6.830 Lab 3 writeup

Katie Siegel

Collaborated with Ankush Gupta

1 Design Decisions

I did not make any unusual or noteworthy design decisions in this lab. Overall, I chose to use iterators heavily and lists sparingly when dealing with the tuples and entries.

For example, with the findLeafPage method, I simply iterate through the pages recursively until I reach a leaf page. I use an iterator every time I need to search through an internal page for the correct parent entry. I make sure that my checks take into account the possibility that both child pages of a parent entry may have the searched-for entry.

With regards to the insert methods, I also heavily use iterators. In the splitLeafPage and splitInternalPage methods, I chose to take a step-by-step approach in implementation. After first breaking down the problem down into concrete, serial steps, I then wrote the code that matched each step within each method. You can see these concrete steps in my code, as each step is clearly commented. I reasoned that following this step-by-step approach would make my code most readable to others. All in all, these split pages methods first search for the relevant parent entries, then shuffle around the child entries, and the finally adjust the parent entries accordingly. This approach generally works bottom-up, an approach I found most intuitive.

With regards to the delete methods, I chose to abstract out the code that deletes the parent entry in the pointer and re-inserts it among the destination page. This code was the same between both the leftSibling and rightSibling cases, so could easily be made into a new method. Even though the leftSibling and rightSibling code was relatively similar with regards to the delete code, there were significant enough structural differences between the two that I chose to not generalize. For example, I chose to use a Stack with the leftSibling entry shuffling code, and an ArrayList with the rightSibling entry shuffling code. These two data structures were most natural to use given the direction in which data was read from these siblings by an iterator and the direction in which the entries had to be inserted into the other page. Finally, with the internal page delete methods, I chose to first handle the former parent entry and then shift over the remaining entries that had to move. This logic meant inserting all entries into the destination page in order, which makes the most intuitive sense.

2 Changes to API

I did not make any changes to the API. I simply edited the BPlusTreeFile java file. I also had to update my BufferPool file to handle eviction, inserting tuples, and deleting tuples properly. In the HeapFile class, I also had to change all calls to BufferPool.PAGE_SIZE to

BufferPool.getPageSize(), as I had previously forgot to generalize these calls.

3 Missing or Incomplete Code

There is no missing or incomplete code in my project; it passes all of the tests.

4 General Comments

This lab was honestly an overwhelming amount of work. I easily spent over 20 hours working on the lab. I felt that there was very little direction as to what to implement in the methods, so I spent hours debugging. However, I was seriously impressed by Rebecca Taft's Piazza response rate–every answer was really helpful, and helped alleviate some frustration with the lab.