CARNEGIE MELLON UNIVERSITY
COMPUTER SCIENCE DEPARTMENT
15-445/645 – DATABASE SYSTEMS (SPRING 2023)
PROF. CHARLIE GARROD

Homework #3 (by Christopher Lim) – Solutions

Due: **Sunday, Feb 26, 2023 @ 11:59pm**

IMPORTANT:

- Enter all of your answers into **Gradescope by 11:59pm on Sunday, Feb 26, 2023**.
- **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: $\approx$ 1 - 2 hours (0.5 - 1 hours for each question)

Revision : 2023/02/27 09:52

Question	Points	Score
Sorting Algorithms	36	
Join Algorithms	43	
Bloom Filter	21	
Total:	100	

Question 1: Sorting Algorithms . . . . . [36 points]

Graded by:

We have a database file with eight million pages (N = 8,000,000 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) **[6 points]** Assume that the DBMS has 200 buffers. How many sorted runs are generated? Note that the final sorted file does not count towards the sorted run count.

☐ 40200 ☐ 40201 ☐ 40202 ☐ 40203 ■ **40204**

Solution:

$$\left\lceil \frac{8,000,000}{200} \right\rceil + \left\lceil \frac{40,000}{199} \right\rceil + \left\lceil \frac{202}{199} \right\rceil = 40204$$

(b) **[6 points]** Again, assuming that the DBMS has 200 buffers. How many passes does the DBMS need to perform in order to sort the file?

☐ 1 ☐ 2 ☐ 3 ■ **4** ☐ 5

Solution:

$$1 + \left\lceil \log_{B-1} \left(\left\lceil \frac{N}{B} \right\rceil \right) \right\rceil = 1 + \lceil \log_{199} (\lceil 8,000,000/200 \rceil) \rceil$$
$$= 1 + 3 = 4$$

(c) **[6 points]** Again, assuming that the DBMS has 200 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)?

☐ 199 ☐ 200 ☐ 201 ☐ 39601 ■ **39800** ☐ 40000 ☐ 40200 ☐ 40401

Solution: On the first pass, B buffer pages will be used to create the sorted runs. From the second pass onward, B-1 runs will be sorted through a K-way merge.

First pass: 200 pages for each sorted run. Second pass: $200 * 199 = 39800$ pages for each sorted run

(d) **[6 points]** Again, assuming that the DBMS has 200 buffers. What is the total I/O cost to sort the file?

☐ 8,000,000 ☐ 16,000,000 ☐ 32,000,000 ■ **64,000,000** ☐ 128,000,000

Solution: $Cost = 2N \times \#passes = 2 \times 8,000,000 \times 4 = 64,000,000$

(e) **[6 points]** What is the smallest number of buffers B such that the DBMS can sort the target file using only six passes?

☐ 8 ☐ 9 ☐ 10 ☐ 14 ■ **15** ☐ 16 ☐ 24 ☐ 25

Solution: We want the smallest integer B such that $N \leq B \times (B-1)^5$. If $B = 15$, then $8,000,000 \leq 15 \times 14^5 = 8,067,360$; any smaller value for B would fail.

(f) **[6 points]** Suppose the DBMS has 445 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?

☐ 2,894,682 ☐ 3,111,696 ☐ 3,185,784 ☐ 7,920,200 ☐ 8,000,000
☐ 8,040,000 ■ **87,725,520** ☐ 88,121,125 ☐ 5,489,031,744 ☐ 5,619,722,976

Solution: We want the largest integer N such that $N \leq B \times (B-1)^2$. The largest such value is $B \times (B-1)^2$ itself, which is $445 \times 444^2 = 87,725,520$

Question 2: Join Algorithms ..... [43 points]

Graded by:

Consider relations `X(a, b)`, `Y(a, c)`, and `Z(a, d, e)` 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 = 1,000 pages in the buffer
- Table `X` spans M = 2,000 pages with 40 tuples per page
- Table `Y` spans N = 500 pages with 250 tuples per page
- Table `Z` spans O = 200 pages with 200 tuples per page

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

☐ 2,500 ☐ 3,000 ■ **3,500** ☐ 4,000 ☐ 4,500 ☐ 5,000 ☐ 5,500

Solution: $N + \lceil \frac{N}{B-2} \rceil \times M = 2,000 + \lceil \frac{2,000}{998} \rceil \times 500 = 2,000 + 1,500 = 3,500$

(b) **[3 points]** What is the I/O cost of a block nested loop join with `Y` as the outer relation and `X` as the inner relation?

■ **2,500** ☐ 3,000 ☐ 3,500 ☐ 4,000 ☐ 4,500 ☐ 5,000 ☐ 5,500

Solution: $M + \lceil \frac{M}{B-2} \rceil \times N = 500 + \lceil \frac{500}{998} \rceil \times 2,000 = 500 + 2,000 = 2,500$

(c) **[3 points]** What is the I/O cost of a simple nested loop join with `Z` as the outer relation and `Y` as the inner relation?

☐ 99,800 ☐ 100,000 ☐ 100,200 ☐ 19,999,800 ☐ 20,000,000
■ **20,000,200**

Solution: $M + m \times N = 200 + 200 \times 200 \times 500 = 20,000,200$

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

i. **[3 points]** What is the cost of sorting the tuples in `X` on attribute `a`?

☐ 2,000 ☐ 4,000 ☐ 6,000 ■ **8,000** ☐ 10,000 ☐ 12,000

Solution: $passes = 1 + \lceil \log_{B-1}(\lceil \frac{M}{B} \rceil) \rceil = 1 + \lceil \log_{999}(\lceil \frac{2,000}{1000} \rceil) \rceil = 1 + 1 = 2$
$2M \times passes = 2 * 2,000 * 2 = 8,000$

ii. **[3 points]** What is the cost of sorting the tuples in `Y` on attribute `a`?

■ **1,000** ☐ 2,000 ☐ 4,000 ☐ 6,000 ☐ 8,000 ☐ 10,000 ☐ 12,000

Solution: $passes = 1 + \lceil \log_{B-1}(\lceil \frac{N}{B} \rceil) \rceil = 1 + \lceil \log_{999}(\lceil \frac{500}{1000} \rceil) \rceil = 1 + 0 = 1$
$2N \times passes = 2 * 500 * 1 = 1,000$

iii. **[3 points]** What is the cost of the merge phase in the worst-case scenario?
☐ 2,000 ☐ 4,000 ☐ 6,000 ☐ 8,000 ☐ 10,000 ☐ 12,000
☐ 500,000 ■ **1,000,000** ☐ 1,500,000 ☐ 2,000,000

Solution: $M \times N = 2,000 \times 500 = 1,000,000$

iv. **[3 points]** What is the cost of the merge phase assuming there are no duplicates in the join attribute?
■ **2,500** ☐ 5,000 ☐ 7,500 ☐ 10,000 ☐ 15,000 ☐ 500,000
☐ 1,000,000 ☐ 2,000,000

Solution: $M + N = 2,000 + 500 = 2,500$

v. **[3 points]** Now consider joining X, Z and then joining the result with Y. Suppose the cost of the final merge phase is 1,000 and assume that there are no duplicates in the join attribute. How many pages did the join result of X and Z span?
☐ 200 ☐ 300 ☐ 400 ■ **500** ☐ 600 ☐ 700 ☐ 800 ☐ 900

Solution: Let J be the number of pages that the join result of X and Z span. We joined those J pages with Y's N pages, assuming no duplicates in the join attribute, and are told that this merge phase cost $1,000$. In other words, $J + N = J + 500 = 1,000$, i.e., $J = 500$.

(e) Consider a hash join with `X` as the outer relation and `Y` as the inner relation. You may ignore recursive partitioning and partially filled blocks.

i. **[3 points]** What is the cost of the probe phase?
☐ 1,000 ■ **2,500** ☐ 5,000 ☐ 7,500 ☐ 10,000

Solution: $(M + N) = (2,000 + 500) = 2,500$

ii. **[3 points]** What is the cost of the partition phase?
☐ 1,000 ☐ 2,500 ■ **5,000** ☐ 7,500 ☐ 10,000

Solution: $2 \times (M + N) = 2 \times (2,000 + 500) = 2 \times 2,500 = 5,000$

(f) **[6 points]** Consider a hash join with `Y` as the outer relation and `Z` as the inner relation. You may ignore recursive partitioning and partially filled blocks. What is the total cost of the hash join?
■ **700** ☐ 1400 ☐ 2100 ☐ 2800 ☐ 3500

Solution: There is enough space in the buffer to hold N + O pages (700 pages), therefore we can skip partition phase. $(N + O) = (500 + 200) = 700$

(g) For the following statements, pick True or False.

i. **[2 points]** Sort merge is ALWAYS slower than hash join.
□ True ■ **False**

Solution: Sort merge join can be just as fast as hash join under specific circumstances. For example, if the sort merge is performed on already-sorted data (i.e. sort cost is 0 and overall cost is M+N), and the hash join is perform on data can fit entirely in memory where overall cost is M+N.

ii. **[2 points]** Hash join performs well on tables with a lot of duplicate data.
□ True ■ **False**

Solution: If there is a lot of duplicate data, this will result in hash collisions which will make hash join inappropriate. Also, recursive partitioning will not work well since the data cannot be further subdivided into sub partitions.

(h) **[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 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 two 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: Bloom Filter.....[21 points]

Graded by:

Assume that we have a bloom filter that is used to register names. The filter uses two hash functions h_1 and h_2 which hash the following strings to the following values:

input	h_1	h_2
"PostgreSQL"	123	321
"MySQL"	456	654
"BusTub"	789	987
"NoisePage"	445	645

(a) **[7 points]** Suppose the filter has 7 bits initially set to 0:

bit 0	bit 1	bit 2	bit 3	bit 4	bit 5	bit 6
0	0	0	0	0	0	0

Which bits will be set to 1 after "PostgresSQL" and "MySQL" have been inserted?
□ 0 ■ **1** □ 2 ■ **3** ■ **4** □ 5 ■ **6**

Solution: Because the filter has 7 bits, we take the modulo of the hashed output and 7.
$123 \mod 7 = 4; 321 \mod 7 = 6; 456 \mod 7 = 1; 654 \mod 7 = 3$

(b) **[7 points]** Suppose the filter has 7 bits set to the following values:

bit 0	bit 1	bit 2	bit 3	bit 4	bit 5	bit 6
0	1	0	0	1	1	0

What will the filter return if we lookup "BusTub"?
□ True ■ **False**

Solution: $789 \mod 7 = 5; 987 \mod 7 = 0$
Because $987 \mod 7 = 0$, bit 1 must be 1 if "BusTub" has been inserted. Because bit 0 is 0, the filter returns false.

(c) **[7 points]** What will the filter from part (b) return if we lookup "NoisePage"?
■ **True** □ False

Solution: $445 \mod 7 = 4; 645 \mod 7 = 1$
Because bit 1 and bit 4 are both 1, the filter returns true.