CS 122A: Introduction to Data Management – Fall 2018

Homework 7: Indexing (100 points)

Due Date: **Friday, November 30 (5:00 PM)**

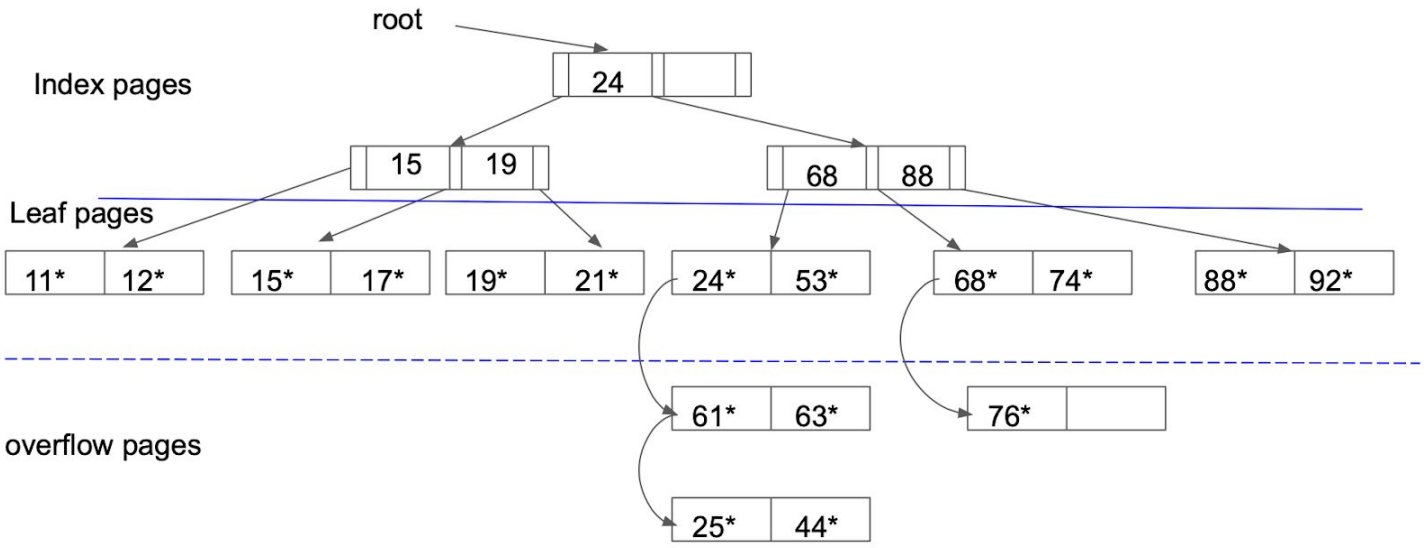
Submission

All HW assignments should contain both your student ID and your name and must be submitted online (e.g., 12345678_John_Doe.pdf) via the HW7 dropbox on **Gradescope**. The pdf file has to contain the answers to the problems. Be sure to download the HW#7 template file and use that as the basis for your submission, as it is **mandatory**.

The questions are presented here for your reference. Submit your answers using the provided template.

ISAM

1) Consider the following ISAM tree. For each part of the question, apply the action(s) asked and draw the resulting tree. **Each action happens based on the original tree.** Indicate the number of reads and writes required to apply the action(s). Note, deleting a value which leads to the deletion of a page is considered 1 combined write, not 2 separate writes.



Each Question is applied onto the original tree diagram.

1.1) Insert 22.
1.2) Insert 75.
1.3) Delete 68.
1.4) Insert 95.
1.5) Delete 18.
1.6) Delete 76.
1.7) Delete 63.
1.8) Delete 11 and then delete 12.

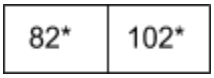
2) Look at the original table. Can you find any sequence of nodes to be inserted or deleted to make the root page full? If yes, mention the sequence, if no, explain why.

B+ Tree

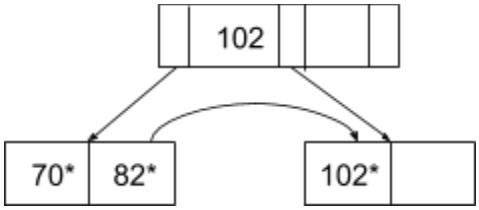
3) At this part of the project, Zootjams has made a big library of contents that needs a faster way of retrieval. CEO of Zootjams has heard that other companies are using B+ tree to increase the speed of searching of data in their systems, as a such, he has asked you to make a B+ tree and add contents' cids one by one to speed up the search of the contents by their cid for Zootjam's users. You told them that in CS122A you learned that bulk loading would be a more efficient approach for building a B+ tree, so they have asked for a proof and good reasoning.

3.1) They have given you the following cids, and asked you to build two B+ trees, one by using bulk loading and the other one by inserting elements one by one. For each B+ tree, you only need to show the final B+ tree.

Each B+ tree has order of 3 (each page can hold two key values). If you need to split a leaf page, and number of nodes are odd, you could put the middle node to the left or right in general and copy up the left-most key in the right child to the parent. **For this question though, you can only put the middle key to the left leaf page. This rule should always hold: "left child values < parent_key <= right child values". Update the parent key if necessary to make it happen.**
For example, if this is your current tree:



Insertion of 70 will only give you the following tree:(Example updated!)



Feel free to work out your solution on paper, but use a software like google slides to make your final B+ trees and put it in the given template.

Cids = [45,44,34,63,61,53,51,72,17,14,12,8,22,28]

3.2) Use current example and give a brief reasoning that why building a B+ tree using bulk load is more efficient than one by one insertion.

3.3) By counting the number of page reads, briefly explain which tree -and why- is acting more efficient in searching for a data entry with search key of 53.

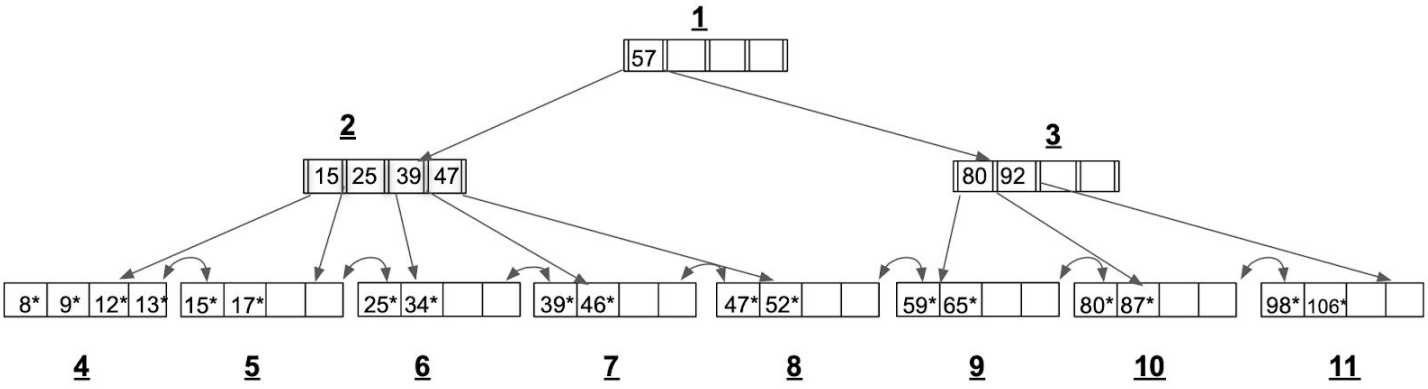
B+ tree- Insertion, Deletion, Lookup

In this section, you are asked to apply some insertion, delete and lookup actions on the following B+ tree. These are some rules that this specific B+ tree is made on, and you should follow it.

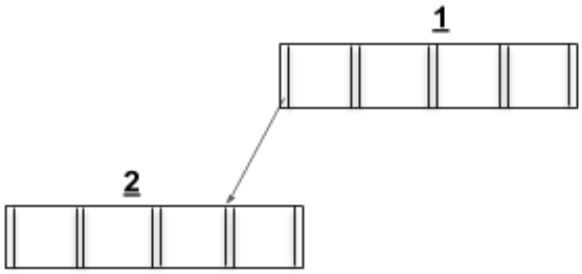
NOTE: Each tree page is labeled with a specific number for easy reference. You can draw the page number without drawing the content when referencing the page that its content has not changed.

During insertion, if a split happens, and the number of elements to be split into two pages are odd, keep floor(n/2)+1 elements to the right and floor(n/2) elements to the left. For example, if you want to split a page to two and you have 7 elements, 4 of those goes to the right page and 3 of those goes to the left page.

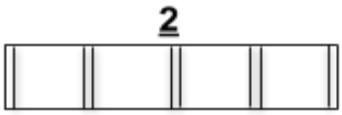
During deletion, if the number of elements in a node other than root drops to less than half, distribution is only possible by borrowing elements from the left sibling, if there is any.



4.1) Show the B+ tree that would result from inserting a data entry with key 10 into this tree (Only draw the root and left tree(page 2) downward. No need to draw pages 3, 9-11).



4.2) Show the B+ tree that would result from deleting a data entry with key 15 from the **original** tree, assuming that only the left sibling is checked for possible redistribution.



4.3) Show the B+ tree that would result from deleting a data entry with key 52 from the **original** tree.

4.4) How many pages and which pages are read for scanning the key values between 25 and 58 inclusive (25<= x <= 58)? You can assume that this is the first time the index structure is being accessed (no caching).