B+ Tree Coding Logic

Names: Youdan Xu, Tengli Fu, Haofei Ying

public T search(K key)

Search method that searches the given key in the tree. It actually calls the search helper method to complete searching.

private T search(Node root, K key)

Search helper method that searches the given key in the given tree of the given root. It is called recursively.

When searches in an IndexNode, it traverses the keys of node, find the key, then go down into the corresponding children node. If root is an LeafNode, it uses binary search to search the key. It returns null if the key is not in this tree.

public T search2(K key)

Another implementation of search mothod that does the same thing as search(K key), but it searches the key iteratively instead of recursively. It goes down until reaches the bottom most level - LeafNode level, then searches the given key and returns the value.

public void insert(K key, T value)

It inserts the key/value pair into the tree. If it inserts into an empty tree, the inserted node becomes the root. If the current root node is full, and there is an overflow from bottom up, a new root will be created, the original root becomes the second level. It inserts the pair into the tree through calling the insert helper method.

private Entry> insert(Node node, K key, T value)

It is the helper method for insert(K key, T value). It inserts the key/value into the given node.

If the node is full, it should be split into two nodes, which is handled in splitLeafNode and splitIndexNode methods.

public Entry> splitLeafNode(LeafNode leaf)

It handles the logic of splitting the leaf node into two leaf nodes - the left one contains D keys, the right one contains D+1 keys. It also handles the pointers which point to previous leaf and next leaf.

public Entry> splitIndexNode(IndexNode index)

It is similar to the splitLeafNode method. The only difference is that it does not need to handle the pointers point to previous node and next node.

public void delete(K key)

It deletes the given key into the tree. It calls the delete helper method to implement deleting.

private int delete(Node node, K key, IndexNode parent)

It is the helper method for delete(K key). The arguments are the root node, the key to delete, and the parent node of the root. It recursively handles the logic of deleting, and returns the index of the key in the parent node. It calls handleLeafNodeUnderflow and handleIndexNodeUnderflow methods to handle the login of underflow when deleting.

public int handleIndexNodeUnderflow(IndexNode left, IndexNode right, IndexNode parent)

It handles the logic of situations of underflow. When there is underflow in an index node, it firstly tries to redistribute between the node left to it. If the total number of keys of the two neighbour nodes is not enough to redistribute, it merges the two nodes, and returns the index of the key to delete in the parent node.

public int handleLeafNodeUnderflow(LeafNode left, LeafNode right, IndexNode parent)

It is similar to the handleLeafNodeUnderflow method. The difference is that it needs to copy the first key of the right node up to the parent node at corresponding index.

public String toString()

It overrides the original toString() method for easier debugging.