Practice Problems

---

(1) We have studies two similar-looking data structures, the binary-search tree and the heap. What is the difference between the two?

(2) An in-order traversal of a binary search tree can be used to print out the elements of an n-node tree in sorted order in $O(n)$ time. Can the same be done with a heap? Explain why or why not.

(3) Given a binary search tree T, sketch an algorithm to find the maximum and minimum key values stored in T. What is the complexity of your algorithm?

(4) Newark airport is developing a computer simulation of air-traffic control, which handles events such as landings and take-ofs. Each event has a time-stamp that denotes when the event occurs. The simulator needs to perform efficiently the following two operations:
—insert a new event with a given time-stamp (i.e. add a future event);
—extract the event with the smallest time-stamp (i.e. determine the next event to process).
Which data structure would you use to support the above operations? Justify your answer.

(5) Given an arbitrary binary tree T with integer keys stored at the nodes, design an efficient algorithm which determines whether or not T is a binary search tree. What is the complexity of your algorithm?

(6) Draw all possible binary trees with keys 1, 2, 3. How many are there? Actually, you don't need to draw them, just argue how many.

(7) Draw all possible binary search trees with keys 1,2,3. How many are there?

(8) Write a method to compute the size (number of nodes) of a tree given as parameter. What is the running time of your method?

(9) Write a method `SmallerThan(int x)` that outputs, in ascending order of keys, all elements in a BST whose keys are smaller than or equal to x. What is the running time of your method?

(10) Implement an `equals` method on Binary Search Trees. Two BSTs are equal if they contain the same elements at the same positions in the tree.

(11) In the search for the element x in a binary search tree, the list of nodes examined (i.e. the nodes that are compared to x) is called the *node sequence* for the search. Can the following sequence be a node sequence for a search?

$$925, 202, 911, 240, 912, 245, 363$$

(12) Assume you want to extend a binary search tree to store a reference to the smallest element in the tree. Assume we modify the insert and delete operations to update this reference to the smallest element. Describe what happens when you insert a new value and when you delete. How long does it take to find the smallest node after insert/delete? (think worst-case).

(13) Describe (not implement) how you could use a stack to check a program to make sure all of the parenthesis are balanced. Parenthesis are balanced when there are as many (s as)s and they are properly nested, etc — just think of a math formula.

(14) Describe the best way to implement two stacks using a single Vector as storage. Your solution should allow for as many elements as possible to be stored. Assume that you know the maximum number of elements inserted in both stacks in total (but not in each stack).

(15) BT&T has 256 million customers. Their current telephone directory consists of many heavy volumes typeset in small point size, which are expensive to print and inconvenient to use. To overcome the above problems, BT&T has decided to set up an online computerized directory, and their software engineers are debating what is the most efficient data structure for the purpose. Assume that the BT&T computer can compare two names in one microsecond.
—One of the engineers, who did not take any course beyond 101, suggests to implement the on-line directory as an unsorted linked list. With this implementation, give an estimate of the worst-case search and insertion times.
—Another engineer, who took 210, wants to implement the online directory as an AVL tree. With this implementation, give an estimate of the worst-case search and insertion times.
—A third engineer proposes the use of a sorted array. What kind of argument can she make tot support her proposal?

(16) Describe how to compute the connected components on an undirected graph $G = (V, E)$ and discuss the complexity of your algorithm.

(17) Assme we have a directed graph G. Describe an algorithm to find out whether G has a cycle. Discuss the complexity of your algorithm.

(18) Assme we have an undirected graph G. Describe an algorithm that determines whether two vertices u and v are linked by a path. Discuss the complexity of your algorithm.

(19) Same problem as above, but on a directed graph.

(20) Maps and hashing: see exercises in leture notes.

(21)