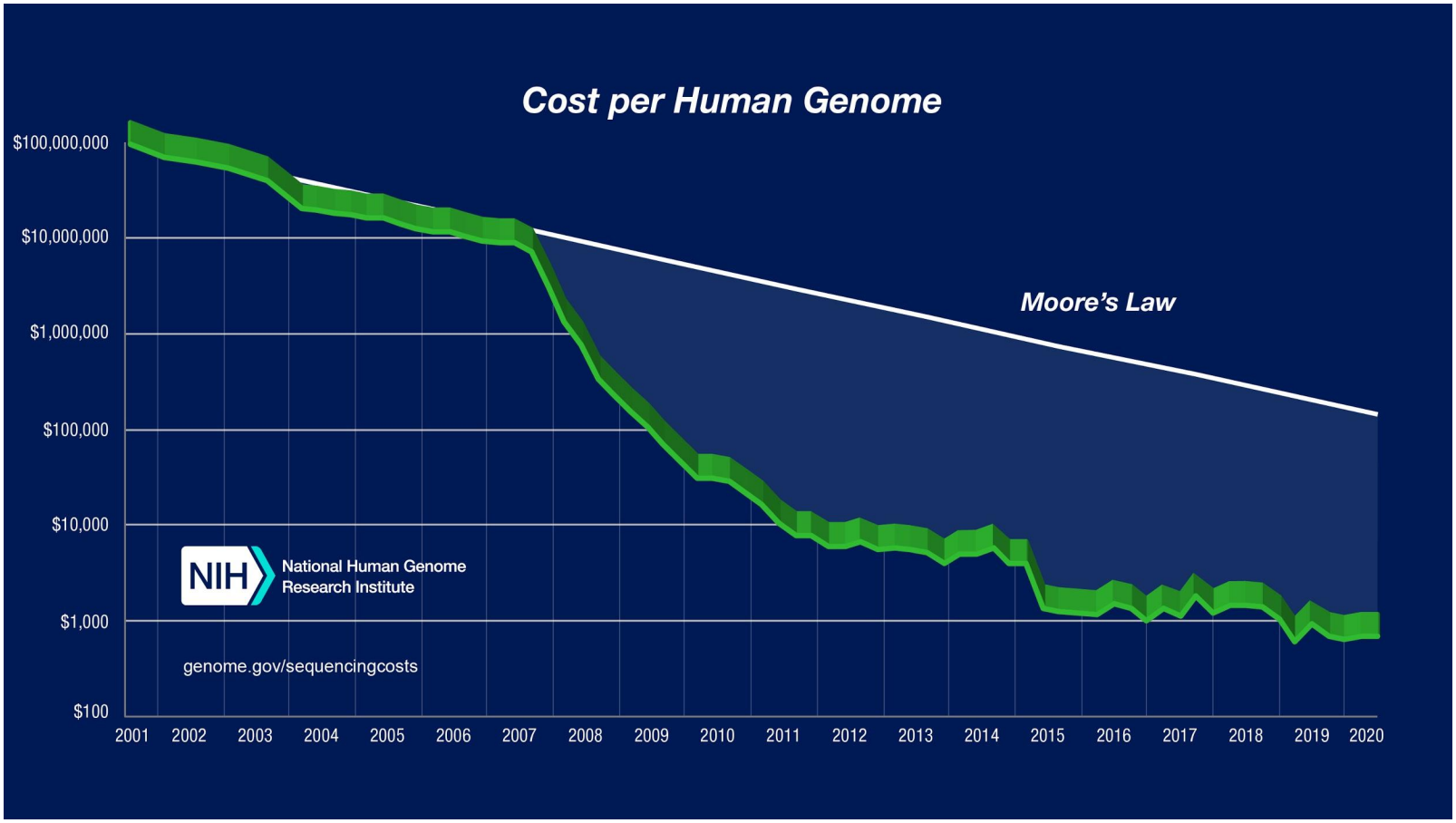
Methods for Indexing and Searching Large-Scale Genomic Data

Prashant Pandey
ppandey@berkeley.edu
Berkeley Lab/UC Berkeley

The *cost* of sequencing is going *down*

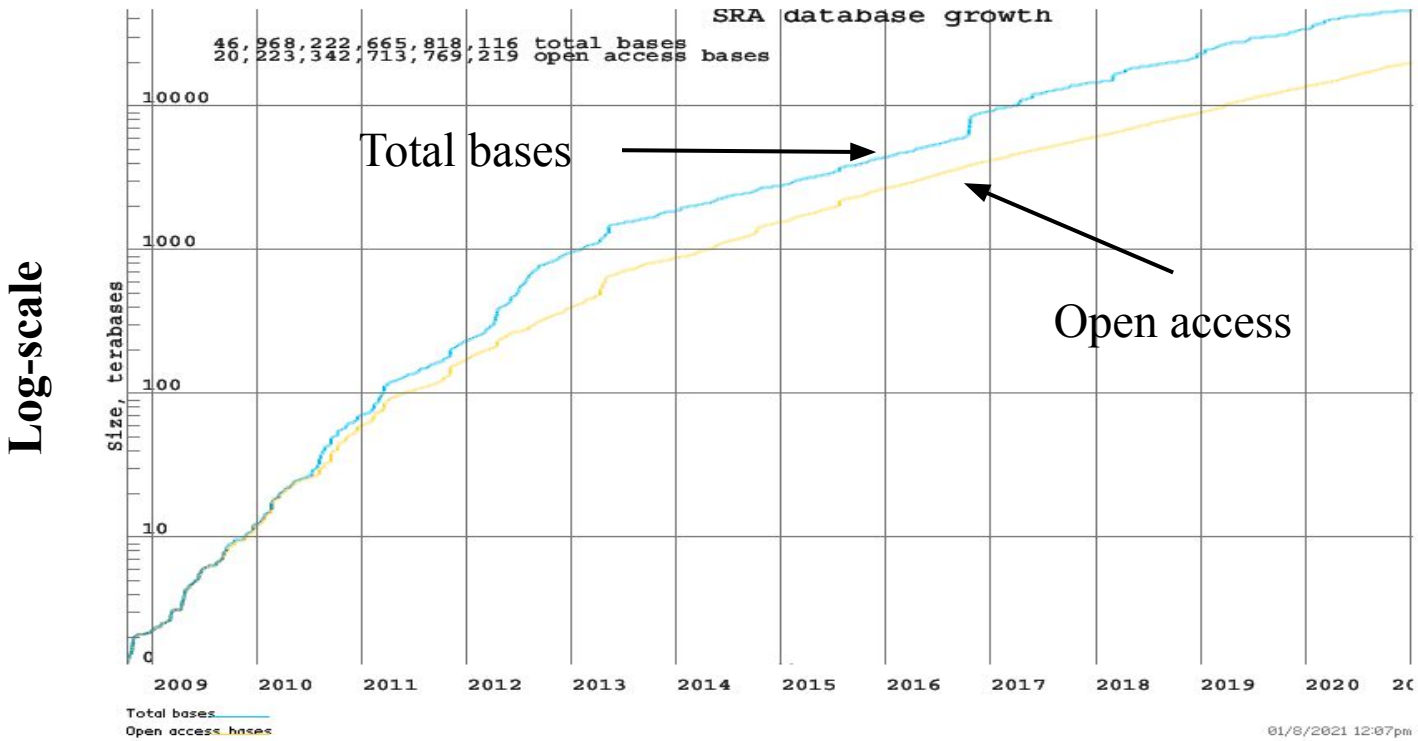
Log-scale



The cost of sequencing human genome is going down over years

Sequence Read Archive (SRA) database growth

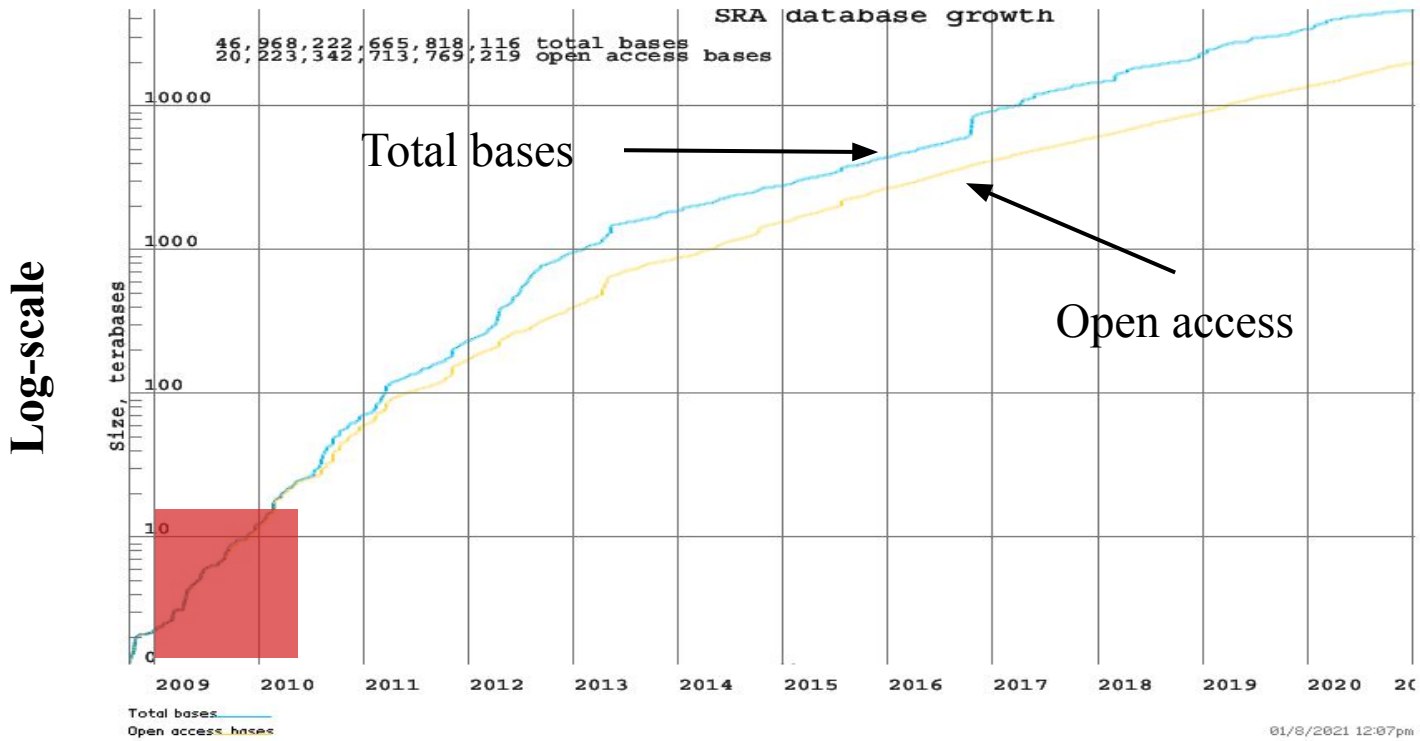
SRA contains a lot of ***diversity information***



Q: What if I find e.g., a new disease-related gene, and want to see if it appeared in other experiments?

New challenges due to data growth

SRA contains a lot of ***diversity information***



Only a small portion of SRA is searchable!

This renders what is otherwise an immensely valuable public resource ***largely inert***

The computational challenge

OPINION Open Access

The real cost of sequencing: scaling computation to keep pace with data generation

Paul Muir[1,2,3], Shantao Li[4], Shaoke Lou[4,5], Daifeng Wang[4,5], Daniel J Spakowicz[4,5], Leonidas Salichos[4,5], Jing Zhang[4,5], George M. Weinstock[6], Farren Isaacs[1,2], Joel Rozowsky[4,5] and Mark Gerstein[4,5,7*]

"This new regime, in which costs scale with the amount of computational processing time, places a premium on driving down the average cost by developing efficient algorithms for data processing."

The computational challenge

OPINION Open Access

The real cost of sequencing: scaling computation to keep pace with data generation

Paul Muir[1,2,3], Shantao Li[4], Shaoke Lou[4,5], Daifeng Wang[4,5], Daniel J Spakowicz[4,5], Leonidas Salichos[4,5], Jing Zhang[4,5], George M. Weinstock[6], Farren Isaacs[1,2], Joel Rozowsky[4,5] and Mark Gerstein[4,5,7*]

"This new regime, in which costs scale with the amount of computational processing time, places a premium on driving down the average cost by developing efficient algorithms for data processing."

Also, it's not just "new" data that is the problem

In addition to new data, re-analysis of existing experiments often desired: In light of new annotations, discoveries, and methodological advancements.

Three approaches to handle massive data

Shrink it

Goal: make data smaller to fit in RAM

Techniques:

- LSH e.g., MinHash
- Filters, e.g., Bloom filter
- Succinct data structures

Three approaches to handle massive data

Shrink it

Goal: make data smaller to fit in RAM

Techniques:

- LSH e.g., MinHash
- Filters, e.g., Bloom filter
- Succinct data structures

Organize it

Goal: organize data in a disk-friendly way

Techniques:

- B-tree
- B^{ε}-tree
- LSM-tree

Three approaches to handle massive data

Shrink it

Goal: make data smaller to fit in RAM

Techniques:

- LSH e.g., MinHash
- Filters, e.g., Bloom filter
- Succinct data structures

Organize it

Goal: organize data in a disk-friendly way

Techniques:

- B-tree
- B^{ε}-tree
- LSM-tree

Distribute it

Goal: partition and distribute data on multiple nodes

Techniques:

- Distributed hash table
- Distributed key-value store

Our solutions to handle massive data

Shrink it

(Counting) Quotient Filter
SIGMOD '17, arXiv '17

Order Min Hash
ISMB '19

Organize it

Buffered Count-Min Sketch
ESA '18

Affine & PDAM model
SPAA '19

Distribute it

Our solutions to handle massive data

Shrink it

(Counting) Quotient Filter
SIGMOD '17, arXiv '17

Order Min Hash
ISMB '19

Squeakr, deBGR, Mantis, Rainbowfish, MST-Mantis
ISMB '17, WABI '17, BIOINFORMATICS '17, RECOMB '18, Cell Systems '18, RECOMB '19, JCB '20

Organize it

Buffered Count-Min Sketch
ESA '18

Affine & PDAM model
SPAA '19

LSM-Mantis, VaraintStore
bioRxiv '20, bioRxiv '21

Distribute it

Distributed GPU-based *k*-mer counting
IPDPS '21

Our solutions to handle massive data

Shrink it

(Counting) Quotient Filter
SIGMOD '17, arXiv '17

Order Min Hash
ISMB '19

Squeakr, deBGR, Mantis, Rainbowfish, MST-Mantis
ISMB '17, WABI '17, BIOINFORMATICS '17, RECOMB '18, Cell Systems '18, RECOMB '19, JCB '20

Organize it

Buffered Count-Min Sketch
ESA '18

Affine & PDAM model
SPAA '19

LSM-Mantis, VaraintStore
bioRxiv '20, bioRxiv '21

BεtrFS file system
FAST '15, TOS 15, FAST '16, TOS 16

Distribute it

Distributed GPU-based *k*-mer counting
IPDPS '21

Our solutions to handle massive data

Shrink it

(Counting) Quotient Filter
SIGMOD '17, arXiv '17

Order Min Hash
ISMB '19

Squeakr, deBGR, Mantis, Rainbowfish, MST-Mantis
ISMB '17, WABI '17, BIOINFORMATICS '17, RECOMB '18, Cell Systems '18, RECOMB '19, JCB '20

Organize it

Buffered Count-Min Sketch
ESA '18

Affine & PDAM model
SPAA '19

LSM-Mantis, VaraintStore
bioRxiv '20, bioRxiv '21

BεtrFS file system
FAST '15, TOS 15, FAST '16, TOS 16

LERTs (Event reporting)
arXiv '19, SIGMOD '20

Distribute it

Distributed GPU-based *k*-mer counting
IPDPS '21

In this talk: Order Min Hash (OMH)

Shrink it

(Counting) Quotient Filter
SIGMOD '17, arXiv '17

Order Min Hash
ISMB '19

Squeakr, deBGR, Mantis, Rainbowfish, MST-Mantis
ISMB '17, WABI '17, BIOINFORMATICS '17, RECOMB '18, Cell Systems '18, RECOMB '19, JCB '20

Bioinformatics, 35, 2019, i127–i135
doi: 10.1093/bioinformatics/btz354
ISMB/ECCB 2019

OXFORD

Locality-sensitive hashing for the edit distance

Guillaume Marçais*, Dan DeBlasio, Prashant Pandey and Carl Kingsford*

Computational Biology Department, Carnegie Mellon University, Pittsburgh, PA 15213, USA

Affine & PDAM model
SPAA '19

LSM-Mantis, VaraintStore
bioRxiv '20, bioRxiv '21

BεtrFS file system
FAST '15, TOS 15, FAST '16, TOS 16

LERTs (Event reporting)
arXiv '19, SIGMOD '20

IPDPS '21

Sequence similarity problem

Sequence similarity is a measure of the similarity of two sequences.

Eg., Edit distance between two sequences is a measure of their similarity.

Low edit distance ⇔ High similarity
High edit distance ⇔ Low similarity

Measuring sequence similarity is the *core* problem in many algorithms in computational biology

- Metagenomic clustering/classification [Wood & Salzberg 2014, Wood et al. 2019]
- Genome assembly (overlap-layout-consensus)[Jaffe et al. 2003, Myers et al. 2000]
- Sequence alignment [Langmead & Salzberg 2012, Ondov et al. 2016, Li 2018, Marcais et al. 2018]

Sequence similarity problem

Sequence similarity is a measure of the similarity of two sequences.

Eg., Edit distance between two sequences is a measure of their similarity.

Low edit distance ⇔ High similarity

Computing quadratic-time edit distance between sequences at scale is computationally not feasible in practice!

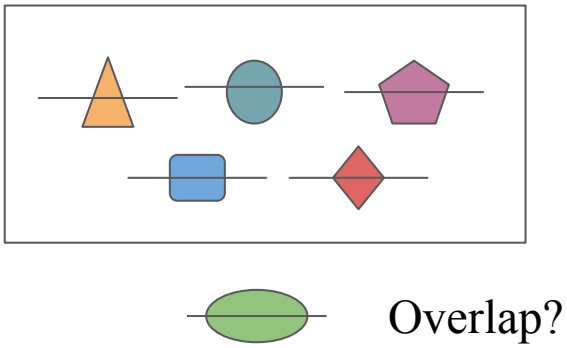
M y

algorithms in computational biology

- Metagenomic clustering/classification [Wood & Salzberg 2014, Wood et al. 2019]
- Genome assembly (overlap-layout-consensus)[Jaffe et al. 2003, Myers et al. 2000]
- Sequence alignment [Langmead & Salzberg 2012, Ondov et al. 2016, Li 2018, Marcais et al. 2018]

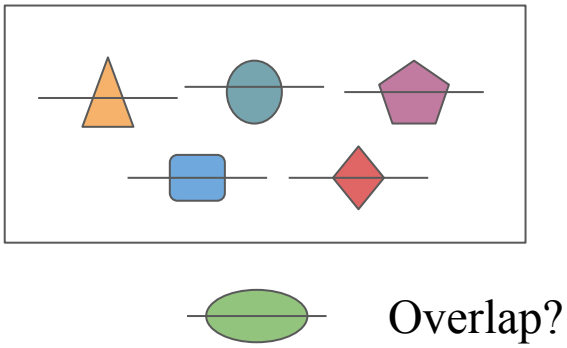
Overlap computation

- Compute overlaps between reads



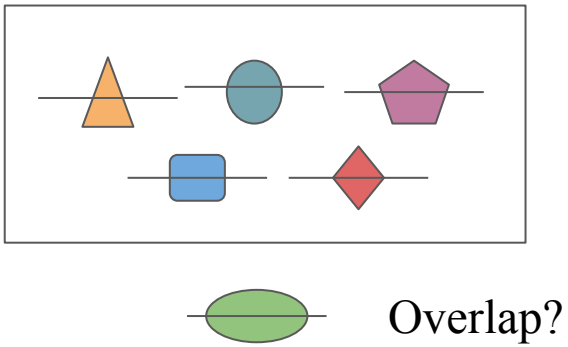
Overlap computation

- Compute overlaps between reads
- Instance of "Nearest Neighbor Problem" for edit distance

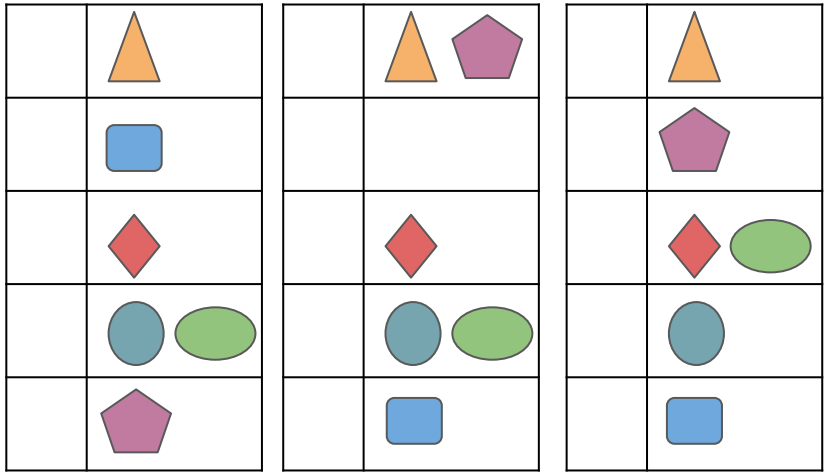


Overlap computation

- Compute overlaps between reads
- Instance of "Nearest Neighbor Problem" for edit distance
- Use multiple hash tables

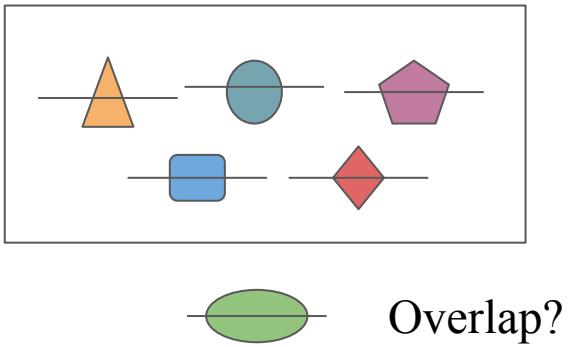


Hash tables

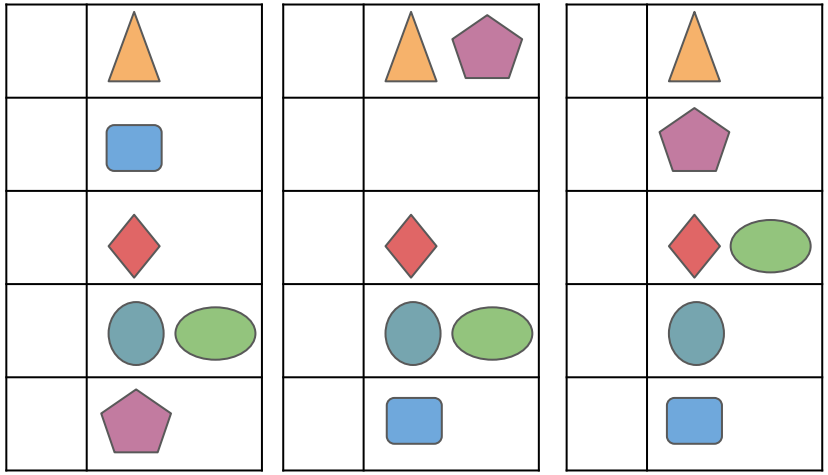


Overlap computation

- Compute overlaps between reads
- Instance of "Nearest Neighbor Problem" for edit distance
- Use multiple hash tables
- Need meaningful hash collisions



Hash tables



Locality Sensitive Hashing (LSH)

Pick $\boldsymbol{h}$ at random from $\mathcal{H}$:

$\Pr[\boldsymbol{h}(\bigcirc) = \boldsymbol{h}(\bigcirc)] > \Pr[\boldsymbol{h}(\bigcirc) = \boldsymbol{h}(\triangle)]$

Locality sensitive hash family

Family of hash functions where similar elements are more likely to have the same value than distant elements.

Locality Sensitive Hashing (LSH)

Pick $\boldsymbol{h}$ at random from $\mathcal{H}$:

Pr[$\boldsymbol{h}$(●) = $\boldsymbol{h}$(⬭)] > Pr[$\boldsymbol{h}$(●) = $\boldsymbol{h}$(△)]

Sketch (●) = {$\boldsymbol{h}_1$ (●), ……., $\boldsymbol{h}_m$(●)}

Locality sensitive hash family

Family of hash functions where similar elements are more likely to have the same value than distant elements.

Locality Sensitive Hashing (Ungaped LSH)

The family $\mathcal{H}$ is sensitive for distance D such that for all

$$x, y \in \mathrm{U}$$

$$\Pr[\boldsymbol{h}(x) = \boldsymbol{h}(y)] = 1 - \mathrm{D}(x, y)$$

Locality sensitive hash family

Family of hash functions where similar elements are more likely to have the same value than distant elements.

Locality Sensitive Hashing (Gaped LSH)

The family $\mathcal{H}$ is sensitive for distance D if there exists $d_1 < d_2, p_1 > p_2$ such that for all

$$x, y \in \mathrm{U}$$

$$\mathrm{D}(x, y) < d_1 \Rightarrow \Pr[\boldsymbol{h}(x) = \boldsymbol{h}(y)] \geq p_1$$

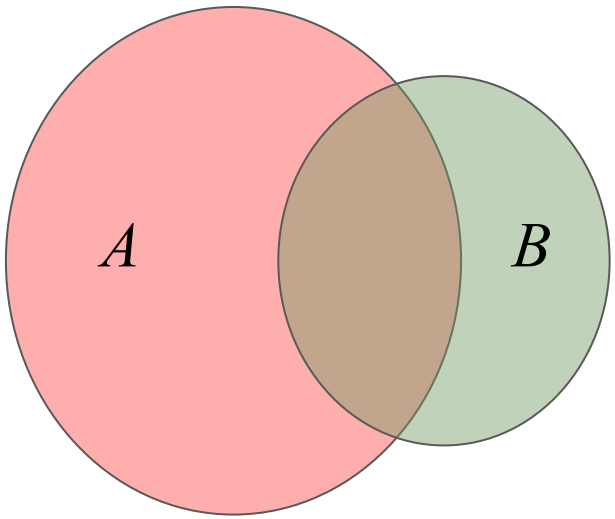
$$\mathrm{D}(x, y) \geq d_2 \Rightarrow \Pr[\boldsymbol{h}(x) = \boldsymbol{h}(y)] \leq p_2$$

Locality sensitive hash family

Family of hash functions where similar elements are more likely to have the same value than distant elements.

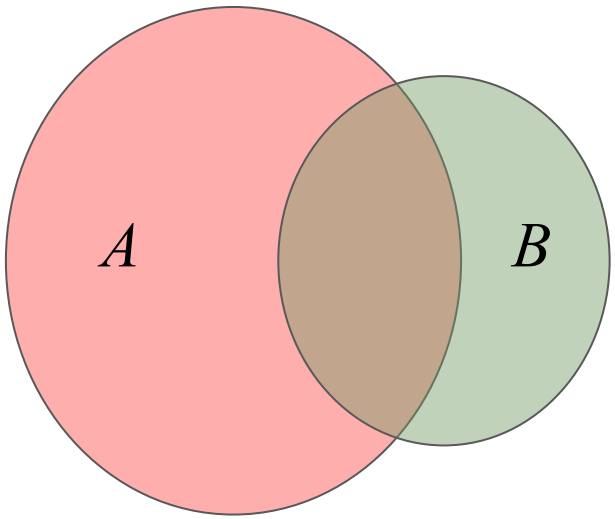
- Low distance ⇔ High collisions
- High distance ⇔ Low collisions

Jaccard distance → proxy for edit distance



$$J(A, B) = 1 - \frac{|A \cap B|}{|A \cup B|}$$

Jaccard distance → proxy for edit distance

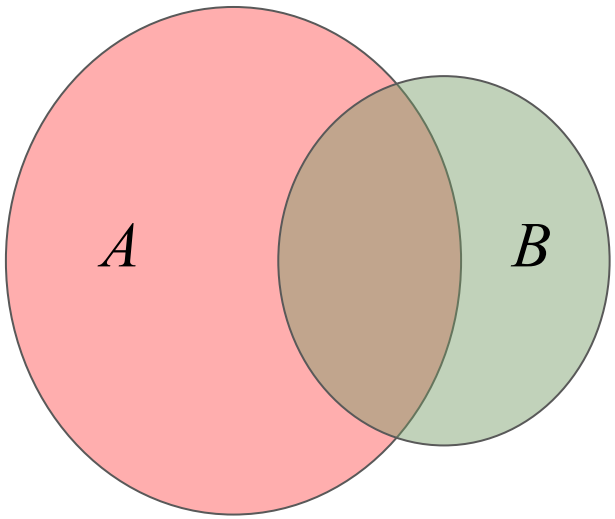


Jaccard distance between sequences x, y:
Jaccard distance of their k-mer sets

$$\mathrm{J}(x, y) = \mathrm{J}(\mathcal{K}(x), \mathcal{K}(y))$$

$$J(A, B) = 1 - \frac{|A \cap B|}{|A \cup B|}$$

Jaccard distance → proxy for edit distance



Jaccard distance between sequences x, y:
Jaccard distance of their k-mer sets

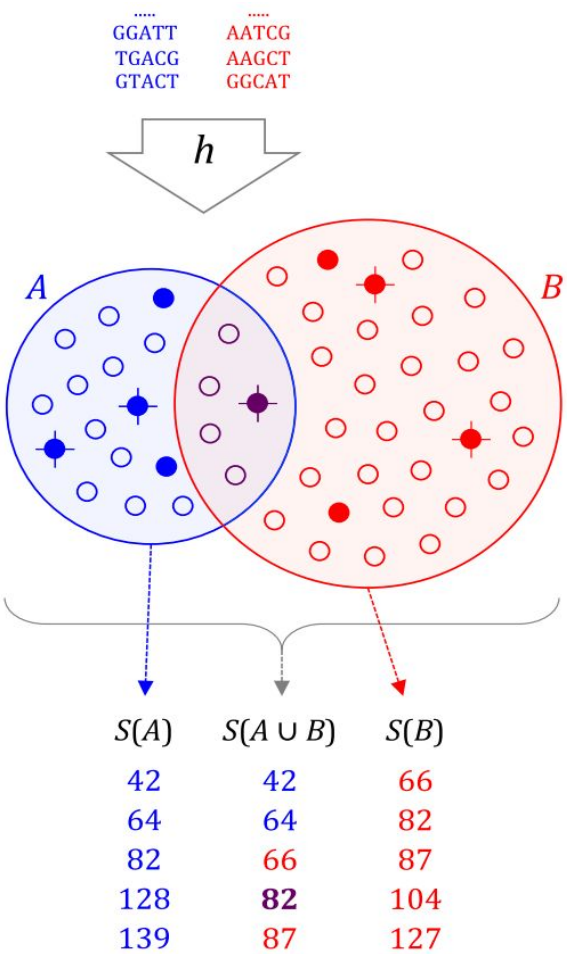
$$\mathrm{J}(x, y) = \mathrm{J}(\mathcal{K}(x), \mathcal{K}(y))$$

- Low $\mathrm{D}(x, y) \Rightarrow$ Low $\mathrm{J}(x, y)$
- High $\mathrm{D}(x, y) \not\Rightarrow$ High $\mathrm{J}(x, y)$

$$J(A, B) = 1 - \frac{|A \cap B|}{|A \cup B|}$$

Mash [Ondov et al. 2016]

Mash extends the **minHash** dimensionality-reduction technique to include a pairwise mutation distance and *P* value significance test, enabling the efficient ***clustering*** and ***search*** of massive sequence collections.



$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \approx \frac{|S(A \cup B) \cap S(A) \cap S(B)|}{|S(A \cup B)|}$$

Jaccard ignore *k*-mer repetition

$n - k$ k

x = AAAAAAAAAAAAAAAACCCC

y = AAAAACCCCCCCCCCCCCCC

k $n - k$

Jaccard ignore *k*-mer repetition

$n - k$ k

x = AAAAAAAAAAAAAAAACCCC → AAAAA, AAAAC, AAACC, AACCC, ACCCC, CCCCC

y = AAAAACCCCCCCCCCCCCCC → AAAAA, AAAAC, AAACC, AACCC, ACCCC, CCCCC

k $n - k$

Jaccard ignore *k*-mer repetition

$n - k$ k

x = AAAAAAAAAAAAAAAACCCC → AAAAA, AAAAC, AAACC, AACCC, ACCCC, CCCCC

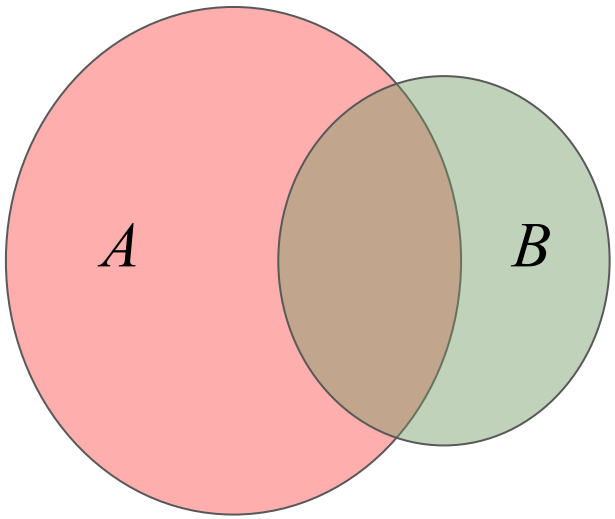
y = AAAAACCCCCCCCCCCCCCC → AAAAA, AAAAC, AAACC, AACCC, ACCCC, CCCCC

k $n - k$

Jaccard distance $J(x, y) = 0$ Edit distance $D(x, y) \geq 1 - 2k/n$

Identical *k*-mer content but high edit distance

Weighted Jaccard handles repetitions



Generalized Jaccard distance for multi-sets

$$J^W(A,B) = 1 - \frac{\sum_{x \in U} min(x_A(x), x_B(x))}{\sum_{x \in U} max(x_A(x), x_B(x))}$$

Weighted Jaccard handles repetitions

x = AAAAAAAAAAAAAAAACCCC (n - k, k) → (AAAAA,11), (AAAAC,1), (AAACC,1), (AACCC,1), (ACCCC,1), (CCCCC,1)

y = AAAAACCCCCCCCCCCCCCC (k, n - k) → (AAAAA,1), (AAAAC,1), (AAACC,1), (AACCC,1), (ACCCC,1), (CCCCC,11)

Jaccard distance $J^W(x, y) = 1 - (k+2)/n$ Edit distance $D(x, y) \geq 1 - 2k/n$

Weighted Jaccard is closer to edit distance than Jaccard

Jaccard and weighted Jaccard ignore relative order

x = CCCCACCAACACAAAACCC

y = AAAACACAACCCCACCAAA

x, y: de Bruijn sequences, contain all 16 possible 4-mers once

Jaccard and weighted Jaccard ignore relative order

x = CCCCACCAACACAAAACCC → AAAA, AAAC, AACA, AACC, ACAA, ACAC, ACCA, ACCC, CAAA, CAAC, CACA, CACC, CCAA, CCAC, CCCA, CCCC

y = AAAACACAACCCCACCAAA → AAAA, AAAC, AACA, AACC, ACAA, ACAC, ACCA, ACCC, CAAA, CAAC, CACA, CACC, CCAA, CCAC, CCCA, CCCC

x, y: de Bruijn sequences, contain all 16 possible 4-mers once

$$J(x, y) = J^{w}(x, y) = 0 \qquad D(x, y) = 0.63$$

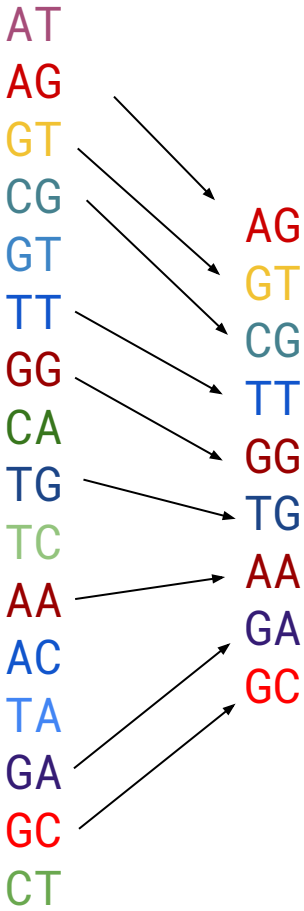
OMH: Order Min Hash

- minHash is an LSH for Jaccard
- OMH is a refinement of minHash
- OMH is sensitive to
  - repeated *k*-mers
  - relative order of *k*-mers

minHash and OMH sketch

x = AGTTGAGCGGAAGGTG k = 2

Order: permutation of Σ^k



minHash and OMH sketch

x = AGTTGAGCGGAAGGTG k = 2 m = 6

Order: permutation of Σ^k

1	2	3	4	5	6
AG	GG	CG	AA	TG	TT
GT	GA	GA	AG	TT	GG
CG	CG	TG	TT	GG	AG
TT	AG	AG	GT	CG	GC
GG	GC	GC	TG	AA	GA
TG	GT	GG	GC	GT	AA
AA	AA	TT	GA	AG	CG
GA	TT	AA	CG	GC	TG
GC	TG	GT	GG	GA	GT

minHash and OMH sketch

x = AGTTGAGCGGAAGGTG k = 2 m = 6

Order: permutation of Σ^k

1	2	3	4	5	6
AG	GG	CG	AA	TG	TT
GT	GA	GA	AG	TT	GG
CG	CG	TG	TT	GG	AG
TT	AG	AG	GT	CG	GC
GG	GC	GC	TG	AA	GA
TG	GT	GG	GC	GT	AA
AA	AA	TT	GA	AG	CG
GA	TT	AA	CG	GC	TG
GC	TG	GT	GG	GA	GT

minHash and OMH sketch

x = AGTTGAGCGGAAGGTG k = 2 m = 6

Order: permutation of $\Sigma^k \times \{1,\ldots, n\}$

1	2	3	4	5	6
AG	GG	CG	AA	TG	TT
GT	GA	GA	AG	TT	GG
CG	CG	TG	TT	GG	AG
TT	AG	AG	GT	CG	GC
GG	GC	GC	TG	AA	GA
TG	GT	GG	GC	GT	AA
AA	AA	TT	GA	AG	CG
GA	TT	AA	CG	GC	TG
GC	TG	GT	GG	GA	GT

1
GA,4
TG,3
AG,5
GT,1
GT,13
AA,10
AG,11
TT,2
AG,0
CG,7
GG,12
GC,6
TG,14
GG,8
GA,9

minHash and OMH sketch

x = AGTTGAGCGGAAGGTG k = 2 m = 6

Order: permutation of $\Sigma^k \times \{1,\ldots, n\}$

1	2	3	4	5	6
AG	GG	CG	AA	TG	TT
GT	GA	GA	AG	TT	GG
CG	CG	TG	TT	GG	AG
TT	AG	AG	GT	CG	GC
GG	GC	GC	TG	AA	GA
TG	GT	GG	GC	GT	AA
AA	AA	TT	GA	AG	CG
GA	TT	AA	CG	GC	TG
GC	TG	GT	GG	GA	GT

1	2	3	4	5	6
GA,4	CG,7	GT,13	AG,0	AA,10	GA,9
TG,3	TG,14	GA,4	TT,2	GT,13	GG,8
AG,5	AG,0	GA,9	AG,11	GA,9	GC,6
GT,1	GA,9	TG,3	AG,5	GT,1	TG,14
GT,13	AG,5	AG,5	AA,10	AG,5	GT,13
AA,10	AG,11	CG,7	GT,13	TT,2	TT,2
AG,11	GA,4	TT,2	CG,7	GA,4	AA,10
TT,2	GT,13	AA,10	GG,8	CG,7	AG,0
AG,0	TT,2	GG,12	GA,4	AG,0	CG,7
CG,7	TG,3	GG,8	GA,9	TG,3	GG,12
GG,12	GG,8	TG,14	TG,14	GG,8	AG,11
GC,6	AA,10	GT,1	TG,3	GG,12	TG,3
TG,14	GG,12	AG,11	GC,6	GC,6	GT,1
GG,8	GT,1	GC,6	GT,1	AG,11	GA,4
GA,9	GC,6	AG,0	GG,12	TG,14	AG,5

minHash and OMH sketch

x = AGTTGAGCGGAAGGTG k = 2 m = 6 l = 2

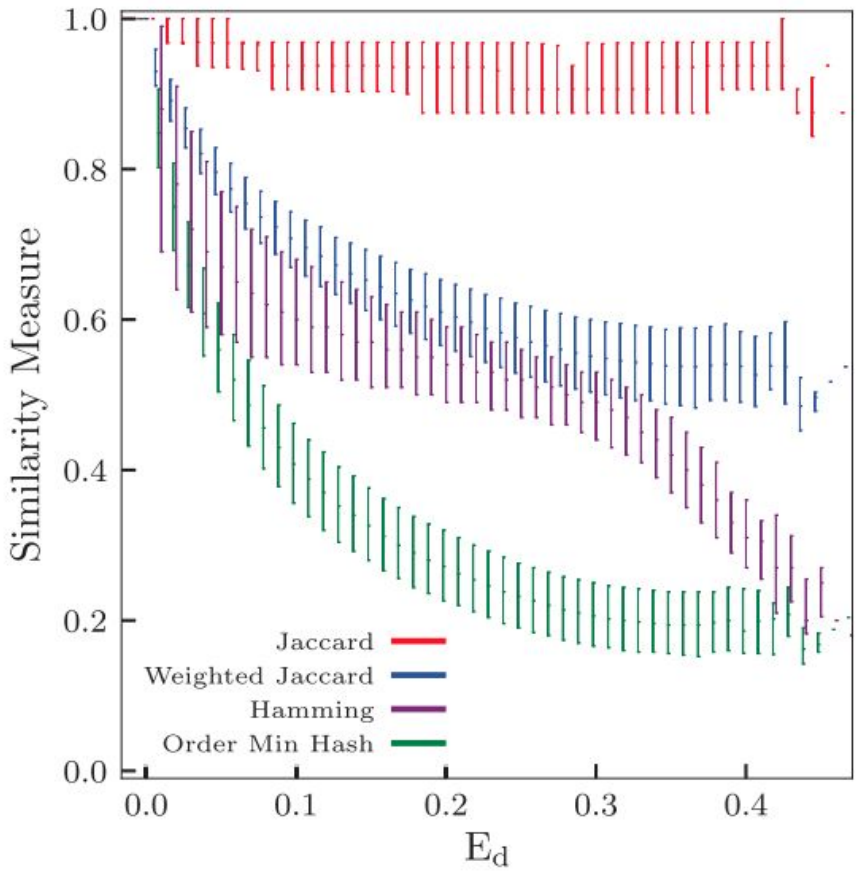
Order: permutation of $\Sigma^k \times \{1,\ldots, n\}$

1	2	3	4	5	6
AG	GG	CG	AA	TG	TT
GT	GA	GA	AG	TT	GG
CG	CG	TG	TT	GG	AG
TT	AG	AG	GT	CG	GC
GG	GC	GC	TG	AA	GA
TG	GT	GG	GC	GT	AA
AA	AA	TT	GA	AG	CG
GA	TT	AA	CG	GC	TG
GC	TG	GT	GG	GA	GT

1	2	3	4	5	6
GA,4	CG,7	GT,13	AG,0	AA,10	GA,9
TG,3	TG,14	GA,4	TT,2	GT,13	GG,8
AG,5	AG,0	GA,9	AG,11	GA,9	GC,6
GT,1	GA,9	TG,3	AG,5	GT,1	TG,14
GT,13	AG,5	AG,5	AA,10	AG,5	GT,13
AA,10	AG,11	CG,7	GT,13	TT,2	TT,2
AG,11	GA,4	TT,2	CG,7	GA,4	AA,10
TT,2	GT,13	AA,10	GG,8	CG,7	AG,0
AG,0	TT,2	GG,12	GA,4	AG,0	CG,7
CG,7	TG,3	GG,8	GA,9	TG,3	GG,12
GG,12	GG,8	TG,14	TG,14	GG,8	AG,11
GC,6	AA,10	GT,1	TG,3	GG,12	TG,3
TG,14	GG,12	AG,11	GC,6	GC,6	GT,1
GG,8	GT,1	GC,6	GT,1	AG,11	GA,4
GA,9	GC,6	AG,0	GG,12	TG,14	AG,5

GC	GA	AG	GG	AG	TG

OMH: conclusion



The Jaccard similarity stays high even for sequences with high edit distance

OMH: conclusion

- an improvement over minHash
- easy to compute
- locality sensitive for edit distance

Next: VariantStore

Shrink it

(Counting) Quotient Filter
SIGMOD '17, arXiv '17

Order Min Hash
ISMB '19

Squeakr, deBGR, Mantis, Rainbowfish, MST-Mantis
ISMB '17, WABI '17, BIOINFORMATICS '17, RECOMB '18, Cell Systems '18, RECOMB '19, JCB '20

Organize it

Buffered Count-Min Sketch
ESA '18

Affine & PDAM model
SPAA '19

LSM-Mantis, VaraintStore
bioRxiv '20, bioRxiv '21

B
FAST '

LER
arXi

Distribute it

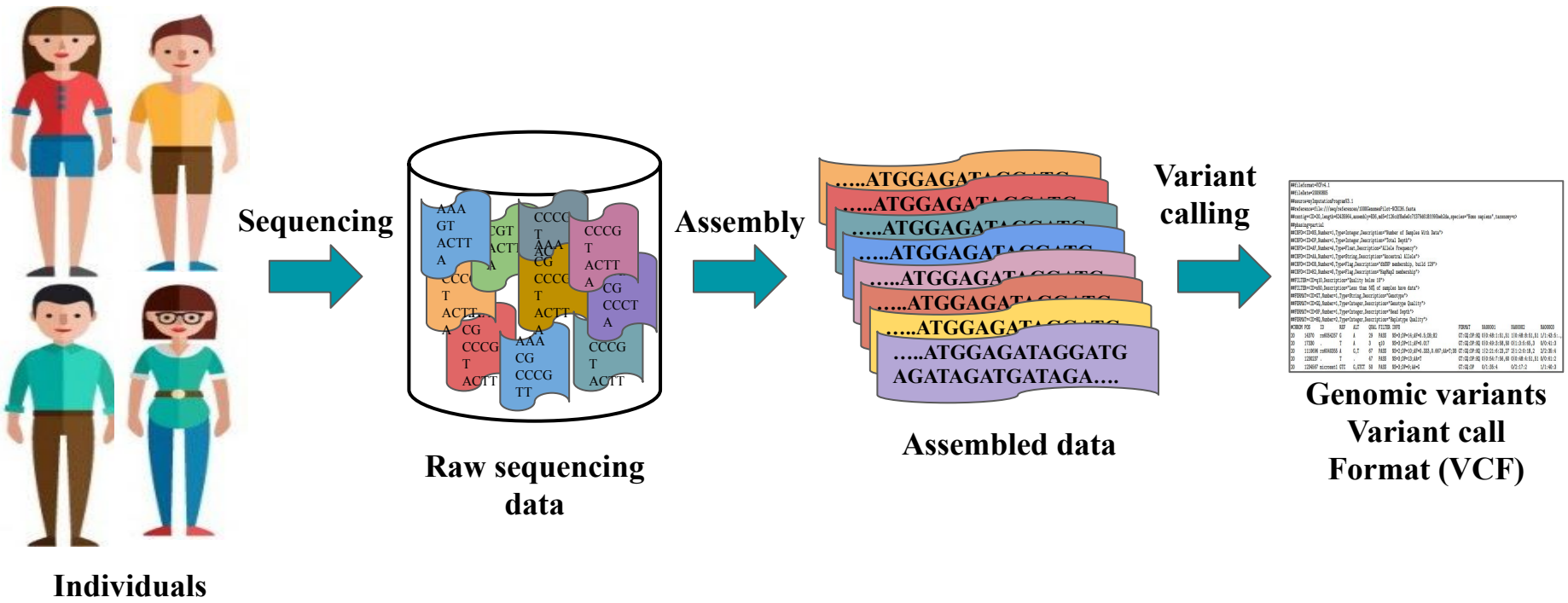
Distributed GPU-based *k*-mer counting
IPDPS '21

VariantStore: A Large-Scale Genomic Variant Search Index

Prashant Pandey[1], Yinjie Gao[1], and Carl Kingsford*[1]

[1]Computational Biology Department, School of Computer Science, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA

Country-scale sequencing efforts produce huge amount of variation data



- 1000 Genomes project [https://www.internationalgenome.org/]
- The Cancer Genome Atlas (TCGA) [https://portal.gdc.cancer.gov/]
- Genotype-Tissue Expression (GTEx) [https://gtexportal.org/home/]

Variation data analysis can improve downstream applications

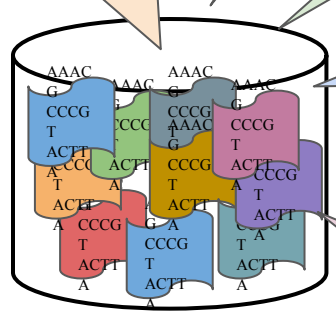
- Population-level disease analysis
- Genome-wide association studies
- Personalized medicine
- Cancer remission-rate prediction
- Colocalization analysis
- PCR primer design
- Genome assembly

Variation data analysis can improve downstream applications

- Population-level disease analysis
- Genome-wide association studies
- Personalized medicine
- Cancer remission-rate prediction
- Colocalization analysis
- PCR primer design
- Genome assembly

Individuals

Sequencing & assembly



Population Genomes

Count the number of variants in a gene

List all people, with > *N* variants in a gene

For person *P*, return the closest variant from position *X*

Return all positions with variants in a gene

List all people, with sequence *S* in a gene

Multiple sample sequences and variants

Sample 1	C	A	A	T	T	T	G	C	T	G	A	T	C	T			
Sample 2	C	A	T	G	C	T	G	A	T	C	T						
Sample 3	C	G	A	T	T	T	G	C	T	G	A	T	C	T			
Sample 4	C	G	A	T	T	T	A	C	G	G	C	T	G	A	T	C	T

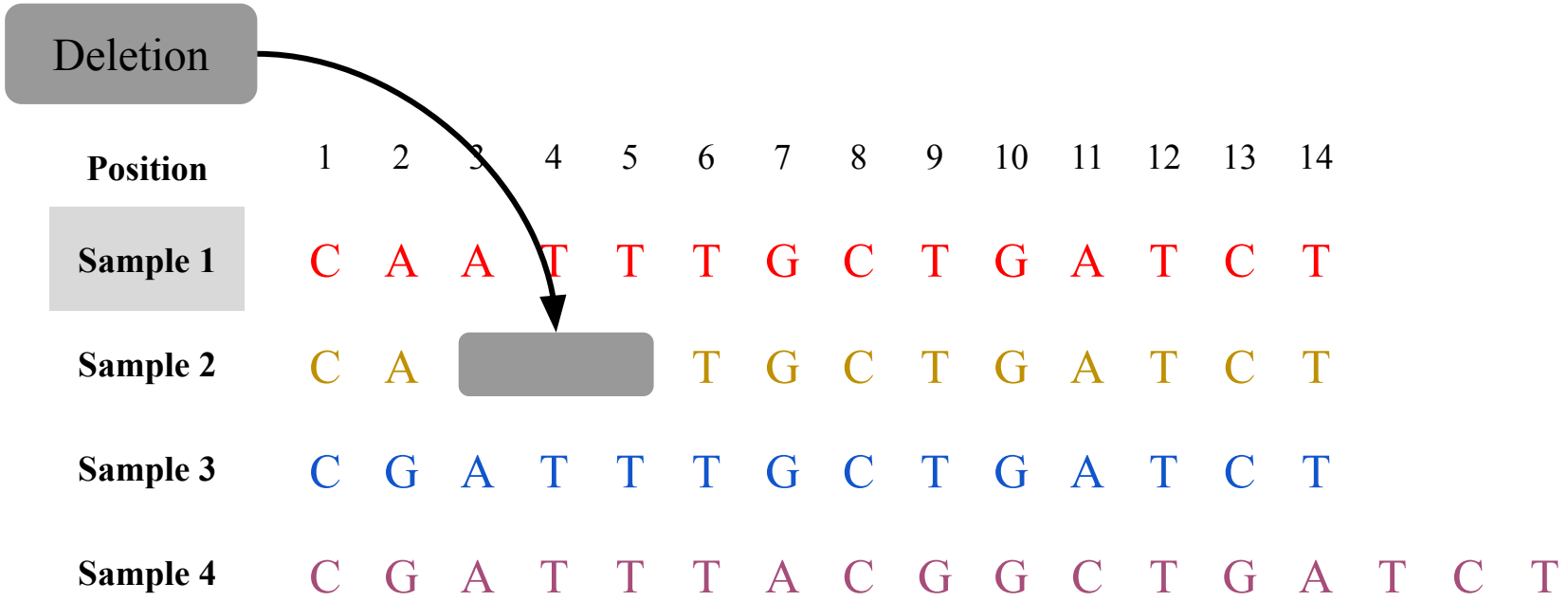
Multiple sample sequences and variants

Position	1	2	3	4	5	6	7	8	9	10	11	12	13	14			
Sample 1	C	A	A	T	T	T	G	C	T	G	A	T	C	T			
Sample 2	C	A	T	G	C	T	G	A	T	C	T						
Sample 3	C	G	A	T	T	T	G	C	T	G	A	T	C	T			
Sample 4	C	G	A	T	T	T	A	C	G	G	C	T	G	A	T	C	T

Treat sample 1 as the ***reference coordinate system*** and identify variants

A ***coordinate system*** uniquely identifies the position of a variant in a given genome

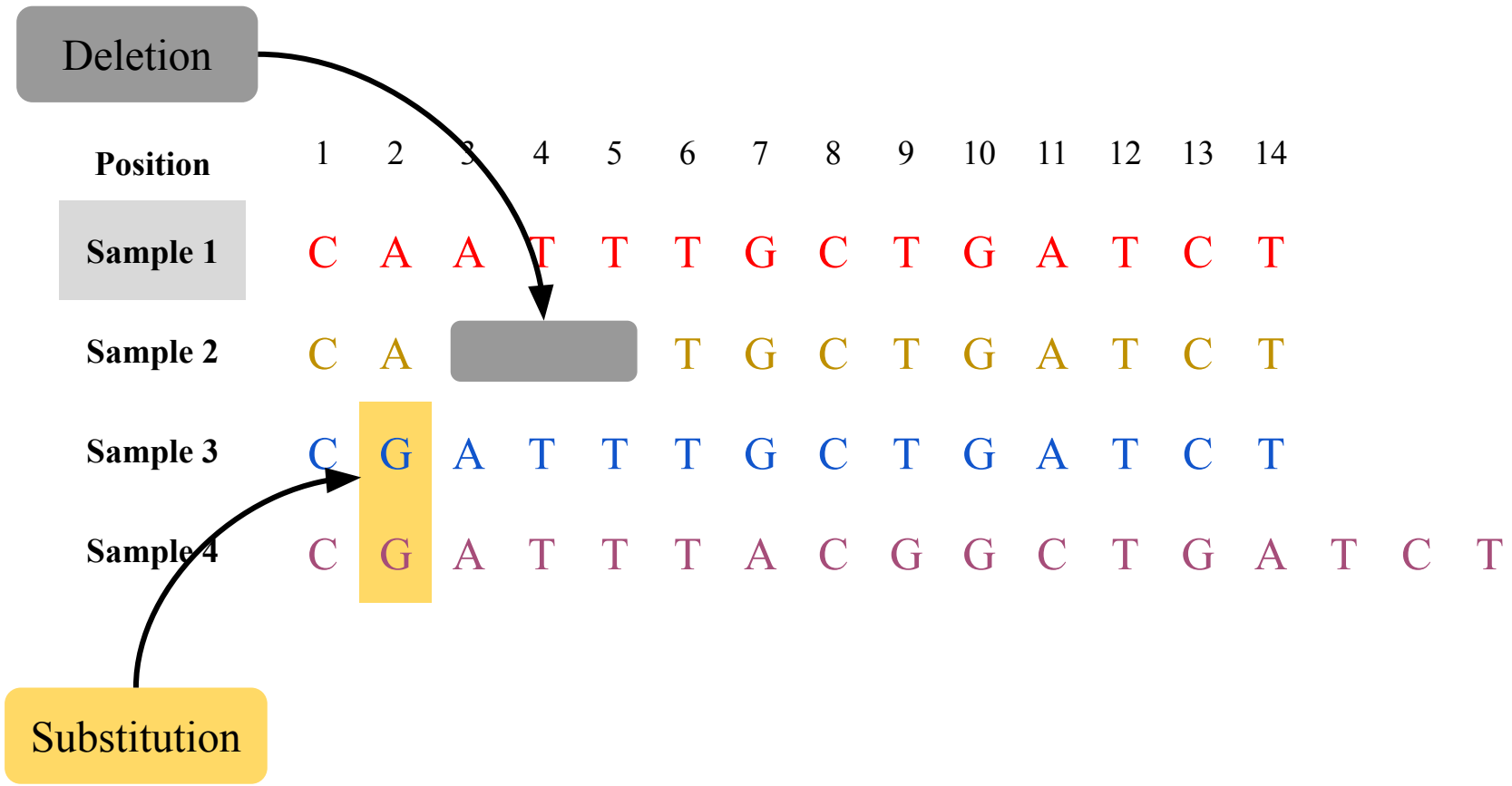
Multiple sample sequences and variants



Treat sample 1 as the ***reference coordinate system*** and identify variants

A ***coordinate system*** uniquely identifies the position of a variant in a given genome

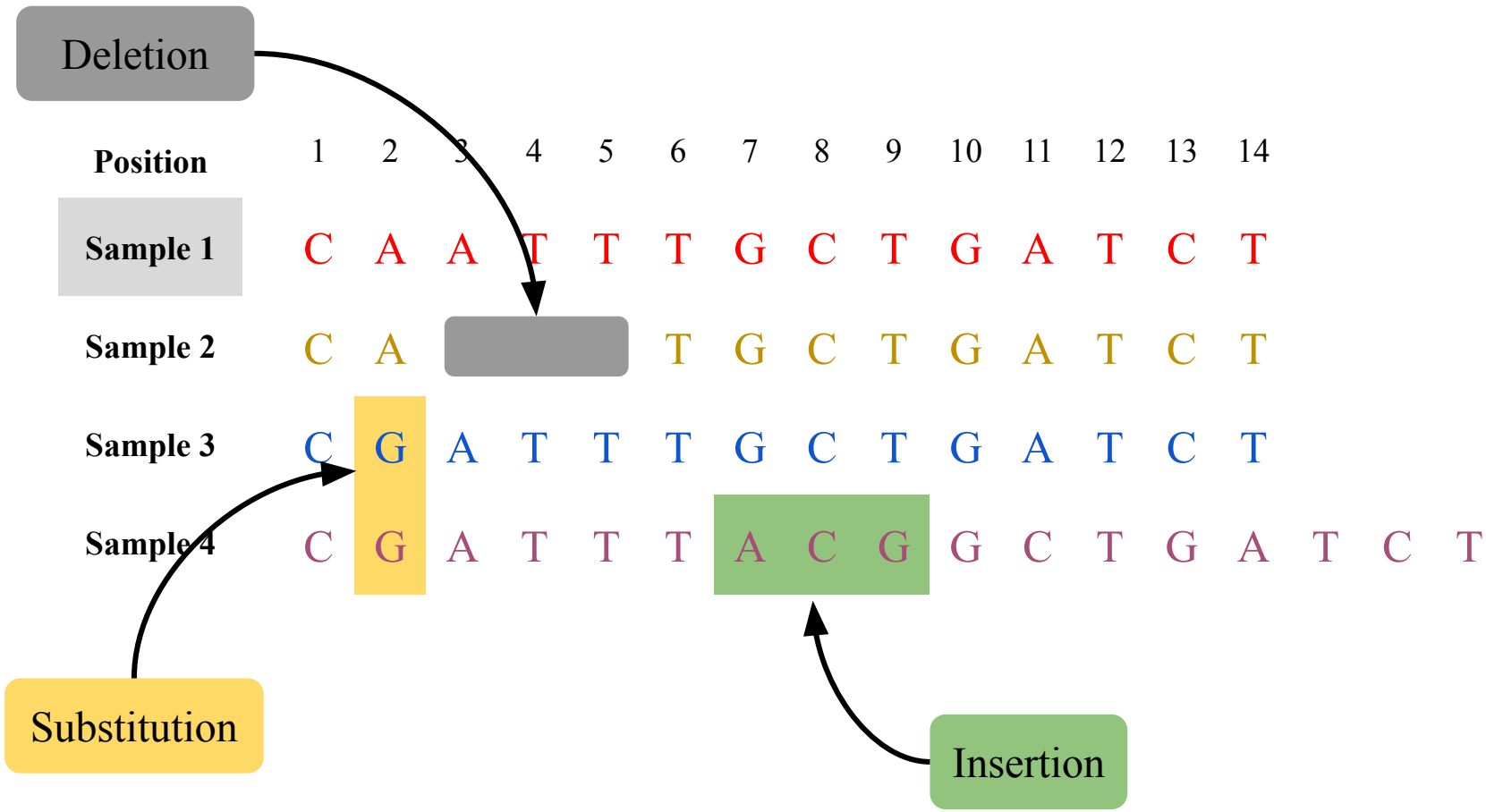
Multiple sample sequences and variants



Treat sample 1 as the ***reference coordinate system*** and identify variants

A ***coordinate system*** uniquely identifies the position of a variant in a given genome

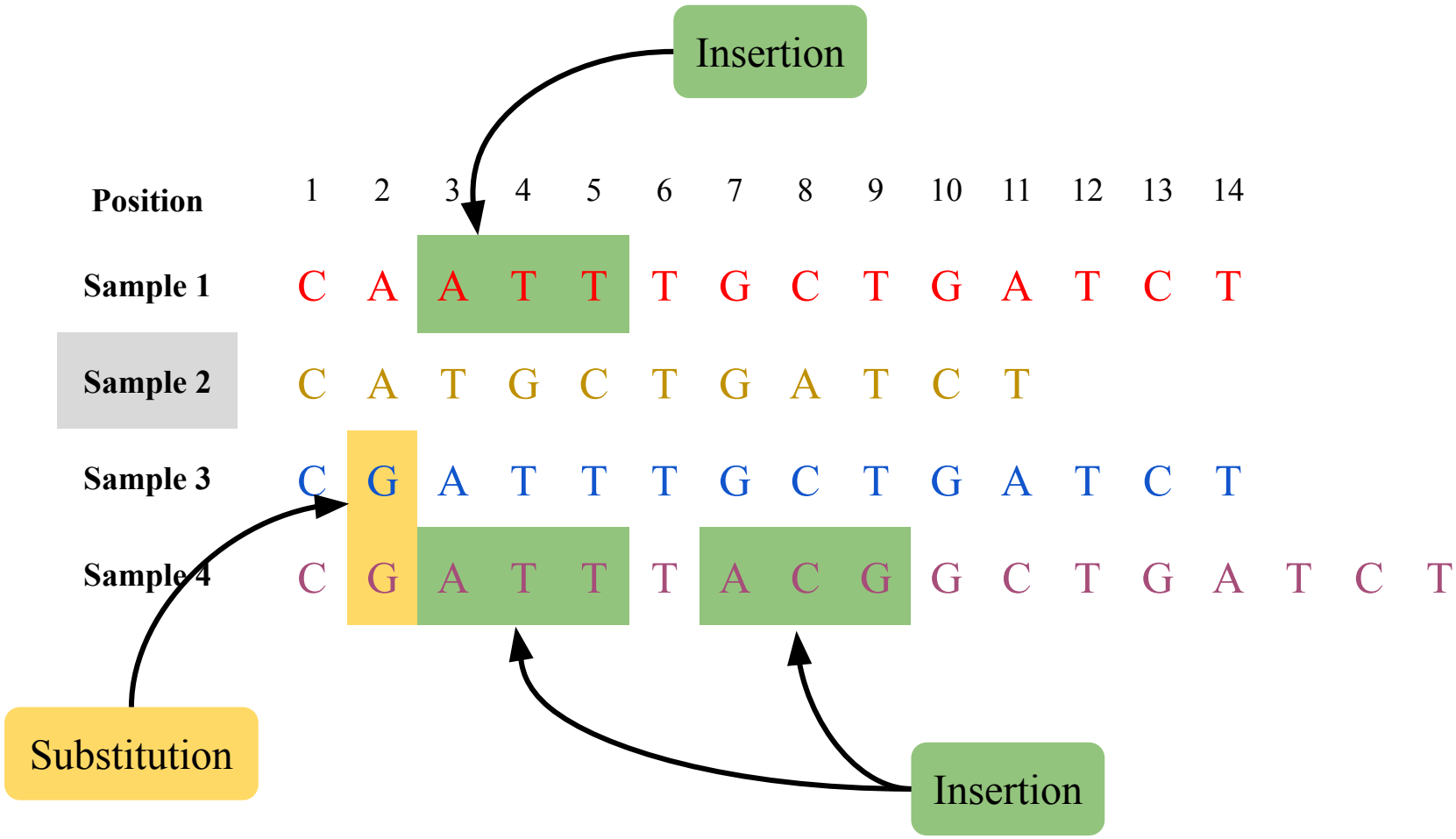
Multiple sample sequences and variants



Treat sample 1 as the ***reference coordinate system*** and identify variants

A ***coordinate system*** uniquely identifies the position of a variant in a given genome

Indels introduce multiple coordinate systems



Each sample can have a ***different*** coordinate system

Variant queries map positions to variants

Reference-only indexes map positions only in the reference coordinate system

$$f(p_i, p_j) \rightarrow (v_i \ldots v_n), \text{ where } p_i \leq p_j$$

Indexing in multiple coordinates is challenging

Reference-only indexes map positions only in the reference coordinate system

$$f(p_i, p_j) \rightarrow (v_i \ldots v_n), \text{ where } p_i \leq p_j$$

Pan-genome analysis involves queries based on sample coordinate systems

$$\text{Num Samples} \left\{ \begin{array}{l} f_1(p_i, p_j) \rightarrow (v_i \ldots v_n), \text{ where } p_i \leq p_j \\ \vdots \\ f_s(p_i, p_j) \rightarrow (v_i \ldots v_n), \text{ where } p_i \leq p_j \end{array} \right.$$

Maintaining thousands of mappings ***increases*** computational ***complexity*** and ***memory footprint***

Limits scalability to population-scale data

Existing solutions do not scale to thousands of samples

- Existing solutions are built to cater to specific applications
- For example, VG toolkit[1] and Seven Bridges[2] are built for read mapping applications
- They encode variants in a ***variation graph*** and perform graph traversals for read mapping
- They support sequence search but ***do not support other kinds of queries***
- The solutions are ***not designed to scale*** with increasing amounts of population-level variation data

[1] Variation graph toolkit improves read mapping by representing genetic variation in the reference. *Nature Biotechnology*, 36:875–879, 2018
[2] Fast and accurate genomic analyses using genome graphs. *Nature Genetics*, 51:354–362, 2019

Reference-only indexes do not support multiple coordinate queries

- GQT[1], BGT[2], and GTC[3] are ***reference-only indexes***
- They are optimized to support positional variant queries but ***do not store sequences for comparison***
- Traditional database-based solutions have proven ***prohibitively slow***

[1] Efficient genotype compression and analysis of large genetic-variation data sets. *Nature Methods*, 13(1):63, 2016
[2] BGT: efficient and flexible genotype query across many samples. *Bioinformatics*, 32(4): 590–592, 2015
[3] GTC: how to maintain huge genotype collections in a compressed form. *Bioinformatics*, 34(11):1834–1840, 2018

Reference-only indexes do not support multiple coordinate queries

- GQT[1], BGT[2], and GTC[3] are ***reference-only indexes***
- They are optimized to support positional variant queries but ***do not store sequences***
- **Existing systems don't support multiple coordinate systems. The ones that do, don't *scale* beyond a few thousand samples.**

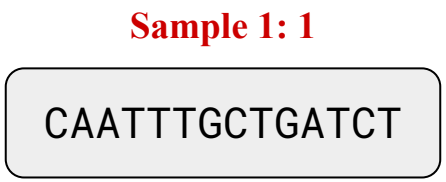
[1] Efficient genotype compression and analysis of large genetic-variation data sets. *Nature Methods*, 13(1):63, 2016
[2] BGT: efficient and flexible genotype query across many samples. *Bioinformatics*, 32(4): 590–592, 2015
[3] GTC: how to maintain huge genotype collections in a compressed form. *Bioinformatics*, 34(11):1834–1840, 2018

VariantStore: a system to efficiently index and query population-level variation data

- Supports querying variants in both reference and ***sample-specific coordinates***
  - Takes between **0.002 -- 3** seconds for different types of variant queries
- ***Scales*** to data containing ***thousands of samples*** and millions of variants
  - 1000 Genomes project, **2500** samples and **924M** variants, 3 Hrs
  - TCGA (BRCA) project, **8640** samples and **5M** variants, 4 Hrs
- ***Efficiently scales out-of-RAM*** to enable memory-efficient construction and query
  - Peak RAM is **10%** the size of the index

Variation graph construction

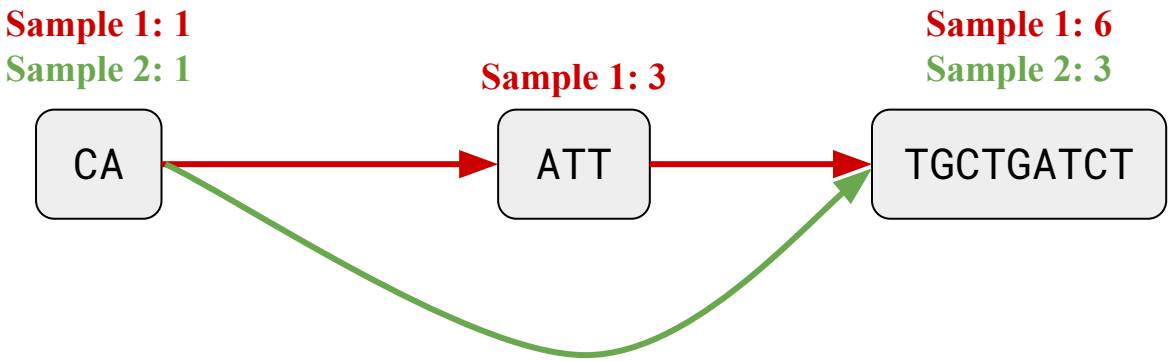
Sample 1: CAATTTGCTGATCT



Variation graph construction

Sample 1: CAATTTGCTGATCT

Sample 2: CATGCTGATCT

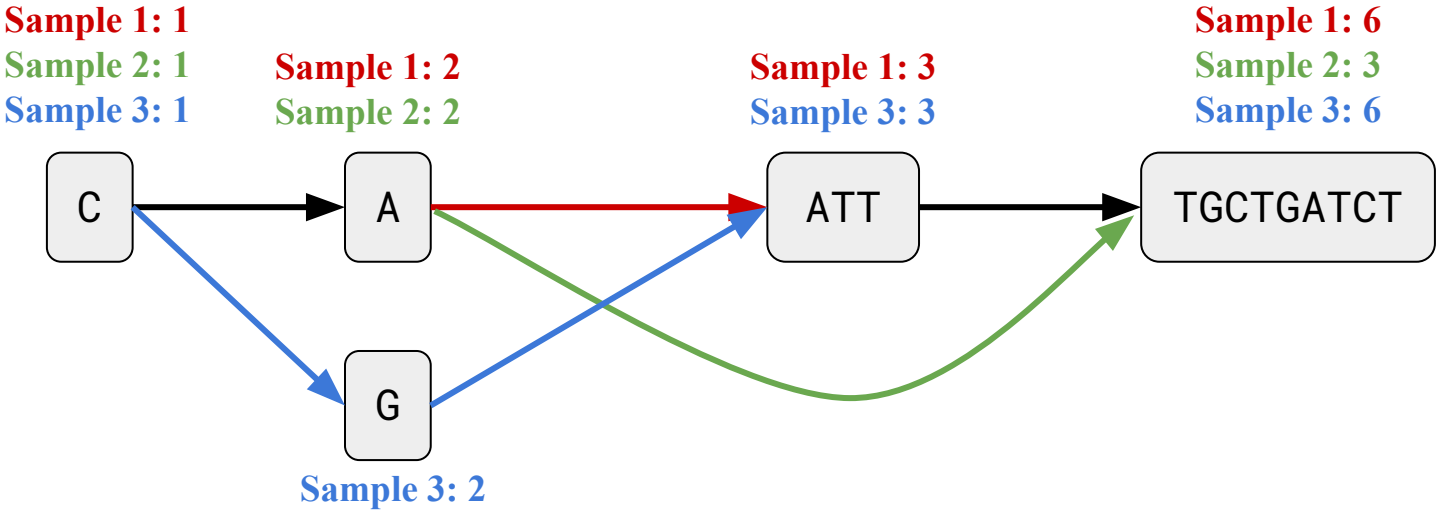


Variation graph construction

Sample 1: CAATTTGCTGATCT

Sample 2: CATGCTGATCT

Sample 3: CGATTTGCTGATCT



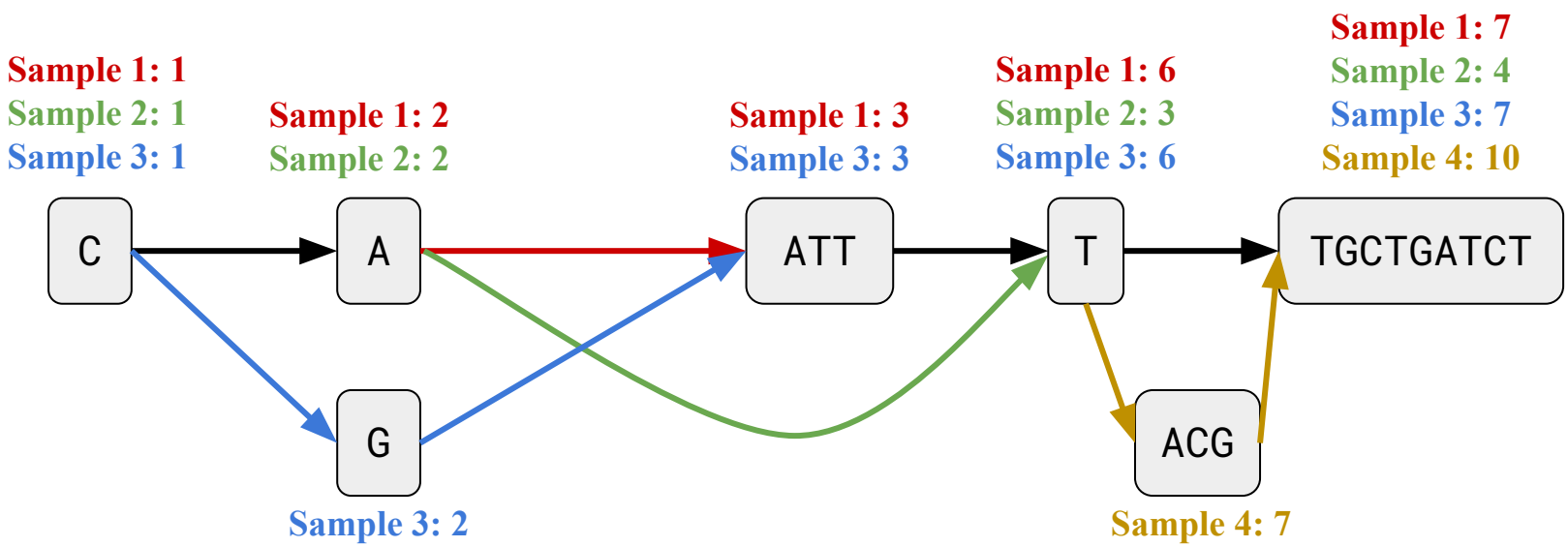
Variation graph construction

Sample 1: CAATTTGCTGATCT

Sample 2: CATGCTGATCT

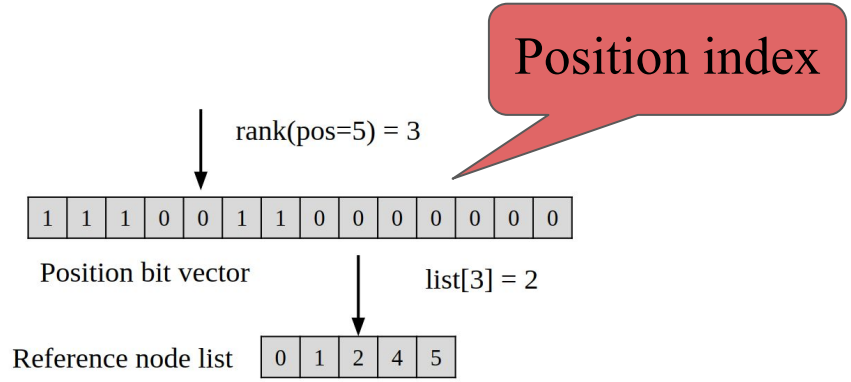
Sample 3: CGATTTGCTGATCT

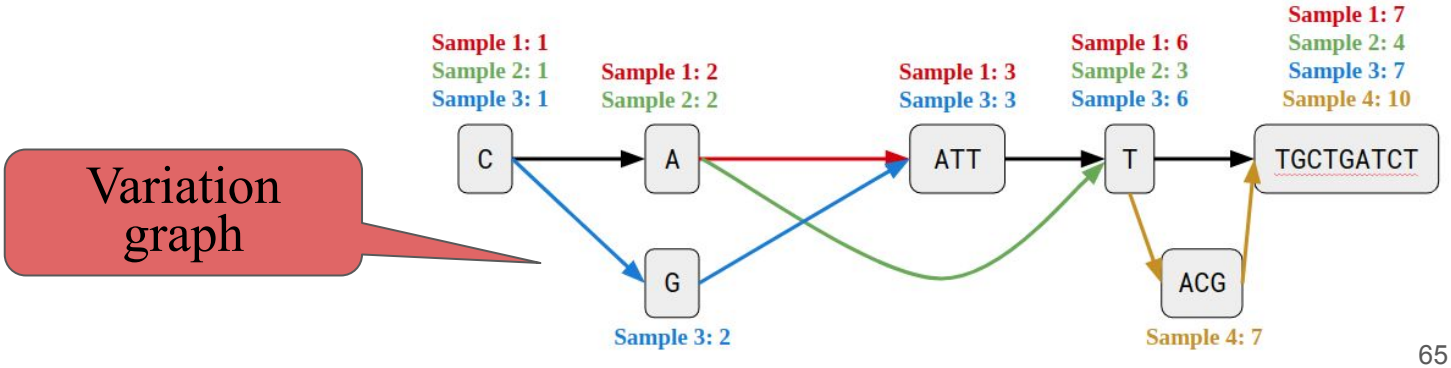
Sample 4: CGATTTACGGCTGATCT



An inverted index on the pan-genome graph

- Succinct index for reference coordinate system
- Local-graph exploration to map position from reference to sample coordinate



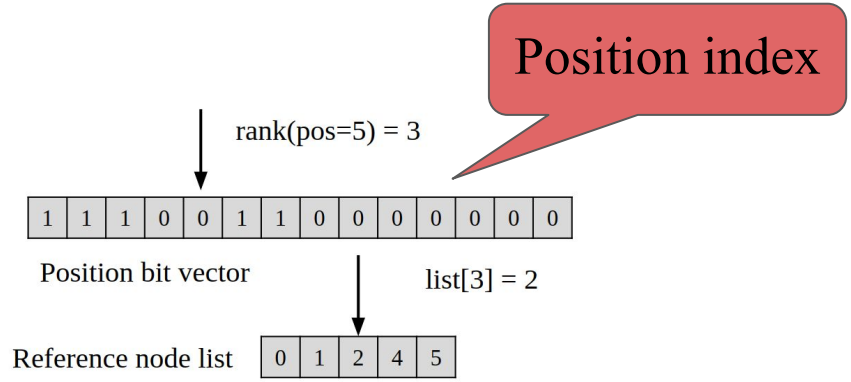


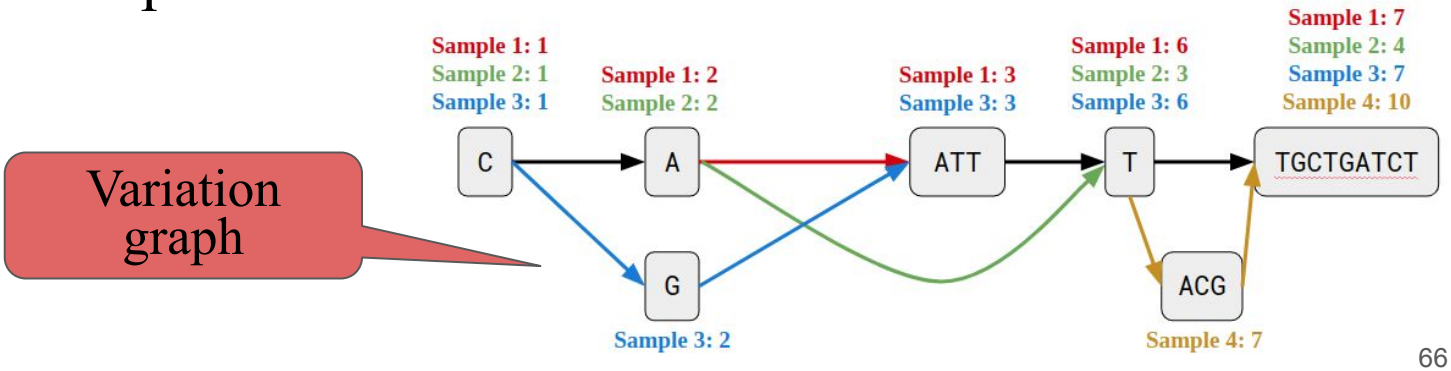
An inverted index on the pan-genome graph

- Partition the variation graph based on coordinate ranges
- Store partitions on disk

} Queries often require loading 1-2 partitions

- Succinct index for reference coordinate system
- Local-graph exploration to map position from reference to sample coordinate





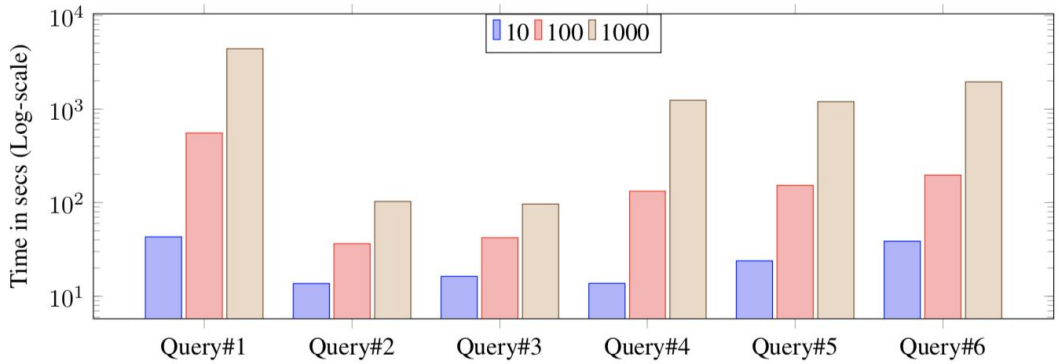
Results for constructing the index

System	Time	Disk space	Peak RAM	Peak RAM Agg.
Dataset		1000 Genomes		
VariantStore	3 Hrs 25 mins	41 GB	8.8 GB	153 GB
VG-toolkit	11 Hrs 10 mins	50 GB	37 GB	450 GB
Dataset		TCGA (OV)		
VariantStore	1 Hr 5 mins	3.4 GB	1.1 GB	17.45 GB
VG-toolkit		11 GB*		
Dataset		TCGA (LUAD)		
VariantStore	1 Hr 20 mins	3.5 GB	2.3 GB	36.05 GB
VG-toolkit		12 GB*		
Dataset		TCGA (BRCA)		
VariantStore	4 Hrs 36 mins	4.2 GB	3.2 GB	53.21 GB
VG-toolkit		14 GB*		

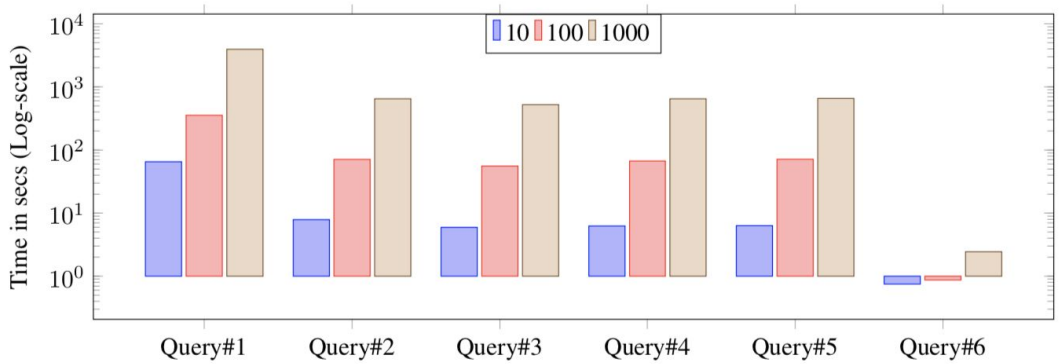
Table 1: Time, space, peak RAM, and peak RAM (aggregate) to construct variant index on the 1000 Genomes and TCGA (OV, LUAD, and BRCA) data using VariantStore and VG toolkit. *VG toolkit could not build GBWT index embedding all sample paths for TCGA data. Space reported is for the XG index that does not contain any path information. We constructed all 24 chromosomes (1 – 22 and X and Y) in parallel. The time and peak RAM reported is for the biggest chromosome (usually chromosome 1 or 2). The space reported is the total space on disk for all 24 chromosomes. The peak RAM (aggregate) is the aggregate peak RAM for all 24 processes.

VariantStore is ***3× faster***, takes ***25% less*** disk space, and ***3× less*** peak RAM than VG toolkit.

Results for variant queries



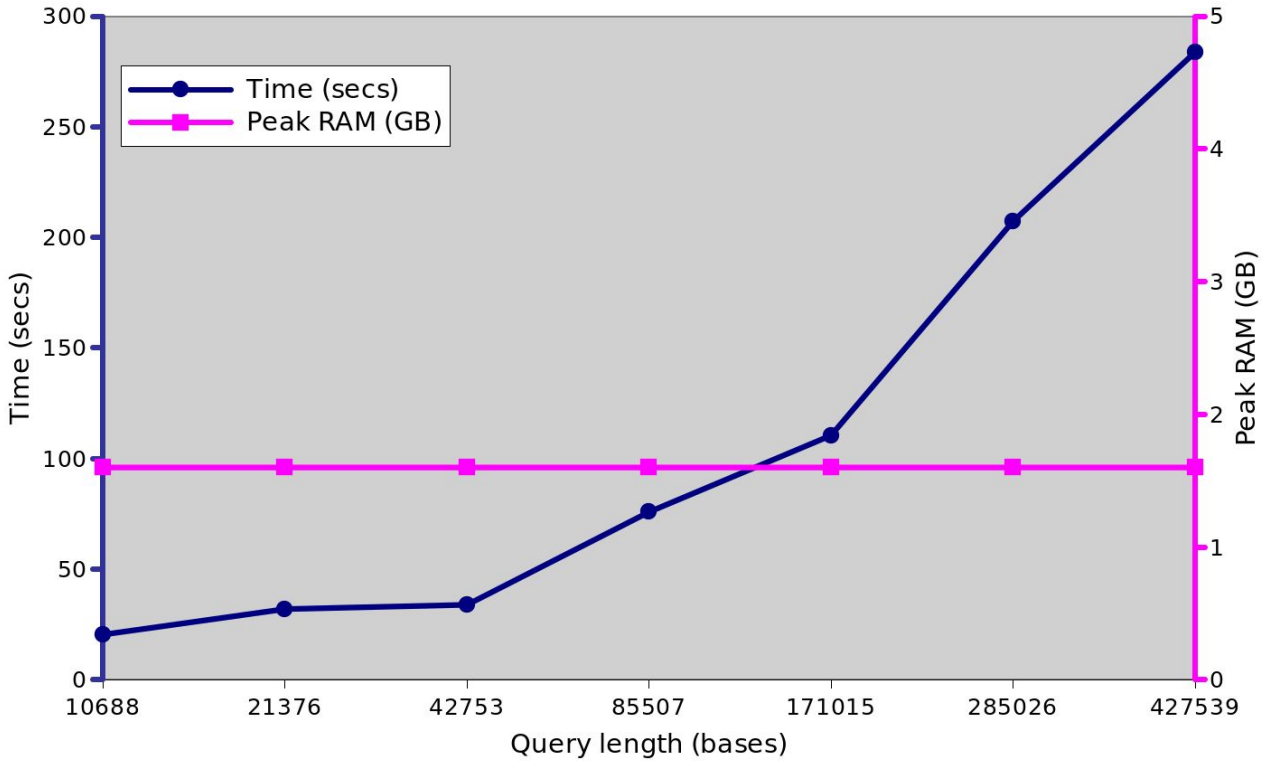
(a) Time for 10, 100, and 1000 queries on Chromosome 22 index in VariantStore for 1000 Genomes data.



(b) Time for 10, 100, and 1000 queries on Chromosome 22 index in VariantStore for TCGA LUAD data.

Aggregate time to execute queries ***increases sublinearly*** with the number of queries

Query analysis based on range size



Memory usage ***remains constant*** regardless of the query length

Conclusion

- The ability to efficiently query population-scale variation data promises to improve multiple downstream applications
- VariantStore enables variant queries across thousands of samples
- VariantStore efficiently scales out of RAM for easy usage in limited memory environments

https://github.com/Kingsford-Group/variantstore

Acknowledgements

Carnegie Mellon University

Carl Kingsford

Guillaume Marcais

Dan DeBlasio

Berkeley Lab/UC Berkeley

Kathy Yelick

Aydin Buluc

Stony Brook University

Rob Patro

Rob Johnson

Michael Bender

Fatemeh Almodaresi

Funding

NSF

Sandia National Laboratories

GORDON AND BETTY MOORE FOUNDATION

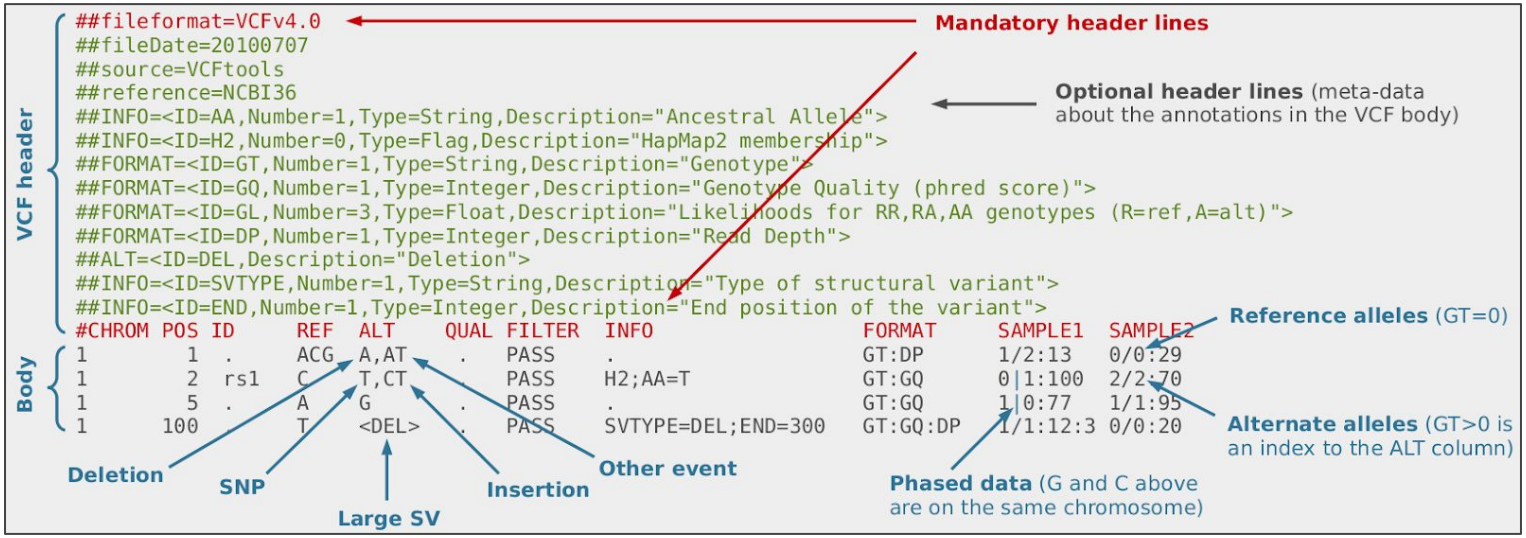
NIH National Institutes of Health

The Shurl and Kay Curci Foundation

U.S. DEPARTMENT OF ENERGY | Office of Science

https://prashantpandey.github.io

Variant call format (VCF) encodes variation information



Picture taken from: http://vcftools.sourceforge.net/VCF-poster.pdf

- The VCF format has been developed to encode variants from large scale sequencing
- These files contain variation as mutations based on a reference genome
  - SNPs and Indels

Applications performing pan-genome variant queries

- To determine the region of interest in PCR primer design [1], applications extract a fixed length sequence up and downstream from a given variant location in samples that share the variant and look for nearby variants affecting the primer.
- In colocalization analysis [2], applications query and compare variants or sequences in a genomic region across samples to determine significant overlaps between genomic regions in order to establish evolutionary or mechanistic relationships.

[1] A multiple-alignment based primer design algorithm for genetically highly variable DNA targets. *BMC bioinformatics*, 14(1):255, 2013.

[2] Colocalization analyses of genomic elements: approaches, recommendations and challenges. *Bioinformatics*, 35(9):1615–1624, 2019.