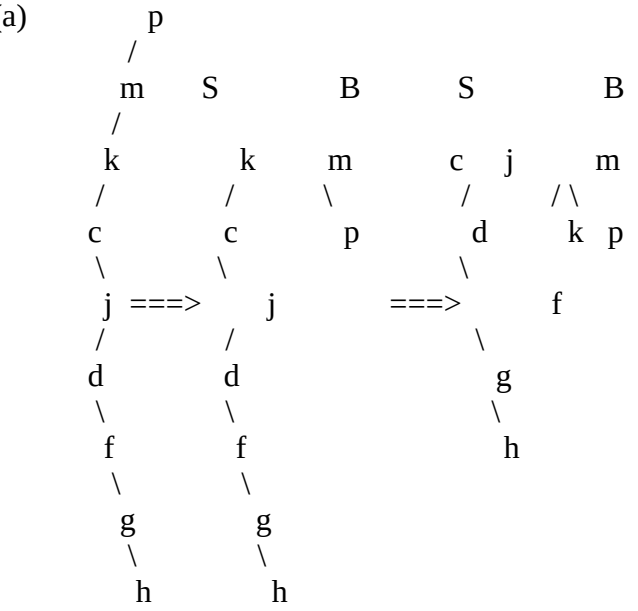
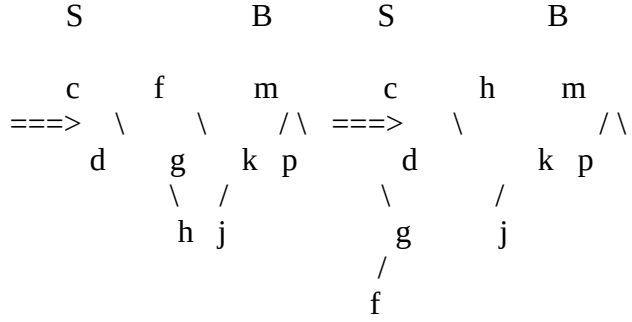
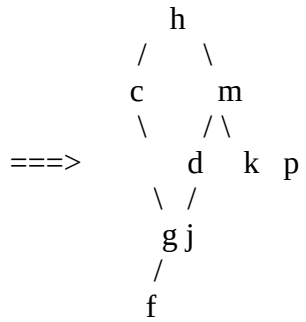
1.

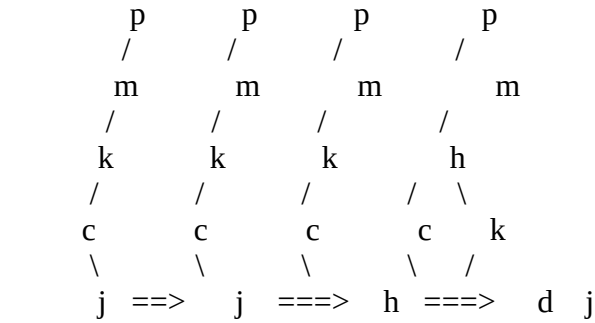
(a)

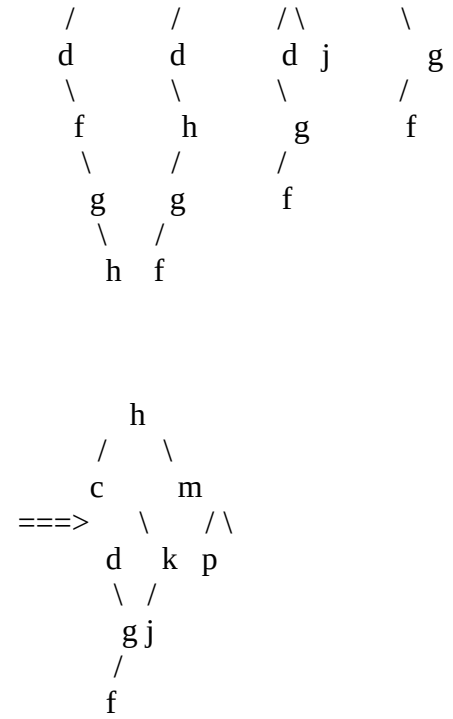

p
m S B S B
k k m c j m
c c p d k p
j ===> j ===> f
d d g
f f h
g g
h h


S B S B
c f m c h m
===> d g k p ===> d k p
h j g j
f


h
c m
===> d k p
g j
f

(b)


p p p p
m m m m
k k k h
c c c c k
j ==> j ===> h ===> d j



2.

(a) f1(17) = 17 mod 13 = 4
f2(17) = 2*17 mod 13 = 8

bit	0	1	2	3	4	5	6	7	8	9	10	11	12
	0	0	0	0	1	0	0	0	1	0	0	0	0

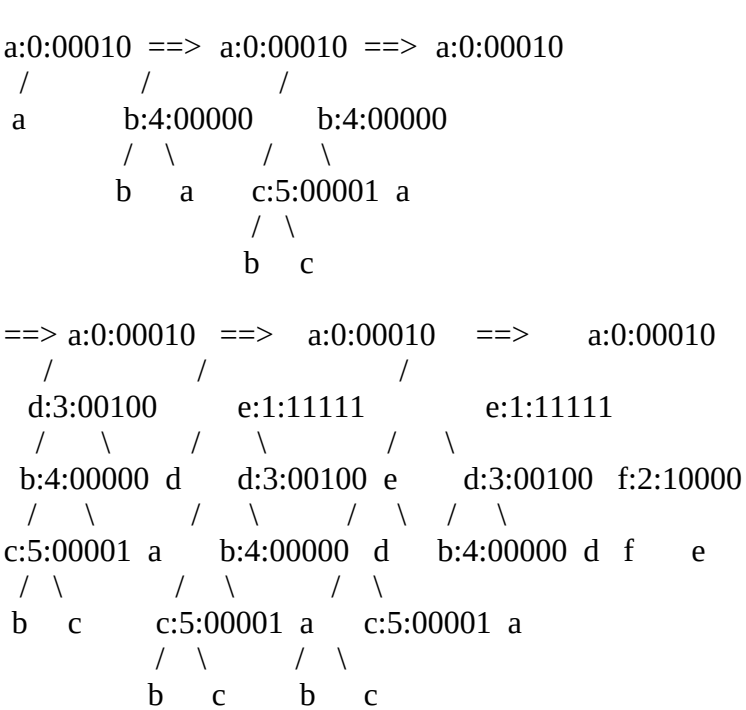
(b) f1(19) = 19 mod 13 = 6
f2(19) = 2*19 mod 13 = 12

bit	0	1	2	3	4	5	6	7	8	9	10	11	12
	0	0	0	0	1	0	1	0	1	0	0	0	1

(c) examples
(1) k=4
f1(4) = 4 mod 13 = 4
f2(4) = 2*4 mod 13 = 8
Thus, the Bloom filter will respond "Maybe" but key "4" is not in the filter.

(2) k=6
f1(6) = 6 mod 13 = 6
f2(6) = 2*6 mod 13 = 12
Thus, the Bloom filter will respond "Maybe" but key "6" is not in the filter.

3. Patricia

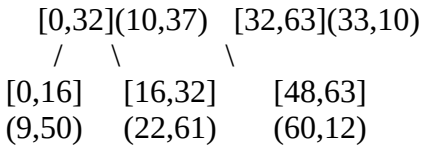


o Node : (NodeLabel:BitNumber:Data)
- NodeLabel is used to show the link of pointer to node.

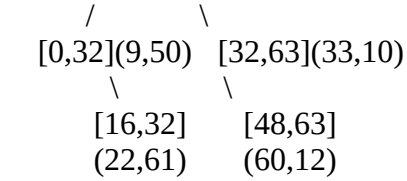
4. A radix priority search tree can be defined as a set of ordered pairs [x,y] over 0..63 of integers maintaining a min-tree on y and a binary search tree on x.

(a) Draw a radix priority search tree which contains pairs of [9,50], [33,10], [20,1], [60,12], [22,61] and [10,37].

(solution)



(b)
(solution)



5. 2D range tree

Assume we have N records with 2 keys, x and y.

A 2-D range tree is a binary tree using the range of x, while in each node we keep a sorted list based on y. The tree always splits the records in half, using a median x key value as the discriminator.

Preprocessing time P:

we need to build a sorted list on y, and another on x. Since we have the sorted list on x, we can find the discriminator easily. And since we have the sorted list on y, we can split the list in linear time on each level (there are logN levels overall). So the preprocessing time is:

P = NlogN + NlogN + N*logN = O(NlogN)

Space required S:

In each node we store an array of the records. At each level the number of records is N. So the space required is:

S = N*logN = O(NlogN)

Query time Q:

Upon each query, we start at the root, decide which branch to take by comparing the boundaries of x with the discriminator. Once we arrive a node, we use binary search on y in the array. Then we report those records that fit the query. So the query time is:

Q = logN*logN + F = O((logN)^2 + F)

where F is the number of records reported.