CARNEGIE MELLON UNIVERSITY
COMPUTER SCIENCE DEPARTMENT
15-445/645 – DATABASE SYSTEMS (FALL 2024)
PROF. ANDY PAVLO

Homework #4 (by William)
Due: **Sunday, Nov 3, 2024 @ 11:59pm**

IMPORTANT:

- Enter all of your answers into **Gradescope by 11:59pm on Sunday, Nov 3, 2024**.
- **Plagiarism**: Homework may be discussed with other students, but all homework is to be completed **individually**.

For your information:

- Graded out of **100** points; **3** questions total
- Rough time estimate: ≈ 2 - 3 hours (0.5 - 1 hours for each question)

Revision : 2024/10/23 08:25

Question	Points	Score
Sorting Algorithms	32	
Join Algorithms	44	
Query Execution, Planning, and Optimization	24	
Total:	100	

Question 1: Sorting Algorithms .... [32 points]

We have a database file with 6 million pages (N = 6,000,000 pages), and we want to sort it using external merge sort. Assume that the DBMS is not using double buffering or blocked I/O, and that it uses quicksort for in-memory sorting. Let B denote the number of buffers.

(a) **[4 points]** Assume that the DBMS has 80 buffers. How many sorted runs are generated? Note that the final sorted file does not count towards the sorted run count.
☐ 75949 ☐ 75950 ☐ 75961 ☐ 75962 ☐ 75963 ☐ 75964

(b) **[4 points]** Again, assuming that the DBMS has 80 buffers. How many passes does the DBMS need to perform in order to sort the file?
☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

(c) **[4 points]** Again, assuming that the DBMS has 80 buffers. How many pages does each sorted run have after the second pass (i.e. Note: this is Pass #1 if you start counting from Pass #0)?
☐ 79 ☐ 80 ☐ 81 ☐ 6320 ☐ 6400 ☐ 6480

(d) **[4 points]** Again, assuming that the DBMS has 80 buffers. What is the total I/O cost to sort the file?
☐ 6,000,000 ☐ 12,000,000 ☐ 24,000,000 ☐ 48,000,000 ☐ 96,000,000

(e) **[4 points]** What is the smallest number of buffers B such that the DBMS can sort the target file using only three passes?
☐ 181 ☐ 182 ☐ 183 ☐ 184 ☐ 185

(f) **[4 points]** Suppose the DBMS has 107 buffers. What is the largest database file (expressed in terms of the number of pages) that can be sorted with external merge sort using three passes?
☐ 11,342 ☐ 11,449 ☐ 1,202,252 ☐ 1,213,594 ☐ 1,225,043

(g) For each of the following statements about sorting, pick True or False.

i. **[4 points]** The DBMS receives a query that requires sorting. Assume that the sort order is a prefix of the index key. Under which scenario(s) will using an unclustered B+Tree index have comparable performance to a clustered B+Tree index:
☐ The sort order exactly matches the index key.
☐ Query contains a LIMIT 1, and the first tuple answers the query.
☐ All attributes accessed by the query are contained in the index.
☐ Using unclustered index will always perform worse than using clustered index.
☐ None of the above.

ii. **[2 points]** The DBMS can implement all aggregates discussed in lecture (i.e., "DISTINCT", "GROUP BY") with a sorting aggregate.
☐ True ☐ False

iii. **[2 points]** The DBMS should always use a hashing aggregate.
☐ True ☐ False

Question 2: Join Algorithms .... [44 points]

Consider relations `X(a, b)`, `Y(a, c, d)`, and `Z(a, e, f)` to be joined on the common attribute `a`. Assume that there are no indexes available on the tables to speed up the join algorithms.

- There are B = 245 pages in the buffer
- Table `X` spans M = 1,950 pages with 300 tuples per page
- Table `Y` spans N = 150 pages with 600 tuples per page
- Table `Z` spans O = 1,200 pages with 180 tuples per page
- The join result of `Y` and `Z` spans P = 90 pages

For the following questions, assume a simple cost model where pages are read and written one at a time. Also assume that one buffer block is needed for the evolving output block and one input block is needed for the current input block of the inner relation. You may ignore the cost of the writing of the final results.

(a) **[2 points]** What is the I/O cost of a simple nested loop join with `Y` as the outer relation and `X` as the inner relation?

☐ 292,650
☐ 585,150
☐ 1,170,150
☐ 54,000,150
☐ 87,750,150
☐ 175,500,150

(b) **[2 points]** What is the I/O cost of a block nested loop join with `Y` as the outer relation and `Z` as the inner relation?

☐ 750
☐ 1,200
☐ 1,350
☐ 1,800
☐ 1,950
☐ 2,100
☐ 2,550

(c) **[2 points]** What is the I/O cost of a block nested loop join with `Z` as the outer relation and `Y` as the inner relation?

☐ 750
☐ 1,200
☐ 1,350
☐ 1,800
☐ 1,950

☐ 2,100
☐ 2,550

(d) For a sort-merge join with Z as the outer relation and X as the inner relation:

i. **[3 points]** What is the cost of sorting the tuples in X on attribute a?
☐ 1,950
☐ 3,900
☐ 4,800
☐ 7,800
☐ 15,600

ii. **[3 points]** What is the cost of sorting the tuples in Z on attribute a?
☐ 2,400
☐ 4,800
☐ 7,200
☐ 9,600
☐ 14,400

iii. **[3 points]** What is the cost of the merge phase in the worst-case scenario?
☐ 1,200
☐ 1,950
☐ 3,150
☐ 180,000
☐ 292,500
☐ 2,046,000
☐ 2,340,000
☐ 2,634,000

iv. **[3 points]** What is the cost of the merge phase assuming there are no duplicates in the join attribute?
☐ 1,200
☐ 1,950
☐ 3,150
☐ 180,000
☐ 292,500
☐ 2,046,000
☐ 2,340,000
☐ 2,634,000

v. **[3 points]** Now consider joining Y, Z and then joining the result with X. What is the cost of the final merge phase assuming there are no duplicates in the join attribute?
☐ 240
☐ 1,290
☐ 2,000

☐ 2,040
☐ 13,500
☐ 108,000
☐ 175,500

(e) **[2 points]** Consider a hash join with `Y` as the outer relation and `X` as the inner relation. You may ignore recursive partitioning and partially filled blocks. What is the cost of the combined probe and partition phases?
☐ 2,100
☐ 2,700
☐ 4,050
☐ 4,200
☐ 6,300
☐ 9,450

(f) **[3 points]** Assume that the tables do not fit in main memory and that a large number of distinct values hash to the same bucket using hash function h_1. Which of the following approaches works the best?
☐ Create two hashtables half the size of the original one, run the same hash join algorithm on the tables, and then merge the hashtables together.
☐ Create hashtables for the inner and outer relation using h_1 and rehash into an embedded hash table using $h_2 \mathrel{!}= h_1$ for large buckets.
☐ Use linear probing for collisions and page in and out parts of the hashtable needed at a given time.
☐ Create hashtables for the inner and outer relation using h_1 and rehash into an embedded hash table using h_1 for large buckets.

(g) For each of the following statements about joins, pick True or False.

i. **[2 points]** If both tables in a simple nested loop join fit entirely in memory, the order of inner and outer tables does not significantly affect I/O costs.
☐ True ☐ False

ii. **[2 points]** If neither table fits entirely in memory, I/O costs would be lower if we process both tables on a per-block basis rather than per-tuple basis.
☐ True ☐ False

iii. **[3 points]** A sort-merge join is faster than a hash join on all circumstances.
☐ True ☐ False

iv. **[3 points]** An index nested loop join requires an index on the outer- and inner- tables.
☐ True ☐ False

v. **[3 points]** For a hash join to work, the inner table (or its partitions) need to fit into memory.
☐ True ☐ False

vi. **[5 points]** A nested loop join can output a sorted stream of tuples under the following condition:
☐ All nested loop joins can output a sorted stream.
☐ No intra-operator parallelism.
☐ Outer- table (or data) is already sorted.
☐ Outer- table (or data) is sorted and no intra-operator parallelism.
☐ Inner- table (or data) is already sorted.
☐ Inner- table (or data) is sorted and no intra-operator parallelism.

Question 3: Query Execution, Planning, and Optimization . . . . . [24 points]

(a) **[3 points]** Assume that the DBMS zone maps are up to date. The DBMS can use these zone maps to answer specific queries without reading any actual table heap tuples:
☐ True ☐ False

(b) **[3 points]** Assuming a query with multiple `OR` predicates. Using a multi-index scan will always perform better than a sequential scan.
☐ True ☐ False

(c) **[3 points]** For OLAP queries, which often involve complex operations on vast datasets, intra-query parallelism is typically not preferred to optimize performance.
☐ True ☐ False

(d) **[3 points]** The process per DBMS worker approach provides better fault isolation and scheduling control than the thread per DBMS worker approach.
☐ True ☐ False

(e) **[3 points]** In OLAP workload, the vectorized model's performance improvements come mainly from the reduction in the number of disk I/O operations.
☐ True ☐ False

(f) **[3 points]** The query optimizer in a database management system always guarantees the generation of an optimal execution plan by exhaustively evaluating all possible plans to ensure the lowest cost for query execution.
☐ True ☐ False

(g) **[3 points]** Predicate and projection pushdown will always improve query performance.
☐ True ☐ False

(h) **[3 points]** Sampling statistics requires evaluating each tuple in the entire table.
☐ True ☐ False