Data Structures and Algorithms

Problem Set #1

Topics : Recursion

Problems

1. **String Reversal** : Reverse a string using recursion.

   - *Input* = "abcde", *Output* = "edcba"

   Palindromic String : A string is said to be a palindrome if it is same when read from both the sides. How would you use the the above problem to check whether a string is palindrome or not?

2. **Fibonacci Numbers** : Fibonacci numbers are defined by the recursive relation $F(n) = F(n-1)+F(n-2)$, where $F_0 = 0$, $F_1 = 1$. Write a program to calculate the $n-th$ fibonacci number. Draw the recursion tree and analyze the execution manually as taught in the class. (For drawing the tree, you can take $n = 5$)

3. **0 - 1 Knapsack** : Suppose you decide to rob a house. You have only one bag with capacity *maxCapacity*. There are n items available in the house. Each item has a fixed value and a fixed cost. Your goal is to maximize the total value that you can steal from the house. (Of course, you cannot exceed the bag capacity). You cannot take a fraction of any element. Either pick it or leave it (0 - 1 Property). Give an algorithm to find the maximum profit possible. Write its pseudocode. [The input consists of 2 arrays, *Values* and *Weight* of length n and an integer *maxCapactiy*.]

   - $Values$ = 10, 20, 30, $Weights$ = 60, 100, 120, *maxCapactiy* = 50
   - The maximum stolen value would be 220 (100 + 120)

4. **Coin Change** : Suppose you have an array that represents the denominations of the coins that you have. Assume that the shopkeeper has unlimited coins of each denomination. You need to find out the number of ways in which you can make change for a given amount. Write an algorithm which returns the number of ways.

   - *Denomination* = 1,2,3, *Amount* = 4. The number of ways is 4. They are (1,1,1,1), (1,1,2), (2,2), (1,3). Note that the order of coins doesn't matter.
   - *Denomination* = 2,5,3,6, *Amount* = 10. The number of ways is 5. They are (2,2,2,2,2), (2,2,3,3), (2,2,6), (2,3,5), (5,5).
   - Try to draw the recursion tree for the first example.

5. **Longest Common Subsequence** : For a string, a subsequence is defined as the string obtained by deleting some characters in the string (possibly zero or all) and maintaining the relative order of the other characters. Notice that the number of subsequences of the string would be 2^n, where n is the length of the string. Given 2 strings as an input, you need to find the longest common subsequence of both the strings, i.e a subsequence which can be formed by both the strings and is of maximal length. Note that you just need to print the length.

    - *First String* = "ABCDGH", *Second String* = "AEDFHR". The answer is of length 3 ["ADH"]
    - *First String* = "AGGTAB", *Second String* = "GXTXAYB". The answer is of length 3 ["GTAB"]

6. **Edit Distance** : Edit distance between 2 strings is defined to be the minimum number of moves required to convert the first string into the second one. There are 3 possible moves (which you can apply only on the first string). You can either insert a character at any position, or you can delete a character from any position, or you can replace a character at any position. All the three of them would cost you exactly one move. Your task is to find the minimum moves required to do the conversion. Write the pseudocode. Note that the strings can be of unequal length.

    - *First String* = "sunday", *Second string* = "saturday". The minimum edit distance is 3. This can be done by replacing n with r, inserting t and finally inserting a.