You will submit your solution to this assignment to the Curator System (as `HW3`). Your solution must be either a plain text file (e.g., NotePad) or a MS Word document; submissions in other formats will not be graded.

Partial credit will only be given if you show relevant work.

---

1. Consider the storage capacity of a B-tree.

   a) [20 points] What is the maximum number of data values a B-tree of order 10 can hold if its height is 4?

   **The levels would hold 1, 10, 100 and 1000 nodes, respectively, for a total of 1111 nodes.**

   **Each node would hold 9 data values, so there would be 9999 of them.**

   b) [20 points] Derive a function of m and h that expresses the maximum number of data values a B-tree of order m can hold if its height is h.

   **Level k (we number starting with level 0) would hold m^k nodes, and each node would hold m-1 data values.**

   **If we have h levels, then the total number of data values would be:**

$$
\begin{aligned}
(m-1)\sum_{k=0}^{h-1} m^k &= (m-1)(m^{h-1} + m^{h-2} + \cdots + m^1 + 1) \\
&= m^h - 1
\end{aligned}
$$

2. Suppose you have a B-tree of order m and height h.

a) [10 points] In the worst case, what is the average-case Θ-complexity of the splitting operation that may be necessary when an insertion is performed. Explain why.

**Splitting a leaf node requires creating a new node (say, Θ(1)), and copying half the data elements from the leaf into the new node, which is Θ(m), and adding one element to the parent node, which is Θ(1). So, splitting a leaf would be Θ(m) .**

**Now, splitting could propagate all the way to the root node, in which case a total of h nodes would be split, so the total cost worst-case cost would be Θ(hm).**

b) [10 points] Assuming no merging occurs, what is the maximum number of times the disk must be accessed (reads and/or writes) when a deletion operation is performed? Explain why.

**The data value to be deleted must be found. If it's in a leaf then one node from each of the h levels must be read from disk. If it's not in a leaf, then its immediate predecessor must be found, and that will be in a leaf. So, either way, a total of h nodes must be read from disk in order to find the relevant data.**

**In the first case (deleting from a leaf), that leaf node must be written back to disk. In the second case, both the leaf and the internal node must be written back. In addition, if the leaf node underflows, then one or both of its siblings must be read from disk in order to borrow a value, and the sibling that donates a value and the parent node must be re-written as well. Note that the parent can't underflow here, since the number of data values in the parent hasn't changed.**

**So the maximum number of reads/writes would be h + 6.**

c) [10 points] Repeat part b assuming that merges may be necessary.

**There's not much difference aside from the fact that merging could propagate back up the tree. You'd still have to find the target value (and perhaps its immediate predecessor) and that would mean reading h nodes from disk. If merging was needed, you'd still face the possibility of having to retrieve two siblings from disk. After the merge, the merged node created from the leaf and the sibling with which it merged would have to be written back to disk, as well as the parent node. If the parent also underflowed (we wouldn't re-write it before merging) so that further merging was necessary, then the parent's siblings would have to be read and the merged node would be re-written, as well as the parent's parent. And so forth…**

**So, the search phase would require reading h nodes from disk. Performing a merge at the each level except the root would require reading up to two siblings from disk, and then re-writing the merged node. At the root, there are no siblings to read, but the root might have to be re-written.**

**The total number of reads and writes would then be h + 2(h-1) + h or 4h-2.**

3. Suppose you have a B-tree of order 101.

a) [10 points] What is the minimum number of values the tree could store if the tree has two levels? Explain.

**The smallest possible tree would have a root node containing 1 data value, and two leaf nodes each containing the minimum number of values (50), for a total of 101 data values.**

b) [10 points] What is the minimum number of values the tree could store if the tree has three levels? Explain.

**To have three levels, the root of a two-level tree must have split, and so there must have been a full complement of nodes (101) in level 2 at that point; note that at least one of those nodes must have been full if the next insertion caused the root to split. So, just before the root split, the tree must have contained at least 100 + 100*50 + 100, or 5200, values. After the insertion causing the root to split, the tree would contain at least 5201 values.**

**Note: at this point, suppose you deleted a value from the tree. Since every leaf is at its minimum fullness after the split that created the current root, this must cause a leaf to underflow. In turn, that forces a merge operation, which removes a value from one of the nodes in level 1. But there are only two nodes in level 1 of this tree, and both would be at minimum fullness, hence they must merge, and that would trigger the removal of the current root. So, the answer above is really the minimum number of values that could be in a B-tree with three levels.**

c) [10 points] What is the maximum number of values the tree could store if the tree has two levels? Explain.

**We must have a B-tree with a full root (100 values) and 101 child nodes, each storing 100 values. So that would be a total of 10200 values.**