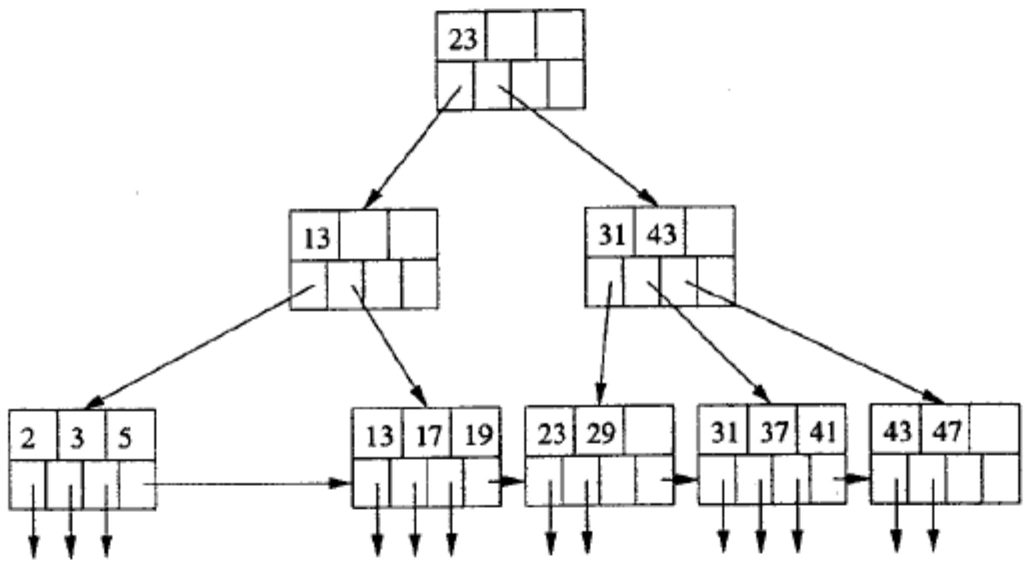
**CS540 - Practice problems on indexing and query processing**

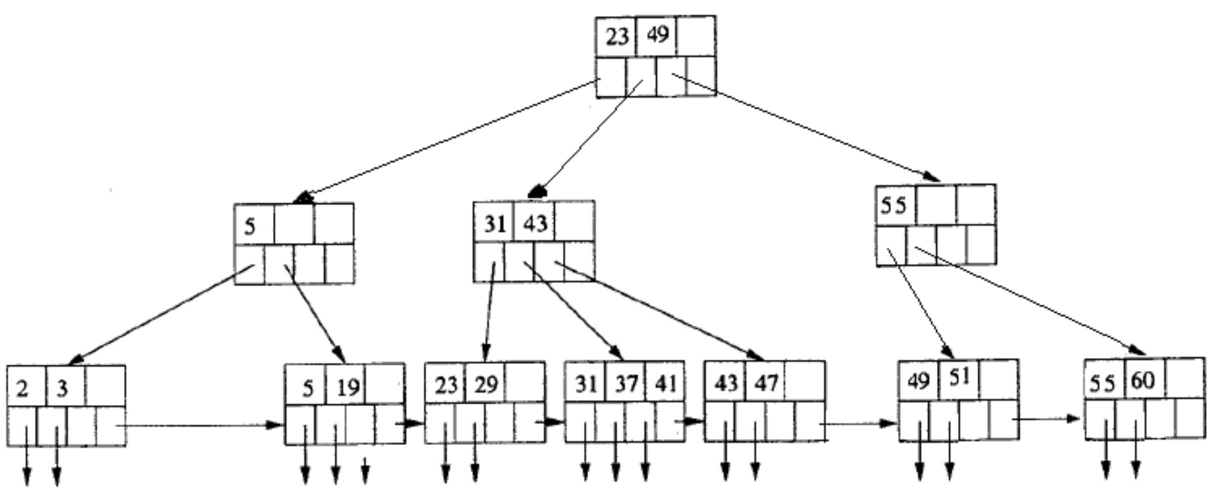
**1-** Consider the B+ tree shown below.

a) Delete keys 13 and 17 from this B+ tree.
b) Insert keys 49, 51, 55 and 60 into this B+ tree.

You only need to show the final picture of the B+ tree.




Solution:



**2-** Consider the following relational schema. Primary keys are underlined and foreign keys are in italics.

Student (<u>Student_Id</u>, Name)

Health_Insurance (<u>Insurance_Id</u>, Date_Started, *Student_Id*)

Assume that number of blocks in relation student is 10,000 and the number of blocks in relation Health_Insurance is 300. We do not have any indexes on these relations. We like to join these relations on Student_ID.

a) If we can have 12,000 blocks in main memory, i.e., M = 20,000, explain what is the fastest join algorithm to join "Student" and "Health_Insurance" on Student_Id and analyze its cost?

Solution: Because we can fit both relations in main memory, we may use an internal memory join algorithms to join them. We may pick the internal memory version of nested-loop, sort-merge, or hash-join as they involve equal number of I/O accesses. The cost of join will be **B (Health_Insurance) + B (Student)** disk I/O's. Note that we do not care about the cost of memory operations when analyzing the cost of query processing algorithms.

b) If we can fit 20 blocks in main memory, M= 20, what is the fastest join algorithm to join these relations? Analyze its cost.

Solution: Hash-join and optimized sort-merge join are the fastest algorithms in the given setting and have equal costs. However, to apply optimized sort-merge join, we should have **B (Health_Insurance) + B (Student)** <= $M^2$ . As **B (Health_Insurance) + B (Student)** is not less than or equal to 400, we cannot use optimized sort-merge algorithm. Hash-join algorithm, however, requires smaller amount of main memory. Because the smaller relation is **Health_Insurance,** we need to have **B(Health_Insurance) <= $M^2$** or **B(Health_Insurance) <= 400** to perform this join using hash-join algorithm. As we have **B(Health_Insurance) <= 400 ,** we can use hash-join algorithm for the join. The cost of hash-join for this join is **3 B(Health_Insurance) + 3 B (Student) = 30900** number of **I**/O access**.**

**3-** Assume that we like to sort a relation R. Answer the following questions.

a) If R is too large to fit in the main memory and does not have any index, explain which algorithm we can use to sort R and analyze it cost for the sort.

Solution:

We may use two-pass, multiway merge-sort algorithm. The cost of sort is 3B(R).

b) Assuming we can fit 200 blocks in main memory, M= 200, what should be the size of relation R (in blocks) to use the algorithm of part a?

Solution:

A relation of B blocks can be sorted using the algorithms of part a, as long as B is no more than $M^2$. Hence, if the size of R is less than or equal to 40,000, we can use two-phase, multiway merge-sort algorithm to sort it.