

polya计数

组合数学 polya计数

定理总结：

计数总数 = 在每一种置换下染色不变的染色数/所有置换数

简单应用

hdoj 3923

题目意思是给我们n个小球，求用m种颜色在一个圈上排列的方法数，我们只需要列举出每一种置换的循环节个数就好。

1.旋转：

旋转的方法有n种，每一种的循环节个数为gcd(n, i)(0<=i<=n), 循环节长度为n/gcd(n, i)。

2.翻转：

翻转要分情况：

(1)n为奇数：当n是奇数时只有一种翻折方式，沿着一个点和对边的中点翻折。循环节个数为(n-1)/2+1。

(2)n为偶数：当n为偶数的时候有两种翻折方式，一个是沿着一个点和对边翻折，循环节为(n-2)/2+2，一个是沿着一条边的中点和对边的中点翻折，循环节是n/2。

```
1. #include <iostream>
2. #include <algorithm>
3. #include <cstdio>
4. #include <cstring>
5. using namespace std;
6. const long long mod = 1e9 + 7;
7. int n, c;
8.
9. long long p(long long base, long long r)
10. {
11.     long long ret = 1;
12.     while (r) {
13.         if (r & 1) {
14.             ret = (ret * base) % mod;
15.         }
16.         r >>= 1;
17.         base = (base * base) % mod;
18.     }
19.     return ret;
20. }
21.
22. int gcd(int a, int b)
23. {
24.     int r;
25.     while (b) {
26.         r = a % b;
27.         a = b;
28.         b = r;
29.     }
30.     return a;
31. }
32.
33. int main()
34. {
35.     int t, tc = 1;
36.     scanf("%d", &t);
37.     while (t --) {
38.         scanf("%d%d", &c, &n);
39.         long long ans;
40.         if (n & 1) {
41.             ans = (1LL * p(c, (n + 1) / 2) * n % mod);
42.         } else {
43.             ans = (1LL * n / 2 * p(c, n / 2) % mod + 1LL * n / 2 * p(c, n / 2 + 1) % mod) % mod;
44.         }
45.         for (int i = 1; i <= n; i ++) {
46.             ans = (ans + p(c, gcd(n, i))) % mod;
```

```

47.     }
48.     ans = (ans * p(2 * n, mod - 2) % mod);
49.     printf("Case #d: %d\n", tc ++, ans);
50. }
51. return 0;
52. }

```

欧拉函数优化

poj2154

题目要求：给出两个整数n和p，代表n个珠子，n种颜色，要求不同的项链数，并对结果mod (p) 处理。
置换只有旋转一种方式，那么共有n个置换。

暴力枚举每一种置换会超时，可以考虑这样的优化：我们只枚举循环节的长度*l*（这样循环节的个数就是 $\frac{n}{l}$ ），显然*l*必然是n的因数，如果可以这样做，我们就可以在 \sqrt{n} 的时间里完成。

我们知道对于第*i*个置换，其循环节长度为*gcd*(*n, i*),我们要找的就是使*gcd*(*n, i*) = *l*的*i*的个数，也就是

$$S_n^l = s_{\frac{n}{l}}^1 = \phi(\frac{n}{l})$$

$$ans = \frac{\sum_{l|n} \phi(l) \cdot n^{\frac{n}{l}}}{n}$$

$$= \sum_{l|n} \phi(l) \cdot n^{\frac{n}{l}-1}$$

```

1. #include <iostream>
2. #include <cstdio>
3. #include <cstring>
4. #include <algorithm>
5. #include <cmath>
6. using namespace std;
7. const int MAX = 1e5 + 10;
8. int n, mod, vis[MAX], prime[MAX], cnt, euler[MAX];
9.
10. void init()
11. {
12.     euler[1] = 1;
13.     for (int i = 2; i < MAX; i++) {
14.         if (!vis[i]) {
15.             prime[cnt++] = i;
16.             euler[i] = i - 1;
17.         }
18.         for (int j = 0; j < cnt; j++) {
19.             if (1LL * i * prime[j] >= MAX) break;
20.             vis[i * prime[j]] = 1;
21.             if (i % prime[j] == 0) {
22.                 euler[i * prime[j]] = euler[i] * prime[j];
23.                 break;
24.             } else {
25.                 euler[i * prime[j]] = euler[i] * (prime[j] - 1);
26.             }
27.         }
28.         //cout << euler[i] << " ";
29.     }
30. }
31.
32. int p(int base, int r)
33. {
34.     int ret = 1;
35.     while (r) {
36.         if (r & 1) {
37.             ret = (ret * base) % mod;
38.         }
39.         r >>= 1;
40.         base = (base * base) % mod;
41.     }
42.     return ret;
43. }
44.
45. int getEuler(int x)
46. {
47.     if (x < MAX) return euler[x];
48.     int ret = x;

```

```

49.     for (int i = 0; i < cnt; i ++) {
50.         if (prime[i] * prime[i] > x) break;
51.         if (x % prime[i] == 0) {
52.             ret = ret / prime[i] * (prime[i] - 1);
53.             x /= prime[i];
54.             while (x % prime[i] == 0) {
55.                 x /= prime[i];
56.             }
57.         }
58.     }
59.     if (x != 1) {
60.         ret = ret / x * (x - 1);
61.     }
62.     return ret;
63. }
64.
65. int main()
66. {
67.     init();
68.     int t;
69.     scanf("%d", &t);
70.     while (t --) {
71.         scanf("%d%d", &n, &mod);
72.         int ans = 0;
73.         for (int l = 1; l * l <= n; l ++) {
74.             if (n % l == 0) {
75.                 // 乘方幂次减一的原因是最后要除以总置换数n
76.                 if (l * l < n) {
77.                     ans = (ans + getEuler(l) % mod * p(n % mod, (n / l - 1) )) % mod;
78.                     ans = (ans + getEuler(n / l) % mod * p(n % mod, (l - 1) )) % mod;
79.                 } else {
80.                     ans = (ans + getEuler(l) % mod * p(n % mod, (l - 1) )) % mod;
81.                 }
82.             }
83.         }
84.         printf("%d\n", ans);
85.     }
86. }

```

矩阵优化

poj 2888

题意：做一串项链，长度为 n ，有 m 种珠子，旋转后项链和旋转前项链视为同一种。有 k 种限制，表示两种颜色的珠子不能挨在一起，求染色数。

除了用到了上面的euler函数优化之外，还用到了一个计数技巧：

在图上求从 a 走到 b 恰好用 i 步的走法 $= (\text{邻接矩阵}^i)[a][b]$

试想：可以把循环节看作是一个回路，假设循环节的长度为 l ，那么第1个珠子和第 $l + 1$ 个珠子的颜色是一样的，枚举第1个珠子的颜色，上色的种数就是从第一个颜色出发，走 l 步回到第1种颜色的路径数。

因此，在循环节长度为 l 的置换下，染色不变的染色数就是邻接矩阵的 l 次幂的迹。

```

1. #include <iostream>
2. #include <cstdio>
3. #include <algorithm>
4. #include <cstring>
5. using namespace std;
6. const int MAX = 1e9 + 10;
7. const int MAXM = 12;
8. const int MAXP = 1e5 + 10;
9. const int mod = 9973;
10. int prime[MAXP], cnt, vis[MAXP], n, k, m;
11.
12. struct Matrix {
13.     int r, c;
14.     int mat[MAXM][MAXM];
15.     Matrix () : r(m), c(m) {
16.         memset(mat, 0, sizeof mat);
17.     }
18.     Matrix (int i) : r(m), c(m) {
19.         memset(mat, 0, sizeof mat);
20.         for (int j = 1; j <= r; j ++) {
21.             mat[j][j] = i;

```

```

22.     }
23. }
24. Matrix (int rr, int cc) : r(rr), c(cc) {
25.     memset(mat, 0, sizeof mat);
26. }
27. void fill(void) {
28.     for (int i = 1; i <= r; i++) {
29.         for (int j = 1; j <= c; j++) {
30.             mat[i][j] = 1;
31.         }
32.     }
33. }
34. void mul(const Matrix& rhs) {
35.     Matrix ret;
36.     for (int i = 1; i <= r; i++) {
37.         for (int j = 1; j <= rhs.c; j++) {
38.             for (int k = 1; k <= c; k++) {
39.                 ret.mat[i][j] = ((ret.mat[i][j] + mat[i][k] * rhs.mat[k][j]) ) % mod;
40.             }
41.         }
42.     }
43.     memcpy(mat, ret.mat, sizeof ret.mat);
44. }
45. static Matrix p(Matrix base, long long r) {
46.     Matrix ret;
47.     ret.c = ret.r = m;
48.     for (int i = 1; i <= m; i++) {
49.         ret.mat[i][i] = 1;
50.     }
51.     while (r) {
52.         if (r & 1) {
53.             ret.mul(base);
54.         }
55.         r >>= 1;
56.         base.mul(base);
57.     }
58.     return ret;
59. }
60. } g;
61.
62. int quickp(int base, int r)
63. {
64.     int ret = 1;
65.     while (r) {
66.         if (r & 1) {
67.             ret = (ret * base) % mod;
68.         }
69.         r >>= 1;
70.         base = (base * base) % mod;
71.     }
72.     return ret;
73. }
74.
75.
76. void getprime()
77. {
78.     for (int i = 2; i < MAXP; i++) {
79.         if (!vis[i]) {
80.             prime[cnt++] = i;
81.         }
82.         for (int j = 0; j < cnt; j++) {
83.             if (i * prime[j] >= MAXP) break;
84.             vis[i * prime[j]] = 0;
85.             if (i % prime[j] == 0) {
86.                 break;
87.             }
88.         }
89.     }
90. }
91.
92. int euler(int x)
93. {
94.     int i = 0, ret = x;
95.     while (prime[i] * prime[i] <= x) {
96.         if (x % prime[i] == 0) {
97.             ret = ret / prime[i] * (prime[i] - 1);
98.             do {
99.                 x /= prime[i];

```

```

100.         } while (x % prime[i] == 0);
101.     }
102.     ++ i;
103. }
104. if (x != 1) {
105.     ret = ret / x * (x - 1);
106. }
107. return ret % mod;
108. }
109.
110. int gettr(const Matrix& mm, long long r)
111. {
112.     int ret = 0;
113.     Matrix mmm = Matrix::p(mm, r);
114.     for (int i = 1; i <= mmm.r; i++) {
115.         ret = (ret + mmm.mat[i][i]) % mod;
116.     }
117.     return ret;
118. }
119.
120. int polya(void)
121. {
122.     int ans = 0;
123.     for (int i = 1; i * i <= n; i++) {
124.         if (n % i == 0) {
125.             if (i * i != n) {
126.                 ans = (ans + euler(n / i) * gettr(g, i)) % mod;
127.                 ans = (ans + euler(i) * gettr(g, n / i)) % mod;
128.             } else {
129.                 ans = (ans + euler(i) * gettr(g, i)) % mod;
130.             }
131.         }
132.     }
133.     ans = (ans * quickp(n % mod, mod - 2)) % mod;
134.     return ans;
135. }
136.
137. int main()
138. {
139.     getprime();
140.     int t;
141.     scanf("%d", &t);
142.     while (t --) {
143.         scanf("%d%d%d", &n, &m, &k);
144.         g.c = g.r = m;
145.         g.fill();
146.         int u, v;
147.         for (int i = 1; i <= k; i++) {
148.             scanf("%d%d", &u, &v);
149.             g.mat[u][v] = g.mat[v][u] = 0;
150.         }
151.         printf("%d\n", polya());
152.     }
153. }

```