100 points (8.75% of course grade)
Assigned: Part 1, 2: Monday, March 06; Part 3, 4, 5: Wednesday, March 22
Due:

Part 1 and 2 : Wednesday, March 22

Part 3, 4, 5: Wednesday, March 29, 11:55 pm EST

This homework should be done in parts as soon as relevant topics are covered in lectures and the corresponding parts are posted. If you wait until the last minute, you might be overwhelmed.

For some problems, you will need to use Gradiance. Access Gradiance via the "Gradiance" link on the course website. There is no need to turn in anything else for these problems; your scores will be tracked automatically.

For other problems, you will need to turn in the required files electronically **on sakai**. Please make sure that ***all*** required files have been uploaded in each resubmission if we submit multiple times.

The non-Gradiance problems in this homework does not require the course VM. You may prepare your answers electronically or on paper (handwritten). In the latter case, scan or photograph the pages and submit the resulting PDF (preferred) or JPG files. Please name your files informatively, e.g., as *n*-<*firstname*>-<*lastname*>.pdf, where *n* is the problem number, and <firstname> and <lastname> are your first and last names.

Problem 1 (8 points)
Complete the Gradiance homework titled "Homework 3.1 (Physical Organization)."

Problem 2 (25 points)
Complete the Gradiance homework titled "Homework 3.2 (Index)."

Problem 3 (10 points)
Complete the Gradiance homework titled "Homework 3.3 (Query Processing)."

Problem 4 (20 points) -- INDEX

Consider a relation R(a, b, c, d, e) containing 5,000,000 records, where each block (unit of I/O) of the relation holds 10 records. R is organized as a sorted file with secondary indexes. Assume that R.a is a candidate key for R, with values lying in the range 0 to 4,999,999, and that R is sorted in R.a order.

For each of the following relational algebra queries, state which of the following three approaches is "most likely to be" the cheapest (an intuitive answer with explanations will suffice, the exact numbers for cost are not needed):

Option A: Access the sorted file for R directly (table scan, but can use sorted order).
Option B: Use a clustered B+ tree index on attribute R.a. Assume accessing the data entries at the leaves requires 2 I/Os.
Option C: Use a hash-based index on attribute R.a. Assume the entire hash-based index structure is in main memory.

Note: $\log_2 500{,}000 = 19$.

Here are the queries:
1. $\sigma_{a<50{,}000}(R)$.
2. $\sigma_{a=50{,}000}(R)$
3. $\sigma_{a > 50{,}000 \text{ and } a<50{,}010}(R)$
4. $\sigma_{a \neq 50{,}000}(R)$

Problem 5 (37 points) -- Join Algorithms and External Sorting

Consider the join $R \bowtie_{R.a=S.b} S$, given the following information about the relations to be joined.

- The cost metric is the number of block I/Os unless otherwise noted, and the cost of writing out the final result should be uniformly ignored, unless explicitly asked. Relation R contains 10,000 tuples and has 10 tuples per block.
- Relation S contains 2,000 tuples and also has 10 tuples per block.
- Attribute b of relation S is the primary key for S.
- M = 52 memory blocks are available.
- There are B+tree indexes on R and S. Assume that it takes 3 I/Os to access a leaf in R, and 2 I/Os to access a leaf in S.
- Assume uniform distribution - i.e. (approximately) the same number of tuples of R joins with a tuple in S.

Q5a. (3 + 1 = 4 pts)
(i) What is the (minimum) cost of joining R and S using a block-based nested loops join?
(ii) What is the minimum number of memory blocks required for this algorithm?

Q5b. (3 + 1 = 4 pts)
(i) What is the (minimum) cost of joining R and S using the improved block-based nested loops join (see Lecture 16, slide 10), where you utilize all available memory blocks.
(ii) What is the minimum number of memory blocks required for this cost to remain unchanged (i.e. can you make the no. of memory blocks smaller than 52)?

Q5c. (3 pts)
Compute the total cost of sorting R using external merge sort. INCLUDE the cost of final write to disk for relation R. Show the cost (how many runs with how many blocks each, cost of reading/writing them) for all passes pass 0, 1, ...)

Q5d. (3 pts)
Compute the cost of sort-merge join of R and S using the cost you calculated in Q5c.
You have to do the same for S. DO NOT include the cost for the final write.

Q5e. (1 + 3 = 4 pts)
Suppose the B+ index on S.b is unclustered, and you are allowed to use only this index on S in an index-nested loop join (i.e. assume you cannot use the index on R).
(i) Which one will be the inner relation?
(ii) What will be the cost?

Q5f. (1 + 3 = 4 pts)
Suppose the B+ index on R.a is unclustered, and you are allowed to use only this index on R in an index-nested loop join (i.e. assume you cannot use the index on S).
(i) Which one will be the inner relation?
(ii) What will be the cost?
.

Q5g. (3 pts)
Suppose the B+ index on S.b is clustered, and you are allowed to use only this index on S in an index-nested loop join (i.e. assume you cannot use the index on R).
What will be the cost?

Q5h. (3 pts)
Suppose the B+ index on R.a is clustered, and you are allowed to use only this index on R in an index-nested loop join (i.e. assume you cannot use the index on S).
What will be the cost?

Q5i. (3 + 2 = 5 pts)
(i) What is the cost of joining R and S using a hash join?
(ii) What is the minimum number of memory blocks required for this cost to remain unchanged (i.e. can you reduce the number of memory blocks from 52)? An approximate answer with analysis will suffice.
Assume uniform distribution for hash functions (i.e. each partition of hash gets about the same number of blocks).

Q5j. (2 + 2 = 4 pts)
What would be the lowest possible I/O cost for joining R and S using any join algorithm (not including the final write), and how many memory blocks (M) would be needed to achieve this cost? Explain briefly.