Data Structures and Algorithms : CS210, CS210A

Practice-sheet 2

Date: *20 August, 2014*

1. Given a positive integer n and a list containing $n-1$ distinct integers in the range $[1, n]$, design an $O(n)$ time algorithm to find the missing number. You are not allowed to modify the list even temporarily. Your algorithm is allowed to use only $O(1)$ extra space. However, you may assume that every arithmetic operation takes $O(1)$ time.

2. Write an iterative as well as recursive function in C for reversing a singly linked list.

3. Given two singly linked lists storing n elements in ascending order. Transform these lists into one list which also contains elements in ascending order. The time complexity of the algorithm has to be O(n).

4. You are given the head of a singly linked list. You need to determine if this list loops (a node whose next pointer points to some node appearing earlier in the list) somewhere or not. Design the most efficient algorithm for this problem. Time complexity of the algorithm has to be O(k) where k denotes the length of the list if there is no loop, otherwise k denotes the length of the longest loopless prefix of the list.

5. Circular linked list is a singly linked list where the next field of the last node points to (stores the address of) the first node of the list. What are the advantages/disadvantages of a circular linked list over a singly linked list and a doubly linked list ?

6. **Operations on Binary Search Tree**

   Given a binary search tree T specified by the address (pointer) of its root node, design an algorithm

   (a) to compute its height.

   (b) to enumerate all the values stored in the tree in the decreasing order.

   (c) to transform this tree into another binary tree which is the **mirror image** of the original tree T. Note that the original tree loses its identity after this operation.

7. **Structure of Binary Tree**

   - What can be the maximum number of nodes in a binary tree of height h ?
   - Depth of a node v in a binary tree T is the number of edges on the path from root to node v. What can be the maximum number of nodes at depth d in a binary tree ?
   - What can be minimum height of a binary tree having n nodes ?

8. **Generating permutations using stack** A stack can be used to generate some (not necessarily all) permutations of first n numbers. For example, for $n = 3$, we can generate $(1,2,3),(1,3,2),(2,1,3),(2,3,1),(3,2,1)$ using stacks but we can not generate permutation $(3,1,2)$ using stack.

   - Design and implement an algorithm which prints all permutations of first n positive integers which can be generated by a stack.
   - (optional) Can you give a closed form expression for the number of permutations generated using a stack ?

9. **Stack with maxima**

   You need to maintain a stack whose elements will be positive numbers. There will be an additional operation called *Report-Maximum.* This operation is supposed to return the maximum element among all those elements present in the stack. You need to provide an implementation that will achieve $O(1)$ time for each operation (including *Report-Maxima* on the stack. At every moment of time, your data structure must occupy $O(n)$ space, where n is the number of elements present in the stack at that time.

10. **Think fresh ...** Company ElGoog has come to IITK for recruiting bright students. There are n students with roll number from 1 to n appearing for interview. Each student has information: (*name, roll no., section,...*) associated with him/her. The interview board calls students in any arbitrary order. The information (record) of each student is destroyed just after the interview. Due to the reason best known to the board, while interviewing a student, with roll number, say i, they want to know the information about the student with maximum roll number less than i who has not been interviewed yet. We call such student as **pred**(i) (a shorthand for predecessor). Likewise we define *succ*(i) (a shorthand for successor). So design a data structure storing information of n students so that each of the following operations can be performed in worst case O(1) time. (No assumption on the sequence of operations is permitted).

    - **Access**(i): return information about a student with roll number i.
    - **Delete**(i): delete the record of a student with roll number i.
    - **Pred**(i): return the roll no. of the predecessor of a student with roll number i.
    - **Succ**(i): return the successor of an existing student with roll number i.

For improving your programming skills ...

The power of Recursion

Recursion is a very powerful and effective way to solve various computational problems. Usually the length of the code of a recursive algorithm is very short. However, one has to be very careful in designing the recursive algorithm. In particular, one must first formulate the solution of the problem recursively in terms of smaller instances of the same problem and then design a recursive algorithm based on it. One must pay a lot of attention while handling the base case. You were taught recursion in ESC101 course. The aim of this exercise is to refresh your knowledge of recursion using some old/new problems. Write pseudo-code for a recursive algorithm for the following problems. You are also advised to implement these pseudo-codes in your favorite programming language.

1. Given an array A storing n distinct elements, and an integer r, generate all possible permutations formed by picking r distinct elements from A.
2. Given an array A storing n distinct elements, and an integer r, generate all possible combinations formed by picking r elements from A.
3. Given an array A storing n elements (duplicates allowed), and an integer r, generate all possible combinations formed by picking r elements from A.