CSE232A Midterm, Winter 2004

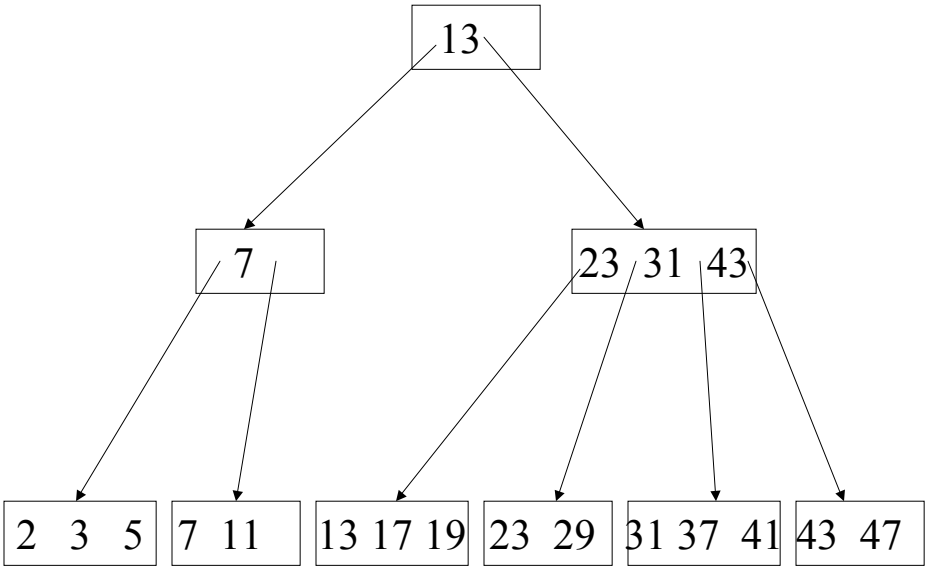
Name:

This is an open book, open notes exam. You have 80 minutes. Express your unique answer to each problem clearly, precisely and succinctly.

Problem 1

Consider the following B+ tree with *n=3*. Notice that the splits have to happen in such a way that no two internal nodes may have the same key value (for example, 13 does not appear in the second level.) Show its state after the following operations happen:

1. Insert a record with key 1
2. Insert records with keys 14 through 16. Show the splits that will happen and the final state after 14, 15, and 16 are inserted.
3. Delete the record with key 23. (Recall that the minimum number of nodes in a leaf is 2.)



Problem 2

Consider two relations *R(A,B)* and *S(B,C)* with the following characteristics:

- The relations have a common attribute *B*, whose size is 10 bytes (80 bits).
- Each tuple of *R* and each of tuple of *S* is 1kbyte
- $T(R) = 10^{12}$
- $T(S) = 10^{8}$
- $V(S,B) = 10^{6}$

We want to compute the antisemijoin of *R* and *S*, which returns the tuples of *R* whose *B* attribute does not appear in *S*. In particular if a tuple *t* appears *n* times in *R* and its *B* value does not appear in the *B* attribute of a *S* tuple then *t* appears *n* times in the output. Otherwise (if its *B* value appears in *R*) then *t* does not appear in the output. The following SQL query is another definition of the antisemijoin of *R* and *S*.

```
SELECT R.A, R.B
FROM R
WHERE R.B NOT IN (SELECT S.B FROM S)
```

Assume that the memory that the database makes available is $1.5\ 10^{7}$ bytes. The block size is 4kbyte.

Write a *one-pass algorithm* that computes the antisemijoin of *R* and *S*, i.e., the algorithm must read the tuples of *R* and *S* just once and output the antisemijoin without storing any intermediate result or data structure in files or disk.

Problem 3

Consider again the relations *R(A,B)* and *S(B,C)* and the following SQL query that computes the antisemijoin of *R* and *S*.

```
SELECT R.A, R.B
FROM R
WHERE R.B NOT IN (SELECT S.B FROM S)
```

Do the following bag algebra expressions correctly compute the above SQL query expression?

1. $R - (\pi_{R.A,R.B} \sigma_{R.B=S.B} (R \times S))$

2. $\pi_{A.\ B} \sigma_{B=D} [R \times [(\pi_{B \to D} R) - (\pi_{B \to D} S)]]$

For each one of the them

- If the answer is yes, provide a proof. Recall, a proof has to show that every tuple *t* that appears *n* times in the result of the SQL query also appears *n* times in the result of the expression and no other tuple appears in the result of the expression.
- If the answer is no, provide a counterexample.

Problem 4

Consider a modification of the INGRES algorithm that

1. whenever there are more than one "small" relations the algorithm proceeds by selecting the small relation that leads to the smallest intermediate result in this step. (It may be that this decision will not lead to the best overall plan.)
2. produces only left-deep join expressions. Note that the right argument of a join operator may be an expression that also involves selection but it is not allowed to also involve join.

Consider the following relations

```
Nation(NationName, NKey)
Customer(CName, NKey, CKey)
Order(Date, CKey, OKey)
LineItem(Product, OKey)
```

with the following characteristics (δ is the duplicate elimination operator)

- the underlined attributes are keys of the corresponding relations (`NKey` and `NationName` are both keys of `Nation`)
- $\delta(\pi_{\texttt{NKey}}\ \texttt{Nation}) = \delta(\pi_{\texttt{NKey}}\ \texttt{Customer})$
- $\delta(\pi_{\texttt{CKey}}\ \texttt{Customer}) = \delta(\pi_{\texttt{CKey}}\ \texttt{Order})$
- $\delta(\pi_{\texttt{OKey}}\ \texttt{Order}) = \delta(\pi_{\texttt{OKey}}\ \texttt{LineItem})$

The following statistics apply:

```
T(Nation) = 5
T(Customer) = 10^6
T(Order) = 10^7
T(LineItem) = 2 10^7
V(LineItem, Product) = 10^4
```

Consider the query

```
SELECT *
FROM Nation, Customer, Order, LineItem
WHERE Nation.NationName = "Canada"
      AND Nation.NKey = Customer.NKey
      AND Customer.CKey = Order.CKey
      AND Order.OKey = LineItem.OKey
      AND LineItem.Product = "ABC123"
```

- Show the initial hypergraph that corresponds to the query and indicate the small relations (or small relation).
- Show the steps that the modified INGRES algorithm will make. For each step show the part of the query plan that has been developed up to that point and the

remaining hypergraph. Eventually you should show the left deep join tree that is produced.

- Estimate the size of the intermediate results of the produced left deep tree.
- Is this the left deep tree that leads to the smallest sum of intermediate result sizes? Justify your answer by showing all other left deep join trees and calculating the sizes of their intermediate results.

Extra Space