Exercises

I. **Buffer replacement policy**
   Show on an example how MRU can be used to prevent sequential flooding

II. **B-tree**
   First, build B-tree from records with sorted keys:
   2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47
   Insert key 1
   Insert keys 14, 15, 16
   Delete key 23
   Delete all keys >23 (in turn)

III. **Extendible hashing**
    Start with hash table with 2 directory slots. Each block can hold up to 3 key-data entries
    Show each step of inserting values from 0000 to 1111.

IV. **Linear hashing**
    Split threshold is 100%, initial number of buckets = 3. Insert keys 1 – 15. Show state after each insertion.

V. **2PMMS**
   Show each step of the 2PMMS algorithm for sorting
   a, f, c, g, b, b, d, m, g, a, c, f, e, a, b, n, a, f, c, g, b, b, d, m.
   Each block can hold maximum 2 records. Total number of blocks that can be held in RAM is 3.

VI. **Index selection**
   Using the relation Sales (day, store, item, color, size), write the following queries in SQL:
   List all colors of shirts and their total sales
   List sales of shirts by store and color
   List sales of all items by store and color
   For each query indicate which indexes (if any) will help to retrieve tuples more efficiently

Exercise 1. Consider the join \( R \bowtie_{R.A=S.B} S \) and the following information on \( R \) and \( S \):

- Relation \( R \) contains 10000 tuples and has 10 tuples per block.
- Relation \( S \) contains 2000 tuples and has 10 tuples per block.
- Attribute \( B \) of relation \( S \) is the primary key for \( S \).
- Neither \( R \) nor \( S \) is sorted on the join attribute.
- Neither relation has any indexes built on it.
- There are 51 main memory buffers available.

a. What is the cost (in disk I/O's) of joining \( R \) and \( S \) using the simple nested loop join? What is the minimum number of buffer pages required for this cost to remain unchanged?

b. What is the cost (in disk I/O's) of joining \( R \) and \( S \) using the block nested loop join? What is the minimum number of buffer pages required for this cost to remain unchanged?

c. What is the cost (in disk I/O's) of joining \( R \) and \( S \) using a sort-merge join? What is the minimum number of buffer pages required for this cost to remain unchanged?

d. What is the cost (in disk I/O's) of joining \( R \) and \( S \) using a hash join? What is the minimum number of buffer pages required for this cost to remain unchanged?

e. What join algorithm yields the least cost if you were free to choose the number of free buffers? Briefly motivate your answer and give the exact optimal cost.

f. How many tuples does the join of \( R \) and \( S \) produce, at most, and how many blocks are required to store the result of the join back on disk?

VIII. Index nested loop joins

Exercise 1. Consider again the join of \( R \) and \( S \) from the previous exercise. Assume that unclustered BTree indexes exist on \( R.A \) and \( S.B \) and that there are 5 buffers available. Number of distinct values in column \( A \) of \( R \) is 1000.

a. Is an index nested loop join using either one of these indexes a cheaper alternative for performing the join than a block-based nested loop join? Explain.

b. Would your answer change if 51 buffers are available?

c. Would your answer change if \( S \) contained only 10 tuples instead of 2000 tuples (and the number of available buffers is 5 as before)?

Exercise 2. Next assume that instead clustered BTree indexes exist on \( R.A \) and \( S.B \). Again, 11 buffers are available. Number of distinct values in column \( A \) of \( R \) is 1000.
Is an index nested loop join using either one of these indexes a cheaper alternative for performing the join than a block-based nested loop join? Explain.