**CS443 -- Lab 5**

**Question 1:**

A PARTS file with Part# as hash key includes records with the following Part# values: 2369, 3760, 4692, 4871, 5659, 1821, 1074, 7115, 1620, 2428, 3943, 4750, 6975, 4981, 9208. The file uses 8 buckets, numbered 0 to 7. Each bucket is one disk block and holds two records. Load these records into the file in the given order using the hash function h(K)=K mod 8. Calculate the average number of block accesses for a random retrieval on Part#.

**Hash values for h(K) = K mod 8:**

h(2369) = 1 h(3760) = 0 h(4692) = 4 h(4871) = 7 h(5659) = 3

h(1821) = 5 h(1074) = 2 h(7115) = 3 h(1620) = 4 h(2428) = 4

h(3943) = 7 h(4750) = 6 h(6975) = 7 h(4981) = 5 h(9208) = 0

|  |  |  |  |
| --- | --- | --- | --- |
| **Bucket** | **Position 1** | **Position 2** | **Overflow** |
| **0** | 3760 | 9208 |  |
| **1** | 2369 |  |  |
| **2** | 1074 |  |  |
| **3** | 5659 | 7115 |  |
| **4** | 4692 | 1620 | 2428 |
| **5** | 1821 | 4981 |  |
| **6** | 4750 |  |  |
| **7** | 4871 | 3943 | 6975 |

Two of the records, 2428 and 6975, can not go into the bucket determined by their hash value since each is the third parts record with the same hash value and each bucket only holds two records. These two records must go into an overflow bucket.

The average number of block accesses for a random retrieval is found by adding the number of blocks needed to find each record and dividing by the number of records. Thirteen of the records will be found with one block access and two will be found with two block accesses. The total is 13 x 1 + 2 x 2 = 17. The total number of parts records is 15 so the average number of block accesses for a random retrieval is 17/15 = 1.1333. This assumes that the two overflow records are stored in the same overflow block or that the overflow block is known from the hash value.

**Question 2:**

Load the records of Question 1 into expandable hash files based on extendible hashing. Show the structure of the directory at each step. Show the directory at each step, and the global and local depths. Use the hash function h(k) = K mod 32 to find the bucket number. For example,

***Record# K h(K) bucket number binary h(K)***

record1 2369 1 00001

record2 3760 16 10000

record3 4692 20 10100

**Complete the table for reference:**

record4 4871 7 00111

record5 5659 27 11011

record6 1821 29 11101

record7 1074 18 10010

record8 7115 11 01011

record9 1620 20 10100

record10 2428 28 11100

record11 3943 7 00111

record12 4750 14 01110

record13 6975 31 11111

record14 4981 21 10101

record15 9208 24 11000

The following figures show the records being inserted one at a time in the order shown above. The new record in each step is shown in bold; the global depth is d and the local depth for each bucket is d’.



Figure 1 The first four records can be inserted at depth 1



Figure 2 The next three records end with all levels at 2



Figure 3 The next two records end with some levels at 3



Figure 4 The tenth record increases the global depth to 4



Figure 5 The eleventh record can be inserted easily



Figure 6 The twelfth record can be inserted easily



Figure 7 The thirteenth record creates a new bucket with a depth of 4



Figure 8 The fourteenth record creates a new bucket with a depth of 3



Figure 9 The fifteenth record can be inserted easily



Figure 10 This is the final directory redrawn for clarity

**Question 3:**

Insert the following into B-tree of order **4**. Show your work step by step with proper illustration of pointers as shown in pages 20-29 in multi-way trees lecture

**8, 2, 80, 25, 26, 27, 28, 55, 71, 15, 51, 20, 21, 22, 23, 24, 63, 90, 35**

The figures below illustrate the index keys listed above being inserted into a B-tree of order 4. The index key inserted in each step is shown in bold.











**Question 4:**

Insert the following into B+ tree of order **3**. Show your work step by step with proper illustration of pointers as shown in pages 47-54 in multi-way trees lecture

**90, 22, 27, 24, 28, 20, 51, 63, 8, 80, 15, 71, 35, 55**

The figures below illustrate the index keys listed above being inserted into a B+ tree of order 3. The index key inserted in each step is shown in bold.









**Question 5:**

Consider the following B-tree.

a) Redraw the tree after deleting 14.

b) Again, redraw the tree after deleting 15.

c) Again, redraw the tree after deleting 25.

**1**

**2**

**3**

**8**

**24**

**23**

**37**

**5**

**6**

**7**

**13**

**14**

**15**

**20**

**18**

**22**

**25**

**16**

**27**

The figures below illustrate the results of the required actions. The first figure is the original figure redrawn for reference in PowerPoint.

a) 14 can simply be deleted since there are still enough keys in the leaf.

b) Deleting 15 with 14 already deleted leaves only one key in the leaf so the keys in the sibling to the left are redistributed so that both leaves have enough keys.

c) Deleting 25 takes several steps:

1. To avoid deleting a key from a non-leaf node, swap 25 with its immediate predecessor 24.
2. Delete key 25.
3. Merge the two nodes so the node containing 23 has more than 1 key.
4. Merge the two nodes so the node containing 22 has more than 1 key.





**Question 6:**

Consider the following B+ tree:

**2**

**9**

**26**

**24**

**53**

**22**

**17**

**10**

**30**

**22**

**29**

**10**

**29**

**65**

1. Delete 29 and redraw the tree after that
2. Next delete 10 and redraw the tree after that
3. Next delete 26 and redraw the tree after that
4. Next delete 24 and redraw the tree after that
5. Next delete 9 and redraw the tree after that

**2**

**9**

**26**

**24**

**53**

**22**

**17**

**10**

**30**

**22**

**29**

**10**

**29**

**65**

1. Delete 29 and redraw the tree after that.

Delete 29 does not cause underflow. It is simply deleted. 29 does not have to deleted in internal node, since it is just a separator.

10

22

29

2

9

10

12

22

24

26

30

53

65

1. Next delete 10 and redraw the tree after that

Delete 10 does not cause underflow. It is simply deleted. 10 does not have to deleted in internal node, since it is just a separator.

10

22

29

2

9

12

22

24

26

30

53

65

1. Next delete 26 and redraw the tree after that

Delete 26 causes underflow. 24, 30, 53, 65 are redistributed. Separator 29 is removed.

10

22

30

2

9

12

22

24

30

53

65

1. Next delete 24 and redraw the tree after that

Delete 24 causes underflow. The leaf is deleted. 30, 53, 65 are redistributed in sibling.

10

22

2

9

12

22

30

53

65

1. Next delete 9 and redraw the tree after that

Delete 9 causes underflow. The leaf is deleted. 2, 12, 22 are redistributed in sibling. Separator 10 is removed.

22

30

53

65

2

122

22