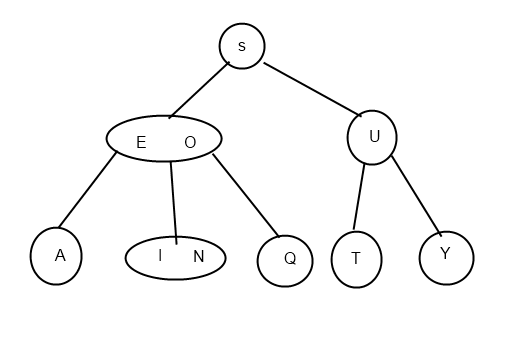
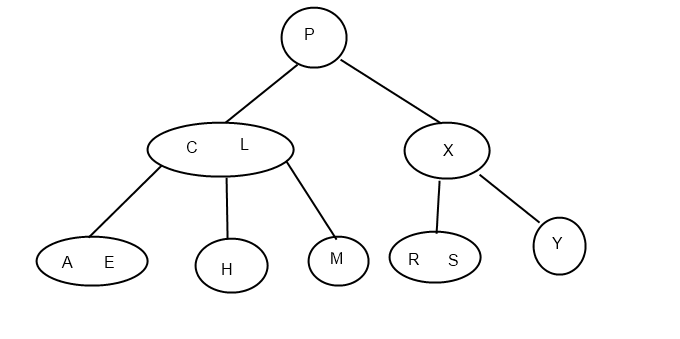
3.3.1 Draw the 2-3 tree that results when you insert the keys E A S Y Q U T I O N in that order into an initially empty tree.



3.3.2 Draw the 2-3 tree that results when you insert the keys Y L P M X H C R A E S in that order into an initially empty tree.



3.3.3 Find an insertion order for the keys S E A R C H X M that leads to a 2-3 tree of height 1.

E R A C H S M X

3.3.4 Prove that the height of a 2-3 tree with N keys is between ~ ⎣log3 N⎦ = .63 lg N (for a tree that is all 3-nodes) and ~⎣lg N⎦ (for a tree that is all 2-nodes).

By the analysis of text we already know a 2-3 tree must be perfect balanced tree.

Its minimum height would be with all nodes are 3 nodes.

A 2-3 tree with full nodes @ height h and every node is a 3 node has totally 3^(h+1) – 1 nodes.

* 3^h – 1 < N <= 3^(h+1) – 1
* 3^h <= N < 3^(h+1)
* Log(3, N) – 1 < h <= Log(3, N)
* h = floor(log(3,N))

Its maximum height would be with all nodes are 2 nodes.

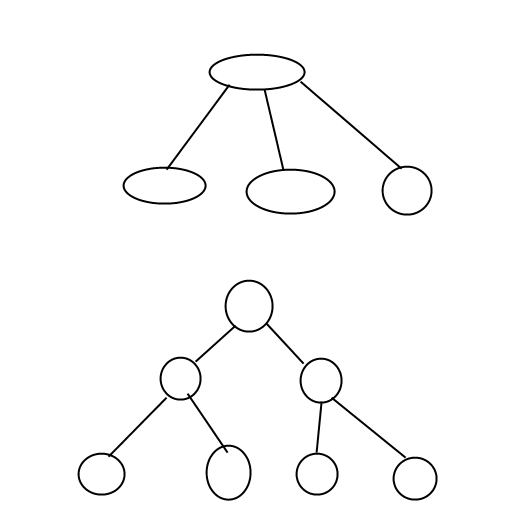
It has totally 2^(h+1) – 1 nodes.

* 2^h – 1 < N <= 2^(h+1) -1
* 2^h <= N < 2^(h+1)
* lgN – 1< h <= lgN
* h = floor(lgN)

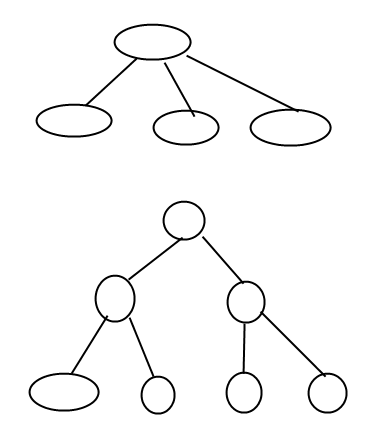
That is to say, log(3,N) <= h <= lgN

3.3.5 The figure at right shows all the structurally different 2-3 trees with N keys, for N from 1 up to 6 (ignore the order of the subtrees). Draw all the structurally different trees for N = 7, 8, 9, and 10.

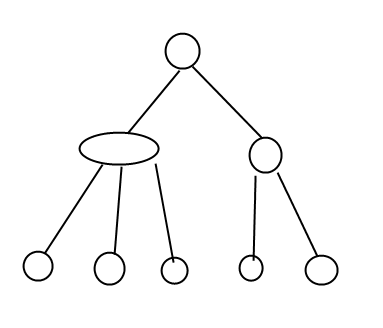
7

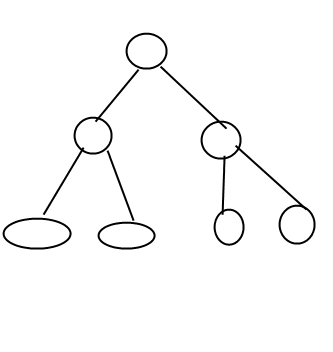


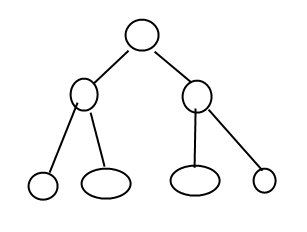
8



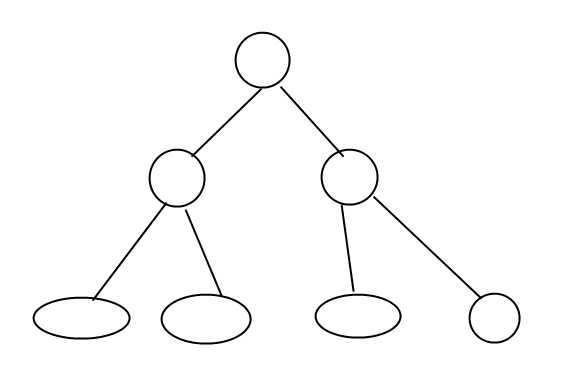
9

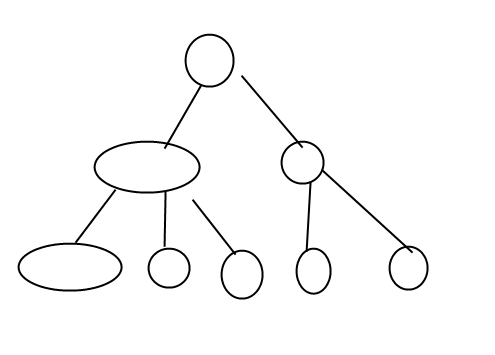






10



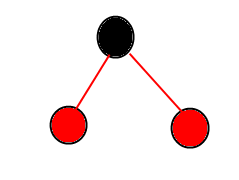


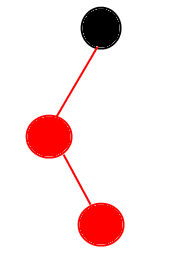
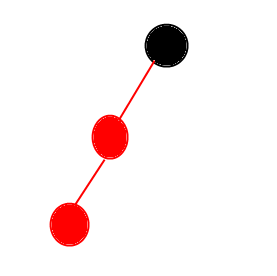
3.3.6 Find the probability that each of the 2-3 trees in Exercise 3.3.5 is the result of the insertion of N random distinct keys into an initially empty tree.

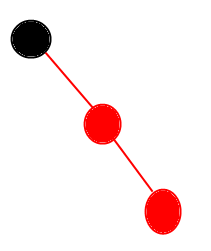
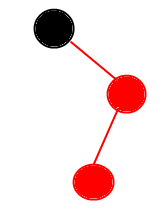
|  |  |
| --- | --- |
| 1 | 1 |
| 2 | 1 |
| 3 | 1 |
| 4 | 1 |
| 5 | 0.4, 0.6 |
| 6 | 1 |
| 7 | 4/7, 3/7 |
| 8 | 4/7 \* 2/8 = 1/7, 6/7 |
| 9 | 6/7 \* 3/9 = 2/7;  1/7 \* 6/9 + 6/7 \* 2/ 9 = 2 / 7;  1/7 \* 3/9 + 6/7 \* 4/9 = 3/7; |
| 10 | 29/35 , 2/7 \* 6/10 = 6/35 |

3.3.7 Draw diagrams like the one at the top of page 428 for the other five cases in the diagram at the bottom of that page.

3.3.8 Show all possible ways that one might represent a 4-node with three 2-nodes bound together with red links (not necessarily left-leaning).





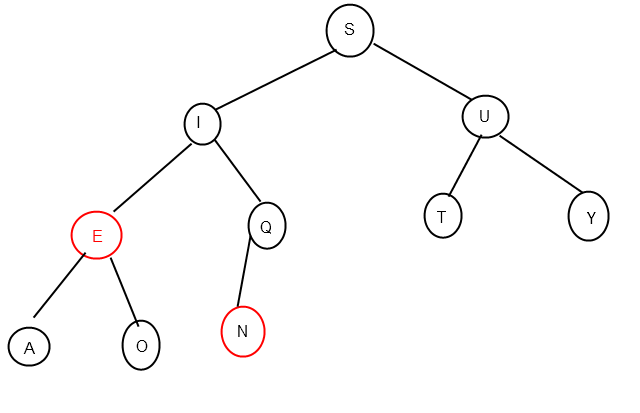


**3.3.9** Which of the following are red-black BSTs?

iii, iv. The black paths length from root to leaves of i and ii are not balanced.

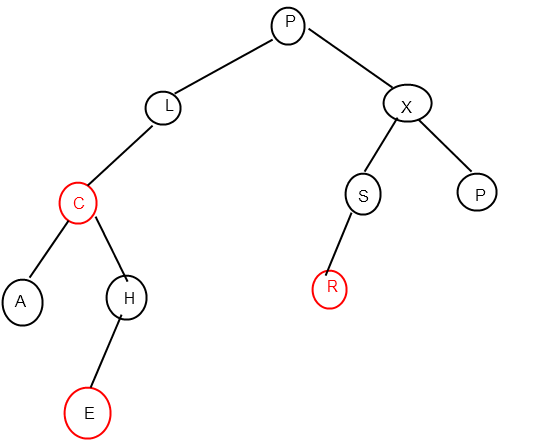
**3.3.10** Draw the red-black BST that results when you insert items with the keys

E A S Y Q U T I O N in that order into an initially empty tree.

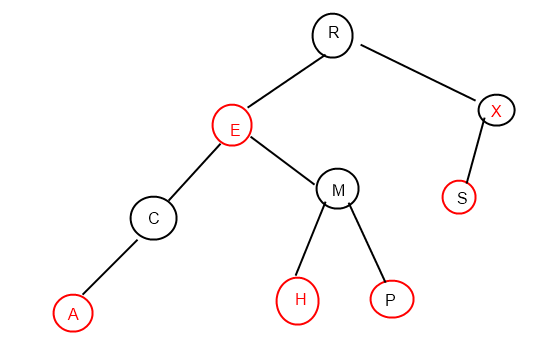


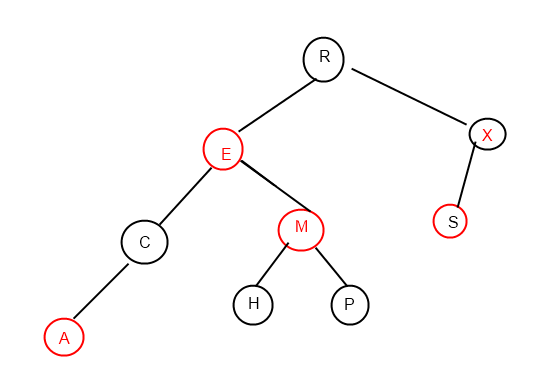
**3.3.11** Draw the red-black BST that results when you insert items with the keys

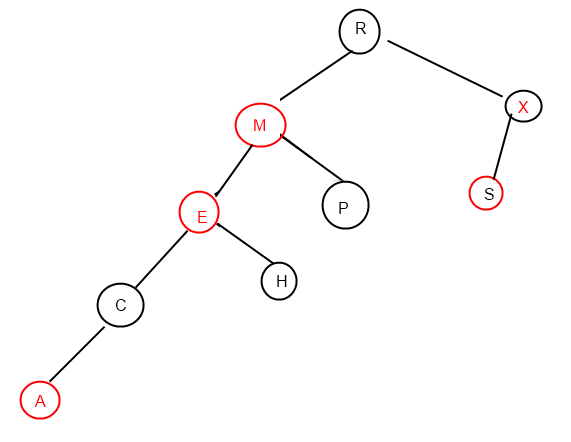
Y L P M X H C R A E S in that order into an initially empty tree.

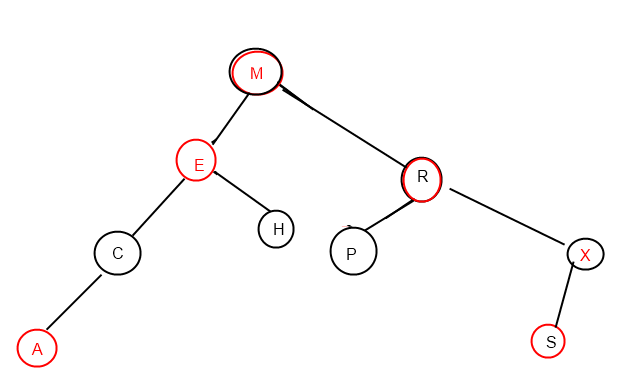


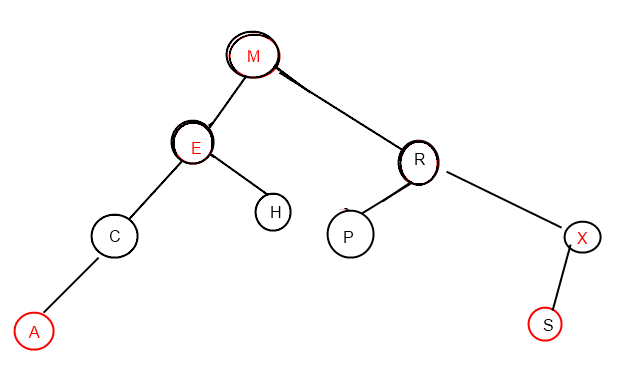
**3.3.12** Draw the red-black BST that results after each transformation (color flip or rotation) during the insertion of P for our standard indexing client.







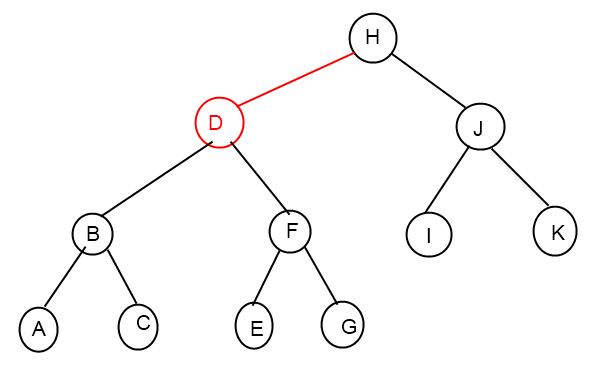




**3.3.13** True or false: If you insert keys in increasing order into a red-black BST, the tree height is monotonically increasing.

True, though not strictly monotonically increasing.

**3.3.14** Draw the red-black BST that results when you insert letters A through K in order into an initially empty tree, then describe what happens in general when trees are built by insertion of keys in ascending order (see also the figure in the text).

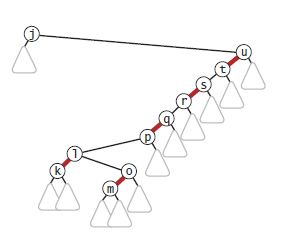
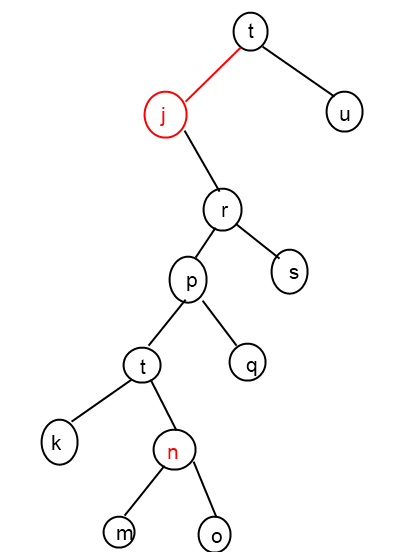


**3.3.15** Answer the previous two questions for the case when the keys are inserted in *descending* order.

No, because the tree could be rotate to right to balance itself. This would decrease the height.

However the final tree is no difference with the one from ascending order.

**3.3.16** Show the result of inserting n into the red-black BST drawn at right (only the search path is shown, and you need to include only these nodes in your answer).

**3.3.17** Generate two random 16-node redblack BSTs. Draw them (either by hand or with a program). Compare them with the (unbalanced) BSTs built with the same keys.

Round: 0

RedBlackTree height:6

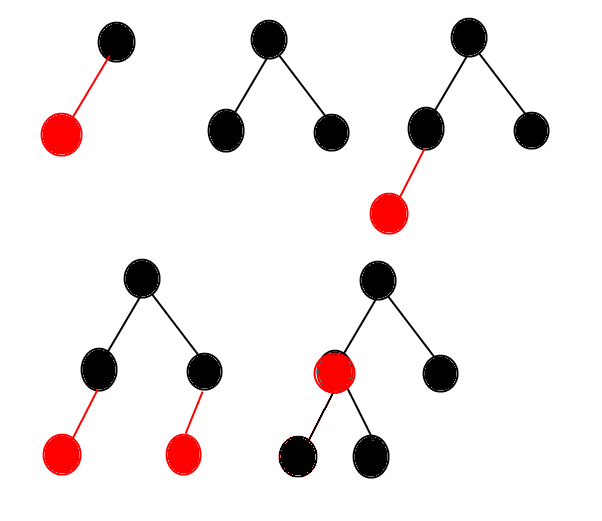
BinarySearchTree height:8

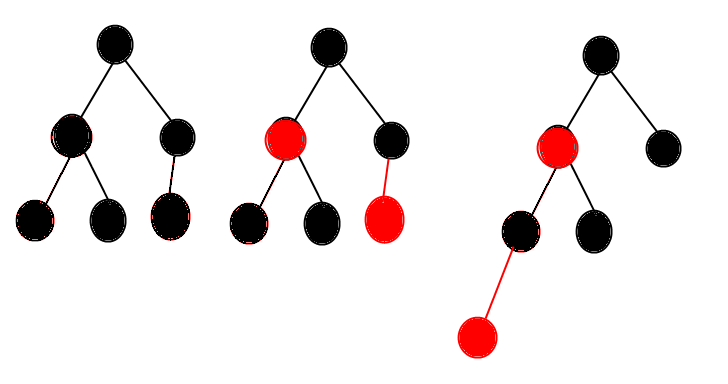
Round: 1

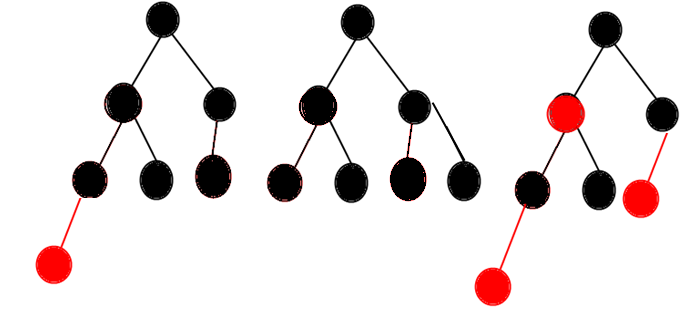
RedBlackTree height:6

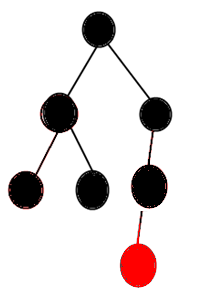
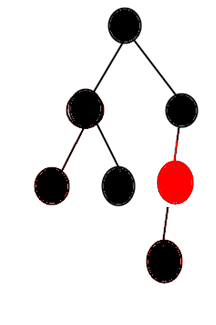
BinarySearchTree height:7

**3.3.18** Draw all the structurally different red-black BSTs with *N* keys, for *N* from 2 up to 10









**3.3.19** With 1 bit per node for color, we can represent 2-, 3-, and 4-nodes. How many bits per node would we need to represent 5-, 6-, 7-, and 8-nodes with a binary tree?

**3.3.20** Compute the internal path length in a perfectly balanced BST of *N* nodes, when *N* is a power of 2 minus 1.

Suppose N = 2m – 1;

Let Sm is the total internal path length of the tree.

Sm =

2 \* Sm **=** =(g = k+1) = = Sm + m\*2m -

* Sm = (m-2)\*2m + 2
* Sm = = O(N)

**3.3.21** Create a test client TestRB.java, based on your solution to

**3.3.22** Find a sequence of keys to insert into a BST and into a red-black BST such that the height of the BST is less than the height of the red-black BST, or prove that no such sequence is possible.

No such sequence, or in other words the height of red-black BST is always smaller or equals to the corresponding standard BST, no matter what the input sequence is.

Because rotate/flip color operations (if necessary) for each insertion from bottom to root will only keep or reduce the height of the tree, never increase it.