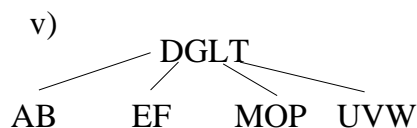
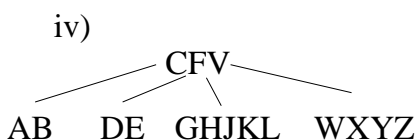
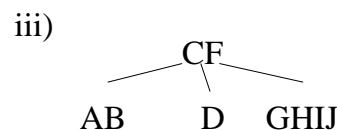
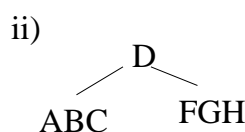
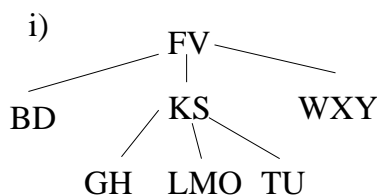


## B-tree Practice Problems

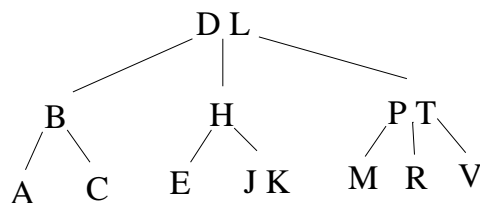
1. Suppose that you have an application in which you want to use B-trees. Suppose that the computer you will be using has disk blocks holding 4096 bytes, the key is 4 bytes long, each child pointer (which is a disk block id) is 4 bytes, the parent is 4 bytes long and the data record reference (which is a disk block id along with a offset within the block) is 8 bytes.

You have an application in which you want to store 1,000,000 items in your B-tree. What value would you select for  $t$ ? (Show how you derived it.) What is the maximum number of disk pages that will be brought into main memory during a search? Remember that the root is kept in main memory at all times.

2. Which of the following are legal B-trees for when the minimum branching factor  $t = 3$ ? For those that are not legal, give one or two sentence very clearly explaining what property was violated.



3. Show the B-tree that results when inserting R,Y,F,X,A,M,C,D,E,T,H,V,L,W,G (in that order) branching factor of  $t = 3$ . You need only draw the trees just before and after each split.
4. Show the B-tree the results when deleting A, then deleting V and then deleting P from the following B-tree with a minimum branching factor of  $t = 2$ .



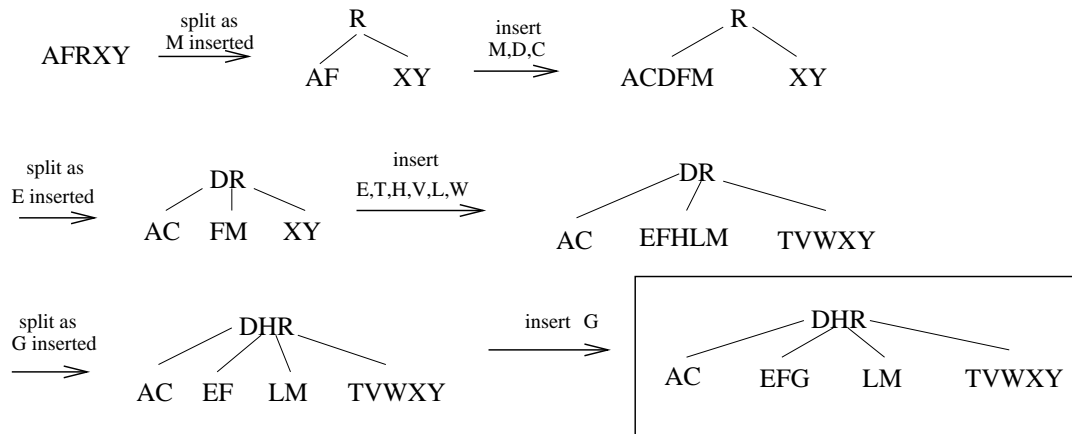
5. Show a way to represent the original B-tree from problem 4 as a red-black tree. You can indicate the color of each node by circling it with red or black or just by putting a  $RB$  or  $BB$  next to it.

The solutions are on the back (except 5 which is on the web page since it didn't fit). Also, there are some more practice problems on the course webpage under Homeworks. I strongly recommend that you solve these BEFORE looking at the solutions.

## SOLUTIONS:

1. We want to select  $t$  so that a full node uses as much of a disk block as possible. In a full node there are  $2t$  keys (4 bytes each),  $2t$  data record references (8 bytes each),  $2t$  child pointers (4 bytes each), a parent pointer (4 bytes), the number of keys (4 bytes) and the leaf bit (which we go ahead and assume takes 4 bytes though 1 bit would do). Hence we want to pick  $t$  as large as we can so that  $12(2t) + 4(2t) + 12 = 32t - 4096$ . Solving for  $t$  yields that we need  $t = 128$ . In class, we argued that the number of disk pages that must be read ( $d-1$  using the notation from class) is at most  $\log_t(n+1)/2$ . Since  $\log_{128}(n+1)/2 = \log_{128} 2.7$  and the number of levels below the root must be an integer, at most 2 disk pages will need to be brought into main memory during a search.
2. (i): Not legal since the height is not balanced. More specifically, both the node with  $BD$  and  $KS$  are at the same level but  $BD$  is a leaf and  $KS$  is not.  
 (ii): This is legal. Remember, that the root can have just a single key.  
 (iii): Not legal since the key  $DD$  has less than the minimum allowable size of 2 keys.  
 (iv): This is legal.  
 (v): Not legal since there is no leaf node corresponding to the keys between  $G$  and  $L$ .

### 3. B-tree insertion problem



### 4. B-tree deletion problem

