CSE 2012- Design and Analysis of Algorithms Practice Problem Sheet (Backtracking Algorithms) N-Queen's Problem

Practice makes you Perfect

N-Queen's Problem :

1. A positive integer n can be partitioned into a sequence of distinct positive integers $a_1, a_2, ..., a_n$ such that $n = a_1+a_2+...+a_k$. We call $a_1+a_2+...+a_k$ as $k-$size distinct partition of n. Here, none of the positive integers a_i,$i = 1, 2..., n$ are same. Distinct partitions of 3 are $2+1$ and 3. We consider the partition $2+1$ and the partition $1+2$, as the same partition. Maximum-size distinct parttion of 3 is $2+1$. Similarly, distinct partitions of 6 are $1+5, 2+4, 3+2+1, 6$. Maximum-size distinct partition of 6 is $3+2+1$. Given a positive integer n, design a back-tracking algorithm to compute a maximum-size distinct partition of n. Analyse the algorithm with the running-time and the time-complexity.

2. Consider the Sudoku Problem : You are given a partially filled 9×9 grid. The task is to fill the 9×9 grid with digits (from 1 to 9) so that each column, each row, and each of the nine 3×3 subgrids contain all of the digits from 1 to 9. Design a backtracking algorithm to solve the Sudoku problem. Analyse the algorithm with the running-time and the time-complexity. Analyse the algorithm with the running-time and the time-complexity.

3. Graph coloring problem involves the assignment of colors to each vertex of a graph G such that no adjacent vertices get same color. The task is to colour the graph G with minimum number of colours. Design a back-tracking algorithm to solve the graph colouring Problem. Analyse the algorithm with the running-time and the time-complexity.

4. Consider the 'String Permutation Problem' wehre the input is a string of symbols S and the output is all the permutations of the symbols of the string S, in an alphabetical order. If S is abc, then tha algorithm should output $abc, acb, bac, bca, cab, cba$. Design a brute-force algortihm and a back-tracking algorithm for the String Permutation Problem. Compare

both the algortihms in sense of time-complexity. In case, both the algorithms are of the same time-complexity, decide which algorithm is more efficient with appropriate justification.

5. Given a non-empty set S of non-negative integers and a positive integer k, problem is to find a subset S' of S such that sum of the integers of S' is equal to k. Design a back-tracking algorithm to solve the above Problem. Analyse the algorithm with the running-time and the time-complexity.