ESO207A: Data Structure & Algorithm

Practice Sheet on Recursion

July 31, 2022

Golden Points

1. It is advisable to try the problems on your own. Refrain from any shortcut means. ☺
2. The course ESO207A requires the basic understanding of recursion and how a recursive program is executed. The aim of the following problems is to make you realize the power of recursion. Make sincere attempts to solve these problem during the semester.
3. Do not worry, if you are not able to solve any problem at first attempt. Giving sufficient time to the problem and making attempts improves one's understanding a lot and is itself a big positive step towards improving the analytical skills. ☺

Practice Questions

1) **Strings of alphabets**

Give an algorithm that generate all strings of length n made of only first k English alphabets.

2) **Special binary strings**

Give a recursive algorithm that generates all strings of length n consisting of 0s and 1s without any two consecutive 1s.

3) **Patterns of special symbols**

Give a recursive algorithm that computes all strings (patterns) formed by n *'s and m @'s .

4) **Paths in a grid**

There is a $n \times n$ grid. You start from bottom left corner and you have to go to the top corner. At each stage you may move either one step up or one step right. Give an algorithm to enumerate all paths to reach top right corner.

Hint: Think in what way this problem is similar to the last problem.

5) **Balanced Parenthesis**

We know that an expression involving parenthesis is valid if for each right parenthesis there is a unique left parenthesis to match. Generate all valid expressions consisting of n left parenthesis and n right parenthesis. For example, for $n = 3$, there are five valid expressions:

{}{}{}
{{}}{}
{}{{}}
{{{}}}
{{}{}}

*Hint: Observe that you need a permutation of n {'s and n }'s such that for each prefix, the number of left parenthesis should be at least equal to the number of right parenthesis. Make use of this observation and suitably modify the problem on the patterns with n *'s and m @'s to achieve the solution.*

6) **Partitions of a positive integer**

a) Given a positive integer n, give a recursive approach to compute all permutations of positive integers such that their sum is n and the value of numbers in permutation is non-decreasing.

   Example: The non-decreasing permutations of integers summing upto 4 are:

   [1+1+1+1]
   [1+1+2]
   [1+3]
   [2+2]
   [4]

b) Solve the above problem but with the change that the value of the numbers in the permutations are strictly increasing.

   Example: For $n = 4$:

   [1+3]
   [4]

c) For a positive number n, give an algorithm to generate all permutations of 1s and 2s which sum upto n.

d) Given a positive integer n, print all those sets of positive numbers whose sum is n.

7) **Permutations of a string with repeated characters**

   Given a string of length n with possibly repeated characters, print all permutations of length L. For example, if string is abac, and $L = 2$, then the set to be enumerated has following strings:

   aa
   ab
   ac
   ba
   bc
   ca
   cb