Written Assignment #3
CAS CS 460: Introduction to Database Systems

Spring 2021

Due: Monday, April 5, 2021 at 11:59PM on Gradescope.

Problem 1. (20 points)

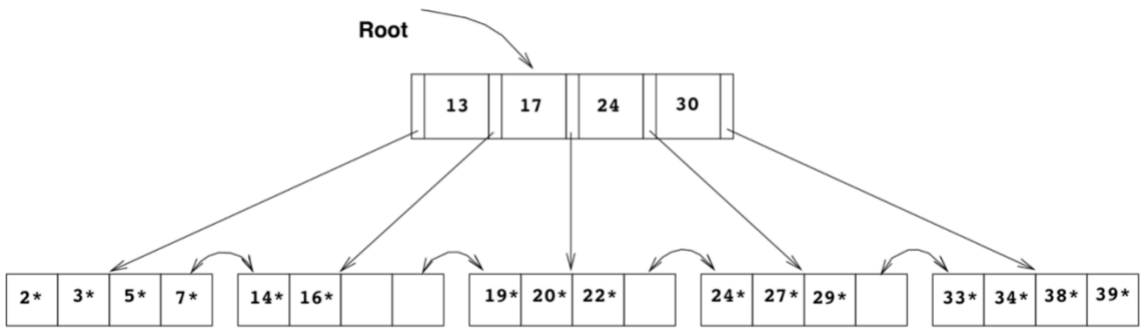
Consider a hard disk drive that has 5 double-sided platters, each surface has 1000 tracks, each track has 256 sectors of size 1024 bytes. Each block (disk page) comprises of 8 sectors. The seek time between adjacent tracks in 1ms and the average seek time between two random tracks is 25ms. The disk rotates at a speed of 7200 rpm (revolutions per minute).

Assume that we have a file of size 1 MB and it contains 2048 equal-sized records.

1. What is the size of a block? How many records fit in a block? How many blocks are required to store the entire file?
2. What is the capacity of each cylinder? How many records can be stored on the disk (total)?
3. If the file is stored "sequentially", how long will it take to read the whole file? Assume that for sequential writes data are written in adjacent cylinders (or tracks) once a cylinder (track) is full.
4. If the blocks in the file are spread "randomly" across the disk, how long will it take to read the whole file?

Problem 2. (20 points)

Consider the following B+-tree instance that has order d=2 (this means that the maximum number of entries per node is 4 and maximum number of pointers is 5).



1) Show the state of the B+-tree after you have inserted the data entries with keys: 9, 15, 35, 23, 18

2) Show the state of the B+-tree after you delete the following keys on the *original* tree shown above: 16, 19, 20

3) Assume that you know in advance the total number of keys that will be inserted into a tree index. In that case, would you consider using ISAM instead of B+-tree for indexing? Explain.

Problem 3. (30 points)

Assume that you have just built a B+ tree index using Alternative (2) on a heap file containing 40,000 records. The index is un-clustered. The key field for this B+ tree index is a 40-byte string, and it is a candidate key. Pointers (i.e., record ids and page ids) are (at most) 10-byte values. The size of one disk page is 1000 bytes. Also, assume that every page (except maybe the last page at each level and the root) are about 67% full (node utilization is ~67%).

1. How many levels does the resulting tree have?

2. For each level of the tree, how many nodes are at that level?

3. How many levels would the resulting tree have if key compression is used and it reduces the average size of each key in an entry to 10 bytes?

4. Consider a range query that has selectivity 1% (retrieves 1% of the total records). Each record has size 100 bytes. Estimate how many disk I/Os you need to perform in order to retrieve all the records. Explain.

5. Consider the previous query with 1% selectivity, but now assume that the index is a clustered index and the records again are 100 bytes each. How many disk I/Os you need to perform to retrieve the records now?

Problem 4. (30 points)

1) Suppose that we are using extensible hashing on a file that contains records with the following search key values:

(449, 124, 654, 831, 1016, 176, 285, 468, 615, 340, 331, 135, 667, 818, 117, 429)

Load these values into a file in the given order using extensible hashing. Assume that every block (bucket) of the hash index can store up to four (4) values. Show the structure of the hash index after every 8 insertions, and the global and local depths. Use the hash function: h(K) = K mod 128 and then apply the extensible hashing technique. Notice that, using this function, every number is mapped first to a number between 0 and 127 and then we take its binary representation. Then, the extensible hashing technique is applied on the binary representation. Furthermore, initially, you start with a single bucket and a single pointer and the global and local depths are zero (0).

2) Consider the Linear Hashing method. Answer the following questions:

i) How does Linear Hashing provide an average-case search cost of only slightly more than one disk I/O, given that overflow buckets are part of its data structure?

ii) Does Linear Hashing guarantee at most one disk access to retrieve a record with a given key value?