Storage and Indexing

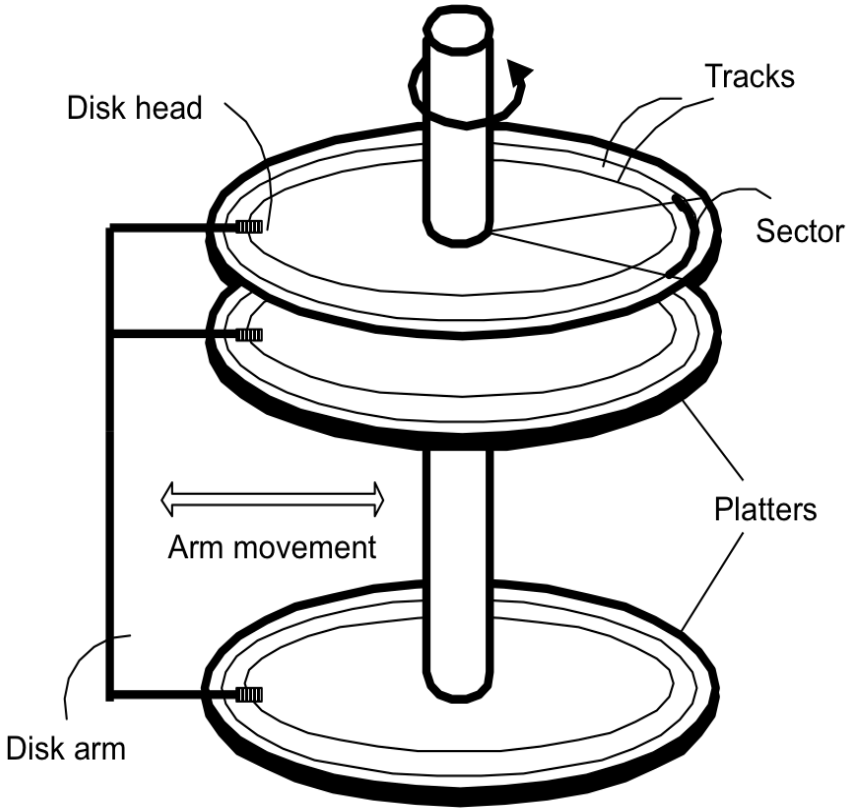
Discussion 9

Question 1

Consider a disk with sector size of 512 bytes, 2000 tracks per surface, 50 sectors per track, five double-sided platters and average seek time of 10 msec. Suppose a page size of 1024 bytes is chosen. Suppose 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 pages

Question 1

- How many pages fit in a track?
- What is the disk capacity in bytes?
- How many records fit onto a page?
- How many pages are required to store the entire file?



Given:
Page size = 1024 bytes
Sector size = 512 bytes
Number of records = 100,000
Size of a record = 100 bytes
Number of tracks/surface = 2000

bytes/track = bytes/sector × sectors/track = 512 × 50 = 25K
bytes/surface = bytes/track × tracks/surface = 25K × 2000 = 50, 000K
bytes/disk = bytes/surface × surfaces/disk = 50,000K × 5 × 2 = 500,000K

1. Number of pages in a track = 25600 / 1024 = 25

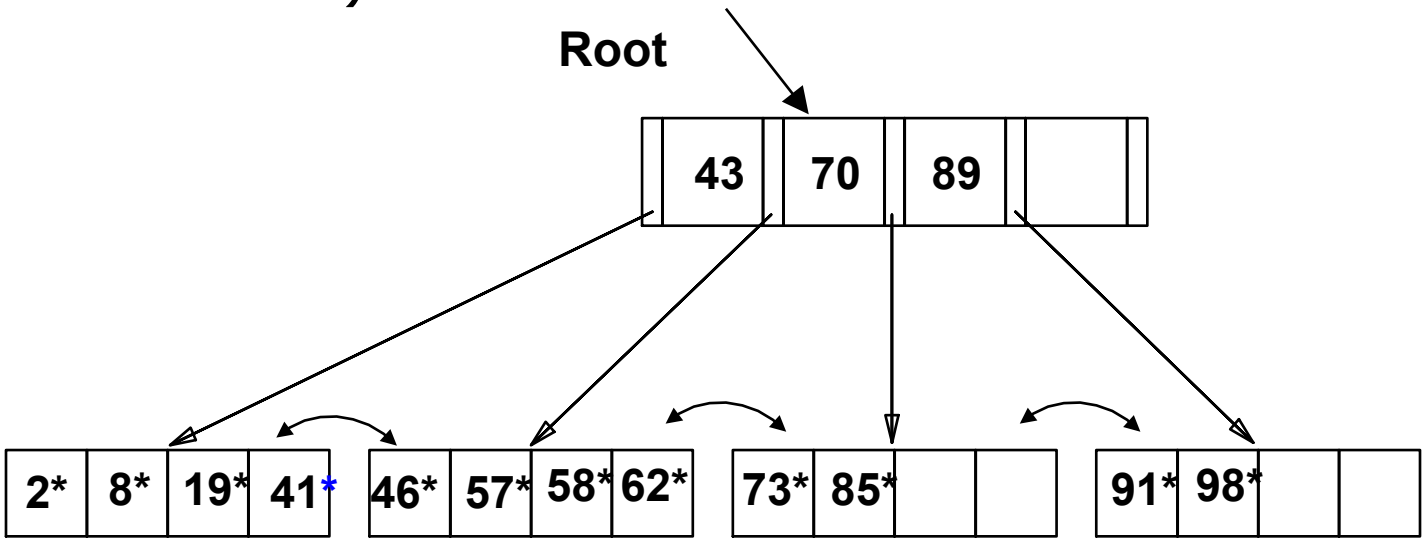
2. Disk capacity = 500,000K

3. records/page = Size of page/ size of record = 1024/100 > 10. So 10 records can fit into a page

4. Number of pages required for the entire file = 100,000/10 = 10000 pages

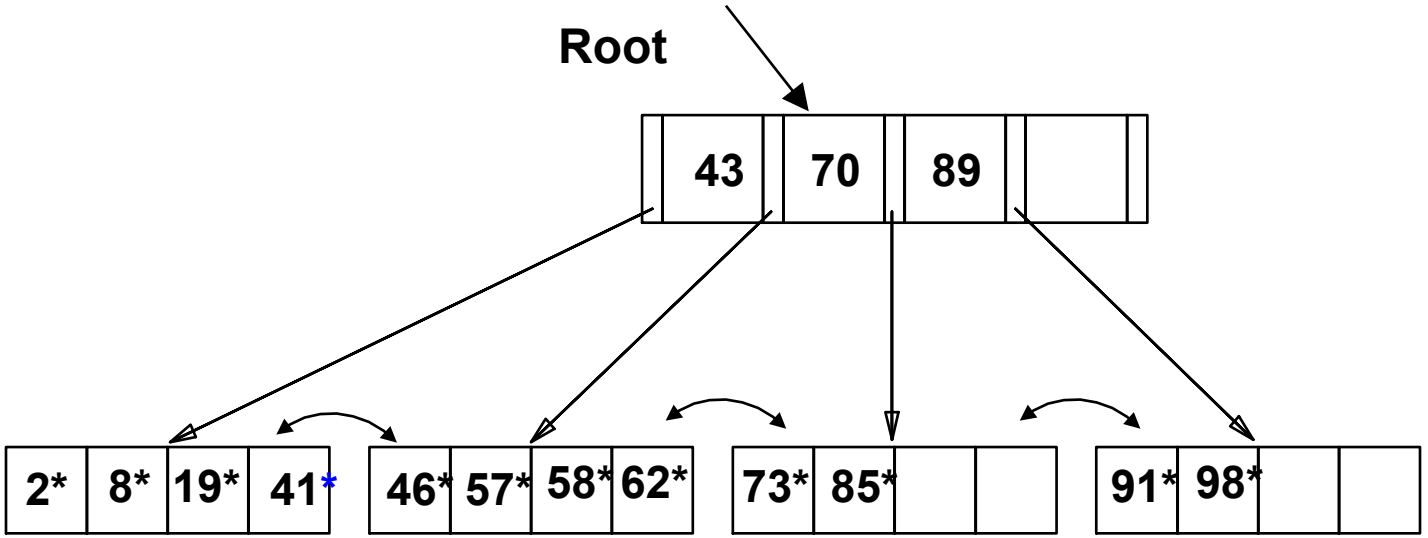
Question 2: B+ trees

Consider the following B+-tree, and assume the convention that the left pointer points to values that are strictly less than the key value. Each node in this B+-tree can hold up to four entries (i.e., the order of the tree is 2).



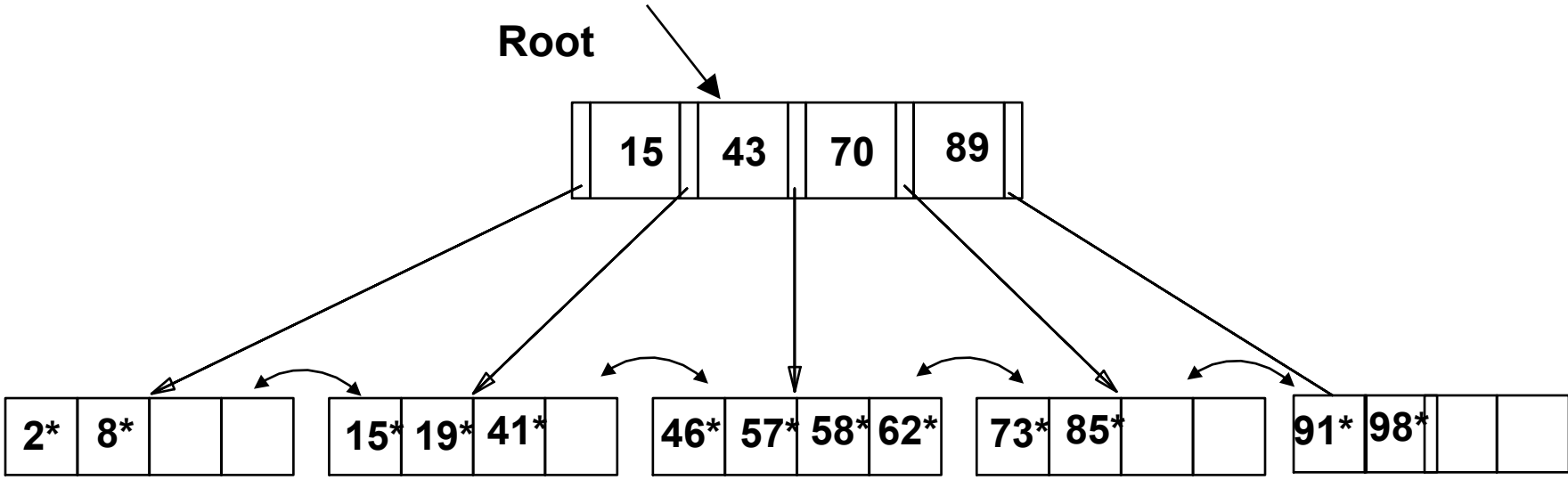
Question 2: B+ trees

Part (a) Draw the B+-tree after inserting data entries with key values **15, 22, 51, and 102 (in that order)** into the B+-tree. Assume that the insert algorithm does not consider redistributing entries.



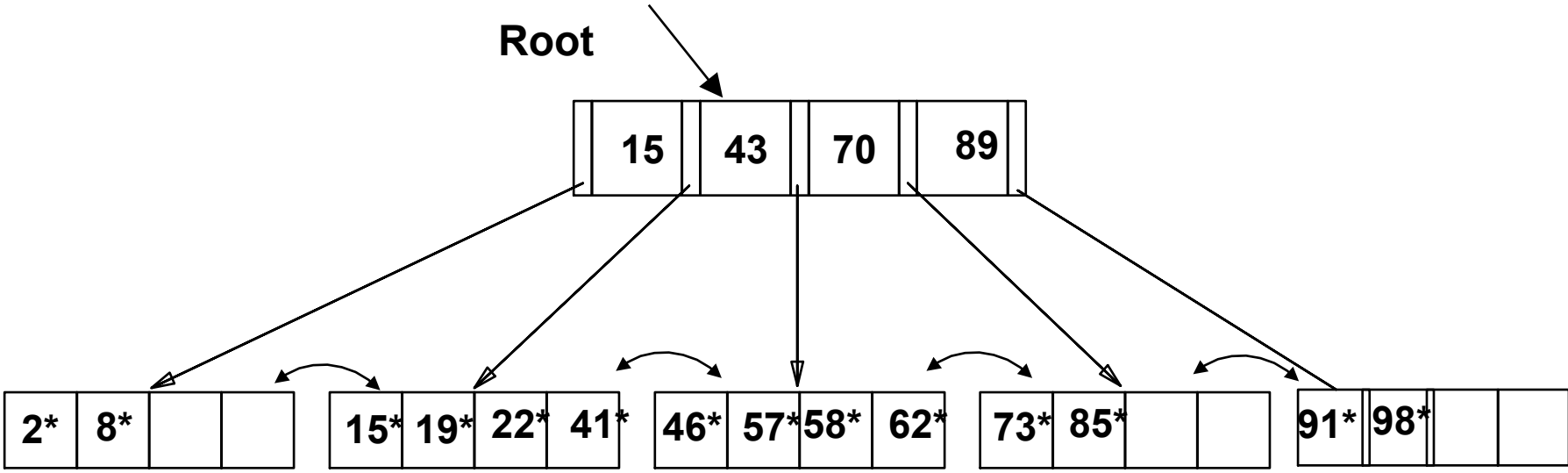
Question 2: B+ trees

Inserting 15



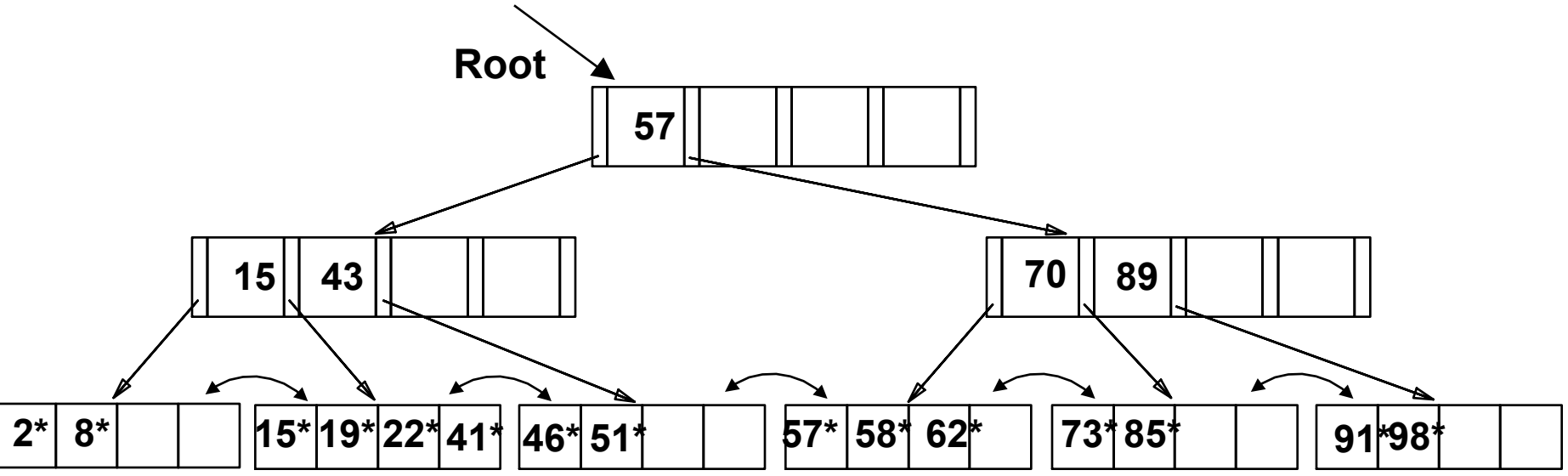
Question 2: B+ trees

Inserting 22



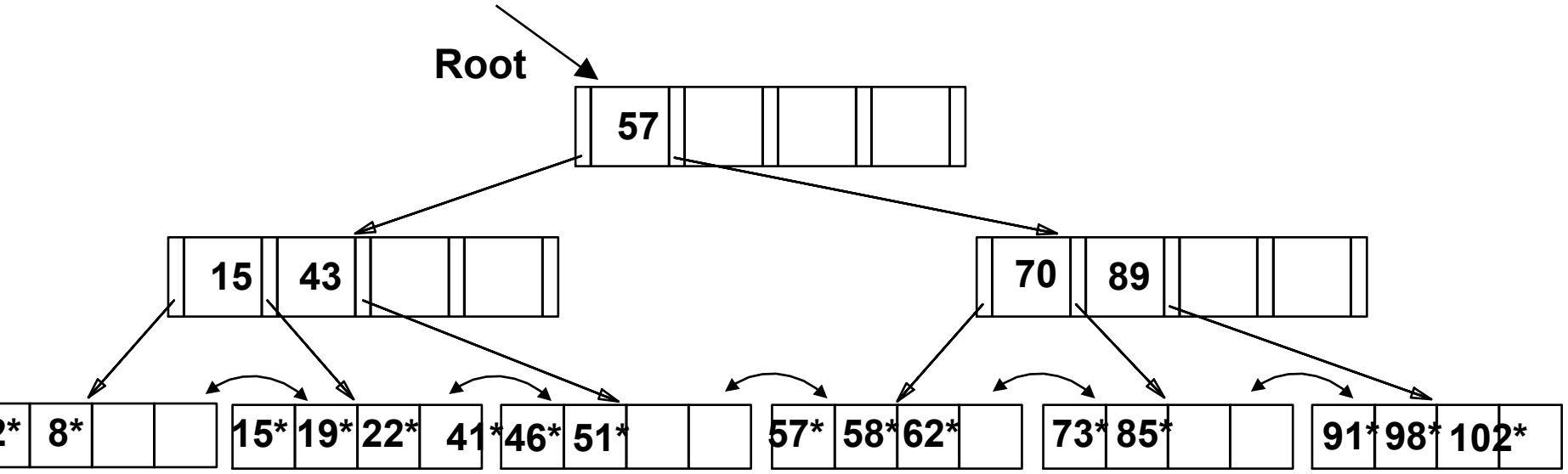
Question 2: B+ trees

Inserting 51



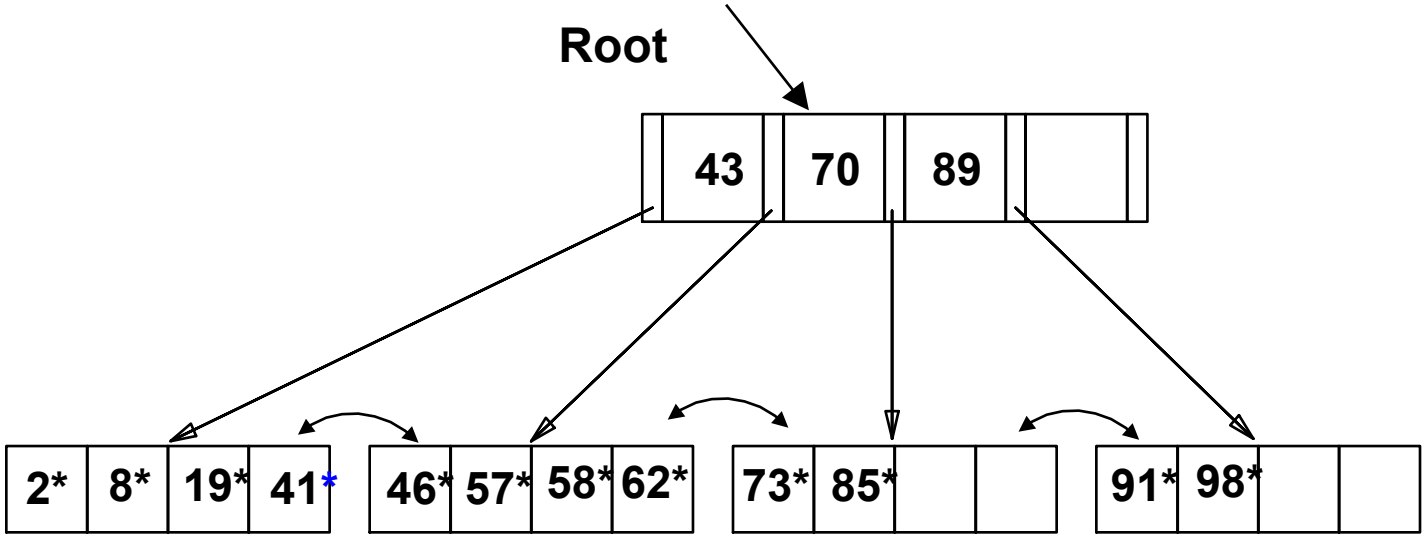
Question 2: B+ trees

Inserting 102



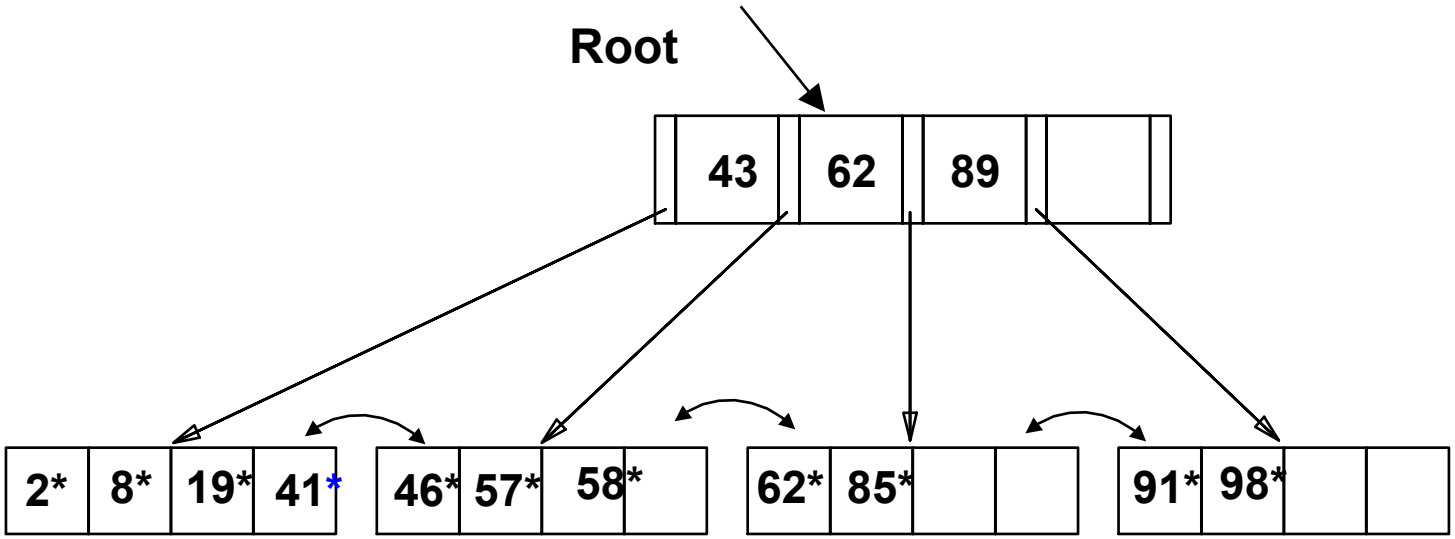
Question 2: B+ trees

Part (b) Consider the above B+-tree (prior to the insertions). Draw the B+-tree after deleting the data entries with key values **73, 46, 2 (in that order). Assume that the delete algorithm** considers redistributing entries.



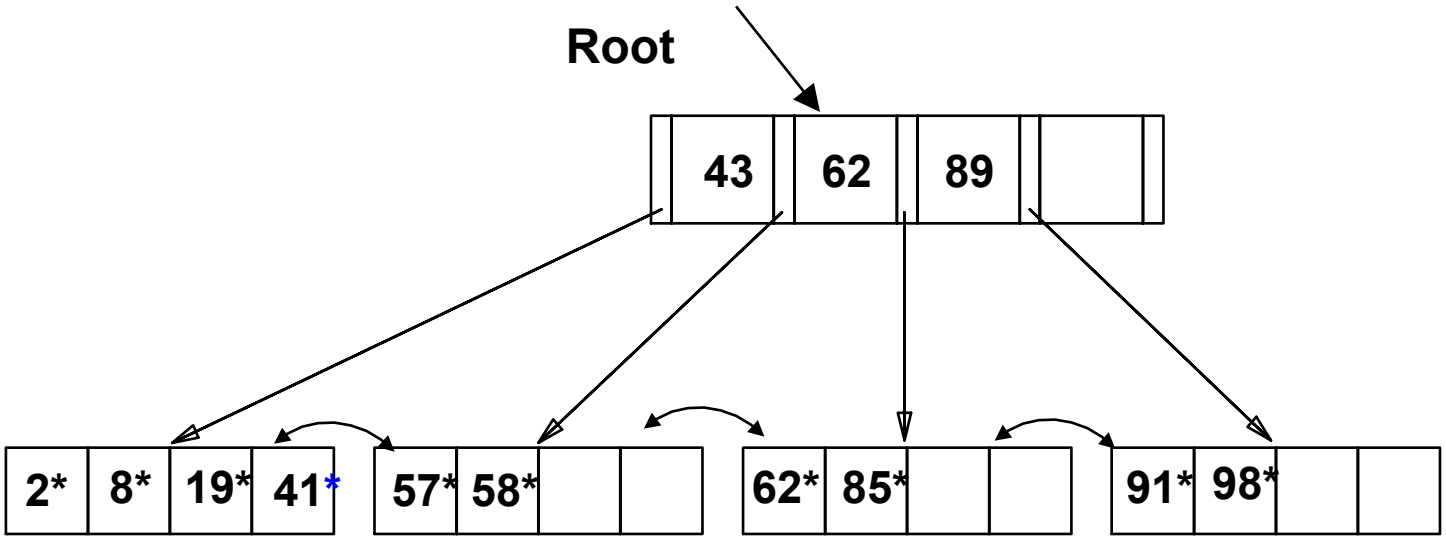
Question 2: B+ trees

Deleting 73



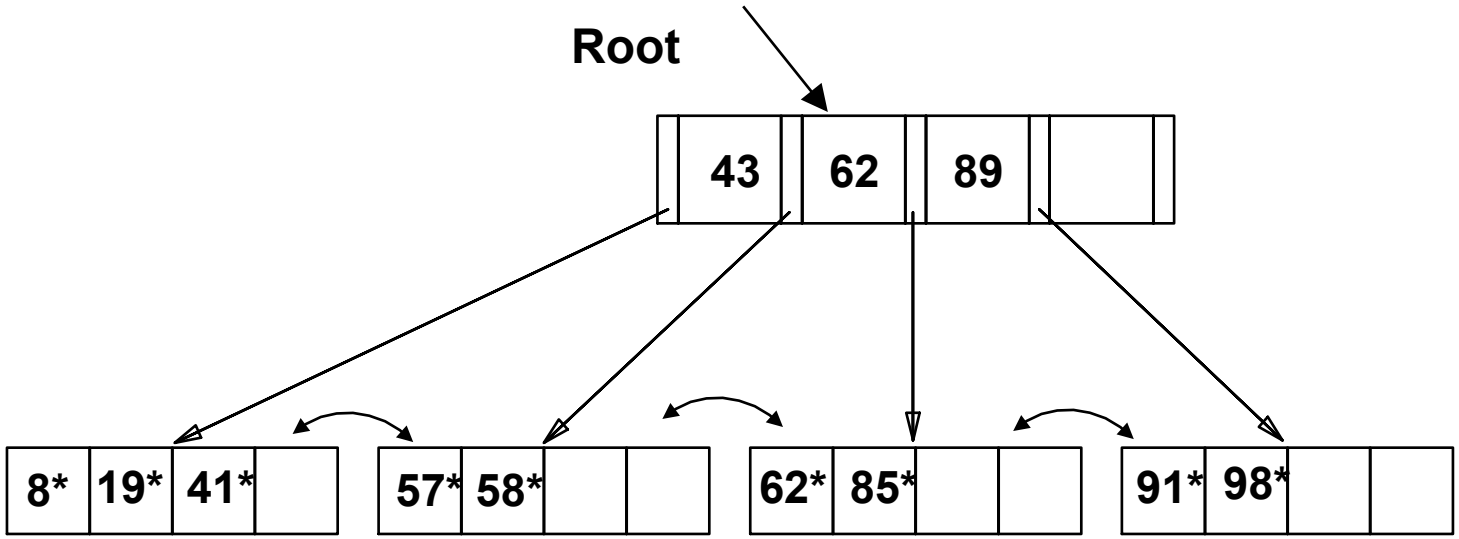
Question 2: B+ trees

Deleting 46



Question 2: B+ trees

Deleting 2

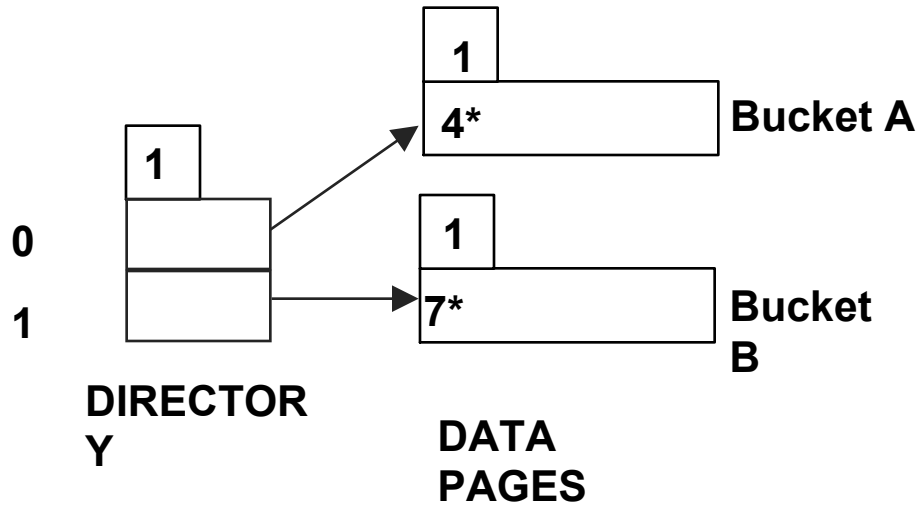


Question 3: Extendible Hashes

a) Consider an extensible hash index that uses the following hash function: **hashvalue = key mod 2^d,** where *d* is the global or local depth of the index. Assume that for this index the bucket capacity is 2 entries. d is initially 1.
Assume that the index is initially empty. Then two entries with key values **4 and 7 are inserted.** Draw the resulting index. *Clearly show the global and local depths and all bucket pointers.*

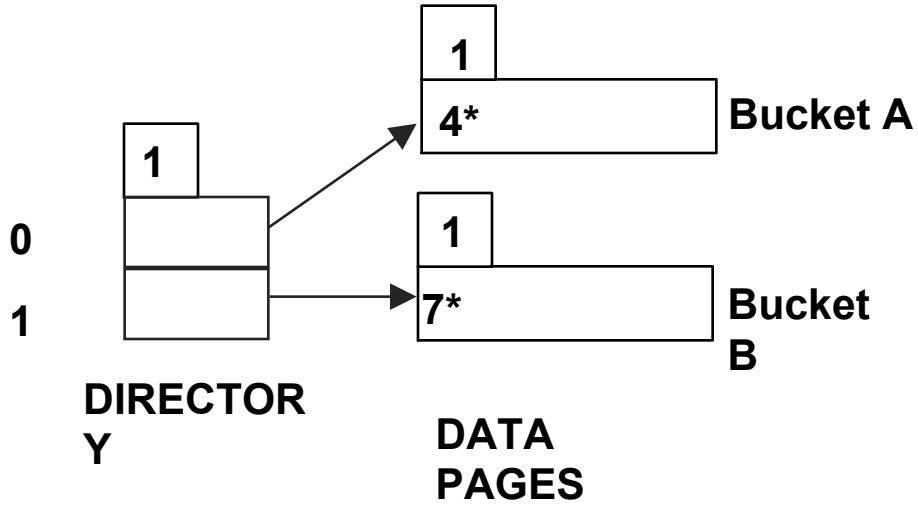
Question 3: Extendible Hashes

a) Assume that the index is initially empty. Then two entries with key values **4 and 7 are inserted.** Draw the resulting index. *Clearly show the global and local depths and all bucket pointers.*



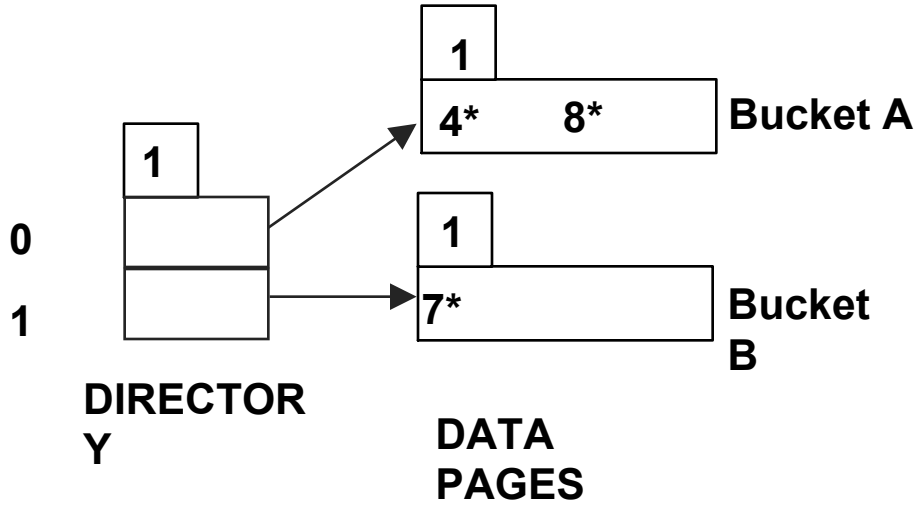
Question 3: Extendible Hashes

b) After inserting the keys 4 and 7 above, the following three additional keys are inserted into the index: **8, 2, 5.** Draw the final index structure with all the five keys. Clearly show the global and local depths, and all bucket pointers



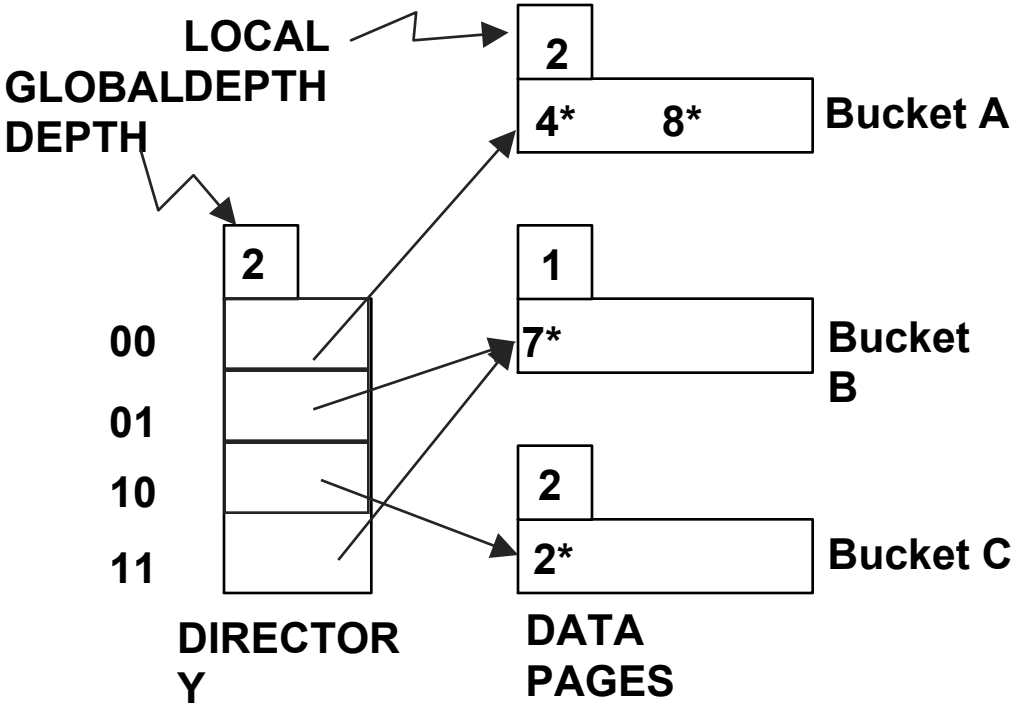
Question 3: Extendible Hashes

Inserting 8



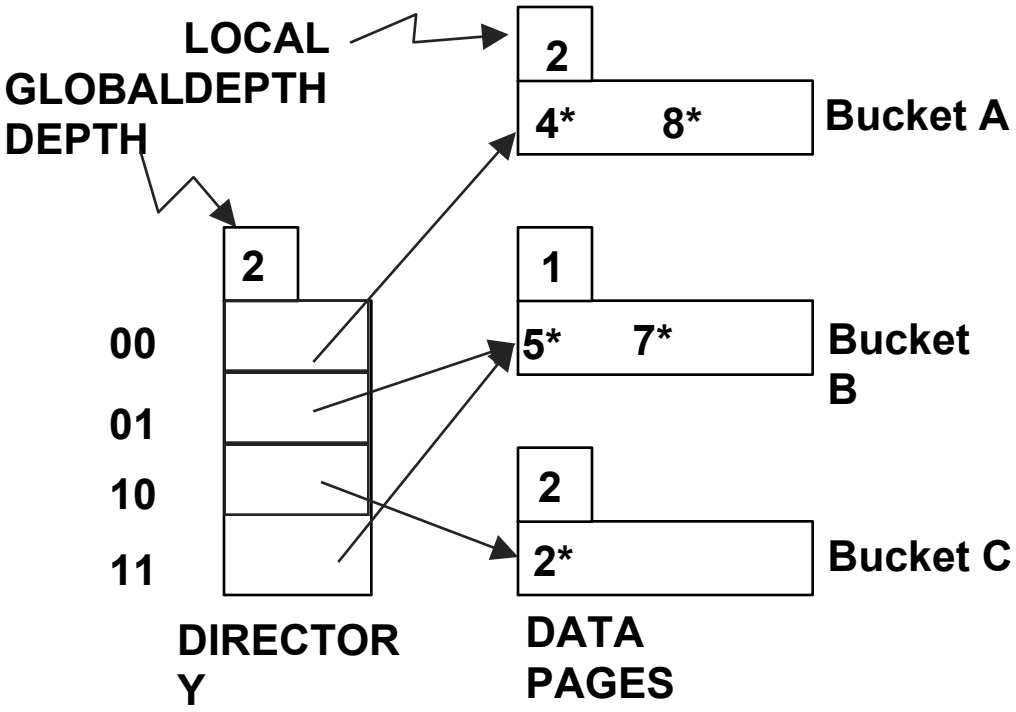
Question 3: Extendible Hashes

Inserting 2



Question 3: Extendible Hashes

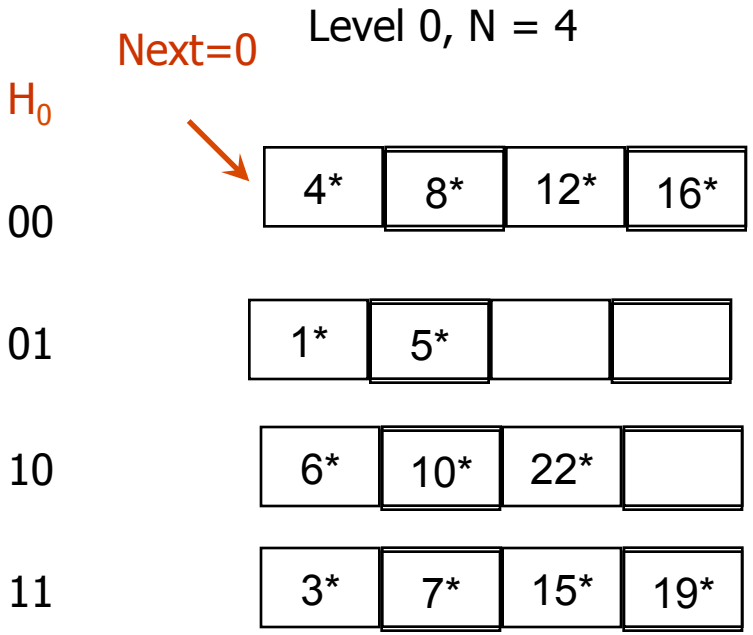
Inserting 5



Question 4: Linear Hashes

Consider a linear hash index. As was the case in the lecture example, let the initial function be h0(k) = f(k) % m, then any later hash function hi(k) = f(k) % 2im. This way, it is guaranteed that if hi hashes a key to bucket j element of [0...2im - 1], hi+1 will hash the same key to either bucket j or bucket j + 2im. At any time, two hash functions hi and hi+1 are used.

Assume we are starting at round 0, with the "Next" pointer set to bucket 0. Each page can hold at most 4 entries. The current state of the hash table is as follows:



Question 4: Linear Hashes

a) Please show the state of the hash buckets after inserting **11.** Be sure to indicate any new buckets, the location of the "Next" pointer, any overflow pages, and the two hash functions currently in use.