CARNEGIE MELLON UNIVERSITY
COMPUTER SCIENCE DEPARTMENT
15-445/645 – DATABASE SYSTEMS (FALL 2022)
PROF. ANDY PAVLO

Homework #3 (by Wan Shen Lim) – Solutions
Due: **Sunday Oct 9, 2022 @ 11:59pm**

IMPORTANT:

- Enter all of your answers into **Gradescope by 11:59pm on Sunday Oct 9, 2022**.
- **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: ≈ 1 - 2 hours (0.5 - 1 hours for each question)

Revision : 2022/10/10 16:03

Question	Points	Score
Sorting Algorithms	35	
Join Algorithms	35	
Query Execution	30	
Total:	100	

Question 1: Sorting Algorithms .... [35 points]

Graded by:

We have a database file with fourteen million pages (N = 14,000,000 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) **[8 points]** Assume that the DBMS has eight buffers. How many passes does the DBMS need to perform in order to sort the file?

□ 8 ■ **9** □ 10 □ 11 □ 12

Solution:

$$\begin{aligned} 1 + \left\lceil \log_{B-1}\left(\left\lceil \frac{N}{B} \right\rceil\right)\right\rceil &= 1 + \lceil \log_7 (\lceil 14,000,000/8 \rceil) \rceil \\ &= 1 + \lceil \log_7 (\lceil 1,750,000 \rceil) \rceil \\ &= 1 + 8 = 9 \end{aligned}$$

(b) **[9 points]** Again, assuming that the DBMS has eight buffers. What is the total I/O cost to sort the file?

□ 224,000,000 ■ **252,000,000** □ 280,000,000 □ 308,000,000 □ 336,000,000

Solution: $Cost = 2N \times \#passes = 2 \times 14,000,000 \times 9$

(c) **[9 points]** What is the smallest number of buffers B that the DBMS can sort the target file using only eight passes?

□ 7 □ 8 ■ **9** □ 2,450 □ 2,451 □ 2,452 □ 3,742 □ 3,743

Solution: We want B where $N \leq B \times (B-1)^7$. If $B = 9$, then $14,000,000 \leq 9 \times 8^7 = 18,874,368$; any smaller value for B would fail.

(d) **[9 points]** Suppose the DBMS has forty-two buffers. What is the largest database file (expressed in terms of N, the number of pages) that can be sorted with external merge sort using four passes?

■ **2,894,682** □ 3,111,696 □ 3,185,784 □ 118,681,962 □ 130,691,232
□ 133,802,928 □ 4,865,960,442 □ 5,489,031,744 □ 5,619,722,976

Solution: We want N such that $N \leq B \times (B-1)^3$. The largest such value is $B \times (B-1)^3$ itself, which is $42 \times 41^3 = 2,894,682$

Question 2: Join Algorithms .... [35 points]

Graded by:

Consider relations `R(a, b, c)`, `S(a, d)`, and `T(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 = 445 pages in the buffer
- Table `R` spans M = 1,500 pages with 80 tuples per page
- Table `S` spans N = 4,500 pages with 150 tuples per page
- Table `T` spans O = 200 pages with 250 tuples per page

Answer the following questions on computing the I/O costs for the joins. You can assume the simplest cost model where pages are read and written one at a time. You can also assume that you will need one buffer block to hold the evolving output block and one input block to hold the current input block of the inner relation. You may ignore the cost of the writing of the final results.

(a) **[4 points]** Block nested loop join with `S` as the outer relation and `R` as the inner relation:
☐ 17,500 ☐ 18,000 ☐ 18,500 ☐ 19,000 ☐ 19,500 ☐ 20,500 ■ **21,000**

Solution: $N + \lceil \frac{N}{B-2} \rceil \times M = 4,500 + \lceil \frac{4,500}{443} \rceil \times 1,500 = 4,500 + 16,500 = 21,000$

(b) **[4 points]** Block nested loop join with `R` as the outer relation and `S` as the inner relation:
☐ 17,500 ☐ 18,000 ☐ 18,500 ☐ 19,000 ■ **19,500** ☐ 20,500 ☐ 21,000

Solution: $M + \lceil \frac{M}{B-2} \rceil \times N = 1,500 + \lceil \frac{1,500}{443} \rceil \times 4,500 = 1,500 + 18,000 = 19,500$

(c) Sort-merge join with `S` as the outer relation and `R` as the inner relation:

i. **[3 points]** What is the cost of sorting the tuples in `R` on attribute `a`?
☐ 3,000 ■ **6,000** ☐ 9,000 ☐ 12,000 ☐ 15,000 ☐ 18,000

Solution: $passes = 1 + \lceil \log_{B-1}(\lceil \frac{M}{B} \rceil) \rceil = 1 + \lceil \log_{444}(\lceil \frac{1,500}{445} \rceil) \rceil = 1 + 1 = 2$
$2M \times passes = 2 * 1,500 * 2 = 6,000$

ii. **[3 points]** What is the cost of sorting the tuples in `S` on attribute `a`?
☐ 3,000 ☐ 6,000 ☐ 9,000 ☐ 12,000 ☐ 15,000 ■ **18,000**

Solution: $passes = 1 + \lceil \log_{B-1}(\lceil \frac{N}{B} \rceil) \rceil = 1 + \lceil \log_{444}(\lceil \frac{4,500}{445} \rceil) \rceil = 1 + 1 = 2$
$2N \times passes = 2 * 4,500 * 2 = 18,000$

iii. **[3 points]** What is the cost of the merge phase in the worst-case scenario?
☐ 1,500 ☐ 3,000 ☐ 4,500 ☐ 6,000 ☐ 12,000 ☐ 2,250,000
■ **6,750,000** ☐ 20,250,000 ☐ 1,080,000

Solution: $M \times N = 1,500 \times 4,500 = 6,750,000$

iv. **[3 points]** What is the cost of the merge phase assuming there are no duplicates in the join attribute?
☐ 1,500 ☐ 3,000 ☐ 4,500 ■ **6,000** ☐ 12,000 ☐ 2,250,000
☐ 6,750,000 ☐ 20,250,000

Solution: $M + N = 1,500 + 4,500 = 6,000$

v. **[3 points]** Now consider joining R, S and then joining the result with T. Suppose the cost of the final merge phase is 800 and assume that there are no duplicates in the join attribute. How many pages did the join result of R and S span?
☐ 2 ☐ 4 ☐ 6 ☐ 8 ☐ 200 ☐ 400 ■ **600** ☐ 800

Solution: Let J be the number of pages that the join result of R and S span. We joined those J pages with T's O pages, assuming no duplicates in the join attribute, and are told that this merge phase cost 800. In other words, $J+O = J+200 = 800$, i.e., $J = 600$.

(d) Hash join with S as the outer relation and R as the inner relation. You may ignore recursive partitioning and partially filled blocks.

i. **[4 points]** What is the cost of the probe phase?
☐ 1,500 ☐ 3,000 ☐ 4,500 ■ **6,000** ☐ 9,000 ☐ 12,000 ☐ 18,050

Solution: $(M + N) = (1,500 + 4,500) = 6,000$

ii. **[4 points]** What is the cost of the partition phase?
☐ 1,500 ☐ 3,000 ☐ 4,500 ☐ 6,000 ☐ 9,000 ■ **12,000** ☐ 18,050

Solution: $2 \times (M + N) = 2 \times (1,500 + 4,500) = 2 \times 6,000 = 12,000$

(e) **[4 points]** Assume that the tables do not fit in main memory and that a high cardinality of distinct values hash to the same bucket using your hash function h_1. Which of the following approaches works the best?

■ **Create hashtables for the inner and outer relation using h_1 and rehash into an embedded hash table using h_2 != h_1 for large buckets**
☐ Create hashtables for the inner and outer relation using h_1 and rehash into an embedded hash table using 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 2 hashtables half the size of the original one, run the same hash join algorithm on the tables, and then merge the hashtables together

Solution: Use Grace hash join with recursive partitioning, which is what the correct option describes.

Question 3: Query Execution .... [30 points]

Graded by:

(a) **[6 points]** Which processing model has on average the smallest working buffer per operator invocation? Ignore optimizations like projection pushdown. Select only one answer.
■ **Iterator** □ Materialization □ Vectorization

Solution: The iterator model processes a single tuple at a time. The materialization model processes its input all at once and emits its output all at once. The vectorization model emits batches of tuples. A single tuple is smaller than multiple tuples.

(b) **[6 points]** In the *iterator* processing model, the logic of an operator is independent of its children and parents. (i.e., the code does not case on what type of iterator the children or parents are)
■ **True** □ False

Solution: Iterators receive tuples as input and emit tuples as output. Consequently, iterators compose cleanly. Any iterator can be used as input or output to any other iterator, without writing custom code for each possible iterator configuration.

(c) **[6 points]** In the *vectorized* processing model, each operator that receives input from multiple children **requires** multi-threaded execution to generate the *Next()* output tuples from each child.
□ True ■ **False**

Solution: Iterators couple dataflow with control flow. When *Next()* returns a batch of tuples in the dataflow graph, control is returned too. Subsequently, the entire query plan can be executed with only one thread.

(d) **[6 points]** The *iterator* processing model often leads to good code locality (in the instruction cache sense).
□ True ■ **False**

Solution: Iterators are typically implemented in a generic way. For example, processing a tuple may involve calling another function to first interpret the tuple's contents, and the *Next()* function itself is usually virtual or invoked by a function pointer. This results in frequent long jumps, which leads to poor code locality.

(e) **[6 points]** An index scan is always better (fewer I/O operations, faster run-time) than a sequential scan, regardless of the processing model.
□ True ■ **False**

Solution: An index scan may require multiple I/O operations (look up the row in the index, look up the row in the heap) whereas a sequential scan will just go through the heap. Moreover, a sequential scan may benefit more from pre-fetching pages. PostgreSQL's optimizer defaults to picking sequential scans over index scans if it thinks that you're asking for more than (very approximately) 10% of all rows in the table.