Database Systems, CSCI 4380-01
Homework # 7 Solutions
Due Thursday April 21, 2011 at 2 pm

Answer the following questions. Turn in a single text or PDF file in the assignment drop box.

Question 1 [20 points]. Suppose you are given a B-tree where each leaf node can address at most 5 and at least 2 tuples. Each internal node can have at most 5 and at least 2 pointers (i.e. between 1-4 key values).

(a) Given this B-tree structure and the given B-tree. Insert the following points: 107, 105, 103, 137, 140. Draw the resulting tree.

(b) Given this B-tree structure and the given B-tree. Delete the following points: 5, 8, 10, 24, 26, 31. Draw the resulting tree.

Note. The sibling pointers at the leaf level are not shown for simplicity.

Question 2 [15 points]. Suppose you are given a B-tree on $R(A, B)$ of height 3 where each node (leaf or internal) contains about 500 entries approximately (key value, pointer pairs). R has a total of 10 million tuples.

Suppose you are given the following queries:

(a) SELECT * FROM R WHERE A = 10 AND B = 20 AND C = 30

The number of nodes of the B-tree are scanned:

We are going to scan the index for the condition A = 10 AND B = 20. Total number of tuples to scan: 2,000 which fits in 4 leaf nodes. Total cost is 2 (root/internal) +4 (leaf).

The number of tuples that are read from the relation is:

We need to read each tuple of R we find in the nodes we scanned from disk to check the C condition. There are 2,000 tuples that need to be read from relation R.

(b) SELECT * FROM R WHERE A >= 1 AND A < 10 AND B = 20

The number of nodes of the B-tree are scanned:

We are going to scan the index for the condition A >=1 AND A < 10. Total number of tuples to scan: 100,000 which fits in 200 leaf nodes. Total cost is 2+200.

When scanning, we will find all the tuples that satisfy the query, i.e. 5,000 tuples.

As we have $SELECT*$, we need to read all the 5,000 tuples from relation R.

(c) SELECT * FROM R WHERE A = 10 AND B >= 1 AND B < 10

The number of nodes of the B-tree are scanned:

We are going to scan the index for the full query condition. Total number of tuples to scan: 8,000 which fits in 16 leaf nodes. Total cost is 2 + 16.

We need to read all the tuples from the relation since we have $SELECT*$, so we need to read all the 8,000 tuples from relation R.

Suppose, the number of tuples satisfying the above conditions are as given below:

Conditions	Number of tuples
A = 10	10,000
B = 20	50,000
C = 30	500
A = 10 and B = 20	2,000
A = 10 AND B = 20 AND C = 30	20
A >= 1 AND A < 10	100,000
A >= 1 AND A < 10 AND B = 20	5,000
B >= 1 AND B < 10	500,000
A = 10 AND B >= 1 AND B < 10	8,000

For each query, assume you are using the B-tree. Write down how many nodes of the B-tree are scanned and how many tuples are read from the relation to answer this query.

Question 3 [10 points]. Suppose you are given a relation R that spans 5,000 pages ($PAGES(R) = 5,000$). What is the cost of sorting this relation if the available memory for the sort is:

(a) $M = 50$ pages

1. Repeatedly fill the M buffers with new tuples from R and sort them, using any main-memory sorting algorithm. Write out 5000/50 = 100 sorted sublist to secondary storage.

Because 100 > 50

2. Repeatedly fill the M buffers with new tuples from R and sort them, using any main-memory sorting algorithm. Write out 100/50 = 2 sorted sublist to secondary storage.

Because 2 < 50

3. Merge the sorted sublists. We allocate one input block to each sorted sublist and one block to the output.

Total cost is 5∗5000 = 25000 pages.

(b) $M = 100$ pages

1. Repeatedly fill the M buffers with new tuples from R and sort them, using any main-memory sorting algorithm. Write out 5000/100 = 50 sorted sublist to secondary storage.

Because 50 < 100

2. Merge the sorted sublists. We allocate one input block to each sorted, and directly write the result to disk.

Total cost is 3∗5000 = 15000 pages.

Show your work.

Question 4 [10 points]. What is the cost of block-nested loop join of R and S where

(a) $PAGES(R) = 5,000$, $PAGES(S) = 500$, $M = 101$.

1. Choose relation R as outer relation and relation S as inner relation: $B(R)+(B(R)B(S))/(M-1) = 5000 + 5000 * 500/(101 - 1) = 30000$.

2. Choose relation S as outer relation and relation R as inner relation: $B(S)+(B(S)B(R))/(M-1) = 500 + 500 * 5000/(101 - 1) = 25500$.

(b) $PAGES(R) = 5,000$, $PAGES(S) = 500$, $M = 1,001$.

1. Choose relation R as outer relation and relation S as inner relation: $B(R)+(B(R)B(S))/(M-1) = 5000 + 5000 * 500/(1001 - 1) = 7500$.

2. Choose relation S as outer relation and relation R as inner relation:

Because the memory size M= 1001 >PAGES(S) = 500, use one pass join;

$B(S) + B(R) = 500 + 5000 = 5500$.

Use the cheapest ordering among R or S (i.e. choosing which relation is outer and which one is the inner relation).

Question 5 [10 points]. Suppose you perform sort based merge join between R and S where $PAGES(R) = 5,000$, $PAGES(S) = 500$, $M = 500$. What is the total cost of the join algorithm (including sort)? Show your work.

Case 1 Using simple Sort-Based Join Algorithm

Because 500*499 = 249500 bigger than both 5000, we can sort relation R with just 2 passes.

And because the size of M is the same as relation S, we can sort relation S with just 1 pass.

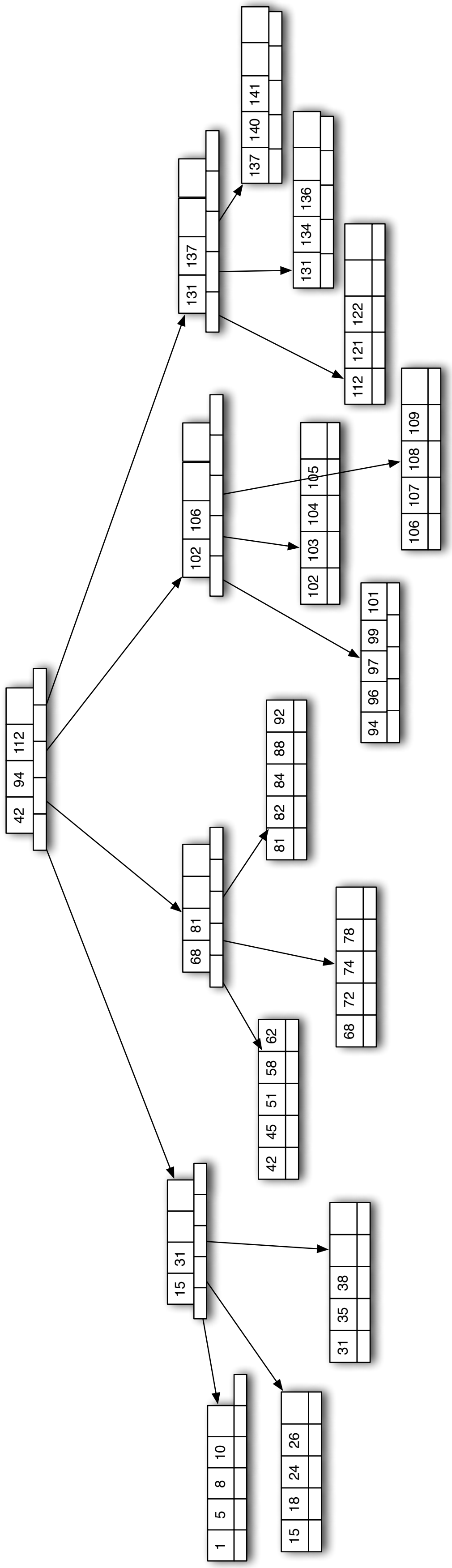
1. The cost of sorting R and writing the result to disk is 4*5000 = 20000 PAGES
2. The cost of sorting S and writing the result to disk is 2*500 = 1000 PAGES
3. The cost of merging the sorted R and S is 5000 + 500 = 5500

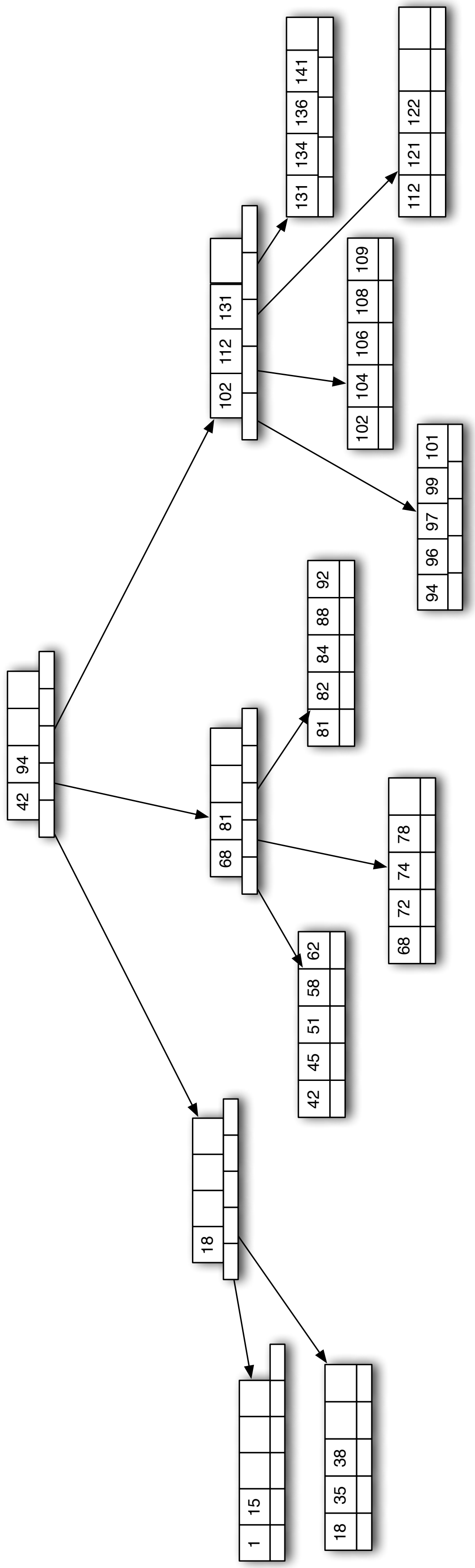
Total cost is $5 * B(R) + 3 * B(S)) = 20000 + 1000 + 5500 = 26500$

Case 2 Using simple Sort-Based Join Algorithm

We can first sort relation S and store the result on disk, then sort relation R with just 2 passes. Becase $500 - 2 > 5000/500 + 1$, we can combine the second phase of sorting the relation R with the join itself.

Total cost is $3(B(R) + B(S)) = 3 * (5000 + 500) = 16500$


42 94 112
15 31
68 81
102 106
131 137
1 5 8 10
15 18 24 26
31 35 38
42 45 51 58 62
68 72 74 78
81 82 84 88 92
94 96 97 99 101
102 103 104 105
106 107 108 109
112 121 122
131 134 136
137 140 141


42 94
18
68 81
102 112 131
1 15
18 35 38
42 45 51 58 62
68 72 74 78
81 82 84 88 92
94 96 97 99 101
102 104 106 108 109
112 121 122
131 134 136 141