CARNEGIE MELLON UNIVERSITY
COMPUTER SCIENCE DEPARTMENT
15-445/645 – DATABASE SYSTEMS (FALL 2022)
PROF. ANDY PAVLO

Homework #2 (by Mike Xu)
Due: **Sunday September 25, 2022 @ 11:59pm**

IMPORTANT:
- Enter all of your answers into **Gradescope by 11:59pm on Sunday September 25, 2022.**
- **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 : 2022/09/22 22:11

Question	Points	Score
Storage Models	16	
Cuckoo Hashing	20	
Extendible Hashing	28	
B+Tree	36	
Total:	100	

Question 1: Storage Models .... [16 points]

Consider a database with a single table `R(q_id,txns,total,failed)`, where `q_id` is the *primary key*, and all attributes are the same fixed width. Suppose `R` has 20,000 tuples that fit into 100 pages, Ignore any additional storage overhead for the table (e.g., page headers, tuple headers). Additionally, you should make the following assumptions:

- The DBMS does *not* have any additional meta-data (e.g., sort order, zone maps).
- `R` does *not* have any indexes (including for primary key `q_id`)
- None of `R`'s pages are already in the buffer pool.

Consider the following query:

```
SELECT total - failed FROM R
   WHERE q_id = 96 AND txns > 420;
```

(a) Suppose the DBMS uses the decomposition storage model (DSM) with implicit offsets

i. **[4 points]** What is the *minimum* number of pages that the DBMS will potentially have to read from disk to answer this query?
☐ 1 ☐ 2-10 ☐ 11-50 ☐ 51-100 ☐ ≥ 101
☐ Not possible to determine

ii. **[4 points]** What is the *maximum* number of pages that the DBMS will potentially have to read from disk to answer this query?
☐ 1 ☐ 2-10 ☐ 11-50 ☐ 51-100 ☐ ≥ 101
☐ Not possible to determine

(b) Suppose the DBMS uses the N-ary storage model (NSM)

i. **[4 points]** What is the *minimum* number of pages that the DBMS will potentially have to read from disk to answer this query?
☐ 1 ☐ 2-10 ☐ 11-50 ☐ 51-100 ☐ ≥ 101
☐ Not possible to determine

ii. **[4 points]** What is the *maximum* number of pages that the DBMS will potentially have to read from disk to answer this query?
☐ 1 ☐ 2-10 ☐ 11-50 ☐ 51-100 ☐ ≥ 101
☐ Not possible to determine

Question 2: 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 fourth and third least significant bits: $h_1(\texttt{x}) = \texttt{(x >> 2) \& 0b11}$.
3. The hashing function of the second table returns the least significant two bits: $h_2(\texttt{x}) = \texttt{x \& 0b11}$.
4. When inserting, try table 1 first.
5. When replacement is necessary, first select an element in the second table.
6. The original entries in the table are shown in the figure below.

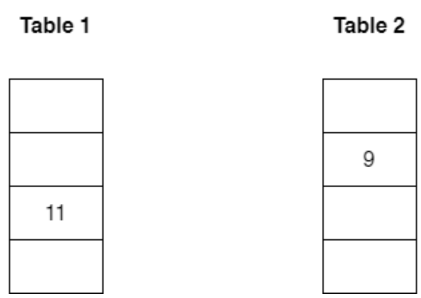
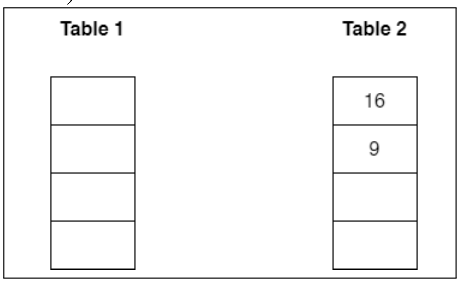


Figure 1: Initial contents of the hash tables.

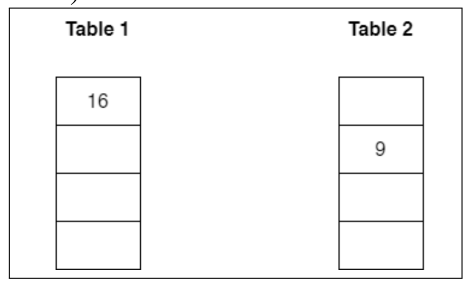
(a) **[3 points]** Select the sequence of insert operations that results in the initial state.
☐ Insert 9, insert 11 ☐ Insert 11, insert 9 ☐ None of the above

(b) **[3 points]** Insert key 16 and delete 11. Select the resulting two tables.

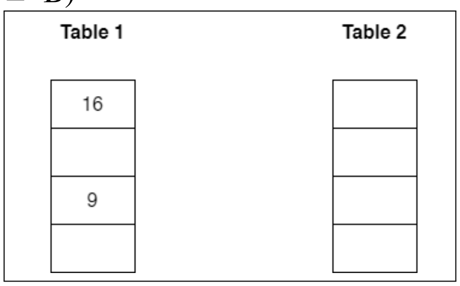
☐ A)



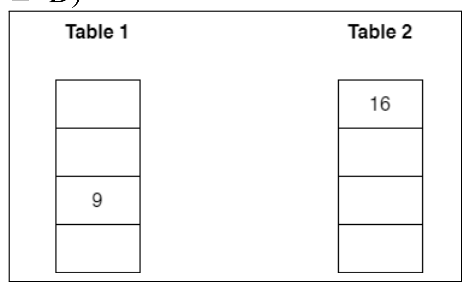
☐ C)



☐ B)

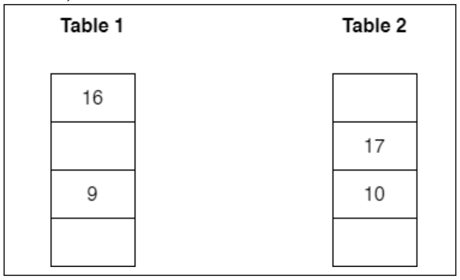


☐ D)

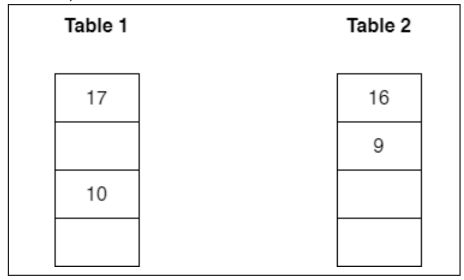


(c) **[4 points]** Then insert 17 followed by 10. Select the resulting two tables.

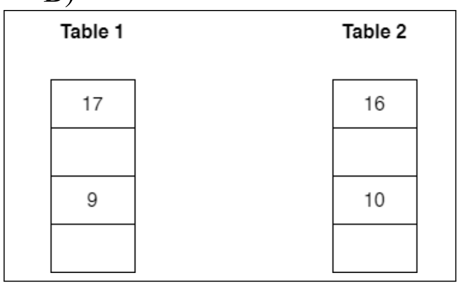
☐ A)



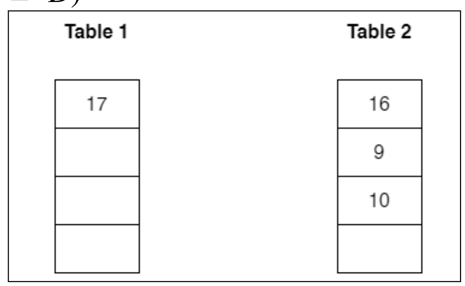
☐ C)



☐ B)



☐ D)



(d) **[5 points]** Finally, insert 33 and delete 16. Select the resulting two tables.

☐ A)

Table 1	Table 2
33	
	9
10	
	17

☐ B)

Table 1	Table 2
17	
	33
9	10

☐ C)

Table 1	Table 2
33	
	17
9	10

☐ D)

Table 1	Table 2
9	33
10	
	17

(e) **[5 points]** What is the smallest key that potentially causes an infinite loop given the tables in **(d)**?

☐ 0 ☐ 1 ☐ 2 ☐ 6 ☐ 9 ☐ 10 ☐ None of the above

Question 3: Extendible Hashing . . . . [28 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, 14, 23, 11, 9.

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

ii. **[4 points]** What is the local depth the bucket containing 15?
☐ 0 ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ None of the above

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

(b) Starting from the result in **(a)**, you insert keys 12, 5, 7, 13, 2.

i. **[4 points]** Which key will first cause a split (without doubling the size of the table)?
☐ 12 ☐ 5 ☐ 7 ☐ 13 ☐ 2 ☐ None of the above

ii. **[4 points]** Which key will first make the table double in size?
☐ 12 ☐ 5 ☐ 7 ☐ 13 ☐ 2 ☐ None of the above

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

1. If two buckets satisfy the following:
   (a) They have the same local depth d
   (b) They share the first $d-1$ bits of their indexes (e.g. `b010` and `b110` share the first 2 bits)
   (c) Their constituent elements fit in a single bucket.

   Then they can be merged into a single bucket with local depth $d-1$.
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 $g-1$.

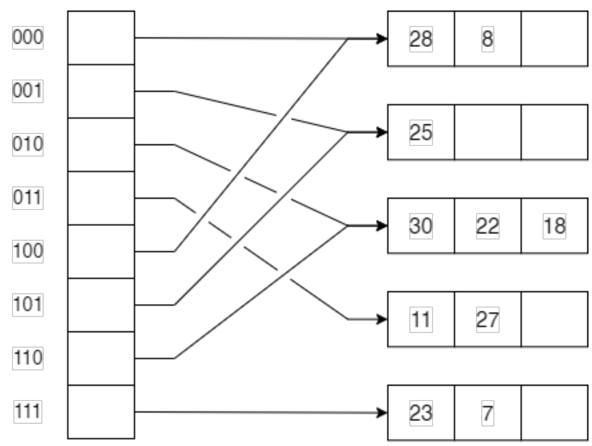


Figure 2: Extendible Hash Table along with the indexes of each bucket

Starting from the table above, delete keys 25, 18, 22, 27, 7.

i. **[4 points]** Which deletion first causes a reduction in a local depth.
☐ 25 ☐ 18 ☐ 22 ☐ 27 ☐ 7 ☐ None of the above

ii. **[4 points]** Which deletion first causes a reduction in global depth.
☐ 25 ☐ 18 ☐ 22 ☐ 27 ☐ 7 ☐ None of the above

Question 4: B+Tree .... [36 points]

Consider the following B+tree.

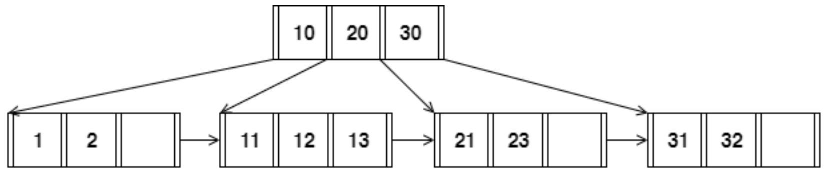


Figure 3: B+ Tree of order $d = 4$ and height $h = 2$.

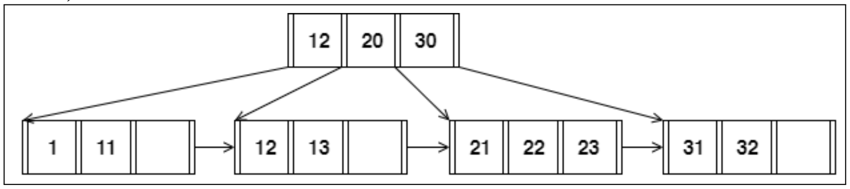
When answering the following questions, be sure to follow the procedures described in class and in your textbook. You can make the following assumptions:

- 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 below $\lceil \frac{d-1}{2} \rceil$.
- An internal node underflows when the number of **pointers** goes below $\lceil \frac{d}{2} \rceil$.

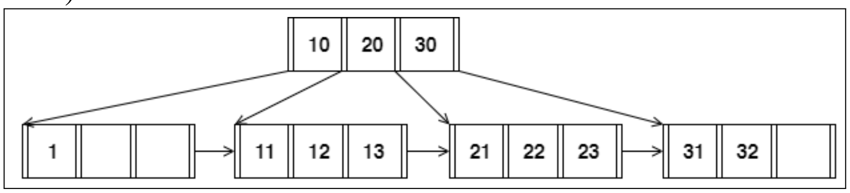
(a) **[2 points]** How many pointers (parent-to-child and sibling-to-sibling) do you chase to find all keys between 9^* and 19^*?
☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ☐ 7

(b) **[6 points]** Insert 22^* into the B+tree, then delete 2^*. Select the resulting tree.

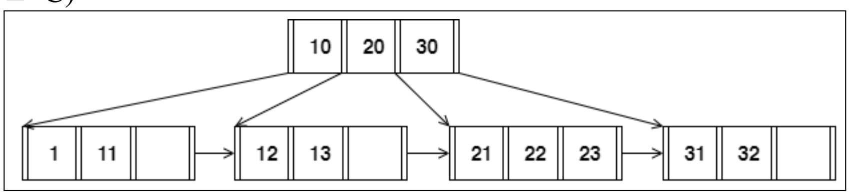
☐ A)



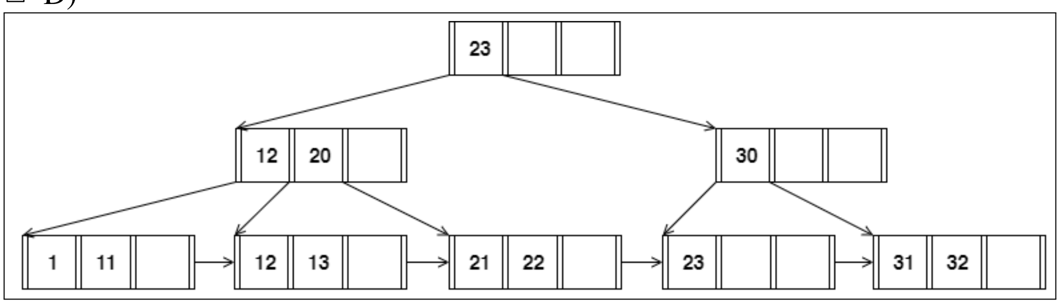
☐ B)



☐ C)

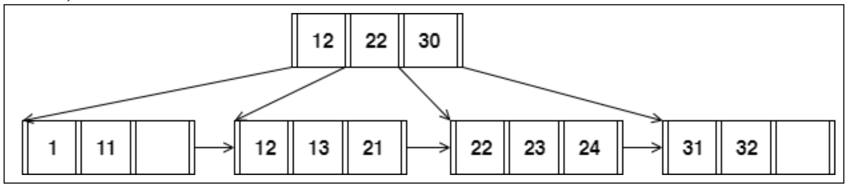


☐ D)

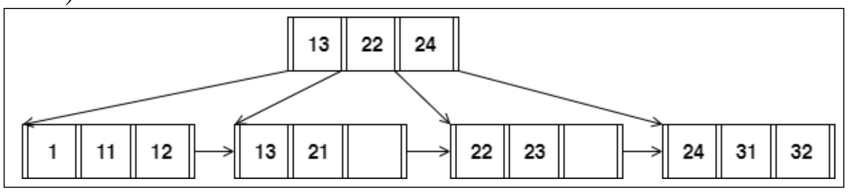


(c) **[10 points]** Then Insert 24^*. Select the resulting tree.

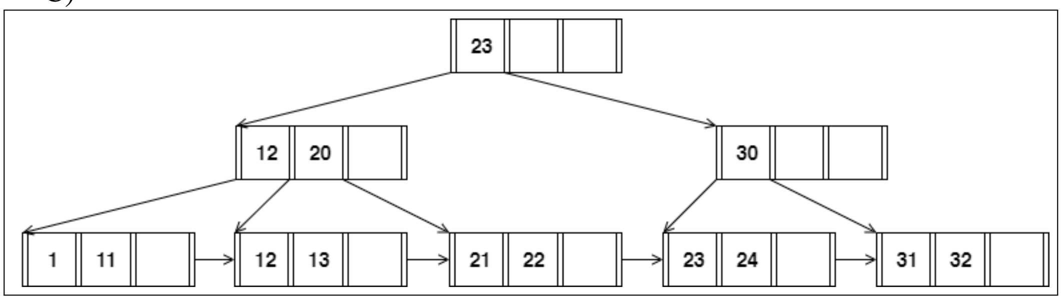
☐ A)



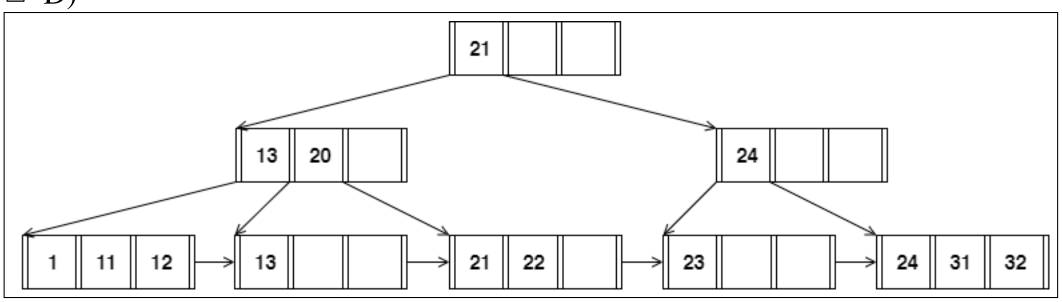
☐ B)



☐ C)

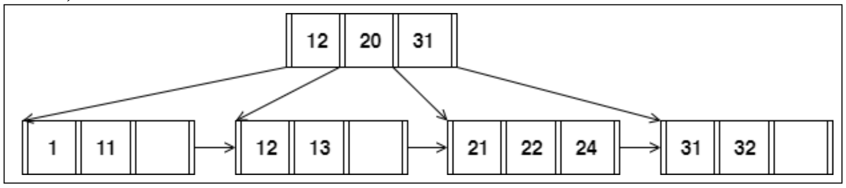


☐ D)

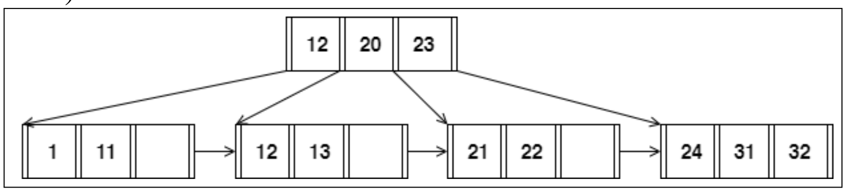


(d) **[10 points]** Finally, delete 23^*. Select the resulting tree.

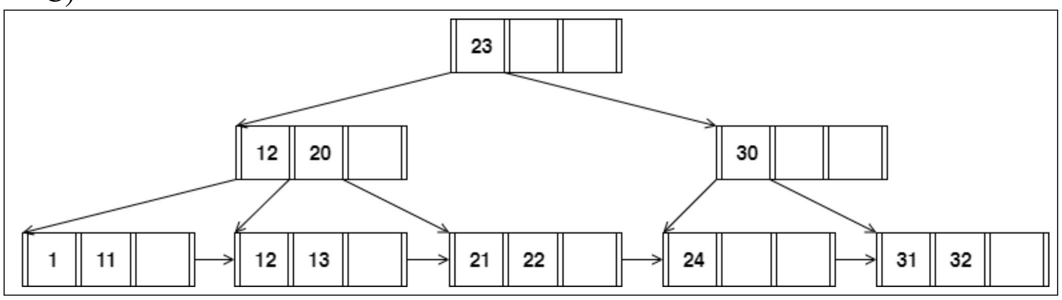
☐ A)



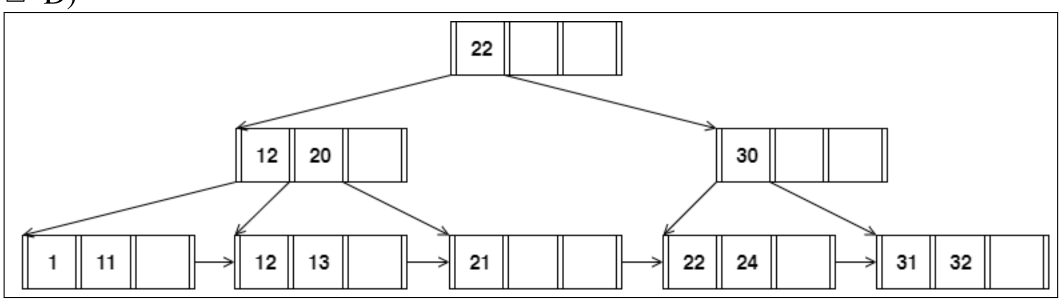
☐ B)



☐ C)



☐ D)



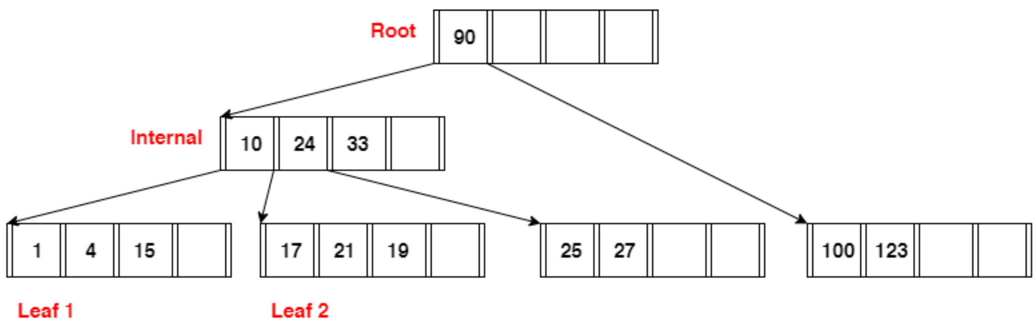


Figure 4: B+tree with violations

The B+Tree shown in Figure 4 is invalid. That is, its nodes violate the correctness properties of B+Trees that we discussed in class. If the tree is invalid, select all the properties that are violated for each node. If the node is valid, then select 'None'. There will be **no** partial credit for missing violations.

Note: If a node's subtrees are not the same height, the balance property is violated at that node only.

i. **[2 points]** Which properties are violated by **Leaf 1**?
☐ Key order property ☐ Half-full property ☐ Balance property
☐ Separator keys ☐ None

ii. **[2 points]** Which properties are violated by **Leaf 2**?
☐ Key order property ☐ Half-full property ☐ Balance property
☐ Separator keys ☐ None

iii. **[2 points]** Which properties are violated by **Internal Node**?
☐ Key order property ☐ Half-full property ☐ Balance property
☐ Separator keys ☐ None

iv. **[2 points]** Which properties are violated by **Root**?
☐ Key order property ☐ Half-full property ☐ Balance property
☐ Separator keys ☐ None