

## 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

版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。

本文链接：<https://blog.csdn.net/giftedpanda/article/details/101279885>

康托展开：求一个  $1 \sim n$  的排列的排名，如  $1 \dots n$  的排名为1

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

$A[i]$ 表示小于当前位置的数的个数。

```

1 // 给定排列 求名次
2 #include<bits/stdc++.h>
3 using namespace std;
4 const int maxn = 10 + 7;
5 typedef long long ll;
6 int a[maxn];
7 int cator(int *a) // 计算排列 a 的名次
8 {
9     ll ans = 0;
10    for(int i = 1; i <= 5; i++) {
11        ll cnt = 0, sum = 1, num = 1;
12        for(int j = i + 1; j <= 5; j++) {
13            if(a[i] > a[j]) cnt++;
14            sum *= num;
15            num++;
16        }
17        ans = (ans + cnt * sum);
18    }
19    cout << ans + 1 << endl;
20    return 0;
21 }
22 int main()
23 {
24     int n;
25     for(int i = 1; i <= 5; i++) a[i] = i;
26     cator(a);
27     return 0;
28 }
```

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  $A[0] * (n-1)! + A[1] * (n-2)! + \dots + A[n-1] * 0!$ 。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

```

1 // 逆康托展开
2 #include<bits/stdc++.h>
3 using namespace std;
4 int f[] = {1, 1, 2, 6, 24, 120, 720, 5040, 40320, 362880}; //阶乘预处理
5 int reversecator(int n, int x) // 逆康托展开 n 个数的排列 名次为 x
6 {
7     vector<int>res; // 当前能选的数
8     vector<int>ans; // 答案
9     for(int i = 1; i <= n; i++) res.push_back(i);
10    for(int i = n; i >= 1; i--) {
11        int r = x % f[i-1]; // 余数
12        int c = x / f[i-1]; // 倍数 小于当前位置的数字有几个
13        x = r;
14        sort(res.begin(), res.end()); // 对剩下的数排序
15        ans.push_back(res[c]); // 选第t+1个数
16        res.erase(res.begin() + c); // 移除当前所选的数
17    }
18    for(auto it = ans.begin(); it != ans.end(); it++) cout << *it << " ";
19    cout << endl;
20    return 0;
}
```

```

21 | }
    | 22 | int main()
23 | {
24 |     int n, x;
25 |     while(scanf("%d %d", &n, &x) == 2) {
26 |         reversecator(n, x - 1);
27 |     }
28 |     return 0;
29 | }

```

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 $\theta(\log(n))$ 。

```

1 | #include<bits/stdc++.h>
2 | using namespace std;
3 | typedef long long ll;
4 | const int maxn = 1000000 + 7;
5 | const ll mod = 998244353;
6 | int n, a[maxn];
7 | ll tree[maxn];
8 | ll f[maxn];
9 | void init() // 阶乘预处理
10 | {
11 |     f[1] = 1;
12 |     for(int i = 2; i <= maxn; i++) f[i] = (f[i-1] * i) % mod;
13 | }
14 | int lowbit(int x)
15 | {
16 |     return x & (-x);
17 | }
18 | int add(int x) // 更新
19 | {
20 |     while(x <= n) {
21 |         tree[x] += 1;
22 |         x += lowbit(x);
23 |     }
24 |     return 0;
25 | }
26 | int query(int x) // 求和
27 | {
28 |     int res = 0;
29 |     while(x > 0) {
30 |         res += tree[x];
31 |         x -= lowbit(x);
32 |     }
33 |     return res;
34 | }
35 | ll cantor(int* a) // 康托展开
36 | {
37 |     memset(tree, 0, sizeof(tree));
38 |     ll ans = 0;
39 |     for(int i = 1; i <= n; i++) {
40 |         ll cnt = query(a[i]);
41 |         add(a[i]);
42 |         // 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
43 |         ans = (ans % mod + (a[i] - cnt - 1) % mod * f[n-i] % mod) % mod;
44 |     }
45 |     return ans;
46 | }
47 | int main()
48 | {
49 |     init();
50 |     while(scanf("%d", &n) == 1) {
51 |         for(int i = 1; i <= n; i++) scanf("%d", &a[i]);
52 |         ll ans = cantor(a);
53 |         printf("%lld\n", ans + 1);
54 |     }
55 |     return 0;
56 | }

```

1024

程序员节，  
为程序员加油！

关闭

有 0 个人打赏

文章最后发布于: 201

©2019 CSDN 皮肤主题: 终极编程指南 设计师: CSDN官方博客

1024

程序员节，为程序员加油！

关闭