原创 【luogu 5367】康托展开 + 逆康托展开 + 树状数组

2019-09-24 13:33:25 我是一只计算鸡 阅读数 7 更多

康托展开：求一个 1 ~ n 的排列的排名，如 1....n 的排名为1

$$A[0]*(n-1)!+A[1]*(n-2)!+...+A[n-1]*0!$$

$A[i]$表示小于当前位置的数的个数。

```
// 给定排列 求名次
#include<bits/stdc++.h>
using namespace std;
const int maxn = 10 + 7;
typedef long long ll;
int a[maxn];
int cator(int *a)  // 计算排列 a 的名次
{
        ll ans = 0;
        for(int i = 1; i <= 5; i++) {
                ll cnt = 0, sum = 1, num = 1;
                for(int j = i + 1; j <= 5; j++) {
                        if(a[i] > a[j]) cnt++;
                        sum *= num;
                        num++;
                }
                ans = (ans + cnt * sum);
        }
        cout << ans + 1 << endl;
        return 0;
}
int main()
{
         int n;
         for(int i = 1; i <= 5; i++) a[i] = i;
         cator(a);
        return 0;
}
```

逆康托展开：已知一个排列的名次，求这个排列。

我们可以通过反解 $A[0]*(n-1)!+A[1]*(n-2)!+...+A[n-1]*0!$ 。很明显是每一个阶乘的系数代表小于当前这个数的个数就能确定

```
// 逆康托展开
#include<bits/stdc++.h>
using namespace std;
int f[] = {1, 1, 2, 6, 24, 120, 720, 5040, 40320, 362880}; //阶乘预处理
int reversecator(int n, int x) // 逆康托展开 n 个数的排列 名次为 x
{
        vector<int>res; // 当前能选的数
        vector<int>ans; // 答案
        for(int i = 1; i <= n; i++) res.push_back(i);
        for(int i = n; i >= 1; i--) {
                int r = x % f[i-1];  // 余数
                int c = x / f[i-1];  // 倍数 小于当前位置的数字有几个
                x = r;
                sort(res.begin(), res.end());  // 对剩下的数排序
                ans.push_back(res[c]);  // 选第t+1个数
                res.erase(res.begin() + c); // 移除当前所选的数
        }
        for(auto it = ans.begin(); it != ans.end(); it++) cout << *it << " ";
        cout << endl;
        return 0;
```

```
}
int main()
{
        int n, x;
        while(scanf("%d %d", &n, &x) == 2) {
                reversecator(n, x - 1);
        }
        return 0;
}
```

康托展开+树状数组：我们在求康托展开时，为了找到小于当前数的个数，我们开始使用$\theta(n)$的时间复杂度。但是我们可以使用树状数组把找小于当前的$\theta(log(n))$。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int maxn = 1000000 + 7;
const ll mod = 998244353;
int n, a[maxn];
ll tree[maxn];
ll f[maxn];
void init()  // 阶乘预处理
{
        f[1] = 1;
        for(int i = 2; i <= maxn; i++) f[i] = (f[i-1] * i) % mod;
}
int lowbit(int x)
{
        return x & (-x);
}
int add(int x)  // 更新
{
        while(x <= n) {
                tree[x] += 1;
                x += lowbit(x);
        }
        return 0;
}
int query(int x)  // 求和
{
        int res = 0;
        while(x > 0) {
                res += tree[x];
                x -= lowbit(x);
        }
        return res;
}
ll cantor(int* a)  // 康托展开
{
        memset(tree, 0, sizeof(tree));
        ll ans = 0;
        for(int i = 1; i <= n; i++) {
                ll cnt = query(a[i]);
                add(a[i]);
                // a[i] 前面的个数是已经被使用过了的,还要去除本身, 所以a[i] - cnt -1
                ans = (ans % mod + (a[i] - cnt - 1) % mod * f[n-i] % mod) % mod;
        }
        return ans;
}
int main()
{
        init();
        while(scanf("%d", &n) == 1) {
                for(int i = 1; i <= n; i++) scanf("%d", &a[i]);
                ll ans = cantor(a);
                printf("%lld\n", ans + 1);
        }
        return 0;
}
```

有 0 个人打赏

文章最后发布于: 201