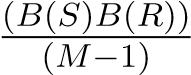
# WORKED OUT ANSWERS

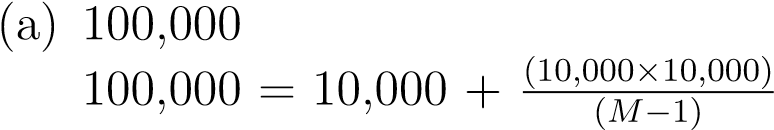
# Part 1

## **Problem 1**

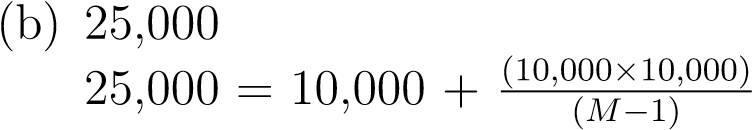
1. Suppose B(R) = B(S) = 10,000. For what value of M would we need to compute R !" S using the nested-loop join algorithm with no more than the following number of I/Os? (8 points, 4 points each)

Using the equation given in Section 15.3.4 of the textbook, solve for M:

I/O = B(S) + 



M = 1,112.1 or ceil(M) = 1,113



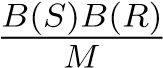
M = 6,667.7 or ceil(M) = 6,668

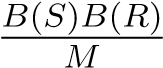
1. If two relations R and S are both unclustered, it seems that the nested-loop join algorithmrequires about T(R)T(S)/M disk I/Os. How can you do signicantly better than this cost? Describe your modied version of the nested-loop algorithm and give the number of disk I/Os required for your algorithm. We assume that M is large enough such that M ? 1 ! M , and that B(R) ! T (R) and B(S) ! T (S); that is, the number of tuples of a relation is much greater than that of blocks of the relation. (8 points)

Note that the cost of algorithm given in the question is T(R)T(S)/M, which means it is using tuple-based nested-loop join. In order to improve the disk I/O cost of nested-loop join algorithm, we need to use block-based nested-loop join. In order to carry out block-based nested loop join efficiently, we need the inner relation clustered, and search structure built on the common attributes of R and S.

Let R be the inner relation (assuming S is smaller):

•• Cost of reading all tuples of R, cluster them, and write them back: T(R) + B(R)Cost of Reading tuples of S, plus the cost of joining them with R in the main memory:

T(S) + 

Therefore the total cost is T(R) + B(R) + T(S) +.

**Problem 2.**

We have two relations R and S where B(R) = B(S) = 10, 000. Give an approximate size of main memory M required and the number of disk I/Os in order to perform the two-pass algorithms for the following operations: (12 points, 4 points each)

1. set union

I/O of set union operation = 3= 3 ×(10,000 + 10,000) ×(B(S) + B(R))

= 60,000

Approximate M requirement for set union operation is !(*B*)*R* + *B*(*S*) = √20*,*000 = 141.42, which the given M satisfies.

1. simple sort-join

I/O of set union operation = 5 ×(B(S) + B(R))

= 5 ×(10,000 + 10,000)

= 100,000

Note that given figures satisfy the required M, which is B(S) and B(R) ≤ M2, i.e. 10,000 ≤ 10,00,000

1. the more efficient sort-join described in Section 15.4.8 of the textbook

I/O of set union operation = 3 ×(B(S) + B(R))

= 3 ×(10,000 + 10,000)

= 60,000

Requirement for M in the e1,000,000 which is satisfied here.fficient sort-join algorithm is B(R) + B(S) ≤ M2 = 20,000

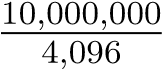
≤

## **Problem 3**

## Two-pass Algorithms Based on Sorting (20 points)

1. Suppose we have a relation with 1,000,000 records and each records requires 10 bytes. Let the disk-block size be 4,096 bytes. (8 points, 4 points each)

1. What is the minimum number of blocks in main memory required for using TPMMS(Two-Phase Multiway Merge-Sort) to sort these records?

The size of the relation in bytes is 1,000,000 × 10 = 10,000,000 bytes, and each disk-block is 4,096 bytes. The minimum number of blocks to hold the relation is ceil( ) = 2,442√2*,*442) =. The minimum M requirement for TPMMS is Bceil(49.4) = 50. ≤ M2, M must

at least be ceil(

1. Following (a), how many disk I/Os are needed to sort all the records?Number of disk I/O for TPMMS is 3B, which is 3 × 2,443 = 7,329

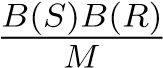
**Problem 4**

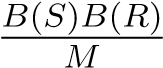
If two relations R and S are both unclustered, it seems that the nested-loop join algorithmrequires about T(R)T(S)/M disk I/Os. How can you do signicantly better than this cost? Describe your modied version of the nested-loop algorithm and give the number of disk I/Os required for your algorithm. We assume that M is large enough such that M ? 1 ! M , and that B(R) ! T (R) and B(S) ! T (S); that is, the number of tuples of a relation is much greater than that of blocks of the relation. (8 points)

Note that the cost of algorithm given in the question is T(R)T(S)/M, which means it is using tuple-based nested-loop join. In order to improve the disk I/O cost of nested-loop join algorithm, we need to use block-based nested-loop join. In order to carry out block-based nested loop join efficiently, we need the inner relation clustered, and search structure built on the common attributes of R and S.

Let R be the inner relation (assuming S is smaller):

•• Cost of reading all tuples of R, cluster them, and write them back: T(R) + B(R)Cost of Reading tuples of S, plus the cost of joining them with R in the main memory:

T(S) + 

Therefore the total cost is T(R) + B(R) + T(S) +.

**Problem 5**

(10 points, 5 points each)

1. We consider two relations R(*A*, *B*, *C*) and S (*C*, *D*, *E*). Convert the following expressions in relational algebra by applying algebraic laws so that we can perform selections and projections as early as possible.

1. *σB*=3*ANDE*=4(R !" *σC>*10(S)) *σ*(*B*=3)*AND*(*C>*10)(R) !" *σC>*10*ANDE*=4(S)

*(b)πA,D*(R !" S) *πA,D*(*πA,C*(R) !" *πC,D*(S))

**Problem 6**

Dynamic Programming (15 points) Compute the optimal plan for R !" S !" T !" U using the technique of dynamic programming. We make the following assumptions (as we did in the class):

* B(R) = 400, B(S) = 800, B(T) = 600, and B(U) = 700.
* B(R1)The size of a join for two relations R1 and R2 is estimated as: B(R1× B(R2). If a subplan is a single relation and does not involve any join, the size!" R2 ) = 0.01 × of its intermediate result is zero.
* The cost of a join is estimated to be the cost of the subplans plus the size of theintermediate results.
* The cost of a scan is zero.

Draw the table for dynamic programming, to show how you compute the optimal plan for all possible join orders allowing all trees.

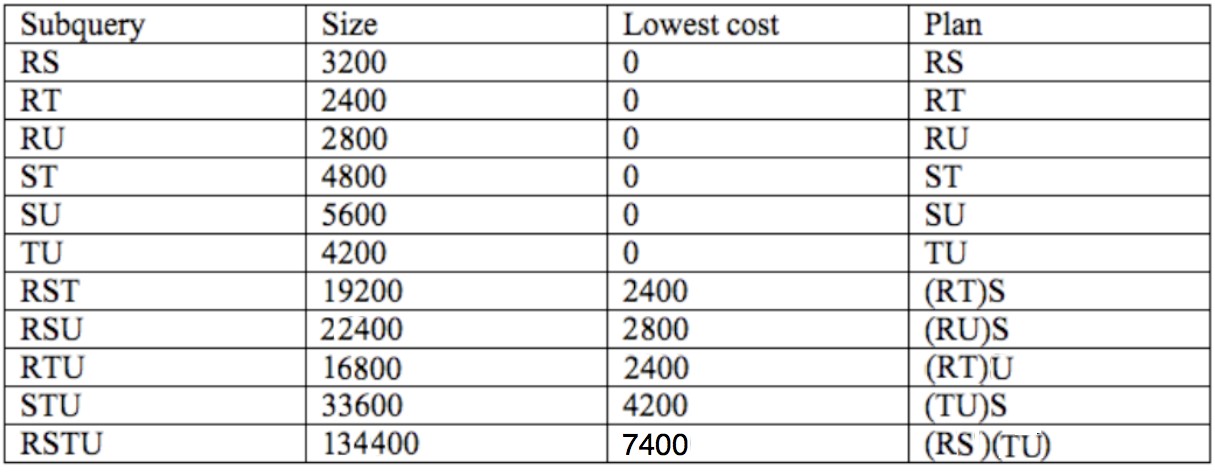


Figure 1: dynamic programming plan