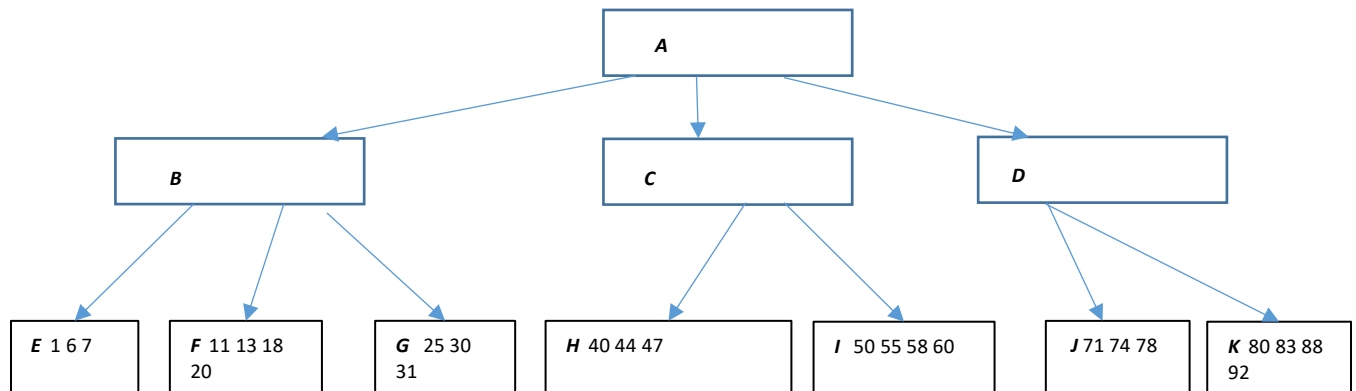


CS 5/7330
Fall 2021
Homework 1

Due: Oct 6th (Thu) 11:59pm.

1. (20 points) Consider building a B+-tree to store numbers. You are given the following constraints on the B+-tree.
 - Each leaf node can store at most 4 numbers
 - Each internal node can store at most 2 numbers (how many children maximum?)
 - Suppose an internal node stores (14, 28). The three children correspond to number with value strictly less than 14, greater than or equal 14 and strictly less than 28, and greater than or equal to 28 respectively.
 - If a node is split, and the number of items of the nodes are odd – this implies one of the split node will have one more item than the other. In such case choose the node on the “right” (containing larger numbers) to be the one who get one more item.
- a. Suppose 100 is inserted into the tree. This will cause nodeK to be split. That means a new key value need to be installed into node D. What is (in theory) the range of the value that can be used there? Explain your answer. (Notice that even though the algorithm use a certain number, but in theory there are other numbers that can be used without violating the B+-tree’s requirements).
- b. Fill in the key values of all nodes A to D, assuming we take the largest possible value for each key value. (Assume the insertion of 75 in part (a) did not happen).
- c. (Ignore part a) Suppose the following numbers are inserted (in order): 57, 4, 99, 12, 53, 5, Every time a node is split, listed the contents of all the nodes being affected. Use the following convention:
 - For any leaf nodes that has changes, list its content after the insertion
 - If a node is split, name the two new nodes by added letter A and B to it. For example, if node K is split, the two new nodes should be named KA and KB. Later if KB is split, then the two nodes should be named KBA and KBB etc.
 - If a new root is created, name the new root N. If subsequently another new root is created then name it NN etc.



For example, for the case of 57:

- 57 is inserted to node I
 - Node I is splitted. New values for IA (50, 55), KB (57, 58, 60)
 - Node C is affected. New values for keys of C is (.....) [fill it in yourself]
 - 4 is inserted to node (to be continued by you)
2. (20 points) Consider you have a Person table with SSN as the primary key, and an attribute *age* (there are other attributes, which is not important to this problem). You are also given the following:
- Each tuple of the table has size of 200 bytes
 - Each sector (page) on the disk have 1600 bytes
 - No tuple is split between two pages
 - There are 250,000 tuples in the table
 - The table is store in a single file, but is split (equally) between 2 tracks on the disk. Within the same track the sector of the file is contiguous
 - Each seek/rotation combination takes 50 milliseconds on average
 - Transferring a single page from the disk to memory take 0.5 millisecond.
 - When you first access a file, you will need to do a single seek/rotation combination
- a. How many tuples can be stored in a single page?
 - b. How many pages are needed to store the table?
 - c. Consider the following SQL query: `SELECT * from Person where SSN="123456789"`. Calculate the *best case* and *worst case* time taken to answer this query if
 - i. There is no index
 - ii. There is a clustering index on SSN (Ignore the time needed for reading the index)
 - d. Now consider the following SQL query: `SELECT * FROM Person WHERE (age > 20) and (age < 29) ORDER BY age`. Assume that there is a non-clustering index on the age

attribute. Assume that you want to use the index to answer the query by first accessing the indexing, and then read each tuple *in the order that the index provided*. (Assume there is NO buffer to store the location of the buffer so as to make reading easier – i.e. you really do have to access each tuple in the order of the age attribute). Calculate the (worst case) time taken to answer this query using the index if

- i. There are 2 tuples that satisfy this query
- ii. There are 10,000 tuples that satisfy this query

(Once again, ignore the time needed for reading the index)

Show your work in all the cases.

3. (14 points) Suppose we want to represent a ternary relationship between three entities. Assume each entity has a unique key attribute, and other attributes that need to be stored.

Now suppose you are only allowed to use embedding to store the information. Suggest what can be a problem. Use an example to illustrate.

Is there any restrictions on the relationship that will make this at least a feasible way of representing the relationship?