Writeup Lab2

Page Eviction Policy

I use LRU algorithm for page eviction, which always evicts the page the least recently used first. To implement LRU algorithm, I use a LinkedList LRUList to store the pages recently used, and a HashMap pageId2Loc to map pageid to the page's location. When using a page, I add it to the tail of LRUList.

Search in B+ Tree

Search from rootnode recursively, until reach a leafnode and return it.

Insert in B+Tree

First, search recursively for the leafnode where the new node should be inserted to. If the leafnode isn't full, just insert it. Else, Split the leafnode, move the right half pages stored on the node to the new leafnode, and add a new entry to parentnode. Note this may cause parentnode split recursively, but it has been implemented by the given code.

Delete in B+ Tree

First, delete the node. If a leafnode become less than half full, it should either steal tuples from one of its siblings or merge with one of its siblings. Remember to update the parentnode's entry accordingly.