Homework 3
Spring 2020

Due: 11:59PM EDT, April 20, 2020. Submit using Blackboard.
(There will be a 10% penalty for each late day. After 5 late days, the homework will not be accepted.)

Part 1: External Sorting (30 Points)

Assume you have a table of 10,000,000 records, each of size 128 Bytes. A disk page is of size 2048 bytes. We want to sort this file. Calculate the **cost** to perform the sort when using:

1. Two-way external Merge Sort. State the number of passes needed.
2. The general external sort-merge algorithm, assuming a buffer of 20 pages. What is the number of passes needed?
3. A clustered B+-tree of height 3 on the table.
   State any assumptions you make.

Part 2: Costing of Operators (45 points)
Consider the following relational schema and SQL query. The schema captures information about employees, departments, and sales.

Books(<u>bid:integer</u>, btitle:char(100), bgenre:char(20))
Publishers(<u>pid:integer</u>, pname:char(100), paddress: char(200))
Sale(<u>sid:integer</u>, bid:integer, pid:integer, cost:real)

```
SELECT B.btitle, P.pname
FROM Books B, Publishers P, Sale S
WHERE
        B.bid = S.bid AND P.pid = S.pid
        AND s.cost >= 100 AND B. bgenre = ‘fiction’;
```

Suppose that the following additional information is available: Unclustered B+ tree indexes exist on Publishers.pid, Books.bid, Books.bgenre, Sale.cost. The system’s statistics indicate that book’s price range from $0 to $1,000, there are 20 different book genres. There is a total of 10,000 books, 100 publishers, and 200,000 sales in the database.

(a) For each of the query’s base relations (Books, Publishers, and Sale), estimate the number of tuples that would be initially selected from that relation if all of the non-join predicates on that relation were applied to it before any join processing begins.

(b) Assume the query optimizer has these constraints: Only left-deep plans are considered. No plan is considered if it requires a cross-product. The only available join method is index nested loops join. What are the possible join orders for this query? Express you answer using a relational algebra expression for each possible join order.

(c) Each possible join order in (b) can be expanded to one or more fully-detailed query plans. From among these query plans, choose the plan with the lowest I/O cost. Sketch this plan and show the computed I/O cost as instructed below.
First, sketch out your chosen query plan as a tree, using the following operators:

Operator	Format
Base table	[table name]:[rows in table]
Select	σ:[column being selected on],[index or on-the-fly],[number of rows passed up to next operator] Note: If a σ operator is the inner relation of a join, “number of rows passed up” should be the rows returned in one inner loop, not the total number of rows returned from that operator.
Join	⋈:[column being joined on],[number of rows passed up to next operator] Note: The “number of rows passed up” by the join will be the product of the rows passed up to it from the outer relation and from the inner relation.

Now that you have written all of your operators out, compute the total number of I/Os incurred by each operator, sum the I/Os, and show the final I/O cost of the query. Note: Do not consider buffer manager constraints for this problem. Assume the following:

Each B+ tree has three levels: root level, inner level, leaf level.
Each leaf node of a B+ tree holds pointers to 100 rows.
The cost of a base table scan is 1 I/O per 20 rows.
The cost of writing out the final result is 1 I/O per 20 rows.

Part 3: Left-deep Trees, Right-deep Trees (25 points)

Suppose we have a database with the following schema:

Students (sid: integer, sname: string)
Courses (cid: integer, cname: string)
Grades (sid, cid, ggrade: string)

Consider the following query over this schema:

SELECT *
FROM Students s, Courses c, Grades g
WHERE g.sid = s.sid AND g.cid = c.cid;

The students table takes 500 pages on disk.
The Courses table takes 100 pages on disk.

The Grades table takes 2000 pages on disk.
Joining Students and Grades produces 300 pages of rows.
Joining Grades and Courses produces 100 pages of rows.
Joining Students, Grades, and Courses produces 50 pages of rows.
The machine running the query has enough memory for 3 buffer pages.
There are no indexes, and all joins are nested loop joins.
(Note: You may not need all of this information to answer the question.)

Now consider the two query plans below:

1. $((s \bowtie g) \bowtie c)$
2. $(s \bowtie (g \bowtie c))$

Plan 1 is left deep, and plan 2 is right deep.

1. Using the information given above, determine what is the minimum number of disk I/O's which must be incurred by each plan. (Note: Larger numbers of I/O are possible if the joins do not go well. For this question, assume the joins go perfectly and incur the smallest possible number of I/Os.
2. Assume that for the right deep tree, we use a temp operator on top of the join of g and c as explained in class so that we do not recompute the join over and over. The temp operator stores the result of the join in disk and then reads from that table afterwards instead of repeatedly recomputing the join. What is the number of disk I/O's for this optimized plan, and how does it compare with the costs of the left- and right-deep plans.
3. Based on what you have studied, can you suggest a query evaluation plan that will be the cheapest of all? Explain your answer.
4. Discussion: What do you think is the most important factor that affects the cost of the query execution plan. Please explain your answer.