Question 1

Consider a relation R with 1600 pages, and we want to sort it using external merge sort. Assume that the DBMS uses quicksort for in-memory sorting (i.e., generate sorted runs of memory size in the first pass). Let B denote the number of buffers. What is the I/O cost to sort R given B is sufficient to sort R in two passes (i.e., generate run in one pass, and then merge these runs in a second pass)?
Answer: 6400
Solution: I/O for generating sorted runs = $|R| + |R| = 3200$
I/O for merging sorted runs = $|R| + |R| = 3200$
Total I/O = $3200 + 3200 = 6400$

Question 2

Consider the same setting in Question 1. What is the smallest number of buffers B that the DBMS can sort R using only three passes (i.e., generate sorted runs, and then need 2 more passes to merge the sorted runs)?
Answer: 13
Solution: B buffers for generating sorted runs, and $B - 1$ buffers for merging sorted runs.
Therefore, $B \times B - 1 \times B - 1 \geq 1600$ is the inequality needed here.
If $B = 12$, Result is 1452 which is not enough to handle 3-pass merge sort.
If $B = 13$, Result is 1872, which is more than enough to handle the 3-pass merge sort.

Question 3

Consider relations R(a, b) and S(a, c, d) to be joined on the common attribute a.
- There are B = 12 buffer pages
- Table R comprises M = 200 pages with 80 tuples per page
- Table S comprises N = 30 pages with 40 tuples per page

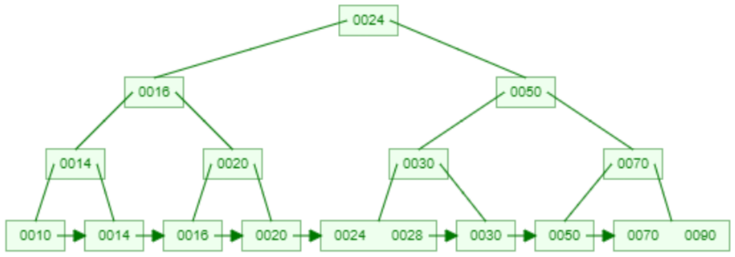
What is the minimal cost (in terms of number of page accesses) to join R and S under Block Nested Loops Join? You should follow the approach taught in the lecture, i.e., you will need one buffer block to hold the evolving output block and one input block to hold the current input block of the inner relation. Also, ignore the cost of writing out the join results.
Answer: 630
Solution: I/O with S as the outer relation = $\frac{|S|}{B-2} \times |R| + |S| = 630$
I/O with R as the outer relation = $\frac{|R|}{B-2} \times |S| + |R| = 800$
Thus, we take the minimum of 630.

Question 4

Consider the same setting in Question 3. What is the minimal cost (in terms of number of page accesses) to join R and S under GRACE Hash Join (as taught in the lecture)? Assume that the tuples are uniformly distributed across all partitions.
Answer: 690
Solution: I/O is simply $2|R| + 2|S| + |R| + |S| = 690$

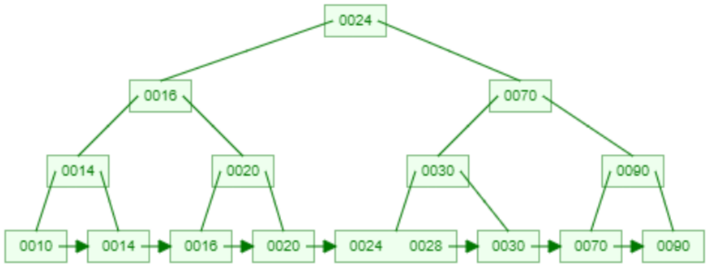
Question 5

Insert the following 10 keys into an initially empty B+-tree of order 1:
$10, 30, 50, 70, 90, 14, 16, 20, 28, 24$
What is the height of the tree?
Answer: 4
Solution:



Question 6

Consider the same setting in Question 5. Delete key 50 from the constructed B+-tree in Question 5. Consider the element/key 90. It appears X times in the tree. What is X?
Answer: 2
Solution:



Question 7

Consider a linear hash table that is initially empty. After inserting some key values, the index now has buckets labelled from 000 to 101 (in binary). Which bucket (in binary) should a key with hash value 111 (binary) be sent to?
Answer: 011
Solution: 111 should belong under bucket 11, aka bucket 011 given that the bucket for 111 is still not available yet

Question 8

Consider a linear hash table whose buckets can hold up to 3 records. Suppose 18 records have been inserted into the hash table. Assume the splitting criterion as discussed in the lecture, i.e., split whenever there is an overflow. What is the smallest number of buckets required (include overflow buckets, if necessary) to hold these records in the best case?
Answer: 6
Solution: Simply, have each bucket hold 3 records. Thus, the number of buckets is simply $\frac{18}{3} = 6$

Question 9

A certain disk unit has 10 surfaces, each with 100 tracks. Each track is divided into 1000 sectors, and a sector holds 2 KB. A page consists of 4 sectors. The disk rotates at 6000 rpm. The average seek time is 15ms. What is the time (in ms; up to 2 decimal points) to retrieve one random page? (NOTE: 1KB = 1024 bytes)
Answer: $20.04ms$
Solution: Time = Seek Time + Rotational Delay + Transfer Time
Rotational Time = $\frac{60}{6000} = 0.01s = 10ms$
Rotational Delay = $\frac{10}{2} = 5ms$
Rotational Time for 1 sector = $\frac{10}{1000} = 0.01ms$
Thus, $15 + 5 + 0.01 \times 4 = 20.04ms$ which is the answer.

Question 10

Consider the same disk in Question 9. Suppose we want to store relation R(A, B, C) on the disk as a contiguous sequential file ordered on the primary key attribute A. Suppose R has 1,000,000 fixed size records. The records in R are 200 bytes in size, and attribute A is 12 bytes (fixed size). Records do not span across sectors. We want to build a sparse B+-tree on A (one pointer per page of R). Suppose a pointer to a track requires 2 bytes. How many bytes (round up to the nearest byte) are required to represent a pointer to a page?
Answer: 3
Solution: Each sector contains $\frac{2048}{200} = 10$ records and each page contains $10 \times 4 = 40$ records.
There are $\frac{1000000}{40} = 25000$ pages and $\frac{25000}{100} = 250$ pages per track.
As a result, a page can be represented in 8 bits given $2^8 = 256$.
Also, to locate a page, need 1 byte for track number and 1 byte for the page.
Thus, a total of 3 bytes are needed.

Question 11

Consider the same setting as Question 10. Assume the B+-tree nodes are of the same size as a data page, and that all nodes are packed as full as possible. What is the height of the B+-tree?
Answer: w
Solution: $2d \times 12 + (2d + 1) \times 3 < 8096$ given 12 bytes for attribute A and 3 bytes for the page
As a result, $d < 269.67$ and thus, $d = 269$ and that the order is $2 \times 269 = 538$
1 level is insufficient as it only can contain 538 nodes
2 levels is sufficient as it can contain $538 \times (538 - 1) = 288906 > 25000$ nodes

Question 12

It is generally better to organize a file cylinder by cylinder instead of surface by surface because organization by cylinder can
Answer: reduce the head movement when reading the file.
Solution: Reading between cylinders is not advisable due to massive amount of read head movement needed.

Question 13

Consider a system where most of the applications use small files. Which of the following is not a valid reason for avoiding large disk pages (e.g., one track)?
Answer: Lengthens the seek time during disk access
Solution: The seek time remains the same irregardless of disk page size.

Question 14

Consider the following sequence of numbers:
$22, 44, 7, 39, 49, 12, 89, 10, 66, 55, 50, 67, 40, 46$
Suppose we use replacement selection to generate sorted runs (as taught in lecture). Moreover, suppose each page has only 1 record. Let the number of buffer pages be 4. What is the length of the longest run?
Answer: 7
Solution:
Given 4 Input Buffers.

Next Tuple	**B1**	**B2**	**B3**	textbfB4	Runs
49	22	44	7	39	()
12	22	44	49	39	(7)
89	**12**	44	49	39	(7, 22)
10	**12**	44	49	89	(7, 22, 39)
66	**12**	**10**	49	89	(7, 22, 39, 44)
55	**12**	**10**	66	89	(7, 22, 39, 44, 49)
50	**12**	**10**	**55**	89	(7, 22, 39, 44, 49, 66)
67	**12**	**10**	**55**	**50**	(7, 22, 39, 44, 49, 66, 89)

Run Produced: $(7, 22, 39, 44, 49, 66, 89)$

Next Tuple	**B1**	**B2**	**B3**	textbfB4	Runs
67	12	10	55	50	()
40	12	67	55	50	(10)
46	40	67	55	50	(10, 12)
	46	67	55	50	(10, 12, 40)

Run Produced: $(10, 12, 40, 46, 50, 55, 67)$

Question 15

Referring to Question 14. What is the number of sorted runs generated?
Answer: 2

Question 16

Consider the following three statements. Which of these are true?
I) Sorting via a clustered index will always be better than applying the external multi-way sort algorithm.
II) Sorting via an unclustered index will always be worse than applying the external multi-way sort algorithm.
III) Sorting via a clustered index will not be inferior to sorting via an unclustered index.
Answer: E
Solution: II may not always be true. This is because if your unclustered index happens to be in perfect order, sorting via external sort may actually be become worse.

Question 17

Which of the following are reasonable criterion/criteria to be used for determining when to split a bucket in linear hashing?
A. Whenever a bucket is full
B. Whenever overall space utilization is 70%
C. Whenever bucket pointed to by the “next” pointer overflows
D. A & B E. B & C F. A & C G. A, B & C
Answer: D
Solution: Only C does not make sense as it is possible for the first bucket to keep overflowing and as a result, it will only split twice and cause massive overflowing in the first bucket. Consider the example $1, 3, 5, 6, 7, 9, 11, 13$ where the maximum number of records per bucket is 2.

Question 18

Consider the join of two relations R and S. Suppose R and S are of the same size. What is the maximum size of R (and therefore S) for the two relations to be joined using GRACE hash join in two passes (as presented in the lecture) if we have a buffer pool of 34 pages? Assume the ideal case that every partition has the same number of tuples.
Answer: 1056
Solution: $B - 1 \times B - 2 = 33 \times 32 = 1056$

Question 19

Consider the join of two relations R and S. Let $|R| = |S| = 1000$. Assume that the join can be performed in two passes using GRACE hash join (as in the lecture). Suppose we want to create the minimum number of partitions during the partitioning phase. Moreover, assume that we use one buffer for input, and the remaining buffers are split evenly for output partitions. In addition, for the joining phase, we do not need an output buffer as results are returned directly to users (without storing the output). Let the buffer size be 201 pages. What is the total number of seeks required to complete the join processing?
Answer: 70
Solution:
Building Phase (partition R tuples):
1) Read 'R' 5 times, 200 pages each (5 seeks)
2) Each time a partition's 40-page output is full, write out. Repeat 5 times for each partition (5 x 5 seeks)
3) Repeat the same for partitioning 'S' (another 5 + 5 * 5)
Joining Phase
1) Load a 200-page partition of R fully (one seek)
2) Read a 200-page partition of S page by page in sequential order (one seek)
3) Repeat the above for the 5 partitions (total 5 + 5 seeks)

Question 20

Which of the following statements is correct (include output cost)?
A. The minimum cost to join two tables R and S (no restrictions on join algorithms, selectivities, memory size or output size, etc) is at least |R| + |S| where |R| and |S| denote the number of pages of R and S respectively.
B. The cost to perform an intersection of two tables is the same as that to union the two tables.
C. We can use an index-only approach to compute an arbitrary aggregate query if the attribute to be aggregated appears as part of the key in the index.
D. A & C E. A & B F. None of A, B & C G. A, B & C
Answer: *F*
Solution:
A is wrong as the minimum cost is $|R| + 1$ (Sort Merge Join)
B is wrong as cost for intersection is $|R| \bowtie |S|$ and cost for union is $|R| + |S|$
C is wrong as it is insufficient if it's just a part of the key in the index. In fact, it should be the entire key in the index, not just a part of it.

Question 21