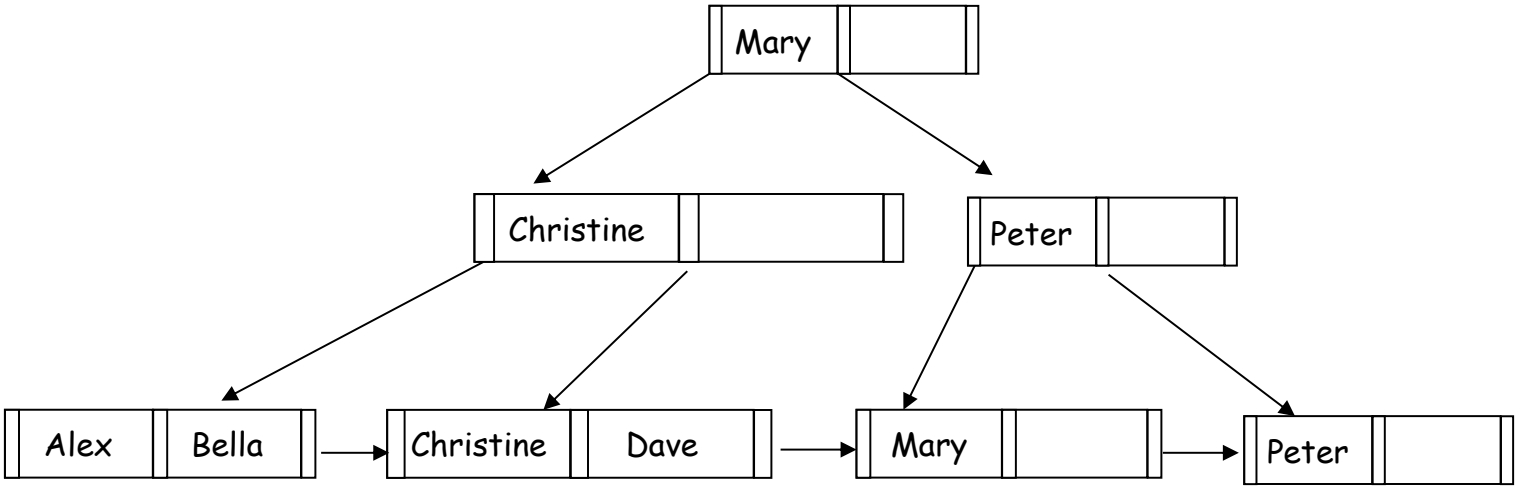
CS1555 Recitation 13 Supplemetary

Objective: To practice B+-tree and Hashing

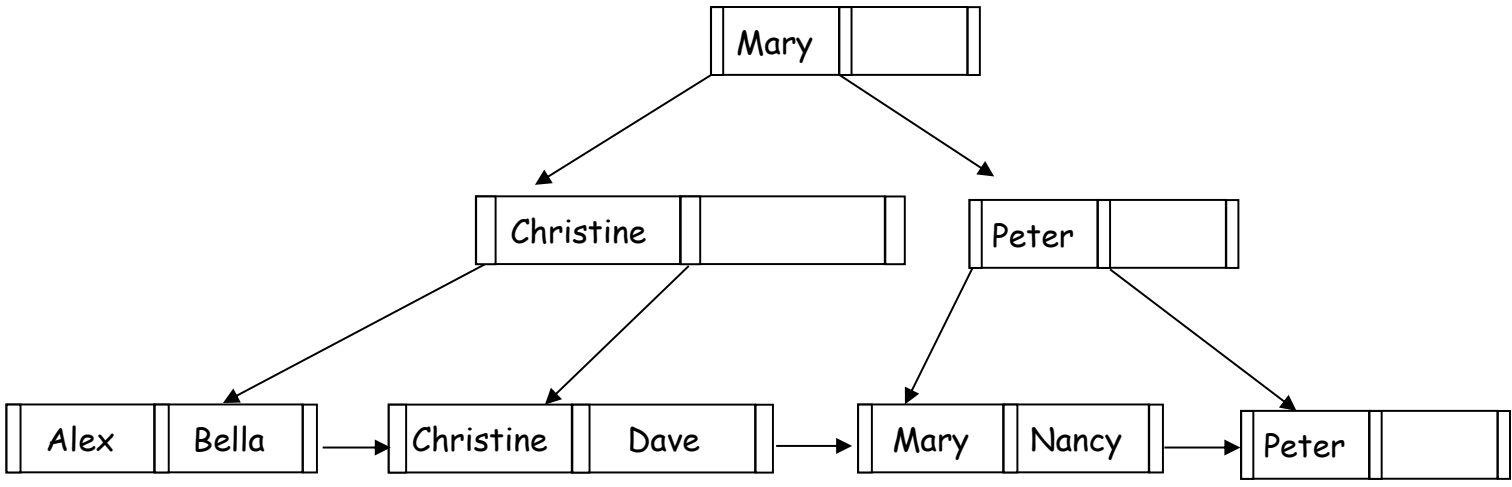
Part 1: B+Tree

Build the B+ Tree maintaining the index on the name of students (n=3), with the following items: Alex, Christine, Bella, Mary, Peter, Dave.

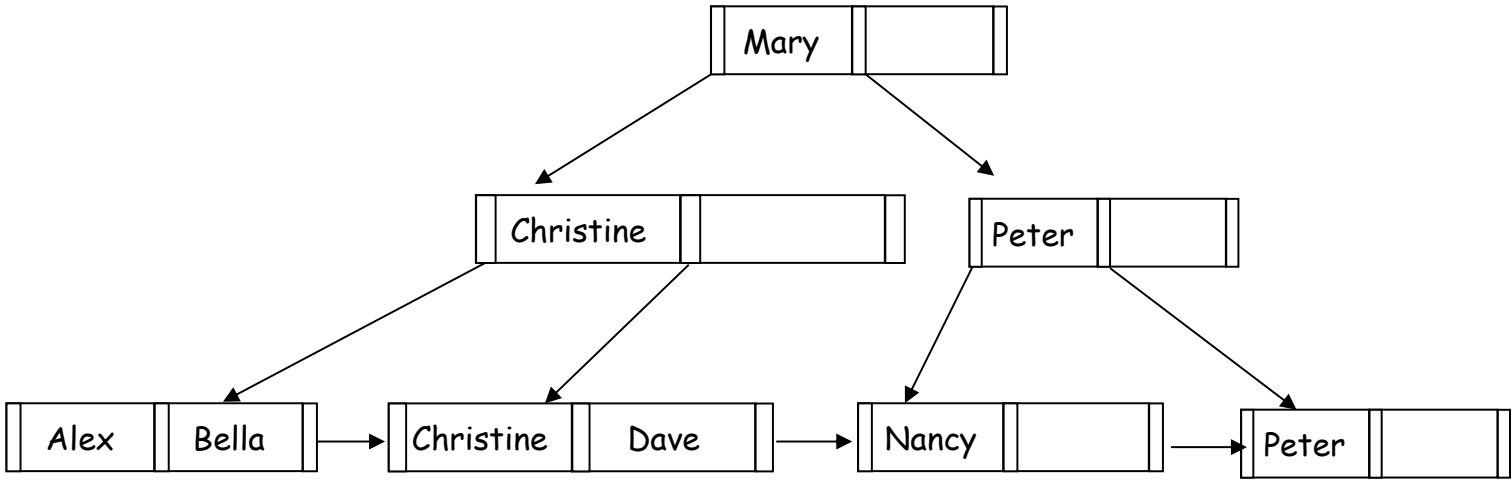
Suppose we use the following variation:
a) Left arrow indicates "<", while right arrow indicates ">=".
b) When splitting an odd number of elements, the new left node has one more than the right node.



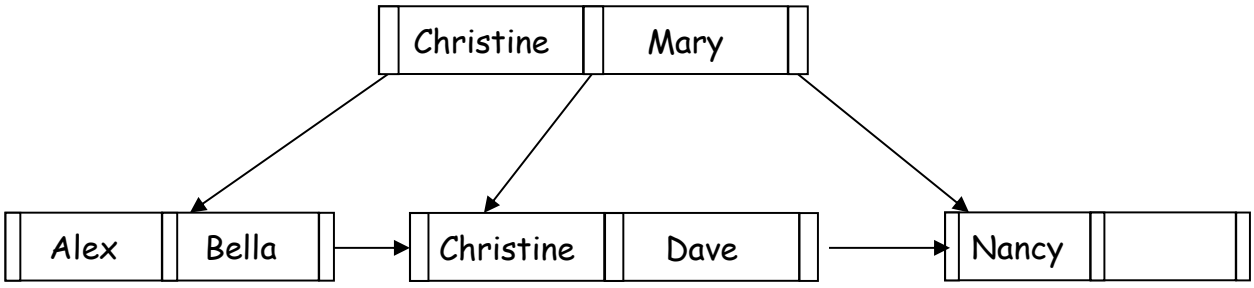
2. a) Add Nancy to the tree



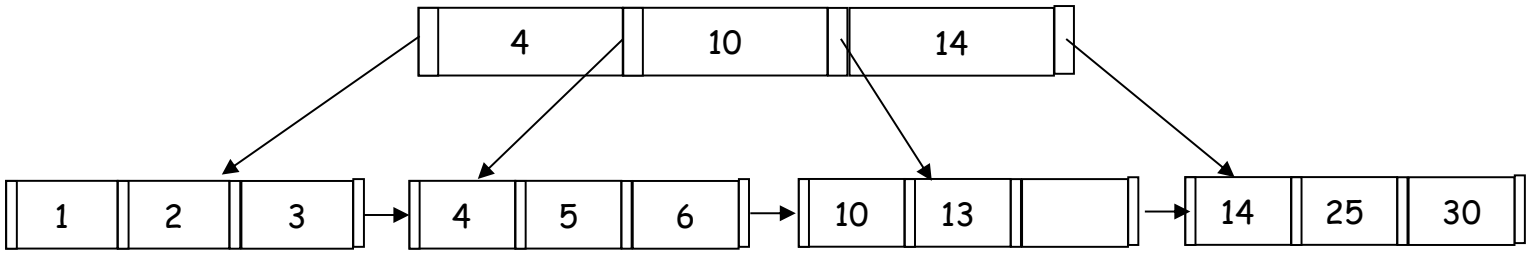
b) Delete “Mary” from the tree: because “Mary” also appears in the internal node, after deletion we pick the left most element (smallest – Nancy) of the node's right sub-tree to replace “Mary”



c) Delete “Peter” from the tree, which results in cascade node merging.



3. Build a B+ tree for n=4 for the following keys (2, 3, 4, 5, 14, 10, 6, 25, 13, 30, 1)



Part 2: Linear Hashing

1. Consider the following record keys: (3, 2, 1, 8, 6, 4, 14, 5, 9). Create the linear hashing structure for the above records after each split assuming $h_0(k) = k \bmod 4$, bfr=2 and:

a) no overflow buckets.
b) 1 overflow bucket.

a) no overflow bucket

Bsplit = 0 Blast =3

8	1	2	3
4		6	
0	*1*	*2*	*3*

Insert 14: Overflow in bucket 2→ start splitting at Bsplit (bucket 0) using h1(k) = k mod 8

Bsplit = 1 Blast =4

8	1	2	3	4
		6		
0	*1*	*2*	*3*	*4*

Continue splitting (bucket 1) since we have not solved the overflow yet, using h1(k)

Bsplit = 2 Blast = 5

8	1	2	3	4	
		6			
0	*1*	*2*	*3*	*4*	*5*

Continue splitting (bucket 2) since we have not solved the overflow yet, using h1(k)

Bsplit = 3, Blast = 6

8	1	2	3	4		6
						14
0	*1*	*2*	*3*	*4*	*5*	*6*

Insert 5: $h_0(5) = 1$, since 1 < Bsplit, compute h1(5) = 5: insert 5 to bucket 5

Bsplit = 3, Blast = 6

8	1	2	3	4	**5**	6
						14
0	*1*	*2*	*3*	*4*	*5*	*6*

Insert 9: $h_0(9) = 1$, since 1 < Bsplit, compute h1(9) = 1: insert 9 to bucket 1

Bsplit = 3, Blast = 6

8	*1*	*2*	*3*	*4*	*5*	*6*
	9					*14*
0	*1*	*2*	*3*	*4*	*5*	*6*

Note that when Blast = s-1 where h1(k) = k mod s, we reset $h_0(k) = h_1(k)$ and Bsplit =0

b) 1 overflow bucket.
Insert 14: overflow, an overflow bucket is given to bucket 2, no splitting is necessary

Bsplit = 0, Blast = 3

4	1	2	3
8		6	
0	*1*	*2*	*3*

14

Insert 5 and then 9: 9 causes overflow, since there is no more bucket available, we need to split bucket 0 and then bucket 1

Split bucket 0 using h1(k) = K mod 8,

Bsplit = 1, Blast = 4

8	1	2	3	4
	5	6		
0	*1*	*2*	*3*	*4*

14

Split bucket 1 using h1(k)

Bsplit = 2, Blast = 5

8	1	2	3	4	5
	9	6			
0	*1*	*2*	*3*	*4*	*5*

14

Note that later on if the bucket 2 is split, the overflow bucket might be freed and become available to be used by another entry. In order to see that let us insert two new values: 10, 17

Insert 10: h0(10) = 10 mod 4 =2

8	1	2	3	4	5
	9	6			
0	*1*	*2*	*3*	*4*	*5*

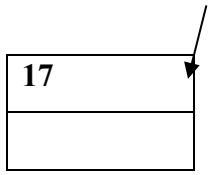
14
10

Insert 17: h0(17) =17 mod 4 =1; Since 1<BSplit use h1(17)=17%8=1

But Bucket #1 is full -> Split Bucket 2
h1(2) =2%8 =2
h1(6) =6%8 =6
h1(14) =14%8 =6
h1(10) =10%8 =2

Then Insert 17 which goes in bucket 1. Because Bucket 1 is full, and we have an empty overflow bucket we will use this one to add value 17.

8	1	2	3	4	5	6
	9	10				14



Part 3: Extendible Hashing

1. **(Exercise 11.1)** Consider the Extendible Hashing index shown in Figure 11.1. Answer the following questions about this index:
   i. What can you say about the last entry that was inserted into the index?
   ii. What can you say about the last entry that was inserted into the index if you know that there have been no deletions from this index so far?
   iii. Suppose you are told that there have been no deletions from this index so far. What can you say about the last entry whose insertion into the index caused a split?
   iv. Show the index after inserting an entry with hash value 68.
   v. Show the index after inserting entries with hash values 17 and 69 into the original tree.
   vi. Show the index after deleting the entry with hash value 21 into the original tree. (Assume that the full deletion algorithm is used.)
   vii. Show the index after deleting the entry with hash value 10 into the original tree. Is a merge triggered by this deletion? If not, explain why. (Assume that the full deletion algorithm is used.)

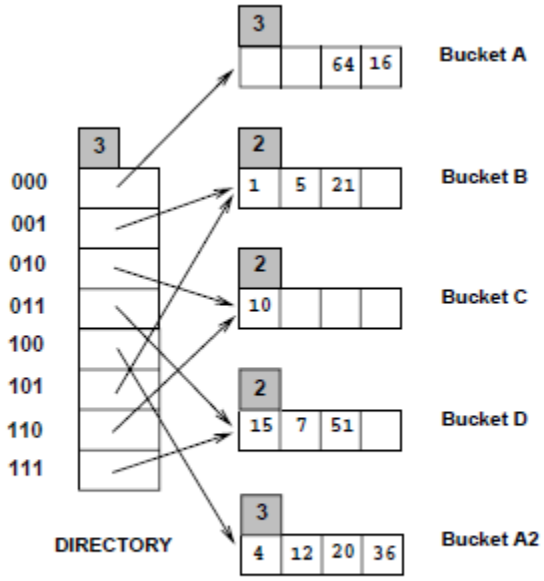


Figure 11.1 Figure for Exercise 11.1

***Answers**:*

i. *It could be any one of the data entries in the index. We can always find a sequence of insertions and deletions with a particular key value, among the key values shown in the index as the last insertion.*

For example, consider the data entry 16 and the following sequence:

1 5 21 10 15 7 51 4 12 36 64 8 24 56 16 56D 24D 8D

ii. *The last insertion could not have caused a split because the total number of data entries in the buckets A and A2 is 6. If the last entry caused a split the total would have been 5.*

iii. *The last insertion which caused a split cannot be in bucket C. Buckets B and C or C and D could have made a possible bucket-split image combination but the total number of data entries in these combinations is 4 and the absence of deletions demands a sum of at least 5 data entries for such combinations. Buckets B and D can form a possible bucket-split image combination because they have a total of 6 data entries between themselves. So do A and A2. But for the B and D to be split images the starting global depth should have been 1. If the starting global depth is 2, then the last insertion causing a split would be in A or A2.*

iv. *Figure 11.2*

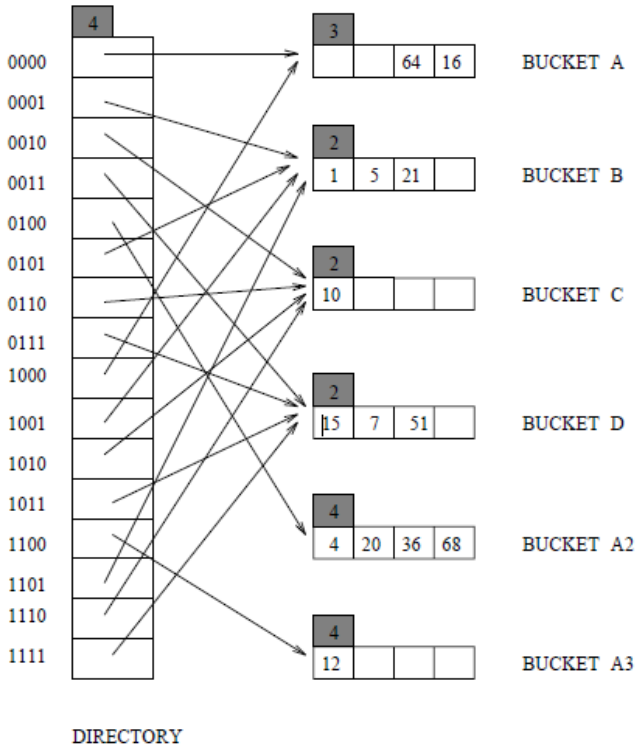


Figure 11.2

v. *Figure 11.3*

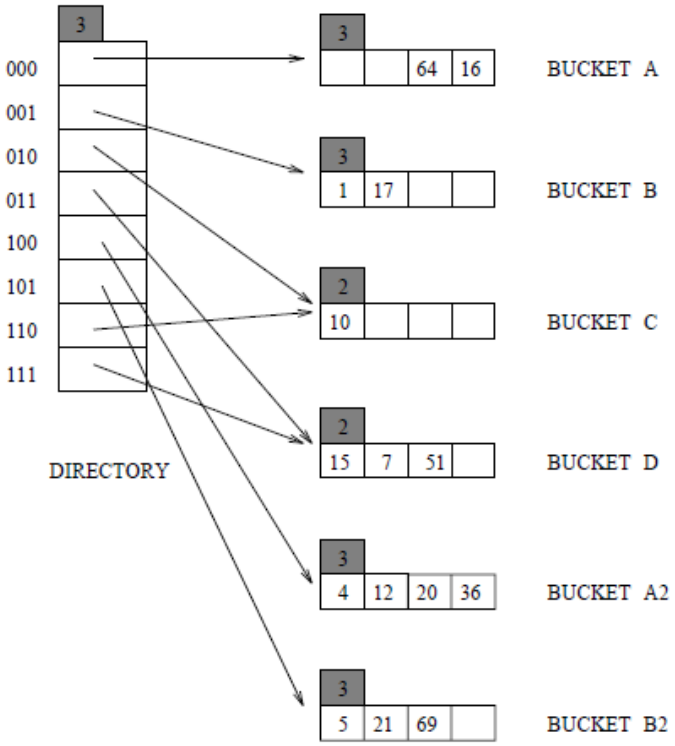


Figure 11.3

vi. *Figure 11.4*

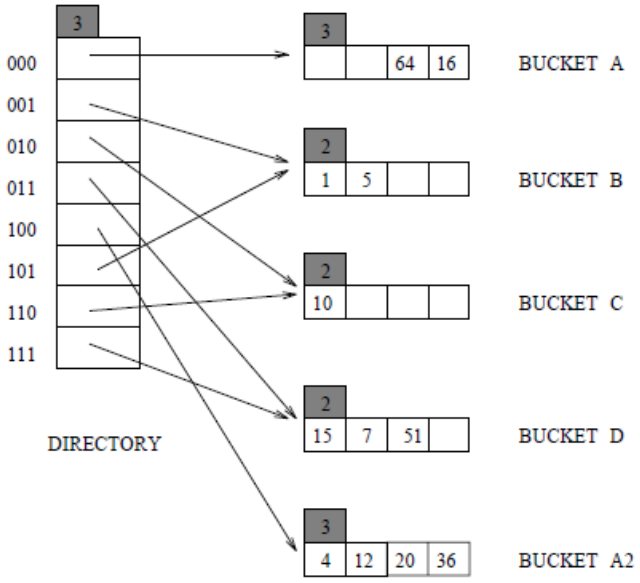


Figure 11.4

vii. The deletion of the data entry 10 which is the only data entry in bucket C doesn't trigger a merge because bucket C is a primary page and it is left as a place holder. Right now, directory element 010 and its split image 110 already point to the same bucket C. We can't do a further merge. See Fig 11.5

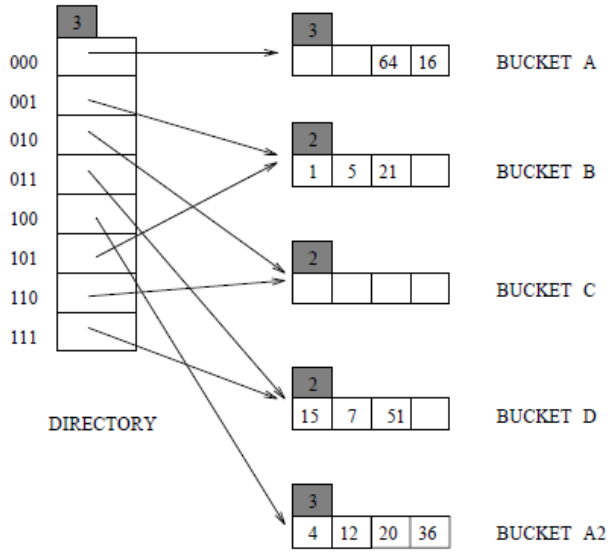


Figure 11.5