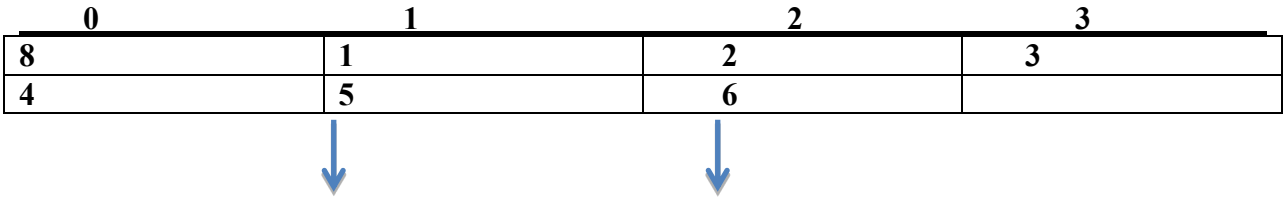
CS1555/CS2055 Recitation 10 Solution

Objective: Practice operations on static hashing, extendible hashing and B+-tree

Part 1. Hash Files

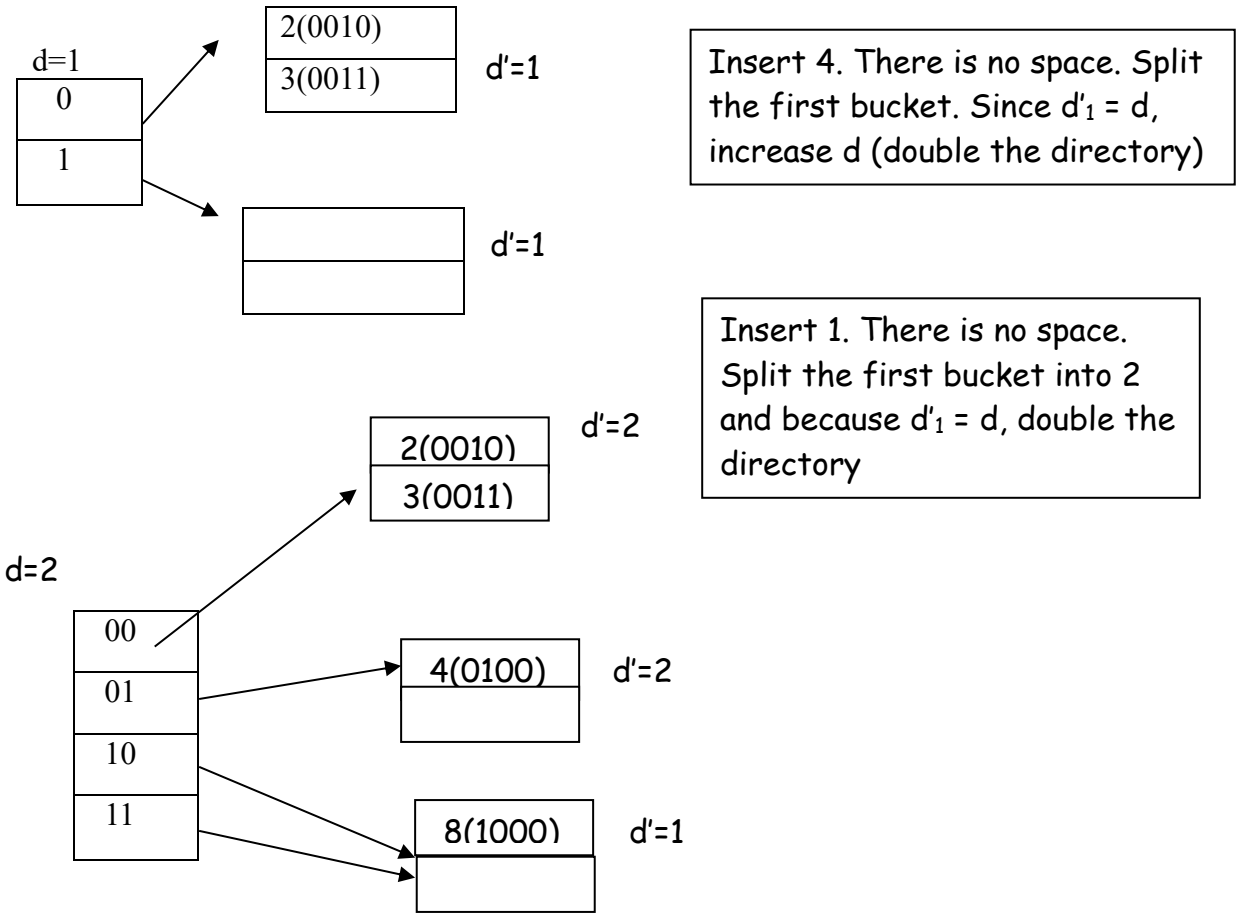
1) Static Hashing

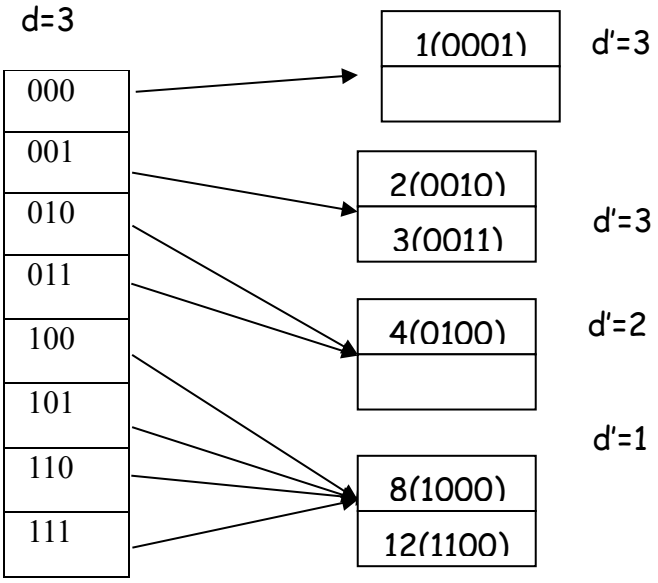
Consider the following record keys: (3, 2, 1, 8, 6, 4, 14, 5, 9). Create the static hash structure, with M=4 main buckets, that will contain the provided records, using the chaining technique. Use h(k) =k mod M as a hashing function. Each bucket can hold 2 records.

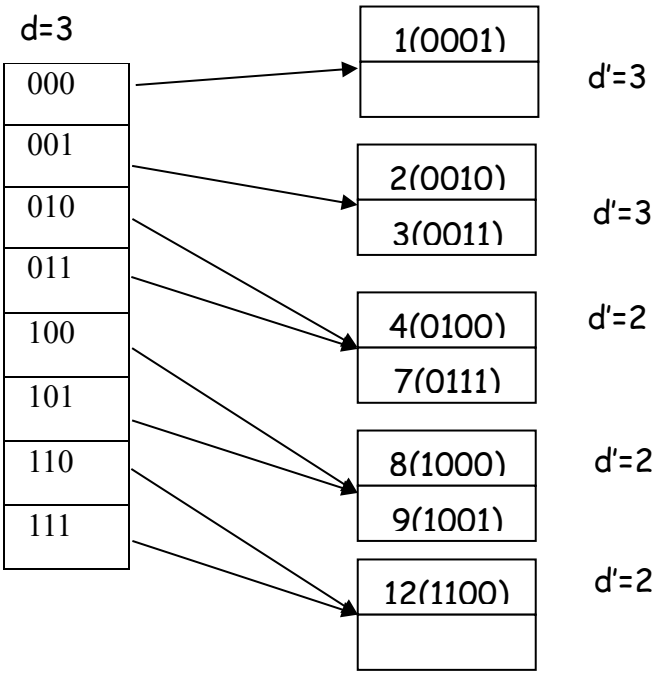


2) Extendible Hashing

Create an extensible hash structure for these record keys: (2, 3, 4, 8, 1, 12, 9, 7) using the most significant bits. Use the 4 bit binary representation of the keys (2=0010, 3=0011, 4=0100, 8=1000, 1=0001, 12=1100, 9=1001, 7=0111). Use a bfr=2.







Insert 9. There is no space. Split the fourth bucket into 2 and because $d'_1 < d$, the directory is not split.

3) 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

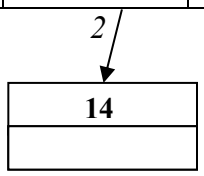
8	*1*	*2*	*3*	*4*	*5*	*6*
	9					*14*
0	*1*	*2*	*3*	*4*	*5*	*6*

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*

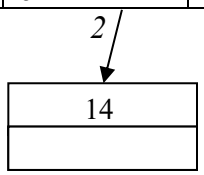


Insert 5 and then 9: 9 causes overflow, since there is no more bucket available, we need to split bucket 0 and then bucket 1

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

Bsplit = 1, Blast = 4

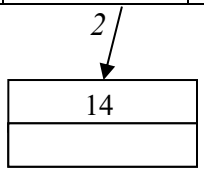
8	1	2	3	4
	5	6		
0	*1*	*2*	*3*	*4*



Split bucket 1 using h1(k)

Bsplit = 2, Blast = 5

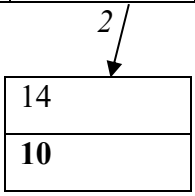
8	1	2	3	4	5
	9	6			
0	*1*	*2*	*3*	*4*	*5*



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*

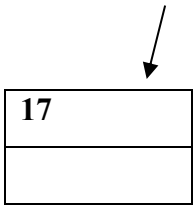


Insert 17: h0(17) =17 mod 4 =1; Since 1<BSplit use h1(17)=17%8=1

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.

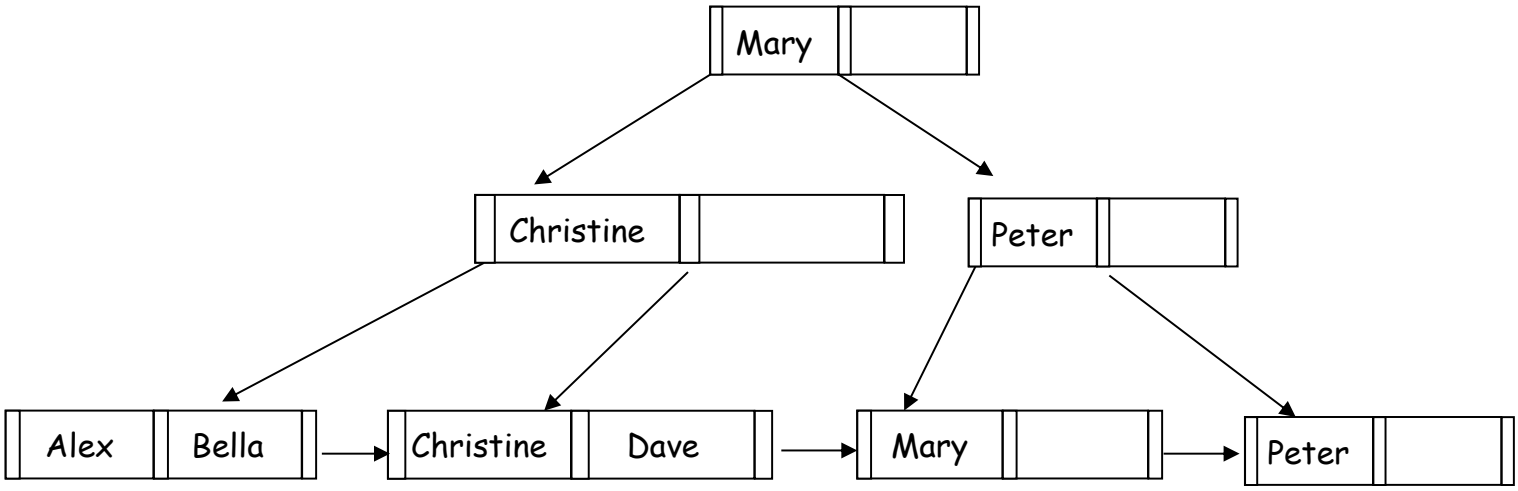
8	1	2	3	4	5	6
	9	10				14



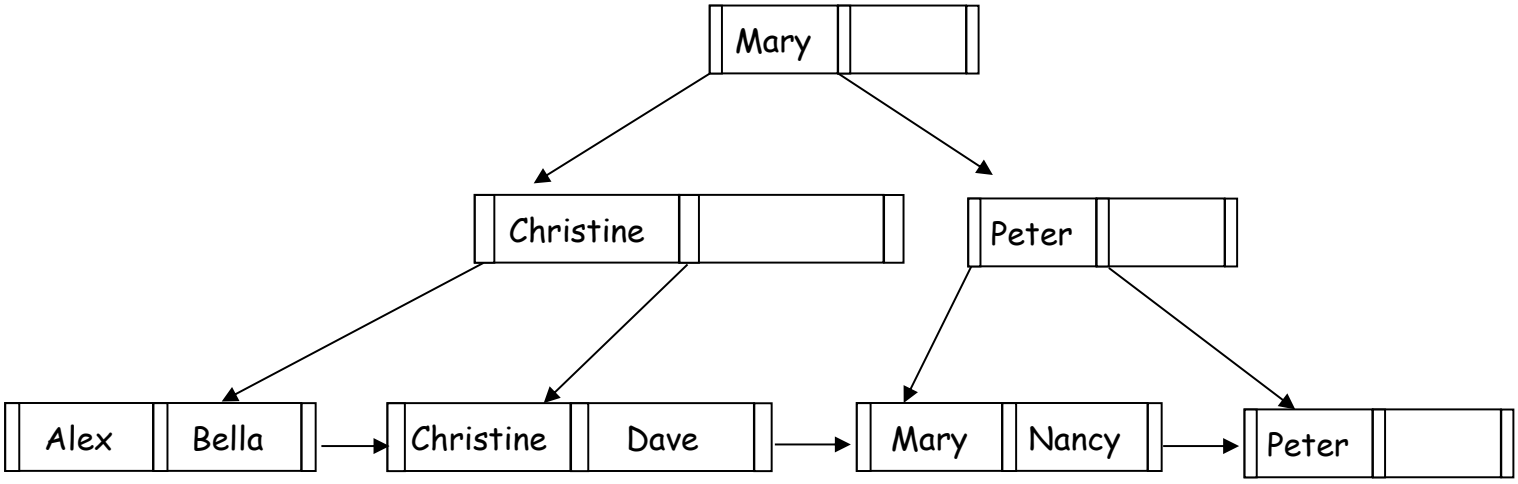
Part 2: B+ Tree

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

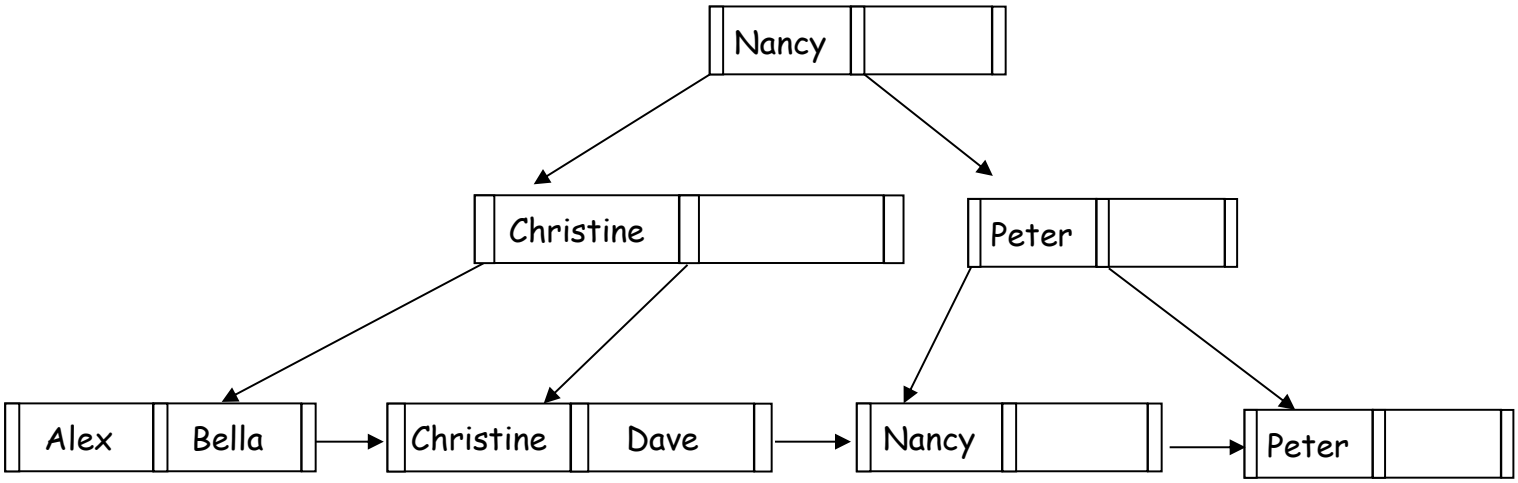
Suppose we use the following variation:
a) Left arrow indicates "<", while right arrow indicates ">=".
b) When splitting an odd number of elements, the new left node has one more than the right node.



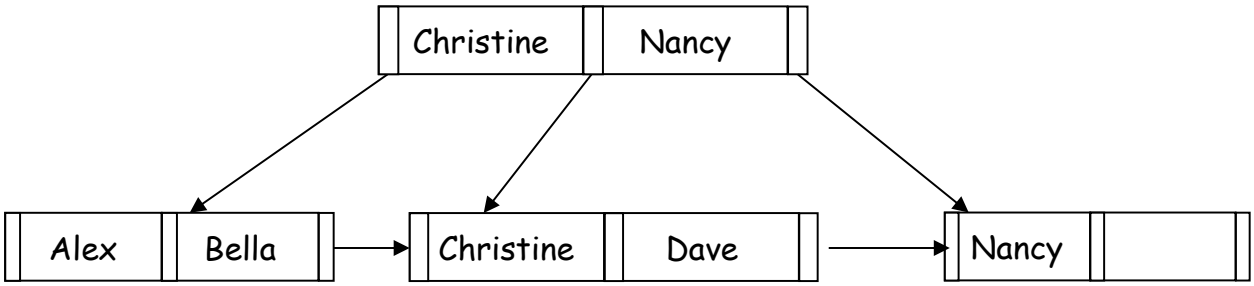
2. a) Add Nancy to the tree



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)