CARNEGIE MELLON UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE
15-445/645 – DATABASE SYSTEMS (FALL 2019)
PROF. ANDY PAVLO

Homework 2 (by Amadou Ngom)

Due: **Monday Sept 30, 2019 @ 11:59pm**

IMPORTANT:

- **Upload this PDF** with your answers to **Gradescope by 11:59pm on Monday Sept 30, 2019**.
- **Plagiarism**: Homework may be discussed with other students, but all homework is to be completed **individually**.

For your information:

- Graded out of **100** points; **4** questions total
- Rough time estimate: ≈1-4 hours (0.5-1 hours for each question)

Revision : 2019/09/25 15:06

Question	Points	Score
Cuckoo Hashing	20	
B+Tree	45	
Extendible Hashing	25	
Suffix Trees	10	
Total:	100	

Number of Days this Assignment is Late:

Number of Late Day You Have Left:

Question 1: Cuckoo Hashing .... [20 points]

Consider the following cuckoo hashing schema:

1. Both tables have a size of 4.
2. The hashing function of the first table returns the lowest two bits: $h_1(\texttt{x}) = \texttt{x \& 0b11}$.
3. The hashing function of the second table returns the next two bits: $h_2(\texttt{x}) = \texttt{(x >> 2) \& 0b11}$
4. When replacement is necessary, first select an element in the second table.
5. The original content is shown in Figure 1.

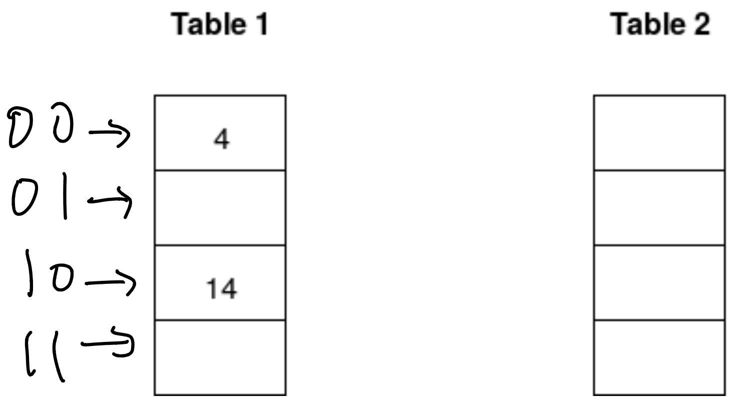


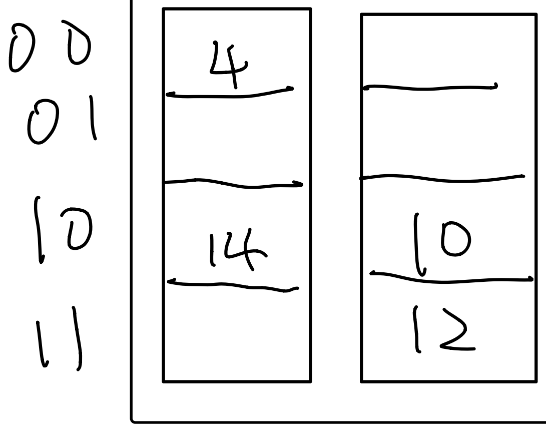
Figure 1: Initial contents of the hash tables.

Use the following template to answer the questions: `https://cmudb.io/fall2019-hw1`.

(a) **[4 points]** Insert keys 12 and 10. Draw the resulting two tables.

12 = 8+4 = 1100 $h_1(12) = 00$ $h_2(12) = 11$

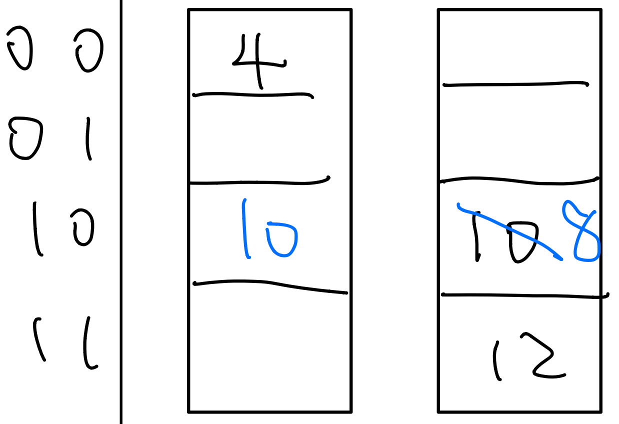
10 = 8+2 = 1010 $h_1(10) = 10$ $h_2(10) = 10$



(b) **[4 points]** Then delete 14, and insert 8. Draw the resulting two tables.

$8 = 1000$ $h_1(8) = 00$

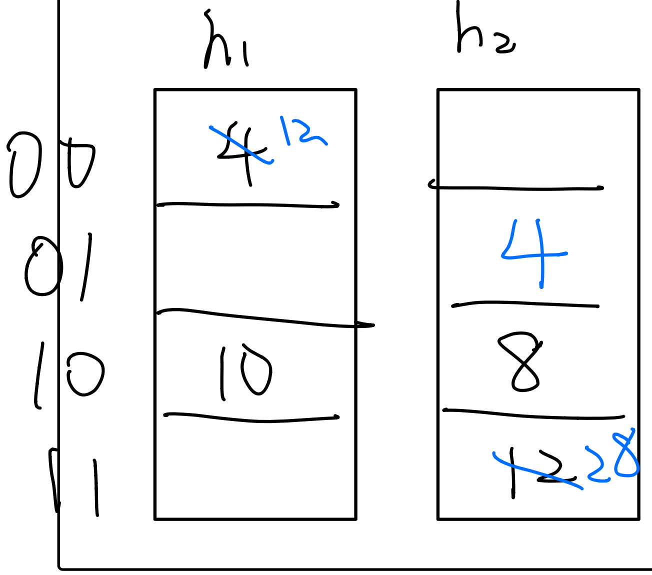
$h_2(8) = 10$



(c) **[6 points]** Finally, insert 28. Draw the resulting two tables.

$28 = 16+8+4 = 11100$ $h_1(28) = 00$

$h_2(28) = 11$



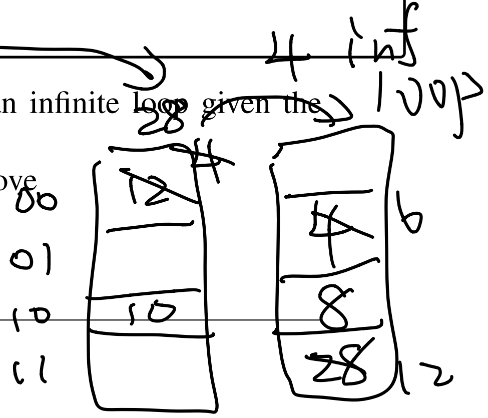
$4 = 0100$ $h_1(4) = 00$

$h_2(4) = 01$

(d) **[6 points]** What is the smallest key that potentially causes an infinite loop given the tables in **(c)**

□ ~~0~~ □ ~~2~~ □ ~~5~~ ☑ 6 □ 7 □ 9 □ None of the above

0010 0011 0110



Question 2: B+Tree ....[45 points]

Consider the following B+tree.

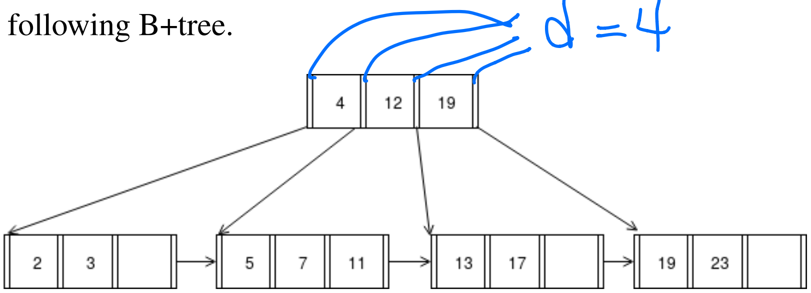


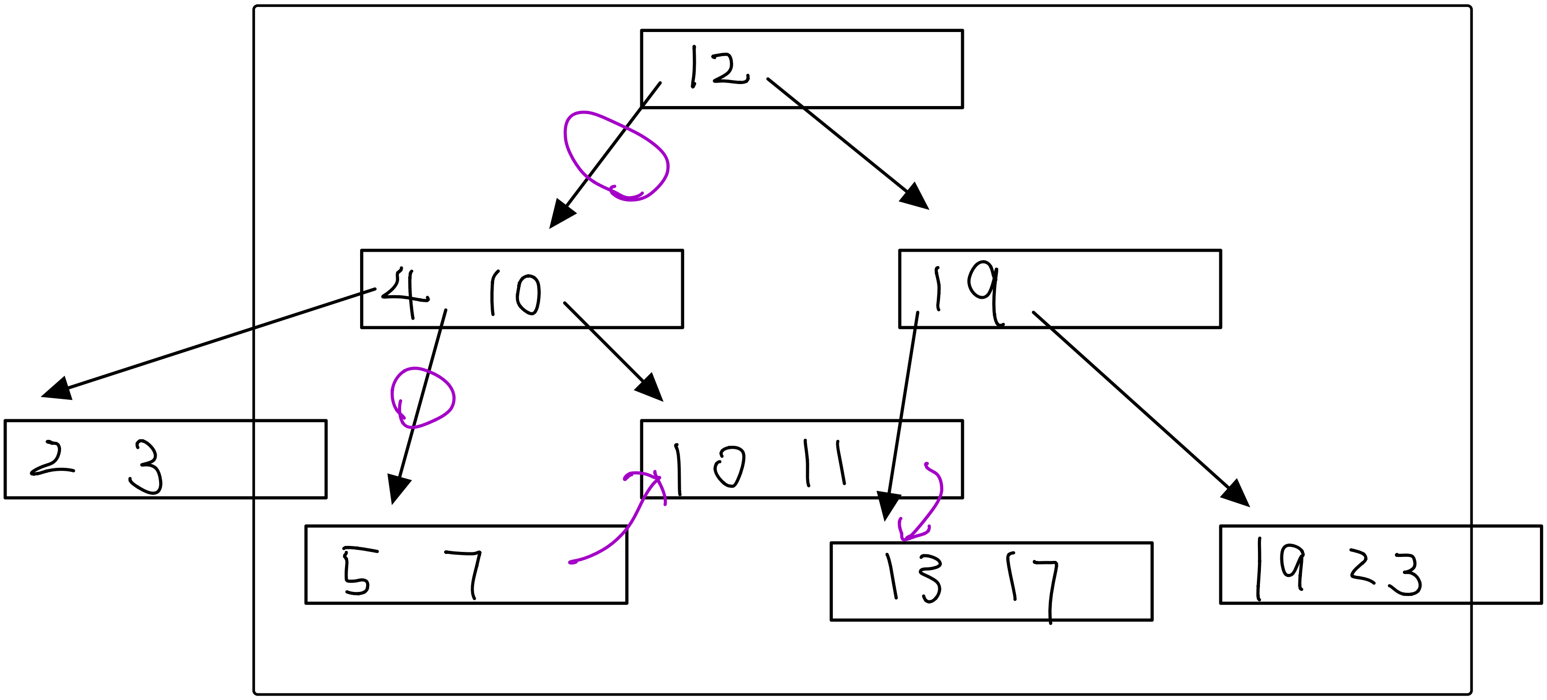
Figure 2: B+ Tree of order $d = 4$ and height $h = 2$.

When answering the following questions, be sure to follow the procedures described in class and in your textbook. You can make the following assumptions:

- A left pointer in an internal node guides towards keys $<$ than its corresponding key, while a right pointer guides towards keys $\geq$.
- A leaf node underflows when the number of **keys** goes bellow $\lceil \frac{d-1}{2} \rceil$.
- An internal node underflows when the number of **pointers** goes below $\lceil \frac{d}{2} \rceil$.

Use the following `draw.io` template for your answers:
`https://cmudb.io/fall2019-hw2`

(a) **[15 points]** Insert 10* into the B+tree. Draw the resulting tree.

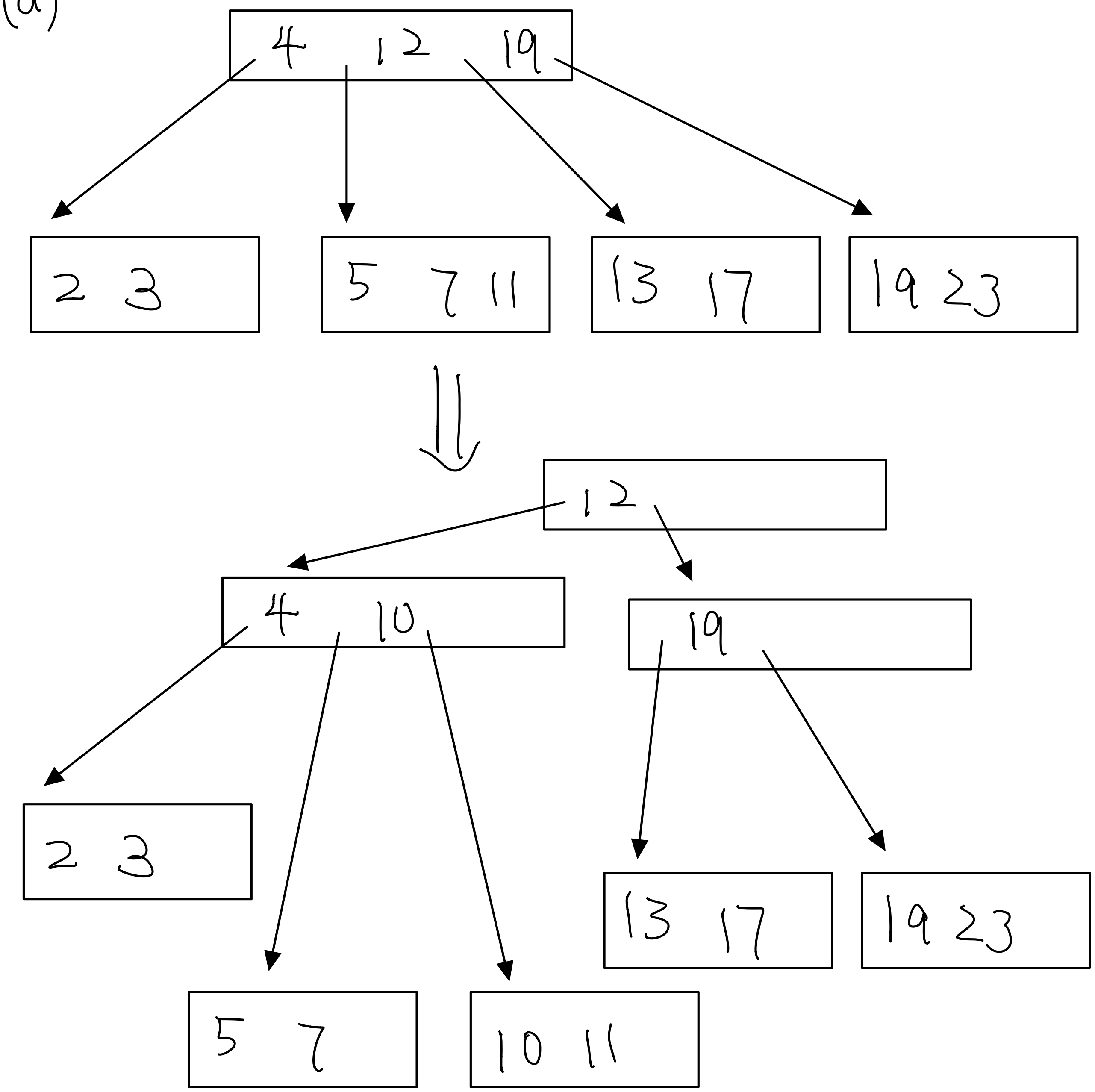


(b) **[5 points]** How many pointers (parent-to-child and sibling-to-sibling) do you chase to find all keys between 5 and 15?

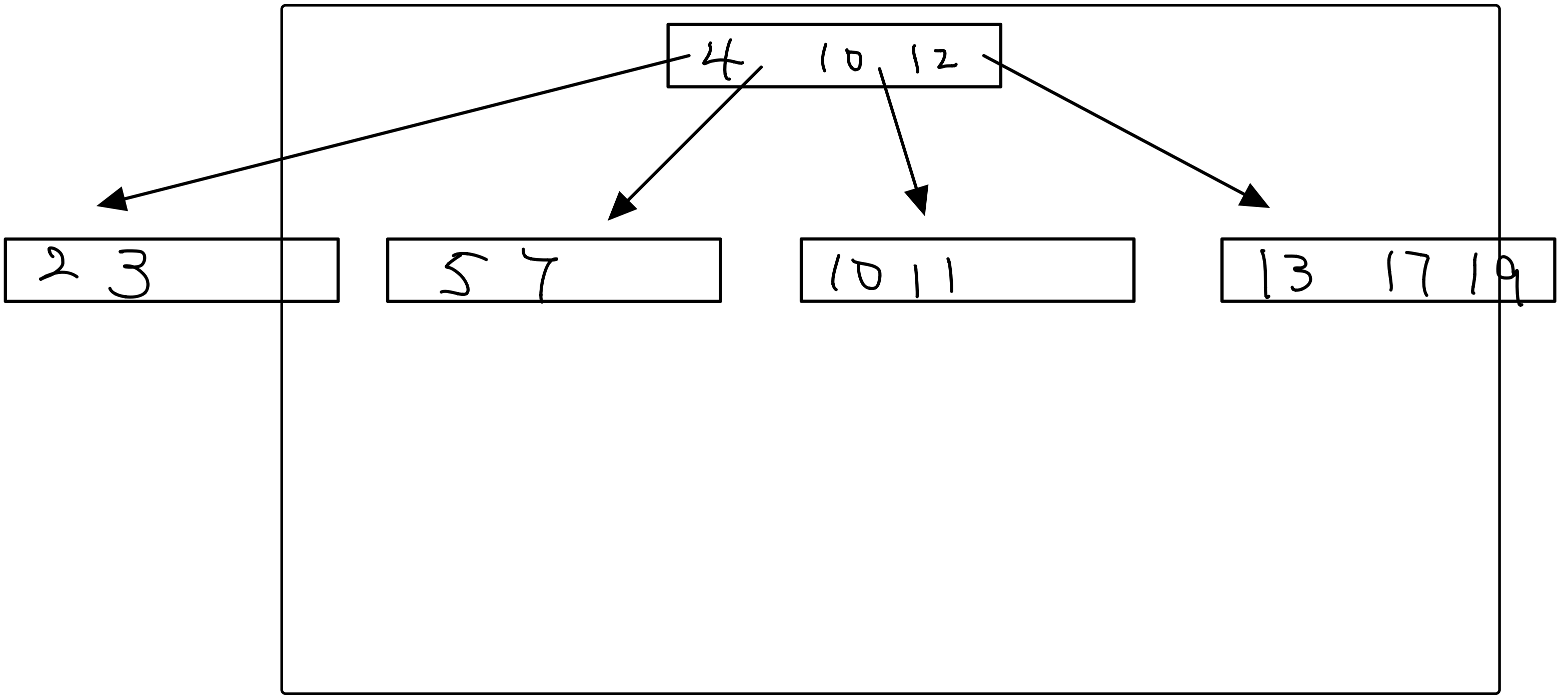
□ 2 □ 3 ☑ 4 □ 5 □ 6 □ 7

Find 5, then keep going right

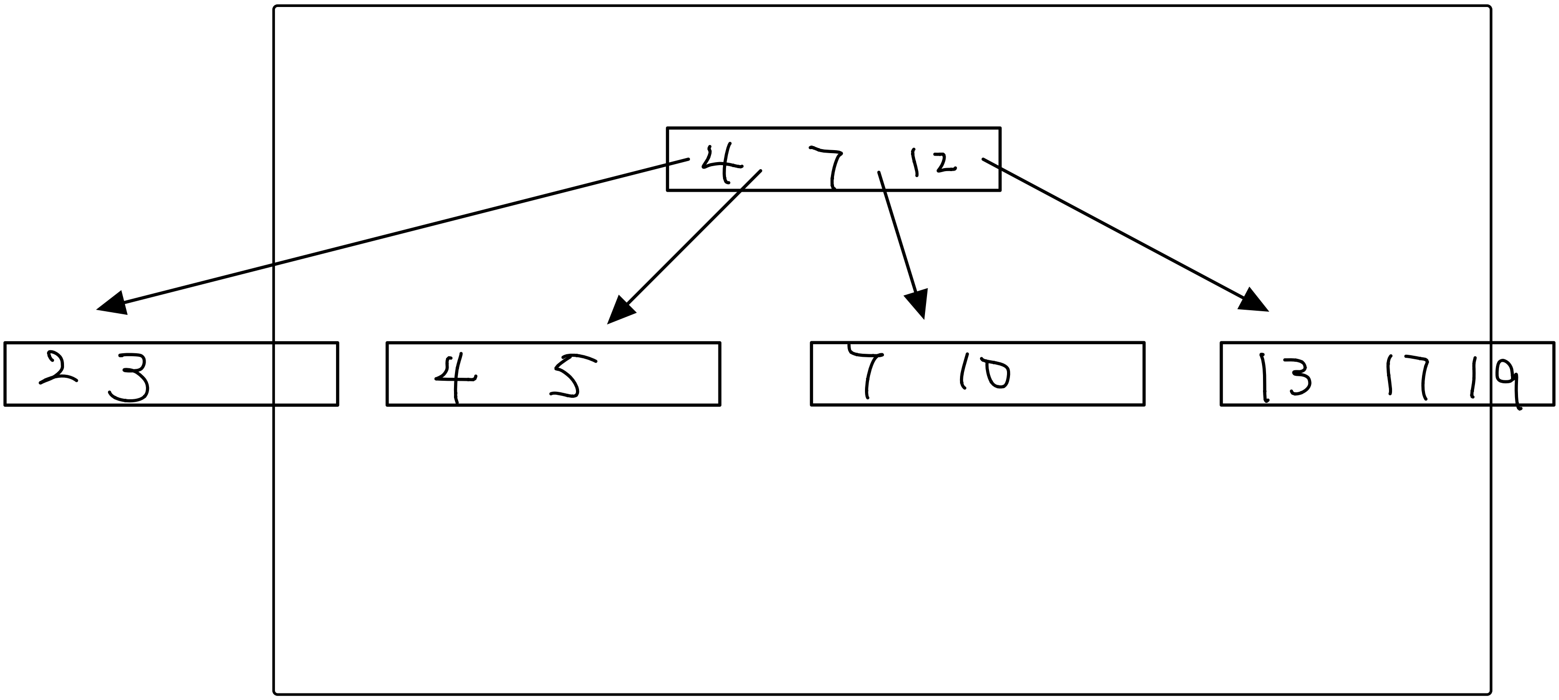
(a)



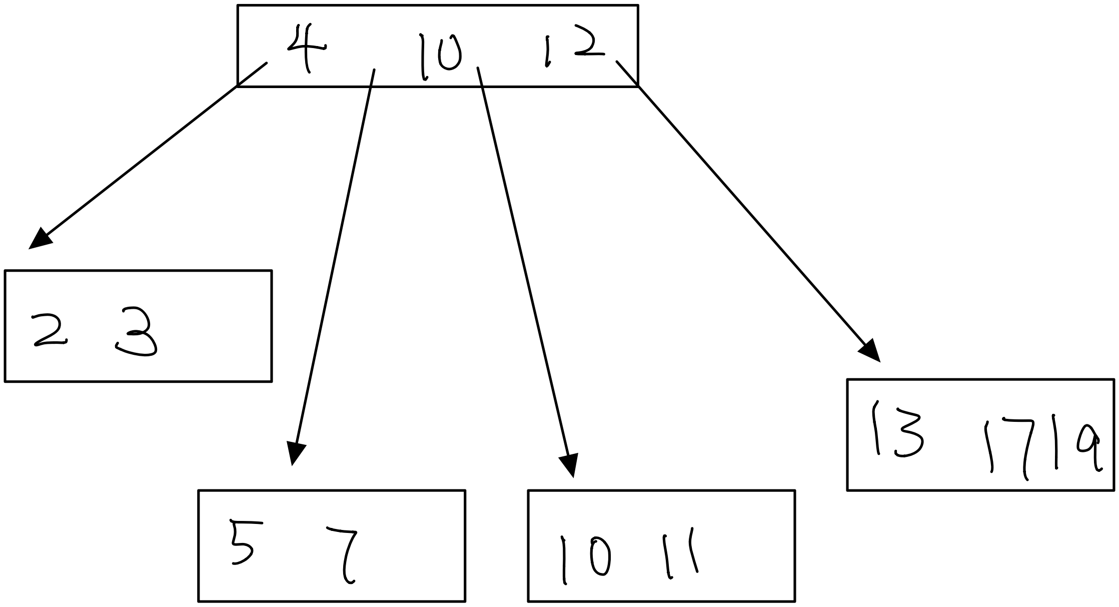
(c) **[15 points]** Then delete 23^*. Draw the resulting tree.



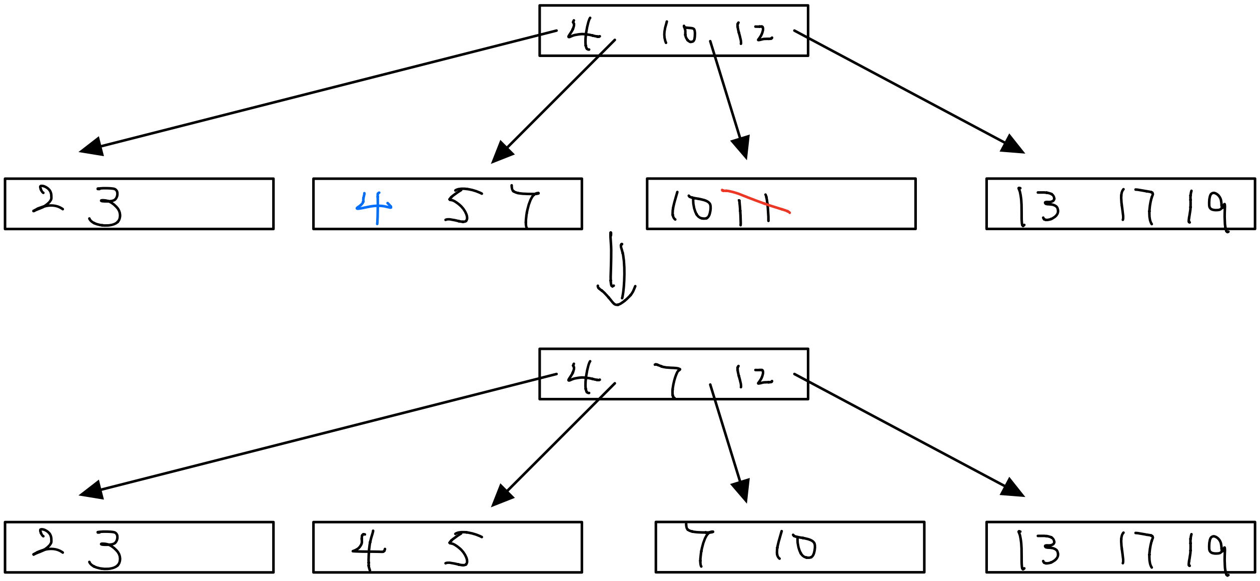
(d) **[10 points]** Finally insert 4^* and delete 11^*. Draw the resulting tree.



(b)


4 10 12
2 3
5 7
10 11
13 17 19

(c)


4 10 12
2 3
4 5 7
10 ~~11~~
13 17 19
⇓
4 7 12
2 3
4 5
7 10
13 17 19

Question 3: Extendible Hashing ....[25 points]

Consider an extendible hashing structure such that:

- Each bucket can hold up to two records.
- The hashing function uses the lowest g bits, where g is the global depth.

(a) Starting from an empty table, insert keys 15, 3, 7, 14.

i. **[3 points]** What is the global depth of the resulting table?
□ 0 □ 1 □ 2 ☑ 3 □ 4 □ None of the above

ii. **[3 points]** What is the local depth the bucket containing 14?
□ 0 ☑ 1 □ 2 □ 3 □ 4 □ None of the above

iii. **[3 points]** What is the local depth of the bucket containing 3?
□ 0 □ 1 □ 2 ☑ 3 □ 4 □ None of the above

(b) Starting from the result in **(a)**, you insert keys 1, 9, 23, 11, 17.

i. **[4 points]** Which key will first cause a split (without doubling the size of the table)?
□ 1 □ 9 □ 23 □ 11 ☑ 17 □ None of the above

ii. **[4 points]** Which key will first make the table double in size?
□ 1 □ 9 ☑ 23 □ 11 □ 17 □ None of the above

(c) Now consider the table below, along with the following deletion rules:

1. If two buckets have the same local depth d, and share the first $d-1$ bits of their indexes (e.g. `010` and `110` share the first 2 bits), then they can be merged if the total capacity fits in a single bucket. The resulting local depth is $d-1$.
2. If the global depth g becomes strictly greater than all local depths, then the table can be halved in size. The resulting global depth is $g-1$.

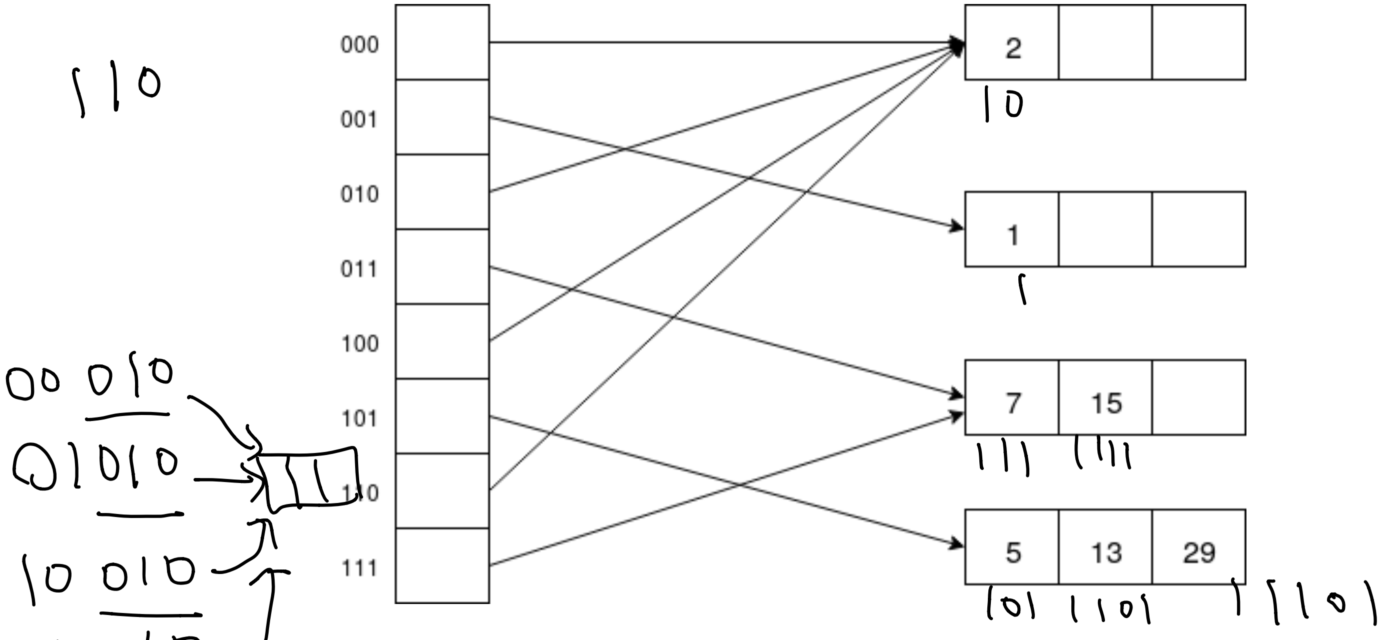


Figure 3: Extendible Hash Table along with the indexes of each bucket

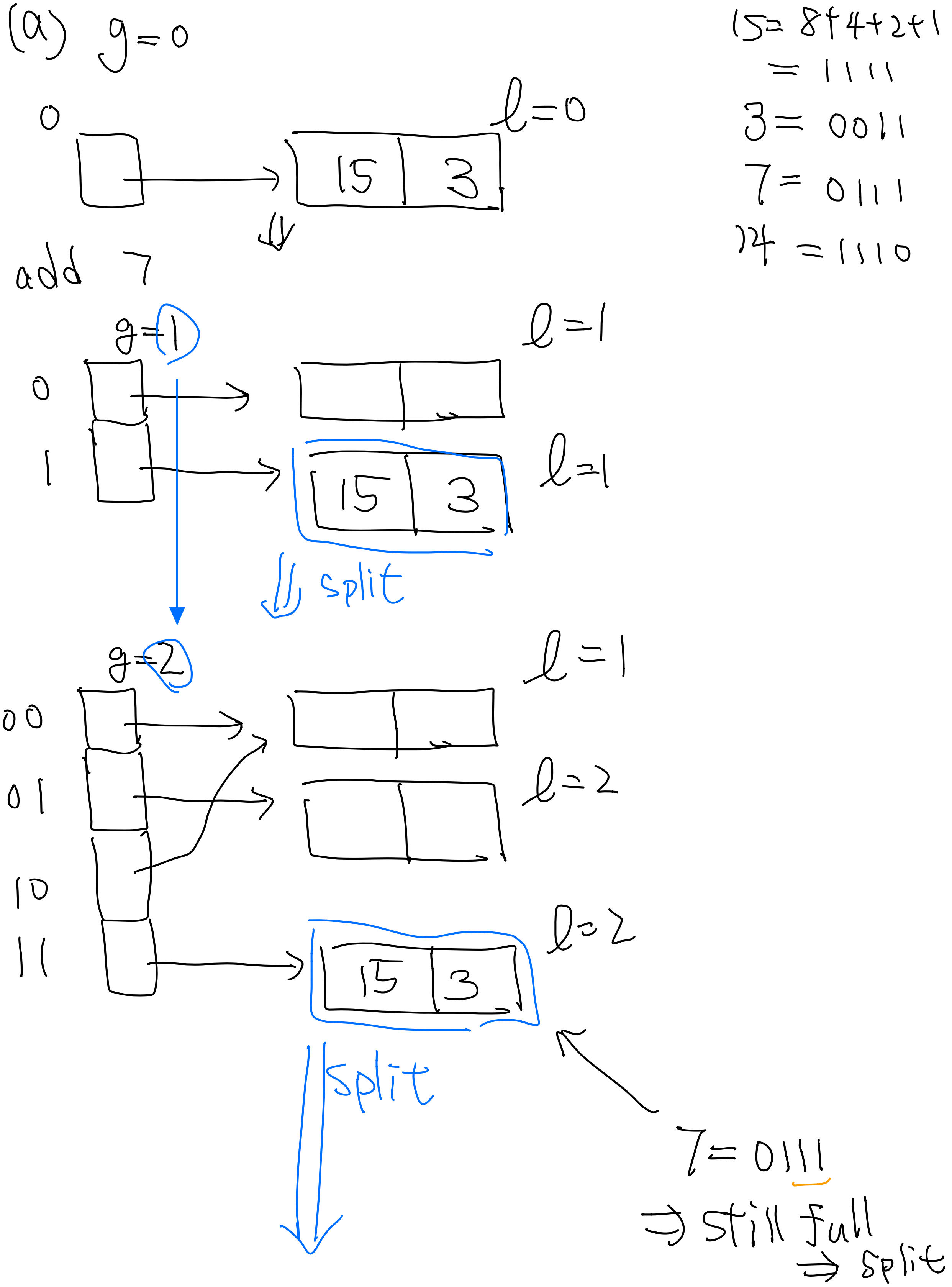
3

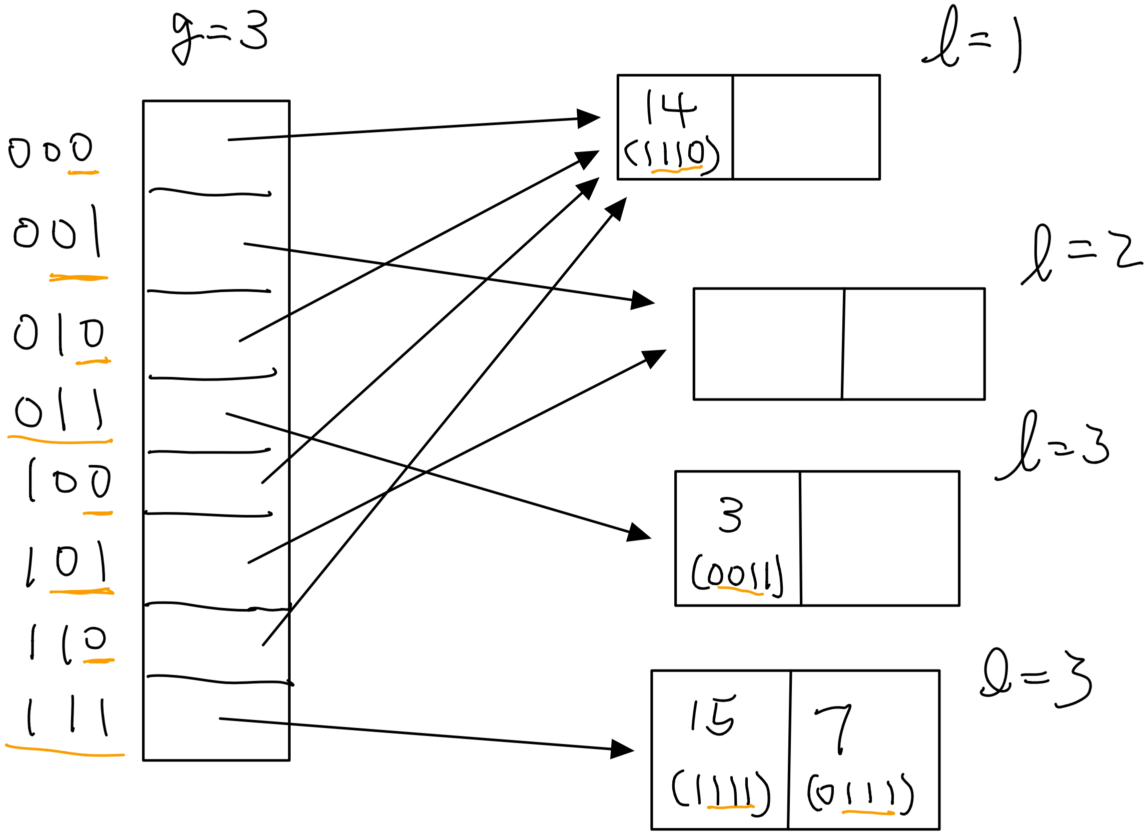
Starting from the table above, delete keys 2, 7, 13, 15, 29.

i. **[4 points]** Which deletion first causes a reduction in a local depth.
□ 2 □ 7 ☑ 13 □ 15 □ 29 □ None of the above

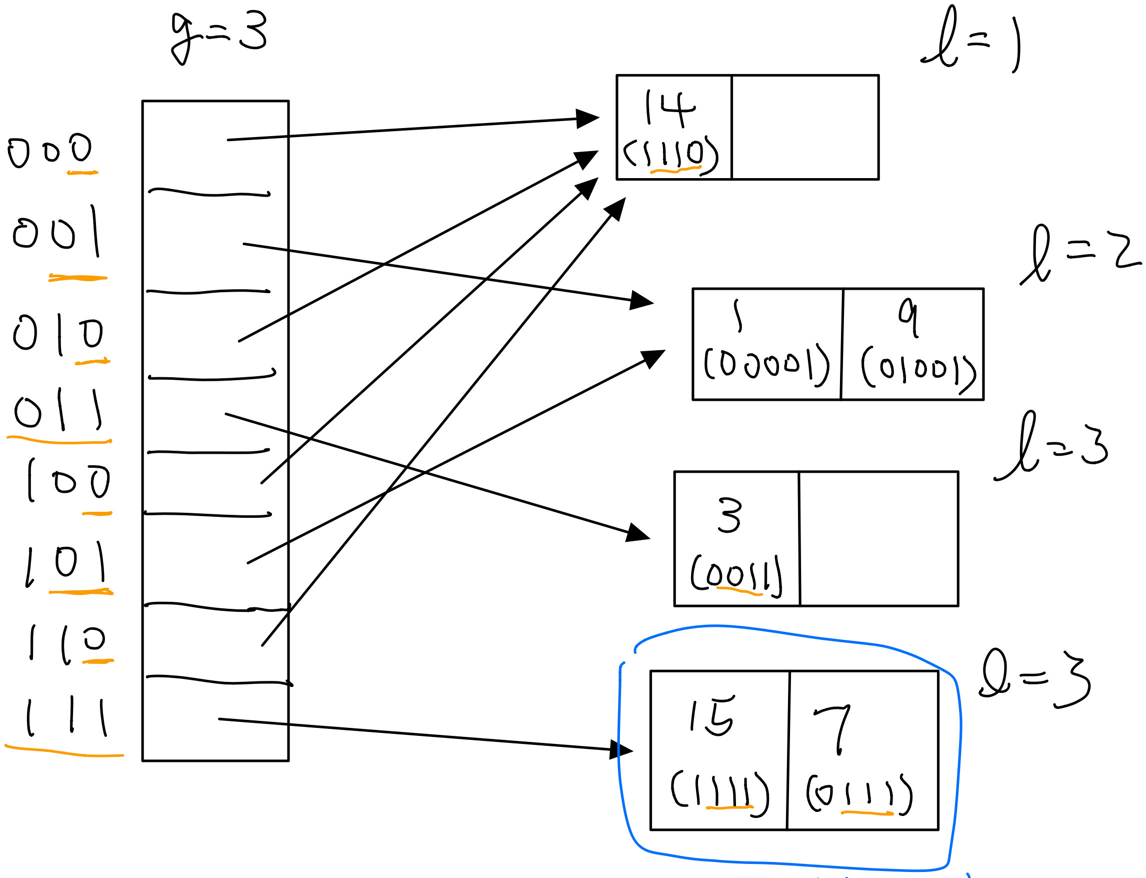
ii. **[4 points]** Which deletion first causes a reduction in global depth.
□ 2 □ 7 ☑ 13 □ 15 □ 29 □ None of the above

(a)


g=0
0
15
3
l=0
add 7
g=1
0
1
l=1
15
3
l=1
split
g=2
00
01
10
11
l=1
l=2
15
3
l=2
split
7=0111
⇒ still full
⇒ split
15=8+4+2+1
=1111
3=0011
7=0111
14=1110


g=3
000
001
010
011
100
101
110
111
l=1
14
(1110)
l=2
l=3
3
(0011)
l=3
15
(1111)
7
(0111)

(b)



Split when
23 is inserted

1, 9, 23, 11, 17

1= 00001

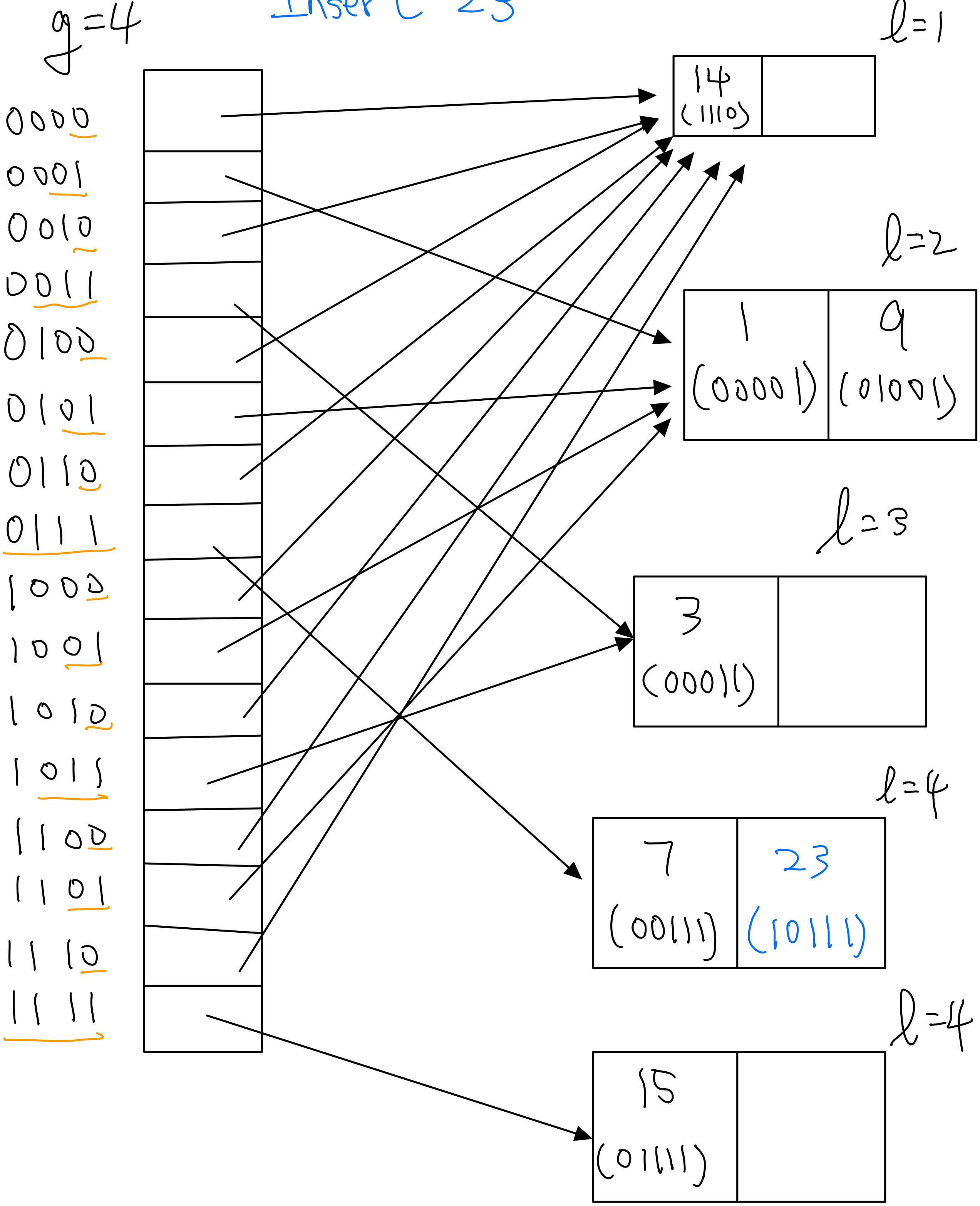
9= 01001

23=16+4+2+1 = 10111

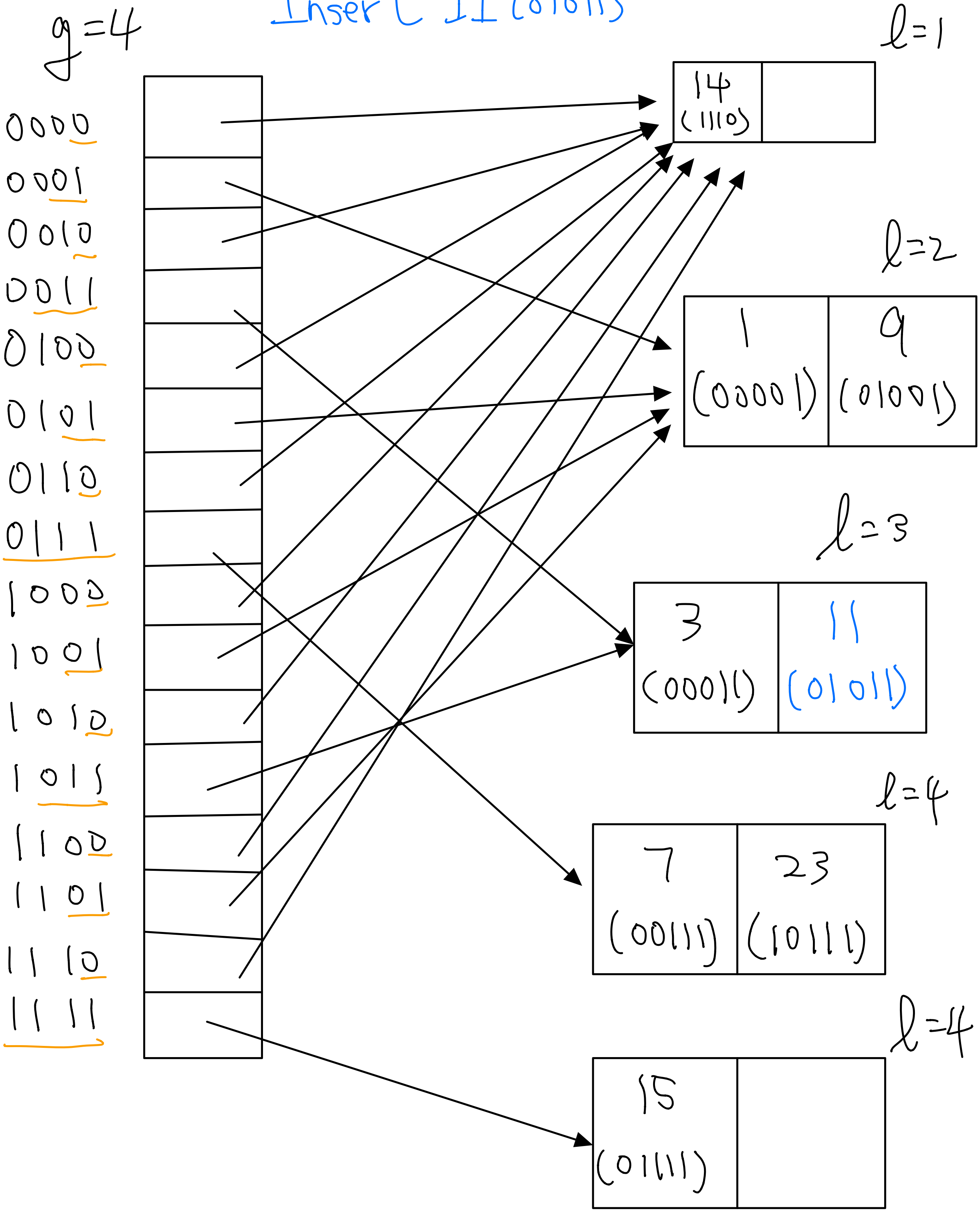
11= 8+2+1 = 01011

17= 16+1 = 10001

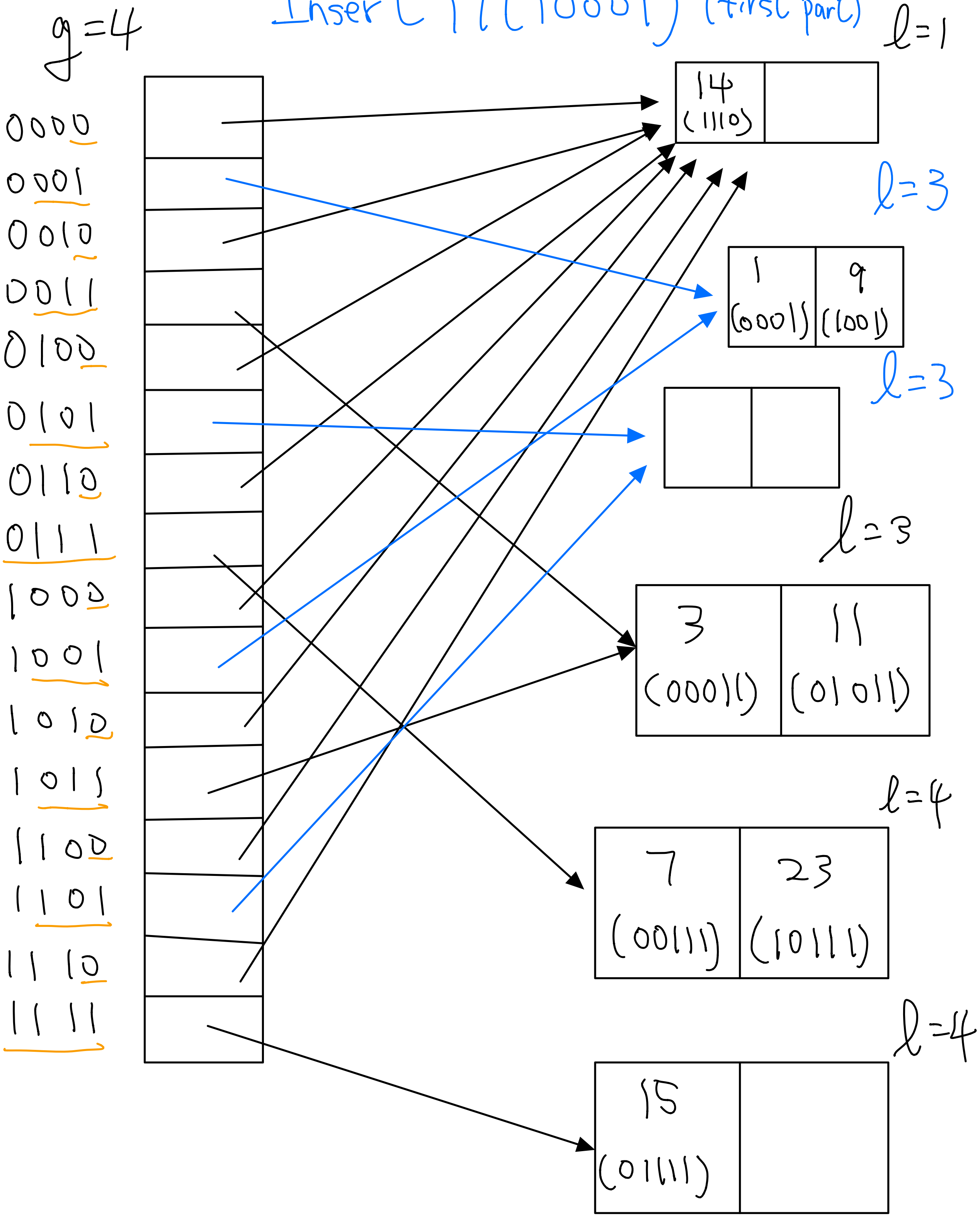
Insert 23


g=4
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111
l=1
14
(1110)
l=2
1
(00001)
9
(01001)
l=3
3
(00011)
l=4
7
(00111)
23
(10111)
l=4
15
(01111)

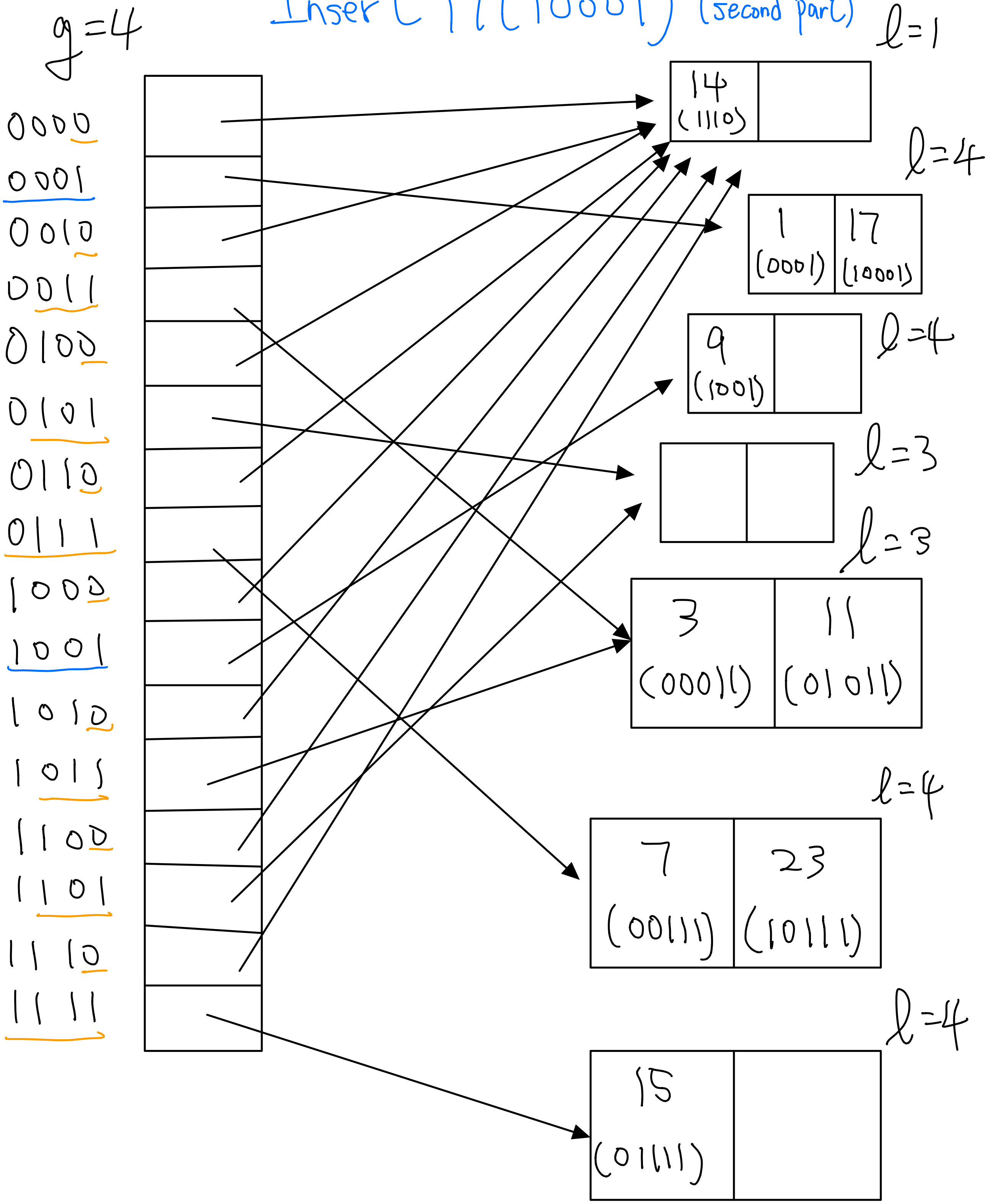
Insert 11 (01011)



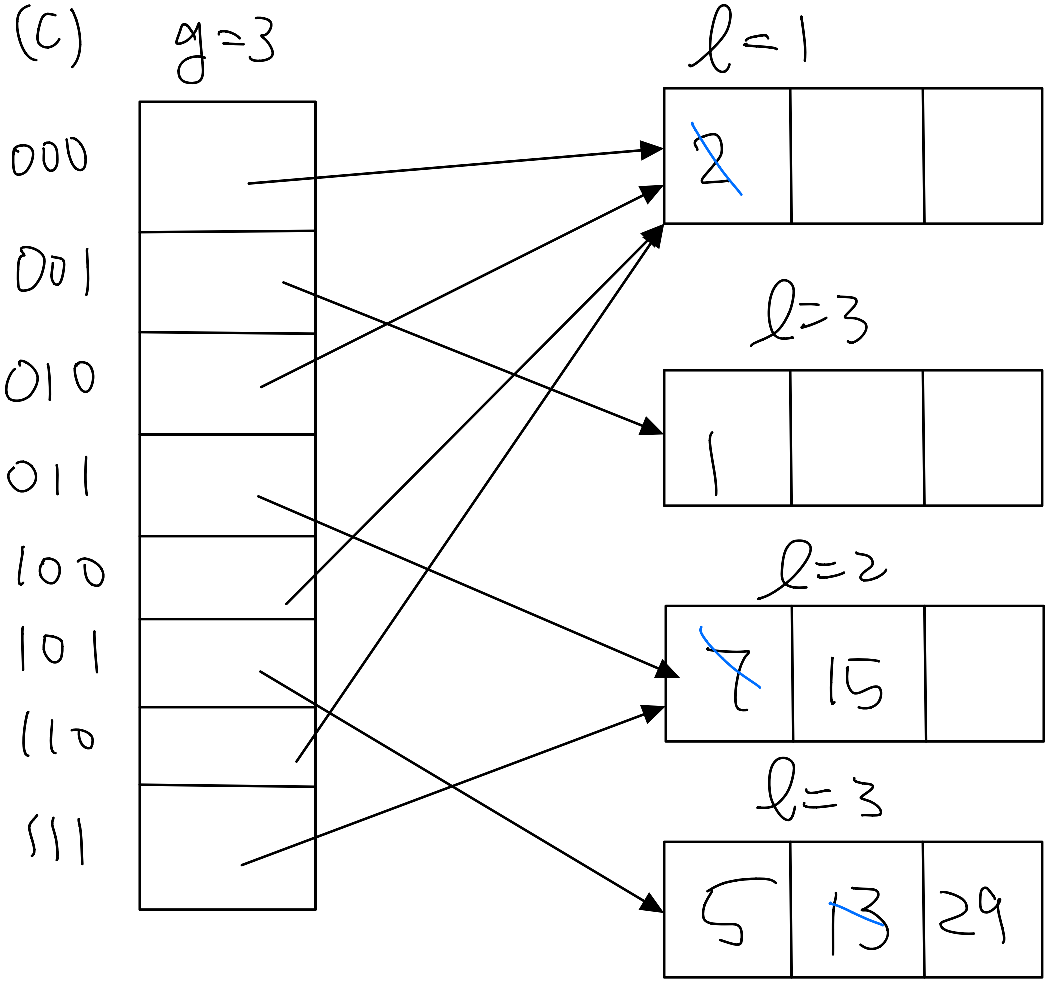
Insert 17 (10001) (first part)

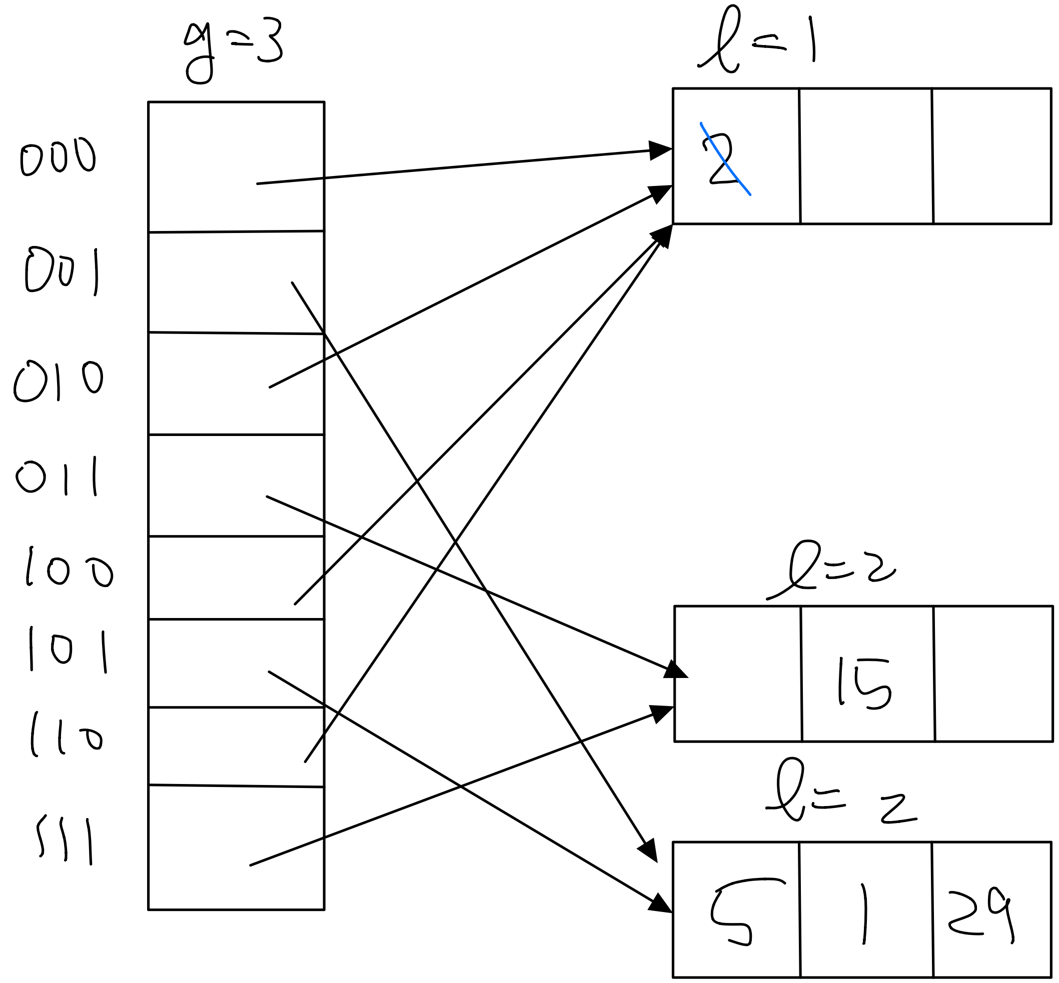


Insert 17 (10001) (second part)

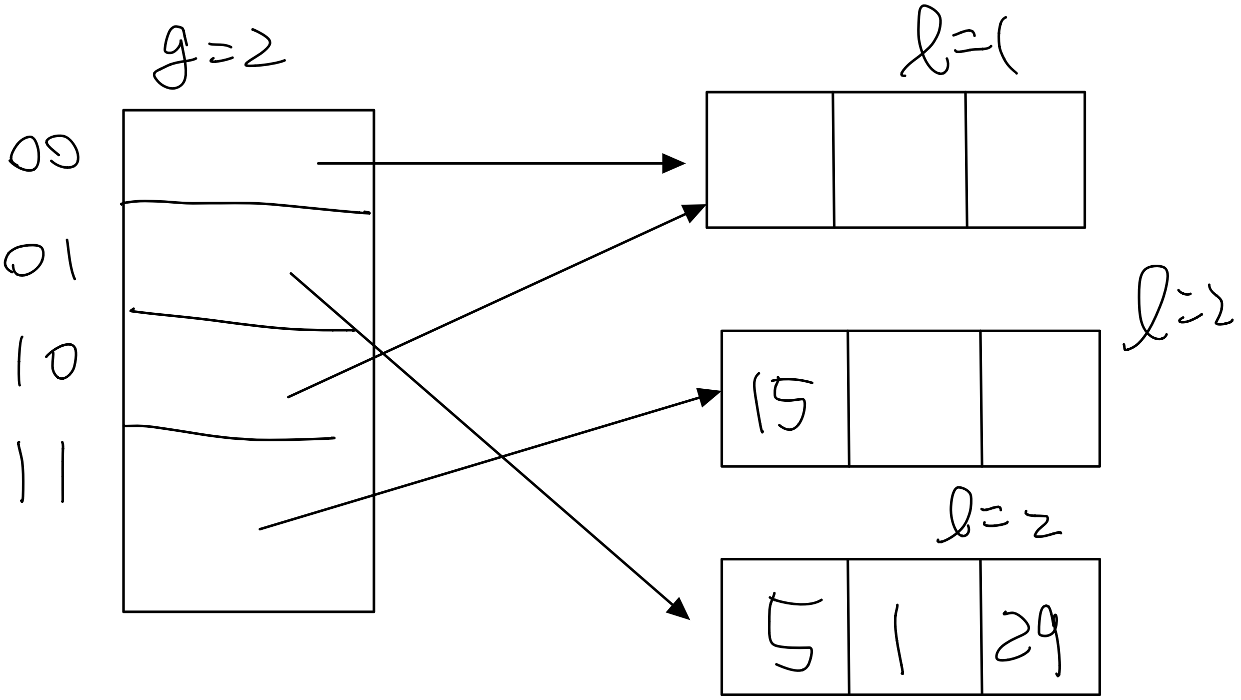


(c)





⇒ halve the size


g=2
00
01
10
11
l=1
l=2
15
l=2
5
1
29

Question 4: Suffix Trees .... [10 points]

Consider the following suffix tree for **unsigned 32-bit integers**.

one hex
= 4 bits

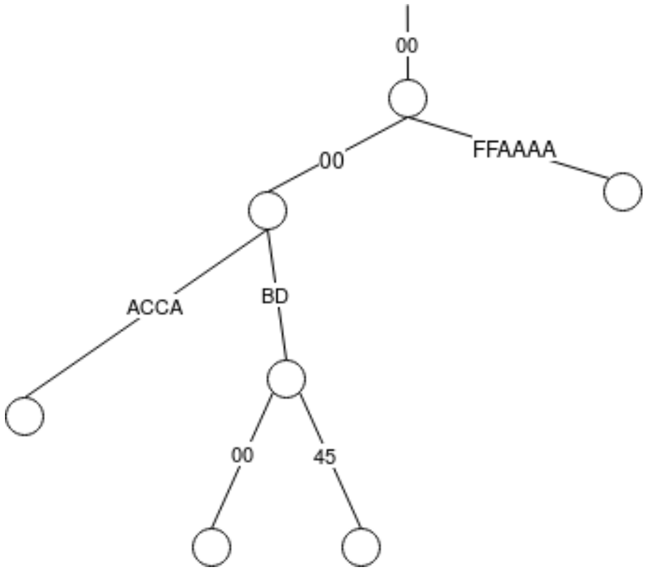
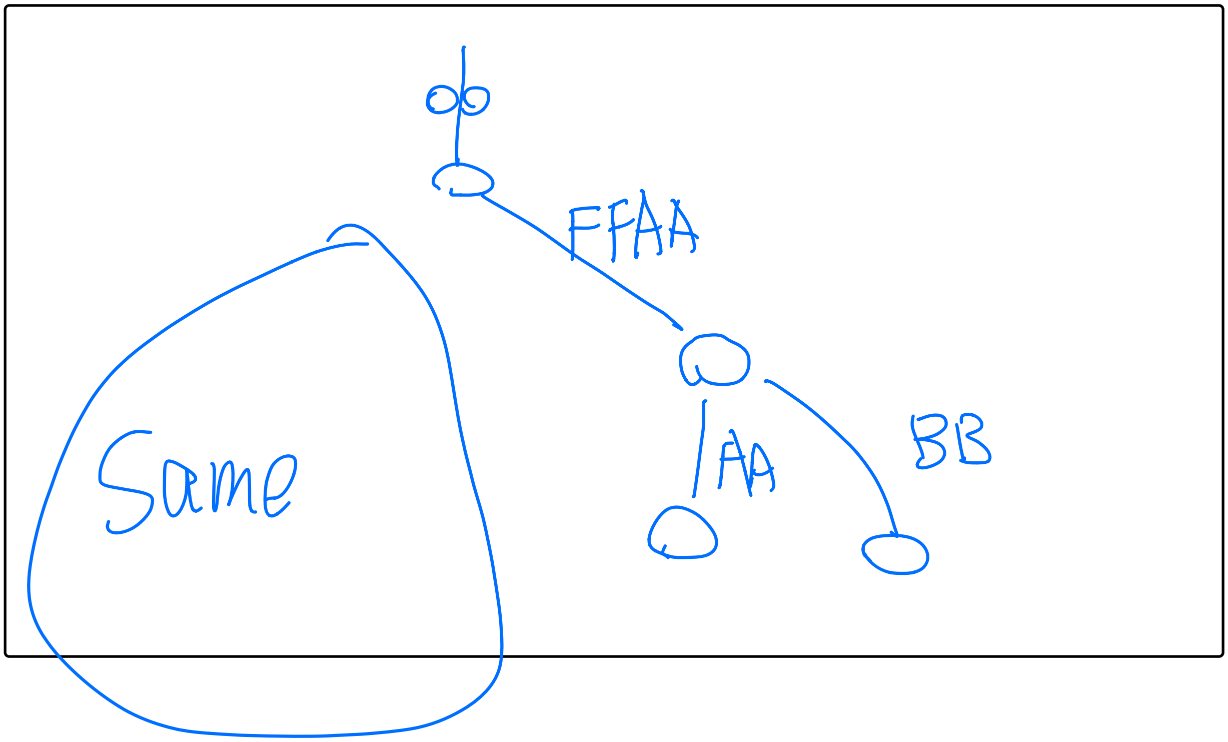


Figure 4: Suffix Tree

(a) **[3 points]** Which of the following elements belong to the suffix tree. Select all that apply.

□ `0x45BD0000` □ `0x0000CAAC` □ `0xFFAAAA00` □ `0xACCA0000` □ `0xBD000000`
☑ None of the above

(b) **[7 points]** Insert the key `0x00FFAABB`. Draw the resulting tree using this template: `https://cmudb.io/fall2019-hw4`.



End of Homework 2