Database Systems, CSCI 4380-01
Homework # 7 Answers
Due Tuesday April 17, 2018 at 11 PM

Homework Statement.

This homework is worth 6% of your total grade. If you choose to skip it, Final Exam will be worth 6% more.

This homework is on B-trees and query processing. Please type your answers in a PDF document and submit on Gradescope. No handwritten homeworks please.

Question 1 (15 points). Suppose the usable space on a disk page is 8,000 bytes for indexing. This is the space filled with key values and pointers. A key value and pointer pair is called an entry. The pointers in a leaf node point to tuples in a relation. Pointers in the upper levels point to index nodes at levels below. Each index node is mapped to a disk page and all pointers are the same size: 16 bytes.

You are creating the following indices: index `i1` on `R(A)` and index `i2` on `R(A, B, C)` where attribute A is 4 bytes long, attribute B is 10 bytes long and attribute C is 8 bytes long.

First, compute the capacity of the nodes: maximum number of entries that can be stored in a node/disk page. Recall that each index node (internal or leaf) contains the same type of information and has the same capacity. You may have one extra pointer in each node, but we assume this is part of the header info that is not part of this computation. So you can safely disregard it. Assume each page contains about 75% of maximum number of entries (the root node or one of the nodes in each level may contain fewer nodes).

Given that `Tuples(R)=500,000`, compute the size of indices i1 and i2 (number of nodes at each level). Show your work.

Answer.

Index i1, size of an entry is 4+16=20. So each node stores 8000/20=400 entries max. If nodes are 75%, they will store 300 entries in general.

Given a total of 500,000 tuples:

Leaf level: ceil(500,000/300)= 1,667 nodes Next level: ceil(1667/300)=6 nodes Next level: ceil(6/300)=1 node (root)

Index i2, size of an entry is 4+10+8+16=38 bytes. So each node stores 8000/38=210 entries max. If nodes are 75%, they will store 157 entries in general.

Given a total of 500,000 tuples:

Leaf level: ceil(500,000/157)= 3,185 nodes Next level: ceil(3185/157)= 20 nodes Next level: ceil(20/157)=1 node (root)

Same height, but slight more nodes.

Question 2 (35 points). You are given the following indices and information.

Index	Columns Indexed	Height	#nodes at leaf level
i1	R(A)	2	1,000
i2	R(A,B)	2	2,000
i3	R(C)	2	4,000

`Tuples(R)=800,000`, `Pages(R)=12,000`.

Query	Number of tuples that match
`Q1: select * from R where A=5`	200
`Q2: select * from R where B <= 20`	80
`Q3: select * from R where C='Pickle Rick'`	2,000
`Q4: select * from R where A=5 and B <= 20`	10
`Q5: select * from R where A=5 and C='Pickle Rick'`	120
`Q6: select * from R where B <= 20 and C='Pickle Rick'`	1
`Q7: select * from R where A=5 and B <= 20 and C='Pickle Rick'`	1

For each query, find the cost of answering the query in terms of the number of pages and total number of seeks for each and every one of the index combinations from the list below:

(1) i1 only (2) i2 only (3) i3 only (4) i1 and i3 together (5) i2 and i3 together.

Assume that all leaf nodes can be read sequentially. Internal nodes are read using random I/O. Assume the worst case scenario when matching tuples can be in many different pages. Relation R spans a total of 200 different cylinders, so reading R sequentially would require a total of 200 seeks.

You can exclude an index combination if it does not apply to this question (for example index on A will not help query Q3). Show your work.

For the remaining questions in this homework, you will compute query costs in terms of number of pages, disregarding whether it is random or sequential I/O. However, as an independent exercises, think about whether the disk operations involved in these operation are more likely to be random or more likely to be sequential I/O.

Answer.

All indices have height 2: root, internal and leaf nodes.

i1 has: 800,000/1,000=800 entries per node

i2 has: 800,000/2,000=400 entries per node

i3 has: 800,000/4,000=200 entries per node

- `Q1.` Selection on A, potential indices are i1 and i2. No need to consider any plan with i3. Makes no sense to use a plan with i1 and i2 together because i2 has all the information in i1.

  (1) i1 only: 200 tuples all consecutive in leaf nodes, so they will fit in ceil(200/800)=1 leaf node (or 2 at most if the range is broken). Total index cost is: 3 to 4 nodes, 3 seeks.

  Read matching 200 tuples from disk, in worst case in 200 pages costing 200 seeks.

  Total: 203-204 pages, total seeks: 203.

  (2) i2 only: Scan index for A=5, tuples are all consecutively stored in leaf nodes, fit in (200/400)= 1 leaf node (or at most 2 as before). The cost is identical to i1 cost.

  Total: 203-204 pages, total seeks: 203.

- `Q2.` Selection on B, only potential index is i2. Other indices are useless.

  (2) i2 only: Scan the whole leaf level of i2 because it is not sorted by B. Cost: 2 internal nodes and 2,000 leaf nodes. Seeks: 3 (assume leaf is all sequential I/O).

  Read matching 80 tuples from disk, cost: 80 pages and 80 seeks in the worst case.

  Total: 2003+80=2083 pages total, 83 seeks. (Suprisingly cheaper than the first query in terms of seeks!)

- `Q3.` Selection on C, only potential index is i3

  (3) i3 only: Scan 2 internal nodes and ceil(2,000/200)=10 leaf nodes. In the worst case, 11 leaf nodes if the range is broken down. Total index cost: 12 or 13 nodes, 3 seeks.

  Read all matching 2,000 tuples from 2,000 pages in the worst case. Seeks can be as bad as 2,000 or if the reads are optimized by the disk controller, it could be about 200. I will use 2,000.

  Total cost: 2012 or 2013 pages, 2003 seeks.

  This is clearly not a great index and it will likely not be used as sequential scan on R is better.

- `Q4.` Selection on A and B, potential indices are i1 or i2.

  (1) i1 only: Cost is the same as i1 in Q1 as A=5 is used to scan i1 and then the matching tuples are read.

  Total: 203-204 pages, total seeks: 203.

  (2) i2 only: Scan i2 for A and B conditions together. 10 tuples will fit in likely 1-2 leaf nodes at most, total 3 or 4 nodes with 3 seeks.

  Read 10 the matching tuples from disk, cost of 10 disk pages and 10 seeks.

  Total: 13 or 14 pages, 13 seeks.

  Clearly, i2 is the best index for this query as A and B together reduce the size of the output considerably and i2 is not a much bigger index to scan.

- `Q5.` Selection on A and C, potential indices are i1, i2 and i3

  (1) i1 only: Same as i1 in Q1: Total: 203-204 pages, total seeks: 203.

  (2) i2 only: Same as i2 in Q1: Total: 203-204 pages, total seeks: 203.

  (3) i3 only: Same as i3 in Q3: Total cost: 2012 or 2013 pages, 2003 seeks.

  (4) i1 and i3 together:

  Scan of i1 is the same as i1 in Q1: Total index cost is: 3 to 4 nodes, 3 seeks.

  Scan of i3 is the same as i3 in Q3: Total index cost: 12 or 13 nodes, 3 seeks.

  Scan the matching 120 tuples from disk, 120 pages (and seeks) in the worst case.

  Total: 135 - 137 pages, 126 seeks.

  (5) i2 and i3 together:

  Scan of i2 is the same as i2 in Q1: Total index cost is: 3 to 4 nodes, 3 seeks.

  Scan of i3 is the same as i3 in Q3: Total index cost: 12 or 13 nodes, 3 seeks.

  Scan the matching 120 tuples from disk, 120 pages (and seeks) in the worst case.

  Total: 135 - 137 pages, 126 seeks.

- `Q6.` Selection on B and C, potential indices are i2 and i3

  (2) i2 only: Same as i2 in Q2. Total: 2003+80=2083 pages total, 83 seeks.

  (3) i3 only: Same as i3 in Q3. Total cost: 2012 or 2013 pages, 2003 seeks.

  (5) i2 and i3 together.

  Scan i2 for B, Cost: 2 internal nodes and 2,000 leaf nodes, 2,002 total. Seeks: 3

  Scan i3 for C, Total index cost: 12 or 13 nodes, 3 seeks.

  Read the matching 1 tuple from disk: Cost: 1 page and seek.

  Total (max): 2,016 pages and 7 seeks.

- `Q7.` Selection on A,B and C, potential indices are i1, i2 and i3

  (1) i1 only: Same as i1 in Q1: Total: 203-204 pages, total seeks: 203.

  (2) i2 only: Same as i2 in Q4: Total: 13 or 14 pages, 13 seeks.

  (3) i3 only: Same as i3 in Q3: Total cost: 2012 or 2013 pages, 2003 seeks.

  (4) i1 and i3 together:

  Scan i1 for A condition: Total index cost is: 3 to 4 nodes, 3 seeks.

  Scan i3 for C condition: Total index cost: 12 or 13 nodes, 3 seeks.

  Read matching 120 tuples (for `A=5 and C='Pickle Rick'`), 120 pages and seeks in the worst case.

  Total: 137 pages, 126 seeks in the worst case.

  (5) i2 and i3 together:

  Scan i2 for A and B conditions as in Q4: Total disk cost: 3 or 4 nodes with 3 seeks.

  Scan i3 for C condition: Total index cost: 12 or 13 nodes, 3 seeks.

  Read the matching 1 tuple from disk: Cost: 1 page and seek.

  Total (max): 18 pages and 7 seeks.

Question 3 (15 points). What is the cost of sorting relation `R` given `Pages(R)=12,000` if:

(a) `M=200`

Answer.

Step 1: Read and sort 200 pages at a time.

Cost: 24,000 pages. Creates: 12,000/200 = 60 sorted groups

Step 2: Merge all 60 groups and output.

Cost: 12,000 pages.

Total: 36,000 pages.

(b) `M=40`

Answer.

Step 1: Read and sort 40 pages at a time.

Cost: 24,000 pages. Creates: 12,000/40 = 300 sorted groups

Step 2: Merge all 300 groups, 40 at a time and write back to disk.

Cost: 24,000 pages, Creates: 300/40=8 sorted groups.

Step 2: Repeated. Merge all 8 sorted groups.

Cost: 12,000 pages.

Total: 60,000 pages.

(c) `M=10`

Answer.

Step 1: Read and sort 10 pages at a time.

Cost: 24,000 pages. Creates: 12,000/10 = 1200 sorted groups

Step 2: Merge all 1,200 groups, 10 at a time and write back to disk.

Cost: 24,000 pages, Creates: 1,200/10=120 sorted groups.

Step 2: Merge all 120 groups, 10 at a time and write back to disk.

Cost: 24,000 pages, Creates: 120/10=12 sorted groups.

Step 2: Merge all 12 groups, 10 at a time and write back to disk.

Cost: 24,000 pages, Creates: 12/10=2 sorted groups.

Step 2: Repeated. Merge 2 sorted groups.

Cost: 12,000 pages.

Total: 108,000 pages.

Question 4 (15 points). What is the cost of block-nested loop join of `R` and `S` given `Pages(R)=12,000` and `Pages(S)=2,000` if:

We will only compute the cheaper version with the smaller relation as the outer relation.

(a). `M=201`

Answer. Read S once, Read R: 2,000/200=10 times.

Total cost: 2,000 + 10*12,000 = 122,000

(b). `M=1,001`

Answer. Read S once, Read R: 2,000/1,000=2 times.

Total cost: 2,000 + 2*12,000 = 26,000

(c). `M=2,001`

Answer. Read both relations once.

Total cost: 2,000 + 12,000 = 14,000

Question 5 (20 points). You are given the following B-tree of capacity 4 for an integer value that takes unique values. In this case, each leaf node stores between 2 to 4 key value and tuple pointer pairs. Each internal node stores between 2-4 pointers to nodes below, with key values to distinguish the division points.

Show the resulting B-tree after

(a) adding key values: 33, 15, 37 in this order to the original B-tree

(b) deleting key values: 69, 61, 84 in this order from the original B-tree

Draw the resulting B-tree. You can use the PDF for the given B-tree and make changes by drawing on it using a PDF source, or you can draw the changes and scanning it. We will only allow hand written components in this part of the homework.

Answer.

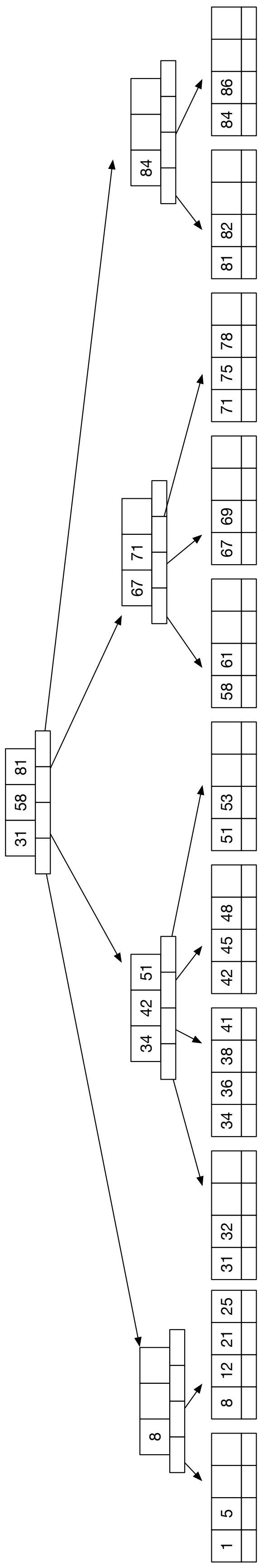


Figure 1: Before any adds or deletes.

Figure 2: After adds to the original B-Tree. Assuming when splitting nodes 3 entries go to the left node. However, putting 3 entries to the right node is also acceptable.

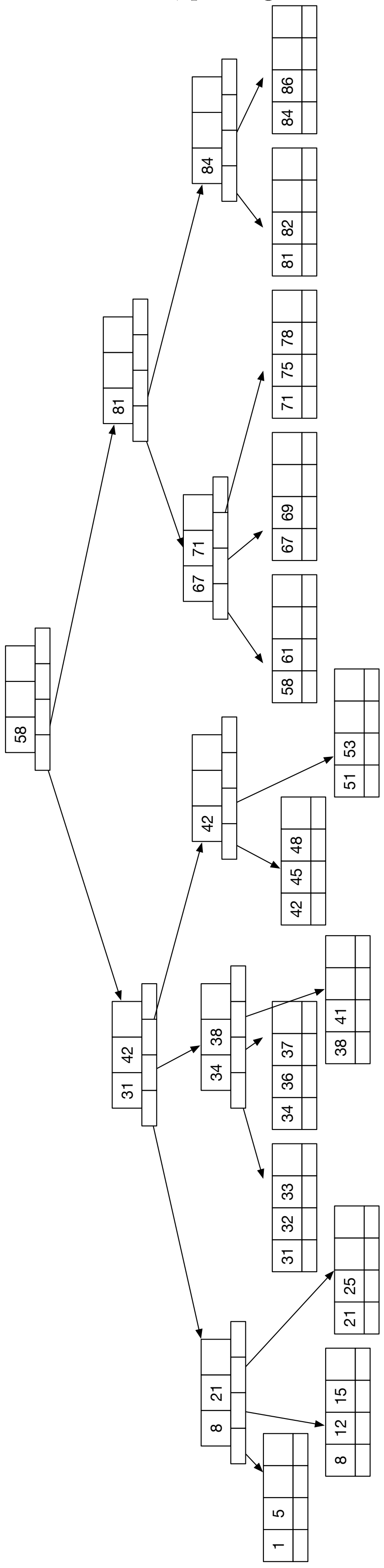
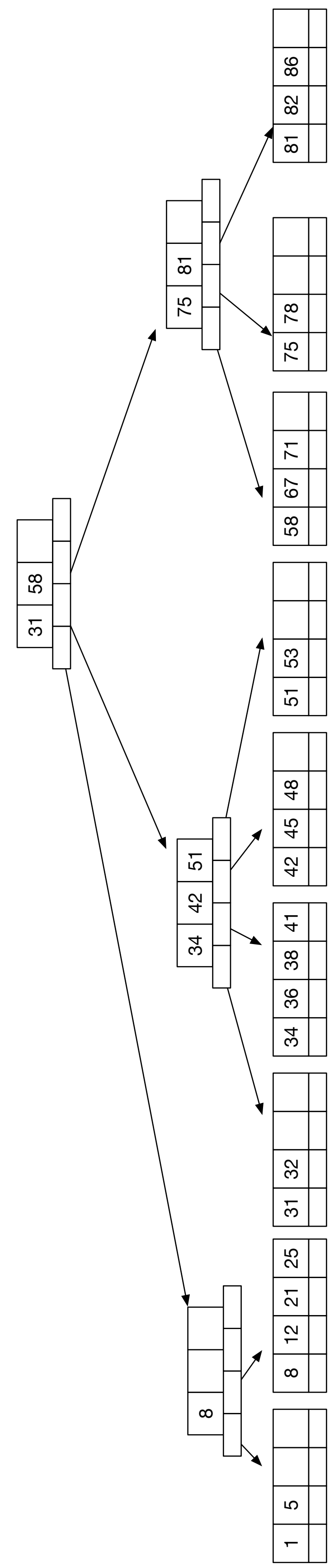


Figure 3: After deletes to the original B-Tree.


31 58
8
34 42 51
75 81
1 5
8 12 21 25
31 32
34 36 38 41
42 45 48
51 53
58 67 71
75 78
81 82 86