


# 剑指17 打印从1到最大的n位数

和大数，数字位数相关的回溯法，当做典型题背诵一下吧

分类：回溯法/DFS，分治算法位运算？（体现在分开计算不同长度数字的结果，最后结果合并起来），递归

题目：输入数字  $n$ ，按顺序打印出从 1 到最大的  $n$  位十进制数。比如输入 3，则打印出 1、2、3 一直到最大的 3 位数 999。

理解：这道题实际是面试题，要考察的是大数。题目出得不好，应该输出字符串数组。

思路：大数放不下 `int` 中，所以需要用 `string` 处理所有数，一位一位拼接，所以问题变成了——字符的拼接组合尝试所有可能。问题的目标要求得到所有的数，而其中的较长的数是可以由短的数后增加字符得到的，第一个字符有 9 种可能，然后第二个字符在这每个分支上又延展出 10 种，如此下去。很容易想到树  的结构，问题就变成了 dfs。

评论：这道题用 **递归** 的 dfs 搜索算法。

```
1 // 总结的标准答案
2 // 最外层n个循环，每次循环都dfs只求出长度为i的所有数。
3 // 内部的dfs：求出长度为len的所有数。变成排列组合，第1位9种可能，第2位10种，第3位10种...，即树的深度优先搜索，且只记录下最底部叶子节点的值，即只push长度为len的树到答案中。
4 class Solution {
5 public:
6     vector<int> res;
7     string cur;
8     void dfs(int k, int len) {
9         if(k == len) { //递归的终止条件
10             res.push_back(stoi(cur));
11             return;
12         }
13         int i = k==0 ? 1 : 0;
14         for(; i <= 9; i++) {
15             cur.push_back(i + '0'); // int数转换为它的ascii码值,直接返回给char接收。
16             dfs(k + 1, len);
17             cur.pop_back();
18         }
19     }
20
21     vector<int> printNumbers(int n) {
22         for(int i = 1; i <= n; i++) {
23             dfs(0, i);
24         }
25         return res;
26     }
27 };
```

1 // 我觉得更好理解的别人的方法。只用了一个DFS：

2 // 全排列字符串的第0位到第n-1位。 存储结果时需去掉字符串前几位的0(0099没有意义，应为99)再放入结果。这样能记

录下长度小于n的那些数。

```
3 class Solution {
4 private:
5     vector<int> res;
6     string s;
7     void savenum()
8     {
9         int p = 0;
10        while(s[p] == '0' && p < s.size()) p++;
11        if(p < s.size()) res.push_back(stoi(s.substr(p)));
12
13    }
14    void dfs(int& n,int index)
15    {
16        if(index == n)
17        {
18            savenum();
19            return;
20        }
21        for(int i = 0;i < 10;i++)
22        {
23            s[index] = '0' + i;
24            dfs(n,index+1);
25        }
26    }
27 public:
28     vector<int> printNumbers(int n) {
29         s.resize(n,'0');
30         dfs(n,0);
31         return res;
32     }
33 };
```

```
1 // 我的思路: BFS
2 class Solution {
3 public:
4     vector<int> printNumbers(int n) {
5         queue<string> myq;
6         vector<int> res;
7         string s;
8         myq.push(""); // push一个起点占位符为空
9         while(!myq.empty()) {
10             s = myq.front();
11             myq.pop();
12             int num = 0;
13             if (s.empty()) {
14                 num = 1;
15             }
16             else {
17                 res.push_back(stoi(s));
18             }
19             if (s.size() < n) {
20                 for(; num <= 9; num++) {
21                     s.push_back(num + '0');
22                     myq.push(s);
23                     s.pop_back();
24                 }
25             }
26         }
27         return res;
28     }
29 };
```

```
25         }
26     }
27     return res;
28 }
29 };
```