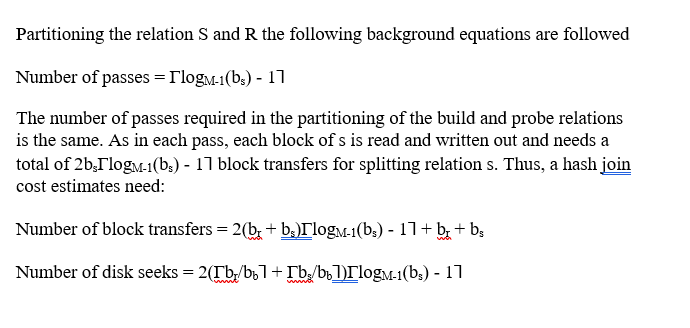
# WORKED OUT ANSWERS

# **Part 1**

## **Problem 1**

****

**Problem 2.**

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

= 3 ×(10,000 + 10,000)

= 60,000

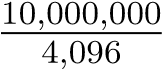
Requirement for M in the e1,000,000 which is satisfied here the efficient sort-join algorithm is B(R) + B(S) ≤ M2 = 20,000

**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**

going to be removed from the result. Thus the right hand side expression can be evaluated more efficiently than the left hand side expression.

R(A,B,C)

T(E,F,G)

S(C,D,E)

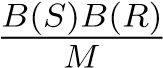
If two relations R and S are both un-clustered, it seems that the nested-loop join-algorithm requires about T(R)T(S)/M disk I/Os. How can you do signicantly better than this cost?

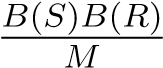
. We assume that M is large enough such that M ? 1 ! M , and that B(R) ! T (R) and B(S) ! T (S); that is, the number of tuples of a relation is much greater than that of blocks of the relation. (8 points)

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**

1. **Pro**(10 points) Π*A*(*R* ∩ *S*) = Π*A*(*R*) ∩ Π*A*(*S*)

The statement is is false as the conjuctive propertieso of the conglomerate of R and S are different fro those of the individual properties

A good example is

3(4\*12)=3(48) and not 3(4)\*3(12)

**Conditional rule of the State**

R∩ S = R∪ S − (R− S) − (S − R)

For our case let

K(R∩S)=K(R) ∩K(S)

For the relation conjuction of the associate derivative of R and S th

1. (10 points) *σθ*(*R* ∪ *S*) = *σθ*(*R*) ∪ *σθ*(*S*)

The statement is true since a conglomerate of the asset branches of either R and S in a mixture are the same as well in the mixture

any tuple t in the output of (R ∪ S) is filtered and thus a selection of the output or the right hand and the left hand are not congruent to each other

Therefore: ∀t, t 6∈ su(R U S) ⇒ t 6∈ su(R) ∪ S

**Problem 6**

Joining Layout of the Infrastructure

Rr=4000*,*

*Rs=3000,*

*Rt=2000,*

*Ru=10000.*

**Assumptions**

assume that all attributes of the relations are of the same length and we use hash join,

**Cost function**

so the cost of joining *X* ∈ {*R,S,T,U*} and *Y* ∈ {*R,S,T,U*} can be expressed as:

*K*(*RX* · *CX* + *CY* · *CY* )

**Cost Calculations**

*R 1 S*

14000*k* , [= (4000 · 2 + 3000 · 2)*k*]

*R 1 T*

12000k , [=(4000\*2+2000\*2)k]

*R 1 U*

10000k, [=(4000\*2+1000\*2)k]

*S 1 T*

10000k, [=(3000\*2+2000

*T 1 U*

6000k, [=(2000\*2+1000\*2)k]

*R 1 S 1 T*

18000k, [=(4000\*2+3000\*2+2000\*2)k]

*R 1 S 1 U*

16000k, [=(4000\*2+3000\*2+1000\*2)k]

*R 1 T 1 U*

14000k, [=(4000\*2+2000\*2+1000\*2)k]

*S 1 T 1 U*

12000k, [=(3000\*2+2000\*2+1000\*2)k]

*R 1 S 1 T 1 U*

20000k, [=(4000\*2+3000\*2+2000\*2+1000\*2)k]

**Problem 6 :Figure of the Dynamic Programming Plan**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Subquery | Size | Cost | | BestPlan |
| *R 1 S* | 3200 | 14000*k* | [= (4000 · 2 + 3000 · 2)*k*] | *R 1 S* |
| *R 1 T* | 2400 | 12000k [=(4000\*2+2000\*2)k] |  | RT |
| *R 1 U* | 2800 | 10000k  [=(4000\*2+1000\*2)k] |  | RU |
| *S 1 T* | 4800 | 10000k  [=(3000\*2+2000\*2)k] |  | ST |
| *S 1 U* | 5600 | 8000k  [=(3000\*2+1000\*2)k] |  | SU |
| *T 1 U* | 4200 | 6000k  [=(2000\*2+1000\*2)k] |  | TU |
| *R 1 S 1 T* | 19200 | 18000k  [=(4000\*2+3000\*2+2000\*2)k] |  | (RT)S |
| *R 1 S 1 U* | 22400 | 16000k  [=(4000\*2+3000\*2+1000\*2)k] |  | (RT)S |
| *R 1 T 1 U* | 16800 | 14000k  [=(4000\*2+2000\*2+1000\*2)k] |  | (RT)U |
| *S 1 T 1 U* | 33600 | 12000k  [=(3000\*2+2000\*2+1000\*2)k] |  | (TU)S |
| *R 1 S 1 T 1 U* | 134400 | 20000k  [=(4000\*2+3000\*2+2000\*2+1000\*2)k] |  | (RS)(TU) |