P1249 最大乘积

题目描述

一个正整数一般可以分为几个互不相同的自然数的和，如 3=1+2，4=1+3，5＝1+4=2+3，6=1+5＝2+4。

现在你的任务是将指定的正整数 n 分解成若干个互不相同的自然数的和，且使这些自然数的乘积最大。

输入格式

只一个正整数 n，（3≤n≤10000）。

输出格式

第一行是分解方案，相邻的数之间用一个空格分开，并且按由小到大的顺序。

第二行是最大的乘积。

输入样例

10

输出样例

2 3 5
30

解析

本题存在两个难点：

1、高精度乘法
2、数字拆分

高精度直接套用计算即可，问题是数字该如何拆分呢。把n分拆成若干个互不相等的自然数的和的分法只有有限种，因而一定存在一种分法，使得这些自然数的乘积最大。

因1作因数，则显然乘积不会最大。把n分拆成若干个互不相等的自然数的和，因数个数越多，乘积越大。

为了使因数个数尽可能多，把n分成2+3+...+k直到满足2+3+...+k <= n且2+3+...+k+(k+1)大于n，令s = 2+3+...+k，一共k-1个数。

若s < n，则还需要分配n - s。如果能把数分到更小的数上，那么乘积更大，但不允许数重复，需要从大数开始向前，依次分配1。

如 s = 2 + 3 + 4 + 5 + 6 + 7

例如 n - s等于1
分配后：2 + 3 + 4 + 5 + 6 + 8
例如n - s等于3
分配后：2 + 3 + 4 + 6 + 7 + 8

即从某个位置开始，全部加1

可视为从2 + 3 + ... + k + (k + 1)使得s > n后删去某个数。

特殊情况，若和比n大1，则因数个数至少减少一个，为了使乘积最大，去掉最小的2，并将最后一个数加上1。

否则，去掉等于s-n的那个数

此时乘法最大(去掉可以使其值变为1，不影响结果)

编码

```
#include <bits/stdc++.h>//万能头文件
using namespace std;
string muli(string num1, string num2) {
    string res = "";
    int a1[3000], a2[3000], len1, len2, a[5000], len;

    memset(a1, 0, sizeof(a1));
    memset(a2, 0, sizeof(a2));
    memset(a, 0, sizeof(a));

    len1 = num1.size();
    for (int i = 0; i < len1; i++) {
        a1[i] = num1[len1 - 1 - i] - '0'; //存储真实数字
```

```
    }
    len2 = num2.size();
    for (int i = 0; i < len2; i++) {
        a2[i] = num2[len2 - 1 - i] - '0';
    }
//计算乘积
    for (int i = 0; i < len1; i++) {
        for (int j = 0; j < len2; j++) {
            a[i + j] += a1[i] * a2[j];
        }
    }
//长度算法1：设定积的长度为两数的位数和
    len = len1 + len2;
//处理进位
    for (int i = 0; i < len; i++) {
        a[i + 1] += a[i] / 10;
        a[i] %= 10;
    }
//处理高位为0的情况
    while (a[len] == 0 && len > 0) {
        len--;
    }
    for (int i = len; i >= 0; i--) {
        res += a[i] + '0';
    }
    return res;
}
string f(int x) {//f函数用来把任意一个整型数字转化为字符串的形式。
    int i = 0, j;
    string p = "";
    char ch[10], t;
    do {
        ch[i] = x % 10 + '0';
        x /= 10;
        i++;
    } while (x != 0);//只要x不为0，就去掉末位。
    ch[i] = '\0';
    for (j = 0, i--; j <= i / 2; j++, i--) {
        t = ch[j];
        ch[j] = ch[i];
        ch[i] = t;
    }
    return ch;//返回这个字符串
}
int n = 0;
//ans数组用来存拆分的数字
int ans[1001];
//s数组用来存每一个数字的字符串，方便做高精度乘法，m存总乘积，初值为"1"。
```

```
string m = "1";
//拆出来的数字总量
int c = 0;
//将数字进行拆分
void Split(int n) {
    //第一轮拆分，将2-k分配到数组中
    for (int i = 2; i <= n; i++) {
        if (n >= i) {
            //用来存储拆分出来的每一个数字
          n -= i;
            //每拆分出1个数，n就减去这个数，在用s数组存下等同于这个数的字符
          ans[c++] = i;
        } else {
            break;//不能再拆分就终止循环
       }
    }
    while (n > 0) {
        //逆序倒推
       for (int i = c - 1; i >= 0; i--) {
            if (n > 0) {
                ans[i]++;
                n--;//多的数分担给其他数
          }
        }
    }
}
int main() {
    scanf("%d", &n);
    Split(n);
    for (int i = 0; i < c; i++) {
        //输出每个拆分数
       cout << ans[i] << " ";
        //调用高精度乘法
       m = muli(f(ans[i]), m);
    }
    cout << endl << m;//输出总乘积
    return 0;
}
```