Sample Solutions

**Question 1** (22 points) Short questions and answers

(1) In a clustered index, records are stored in the same order as the keys in the index.
(2) No, since hash table doesn’t support range queries.
(3) Spatial data: R-tree; Text data: inverted index.
(4) In the last merge phase of the sorts, combine it with the join phase of the two relations.
(5) UNION: remove duplicates. UNION ALL: Keep all records of R and S without removing duplicates.
(6) Each operator has three functions, open(), getNext(), and close(). Advantages: in a query tree, each operator can just use these functions of its children without considering the type of these operators.
(7) • Left-deep tree: ((R JOIN S) JOIN T) JOIN W
  • Right-deep tree: R JOIN (S JOIN (T JOIN W))
  • Linear tree: ((R JOIN S) JOIN T) JOIN W
  • Bushy tree: (R JOIN S) JOIN (T JOIN W)

**Question 2** (6 pts) Transformation rules in Query Optimization

R3 -> R1 -> R2

**Question 3** (12 pts) Histograms

(1) (0,20]: 8, (20,40]: 16, (40,60]: 24, (60,80]: 16, (80,100]: 16
(2) \(18/20 * 16 + 24 + 16 + 4/20*16 = 57.6\)
(3) Sort the records based on “price.” For each unique price value, keep its count. After that, scan the sorted count frequency, and decide the boundaries of the buckets.

**Question 4** (20 pts) Merge Sort

(1) The OS’s buffer manager does not utilize the special characteristics of the merge operations to do buffer replacement.
(2) 16 * 3 * 110 (all random IOs) + (15 + 11) * 10 (writes in the first pass)
  \[1,760 \text{ (R)} + 260 \text{ (W)} + 1,760 \text{ (R)} + 1,760 \text{ (W)} = 5,540 \text{ms}\]
(3) 1st pass: (15 + 1) * 110 (random reads) + (15 + 11) * 10 (sequential writes)
  2nd pass:
  Reads: (15 + 2) * 5 * 10 (2 pages each IO) + (15 + 1) * 10 (1 page each IO)
Writes: \((15 + 1) \times 1 \times 110\) (random IOs)

3rd pass:

Reads: \((15 + 5) \times 11 \times 2\) (5 pages each IO)

Writes: \((15 + 1) \times 1 \times 110\) (all random IOs)

\[1,760 \text{ (R)} + 260 \text{ (W)} + 1,010 \text{ (R)} + 1,760 \text{ (W)} + 440 \text{ (R)} + 1,760 \text{ (W)} = 6,990\text{ms}\]

**Question 5 (20 pts) Join**

1. \(100 + 100 \times 200 = 20,100\); 3 pages
2. \(100 + 100/50 \times 200 = 500\)
3. \((100 + 200) \times 3 = 900; \sqrt{100} + 2 = 12\)
4. Use a portion of the buffer for an in-memory hash table for R. Use the rest of the buffer for grace-style partition buffering.

**Question 6 (20 pts) System-R Optimizer**

1. E: eid, did
   D: did, pid
   P: pid
   E join D: eid, pid
   D join P: none
   E join D join P: eid

2. Scan
   Scan on the B+ tree of the salary attribute for “salary = 50K”
   Scan on the B+ tree of the eid attribute (interesting order on “eid”)

3. Start from single relations, and consider all access methods. Apply selections as early as possible.

   For each subset of K relations, consider all possibilities of joining one of the relations with a best plan of the remaining k-1 relations. Keep the best plan for this subset.

   Consider interesting orders for each subset of relations. For each interesting order, keep the best plan for a set of relations.

   Only consider left-deep trees. Apply group by, order by, distinct operators at the end.