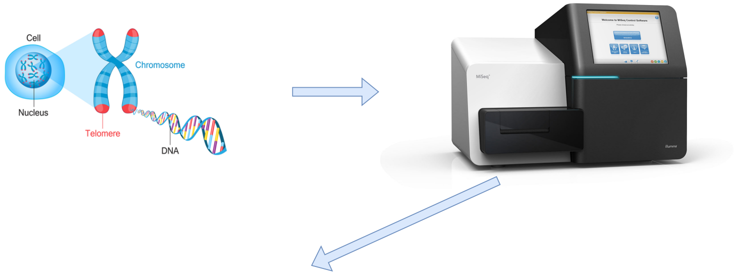
Cdbgtricks: strategies to update a compacted de Bruijn graph

PSC 2024

Presented by:
Khodor Hannoush (Genscale, Inria, Rennes university, France)
Co-authors:
Camille Marchet (BONSAI, CNRS, Lille university, France)
Pierre Peterlongo (Genscale, Inria, Rennes university, France)

Aug 26-27, 2024

Reading the DNA



```
> Sequence_1
CCGAGACGGCCAATTATTGGGGAGTATAGATGTTAGCTAGGCGGTATGGGGTAAACGTTT
GGCTTATTAAGTCTATGACCCCACCTAGAGATTCGTGCAGGTCGGAAAGTTCCATATCTG
ACCTTTCTCGATTTCCCTAAGGTCGCTCGTACGTTTGCCACGGGCACCCCATGATCATTC
TCCGTACAAACCGTGGAGGT
> Sequence_2
GATCAAGGGGCAGCGAGCATTGGCGAATGCTTTAACCGGTAAACGTTGCAAGGTCTCACC
GTGGGAGTGCCACTATGCAGGAGTGCTTCTTAGGGGGGATCGTGACGGTTATGAGTTCTA
CAACAAACCCTAACCCACCCAACCCGCCTACACCCAGCTGTACCCAGTCCCTCGGGAATT
GCTTAAGGAAAGGGTGCAGC
```

Figure: Accessing the DNA

DNA as a string

The alphabet of the genetic language

Computer scientists treat DNA as a language of alphabet $\delta = \{A, C, G, T\}$.

DNA as a string

The alphabet of the genetic language

Computer scientists treat DNA as a language of alphabet $\delta = \{A, C, G, T\}$.

Grammar of the DNA

A DNA sequence is a string over δ. For example ACGCCGTAA.

DNA as a string

The alphabet of the genetic language

Computer scientists treat DNA as a language of alphabet $\delta = \{A, C, G, T\}$.

Grammar of the DNA

A DNA sequence is a string over δ. For example ACGCCGTAA.

k-mer

A k-mer is a string of k characters over δ. ACGGT is a 5-mer.

Storage of DNA sequences

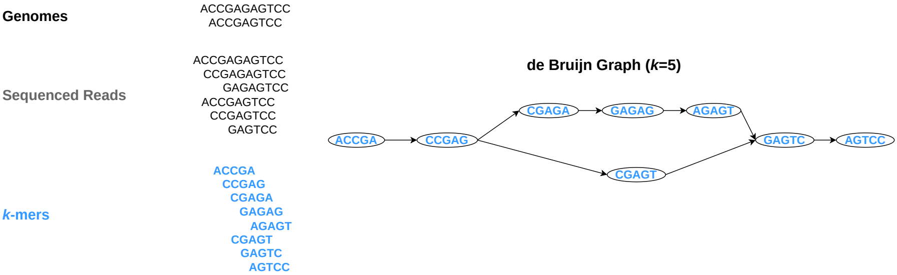


Figure: Storing sequences in a de Bruijn graph

Compacted de Bruijn graph

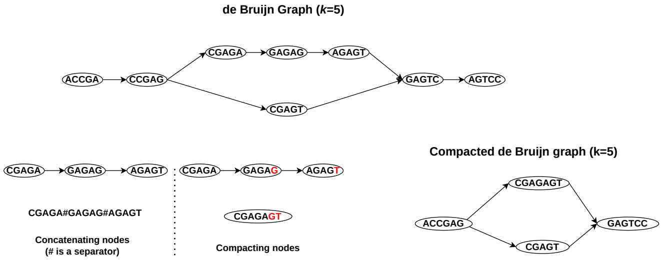


Figure: Compacting the de Bruijn graph

Unitig

A unitig is a maximal non-branching path $p = \{f, v_1, v_2, ..., v_n, l\}$ such that every node $s \in p$ has only one in-coming and one out-going edges except for f and l.

Objective: Augmenting a cDBG

Given a compacted de Bruijn graph G and a new genome sequence S not in G, we need to add S to G.

Objective: Augmenting a cDBG

Given a compacted de Bruijn graph G and a new genome sequence S not in G, we need to add S to G.

The update operation should be time-efficient as we need to support large datasets.

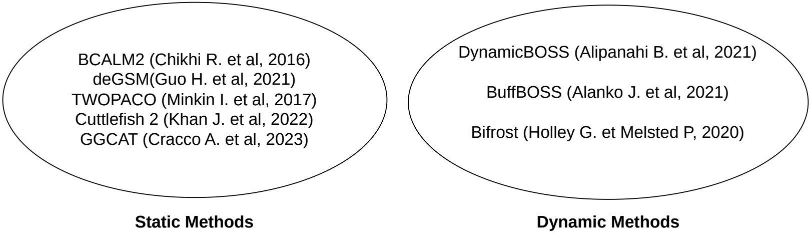
Objective: Augmenting a cDBG

Given a compacted de Bruijn graph G and a new genome sequence S not in G, we need to add S to G.

The update operation should be time-efficient as we need to support large datasets.

We need to be able to identify the regions in the graph where the update will take place.

Existing Methods



BCALM2 (Chikhi R. et al, 2016)
deGSM(Guo H. et al, 2021)
TWOPACO (Minkin I. et al, 2017)
Cuttlefish 2 (Khan J. et al, 2022)
GGCAT (Cracco A. et al, 2023)

Static Methods

DynamicBOSS (Alipanahi B. et al, 2021)

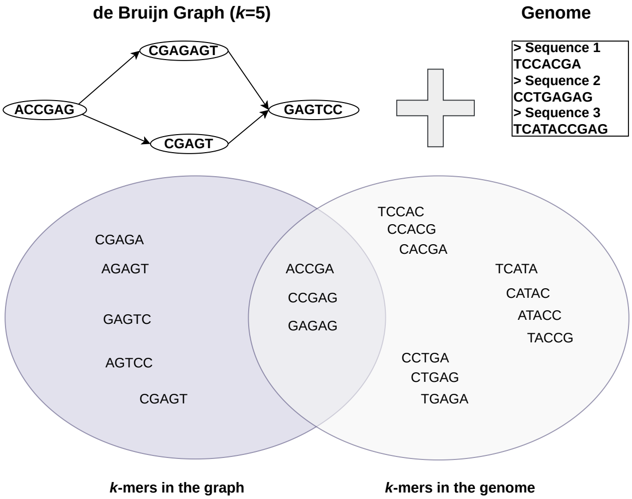
BuffBOSS (Alanko J. et al, 2021)

Bifrost (Holley G. et Melsted P, 2020)

Dynamic Methods

Figure: Methods to construct and update a de Bruijn graph

Graph and genomes as sets of *k*-mers



de Bruijn Graph (*k*=5)

ACCGAG
CGAGAGT
CGAGT
GAGTCC

Genome

> Sequence 1
TCCACGA
> Sequence 2
CCTGAGAG
> Sequence 3
TCATACCGAG

CGAGA
AGAGT
GAGTC
AGTCC
CGAGT

ACCGA
CCGAG
GAGAG

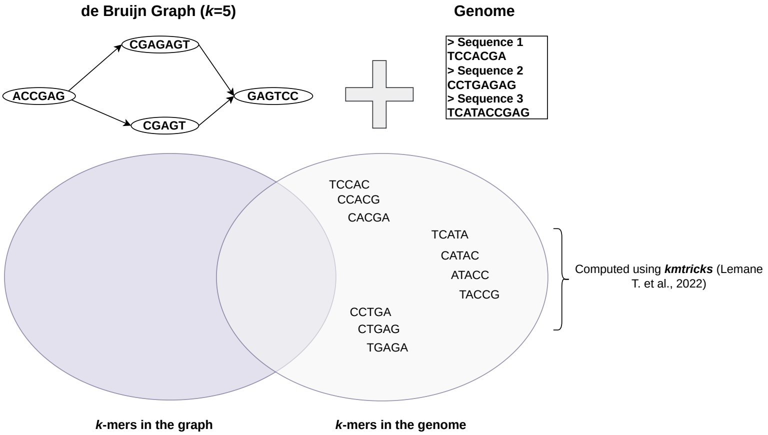
TCCAC
CCACG
CACGA
TCATA
CATAC
ATACC
TACCG
CCTGA
CTGAG
TGAGA

***k*-mers in the graph**

***k*-mers in the genome**

Figure: Venn diagram of the graph and the new genome

Set of new *k*-mers



de Bruijn Graph (*k*=5)

ACCGAG
CGAGAGT
CGAGT
GAGTCC

Genome

> Sequence 1
TCCACGA
> Sequence 2
CCTGAGAG
> Sequence 3
TCATACCGAG

TCCAC
CCACG
CACGA
TCATA
CATAC
ATACC
TACCG
CCTGA
CTGAG
TGAGA

Computed using ***kmtricks*** (Lemane T. et al., 2022)

***k*-mers in the graph**

***k*-mers in the genome**

Figure: Set of new *k*-mers

Possible cases

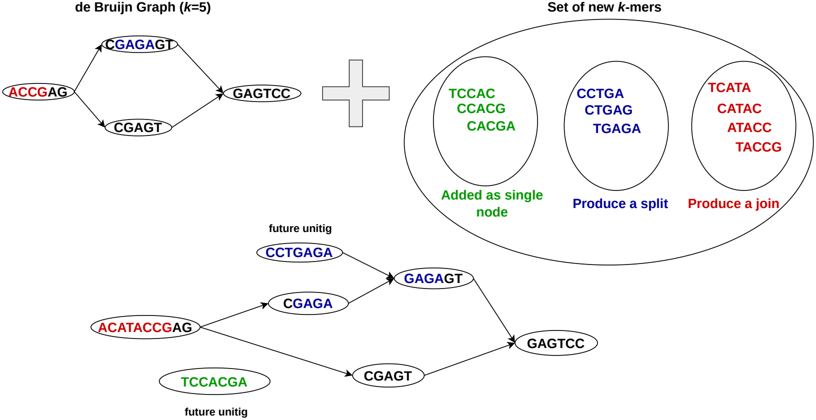


Figure: Possible cases while compacting the set of new k-mers

Indexing the $(k-1)$-mers

For each added k-mer we need to query its $(k-1)$-mers prefix and suffix.

Indexing the $(k-1)$-mers

For each added k-mer we need to query its $(k-1)$-mers prefix and suffix.

The query should be performed in constant time.

Indexing the $(k-1)$-mers

For each added k-mer we need to query its $(k-1)$-mers prefix and suffix.

The query should be performed in constant time.

Solution: index the $(k-1)$-mer of the graph.

Indexing the $(k-1)$-mers: **Drawback**

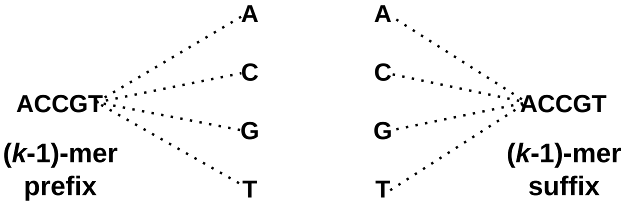


Figure: Maximum number of occurrences of a $(k-1)$-mer

Indexing the $(k-1)$-mers: **Drawback**

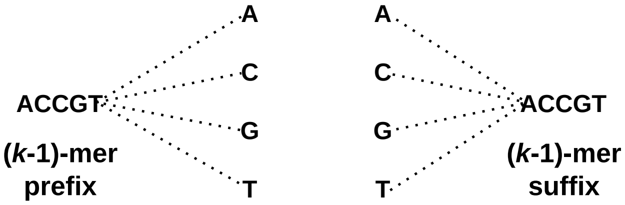


Figure: Maximum number of occurrences of a $(k-1)$-mer

A $(k-1)$-mer x may have up to 8 occurrences in the graph.

Indexing the $(k-1)$-mers: **Drawback**

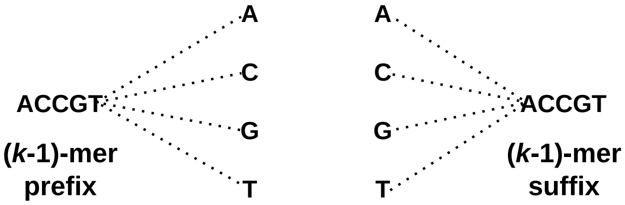


Figure: Maximum number of occurrences of a $(k-1)$-mer

A $(k-1)$-mer x may have up to 8 occurrences in the graph.

It is not efficient to allocate 8 slots for each $(k-1)$-mer in the graph especially for graphs that contains billions of $(k-1)$-mers.

Indexing k-mers to query $(k-1)$-mers

The k-mers in a compacted de Bruijn graph are unique, hence it is more memory efficient to index k-mers instead of $(k-1)$-mers.

Indexing k-mers to query $(k-1)$-mers

The k-mers in a compacted de Bruijn graph are unique, hence it is more memory efficient to index k-mers instead of $(k-1)$-mers.

Given a $(k-1)$-mer x, we can query the 8 possible k-mers for which x is either a suffix or a prefix.

Indexing the k-mers

First possible solution

- Index the k-mers in a hash table.

Indexing the k-mers

First possible solution

- Index the k-mers in a hash table.
- **Drawback**: We need to store the k-mers and their positions in the graph.

Indexing the k-mers

First possible solution

- Index the k-mers in a hash table.
- **Drawback**: We need to store the k-mers and their positions in the graph.

Second possible solution

- Compute one minimal perfect hash function (MPHF) f from the k-mers of the graph. An MPHF is a hash function that bijectively maps a set of N keys to the set $\{i \mid 0 \leq i < N\}$.

Indexing the *k*-mers

First possible solution

- Index the *k*-mers in a hash table.
- **Drawback**: We need to store the *k*-mers and their positions in the graph.

Second possible solution

- Compute one minimal perfect hash function (MPHF) f from the *k*-mers of the graph. An MPHF is a hash function that bijectively maps a set of N keys to the set $\{i \mid 0 \leq i < N\}$.
- **Drawback**: The MPHF is static, for every addition of *k*-mers we need to re-compute f.

Indexing the k-mers

First possible solution

- Index the k-mers in a hash table.
- **Drawback**: We need to store the k-mers and their positions in the graph.

Second possible solution

- Compute one minimal perfect hash function (MPHF) f from the k-mers of the graph. An MPHF is a hash function that bijectively maps a set of N keys to the set $\{i \mid 0 \leq i < N\}$.
- **Drawback**: The MPHF is static, for every addition of k-mers we need to re-compute f.

Our solution

Solution: partition the k-mers into **buckets**, and compute one MPHF for each bucket.

Constructing future unitigs

From the set of new *k*-mers, we construct the future unitigs (**funitigs**) that get added to the graph.

Updating the index

The position of *k*-mers in the unitigs that went into splits or joins are changed.

Updating the index

The position of *k*-mers in the unitigs that went into splits or joins are changed.

Re-computing MPHFs

The MPHF of the buckets to which we added new *k*-mers are re-computed.

Cdbgtricks: product of this work

These functionalities (indexing and updating a compacted de Bruijn graph) are available in one open source software **Cdbgtricks**.

Cdbgtricks: product of this work

These functionalities (indexing and updating a compacted de Bruijn graph) are available in one open source software **Cdbgtricks**.

Cdbgtricks is available at https://github.com/khodor14/Cdbgtricks

Test Data and Tools

- Datasets
  - 7054 *E. coli* genomes (1 *E. coli genome* $\approx 4Mb$).
  - 100 human genomes (1 *human genome* $\approx 3Gb$).

Test Data and Tools

- Datasets
  - 7054 *E. coli* genomes (1 $E.\ coli\ genome \approx 4Mb$).
  - 100 human genomes (1 $human\ genome \approx 3Gb$).
- Competitor tool of Cdbgtricks
  - Bifrost update [Holley, G. and Melsted, P,2020].
  - GGCAT (only for reconstruction)[Andrea Cracco and Alexandru I. Tomescu,2022].

Test Data and Tools

- Datasets
  - 7054 *E. coli* genomes (1 $E.\ coli\ genome \approx 4Mb$).
  - 100 human genomes (1 $human\ genome \approx 3Gb$).
- Competitor tool of Cdbgtricks
  - Bifrost update [Holley, G. and Melsted, P,2020].
  - GGCAT (only for reconstruction)[Andrea Cracco and Alexandru I. Tomescu,2022].
- Experimental settings
  - The value of k is 31.
  - The experiments were executed using 32 threads.

Results of benchmark - *E.coli* genomes

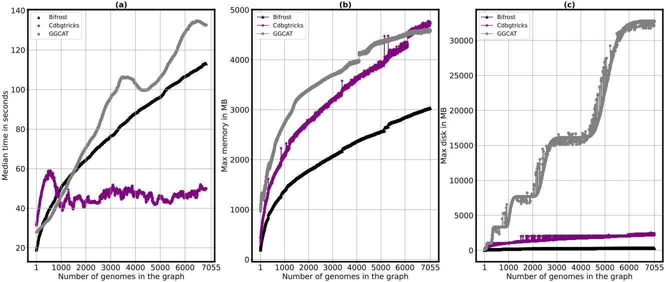


Figure: Results of *E.coli* genomes dataset. Time (a), Memory (b) and Disk (c).

Cdbtricks is 2 to 3x faster than Bifrost and GGCAT on the *E. coli* genomes dataset.

Results of benchmark - Human genomes

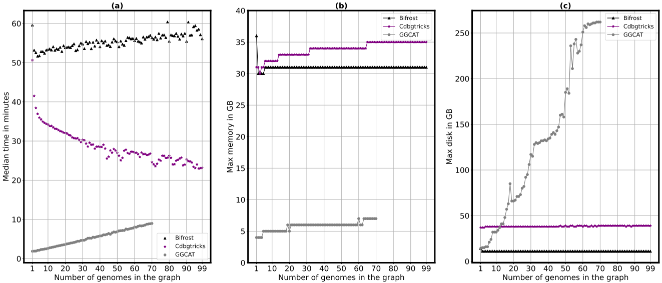


Figure: Results of human genomes dataset. Time (a), Memory (b) and Disk (c).

Cdbtricks is 2 to 3x faster than Bifrost and has the potential to be faster than GGCAT on a larger human genomes dataset.

Conclusion

Cdbgtricks

A novel method to update a compacted de Bruijn graph and its index

Conclusion

Cdbgtricks

A novel method to update a compacted de Bruijn graph and its index

Indexing a compacted de Bruijn graph

Cdbgtricks indexes a compacted de Bruijn graph

Conclusion

Cdbgtricks

A novel method to update a compacted de Bruijn graph and its index

Indexing a compacted de Bruijn graph

Cdbgtricks indexes a compacted de Bruijn graph

Performance

Cdbgtricks outperforms the state-of-the-art tools dedicated to the creation of the update of a compacted de Bruijn graph on a data set of thousands of *E. coli* genomes and another data set of 100 *E. coli* genomes.

Perspectives

Optimise *k*-mer partitioning

The aim is to be able to locate consecutive *k*-mers in the same bucket.

Colored and compacted de Bruijn graph

Store the references (**colors**) of *k*-mers in **Cdbgtricks** and implement a mechanism to update the set of colors.

Acknowledgements

My Supervisors

Thanks to my supervisors Camille Marchet (BONSAI, CNRS, Lille) and Pierre Peterlongo (Genscale, Inria, Rennes).

Alpaca Project

This work is part of the Alpaca project that has received funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No 956229.

Questions?

Preliminaries

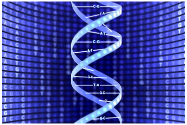


Figure: Double helix structure of DNA sequence.

Preliminaries

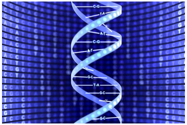


Figure: Double helix structure of DNA sequence.

Reverse Complement of DNA Sequence

The reverse complement $\bar{s}$ of a DNA sequence is obtained by reversing it and replacing each character by its complement (A:T, C:G,T:A,G:C). The reverse complement of ACCT is AGGT.

Preliminaries

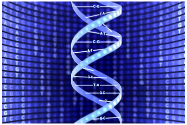


Figure: Double helix structure of DNA sequence.

Reverse Complement of DNA Sequence

The reverse complement $\bar{s}$ of a DNA sequence is obtained by reversing it and replacing each character by its complement (A:T, C:G,T:A,G:C). The reverse complement of ACCT is AGGT.

Canonical Sequence

The canonical sequence of A DNA sequence s is the smallest sequence in lexicographical order between s and its reverse complement $\bar{s}$. The canonical sequence of ACCT is ACCT.

Bit encoding of a unitig

A	**00**
C	**01**
G	**10**
T	**11**

ACCGATTATTA

0001011000111100111100

Figure: Encoding unitig using 2 bits per character

Minimizer

***k*=8** **ACCGTAAT**

	***m*-mer**	**hash value**	
	ACC	**6**	
	CCG	**9**	
***m*=3**	**CGT**	**3**	***m*-mer with the smallest hash value**
	GTA	**11**	
	TAA	**5**	
	AAT	**8**	**lexicographically smallest *m*-mer**

Figure: Hash based versus lexicographic based minimizer

Minimal perfect hashing

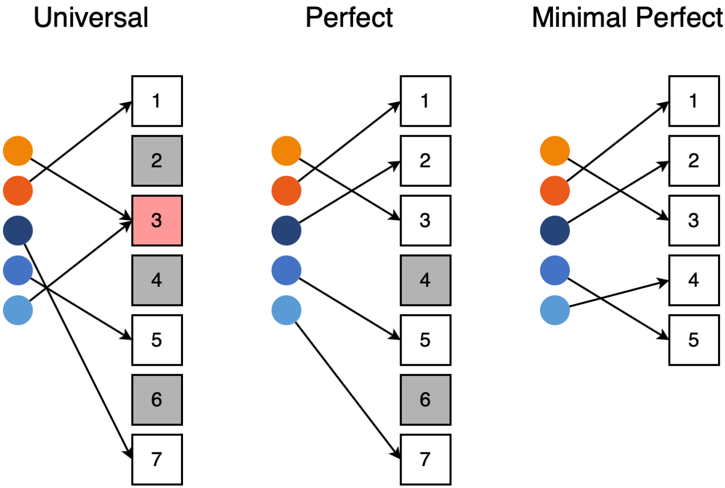
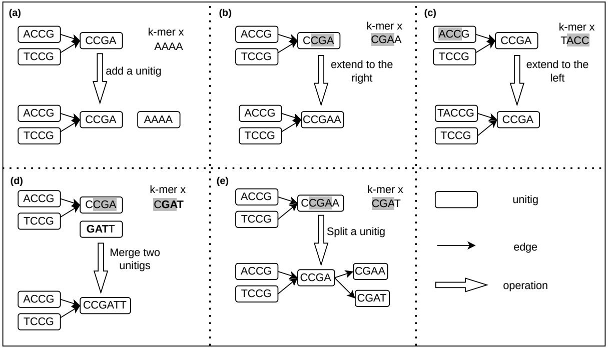


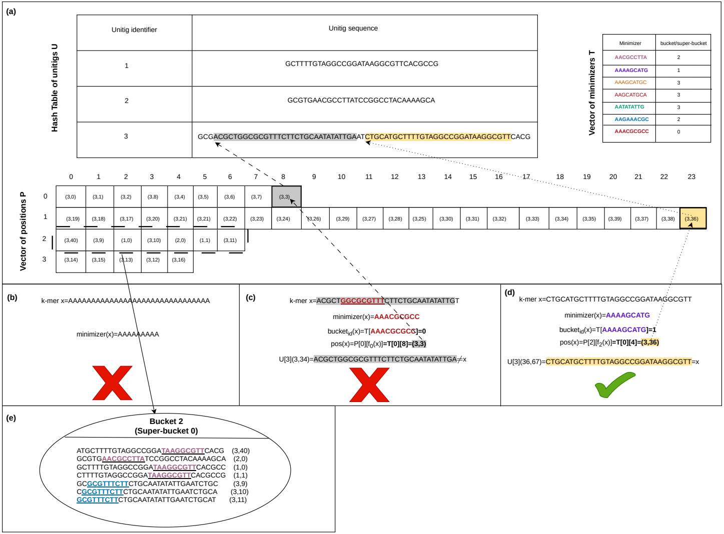
Figure: Types of hashing

Operations on adding a k-mer



- a. The k-mer x forms a new unitig
- b. A unitig of the graph is extended from right
- c. A unitig of the graph is extended from left
- d. Two unitigs in the graph get merge
- e. A unitig is split into two unitigs

Graph index in Cdbgtricks



Unitig identifier	Unitig sequence
1	GCTTTTGTAGGCCGGATAAGGCGTTCACGCCG
2	GCGTGAACGCCTTATCCGGCCTACAAAAGCA
3	GCGACGCTGGCGCGTTTCTTCTGCAATATATTGAATCTGCATGCTTTTGTAGGCCGGATAAGGCGTTCACG

Minimizer	bucket/super-bucket
AACGCCTTA	2
AAAAGCATG	1
AAAGCATGC	3
AAGCATGCA	3
AATATATTG	3
AAGAAACGC	2
AAACGCGCC	0

	0	1	2	3	4	5	6	7	8	9	10	11
0	(3,0)	(3,1)	(3,2)	(3,8)	(3,4)	(3,5)	(3,6)	(3,7)	(3,3)			
1	(3,19)	(3,18)	(3,17)	(3,20)	(3,21)	(3,21)	(3,22)	(3,23)	(3,24)	(3,26)	(3,29)	(3,27)
2	(3,40)	(3,9)	(1,0)	(3,10)	(2,0)	(1,1)	(3,11)					
3	(3,14)	(3,15)	(3,13)	(3,12)	(3,16)							

	12	13	14	15	16	17	18	19	20	21	22	23
0												
1	(3,28)	(3,25)	(3,30)	(3,31)	(3,32)	(3,33)	(3,34)	(3,35)	(3,39)	(3,37)	(3,38)	(3,36)
2												
3												

(b)

k-mer x=AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

minimizer(x)=AAAAAAAAA

✗

(c)

k-mer x=ACGCTGGCGCGTTTCTTCTGCAATATATTGT

minimizer(x)=AAACGCGCC

bucket_{id}(x)=T[AAACGCGCC]=0

pos(x)=P[0][f_0(x)]=T[0][8]=(3,3)

U[3](3,34)=ACGCTGGCGCGTTTCTTCTGCAATATATTGA≠x

✗

(d)

k-mer x=CTGCATGCTTTTGTAGGCCGGATAAGGCGTT

minimizer(x)=AAAAGCATG

bucket_{id}(x)=T[AAAAGCATG]=1

pos(x)=P[2][f_2(x)]=T[0][4]=(3,36)

U[3](36,67)=CTGCATGCTTTTGTAGGCCGGATAAGGCGTT=x

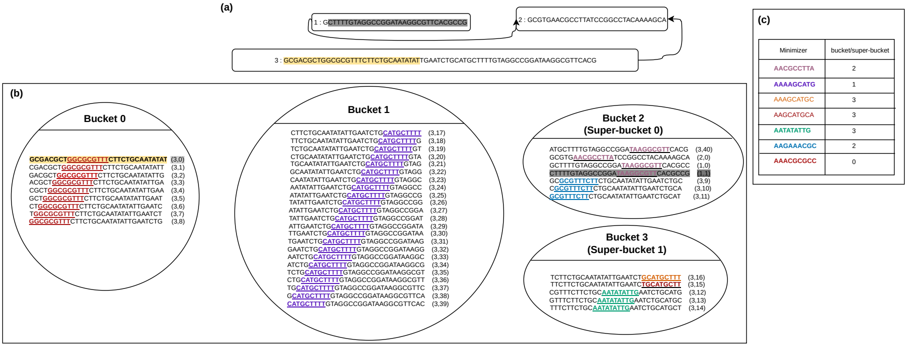
✓

(e)

Bucket 2
(Super-bucket 0)

ATGCTTTTGTAGGCCGGATAAGGCGTTCACG	(3,40)
GCGTGAACGCCTTATCCGGCCTACAAAAGCA	(2,0)
GCTTTTGTAGGCCGGATAAGGCGTTCACGCC	(1,0)
CTTTTGTAGGCCGGATAAGGCGTTCACGCCG	(1,1)
GCGCGTTTCTTCTGCAATATATTGAATCTGC	(3,9)
CGCGTTTCTTCTGCAATATATTGAATCTGCA	(3,10)
GCGTTTCTTCTGCAATATATTGAATCTGCAT	(3,11)

Partitioning the *k*-mers



(a)

1 : GCTTTTGTAGGCCGGATAAGGCGTTCACGCCG

2 : GCGTGAACGCCTTATCCGGCCTACAAAAGCA

3 : GCGACGCTGGCGCGTTTCTTCTGCAATATATTGAATCTGCATGCTTTTGTAGGCCGGATAAGGCGTTCACG

(b)

Bucket 0

GCGACGCTGGCGCGTTTCTTCTGCAATATAT	(3,0)
CGACGCTGGCGCGTTTCTTCTGCAATATATT	(3,1)
GACGCTGGCGCGTTTCTTCTGCAATATATTG	(3,2)
ACGCTGGCGCGTTTCTTCTGCAATATATTGA	(3,3)
CGCTGGCGCGTTTCTTCTGCAATATATTGAA	(3,4)
GCTGGCGCGTTTCTTCTGCAATATATTGAAT	(3,5)
CTGGCGCGTTTCTTCTGCAATATATTGAATC	(3,6)
TGGCGCGTTTCTTCTGCAATATATTGAATCT	(3,7)
GGCGCGTTTCTTCTGCAATATATTGAATCTG	(3,8)

Bucket 1

CTTCTGCAATATATTGAATCTGCATGCTTTT	(3,17)
TTCTGCAATATATTGAATCTGCATGCTTTTG	(3,18)
TCTGCAATATATTGAATCTGCATGCTTTTGT	(3,19)
CTGCAATATATTGAATCTGCATGCTTTTGTA	(3,20)
TGCAATATATTGAATCTGCATGCTTTTGTAG	(3,21)
GCAATATATTGAATCTGCATGCTTTTGTAGG	(3,22)
CAATATATTGAATCTGCATGCTTTTGTAGGC	(3,23)
AATATATTGAATCTGCATGCTTTTGTAGGCC	(3,24)
ATATATTGAATCTGCATGCTTTTGTAGGCCG	(3,25)
TATATTGAATCTGCATGCTTTTGTAGGCCGG	(3,26)
ATATTGAATCTGCATGCTTTTGTAGGCCGGA	(3,27)
TATTGAATCTGCATGCTTTTGTAGGCCGGAT	(3,28)
ATTGAATCTGCATGCTTTTGTAGGCCGGATA	(3,29)
TTGAATCTGCATGCTTTTGTAGGCCGGATAA	(3,30)
TGAATCTGCATGCTTTTGTAGGCCGGATAAG	(3,31)
GAATCTGCATGCTTTTGTAGGCCGGATAAGG	(3,32)
AATCTGCATGCTTTTGTAGGCCGGATAAGGC	(3,33)
ATCTGCATGCTTTTGTAGGCCGGATAAGGCG	(3,34)
TCTGCATGCTTTTGTAGGCCGGATAAGGCGT	(3,35)
CTGCATGCTTTTGTAGGCCGGATAAGGCGTT	(3,36)
TGCATGCTTTTGTAGGCCGGATAAGGCGTTC	(3,37)
GCATGCTTTTGTAGGCCGGATAAGGCGTTCA	(3,38)
CATGCTTTTGTAGGCCGGATAAGGCGTTCAC	(3,39)

Bucket 2
(Super-bucket 0)

ATGCTTTTGTAGGCCGGATAAGGCGTTCACG	(3,40)
GCGTGAACGCCTTATCCGGCCTACAAAAGCA	(2,0)
GCTTTTGTAGGCCGGATAAGGCGTTCACGCC	(1,0)
CTTTTGTAGGCCGGATAAGGCGTTCACGCCG	(1,1)
GCGCGTTTCTTCTGCAATATATTGAATCTGC	(3,9)
CGCGTTTCTTCTGCAATATATTGAATCTGCA	(3,10)
GCGTTTCTTCTGCAATATATTGAATCTGCAT	(3,11)

Bucket 3
(Super-bucket 1)

TCTTCTGCAATATATTGAATCTGCATGCTTT	(3,16)
TTCTTCTGCAATATATTGAATCTGCATGCTT	(3,15)
CGTTTCTTCTGCAATATATTGAATCTGCATG	(3,12)
GTTTCTTCTGCAATATATTGAATCTGCATGC	(3,13)
TTTCTTCTGCAATATATTGAATCTGCATGCT	(3,14)

(c)

Minimizer	bucket/super-bucket
AACGCCTTA	2
AAAAGCATG	1
AAAGCATGC	3
AAGCATGCA	3
AATATATTG	3
AAGAAACGC	2
AAACGCGCC	0

Figure: Partitioning the *k*-mers into buckets using their minimizers (the smallest *m*-mer according to some order where $0 < m < k$)

Querying a compacted de Bruijn graph

Presence/absence

The Cdbgtricks index helps validate whether or not the k-mers of a query sequence are in the graph.

Uni-MEMs

If some k-mers are present, Cdbgtricks can output the unitig id and the offset in this unitig where these present k-mer appear.

Test Data and Tools

- Data sets
  - 15,006 *E. coli*.
  - 10 human genomes.
- Competitor tool of Cdbgtricks
  - Bifrost [Holley, G. and Melsted, P,2020].
  - GGCAT (only for reconstruction)[Andrea Cracco and Alexandru I. Tomescu,2022].
  - SSHash (Pibiri, G., SSHash).

Results query

Dataset	Query type	Tool	Memory (MB)	Disk (MB)	time (mm:ss)
E. coli	Negative	Cdbgtricks	4723	0	10:02
		Bifrost	4362	0	06:43
		SSHash	725	0	00:07
		GGCAT	560	3325	01:32
	Positive	Cdbgtricks	4724	0	02:15
		Bifrost	4362	0	01:43
		SSHash	725	0	01:10
		GGCAT	644	2978	01:26
human	Negative	Cdbgtricks	25520	0	12:25
		Bifrost	27376	0	11:37
		SSHash	6090	0	00:07
		GGCAT	615	6861	4:55
	Positive	Cdbgtricks	25520	0	04:23
		Bifrost	27376	0	06:37
		SSHash	6090	0	01:14
		GGCAT	746	7053	05:04

Performing splits and joins

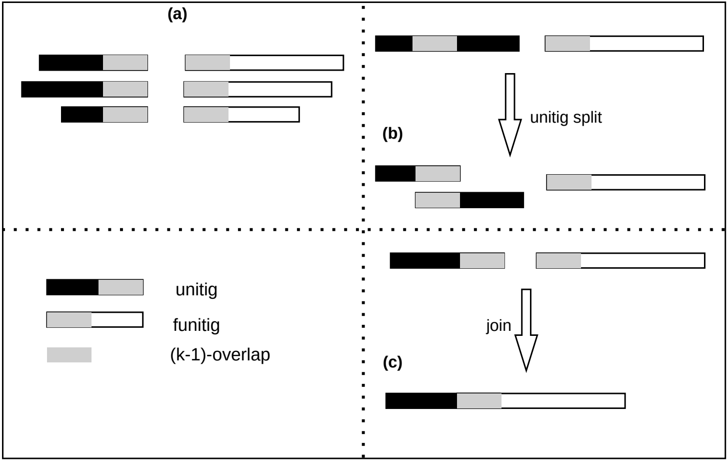


Figure: **a**. Unitigs remain unchanged. **b**. The split case. **c**. The join case.