# CS 245 Database Systems Principles – Summer 2001

# Assignment 3

- Due Date: Monday July 23rd 2001, 5 pm.
- Submission through
    - o Box Placed outside Gates 412.
    - o Email solution to masood@cs.stanford.edu (only pdf, ps or plain text files).
    - o SITN homework delivery.
- The deadline is hard, No Late days.
- Do not forget to write your Leland Ids at the start of your solution.
- State all assumptions.
- Email questions to cs245-staff@lists.stanford.edu

---

## *Problem 1*                                                 20 Points

Consider an extensible hash structure where buckets can hold up to three records.
Initially, the structure is empty. The hashed key values are the following (in binary):

| | |
|---|---|
| h(Biochemistry) | = 00001 |
| h(Biology) | = 00011 |
| h(Chemistry) | = 00110 |
| h(Computer Science ) | = 01110 |
| h(Economics) | = 01111 |
| h(Electrical Eng ) | = 10001 |
| h(History ) | = 10101 |
| h(Math) | = 10111 |
| h(Philosophy) | = 11000 |
| h(Physics) | = 11001 |
| h(Psychology ) | = 11101 |
| h(Statistics) | = 11111 |

a) First, suppose we insert the values in the order given above. Show the structure after all the values have been inserted.

b) Once again, assume we start with an empty structure. Now suppose we insert all the values in the following order:

Physics, Chemistry, Biology, Math, Electrical Eng, Computer Science, Psychology, Statistics, History, Philosophy, Economics, Biochemistry.

Show the structure after all the values have been inserted.

## Problem 2                                                                    20 Points
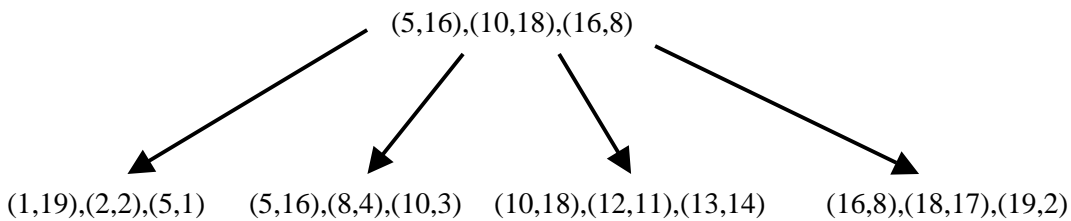
Consider a linear hash structure where buckets can hold up to two records. Initially the structure is empty. Assume that the utilization threshold value is 100% (i.e., when the average number of keys per non-overflow slot is greater that 1, we allocate another bucket). The hashed keys are the same as in Problem 1.

a) Show the structure after all the values have been inserted in the order they are given (Problem 1 a).

## Problem 3                                                                    15 Points

Consider the following Multi-key B Tree over keys that are points in two dimensions (i.e. x,y co-ordinates). This tree has one root and four leaf nodes.



(5,16),(10,18),(16,8)

(1,19),(2,2),(5,1)     (5,16),(8,4),(10,3)     (10,18),(12,11),(13,14)     (16,8),(18,17),(19,2)
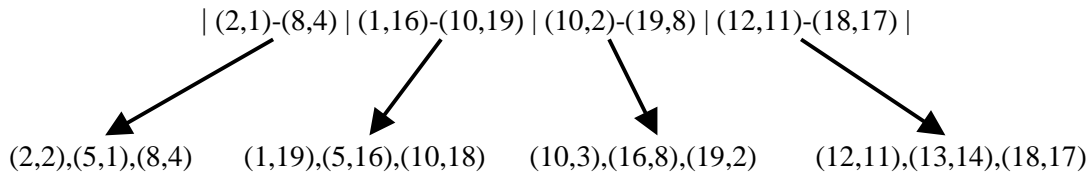
Note: The ordering is determined by first comparing the x co-ordinate. If the x co-ordinates are identical, then the order is determined by comparing the y co-ordinates. For example, (2,1) < (12,2) and (12,2) < (12,4).

In this problem, rectangles in two dimensions are represented by a pair of two-dimensional points. A rectangle representation has the form: (x1,y1)-(x2,y2). The two points define the lower-left and upper-right corners of the rectangle, respectively.

a) How many nodes (blocks) of the index will be read while executing the following lookups:

i)      X <= 10
ii)     X <= 9
iii)    Y <= 10
iv)     (X,Y) in the rectangle defined by (5,5)-(15,15).
v)      (X,Y) in the rectangle defined by (2,2)-(8,17).

vi)     (X,Y) in the rectangle defined by (4,2)-(19,8).

Now Consider a R-Tree index over the same set of points:

| (2,1)-(8,4) | (1,16)-(10,19) | (10,2)-(19,8) | (12,11)-(18,17) |

(2,2),(5,1),(8,4)     (1,19),(5,16),(10,18)     (10,3),(16,8),(19,2)     (12,11),(13,14),(18,17)

Recall that the non-leaf nodes of an R-Tree have one key per child pointer. Keys in non-leaf nodes are rectangles (represented by a pair of points).

   b)  How many nodes (blocks) of the R-Tree index will be read while executing the six lookups given in part (a)?

## Problem 4                                                              10 Points

Given: M = # of memory buffer pages = 20 blocks
        Relation R with B(R) = # of blocks = 520

Lets say we want to sort R using the external merge sort algorithm (without the heap-sort in memory optimization).

   a)  How long will the runs be after the first pass?
   b)  How many runs will there be after the first pass?
   c)  Can the sort be performed in two passes?
   d)  If not, how much memory would we need in order to sort R in two passes?

## Problem 5                                                              20 Points

Trace the execution of the merging phase of the merge-join algorithm. Assume that the two relations have already been fully sorted. Show a sequence of values for "i" and "j" as used in the pseudo-code in lecture notes 6. Indicate which (i, j) pairs generate a result (joined) tuple.

The sorted relations R and S have the following keys:

R: 1, 2, 3, 3, 4, 5, 5, 6, 8
S: 1, 1, 3, 4, 5, 5, 7, 8, 9, 10

Consider a hash-join of two relations R and S having $B(R) = 1000$ and $B(S) = 500$.

The values in R and S are skewed such that the hash function assigns twice as many tuples to even-numbered hash buckets as to odd-numbered buckets.

How much memory would be required to perform the join in two passes?

---

**Solution**

In a two-pass hash join we use the smaller relation, S, for building the buckets. With $M$ buffers we partition S into roughly $M$ buckets. For the join to complete in two passes, each bucket of S must fit in memory.

If the distribution were uniform, each bucket would hold $B(S)/M = 500/M$ blocks, and we would need

$$\frac{500}{M} \le M \quad\Rightarrow\quad M \ge \sqrt{500} \approx 22.4$$

i.e. about 23 buffers.

With the given skew, an even-numbered bucket receives twice as many tuples as an odd-numbered bucket. If half of the buckets are even and half odd, the total "weight" is

$$\tfrac{M}{2}(2) + \tfrac{M}{2}(1) = 1.5M,$$

so an even bucket holds a fraction $\dfrac{2}{1.5M} = \dfrac{4}{3M}$ of the tuples. The largest (even) bucket of S therefore contains

$$\frac{4}{3}\cdot\frac{500}{M} = \frac{2000}{3M} \text{ blocks.}$$

For this to fit in memory we need

$$\frac{2000}{3M} \le M \quad\Rightarrow\quad M^2 \ge \frac{2000}{3} \approx 667 \quad\Rightarrow\quad M \ge 26.$$

So about **26 blocks of memory** are required (compared with ~23 in the unskewed case).