CARNEGIE MELLON UNIVERSITY
COMPUTER SCIENCE DEPARTMENT
15-445/645 – DATABASE SYSTEMS (FALL 2025)
PROF. ANDY PAVLO

# Homework #4 (by Will) – Solutions
Due: **Sunday, Nov 2, 2025 @ 11:59pm**

**IMPORTANT:**

- Enter all of your answers into **Gradescope by 11:59pm on Sunday, Nov 2, 2025**.
- **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)
- Each part is all or nothing. There is no partial credit.

*Revision* : 2025/11/09 20:54

| Question | Points | Score |
|---|---|---|
| Join Algorithms | 45 | |
| Query Execution, Planning, and Optimization | 30 | |
| Cardinality Estimation | 25 | |
| Total: | 100 | |

## Question 1: Join Algorithms ................................. [45 points]

**Graded by:**

Consider relations `X(a, b)`, `Y(a, c, e)`, and `Z(a, d, 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 $B$ = 100 pages in the buffer
- Table `X` spans $M$ = 1,000 pages with 100 tuples per page
- Table `Y` spans $N$ = 500 pages with 1,000 tuples per page
- Table `Z` spans $O$ = 3,000 pages with 3000 tuples per page
- The join result of `Y` and `Z` spans $P$ = 200 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?

☐ 51,000
☐ 101,000
☐ 675,250
☐ 500,550
☐ 50,001,000
■ **500,000,500**

**Solution:** $N + n \times M = 500 + 500 \times 1,000 \times 1,000 = 500,000,500$

(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?

☐ 1,650
☐ 3,250
☐ 15,500
■ **18,500**
☐ 21,250
☐ 96,000

**Solution:** $N + \lceil \frac{N}{B-2} \rceil \times O = 500 + \lceil \frac{500}{98} \rceil \times 3,000 = 500 + 6 \times 3,000 = 18,500$

(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?

☐ 17,500

- ☐ 18,000
- ■ **18,500**
- ☐ 19,000
- ☐ 19,500

**Solution:** $O + \lceil \frac{O}{B-2} \rceil \times N = 3,000 + \lceil \frac{3,000}{98} \rceil \times 500 = 3,000 + 31 \times 500 = 18,500$

(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?

- ☐ 1,000
- ☐ 2,000
- ■ **4,000**
- ☐ 6,000
- ☐ 8,000

**Solution:** $passes = 1 + \lceil \log_{B-1}(\lceil \frac{M}{B} \rceil) \rceil = 1 + \lceil \log_{99}(\lceil \frac{1,000}{100} \rceil) \rceil = 1 + 1 = 2$
$2M \times passes = 2 \times 1,000 \times 2 = 4,000$

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

- ☐ 3,000
- ☐ 6,000
- ■ **12,000**
- ☐ 18,000
- ☐ 24,000

**Solution:** $passes = 1 + \lceil \log_{B-1}(\lceil \frac{O}{B} \rceil) \rceil = 1 + \lceil \log_{99}(\lceil \frac{3,000}{100} \rceil) \rceil = 1 + 1 = 2$
$2O \times passes = 2 \times 3,000 \times 2 = 12,000$

iii. **[3 points]** What is the cost of the merge phase in the worst-case scenario?

- ☐ 1,000
- ☐ 3,000
- ☐ 4,000
- ☐ 300,000
- ■ **3,000,000**

**Solution:** $O \times M = 3,000 \times 1,000 = 3,000,000$

iv. **[3 points]** What is the cost of the merge phase assuming there are no duplicates in the join attribute?

- ☐ 1,000
- ☐ 3,000
- ■ **4,000**
- ☐ 300,000

☐ 3,000,000

**Solution:** $O + M = 3,000 + 1,000 = 4,000$

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?
☐ 700
■ **1,200**
☐ 3,200
☐ 100,000
☐ 200,000
☐ 600,000

**Solution:** $P + X = 200 + 1,000 = 1,200$

(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?
☐ 3,000
■ **4,500**
☐ 6,000
☐ 9,000
☐ 12,000

**Solution:** $3(N + M) = 4,500$

(f) **[4 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 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 $h_1$ for large buckets.
☐ Use linear probing for collisions and page in and out parts of the hashtable needed at a given time.

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

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

ii. **[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

**Solution:** If both tables fit entirely in memory, then they can be read just once and therefore the order would not be very important.

iii. **[3 points]** An index nested loop join requires an index on only the inner- table.
■ **True** ☐ False

**Solution:** An index nested loop join only requires the inner- table to have an index.

iv. **[3 points]** A sort-merge join is faster 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. **[3 points]** For a hash join to work, the inner table (or its partitions) need to fit into memory.
☐ True ■ **False**

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

vi. **[5 points]** Select all true statements about joins:
☐ Each tuple is scanned exactly once in a naive nested loop join.
☐ Hash Joins are always preferred over a Sort-Merge Join when both relations are already sorted on the join key.
☐ Sort-Merge Join requires both input relations to be pre-sorted.
■ **For block nested loop joins, increasing the buffer pool could reduce the number of disk I/Os.**
☐ Index Nested Loop Join requires pre-sorting the inner relation.
☐ None of the above.

**Solution:** Only (D) is true.

## Question 2: Query Execution, Planning, and Optimization . . . . . [30 points]

**Graded by:**

(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

**Solution:** True. If the zone maps are up-to-date, the DBMS could answer queries asking for the minimum value of a given attribute (amongst others) by only looking at zone maps.

(b) **[3 points]** Assume a query with multiple `OR` predicates with high selectivity (i.e., retrieves a few tuples). Using a multi-index scan is preferred over a sequential scan for extremely large tables (i.e., >1,000,000 tuples).
■ **True** □ False

**Solution:** True.

(c) **[3 points]** The iterator model allows tuples to continously flow through the entire sequence of operators in the execution plan before retrieving the next tuple.
□ True ■ **False**

**Solution:** False. The statement is true only for a single pipeline, but a query can have multiple pipelines. If an operator is a pipeline breaker (e.g. build-side of hash join, subqueries, order-by), it cannot emit tuples until all its children emit all their tuples.

(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**

**Solution:** False. While the process per DBMS worker approach does provide better fault isolation, thread per worker gives the DBMS finer control over scheduling.

(e) **[3 points]** For OLAP queries, which often involve complex operations on vast datasets, intra-query parallelism can improve performance.
■ **True** □ False

**Solution:** True. OLAP queries, characterized by their complex operations on large volumes of data, can greatly benefit from intra-query parallelism. By executing the operations of a single query in parallel, it helps in significantly decreasing the latency, thus optimizing the performance of these types of queries.

(f) **[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**

**Solution:** False. While the Vectorized Model can reduce some I/O operations due to its batch processing, its primary advantage is from reducing CPU overhead, optimizing cache utilization, and leveraging SIMD instructions.

(g) **[2 points]** The execution plan with the lowest cost is guaranteed to be the most efficient among all execution plans enumerated by the query optimizer.
☐ True ■ **False**

**Solution:** False. The execution plan is not guaranteed to be the most efficient, as the statistics used to compute cardinality might be stale and the cost model may not accurately reflect the actual performance of the plan (e.g. assume all predicates in a conjunction is indepedent, while there might be some correlation).

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

**Solution:** False. If the predicate/projection being pushed down involves an expensive function call (i.e., UDF), the query may benefit from deferring it to later.

(i) **[2 points]** Sampling statistics requires evaluating a subset of the table's tuples.
■ **True** ☐ False

**Solution:** True. Sampling only needs to look at a subset (or representative sample) of the table.

(j) **[2 points]** Equi-depth histogram maintains counts for a group of values instead of each unique key to reduce memory footprint and uses the same range size for each bucket.
☐ True ■ **False**

**Solution:** False. Equi-depth histogram varies the range size of buckets so that the total number of occurrences for each bucket is roughly the same.

(k) **[3 points]** Consider two options to estimate the selectivity of a selection predicate: a *histogram* built on the selection column or using *sampling* to estimate the predicate selectivity. Which of the following statements is true? Select all that apply.
☐ Histograms are more accurate than sampling for skewed data distributions
■ **Sampling can adapt better to arbitrary predicates than histograms**
☐ Sampling always provides faster estimates than histograms
☐ Histograms are generally more effective for columns with floating-point values than for columns with integer values

**Solution:** Histograms often struggle with skewed data – you have to get the bin boundaries right, which is difficult. Sampling can generally adapt better to arbitrary predicates as it does not rely on pre-defined bins, and can estimate selectivity for any predicate, accurately provided the sample is representative of the actual data. Sampling is slower than histograms, though. All this holds regardless of the data type of the column.

## Question 3: Cardinality Estimation ........................... [25 points]

**Graded by:**

(a) A database contains a single table: `University(id,name,state,city)`. You need to estimate the cardinality of the following query:

`SELECT * FROM University WHERE state = ‘PA’ AND city = ‘Pittsburgh’`

For the following questions, assume `University` has 10,000 rows with $10\%$ having `state = ‘PA’`, and $0.5\%$ having `city = ‘Pittsburgh’`.

i. **[4 points]** Under uniform data assumption and independent predicates assumption, what is the estimated cardinality $c$ of this query? Take $\lceil c \rceil$ of the result.

■ **5**
□ 25
□ 50
□ 100
□ 500
□ 1,000

**Solution:** $\lceil 10,000 \times 0.1 \times 0.005 \rceil = \lceil 5 \rceil = 5.$

ii. **[2 points]** Is the result from previous question likely an overestimate or underestimate of the true cardinality?

□ Overestimate ■ **Underestimate**

**Solution:** It is likely an underestimate of the true cardinality since the state and city are not independent attributes - they are highly correlated.

(b) Consider the following schema and statistics:

- $N_1$=4,000 tuples in $R1(A, B)$, with
  - $V(A, R1)$= 250 distinct values of $A$
  - $V(B, R1)$= 500 distinct values of $B$
- $N_2$=8,000 tuples in $R2(B, C)$
  - $V(B, R2)$= 500 distinct values of $B$
  - $V(C, R2)$= 800 distinct values of $C$
- $N_3$=16,000 tuples in $R3(C, D,$ FOREIGN KEY $(D)$ REFERENCES $R4(D))$
  - $V(C, R3)$= 800 distinct values of $C$
  - $V(D, R3)$= 900 distinct values of $D$
- $N_4$=4,000 tuples in $R4(D$ PRIMARY KEY $, E)$
  - $V(D, R4)$= 4,000 distinct values of $D$
  - $V(E, R4)$= 1,200 distinct values of $E$

Assume that:

- *Uniformity*: the values are uniformly distributed in each table.
- *Perfect overlap*: the distinct values of $B$ in $R1$ are the same as the distinct values of $B$ in $R2$ (and ditto for $C$, etc).

i. **[1 point]** Estimate the number of tuples for the query $\sigma_{D=100}(R4)$.
■ **1** ☐ 900 ☐ 1,200 ☐ 4,000 ☐ None of the above

**Solution:** R4.D is a primary key. Thus, a query optimizer would estimate that 1 tuple exists.

ii. **[1 point]** Estimate the number of tuples for the natural join query $A = R3 \bowtie R4$.
☐ 1 ☐ 800 ☐ 900 ☐ 1,200 ☐ 4,000 ■ **None of the above**

**Solution:** R3.D is a foreign key reference to R4.D and R4.D is a primary key, so each tuple in R3 will join with exactly one tuple in R4.

iii. **[2 points]** Estimate the number of tuples for the query $\sigma_{A=5}(R1)$.
☐ 1 ☐ 8 ■ **16** ☐ 250 ☐ 4,000 ☐ None of the above

**Solution:** $N_1/V(A, R1) = 4000/250 = 16$

iv. **[2 points]** Estimate the number of tuples for the query $\sigma_{C\neq 99}(R3)$.
☐ 1 ☐ 20 ☐ 800 ■ **15,980** ☐ 16,000 ☐ None of the above

**Solution:** $N_2 - (N_2/V(C, R3)) = 16,000 - (16,000/800) = 15,980$

v. **[4 points]** Estimate the number of tuples for the query $\sigma_{B=2\wedge C=2}(R2)$
■ **0.02** ☐ 10 ☐ 16 ☐ 8,000 ☐ None of the above

**Solution:** $N_2 \times (1/V(B, R2)) \times (1/V(C, R2)) = 8,000/500/800 = 0.02$

vi. **[4 points]** Estimate the number of tuples for the natural join query $A = R1 \bowtie R2$.
☐ 4,000 ☐ 8,000 ☐ 40,000 ■ **64,000** ☐ 128,000 ☐ None of the above

**Solution:** $N_1 \times N_2/V(B, R1) = 4000 \times 8,000/500 = 64,000$

vii. **[5 points]** Estimate the number of qualifying tuples for the natural join query $B = R1 \bowtie R2 \bowtie R3$. Assume that the number of distinct values of $C$ is always 800.
☐ 16,000
☐ 64,000
☐ 800,000
■ **1,280,000**
☐ 2,048,000
☐ 2,560,000
☐ None of the above

**Solution:** $|R1 \bowtie R2| \times N_3/V(C, R2) = 64,000 \times 16,000/800 = 1,280,000$