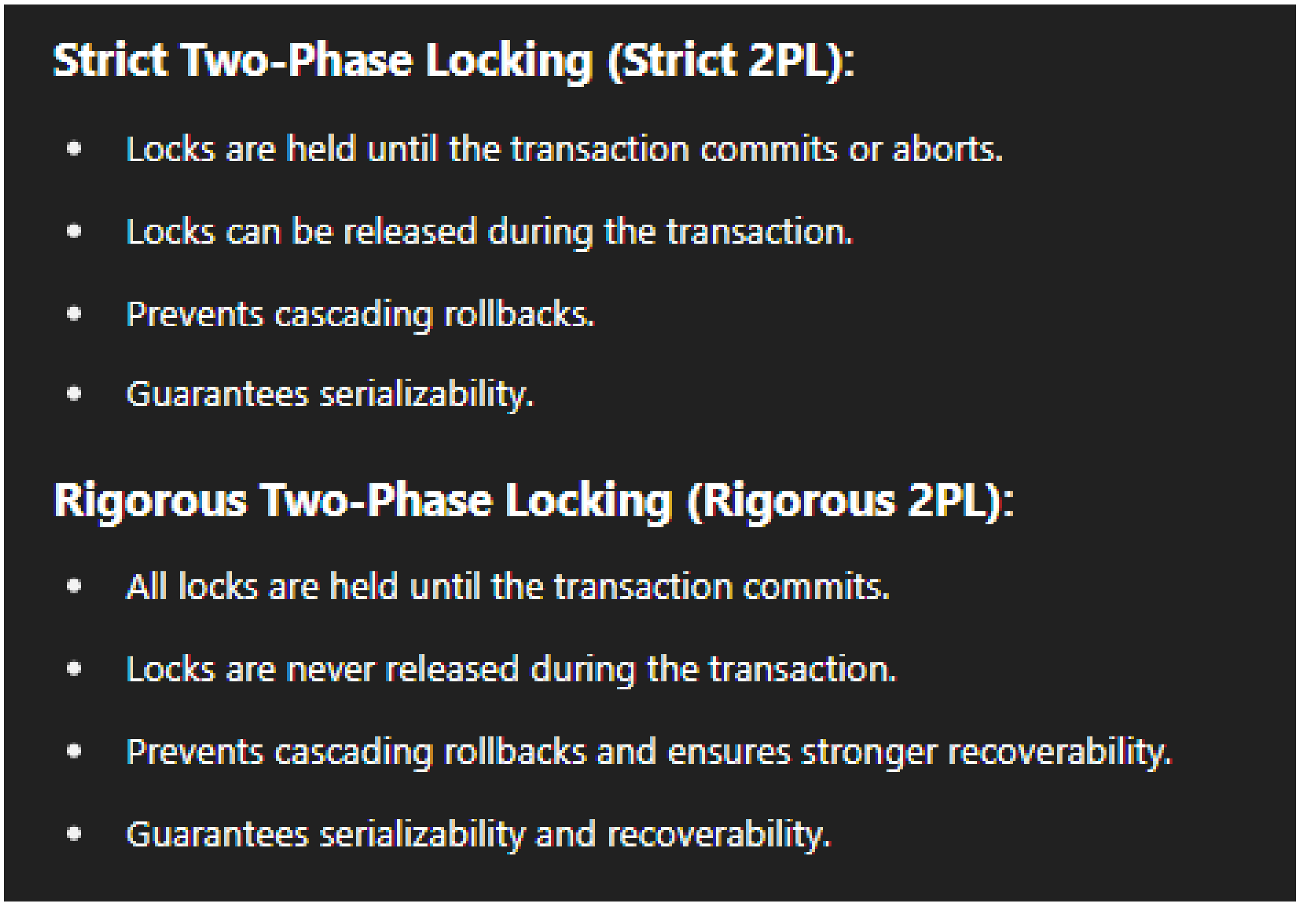
1. Differentiate strict two-phase locking protocol and rigorous two-phase locking protocol.



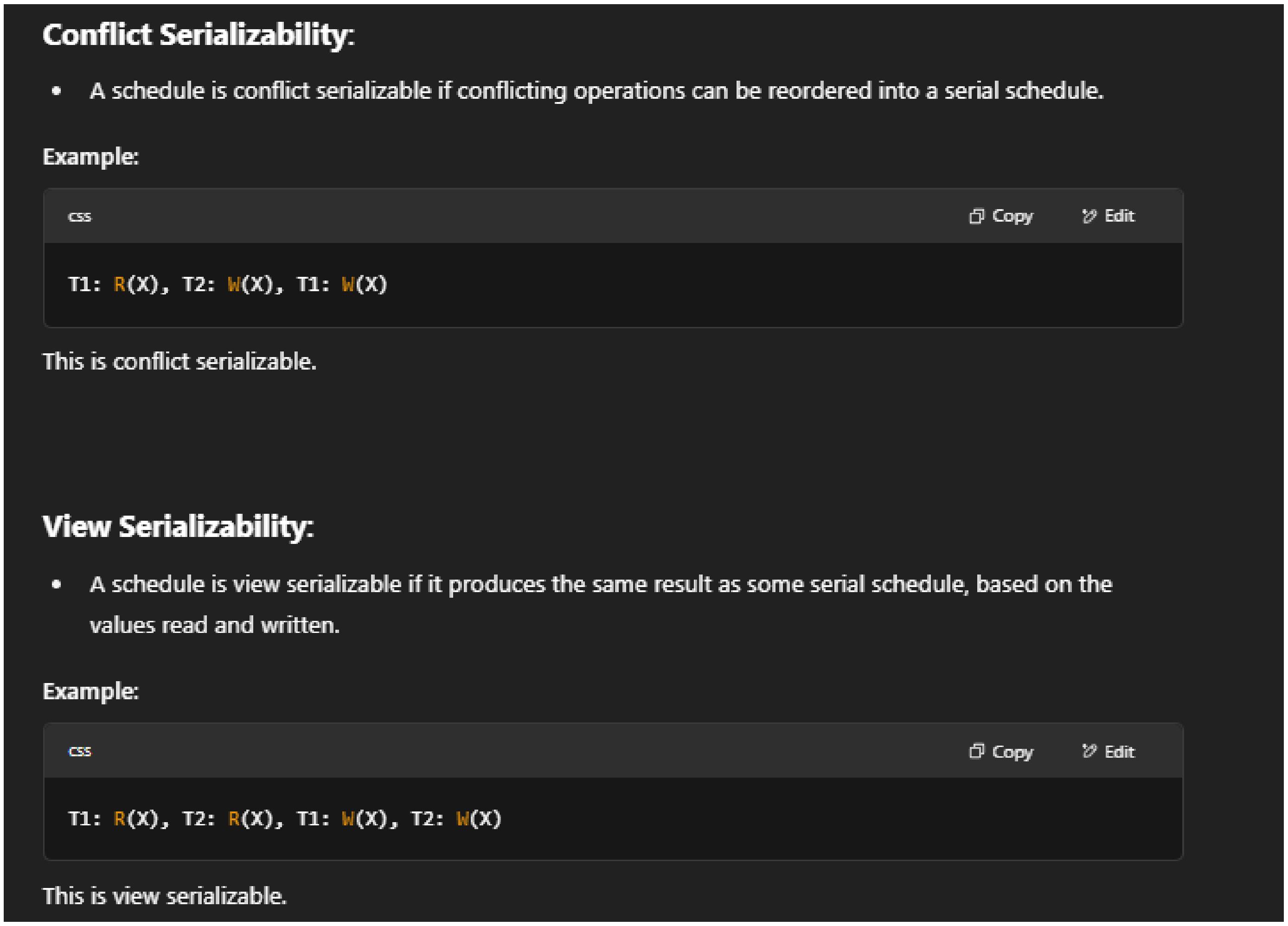
Strict Two-Phase Locking (Strict 2PL):

- Locks are held until the transaction commits or aborts.
- Locks can be released during the transaction.
- Prevents cascading rollbacks.
- Guarantees serializability.

Rigorous Two-Phase Locking (Rigorous 2PL):

- All locks are held until the transaction commits.
- Locks are never released during the transaction.
- Prevents cascading rollbacks and ensures stronger recoverability.
- Guarantees serializability and recoverability.

2. Illustrate the conflict serializability and view serializability with an example.



Conflict Serializability:

- A schedule is conflict serializable if conflicting operations can be reordered into a serial schedule.

Example:

```css
T1: R(X), T2: W(X), T1: W(X)
```

This is conflict serializable.

View Serializability:

- A schedule is view serializable if it produces the same result as some serial schedule, based on the values read and written.

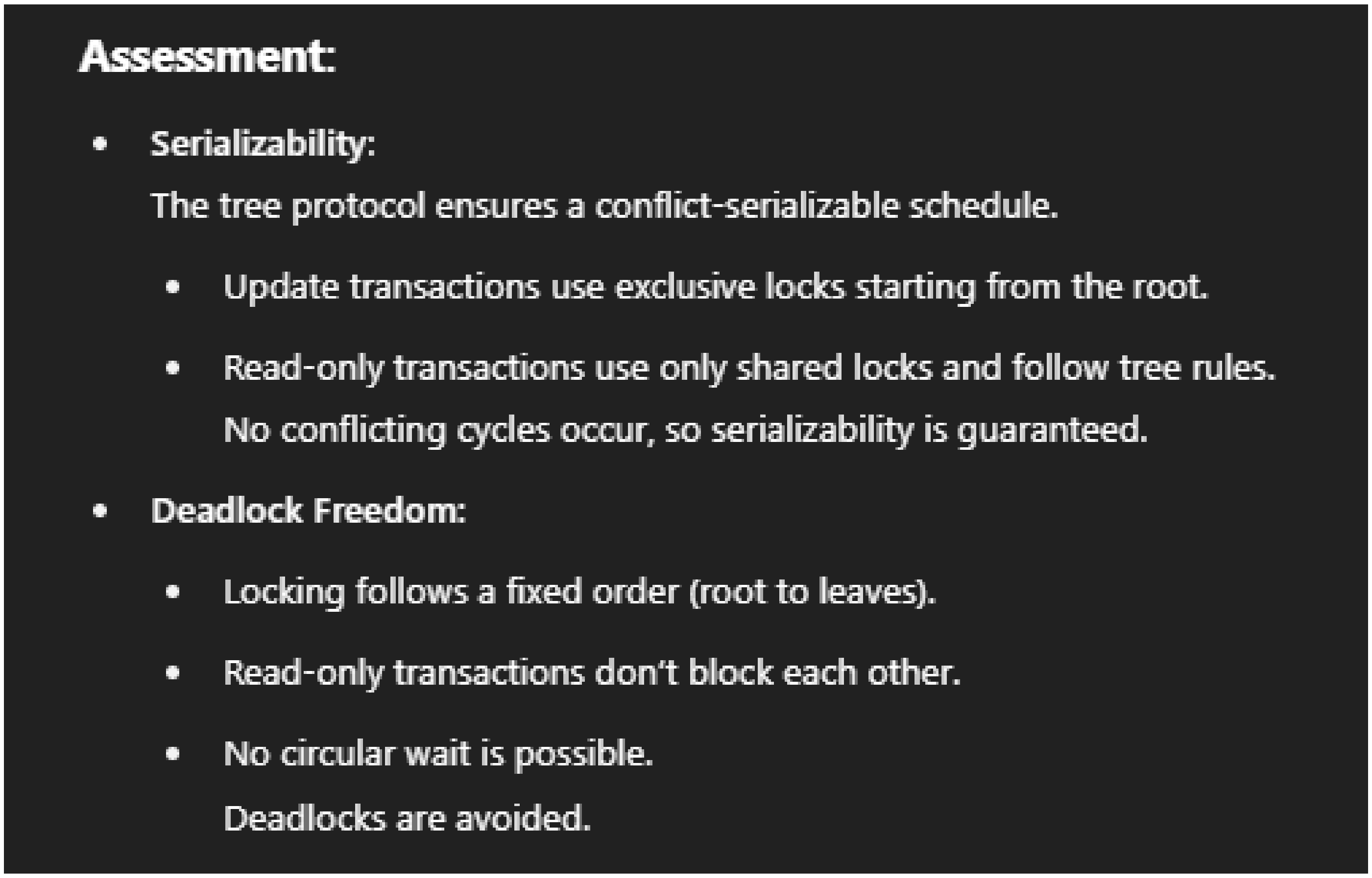
Example:

```css
T1: R(X), T2: R(X), T1: W(X), T2: W(X)
```

This is view serializable.

3. Consider the following extension to the tree-locking protocol, which allows both shared and exclusive locks:

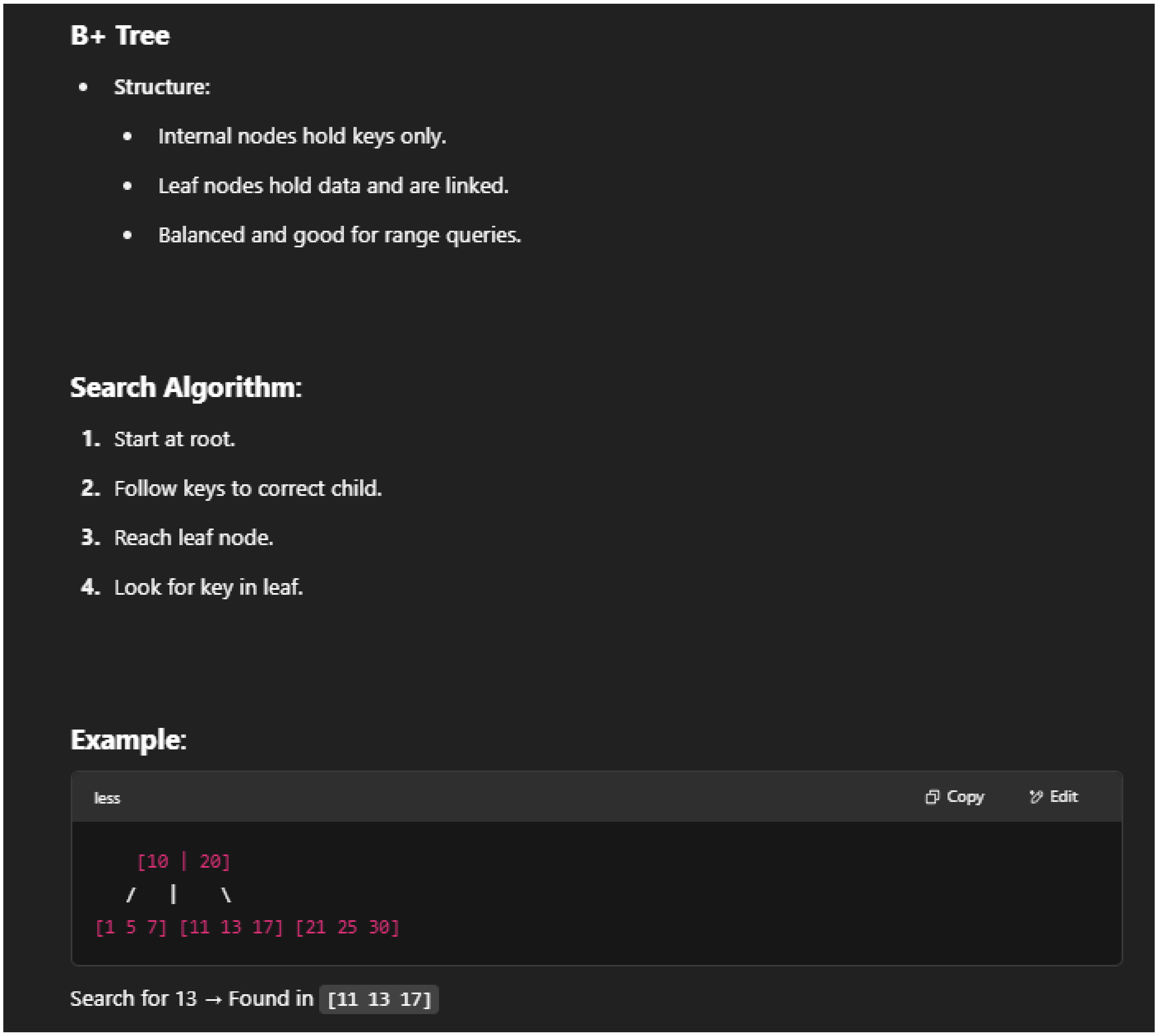
• A transaction can be either a read-only transaction, in which case it can request only shared locks, or an update transaction, in which case it can request only exclusive locks.
• Each transaction must follow the rules of the tree protocol. Read-only transactions may lock any data item first, whereas update transactions must lock the root first. Assess on that the protocol ensures serializability and deadlock freedom.



Assessment:

- **Serializability:**
  The tree protocol ensures a conflict-serializable schedule.
  - Update transactions use exclusive locks starting from the root.
  - Read-only transactions use only shared locks and follow tree rules.
    No conflicting cycles occur, so serializability is guaranteed.
- **Deadlock Freedom:**
  - Locking follows a fixed order (root to leaves).
  - Read-only transactions don't block each other.
  - No circular wait is possible.
    Deadlocks are avoided.

4. Demonstrate the structure of B+ tree and give the algorithm for search in the B+ tree with example



B+ Tree

- **Structure:**
    - Internal nodes hold keys only.
    - Leaf nodes hold data and are linked.
    - Balanced and good for range queries.

Search Algorithm:

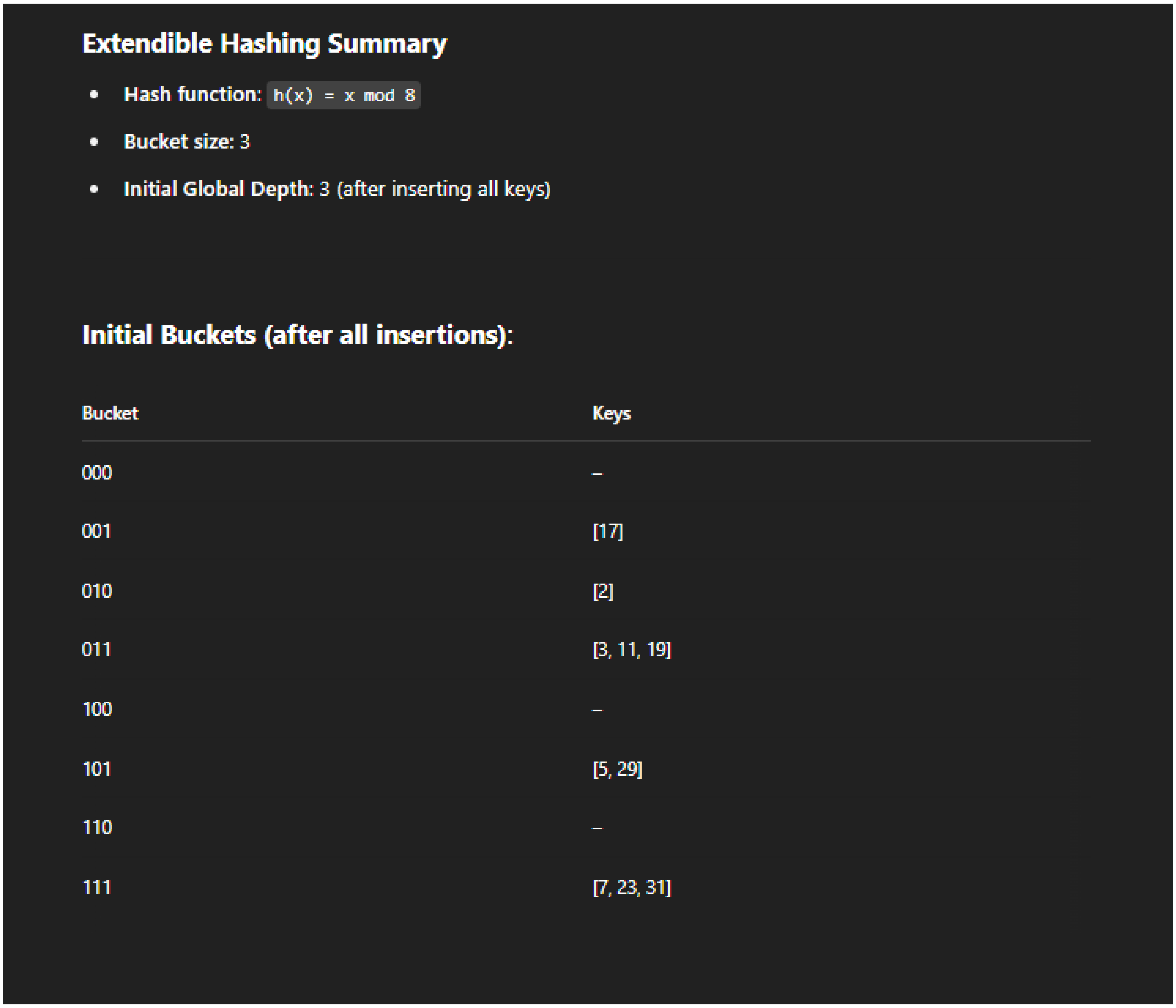
1. Start at root.
2. Follow keys to correct child.
3. Reach leaf node.
4. Look for key in leaf.

Example:

```less
    [10 | 20]
   /   |    \
[1 5 7] [11 13 17] [21 25 30]
```

Search for 13 → Found in `[11 13 17]`

5. The following key values are organized in an extendable hashing technique.
2, 3, 5, 7, 11, 17, 19, 23, 29, 31. Show the extendable hash structure for this file if the hash function is h(x)=x mod 8 and buckets can hold three records. Assess how the extendable hash structure changes as the result of each of the following steps: (15)
DELETE 11
DELETE 31
INSERT 1
INSERT 15

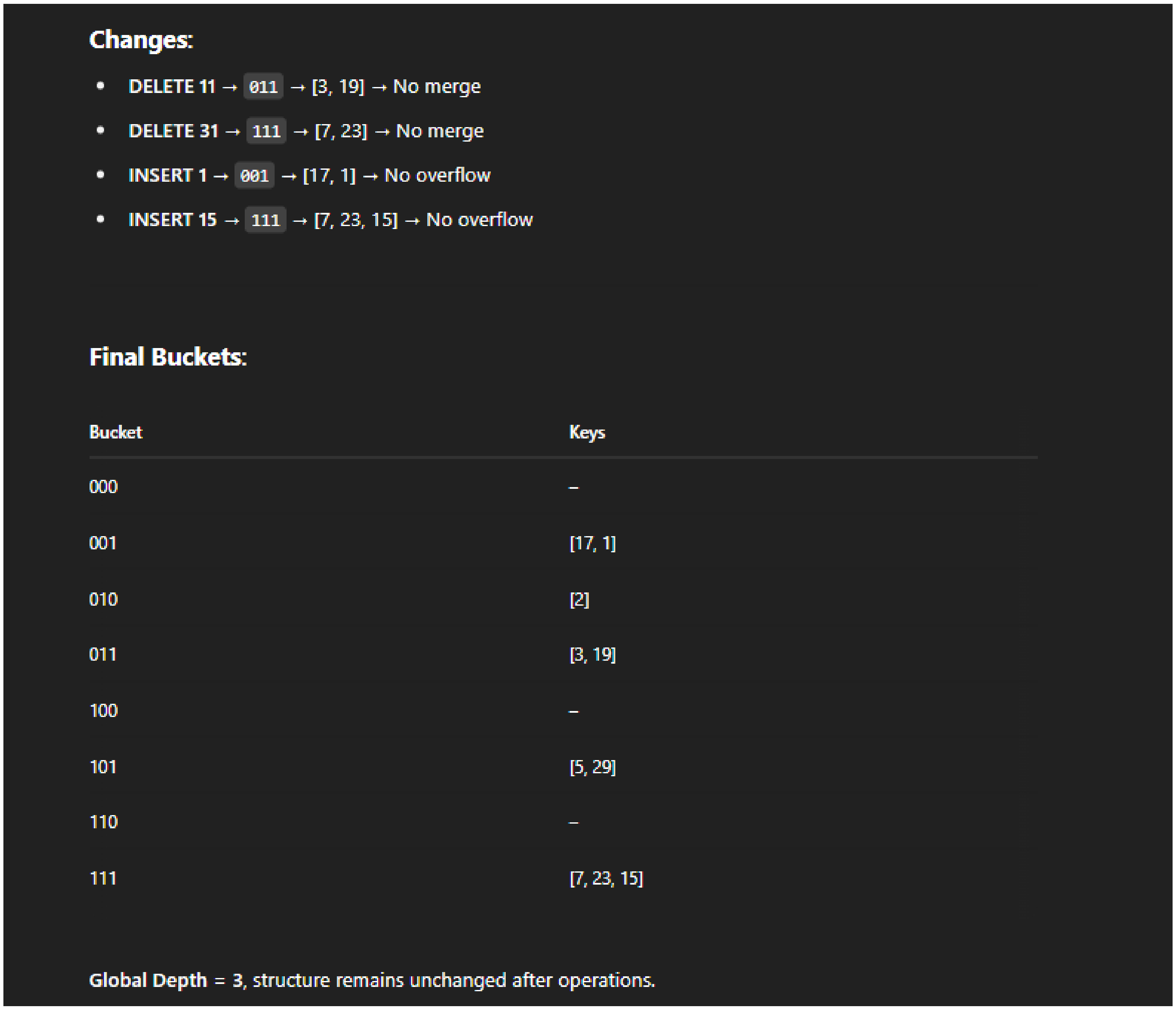


Extendible Hashing Summary

- **Hash function:** `h(x) = x mod 8`
- **Bucket size:** 3
- **Initial Global Depth:** 3 (after inserting all keys)

Initial Buckets (after all insertions):

Bucket	Keys
000	–
001	[17]
010	[2]
011	[3, 11, 19]
100	–
101	[5, 29]
110	–
111	[7, 23, 31]



Changes:

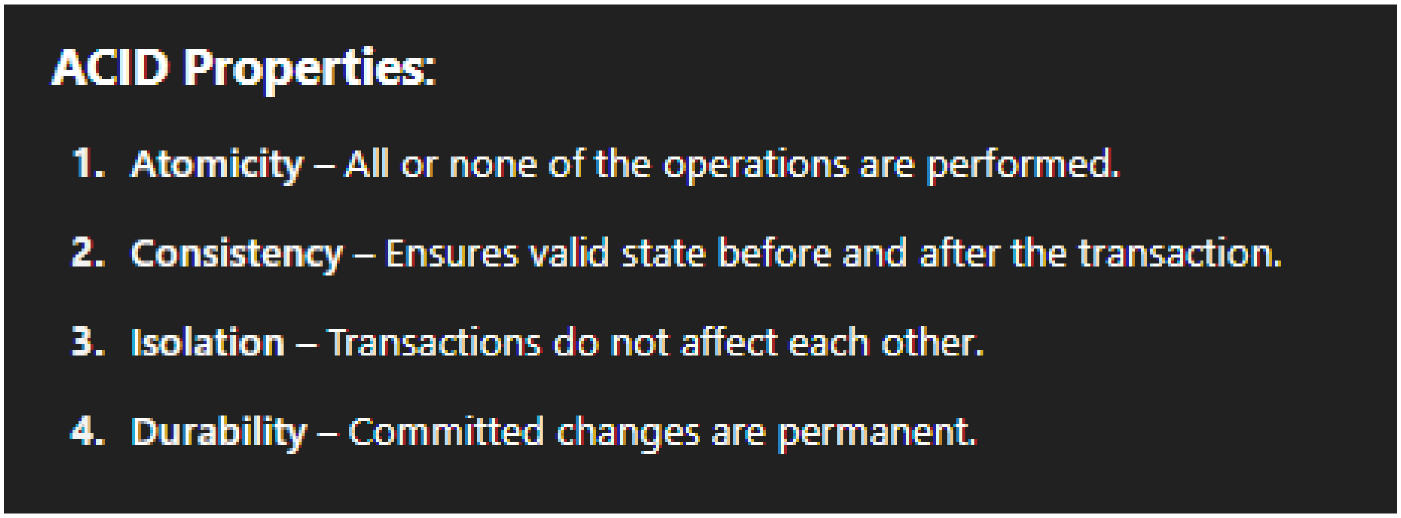
- **DELETE 11** → `011` → [3, 19] → No merge
- **DELETE 31** → `111` → [7, 23] → No merge
- **INSERT 1** → `001` → [17, 1] → No overflow
- **INSERT 15** → `111` → [7, 23, 15] → No overflow

Final Buckets:

Bucket	Keys
000	–
001	[17, 1]
010	[2]
011	[3, 19]
100	–
101	[5, 29]
110	–
111	[7, 23, 15]

Global Depth = 3, structure remains unchanged after operations.

6. Describe the ACID Properties of a transaction



ACID Properties:

1. **Atomicity** – All or none of the operations are performed.
2. **Consistency** – Ensures valid state before and after the transaction.
3. **Isolation** – Transactions do not affect each other.
4. **Durability** – Committed changes are permanent.