

Algorithm Library

stlvdv

2021 年 10 月 20 日

目录

1	多项式	1
1.1	FFT - tourist	1
1.2	形式幂级数	6
2	数论	6
2.1	简单的防爆模板	6
2.2	筛法	7
2.2.1	线性素数筛	7
2.3	中国剩余定理	8
2.3.1	CRT	8
2.3.2	EXCRT	8
2.4	BSGS	8
3	线性代数	8
3.1	高斯-约旦消元法	8
3.2	高斯消元法-bitset	8
3.3	线性基	8
3.4	矩阵树定理	8
3.5	LGV 引理	8
4	组合数学	8
4.1	组合数预处理	8
4.2	小球盒子模型	8
4.3	斯特林数	8
5	图论	8
5.1	并查集	8
5.2	最小树形图	9
5.3	最近公共祖先	9
5.4	最大流	9
5.5	2-sat	9
5.6	最大团	9
6	字符串	9
6.1	KMP	9
6.2	Manacher	9
6.3	Trie	9
6.4	01-Trie	9

1 多项式

1.1 FFT - tourist

```

1  /* copy from tourist */
2  namespace FFT {
3      typedef double dbl;
4
5      struct num {
6          dbl x, y;
7          num() { x = y = 0; }
8          num(dbl x, dbl y) : x(x), y(y) { }
9      };
10
11     inline num operator+(num a, num b) { return num(a.x + b.x, a.y + b.y); }
12     inline num operator-(num a, num b) { return num(a.x - b.x, a.y - b.y); }
13     inline num operator*(num a, num b) { return num(a.x * b.x - a.y * b.y, a
        .x * b.y + a.y * b.x); }
14     inline num conj(num a) { return num(a.x, -a.y); }
15
16     int base = 1;
17     vector<num> roots = { {0, 0}, {1, 0} };
18     vector<int> rev = { 0, 1 };
19
20     const dbl PI = acos(-1.0);
21
22     void ensure_base(int nbase) {
23         if (nbase <= base) {
24             return;
25         }
26         rev.resize(1 << nbase);
27         for (int i = 0; i < (1 << nbase); i++) {
28             rev[i] = (rev[i >> 1] >> 1) + ((i & 1) << (nbase - 1));
29         }
30         roots.resize(1 << nbase);
31         while (base < nbase) {
32             dbl angle = 2 * PI / (1 << (base + 1));
33             for (int i = 1 << (base - 1); i < (1 << base); i++) {
34                 roots[i << 1] = roots[i];
35                 dbl angle_i = angle * (2 * i + 1 - (1 << base));
36                 roots[(i << 1) + 1] = num(cos(angle_i), sin(angle_i));
37             }
38             base++;
39         }

```

```

40     }
41
42     void fft(vector<num>& a, int n = -1) {
43         if (n == -1) {
44             n = a.size();
45         }
46         assert((n & (n - 1)) == 0);
47         int zeros = __builtin_ctz(n);
48         ensure_base(zeros);
49         int shift = base - zeros;
50         for (int i = 0; i < n; i++) {
51             if (i < (rev[i] >> shift)) {
52                 swap(a[i], a[rev[i] >> shift]);
53             }
54         }
55         for (int k = 1; k < n; k <= 1) {
56             for (int i = 0; i < n; i += 2 * k) {
57                 for (int j = 0; j < k; j++) {
58                     num z = a[i + j + k] * roots[j + k];
59                     a[i + j + k] = a[i + j] - z;
60                     a[i + j] = a[i + j] + z;
61                 }
62             }
63         }
64     }
65
66     vector<num> fa, fb;
67
68     vector<long long> multiply(vector<int>& a, vector<int>& b) {
69         int need = a.size() + b.size() - 1;
70         int nbase = 1;
71         while ((1 << nbase) < need) nbase++;
72         ensure_base(nbase);
73         int sz = 1 << nbase;
74         if (sz > (int)fa.size()) {
75             fa.resize(sz);
76         }
77         for (int i = 0; i < sz; i++) {
78             int x = (i < (int)a.size() ? a[i] : 0);
79             int y = (i < (int)b.size() ? b[i] : 0);
80             fa[i] = num(x, y);
81         }
82         fft(fa, sz);

```

```

83     num r(0, -0.25 / (sz >> 1));
84     for (int i = 0; i <= (sz >> 1); i++) {
85         int j = (sz - i) & (sz - 1);
86         num z = (fa[j] * fa[j] - conj(fa[i] * fa[i])) * r;
87         if (i != j) {
88             fa[j] = (fa[i] * fa[i] - conj(fa[j] * fa[j])) * r;
89         }
90         fa[i] = z;
91     }
92     for (int i = 0; i < (sz >> 1); i++) {
93         num A0 = (fa[i] + fa[i + (sz >> 1)]) * num(0.5, 0);
94         num A1 = (fa[i] - fa[i + (sz >> 1)]) * num(0.5, 0) * roots[(sz
95             >> 1) + i];
96         fa[i] = A0 + A1 * num(0, 1);
97     }
98     fft(fa, sz >> 1);
99     vector<long long> res(need);
100    for (int i = 0; i < need; i++) {
101        if (i % 2 == 0) {
102            res[i] = fa[i >> 1].x + 0.5;
103        } else {
104            res[i] = fa[i >> 1].y + 0.5;
105        }
106    }
107    return res;
108 }
109
110 vector<long long> square(const vector<int>& a) {
111     int need = a.size() + a.size() - 1;
112     int nbase = 1;
113     while ((1 << nbase) < need) nbase++;
114     ensure_base(nbase);
115     int sz = 1 << nbase;
116     if ((sz >> 1) > (int)a.size()) {
117         fa.resize(sz >> 1);
118     }
119     for (int i = 0; i < (sz >> 1); i++) {
120         int x = (2 * i < (int)a.size() ? a[2 * i] : 0);
121         int y = (2 * i + 1 < (int)a.size() ? a[2 * i + 1] : 0);
122         fa[i] = num(x, y);
123     }
124     fft(fa, sz >> 1);
125     num r(1.0 / (sz >> 1), 0.0);

```

```

125     for (int i = 0; i <= (sz >> 2); i++) {
126         int j = ((sz >> 1) - i) & ((sz >> 1) - 1);
127         num fe = (fa[i] + conj(fa[j])) * num(0.5, 0);
128         num fo = (fa[i] - conj(fa[j])) * num(0, -0.5);
129         num aux = fe * fe + fo * fo * roots[(sz >> 1) + i] * roots[(sz
            >> 1) + i];
130         num tmp = fe * fo;
131         fa[i] = r * (conj(aux) + num(0, 2) * conj(tmp));
132         fa[j] = r * (aux + num(0, 2) * tmp);
133     }
134     fft(fa, sz >> 1);
135     vector<long long> res(need);
136     for (int i = 0; i < need; i++) {
137         if (i % 2 == 0) {
138             res[i] = fa[i >> 1].x + 0.5;
139         } else {
140             res[i] = fa[i >> 1].y + 0.5;
141         }
142     }
143     return res;
144 }
145
146 vector<int> multiply_mod(vector<int>& a, vector<int>& b, int m, int eq =
    0) {
147     int need = a.size() + b.size() - 1;
148     int nbase = 0;
149     while ((1 << nbase) < need) nbase++;
150     ensure_base(nbase);
151     int sz = 1 << nbase;
152     if (sz > (int)fa.size()) {
153         fa.resize(sz);
154     }
155     for (int i = 0; i < (int)a.size(); i++) {
156         int x = (a[i] % m + m) % m;
157         fa[i] = num(x & ((1 << 15) - 1), x >> 15);
158     }
159     fill(fa.begin() + a.size(), fa.begin() + sz, num{ 0, 0 });
160     fft(fa, sz);
161     if (sz > (int)fb.size()) {
162         fb.resize(sz);
163     }
164     if (eq) {
165         copy(fa.begin(), fa.begin() + sz, fb.begin());

```

```

166     } else {
167         for (int i = 0; i < (int)b.size(); i++) {
168             int x = (b[i] % m + m) % m;
169             fb[i] = num(x & ((1 << 15) - 1), x >> 15);
170         }
171         fill(fb.begin() + b.size(), fb.begin() + sz, num{ 0, 0 });
172         fft(fb, sz);
173     }
174     dbl ratio = 0.25 / sz;
175     num r2(0, -1);
176     num r3(ratio, 0);
177     num r4(0, -ratio);
178     num r5(0, 1);
179     for (int i = 0; i <= (sz >> 1); i++) {
180         int j = (sz - i) & (sz - 1);
181         num a1 = (fa[i] + conj(fa[j]));
182         num a2 = (fa[i] - conj(fa[j])) * r2;
183         num b1 = (fb[i] + conj(fb[j])) * r3;
184         num b2 = (fb[i] - conj(fb[j])) * r4;
185         if (i != j) {
186             num c1 = (fa[j] + conj(fa[i]));
187             num c2 = (fa[j] - conj(fa[i])) * r2;
188             num d1 = (fb[j] + conj(fb[i])) * r3;
189             num d2 = (fb[j] - conj(fb[i])) * r4;
190             fa[i] = c1 * d1 + c2 * d2 * r5;
191             fb[i] = c1 * d2 + c2 * d1;
192         }
193         fa[j] = a1 * b1 + a2 * b2 * r5;
194         fb[j] = a1 * b2 + a2 * b1;
195     }
196     fft(fa, sz);
197     fft(fb, sz);
198     vector<int> res(need);
199     for (int i = 0; i < need; i++) {
200         long long aa = fa[i].x + 0.5;
201         long long bb = fb[i].x + 0.5;
202         long long cc = fa[i].y + 0.5;
203         res[i] = (aa + ((bb % m) << 15) + ((cc % m) << 30)) % m;
204     }
205     return res;
206 }
207
208 vector<int> square_mod(vector<int>& a, int m) {

```

```

209     return multiply_mod(a, a, m, 1);
210 }
211 };

```

1.2 形式幂级数

2 数论

2.1 简单的防爆模板

```

1 namespace SimpleMod {
2     constexpr int MOD = (int)1e9 + 7;
3     inline int norm(long long a) { return (a % MOD + MOD) % MOD; }
4     inline int add(int a, int b) { return a + b >= MOD ? a + b - MOD : a + b
5         ; }
6     inline int sub(int a, int b) { return a - b < 0 ? a - b + MOD : a - b; }
7     inline int mul(int a, int b) { return (int)((long long)a * b % MOD); }
8     inline int powmod(int a, long long b) {
9         int res = 1;
10        while (b > 0) {
11            if (b & 1) res = mul(res, a);
12            a = mul(a, a);
13            b >>= 1;
14        }
15        return res;
16    }
17    inline int inv(int a) {
18        a %= MOD;
19        if (a < 0) a += MOD;
20        int b = MOD, u = 0, v = 1;
21        while (a) {
22            int t = b / a;
23            b -= t * a; swap(a, b);
24            u -= t * v; swap(u, v);
25        }
26        assert(b == 1);
27        if (u < 0) u += MOD;
28        return u;
29    }
30 }

```


2.2 筛法

2.2.1 线性素数筛

```
1  vector<bool> isPrime; // true 表示非素数  false 表示是素数
2  vector<int> prime; // 保存素数
3  int sieve(int n) {
4      isPrime.resize(n + 1, false);
5      isPrime[0] = isPrime[1] = true;
6      for (int i = 2; i <= n; i++) {
7          if (!isPrime[i]) prime.emplace_back(i);
8          for (int j = 0; j < (int)prime.size() && prime[j] * i <= n; ++j) {
9              isPrime[prime[j] * i] = true;
10             if (!(i % prime[j])) break;
11         }
12     }
13     return (int)prime.size();
14 }
```

2.3 欧拉定理

2.4 欧拉函数

2.5 中国剩余定理

2.5.1 CRT

2.5.2 EXCRT

2.6 BSGS

3 线性代数

3.1 高斯-约旦消元法

3.2 高斯消元法-bitset

3.3 线性基

3.4 矩阵树定理

3.5 LGV 引理

4 组合数学

4.1 组合数预处理

4.2 卢卡斯定理

4.3 小球盒子模型

4.4 斯特林数

5 博弈论

6 其他数学

6.1 蔡勒公式

7 图论

7.1 并查集

```
1 struct dsu {  
2     private:  
3         // number of nodes  
4         int n;  
5         // root node: -1 * component size  
6         // otherwise: parent  
7         std::vector<int> pa;
```

```
8 public:
9     dsu(int n_ = 0) : n(n_), pa(n_, -1) {}
10    // find node x's parent
11    int find(int x) {
12        return pa[x] < 0 ? x : pa[x] = find(pa[x]);
13    }
14    // merge node x and node y
15    // if x and y had already in the same component, return false, otherwise
16    // return true
17    // Implement (union by size) + (path compression)
18    bool unite(int x, int y) {
19        int xr = find(x), yr = find(y);
20        if (xr != yr) {
21            if (-pa[xr] < -pa[yr]) std::swap(xr, yr);
22            pa[xr] += pa[yr];
23            pa[yr] = xr; // y -> x
24            return true;
25        }
26        return false;
27    }
28    // size of the connected component that contains the vertex x
29    int size(int x) {
30        return -pa[find(x)];
31    }
32};
```

7.2 最小树形图

7.3 最近公共祖先

7.4 最大流

7.5 全局最小割

7.6 二分图最大权匹配

7.7 一般图最大匹配

7.8 2-sat

7.9 最大团

8 字符串

8.1 KMP

8.2 Manacher

8.3 Trie

8.4 01-Trie

9 计算几何