Explanation of the code for the 3rd Assignment in Data Structures

George Valavanis

B+tree : To implement the tree, first of all instead of nodes now the pointers of a node (pointer for the left-right child/sibling, parent) are integers that indicate on which page of the file the right-left child/sibling or parent of the node is located. There is a "memory" called StorageCache that holds the last node read on disk and its role is to reduce disk accesses. But in general, when we need access to a node we check if it exists in StorageCache otherwise we read from disk. There are appropriate methods that support B+tree functions such as split,borrow,merge .

Random Key Insertion: To achieve random node insertion we first generate a random key from a random number generator and then insert it into the tree. To insert N random nodes a random number generator returns a table of unique random keys and then one by one it is inserted into the tree.

Random Key Deletion: A generator generates a random key and then calls the delete method delete , which searches for the node with that key and deletes it if it exists.

Random Key Value Search: A generator generates a random key and then calls the search method , which searches for the specific node and returns its data if it exists, otherwise returns null.

Key Value Range Queries: A random number generator first generates the random range of values. It then uses findRange to search for the node with a value equal to the lower bound of the range, if none exists then it returns the node with the closest value to the lower bound of the range. It then visits each key of the node serially until it finds a key greater than the upper limit of the range. If no key greater than the upper limit is found in the same node then it visits its right sibling and checks its keys serially.