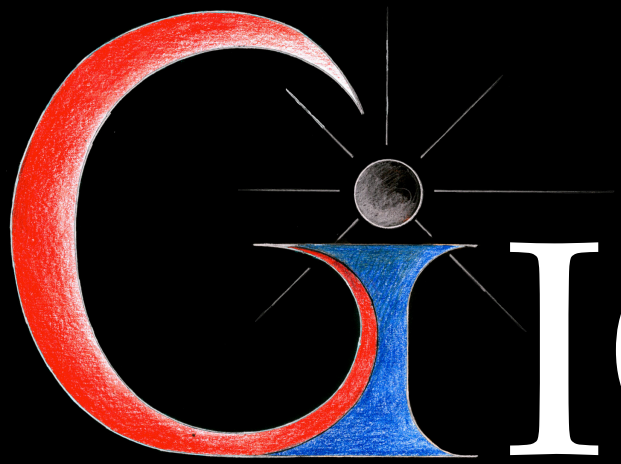


GIGGLE

github.com/ryanlayer/giggle

Search **ALL** Genome Annotations

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

USTAR Center for Genetic Discovery

UNIVERSITY OF UTAH SCHOOL OF MEDICINE

GFF

NARROWPEAK

VCF

BROADPEAK

BEDPE

CRAM

BED

SAM

BCF

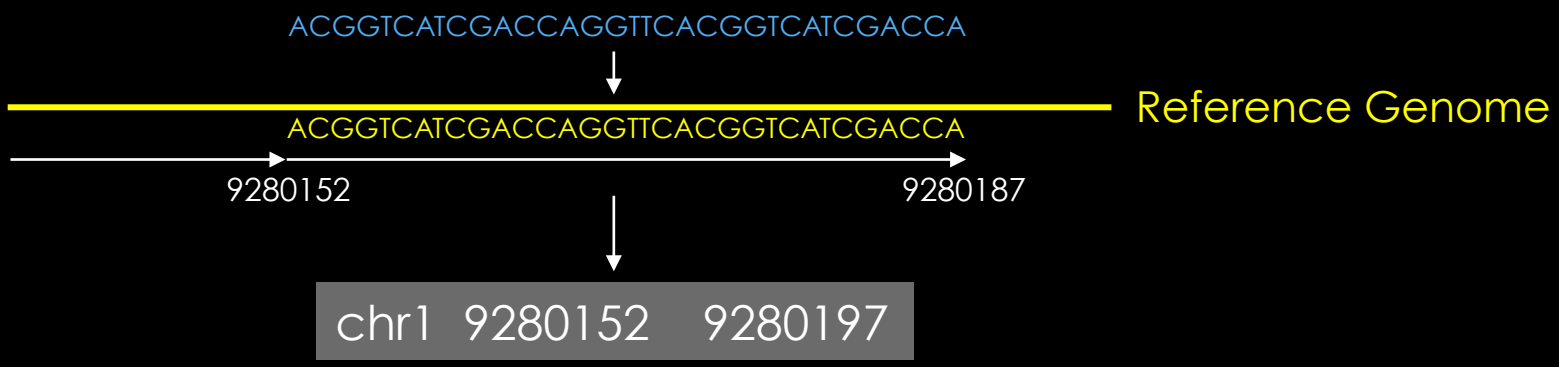
BAM

BIGWIG

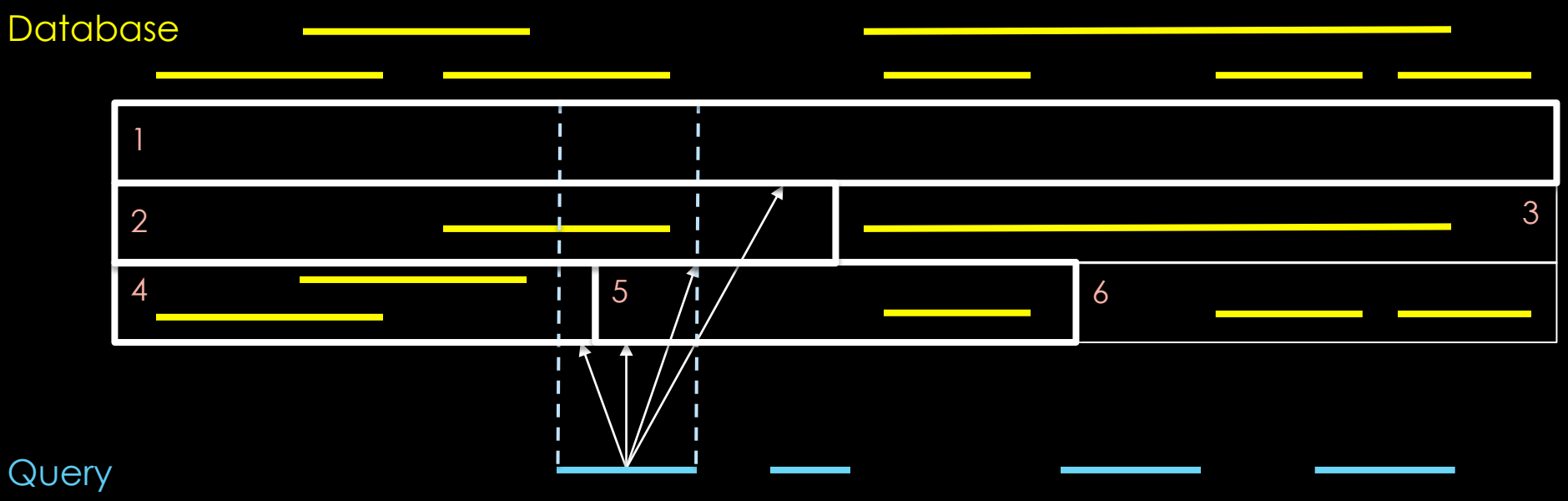
BEDGRAPH

WIG

BIGBED

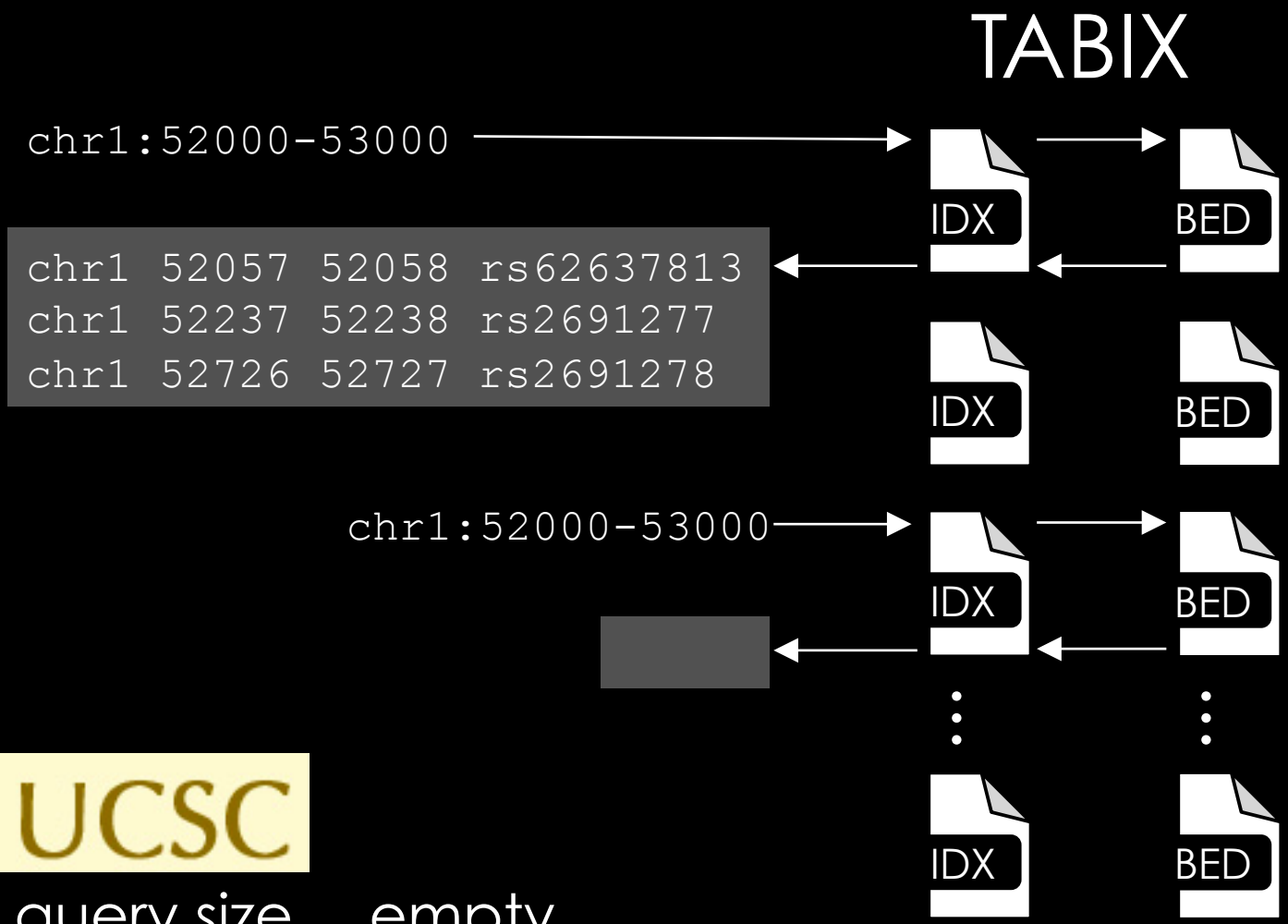


Hierarchical Binning [UCSC Kent 2002, TABIX Li 2011, BEDTOOLS Quinlan 2010]



Sweep [BEDTOOLS]

TABIX

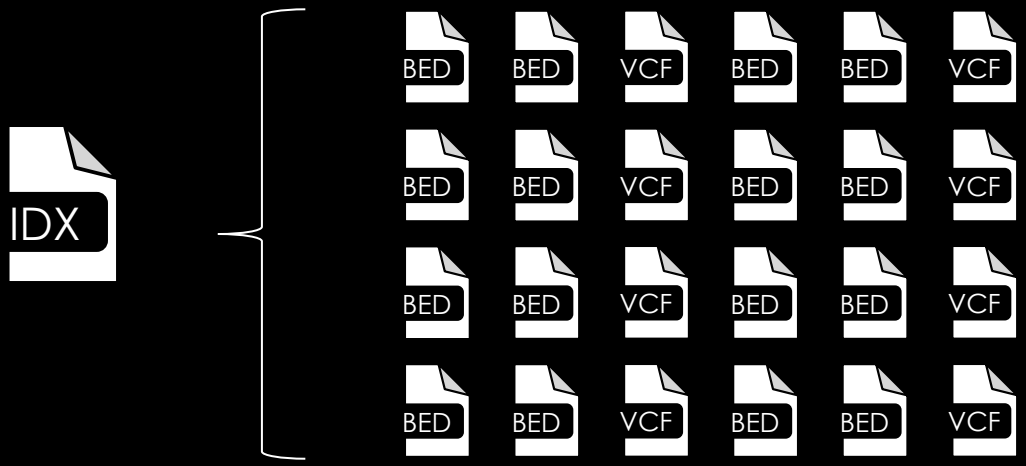


UCSC

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

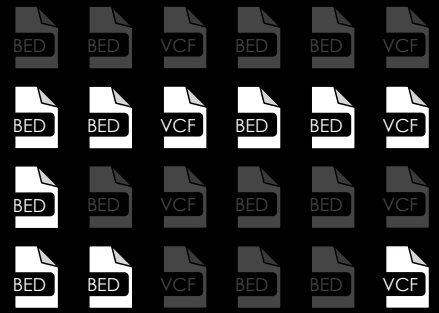
```
chr1:52000-53000
```


```
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
```



GIGGLE

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

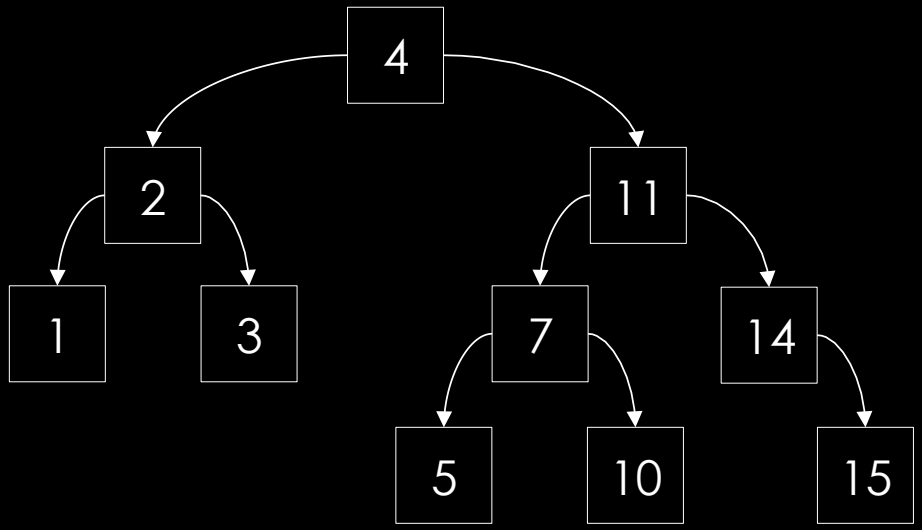
In this paper, we describe a new indexing technique, the *time index*, for improving the performance of certain classes of temporal queries. The time index can be used to retrieve versions of objects that are valid during a specific time period. It supports the processing of the temporal WHEN operator 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.

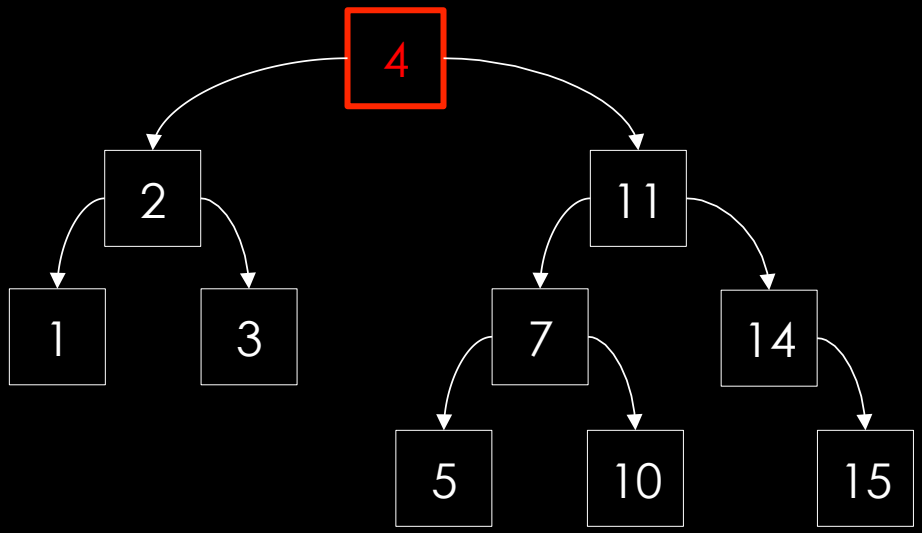
The storage techniques for temporal data proposed in [AS88, Lum84] index or link the versions of each individual object separately. In order to retrieve object versions that are valid during a certain time period, it is necessary to first locate the first (current) version of each object, and then search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

VLDB 1990

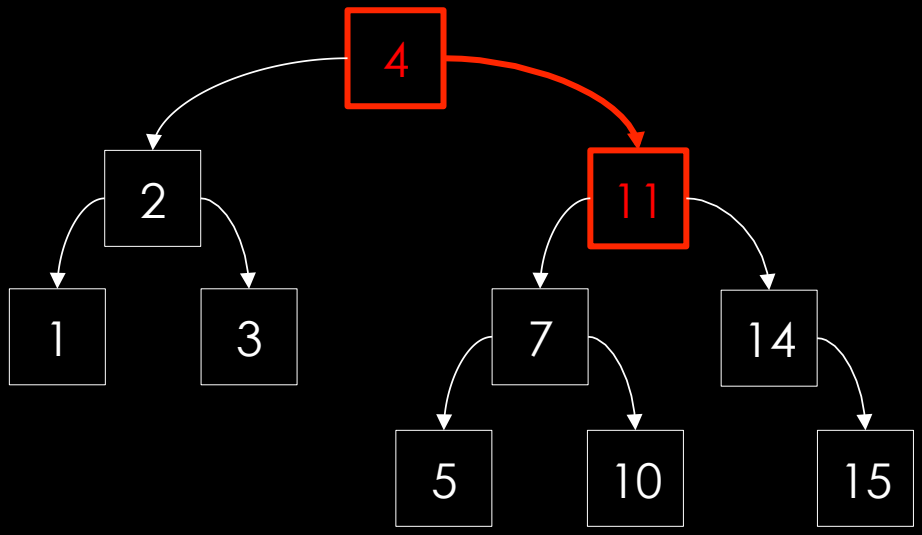
Binary Tree



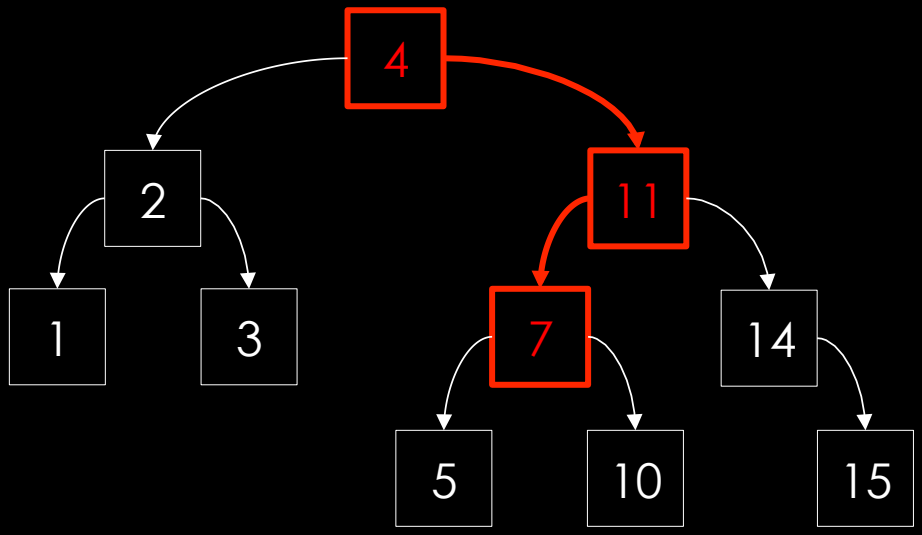
Binary Tree



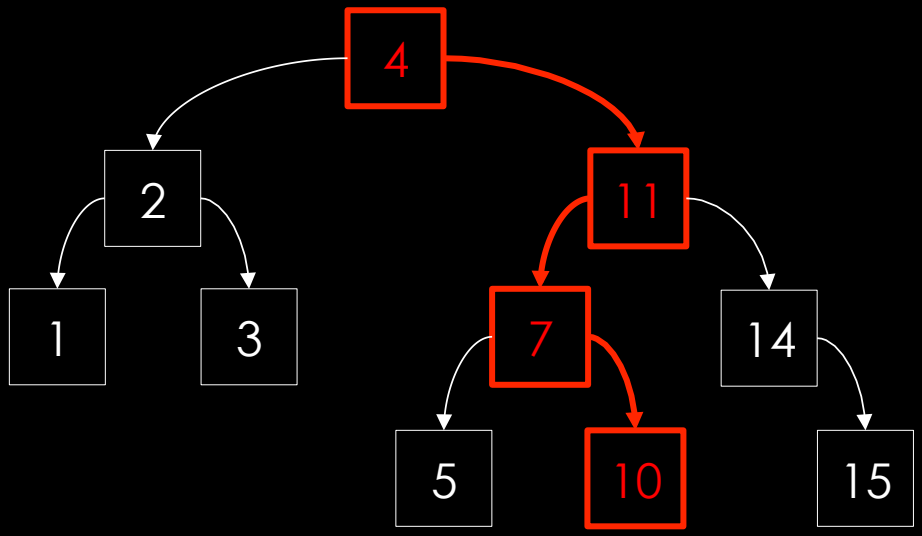
Binary Tree

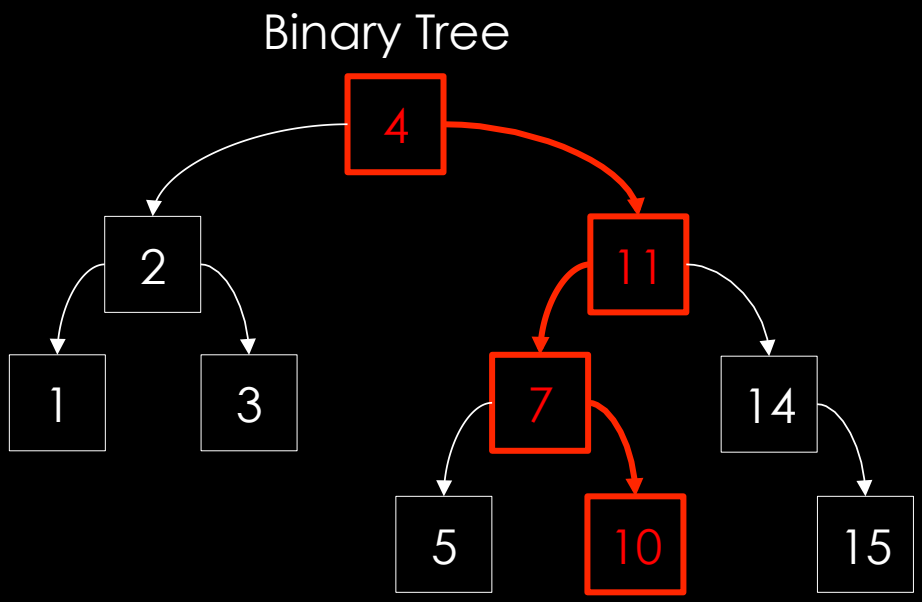


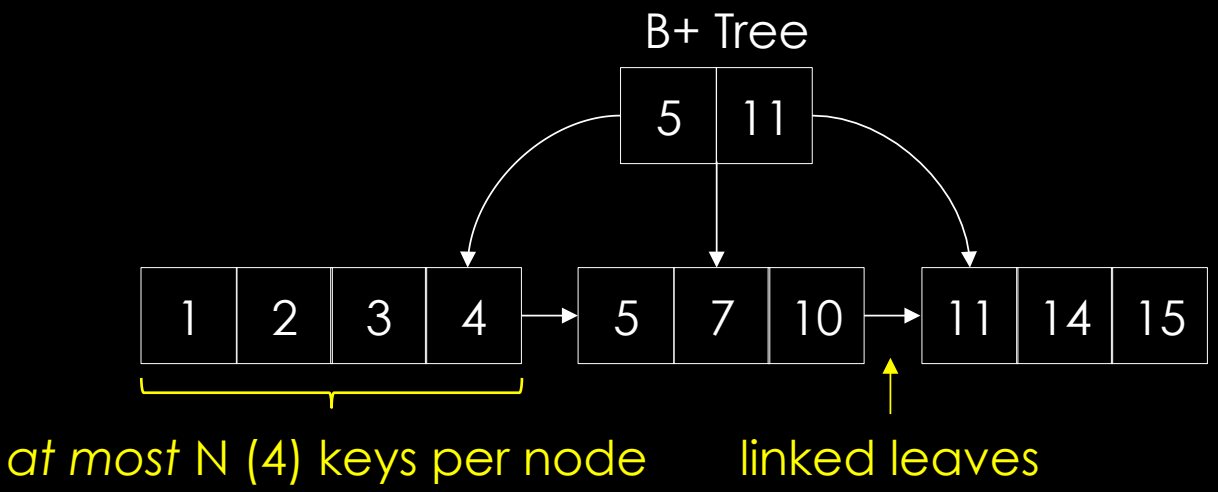
Binary Tree

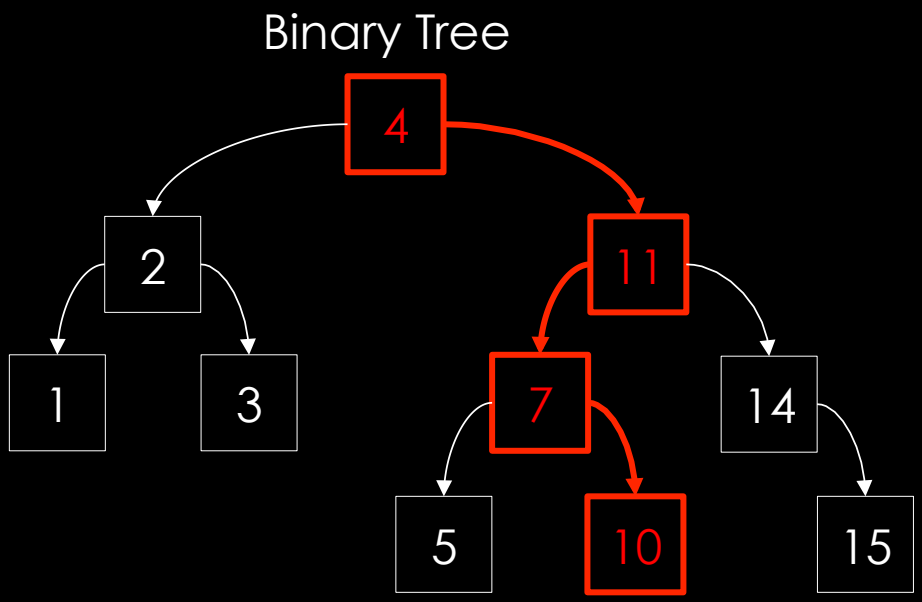


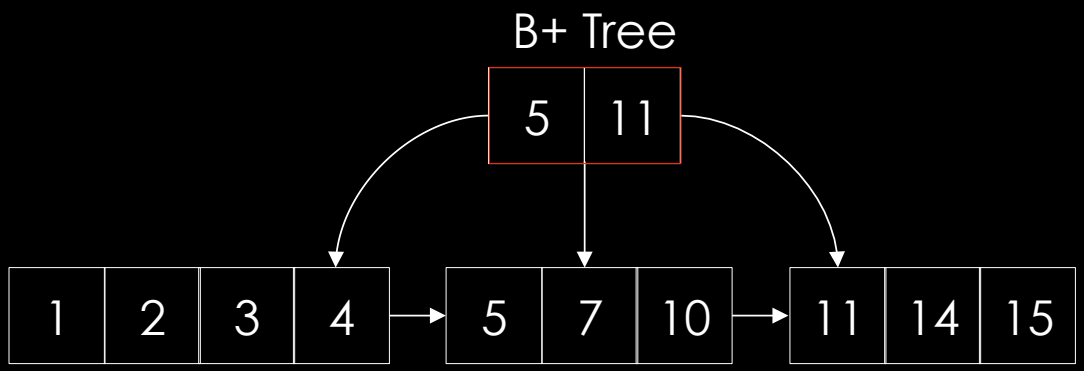
Binary Tree

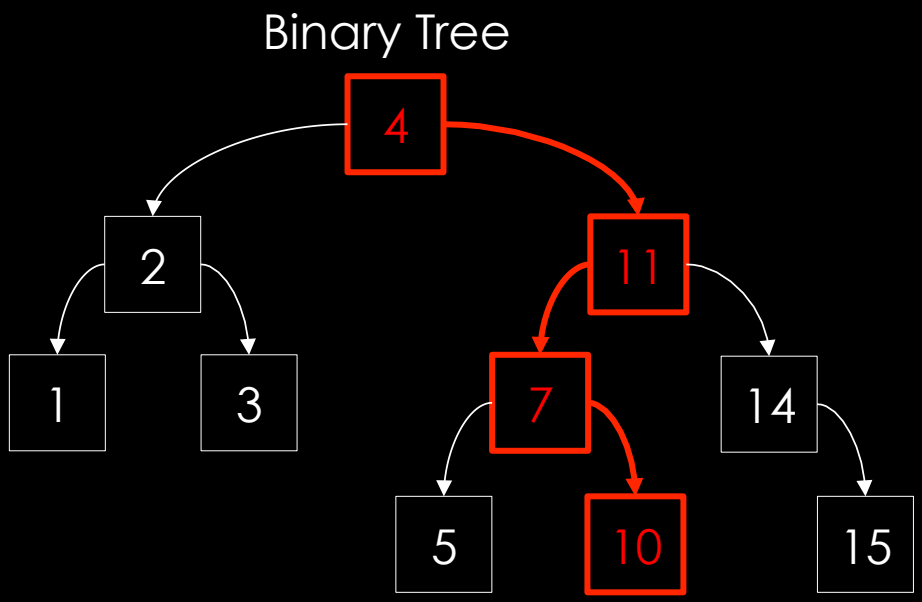


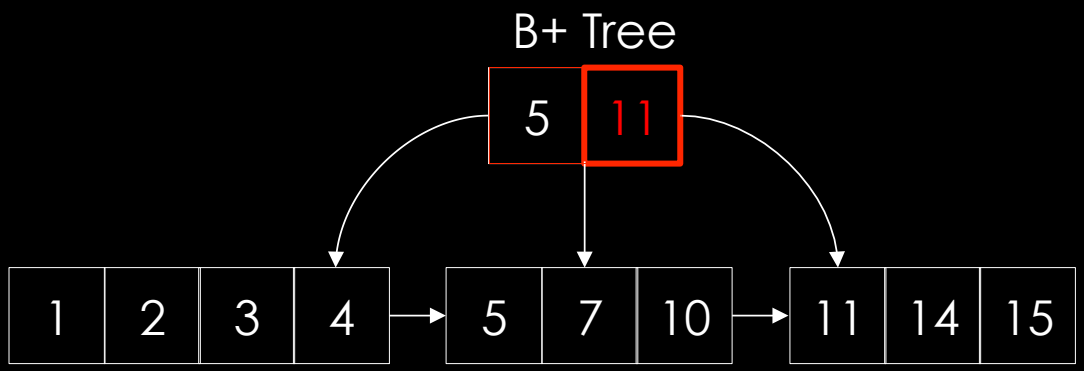

Binary Tree
4
2
11
1
3
7
14
5
10
15

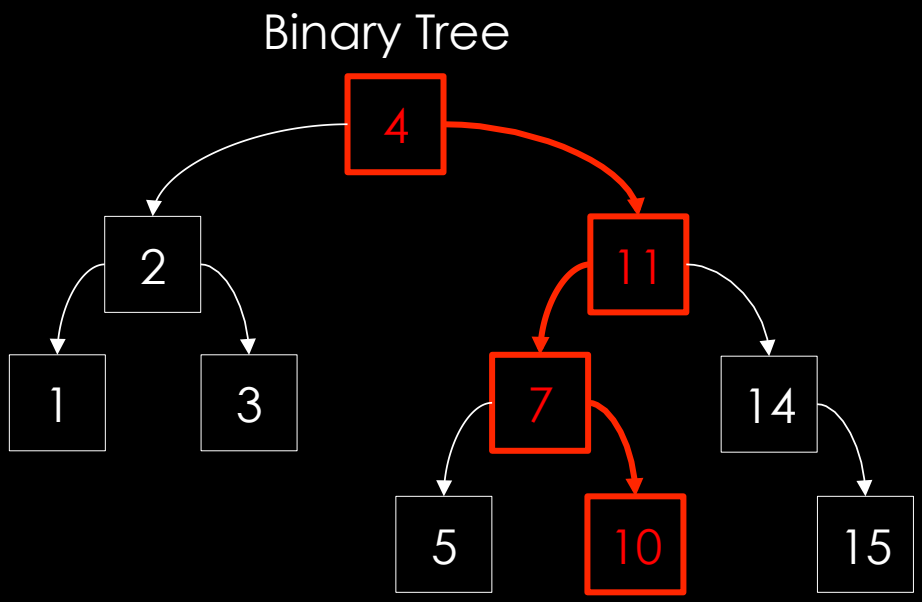

B+ Tree
5
11
1
2
3
4
5
7
10
11
14
15
at most N (4) keys per node
linked leaves

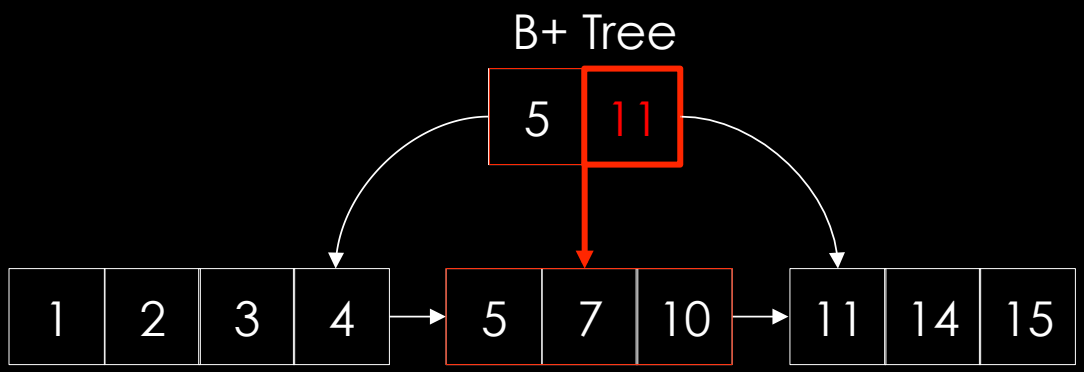

Binary Tree
4
2
11
1
3
7
14
5
10
15

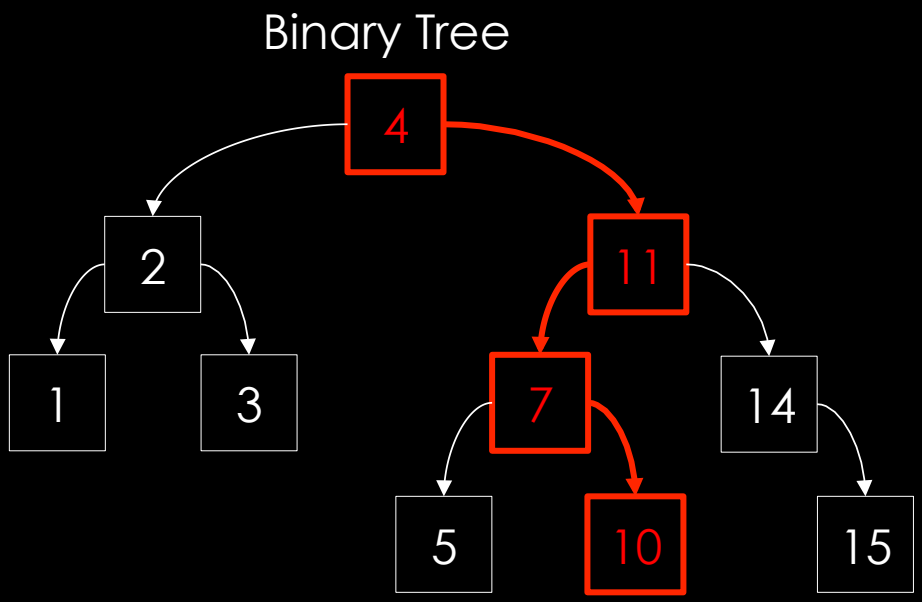

B+ Tree
5
11
1
2
3
4
5
7
10
11
14
15

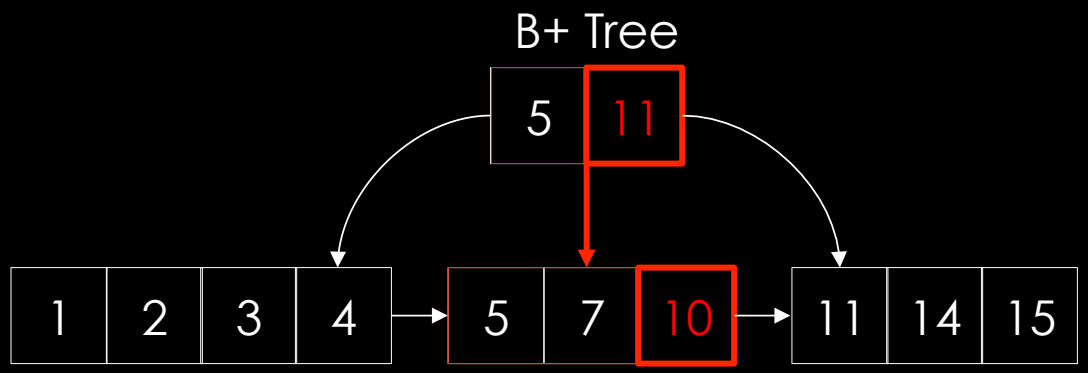

Binary Tree
4
2
11
1
3
7
14
5
10
15

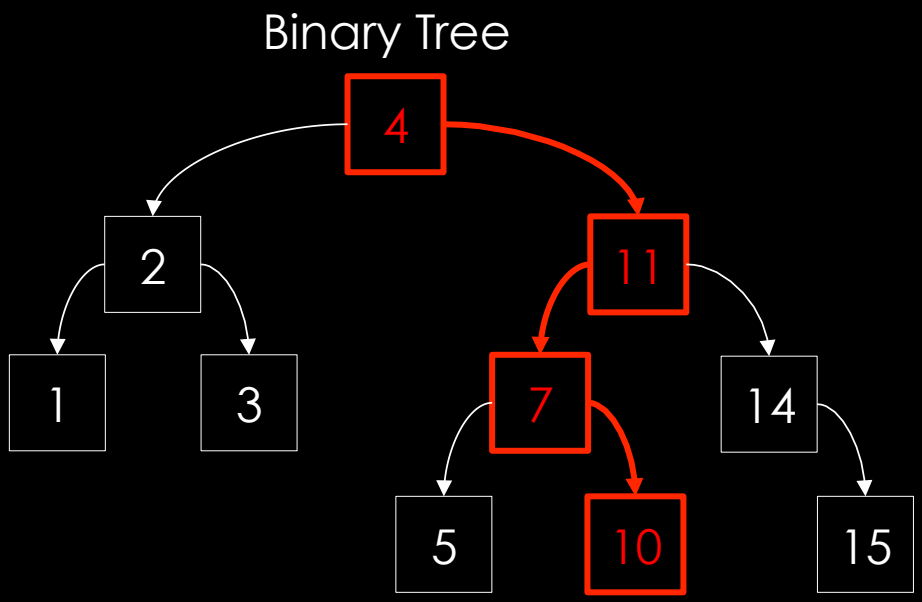

B+ Tree
5
11
1
2
3
4
5
7
10
11
14
15


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

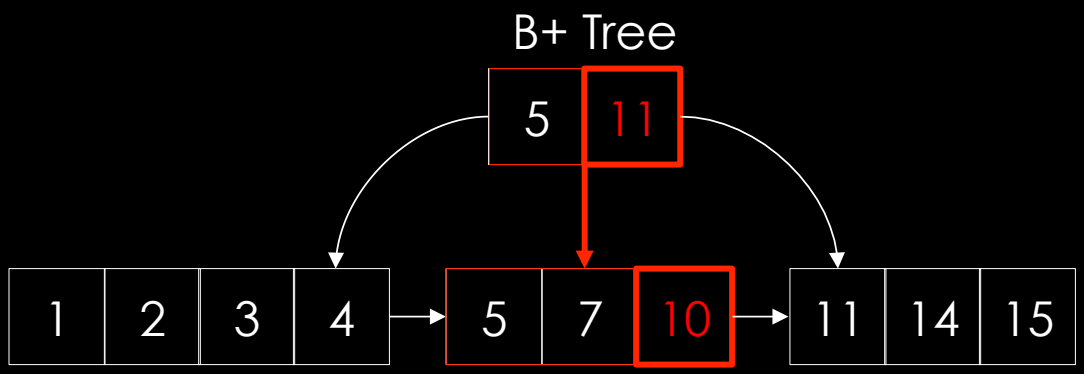

B+ Tree
5
11
1
2
3
4
5
7
10
11
14
15


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

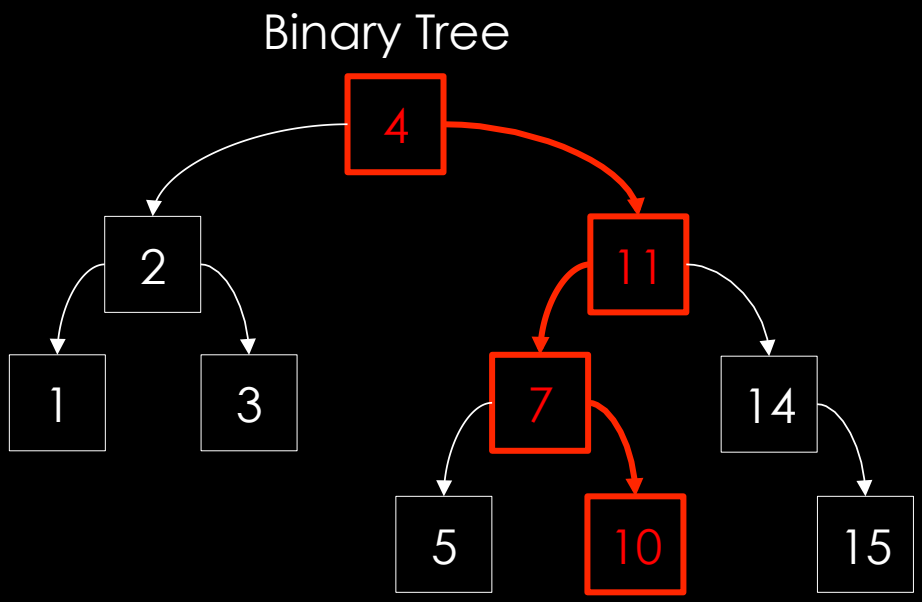

B+ Tree
5
11
1
2
3
4
5
7
10
11
14
15

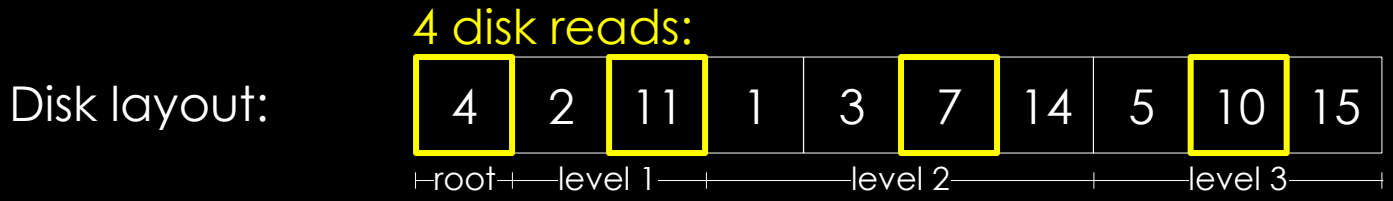


Disk layout:



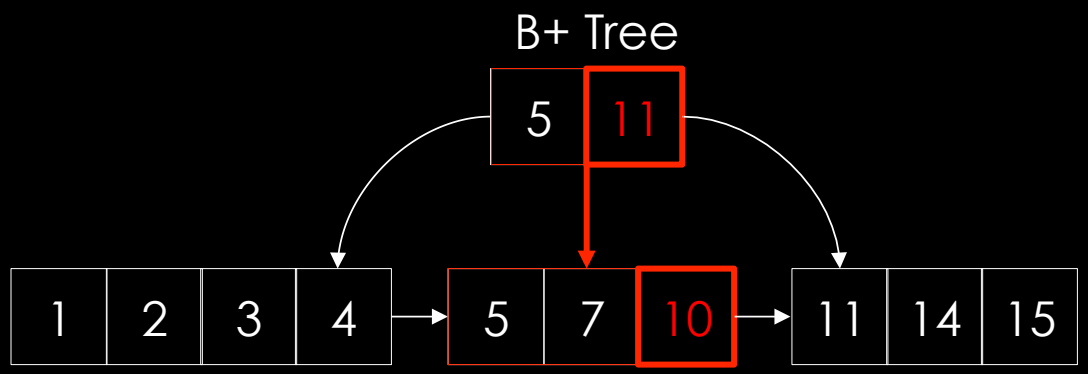
Disk layout:

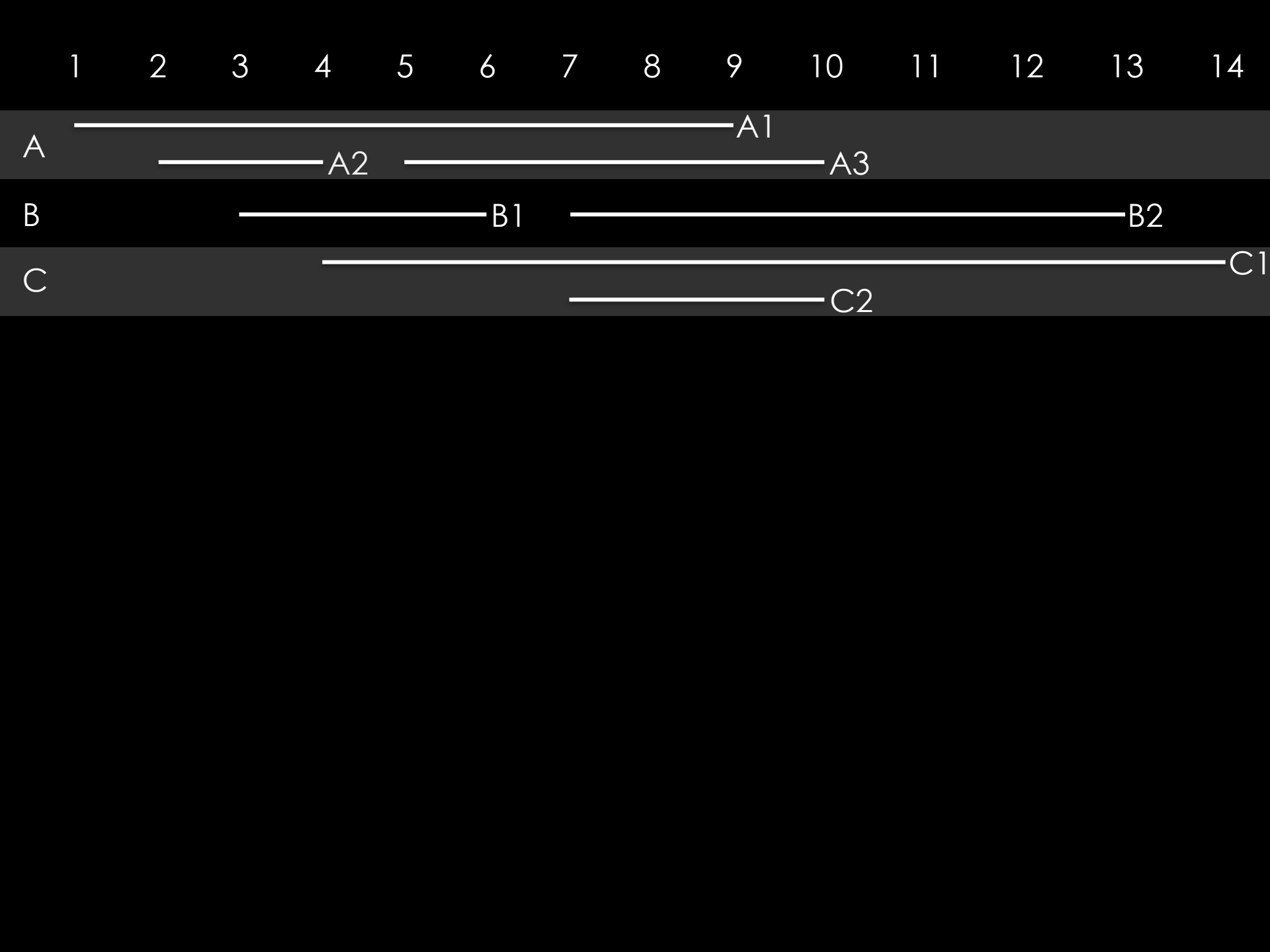


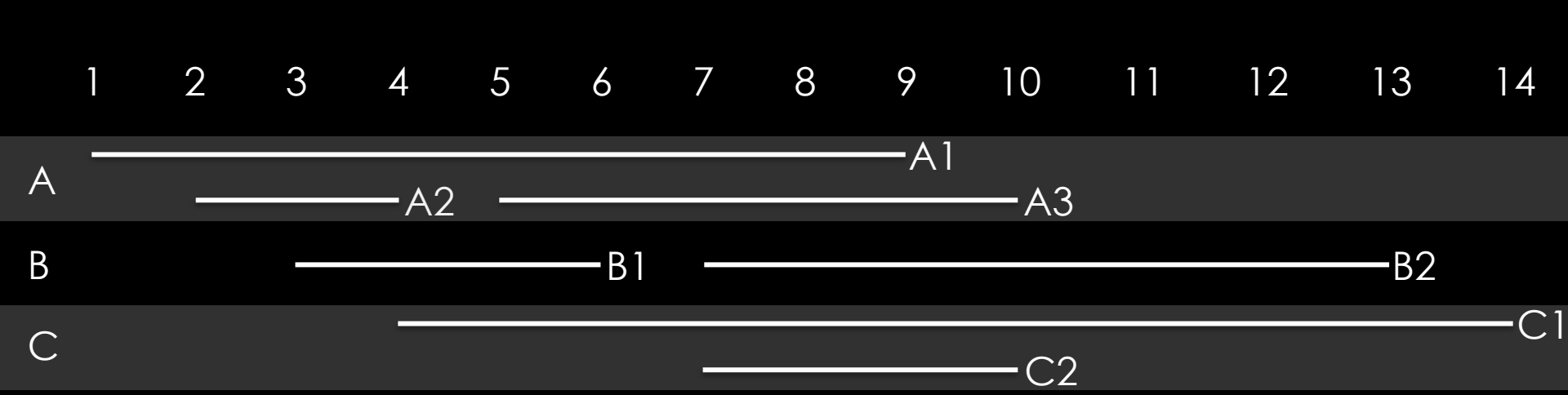


SSD disk speed
100,000 IOPS

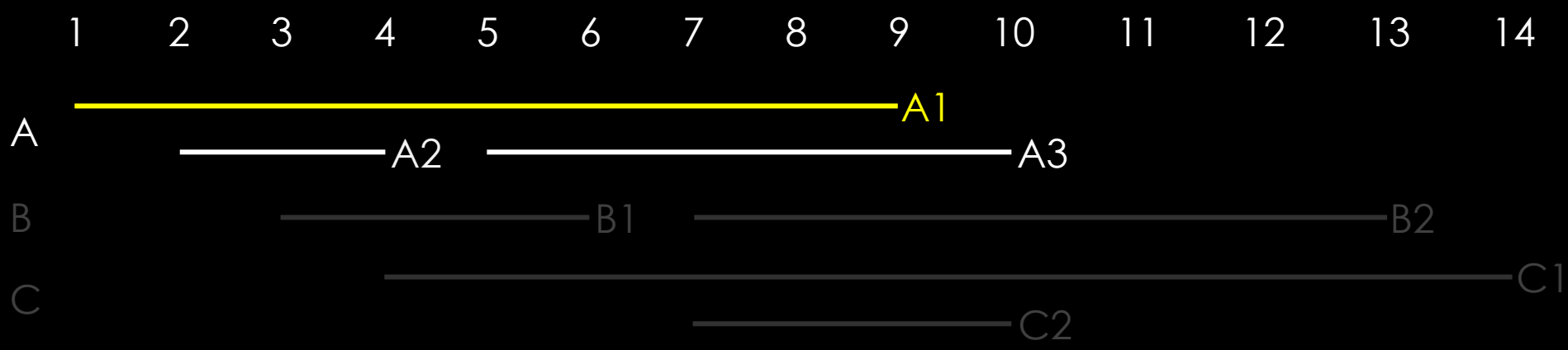
~2,000,000X
slower than CPU



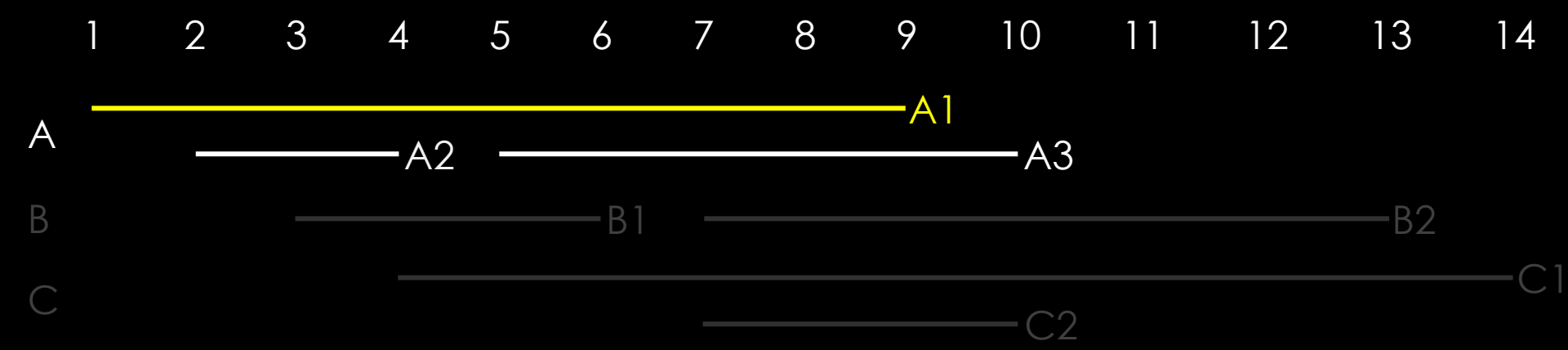

1
2
3
4
5
6
7
8
9
10
11
12
13
14
A
A1
A2
A3
B
B1
B2
C
C1
C2



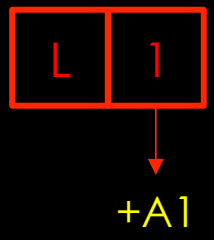

```
index(X, start, end)
 insert +X at start
 insert -X at end + 1
 append X at spanned L's
```



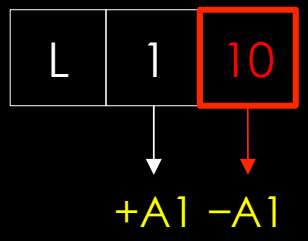

```
index(A1, 1, 9)
MAX_KEYS = 4
```

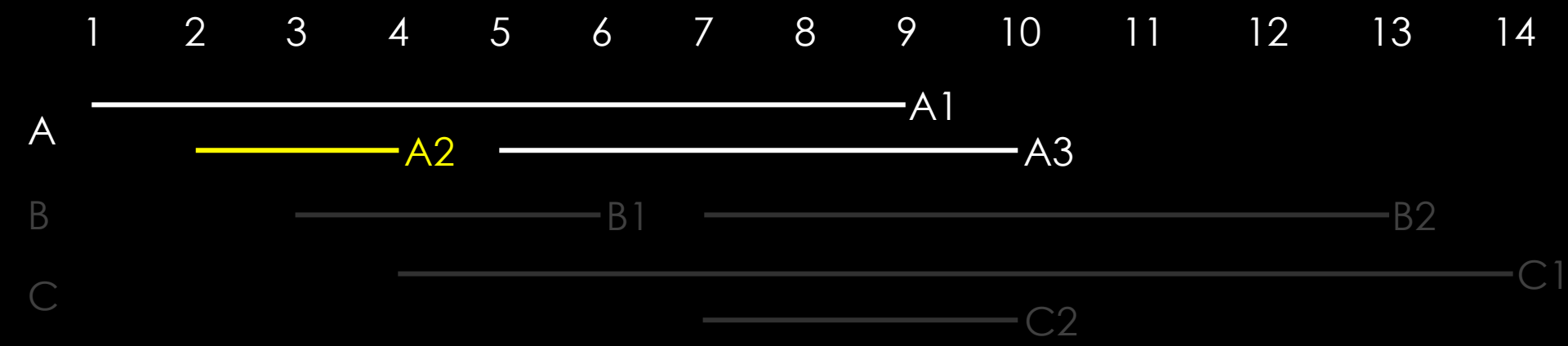



```
index(A1, 1, 9)
```

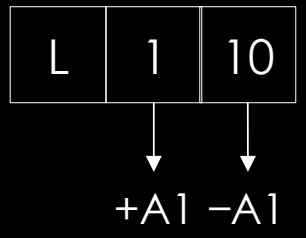


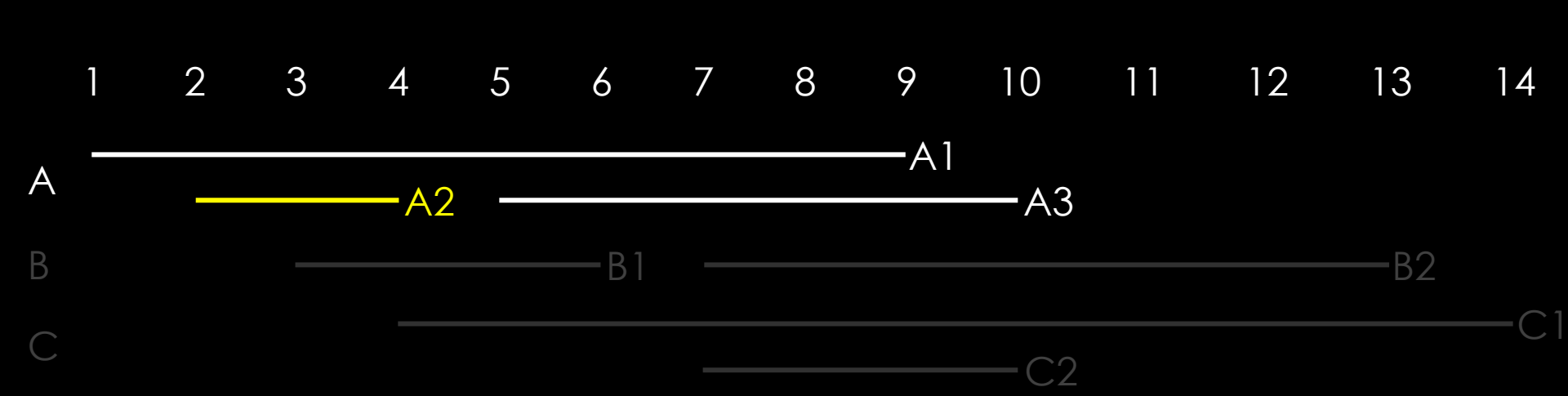

```
index(A1, 1, 9)
```



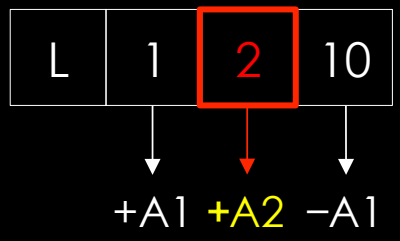


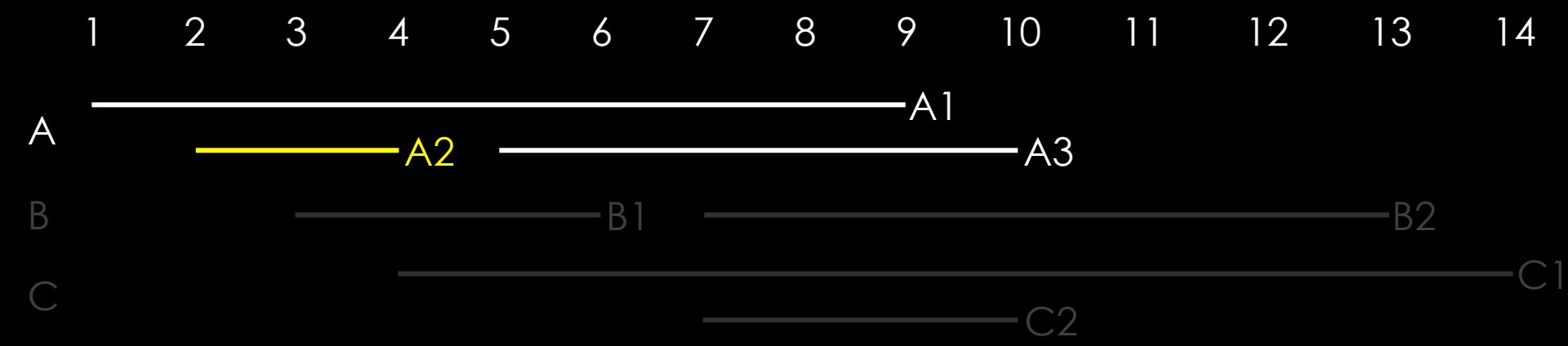

```
index(A2, 2, 4)
```



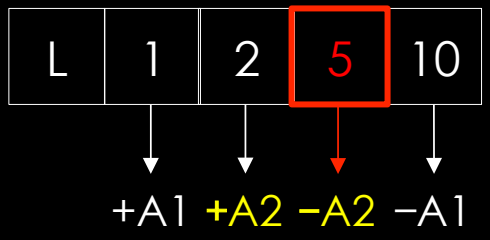


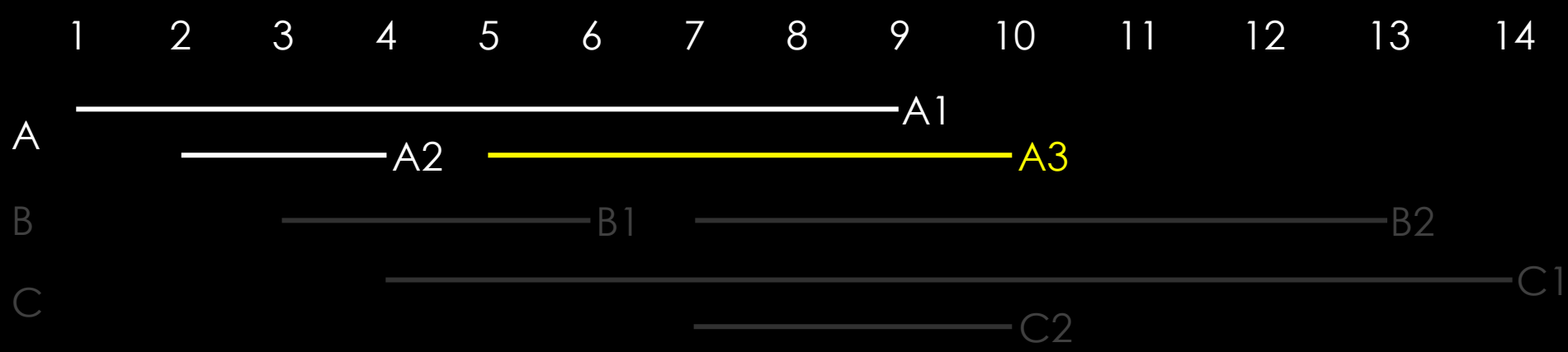

```
index(A2, 2, 4)
```



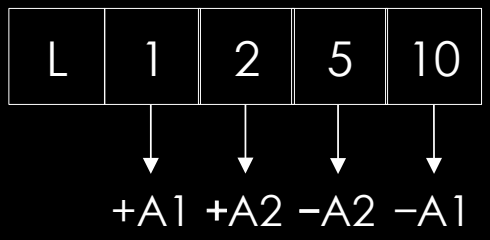

1 2 3 4 5 6 7 8 9 10 11 12 13 14
A
A1
A2
A3
B
B1
B2
C
C1
C2

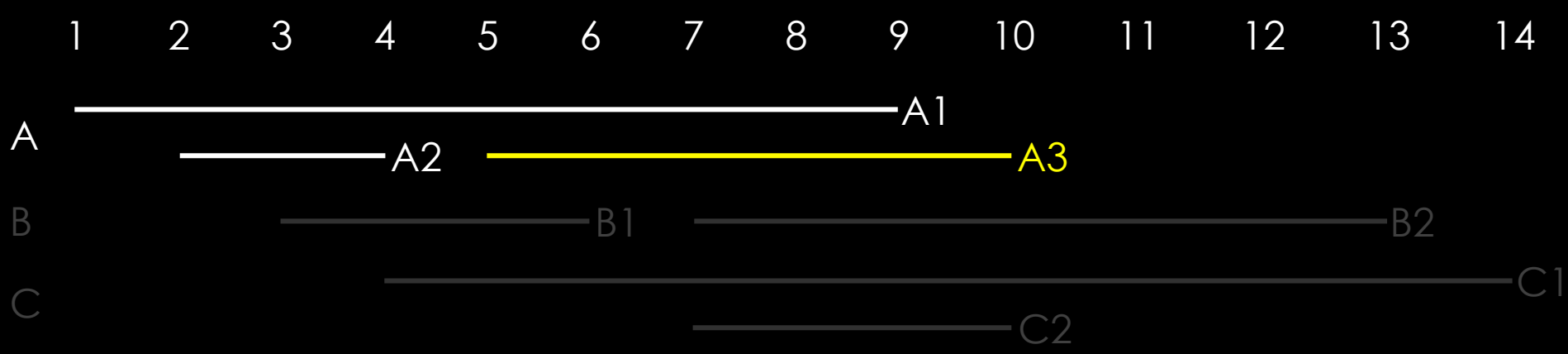
`index(A2, 2, 4)`


L
1
2
5
10
+A1 +A2 −A2 −A1

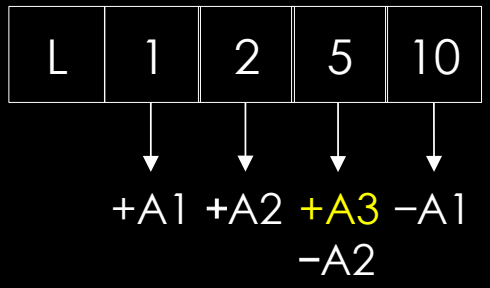

1 2 3 4 5 6 7 8 9 10 11 12 13 14
A
A1
A2
A3
B
B1
B2
C
C1
C2

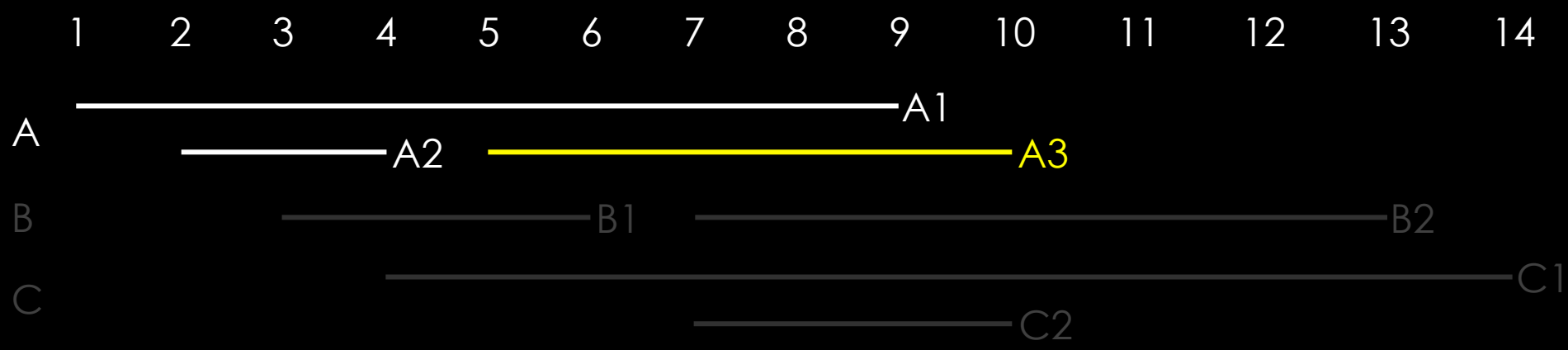
```
index(A3, 5, 10)
```


L
1
2
5
10
+A1 +A2 −A2 −A1

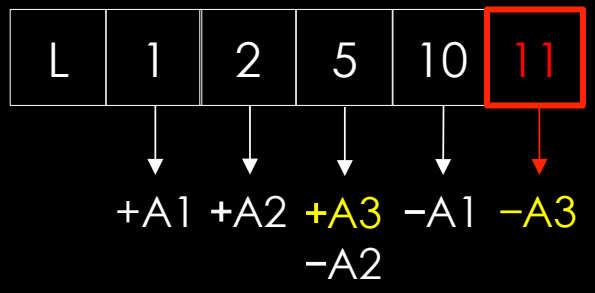

1 2 3 4 5 6 7 8 9 10 11 12 13 14
A
A1
A2
A3
B
B1
B2
C
C1
C2

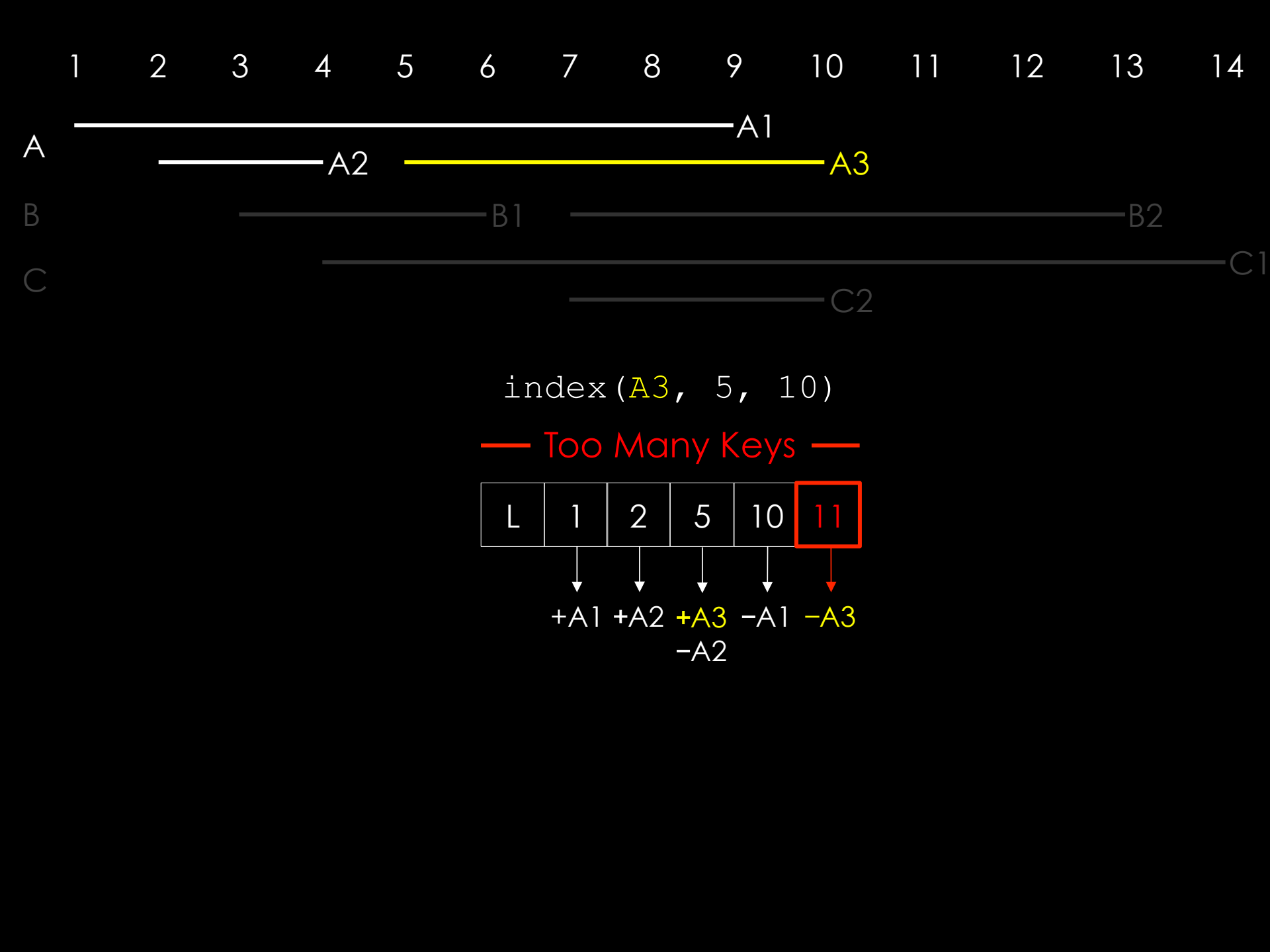
```
index(A3, 5, 10)
```

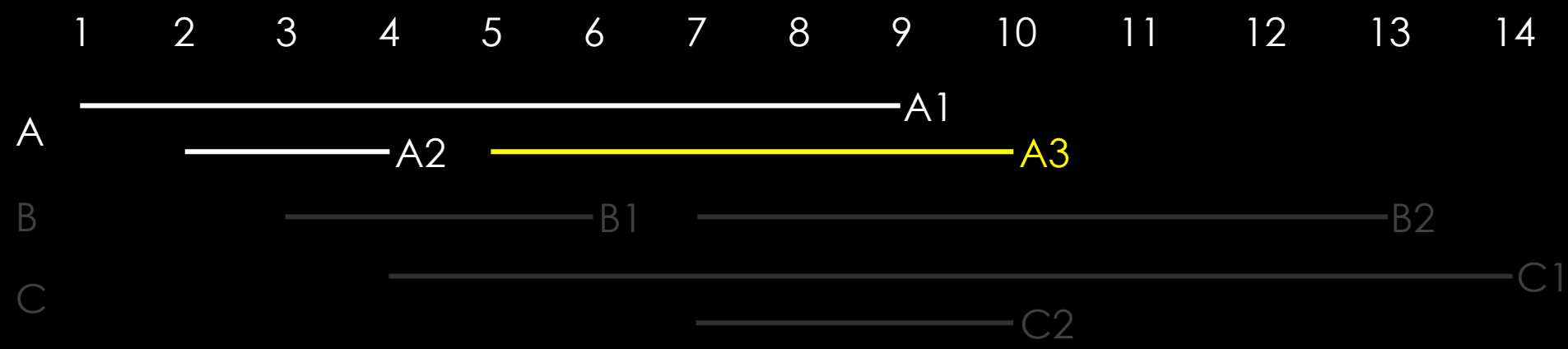

L
1
2
5
10
+A1
+A2
+A3
−A1
−A2


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

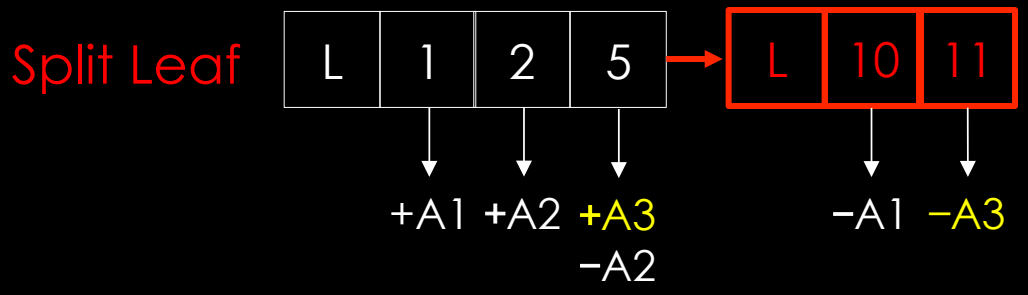
```
index(A3, 5, 10)
```

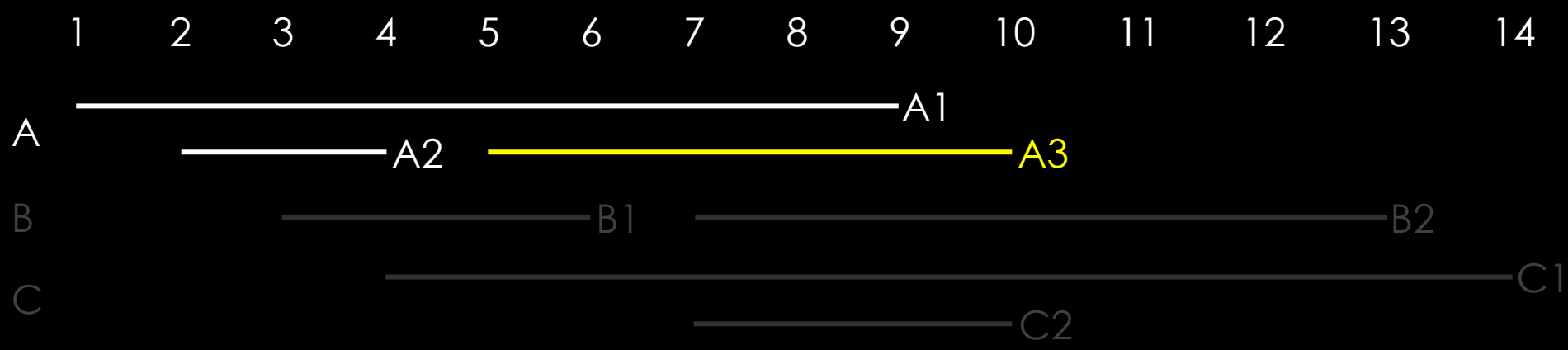

L
1
2
5
10
11
+A1 +A2 +A3 −A1 −A3
−A2


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

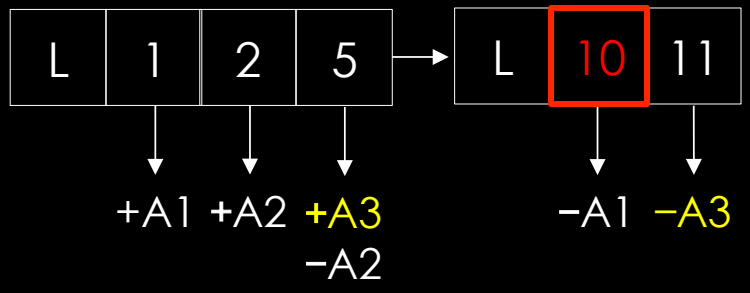
```
index(A3, 5, 10)
```

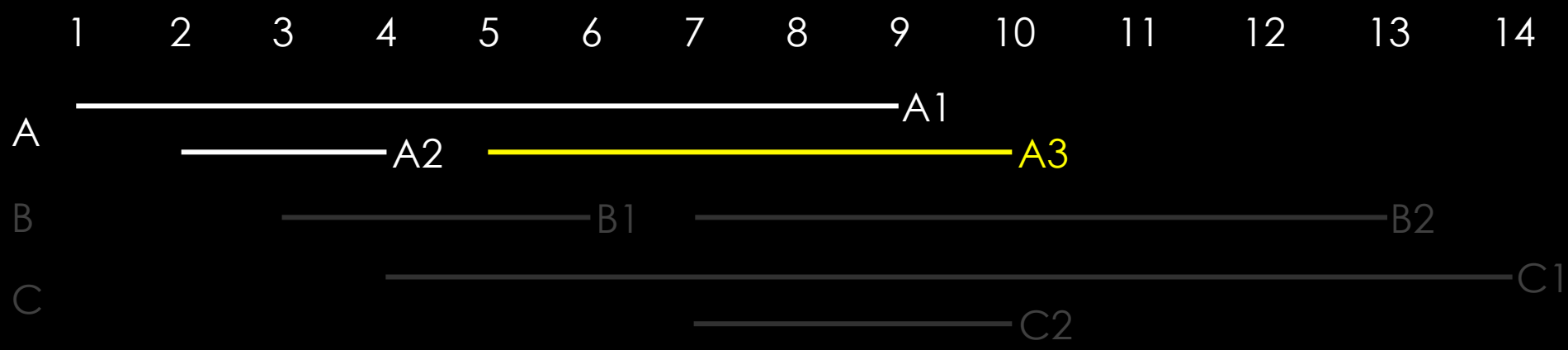

Split Leaf
L
1
2
5
L
10
11
+A1 +A2 +A3
−A2
−A1 −A3




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

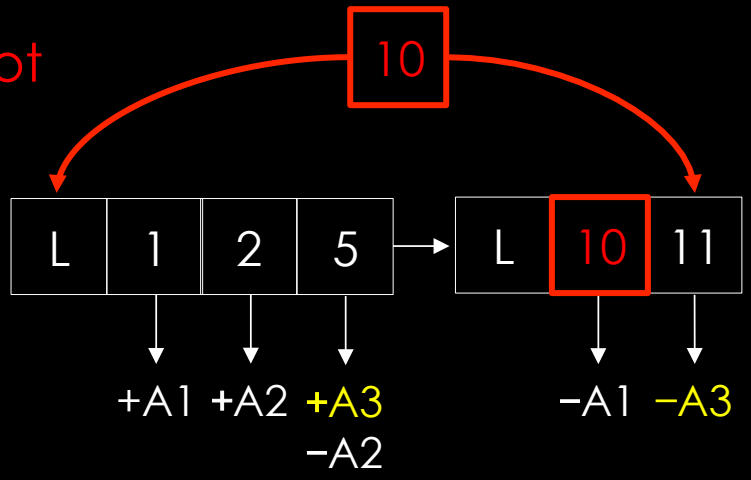
Promote a New Root

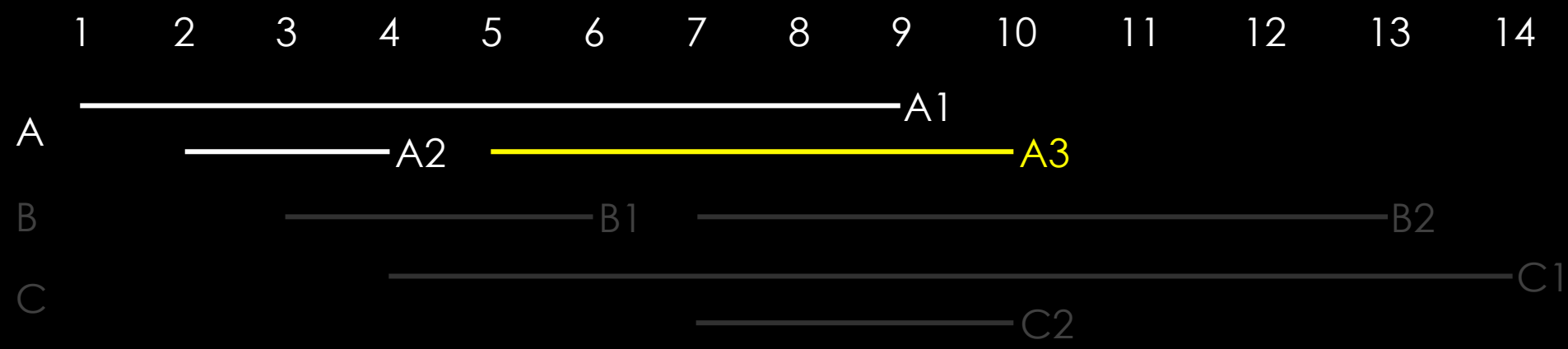





```
index(A3, 3, 6)
```

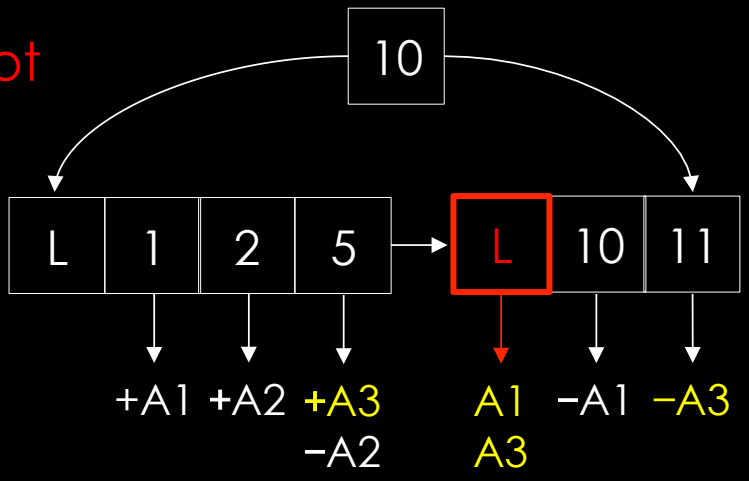
Promote a New Root

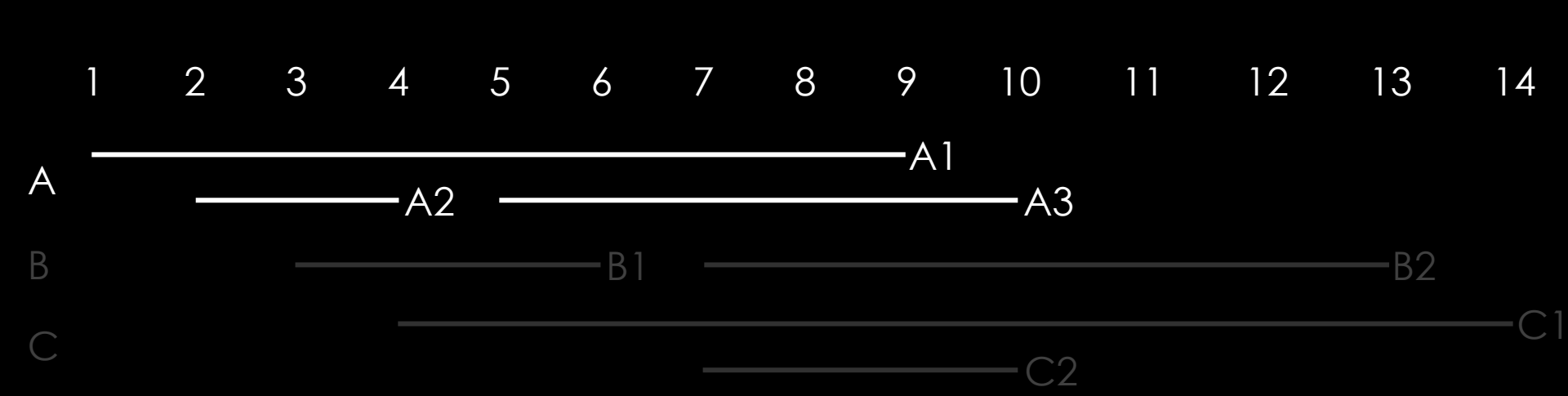


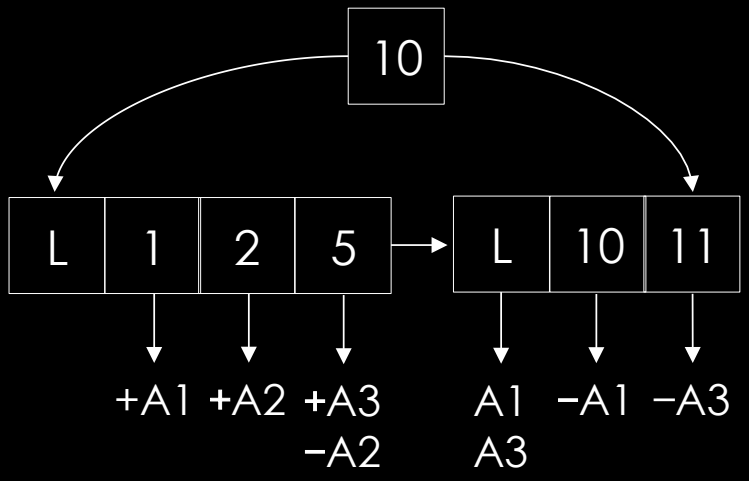


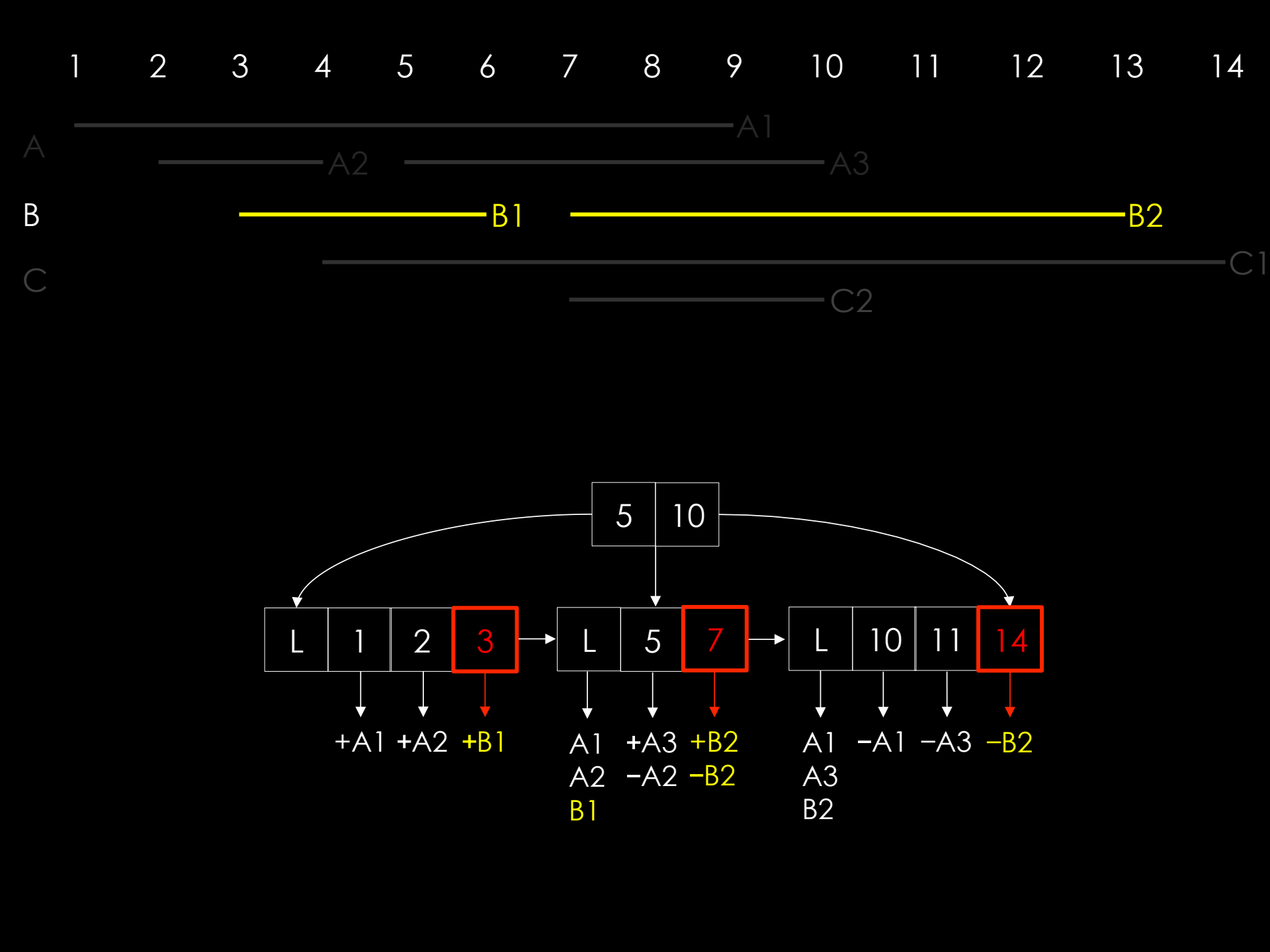

```
index(A3, 3, 6)
```

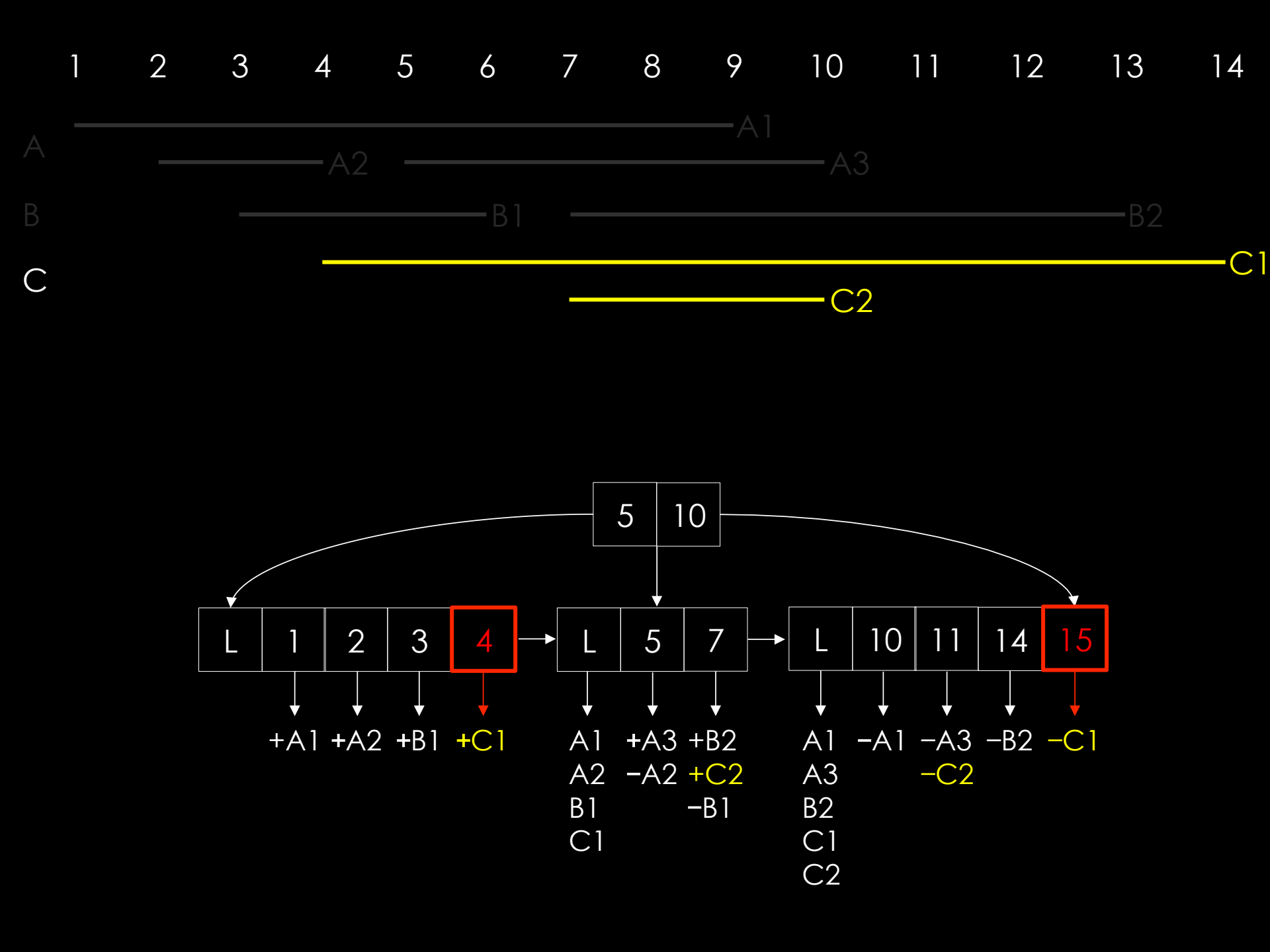
Promote a New Root

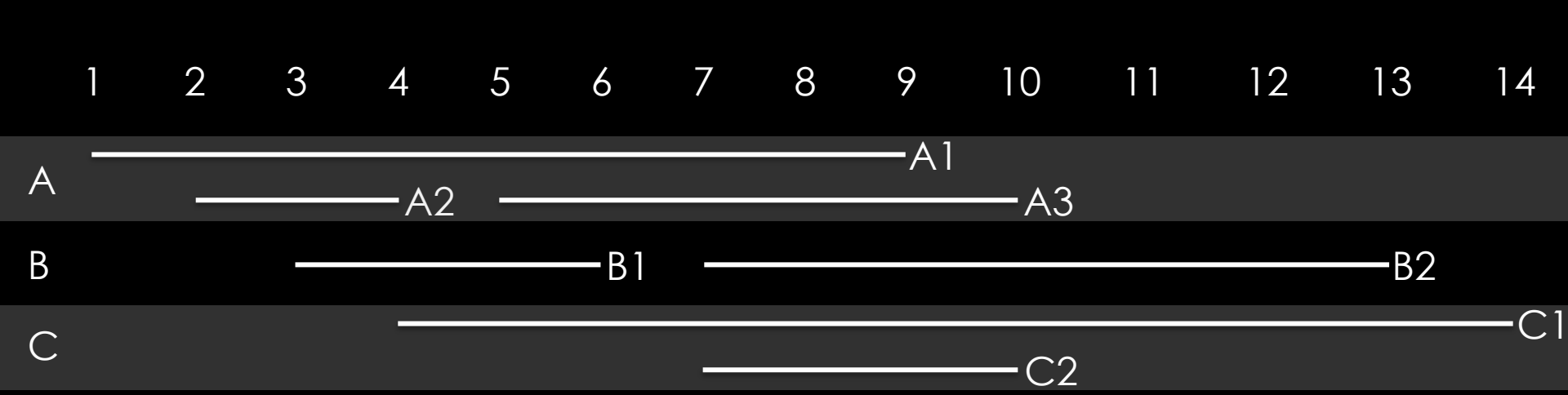



1 2 3 4 5 6 7 8 9 10 11 12 13 14
A
A1
A2
A3
B
B1
B2
C
C1
C2

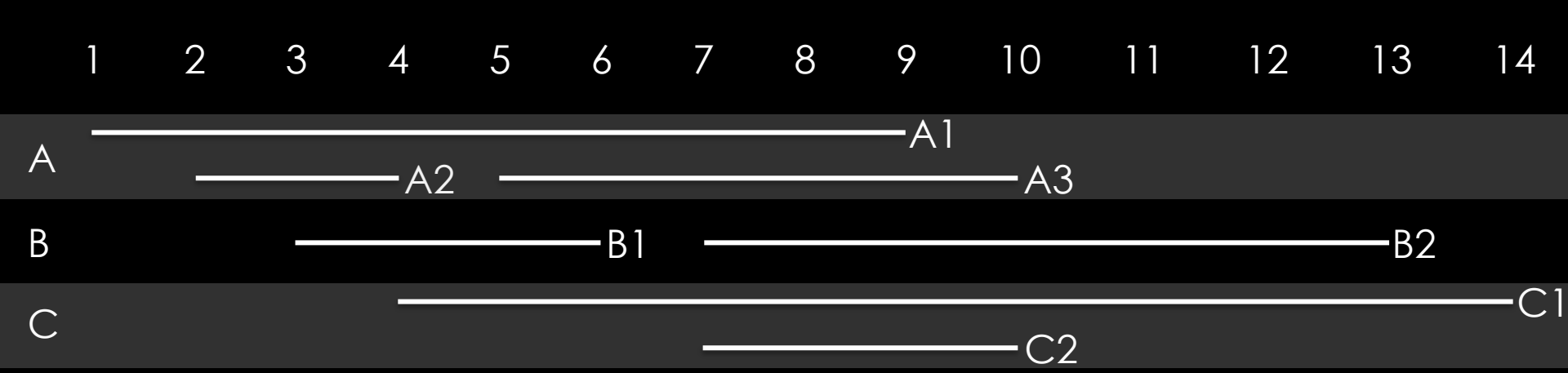

10
L 1 2 5
L 10 11
+A1 +A2 +A3
−A2
A1
A3
−A1 −A3

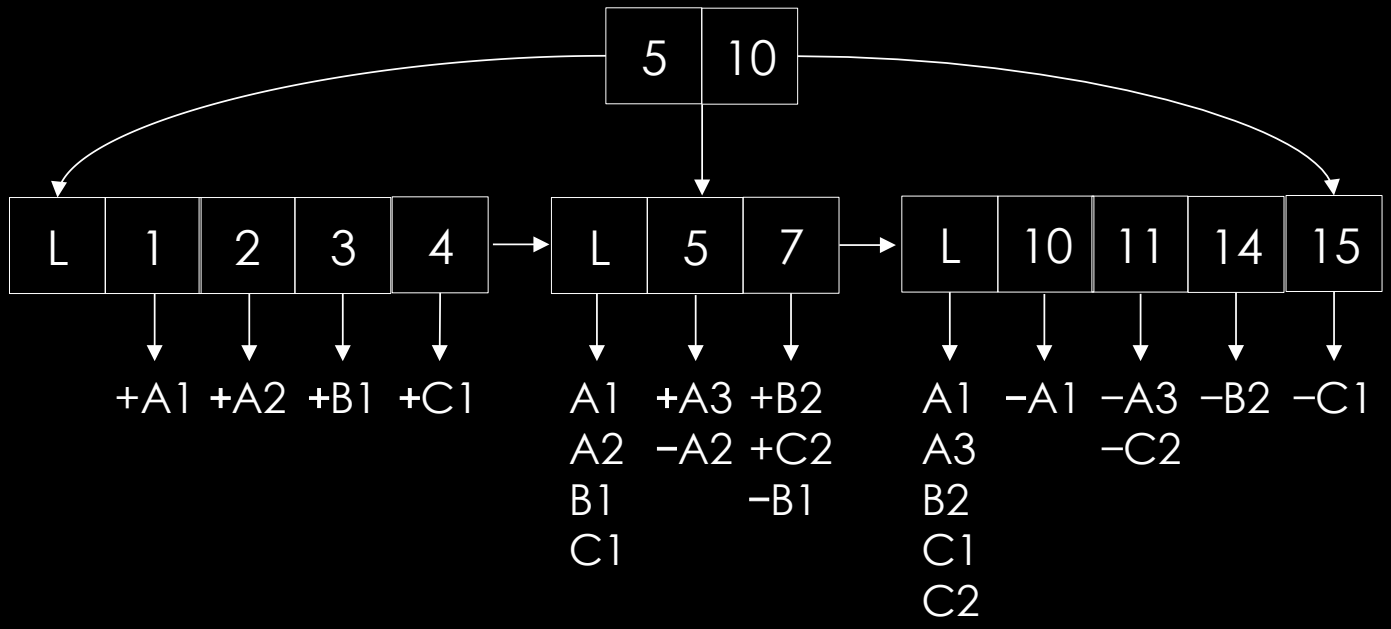

1 2 3 4 5 6 7 8 9 10 11 12 13 14
A
A1
A2
A3
B
B1
B2
C
C1
C2
5 10
L 1 2 3
L 5 7
L 10 11 14
+A1 +A2 +B1
A1 A2 B1
+A3 −A2 +B2 −B2
A1 A3 B2
−A1 −A3 −B2

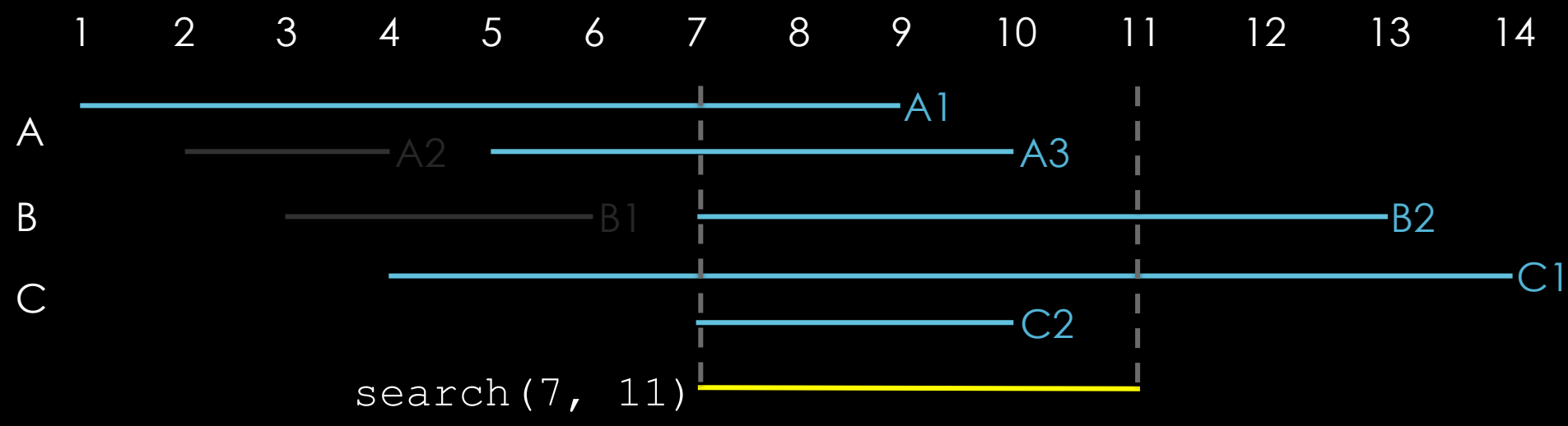

1 2 3 4 5 6 7 8 9 10 11 12 13 14
A
A1
A2
A3
B
B1
B2
C
C1
C2
5 10
L 1 2 3 4
L 5 7
L 10 11 14 15
+A1 +A2 +B1 +C1
A1 A2 B1 C1
+A3 −A2 +C2 −B1
A1 A3 B2 C1 C2
−A1 −A3 −C2 −B2 −C1

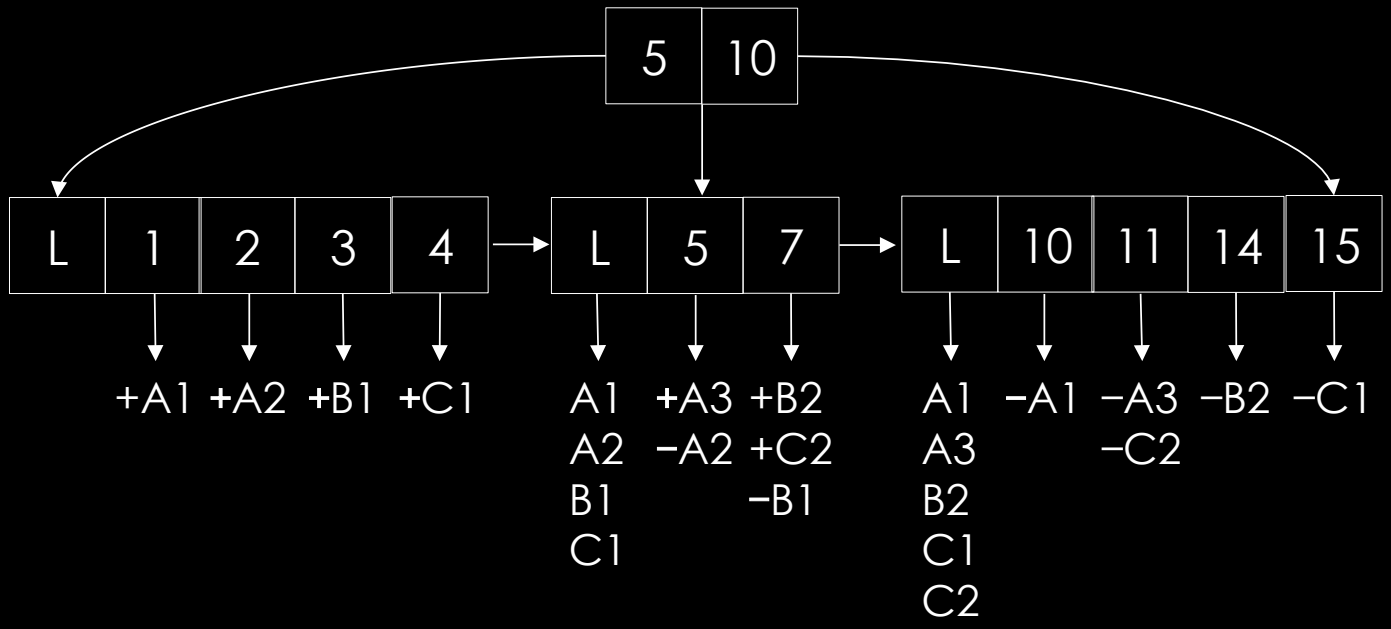


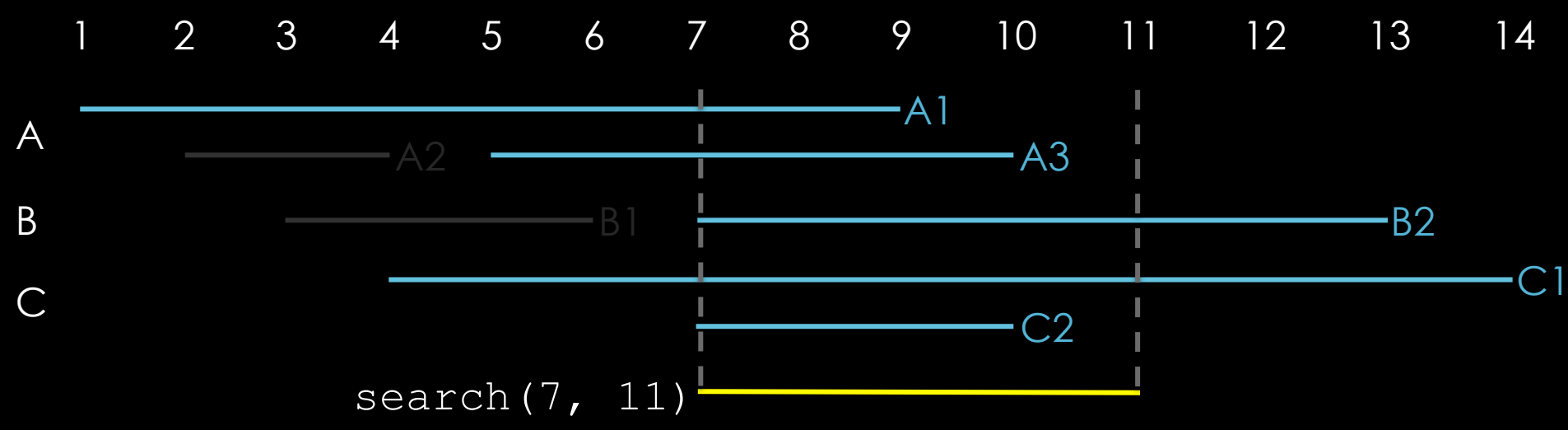

```
search(start, end)
  I = []
  from = find(start)
  to = find(end);
  I += from.leaf.L
  for node upto from:
    I += node.starts
    I -= node.ends
  for node in [from.next, to]:
    I += node.starts
```


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)

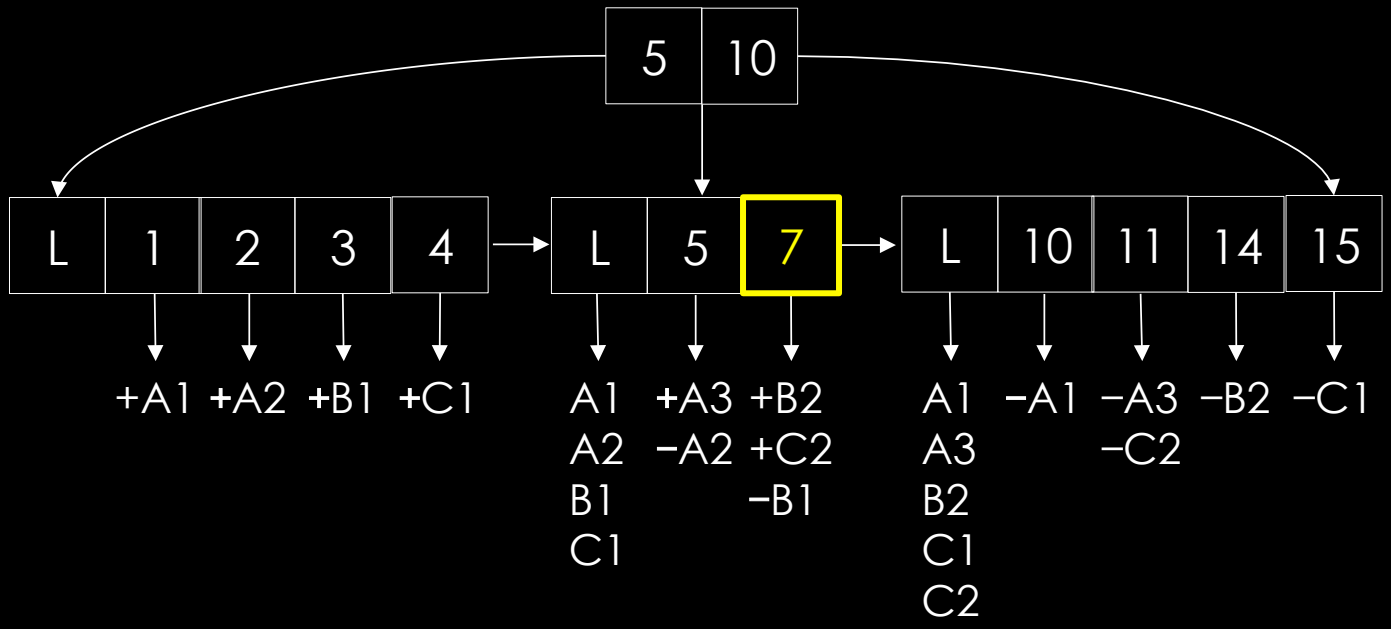

5 10
L 1 2 3 4
L 5 7
L 10 11 14 15
+A1 +A2 +B1 +C1
A1 +A3 +B2
A2 −A2 +C2
B1 −B1
C1
A1 −A1 −A3 −B2 −C1
A3 −C2
B2
C1
C2

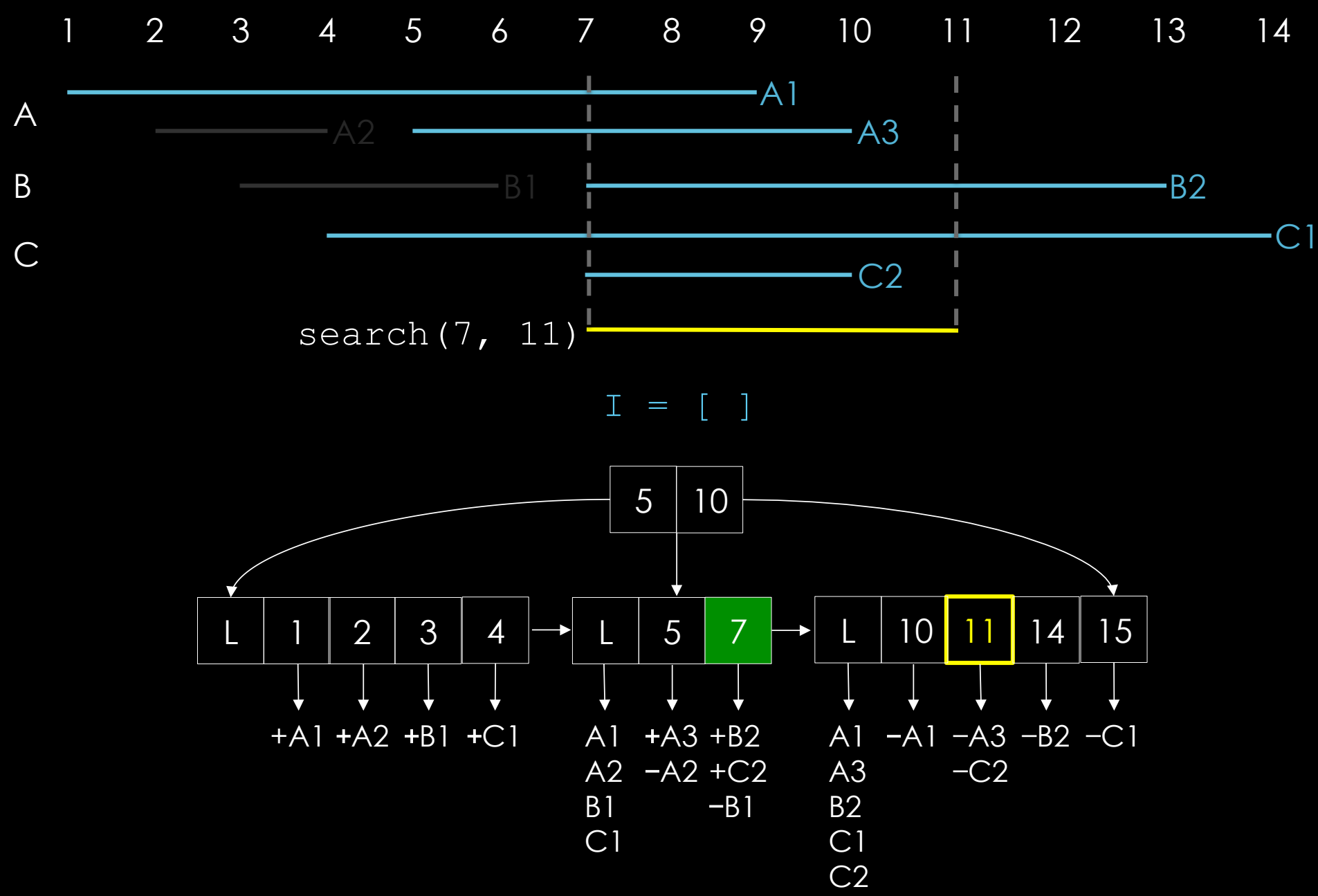

1 2 3 4 5 6 7 8 9 10 11 12 13 14
A
B
C
A1
A2
A3
B1
B2
C1
C2
search(7, 11)

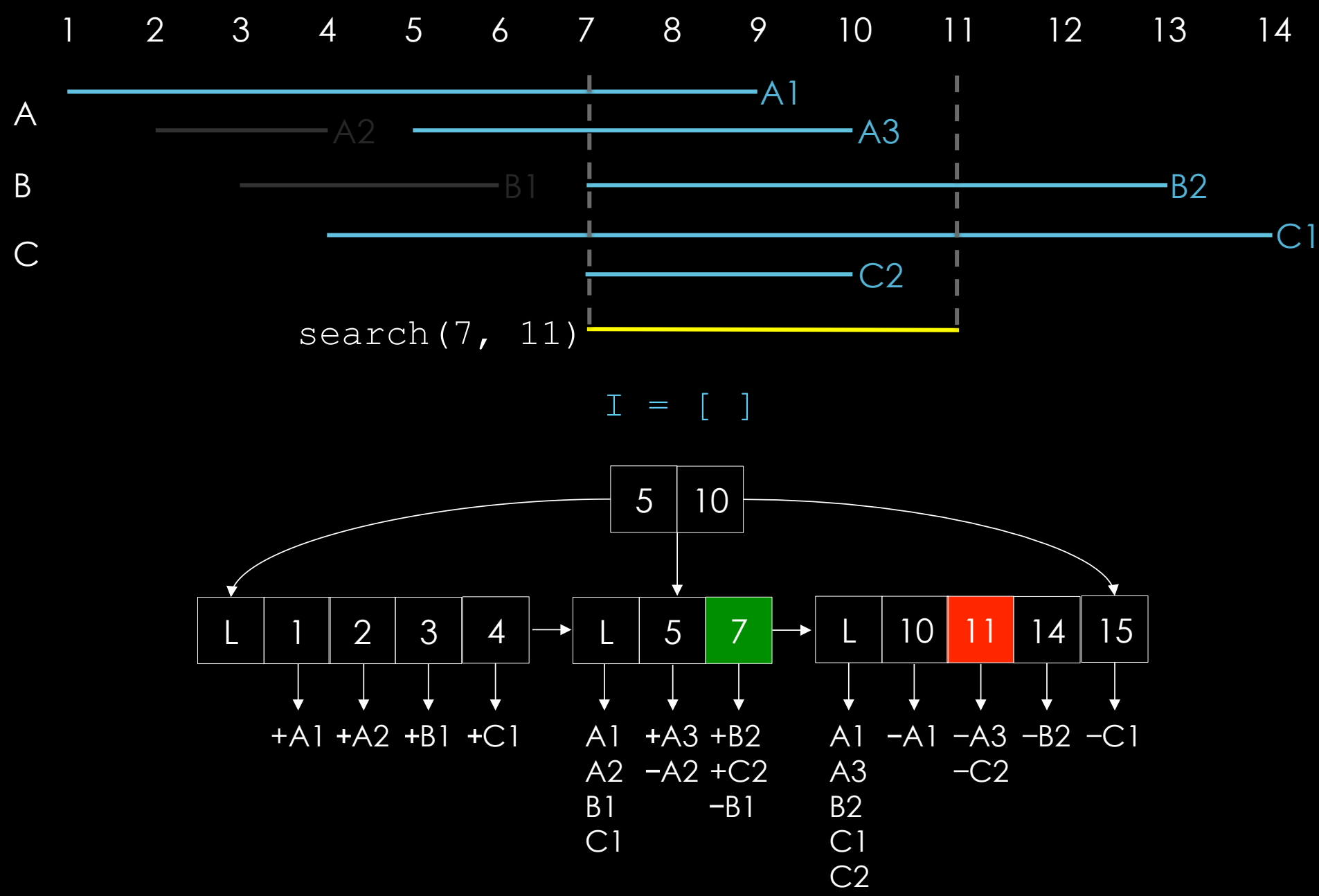

5 10
L 1 2 3 4
L 5 7
L 10 11 14 15
+A1 +A2 +B1 +C1
A1 +A3 +B2
A2 −A2 +C2
B1 −B1
C1
A1 −A1 −A3 −B2 −C1
A3 −C2
B2
C1
C2

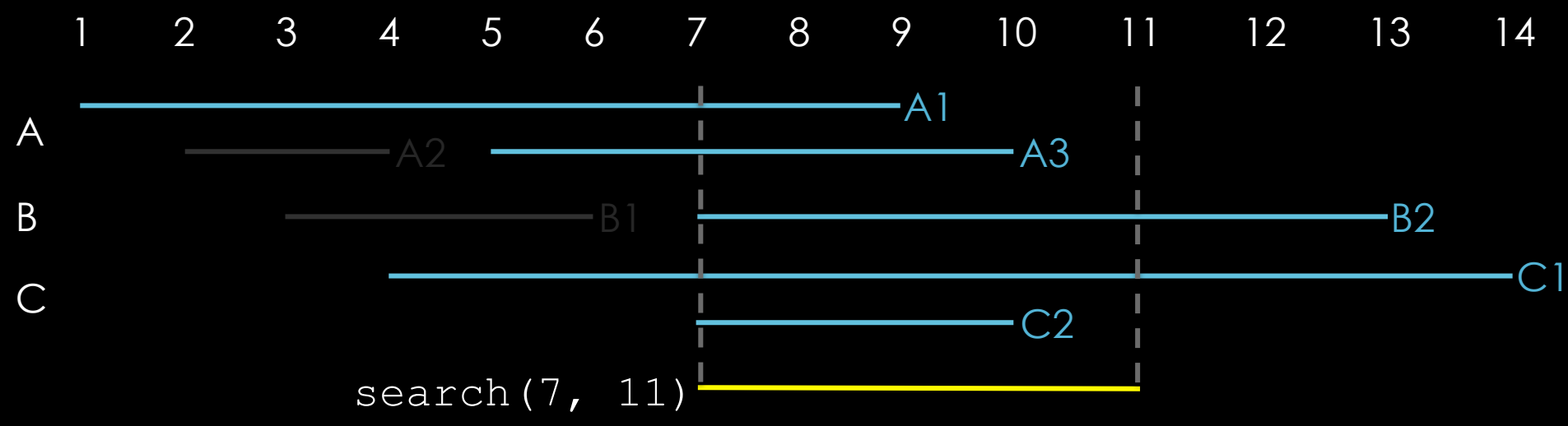

1 2 3 4 5 6 7 8 9 10 11 12 13 14
A
B
C
A1
A2
A3
B1
B2
C1
C2
search(7, 11)

I = [ ]

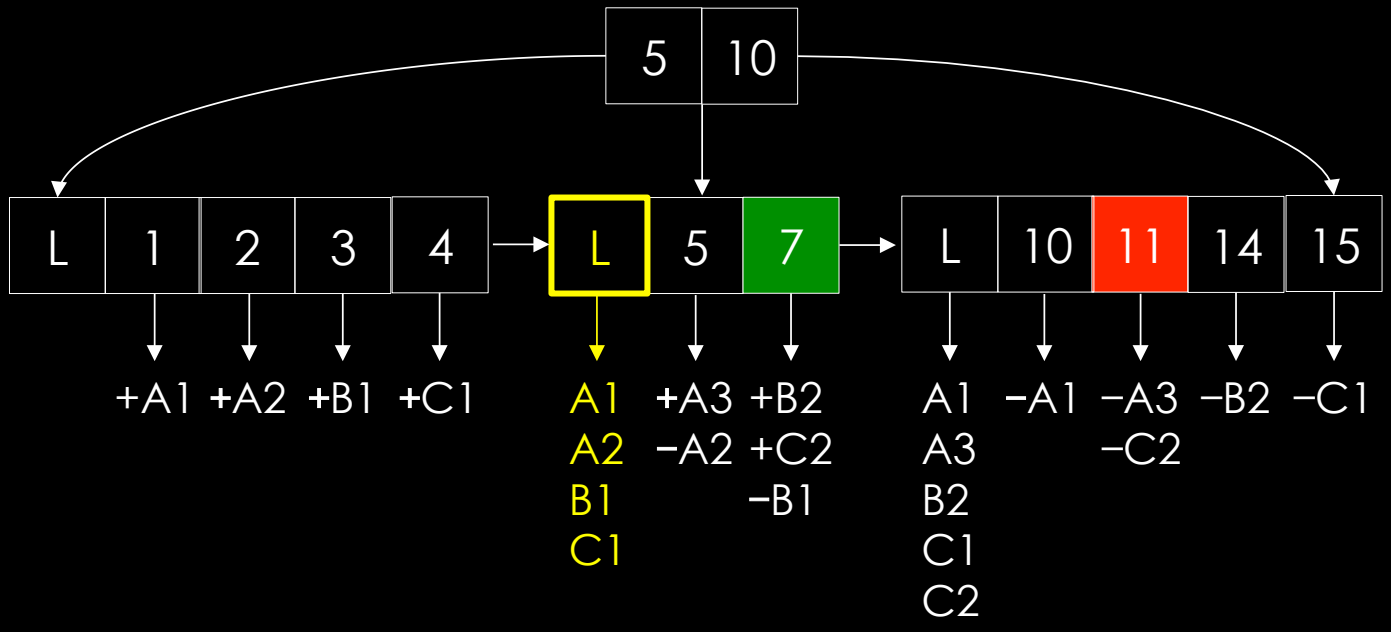

5 10
L 1 2 3 4
L 5 7
L 10 11 14 15
+A1 +A2 +B1 +C1
A1 +A3 +B2
A2 −A2 +C2
B1 −B1
C1
A1 −A1 −A3 −B2 −C1
A3 −C2
B2
C1
C2

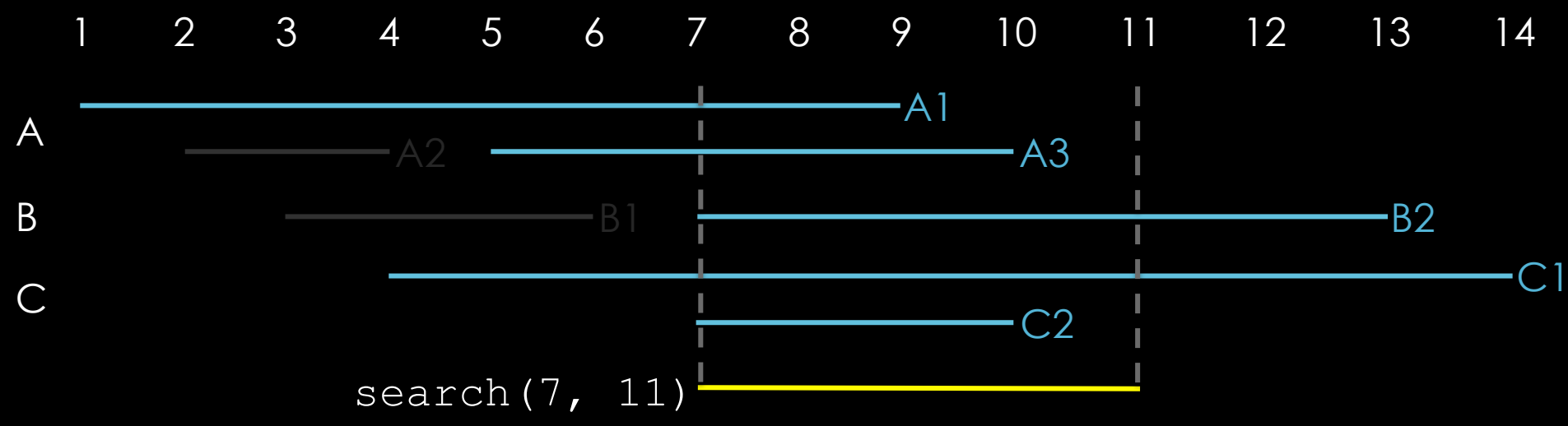

1 2 3 4 5 6 7 8 9 10 11 12 13 14
A
B
C
A1
A2
A3
B1
B2
C1
C2
search(7, 11)
I = [ ]
5 10
L 1 2 3 4
L 5 7
L 10 11 14 15
+A1 +A2 +B1 +C1
A1 A2 B1 C1
+A3 −A2
+B2 +C2 −B1
A1 A3 B2 C1 C2
−A1
−A3 −C2
−B2
−C1


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

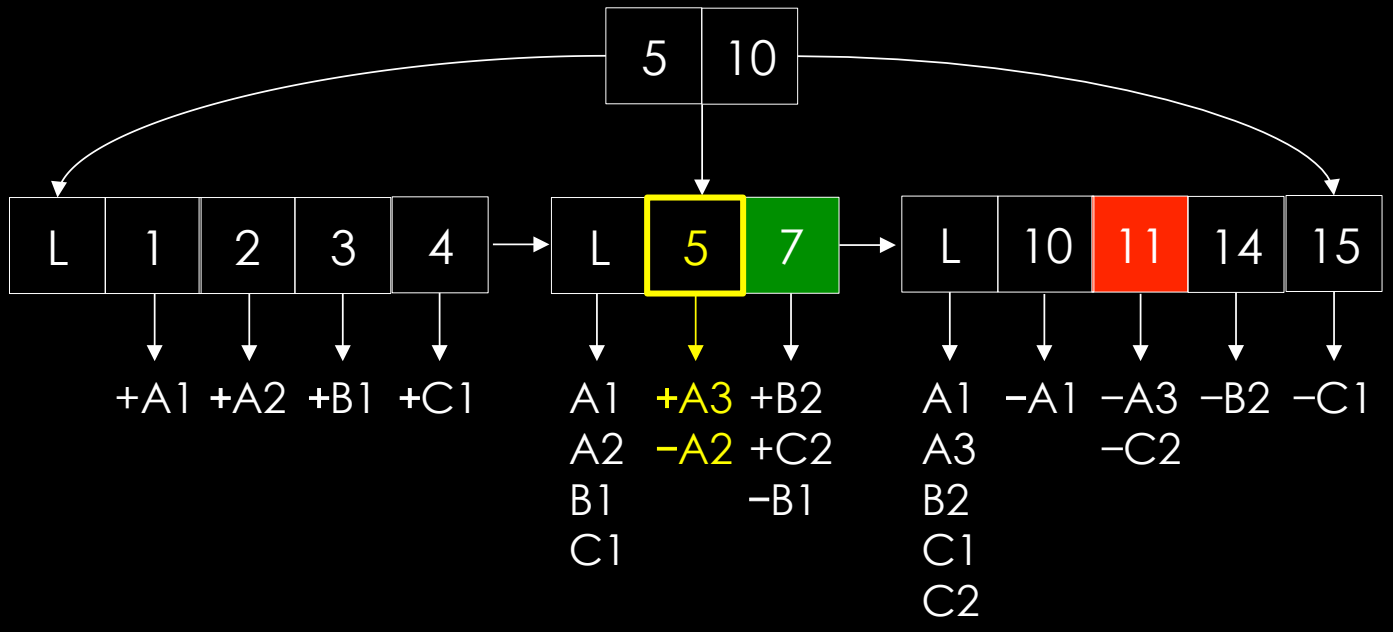

1 2 3 4 5 6 7 8 9 10 11 12 13 14
A
B
C
A1
A2
A3
B1
B2
C1
C2
search(7, 11)

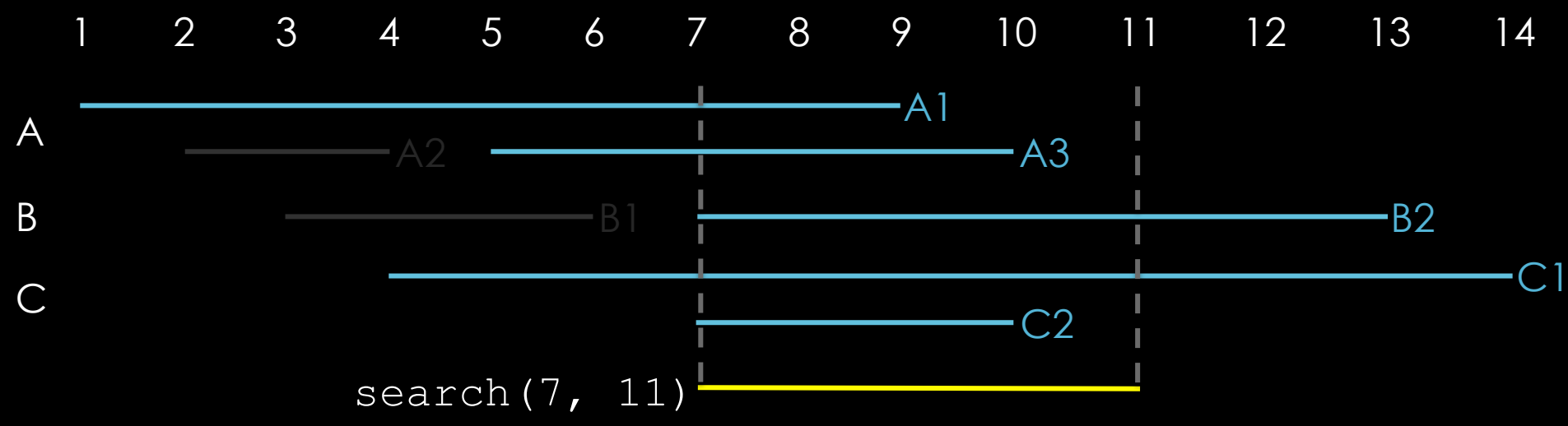
```
I = [A1 A2 B1 C1]
```


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

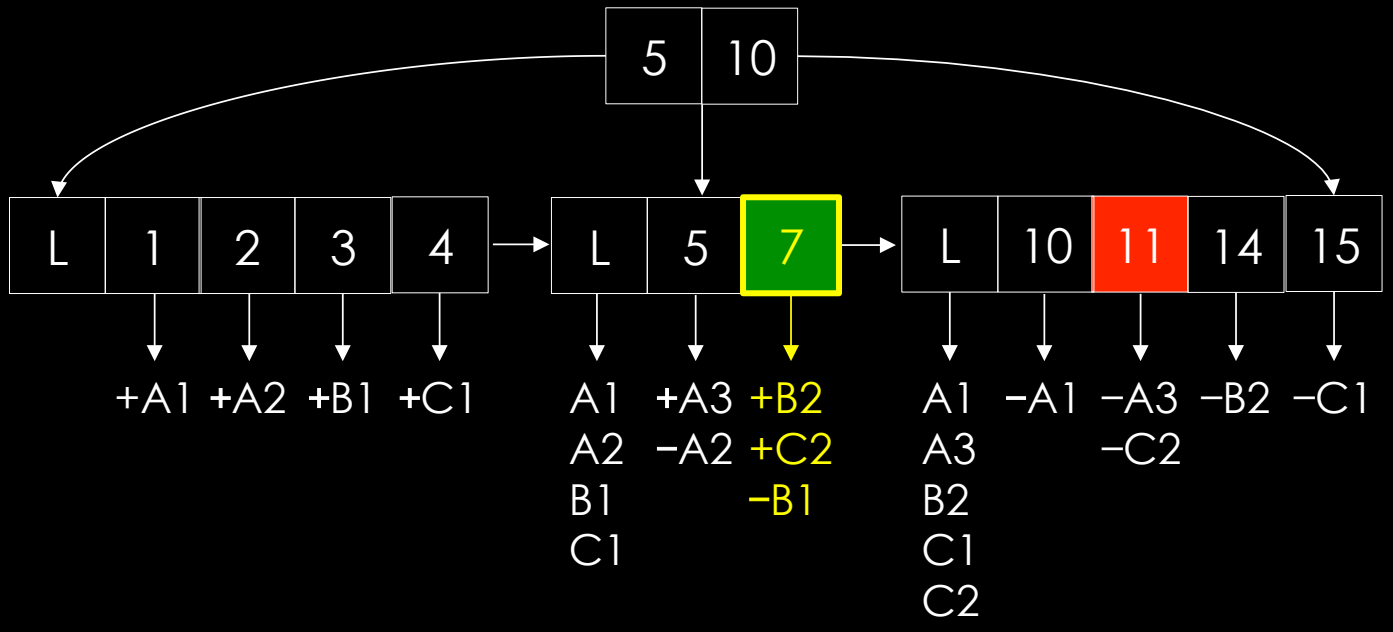


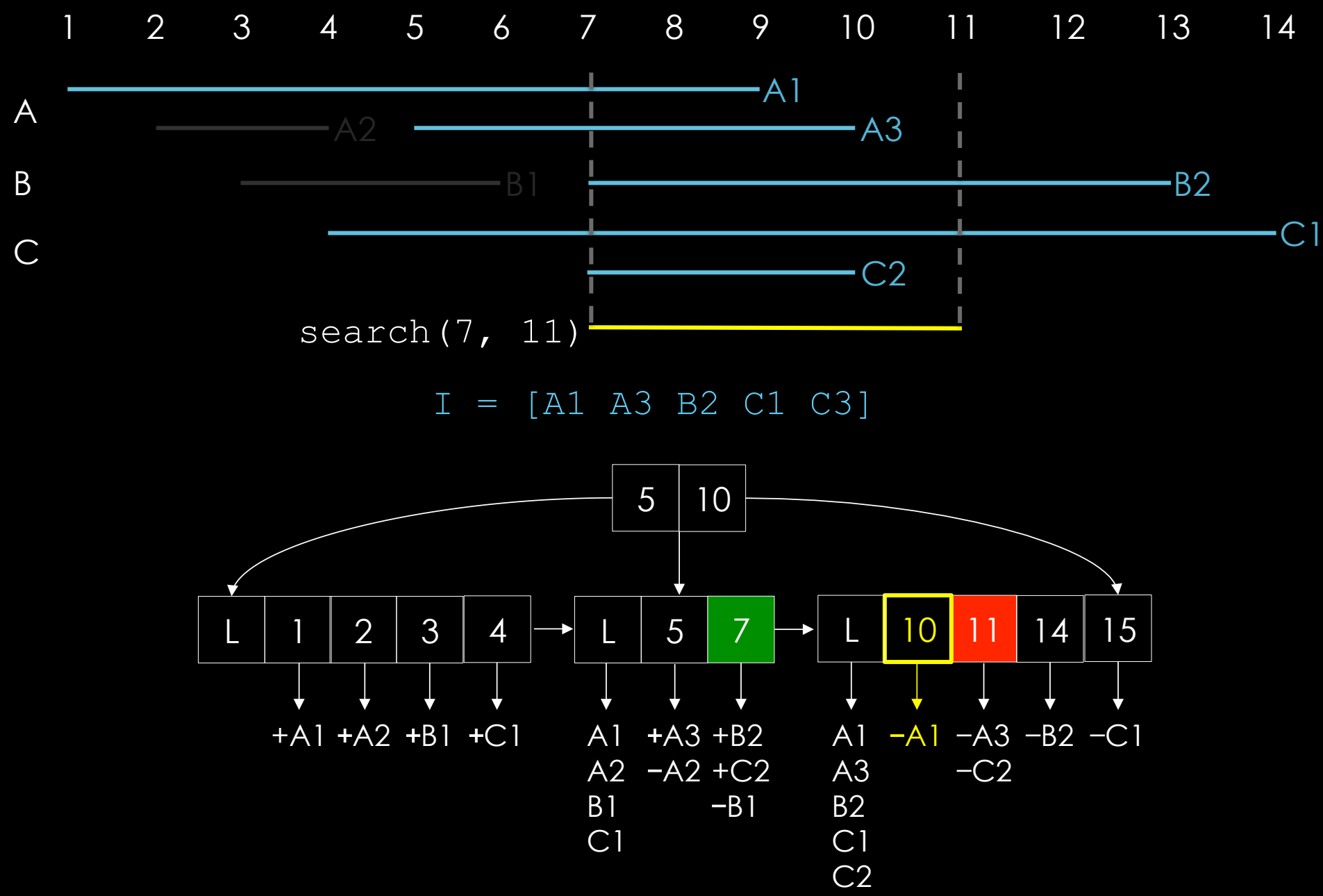

```
I = [A1 A2 B1 C1] + [A3] − [A2] = [A1 A3 B1 C1]
```

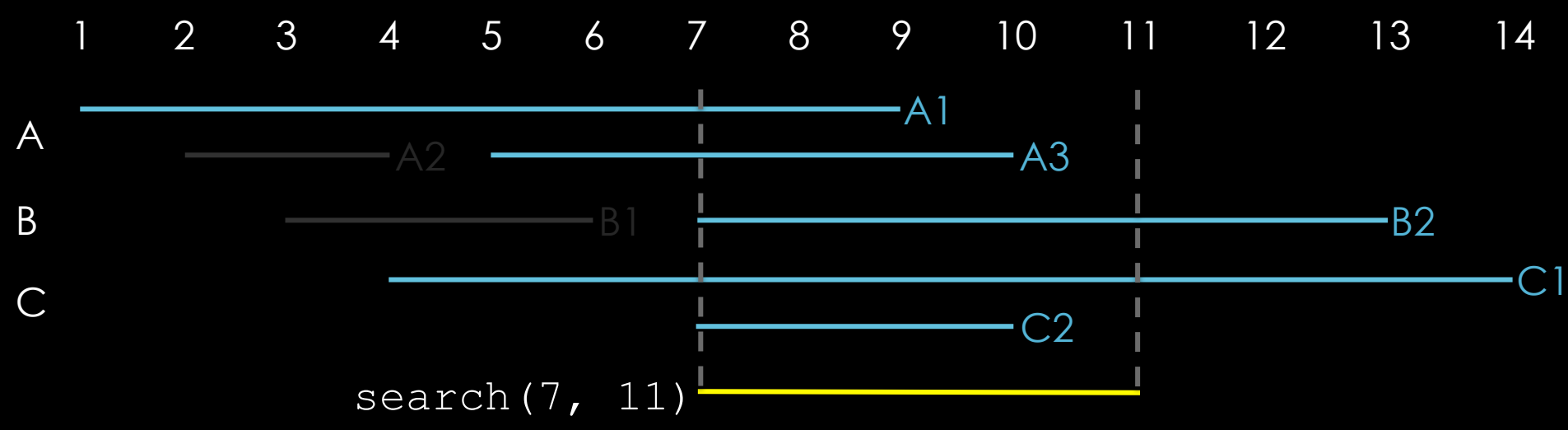




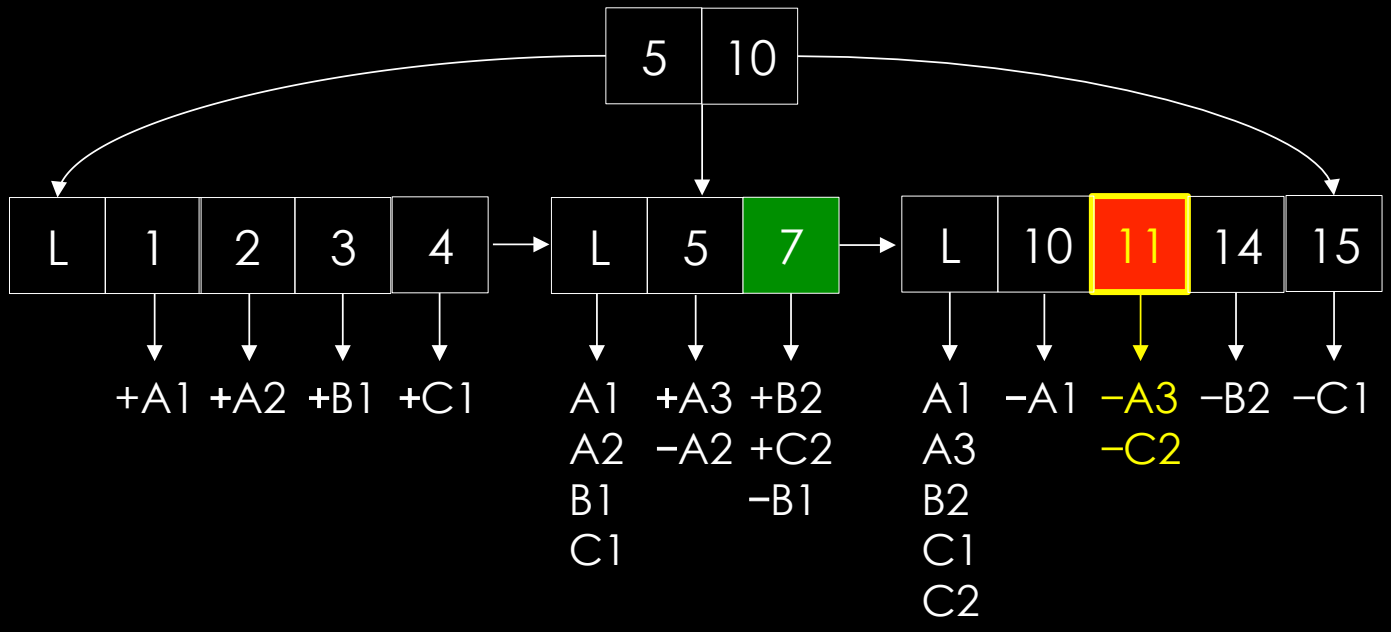

```
I = [A1 A3 B1 C1] + [B2 C2] - [B1] = [A1 A3 B2 C1 C3]
```




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]
```



- B+ tree
  - Efficient on-disk database
  - Traversal between leaves

- Many files in one index

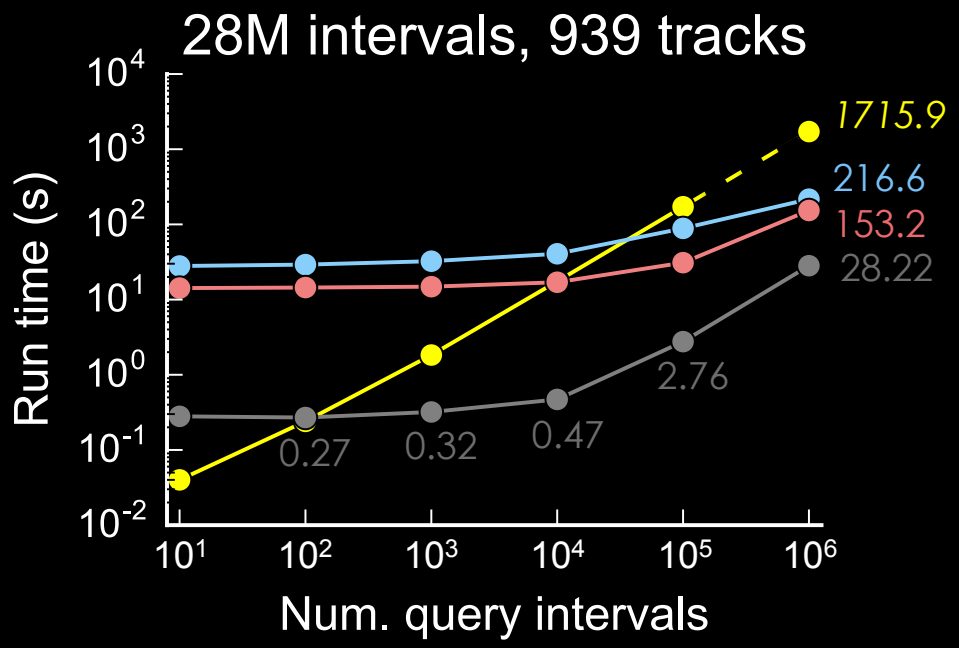
- Count intersections within the index

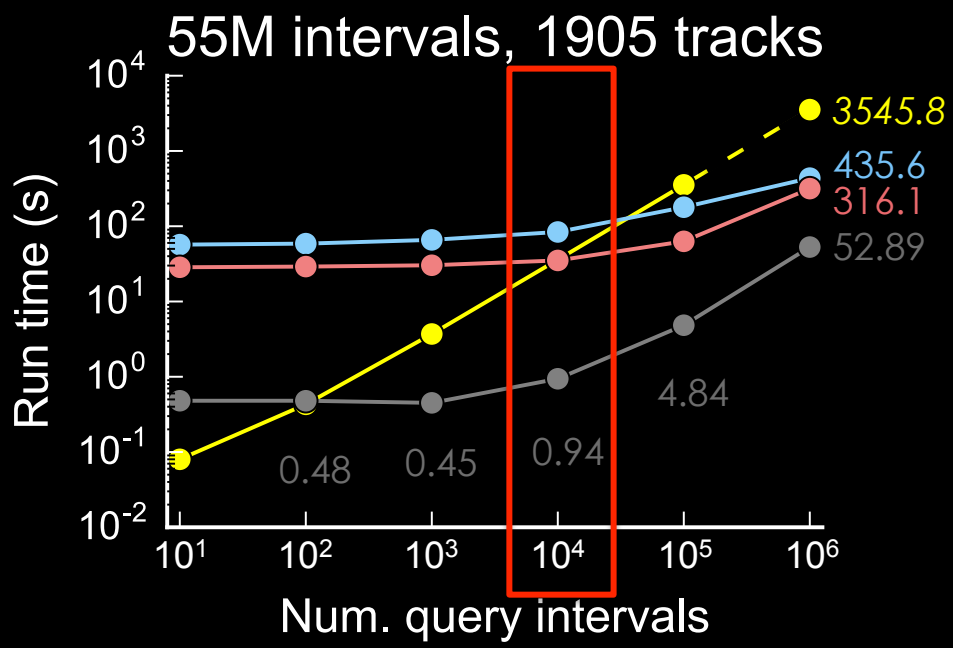
Count intersections for 10 -1M 100bp intervals

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

SQLITE3 w/ UCSC binning
TABIX
BEDTOOLS sorted
GIGGLE

2.6GHz Intel Xeon CPU, 20MB cache





Speed up: 60.8X 5.4X 7.7X

67X 5.9X 8.2X

my_chipseq.bed

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

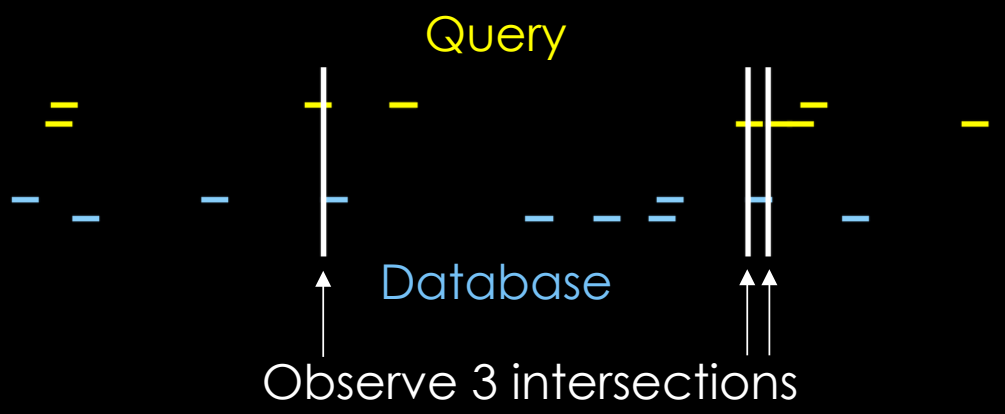
DNase-seq of K562
Homo sapiens, adult 53 year
Lab: John Stamatoyannopoulos, UW
Project: ENCODE

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

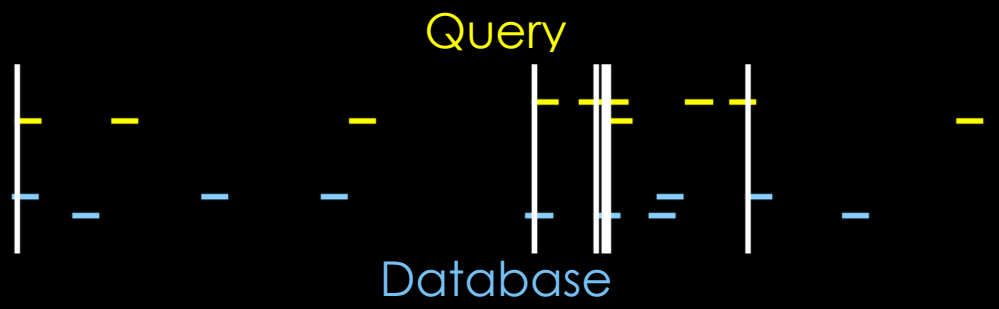
ChIP-seq of heart
Mus musculus, embryonic 10.5 day
Target: H3K27me3
Lab: Bing Ren, UCSD
Project: ENCODE

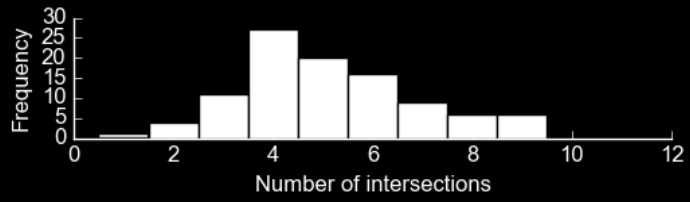
ChIP-seq of embryonic facial prominence
Mus musculus, embryonic 10.5 day
Target: H3K27me3
Lab: Bing Ren, UCSD
Project: ENCODE

Monte Carlo simulations



Monte Carlo simulations



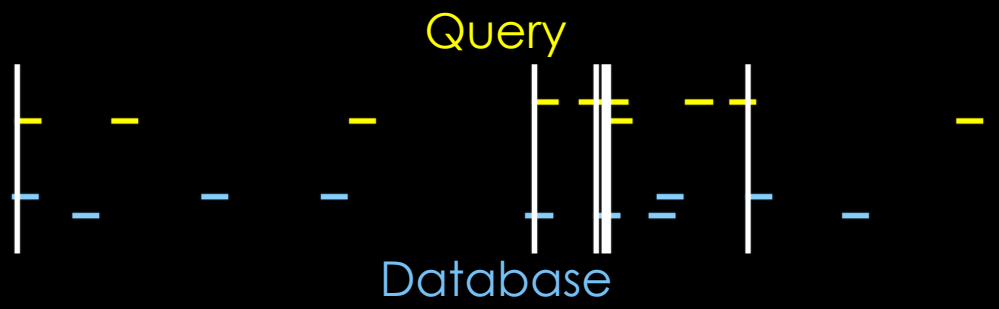


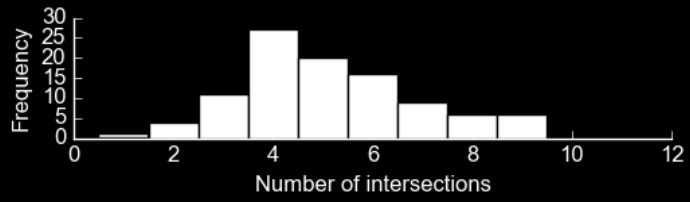
Observe 3 intersections

Expect ~4

0.75 "enrichment"

Monte Carlo simulations





Observe 3 intersections

Expect ~4

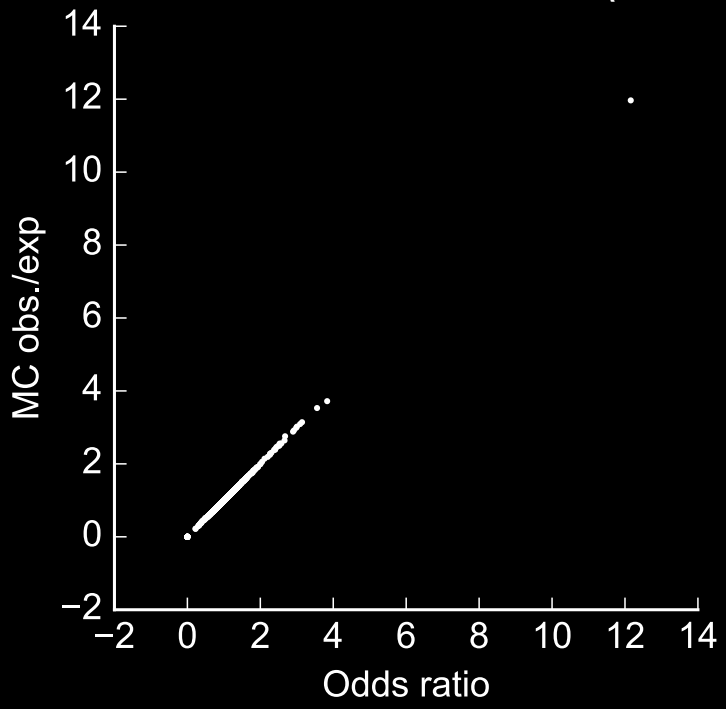
0.75 "enrichment"

Contingency table

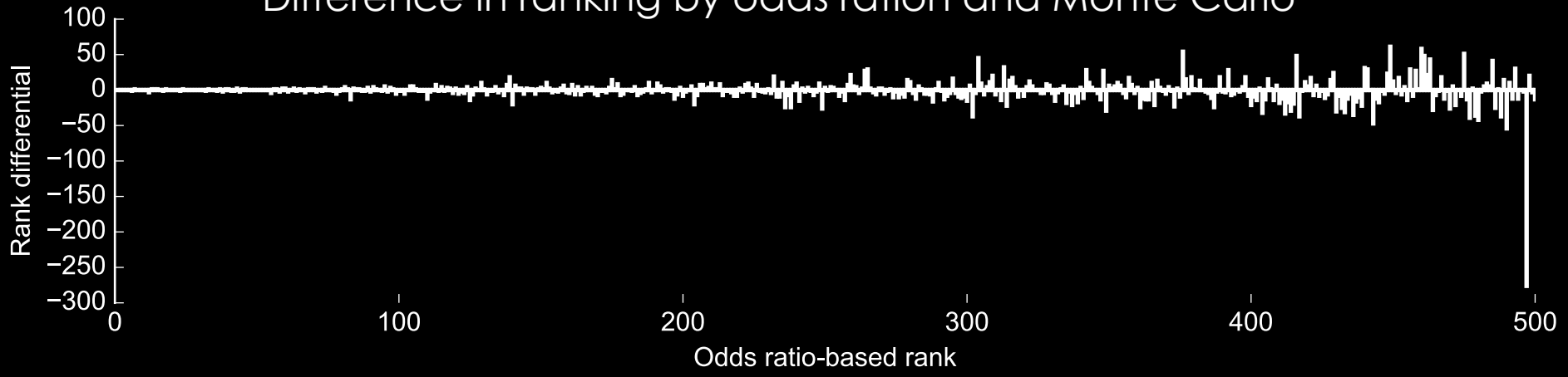
	In target	Not in target
In query	| Q T |	|Q | - | Q T |
Not In query	|T| - | Q T |	$\frac{\text{Genome size}}{\mu(Q) + \mu(T)} - (\,|Q| + |T| - |Q\,T|\,)$

Brent Pedersen

Odds ratio vs. Monte Carlo(N=10000)



Difference in ranking by odds ration and Monte Carlo



Genotype Query Tools

github.com/ryanlayer/gqt

+

Giggle

SFARI 40 families 20M variants

ROADMAP epigenomics PROJECT 127 tissues 15 genomic states

De Novos in:

Affected child

```
gqt query -i all.vcf.gz
-p "phenotype == 2"
-g "count(HET) == 1"
-p "phenotype == 1"
-g "HOM_REF"
```

Unaffected child

```
-p "phenotype == 1
    & parental_
-g "count(HET)
-p "phenotype !
    || parental
-g "HOM_REF"
```

< 6s

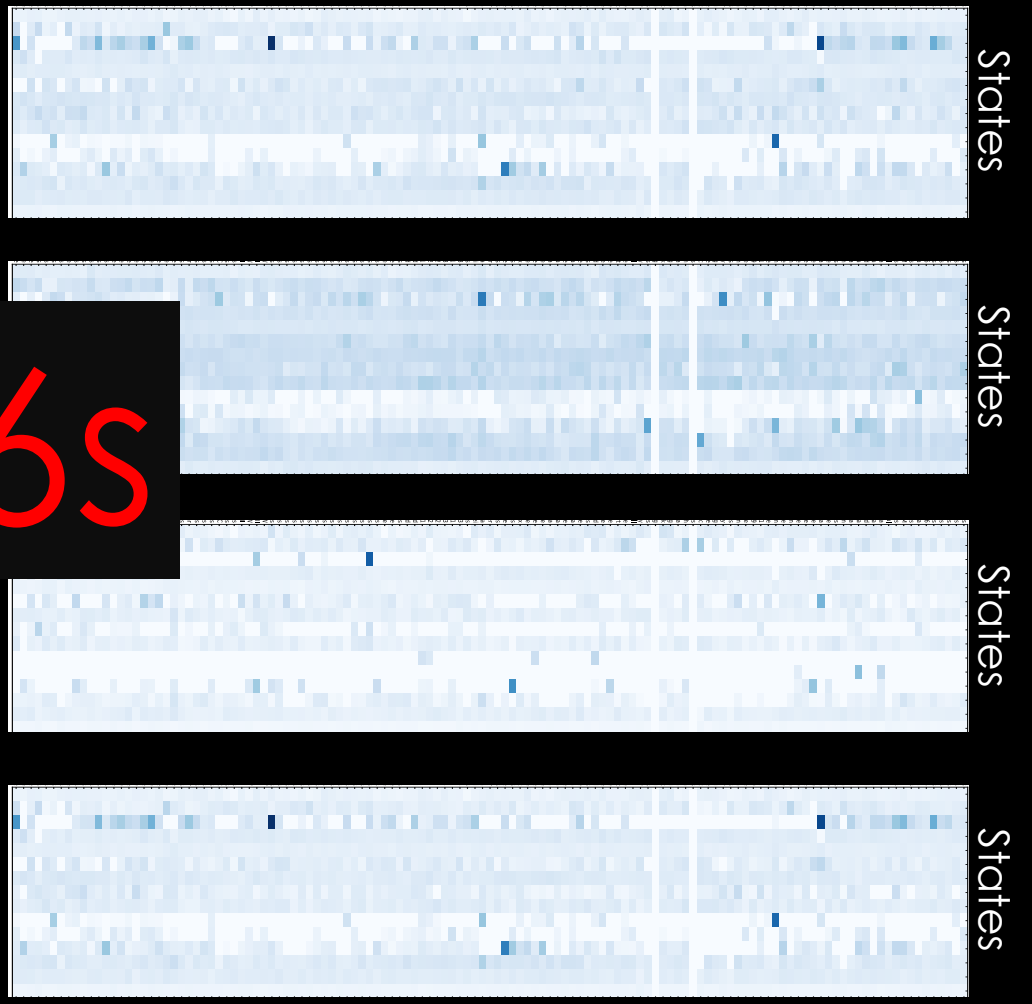
Affected female

```
-p "phenotype =
    & sex == 2"
-g "count(HET) == 1"
-p "phenotype == 1"
-g "HOM_REF"
```

Affected European

```
-p "phenotype == 2
    & population == 'EUR'"
-g "count(HET) == 1"
-p "phenotype == 1"
-g "HOM_REF"
```

Tissues and Cell Types



Tonya Di Sera

Scale
chr1:
100 bases
752,650
752,700
752,750
hg19
Assembly from Fragments
Assembly
BAC End Pairs
BAC End Pairs
Chromosome Bands Localized by FISH Mapping Clones
Chromosome Band
1p36.33
Fosmid End Pairs
Fosmid End Pairs
GC Percent in 5-Base Windows
GC Percent
Genome Reference Consortium Map Contigs
GRC Map Contigs
HSCHR1_CTG3
Contigs Dropped or Changed from GRCh37(hg19) to GRCh38(hg38)
Hg38 Diff
Accession at INSDC - International Nucleotide Sequence Database Collaboration
INSDC
Physical Map Contigs
Map Contigs
GL000003.1
Recombination Rate from deCODE. Marshfield. or Genethon Maps (deCODE default)
Recomb Rate
UCSC Genes (RefSeq, GenBank, CCDS, Rfam, tRNAs & Comparative Genomics)
RefSeq Genes
RefSeq Genes
AceView Gene Models With Alt-Splicing
AceView Genes
Augustus gene predictions v3.1
Augustus
Ensembl Gene Predictions - archive 75 - feb2014
Ensembl Genes
Geneid Gene Predictions
Geneid Genes
N-SCAN Gene Predictions
N-SCAN
SGP Gene Predictions Using Mouse/Human Homology
SGP Genes
Swiss Institute of Bioinformatics Gene Predictions from mRNA and ESTs
SIB Genes
Publications: Sequences in Scientific Articles
Sequences
SNPs
Coriell Cell Line Copy Number Variants
Coriell CNVs
Human Quantitative Trait Locus from RGD
Blood pressure QTL 9
RGD Human QTL
DNA Sequences in Web Pages Indexed by Bing.com / Microsoft Research
Web Sequences
Human mRNAs from GenBank
Human mRNAs
Gene Boundaries as Defined by RNA and Spliced EST Clusters
Gene Bounds
H-Invitational Genes mRNA Alignments
H-Inv
Alternative Splicing Graph from Swiss Institute of Bioinformatics
SIB Alt-Splicing
UniGene Alignments
UniGene
H3K27Ac Mark (Often Found Near Active Regulatory Elements) on 7 cell lines from ENCODE
100
Layered H3K27Ac
0
DNaseI Hypersensitivity Clusters in 125 cell types from ENCODE (V3)
DNase Clusters
Transcription Factor ChIP-seq (161 factors) from ENCODE with Factorbook Motifs
Txn Factor ChIP
SwitchGear Genomics Transcription Start Sites
SwitchGear TSS
4.88
100 vertebrates Basewise Conservation by PhyloP
100 Vert. Cons
0
-4.5
Multiz Alignments of 100 Vertebrates
Rhesus
Mouse
Dog
Elephant
Chicken
X_tropicalis
Zebrafish
Lamprey
Neandertal Reconstructed DNA Methylation Map
Neandertal Methyl
SNPS Used for Selective Sweep Scan (S)
S SNPs
Selective Sweep Scan (S) on Neandertal vs. Human Polymorphisms (Z-Score +- Variance)
Sel Swp Scan (S)
Denisova Reconstructed DNA Methylation Map
Denisova Methyl
Simple Nucleotide Polymorphisms (dbSNP 146) Found in >= 1% of Samples
Common SNPs(146)
1000 Genomes Phase 3 Integrated Variant Calls: SNVs, Indels, SVs
1000G Ph3 Vars
Simple Nucleotide Polymorphisms (dbSNP 144)
All SNPs(144)
Repeating Elements by RepeatMasker
RepeatMasker
Duplications of >1000 Bases of Non-RepeatMasked Sequence
Segmental Dups
Human Chained Self Alignments
Self Chain

Command line

```
giggle index -i "tracks/*gz" -o tracks_i
giggle search -i tracks_i -r chr1:1-1000
giggle search -9 tracks_i -q query.bed.gz
```

C API

```
struct giggle_index *gi =
  giggle_load(index_dir_name, uint32_t_ll_giggle_set_data_handler);

struct giggle_query_result *gqr = giggle_query(gi ,chr ,beg ,end, gqr);

for(i = 0; i < gqr->num_files; i++) {
  struct giggle_query_iter *gqi = giggle_get_query_itr(gqr, i);

  while (giggle_query_next(gqi, &result) == 0)
    printf("%s\n", result);

  giggle_iter_destroy(&gqi);
}
```

Python API COMING SOON

Brent Pedersen

TODO:

- Index remote files
- Faster indexing
- Index updating (append)
- Integration with Genotype Query Tools (GQT)

github.com/ryanlayer/giggle

Ryan M. Layer
ryan.layer@gmail.com
@ryanlayer

Aaron Quinlan, Brent Pedersen, Tonya Di Sera

USTAR Center for Genetic Discovery

UNIVERSITY OF UTAH SCHOOL OF MEDICINE