Problem Set 4

1. **Evaluation of Relational Operators**

Consider the following query:  
SELECT R.a, T.k  
FROM R,S,T  
WHERE R.b=S.b AND S.f=T.f  
AND R.c=10 AND S.g>1000;  
  
over the relations R(a, b, c, d), S (b, e, f, g) and T( f, k).  
  
Assume that:

* + R has 12,000 tuples in 120 pages.
  + S has 4,000 tuples in 40 pages.
  + T has 2,000 tuples in 10 pages.
  + There are 10 available buffer pages.
  + Each tuple of S joins with (on average) 3 tuples of R.
  + Each tuple of T joins with (on average) 2 tuples of S .

**(a)** Which are the possible join orderings a query optimizer would consider

**(b)** What would be the I/O cost (excluding output) of the following join operations?

* + R ⋈ S using a nested loop join?
  + S ⋈ R using a nested loop join?
  + R ⋈ S using a block nested loop join?
  + S ⋈ R using a block nested loop join?
  + R ⋈ S using a sort-merge join?
  + Assuming unlimited buffer space; what would be the most efficient implementation for joining R and S ? What would its cost be? How much buffer space would you need?

**(c)** Suggest a reasonable query plan. What would be the cost of your plan, assuming that 10% of R tuples match the R.c = 10 condition, and 50% of S tuples match the S.g > 1000 condition? You can use indexes if you wish, but if you do you have to describe them, and mention their impact on cost. (10 points)

1. **Evaluation of Relational Operators : Textbook Problems: 14.4, 14.3**
2. **External Sorting: Textbook Problems: 13.1, 13.2**
3. **Evaluation of Relational Operators**

Assume that you want to join two relations R(A,B) and S(B,C). The two relations are stored as simple (unsorted) heap files.

a) Briefly describe one scenario where you would prefer a hash-join to a sort-merge join.  
b) Briefly describe two different scenarios where you would prefer a sort-merge join to a hash-join

1. **Query Optimization**

**1)** Consider the following SQL query:

SELECT P.A, Q.B, R.C

FROM P, Q, R

WHERE P.A = Q.A and Q.B = R.B

a) How many left-deep join orderings are there for the above query?

b) How many of these left-deep orderings are considered by a System R style optimizer?

c) What are the total number of join orderings (both left-deep and not left-deep) for this query?

1. **Database Security: Textbook Problem 21.2**