Report of Lab2

Yuxuan Duan 516030910573

1 Design Decisions

1.1 Eviction Policy

I implemented two policies: Simple Least Recently Used (LRU) Policy and Random Policy. Simple LRU Policy is currently the default, yet the policy can be changed via *BufferPool.EVICT_POLICY = LRU_POLICY* or *RANDOM_POLICY*.

The LRU Policy I have implemented is regarded as simple because the buffer pool only records the timestamp when a page was last visited, checks the timestamp of every page in the buffer pool and chooses the least recently visited one. There are more efficient yet more complicated algorithms to implement the LRU Policy, but I did not implement them due to the limit of time.

On the unit tests provided, the efficiencies of Simple LRU Policy and Random Policy do not differ much.

1.2 B+ Tree Operations

I did not do major special design to tuple inserting and deleting in my code. Most of the work has followed the document.

2 API Changes

In Lab1 I used *java.io.RandomAccessFile* to read pages, and have made minor changes to the API (with most of them only added *throws IOException*). While, in Lab2 I catch the IOExceptions just in *readPage()*, so the API of many methods were reset.

3 Missing or Incomplete Elements

There was no more missing or incomplete element than those in Lab1 to the best of my knowledge.

4 Time Spent and Difficulties

I roughly spent 18 hours on Lab2. I had spent much time on the EvictionTest before I discovered that the HeapFileIterator I had implemented in Lab1 was space-consuming, which was the key reason that the memory limit was exceeded and failed the EvictionTest. Insert in B+ Tree also took me hours debugging because I was not familiar with the utilities provided and it was also difficult to check the contents of the B+ Tree.