CS3223 Tutorial 1: Storage & Indexing

Wk 3, Sem 2, 2022/23

Questions to be discussed: 2, 4, 5 & 8.

1. (Exercise 9.6, R&G)

   Consider a disk with a sector size of 512 bytes, 2000 tracks per surface, 50 sectors per track, five double-sided platters, and an average seek time of 10 msec. Assume that the disk platters rotate at 5400 rpm (revolutions per minute), the disk can read/write from only one head at a time, and a block size of 1024 bytes is chosen.

   Suppose that a file containing 100,000 records of 100 bytes each is to be stored on such a disk and that no record is allowed to span two blocks.

   (a) How many records fit onto a block?

   (b) How many blocks are required to store the entire file? If the file is arranged sequentially on the disk, how many surfaces are needed?

   (c) How many records of 100 bytes can be stored using this disk?

   (d) If pages are stored sequentially on disk, with page 1 on block 1 of track 1, what page is stored on block 1 of track 1 on the next disk surface?

   (e) What time is required to read a file containing 100,000 records of 100 bytes each sequentially?

   (f) What is the time required to read a file containing 100,000 records of 100 bytes each in a random order? To read a record, the block containing the record has to be fetched from disk. Assume that each block request incurs the average seek time and rotational delay.

2. Consider a buffer pool with a total of 10 buffer frames (numbered 0 to 9). Suppose that an application needs to make three repeated sequential scans of a file that has a total of 11 pages (p_1 to p_{11}) resulting in the following sequence of page requests:

   $$\underbrace{p_1, p_2, \cdots, p_{11}}_{1^{st}\text{ scan}}, \underbrace{p_1, p_2, \cdots, p_{11}}_{2^{nd}\text{ scan}}, \underbrace{p_1, p_2, \cdots, p_{11}}_{3^{rd}\text{ scan}}.$$

   Assume the following for this question:

   1. All the buffer frames are initially in the free list.
   2. Access to a page p requires a disk page read if p is not in the buffer pool.
   3. For each accessed page p, the application always unpins p before making the next page request (i.e., there are no pinned pages in the buffer pool when a page request is issued).

   (a) How many disk page reads are incurred under the LRU replacement policy?

   (b) How many disk page reads are incurred under the Clock replacement policy?

   (c) How many disk page reads are incurred under the MRU replacement policy?

3. Consider a B^+-tree index with a page size of 8192 bytes, 4-bytes key and 8-bytes disk page address.

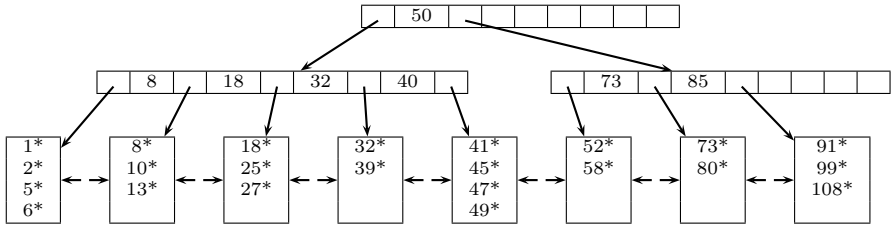
   (a) What is the maximum order of the index?

   (b) Assume that the order of the index is given by your answer for part (a) and the height of the index is 3 (i.e., the index has 3 levels of internal nodes).

      i. What is the minimum number of leaf nodes in the index?

      ii. What is the maximum number of leaf nodes in the index?

4. Consider the B$^+$-tree index of order d=2 shown below.

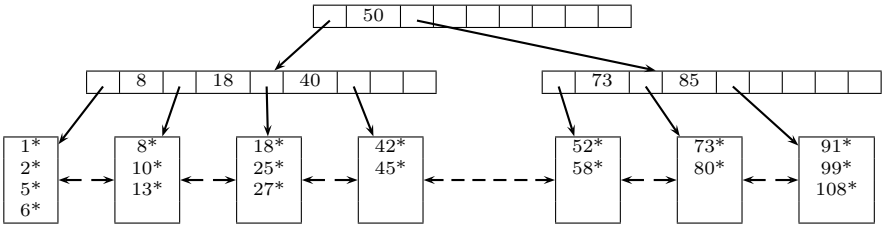


In this question, assume that an overflowed node is always split.

(a) Show the B$^+$-tree index that would result from inserting a data entry with key 3 into this index.

(b) Show the B$^+$-tree index that would result from inserting a data entry with key 48 into the B$^+$-tree index obtained from part (a).

5. Consider the B$^+$-tree index of order d=2 shown below.



Show the B$^+$-tree index that would result from deleting the data entry with key 80 from this index under each of the following scenarios:

(a) Redistribution is used whenever possible.

(b) Redistribution is not used for leaf nodes but it is used whenever possible for internal nodes.

(c) Redistribution is not used for both leaf and internal nodes.

6. In this question, assume that an overflowed node is always split.

(a) Show the B$^+$-tree index (of order d = 2) constructed by inserting the following sequence of 14 data entries one at a time:

$$11*, 5*, 3*, 31*, 19*, 2*, 7*, 17*, 23*, 29*, 4*, 50*, 40*, 44*$$

(b) Show the B$^+$-tree index (of order d = 2) constructed by bulk-loading the same set of data entries in (a).

7. This question considers the update cost for handling overflowed nodes in B$^+$-tree indexes.

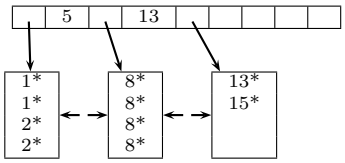
(a) What is the maximum number of nodes that need to be updated when an overflowed leaf node is resolved using redistribution?

(b) What is the maximum number of nodes that need to be updated when an overflowed leaf node is split without any overflow propagation?

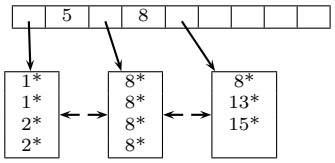
(c) What is the maximum number of nodes that need to be updated when an overflowed internal node is resolved using redistribution?

(d) What is the maximum number of nodes that need to be updated when an overflowed internal node is split without any overflow propagation?

8. This question considers an issue with non-unique B^+-tree indexes. Consider the following non-unique B^+-tree index with order = 2 shown below.



Suppose that we need to insert a new data entry with a key value of 8 into this index. Using redistribution to handle the overflowed leaf node caused by the insertion will result in the following index:



However, the search algorithm discussed in class is no longer sound if the same key value could be stored in multiple leaf nodes (e.g., searching for k=8 will search only the third leaf node); thus, we will need to modify the logic of the search algorithm for non-unique B^+-tree indexes.

Suggest a way to handle duplicate key values that does not require modifying the search algorithm.