1 Practicing $B^+$ Tree Insertions

1. Consider a $B^+$ tree of order 2.
   (a) How many maximum number of keys can we have in a single node? 
   (b) What is the least number of keys we can have in a root node? 
   (c) What is the least number of keys we can have in a non-root node? 
   (d) What is the maximum number of pointers for a non-leaf page? 
   (e) Starting from an empty $B^+$ , insert the following keys in the same order as shown (we need not show the tree at each step; just the final one):
      $15, 21, 13, 30, 42, 50$
(f) For each of the following sub-questions, we will be doing some more insertions on the resultant tree from question 1.

i. Insert key 60.
ii. Insert key 70.
iii. Insert key 28.
iv. Insert key 80.

We will be doing each of these insertions in two ways (show the tree after each step):

1. Use only splitting of nodes (if necessary).
2. **Try to redistribute** with neighbors if possible, otherwise, split (if necessary).

*Sub-questions repeated below for reference:*

i. Insert key 60.
ii. Insert key 70.
iii. Insert key 28.
iv. Insert key 80.

2. What is the **minimum number of insertions** that we must perform on the resultant tree from question 1 to increase the height of the tree by 1?
2 $B^+$ Tree Deletions

We have seen how to look-up and insert keys into a $B^+$ tree. We will now look at how to perform deletions.

To delete key $K$ from a $B^+$ tree:

- Start at the root, and find the leaf $L$ where entry $K$ belongs to.
- Remove the entry.
- If $L$ is at least half-full, we are done!
- If $L$ underflows:
  - Try to **re-distribute** (i.e., borrow from a "rich sibling" and "copy up" its lowest key).
  - If re-distribution fails, **merge $L$** and a "poor sibling."
    * Update parent.
    * And possibly, merge **recursively**.

2.1 Examples

Suppose we have the following $B^+$ tree with order 2.

![Figure 1: Our initial $B^+$ tree](image)

1. Deleting key 18 from the original tree (Figure 1) will result in the following:

![Figure 2: Our resulting $B^+$ tree after deleting 18 from Figure 1](image)

We simply found our way to the correct leaf, and removed the key.
2. Deleting key 5 from the resultant tree (Figure 2) will result in the following:

![Diagram of B^+ tree after deleting 5 from Figure 2]

Figure 3: Our resulting B^+ tree after deleting 5 from Figure 2

This was a bit more involved than in the previous example. Here’s what happened:

- We found a leaf where k resides (leaf with keys 4 and 5).
- We deleted key 5.
- This resulted in an underflow! Our leaf now has less than the d keys. We fix it by:
  - Redistribution: checking if we have a ‘rich neighbor’ we can borrow from. Indeed, if we check our right neighbor, we can borrow an entry. Therefore, we move 7 to our leaf.
  - Last step: we need to ‘copy up’ the lowest value in the leaf from which we borrowed from. In this case, this value is 8.

3. Deleting key 9 from the resultant tree (Figure 3) will result in the following:

![Diagram of B^+ tree after deleting 9 from Figure 3]

Figure 4: Our resulting B^+ tree after deleting 9 from Figure 3

This was a bit further involved than in the previous example. Here’s what happened:

- We found a leaf where k resides (leaf with keys 7, 8 and 9).
- We deleted key 9.
- This resulted in an underflow! Our leaf now has less than the d keys. We fix it by:
  - Redistribution: checking if we have a ‘rich neighbor’ we can borrow from. However, we don’t have any neighbors, so we must ‘merge’!
  - We merge this leaf with the previous one:
    * 8 is merged with the the leaf containing 4 and 7.
    * The 8 from the parent is ‘tossed’ as the page it points to doesn’t exist anymore.
    * Now, we have a new problem; tossing 8 from the parent resulted in another underflow! We fix this by doing exactly the steps we did when we removed 9:
      · Try to redistribute (doesn’t work here! The only other neighbor is the node with 15 and 20, which is ‘poor’).
      · Merge again!
2.2 Practicing $B^+$ Tree deletions

Suppose we have the following $B^+$ tree with order 2.

1. Delete the entry with key 2.

2. Delete the entry with key 1.