CS660 PA2

1.Description

`evictPage()`

- use strategy of LRU, Least Recently Used, to evict pages;
- use a hash map to store the page id and times that each page is used;

`BTreeFile.findLeafPage()`

- If pid is a leaf page, just get the required page;
- If pid is a internal page, do binary search recursively to find the required page;

`splitLeafPage()`

- create a right leaf page, and copy the last half of current leaf page to it;
- copy midfield to parent page;
- return the required page according to field;

`splitInternalPage()`

- create a right leaf page, and copy the last half of current leaf page to it;
- push midfield to parent page;
- return the required page according to field;

`BTreeFile.stealFromLeafPage()`

- sibling has s tuples and page has t tuples, we have to move (s-t+1)/2 tuples from sibling to page, so that the tuples are evenly distributed;
- update pointer of parent;

`BTreeFile.stealFromLeftInternalPage()`

- left sibling has s tuples and page has t tuples, move (s-t+1)/2 tuples from left to page;
- put last key of left into parent;
- update parent pointer;

`BTreeFile.stealFromRightInternalPage()`

- move (s-t+1)/2 tuples from right to page;
- put first key of right into parent;
- update parent pointer;

`BTreeFile.mergeLeafPages()`

- move tuples from right to left;
- update sibling pointers of left;
- delete the entry in the parent;
- update dirty pages of left and parent;

`BTreeFile.mergeInternalPages()`

- pull down the parentEntry key;
- move entries from right to left;
- update parent pointers;
- delete the parent entry;
- update dirty pages of left and parent;

2. No change to the API.

3. Complete all requirements of PA2.

4. Totally spend about 7 hours on this lab.