# Sqrt(x) I/II/Variant

变种:求上界，和求下界的唯一区别，将赋值结果放到right区间里，判断条件等一律不变；同时为了规避int越界，用long来定义middle/left/right等元素；

II注意点： 对于精度处理在while循环里，不需要再改变索引的时候使用；

Double数据类型处理需要注意，如果x小于1，则索引end应该取1；

Double处理使用牛顿法？

# LIS各种变种

相邻递增 =》 不相邻递增1 =》 不相邻递增

=》 二维矩阵相邻递增

# Word break I/II/variant

I: DP求解1可能性，dp中存放1/0表示是否可达，dp头结点初始化为1；

II: DP求解3可行解的总数，dp中存放可达的可行解总数，dp头结点初始化为1；

III: DFS求解所有可行解，使用哈希表缓存每个索引点对应的可能匹配结果；如果遍历到一个索引节点，发现有缓存结果，代表之前已经在此路径上找到过可行解了，不在DFS，直接利用哈希表的缓存输出到最终结果集

Variant: 求可行解中最大集的个数，dp中存放可行解的最大个数，dp头结点初始化为0；在当前索引节点找到对应匹配时，比较当前索引值的dp值+1 vs 目标索引值的dp值，取小的那个；

注意处理非头结点时，如果该节点不可达直接跳过不处理；

# Spiral matrix I/II

注意n\*n的矩阵每遍历一次（横纵横纵），剩下的还是一个标准矩阵，或者一个点；横纵横纵中的每一次扫描结束后不需要判断横纵是否越界；

但是如果横纵不相等n\*m，则可能出现最后是一个横，或者是一个列；在遍历的过程中，横纵中的每一次扫描结束后都需要判断当前起始和终止位置是否已经越界了；

# Number of Islands I/II/Variant

I:矩阵分布0/1，求全1的连通块个数。每个1各自代表一个连通块，统计1的总数即连通块总数，每一次union后，连通块个数减少1个；

II:矩阵默认为空，随机放1进去，统计每次连通块个数。定义标准并查集，每放入一个1，统计1的总数，同时检查四周邻居，可以union则union并同时记录union的个数。同I的解法一样，1的总数减去union的次数就是岛屿数量；

Variant: 矩阵分布1-n，同一数字连接为一个连通块，求最大连通块size。并查集中引入size数组记录每个连通块的大小，union的同时记录当前记录到的最大size

# LCS I/II/III

I: Node A and B exist in the tree.

II: Node A and B exist in tree and each tree node maintains a pointer points to its parent.

III: Node A and B may not exist in the tree.

Variant: Print path between A and B

# Add Binary：

注意字符串处理，StringBuilder的使用；

主要是代码的书写是否简洁