National University of Computer and Emerging Sciences, Lahore Campus

	Course:	**Advance Database Concepts**
	Program:	**BS (Computer Science)**
	Instructor:	**Muhammad Ishaq Raza**
	Practice Problems:	**File Structures and Hashing**

SOLUTION

Topic: File Structures and Hashing

Q1. Assume a relation R (A, B, C) is given; R is stored as an ordered file (un-spanned) on non-key field C and contains 500,000 records. Attributes A, B and C need 5 bytes of storage each, and blocks have a size of 2048 Bytes. Each A value occurs at an average 5 times in the database, each B value occurs 50 times in the database, and each C value occurs 50,000 times in the database. Assume there is no index structure exists.

Estimate the number of block fetches needed to compute the following queries (where C_a and C_c are integer constants):

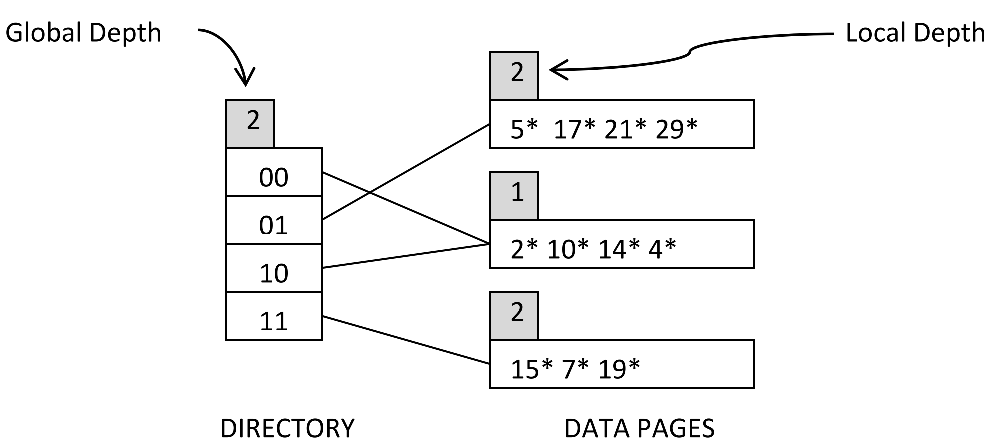
a. SELECT B, C FROM R WHERE A = C_a;

b. SELECT B, C FROM R WHERE C = C_c;

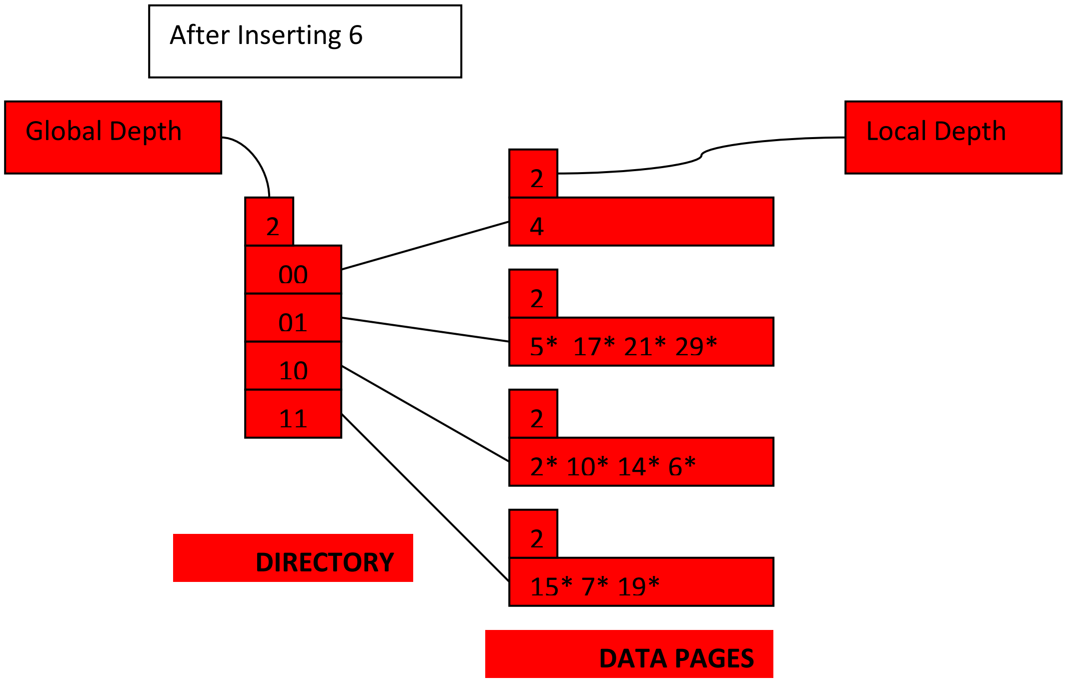
Answer:
r=500,000; R=15 bytes; B=2048 bytes; bfr=2048/15=<u>136</u>; b=500,000/136= <u>3677</u>
a. O(b) = 3677
b. O(log(b) + s/bfr - 1) = O(12 + 50,000/136 – 1) = O(12 + 368 - 1) = <u>379</u>

Q2. Assume the extendible hash index in the figure. Each bucket overflow leads to a split. Draw the index after the insertion of record with the search keys: **6, 9** (*<u>two diagrams one after inserting each record are required</u>*).

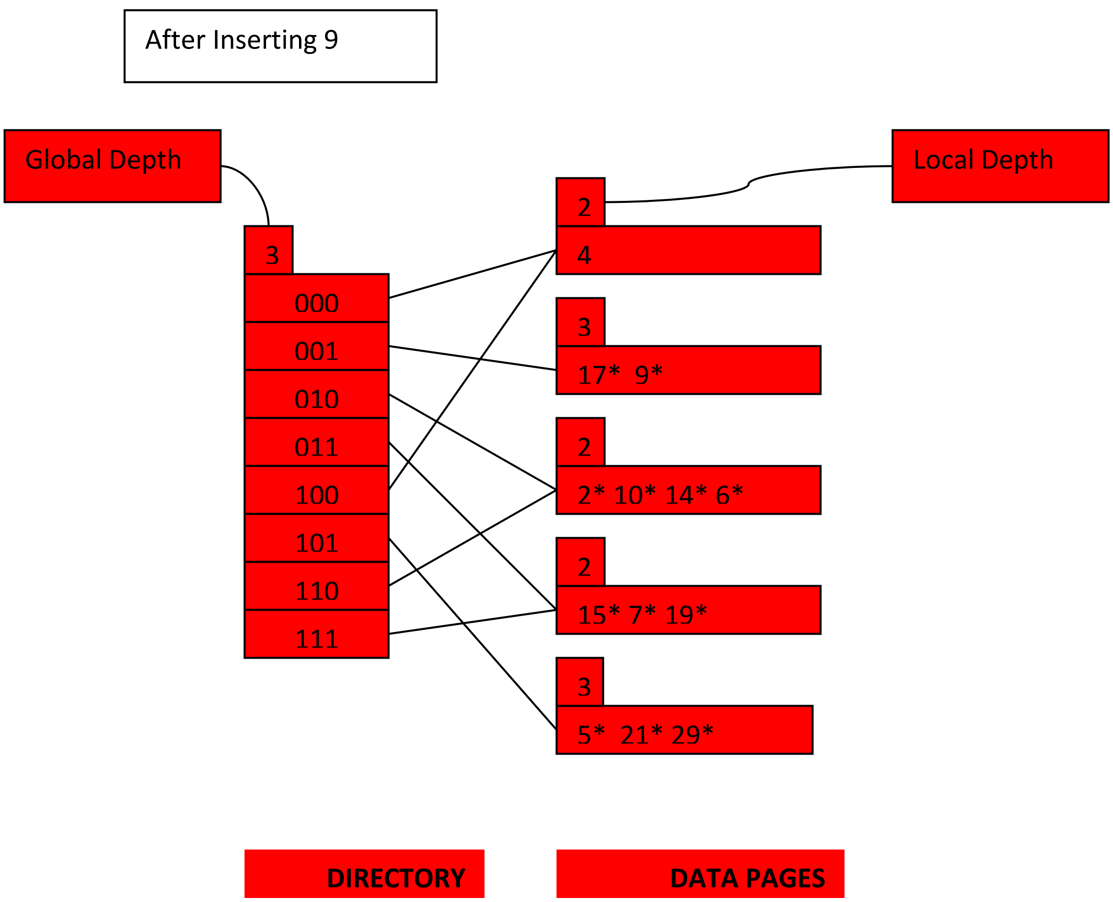
- Directory array size (i.e. bucket size) is 4
- To find the Bucket for r, take last 'global depth' number of bits of *h(r)*, we denote r by *h(r)*. If *h(r)* = 5 = binary 101 it is in the bucket pointed to by 01 (*<u>least significant bits</u>*).



Answer:



After Inserting 6
Global Depth
Local Depth
2
00
01
10
11
2
4
2
5* 17* 21* 29*
2
2* 10* 14* 6*
2
15* 7* 19*
DIRECTORY
DATA PAGES



After Inserting 9
Global Depth
Local Depth
3
000
001
010
011
100
101
110
111
2
4
3
17* 9*
2
2* 10* 14* 6*
2
15* 7* 19*
3
5* 21* 29*
DIRECTORY
DATA PAGES

Q3. Assume the linear hashing index in the figure. Insert the records with the following search keys: **22, 21, 12**

n

1

Buckets

0	8		
1	5		
18	2	10	6
7	3	19	11
4			

Overflow

14			

h = k mod 4

Answer:

Solution

n

2

Buckets

0	8		
1			
18	2	10	6
7	3	19	11
4	12		
5	21		

Overflow

14	22		

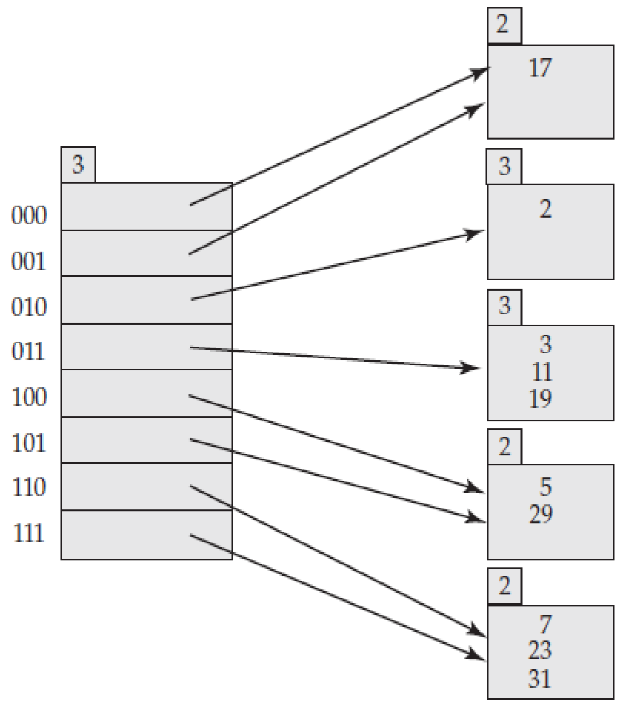
$h_i(k) = k \bmod 4$ [Initial hash function]

$h_{i+1}(k) = k \bmod 8$ [use if hash value m < n]

Q4. Suppose that we are using extendable hashing on a file that contains records with the following search-key values: 5, 7, 11, 17, 18, 19, 23, 27, 37, 39

Show the extendable hash structure for this file if the hash function is *h(k)* = *k* mod 8 and buckets can hold three records. Show your working.

Answer: Extendable hash structure:



Q5. Suppose you are building an extensible hash index on a table of 25,000 rows. Key values are 8 bytes, a pointer (block/record) to a row is 8 bytes, and a disk block is 2048 bytes. Assume all keys are distinct.

a. What is the (lowest possible) global depth? Provide valid reasons.

b. What is the average occupancy of a bucket, assuming all buckets have a local depth equal to the global depth from part (a)? Justify your answer.

Answer:

a. Bucket entries will be key/pointer pairs, so 16 bytes each. Floor(2048/16) = 128 entries / bucket. 25,000/128 = at least 196 buckets needed. Since the directory is always a power of 2 size, it will have at least 2^8 = 256 entries, so the global depth is 8.

b. If all buckets have local depth equal to global depth, then every pointer in the directory points to a unique bucket. Thus, there are 256 buckets. 256 * 128 = capacity of 32,768. 25,000/32,768 ~= 76.3% occupancy.

Q6. Consider the following values: 5, 7, 12, 11, 9.

a. Show the extendable hash structures when the above hash values are added in the file (in order) and buckets can hold 2 records. Show your working.

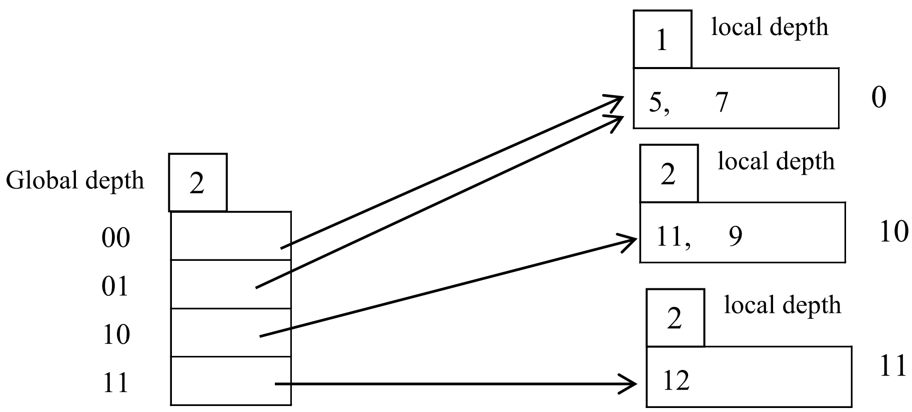
b. Show the dynamic hash structures when the above hash values are added in the file (in order) and buckets can hold 2 records. Show your working.

c. Show the linear hash structures when the above hash values are added in the file (in order), start with empty table with 2 buckets (M = 2), split = 0, and a load factor threshold = 0.9. Splitting must be controlled by monitoring the file load factor with *l* = *r/N*, where r is the current number of file records, and N is the current number of file buckets. Use the mod hash function, first (initial hash function) h_0 = K mod M, second h_1 = K mod 2M, etc.

Answer:

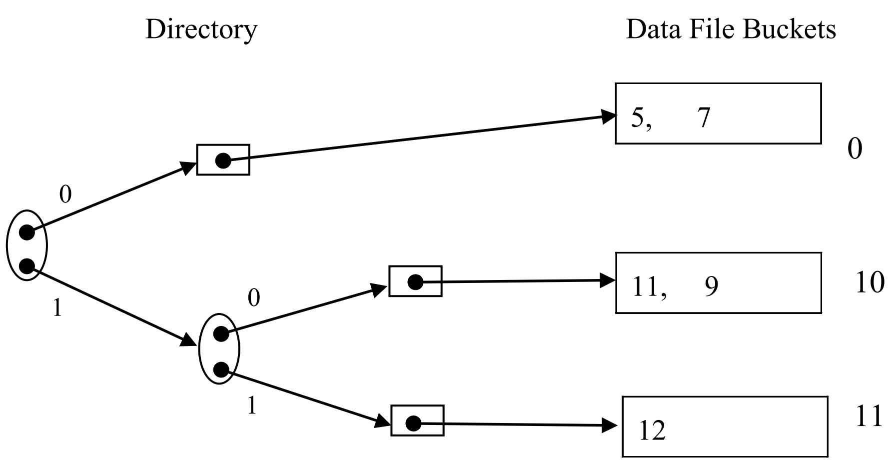
a. Extendible Hashing

h(K) values: **5 (0101), 7 (0111), 12 (1100), 11 (1011), 9 (1001).**



b. Dynamic Hashing

h(K) values: **5 (0101), 7 (0111), 12 (1100), 11 (1011), 9 (1001).**



c. Answer: Linear Hashing

Next Split at	Bucket no	Hash function	Elements	Comments
0	0	Mod 2		
	1	Mod 2		

5

Next Split at	Bucket no	Hash function	Elements	Comments
0	0	Mod 2		
	1	Mod 2	5	Load factor 0.5<0.9

7

Next Split at	Bucket no	Hash function	Elements	Comments
0	0	Mod 2		
	1	Mod 2	5, 7	Load factor 1>0.9; need split

After the split

Next Split at	Bucket no	Hash function	Elements	Comments
1	0	Mod 4		
	1	Mod 2	5, 7	Load factor .67<0.9;
	2	Mod 4		

12

Next Split at	Bucket no	Hash function	Elements	Comments
1	0	Mod 4	12	
	1	Mod 2	5, 7	Load factor1>0.9; need split
	2	Mod 4		

Next Split at	Bucket no	Hash function	Elements	Comments
0	0	Mod 4	12	
	1	Mod 4	5	Load factor 0.75<0.9;
	2	Mod 4		
	3	Mod 4	7	

After split (Now M=4)

11

Next Split at	Bucket no	Hash function	Elements	Comments
0	0	Mod 4	12	
	1	Mod 4	5	Load factor 1>0.9; Need split
	2	Mod 4		
	3	Mod 4	7, 11	

Next Split at	Bucket no	Hash function	Elements	Comments
1	0	Mod 8		
	1	Mod 4	5	Load factor 0.75<0.9;
	2	Mod 4		
	3	Mod 4	7,11	
	4	Mod 8	12	

Next Split at	Bucket no	Hash function	Elements	Comments
1	0	Mod 8		
	1	Mod 4	5, 9	Load factor 1>0.9; Split
	2	Mod 4		
	3	Mod 4	7,11	
	4	Mod 8	12	

Next Split at	Bucket no	Hash function	Elements	Comments
2	0	Mod 8		
	1	Mod 8	9	
	2	Mod 4		
	3	Mod 4	7,11	
	4	Mod 8	12	
	5	Mod 8	5	