BIG DATA MANAGEMENT

CE/CZ4123

KEY-VALUE STORE LSM-TREE (EXTENSIONS)

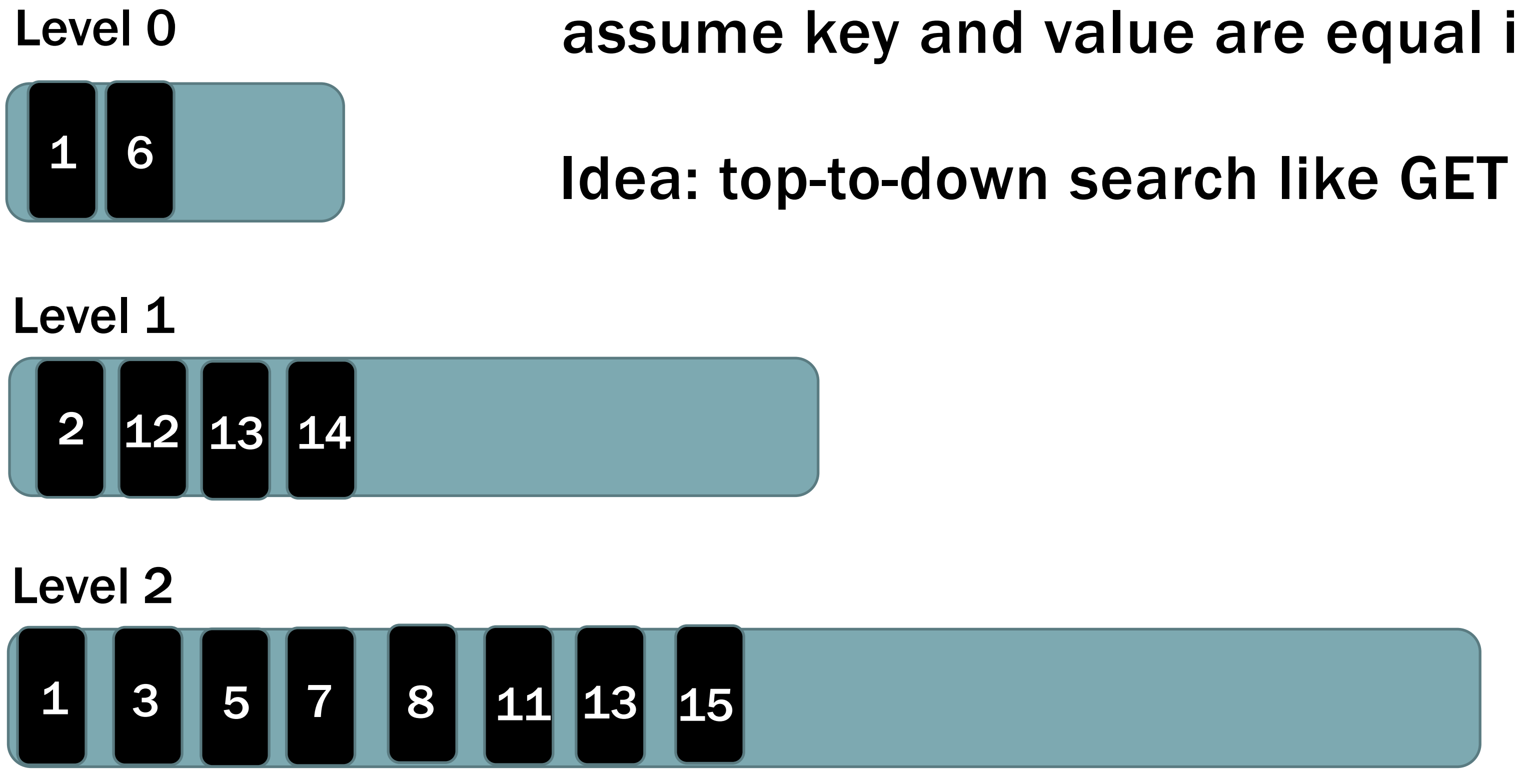
Siqiang Luo

Assistant Professor

How to implement Range-Get ([4,9])?

Range-Get([4,9]) → find out ALL values with keys larger than 4 and smaller than 9 in the LSM-tree (For simplicity, we assume key and value are equal in the example.

Idea: top-to-down search like GET function



Level 0

1 6

Level 1

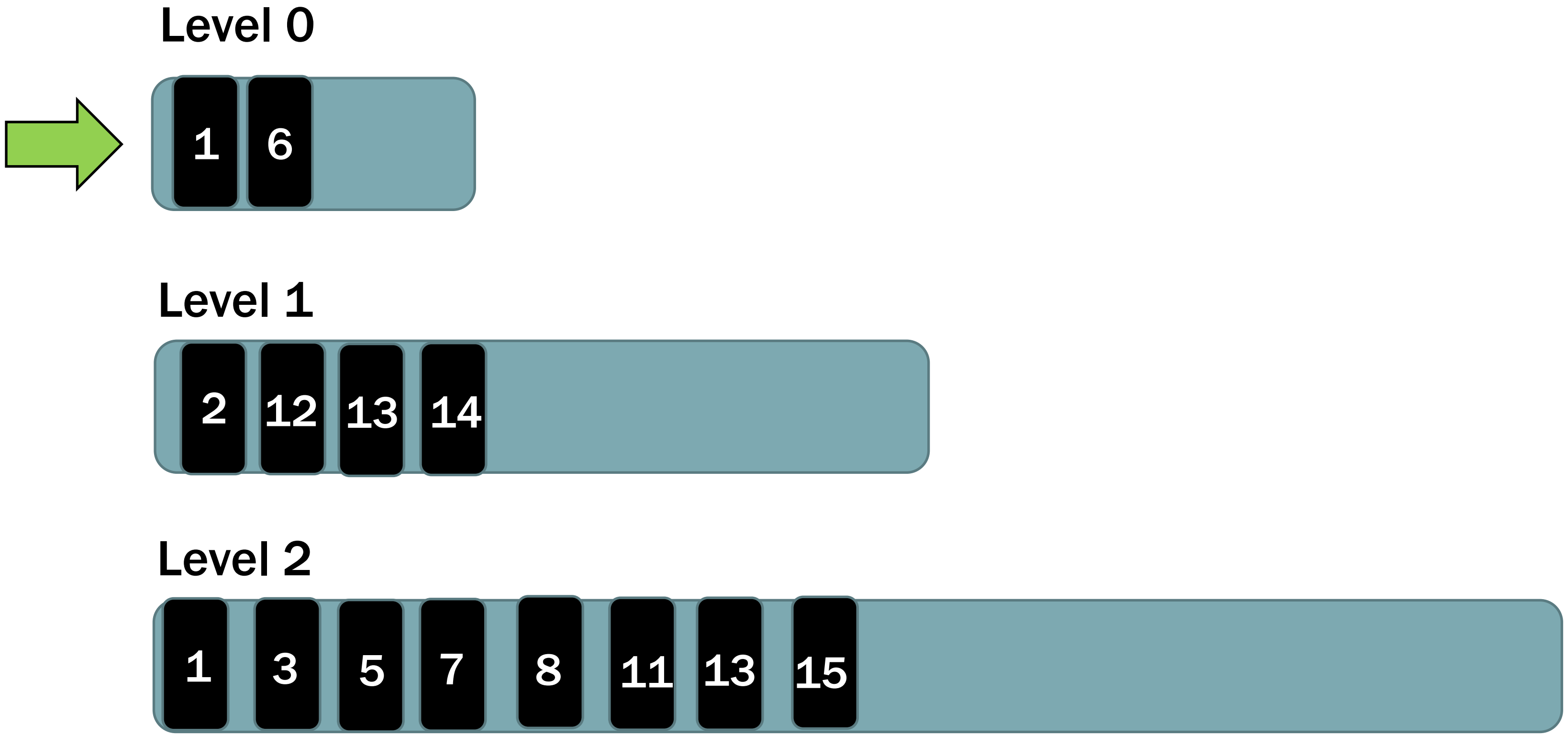
2 12 13 14

Level 2

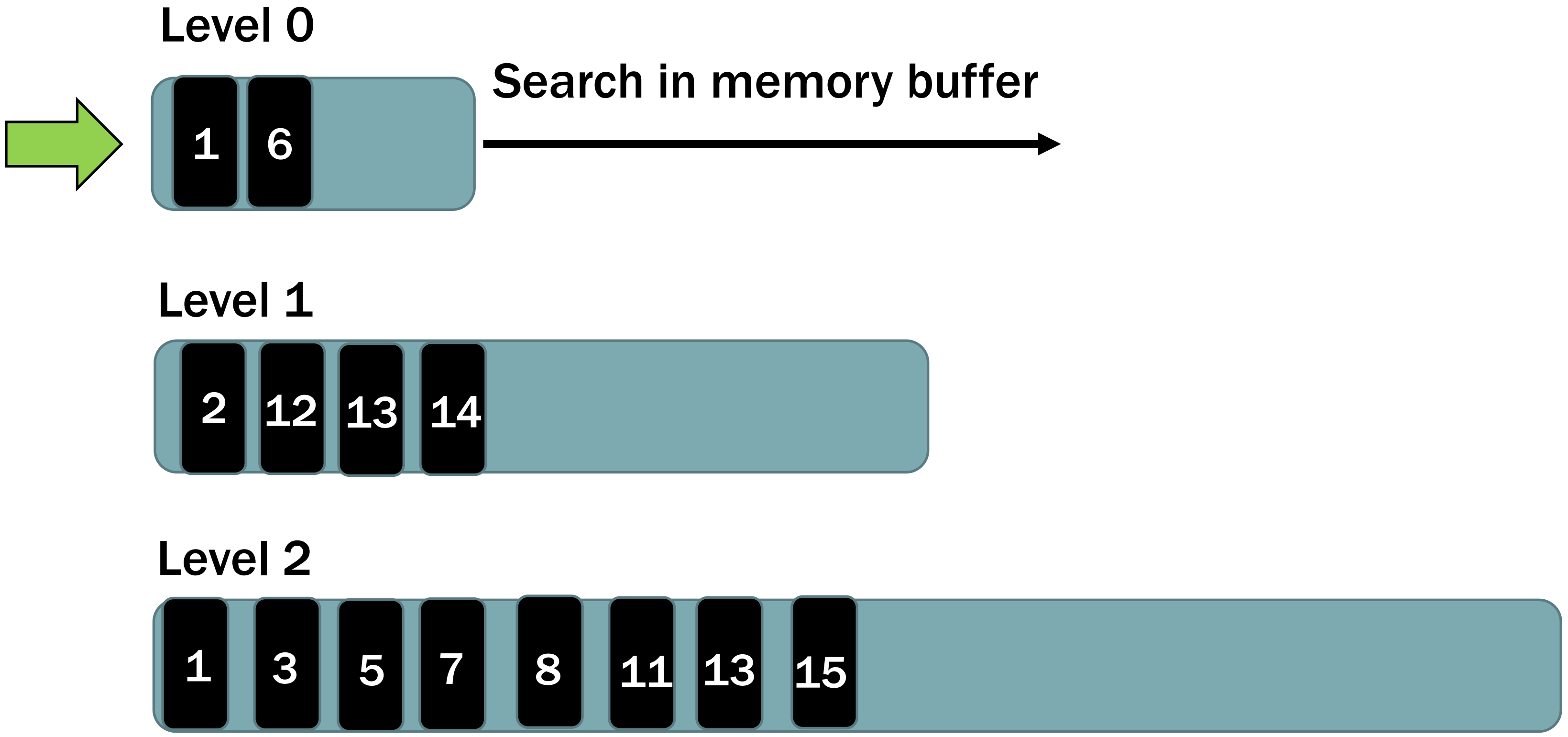
1 3 5 7 8 11 13 15

Leveling Structure

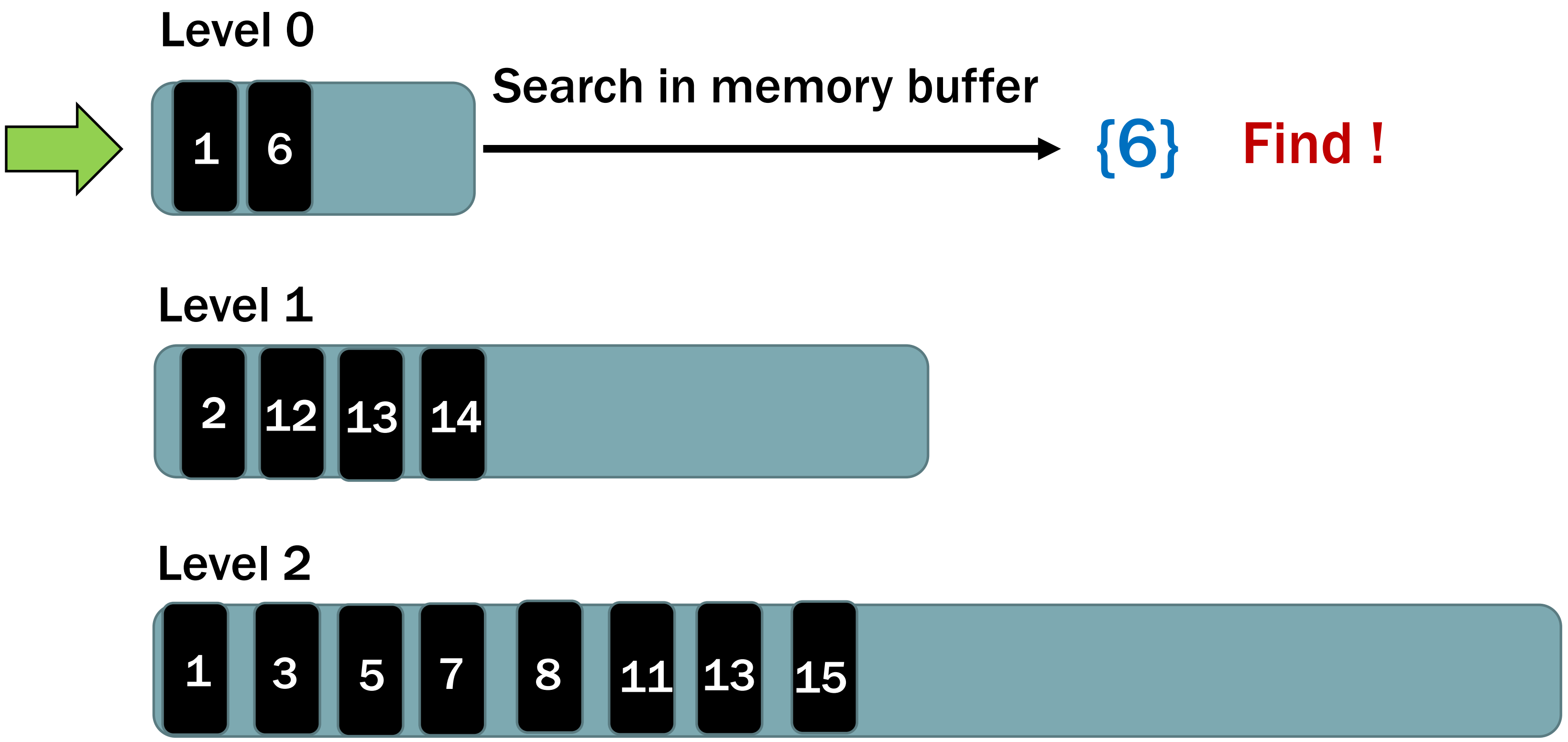
Range-Get([4,9]) → find out ALL keys larger than 4 and smaller than 9



Range-Get([4,9]) → find out ALL keys larger than 4 and smaller than 9



Range-Get([4,9]) → find out ALL keys larger than 4 and smaller than 9



Level 0

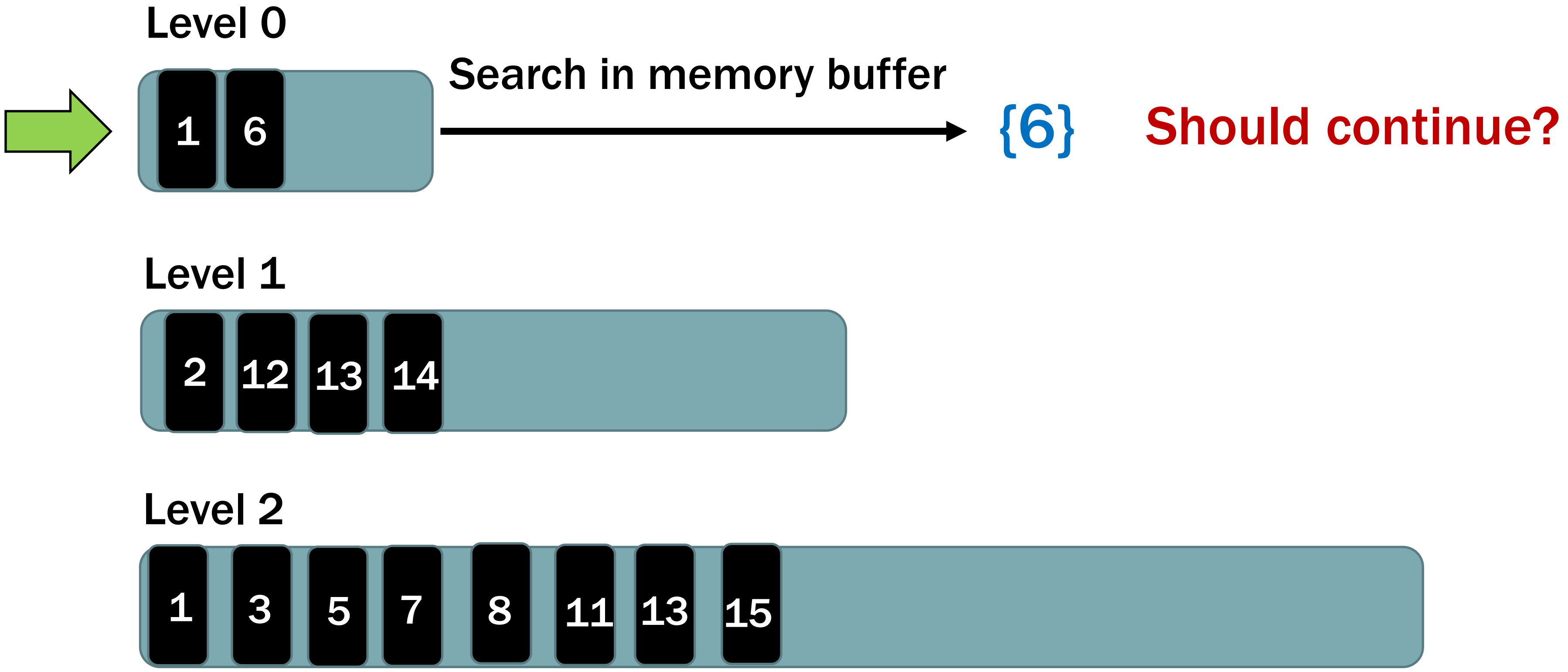
Search in memory buffer

{6} Find !

Level 1

Level 2

Range-Get([4,9]) → find out ALL keys larger than 4 and smaller than 9



Level 0

Search in memory buffer

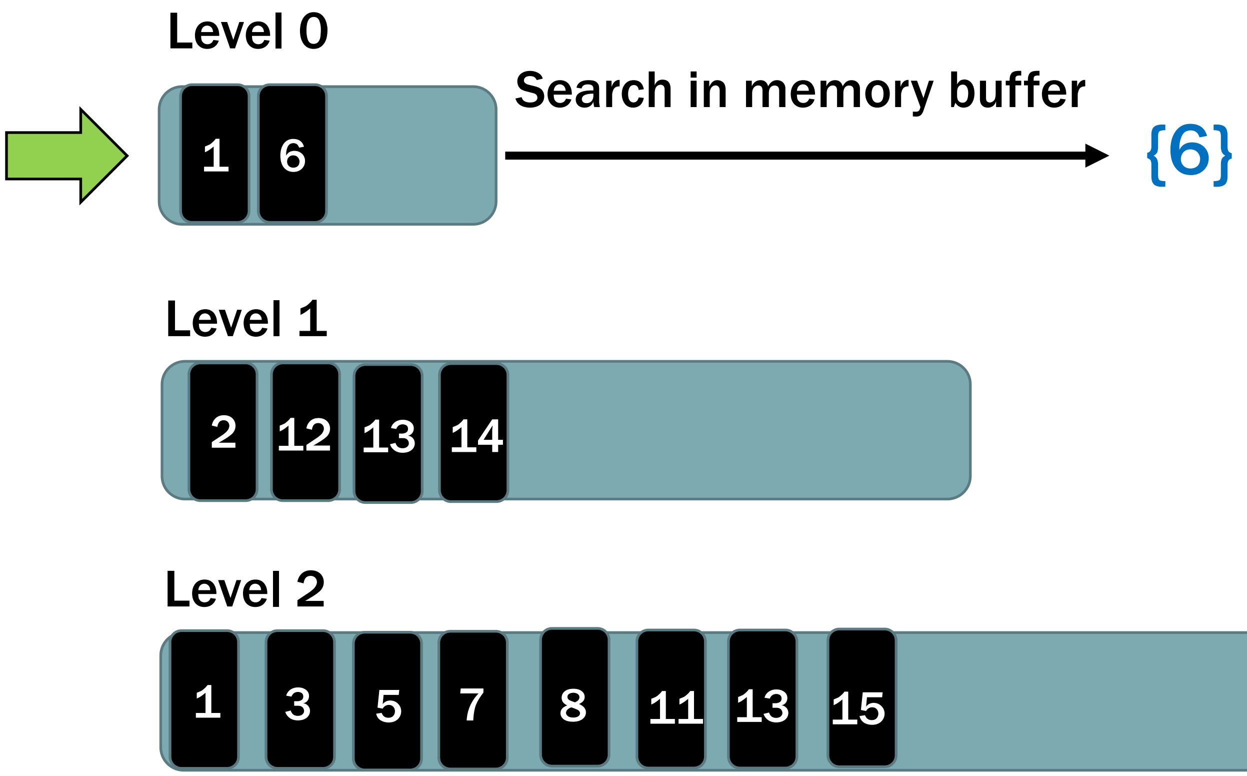
{6}

Should continue?

Level 1

Level 2

Range-Get([4,9]) → find out ALL keys larger than 4 and smaller than 9

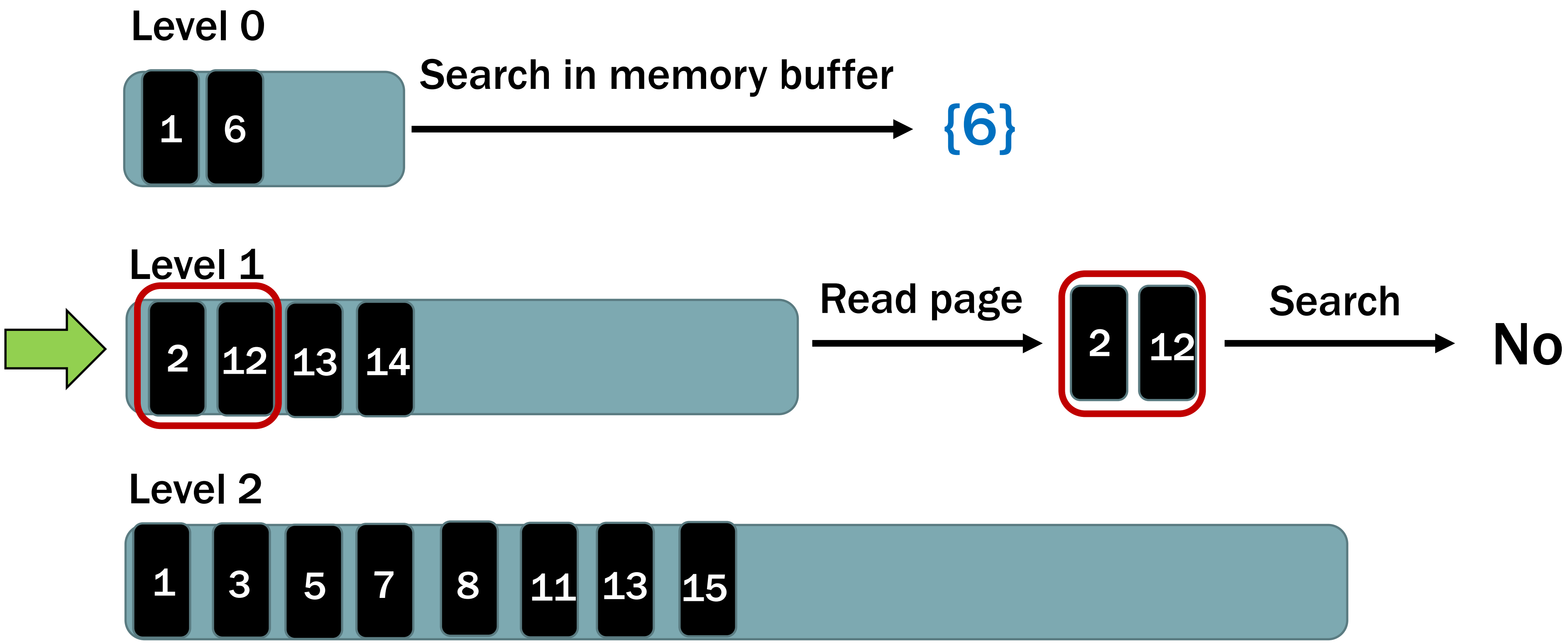


Should continue?

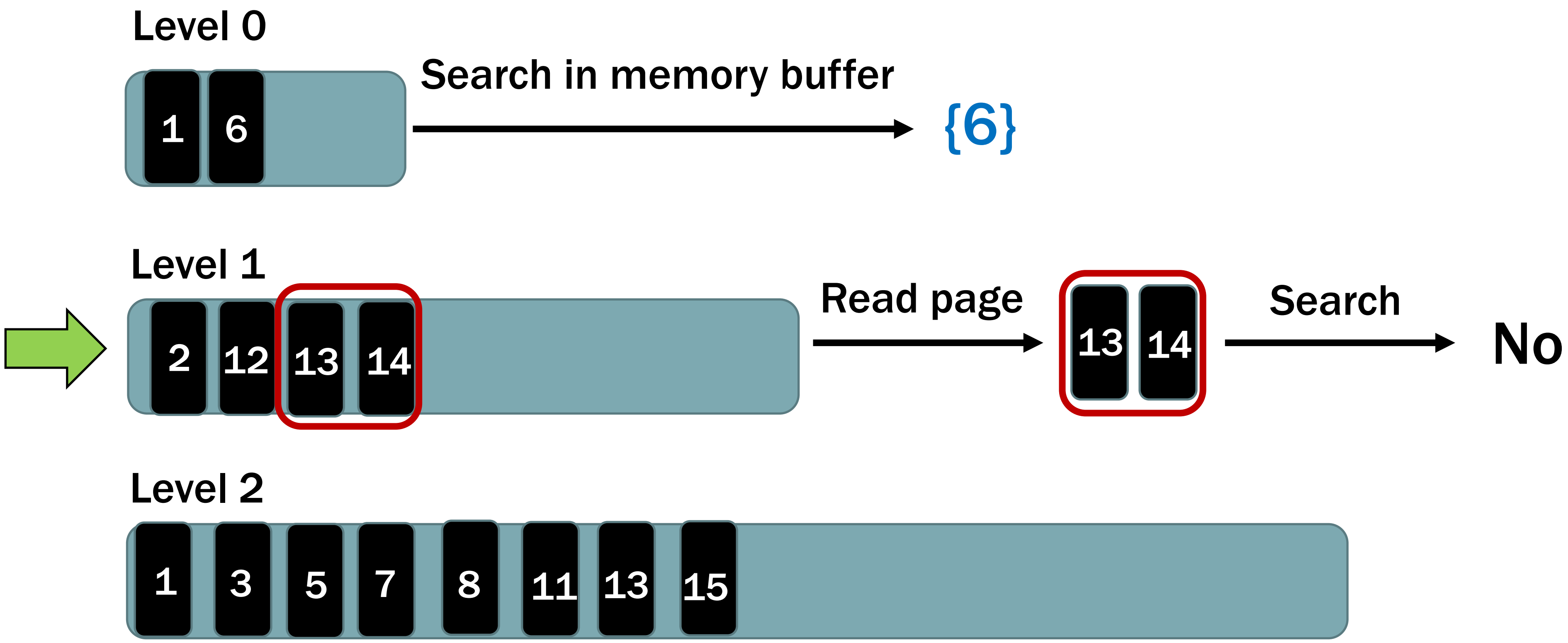
Yes!

Note: This is different from the Get function.

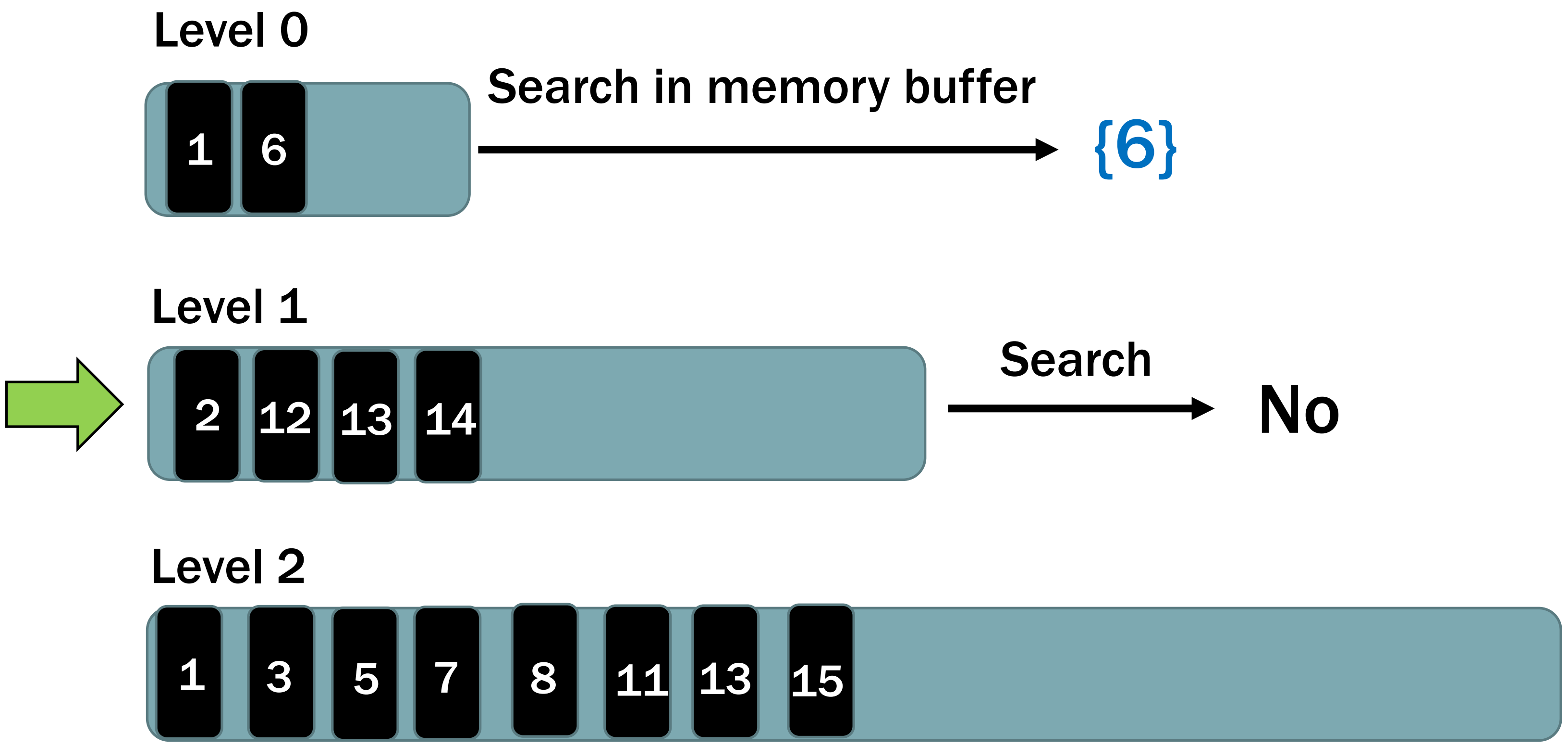
Range-Get([4,9]) → find out ALL keys larger than 4 and smaller than 9



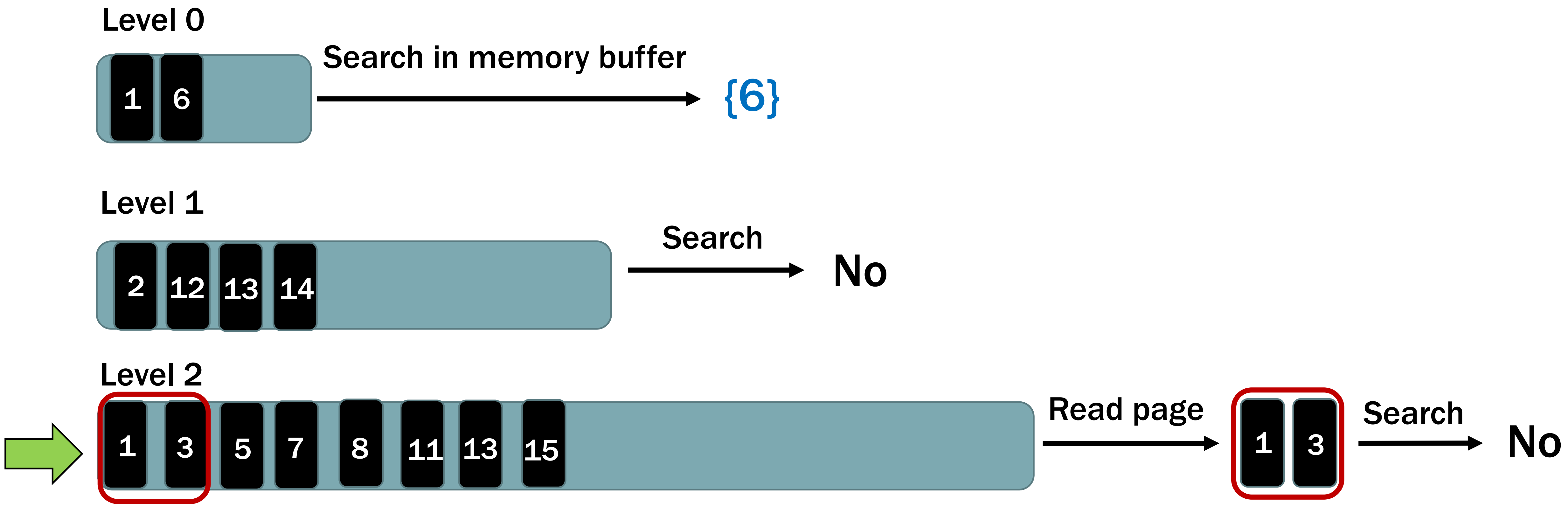
Range-Get([4,9]) → find out ALL keys larger than 4 and smaller than 9



Range-Get([4,9]) → find out ALL keys larger than 4 and smaller than 9



Range-Get([4,9]) → find out **ALL** keys larger than 4 and smaller than 9



Level 0

1 6

Search in memory buffer

{6}

Level 1

2 12 13 14

Search

No

Level 2

1 3 5 7 8 11 13 15

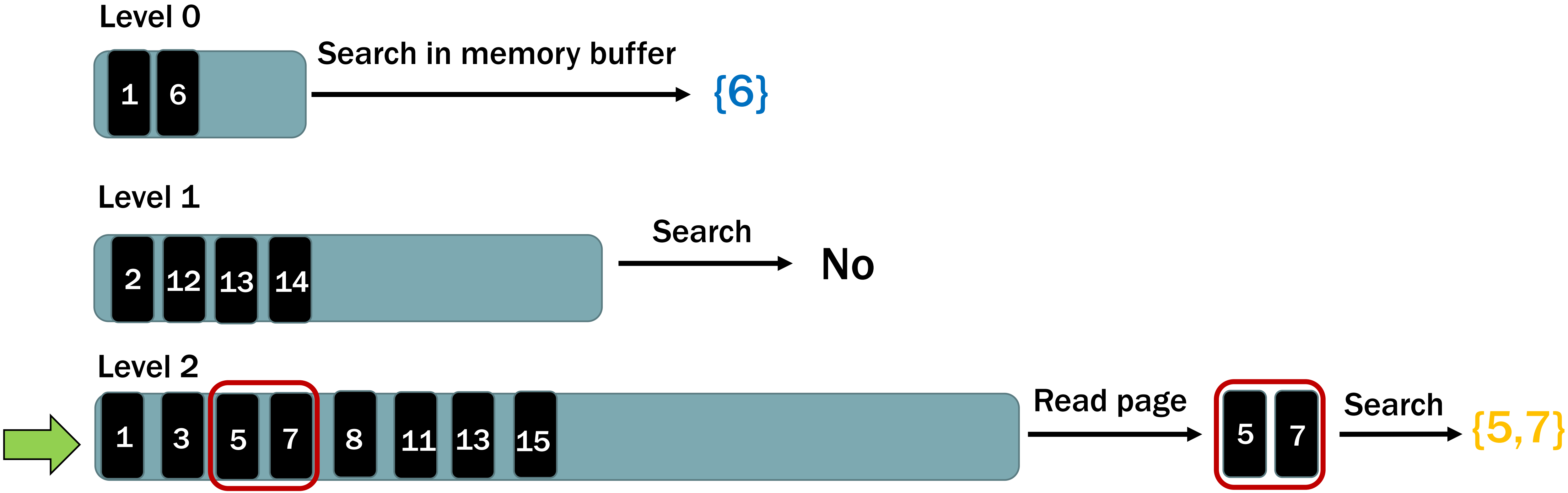
Read page

1 3

Search

No

Range-Get([4,9]) → find out ALL keys larger than 4 and smaller than 9



Level 0

Search in memory buffer → {6}

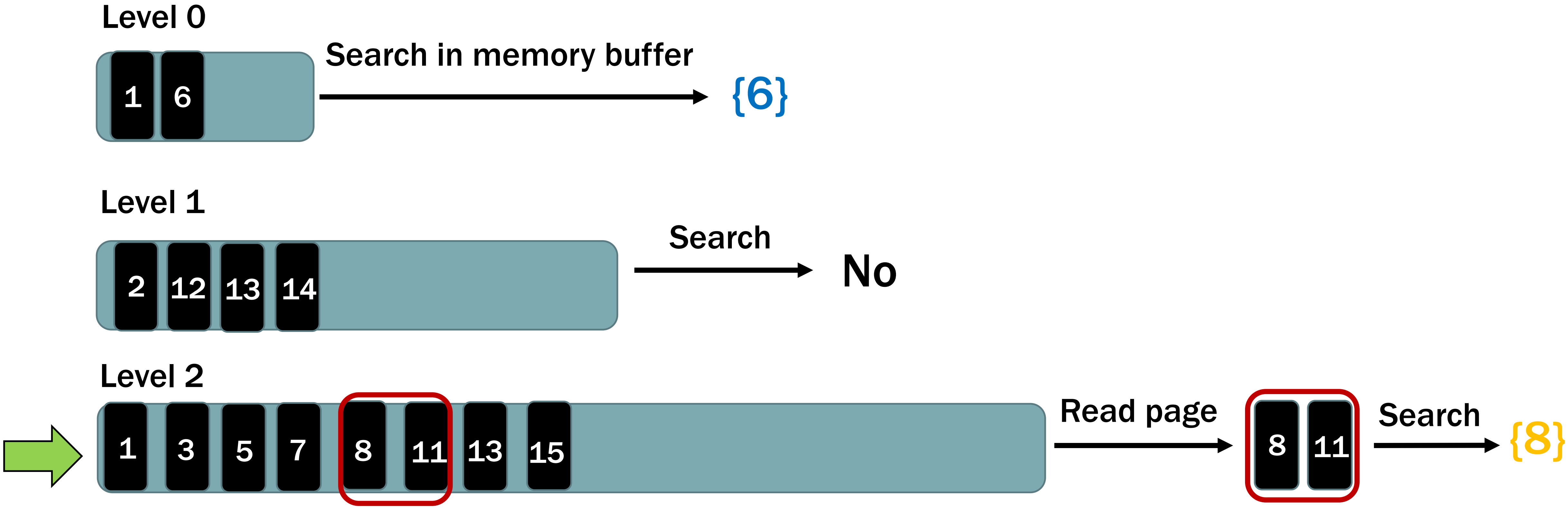
Level 1

Search → No

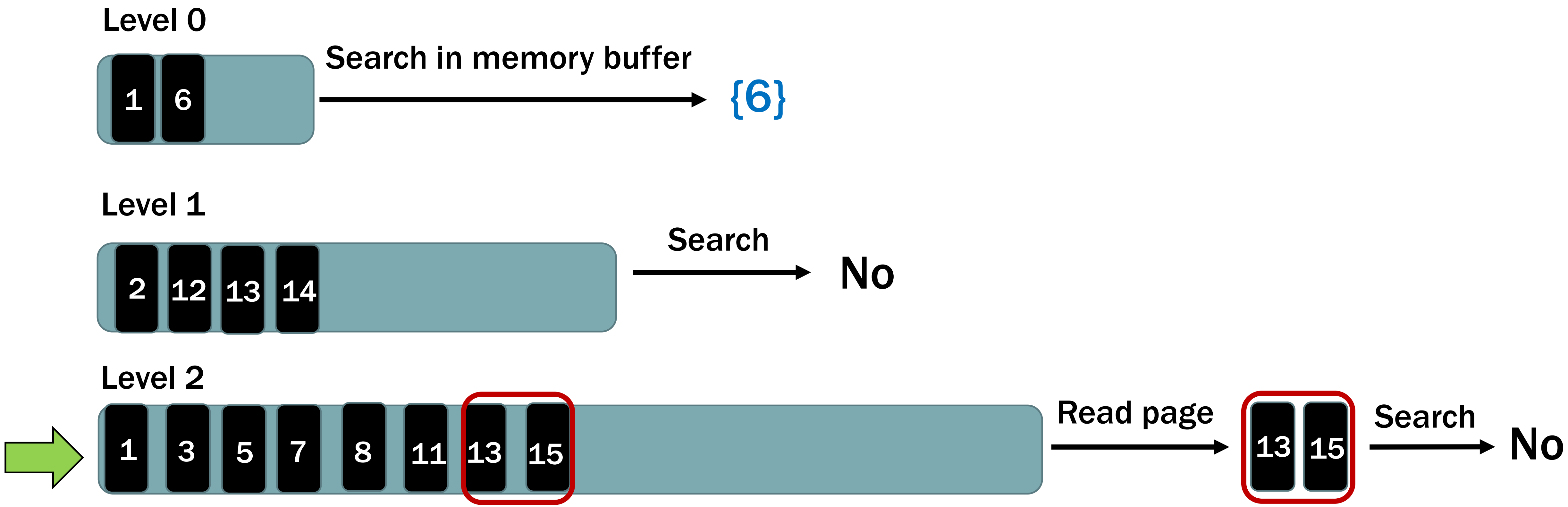
Level 2

Read page → Search → {5,7}

Range-Get([4,9]) → find out ALL keys larger than 4 and smaller than 9



Range-Get([4,9]) → find out ALL keys larger than 4 and smaller than 9



Level 0

Search in memory buffer

{6}

Level 1

Search

No

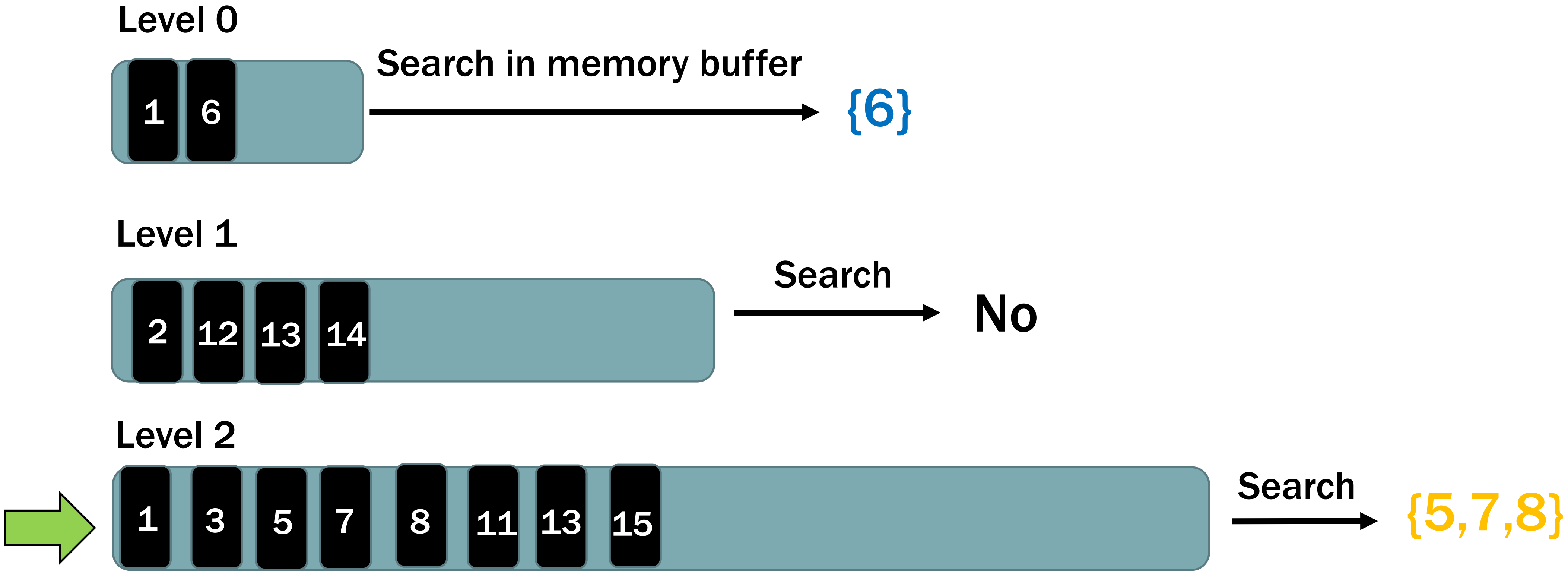
Level 2

Read page

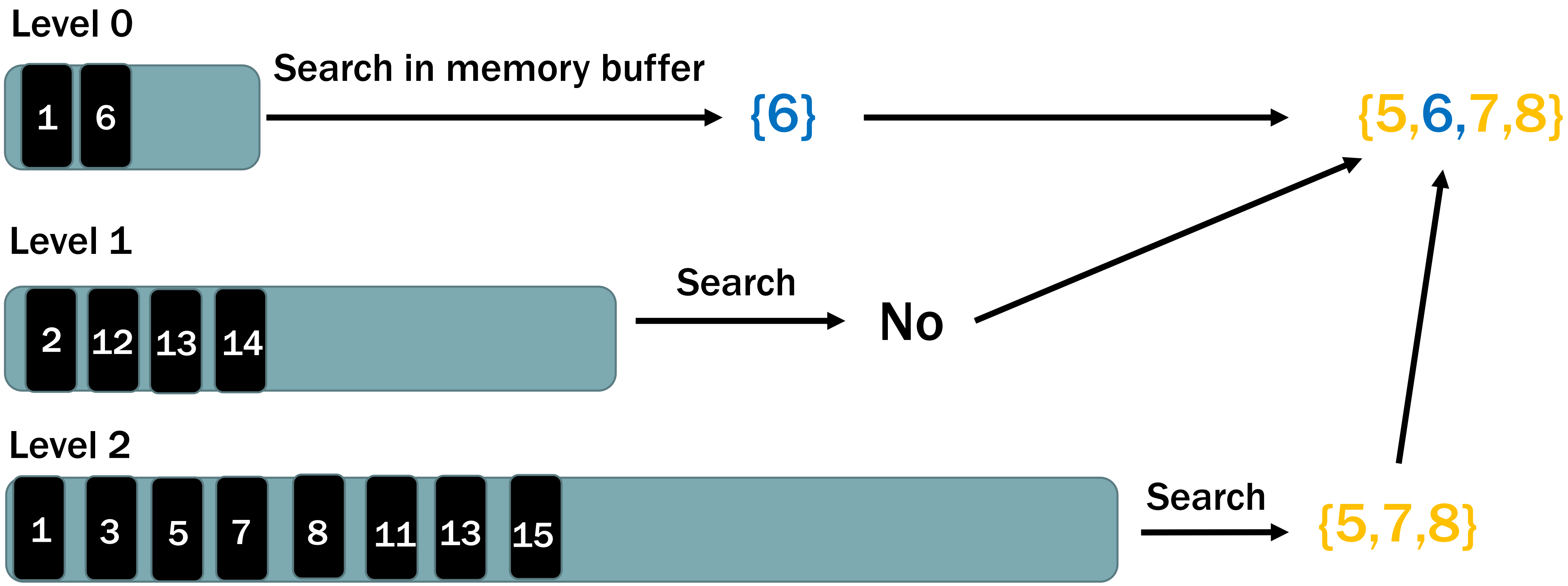
Search

No

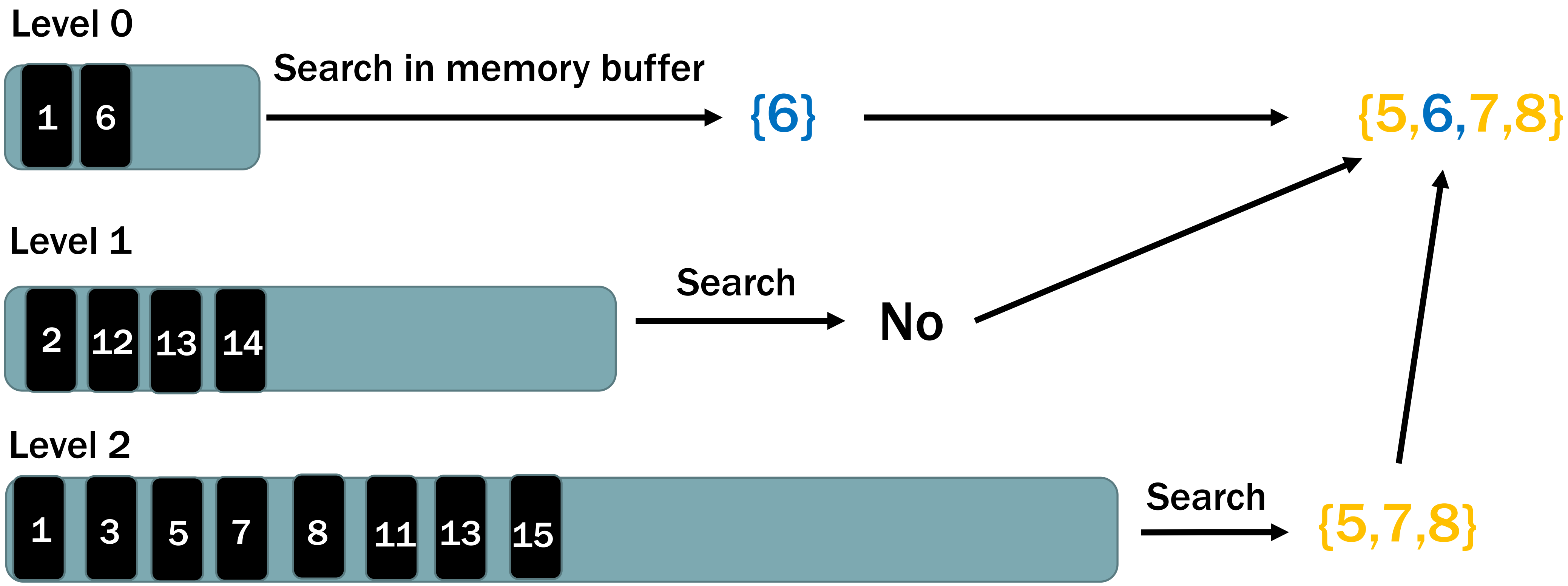
Range-Get([4,9]) → find out ALL keys larger than 4 and smaller than 9



Range-Get([4,9]) → find out ALL keys larger than 4 and smaller than 9



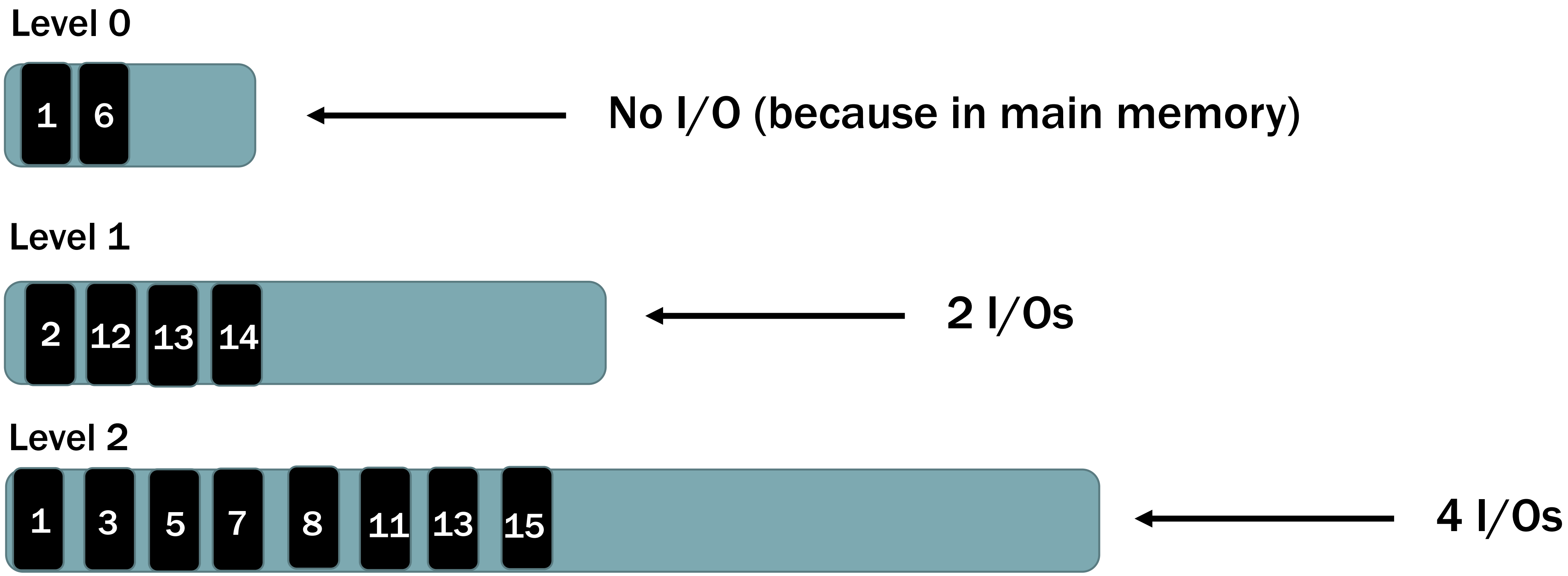
Range-Get([4,9]) → {5,6,7,8}



THE COST FOR RANGE_GET([4,9])

Range-Get([4,9]) → {5,6,7,8}

All the existing data in the LSM-tree should be retrieved which introduces huge I/O cost.



Level 0

No I/O (because in main memory)

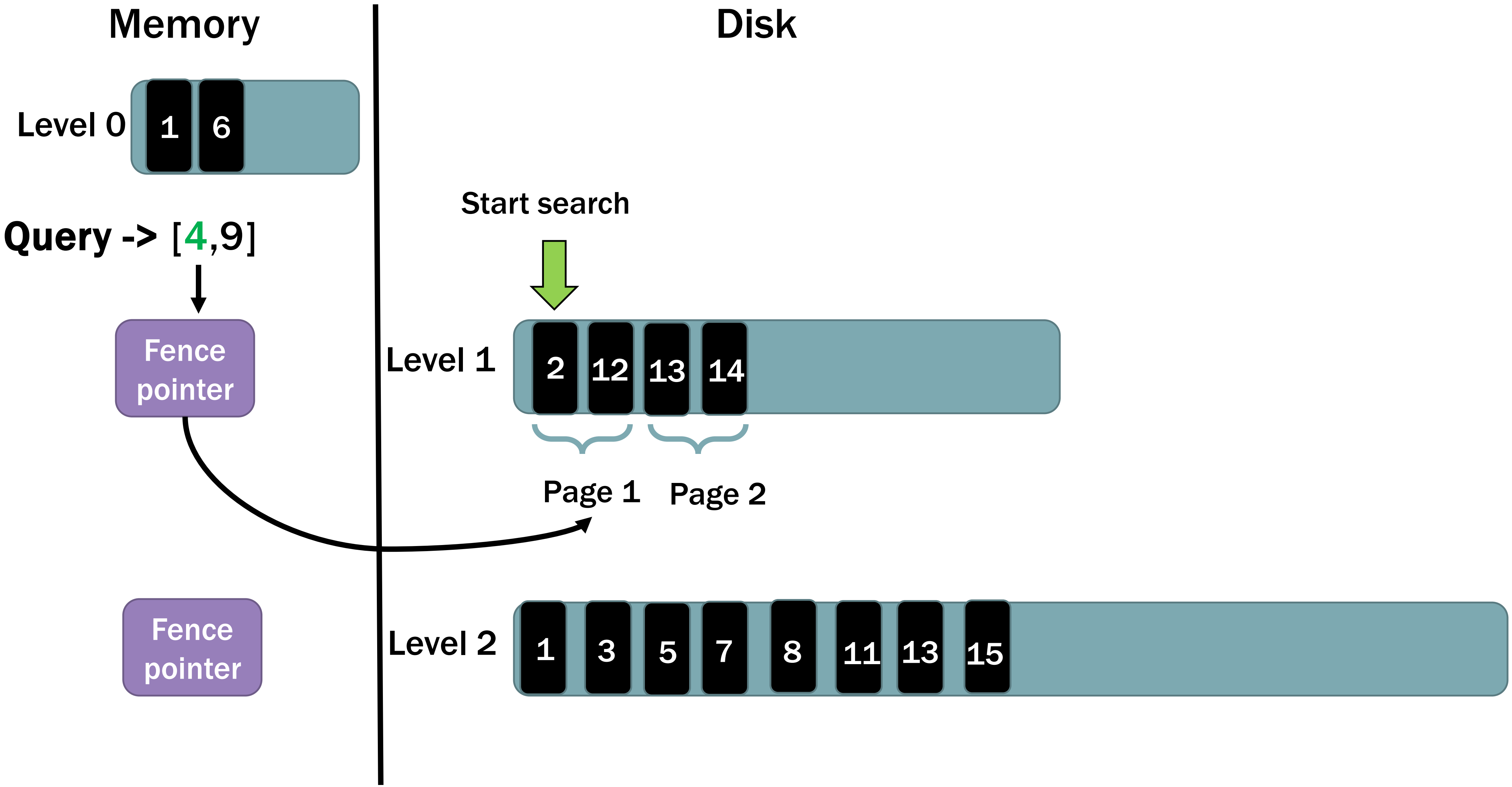
Level 1

2 I/Os

Level 2

4 I/Os

OPTIMIZATION – FENCE POINTERS



Memory

Disk

Level 0

1 6

Query -> [4,9]

Fence pointer

Start search

Level 1

2 12 13 14

Page 1

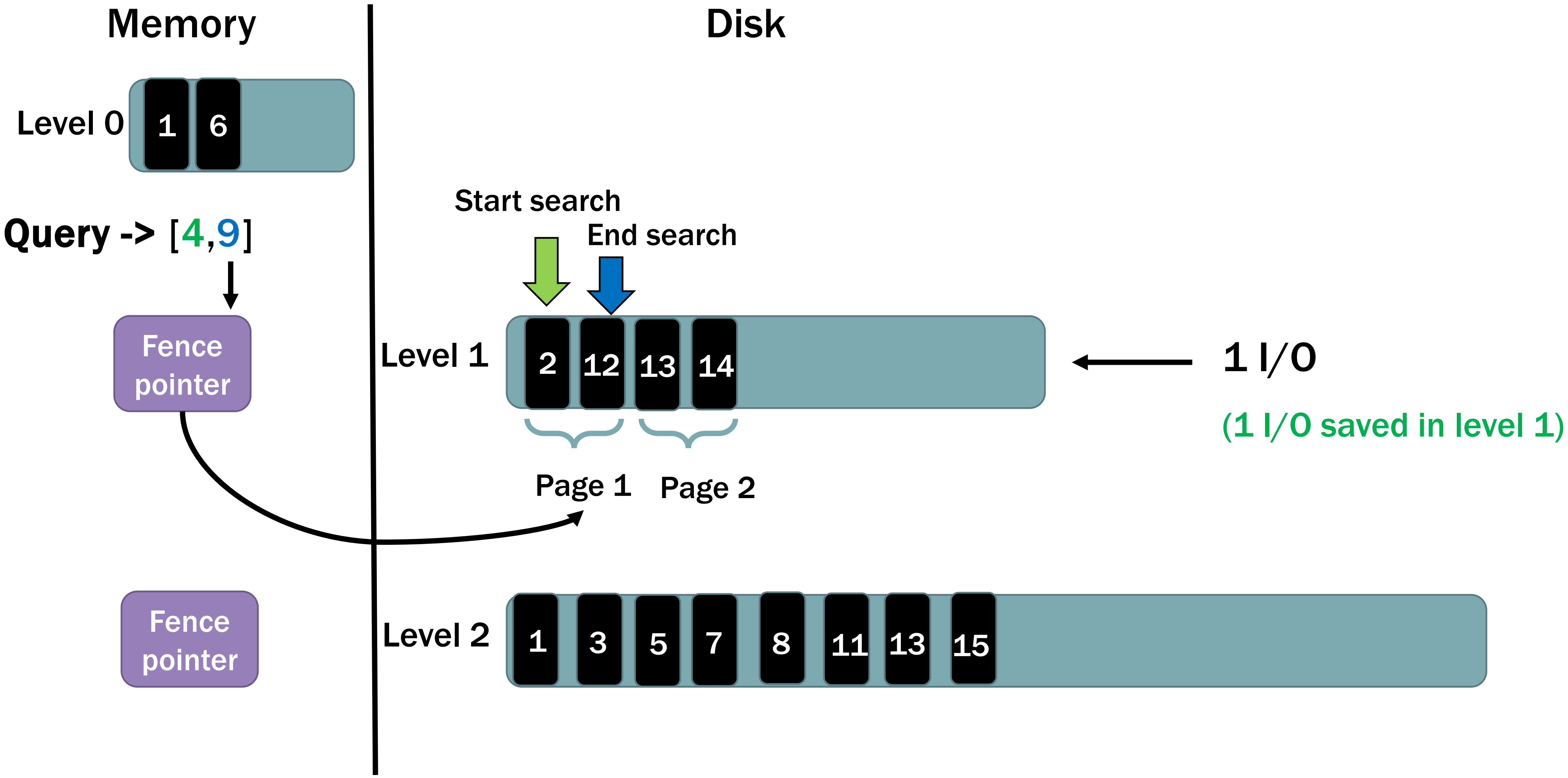
Page 2

Fence pointer

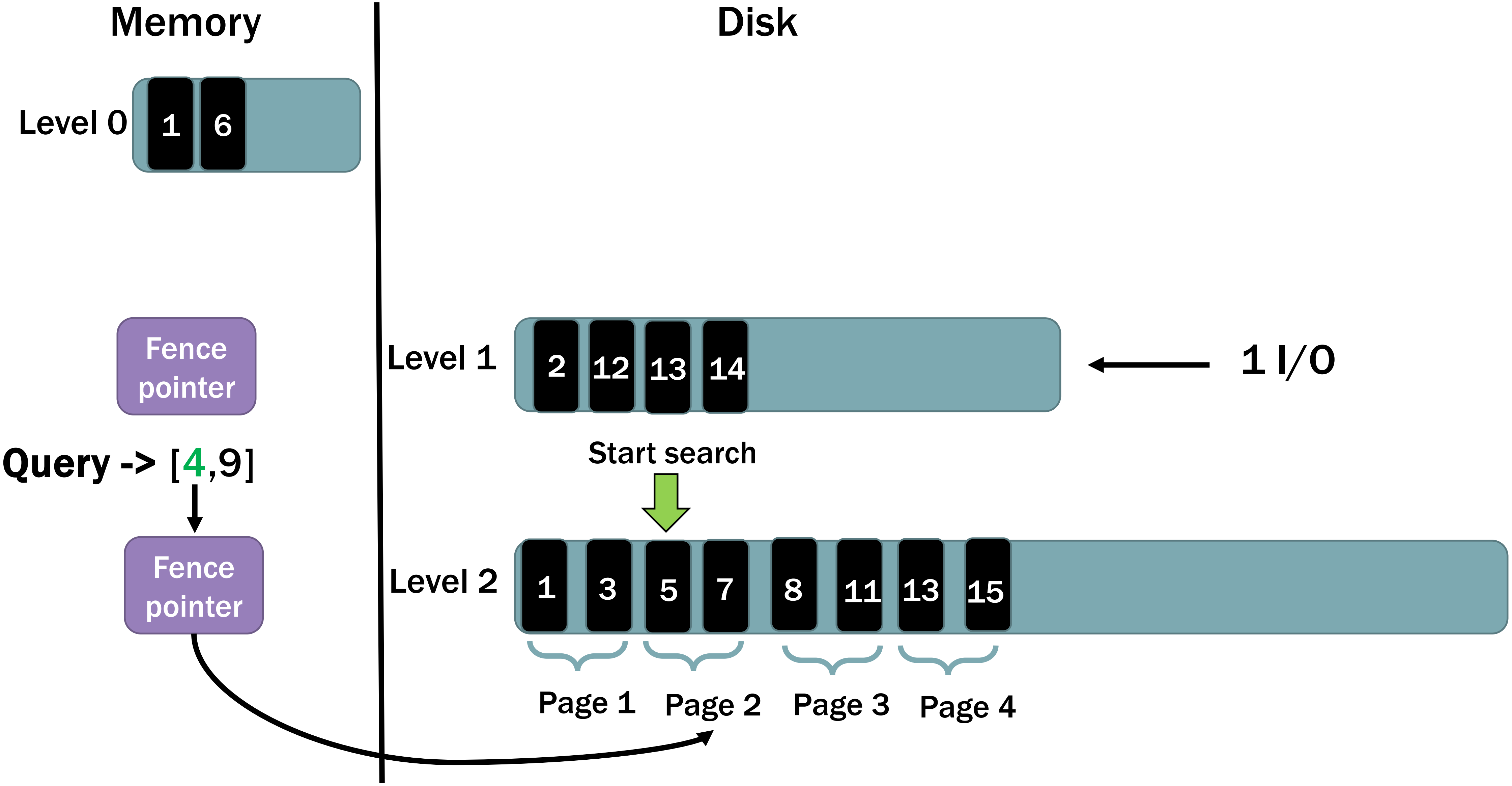
Level 2

1 3 5 7 8 11 13 15

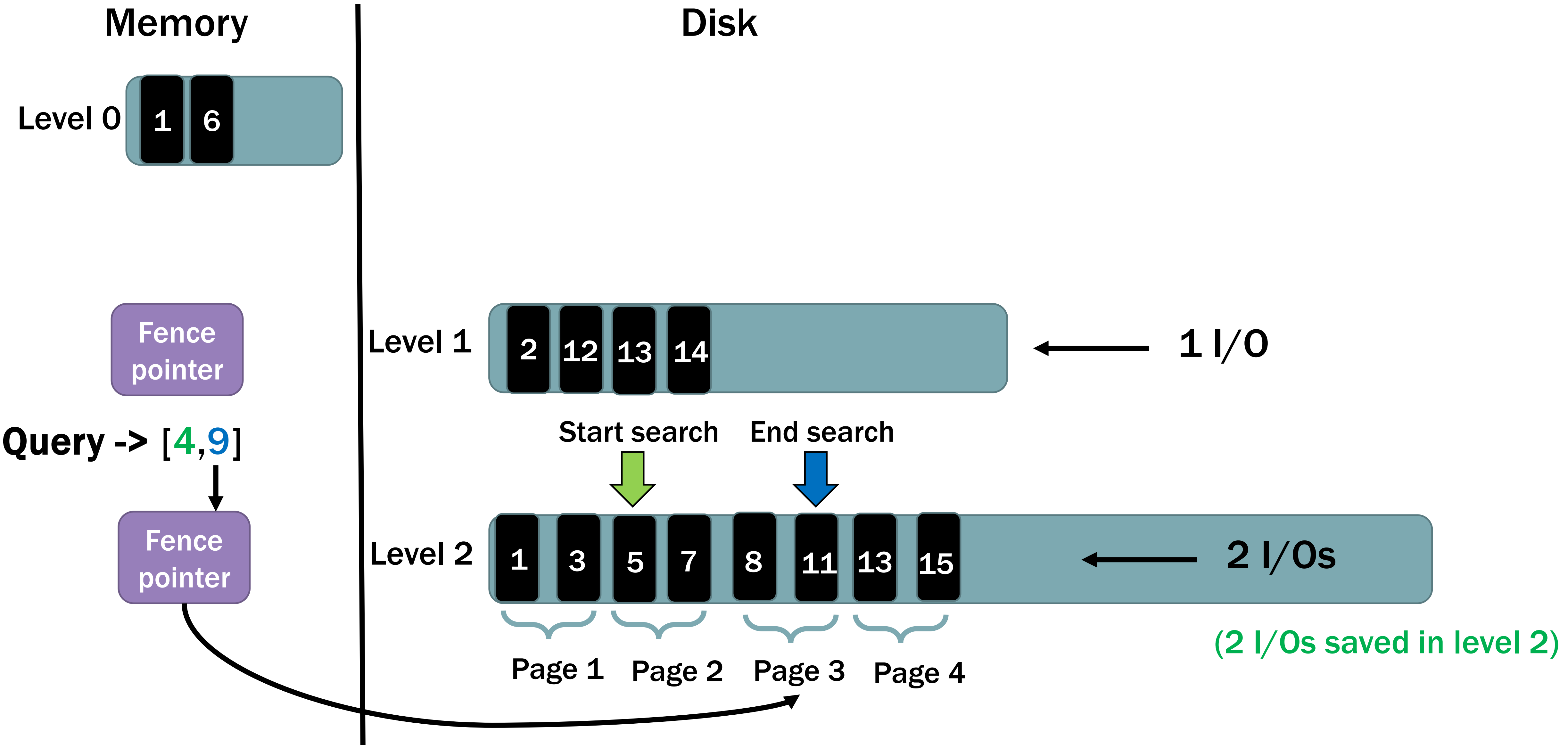
OPTIMIZATION – FENCE POINTERS



OPTIMIZATION – FENCE POINTERS



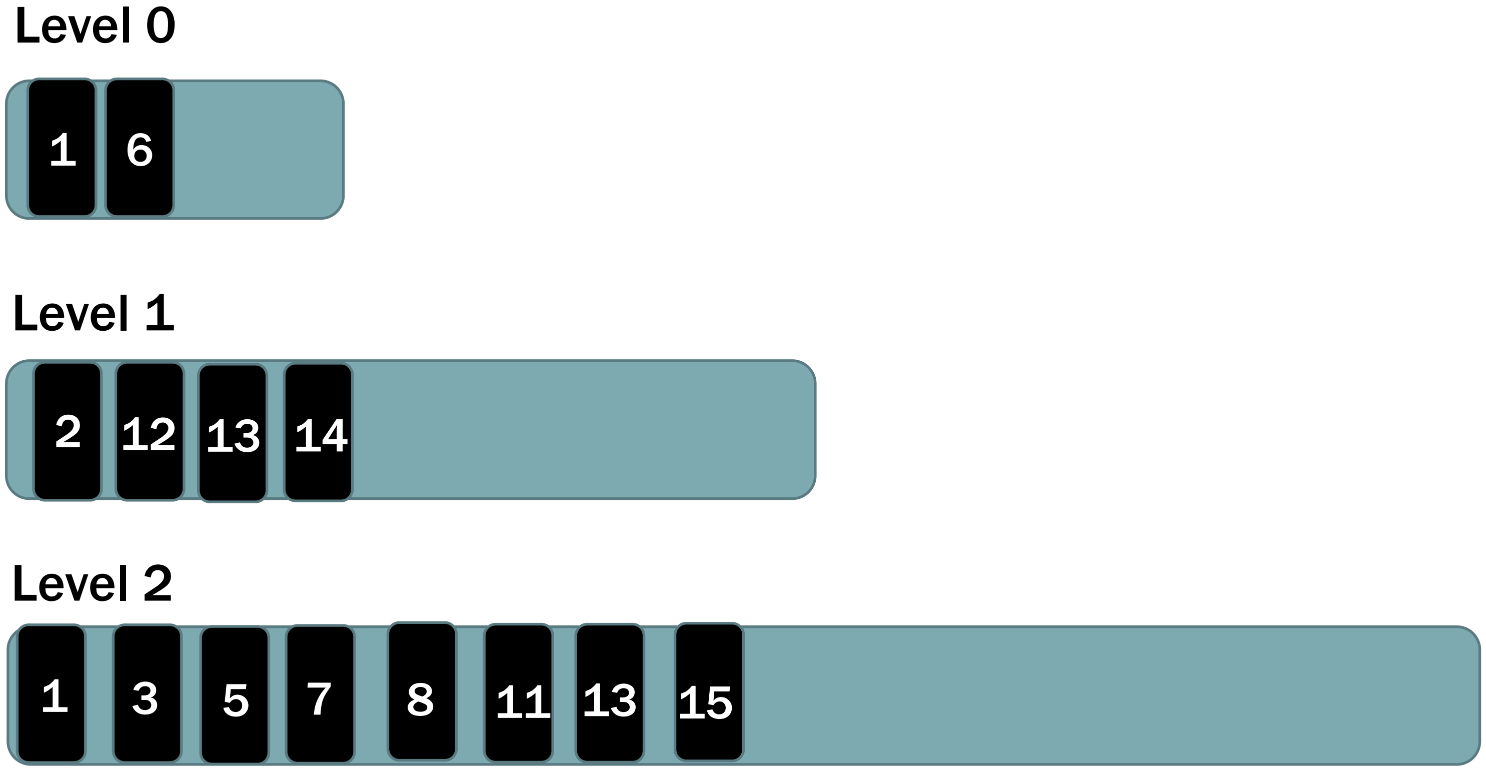
OPTIMIZATION – FENCE POINTERS



OPTIMIZATION – RANGE FILTERS (NOT EXAMINABLE)

Consider:

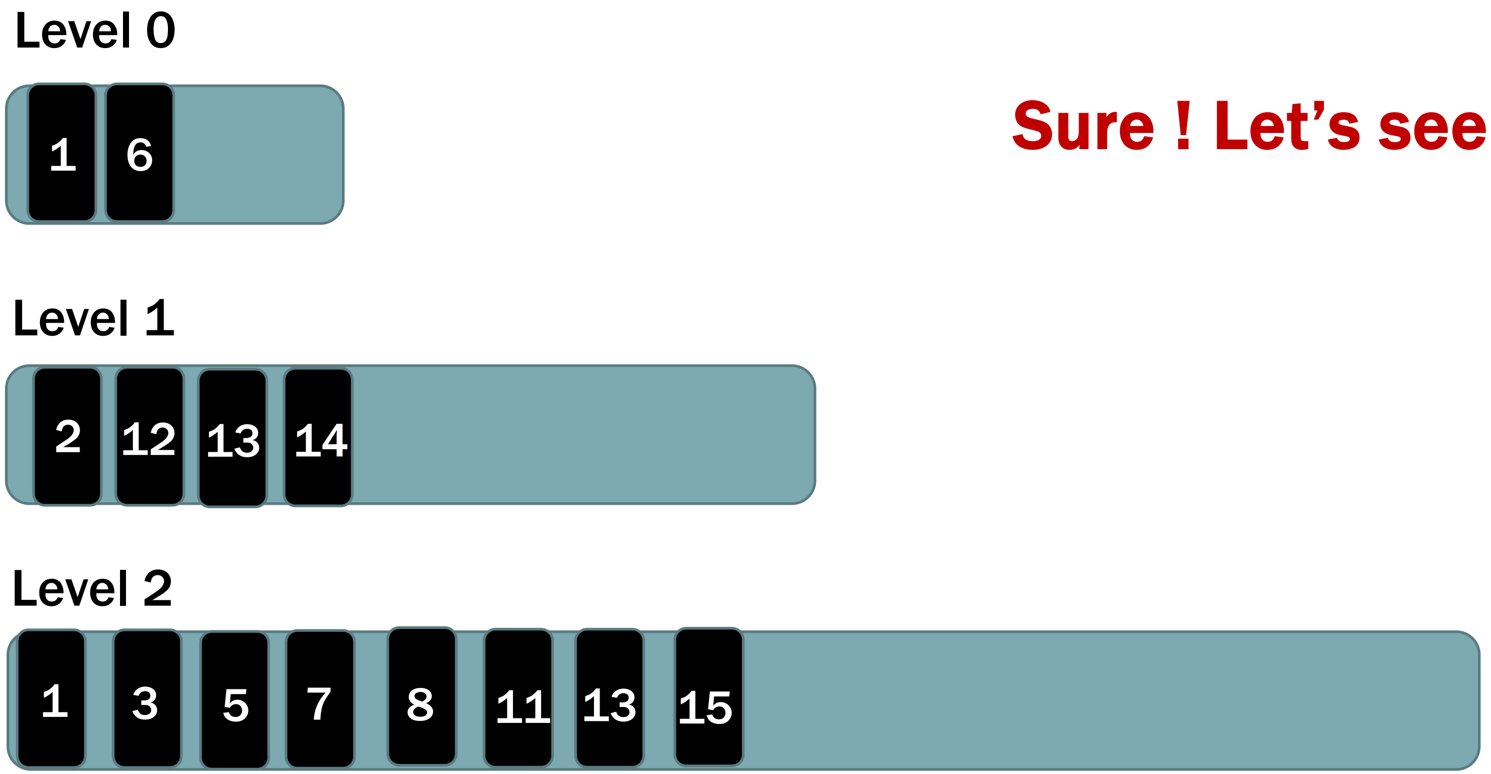
Can we introduce a filter for Range_Get like GET function to reduce I/O cost?



OPTIMIZATION – RANGE FILTERS

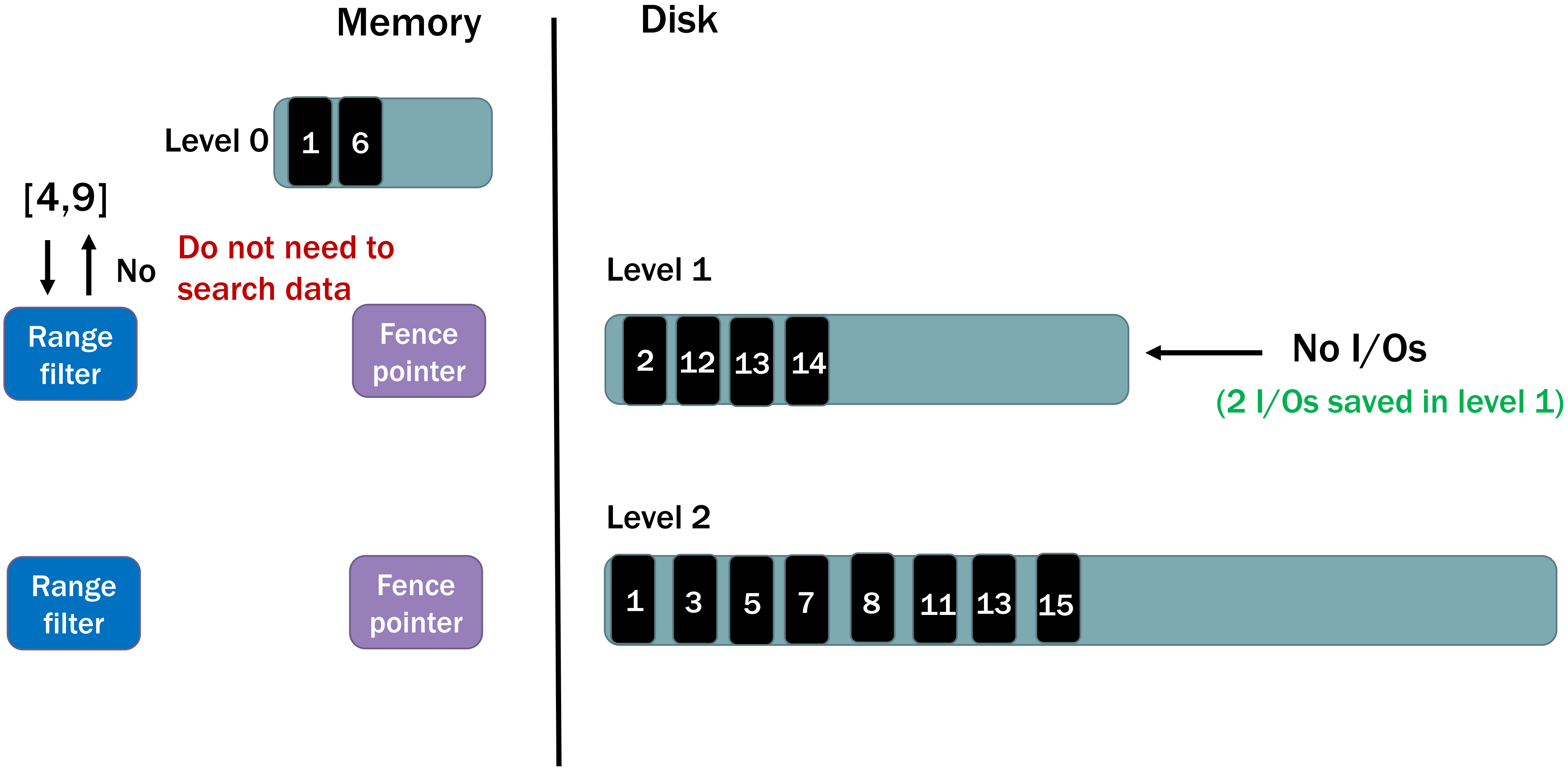
Consider:

Can we introduce a filter for Range_Get like GET function to reduce I/O cost?

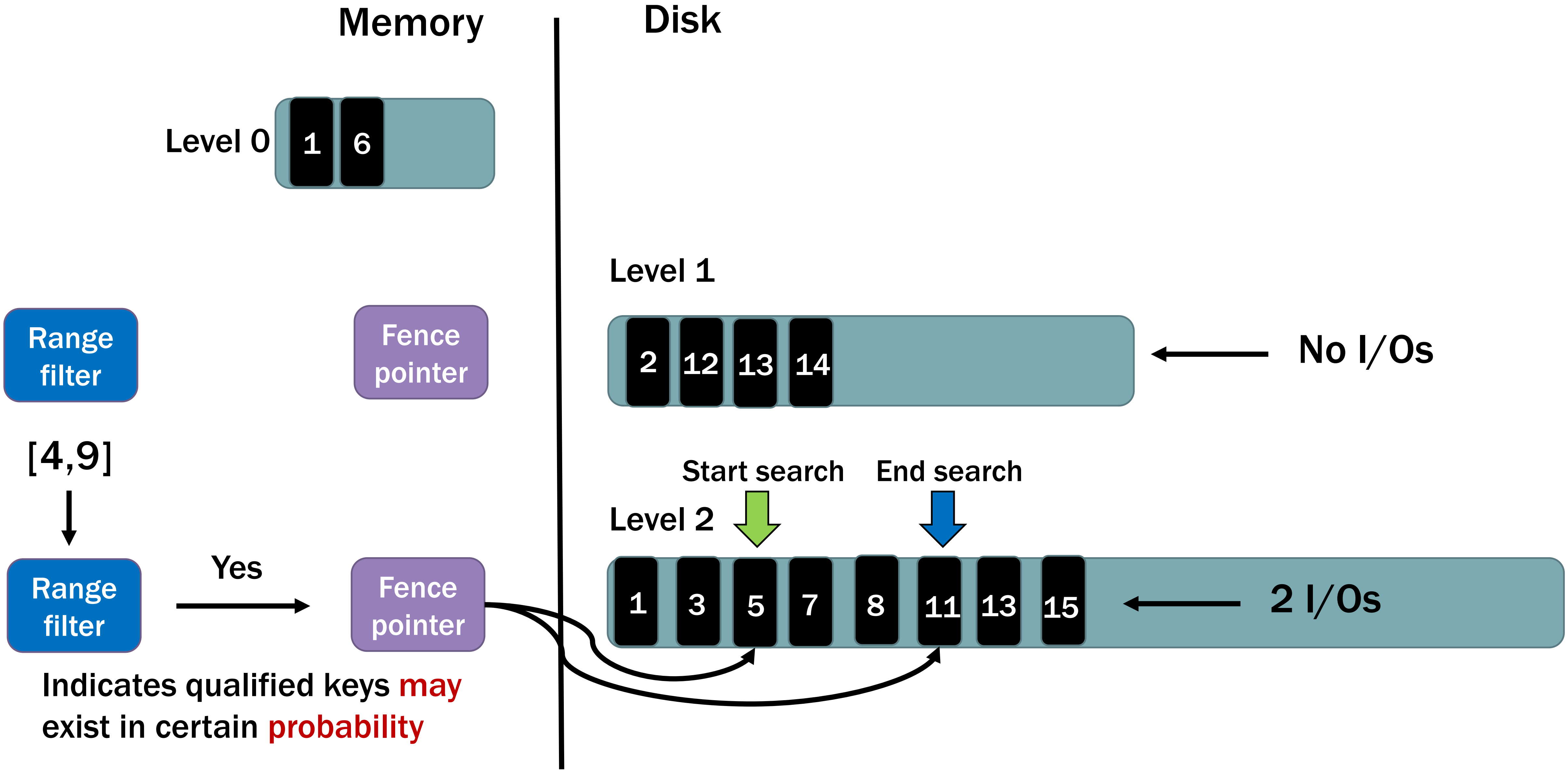


Sure ! Let's see how does it work.

OPTIMIZATION – RANGE FILTERS

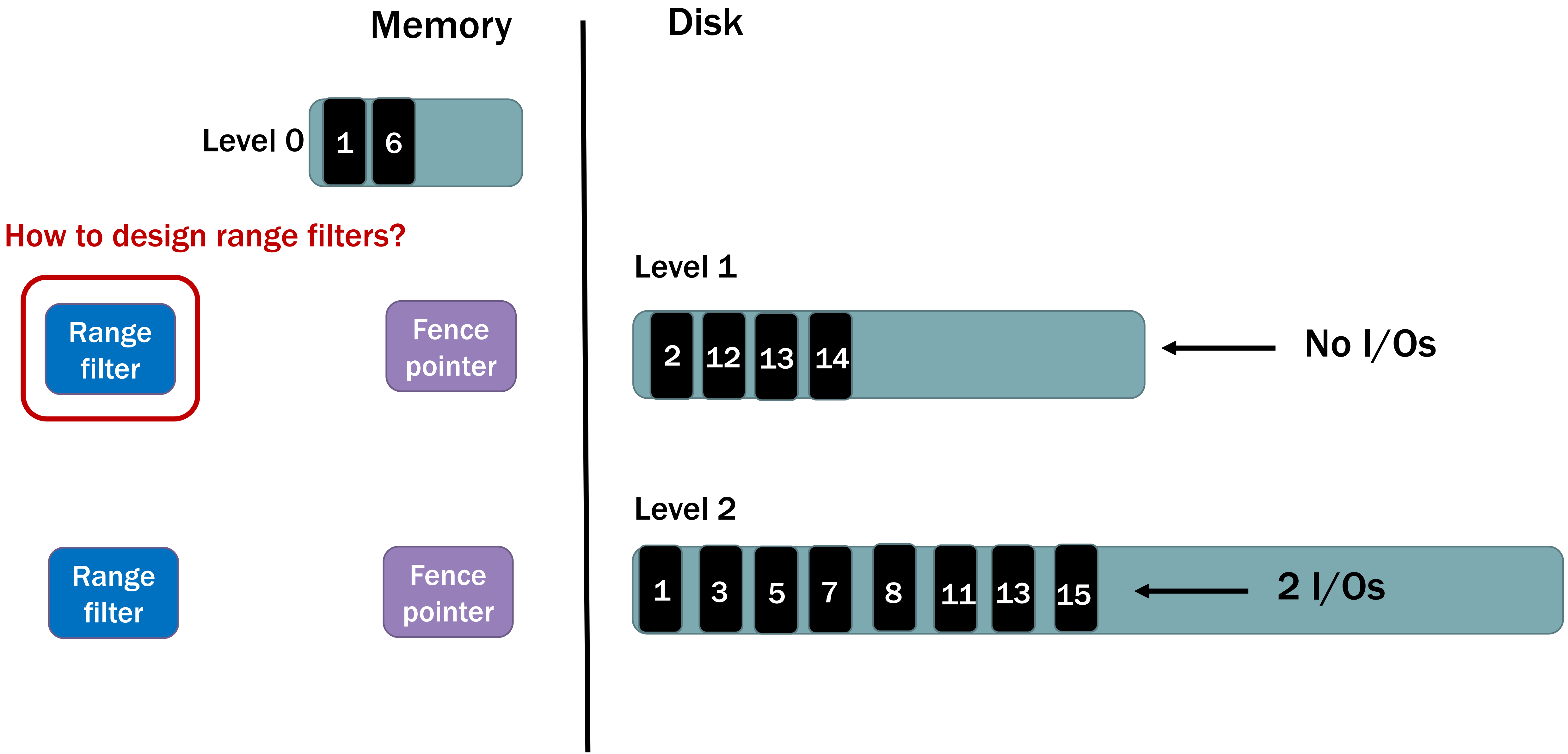


OPTIMIZATION – RANGE FILTERS



Indicates qualified keys may exist in certain probability

OPTIMIZATION – RANGE FILTERS



Memory

Disk

Level 0

1 6

How to design range filters?

Range filter

Fence pointer

Level 1

2 12 13 14

No I/Os

Range filter

Fence pointer

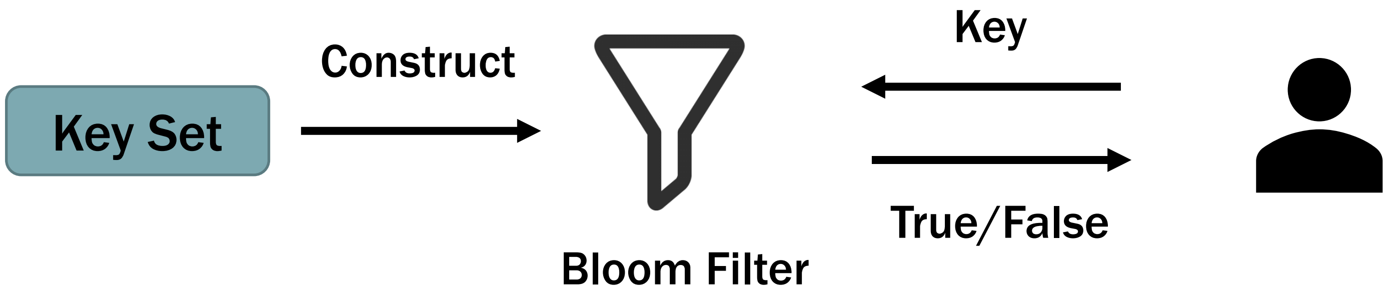
Level 2

1 3 5 7 8 11 13 15

2 I/Os

RANGE FILTER – PREFIX BLOOM FILTER

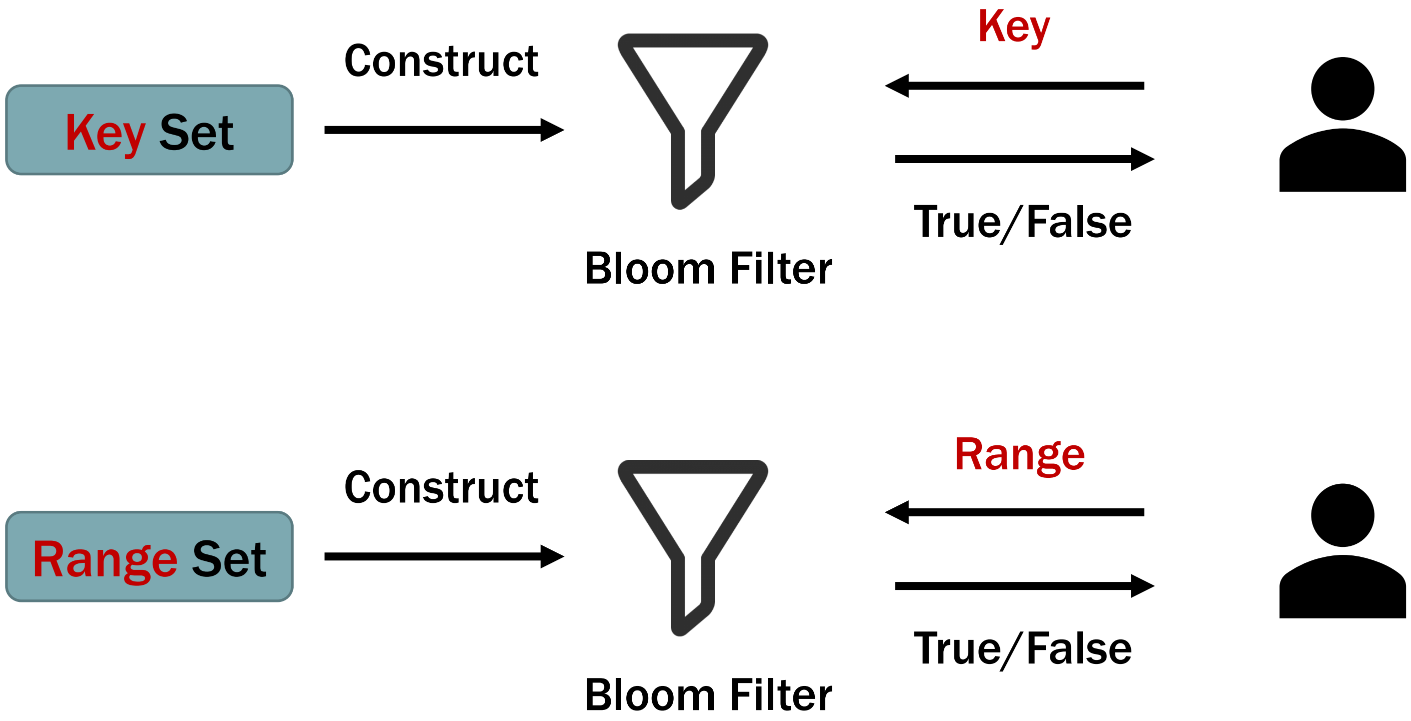
Revisit our old friend: Bloom filter



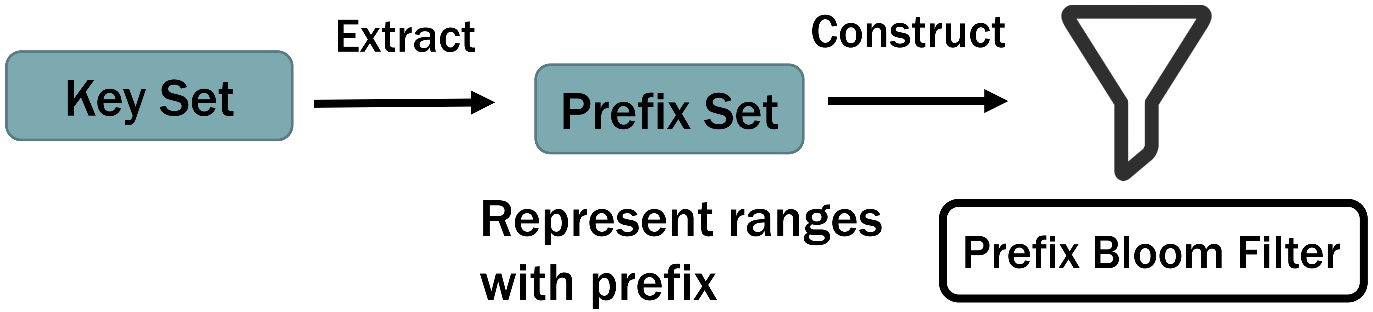
Can we extent to range query?

RANGE FILTER – PREFIX BLOOM FILTER

Extent Bloom filter to facilitate range query



RANGE FILTER – PREFIX BLOOM FILTER



Key Set

Extract

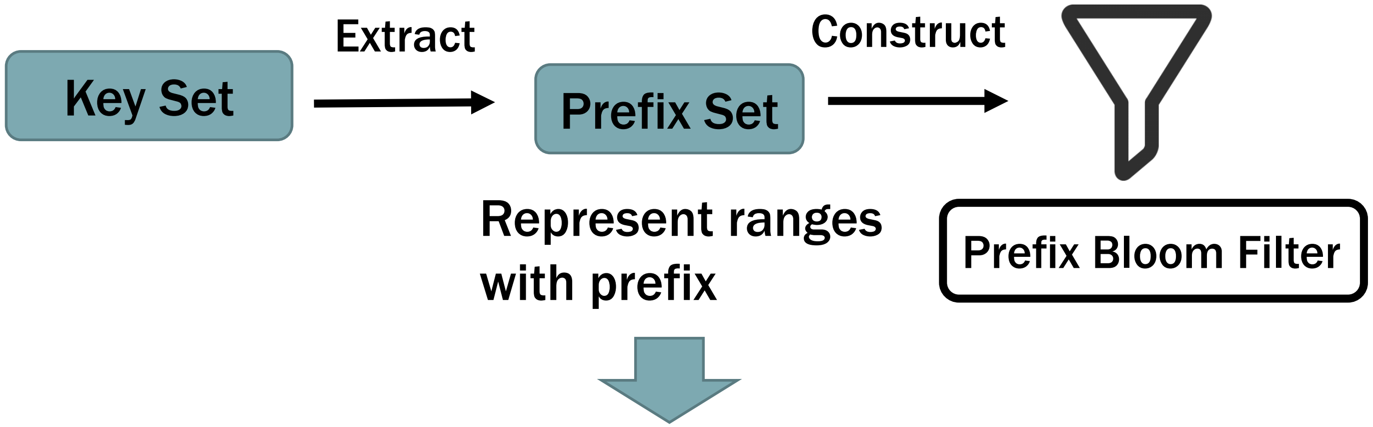
Prefix Set

Represent ranges with prefix

Construct

Prefix Bloom Filter

RANGE FILTER – PREFIX BLOOM FILTER



Key Set

Extract

Prefix Set

Construct

Prefix Bloom Filter

Represent ranges with prefix

Example: Assume the key space is [0-15] and prefix length is 2

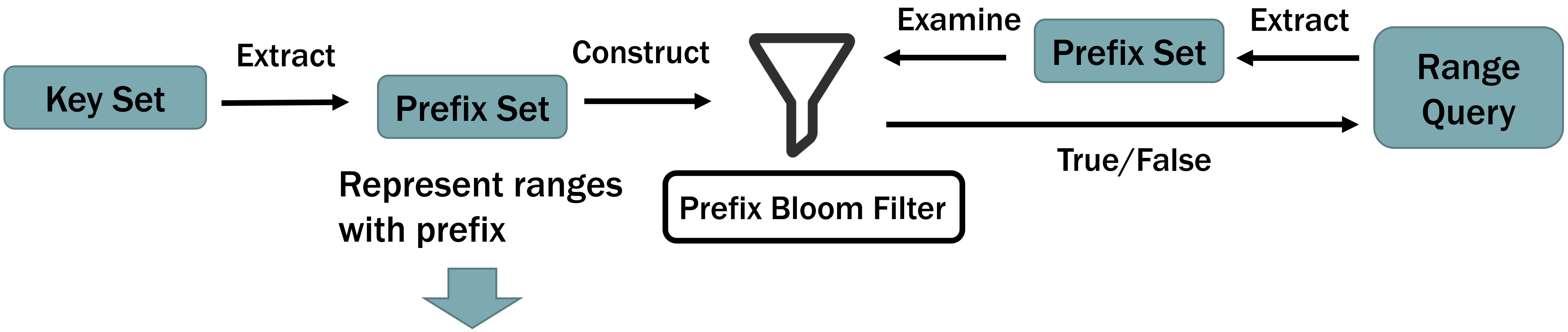
6 -> 0110

↑

prefix

prefix	range
00	[0-3]
01	[4-7]
10	[8-11]
11	[12-15]

RANGE FILTER – PREFIX BLOOM FILTER



Key Set

Extract

Prefix Set

Construct

Prefix Bloom Filter

Examine

Prefix Set

Extract

Range Query

True/False

Represent ranges with prefix

Example: Assume the key space is [0-15] and prefix length is 2

6 -> 0110

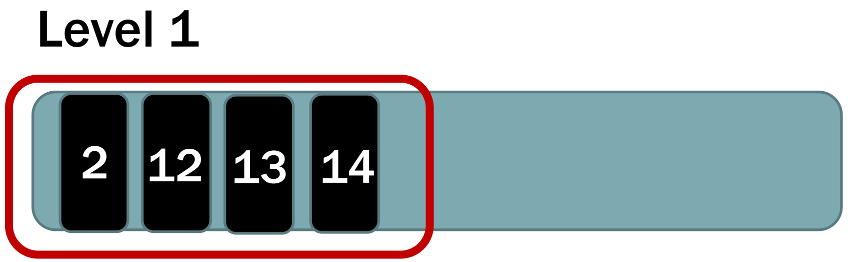
↑

prefix

prefix	range
00	[0-3]
01	[4-7]
10	[8-11]
11	[12-15]

RANGE FILTER – PREFIX BLOOM FILTER

Example: construct prefix bloom filter for level 1



Bloom filter info.

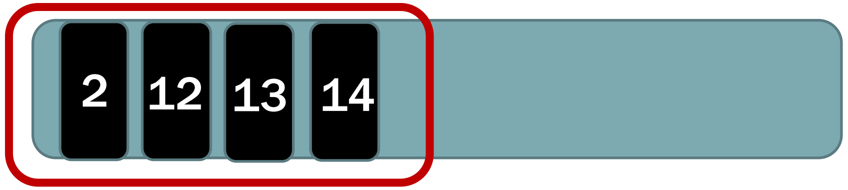
$h_1(x) = x \bmod 8$

$h_2(x) = (3x+5) \bmod 8$

RANGE FILTER – PREFIX BLOOM FILTER

Example: construct prefix bloom filter for level 1

Level 1



$h_1(x)$ = x mod 8

$h_2(x)$ = $(3x+5)$ mod 8

2 -> 0010

12 -> 1100

13 -> 1101

14 -> 1110

Extract prefix →

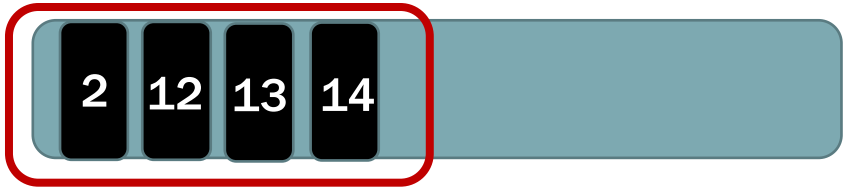
00(0)

11(3)

RANGE FILTER – PREFIX BLOOM FILTER

Example: construct prefix bloom filter for level 1

Level 1



$h_1(x) = x \bmod 8$

$h_2(x) = (3x+5) \bmod 8$

2 -> 0010

12 -> 1100

13 -> 1101

14 -> 1110

Extract prefix →

00(0) → $h_1(0) = 0$, $h_2(0) = 5$ →

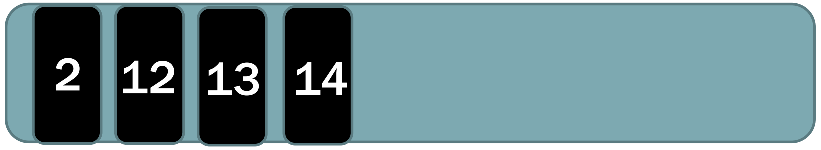
1	0	0	0	0	1	0	0

11(3)

RANGE FILTER – PREFIX BLOOM FILTER

Example: construct prefix bloom filter for level 1

Level 1



$h_1(x)$ = x mod 8

$h_2(x)$ = $(3x+5)$ mod 8

2 -> 0010

12 -> 1100

13 -> 1101

14 -> 1110

Extract prefix →

00(0) →

1	0	0	0	0	1	0	0

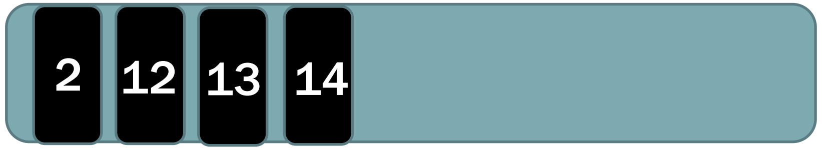
11(3) → $h_1(3) = 3$, $h_2(3) = 6$ →

0	0	0	1	0	0	1	0

RANGE FILTER – PREFIX BLOOM FILTER

Response to the range query

Level 1



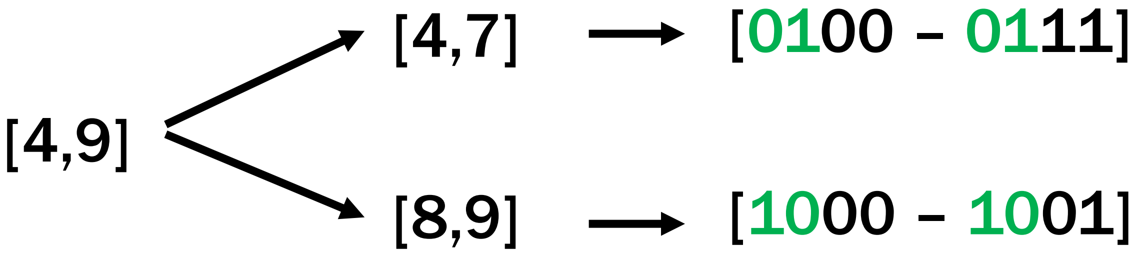
$h_1(x) = x \bmod 8$

$h_2(x) = (3x+5) \bmod 8$

Prefix Bloom Filter

1	0	0	1	0	1	1	0

Query -> [4,9]



[4,9]

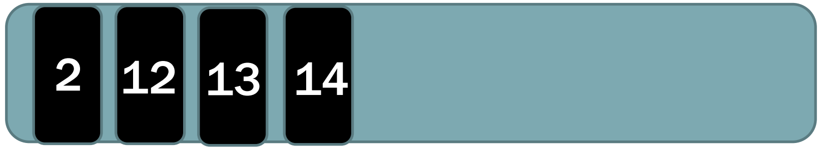
[4,7] → [0100 – 0111]

[8,9] → [1000 – 1001]

RANGE FILTER – PREFIX BLOOM FILTER

Response to the range query

Level 1



2 12 13 14

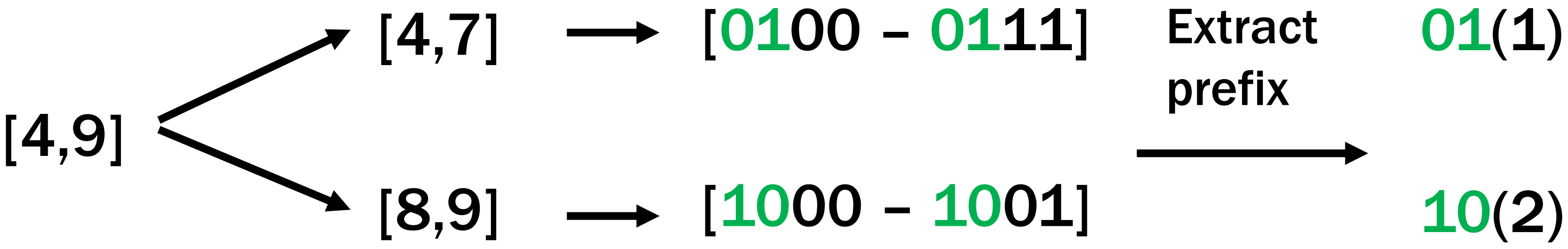
$h_1(x) = x \bmod 8$

$h_2(x) = (3x+5) \bmod 8$

Prefix Bloom Filter

1	0	0	1	0	1	1	0

Query -> [4,9]



[4,9] → [4,7] → [0100 – 0111]

[4,9] → [8,9] → [1000 – 1001]

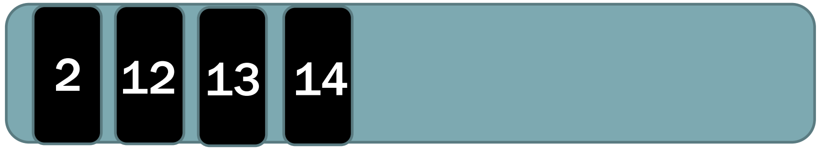
Extract prefix →

01(1)

10(2)

RANGE FILTER – PREFIX BLOOM FILTER

Level 1



2 12 13 14

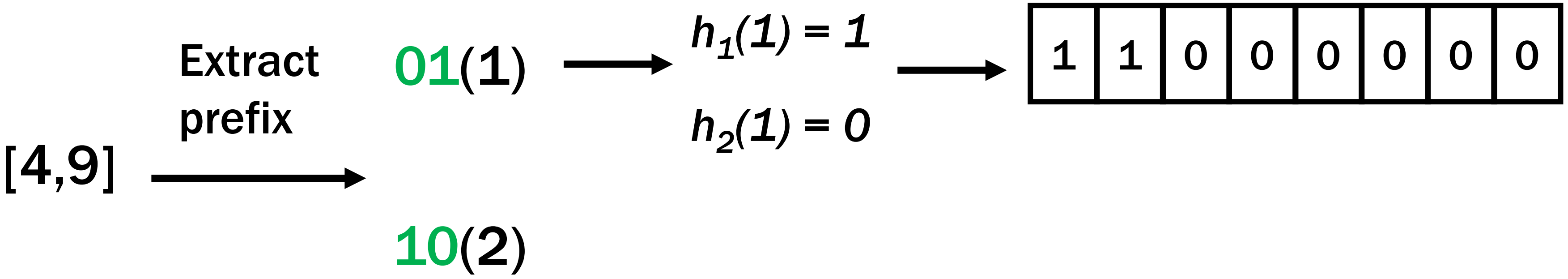
$h_1(x) = x \bmod 8$

$h_2(x) = (3x+5) \bmod 8$

Prefix Bloom Filter

1	0	0	1	0	1	1	0

Query -> [4,9]

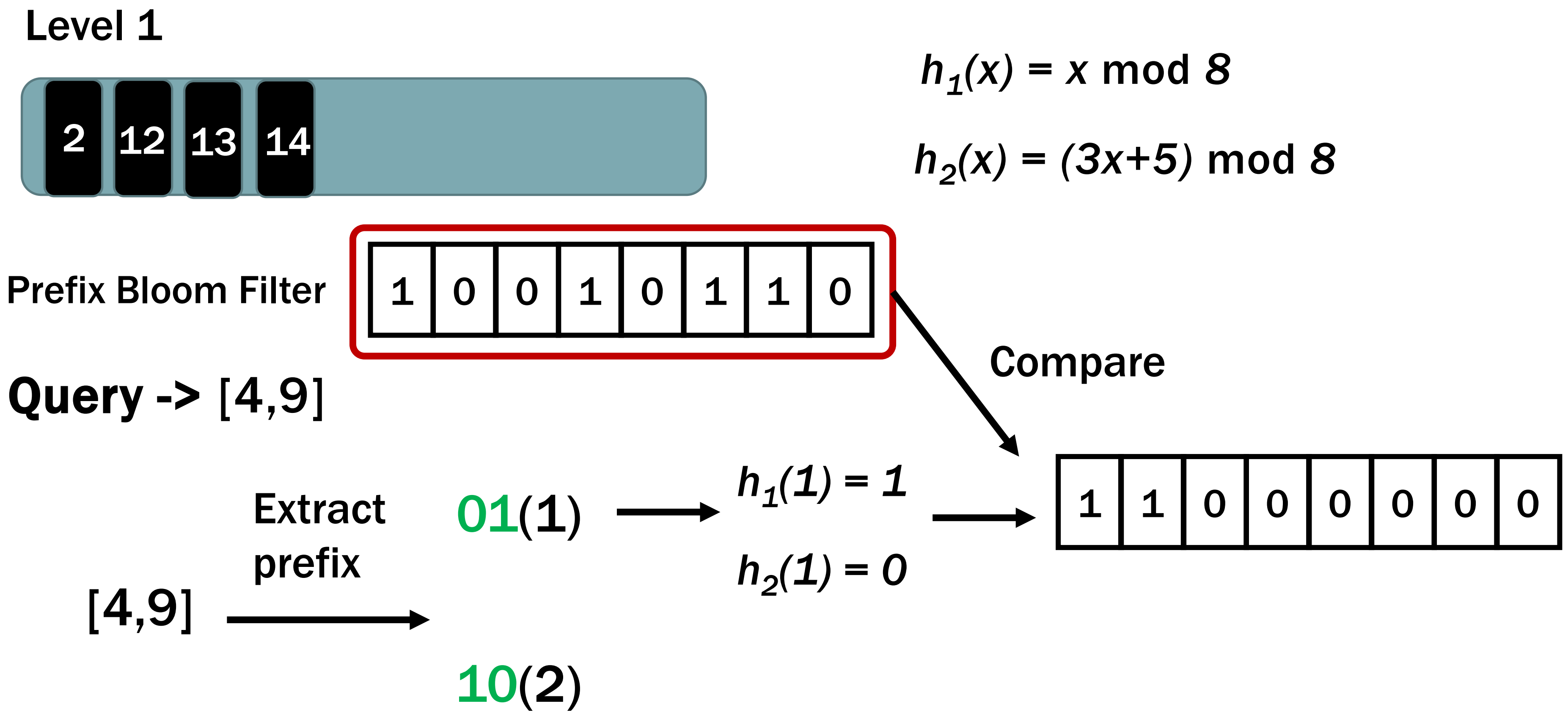


[4,9] → Extract prefix → 01(1), 10(2)

01(1) → $h_1(1) = 1$, $h_2(1) = 0$ →

1	1	0	0	0	0	0	0

RANGE FILTER – PREFIX BLOOM FILTER



Level 1

2	12	13	14

$h_1(x) = x \bmod 8$

$h_2(x) = (3x+5) \bmod 8$

Prefix Bloom Filter

1	0	0	1	0	1	1	0

Compare

Query -> [4,9]

[4,9] → Extract prefix

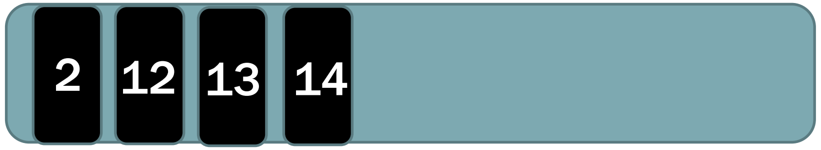
$\color{green}{01}(1) \rightarrow h_1(1) = 1$, $h_2(1) = 0$ →

1	1	0	0	0	0	0	0

$\color{green}{10}(2)$

RANGE FILTER – PREFIX BLOOM FILTER

Level 1



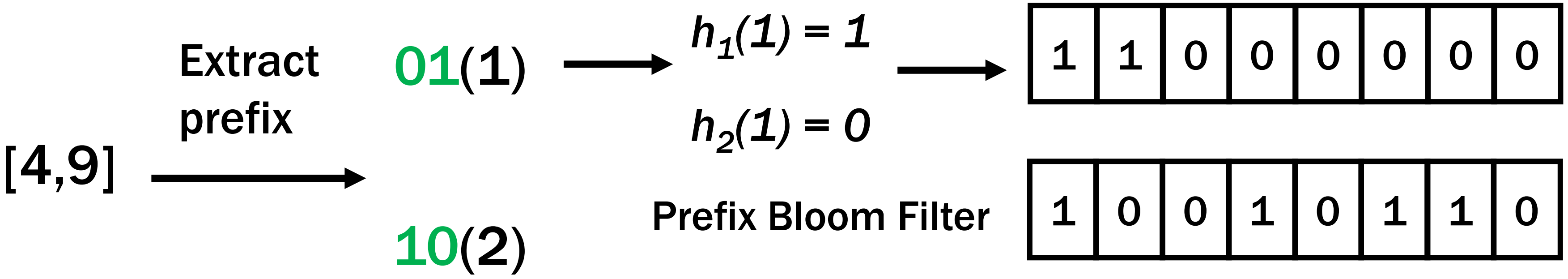
$h_1(x) = x \bmod 8$

$h_2(x) = (3x+5) \bmod 8$

Prefix Bloom Filter

1	0	0	1	0	1	1	0

Query -> [4,9]



[4,9] $\longrightarrow$ Extract prefix

01(1) $\longrightarrow$ $h_1(1) = 1$, $h_2(1) = 0$ $\longrightarrow$

1	1	0	0	0	0	0	0

10(2)

Prefix Bloom Filter

1	0	0	1	0	1	1	0

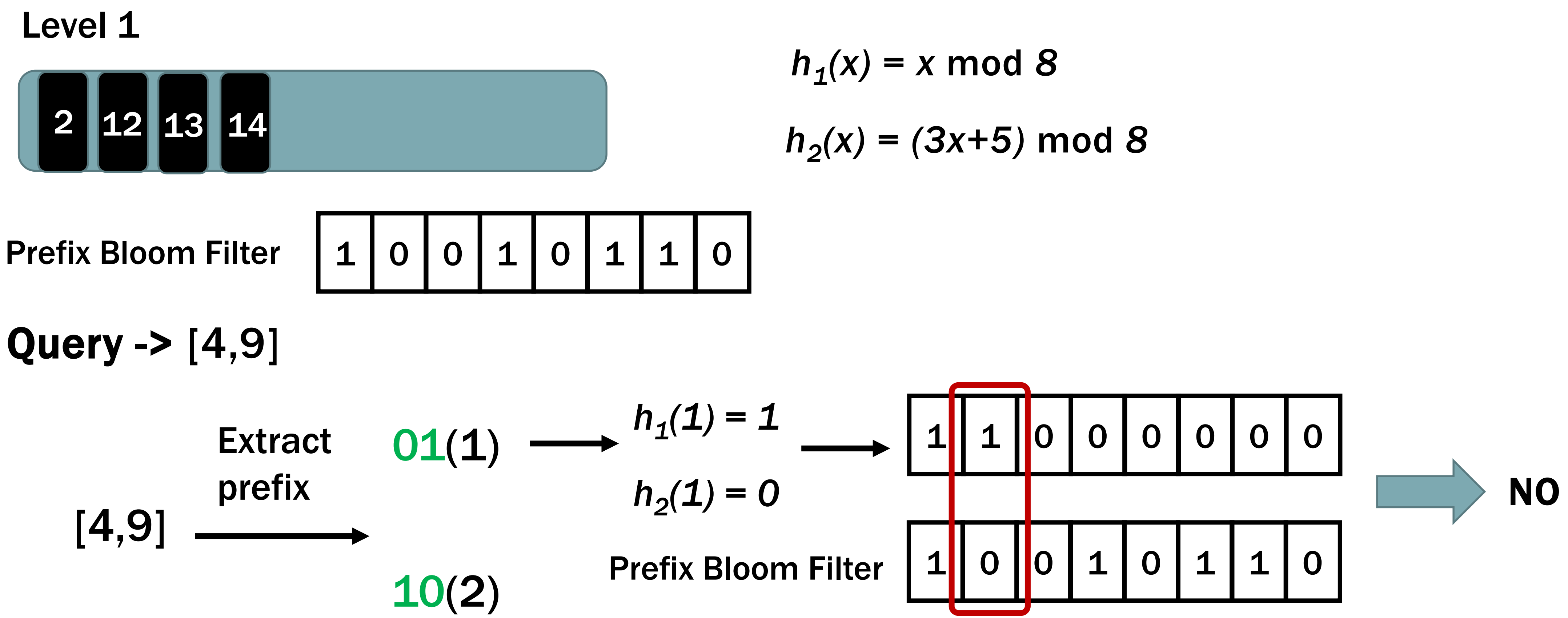
If every “1” bit is set to “1” in Prefix Bloom filter:

Response YES

Else:

Response NO

RANGE FILTER – PREFIX BLOOM FILTER



Level 1

2 12 13 14

$h_1(x)$ = x mod 8

$h_2(x)$ = $(3x+5)$ mod 8

Prefix Bloom Filter

1	0	0	1	0	1	1	0

Query -> [4,9]

[4,9] $\xrightarrow{\text{Extract prefix}}$

01(1) → $h_1(1) = 1$, $h_2(1) = 0$ →

1	1	0	0	0	0	0	0

10(2)

Prefix Bloom Filter

1	0	0	1	0	1	1	0

NO

If every “1” bit is set to “1” in Prefix Bloom filter:

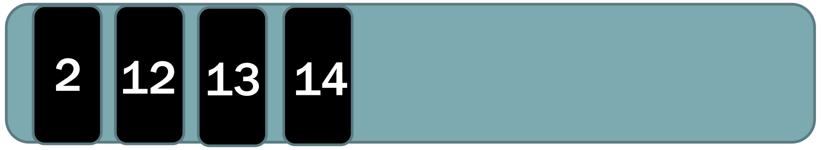
Response YES

Else:

Response NO

RANGE FILTER – PREFIX BLOOM FILTER

Level 1



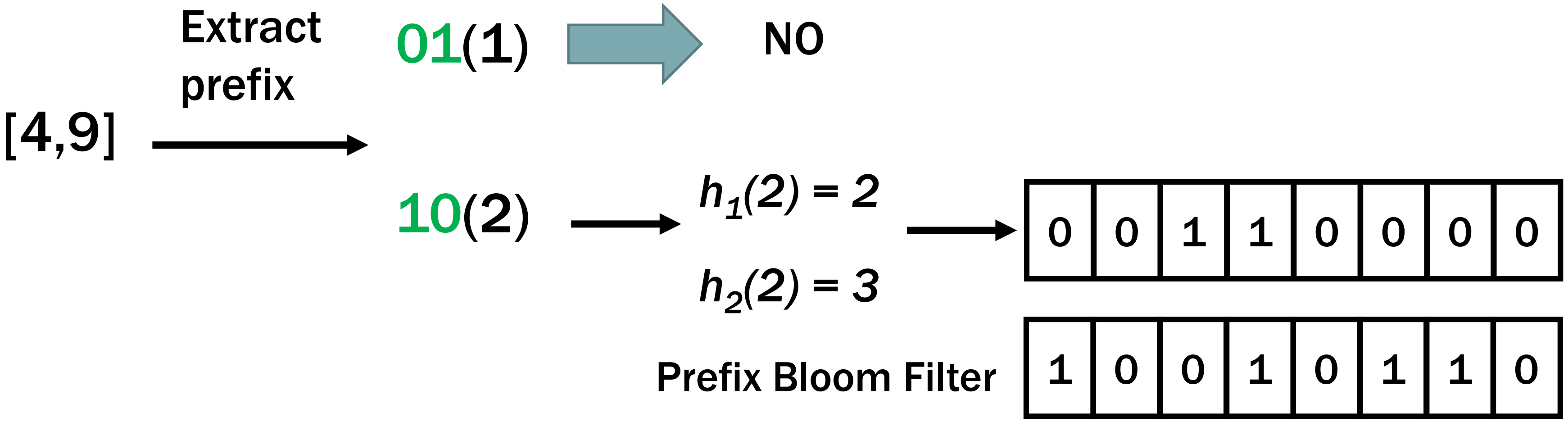
$h_1(x)$ = x mod 8

$h_2(x)$ = (3x+5) mod 8

Prefix Bloom Filter

1	0	0	1	0	1	1	0

Query -> [4,9]



[4,9] Extract prefix →

01(1) → NO

10(2) → $h_1(2) = 2$, $h_2(2) = 3$ →

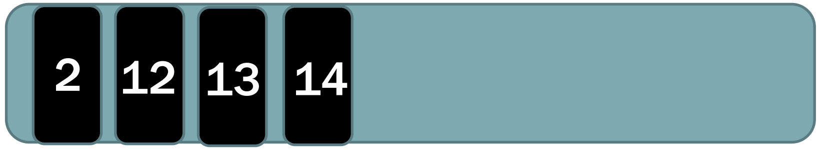
0	0	1	1	0	0	0	0

Prefix Bloom Filter

1	0	0	1	0	1	1	0

RANGE FILTER – PREFIX BLOOM FILTER

Level 1



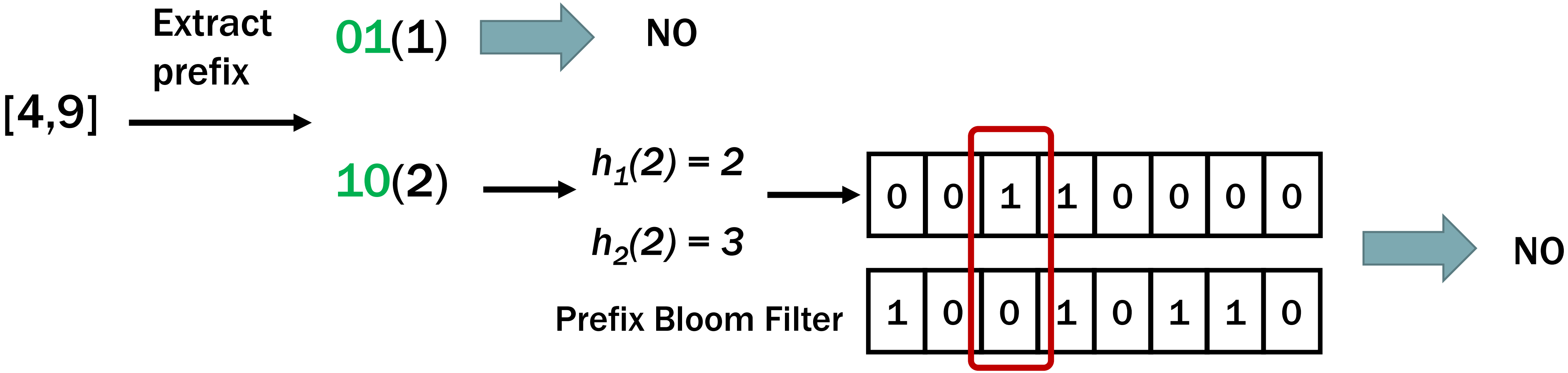
$h_1(x) = x \bmod 8$

$h_2(x) = (3x+5) \bmod 8$

Prefix Bloom Filter

1	0	0	1	0	1	1	0

Query -> [4,9]



[4,9] Extract prefix

01(1) NO

10(2) $h_1(2) = 2$ $h_2(2) = 3$

0	0	1	1	0	0	0	0

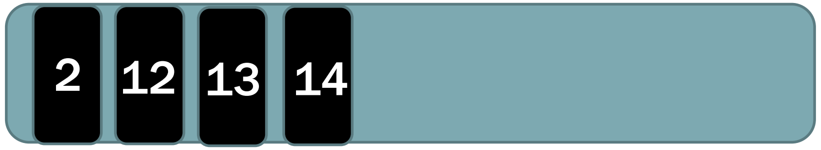
Prefix Bloom Filter

1	0	0	1	0	1	1	0

NO

RANGE FILTER – PREFIX BLOOM FILTER

Level 1



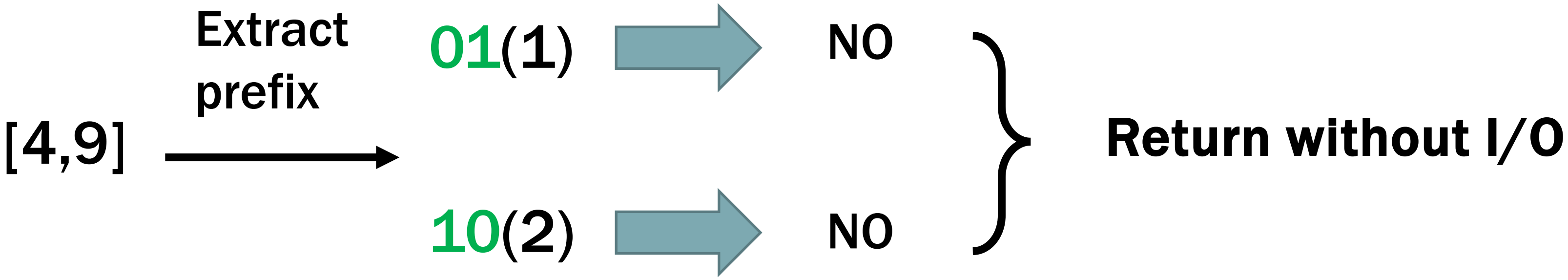
$h_1(x) = x \bmod 8$

$h_2(x) = (3x+5) \bmod 8$

Prefix Bloom Filter

1	0	0	1	0	1	1	0

Query -> [4,9]



[4,9]

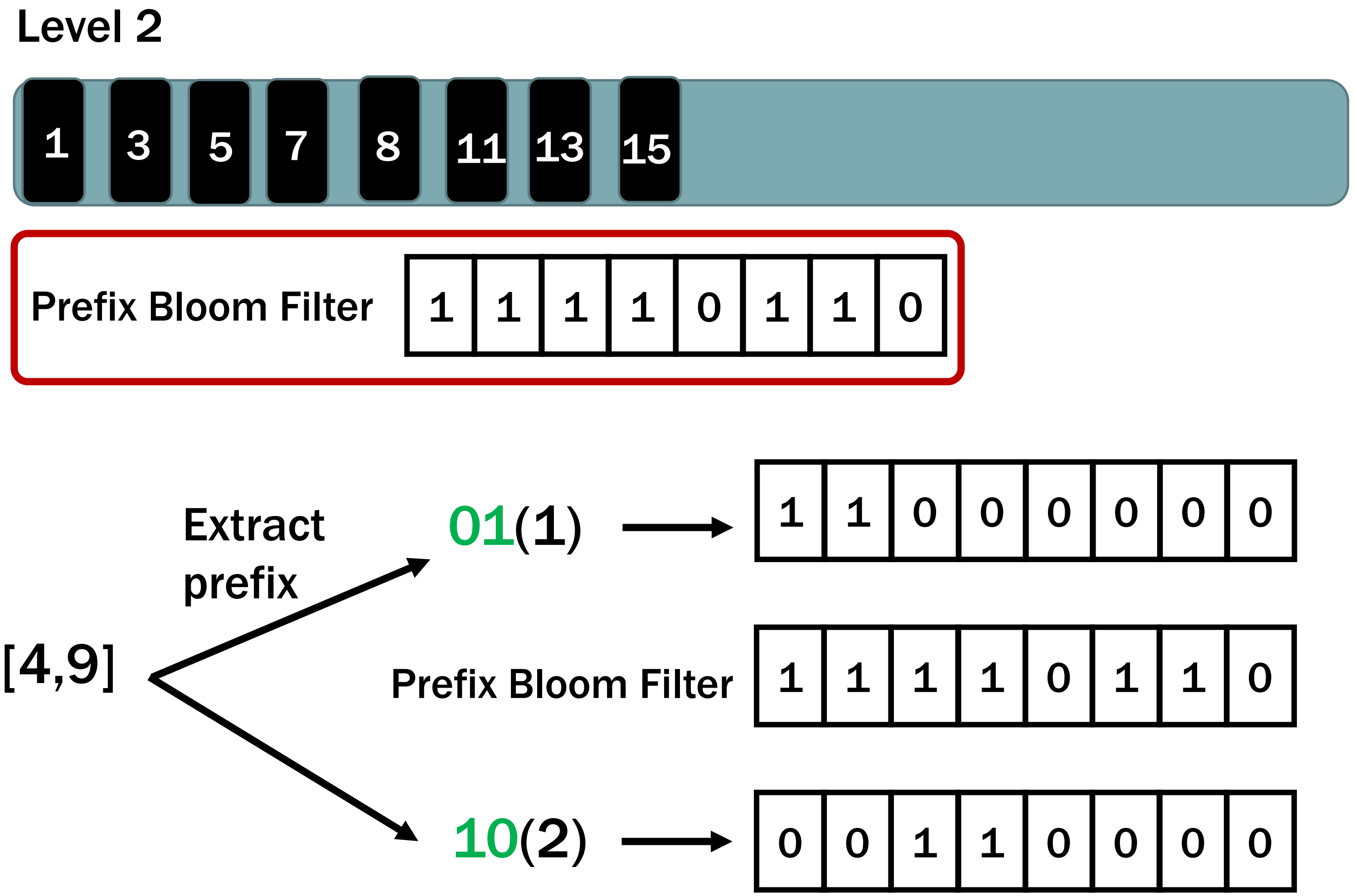
Extract prefix

01(1) → NO

10(2) → NO

Return without I/O

RANGE FILTER – PREFIX BLOOM FILTER



Level 2

1	3	5	7	8	11	13	15

Prefix Bloom Filter

1	1	1	1	0	1	1	0

[4,9]

Extract prefix

01(1) →

1	1	0	0	0	0	0	0

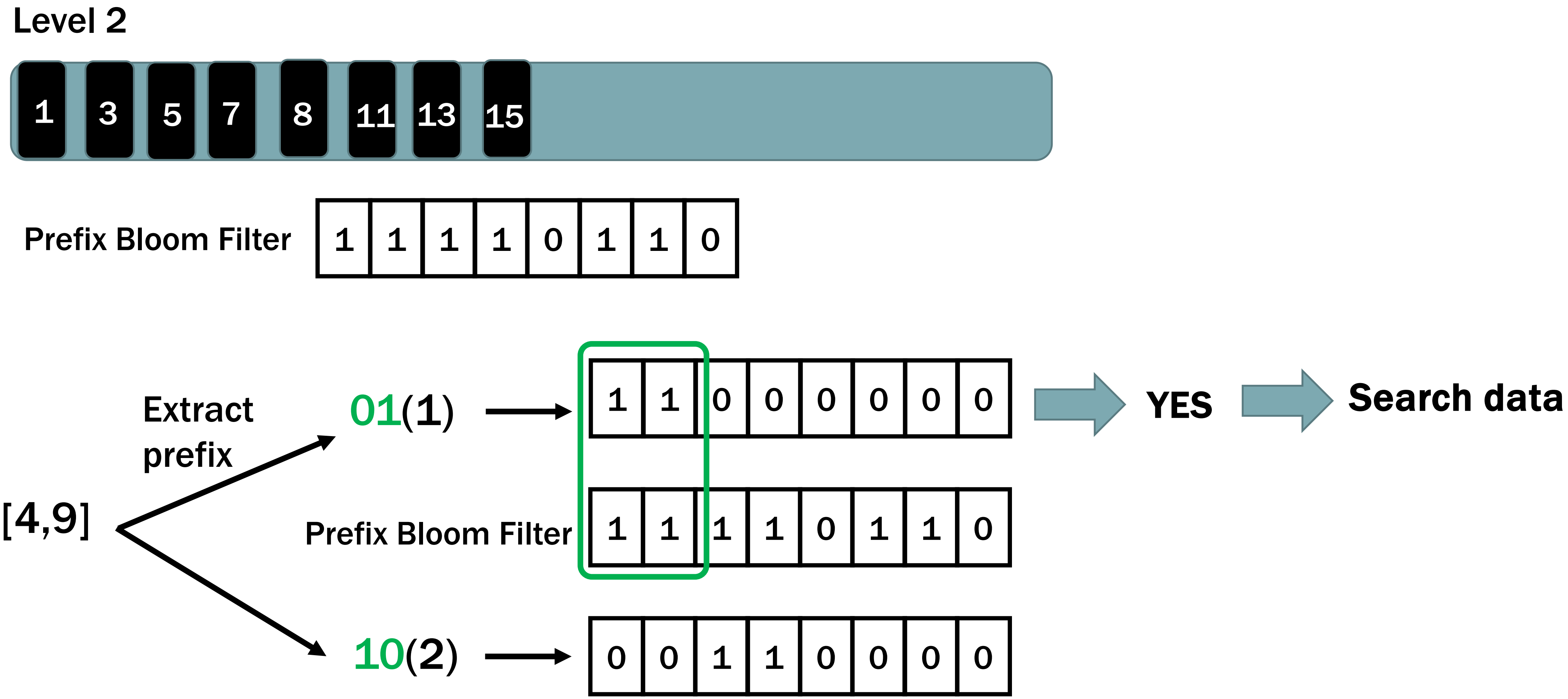
Prefix Bloom Filter

1	1	1	1	0	1	1	0

10(2) →

0	0	1	1	0	0	0	0

RANGE FILTER – PREFIX BLOOM FILTER



Level 2

1 3 5 7 8 11 13 15

Prefix Bloom Filter 1 1 1 1 0 1 1 0

[4,9]

Extract prefix

01(1)

1 1 0 0 0 0 0 0

YES

Search data

Prefix Bloom Filter 1 1 1 1 0 1 1 0

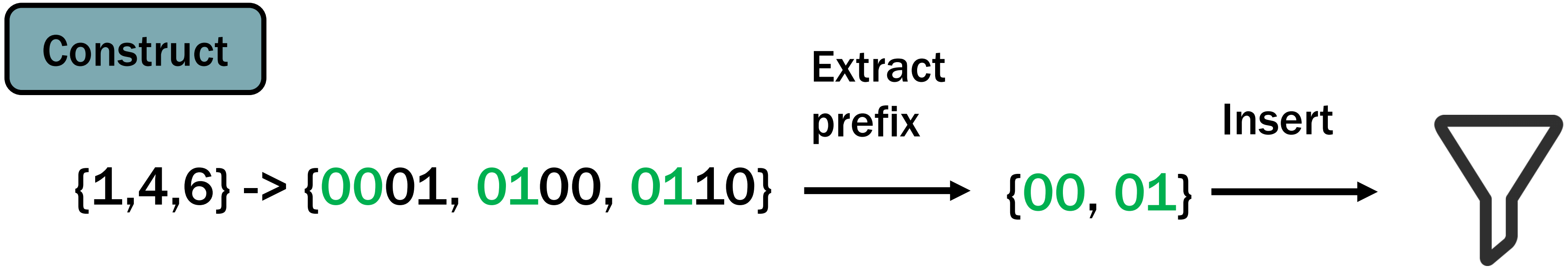
10(2)

0 0 1 1 0 0 0 0

RANGE FILTER – PREFIX BLOOM FILTER

Consider:

How to construct Prefix Bloom filter with the key set {1,4,6} and query with the range [9-14]?



Construct

{1,4,6} -> {0001, 0100, 0110}

Extract prefix

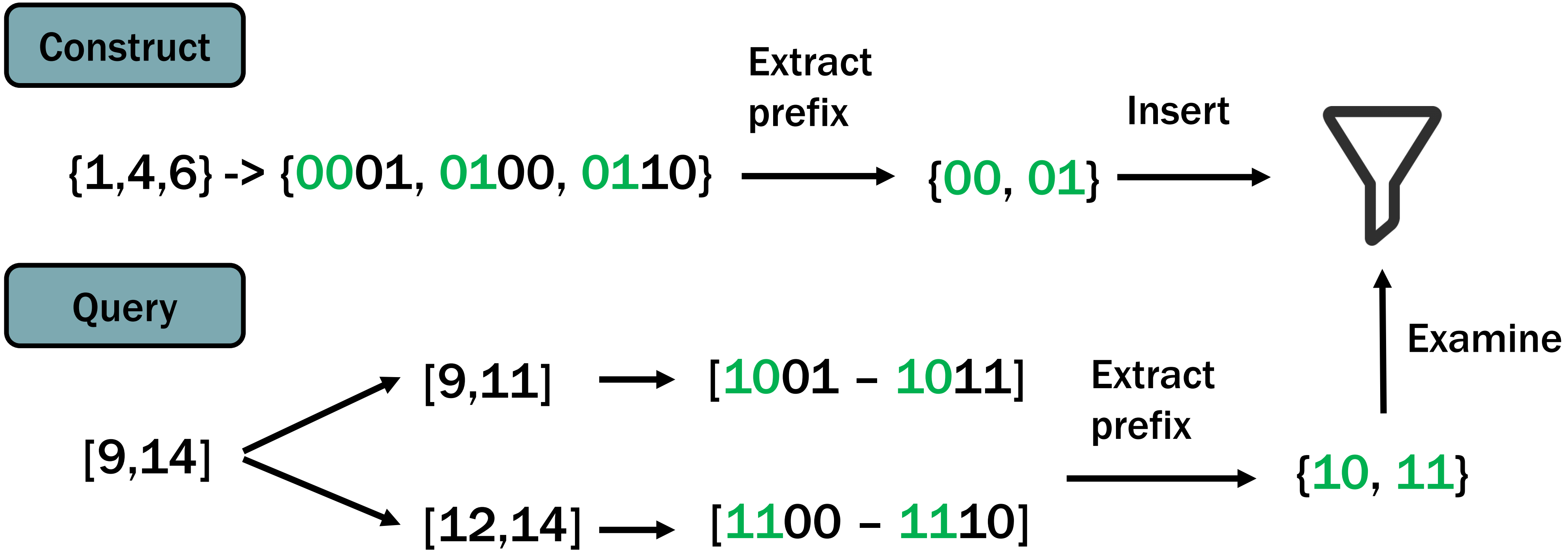
{00, 01}

Insert

RANGE FILTER – PREFIX BLOOM FILTER

Consider:

How to construct Prefix Bloom filter with the key set {1,4,6} and query with the range [9-14]?



Construct

{1,4,6} -> {0001, 0100, 0110} Extract prefix → {00, 01} Insert →

Query

[9,14]

[9,11] → [1001 – 1011]

[12,14] → [1100 – 1110]

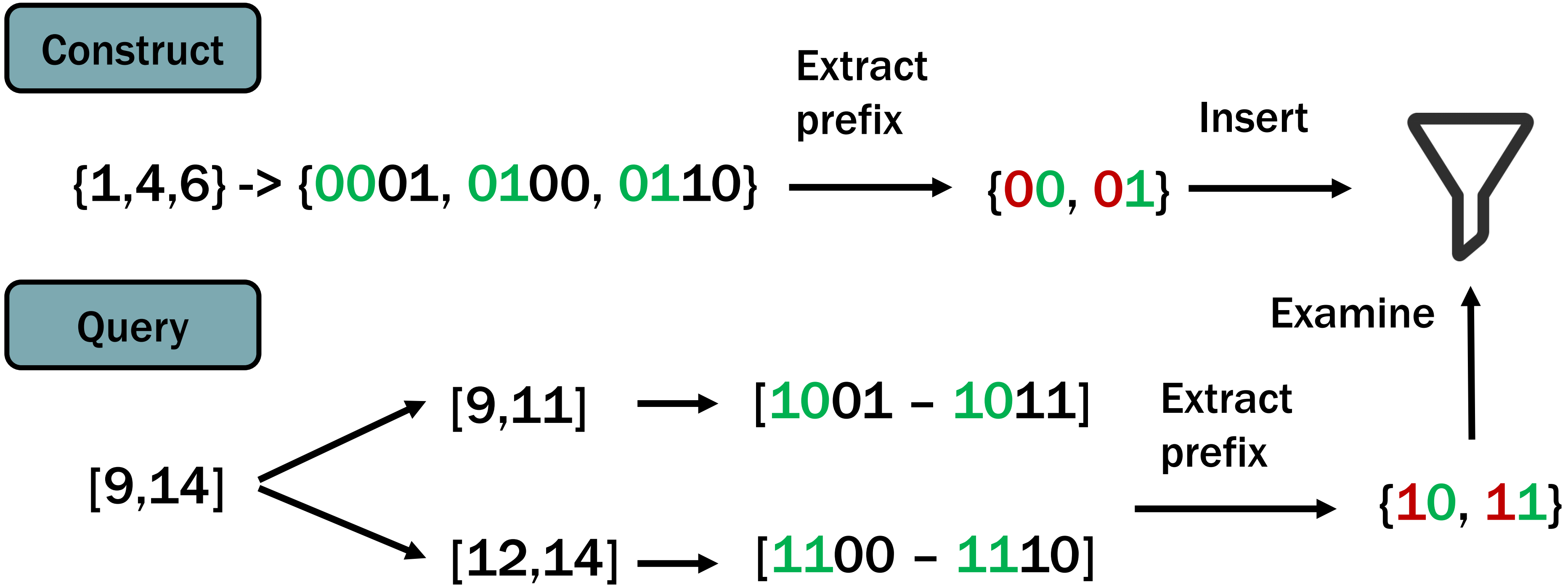
Extract prefix → {10, 11}

Examine

RANGE FILTER – PREFIX BLOOM FILTER

Consider:

How to construct Prefix Bloom filter with the key set {1,4,6} and query with the range [9-14]?



Construct

{1,4,6} -> {0001, 0100, 0110} → Extract prefix → {00, 01} → Insert → filter

Query

[9,14] → [9,11] → [1001 – 1011]

[9,14] → [12,14] → [1100 – 1110]

Extract prefix → {10, 11} → Examine → filter

The query range distinct from the key set with a **shorter prefix**

Enhancement?

RANGE FILTER – PREFIX BLOOM FILTER

Idea: use hierarchical prefix Bloom filters to encode different length of prefixes

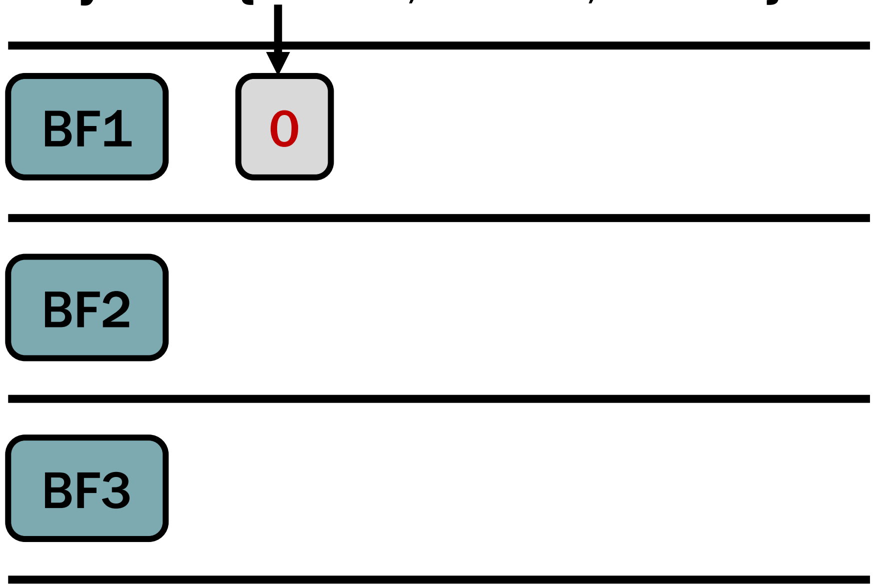
Example: set prefix length from 1 to 3.

RANGE FILTER – PREFIX BLOOM FILTER

Idea: use hierarchical prefix Bloom filters to encode different length of prefixes

Example: set prefix length from 1 to 3.

Key set: {0001, 0100, 0110}



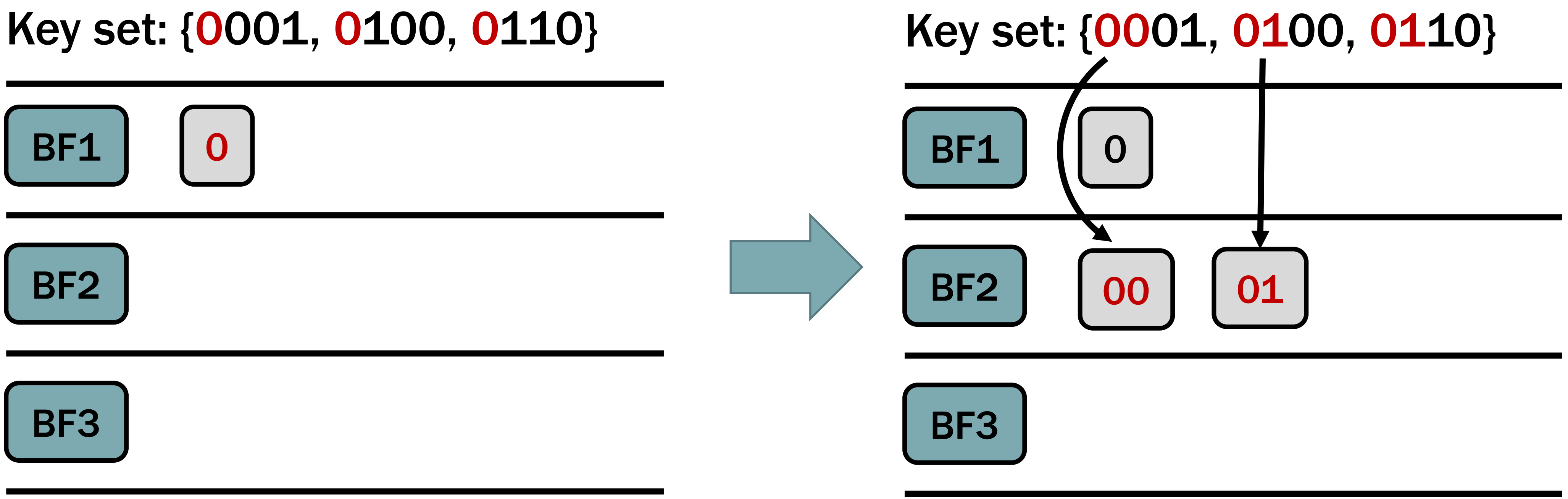
For BF1:

0 -> [0000,0111] Range size = 8

For clarity, the elements inserted into a BF are listed instead of the actual BF

RANGE FILTER – PREFIX BLOOM FILTER

Example: set prefix length from 1 to 3.



Each element represents smaller range compared to BF1

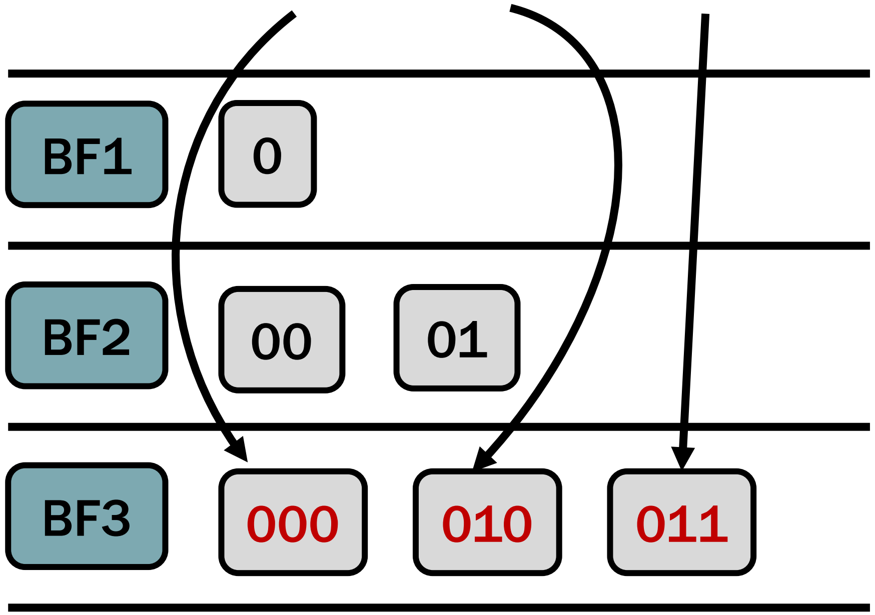
00 -> [0000,0011] Range size = 4

01 -> [0100,0111]

RANGE FILTER – PREFIX BLOOM FILTER

Example: set prefix length from 1 to 3.

Key set: {0001, 0100, 0110}



000 -> [0000,0001]

010 -> [0100,0101]

011 -> [0110,0111]

Range size = 2

RANGE FILTER – PREFIX BLOOM FILTER

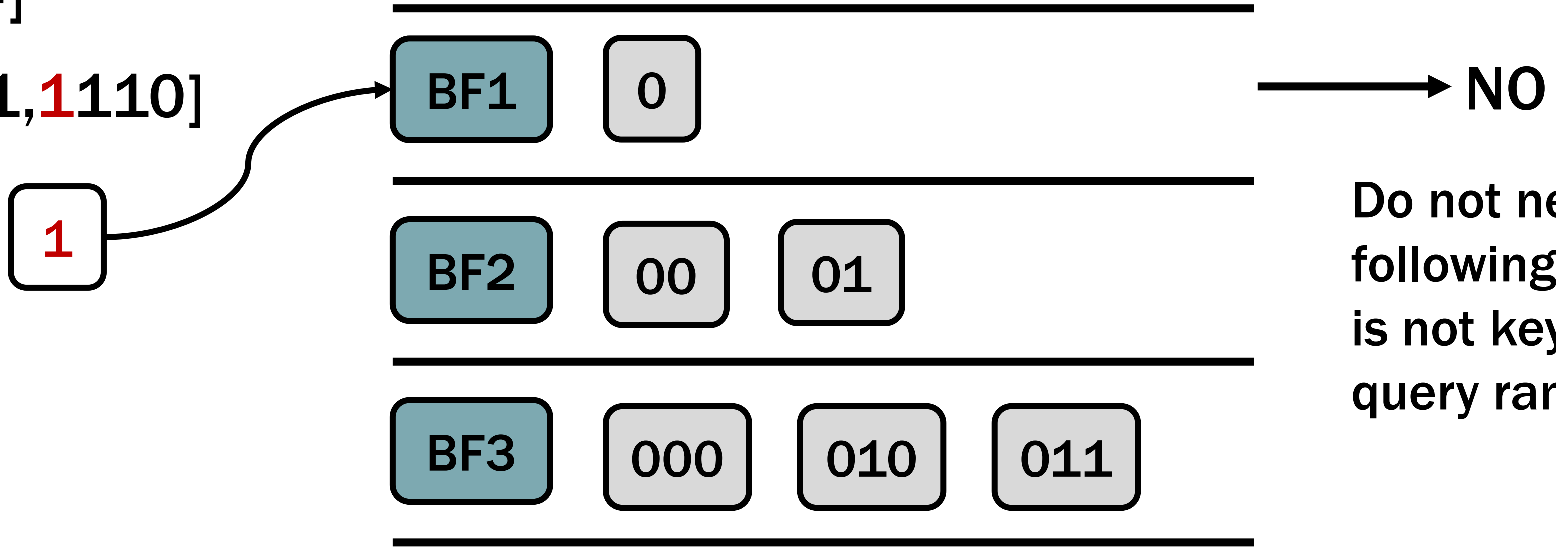
Example: set prefix length from 1 to 3.

Query

-> [9,14]

-> [1001,1110]

Key set: {0001, 0100, 0110}



Do not need to query the following BFs due to there is not key in the extended query range [1000,1111]

RANGE FILTER – PREFIX BLOOM FILTER

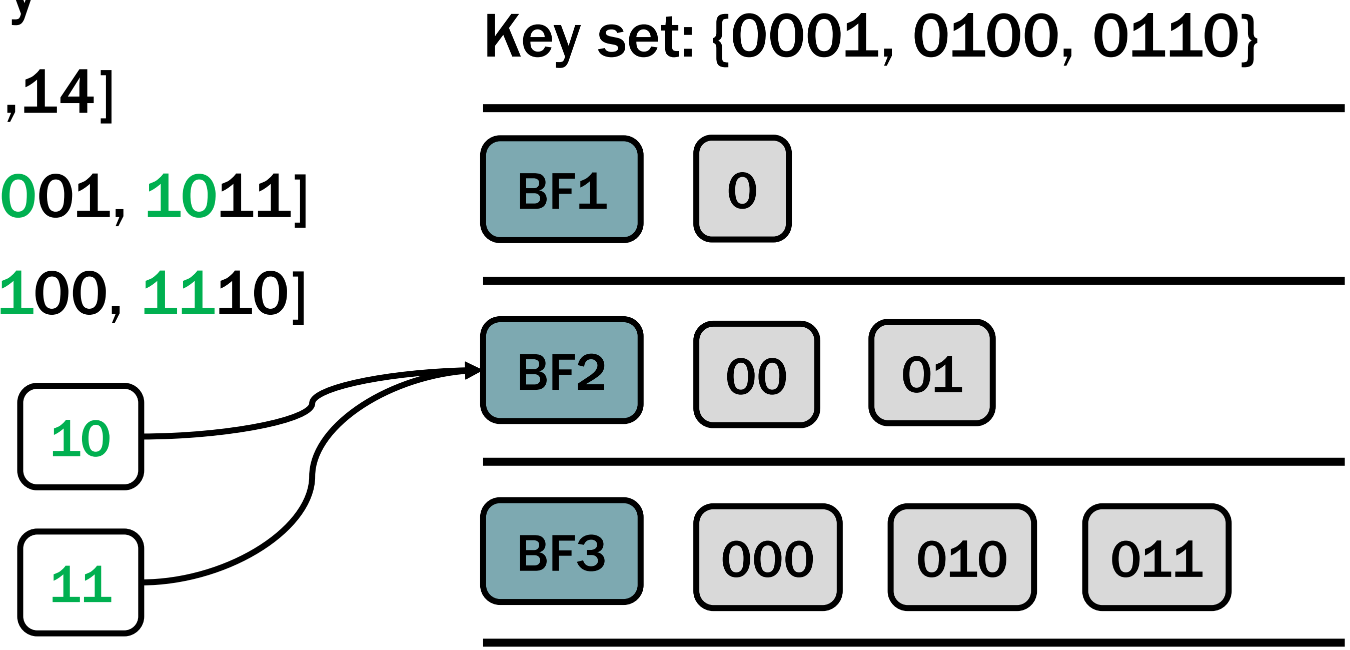
Example: set prefix length from 1 to 3.

Query

-> [9,14]

-> [1001, 1011]

[1100, 1110]

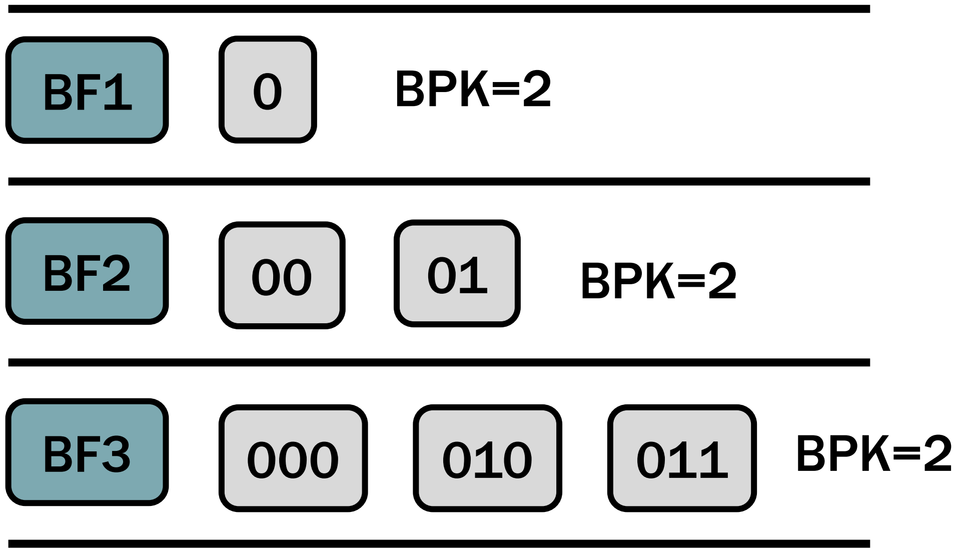


This step is unnecessary for the reason mentioned

Query 1 bit is sufficient, more efficient.

RANGE FILTER – PREFIX BLOOM FILTER

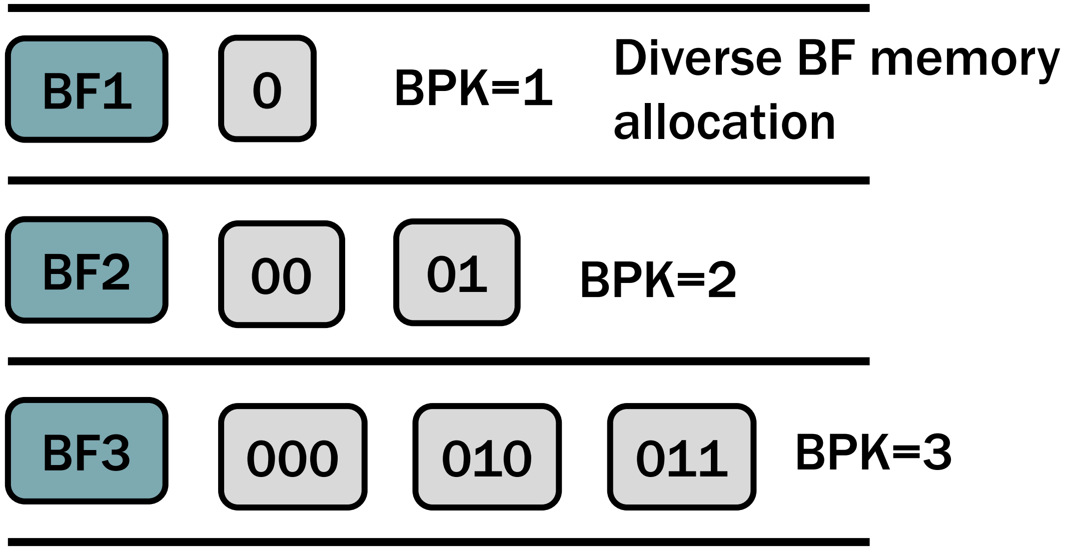
Key set: {0001, 0100, 0110}



BF1 0 BPK=2

BF2 00 01 BPK=2

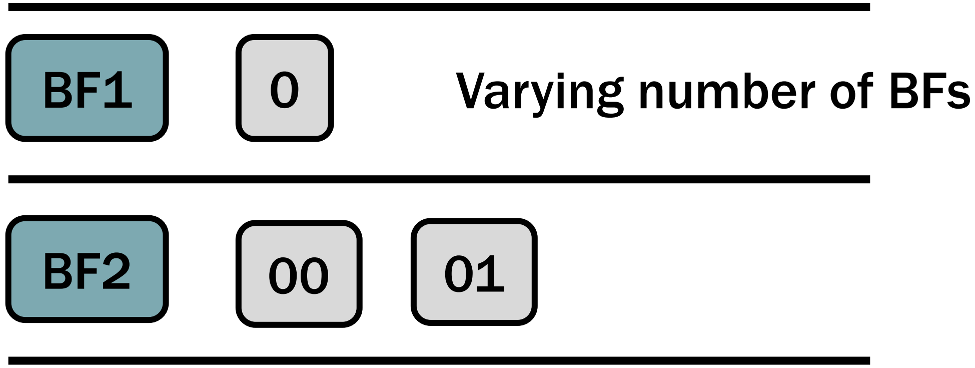
BF3 000 010 011 BPK=2



BF1 0 BPK=1 Diverse BF memory allocation

BF2 00 01 BPK=2

BF3 000 010 011 BPK=3



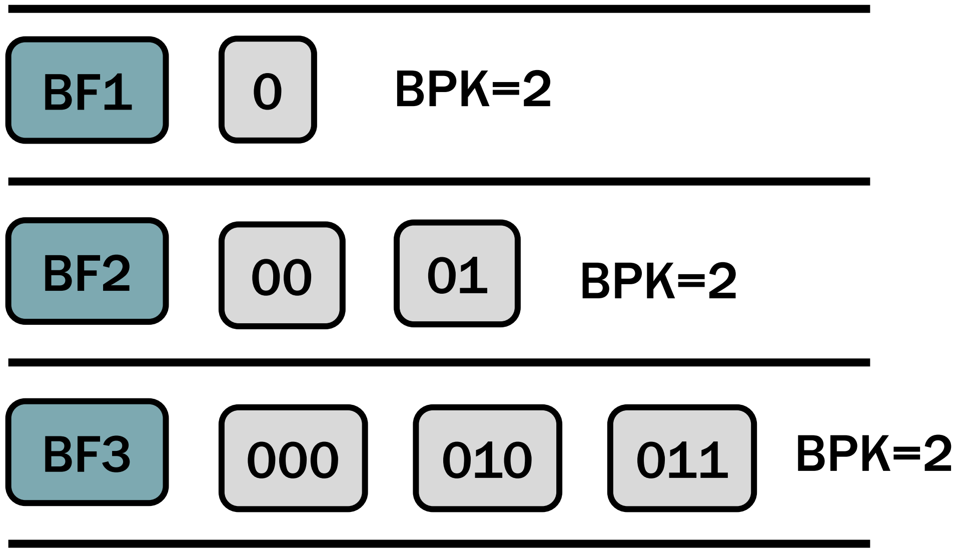
BF1 0 Varying number of BFs

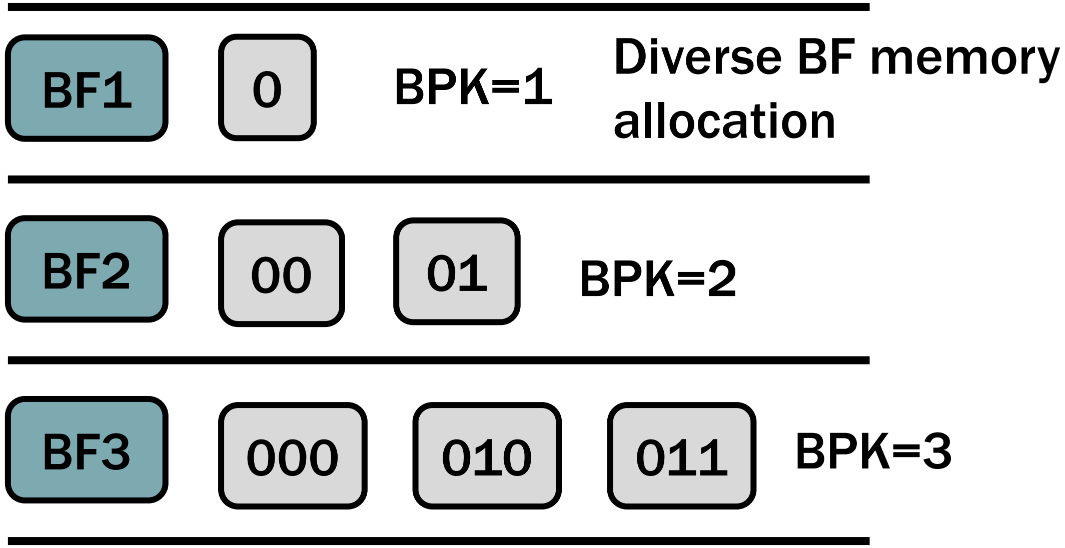
BF2 00 01

More **flexible structure** to adapt to different work condition.

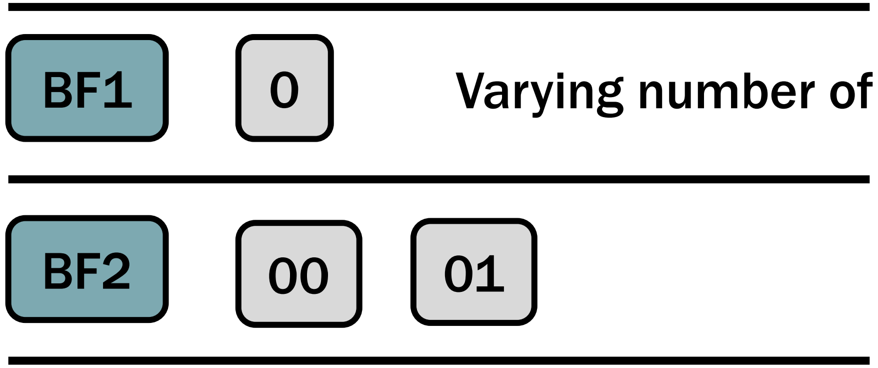
RANGE FILTER – PREFIX BLOOM FILTER

Key set: {0001, 0100, 0110}





Diverse BF memory allocation



Varying number of BFs

However, more BF could take up more memory space

The End
Thank you!