CS110: Computer Programming Lab
Department of CSE
IIT, Guwahati                    Jan-May 2018

# RECURSIVE FUNCTIONS: A COMMON STRATEGY

Recursive functions are very useful strategy for writing programs. Many problems that are best suited for this kind of programming are problems that have a "repeating" nature. That is, the solution producing function that is suitable for a bigger problem is also suitable function for solving the smaller sub-problem.

For example, to sort the names of students in a class, we may choose a letter (alphabet) and put all names falling on or before the chosen letter in one sub-list. And, all those after it in another sub-list. Now these lists can be sorted using exactly the same trick! This is called recursion. Once two lists are sorted, the combined list is simply one list (already sorted list of larger letters) added at the end of the other sorted list.

The problems here in this column follow the similar pattern.

We have a problem of letter-digit puzzle. Letters in a few words are to be assigned digits. One slow way is to try all combinations of the pairing of letter and digits. A more efficient and faster way is to make a pair; determine as soon as possible if the pairing causes any conflict. If it does discard the choice and pick an alternate pairing. If a pairing is ok, then choose values for other (remaining unpaired letters) recursively. Often the recursive attempt will report that the pairing of remaining letters is not possible. If the recursive step reports trouble, we must discard the choice we made and repeat the recursive step by picking yet another pairing for the current letter we had.

The problem of Knight tour also is based on the similar recursive strategy. There are 8 possible next moves for a knight. Some cause conflict. Others do not cause conflict immediately but may have problems later. Once again the recursive program that we write makes a choice of the next move of knight and let the recursive step begin. If the recursion succeeds we are done. If it does not complete the tour, we have a failed along the path we chose. Undo the move and make alternate move to recursive call. If not further choice remains to choose, report failure to the caller. The caller is likely to be a previous recursive call. The call will most likely make a different choice and resume recursive call.

Sudocu problem is similarly recursive. For each unspecified square, we can add a digit. So each square has 10 choices of digit. We try them one by one. See is the choice causes an immediate conflict. If not then we do the same steps for the remaining (unfilled) board. If that does not succeed, we must choose a different digit for this square.

Putting 8 queens on chess board is the same strategy. Try to put one queens in the rows from one edge of the chess board. Find a non-conflicted square in the next row. Place a new queen there and then recursively try the remaining rows. If the reaming rows give no solution, the queen in the current row may be moved to different square and recursive step to place other queens re-attempted.

Counting the number of possible solutions for 8-queens problem is similar strategy with a small variation. For each possible placement of queen in first row recursive step will be used to count the number of solutions they have. Add these numbers to get the total number of solutions. The recursive steps will be performing the similar action. They too will be adding the count of solution counts returned by their (recursive) sub-problems. At the end of these chains is obviously a count of solutions in which all queens are in the board. This end-of-recursion step will return either count of 1 or 0 depending on if it is a non-conflicted or conflicted situation. The calling recursive step will add these numbers and will pass that sum as result to its caller. The original caller to this recursive function will get sum of all counts. This is the answer a problem in the assessment exercises set seeks.

# Error Reporting and Suggestions for Improvements

My sincere apologies if the document has errors or mistakes. Please report errors in this document to vmm@iitg.ernet.in. Also, I welcome suggestions and advice to improve the quality of the document for the students of CS110.