

CS525: Test Cases for the B⁺-Tree

Wai Gen Yee

November 27, 2004

1 Introduction

We will be testing the ability of the B⁺-tree to correctly handle inserts and searches. To do this, we will count the number of blocks accessed per operation, and compare them to predicted values.

We will also test whether the buckets contain the right values. When they split, they should contain the right values.

2 Tests

Open index, called index.idx.

The index.idx file should be 128B long, containing one header page and one leaf bucket.

2.1 First Insert

Insert "0, 1", where 0 is the key, and 1 is the value. See Figure 1.

2.2 Filling the Leaf

Insert "5, 2." Now the leaf should be full. See Figure 2.

Look up key 5. This should return 2, and indicate that 1 bucket was read.

Look up key 3. This should return an error, and indicate that 1 bucket was read.

2.3 Splitting a Leaf

Now insert "3, 3." This should cause a split of the leaf bucket. Half the keys should go to the left leaf, and half should go to the right leaf. If the number of keys is odd, put the extra one in the left leaf.

If the "split debug flag" (function 11) were turned on, the index would have printed the message "Splitting bucket 0. Creating bucket 1. Inserting key 5 into parent. Creating parent bucket 2." Or something like that.

0	
1	

Figure 1: After First Insert.

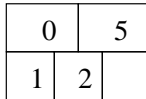


Figure 2: After Second Insert.

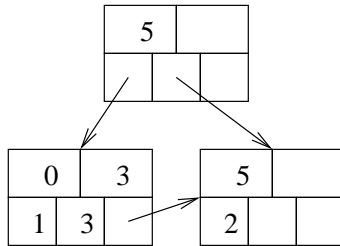


Figure 3: After Third Insert.

The size of index.idx should be $4 \times 64 = 256\text{B}$. As in Figure 3, there should be three buckets in the B+-tree. I use arrows to indicate pointers. In your implementation, you should use bucket numbers, of course.

Now, search for key 5. The index should return the value 2, and that it took 2 bucket reads to find this key.

Insert key "3, 4." This should return an error, because there already is a 3 in the index.

2.4 Inserting into an Internal Node

Insert key "4, 4." This will cause a split of the leftmost leaf, which is already full. See Figure 4.

Now, the file should be $5 \times 64\text{B} = 320\text{B}$ long.

If we do a lookup of 5, the index should return 2, and indicate that 2 buckets were searched.

If we do a "naive lookup" of 5, the index should return that 4 buckets were searched. One for the root, and three to search the leaves from left to right.

2.5 Overflowing the Internal Nodes

We will now insert "2,5" into the index. This overflows the leftmost leaf, causing a split. It's special, because it also overflows the leaf's parent, causing a split. Ultimately, the tree gets taller, and consumes 7 buckets. See Figure 5.

If the split debug flag were turned on, then the index would have indicated that two buckets were split by this insert.

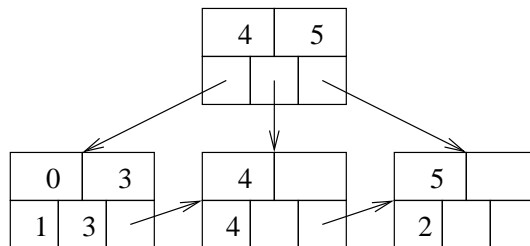


Figure 4: After Fourth Insert.

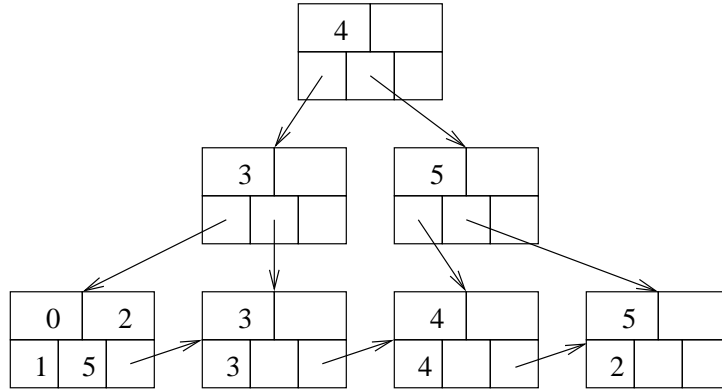


Figure 5: After Fifth Insert.

In these example, when we look at your pages, we should be able to see the correct keys and pointers.

2.6 Other Tests

Other tests: I will give you a file containing lots (potentially 1000's) of key/pointer pairs, of the format:

0,1
 5,2
 3,3
 4,4
 2,5

etc...

And will run the lookup tests on the resultant index.

3 Conclusion

Try these tests out, and give me your comments. Thanks and good luck!