E2-D题题解

> 作者：余绍函

题目描述

一个正整数为 n 元数，当且仅当其各位数的 n 次方之和等于这个正整数本身。
例如 153 是一个 3 元数，因为 $1^3+5^3+3^3=153$ 。
给定正整数 n ，请你求出所有 n 元数之和。

输入

本题测试点包含多组数据。
第一行，一个正整数 $T(1\leq T\leq 12)$, 表示数据组数。
对于每组数据：
一行，一个正整数 $n(1\leq n\leq 12)$，含义同题目描述。

输出

对于每组数据：
输出一行，一个正整数，表示所有 n元数之和。

输入样例

```
1
1
```

输出样例

```
45
```

解题思路

水仙花数

介题是神魔？水仙花数？

水仙花数（Narcissistic number）也被称为超完全数字不变数（pluperfect digital invariant, PPDI）、自恋数、自幂数、阿姆斯壮数或阿姆斯特朗数（Armstrong number），水仙花数是指一个 3 位数，它的每个数位上的数字的 3次幂之和等于它本身。例如：1^3 + 5^3+ 3^3 = 153。

不是。准确来说，水仙花数是自幂数的一个子集。举个例子：1 是一元数、二元数、三元数......但是只是一个**独身数**（一位数的每个数位上的数字的1次幂之和等于它本身），不是一个水仙花数、也不是四叶玫瑰数、五角星数等等（定义可以参考水仙花数，别问我为啥名字这么奇怪）。

暴力解法

不管你是否了解水仙花数的求法，相信大多数人第一反应会想到暴力解法。

```
#define n 3
for(int i = 100; i < 1000; i++)
{
        int sum = 0, tmp = i;
        while(tmp > 0)
        {
                sum += pow(tmp % 10, n);
                tmp /= 10;
        }
        if(sum == i)
                printf("%d\n", i);
}
```

这是用来求解水仙花数的暴力解法。我们发现，只要把循环的范围变得更大，通过调整我们肯定能找到所有的n元数。但是如果你求 12 元数，时间复杂度是否太大了些？先不说在accoding上能不能跑过，给你个1小时，能不能跑出来还是一回事。因此，如果这种方法能跑出来，可以用来跑出来打表（bushi），但绝对不是考察的方法。

换个思路

比起遍历每个数，我们还有没有什么遍历思路？当然有。举个例子，三元数是由`1, 8, 27, 64, 125, 216, 343, 512, 729` 的和组成的。那如果我们枚举这9个数字的和，再判断和是否是一个 n 元数，肯定能降低很多时间复杂度。怎么枚举呢？首先，枚举无顺序，比如`1 + 8 + 27`和`27 + 8 + 1`表示的意义是一样的。其次，一个数字可以被反复枚举。最后，枚举的数字个数应该等于和的数字位数，比如`1 + 1 + 1 = 3`是一位数不合法，但`125 + 216 + 64 = 405`是三位数合法。有了这个思路，我们直接来看代码：

```c
#include<stdio.h>
#define ll long long
#define max(a, b) (a)>(b)?(a):(b)

int n;
ll pw[10], cnt[10], bit[10], ans;

// check: 用来判断数字0-9在s中出现的次数是否与给定数组cnt相同。
int check(ll s)
{
    for(int i = 0; i < 10; i++)
        bit[i] = 0;
    while(s)
    {
        bit[s % 10]++;
        s /= 10;
    }
    for(int i = 0; i < 10; i++)
        if(cnt[i] != bit[i])
            return 0;
    return 1;
}

// ubound: 可供选择的n次方最大的底数, num: 求得的数字, limit: num的下界乘10
void dfs(int ubound, ll num, ll limit)
{
    if(num < limit / 10) // 递归的终止条件：num比下界更小
        return;
    ans += check(num) * num; // 如果是n元数，加到结果上
    for(int i = 0; i <= ubound; i++)
    {
        cnt[i]++;
        dfs(i, num + pw[i], limit * 10);
        cnt[i]--;
    }
}

int main()
{
    int T;
    scanf("%d", &T);
    while (T--)
    {
```

```
        scanf("%d", &n);
        ans = 0;
        for (int i = 0; i < 10; i++)
        {
            cnt[i] = 0;
            pw[i] = 1;
            for (int j = 1; j <= n; j++)
                pw[i] *= i;
        }
        dfs(9, 0, 1);
        printf("%lld\n", ans);
    }
    return 0;
}
```

OEIS大法

> 打表诚可贵，递归价更高。做题不规范，上机两行泪。

把这个方法写在最后，就是因为这并不算得上是一个很好方法。如果你理解了上文的算法，那么你已经收获了这篇题解的全部精华。写这个投机取巧的方法，主要是因为我是这么做的。
首先，OEIS上并没有这个数列的原数列。如果利用暴力法求出了一元数、二元数、三元数等等，把它们连在一起，就可以在OEIS上找到这个数列，给出了全部的一元数到十元数。

A252648 as a simple table

n	a(n)
0	1
1	0
2	1
3	2
4	3
5	4
6	5
7	6
8	7
9	8
10	9
11	0
12	1
13	0
14	1
15	153
16	370
17	371
18	407
19	0
20	1
21	1634
22	8208
23	9474
24	0
25	1
26	4150
27	4151
28	54748
29	92727
30	93084
31	194979

但是却缺少了十一元数和十二元数，这怎么办呢？从概率的角度讲，当 n 尽量大的时候，n 元数更难凑成。猜想 n 元数比 n 位自幂数只多个一， 就 AC 了！猜想正确！