CARNEGIE MELLON UNIVERSITY
COMPUTER SCIENCE DEPARTMENT
15-445/645 – DATABASE SYSTEMS (SPRING 2024)
PROF. JIGNESH PATEL

Homework #3 (by Yuchen and Ruijie) – Solutions

Due: **Sunday, Feb 25, 2024 @ 11:59pm**

IMPORTANT:

- Enter all of your answers into **Gradescope by 11:59pm on Sunday, Feb 25, 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 : 2024/02/26 16:49

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 1 million pages (N = 1,000,000 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 <u>30</u> buffers. How many sorted runs are generated? Note that the final sorted file does not count towards the sorted run count.

☐ 34521 ☐ 34524 ☐ 34525 [✓ **34526**] ☐ 34528

Solution:

$$\left\lceil \frac{1,000,000}{30} \right\rceil + \left\lceil \frac{33,334}{29} \right\rceil + \left\lceil \frac{1,150}{29} \right\rceil + \left\lceil \frac{40}{29} \right\rceil = 34526$$

(b) **[6 points]** Again, assuming that the DBMS has <u>30</u> buffers. How many passes does the DBMS need to perform in order to sort the file?

☐ 1 ☐ 2 ☐ 3 ☐ 4 [✓ **5**]

Solution:

$$\begin{aligned} 1 + \left\lceil \log_{B-1}\left(\left\lceil \frac{N}{B} \right\rceil\right) \right\rceil &= 1 + \lceil \log_{29}\left(\lceil 1,000,000/30 \rceil\right) \rceil \\ &= 1 + 4 = 5 \end{aligned}$$

(c) **[6 points]** Again, assuming that the DBMS has <u>30</u> buffers. How many pages does each sorted run have after the third pass (i.e. Note: this is Pass #2 if you start counting from Pass #0)?

☐ 29 ☐ 30 ☐ 31 ☐ 841 ☐ 870 ☐ 900 ☐ 24389 [✓ **25230**]

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: 30 pages for each sorted run. Second pass: $30 * 29 = 870$ pages for each sorted run. Third pass: $30 * 29 * 29 = 25230$ pages for each sorted run.

(d) **[6 points]** Again, assuming that the DBMS has <u>30</u> buffers. What is the total I/O cost to sort the file?

☐ 2,000,000 ☐ 5,000,000 [✓ **10,000,000**] ☐ 20,000,000 ☐ 100,000,000

Solution: $Cost = 2N \times \#passes = 2 \times 1,000,000 \times 5 = 10,000,000$

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

Question 1 continues. . .

☐ 97 ☐ 98 ☐ 99 ☐ 100 ■ **101** ☐ 102 ☐ 103

Solution: We want the smallest integer B such that $N \leq B \times (B-1)^2$. If $B = 101$, then $1,000,000 \leq 101 \times 100^2 = 1,010,000$; any smaller value for B would fail.

(f) **[6 points]** Suppose the DBMS has 410 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?

☐ 167,281 ☐ 167,690 ☐ 168,100 ☐ 68,417,929 ■ **68,585,210**
☐ 68,752,900 ☐ 68,921,000 ☐ 28,051,350,890 ☐ 28,119,936,100

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 $410 \times 409^2 = 68,585,210$

Homework #3 continues…

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

Graded by:

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 = 645 pages in the buffer
- Table `X` spans M = 2,400 pages with 60 tuples per page
- Table `Y` spans N = 500 pages with 240 tuples per page
- Table `Z` spans O = 1,800 pages with 100 tuples per page
- The join result of `Y` and `Z` spans P = 300 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) **[3 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?

☐ 200,600 ☐ 576,500 ☐ 1,200,500 ☐ 72,002,400 ☐ 120,000,600 [✓ **288,000,500**]

Solution: $N + n \times M = 500 + 500 \times 240 \times 2,400 = 288,000,500$

(b) **[3 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?

☐ 1,500 ☐ 1,900 [✓ **2,300**] ☐ 3,000 ☐ 3,300 ☐ 3,600 ☐ 3,900 ☐ 5,100

Solution: $N + \lceil \frac{N}{B-2} \rceil \times O = 500 + \lceil \frac{500}{643} \rceil \times 1,800 = 500 + 1,800 = 2,300$

(c) **[3 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?

☐ 1,500 ☐ 1,900 ☐ 2,300 ☐ 3,000 [✓ **3,300**] ☐ 3,600 ☐ 3,900 ☐ 5,100

Solution: $O + \lceil \frac{O}{B-2} \rceil \times N = 1,800 + \lceil \frac{1,800}{643} \rceil \times 500 = 1,800 + 1,500 = 3,300$

(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`?

☐ 2,400 ☐ 4,800 ☐ 7,200 [✓ **9,600**] ☐ 14,400

Question 2 continues…

Solution: $passes = 1 + \lceil \log_{B-1}(\lceil \frac{M}{B} \rceil) \rceil = 1 + \lceil \log_{644}(\lceil \frac{2{,}400}{645} \rceil) \rceil = 1 + 1 = 2$
$2M \times passes = 2 \times 2,400 \times 2 = 9,600$

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

Solution: $passes = 1 + \lceil \log_{B-1}(\lceil \frac{O}{B} \rceil) \rceil = 1 + \lceil \log_{644}(\lceil \frac{1{,}800}{645} \rceil) \rceil = 1 + 1 = 2$
$2O \times passes = 2 \times 1,800 \times 2 = 7,200$

iii. **[3 points]** What is the cost of the merge phase in the worst-case scenario?
☐ 1,500 ☐ 2,000 ☐ 2,500 ☐ 4,200 ☐ 900,000 ☐ 2,000,000
☐ 3,250,000 ☐ 3,750,000 [✓ **4,320,000**] ☐ 5,000,000

Solution: $O \times M = 1,800 \times 2,400 = 4,320,000$

iv. **[3 points]** What is the cost of the merge phase assuming there are no duplicates in the join attribute?
☐ 1,500 ☐ 2,000 ☐ 2,500 [✓ **4,200**] ☐ 900,000 ☐ 2,000,000
☐ 3,250,000 ☐ 3,750,000 ☐ 4,500,000 ☐ 5,000,000

Solution: $M + O = 1,800 + 2,400 = 4,200$

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?
☐ 1,000 ☐ 2,000 [✓ **2,700**] ☐ 4,700 ☐ 4,320,000

Solution: $P + X = 300 + 2,400 = 2,700$

(e) 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.

i. **[3 points]** What is the cost of the probe phase?
☐ 2,000 ☐ 2,700 ☐ 2,800 [✓ **2,900**] ☐ 3,000 ☐ 5,400 ☐ 10,000

Solution: $(N + M) = (500 + 2,400) = 2,900$

ii. **[3 points]** What is the cost of the partition phase?
☐ 2,000 ☐ 2,400 ☐ 2,900 [✓ **5,800**] ☐ 6,000 ☐ 8,700 ☐ 10,000

Solution: $2 \times (N + M) = 2 \times (500 + 2,400) = 2 \times 2,900 = 5,800$

(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 != 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.

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

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

i. **[2 points]** In a simple nested loop join where one of the tables fits entirely in memory, it is beneficial to use that table as the inner table.

[✓ **True**] ☐ False

Solution: Using the table that fits in memory as the inner table would reduce I/O costs as it would then be read only once, instead of repeatedly for each tuple of the outer table.

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

Solution: A block nested loop join has fewer disk accesses when compared to a simple nested loop join.

iii. **[2 points]** For a block nested loop join, in the worst case, each block in the inner table has to be read once for each tuple in the outer table.

☐ True [✓ **False**]

Solution: Even in the worst case, each block in the inner table is read only once for each *block* in the outer table.

iv. **[2 points]** A sort-merge join is slower than a hash join on all circumstances.

☐ 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.

v. **[2 points]** For a hash join to work, the inner table (or its partitions) need to fit into memory.

Question 2 continues. . .

☐ True [✓ ■ **False**]

Solution: The inner table can be any size. Only outer table (or its partitions) need to fit in memory.

Question 3: Bloom Filter..[21 points]

Graded by:

Assume that we have a bloom filter that is used to register database 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
“ChiDB”	999	996
“YourSQL”	233	666
“RusTub”	235	468
“GooseDB”	721	445

(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 “ChiDB” and “YourSQL” 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.
$999 \mod 7 = 5$; $996 \mod 7 = 2$; $233 \mod 7 = 2$; $666 \mod 7 = 1$

(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	1

What will we learn if we lookup “RusTub”?

☐ RusTub has been inserted ☐ RusTub has not been inserted

[✓ **RusTub may have been inserted**]

Solution: $235 \mod 7 = 4$; $468 \mod 7 = 6$

Because bit 4 and bit 6 are both 1, filter will return True, meaning we might have RusTub inserted.

(c) **[7 points]** What will we learn using the filter from part (b) if we lookup “GooseDB”?

☐ GooseDB has been inserted [✓ **GooseDB has not been inserted**]

☐ GooseDB may have been inserted

Solution: $721 \mod 7 = 0$; $445 \mod 7 = 4$

Because bit 0 is 0, the filter will just return false.