# Compacting de Bruijn graphs from sequencing data quickly and in low memory

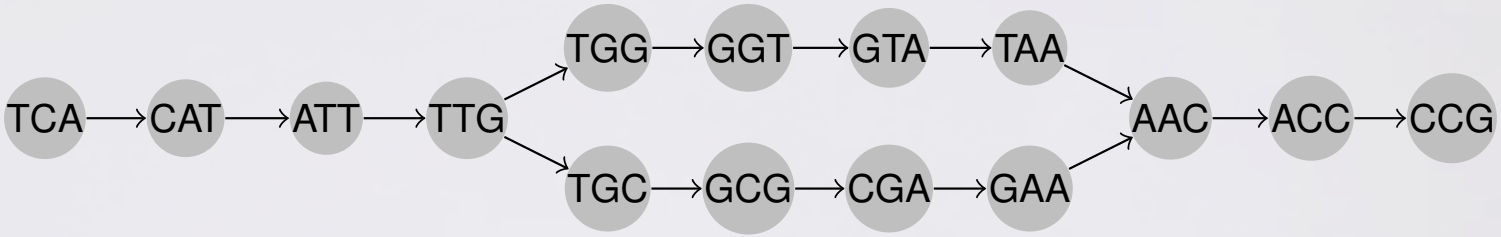
Rayan Chikhi

joint work with A. Limasset (ENS Rennes), P. Medvedev (Penn State)

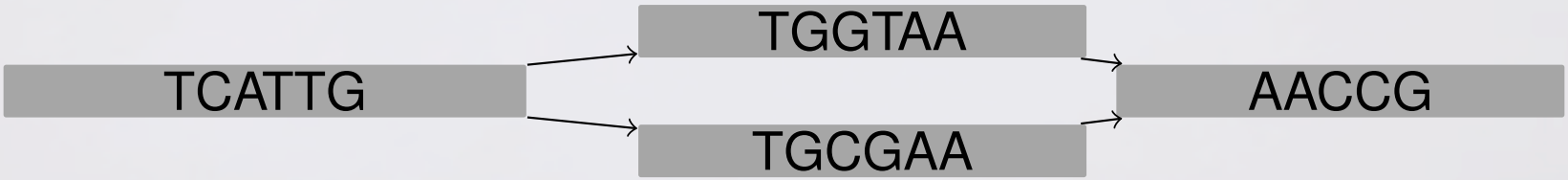
Workshop on Graph Assembly Algorithms for omics data
(adapted from a talk at ISMB 2016)

# Compacted de Bruijn Graph

(non-compacted) de Bruijn graph (nodes = $k$-mers):



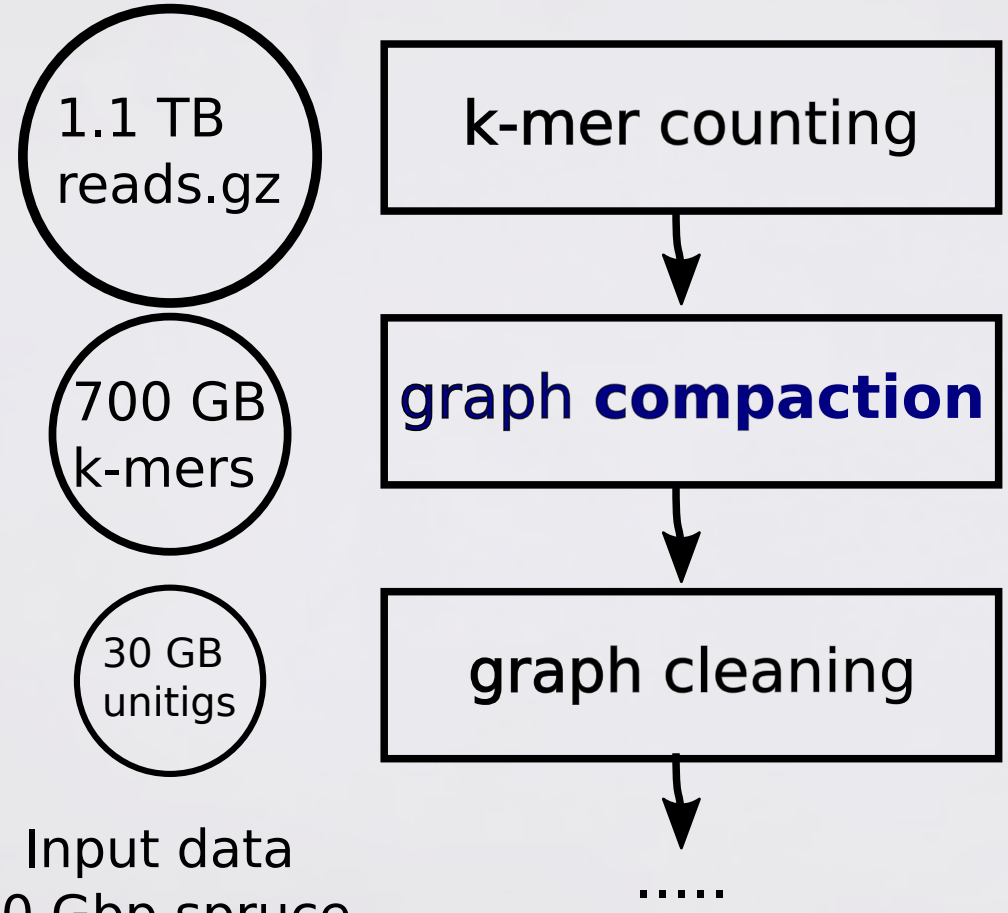

**Compacted** de Bruijn graph:




Each non-branching path becomes a single node (*unitig*).

- no loss of information
- less space

# Steps of de Bruijn graph assemblers




Recent progress,
Stand-alone software
(KMC2, DSK2, Jellyfish2)

Integrated in assemblers,
high-memory or slow
**This work**

Integrated in assemblers,
Heuristics

Input data
20 Gbp spruce
[Birol 2013]

- computationally intensive
- bottlenecks at early stages

# 20 Gbp spruce and 22 Gbp pine

Previous assemblies

- spruce: 2 days, 1380 cores, 4.3 TB RAM [Birol 2013]
- pine: 3 months, 32 cores, 0.8 TB RAM [Zimin 2014]

This work:
improve performance by **orders of magnitude** (up to compaction step)

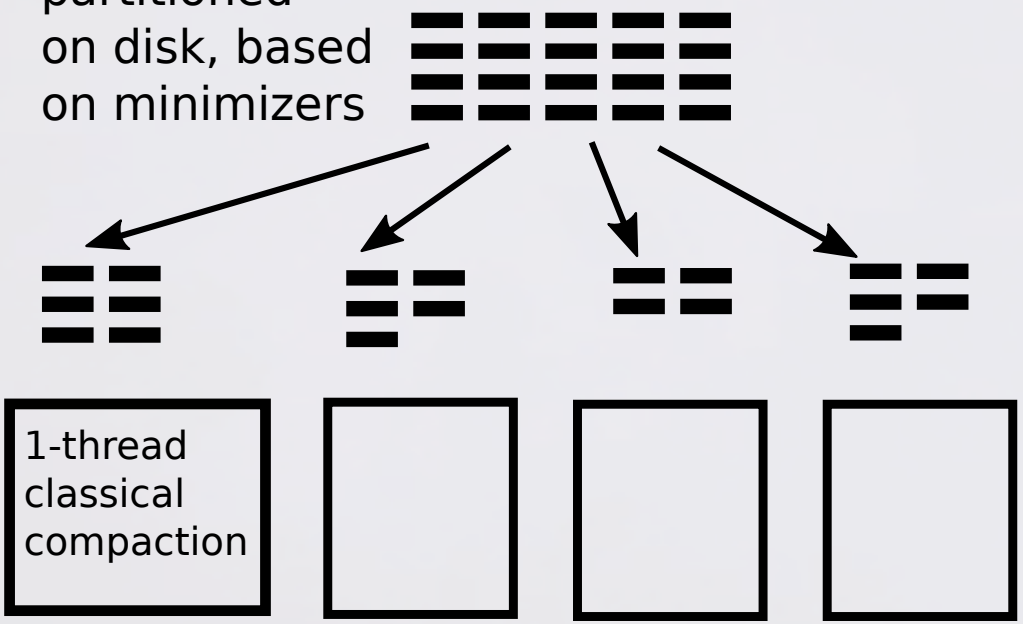
# BCALM 2

Software for *constructing* (not indexing) compacted de Bruijn graphs

Successor of BCALM 1 (single-threaded)

Parallel graph compaction is non-trivial, let's see why..

# Parallel compaction, first attempt



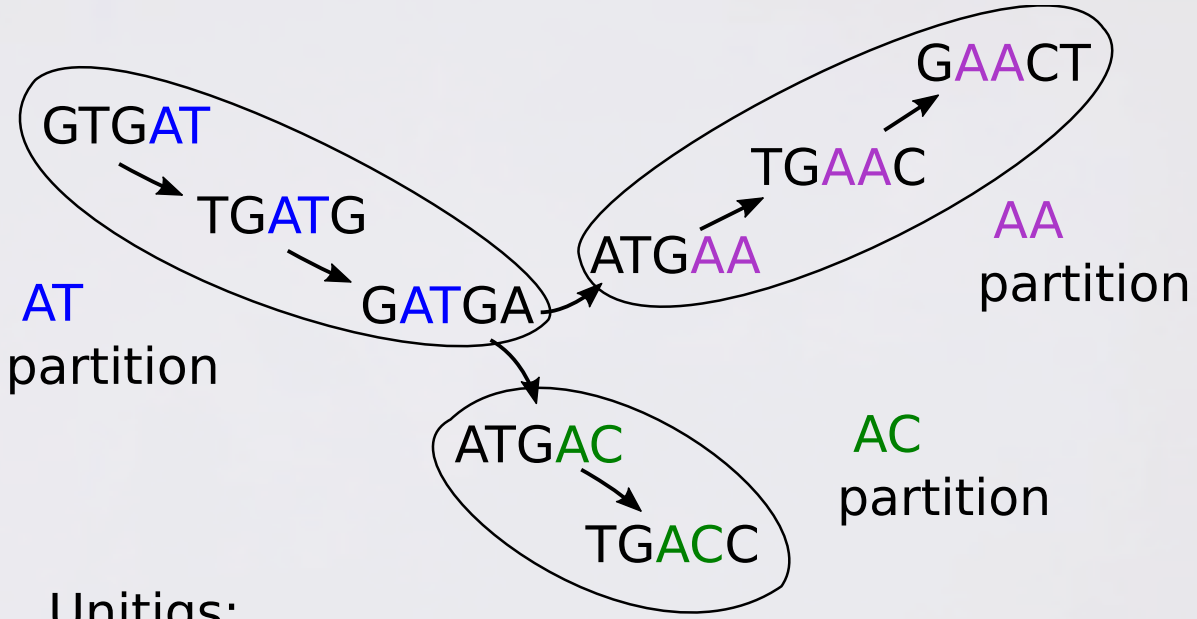

**minimizer** of *s*:
smallest $\ell$-mer in *s*
[Roberts *et al*, 2004]

e.g. ($\ell = 2$, lexicographical order)

```
TGACGGG
 GACGGGT
  ACGGGTC
   CGGGTCA
    GGGTCAG
     GGTCAGA
```

Frequency ordering
$\rightarrow$ better repartition.
[RECOMB'14]

# Compaction of partitions

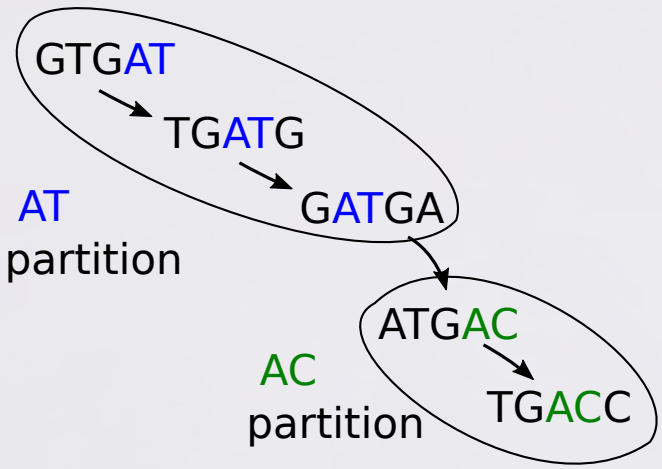



Unitigs:

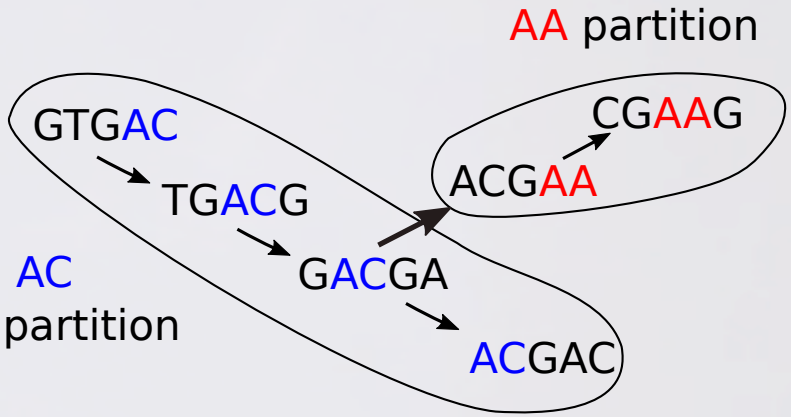
GTGATGA
ATGACC
ATGAACT

*k*-mers are partitioned w.r.t minimizer.
In this case, compacting all partitions returns exactly all the unitigs.

# Can’t just compact partitions




Real unitig:
GTGATGACC

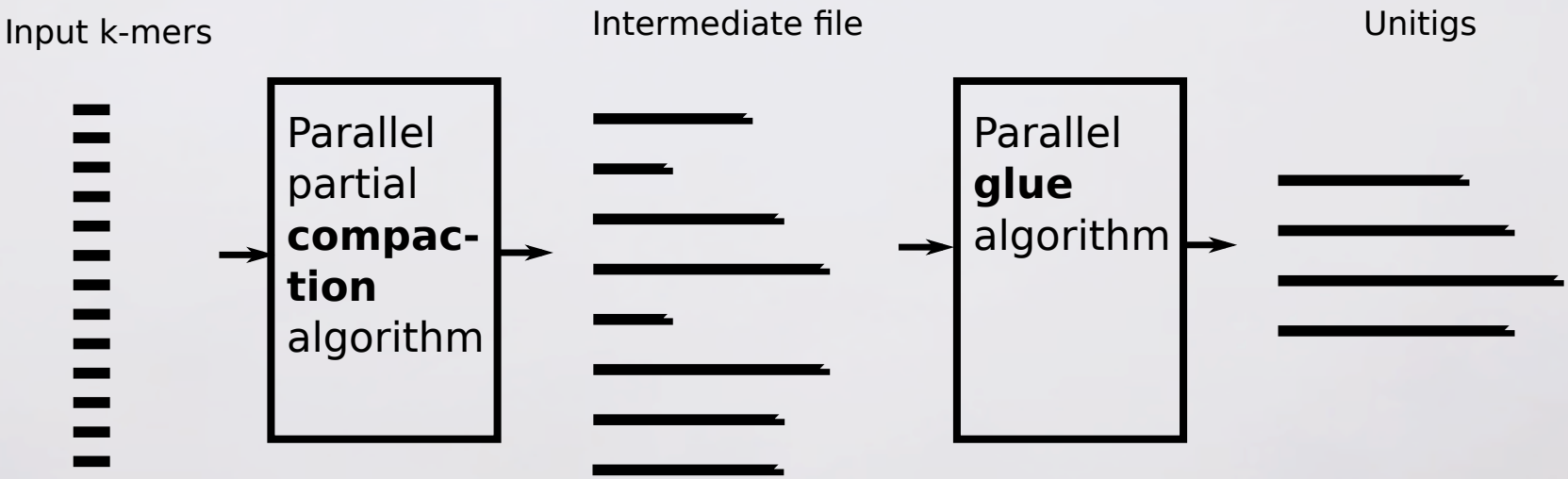



Real unitigs:
GTGACGA
ACGAC
ACGAAG

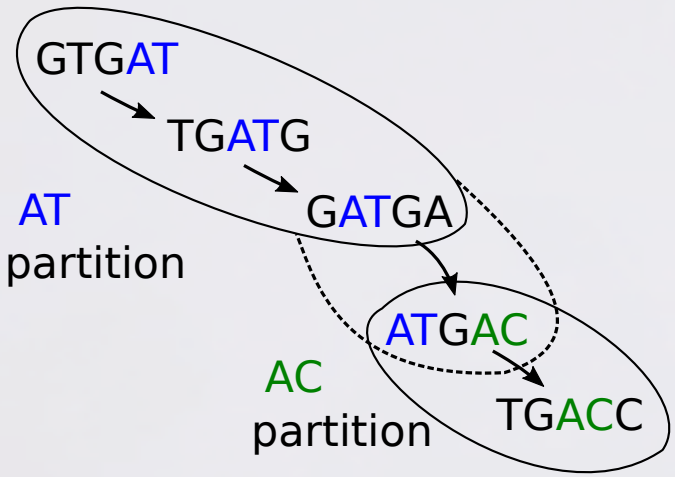
Left: 1 unitig = multiple partitions. (*Merge them?*)
Right: 1 partition = multiple unitigs. (*Split them?*)

# Strategy

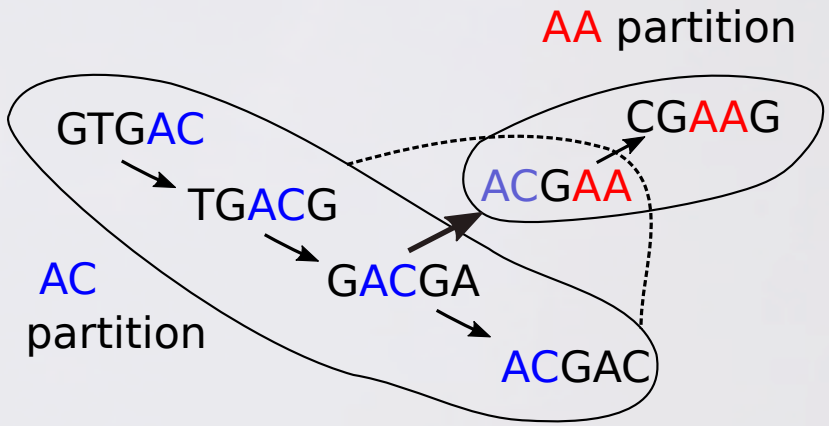
- Put certain $k$-mers into two partitions.
- x is a **doubled kmer** when minimizer($x[1..k-1]$) $\neq$ minimizer($x[2\ldots k]$).

# Partitions with doubled *k*-mers



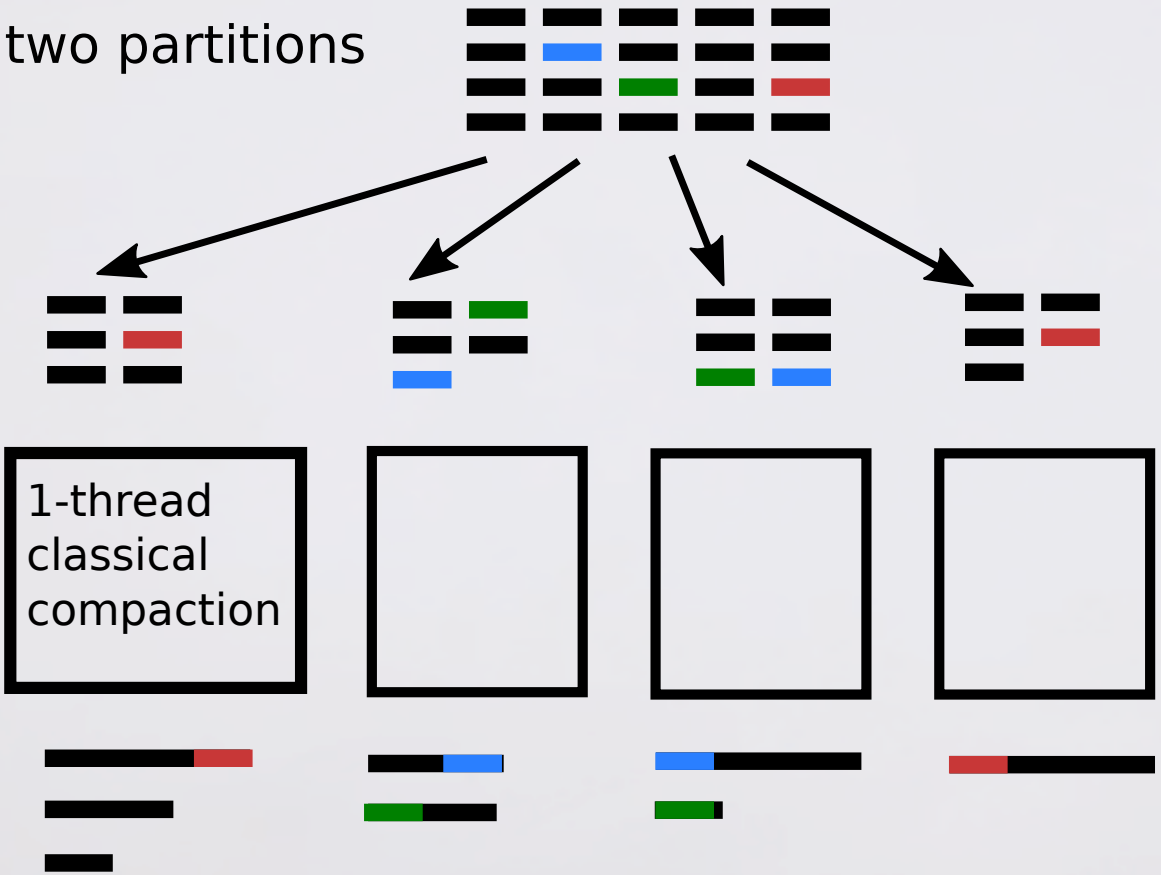

Compacted partitions:
GTGATGAC
ATGACC




Compacted partitions:
GTGACGA
ACGAC
ACGAA
ACGAAG

# BCALM 2's partial compaction module

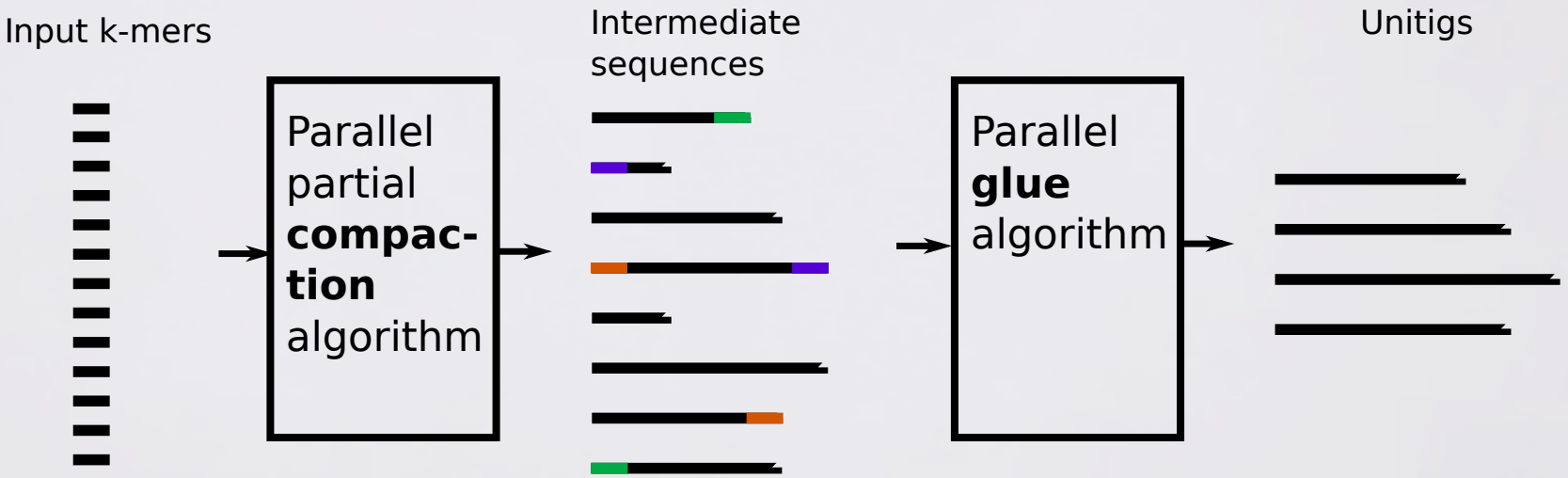
**Doubled kmers**
are inserted in
two partitions




**Lemma** 1:
doubled *k*-mers
appear as
prefixes or
suffixes of
compacted
strings.

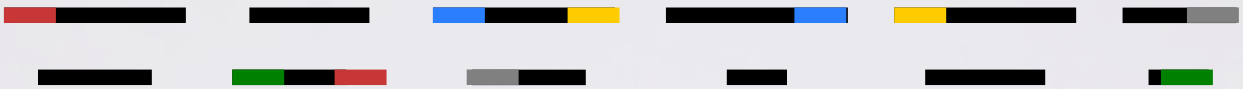
**Lemma** 2:
Gluing together
strings with
matching
doubled *k*-mers
yield unitigs.

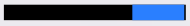
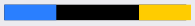
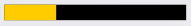
# Big picture


Input k-mers
Parallel partial **compac-tion** algorithm
Intermediate sequences
Parallel **glue** algorithm
Unitigs

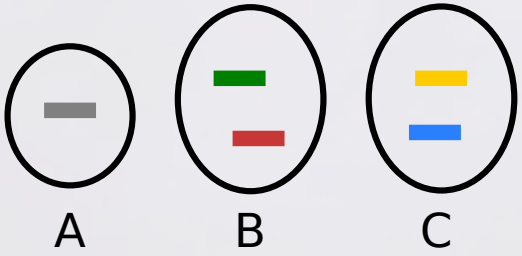
# BCALM 2's glue module

Input sequences

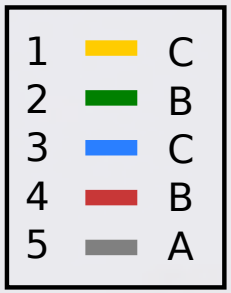


Cannot load all sequences in memory. Need again to partition.
Would like to have  ,  and  in the same partition.

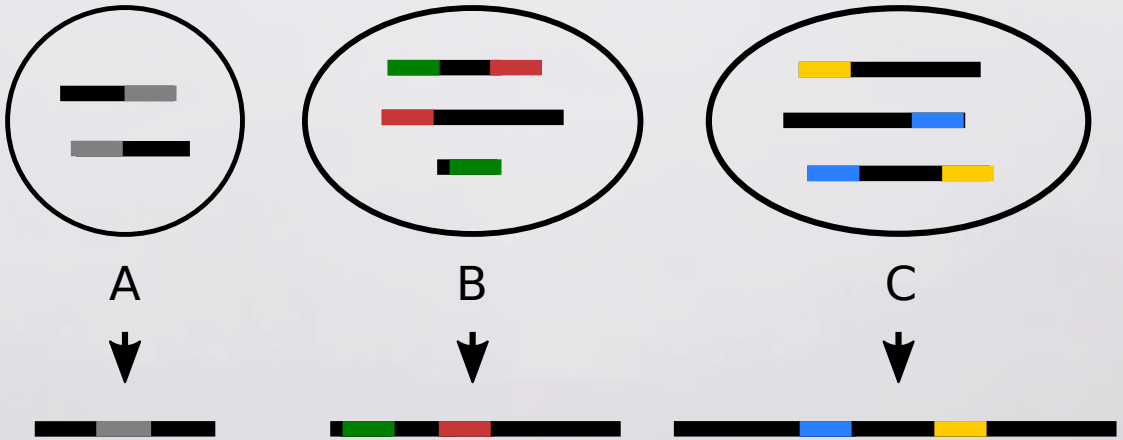
**Union-find**
of doubled kmers




Minimal perfect
hash table

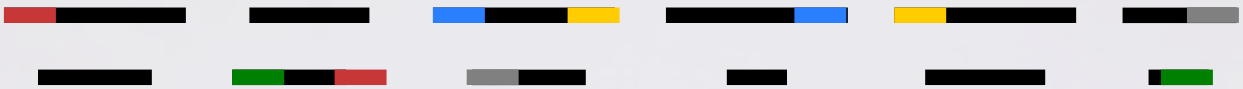


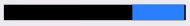
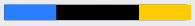
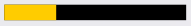

Sequences of
each U-F class
are loaded
and glued
in parallel.



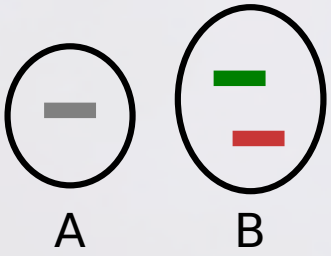
# BCALM 2's glue module

Input sequences



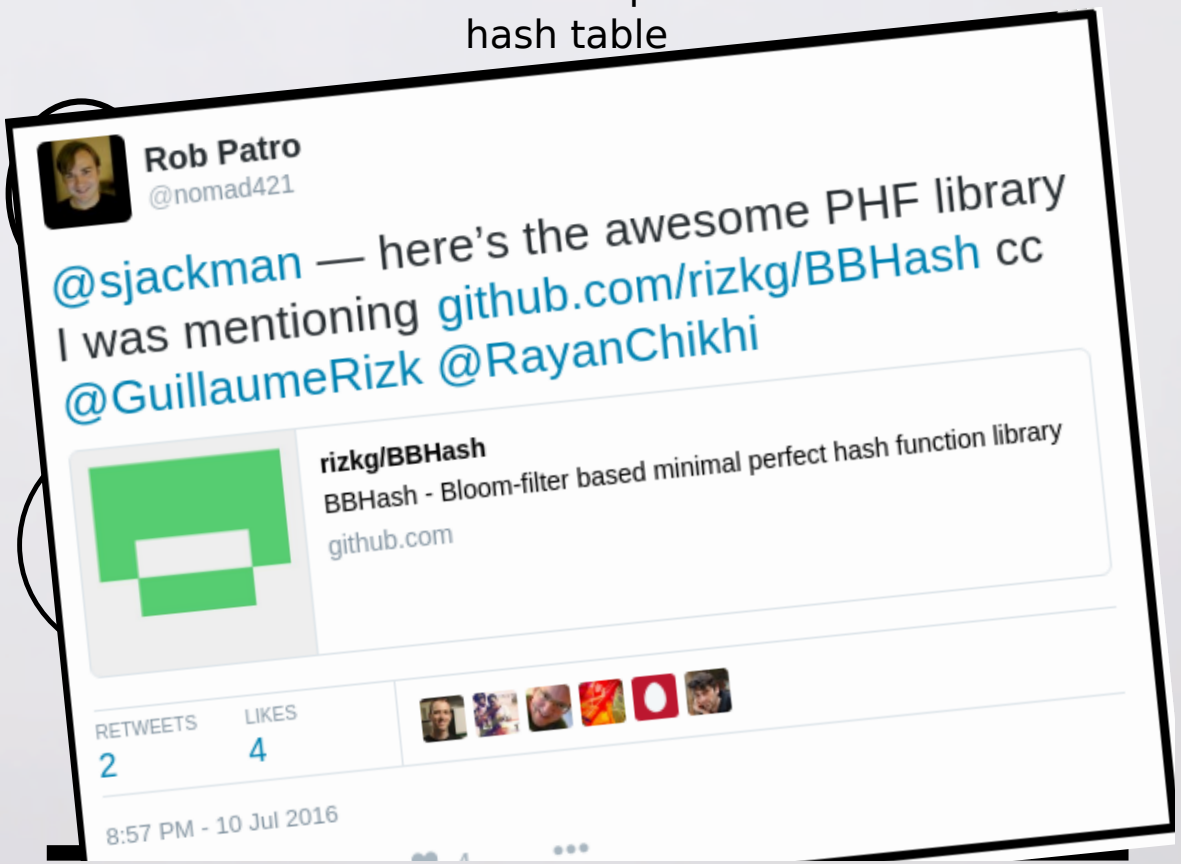
Cannot load all sequences in memory. Need again to partition.
Would like to have  ,  and  in the same partition.

**Union-find**
of doubled kmers




Sequences of each U-F class are loaded and glued in parallel.

Minimal perfect hash table



# 20 Gbp spruce and 22 Gbp pine

Previously,
spruce: 2 days, 1380 cores, 4.3 TB RAM (Abyss) [Birol 2013]
pine: 4 months, 32 cores, 0.8 TB RAM (MaSuRCA) [Zimin 2014]

| BCALM 2 | Pine | Spruce |
|---|---|---|
| Time | 8 h 25 m | 8 h 52 m |
| Memory | 17 GB | 31 GB |
| Unitigs | 30.5 Gbp | 56.0 Gbp |
| # | 257 M | 580 M |

1.1/1.2 TB compressed reads
$k = 61$, abundance cut-off 7, 8/16 threads (pine/spruce)
$k$-mer counting time not included: 2 days, $\leq$ 30 GB memory, DSK 2

# Human dataset

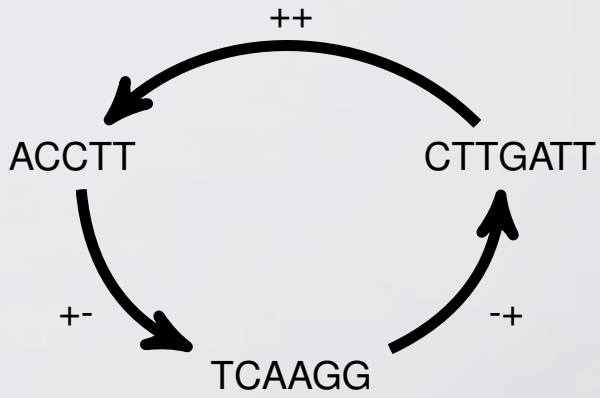
| Human NA18507 | Bcalm 2 | Bcalm 1 | ABySS-P 1.9 |
|---|---|---|---|
| Time | **2 h** | 13 h | 6.5 h |
| Memory | **2.8 GB** | 43 MB | 89 GB |

54 GB compressed reads
$k = 55$, abundance cut-off 7, 16 threads
$k$-mer counting time included in BCALM 1&2: 46 mins, 2 GB memory, DSK 2
Meraculous: 16 hours, $\leq$ 1 TB [Georganas 2014]

# Output in GFA format

- FASTG
- **GFA**
- GFA2

```
H   VN:Z:1.0
S   11  ACCTT
S   12  TCAAGG
S   13  CTTGATT
L   11  +   12  -   4M
L   12  -   13  +   5M
L   11  +   13  +   3M
P   14  11+,12-,13+ 4M,5M
```



# Conclusion

Compacting de Bruijn graphs:

- from **reads** to the **unitig graph**
- efficient
  - 1 hour, 2 GB memory per genome Gbp
- designed as a **small module** for *other* tools (through GFA)
- unitigs to replace $k$-mers in some applications

Observations:

- bottleneck becomes $k$-mer counting again
- not a data structure (construction algorithm, no queries)

Contact:

- @RayanChikhi, @pashadag, @NP_Malfoy on Twitter
- C++ library: `http://gatb.inria.fr`