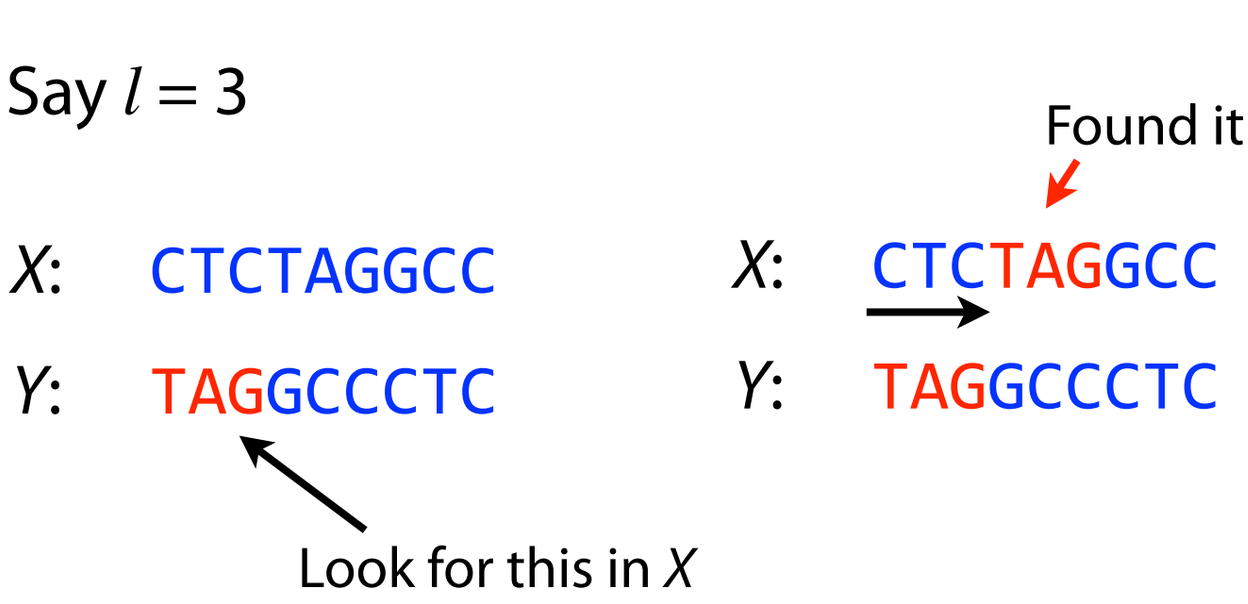
Pick nucleotide sequence for each contig

Finding overlaps

Overlap: Suffix of *X* of length $\geq l$ matches prefix of *Y; l* is given

Naive: look in *X* for occurrences of *Y*'s length-l prefix. Extend matches to the right to confirm whether entire suffix of *X* matches.

Say $l = 3$



See `suffixPrefixMatch` function in HW5 Q4 (Assembly Challenge)

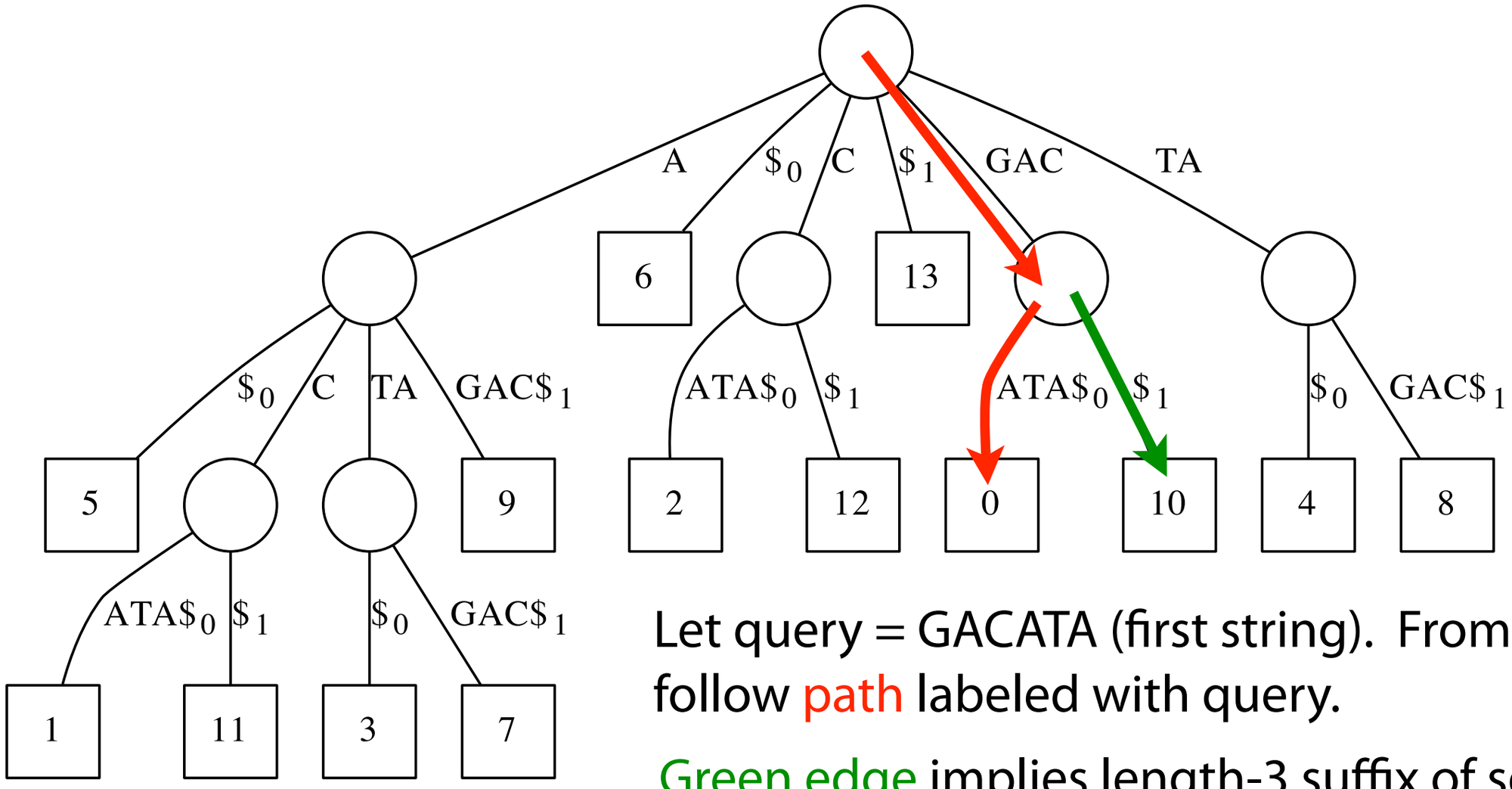
Finding overlaps

With suffix tree?

Given a collection of strings *S*, for each string *x* in *S* find all overlaps involving a prefix of *x* and a suffix of another string *y*

Finding overlaps with suffix tree

Generalized suffix tree for {"GACATA", "ATAGAC"} GACATA$_0$ATAGAC$_1$



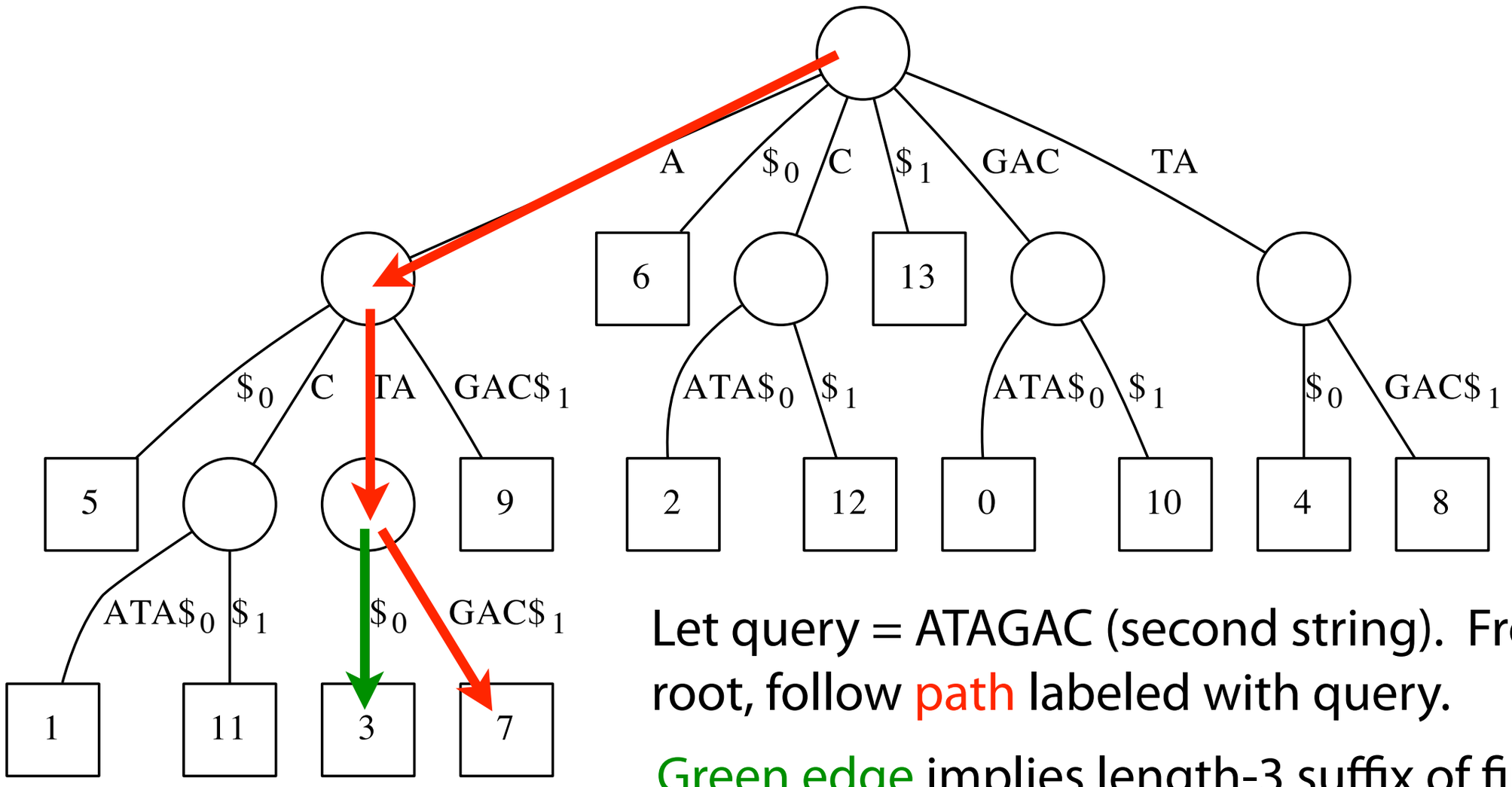
Let query = GACATA (first string). From root, follow path labeled with query.

Green edge implies length-3 suffix of second string equals length-3 prefix of query

ATAGAC
|||
GACATA

Finding overlaps with suffix tree

Generalized suffix tree for {"GACATA", "ATAGAC"} GACATA$_0$ATAGAC$_1$



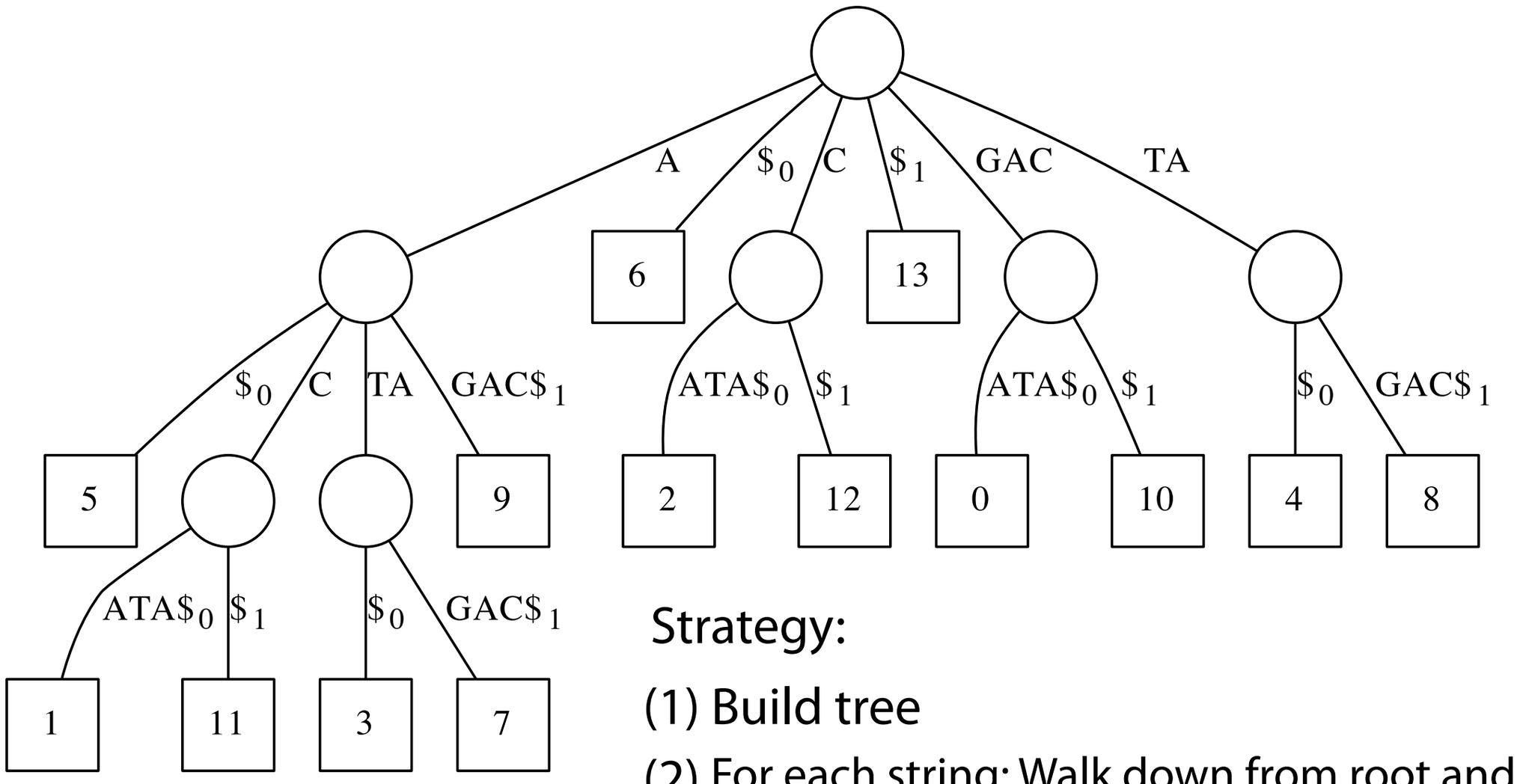
GACATA
|||
ATAGAC

Let query = ATAGAC (second string). From root, follow path labeled with query.

Green edge implies length-3 suffix of first string equals length-3 prefix of query

Finding overlaps with suffix tree

Generalized suffix tree for {"GACATA", "ATAGAC"} GACATA$_0$ATAGAC$_1$

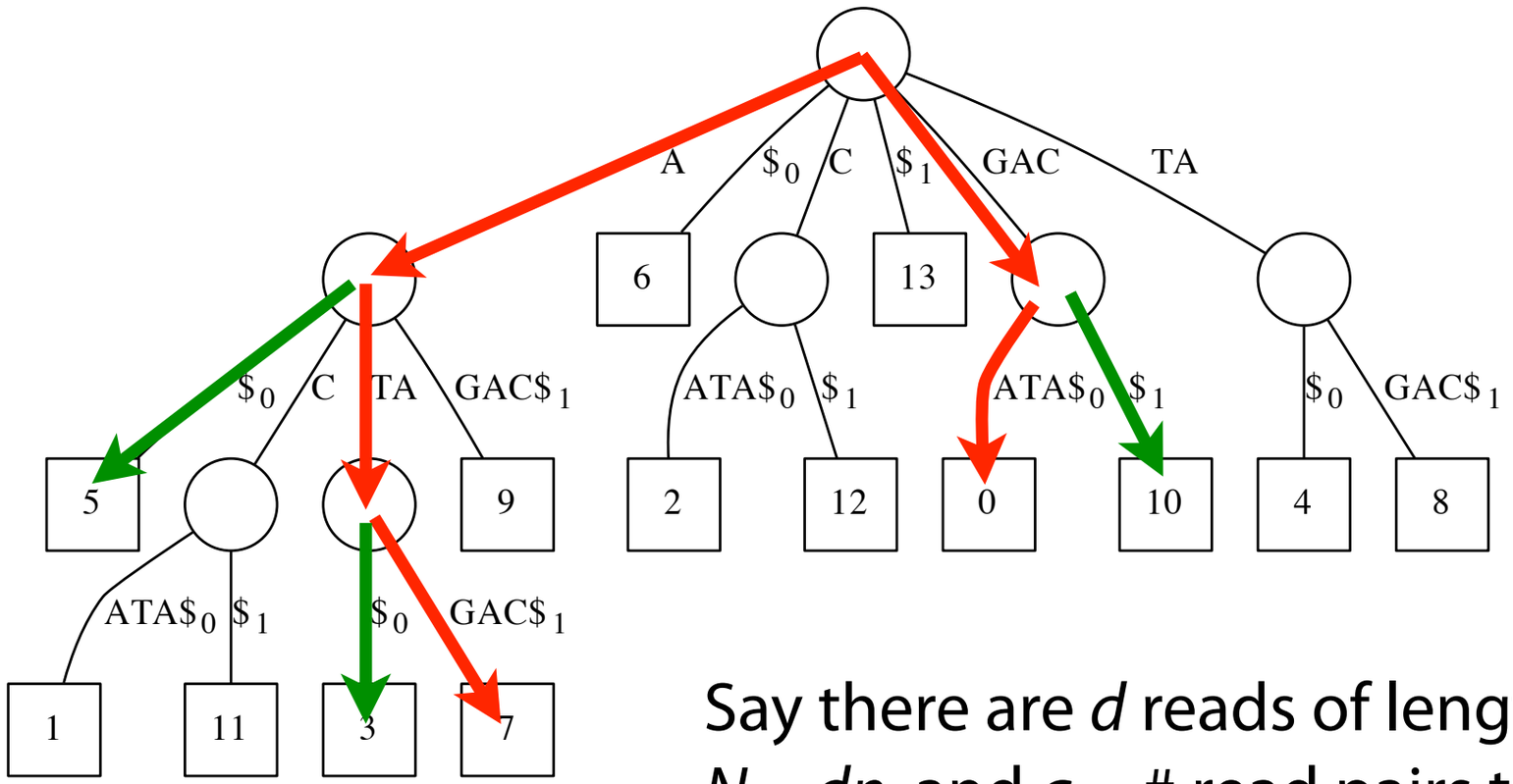


Strategy:

(1) Build tree

(2) For each string: Walk down from root and report any outgoing edge labeled with a separator. Each corresponds to a prefix/suffix match involving prefix of query string and suffix of string ending in the separator.

Finding overlaps with suffix tree



Say there are *d* reads of length *n*, total length $N = dn$, and a = # read pairs that overlap

Assume for given string pair we report only the longest suffix/prefix match

Time to build generalized suffix tree:	O(*N*)
... to walk down red paths:	O(*N*)
... to find & report overlaps (green):	O(*a*)
Overall:	O(*N* + *a*)

Finding overlaps

What about *approximate* suffix/prefix matches?

```
X: CTCGGCCCTAGG
       ||| |||||
Y:     GGCTCTAGGCCC
```

Dynamic programming

Finding overlaps with dynamic programming

```
X: CTCGGCCCTAGG
       |||  |||||
Y:     GGCTCTAGGCCC
```

Use *global alignment* recurrence and score function

$$D[i,j] = \min \begin{cases} D[i-1,j] + s(x[i-1], -) \\ D[i,j-1] + s(-, y[j-1]) \\ D[i-1,j-1] + s(x[i-1], y[j-1]) \end{cases}$$

$s(a,b)$

	A	C	G	T	-
A	0	4	2	4	8
C	4	0	4	2	8
G	2	4	0	4	8
T	4	2	4	0	8
-	8	8	8	8	

How do we force it to find prefix / suffix matches?

Finding overlaps with dynamic programming

$s(a, b)$

	A	C	G	T	-
A	0	4	2	4	8
C	4	0	4	2	8
G	2	4	0	4	8
T	4	2	4	0	8
-	8	8	8	8	

How to initialize first row & column so suffix of *X* aligns to prefix of *Y*?

First column gets 0s
(any suffix of *X* is possible)

First row gets ∞s
(must be a prefix of *Y*)

Backtrace from last row

Y

X		-	G	G	C	T	C	T	A	G	G	C	C	C
	-	0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
	C	0	4	12	20									
	T	0	4	8	14									
	C	0	4	8	8									
	G	0	0	4	12									
	G	0	0	0	8	16	16	24	26	30	36	44	52	60
	C	0	4	4	0	8	16	18	26	30	34	36	44	52
	C	0	4	8	4	0	8	16	22	30	34	34	36	44
	C	0	4	8	8	6	0	10	18	26	34	34	34	36
	T	0	4	8	10	8	8	0	10	18	26	34	36	36
	A	0	2	6	12	14	12	10	0	10	18	26	34	40
	G	0	0	2	10	16	18	16	10	0	10	18	26	34
	G	0	0	0	6	14	20	22	18	10	0	10	18	26

X: CTCGGCCCTAGG
||| |||||
Y: GGCTCTAGGCCC

Finding overlaps with dynamic programming

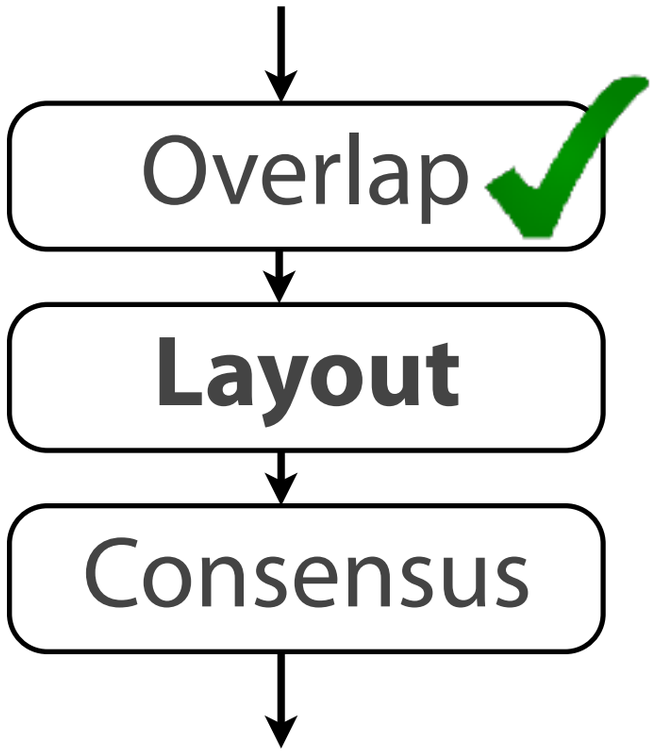
Say there are d reads of length n, total length $N = dn$, and a is total number of pairs with an overlap

# overlaps to try:	$O(d^2)$
Size of each DP matrix:	$O(n^2)$
Overall:	$O(d^2n^2)$, or $O(N^2)$

Contrast $O(N^2)$ with suffix tree: $O(N + a)$, but where a is worst-case $O(d^2)$

Real-world overlappers mix the two; index filters out vast majority of non-overlapping pairs, dynamic programming used for remaining pairs

Overlap Layout Consensus



Build overlap graph

Coalesce paths into *contigs*

Pick nucleotide sequence for each contig

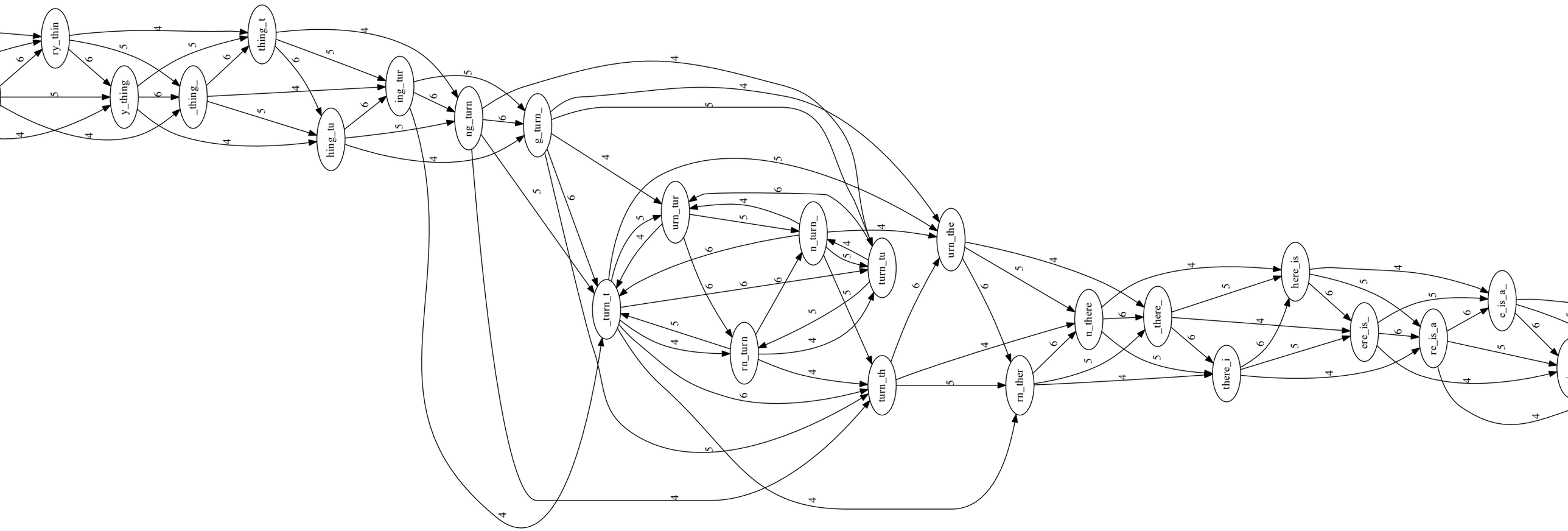
Layout

Overlap graph is big and messy. Contigs don't "pop out" at us.

Below: part of the overlap graph for

`to_every_thing_turn_turn_turn_there_is_a_season`

$l = 4, k = 7$

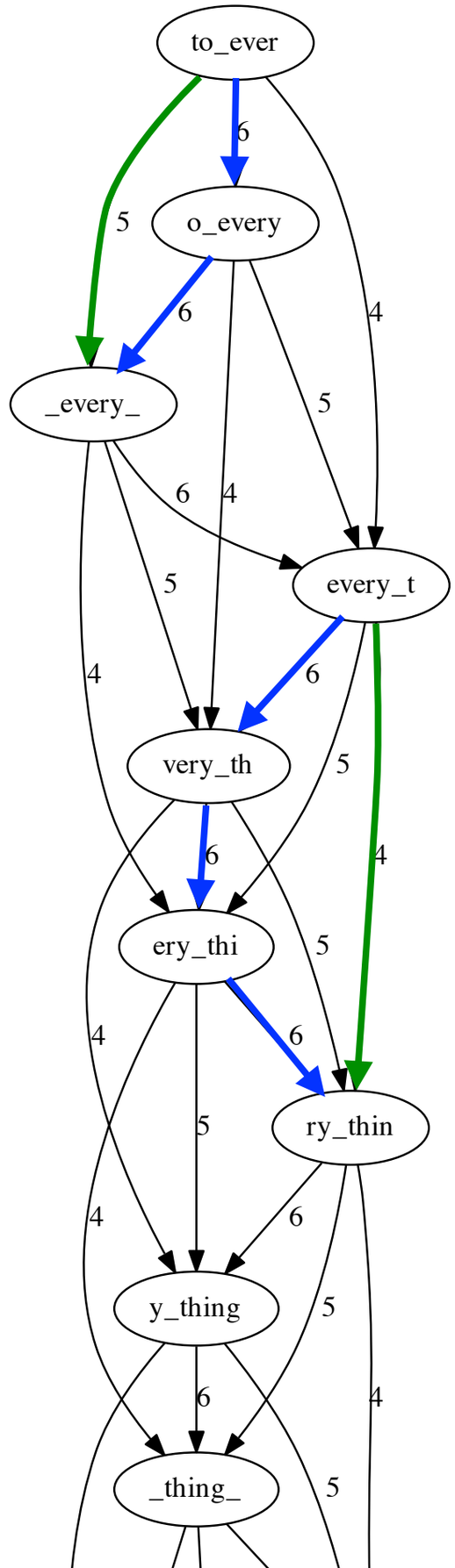


Layout

Anything redundant about this part of the overlap graph?

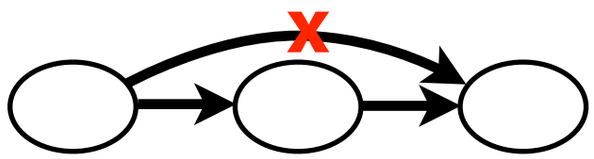
Some edges can be *inferred* (*transitively*) from other edges

E.g. green edge can be inferred from blue

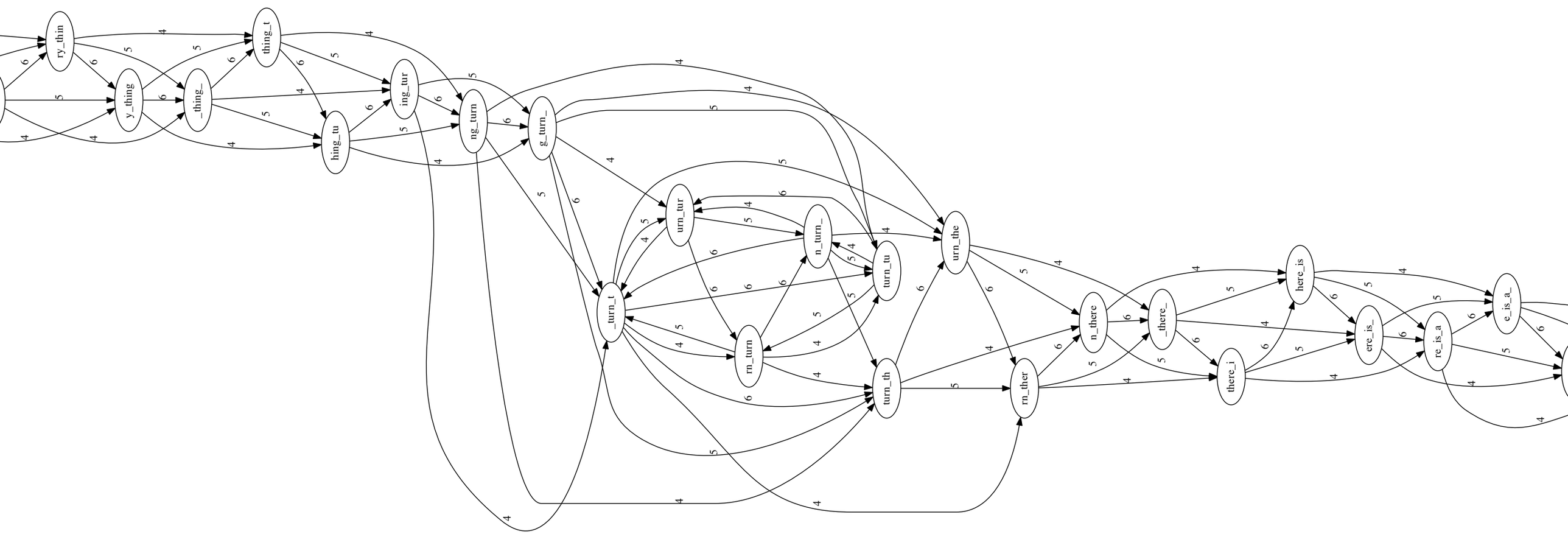


Layout

Remove transitively inferrable edges, starting with edges that skip one node:

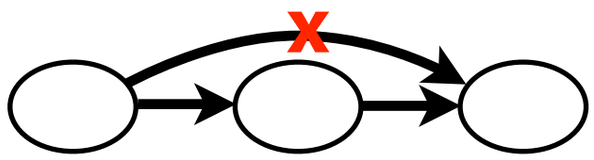


Before:

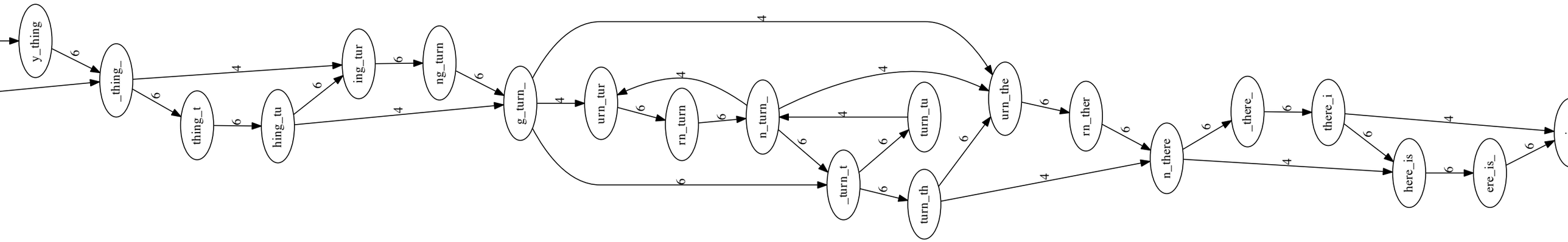


Layout

Remove transitively inferrable edges, starting with edges that skip one node:

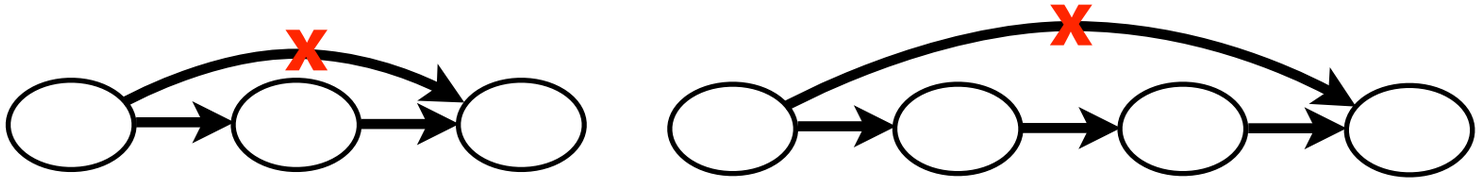


After:

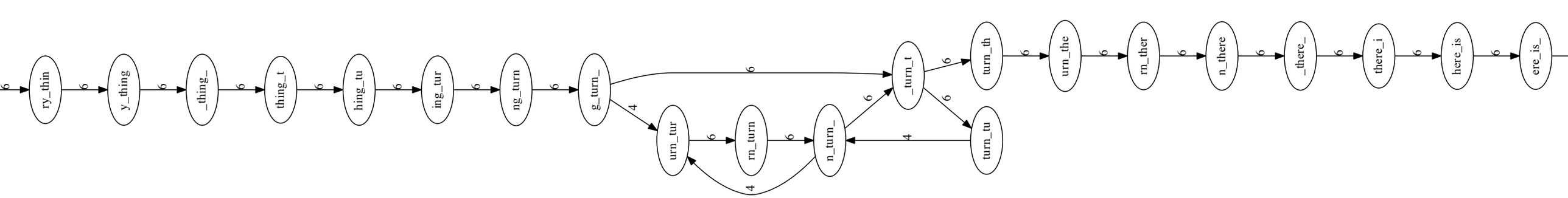


Layout

Now remove edges that skip one or two nodes:



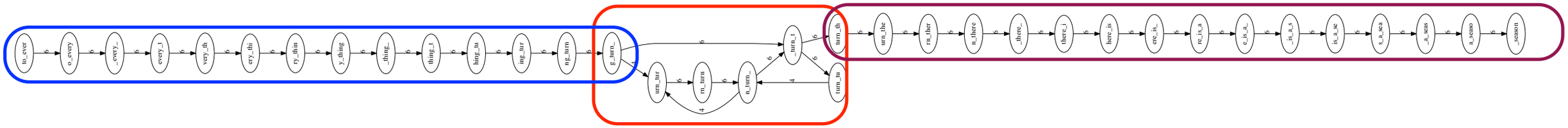
After:



Even simpler

Layout

Emit *contigs* corresponding to the non-branching stretches



Contig 1

`to_every_thing_turn_`

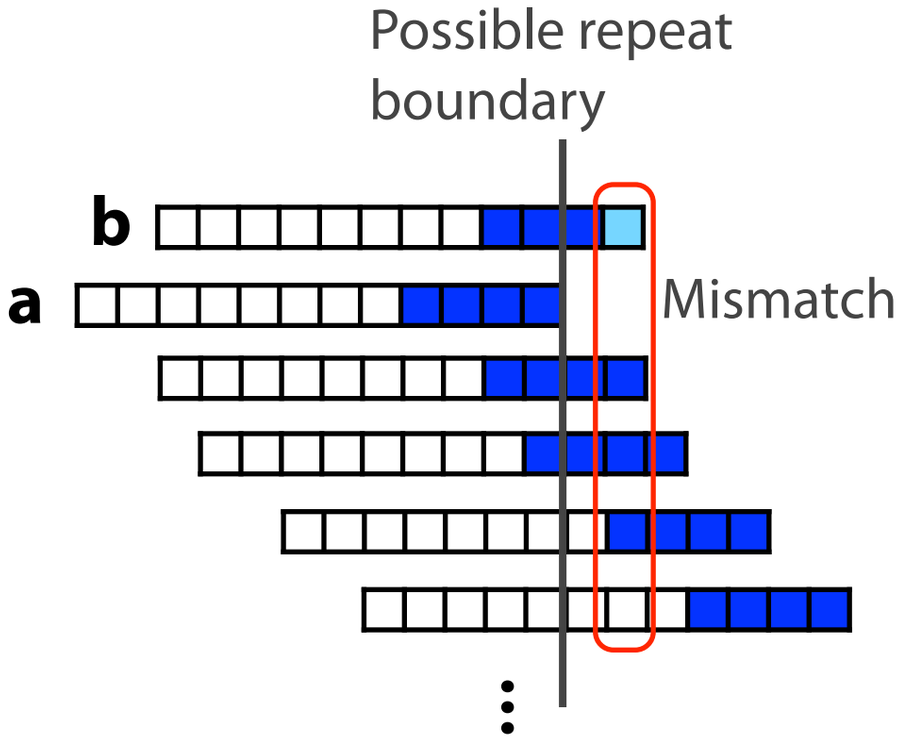
Contig 2

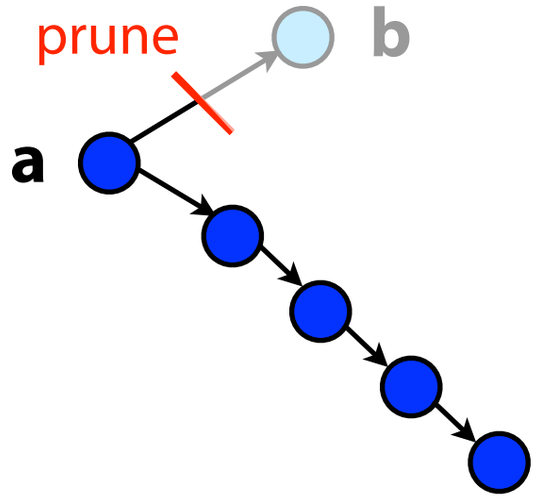
`turn_there_is_a_season`

Unresolvable repeat

Layout

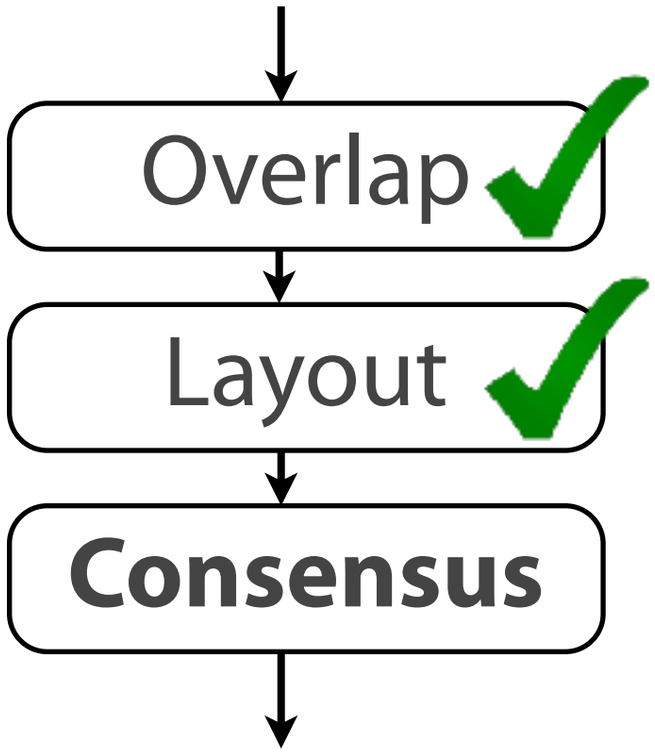
Must handle subgraphs that are spurious, e.g. because of sequencing error





Mismatch could be due to sequencing error or repeat. Since the path through **b** ends abruptly we might conclude it's an error and prune **b**.

Overlap Layout Consensus



Build overlap graph

Bundle stretches of the overlap graph into *contigs*

Pick most likely nucleotide sequence for each contig

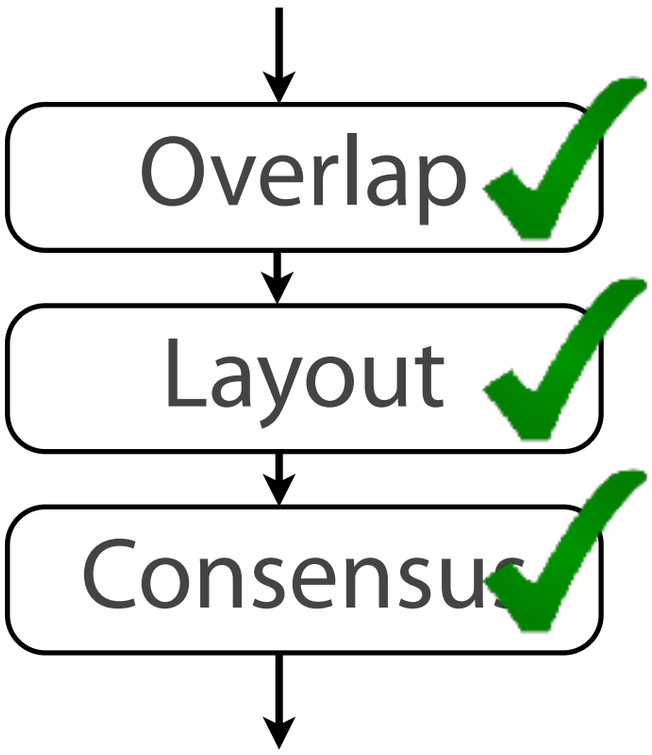
Consensus

Take reads that make up a contig and line them up

Take *consensus*, i.e. majority vote

Complications: (a) sequencing error, (b) ploidy

Overlap Layout Consensus



Build overlap graph

Coalesce paths into *contigs*

Pick nucleotide sequence for each contig

OLC drawbacks

Building overlap graph is slow. We saw $O(N + a)$ and $O(N^2)$ approaches.

Overlap graph is big; one node per read, # edges can grow superlinearly with # reads

Sequencing datasets are ~ 100s of millions or billions of reads

Assembly alternatives

Alternative 1: Overlap-Layout-Consensus (OLC) assembly

Alternative 2: De Bruijn graph (DBG) assembly