DFS-II (class 17)

1. Summary

1.1 DFS Principle

- How many levels in the recursion tree? What does it store on each level?
- How many different states should we try to put on each level? (how many different cases under one case?)

1.2 用途：all possible

枚举所有可能性。答案可以用排列 or 组合表达，就用DFS。

注意虽然是枚举所有可能性，但是不一定要求"输出所有可能性"

2. All types

2.1 Subset

Subset

Name	I/O	问题关键	关键解法	Done!	Column
All Subsets II	输入set，输出list of subsets	all possible subsets； 有重复字母，可能会产生ab1 和 ab2 这种重复	1. 每层 —> 判断一个字母是否加入； 2. 防止重复字母导致重复，需要先对set排序，一旦不用某个字母就要先跳过所有重复，直到结尾或者下个字母，才进入下层循环。 （不是先进入下层循环，出来后才跳过） base case是 idx越界了	☑	
All Subsets of Size K	输入set和k，输出list of subsets长度都恰好为k	恰好长度k的subsets 无重复	直接k层，每层选没用过的（传递下次用的index） 需要sort，所以要变成char array 注意Arrays.sort() 的时间复杂度（分类讨论） base case 是stringBuilder的大小 == k了。	☑	
All Subsets II of Size K	输入set和k，输出list of subsets长度都恰好为k	恰好长度k的subsets 有重复	当于缝合了。 注意要sort，所以要变成char array base case有两个： 1. 长度恰好为k了 - 加入results 2. idx越界了 - 单纯return(放在后面，因为长度为k的时候如果包括最后一个字母，那么其实当前idx也是越界的，如果放前面会漏掉一个result）	☑	
Two Subsets With Min Difference	输入set，求产生size/2的两个subset时，最小的diff	1. sum diff最小 2. size对半（奇数可以相差一个） 不需要返回所有结果，只要最小的diff（依然要枚举所有可能性）	1. 层数 = array.length / 2，每层枚举所有可能性 2. 最小的diff可以表达为：total - 2*currentSum 其中currentSum是一种选择下的sum。更新globalMin就行。 base case只需要处理 levelLeft == 0. 对于那些走到头但是levelLeft不为0的，进不去for循环，会自动走到结尾的"return"而返回上一层，自动作废。	☑	
https://leetcode.com/problems/partition-equal-subset-sum/	输入set，问是否能产生两个sum相同的subset	对size无要求 返回boolean	1. 可以动态规划，这里先考虑用DFS暴力破解 2. 没有size限制，用All Subsets I 并check是否让subsum == 0 （subsum从half开始，记录还剩多少） 3. base case：让subsum == 0 或者index用完了 sum(set A) = sum(set B) and C = A union B —> **sum(set A) = sum(set C) / 2**	☐	
	Untitled			☐	

2.2 Parenthesis （force an order）

Parenthesis

Name	I/O	问题关键	关键解法	Column
All Valid Permutations of Parenthesis II	三种括号的数量，产出匹配的所有情况	三种括号的匹配	1. 加入右括号的条件是栈顶元素的左括号和我匹配。不然会出现 [ )( ] 2. 加入左括号的条件是还有左括号可以用。 3. 不需要用三个stack，一个stack就够了。 stack是用来保证匹配的，不是记录什么元素被添加了： **使用方法 — 左括号入栈，右括号看到match的top后会把它弹出。（然后递归调用），随后恢复现场。 栈里面不需要是具体的符号，可以是PS的index**	☑
All Valid Permutations of Parenthesis III	还要求有优先级	怎么定义优先级？怎么产生结果？平级不能嵌套，即不能有(())只能有 () () 或者 <()>	考虑优先级，可以设计一个列表： 1. 保证右括号和更宽泛的括号的index比某左括号大 2. 方便找左右括号是否匹配 ( ) [ ] { } 0 1 2 3 4 5 左括号 i % 2 == 0 右括号的左括号是 i - 1 想加入左括号还需要保证：栈顶要么是空的、**要么是比我更宽泛的括号（即所有右括号和更宽泛的左括号）** 这是和II唯一的区别，为了达到这个效果，**需要把idx加入stack。** 为了保证不同类型的括号匹配，需要引入一个stack来控制什么能选中，什么不能选中。 想加入右括号，栈顶元素必须是和我配对的左括号。【stack.peek() == parentheses[i - 1] 会出NPE错，因为peek可能是空，然后 == 的时候 auto unboxing。 应该先判断是否为空，非空才行。】 【怎么产生结果？ 如何操作stack和stringBuilder】 一个stack，一个stringBuilder。 stringBuilder存储每次操作，stack只存储当前未配对的内容。 举例来说，如果找到了一个右括号，就应该把对应左括号poll了。 让栈顶变成之前压入的左括号。（否则不会有嵌套的出现） —> **而且记得恢复原状，这次是“吐了要吃”** 每层：尝试尝试一遍目前尚能获取的所有括号类型。 层数：最多是 l+m+n层	☑
Factor Combinations	输入n，输出List of List of Array。里面每个小List是一组合法的factors 最小factor从2开始	避免重复？比如6 = 3x2 = 2x3	指定一个order，我尝试过3以后，不允许下一层迭代尝试2 （即只能从我尝试的这个值开始） 这样factor也会从大到小排列。 另外，for循环应该尝试 minFactor 到 remain的所有值。 然后在base case里面把只有remain自己一个的答案排除掉。如果只尝试到 remain / 2 , 那么对于input 4，会漏掉 2x2这个解。 第一层和后续层的定义不一样，第一层中：for first level, i should < n, because input self can't be in result 后续层中：for lower levels, i should <= remain https://docs.google.com/document/d/1WVFps5vWCS3dkrQ6Ikt79xRizCjRZKtPcXvBRgfBiOY/edit 这个方案可以加速，事先把所有factor找出来，然后后续只在factor里面找解。	☑
	Untitled			☐

2.3 Coins

-

2.4 String Permutation / Combination

String

Name	I/O	问题关键	关键解法	Done

Name	I/O	问题关键	关键解法	Done
All permutations of a string and all its subsets(without duplicate letters)	input: string without duplicate char output: List of strings，all permutations	abc —> a,b,c, ab, ac, ba, bc, ca, cb, abc, acb, bac, bca, cab, cba. 如何产生这些中间结果？	递归树每层：代表尝试一个position 每层的元素：当前idx及之后的元素中全部通过swap放一次当前位置。（一定从自己位置开始swap(i,i)，因为abc的permutation有一种是 abc，即不换位置）【这是为了避免重复，如果不是从当前位置开始，而是所有位置逗弄，那么就会有重复的swap出现】 - 记住 全排列permutation是 n! **难点：API使用上。 提供的string输入，我们为了swap方便变成了char[] 怎么获得每次的中间结果呢？ 我们会记录当前try的是第几个position（idx） 用 new String(array, 0, idx) 就可以获得一个这一段的string。**	☑
Keep Distance For Identical Elements	输入 two copies of all integers from 1 to n. 输出 判断能否让两个i之间恰好有i个数字。	怎么放？ DFS每层什么意思？	DFS有n层，每层尝试所有放置 number为 i 的pair的方法。 第1层放 n 第2层放 n-1 .... 第n层放 1 难点：怎么保障正确答案的传输？ 只有一个pad的话，后续的操作会覆盖前面的，而且可能中间就出现正解了，却无法反馈。 解答：可以给DFS一个返回值：true or false **需要重新做一次。 感觉挺难的 1. 一共k层 2. 每层的每个node返回当前subtree有没有结果 - 我尝试所有位置，并让每个位置告诉我这个位置能否成立。如果都不成立返回false，有一个成立立刻返回true。 - 由于立刻返回，所以中断了覆盖。 可以保证pad是触底获得true的那个情况下的。 - 由于helper返回true or false，所以可以在true的时候返回pad，false的时候返回null**	☑
Restore IP Address	输入，一串数字 输出：所有可能的ip地址组合	理解ip地址的合法性，好安排 . 的位置	直接看leetcode解析： 1. 一共三层，每层放一个点。 base case是结尾seg 2. 每层放点的时候，要保证它位置前面的seg是合法的。 base case是在三个点都放了以后，检查剩余部分是否合法（小于等于255） 记录结果：用一个seg数组，记录当前合法segs，return的时候修改最后一个seg。 // 递归树： 一共有4层，每层添加1个、2个、3个数字。 // 答案方法：把for循环展开，添加1个、2个、3个数字的方法是拆开来的，这样可以避免一个特别复杂的getValidSegment 以后遇到了“尝试1个、2个...k个字符”的时候，可以试试不用for循环而是直接展开写k个branch。 规则： **递归node规则：** 1个字符 - 添加即可，然后递归，删掉这个 2个字符 - 开头不能为'0' 3个字符 - 开头不能为'0', value不超过255 **base case规则：** 到第四层的时候，如果发现stringBuilder中的字符数量是都用上了的数量，则知道是base case，把第四个seg后多余的那个点点删掉后就可以加入results。	☑
Combinations For Telephone Pad I	输入：九宫格手机键盘和一个数列 输出：这个数列对应的所有可能性	很常规	1. 多少层？ 多少个数字多少层 2. 每层多少个state：取决于这个数字对应的字母有几个。 发现走到了结尾，就加入结果然后离开	☐
Untitled				☐

2.5 Generate Random Maze

- Generate Random Maze
  - IO：输入n，输出一个 n x n的迷宫 （n是奇数大于等于1）
  - 要求：corridor和wall的宽度都是1cell，每一对corridor之间只有一条路径连接
    - wall是1，corridor是0

> **Examples**
>
> N = 5, one possible maze generated is
>
> 0 0 0 1 0
>
> 1 1 0 1 0

> 0 1 0 0 0
>
> 0 1 1 1 0
>
> 0 0 0 0 0

- 翻译一下：
  - 要求是不能让两个0之间不可达，也不能用出现多条可达路径。
  - 这个要求可以通过”0不成环“这个条件来达成。
- Laioffer答案解法：
  - 准备工作：用enum做了一个叫Dir的枚举类型，记录了上下左右对应的操作，返回移动后结果
  - 先初始化，除了左上角(0,0)是0，其他都设置为1（wall）然后DFS（generate）
    - DFS一共有多少层？直到走完所有的可能性
    - DFS每层几个state：上下左右四个状态中可以走到合法的wall的状态，注意顺序要打乱，不能总是”上下左右“
  - generate函数：
    - 调用shuffle函数，产生一组随机direction序列（上下左右的一个排列）
    - 为了防止把墙打穿，需要看某个方向上走两步是否会还是wall。
      - 走两步走到边界外面：跳过看下一种state（方向）
      - 走两步会站在valid wall上，意味着可以走两步把这个方向上两个wall都变成corridor。
      - 走两步站在了corridor上，这意味着中间隔着一堵墙，不能把墙打穿，那样会导致环路。
    - **上述是如何避免产生环路的？**
      - **因为走两步都是wall，其实第三步可能是corridor，还是会造成联通的啊？？？**
      - **看了一下，只要每次都是走两步，就不会出现第三步可能是corridor，因为这样必然意味着我们之间的间隔是偶数（2），而每次走两步只能产生奇数个间隔（1，3，xxx）。**
- 看看怎么产生随机路径的：
  - Dir.values()获得一个长度为4的四个可能方向
  - 通过random获得一个第i位置后面的某个位置j，让第i个位置的dir和j互换，以产生一个（上下左右）的随机排列：
  - 查看一个当前随机步骤的结果（上下左右之一），如果去的位置是合法的一个wall，就可以把它标记为corridor，然后在那个位置上递归调用。
  - 回来当前位置，探索下一个方向（上下左右之一）

456是总phone interview的题 没有link：
- 第4题是问 问target的interval和given的一组interval有没有overlap
- 第5题是给了个2d matrix，0代表能走，1代表障碍物，问shortest path from (0, 0) to (row, col)，follow up: 如果你有k条命，你可以走障碍物上，但是会少一条命，问新的known info的最短路径
- 第6题类似于class schedule，只不过又加了一个priority，相同等选的课要先选priority高的