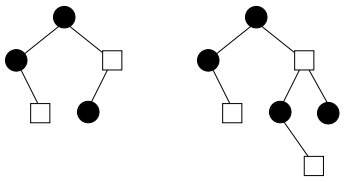
Problem set 8.
Binary Search Trees, Red-Black Trees.

1. (a) Insert the following elements into an empty binary search tree: $7, 13, 1, 9, 10, 14, 12, 2$.
   (b) What is the Preorder, Inorder and Postorder of the elements?
   (c) Delete $2, 9$ and 7 from the tree in that order.

2. Is it possible that the Preorder of the elements of a red-black tree is : $6, 1, 5, 3, 2, 4$?

---

3. Is it possible that the figure belows shows the vertices of a red-black tree. The empty (NIL) leaves have not been drawn and a black circle indicates a black vertex while a white cube indicates a red vertex.



4. We are given a binary search tree containing all distinct integers. Is it possible that during a call to SEARCH(x), we see the key-values $20, 18, 3, 15, 5, 8, 9$ in this order along the search path? If it is not possible, give a reason why not, and if it is possible, then specify all the integers x for which this can happen.

5. In a red-black tree, the path from the root to one certain leaf have the colours in the following order: black, red, black, black. What is the minimum number of items stored in the tree?

6. Let the Inorder sequence of the elements of a binary search tree be $j, b, k, g, i, a, c, d, f, e, h$, and let the Preorder be: $a, b, j, g, k, i, d, c, e, f, h$. Use this information to reconstruct the tree.

7. We are given two binary search trees, each containing n different elements. Give an algorithm with running time $O(n)$ that determines whether the two trees contain the same numbers.

8. Design a data structure that stores natural numbers (with possible repetitions) where the following operations are supported: INSERT(i): store a new instance of i
   DELETE(i): an instance of i is deleted
   CLEAR(i): delete all instances of i
   COUNT(i): return the number of instances of i
   The cost of the data structure should be such that if it contains m *different values*, then each operation should have running time of $O(\log m)$.
   (For example, if the elements stored are $1, 1, 3, 3, 3, 8$, then COUNT(1)=2, m =3.)

---

9. We are given two complete binary search trees, each containing n elements. Give an algorithm with running time $O(\log n)$ that determines whether all elements of the first tree are greater than all elements of the second tree.

10. We are given a binary search tree with n vertices, all the values of which are different. For each vertex v of this tree, we want to determine how many elements in the subtree with root v are smaller than v. Give an algorithm that does this in $O(n)$ steps.

11. In a doctor's office, you have to check in at the registration desk, where the staff decide whether the patient must be assigned to doctor A or to doctor B or if the patient can be assigned to either one of them. Also, after inpecting the referral note, the patient will be assigned a number indicating the urgency of the case. Assume that a smaller number indicates a greater emergency. When a doctor has finished with a patient, he picks a patient with the smallest emergency number out of all

the patients who can/must be treated by him. Assume that the emergency numbers assigned to all patients waiting are different. Describe a data-structure that, in the case where n patients are waiting, allows the registration of a new patient and the selection of the next patient by the doctors in $O(\log n)$ steps.

12. Design a data structure that can be used to store the elements of an ordered set. We want to be able to support the following operations:
BUILD(n): builds an ordered set of n elements.
DELETEMIN, DELETEMAX: deletion of the maximum or the minimum element.
INSERT: insert an element in the set.
We want the above operations to have a running time of $O(n)$ for BUILD(n) and $O(\log n)$ for all other operations.

13. On a rooted leveled tree, A and B play the following game: they take turns moving a piece that is initially on the first level at the root. Then they alternatingly move the piece to a child of the vertex it is at. The game ends when the piece is placed in a leaf of the tree. Some of the leaves are painted green while the others are colorless. We say that the first player A wins if the game ends in a green leaf.

    Given the adjacency matrix of the graph and a vertex attribute stating if a vertex is green or colorless, design an algorithm with running time $O(n)$ that determines how player A should play to ensure that he wins (assuming he has such a winning strategy).