40、猜密码，考点 or 实现——深度优先搜索 DFS

题目描述

小杨申请了一个保密柜，但是他忘记了密码。只记得密码都是数字，而且所有数字都是不重复的。

请你根据他记住的数字范围和密码的最小数字数量，帮他算下有哪些可能的组合，**规则如下**：

1. 输出的组合都是从可选的数字范围中选取的，且不能重复；
2. 输出的密码数字要按照从小到大的顺序排列，密码组合需要按照字母顺序，从小到大的顺序排序。
3. 输出的每一个组合的数字的数量要大于等于密码最小数字数量；
4. 如果可能的组合为空，则返回“None”

输入描述

输入的第一行是可能的密码数字列表，数字间以半角逗号分隔
输入的第二行是密码最小数字数量

输出描述

可能的密码组合，每种组合显示成一行，每个组合内部的数字以半角逗号分隔，从小到大的顺序排列。

输出的组合间需要按照字典序排序。
比如：2,3,4放到2,4的前面

备注

字典序是指按照单词出现在字典的顺序进行排序的方法，比如：

- a排在b前
- a排在ab前
- ab排在ac前
- ac排在aca前

用例

输入	2,3,4 2
输出	2,3 2,3,4 2,4 3,4
说明	最小密码数量是两个，可能有三种组合： 2,3 2,4 3,4 三个密码有一种： 2,3,4

输入	2,0 1
输出	0 0,2 2
说明	可能的密码组合，一个的有两种: 0 2 两个的有一个: 0,2

题目解析

这题就是一道求组合的题目，可以考虑使用dfs，只是通常情况下，每一个组合的层级数都是固定的，但是本题的组合层级数是动态的，即必须要大于等于第二行输入的值。

这个其实只要变通一下dfs的递归结束条件即可。大家可以参考算法源码中实现。

Java算法源码

```java
import java.util.*;

public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    String[] arr = sc.nextLine().split(",");
    int level = Integer.parseInt(sc.nextLine());
    getResult(arr, level);
  }

  public static void getResult(String[] arr, int level) {
    Arrays.sort(arr);

    ArrayList<String> res = new ArrayList<>();
    dfs(arr, 0, level, new LinkedList<>(), res);

    if (res.size() > 0) {
      for (String v : res) {
        System.out.println(v);
      }
    } else {
      System.out.println("None");
    }
  }

  public static void dfs(
      String[] arr, int index, int level, LinkedList<String> path, ArrayList<String> res) {
    if (path.size() >= level) {
      StringJoiner sj = new StringJoiner(",");
      for (String v : path) sj.add(v);
      res.add(sj.toString());
    }

    for (int i = index; i < arr.length; i++) {
      path.add(arr[i]);
      dfs(arr, i + 1, level, path, res);
      path.removeLast();
    }
  }
}
```

JS算法源码

```
/* JavaScript Node ACM模式 控制台输入获取 */
const readline = require("readline");

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});

const lines = [];
rl.on("line", (line) => {
  lines.push(line);

  if (lines.length === 2) {
    const arr = lines[0].split(",");
    const level = parseInt(lines[1]);

    getResult(arr, level);

    lines.length = 0;
  }
});

function getResult(arr, level) {
  arr.sort();
```

```
25
26    const res = [];
27    dfs(arr, 0, level, [], res);
28
29    if (res.length) {
30      res.forEach((combination) => {
31        console.log(combination);
32      });
33    } else {
34      console.log("None");
35    }
36  }
37
38  function dfs(arr, index, level, path, res) {
39    if (path.length >= level) {
40      res.push(path.join(","));
41    }
42
43    for (let i = index; i < arr.length; i++) {
44      path.push(arr[i]);
45      dfs(arr, i + 1, level, path, res);
46      path.pop();
47    }
48  }
```

Python算法源码

```
1  # 输入获取
2  arr = input().split(",")
3  level = int(input())
4
5
6  def dfs(arr, index, level, path, res):
7      if len(path) >= level:
8          res.append(",".join(path))
9
10     for i in range(index, len(arr)):
11         path.append(arr[i])
12         dfs(arr, i + 1, level, path, res)
13         path.pop()
14
15
16 # 算法入口
17 def getResult():
18     arr.sort()
19
20     res = []
21     dfs(arr, 0, level, [], res)
22
23     if len(res) > 0:
24         for s in res:
25             print(s)
```

```
            print(s)
    else:
        print("None")


# 调用算法
getResult()
```