# Design and Analysis of Algorithms

Motivation:

Demonstrate the relationship between input size and time. In particular show that an exponential time program is too slow even for very small input sizes.

Practice writing recursive code.

Prepare for interview questions. Code for generating all sequences is a commonly asked interview question

**Basic steps of recursive programs**

Every recursive program follows the same basic sequence of steps:

Initialize the algorithm. Recursive programs often need a seed value to start with. This is accomplished either by using a parameter passed to the function or by providing a gateway function that is non-recursive but that sets up the seed values for the recursive calculation.

Check to see whether the current value(s) being processed match the base case. If so, process and return the value.

Redefine the answer in terms of smaller or simpler sub-problem or sub-problems.

Combine the results of the sub-problems in the formulation of the answer.

Return the results.

Write a recursive function to generate all the sequences of length *n* of the numbers 1 to *n.* [100%]

Your program will take a command line arguments n. You need to write code for a program that creates all possible sequences of the numbers 1, 2,…, n. For example if n=3 your program should write:

111, 112, 113, 121, 122, 123, 131, 132, 133,

211, 212, 213, 221, 222, 223, 231, 232, 233,

311, 312, 313, 321, 322, 323, 331, 332, 333”

out1.txt: Run your program for n=3. The output should be the three lines above.

out2.txt: Run your code for 4<=n<=11:

Line 1: “The Run time for generating all sequences of size” 4 “is” <y4> “seconds”

…

Line 8: “The Run time for generating all sequences of size” 11 “is” <y11> “seconds”

Grade:

Program compiles but does not run 45%

Program runs but output is incorrect 45%-80%

Program is correct and well written 100%

Extra credit [10%]

Same requirements but now generate all permutations. For n =3 you should print now in some order the following permutations.

123

132

213

231

312

321

**Hint**: The main difficulty in writing code for generating all permutations is to make sure that if you already generated part of a permutation with numbers *p*1≠ *p*2≠ …≠ *pi* where *pj* for *j* = 1, …, *i* satisfies 1*≤* *pj≤ n*, that your next choice for 1*≤ pi*+1*≤ n* will be different from *p*1≠ *p*2≠ …≠ *pi*.

You may write the code using C, C++, or Java. The programs must compile on Bingsuns.