SJTU 1296

题目：计算序列 $\{a_i\}$ 本质不同的最长下降子序列数目。

本题关键在于**清除额外的分支**。我们设 $f(i)$ 为以第 i 个元素结尾的最长下降子序列长度，$d(i)$ 为以第 i 个元素结尾的最长下降子序列个数，那么在计算完 $f(i)$，发现存在 $j<i$ 使得 $f(i)=f(j)$ 且 $a_i=a_j$ 时，显然 $f(i)$ 的方案数目包含了 $f(j)$ 的。这时候就令 $d(j):=0$ 以防止后续重复计算。

注意这里递推的时候，我们实际上隐藏了一个状态的维度：当前元素。也就是说，完整的状态设计应该是设 $f(i,j)$ 为考虑了前 i 个元素，以第 j 个元素结尾的最长下降子序列长度，$d(i,j)$ 为考虑了前 i 个元素，**经历了去重之后**以第 j 个元素结尾的最长下降子序列长度的个数。这一被隐藏的维度实际上是用滚动数组的手法消去了，但在列方程的时候必须考虑到这一点。

此外，由于本题的数据量很大，需要用到高精度。

```cpp
#include <bits/stdc++.h>
#define INF 2000000000
using namespace std;
typedef unsigned long long ll;
int read(){
    int f = 1, x = 0;
    char c = getchar();
    while(c < '0' || c > '9'){if(c == '-') f = -f; c = getchar();}
    while(c >= '0' && c <= '9')x = x * 10 + c - '0', c = getchar();
    return f * x;
}
struct Simple_Bigint{
    int num[105];
    Simple_Bigint(){
        memset(num, 0, sizeof(num));
        num[0] = 1, num[1] = 0;
    }
    Simple_Bigint& operator += (const Simple_Bigint& b){
        int ws = max(num[0], b.num[0]), x = 0;
        for (int i = 1; i <= ws; ++i)   {
            x = x + num[i] + b.num[i];
            num[i] = x % 10;
            x /= 10;
        }
        num[0] = ws;
        if (x != 0) num[++num[0]] = x;
        return *this;
    }
    Simple_Bigint& operator= (int x){
        memset(num, 0, sizeof(num));
        num[0] = 0;
        while (x > 0)
            num[++num[0]] = x % 10, x /= 10;
        return *this;
    }
    Simple_Bigint& operator= (const Simple_Bigint& b){
        memset(num, 0, sizeof(num));
        num[0] = b.num[0];
        for (int i = 1; i <= num[0]; ++i)
```

```
            num[i] = b.num[i];
        return *this;
    }
    bool operator! (){
        return num[0] == 1 && num[1] == 0;
    }
    void output(){
        for (int i = num[0]; i >= 1; --i)
            putchar(num[i] + '0');
    }
};
int n, a[5005];
int f[5005] = {0};
Simple_Bigint d[5005];
void init(){
    n = read();
    for (int i = 1; i <= n; ++i)
        a[i] = read();
}
void solve(){
    f[1] = 1, d[1] = 1;
    int ans = 1;
    Simple_Bigint ans1;
    ans1 = 0;
    for (int i = 2; i <= n; ++i){
        f[i] = 1;
        for (int j = i - 1; j >= 1; --j){
            if(a[j] > a[i]) {
                if(f[j] + 1 == f[i]) d[i] += d[j];
                else if(f[i] < f[j] + 1) f[i] = f[j] + 1, d[i] = d[j];
            }
        }
        for (int j = 1; j < i; ++j)
            if (a[i] == a[j] && f[i] == f[j])
                d[j] = 0;
        if(!d[i]) d[i] = 1;
        ans = max(ans, f[i]);
    }
    for (int i = 1; i <= n; ++i)
        if(f[i] == ans) ans1 += d[i];
    printf("%d ", ans);
    ans1.output();
    printf("\n");
}
int main(){
    init();
    solve();
    return 0;
}
```