Methods for Exploring Large-Scale Genetic Datasets

Ryan M. Layer
ryan.layer@gmail.com
@ryanlayer
University of Utah
quinlanlab.org

Reference:

AGCGTGCATTCGCCGCGGTCCAACGTGTCAATAGACAACATTTCTCGTCGAAATACACAGTTCTGGGAGTGATGTGGCTAGGGAATCACTG

CGTGTCAATA

GTGCATTCGG

AGCGTGCATT

ATTCGGCGCG

GGCGCGGTCC

CGTCGAAATA

TCAATAGAC

CAACCTTTCT

CCGTCCAAGG

ATAGACAACC

ATTTCTCGTC

ACACAGTTTC

AGCG

CTGGGAGTGA

CGAAATACAC

GAGTGATGTG

GCTACGGAAT

TACGGAATCA

GTTTCTGGGA

ATGTGGCTAC

CTG

AATCACTG

Reference:

AGCGTGCATTCGCCGCGGTCCAACGTGTCAATAGACAACATTTCTCGTCGAAATACACAGTTTCTGGGAGTGATGTGGCTAGGGAATCACTG

AGCGTGCATT GGCGCGGTCC TCAATAGAC ATTTCTCGTC ACACAGTTTC GAGTGATGTG TACGGAATCA

GTGCATTCGG CCGTCCAAGG ATAGACAACC CGTCGAAATA GTTTCTGGGA ATGTGGCTAC AATCACTG

AGCG ATTCGGCGCG CGTGTCAATA CAACCTTTCT CGAAATACAC CTGGGAGTGA GCTACGGAAT CTG

Reference:

AGCGTGCATTCGCCGCGGTCCAACGTGTCAATAGACAACATTTCTCGTCGAAATACACAGTTTCTGGGAGTGATGTGGCTAGGGAATCACTG

G C G C C
G G C A C

AGCGTGCATT GGCGCGGTCC TCAATAGAC ATTTCTCGTC ACACAGTTTC GAGTGATGTG TACGGAATCA
GTGCATTCGG CCGTCCAAGG ATAGACAACC CGTCGAAATA GTTTCTGGGA ATGTGGCTAC AATCACTG
AGCG ATTCGGCGCG CGTGTCAATA CAACCTTTCT CGAAATACAC CTGGGAGTGA GCTACGGAAT CTG

Reference:

AGCGTGCATTCGCCGCGGTCCAACGTGTCAATAGACAACATTTCTCGTCGAAATACACAGTTTCTGGGAGTGATGTGGCTAGGGAATCACTG

G C G C C
G G C A C

ChIP-seq

AGCGTGCATTCGGCGCGGTCCAACGTGTCAATAGACAACATTTCTCGTCGAAATACACAGTTTCTGGGAGTGATGTGGCTACGGAATCACTG

Reference:

AGCGTGCATTCGCCGCGGTCCAACGTGTCAATAGACAACATTTCTCGTCGAAATACACAGTTTCTGGGAGTGATGTGGCTAGGGAATCACTG

G C G C C
G G C A C

Reference:

AGCGTGCATTCGCCGCGGTCCAACGTGTCAATAGACAACATTTCTCGTCGAAATACACAGTTTCTGGGAGTGATGTGGCTAGGGAATCACTG

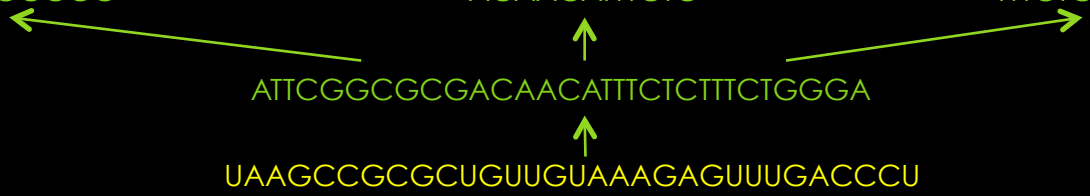
G	C	G	C	C
G	G	C	A	C

CGTGCATTCG CATTTCTCGTCG TGGGAGTGAT

ChIP-seq

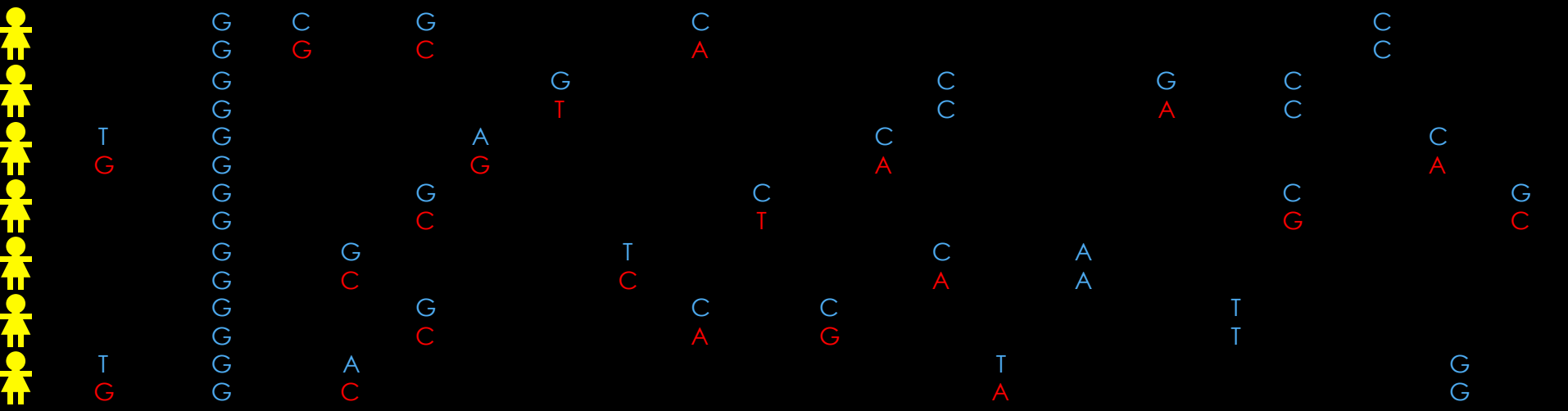
ATTCGGCGCG ACAACATTTCTC TTTCTGGGA

RNA-seq



Reference:

AGCGTGCATTCGCCGCGGTCCAACGTGTCAATAGACAACATTTCTCGTCGAAATACACAGTTCTGGGAGTGATGTGGCTAGGGAATCACTG



CGTGCATTCG CATTTCTCGTCG TGGGAGTGAT ChIP-seq

ATTCGGCGCG ACAACATTTCTC TTTCTGGGA RNA-seq

Reference:

AGCGTGCATTCGCCGCGGTCCAACGTGTCAATAGACAACATTTCTCGTCGAAATACACAGTTCTGGGAGTGATGTGGCTAGGGAATCACTG

CGTGCATTCG
ATTCGGCGCG

CATTTCTCGTCG
ACAACATTTCTC

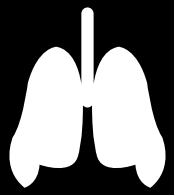
TGGGAGTGAT
TTTCTGGGA

GCATTCGCCG
ATTCGCCGCG

ACAACATTTCTC

TTTCTGGGA

GATGTGGCT



GCATTCGCCG
ATTCGCCGCGG

ATTTCTCGTCGAAATA

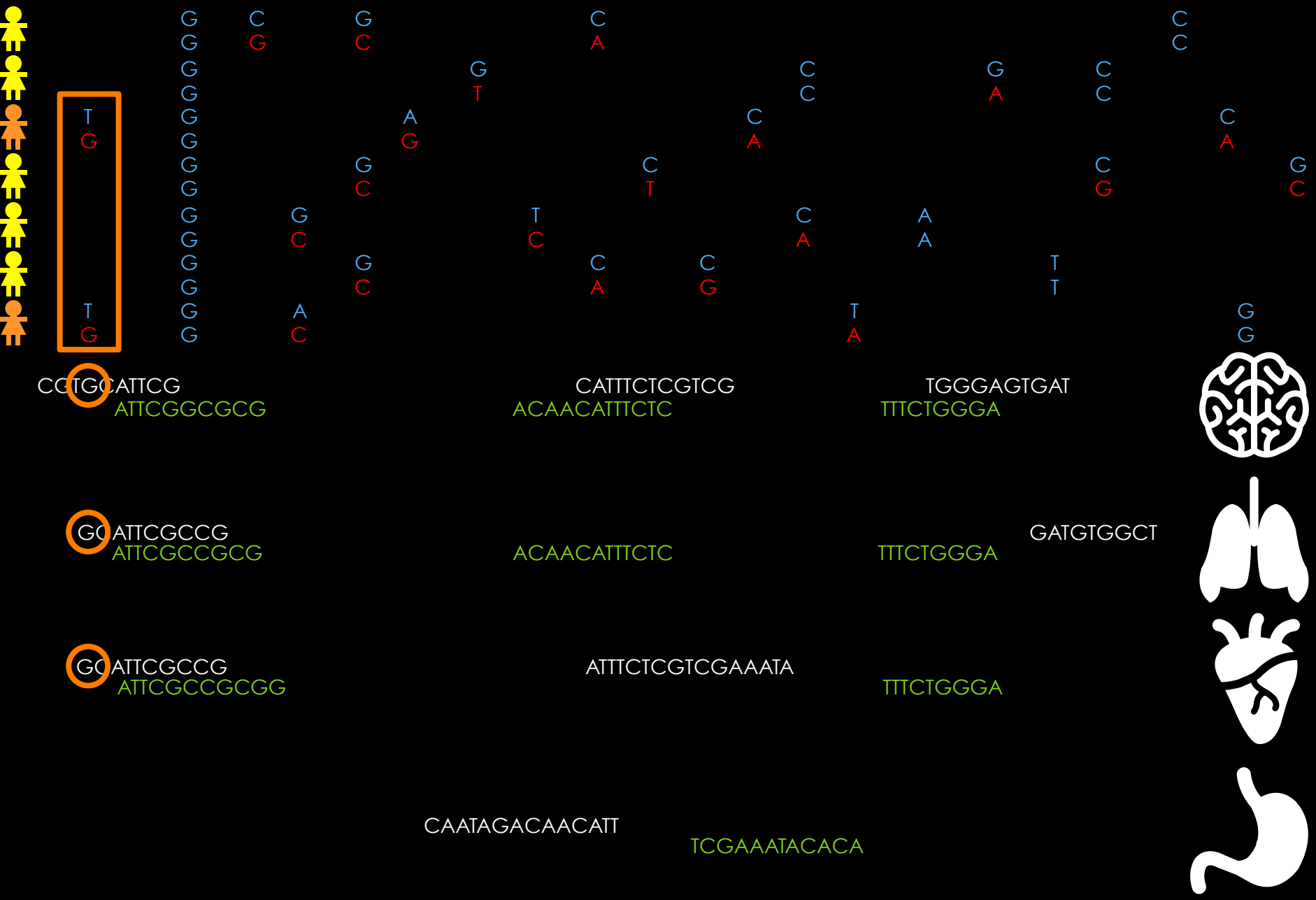
TTTCTGGGA

CAATAGACAACATT

TCGAAATACACA

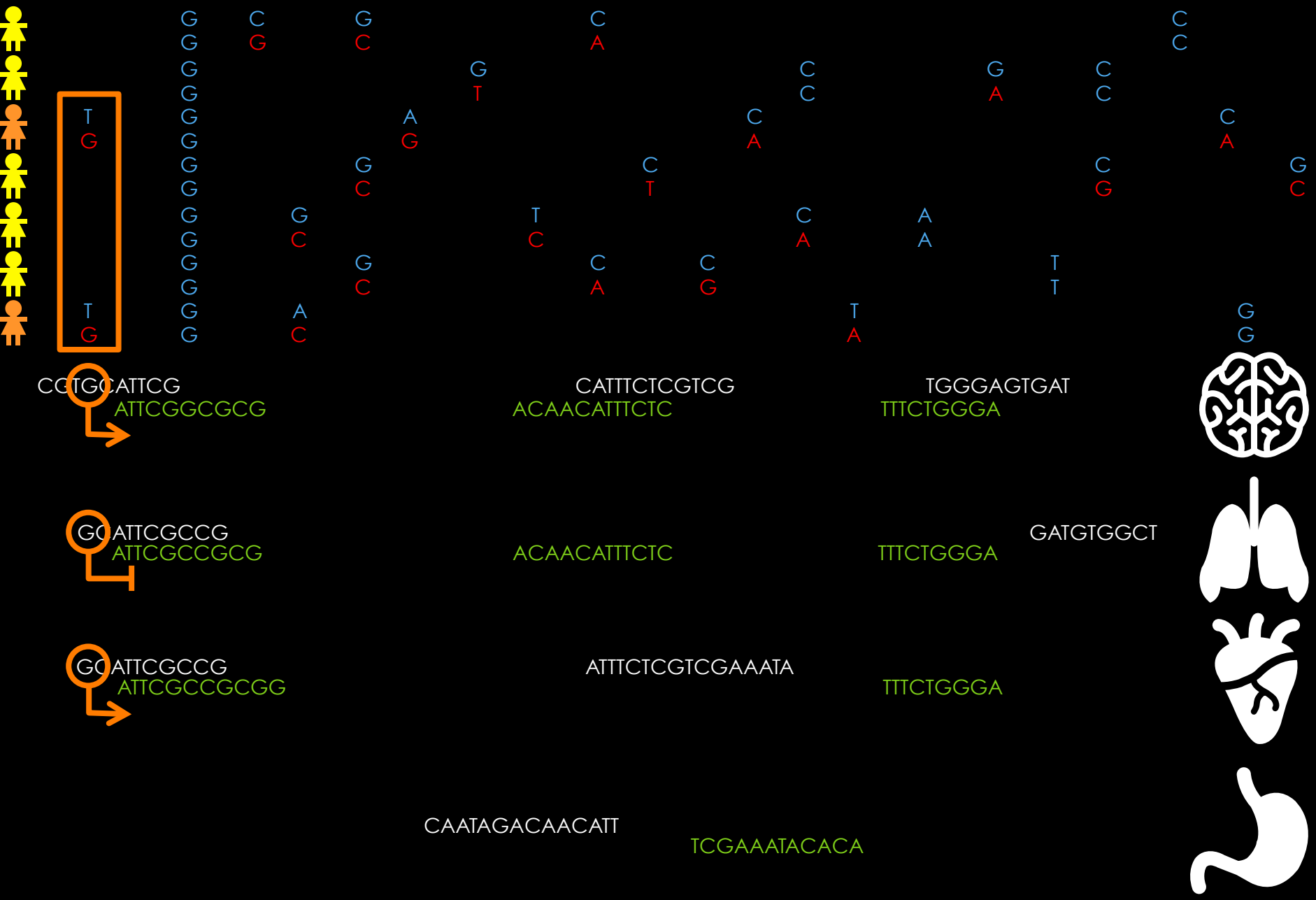
Reference:

AGCGTGCATTCGCCGCGGTCCAACGTGTCAATAGACAACATTTCTCGTCGAAATACACAGTTTCTGGGAGTGATGTGGCTAGGGAATCACTG



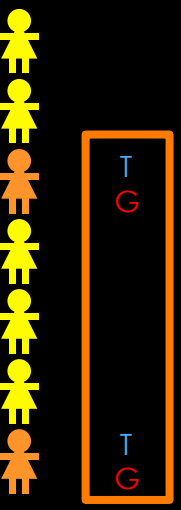
Reference:

AGCGTGCATTCGCCGCGGTCCAACGTGTCAATAGACAACATTTCTCGTCGAAATACACAGTTTCTGGGAGTGATGTGGCTAGGGAATCACTG



Reference:

AGCGTGCATTCGCCGCGGTCCAACGTGTCAATAGACAACATTTCTCGTCGAAATACACAGTTTCTGGGAGTGATGTGGCTAGGGAATCACTG



GENOTYPE QUERY TOOLS

CCTGCATTCG
ATTCGGCGCG

GCATTCGCCG
ATTCGCCGCG

GCATTCGCCG
ATTCGCCGCGG

CATTTCTCGTCG
ACAACATTTCTC

ACAACATTTCTC

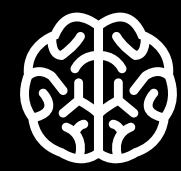
ATTTCTCGTCGAAATA

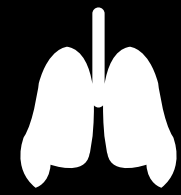
CAATAGACAACATT
TCGAAATACACA

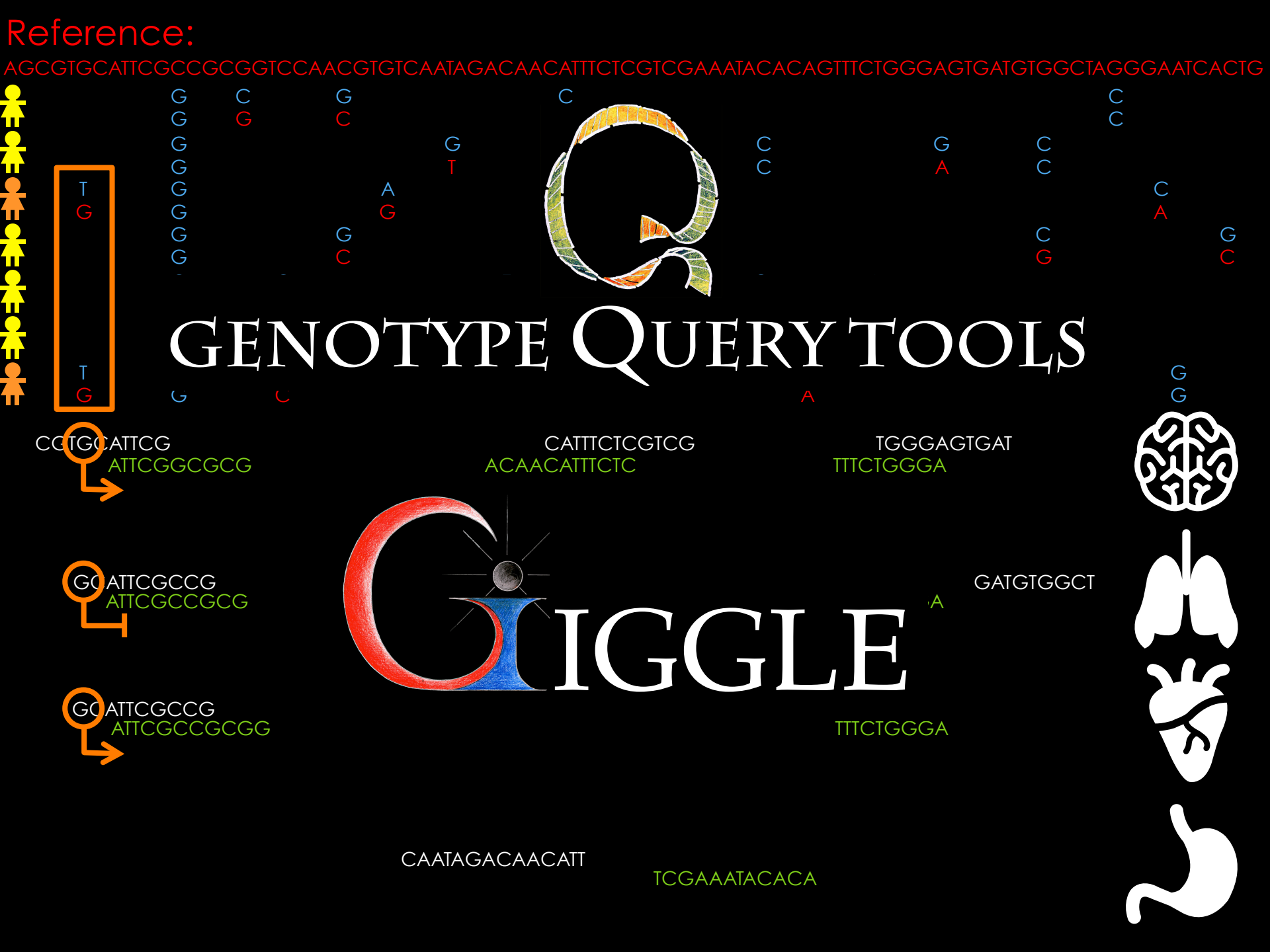
TGGGAGTGAT
TTTCTGGGA

GATGTGGCT
TTTCTGGGA

TTTCTGGGA

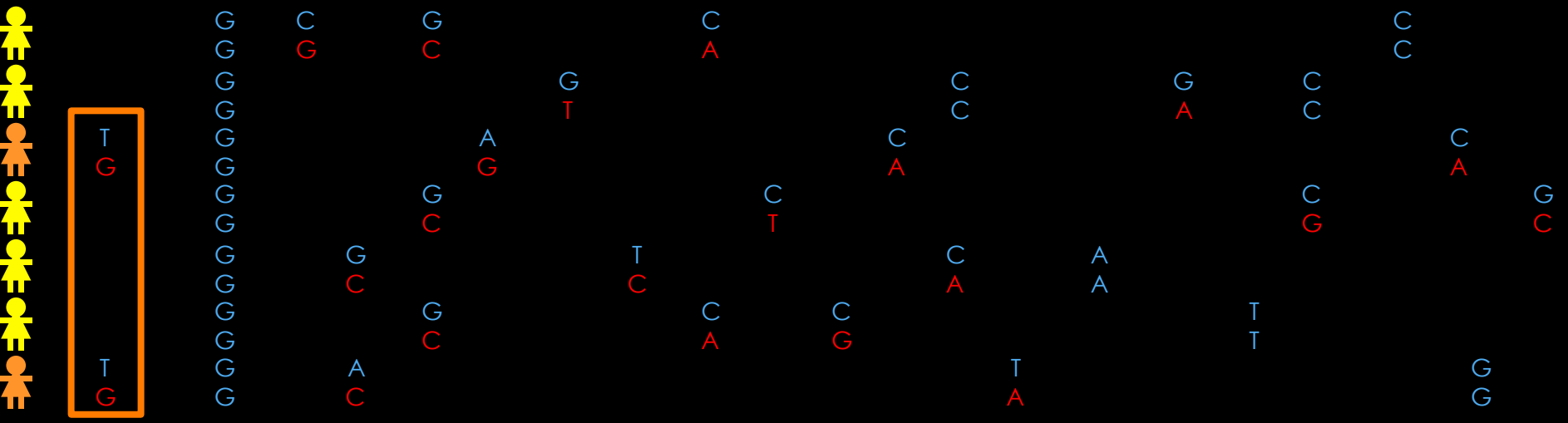





Reference:
AGCGTGCATTCGCCGCGGTCCAACGTGTCAATAGACAACATTTCTCGTCGAAATACACAGTTTCTGGGAGTGATGTGGCTAGGGAATCACTG
GENOTYPE QUERY TOOLS
CGTGCATTCG
ATTCGGCGCG
GCATTCGCCG
ATTCGCCGCG
GCATTCGCCG
ATTCGCCGCGG
CATTTCTCGTCG
ACAACATTTCTC
TGGGAGTGAT
TTTCTGGGA
GIGGLE
GATGTGGCT
TTTCTGGGA
CAATAGACAACATT
TCGAAATACACA

GENOTYPE QUERY TOOLS

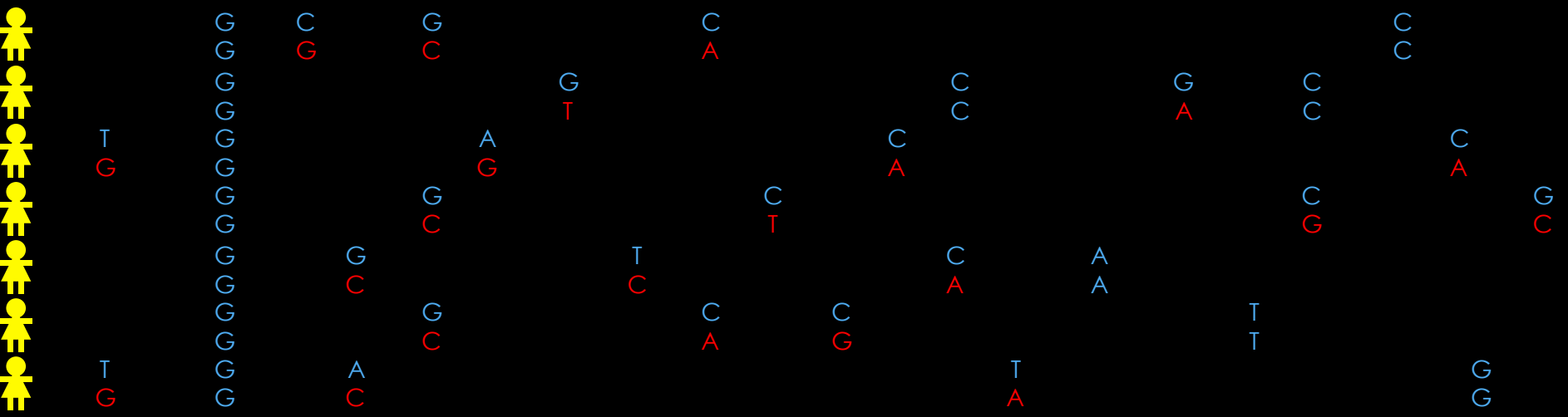
github.com/ryanlayer/gqt

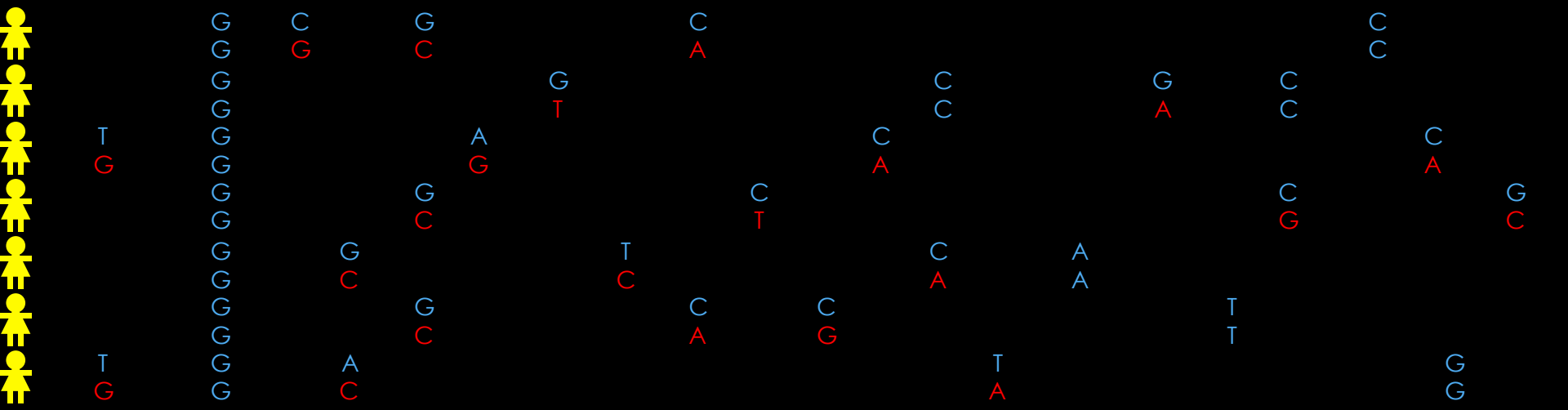


Efficient genotype compression and analysis of large genetic-variation data sets
RM Layer, N Kindlon, Exome Aggregation Consortium, KJ Karczewski, AR Quinlan.
Nature methods, 2016

Reference:

AGCGTGCATTCGCCGCGGTCCAACGTGTCAATAGACAACATTTCTCGTCGAAATACACAGTTTCTGGGAGTGATGTGGCTAGGGAATCACTG

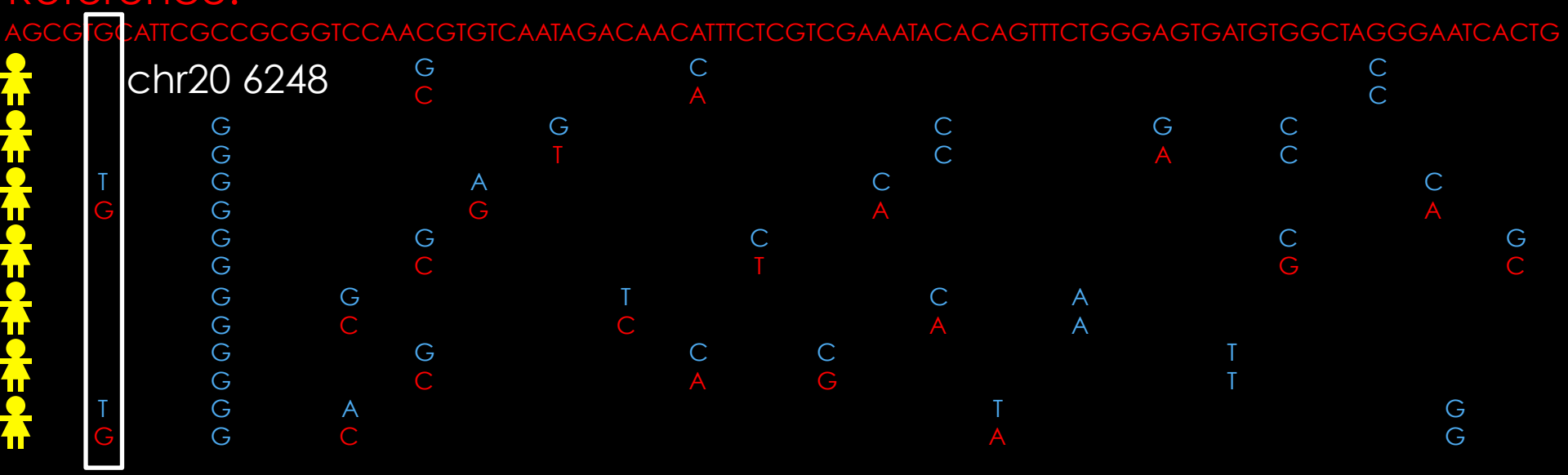




Variant Call Format (VCF)

Variant Data

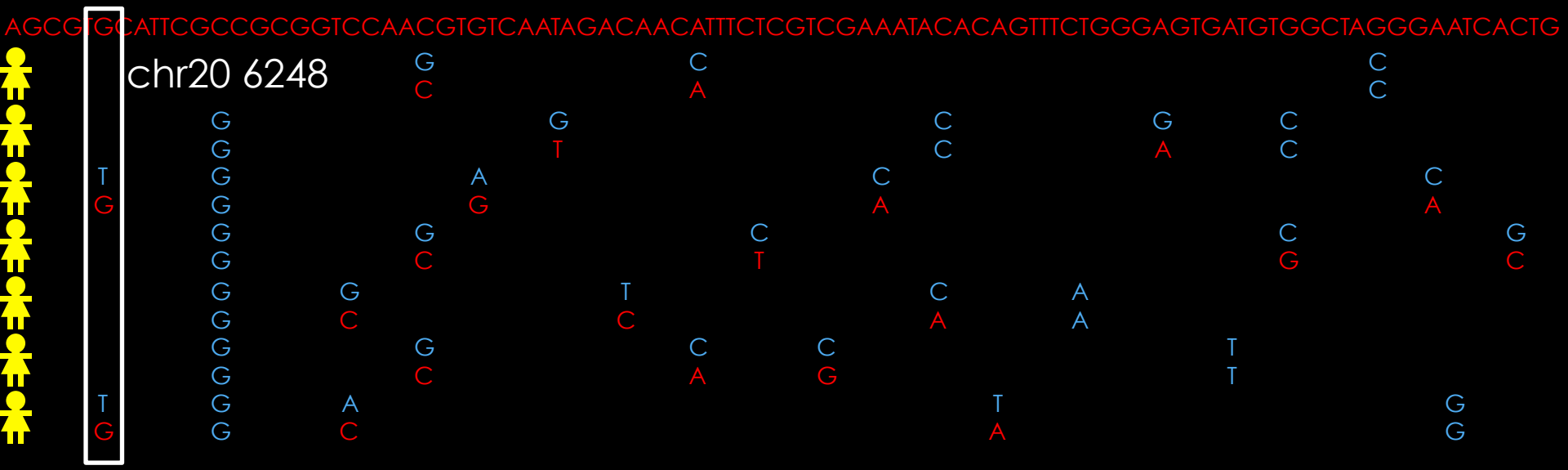
Individual Genotypes



Variant Call Format (VCF)

Variant Data

Individual Genotypes

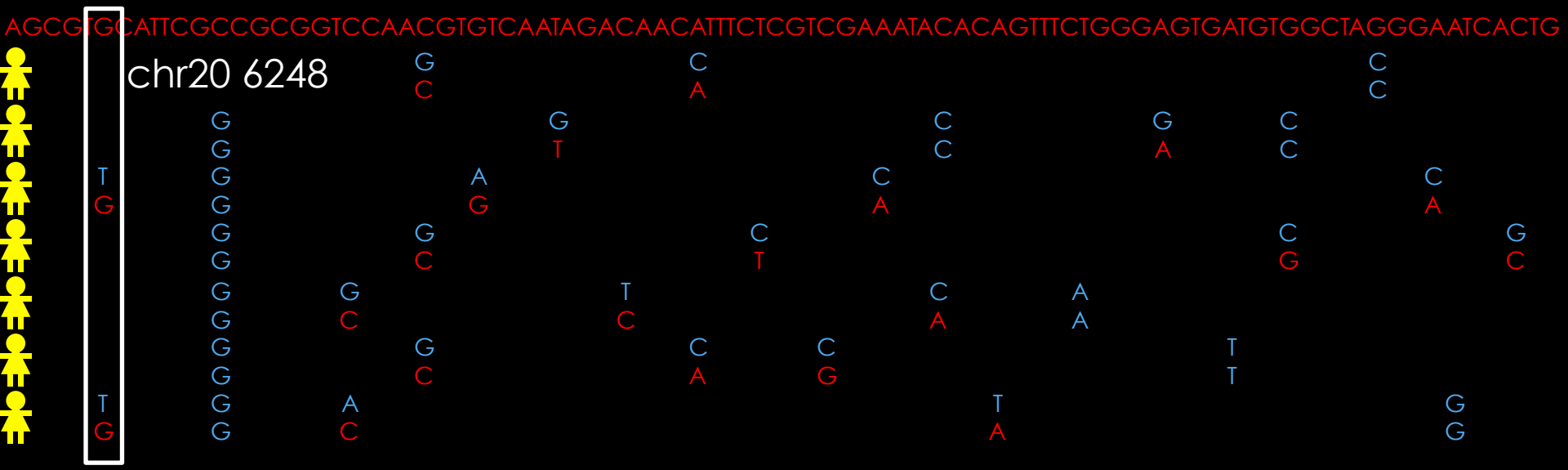


Variant Call Format (VCF)

Variant Data

Individual Genotypes

```
chr20      6248
```

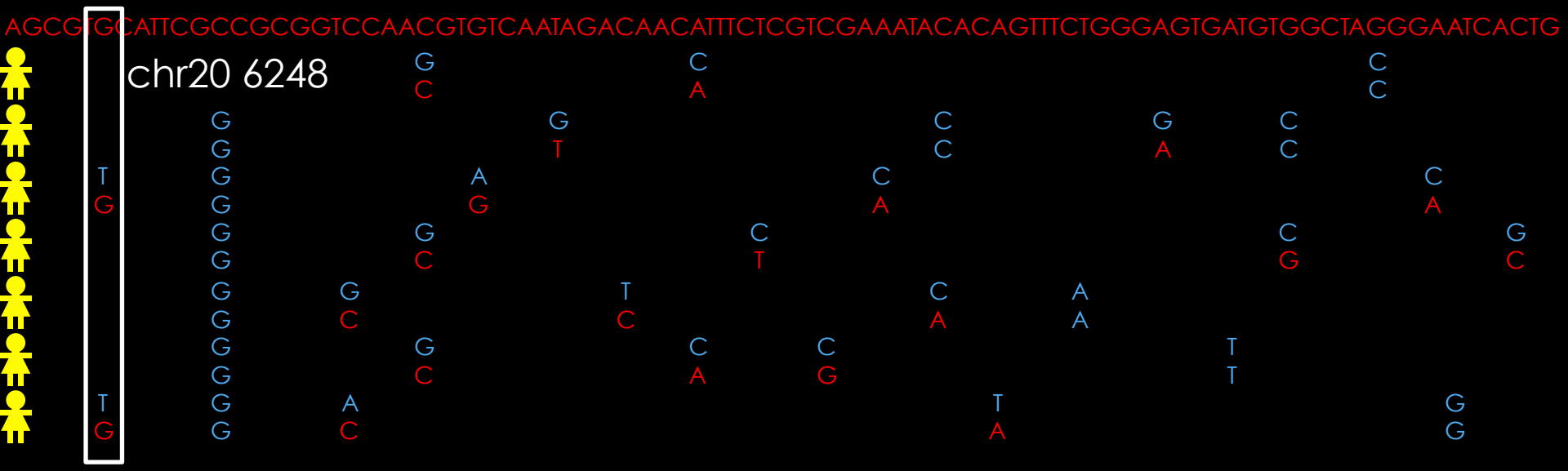


Variant Call Format (VCF)

Variant Data

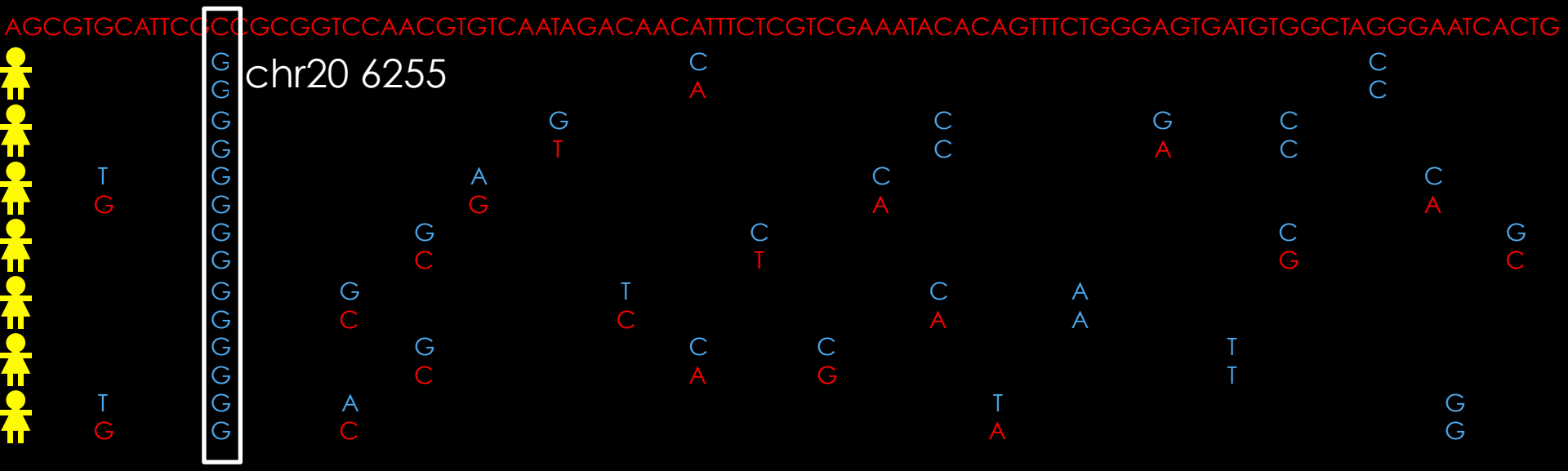
Individual Genotypes

```
chr20    6248    G    T
```



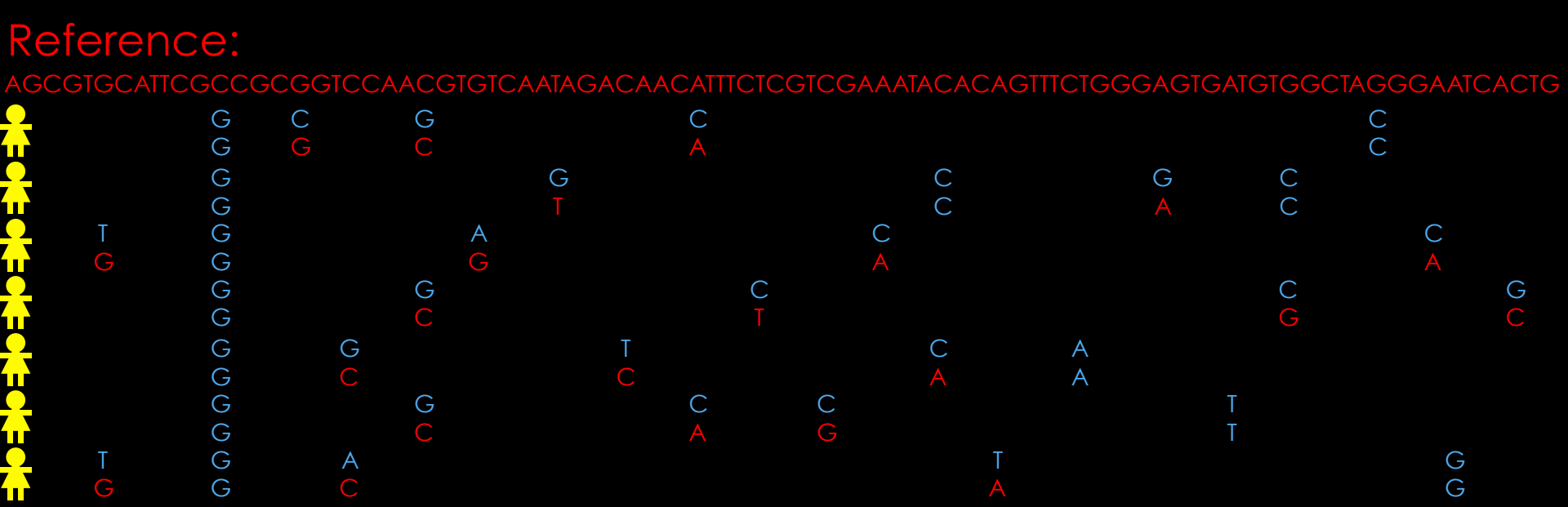
Variant Call Format (VCF)

Variant Data				Individual Genotypes						
chr20	6248	G	T	0/0	0/0	1/0	0/0	0/0	0/0	1/0



Variant Call Format (VCF)

Variant Data				Individual Genotypes						
chr20	6248	G	T	0/0	0/0	1/0	0/0	0/0	0/0	1/0
chr20	6255	C	G	1/1	1/1	1/1	1/1	1/1	1/1	1/1

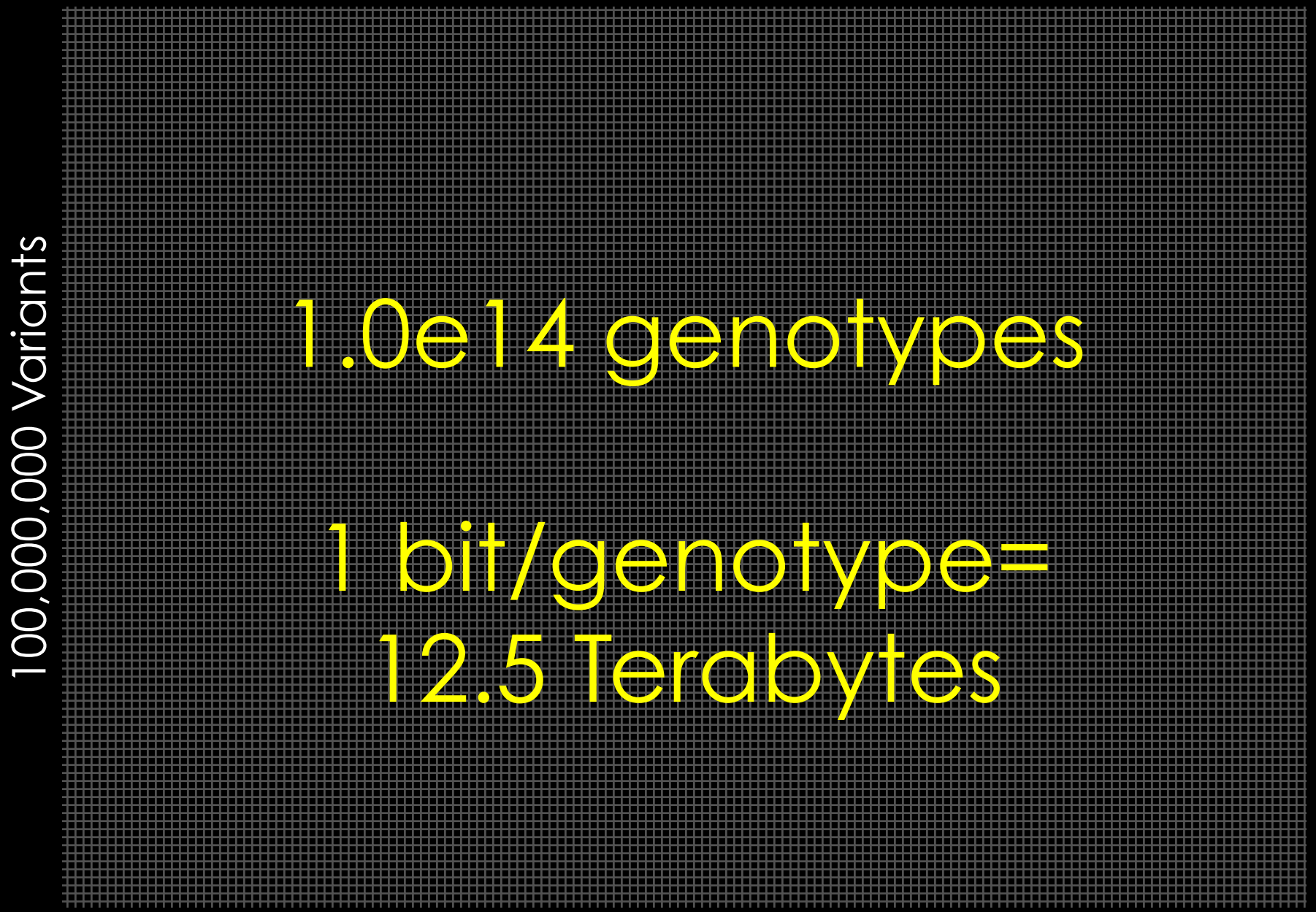


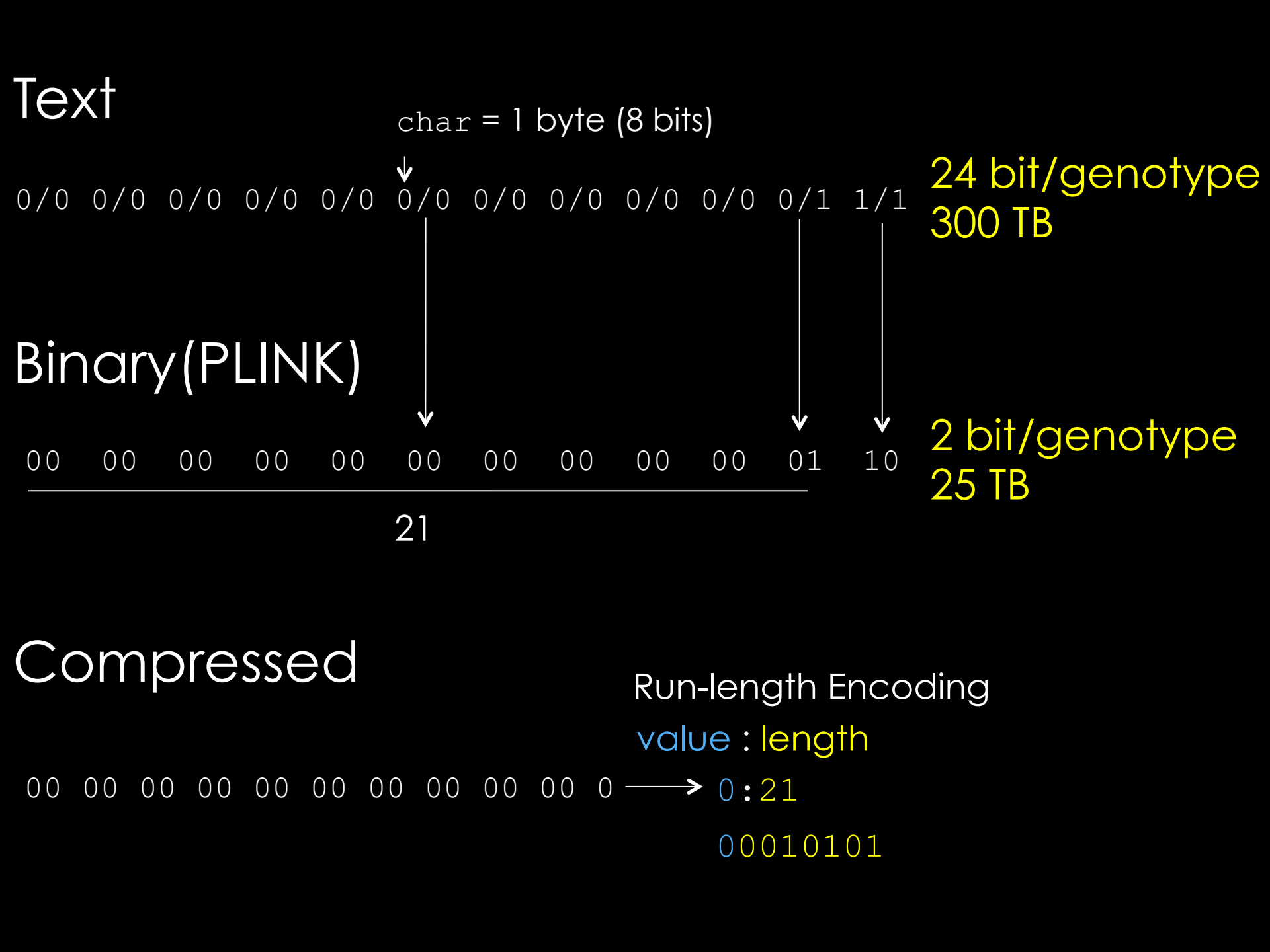
Variant Call Format (VCF)

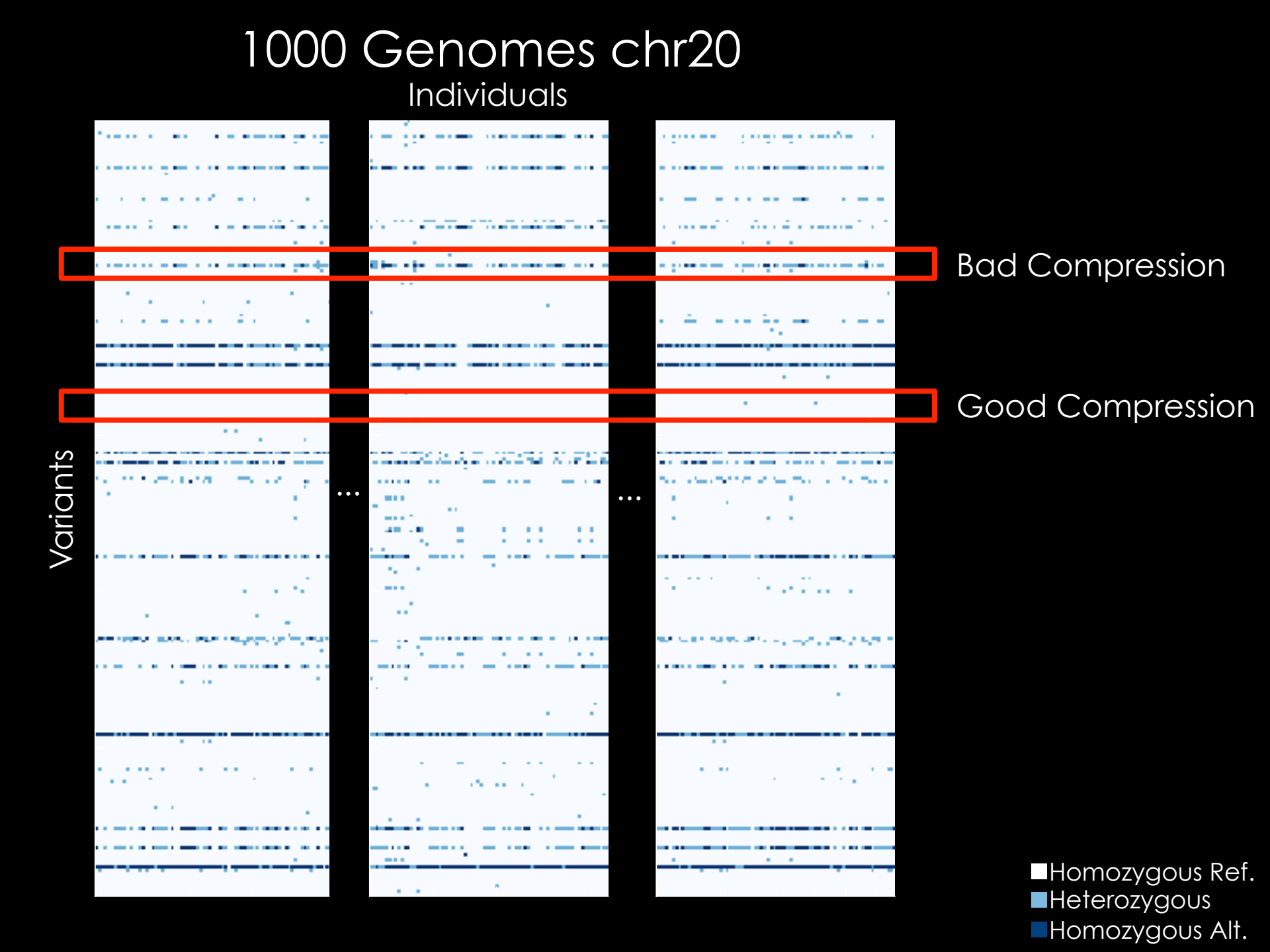
Variant Data

Individual Genotypes

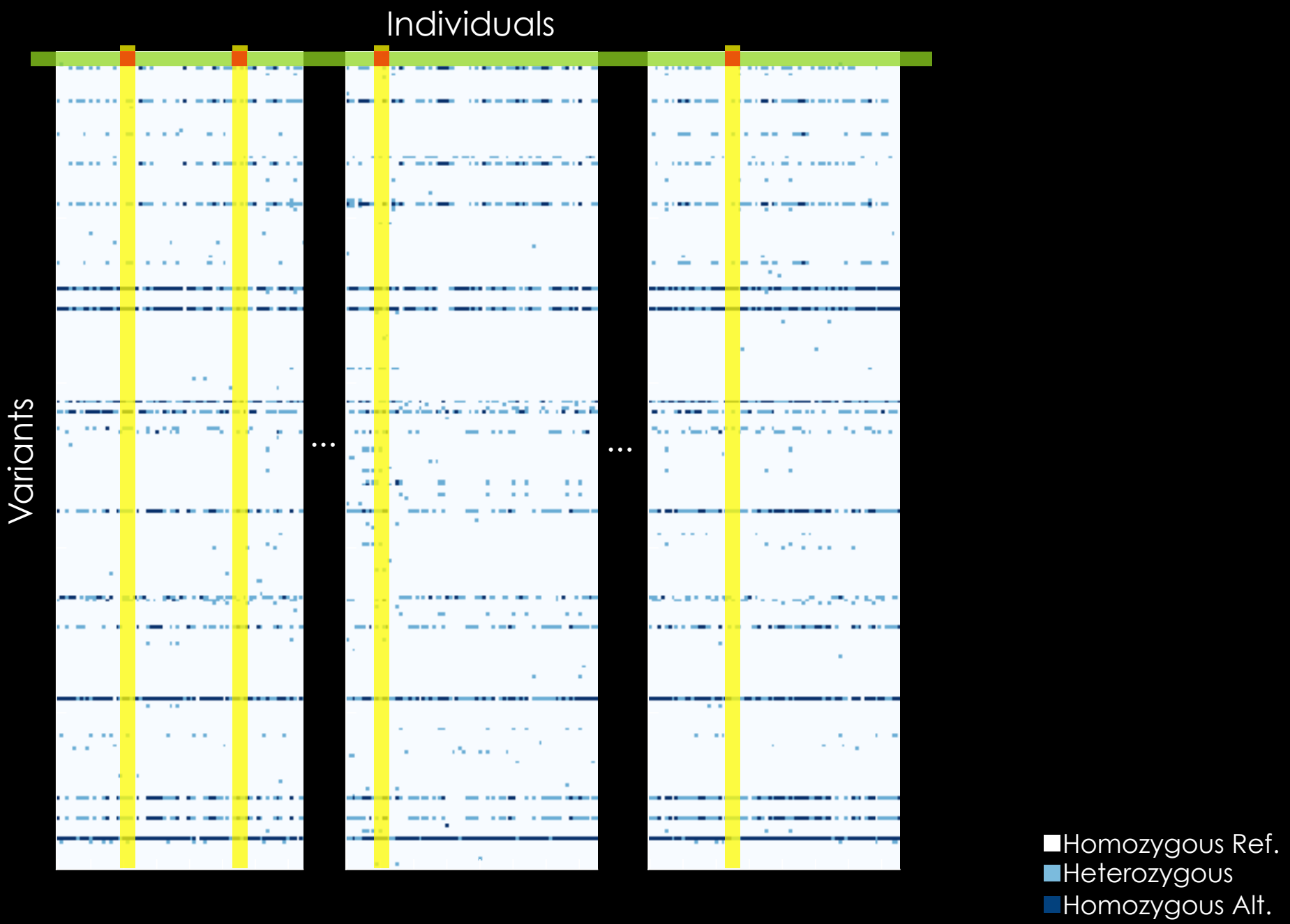
chr20	6248	G	T	0/0	0/0	1/0	0/0	0/0	0/0	1/0
chr20	6255	C	G	1/1	1/1	1/1	1/1	1/1	1/1	1/1
chr20	6259	G	C	1/0	0/0	0/0	0/0	0/0	0/0	0/0
chr20	6262	C	G	0/0	0/0	0/0	0/0	1/0	0/0	1/0
chr20	6266	C	G	1/0	0/0	0/0	1/0	0/0	1/0	0/0
chr20	6269	G	A	0/0	0/0	1/0	0/0	0/0	0/0	0/0
chr20	6274	T	G	0/0	1/0	0/0	0/0	0/0	0/0	0/0
chr20	6278	C	T	0/0	0/0	0/0	0/0	1/0	0/0	0/0
chr20	6282	A	C	1/0	0/0	0/0	0/0	1/0	1/0	0/0
chr20	6287	T	C	0/0	0/0	0/0	1/0	0/0	0/0	0/0
chr20	6291	G	C	0/0	0/0	0/0	0/0	0/0	1/0	0/0
chr20	6294	A	C	0/0	0/0	0/0	0/0	1/1	0/0	0/0
chr20	6298	A	G	0/0	0/0	1/0	0/0	0/0	0/0	0/0
chr20	6301	T	A	0/0	1/1	0/0	0/0	1/0	0/0	0/0
chr20	6306	A	G	1/0	0/0	0/0	0/0	1/0	1/0	0/0
chr20	6311	G	C	0/0	1/1	0/0	1/0	0/0	0/0	0/0


1,000,000 Individuals
100,000,000 Variants
1.0e14 genotypes
1 bit/genotype=
12.5 Terabytes

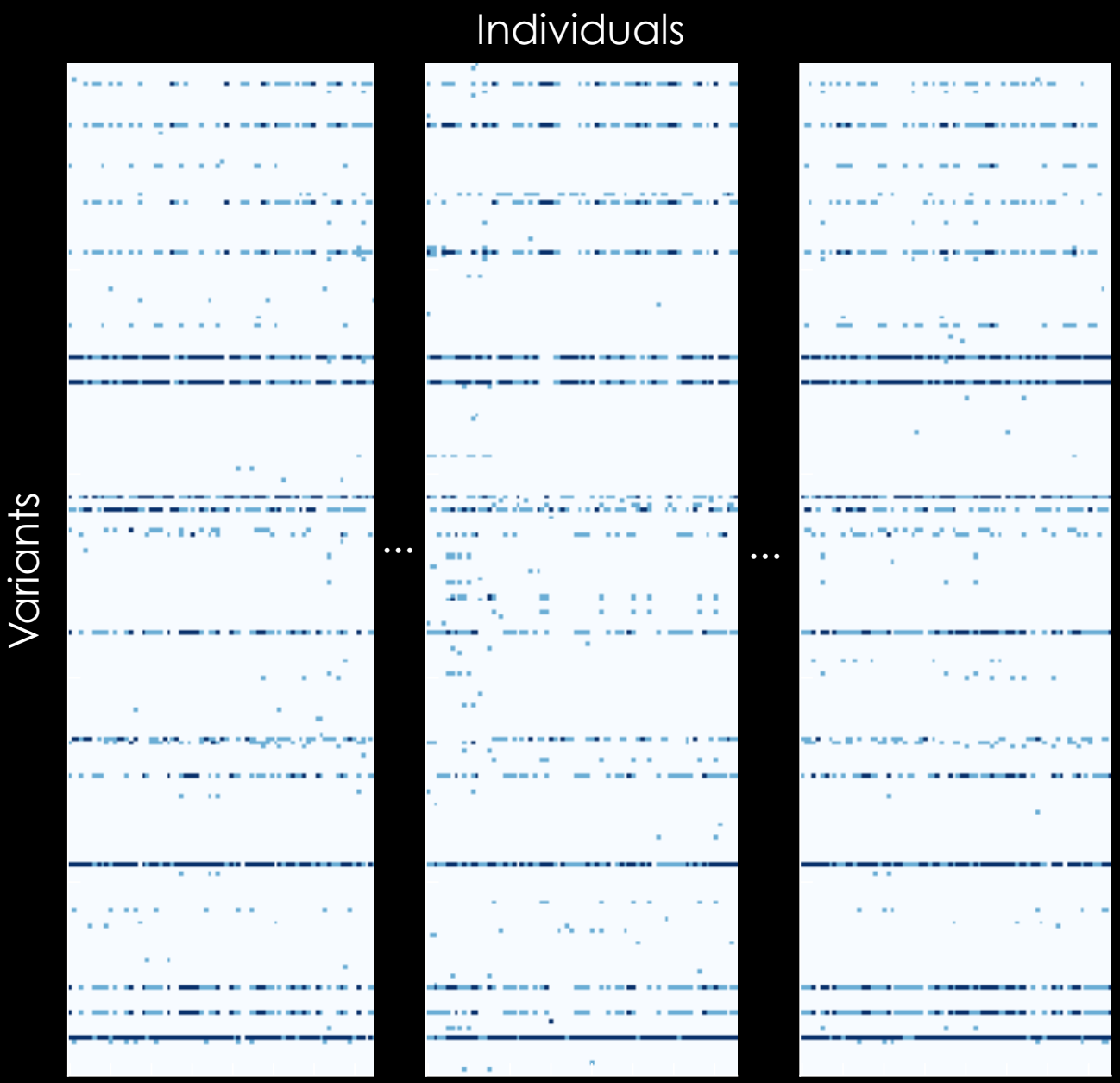

Text
char = 1 byte (8 bits)
0/0 0/0 0/0 0/0 0/0 0/0 0/0 0/0 0/0 0/0 0/1 1/1
24 bit/genotype
300 TB
Binary(PLINK)
00 00 00 00 00 00 00 00 00 00 01 10
2 bit/genotype
25 TB
21
Compressed
Run-length Encoding
value : length
00 00 00 00 00 00 00 00 00 00 0
0:21
00010101


1000 Genomes chr20
Individuals
Variants
Bad Compression
Good Compression
Homozygous Ref.
Heterozygous
Homozygous Alt.

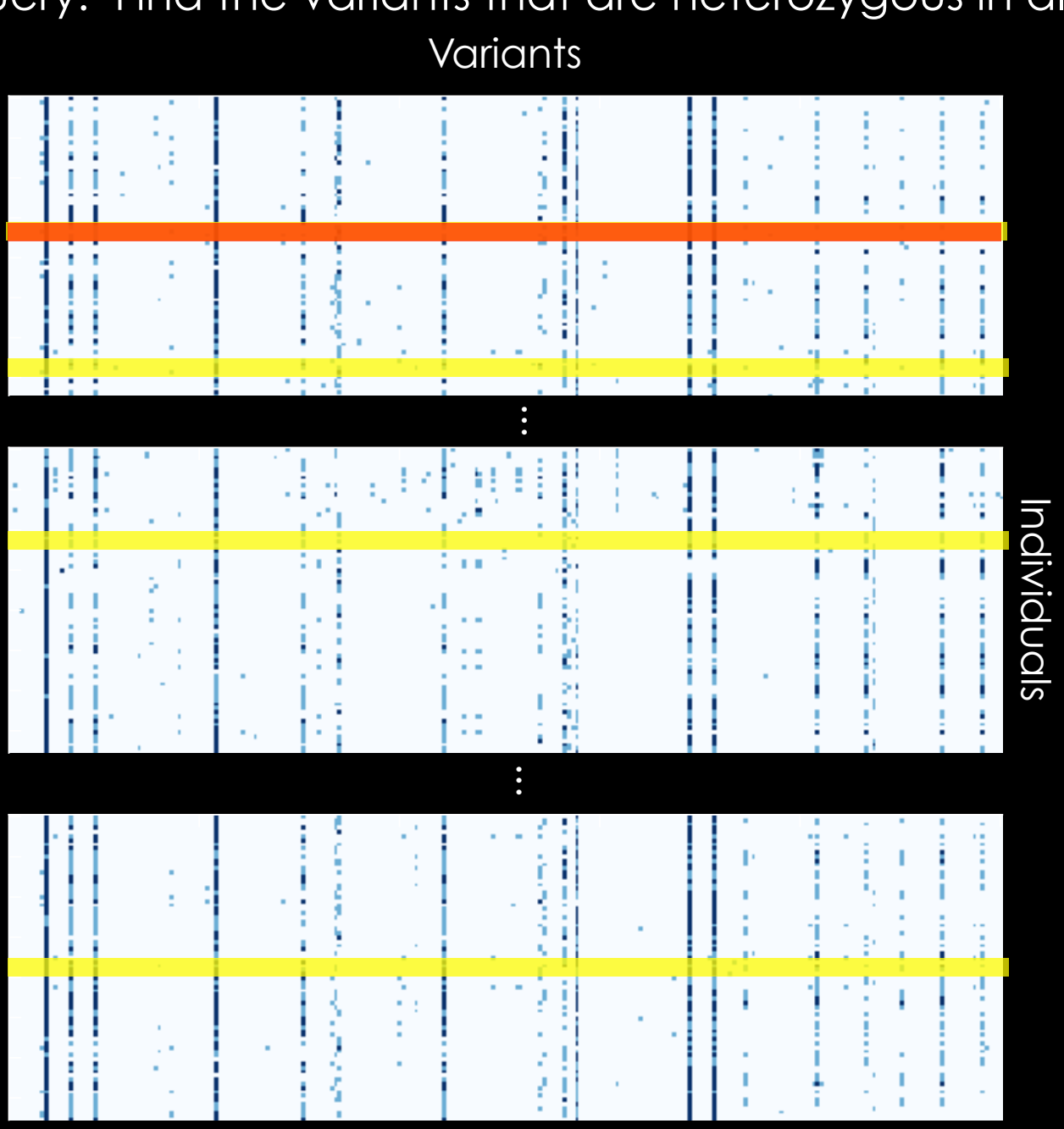
Query: Find the variants that are heterozygous in all affected individuals



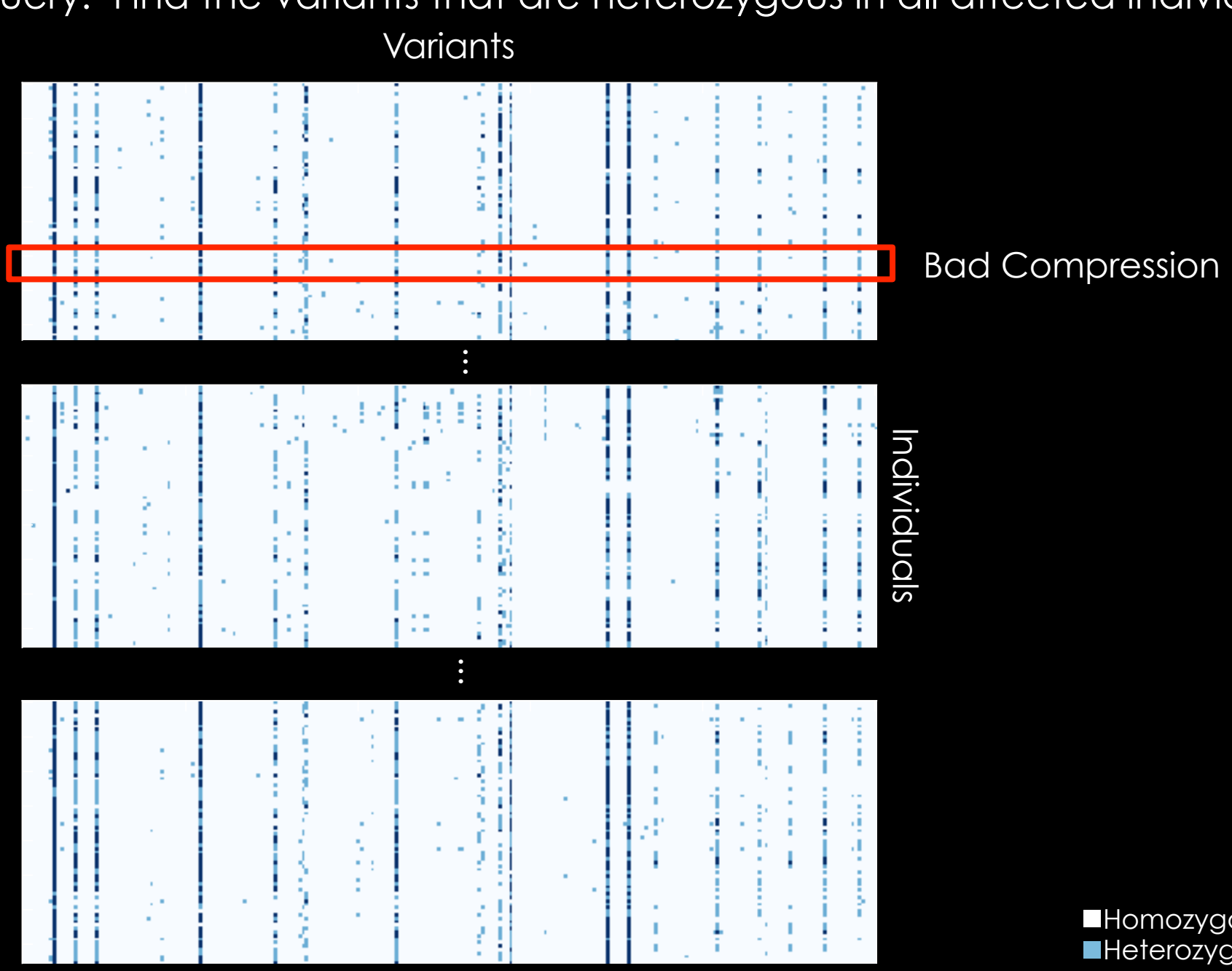
Query: Find the variants that are heterozygous in all affected individuals



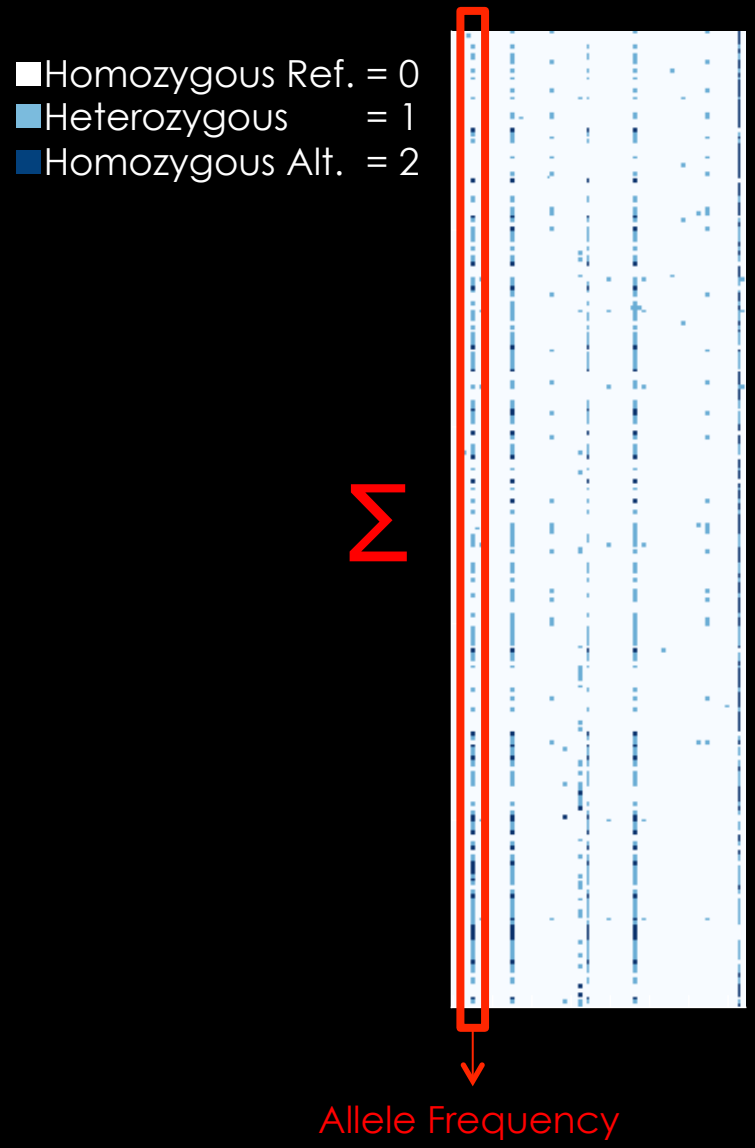
Query: Find the variants that are heterozygous in all affected individuals



Query: Find the variants that are heterozygous in all affected individuals

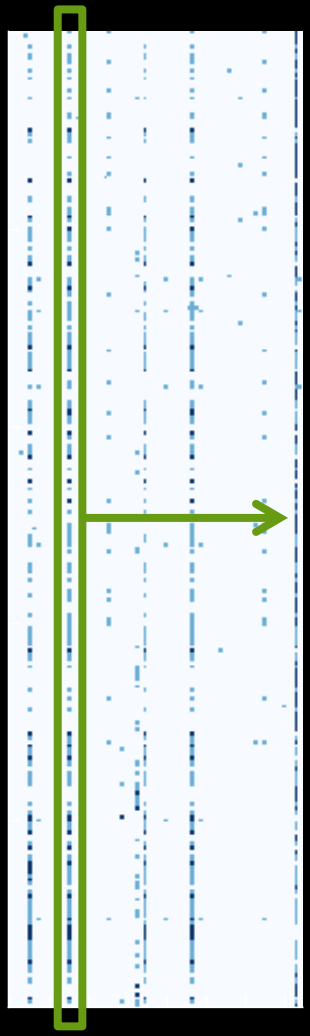


Sort Variants by Allele Frequency



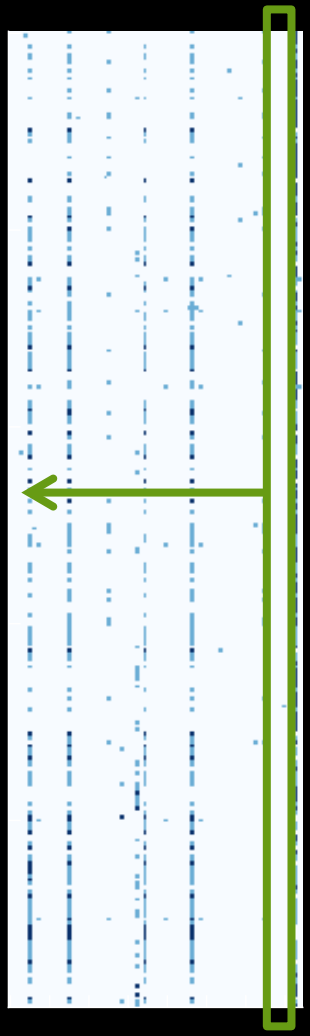
Sort Variants by Allele Frequency

Homozygous Ref. = 0
Heterozygous = 1
Homozygous Alt. = 2



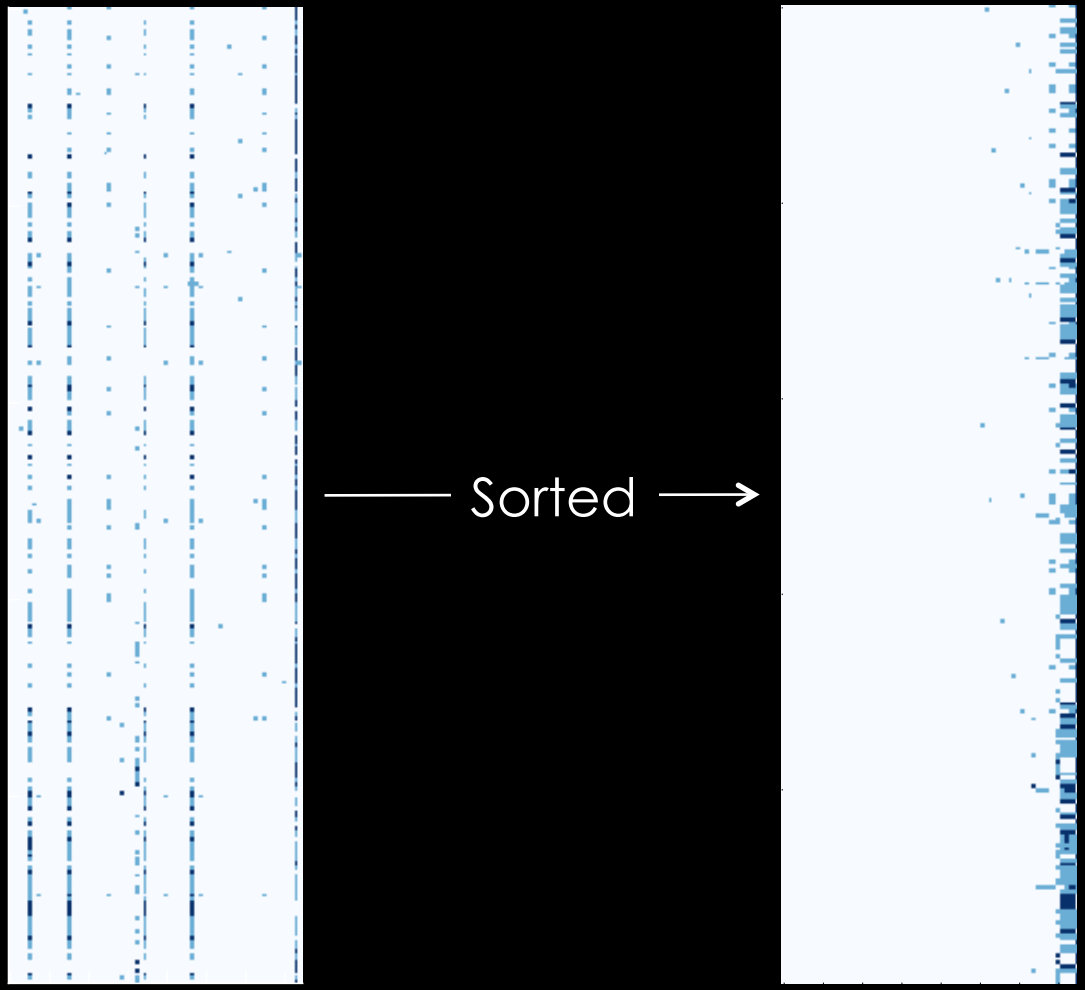
Sort Variants by Allele Frequency

Homozygous Ref. = 0
Heterozygous = 1
Homozygous Alt. = 2

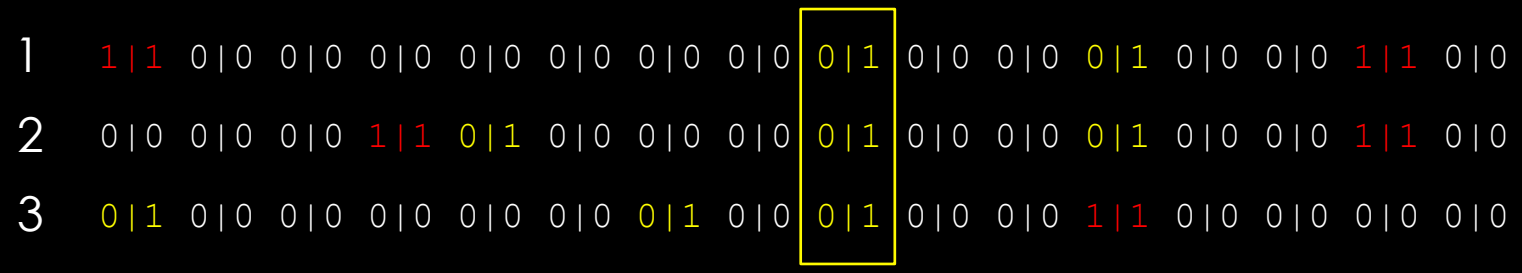


Sort Variants by Allele Frequency

Homozygous Ref. = 0
Heterozygous = 1
Homozygous Alt. = 2



Query: Find the variants that are heterozygous in individuals 1, 2, and 3



Bitmap Index: map records to bit arrays then answer queries using bitwise logical operations

1 1|1 0|0 0|0 0|0 0|0 0|0 0|0 0|0 0|0 0|1 0|0 0|0 0|1 0|0 0|0 1|1 0|0

Bitmap Index: map records to bit arrays then answer queries using bitwise logical operations

1	1|1	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|1	0|0	0|0	0|1	0|0	0|0	1|1	0|0
Homo Ref	0	1	1	1	1	1	1	1	0	1	1	0	1	1	0	1

Bitmap Index: map records to bit arrays then answer queries using bitwise logical operations

1	1|1	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|1	0|0	0|0	0|1	0|0	0|0	1|1	0|0
Homo Ref	0	1	1	1	1	1	1	1	0	1	1	0	1	1	0	1
Het	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0

Bitmap Index: map records to bit arrays then answer queries using bitwise logical operations

1	1|1	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|1	0|0	0|0	0|1	0|0	0|0	1|1	0|0
Homo Ref	0	1	1	1	1	1	1	1	0	1	1	0	1	1	0	1
Het	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0
Homo Alt	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Bitmap Index: map records to bit arrays then answer queries using bitwise logical operations

1	1|1	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|1	0|0	0|0	0|1	0|0	0|0	1|1	0|0
Homo Ref	0	1	1	1	1	1	1	1	0	1	1	0	1	1	0	1
Het	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0
Homo Alt	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
Unknown	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bitmap Index: map records to bit arrays then answer queries using bitwise logical operations

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1|1	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|1	0|0	0|0	0|1	0|0	0|0	1|1	0|0
Homo Ref	0	1	1	1	1	1	1	1	0	1	1	0	1	1	0	1
Het	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0
Homo Alt	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
Unknown	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

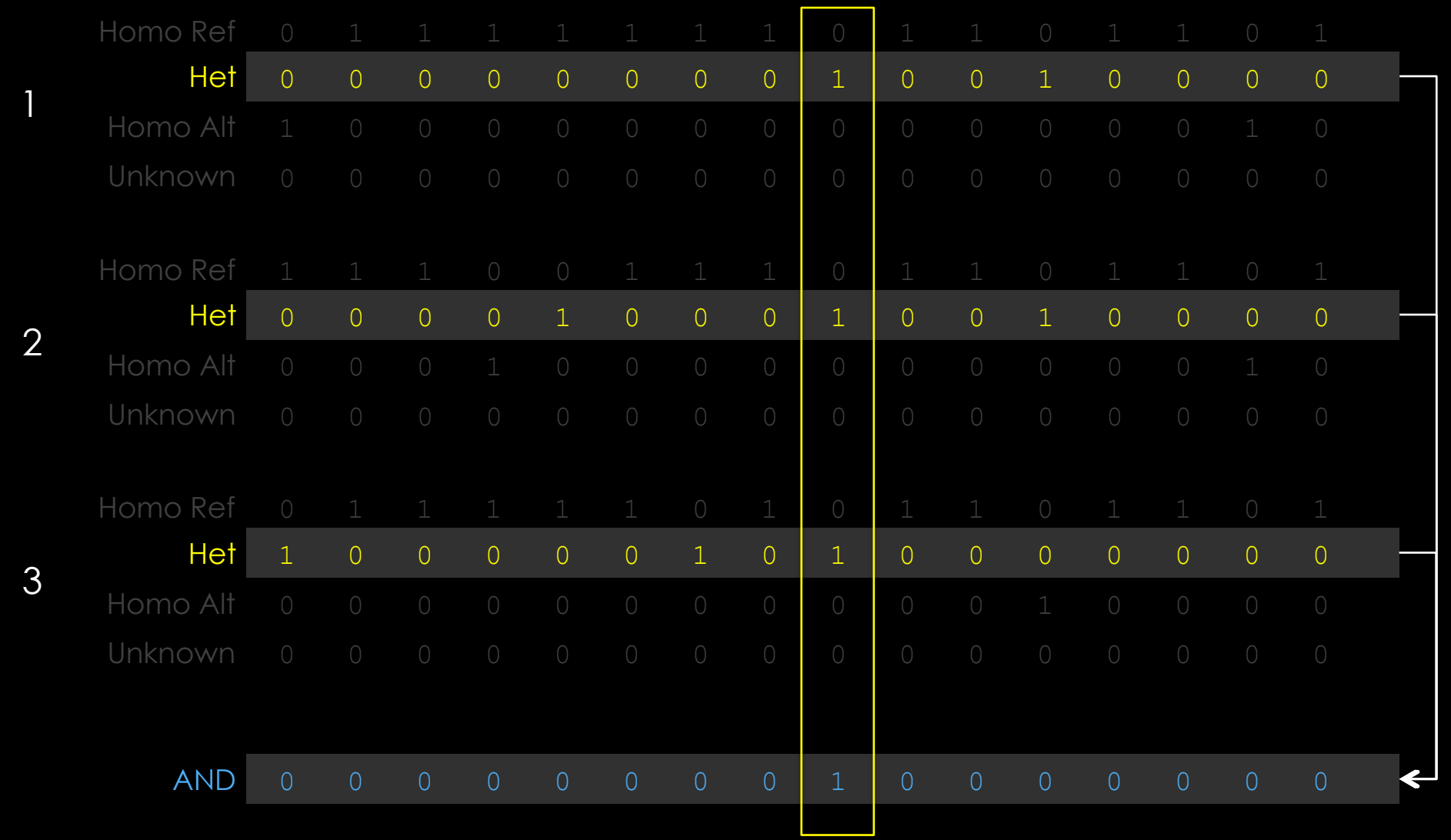
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	0|0	0|0	0|0	1|1	0|1	0|0	0|0	0|0	0|1	0|0	0|0	0|1	0|0	0|0	1|1	0|0
Homo Ref	1	1	1	0	0	1	1	1	0	1	1	0	1	1	0	1
Het	0	0	0	0	1	0	0	0	1	0	0	1	0	0	0	0
Homo Alt	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0
Unknown	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
3	0|1	0|0	0|0	0|0	0|0	0|0	0|1	0|0	0|1	0|0	0|0	1|1	0|0	0|0	0|0	0|0
Homo Ref	0	1	1	1	1	1	0	1	0	1	1	0	1	1	0	1
Het	1	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
Homo Alt	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
Unknown	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Query: Find the variants that are heterozygous in individuals 1, 2, and 3

1	Homo Ref	0	1	1	1	1	1	1	1	0	1	1	0	1	1	0	1
	Het	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0
	Homo Alt	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
	Unknown	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	Homo Ref	1	1	1	0	0	1	1	1	0	1	1	0	1	1	0	1
	Het	0	0	0	0	1	0	0	0	1	0	0	1	0	0	0	0
	Homo Alt	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0
	Unknown	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	Homo Ref	0	1	1	1	1	1	0	1	0	1	1	0	1	1	0	1
	Het	1	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
	Homo Alt	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
	Unknown	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Query: Find the variants that are heterozygous in individuals 1, 2, and 3



Genotype Filters

```
"HET"
"HOM_REF"
"HOM_ALT"
```

Summary Statistics

```
"count(HET)"
"pct(HET HOM_ALT)"
"maf()"
```

Statistic Filters

```
"pct(HET HOM_ALT) <= 0.01"
"count(HOM_REF) > 1"
"maf() < 0.1"
```

Word Aligned Hybrid Wu, TODS 2006

VCF `1|1 0|0 0|0 0|0 0|0 0|0 0|0 0|0 0|1 0|0` 24 bits/genotype

Binary `10000000 00000000 01000001 00001000` 2 bits/genotype

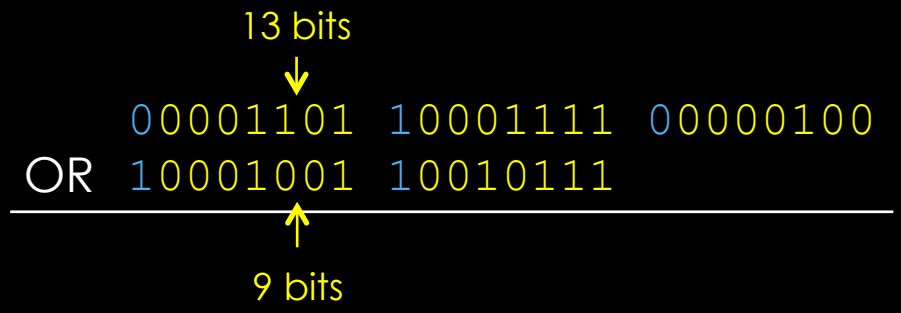
Bit Map

```
01111111 01101101
00000000 10010000
10000000 00000010
00000000 00000000
```

4 bits/genotype

Run-Length Encoding Complicates* Logical Operations

```
    00000000 00000111 11111111 11110000
OR  11111111 10000000 00000000 00000000
-------------------------------------------
    11111111 10000111 11111111 11111111
```



Succinct Data Structure

- Near-optimal compression
- Allow operations without inflation

*likely, but not proven to be impractical

Word Aligned Hybrid [Wu, TODS 2006]

Run Length Encoding

`10010000` →

value : length (bits)

`1:1 0:2 1:1 0:4`

`10000001 00000010 10000001 00000100`

Word Aligned Hybrid

Literal

`1001000` →

type : uncompressed value

`0101000`

Fill

`0000000 0000000` →

type : value : length (words)

`1000010`

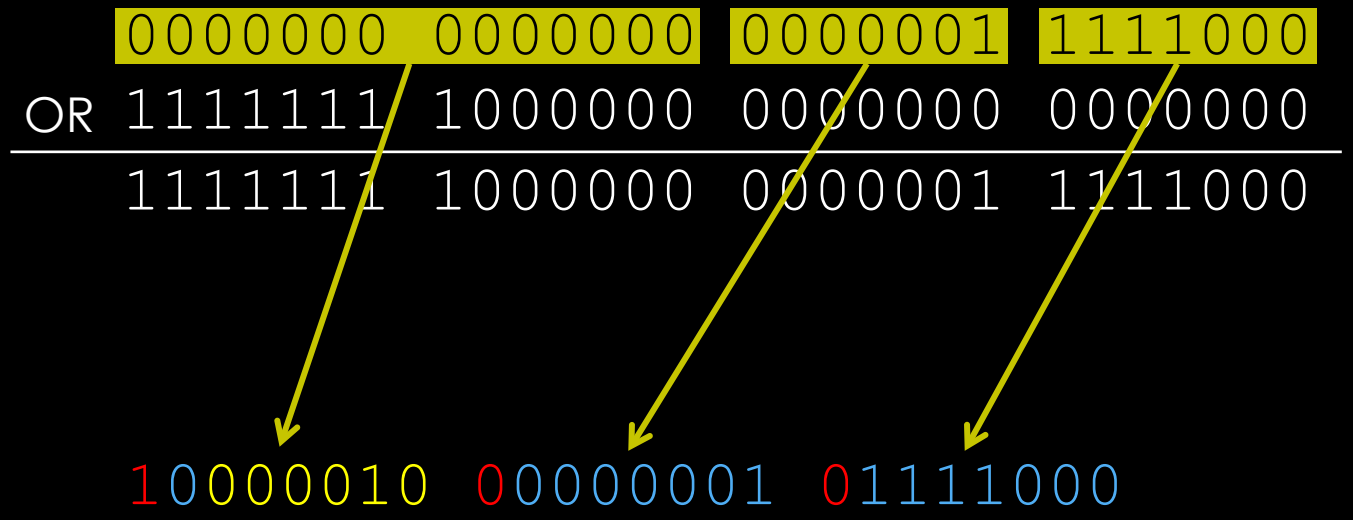
`1111111 1111111 1111111` →

`1100011`

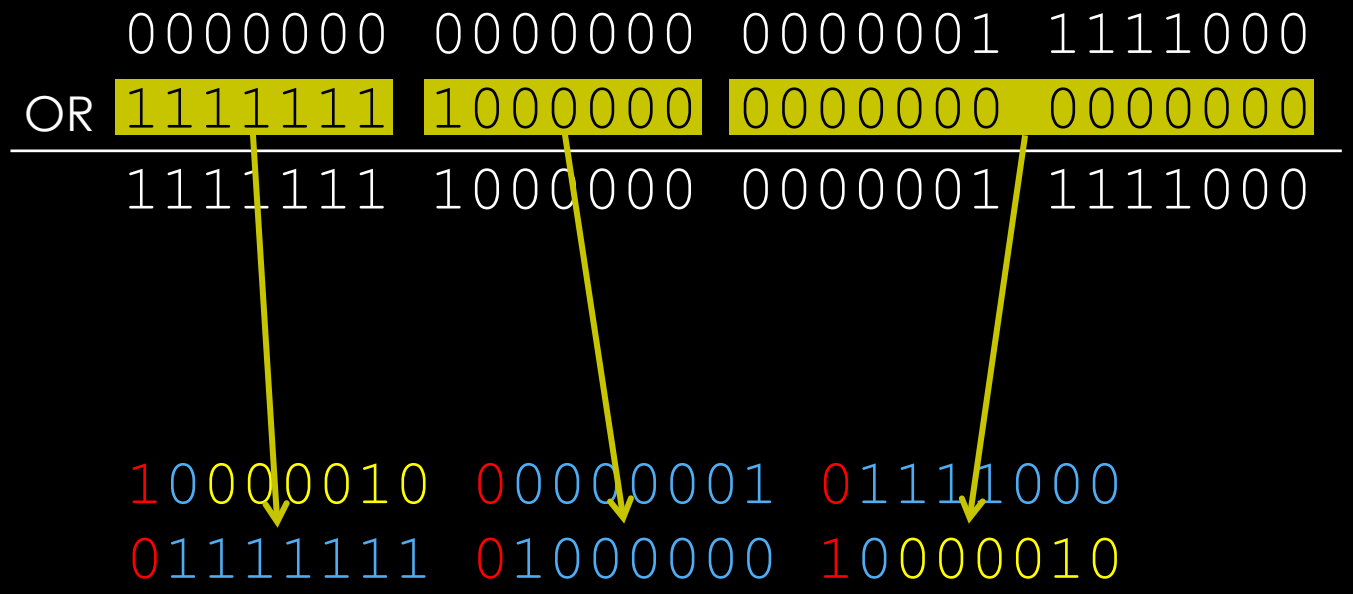
Word Aligned Hybrid [Wu, TODS 2006]

```
    0000000 0000000 0000001 1111000
OR  1111111 1000000 0000000 0000000
-----------------------------------
    1111111 1000000 0000001 1111000
```

Word Aligned Hybrid [Wu, TODS 2006]



Word Aligned Hybrid [Wu, TODS 2006]



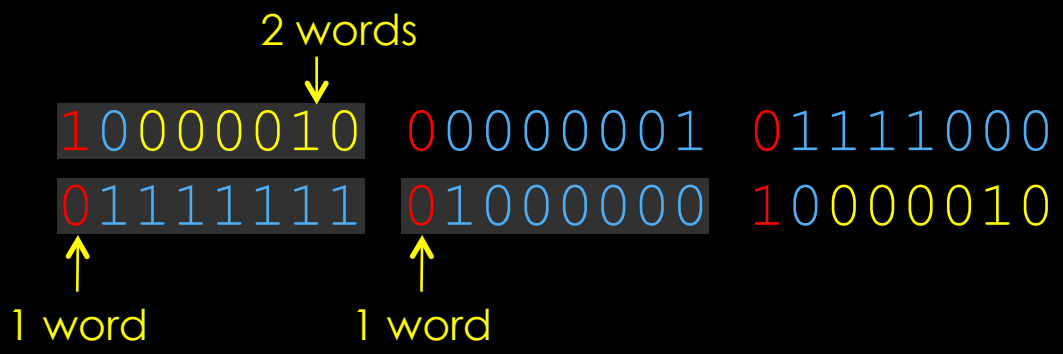
Word Aligned Hybrid [Wu, TODS 2006]

```
    0000000 0000000 0000001 1111000
OR  1111111 1000000 0000000 0000000
-------------------------------------
    1111111 1000000 0000001 1111000
```

```
10000010 00000001 01111000
01111111 01000000 10000010
```

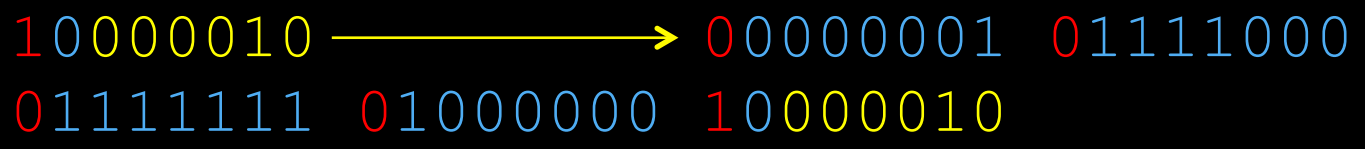
Word Aligned Hybrid [Wu, TODS 2006]

```
    0000000 0000000 0000001 1111000
OR  1111111 1000000 0000000 0000000
-------------------------------------
    1111111 1000000 0000001 1111000
```



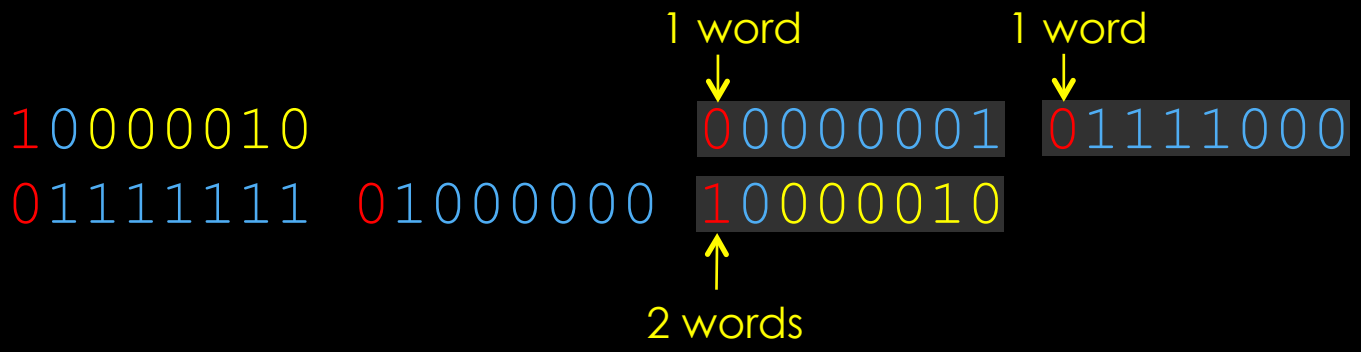
Word Aligned Hybrid [Wu, TODS 2006]

```
   0000000 0000000 0000001 1111000
OR 1111111 1000000 0000000 0000000
-------------------------------------
   1111111 1000000 0000001 1111000
```



Word Aligned Hybrid [Wu, TODS 2006]

```
    0000000 0000000 0000001 1111000
OR  1111111 1000000 0000000 0000000
-------------------------------------
    1111111 1000000 0000001 1111000
```



Word Aligned Hybrid [Wu, TODS 2006]

```
    0000000 0000000 0000001 1111000
OR  1111111 1000000 0000000 0000000
-------------------------------------
    1111111 1000000 0000001 1111000
```

```
    10000010            00000001 01111000
OR  01111111 01000000 10000010
-------------------------------------------
```

Word Aligned Hybrid [Wu, TODS 2006]

```
    0000000 0000000 0000001 1111000
OR  1111111 1000000 0000000 0000000
-----------------------------------
    1111111 1000000 0000001 1111000
```

```
    10000010            00000001 01111000
OR  01111111 01000000 10000010
-----------------------------------
    01111111 01000000 00000001 01111000
```

Word Aligned Hybrid [Wu, TODS 2006]

Smaller Files:

Compresses with run-length & literal values

Faster Queries:

Bitwise logical operations without inflation

GQT v BCFTOOLS & PLINK v1.9

The Exome Aggregation Consortium (ExAC)

60K individual 9.3M variants

Linux v3.13, gcc v4.8.2, 4.0 GHz Intel Core i7 (Haswell)

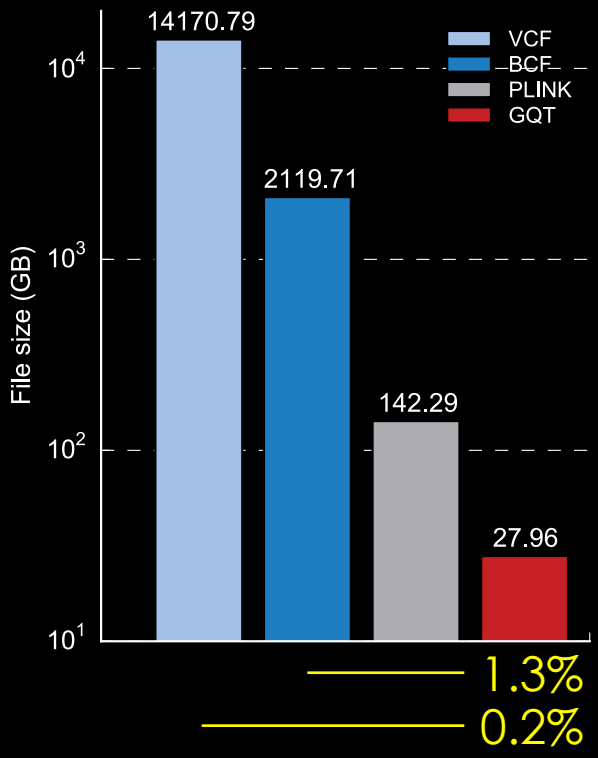
GQT v BCFTOOLS & PLINK v1.9

The Exome Aggregation Consortium (ExAC)

60K individual 9.3M variants

Linux v3.13, gcc v4.8.2, 4.0 GHz Intel Core i7 (Haswell)

Index size



Konrad Karczewski

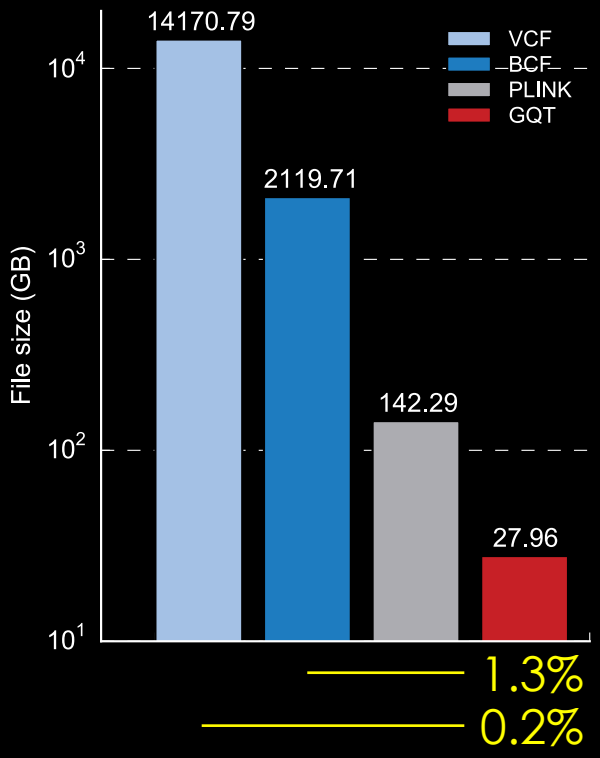
GQT v BCFTOOLS & PLINK v1.9

The Exome Aggregation Consortium (ExAC)

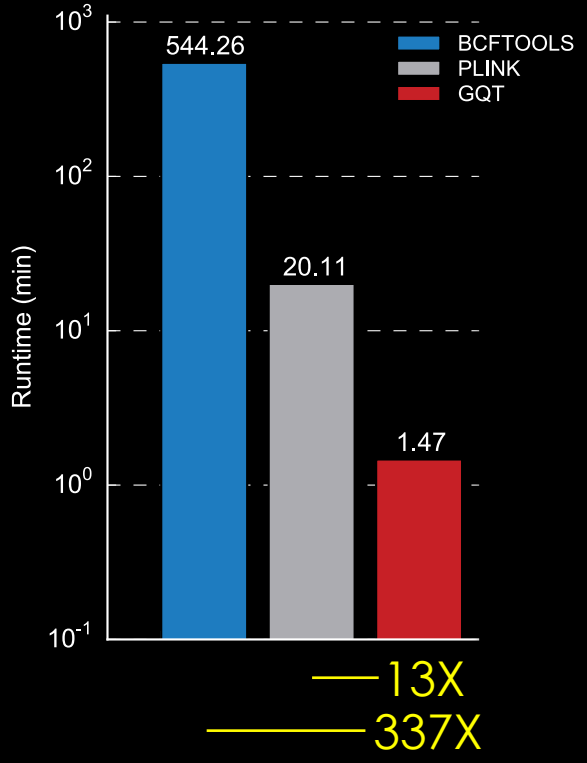
60K individual 9.3M variants

Linux v3.13, gcc v4.8.2, 4.0 GHz Intel Core i7 (Haswell)

Index size



Alt. Allele Count



Konrad Karczewski

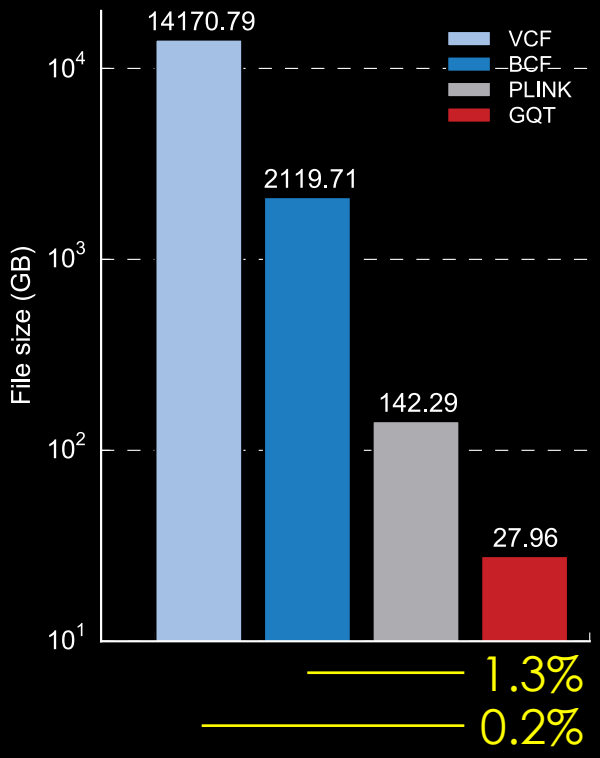
GQT v BCFTOOLS & PLINK v1.9

The Exome Aggregation Consortium (ExAC)

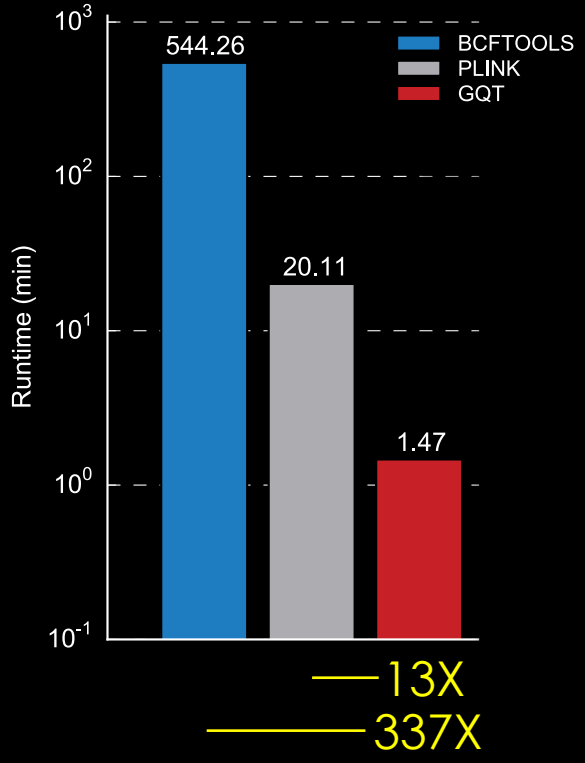
60K individual 9.3M variants

Linux v3.13, gcc v4.8.2, 4.0 GHz Intel Core i7 (Haswell)

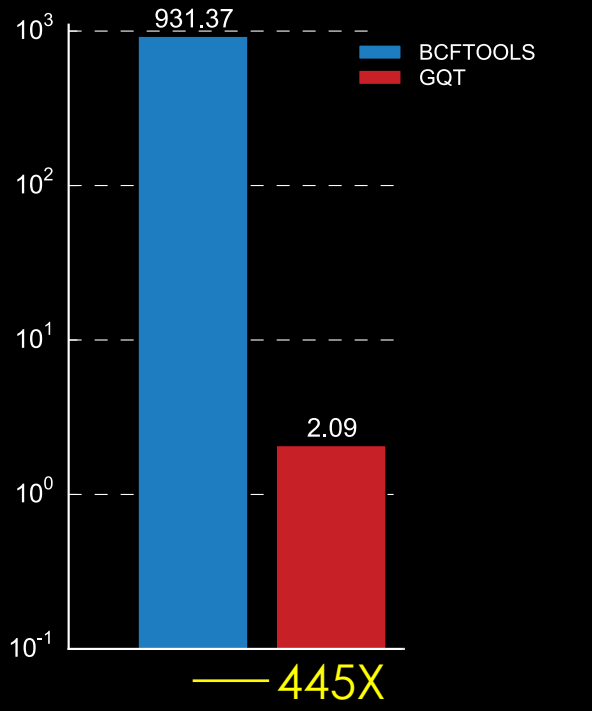
Index size



Alt. Allele Count

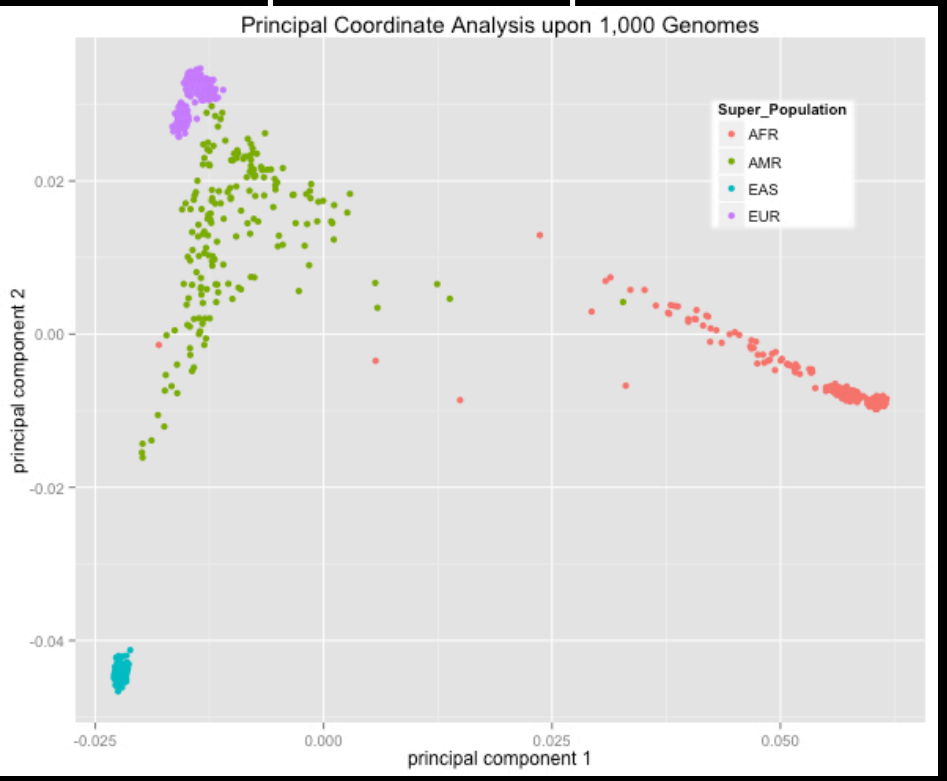


Rare Variant Search



Konrad Karczewski

Google Genomics & Apache Spark

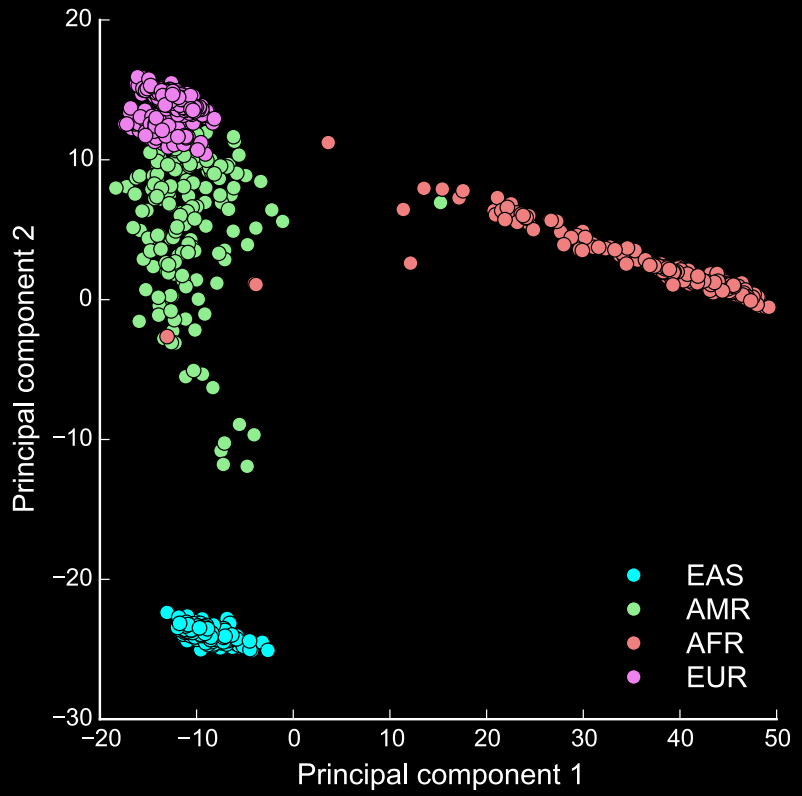


"... about 2 hours

on 60 eight core machines."[1]

2 x 60 x 8 = 960 compute hours

GQT



32 min 31 sec

on 1 core

0.542 compute hours

~1,800X speedup

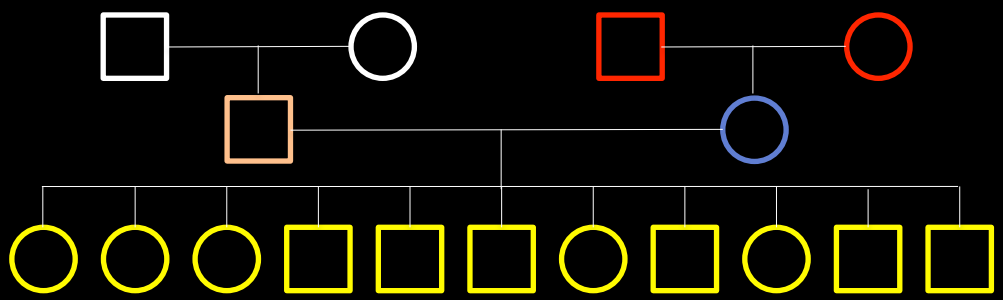
1. C. Curtin "ASHG: Data Science to Help Genomics Move from 'Artisanal' to 'Factory' " *genomeweb* Oct. 20, 2014


100,000,000 Variants
1,000,000 Individuals

Pedigree File (PED)

Family ID	Individual ID	Paternal ID	Maternal ID	Sex	Population	Phenotype
1463	NA12889	0	0	1	CEU	0
1463	NA12890	0	0	2	CEU	0
1463	NA12891	0	0	1	CEU	0
1463	NA12892	0	0	2	CEU	0
1463	NA12877	NA12889	NA12890	1	CEU	0
1463	NA12878	NA12891	NA12892	2	CEU	0
1463	NA12879	NA12877	NA12878	2	CEU	0
1463	NA12880	NA12877	NA12878	2	CEU	0
1463	NA12881	NA12877	NA12878	2	CEU	0
1463	NA12882	NA12877	NA12878	1	CEU	0
1463	NA12883	NA12877	NA12878	1	CEU	0
1463	NA12884	NA12877	NA12878	1	CEU	0
1463	NA12885	NA12877	NA12878	2	CEU	0
1463	NA12886	NA12877	NA12878	1	CEU	0
1463	NA12887	NA12877	NA12878	2	CEU	0
1463	NA12888	NA12877	NA12878	1	CEU	0
1463	NA12893	NA12877	NA12878	1	CEU	0

CEPH/Utah Pedigree 1463



Phenotype Query

Parents

```
"Individual_ID IN ('NA12878', 'NA12877')"
```

Female children

```
"Sex = 2 AND Maternal_ID = 'NA12878'"
```

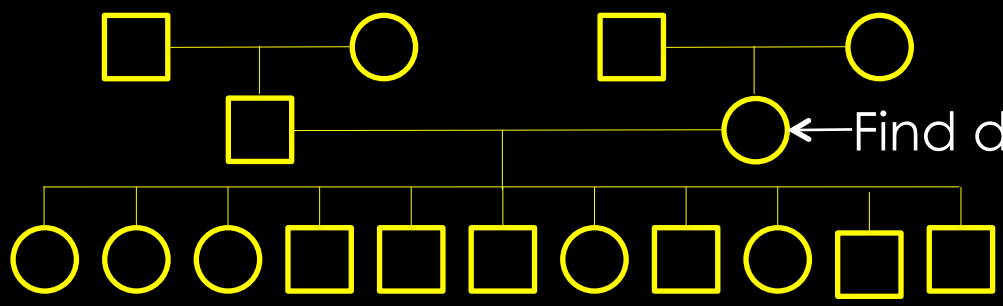
Affected across the CEU population

```
"Phenotype = 2 AND Population ='CEU'"
```

Subphenotypes

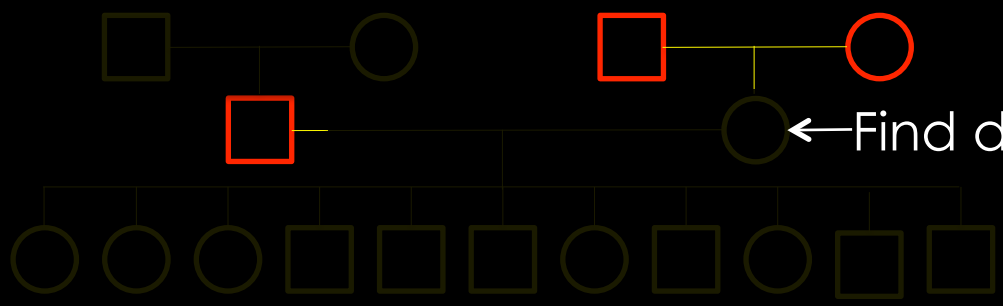
```
"LDL > 300"
"Maternal_infection = 1"
"Kidney_failture = 1"
```

Chaining Phenotype & Genotype Queries




```
gqt query -i ceph1463.gqt -d ceph1463.ped.db \
```

Chaining Phenotype & Genotype Queries

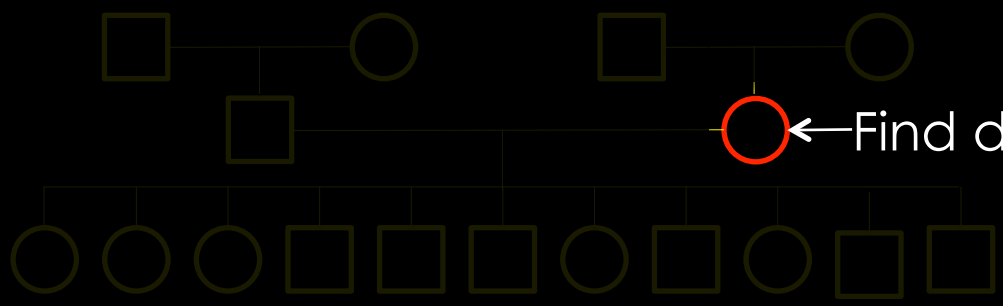



```
gqt query -i ceph1463.gqt -d ceph1463.ped.db \
  -p "sample_id in ('NA12891','NA12892','NA12877')" \
  -g "HOM_REF" \
```

Get all variants that are:

1. homozygous reference in the grandparents & father

Chaining Phenotype & Genotype Queries

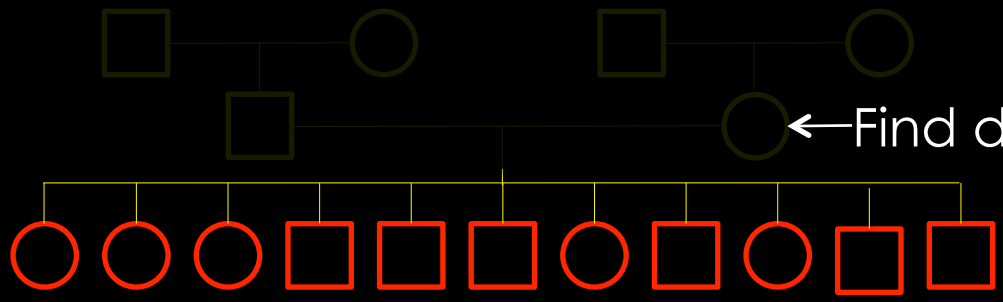



```
gqt query -i ceph1463.gqt -d ceph1463.ped.db \
  -p "sample_id in ('NA12891','NA12892','NA12877')" \
  -g "HOM_REF" \
  -p "sample_id = 'NA12878'" \
  -g "HET" \
```

Get all variants that are:

1. homozygous reference in the grandparents & father
2. heterozygous in the mother

Chaining Phenotype & Genotype Queries




```
gqt query -i ceph1463.gqt -d ceph1463.ped.db \
  -p "sample_id in ('NA12891','NA12892','NA12877')" \
  -g "HOM_REF" \
  -p "sample_id = 'NA12878'" \
  -g "HET" \
  -p "maternal_id = 'NA12878'" \
  -g "count(HET)>=1"
```

Get all variants that are:

1. homozygous reference in the grandparents & father
2. heterozygous in the mother
3. heterozygous in at least 1 of the children

GENOTYPE QUERY TOOLS

github.com/ryanlayer/gqt

Millions of Individuals

100 Millions of genotypes

Fast Queries

Powerful Interface

UCSC

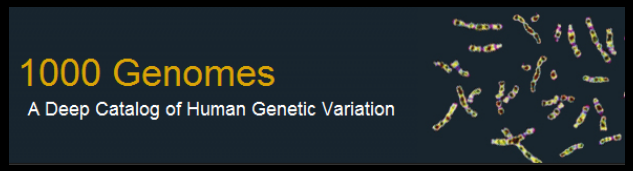
300M Intervals

GTEx

8.5K Samples

ENCODE

7K Samples



2.5K Genomes

ExAC

125K Exomes
15K Genomes

5K Phenotypes
3K Genes

FANTOM

FUNCTIONAL ANNOTATION OF THE MAMMALIAN GENOME

400 Tissues

ROADMAP epigenomics PROJECT

127 Tissues
55M Intervals

Genomics england

100K Genomes

TOPMed

100K Genomes

CCDG

200K Genomes

GFF

NARROWPEAK

VCF

BROADPEAK

BEDPE

CRAM

BED

SAM

BCF

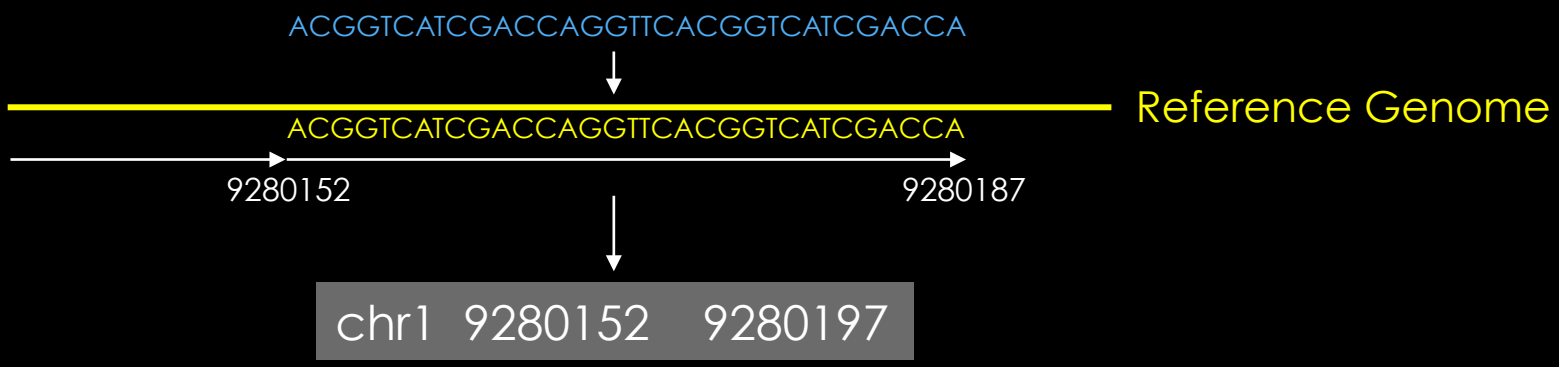
BAM

BIGWIG

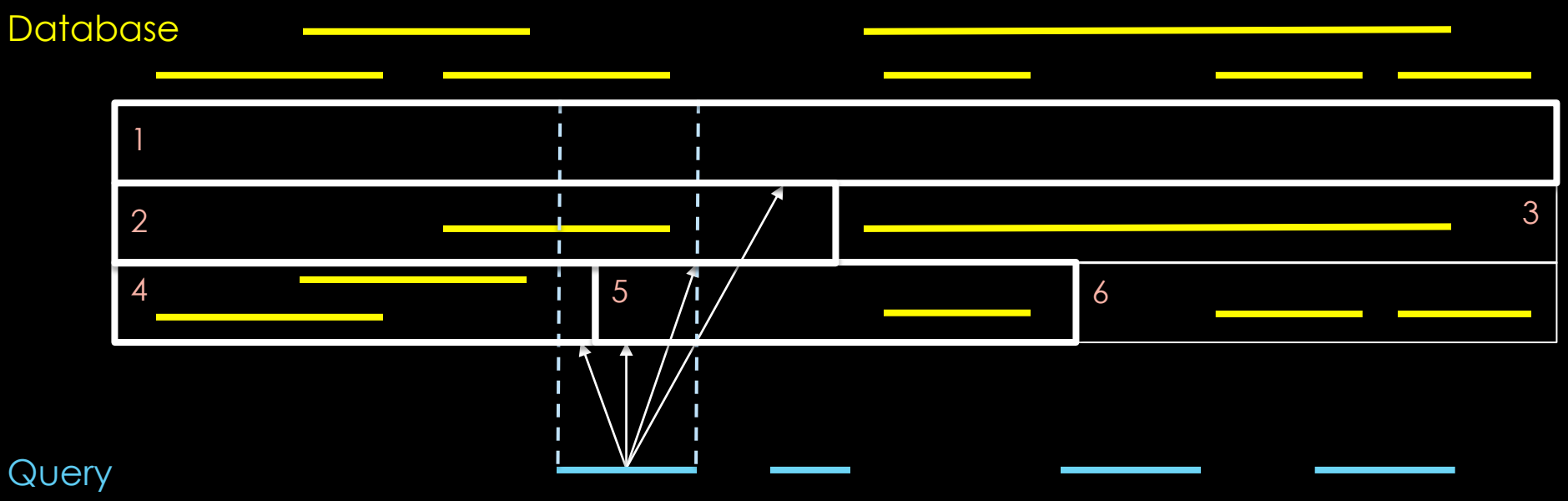
BEDGRAPH

WIG

BIGBED

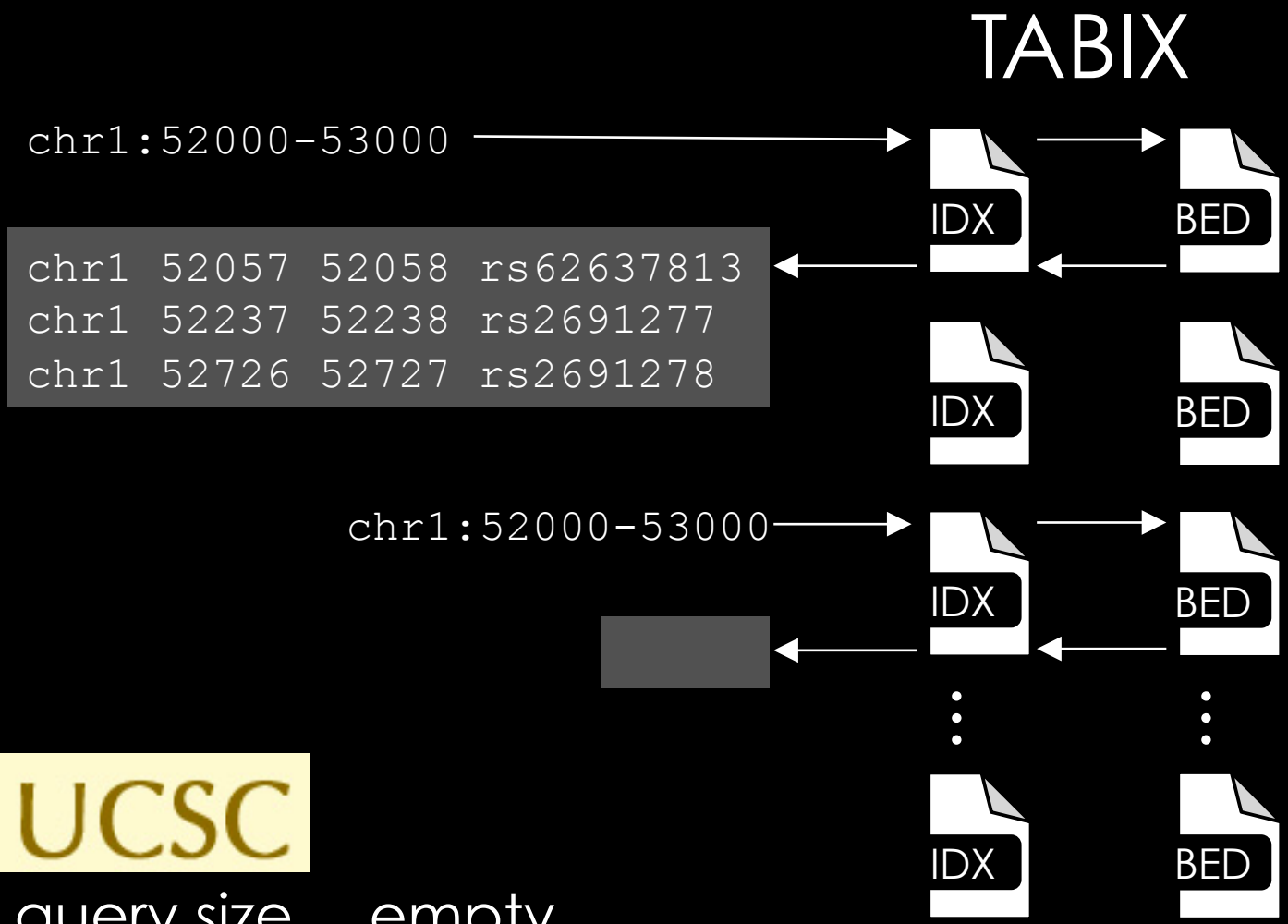


Hierarchical Binning [UCSC Kent 2002, TABIX Li 2011, BEDTOOLS Quinlan 2010]



Sweep [BEDTOOLS]

TABIX

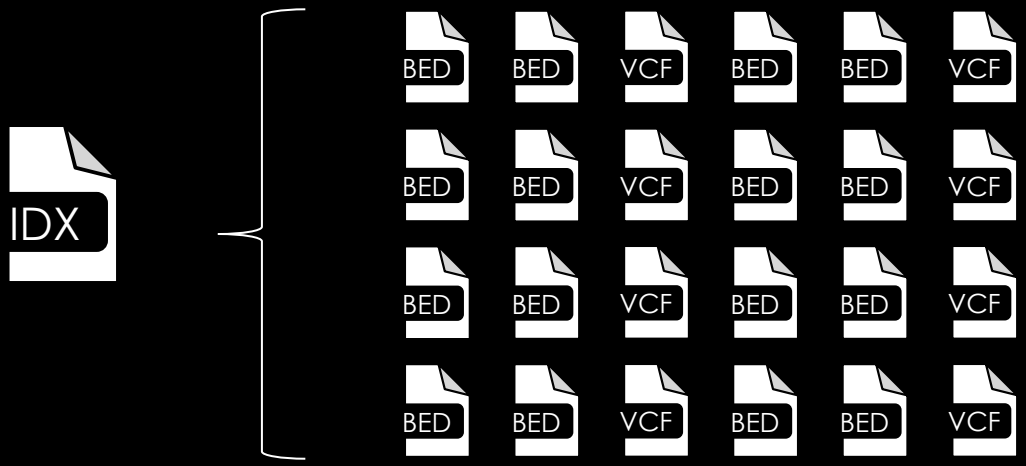


UCSC

query size	empty
100bp	0.73
1000bp	0.67
10000bp	0.58

GIGGLE

B+ Tree index of
positions and offsets



GIGGLE

Query

# of intersections

Operation

```
chr1:52000-53000
```


```
12039
```

Intersections per file

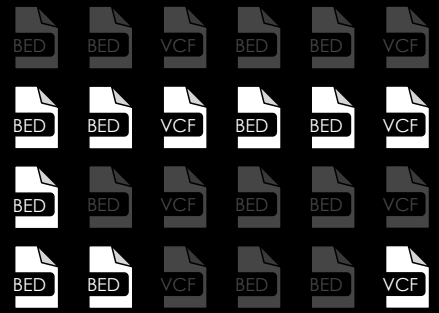
```
chr1:52000-53000
```


```
snp144                  13
neandertalMethylation   5
genomicSuperDups        2
```

Intersecting intervals

```
chr1:52000-53000
```


```
#snp144
chr1 52057 52058 rs62637
chr1 52095 52096 rs36775
#neandertalMethylation
chr1 52015 52016 100
chr1 52028 52029 100
```



THE TIME INDEX:
AN ACCESS STRUCTURE FOR TEMPORAL DATA

Ramez Elmasri[1,*], Gene T. J. Wuu[2], and Yeong-Joon Kim[1]

[1]Department of Computer Science, University of Houston, Houston, TX 77204
[2]Bell Communications Research, 444 Hoes Lane, Piscataway, NJ 08854

ABSTRACT

In this paper, we describe a new indexing technique, the *time index*, for improving the performance of certain classes of temporal queries. The time index can be used to retrieve versions of objects that are valid during a specific time period. It supports the processing of the temporal WHEN operator and temporal aggregate functions efficiently. The time indexing scheme is also extended to improve the performance of the temporal SELECT operator, which retrieves objects that satisfy a certain condition during a specific time period. We will describe the indexing technique, and its search and insertion algorithms. We also describe an algorithm for processing a commonly used temporal JOIN operation. Some results of a simulation for comparing the performance of the time index with other proposed temporal access structures are presented.

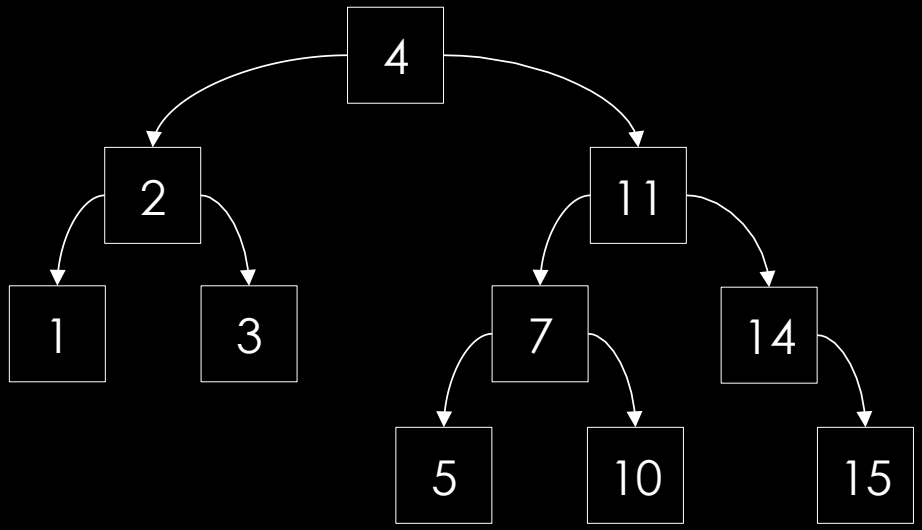
languages [SS87, EW90]. These temporal data models define powerful operations for specifying complex temporal queries. There has been relatively less research in the area of defining efficient storage structures and access paths for temporal data [Lum84, Ahn86, AS88, SG89, GSsu, RS87, KS89, LS89]. These proposals do not discuss indexing schemes for supporting the high-level temporal operators defined in [GY88, EW90]. This paper describes indexing techniques for improving the efficiency of temporal operations, such as *when*, *select*, and *join* [GY88], *temporal selection* and *temporal projection* [EW90], and aggregation functions.

The storage techniques for temporal data proposed in [AS88, Lum84] index or link the versions of each individual object separately. In order to retrieve object versions that are valid during a certain time period, it is necessary to first locate the first (current) version of each object, and then search through the version in-

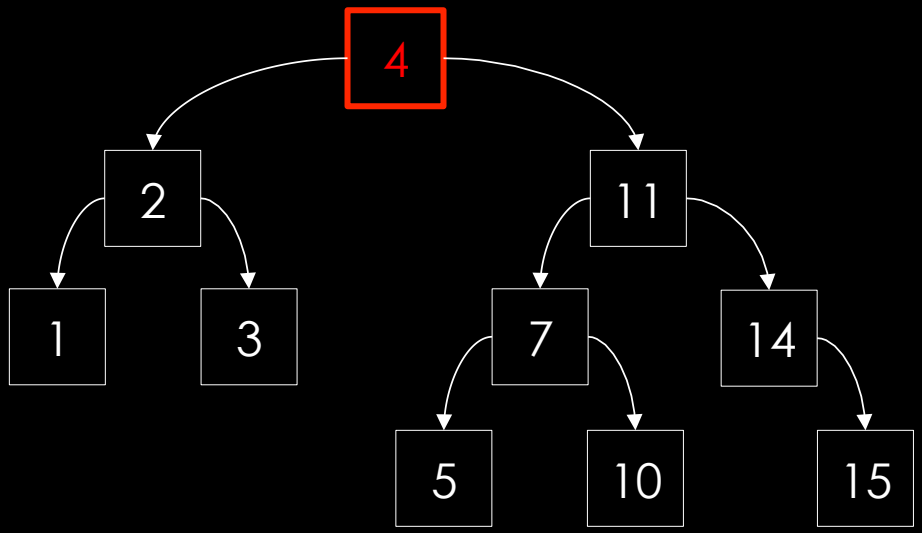
VLDB 1990

B+ Tree: efficient on-disk retrieval
efficient in-order traversal

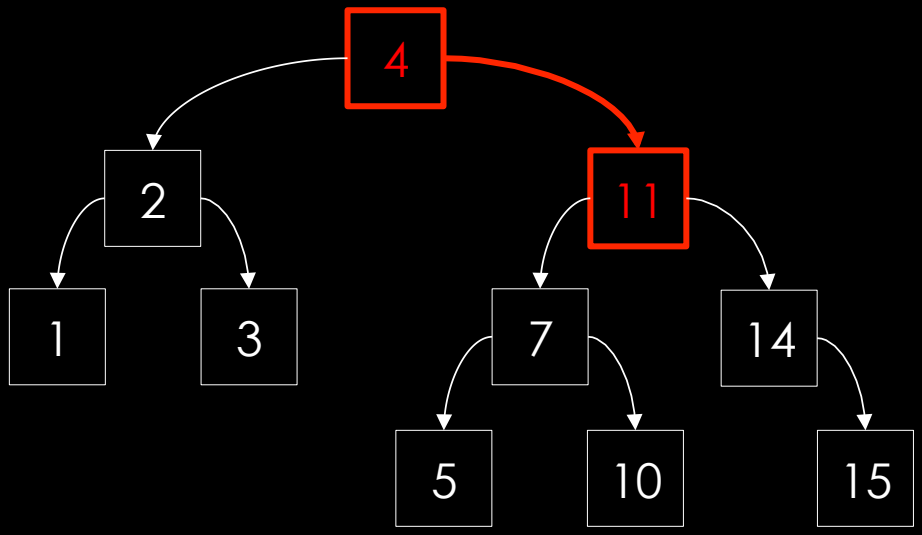
Binary Tree



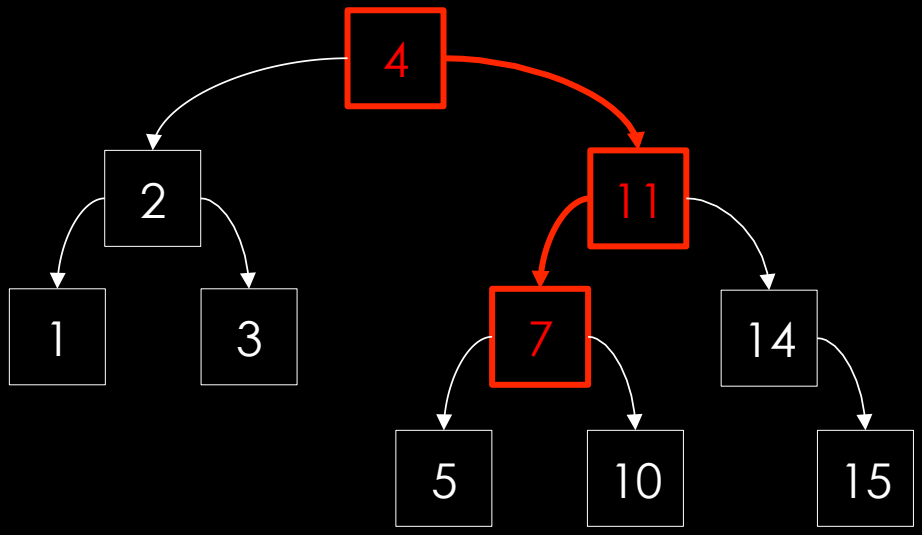
Binary Tree



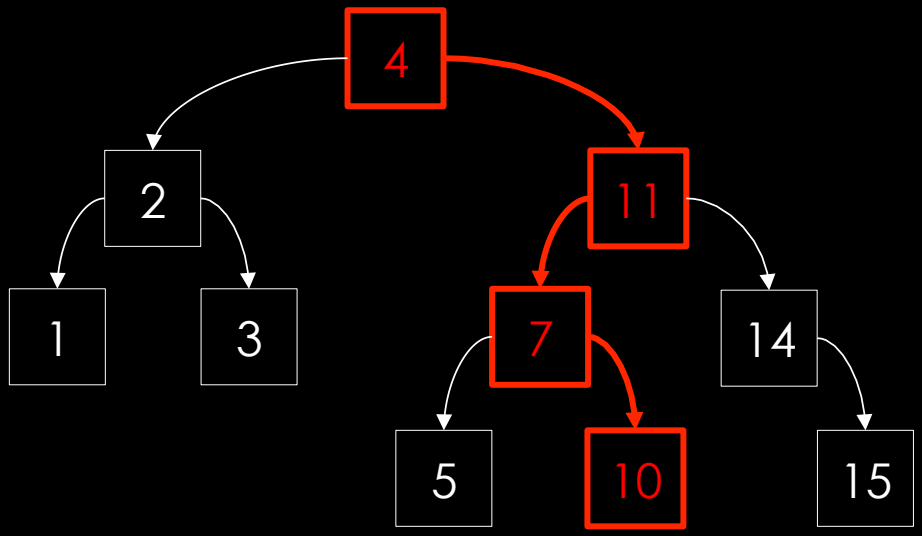
Binary Tree

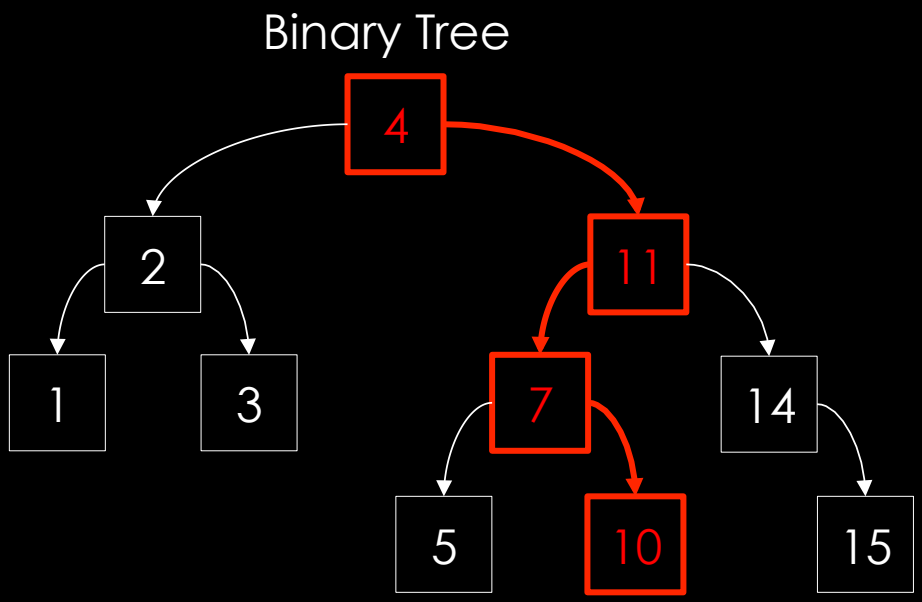


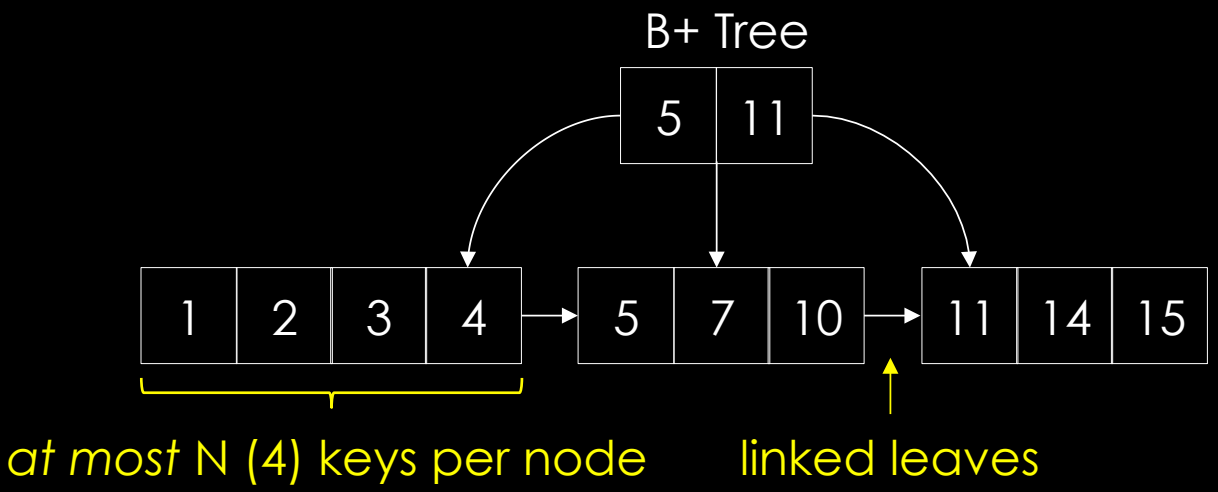
Binary Tree

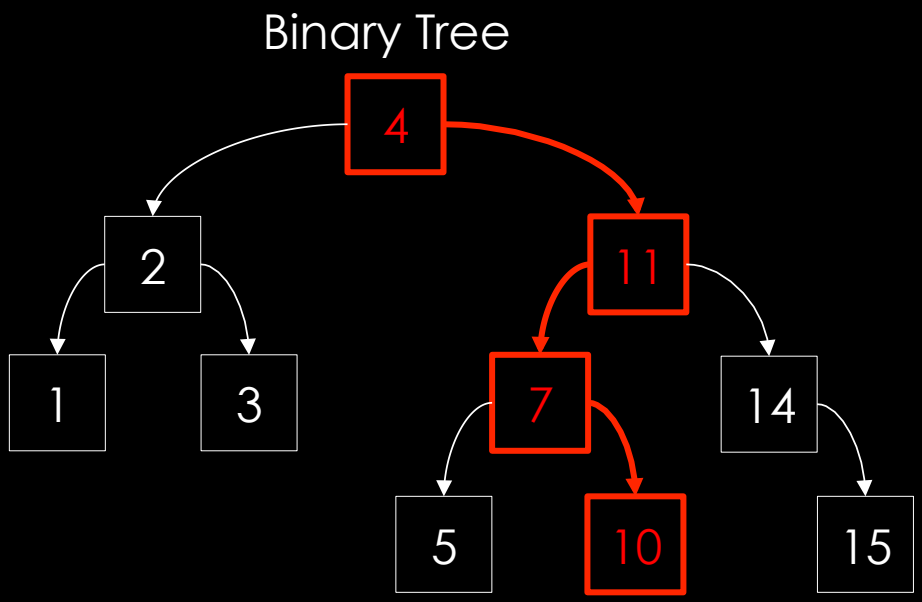


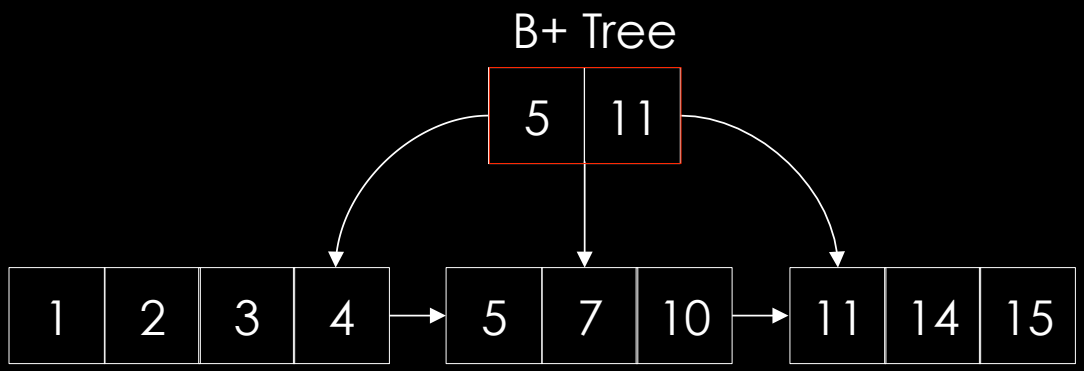
Binary Tree

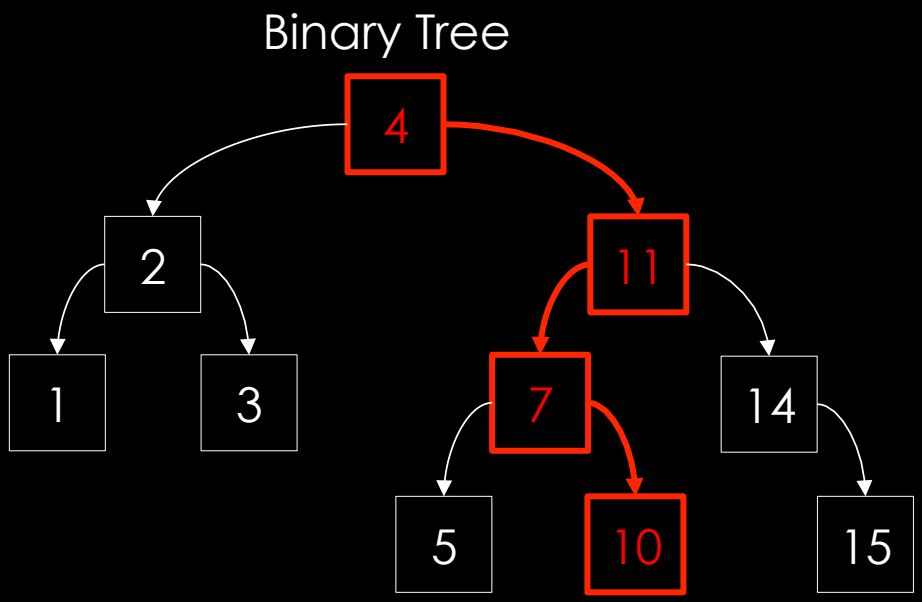


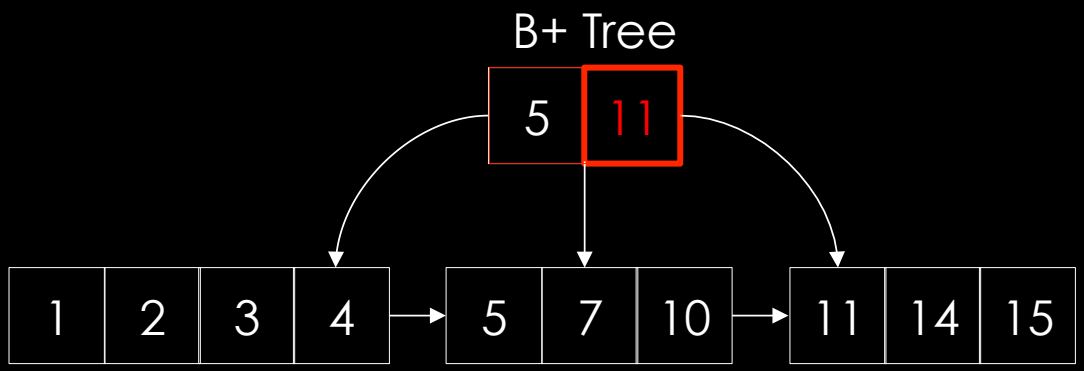

Binary Tree
4
2
11
1
3
7
14
5
10
15

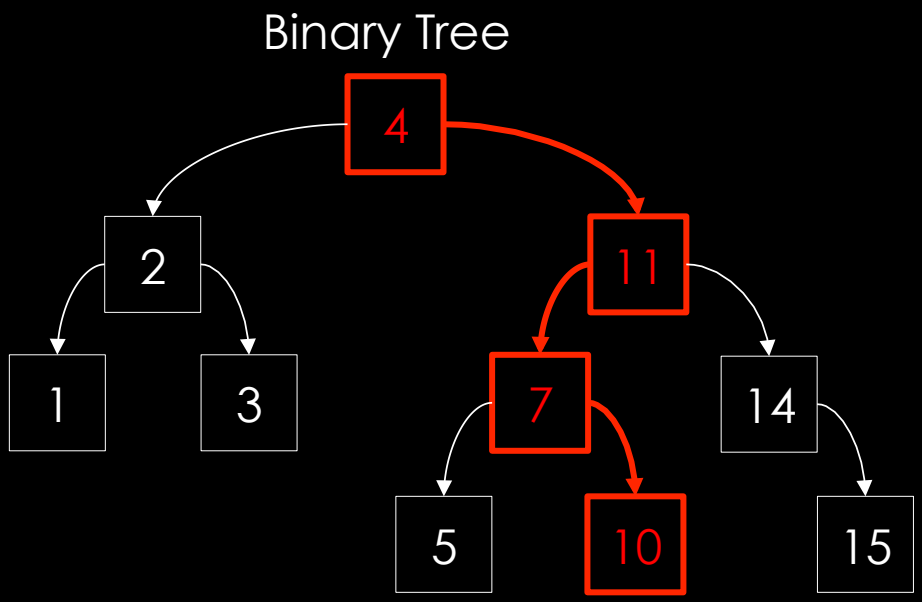

B+ Tree
5
11
1
2
3
4
5
7
10
11
14
15
at most N (4) keys per node
linked leaves

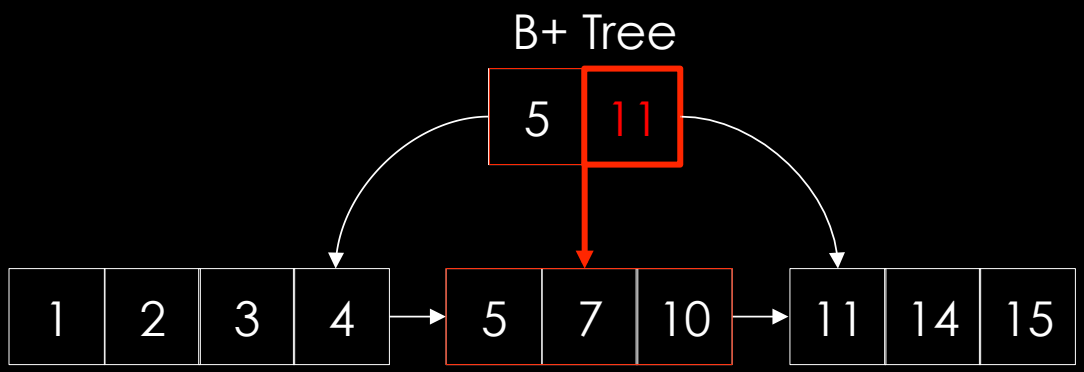

Binary Tree
4
2
11
1
3
7
14
5
10
15

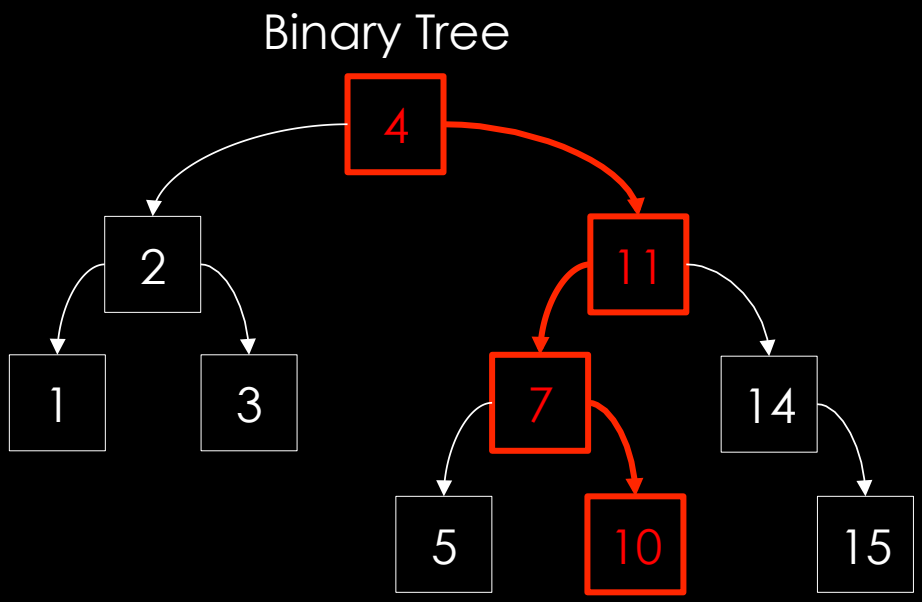

B+ Tree
5
11
1
2
3
4
5
7
10
11
14
15

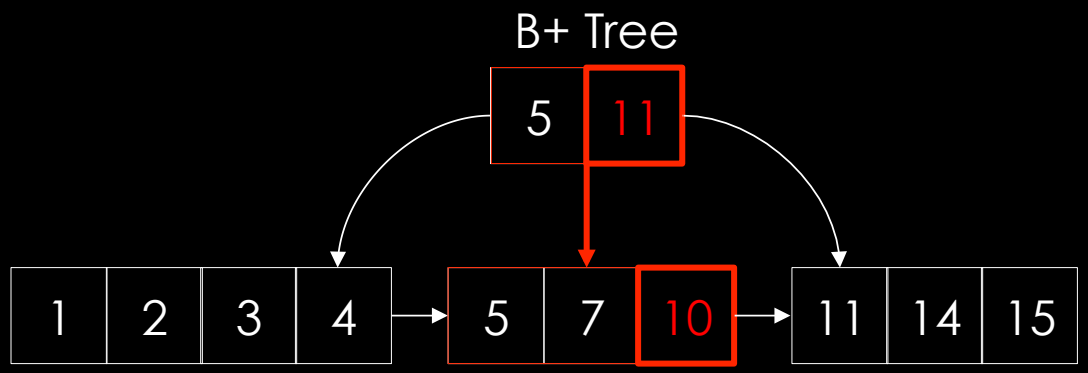

Binary Tree
4
2
11
1
3
7
14
5
10
15

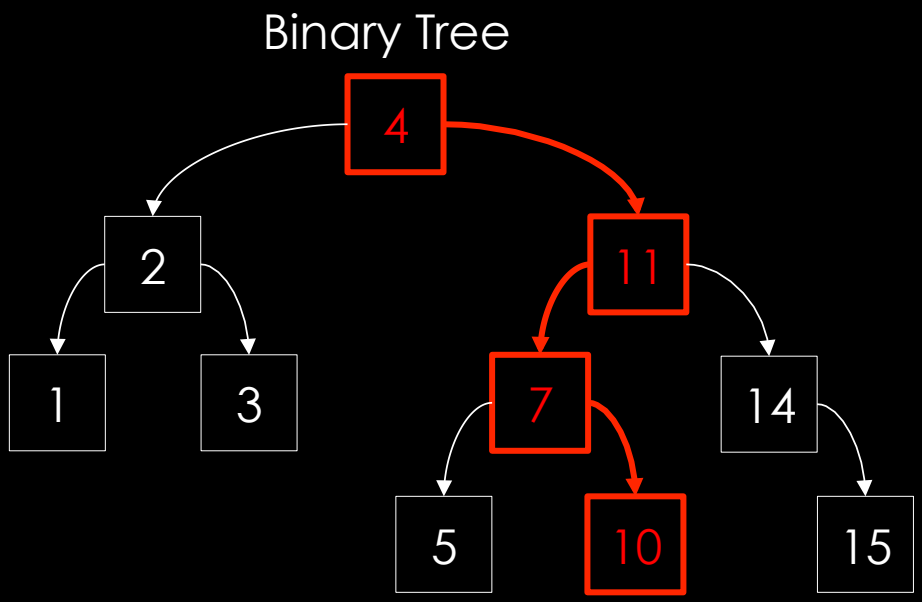

B+ Tree
5
11
1
2
3
4
5
7
10
11
14
15


Binary Tree
4
2
11
1
3
7
14
5
10
15

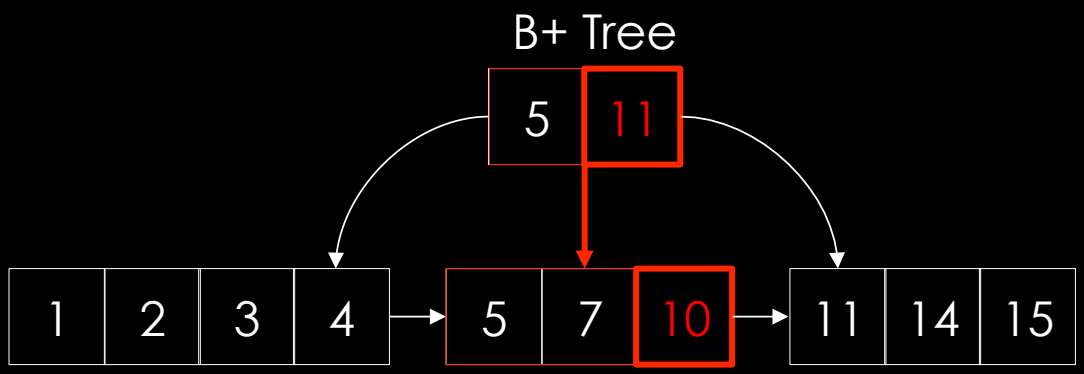

B+ Tree
5
11
1
2
3
4
5
7
10
11
14
15


Binary Tree
4
2
11
1
3
7
14
5
10
15

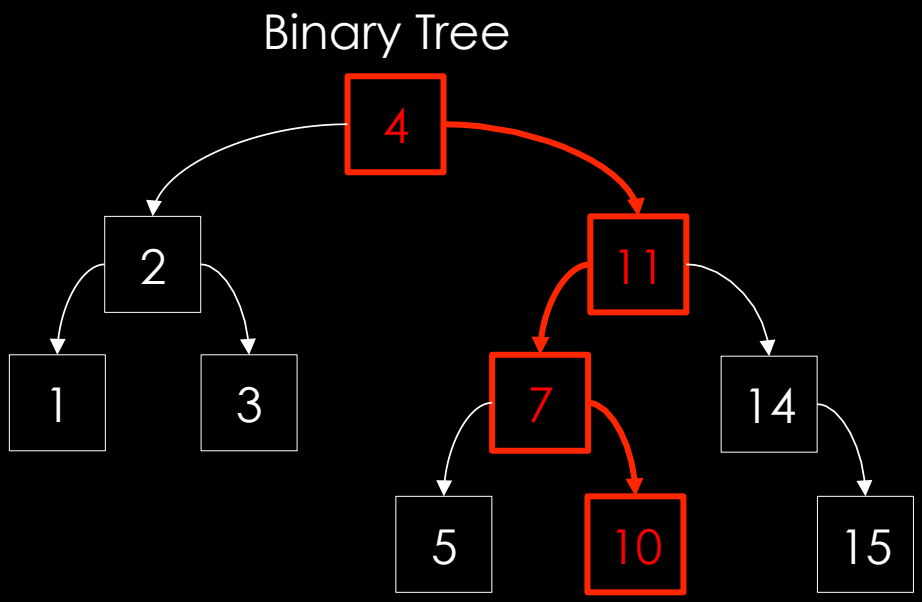

B+ Tree
5
11
1
2
3
4
5
7
10
11
14
15

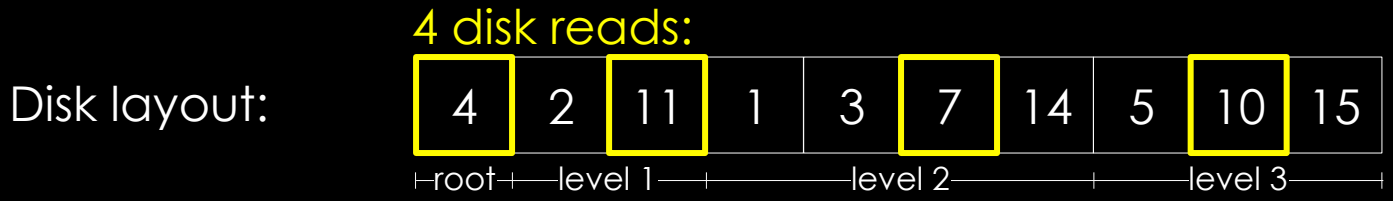


Disk layout:



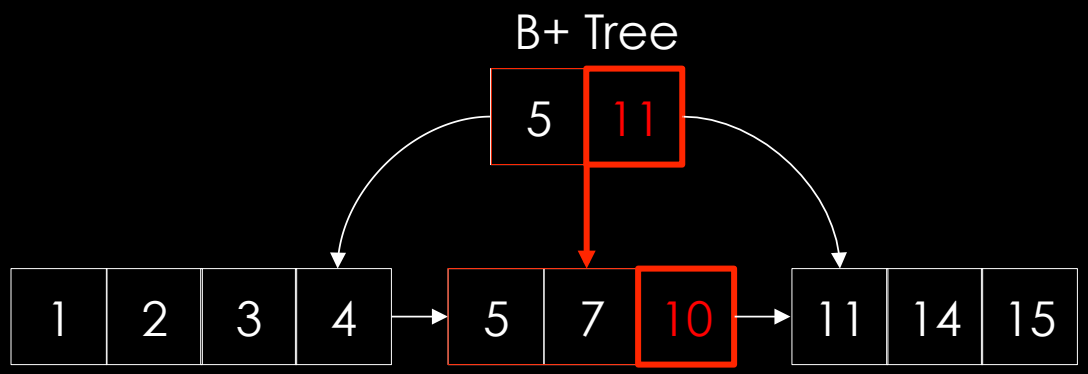
Disk layout:

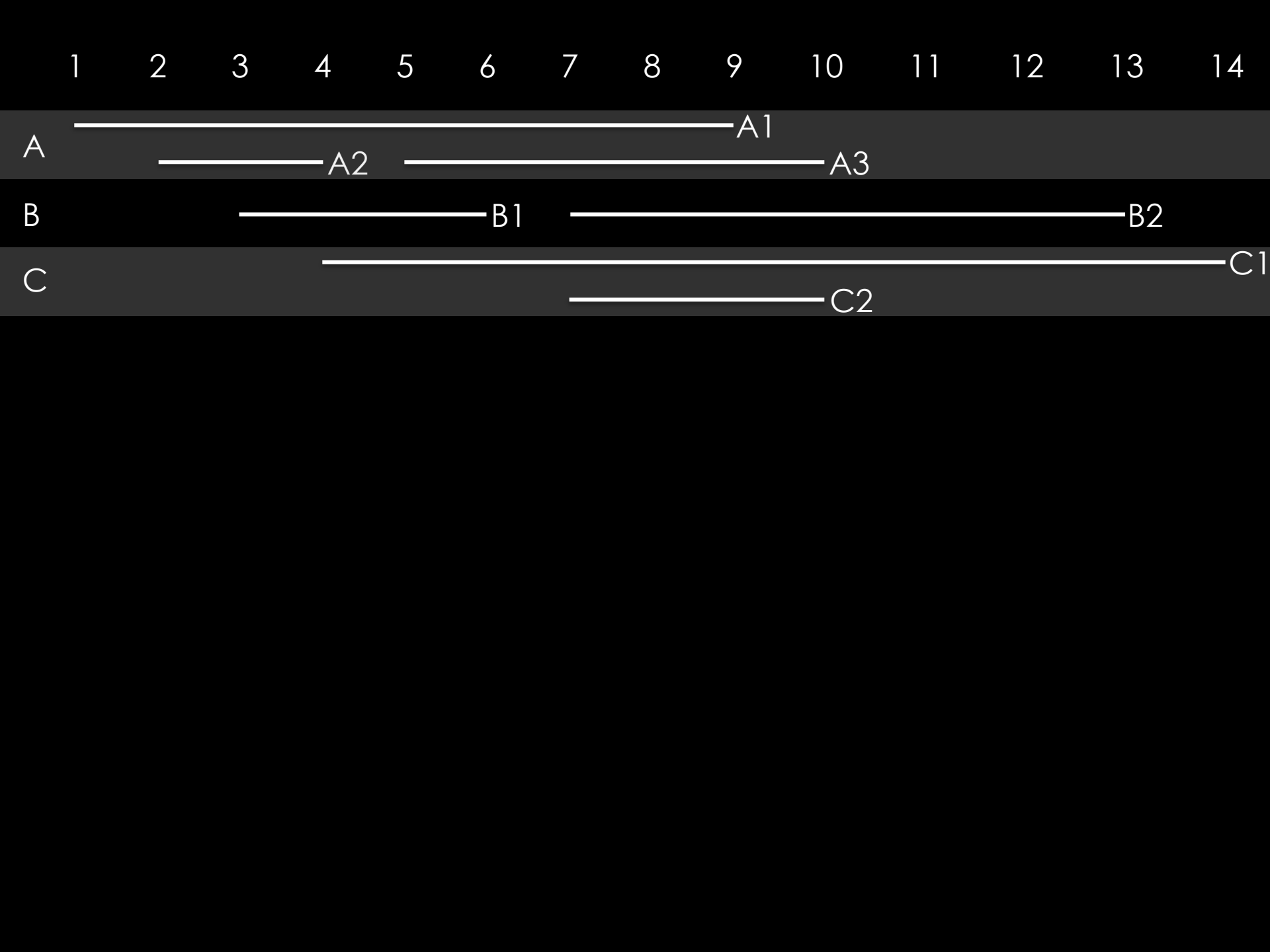


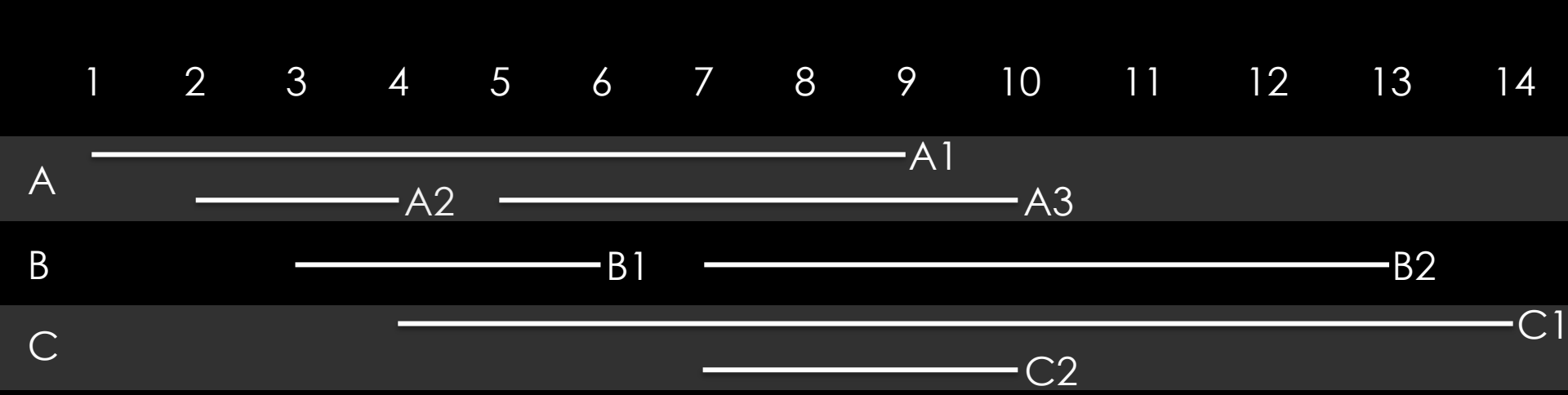


SSD disk speed
100,000 IOPS

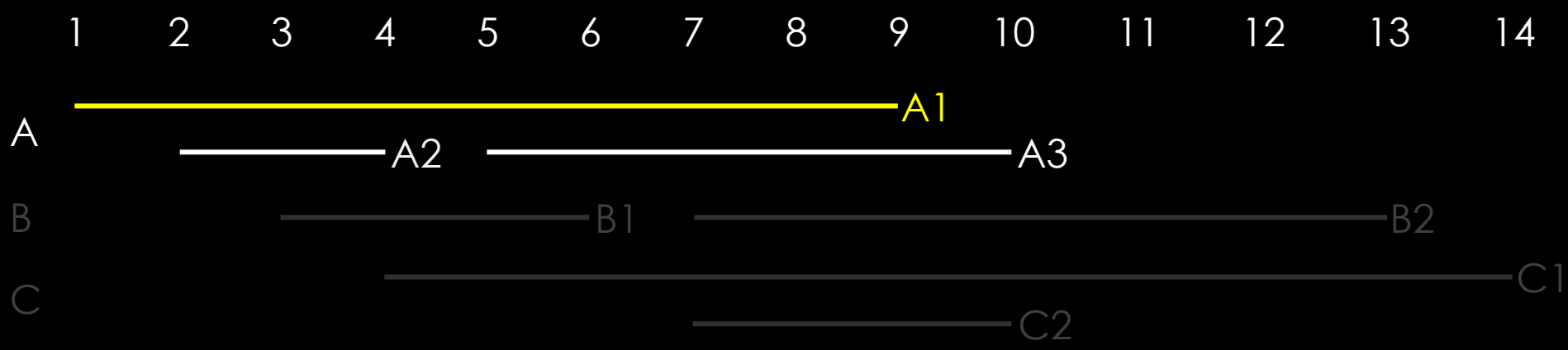
~2,000,000X
slower than CPU



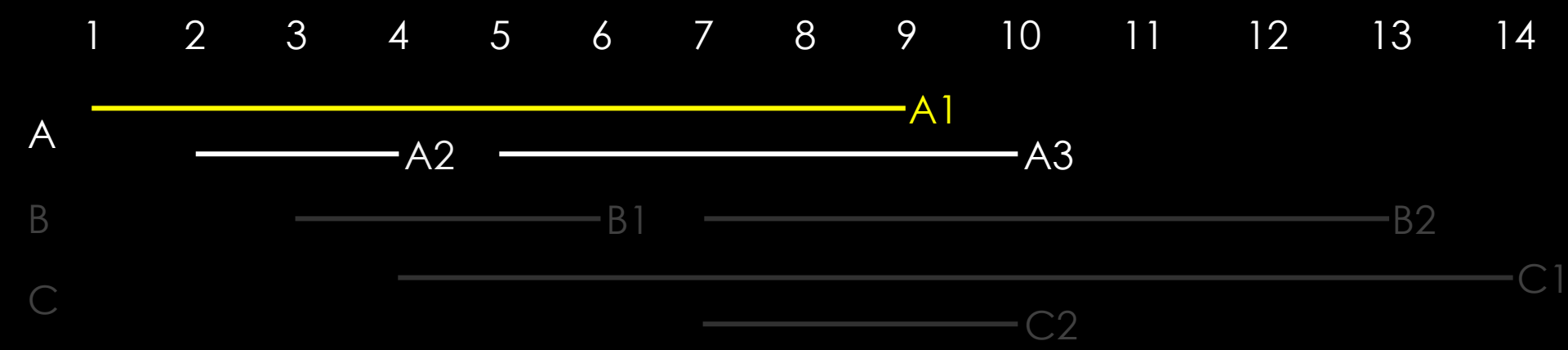

1
2
3
4
5
6
7
8
9
10
11
12
13
14
A
A1
A2
A3
B
B1
B2
C
C1
C2



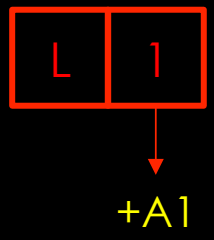

```
index(X, start, end)
 insert +X at start
 insert -X at end + 1
 append X at spanned L's
```



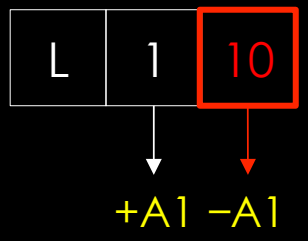

```
index(A1, 1, 9)
MAX_KEYS = 4
```

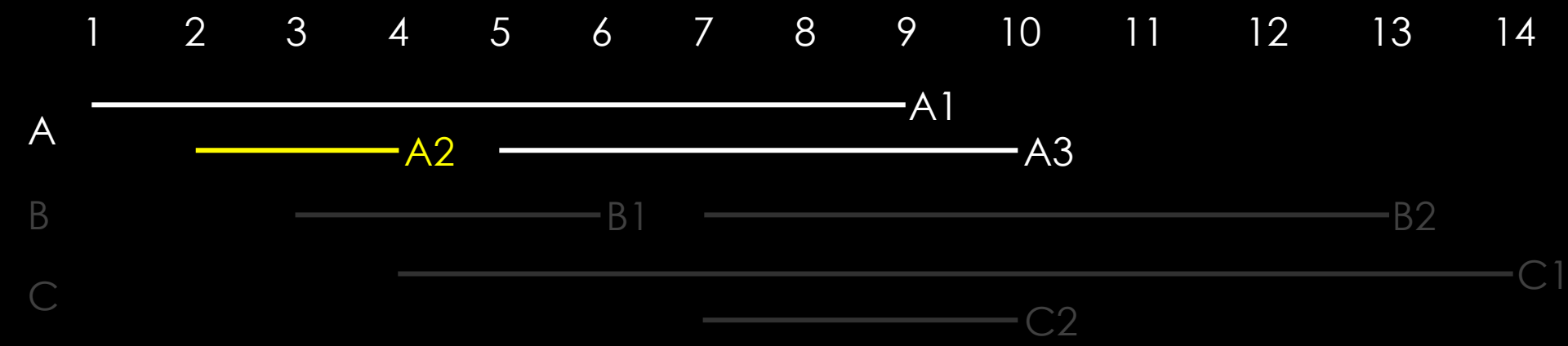



```
index(A1, 1, 9)
```

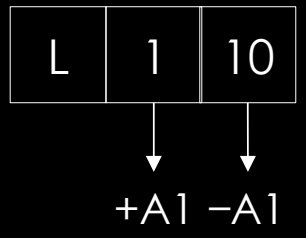


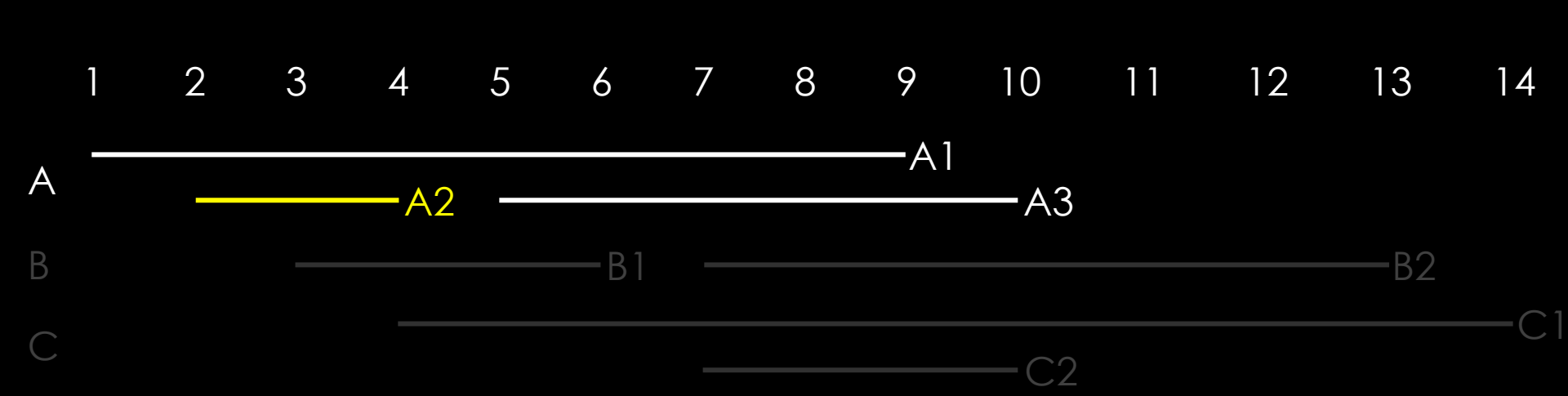

```
index(A1, 1, 9)
```



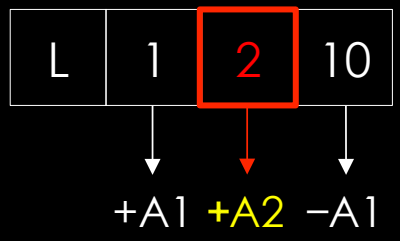


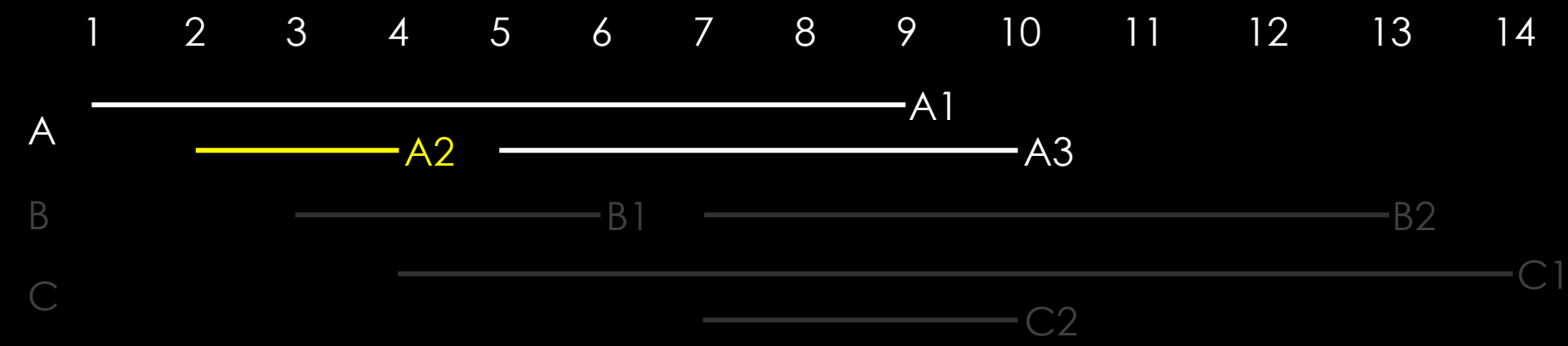

```
index(A2, 2, 4)
```



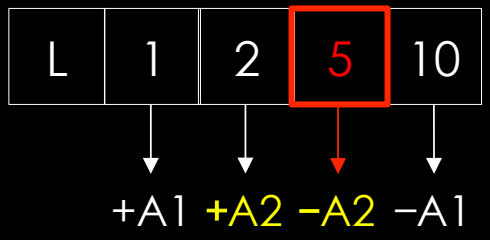


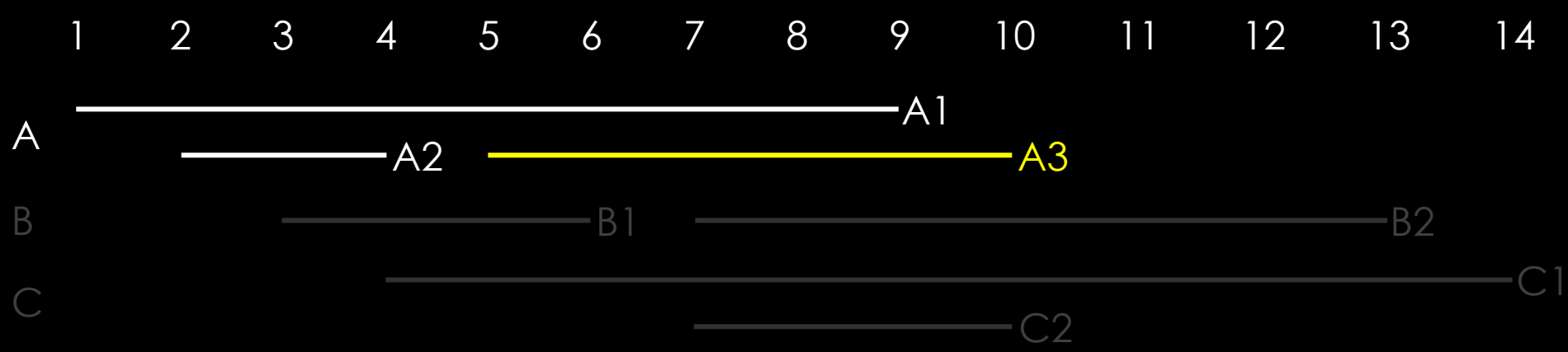

```
index(A2, 2, 4)
```



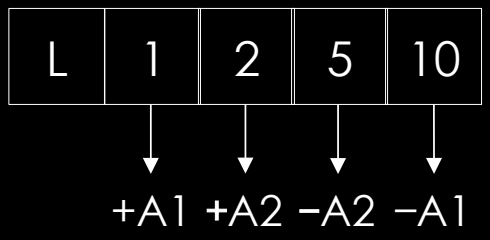

1 2 3 4 5 6 7 8 9 10 11 12 13 14
A
A1
A2
A3
B
B1
B2
C
C1
C2

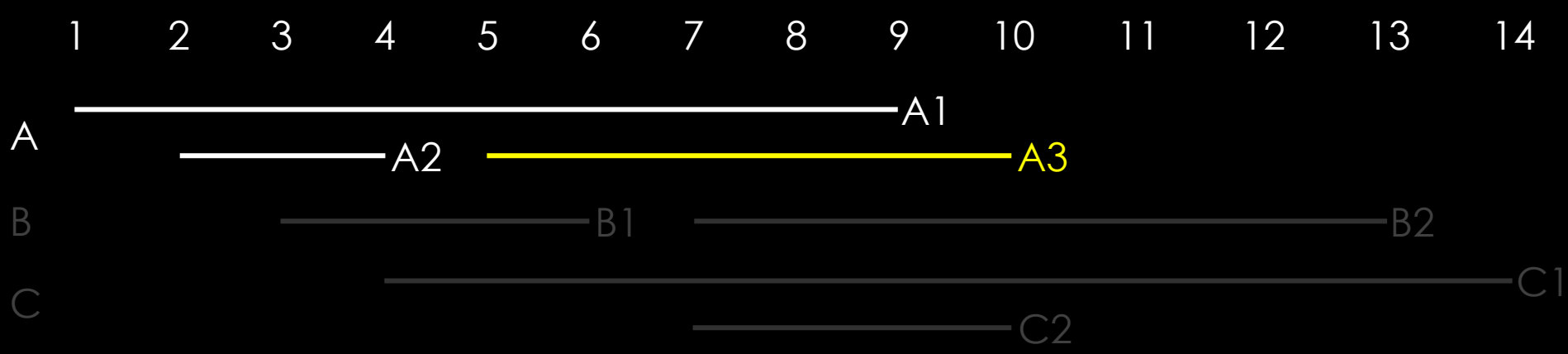
`index(A2, 2, 4)`


L
1
2
5
10
+A1 +A2 −A2 −A1

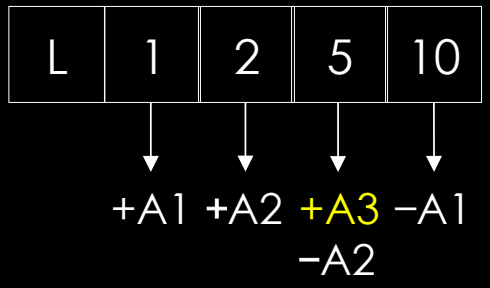

1 2 3 4 5 6 7 8 9 10 11 12 13 14
A
A1
A2
A3
B
B1
B2
C
C1
C2

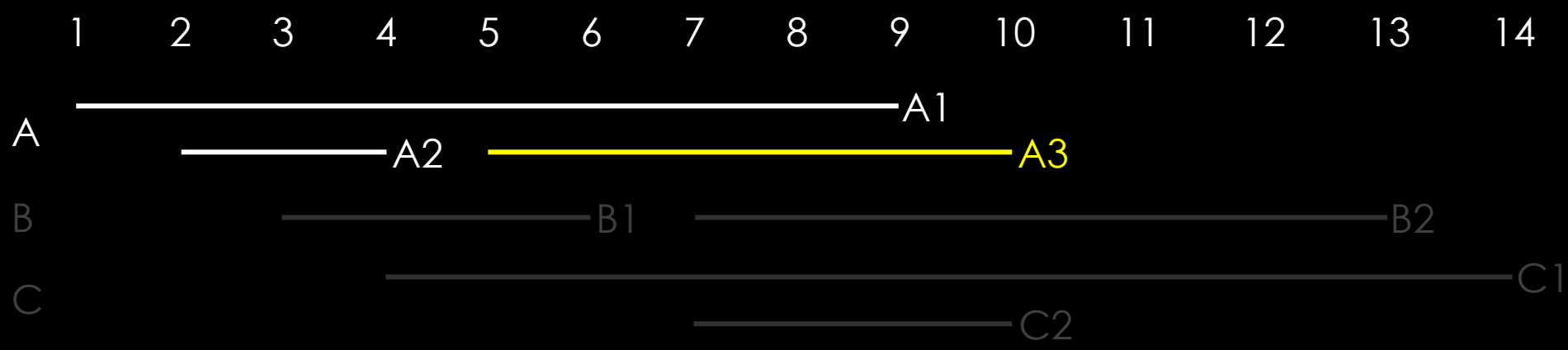
```
index(A3, 5, 10)
```


L
1
2
5
10
+A1 +A2 −A2 −A1

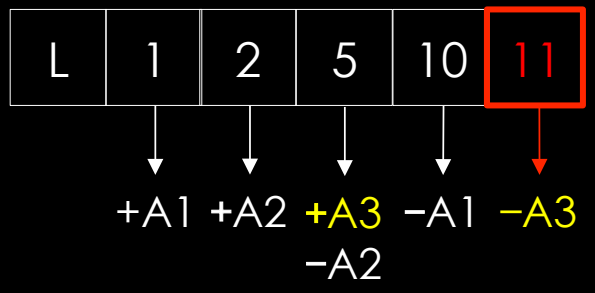

1 2 3 4 5 6 7 8 9 10 11 12 13 14
A
A1
A2
A3
B
B1
B2
C
C1
C2

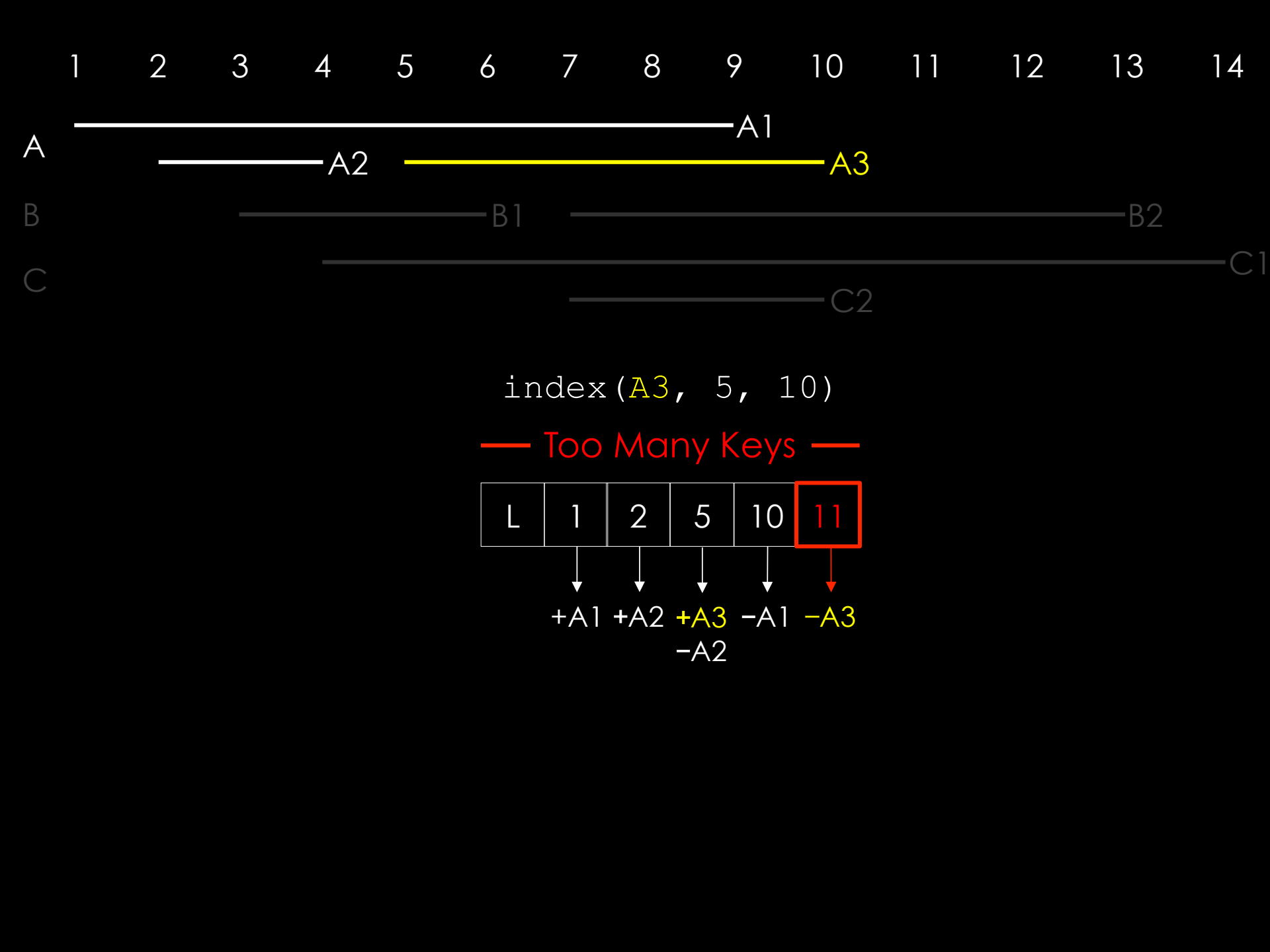
```
index(A3, 5, 10)
```

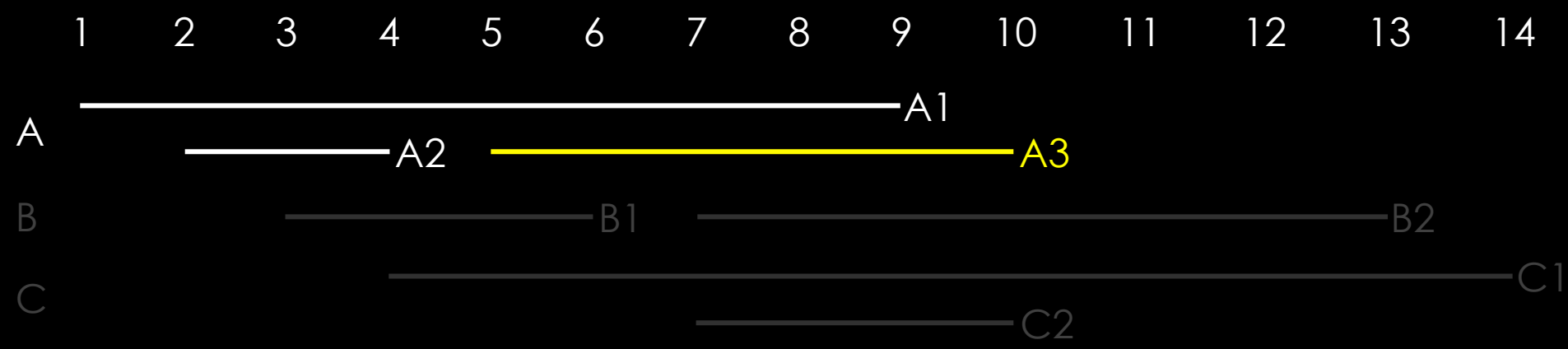

L
1
2
5
10
+A1
+A2
+A3
−A1
−A2


1 2 3 4 5 6 7 8 9 10 11 12 13 14
A
A1
A2
A3
B
B1
B2
C
C1
C2

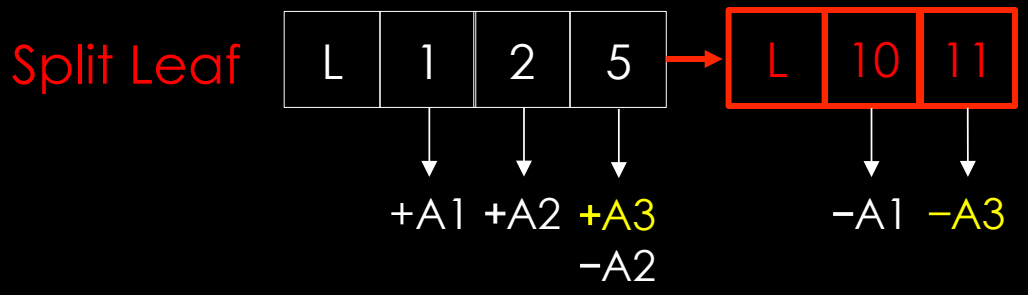
```
index(A3, 5, 10)
```

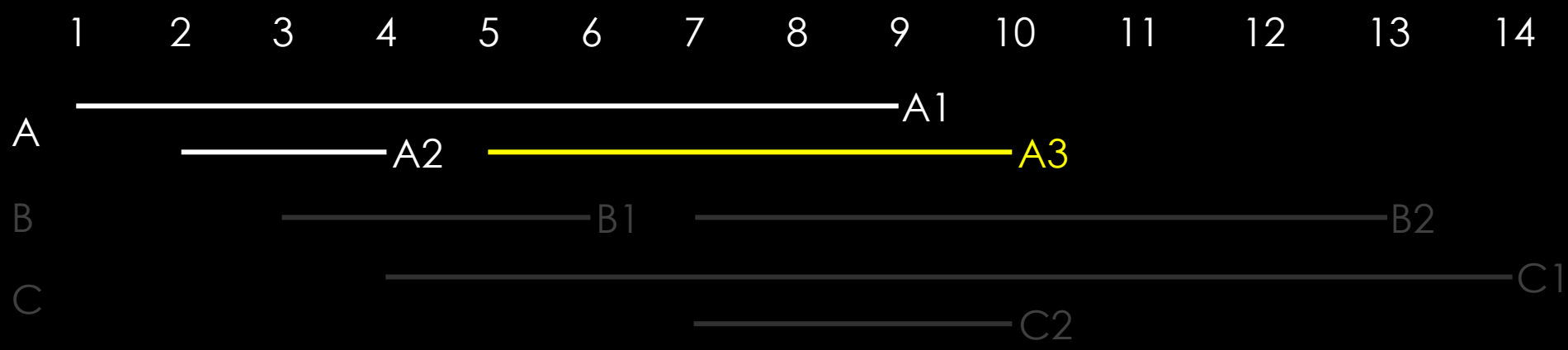

L
1
2
5
10
11
+A1 +A2 +A3 −A1 −A3
−A2


1 2 3 4 5 6 7 8 9 10 11 12 13 14
A
A1
A2
A3
B
B1
B2
C
C1
C2
index(A3, 5, 10)
Too Many Keys
L
1
2
5
10
11
+A1 +A2 +A3 −A1 −A3
−A2


1 2 3 4 5 6 7 8 9 10 11 12 13 14
A
A1
A2
A3
B
B1
B2
C
C1
C2

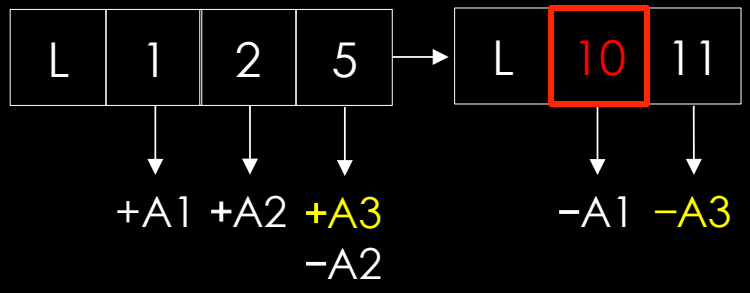
```
index(A3, 5, 10)
```

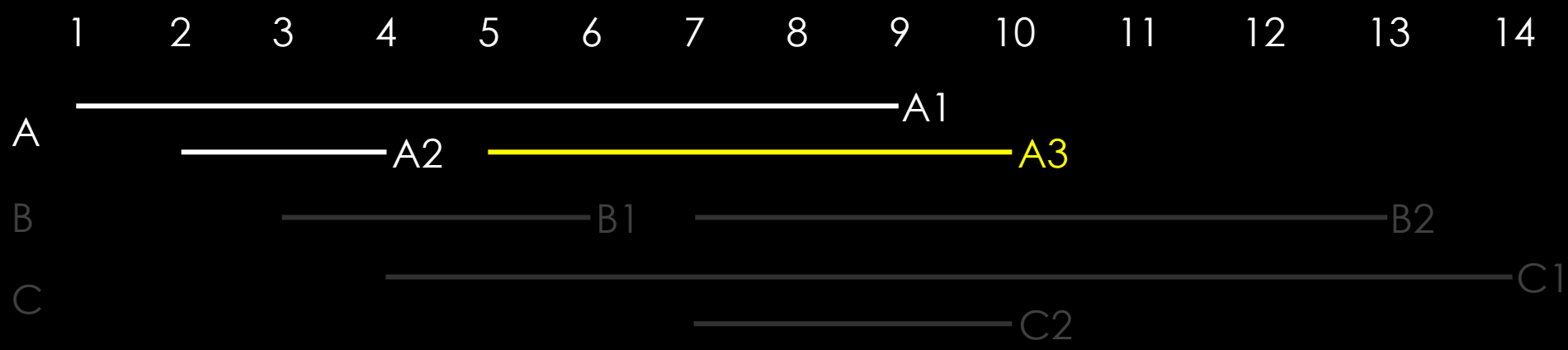

Split Leaf
L
1
2
5
L
10
11
+A1 +A2 +A3
−A2
−A1 −A3




```
index(A3, 5, 10)
```

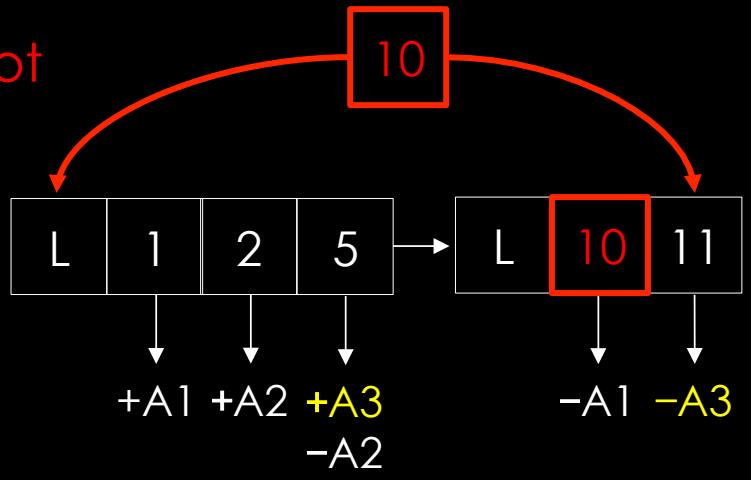
Promote a New Root

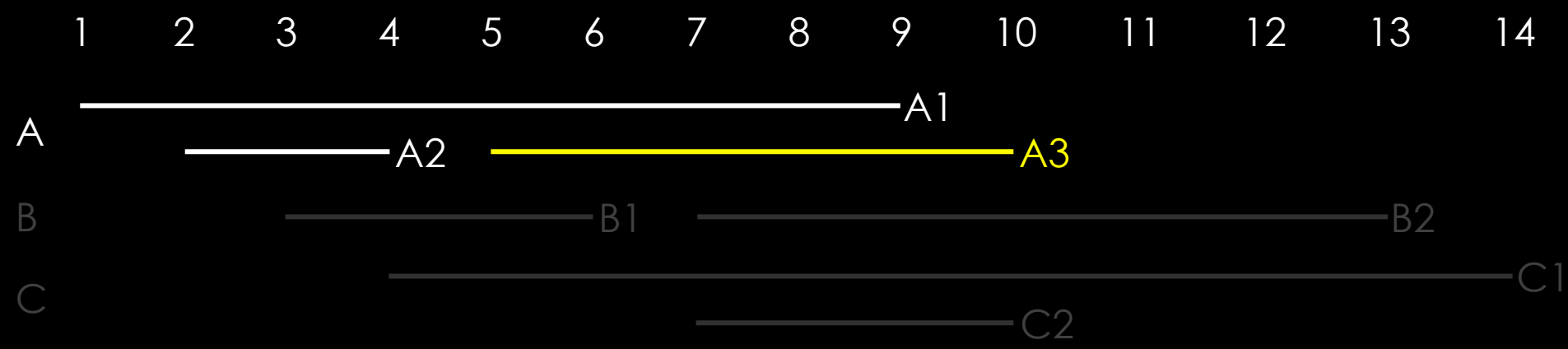





```
index(A3, 3, 6)
```

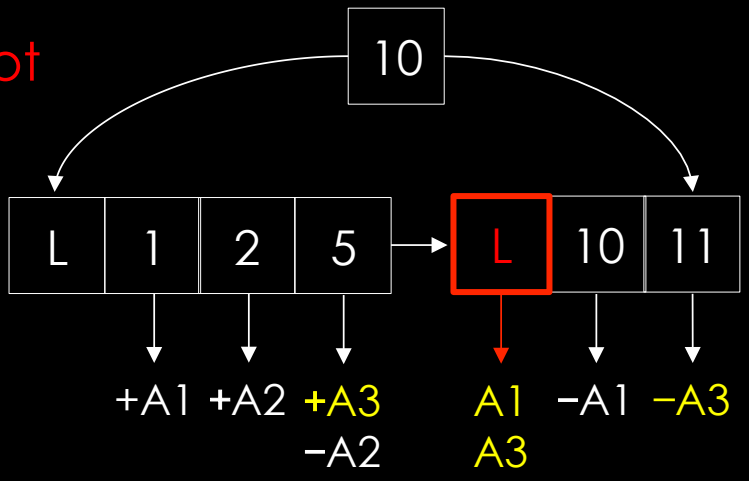
Promote a New Root

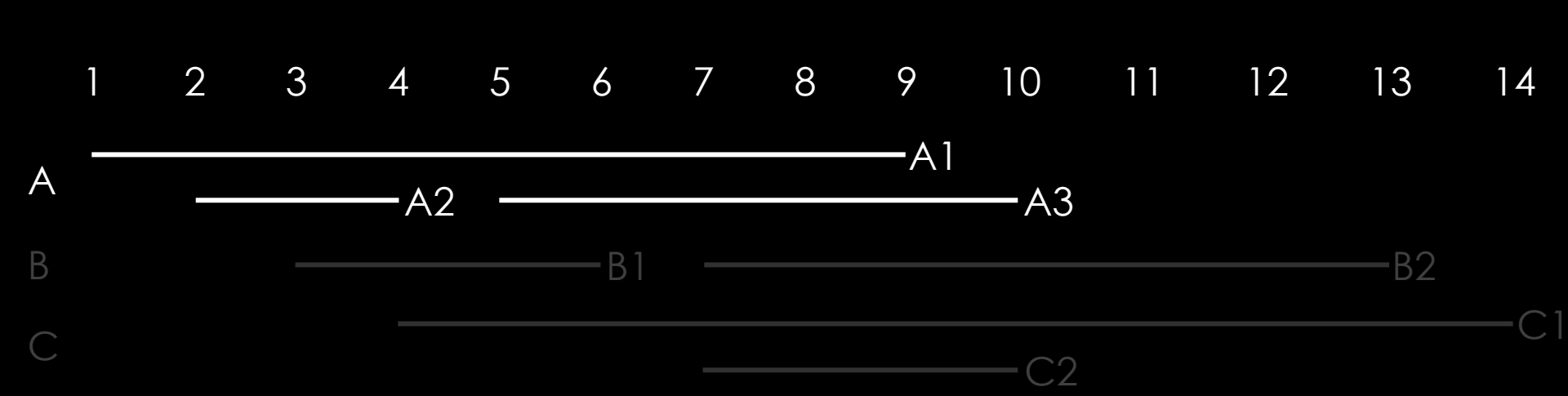


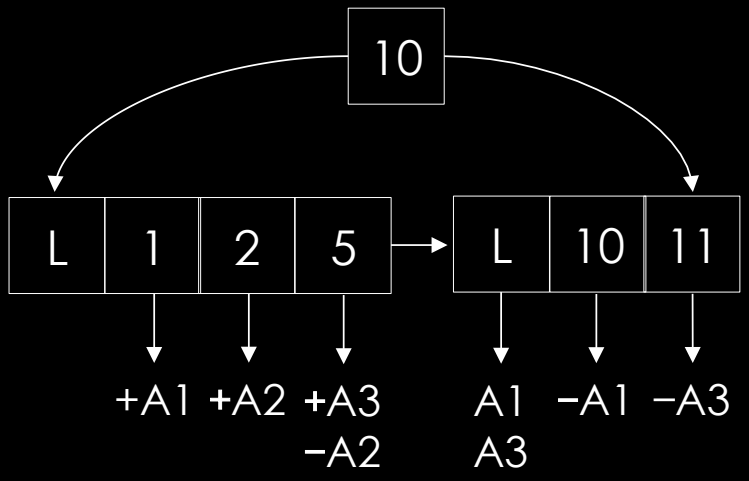


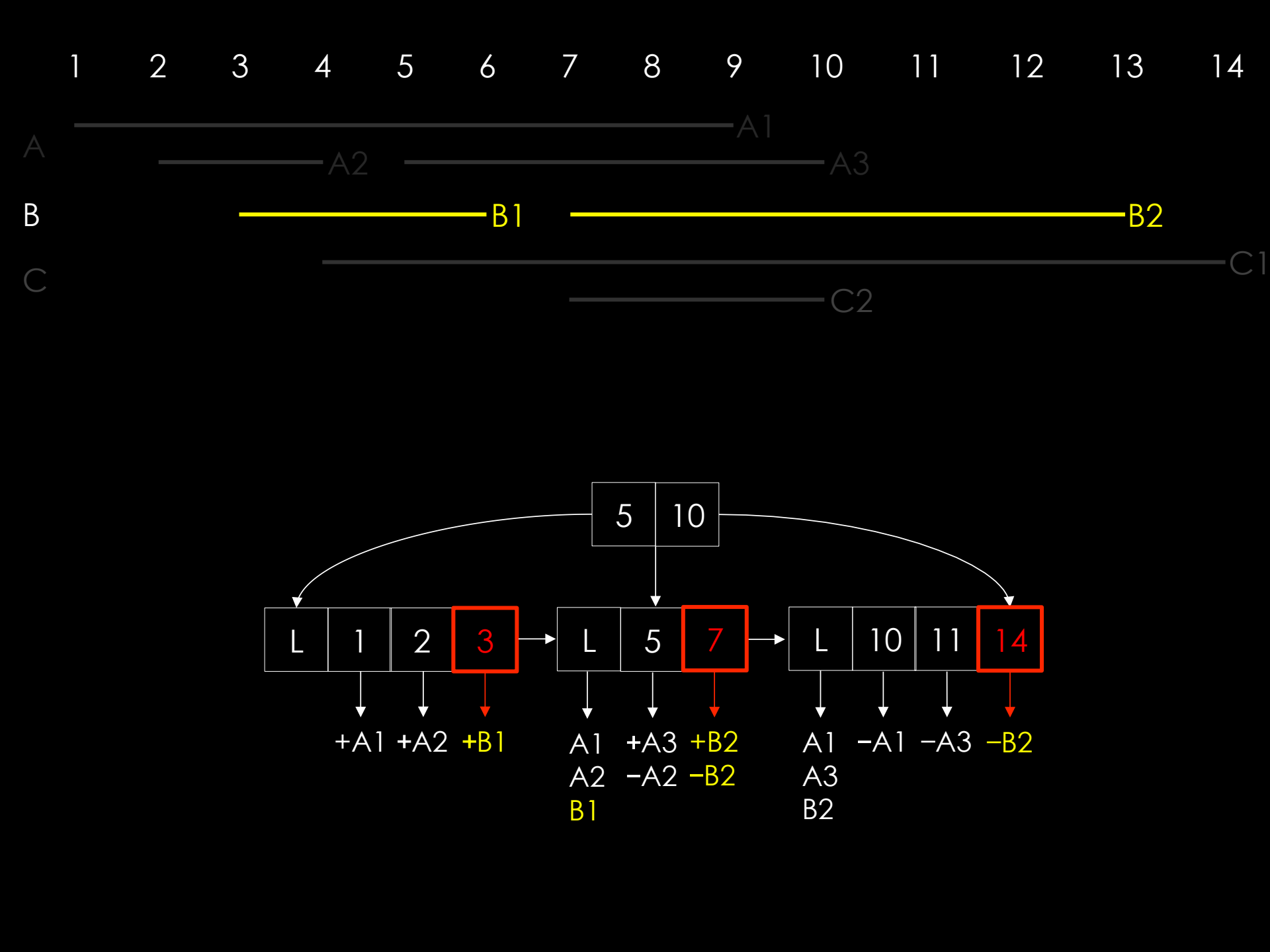

```
index(A3, 3, 6)
```

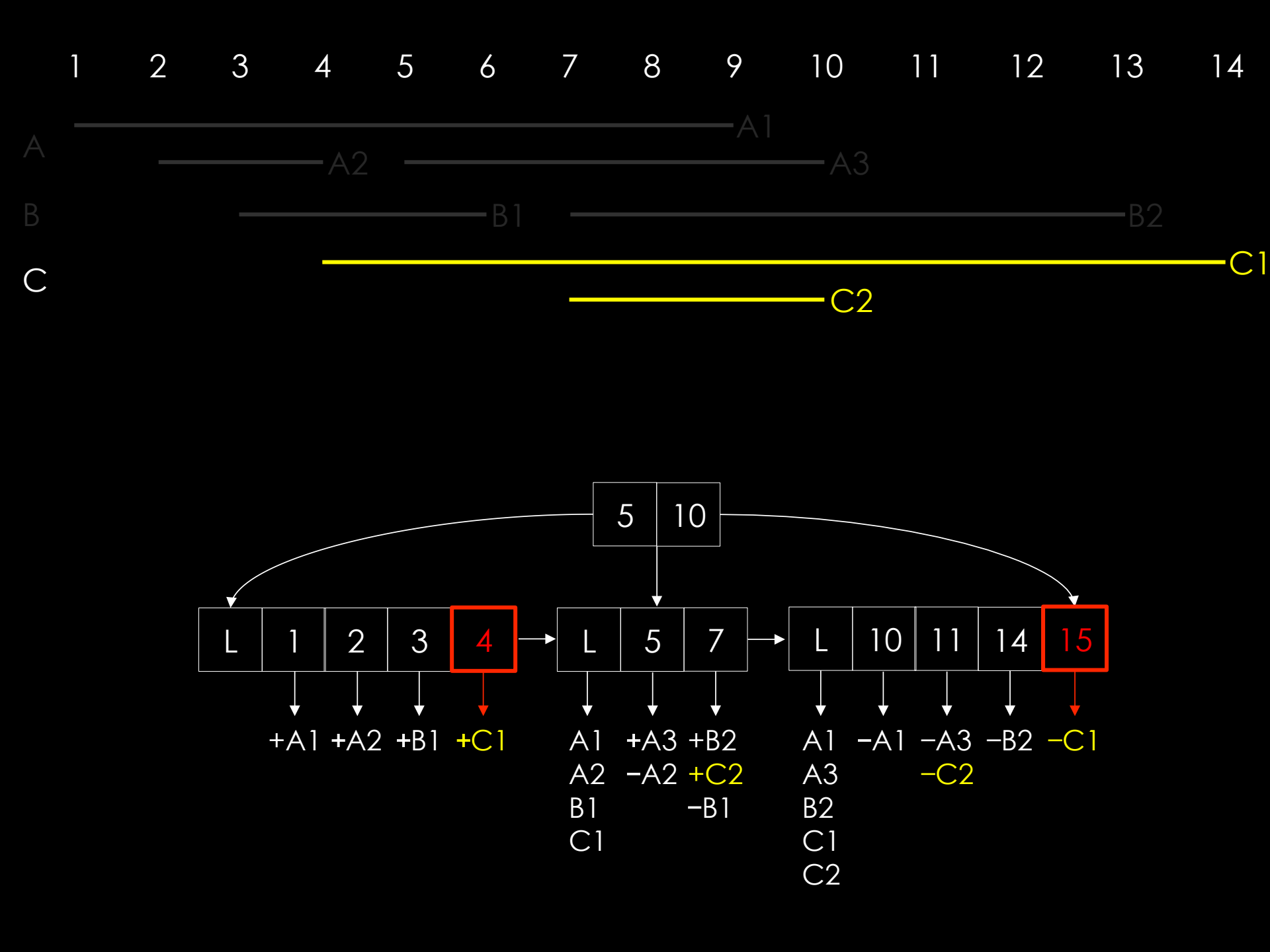
Promote a New Root

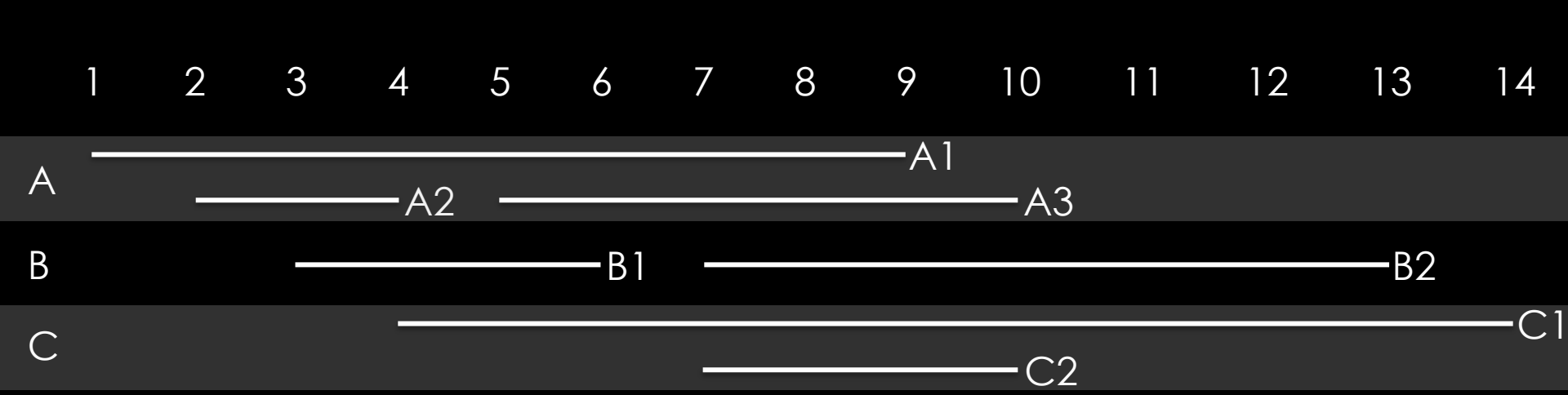



1 2 3 4 5 6 7 8 9 10 11 12 13 14
A
A1
A2
A3
B
B1
B2
C
C1
C2

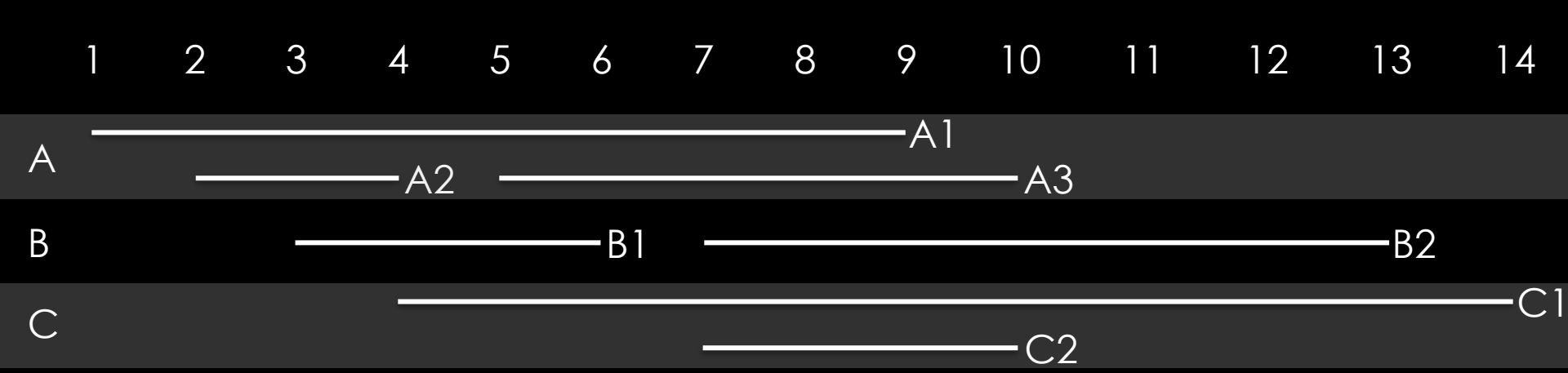

10
L 1 2 5
L 10 11
+A1 +A2 +A3
−A2
A1
A3
−A1 −A3

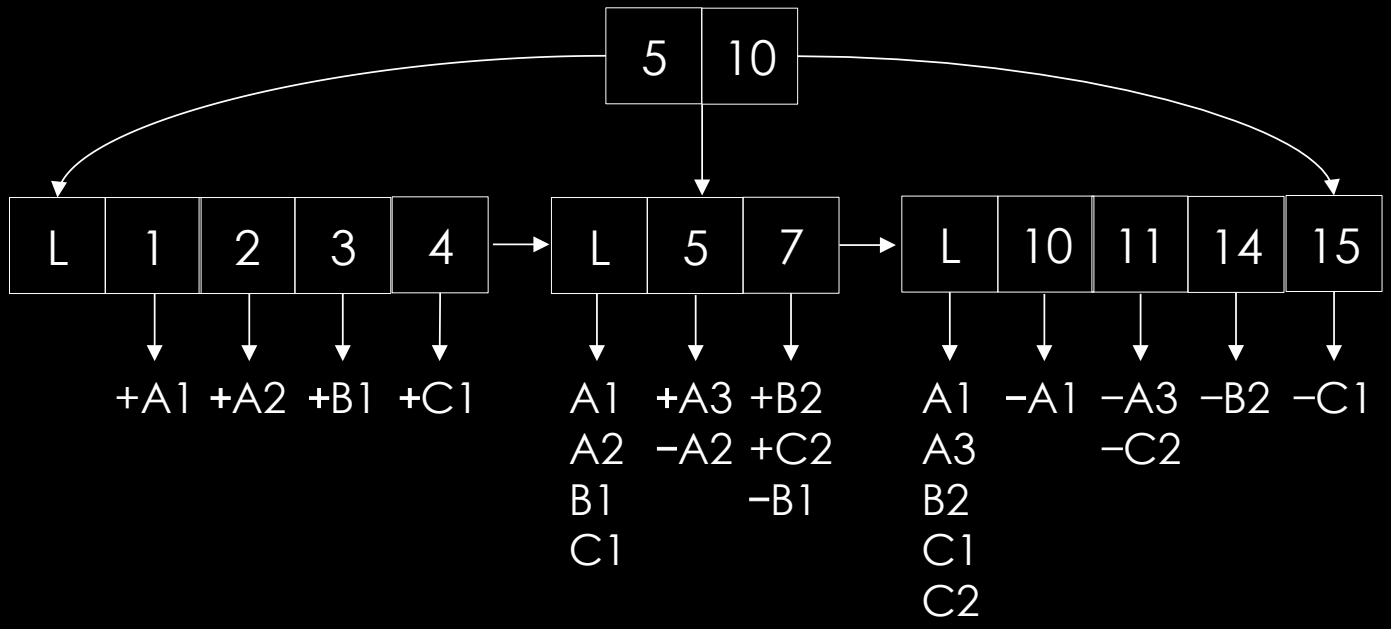

1 2 3 4 5 6 7 8 9 10 11 12 13 14
A
A1
A2
A3
B
B1
B2
C
C1
C2
5 10
L 1 2 3
L 5 7
L 10 11 14
+A1 +A2 +B1
A1 A2 B1
+A3 −A2 +B2 −B2
A1 A3 B2
−A1 −A3 −B2

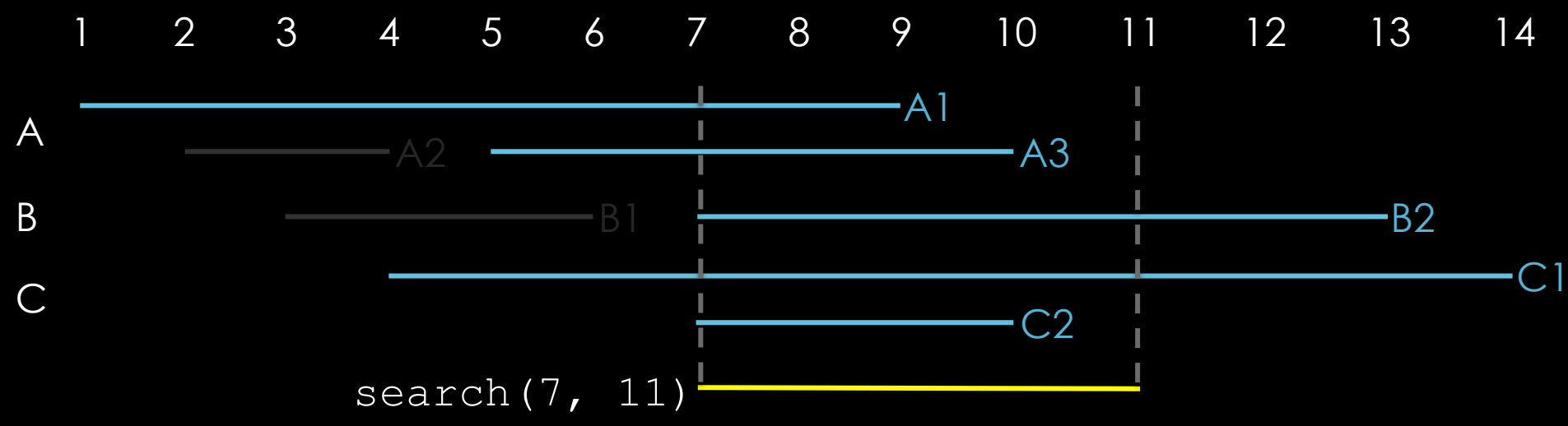

1 2 3 4 5 6 7 8 9 10 11 12 13 14
A
A1
A2
A3
B
B1
B2
C
C1
C2
5 10
L 1 2 3 4
L 5 7
L 10 11 14 15
+A1 +A2 +B1 +C1
A1 A2 B1 C1
+A3 −A2 +C2 −B1
A1 A3 B2 C1 C2
−A1 −A3 −C2 −B2 −C1

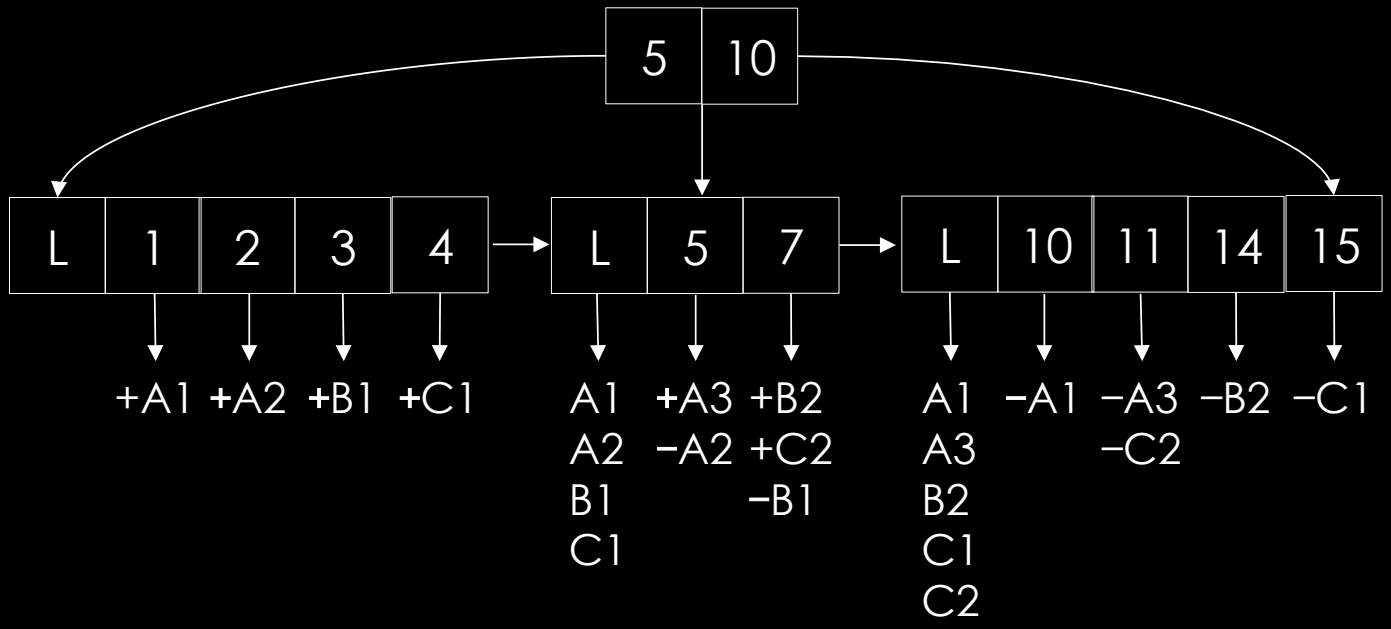


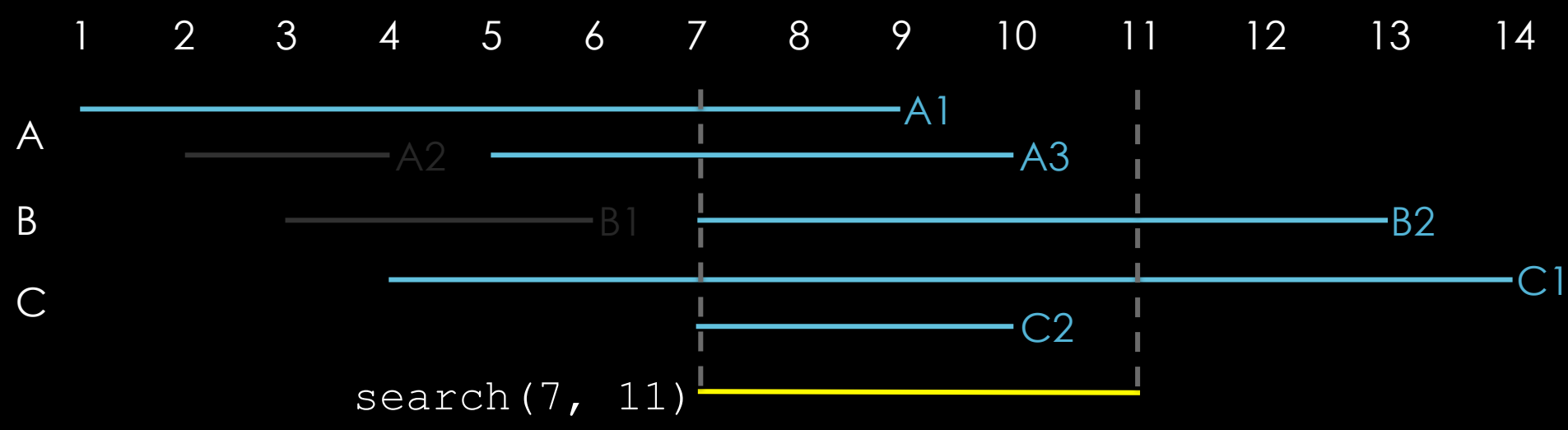

```
search(start, end)
  I = []
  from = find(start)
  to = find(end);
  I += from.leaf.L
  for node upto from:
    I += node.starts
    I -= node.ends
  for node in [from.next, to]:
    I += node.starts
```


1 2 3 4 5 6 7 8 9 10 11 12 13 14
A
A1
A2
A3
B
B1
B2
C
C1
C2
search(7, 11)

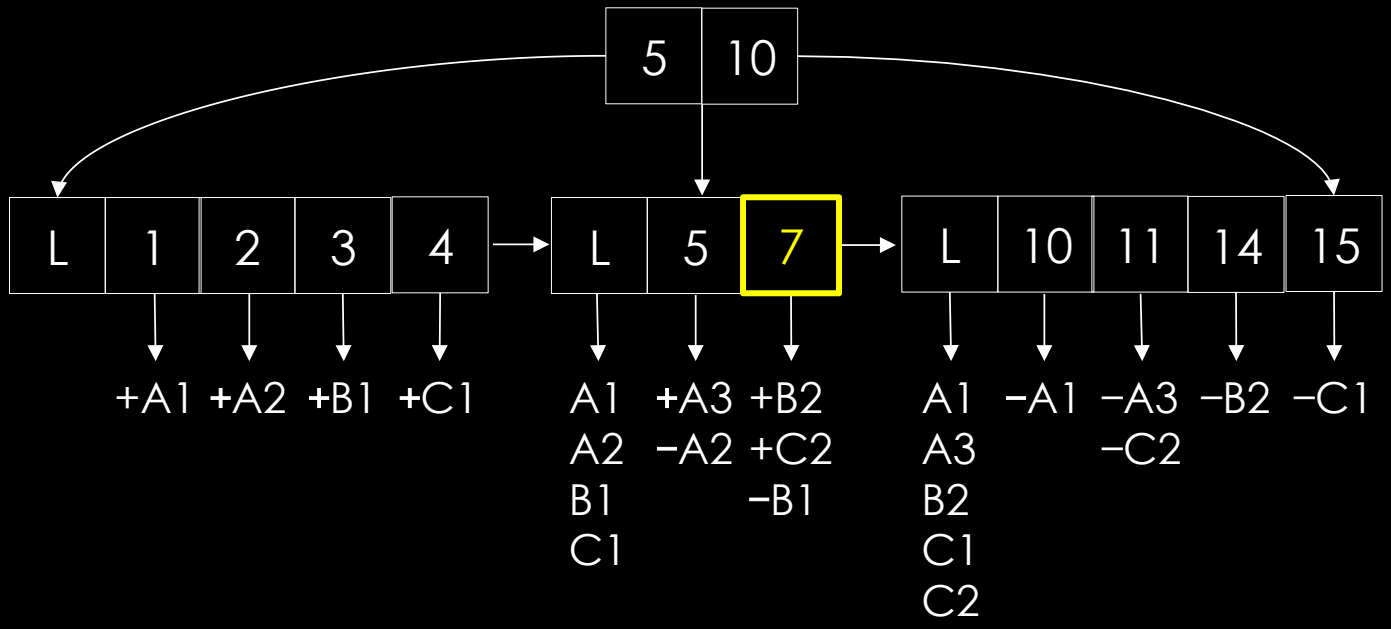

5 10
L 1 2 3 4
L 5 7
L 10 11 14 15
+A1 +A2 +B1 +C1
A1 +A3 +B2
A2 −A2 +C2
B1 −B1
C1
A1 −A1 −A3 −B2 −C1
A3 −C2
B2
C1
C2

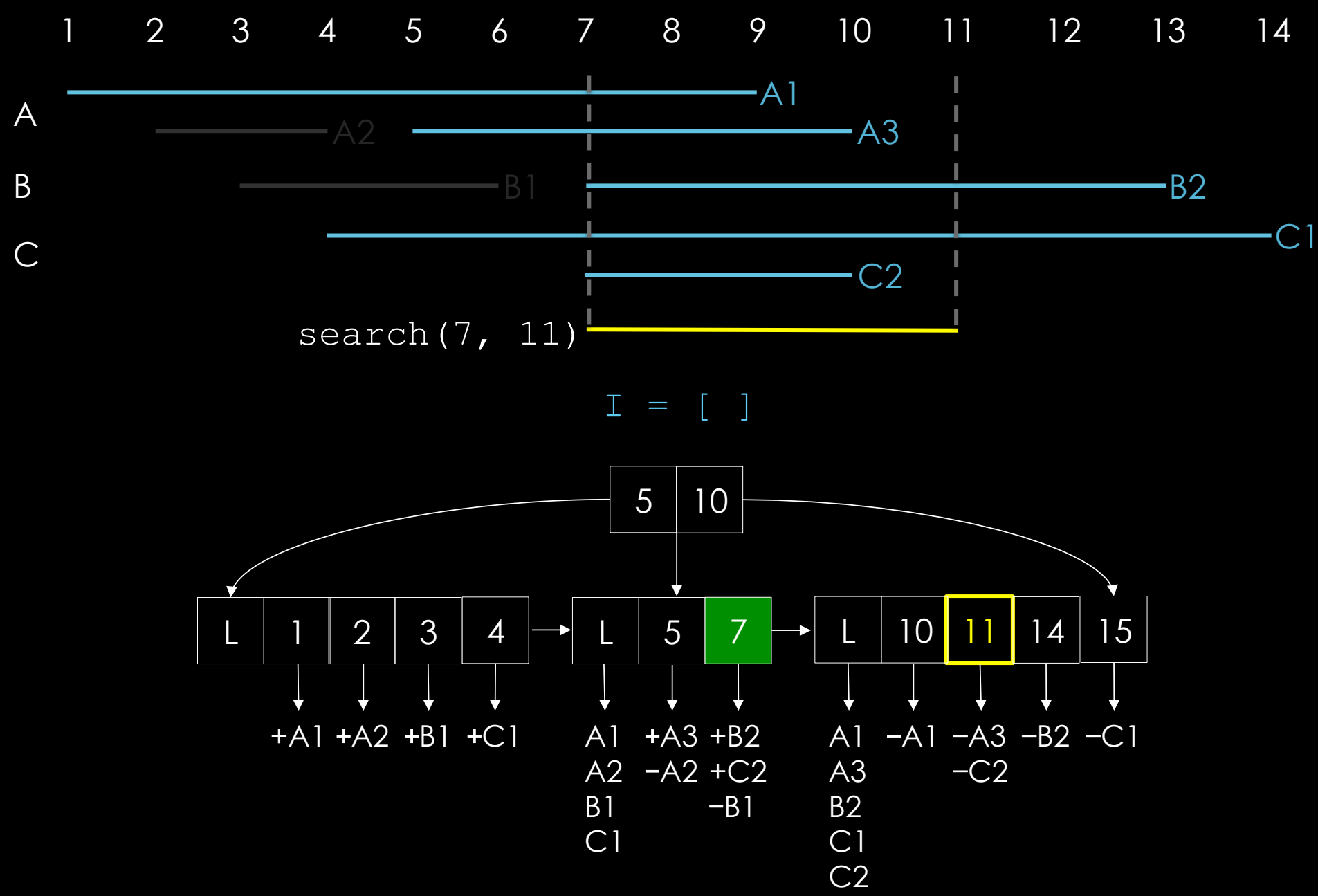

1 2 3 4 5 6 7 8 9 10 11 12 13 14
A
B
C
A1
A2
A3
B1
B2
C1
C2
search(7, 11)

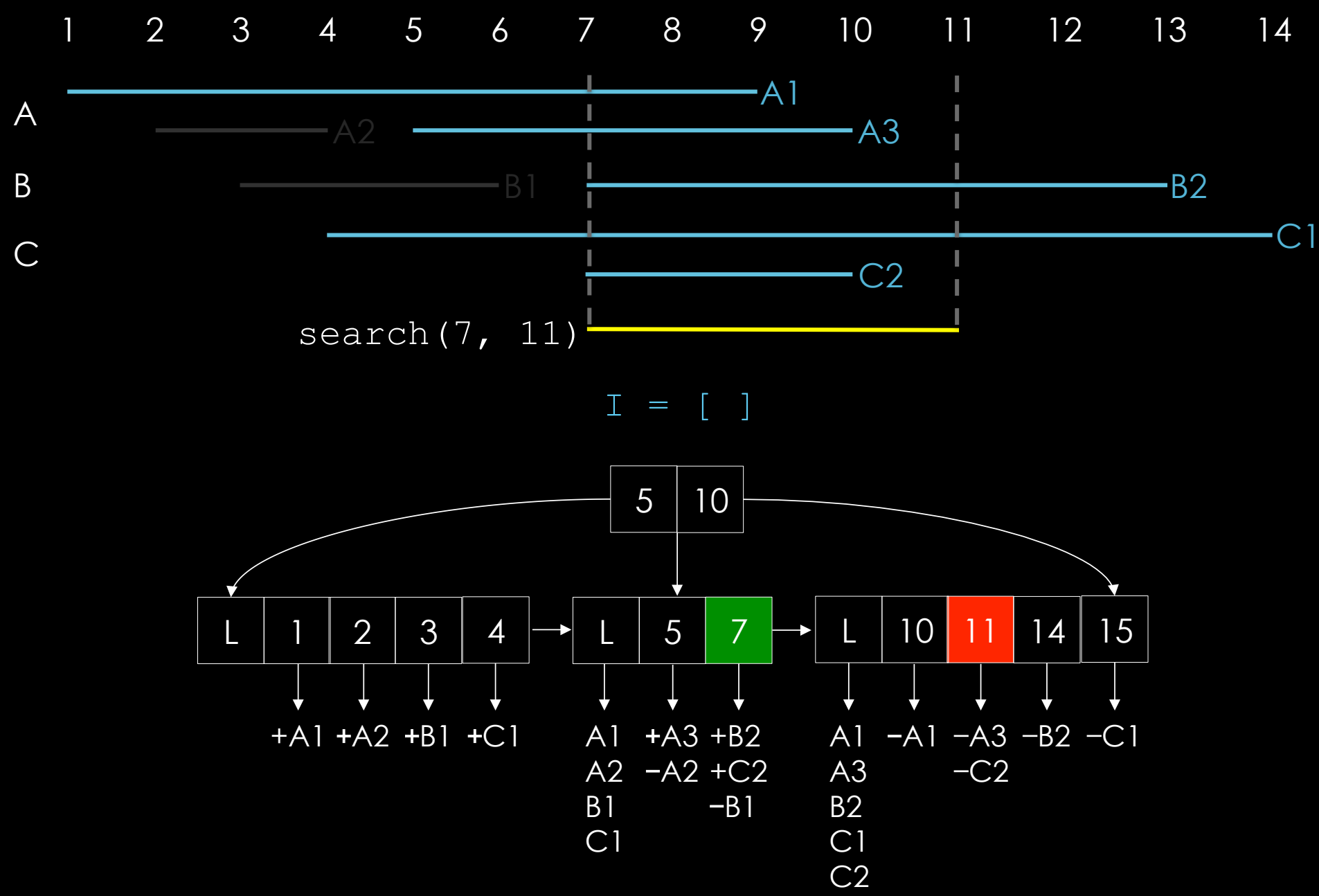

5 10
L 1 2 3 4
L 5 7
L 10 11 14 15
+A1 +A2 +B1 +C1
A1 +A3 +B2
A2 −A2 +C2
B1 −B1
C1
A1 −A1 −A3 −B2 −C1
A3 −C2
B2
C1
C2

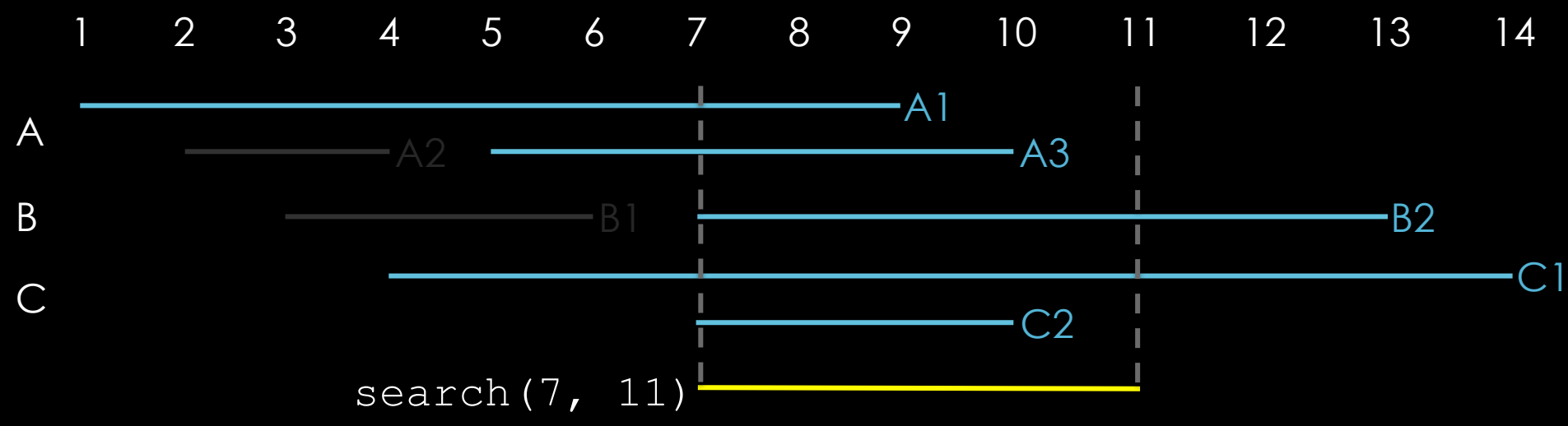

1 2 3 4 5 6 7 8 9 10 11 12 13 14
A
B
C
A1
A2
A3
B1
B2
C1
C2
search(7, 11)

I = [ ]

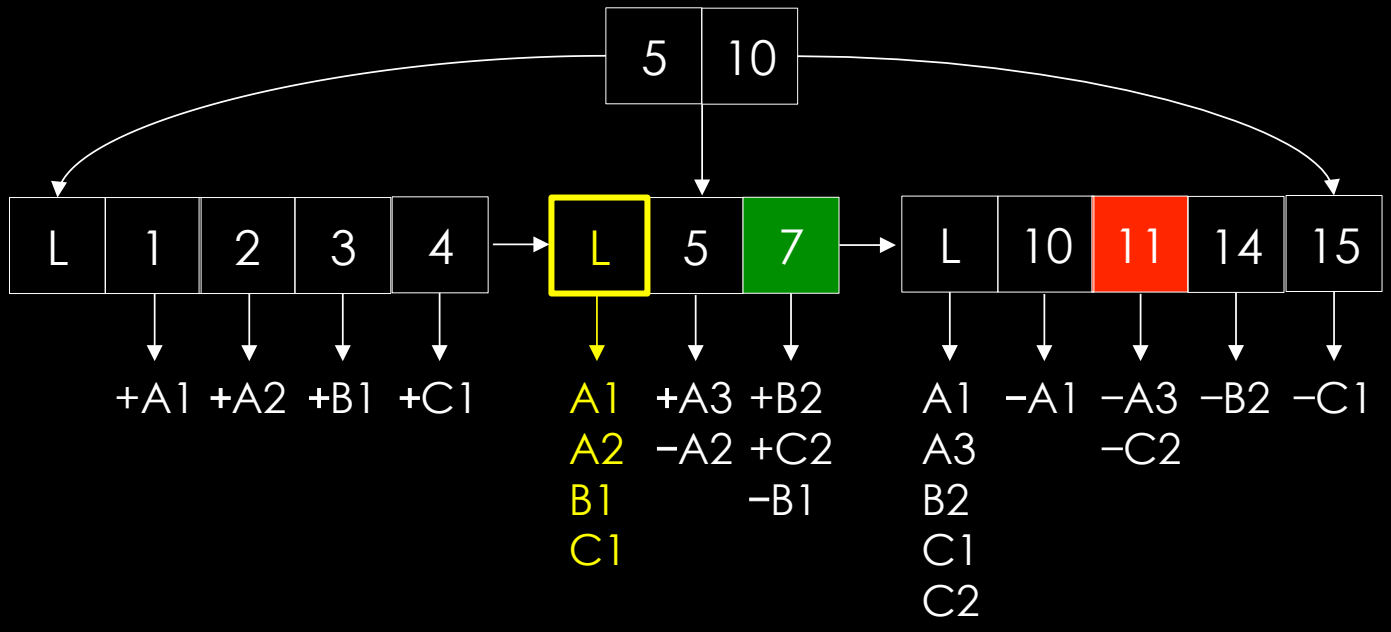

5 10
L 1 2 3 4
L 5 7
L 10 11 14 15
+A1 +A2 +B1 +C1
A1 +A3 +B2
A2 −A2 +C2
B1 −B1
C1
A1 −A1 −A3 −B2 −C1
A3 −C2
B2
C1
C2

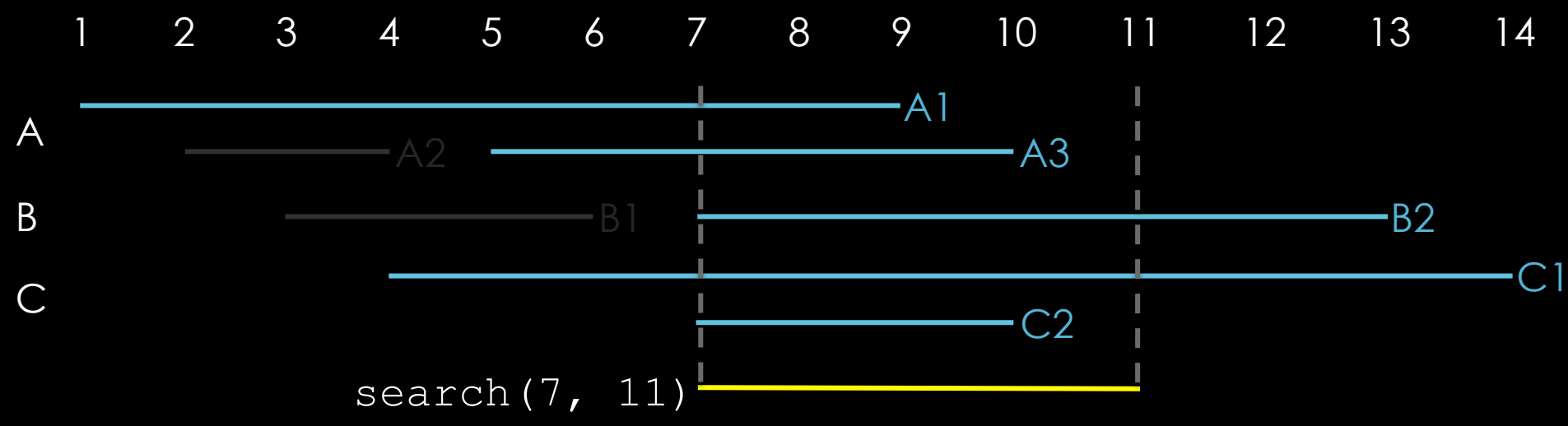

1 2 3 4 5 6 7 8 9 10 11 12 13 14
A
B
C
A1
A2
A3
B1
B2
C1
C2
search(7, 11)
I = [ ]
5 10
L 1 2 3 4
L 5 7
L 10 11 14 15
+A1 +A2 +B1 +C1
A1 A2 B1 C1
+A3 −A2
+B2 +C2 −B1
A1 A3 B2 C1 C2
−A1
−A3 −C2
−B2
−C1


1 2 3 4 5 6 7 8 9 10 11 12 13 14
A
B
C
A1
A2
A3
B1
B2
C1
C2
search(7, 11)
I = [ ]
5 10
L 1 2 3 4
L 5 7
L 10 11 14 15
+A1 +A2 +B1 +C1
A1 +A3 +B2
A2 −A2 +C2
B1 −B1
C1
A1 −A1 −A3 −B2 −C1
A3 −C2
B2
C1
C2

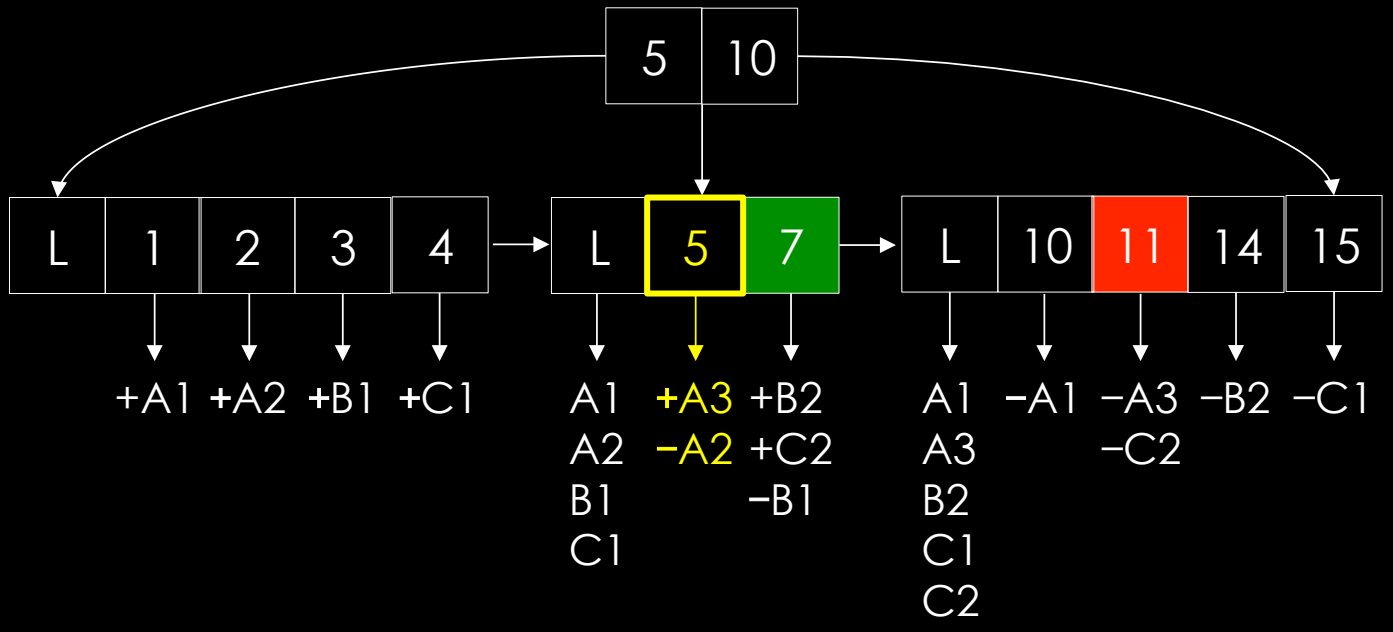

1 2 3 4 5 6 7 8 9 10 11 12 13 14
A
B
C
A1
A2
A3
B1
B2
C1
C2
search(7, 11)

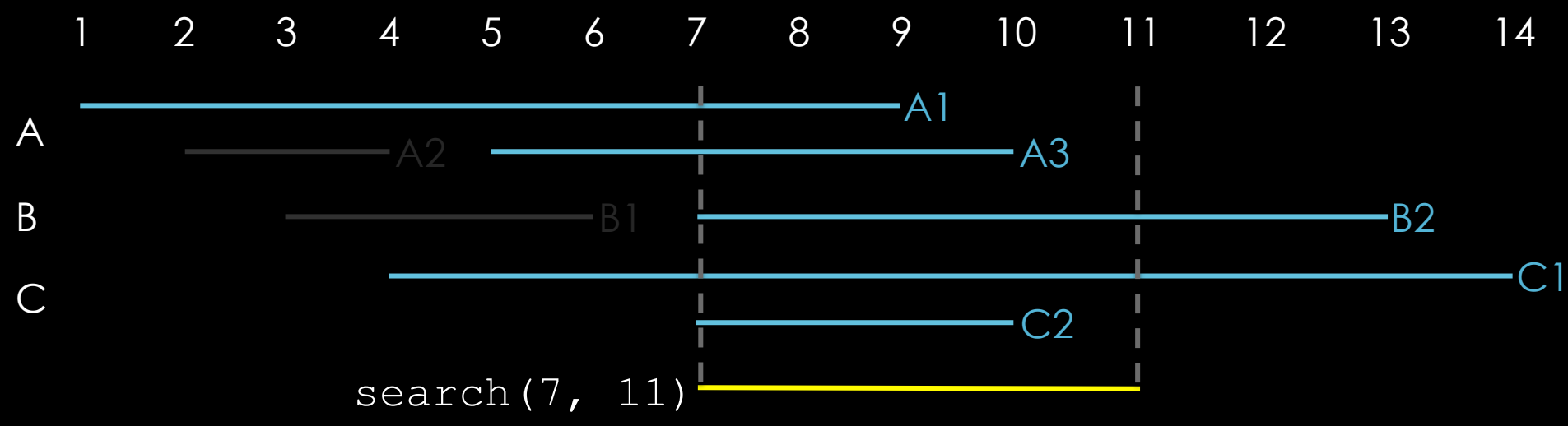
```
I = [A1 A2 B1 C1]
```


5 10
L 1 2 3 4
L 5 7
L 10 11 14 15
+A1 +A2 +B1 +C1
A1 A2 B1 C1
+A3 −A2 +B2 +C2 −B1
A1 A3 B2 C1 C2
−A1 −A3 −C2 −B2 −C1

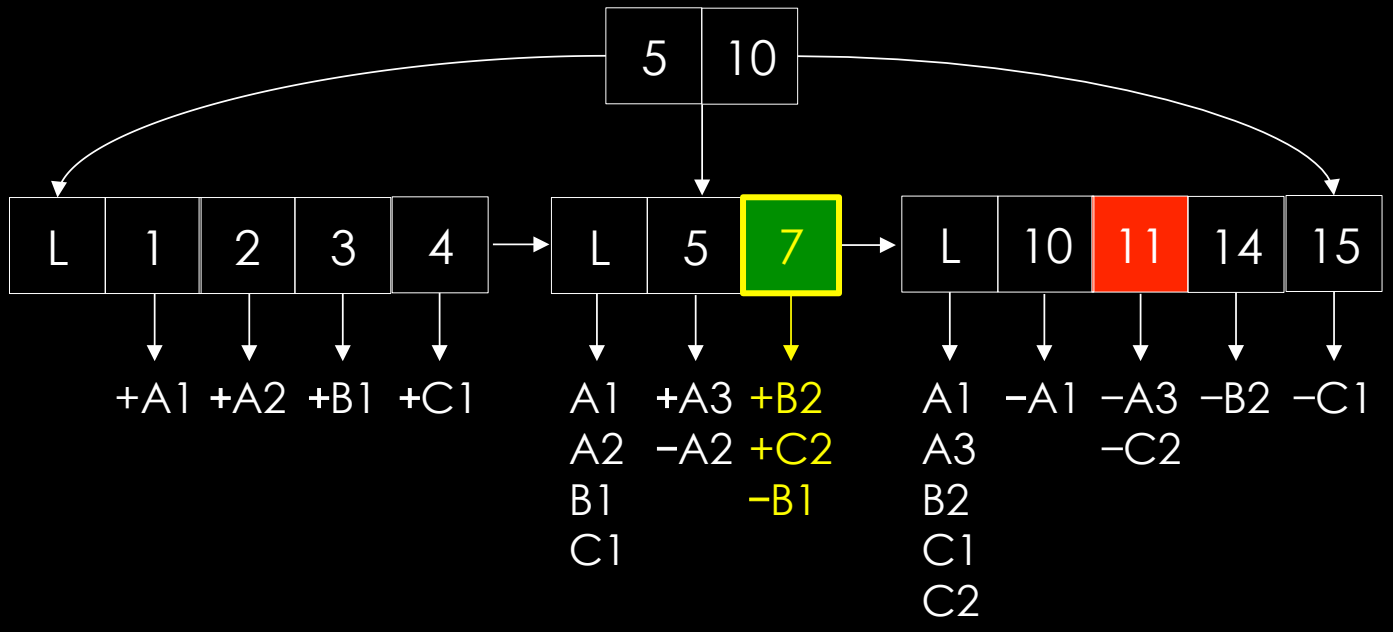


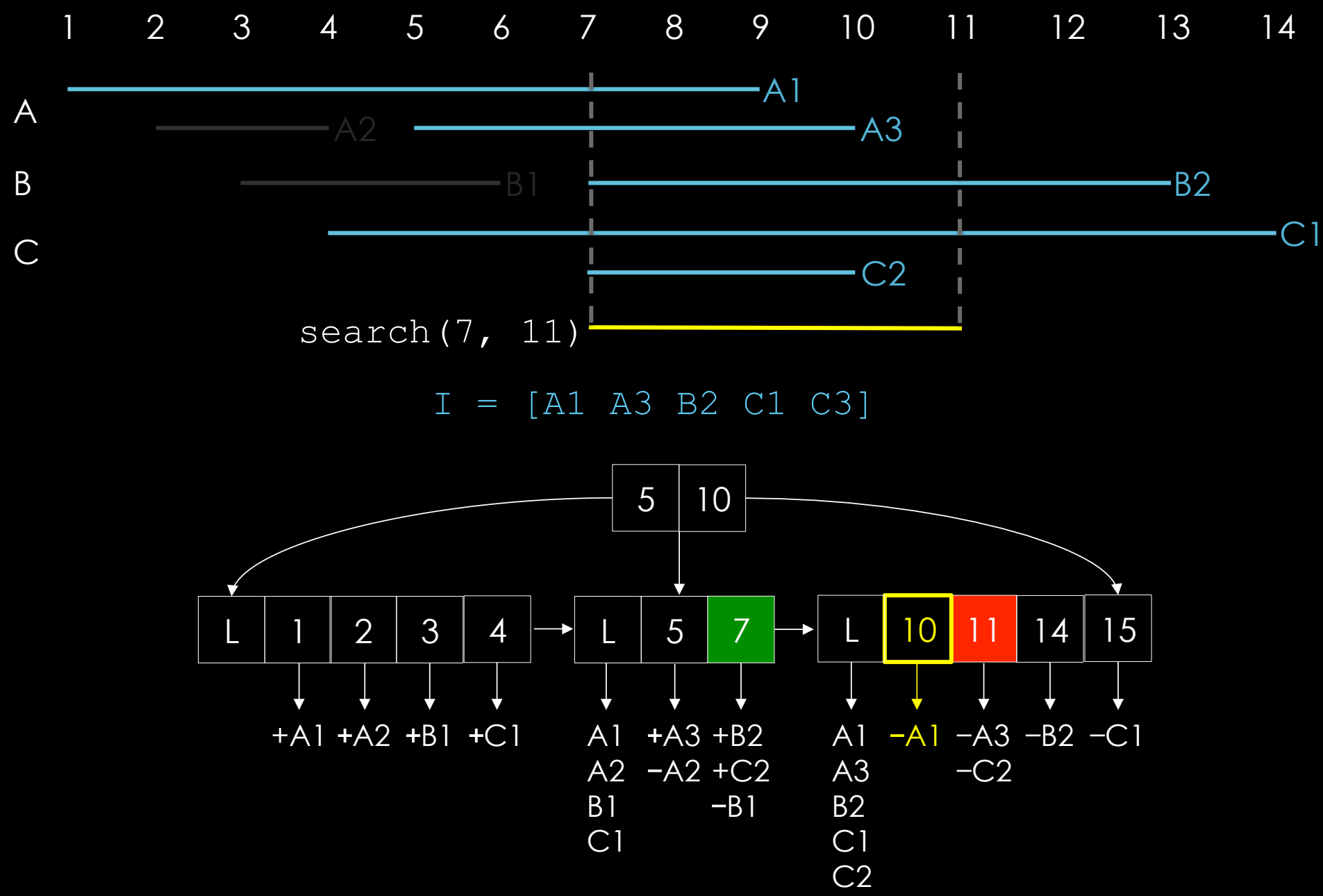

```
I = [A1 A2 B1 C1] + [A3] − [A2] = [A1 A3 B1 C1]
```

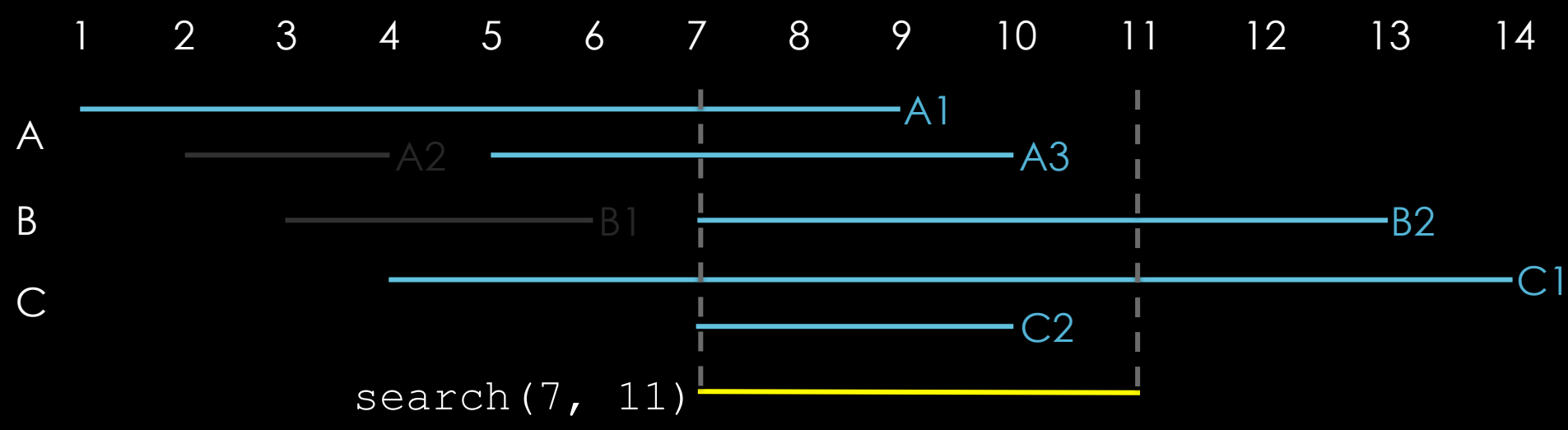




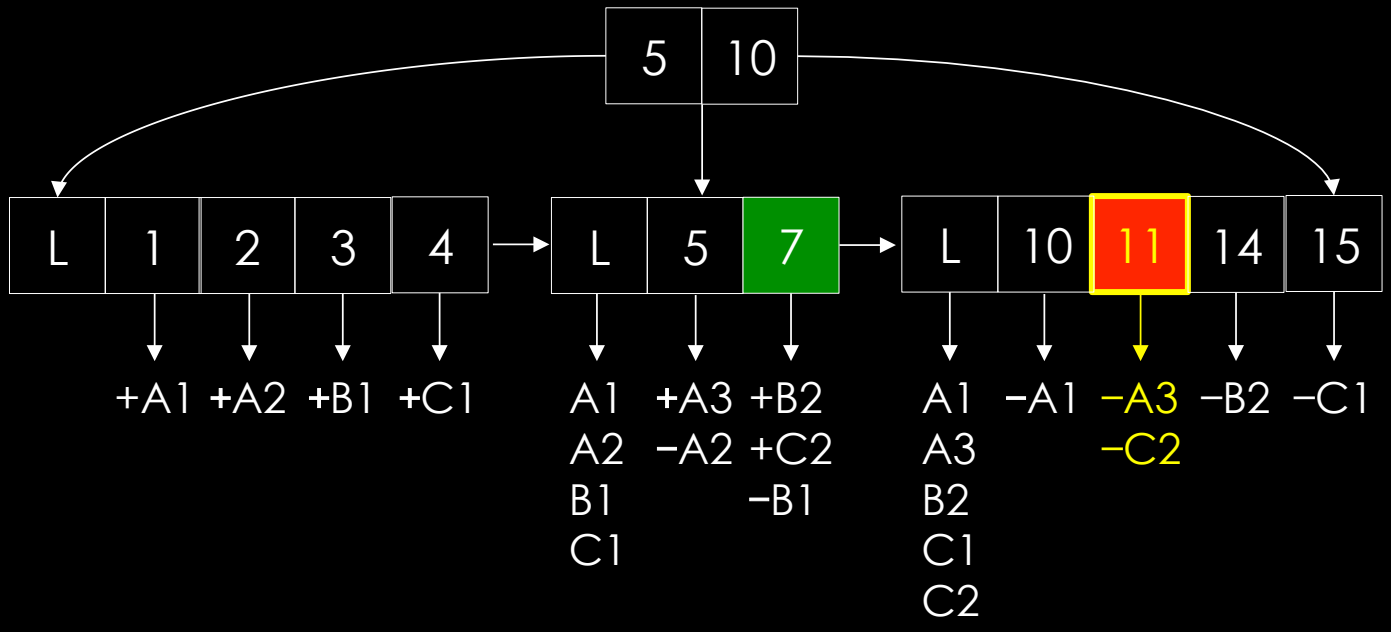

```
I = [A1 A3 B1 C1] + [B2 C2] - [B1] = [A1 A3 B2 C1 C3]
```




1 2 3 4 5 6 7 8 9 10 11 12 13 14
A
B
C
A1
A2
A3
B1
B2
C1
C2
search(7, 11)
I = [A1 A3 B2 C1 C3]
5 10
L 1 2 3 4
L 5 7
L 10 11 14 15
+A1 +A2 +B1 +C1
A1 A2 B1 C1
+A3 −A2 +C2 −B1
A1 A3 B2 C1 C2
−A1
−A3 −C2
−B2
−C1




```
I = [A1 A3 B2 C1 C3]
```

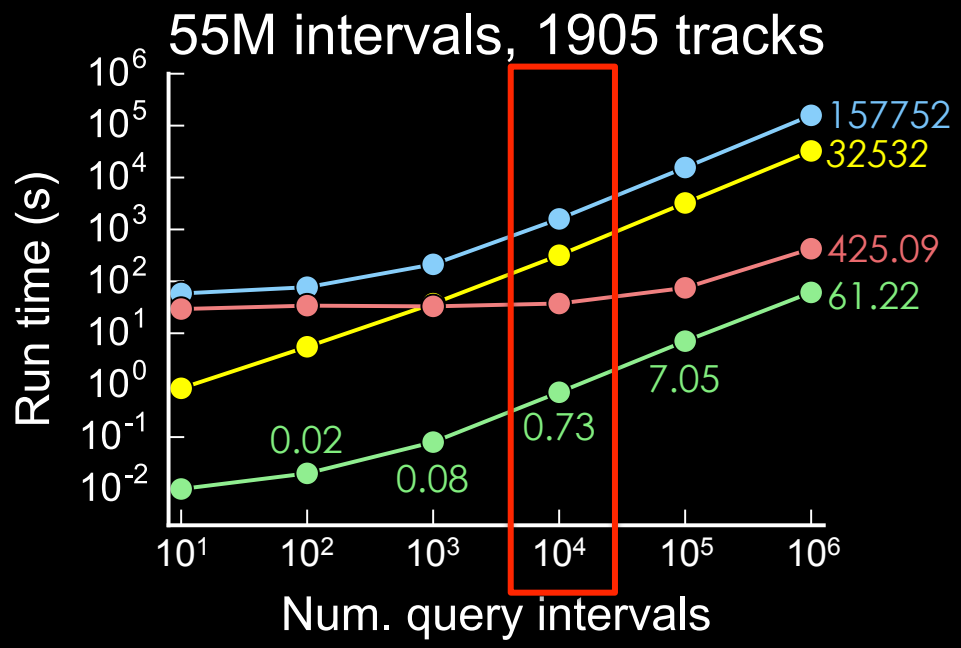


Count intersections for 10 -1M 100bp intervals

127 reference epigenomes
15 genomic states [CHROMHMM Ernst 2012]
1905 tracks

SQLITE3 w/ UCSC binning
TABIX
BEDTOOLS sorted
GIGGLE

2.6GHz Intel Xeon CPU, 20MB cache



2576X 531X 6.9X

my_chipseq.bed

All **Encode** 1KG GTEx ExAC TOPMed UCSC

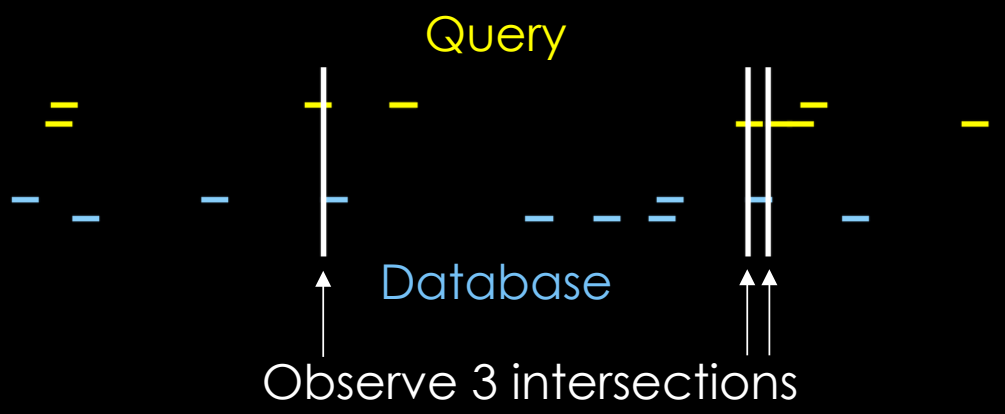
DNase-seq of K562
Homo sapiens, adult 53 year
Lab: John Stamatoyannopoulos, UW
Project: ENCODE

ChIP-seq of forebrain
Mus musculus, embryonic 10.5 day
Target: H3K9me3
Lab: Bing Ren, UCSD
Project: ENCODE

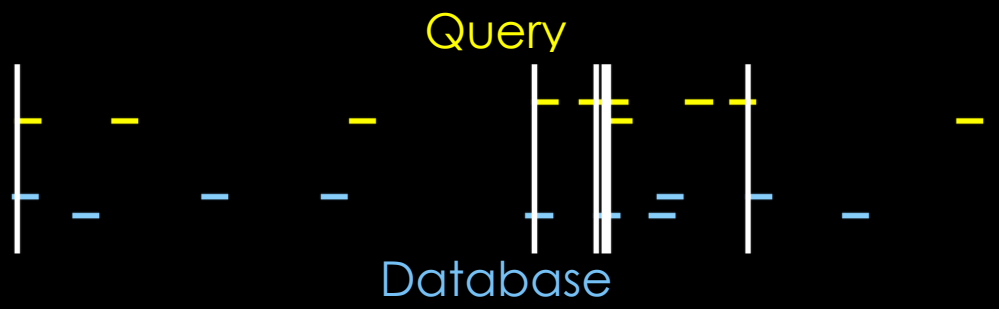
ChIP-seq of heart
Mus musculus, embryonic 10.5 day
Target: H3K27me3
Lab: Bing Ren, UCSD
Project: ENCODE

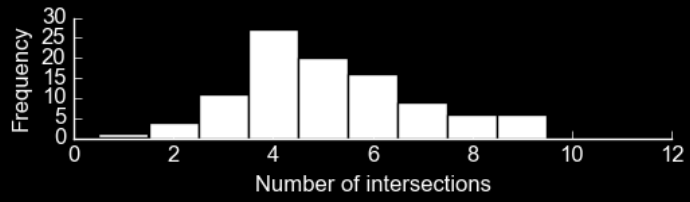
ChIP-seq of embryonic facial prominence
Mus musculus, embryonic 10.5 day
Target: H3K27me3
Lab: Bing Ren, UCSD
Project: ENCODE

Monte Carlo simulations



Monte Carlo simulations



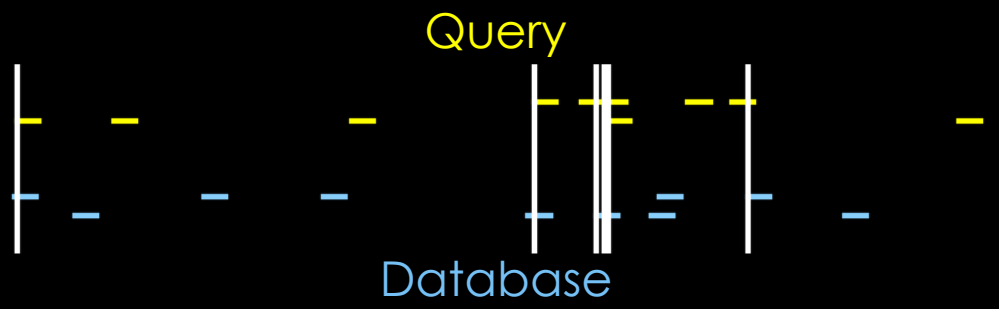


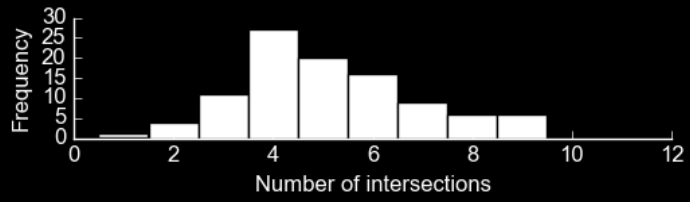
Observe 3 intersections

Expect ~4

0.75 "enrichment"

Monte Carlo simulations





Observe 3 intersections

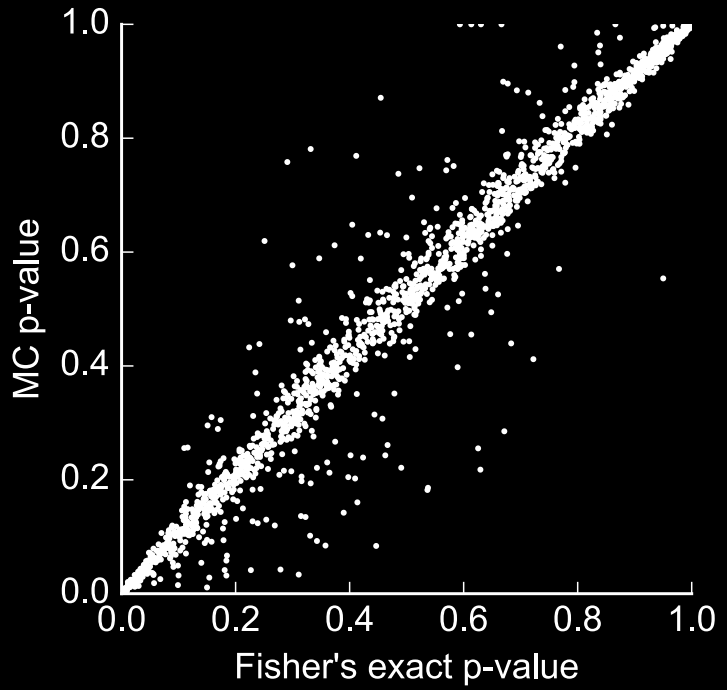
Expect ~4

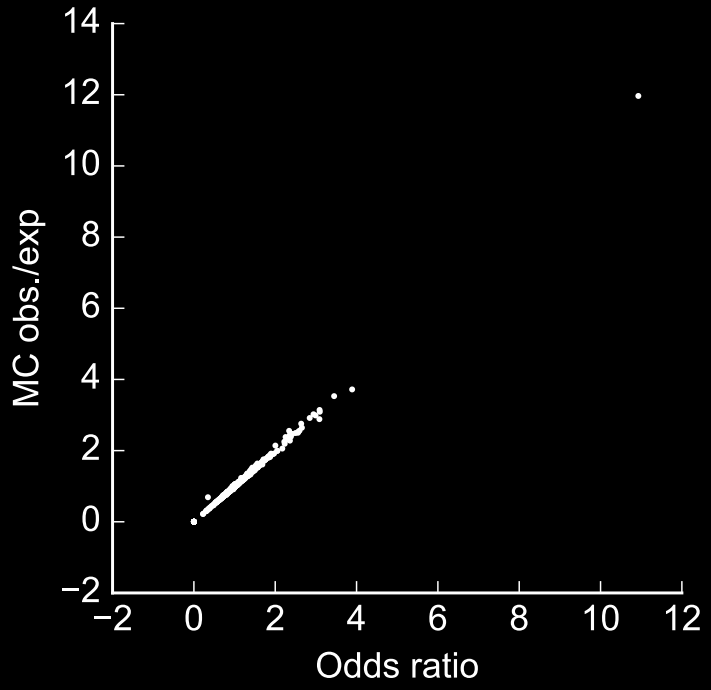
0.75 "enrichment"

Contingency table

	In target	Not in target
In query	| Q T |	|Q | - | Q T |
Not In query	|T| - | Q T |	$\frac{\text{Genome size}}{\mu(Q) + \mu(T)} - (\,|Q| + |T| - |Q\,T|\,)$

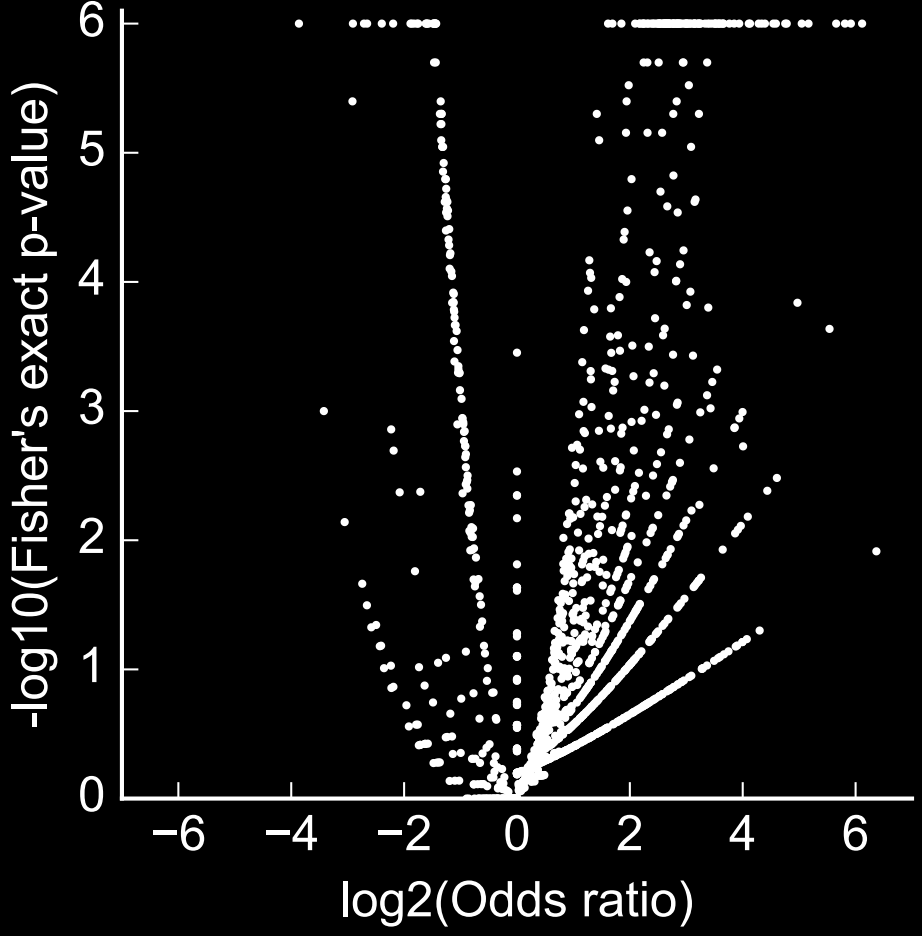
Brent Pedersen





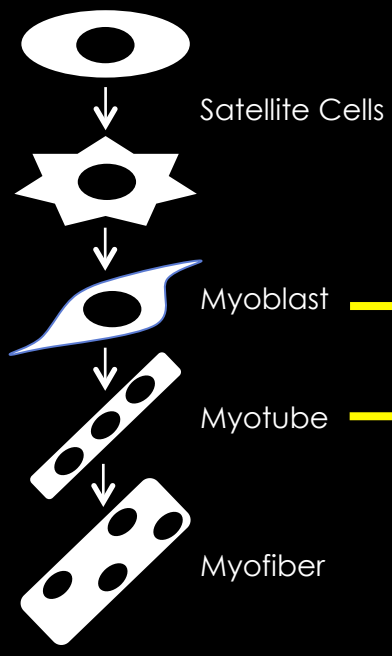
False Positive:
Small fold change, Small variance

False Negative:
Large fold change, Large variance



A novel significance score for gene selection and ranking, Xiao 2014

MyoD



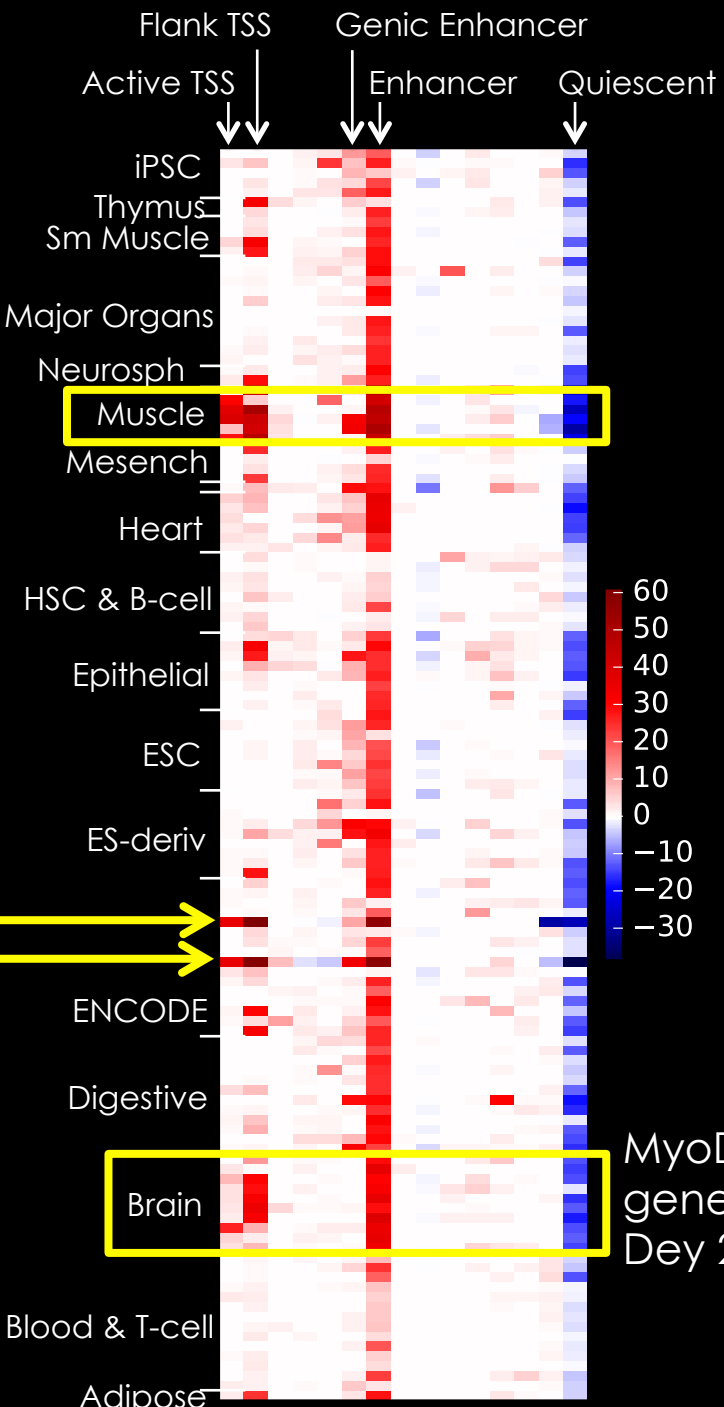
Zammit, 2006

ChIP-Seq MacQuarrie, 2013

+

ROADMAP epigenomics PROJECT

127 tissues
15 genomic states

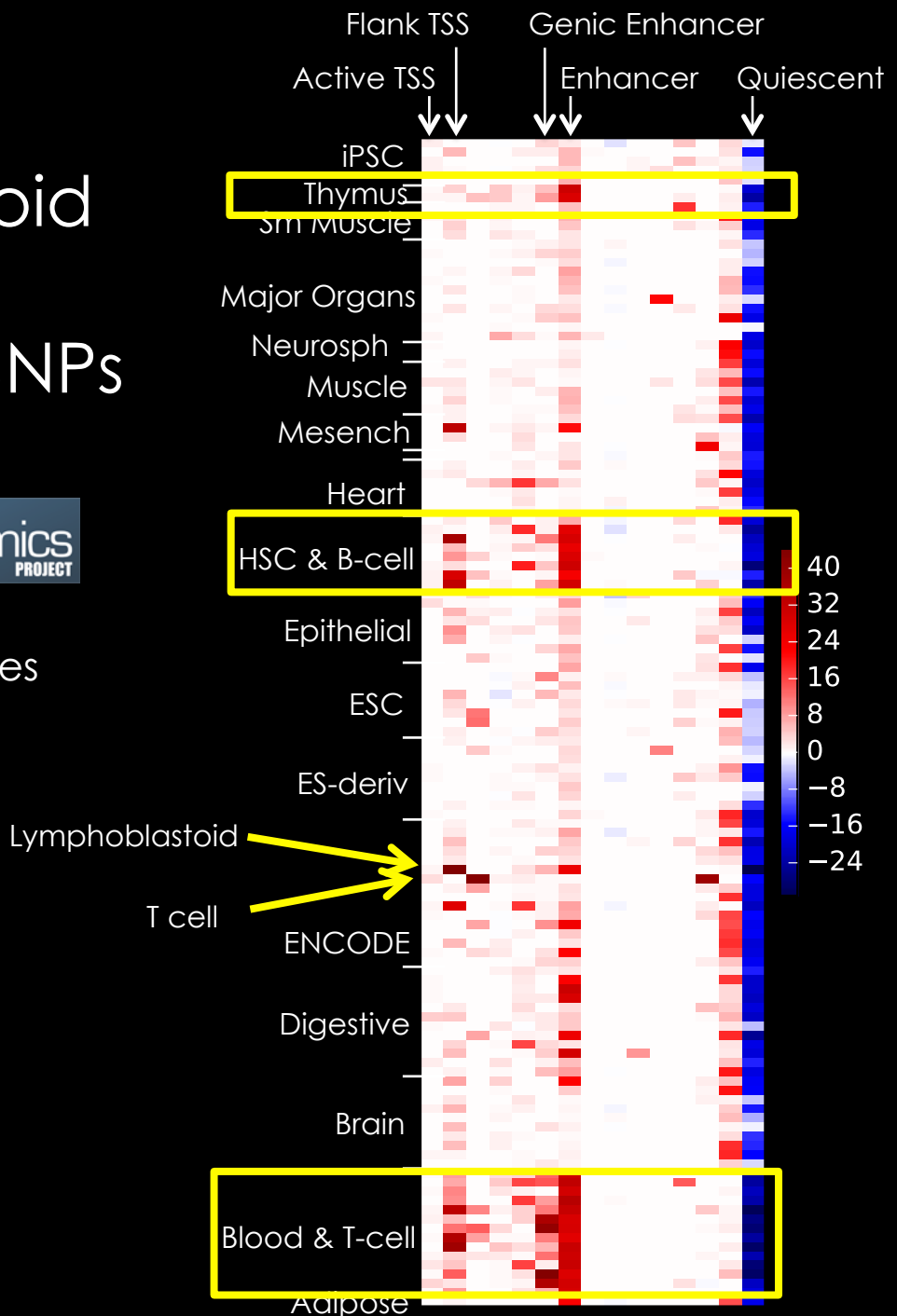


MyoD is a tumor suppressor gene in medulloblastoma
Dey 2012, Cancer Res

Rheumatoid Arthritis
84 GWAS SNPs
+

ROADMAP epigenomics PROJECT

127 tissues
15 genomic states



UCSC

Centers for Common Disease Genomics

OMIM

1000 Genomes
A Deep Catalog of Human Genetic Variation

GTEx

ENCODE

ClinVar

GGTATGGGGCCAAGAGATATATCT
CGGCTGTCATCACTTAGACCTCAC
TGGGCATAAAAGTCAGGGCAGAGC
TGCATCTGACTCCTGAGGAGAAGT
TGGTATCAAGGTTACAAGACAGGT
GACTCTCTCTGCCTATTGGTCTAT

Tonya Di Sera

24. chr3 8,635,600-8,636,200

CLOSE

Skeletal Muscle Female
Small Intestine
Stomach Mucosa
Stomach Smooth Muscle
Thymus
Spleen

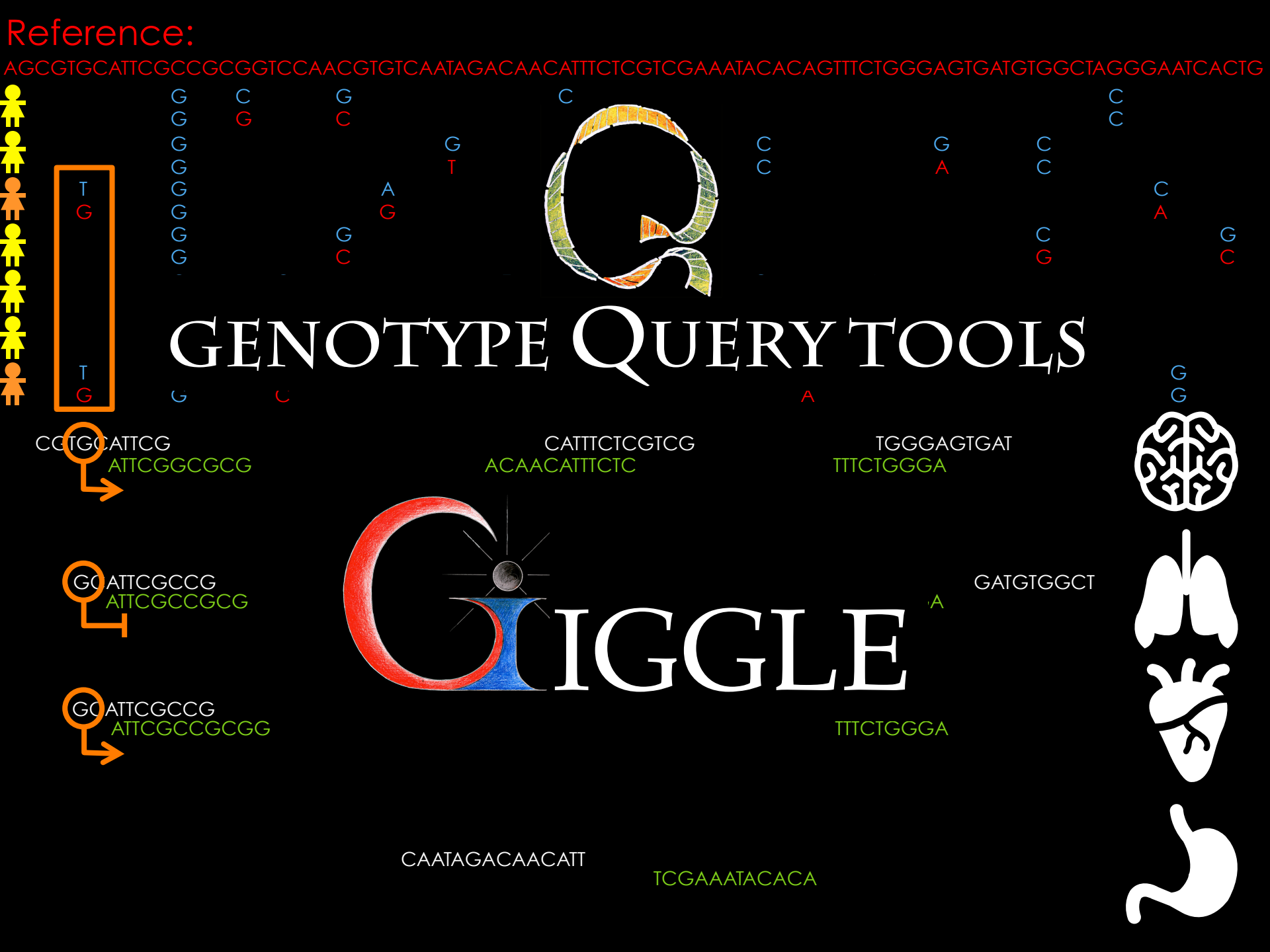
github.com/ryanlayer/giggle

Thousands of Annotations

100 Millions of Intervals

Fast Queries

High Quality Enrichment Score


Reference:
AGCGTGCATTCGCCGCGGTCCAACGTGTCAATAGACAACATTTCTCGTCGAAATACACAGTTTCTGGGAGTGATGTGGCTAGGGAATCACTG
GENOTYPE QUERY TOOLS
CGTGCATTCG
ATTCGGCGCG
GCATTCGCCG
ATTCGCCGCG
GCATTCGCCG
ATTCGCCGCGG
CATTTCTCGTCG
ACAACATTTCTC
TGGGAGTGAT
TTTCTGGGA
GIGGLE
GATGTGGCT
TTTCTGGGA
CAATAGACAACATT
TCGAAATACACA

SFARI

519

gatk 70M SNVs

lumpy 100K SVs

GiQ

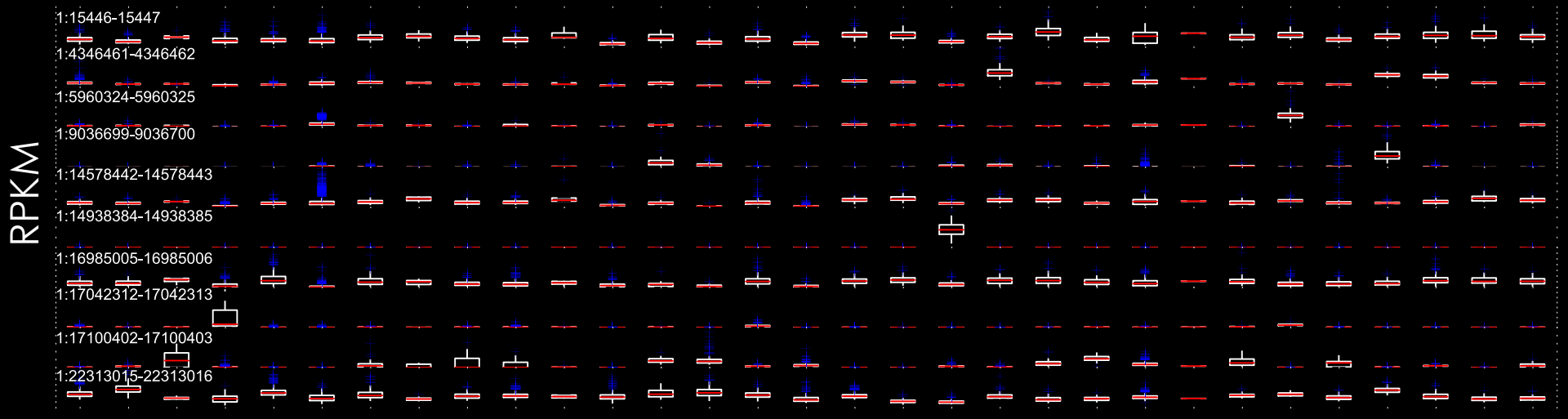
UCSC ← Not in problematic regions (SEGDUPs)

ExAC ← 1000 Genomes A Deep Catalog of Human Genetic Variation ← ClinVar ← OMIM ← Not previously observed

de novo in affected

GTEx ← Gene expression

Tissues



SFARI

519

gatk 70M SNVs

lumpy 100K SVs

GiQ

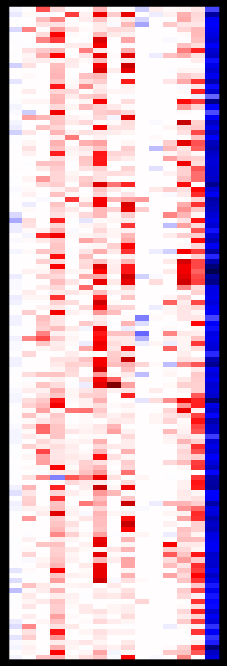
UCSC ← Not in problematic regions (SEGDUPs)

ExAC ← 1000 Genomes A Deep Catalog of Human Genetic Variation ← ClinVar ← OMIM ← Not previously observed

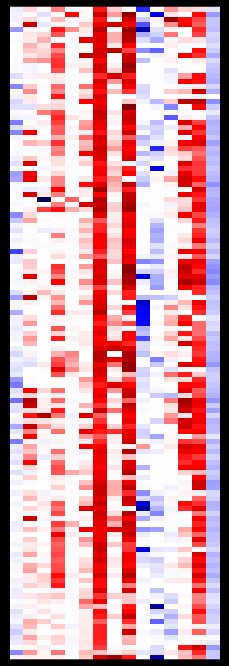
de novo in affected

de novo in unaffected

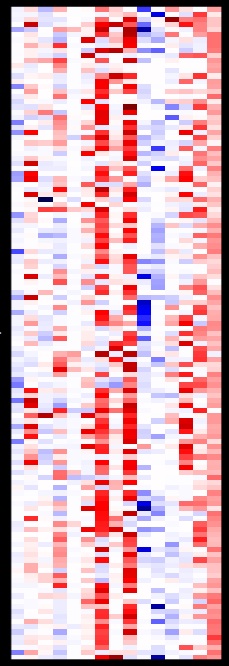
ROADMAP epigenomics PROJECT



ROADMAP epigenomics PROJECT



difference



SFARI

519

gatk 70M SNVs

lumpy 100K SVs

GiQ

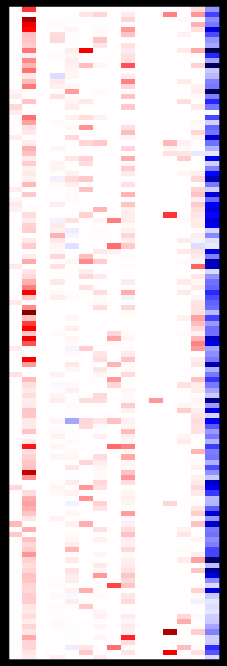
UCSC ← Not in problematic regions (SEGDUPs)

ExAC ← 1000 Genomes A Deep Catalog of Human Genetic Variation ← ClinVar ← OMIM ← Not previously observed

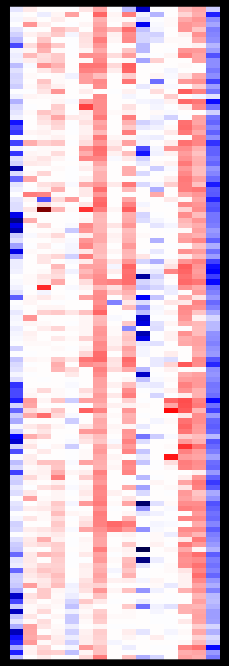
de novo in affected AFR

de novo in unaffected AFR

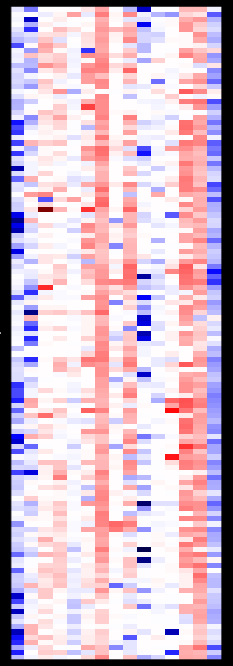
ROADMAP epigenomics PROJECT



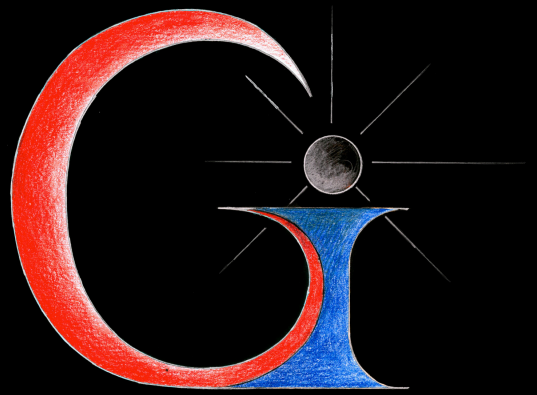
ROADMAP epigenomics PROJECT



difference



&



GENOTYPE QUERY TOOLS

GIGGLE

github.com/ryanlay

Questions?

n M e

ryan.layer@

@ryan

Aaron Quinlan, Brent a

USTAR Center for Genetic Discovery