TASK-2

Things to add in code:

- Pager
- Writing to Disc
- Cursor
- B-Tree

Persistence to disk

- After .exit, data is lost
- Create a .db file to store this data, read/write to this file
- $ a.out filename.db
- We ask the pager for page number x, and the pager gives us back a block of memory. It first looks in its cache. On a cache miss, it copies data from disk into memory (by reading the database file).

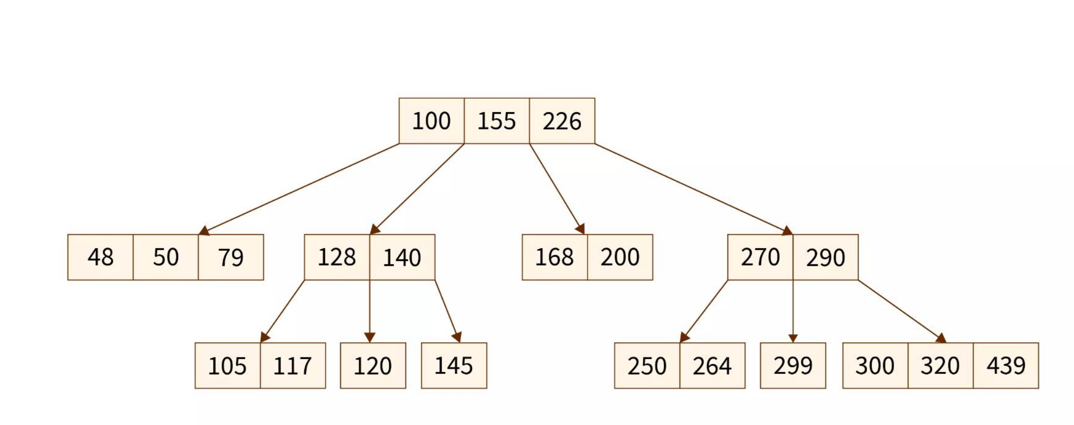
B-Tree

(NOT binary tree)
By Rudolf Bayer while working at Boeing labs

Advantages over Binary Tree

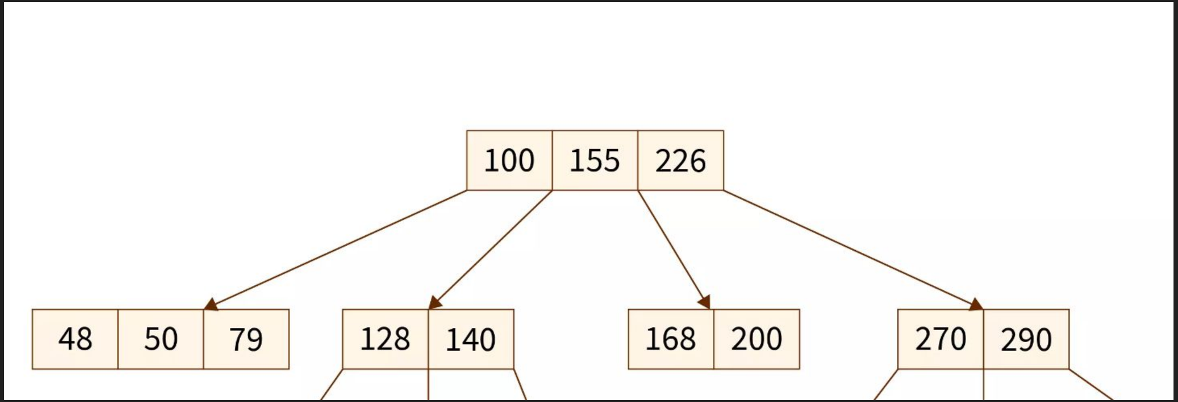
- B-Trees can store a large number of keys (here, page numbers) in a single node
- larger branching factor, shallower height, faster search and insertion operations
- Search, insert, delete in logarithm time
- Self Balancing

B-Tree Properties:



- B-Tree is defined by the term minimum degree ‘t‘.
- Every node except the root must contain at least t-1 keys.
- All nodes (including root) may contain at most (2*t – 1) keys.

B-Tree Properties:

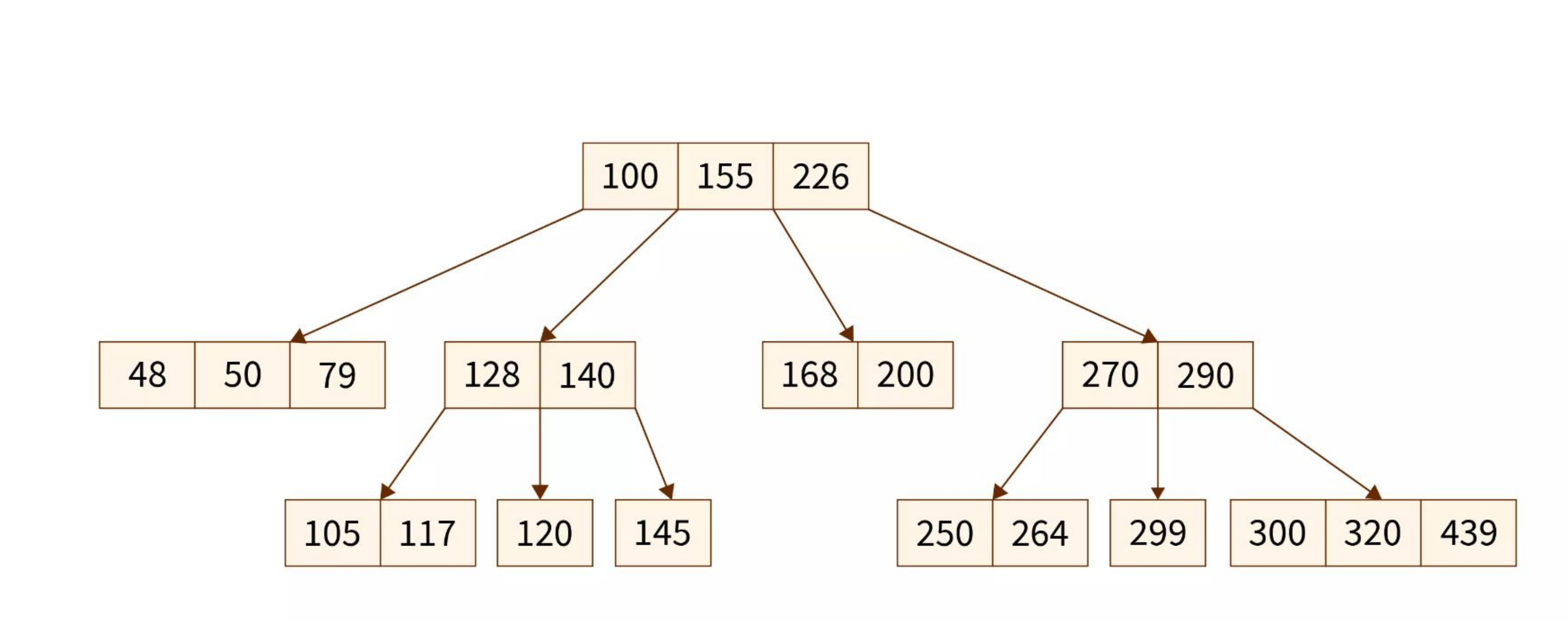


- Number of children of a node is equal to the number of keys in it plus 1. (Each node must have ‘t’ children)
- All keys of a node are sorted in increasing order. The child between two keys k1 and k2 contains all keys in the range from k1 and k2.
- Insertion of a Node in B-Tree happens only at Leaf Node.

B-Tree Properties: Root, Leaf

- Leaf nodes have 0 children
- The root node can have fewer than m children but must have at least 2, The root may contain a minimum of 1 key.
- If the root node is a leaf node (the only node), it still has 0 children

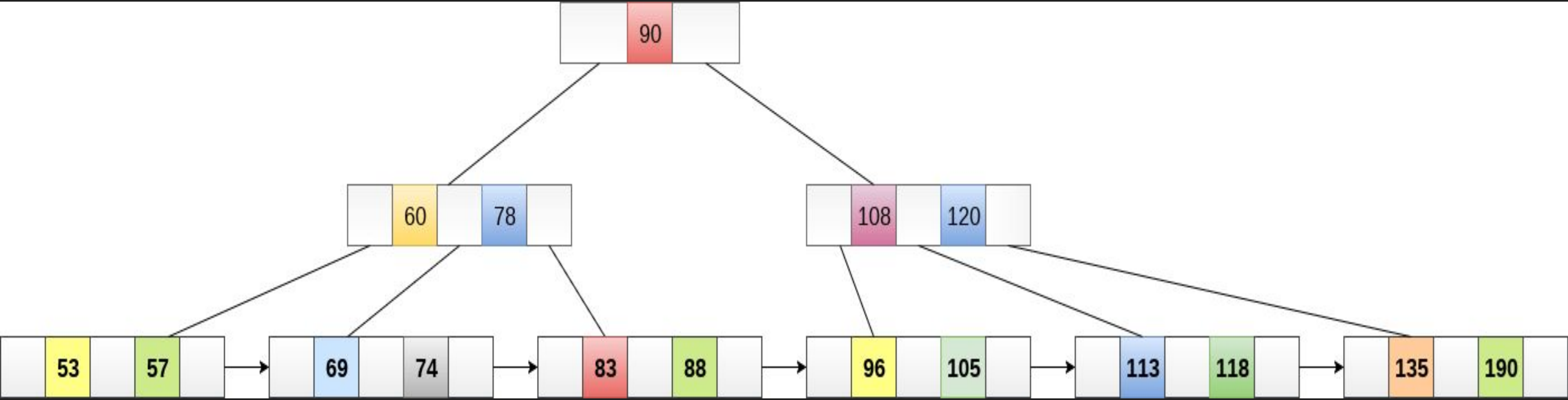
Search (Similar to Binary Tree)

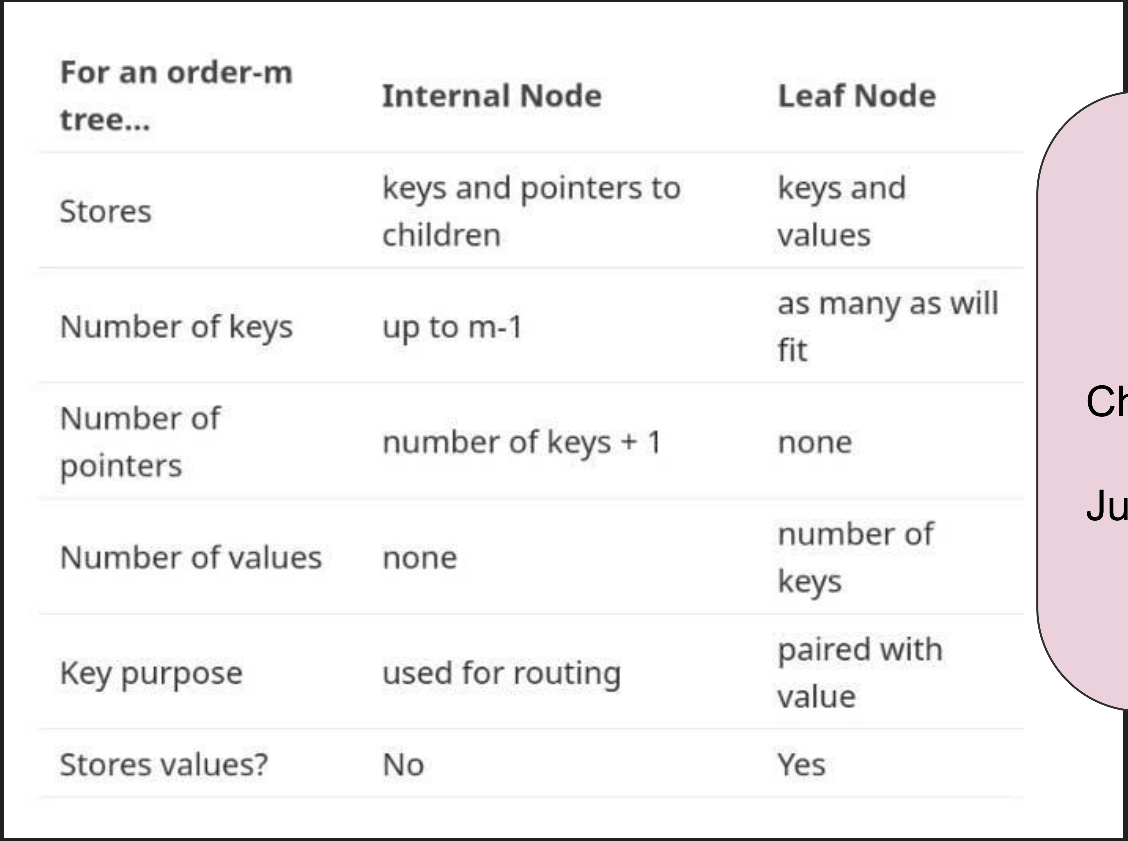


B+ Tree

- B + Tree is a variation of the B-tree data structure.
- In a B + tree, data pointers are stored only at the leaf nodes of the tree, structure of a leaf node differs from the structure of internal nodes.
- The leaf nodes have an entry for every value of the search field, along with a data pointer to the record.
- Internal nodes of a B+ tree are used to guide the search.

Leaf nodes form a linked list for efficient range-based queries





For an order-m tree...	Internal Node	Leaf Node
Stores	keys and pointers to children	keys and values
Number of keys	up to m-1	as many as will fit
Number of pointers	number of keys + 1	none
Number of values	none	number of keys
Key purpose	used for routing	paired with value
Stores values?	No	Yes

Min________Max

Keys: ceil(m/2)-1______m-1

Children: ceil(m/2)________m

Just for root: min=1

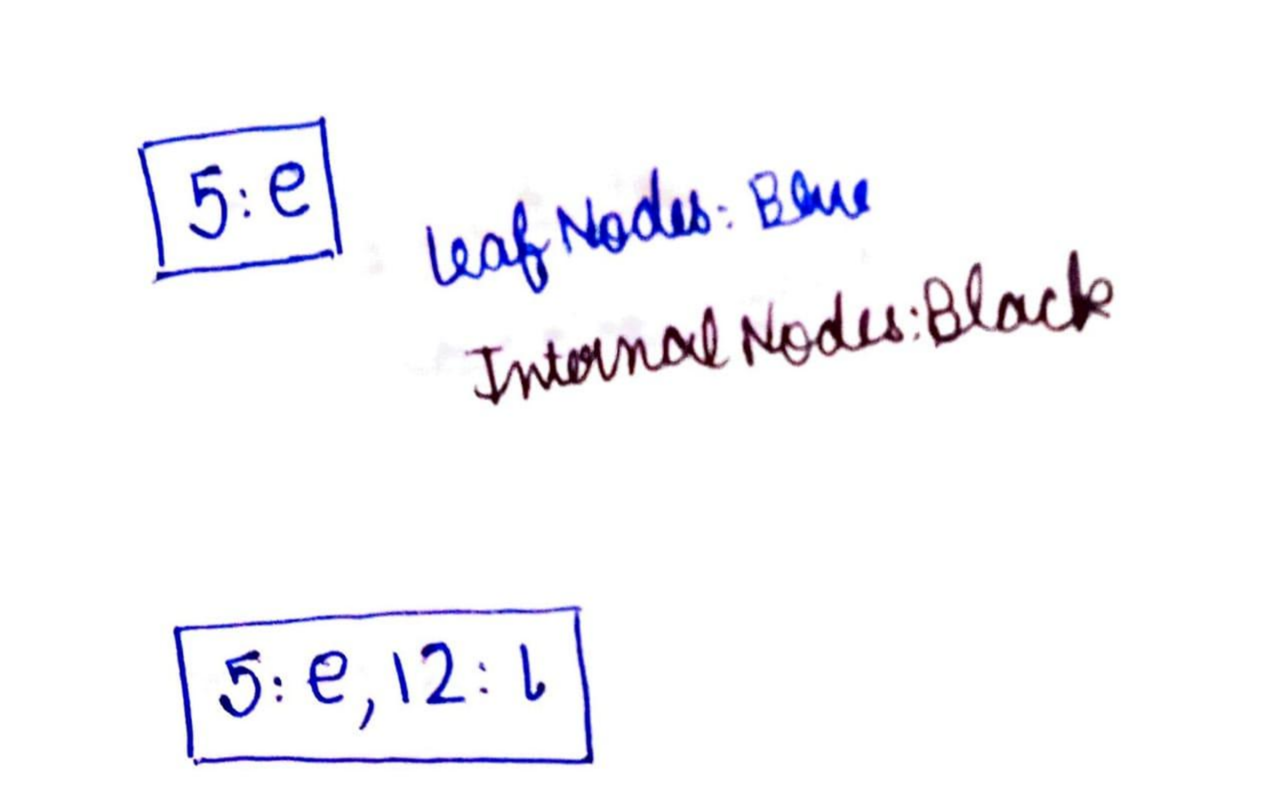
Construction:

- Every element is inserted into a leaf node. Go to the appropriate leaf node.
- In Case of overflow

  Split the leaf node into two nodes. First node contains ceil(m/2) values. Second node contains the remaining values. Copy the largest search key value from first node to the parent node.

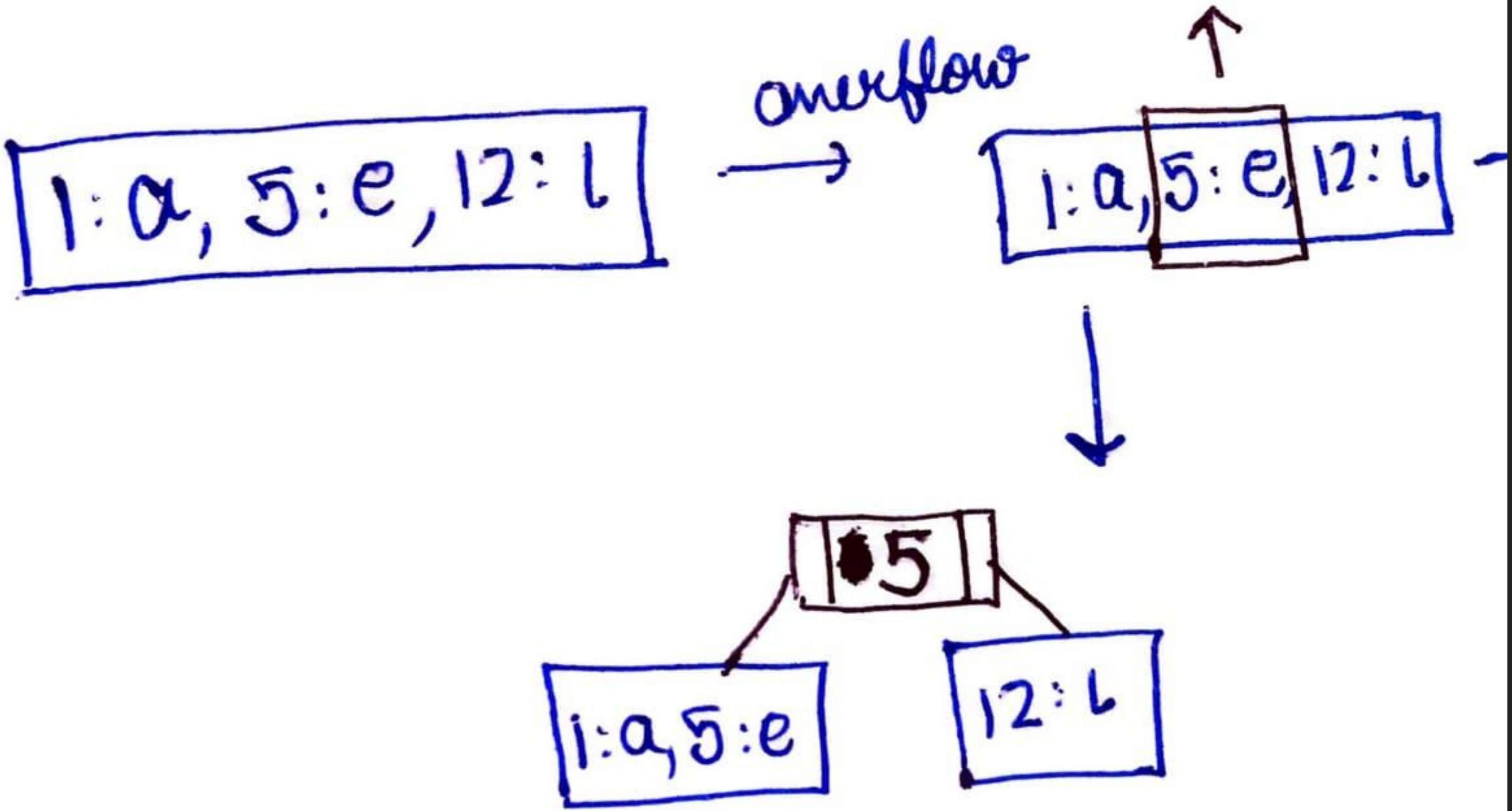
- In case of overflow in parent node: repeat Step 2

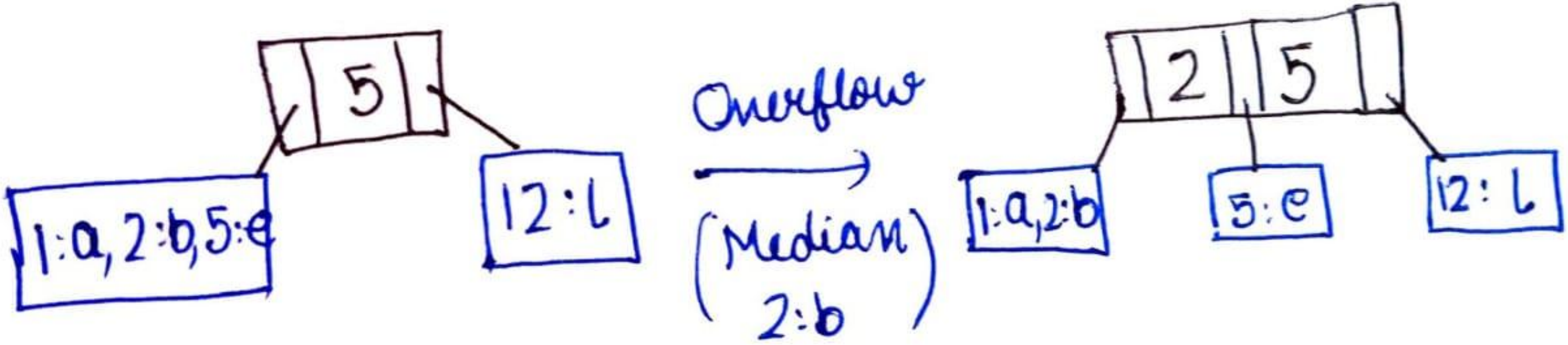
Example: Order=3, Keys: 5,12,1,2,18,21 Values: alphabets

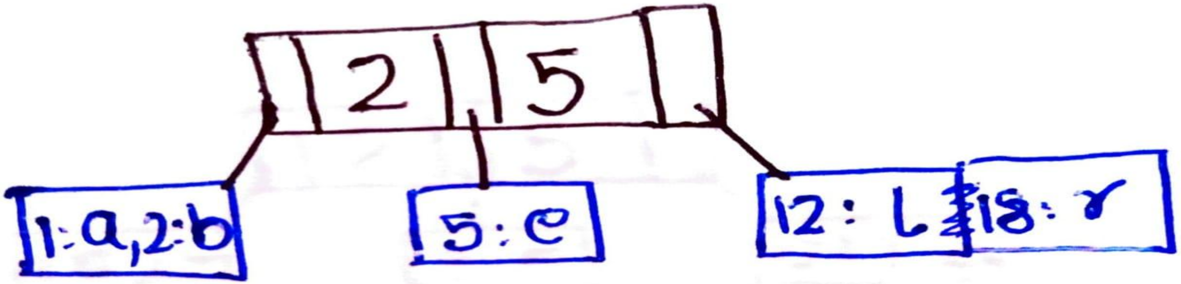


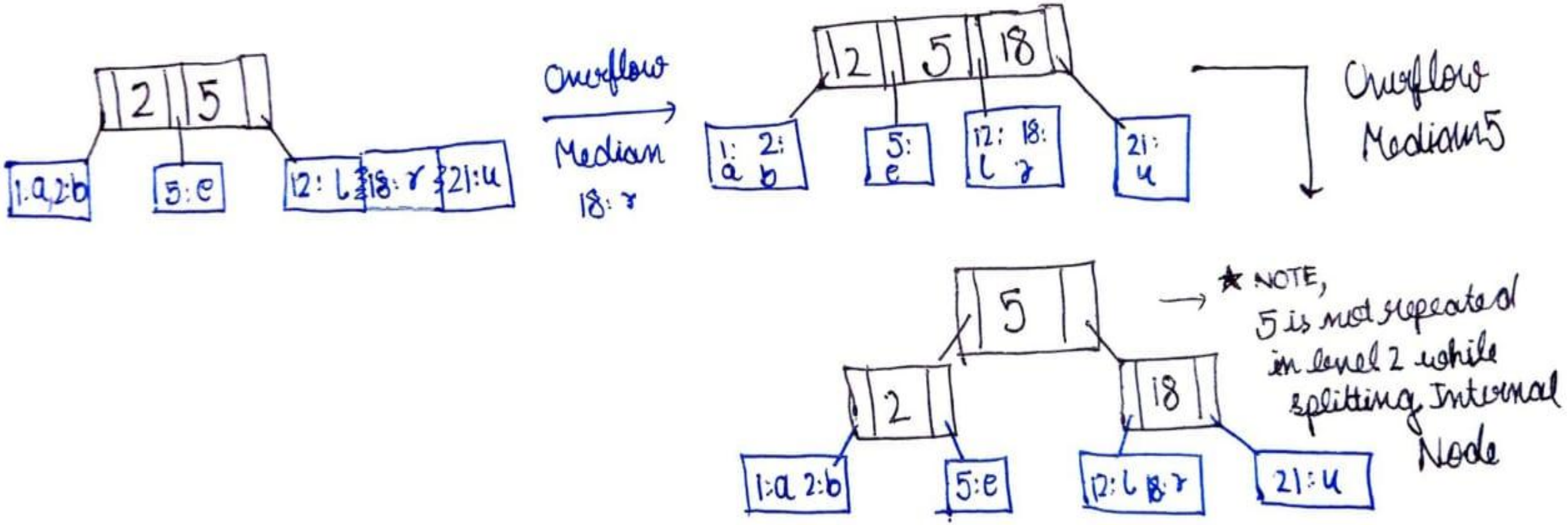
Leaf Nodes: Blue

Internal Nodes: Black


1:a, 5:e, 12:l
overflow
1:a, 5:e, 12:l
5
1:a, 5:e
12:l


5
1:a,2:b,5:e
12:l
Overflow
(Median 2:b)
2
5
1:a,2:b
5:e
12:l


2
5
1:a,2:b
5:e
12:l
18:r



★ NOTE,
5 is not repeated
in level 2 while
splitting Internal
Node