Discussion 10

Query Evaluation and Homework 3 solutions

Query Evaluation

- B+ tree can be used for range and equality selection
- Hash based indexing can be used for only equality selection
- Composite keys - combination of keys
  - Hash: Requires all the keys on which hash is created
  - B+ tree : Prefix is sufficient

Ex 12.5 (1)

- Consider the schema with the Sailors relation:
  - Sailors(*sid:* integer, *sname:* string, *rating:* integer, *age:* real)
  - Sailor tuple size = 50 bytes
  - A page can hold 80 tuples
  - Total 500 pages

What are the total number of tuples?

Total number of tuples = total number of pages * tuples/page = 80 * 500 = 40000

Ex 12.5 (1)

- Consider the schema with the Sailors relation:
  - Sailors(*sid:* integer, *sname:* string, *rating:* integer, *age:* real)
  - Sailor tuple size = 50 bytes
  - A page can hold 80 tuples
  - Total 500 pages
- Assume that we have a B+-tree index T on the search key Sailors.sid , and assume that Height(T) = 4, INPages(T) = 50, Low(T) = 1, and High(T) = 100,000. Assume that the root is in main memory
  - σSailors.sid<50,000(Sailors)
  - σSailors.sid=50,000(Sailors)
- Estimate the number of pages retrieved

Ex 12.5 (1) Solution

- σSailors.sid < 50,000(Sailors)
- Assuming uniform distribution, 20,000 tuples will match the condition
- Clustered index:
  - Cost for finding leaf node: 4 I/Os
  - Cost for accessing leaf nodes: 24 I/Os
  - Cost for accessing matching tuples: 250 I/Os
  - Total cost = 278 I/Os
- Unclustered index:
  - Cost for accessing matching tuples: 20,000 I/Os
  - Total cost = 20,028 I/Os using index
  - Using file scan = 500 I/Os
- σSailors.sid = 50,000(Sailors)
- Sid is primary key so only one matching tuple
- Cost = Height of tree + cost to read qualifying tuple = 4+1=5 I/Os

Ex 12.5 (2)

- Consider the schema with the Sailors relation:
  - Sailors(*sid:* integer, *sname:* string, *rating:* integer, *age:* real)
  - Sailor tuple size = 50 bytes
  - A page can hold 80 tuples
  - Total 500 pages

- Assume that we have a hash index T on the search key Sailors.sid , and assume that IHeight(T) = 2, INPages(T) = 50, Low(T) = 1, and High(T) = 100,000.
  - σSailors.sid<50,000(Sailors)
  - σSailors.sid=50,000(Sailors)

- Estimate the number of pages retrieved

Ex 12.5 (2) Solution

- σSailors.sid < 50,000(Sailors)
  - Hash index on sid cannot be used for range queries
  - Need to do a file scan that require 500 I/Os

- σSailors.sid = 50,000(Sailors)
  - Sid is primary key so only one matching tuple
  - Cost = Depth (or levels) of hash table + cost to read qualifying tuple = 2+1=3 I/Os

External Merge Sort

- **Run:** Sorted subfile
- B buffer pages
- Pass 0: Uses B buffer pages
    - Produces ceiling(N/B) sorted runs of B pages each
- Pass 1, 2,..: merge ***B-1*** runs, using (B-1)-way merge
- Number of passes:

$$1 + \lceil \log_{B-1} \lceil N/B \rceil \rceil$$

- Cost = 2N * (# of passes)

Ex 13.1

- Suppose you have a file with 20,000 pages and you have five buffer pages. Answer the following questions:
  - How many runs will you produce in the first pass?
  - How many passes will it take to sort the file completely?
  - What is the total I/O cost of sorting the file?

Ex 13.1 Solution

- Number of runs after the first pass
- Pass 0 produces $\lceil N / B \rceil$ runs of B pages each, where N is the number of file pages and B is the number of available buffer pages
- ceil(20000/5) = 4000 sorted runs.

- Passes to completely sort the file
- Number of passes = $\lceil \log(B-1)N1 \rceil + 1$ where N1 = $\lceil N / B \rceil$

  $= \lceil \log 4(4000) \rceil + 1$ = 7 passes

- Total number of page I/Os = 2 * N *(#passes)

  = 2 * 20000 * 7 = 280000

Question 1

Consider a hard disk with 5400 RPM and 2 double-sided platters. Each surface has 300,000 tracks and 3000 sectors per track. (For simplicity, we assume that the number of sectors per track does not vary between the outer and inner area of the disk.) Each sector has 512 bytes. Suppose a page size of 4096 bytes is chosen. A file containing 100,000 records of 400 bytes each is to be stored on such a disk and that no record is allowed to span two blocks.

Question 1

- What is the capacity(max number of bytes) of the disk?
- How many records fit in a page?
- How many I/Os are required to read the entire file sequentially?
- How many records of 100 bytes each can be stored using this disk?

Given:

Page size = 4096 bytes

Sector size = 512 bytes

Number of records = 100,000

Size of a record = 400 bytes

# of tracks/surface = 300,000

# of sectors/track = 3000

# of bytes/sector = 512 bytes

# of surfaces/disk = 4

bytes/track = bytes/sector × sectors/track = 512 × 3000 = 1,536,000

bytes/surface = bytes/track × tracks/surface = 1536K × 300K = 460,800 * 10^6

bytes/disk = bytes/surface × surfaces/disk = 460,800 * 10^6 * 4 = 1843200 * 10^6

Number of blocks in a track = 1536000 / 4096 = 375

1) What is the capacity of the disk?

A) 1843200 * 10^6 bytes

2) How many records fit into a page?

A Size of page/ size of record = 4096/400 > 10. So 10 records can fit into a block

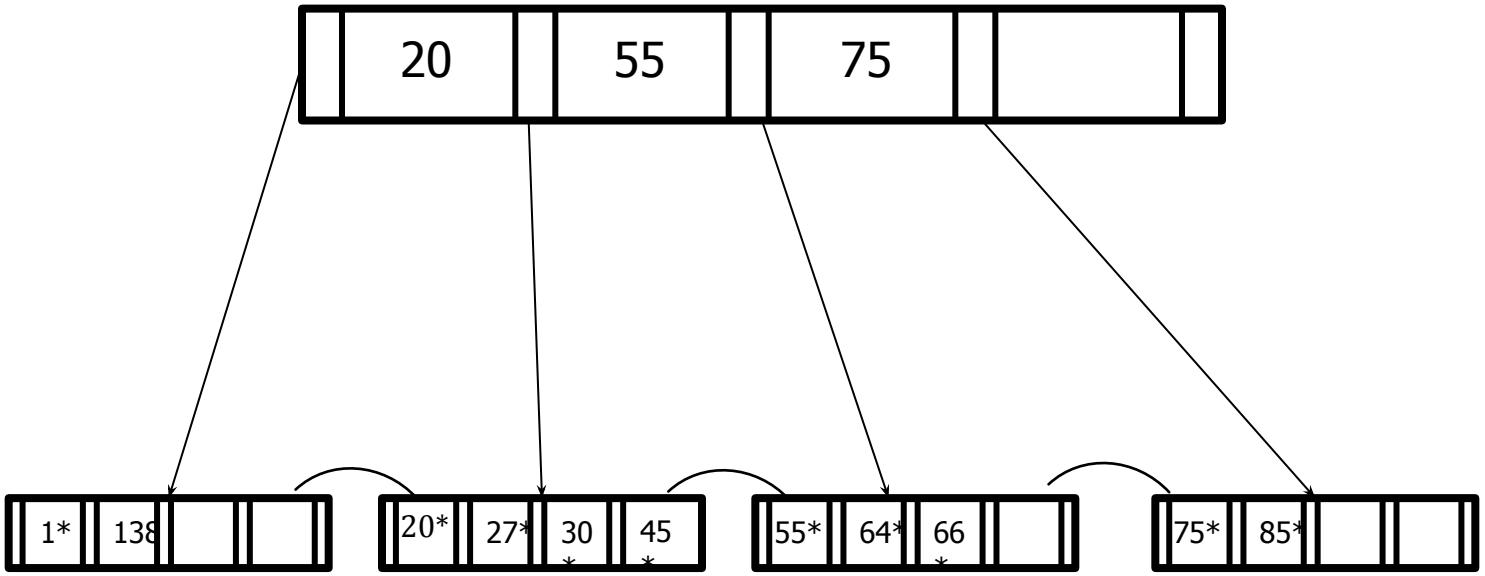
3) How many I/Os are required to read the entire file sequentially?
A 100,000/10 = 10000 I/Os

4) How many records of 100 bytes each can be stored using this disk?

Number of pages in a disk = Disk capacity / 4096 bytes = 1843200* 10^6 / 4096 = 450 * 10^6 pages
Number of records that can fit in a page = 4096/100 > 40
Total number of records that can fit = 40 * 450 * 10^6 = 18 * 10^9

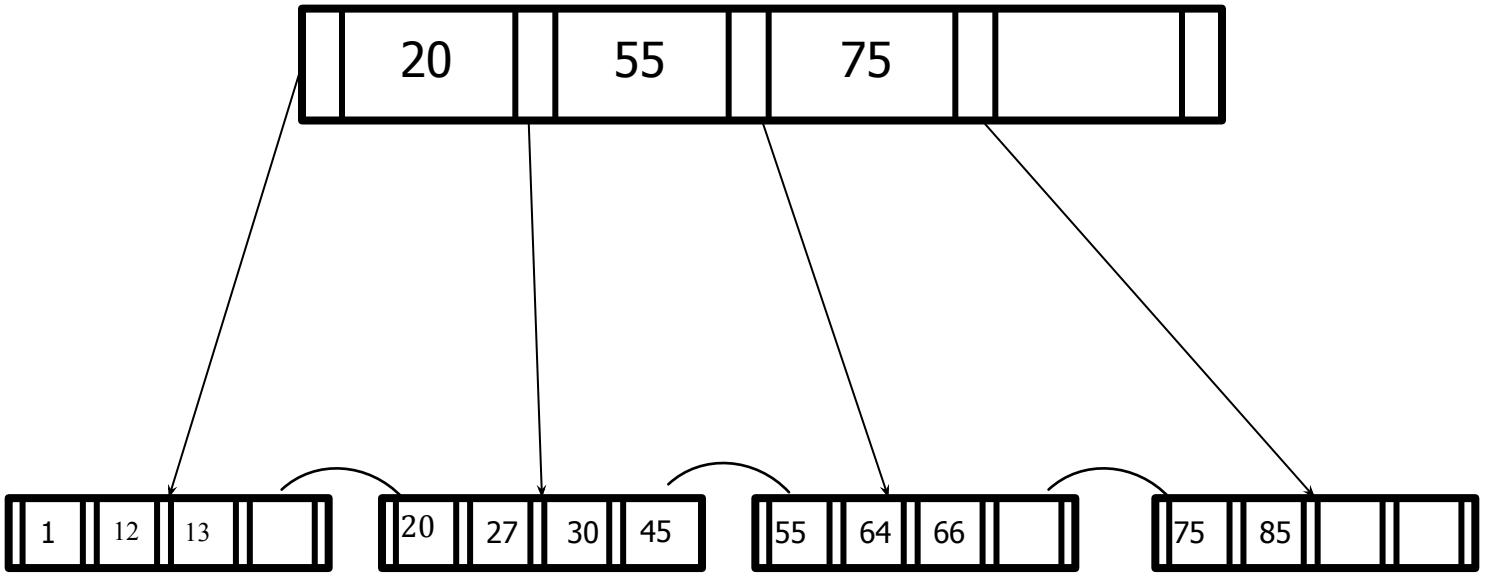
Question 2

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).



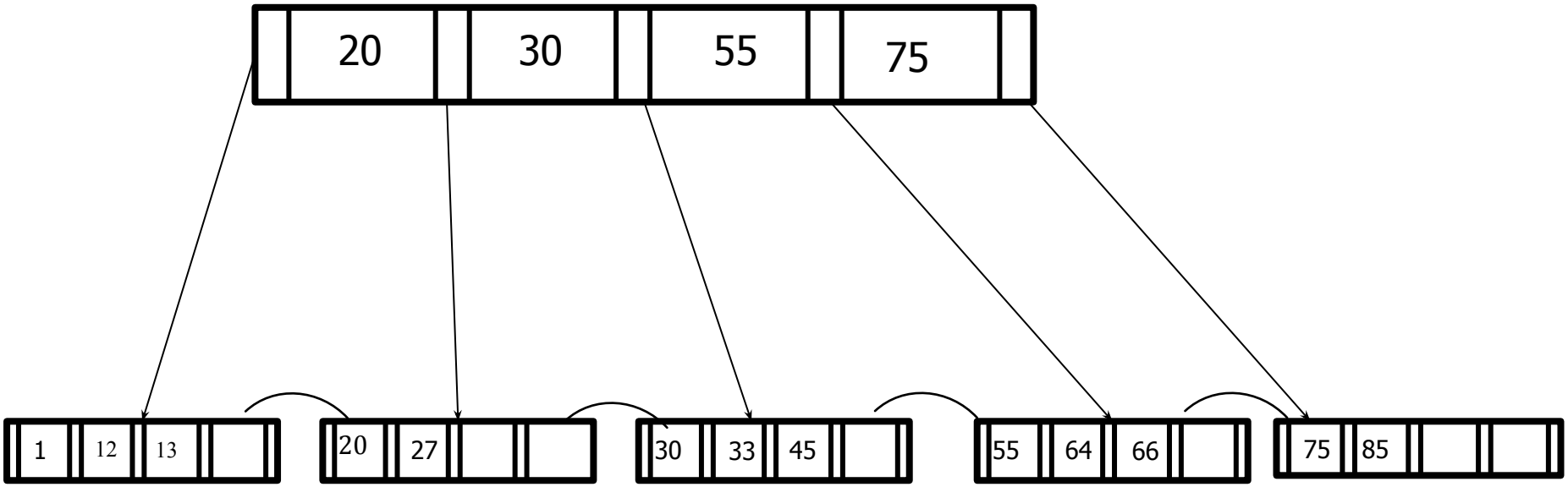
Question 2 - Solution

Inserting 12



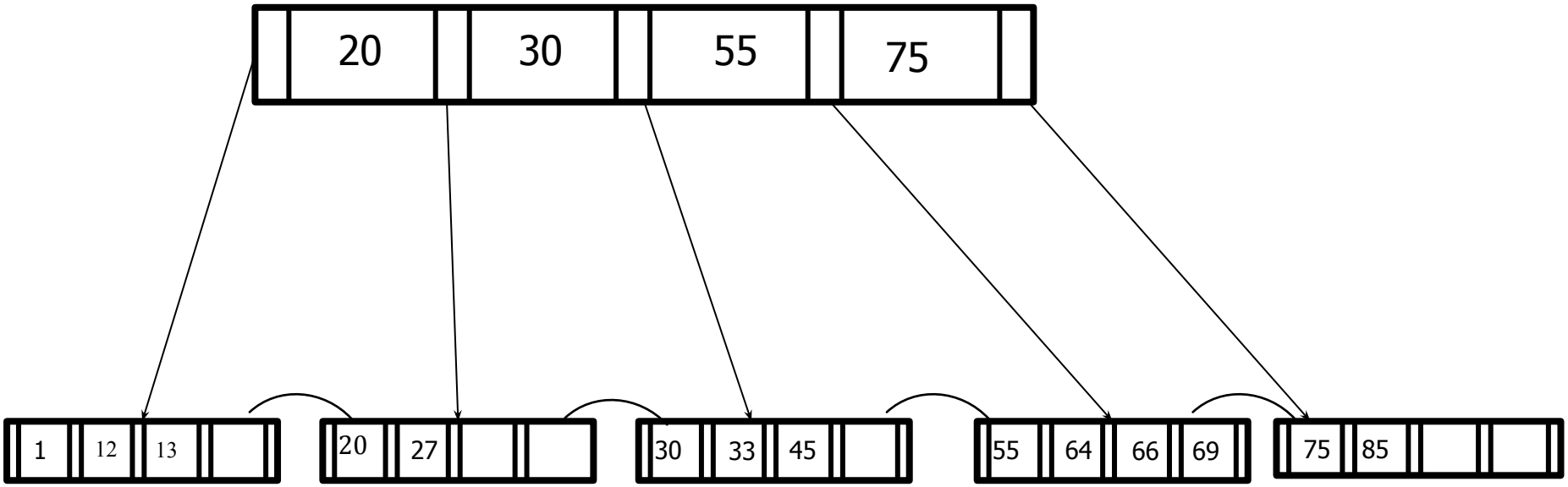
Question 2 - Solution

Inserting 33



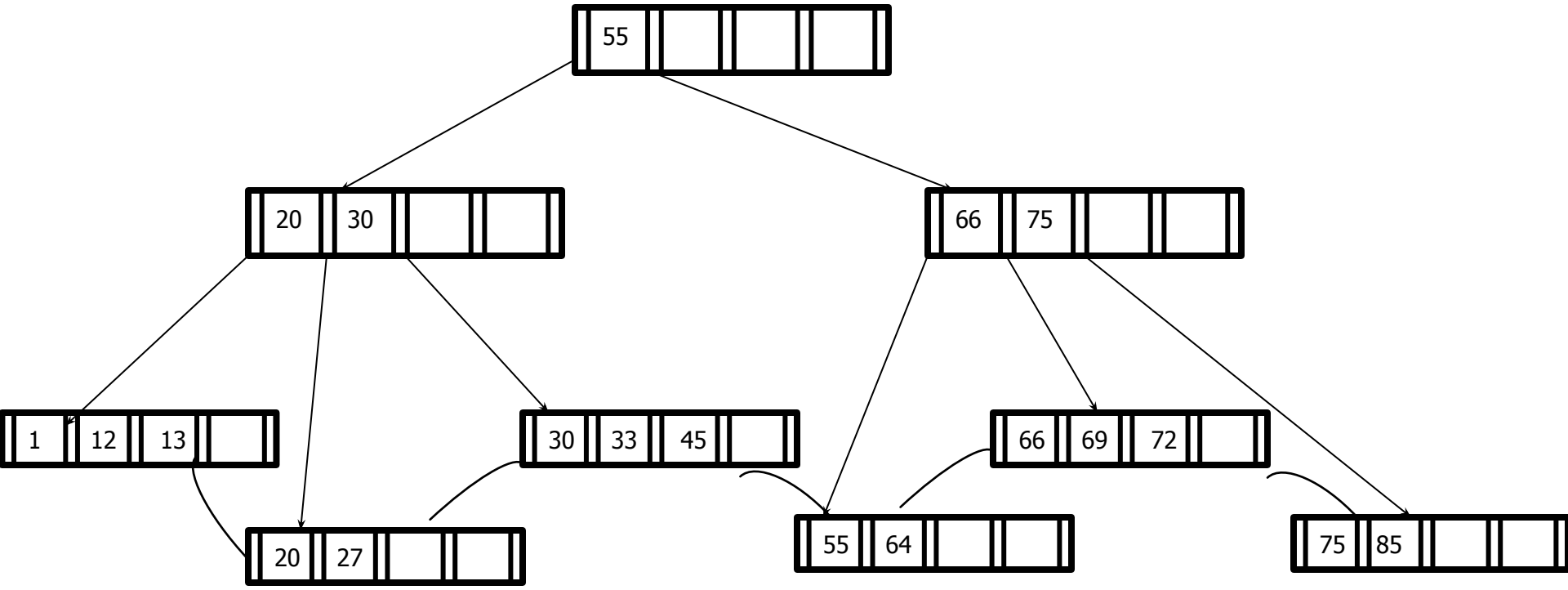
Question 2 - Solution

Inserting 69



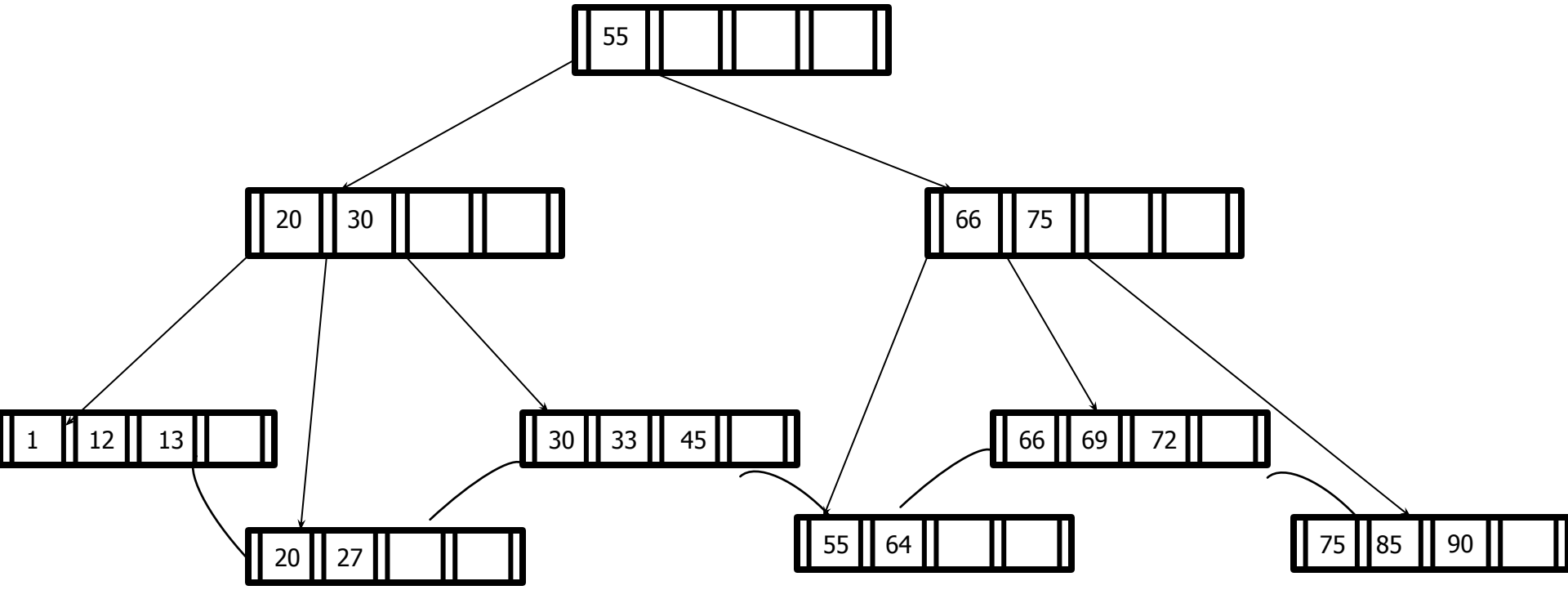
Solution 2a

Inserting 72 Assume no redistribution of entries.



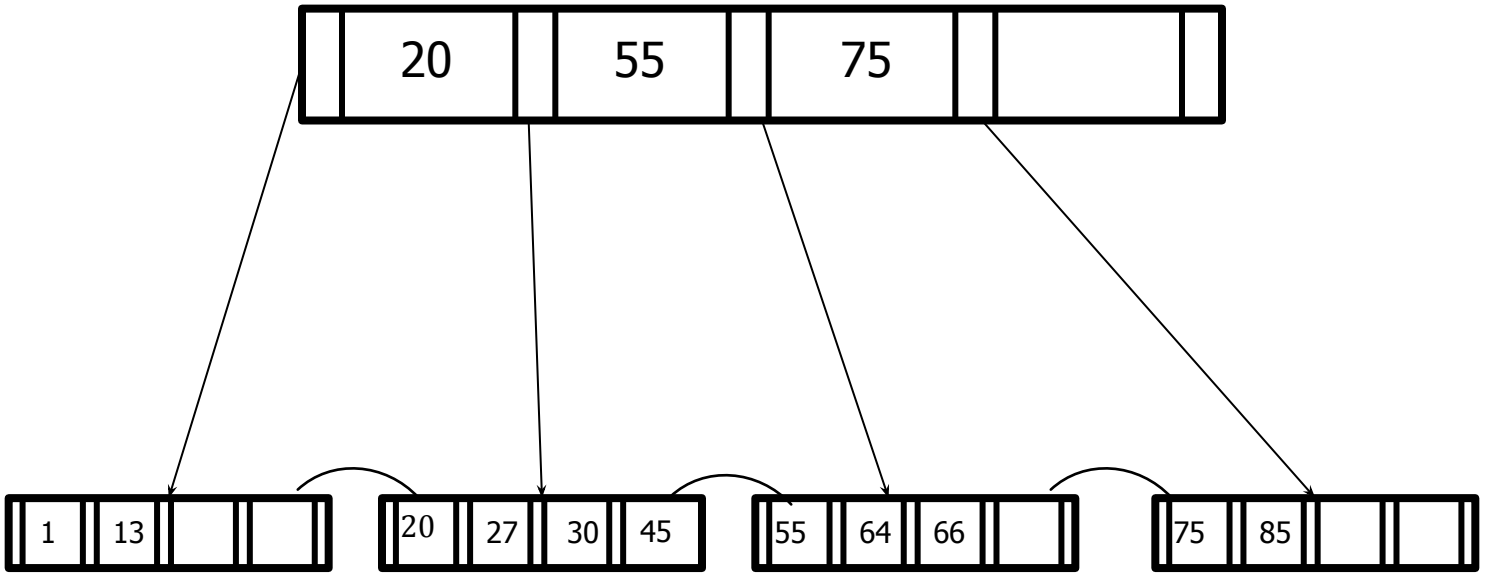
Solution 2a

Inserting 90 Assume no redistribution of entries.



Problem 2b

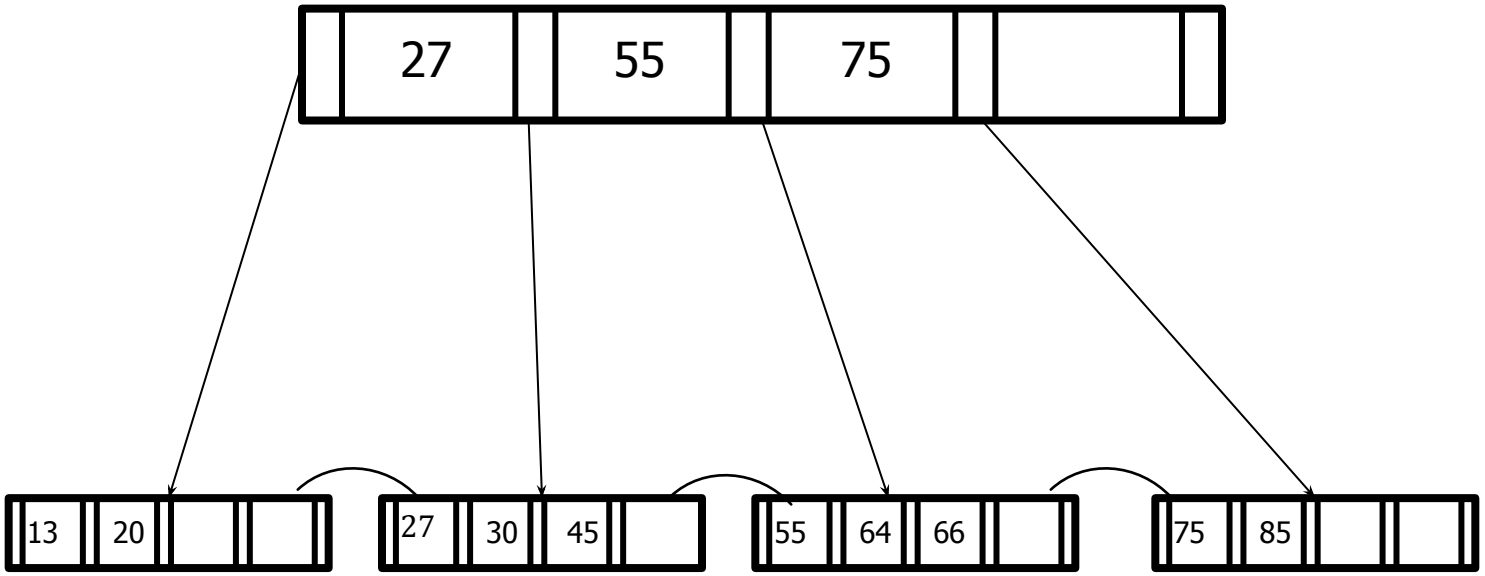
Delete the data entries with key values 1, 75, 20, 64 (in that order).
Assuming redistribution of entries.



Solution 2b

Delete the data entries with key values 1, 75, 20, 64 (in that order).
Assuming redistribution of entries.

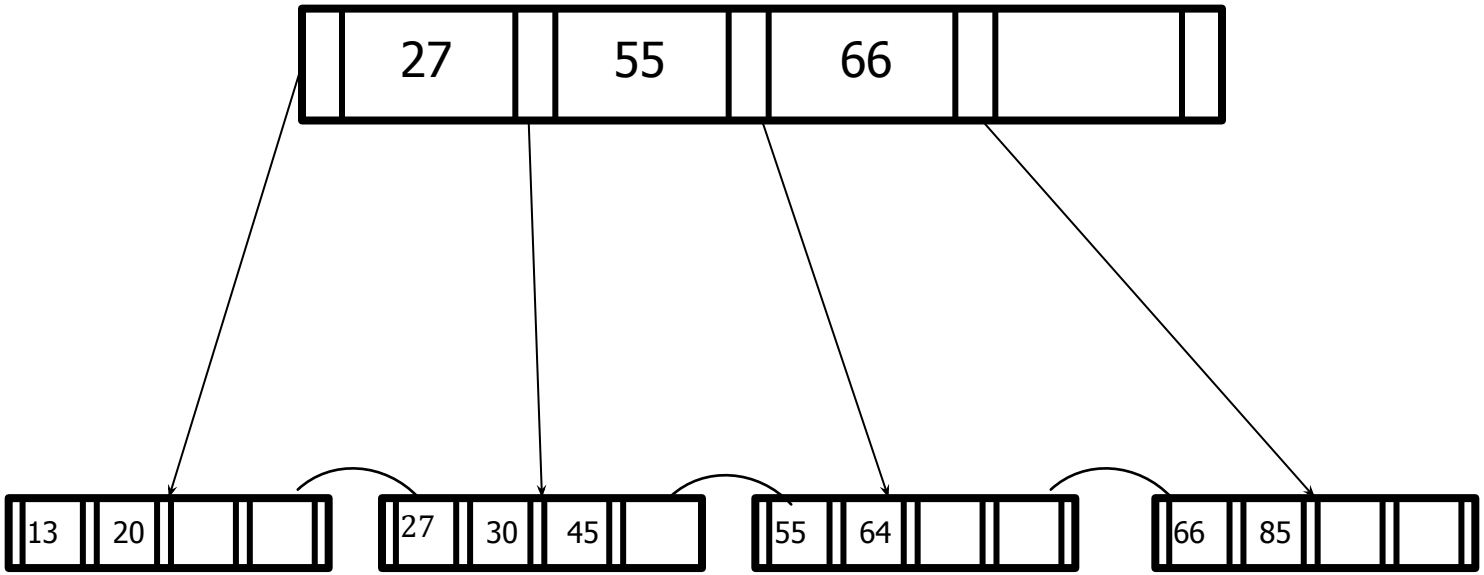
DELETING 1



Solution 2b

Delete the data entries with key values 1, 75, 20, 64 (in that order).
Assuming redistribution of entries.

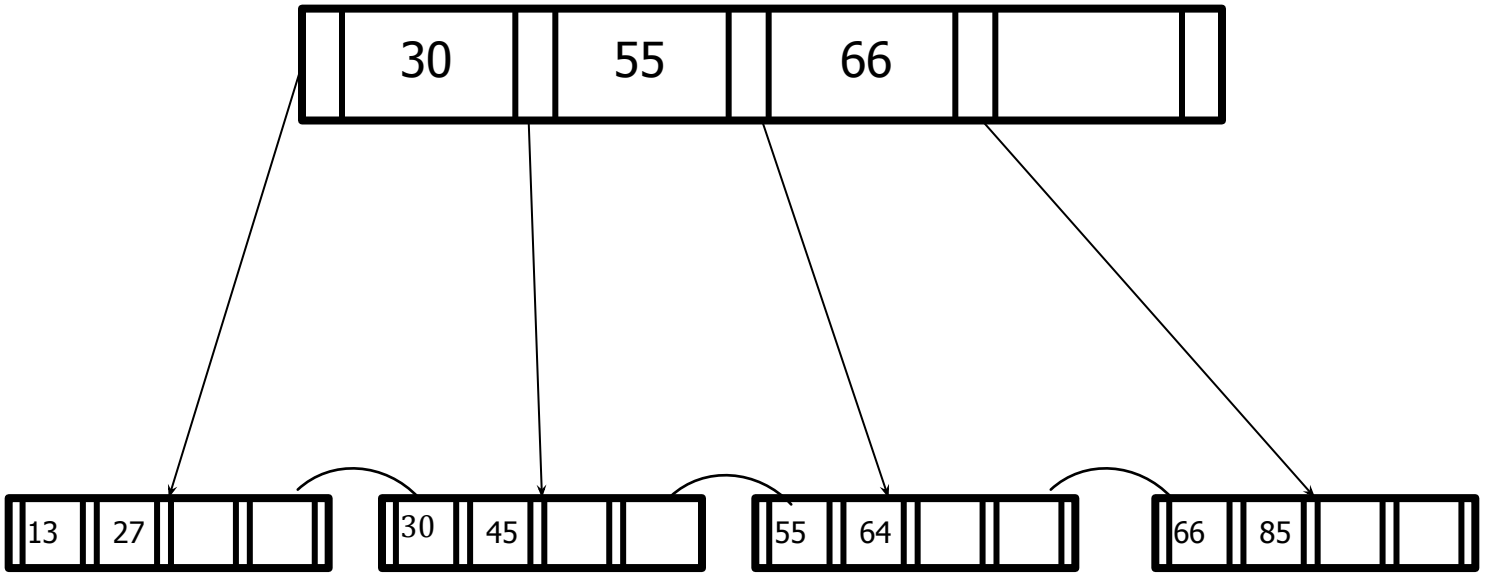
DELETING 75



Solution 2b

Delete the data entries with key values 1, 75, 20, 64 (in that order). Assuming redistribution of entries.

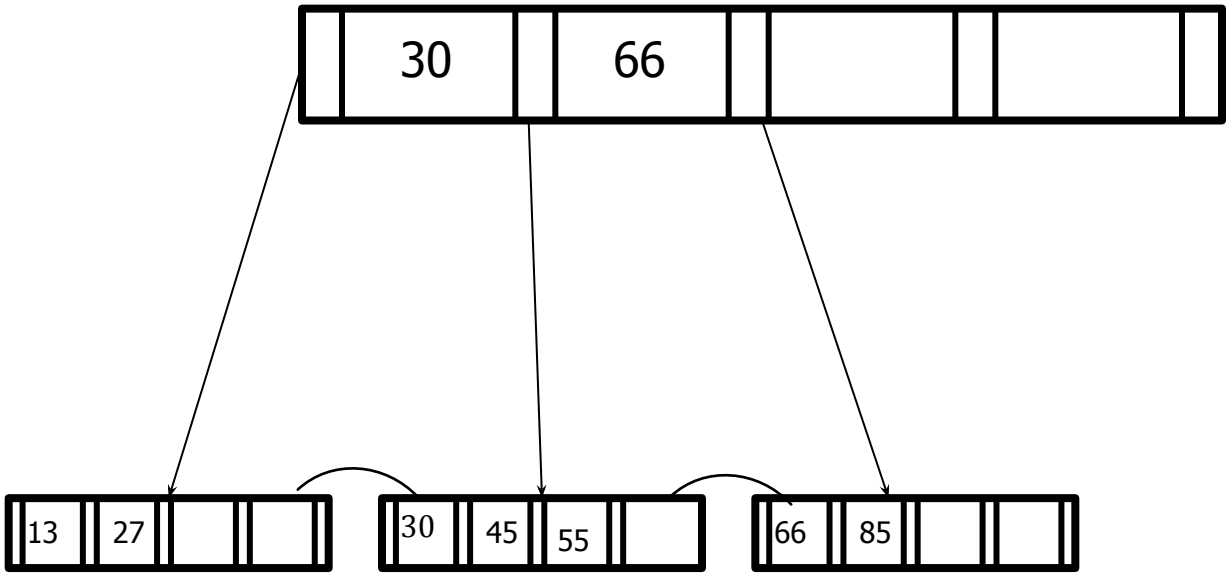
DELETING 20



Solution 2b

Delete the data entries with key values 1, 75, 20, 64 (in that order).
Assuming redistribution of entries.

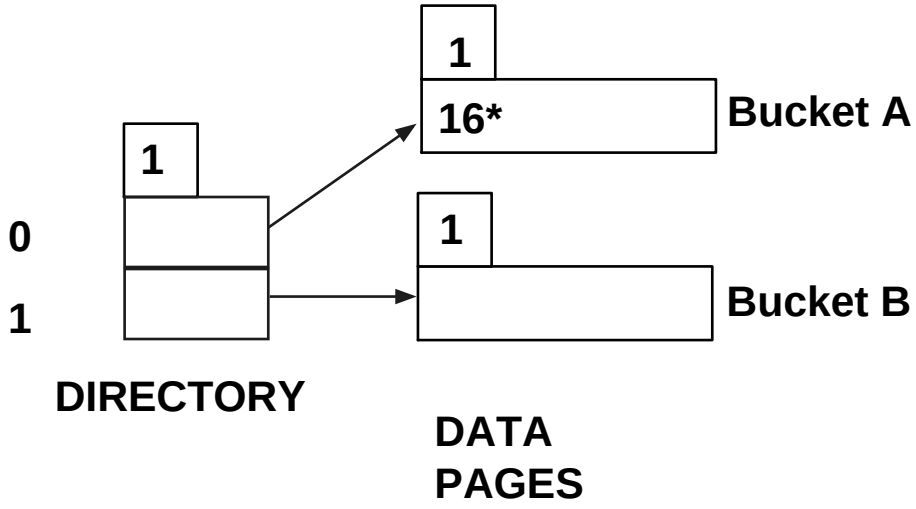
DELETING 64



Question 3(a)

- Consider an extendible hash index that uses the following hash function: **hashvalue = least significant bits**, where *d* is the global or local depth of the index. Assume that for this index the bucket capacity is 2 entries.

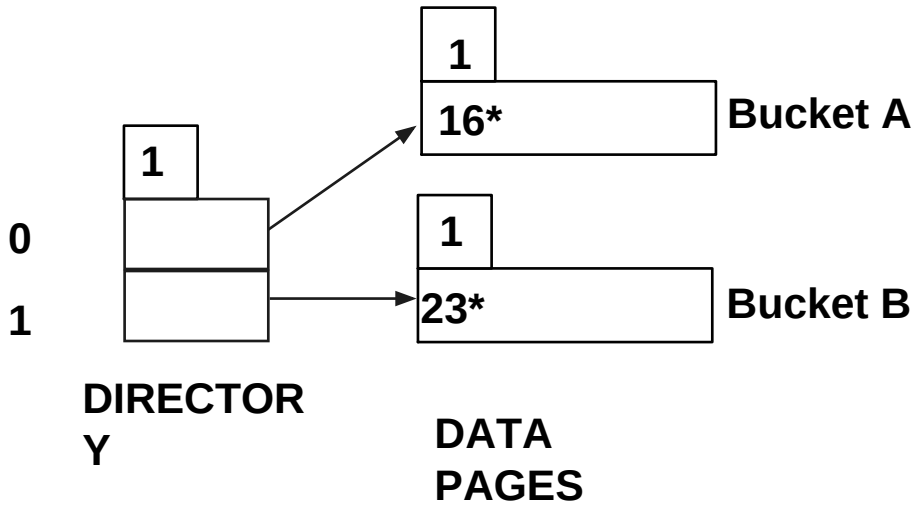
  Assume that the index is initially empty. Then two entries with key values **16** and **23** are inserted. Draw the resulting index. *Clearly show the global and local depths and all bucket pointers*.



After adding 16

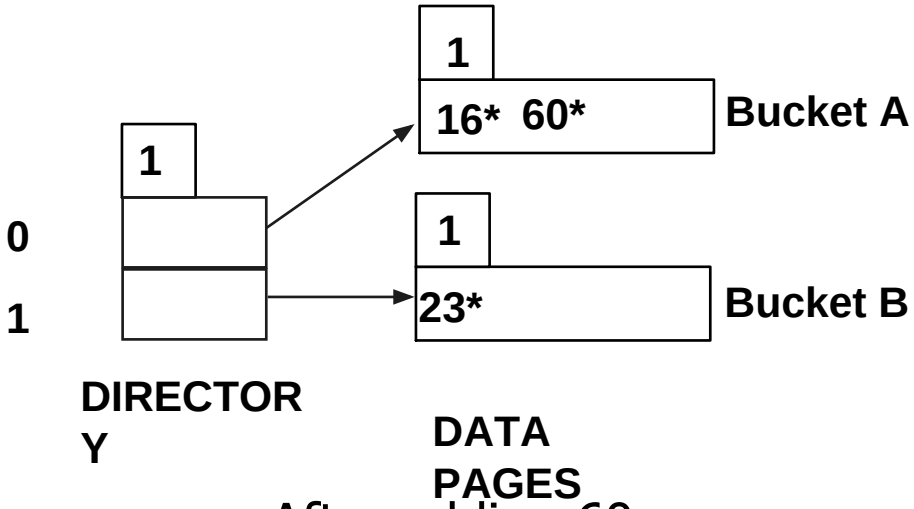
Question 3(a)

After adding 23



Question 3(b)

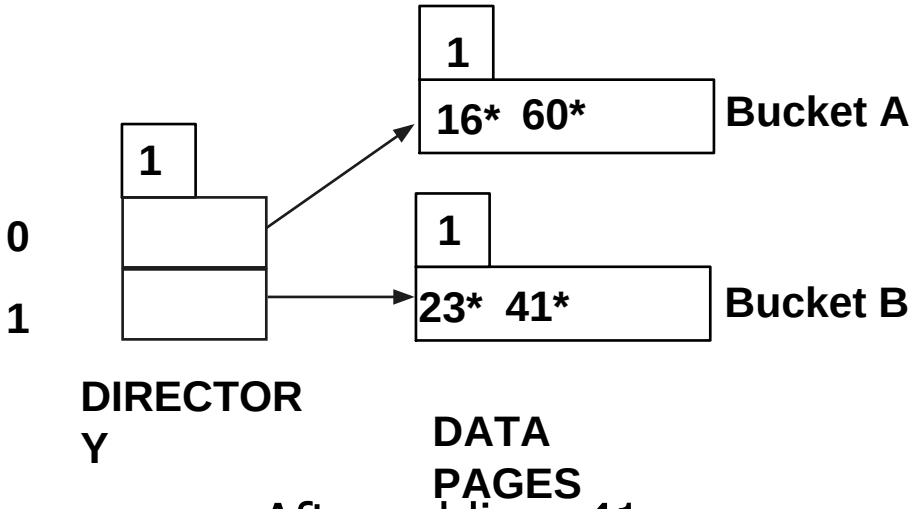
- After inserting the keys 16 and 23 above, the following three additional keys are inserted into the index: **60**,**41**, **26**, **37**, **28**. Draw the final index structure with all the five keys. Clearly show the global and local depths, and all bucket pointers.



After adding 60

Question 3(b)

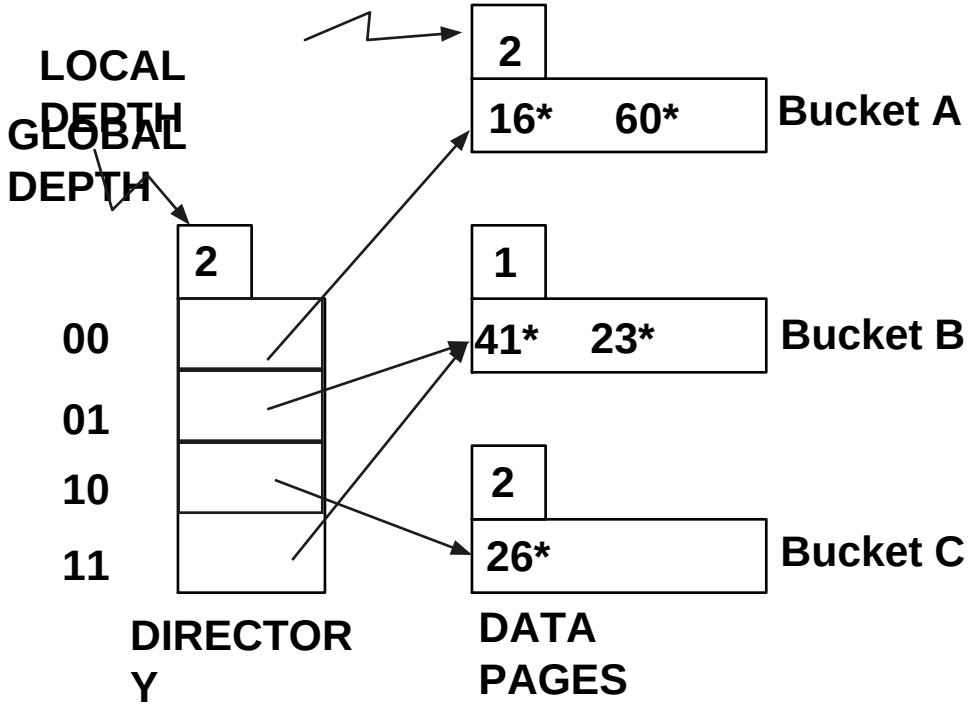
- After inserting the keys 16 and 23 above, the following three additional keys are inserted into the index: **60**,**41**, **26**, **37**, **28**. Draw the final index structure with all the five keys. Clearly show the global and local depths, and all bucket pointers.



After adding 41

Question 3(b)

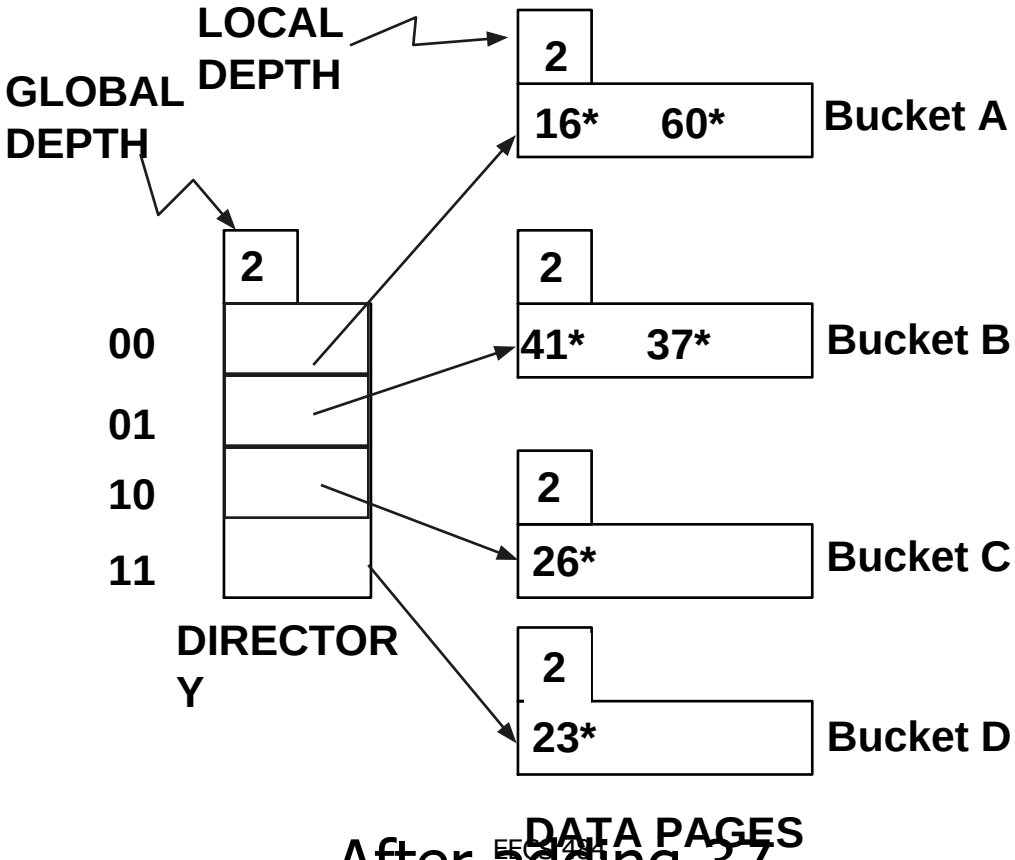
- After inserting the keys 16 and 23 above, the following three additional keys are inserted into the index: **60,41, 26, 37, 28**. Draw the final index structure with all the five keys. Clearly show the global and local depths, and all bucket pointers.



After adding 26

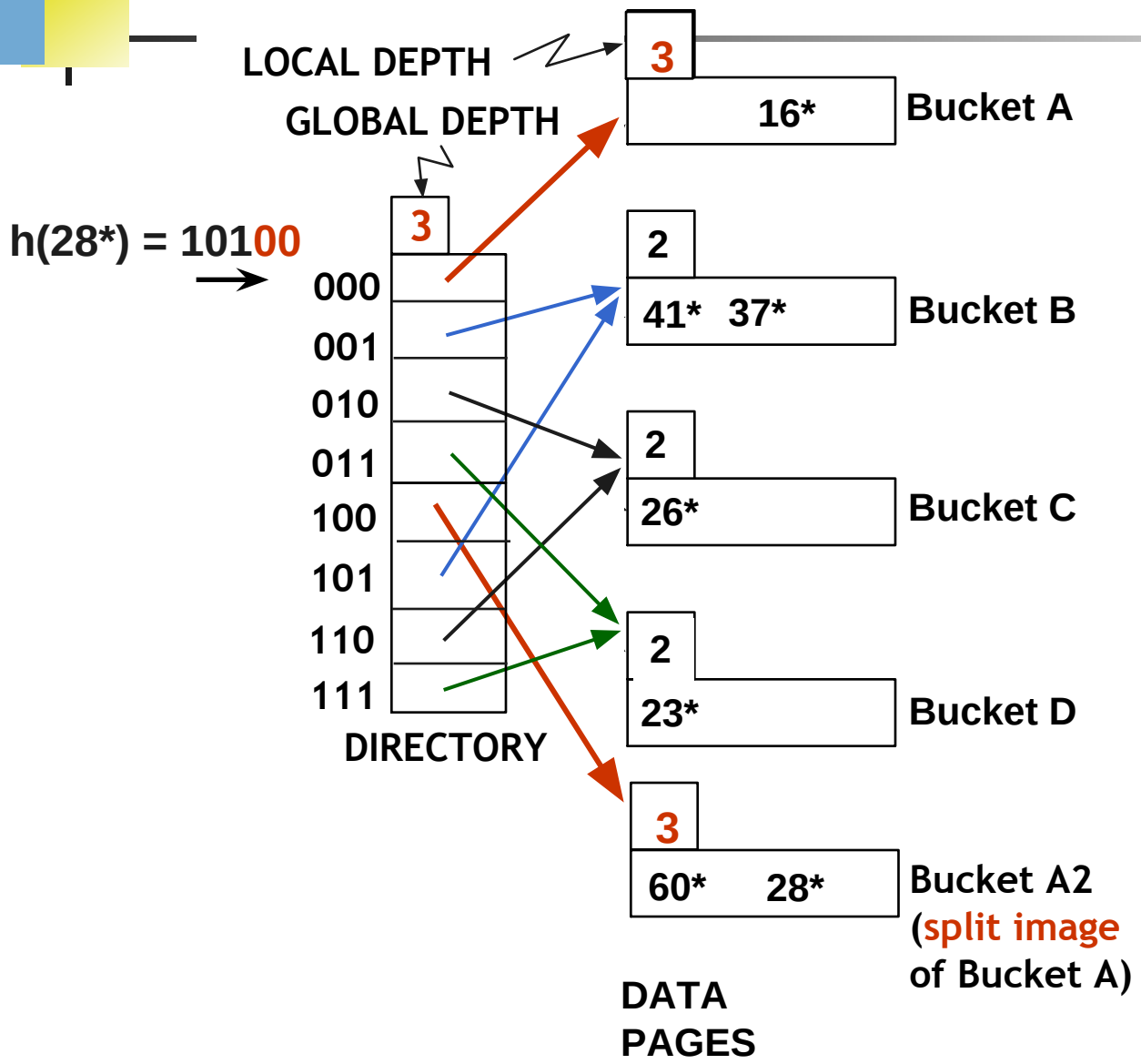
Question 3(b)

- After inserting the keys 16 and 23 above, the following three additional keys are inserted into the index: **60,41, 26, 37, 28**. Draw the final index structure with all the five keys. Clearly show the global and local depths, and all bucket pointers.



After adding 37

After adding 26

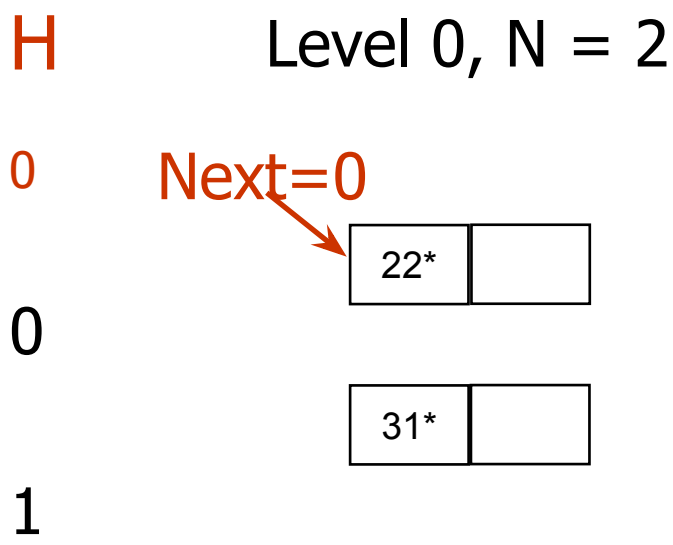


c) What happens when the local depth becomes more than the global depth?

Possible answers:

- The index structure is split.
- This situation is impossible because when a split occurs the directory structure is doubled and then the global and local depth are increased simultaneously.

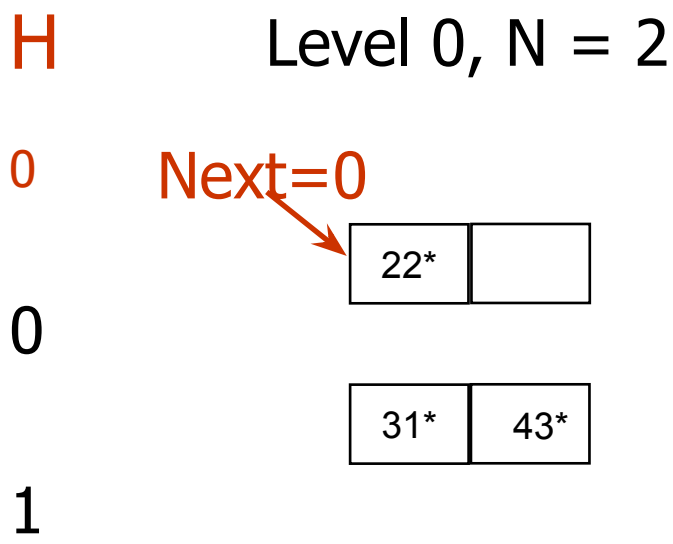
Question 4



Primary bucket Pages in the Hash File

Question 4 - Solution

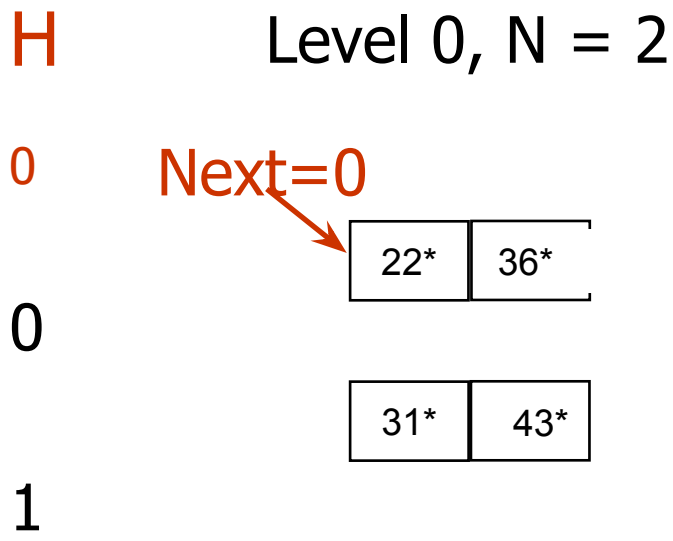
Inserting 43



Primary bucket Pages in the Hash File

Question 4 - Solution

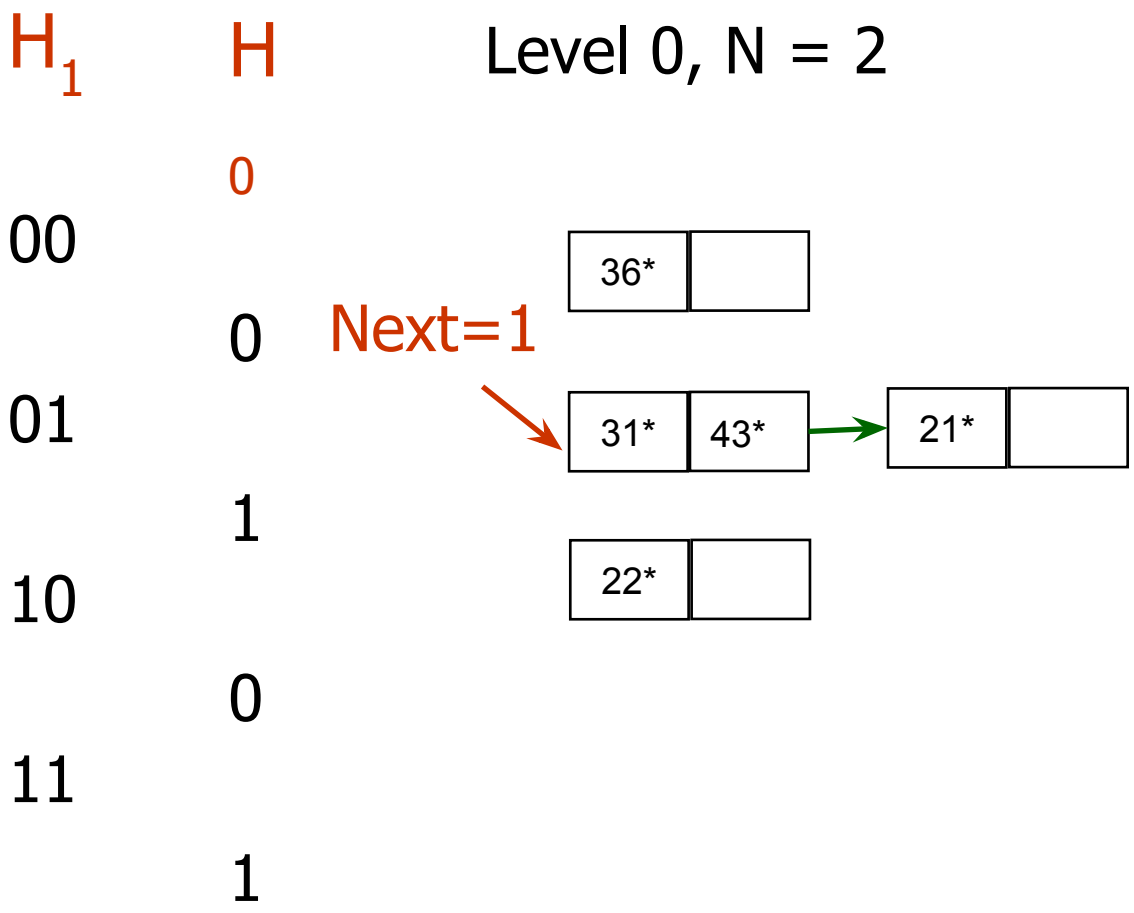
Inserting 36



Primary bucket Pages in the Hash File

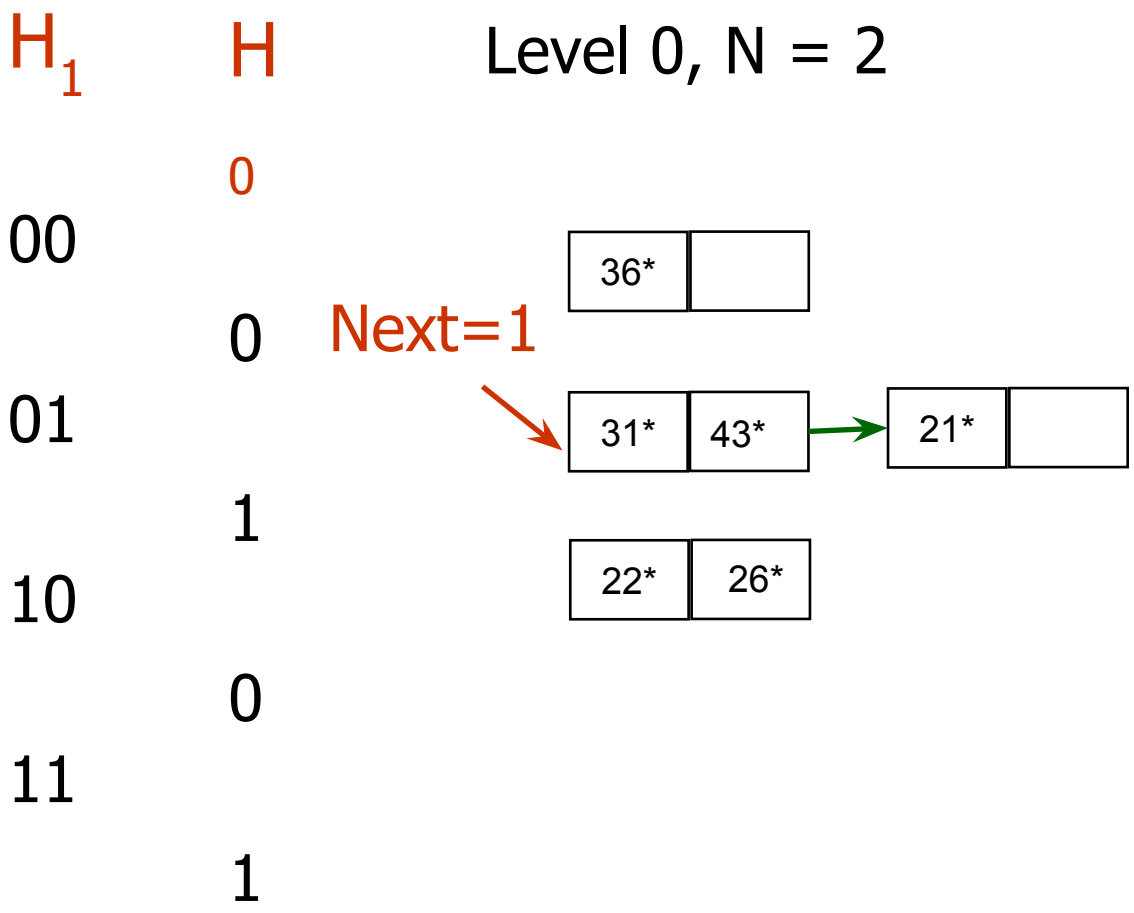
Question 4 - Solution

Inserting 21



Question 4 - Solution

Inserting 26



Question 4 - Solution

Inserting 54