

字符串

KMP

```
1  int len1, len2, nxt[MAXN], ans[MAXN];
2  char s1[MAXN], s2[MAXN]; //s1 文本串, s2 模式串
3
4  int main()
5  {
6      scanf("%s%s", s1+1, s2+1);
7      len1=strlen(s1+1), len2=strlen(s2+1);
8      for(int i=2, p=0; i<=len2; i++)
9      {
10         while(p && s2[i]!=s2[p+1]) p=nxt[p];
11         if(s2[i]==s2[p+1]) p++;
12         nxt[i]=p;
13     }
14     for(int i=1, p=0; i<=len1; i++)
15     {
16         while(p && s1[i]!=s2[p+1]) p=nxt[p];
17         if(s1[i]==s2[p+1]) p++;
18         // p==len2 时, 存在一个匹配
19     }
20 }
```

AC自动机

```
1  int n, tot, fail[MAXN], t[MAXN][26], num[MAXN];
2  char s2[MAXN], s1[MAXN]; //s1 文本串, s2 模式串
3
4  queue<int> q;
5  vector<int> vec[MAXN];
6
7  void insert(char *s, int id)
8  {
9      int len=strlen(s), p=0;
10     for(int i=0; i<len; i++) {
11         int ch=s[i]-'a';
12         if(!t[p][ch]) t[p][ch]=++tot;
13         p=t[p][ch];
14     }
15     vec[p].push_back(id);
16 }
17
18 void getfail()
19 {
20     for(int i=0; i<26; i++)
```

```

21         if(t[0][i]) fail[t[0][i]]=0, q.push(t[0][i]);
22     while(!q.empty()) {
23         int x=q.front(); q.pop();
24         for(int i=0; i<26; i++) {
25             if(t[x][i]) {
26                 fail[t[x][i]]=t[fail[x]][i];
27                 q.push(t[x][i]);
28             } else {
29                 t[x][i]=t[fail[x]][i];
30             }
31         }
32     }
33 }
34
35 void solve(char *s)
36 {
37     int len=strlen(s), p=0;
38     for(int i=0; i<len; i++) {
39         int ch=s[i]-'a';
40         p=t[p][ch];
41         for(int j=p; j; j=fail[j])
42             for(int e: vec[j]) num[e]++;
43     }
44 }
45
46 int main()
47 {
48     scanf("%d", &n);
49     for(int i=1; i<=n; i++) {
50         scanf("%s", s2);
51         insert(s2, i);
52     }
53     getfail();
54     scanf("%s", s1);
55     solve(s1);
56     //num[i] 为模式串 i 在文本串中出现次数
57     return 0;
58 }

```

后缀自动机

```

1 void insert(int ch)
2 {
3     /* 广义后缀自动机
4     if(a[last].nxt[ch]) {
5         int p=last, x=a[p].nxt[ch];
6         if(a[p].len+1==a[x].len) last=x;
7         else {
8             int y=++tot;

```

```

9         memcpy(a[y].nxt, a[x].nxt, sizeof(a[y].nxt));
10        a[y].len=a[p].len+1; a[y].fa=a[x].fa;
11        for(; p && a[p].nxt[ch]==x; p=a[p].fa) a[p].nxt[ch]=y;
12        a[x].fa=y; last=y;
13    }
14    return;
15 }
16 */
17 int cur=++tot, p=last;
18 a[cur].len=a[last].len+1;
19 a[cur].val=1;
20 for(; p && !a[p].nxt[ch]; p=a[p].fa) a[p].nxt[ch]=cur;
21 if(!p) a[cur].fa=1;
22 else {
23     int x=a[p].nxt[ch];
24     if(a[p].len+1==a[x].len) a[cur].fa=x;
25     else {
26         int y=++tot;
27         memcpy(a[y].nxt, a[x].nxt, sizeof(a[y].nxt));
28         a[y].len=a[p].len+1; a[y].fa=a[x].fa;
29         for(; p && a[p].nxt[ch]==x; p=a[p].fa) a[p].nxt[ch]=y;
30         a[x].fa=a[cur].fa=y;
31     }
32 }
33 last=cur;
34 }

```

线性代数

高斯消元

```
1  int n;
2  double a[MAXN][MAXN];
3
4  int main()
5  {
6      scanf("%d", &n);
7      for(int i=1; i<=n; ++i)
8          for(int j=1; j<=n+1; ++j) scanf("%lf", &a[i][j]);
9      int c=1, r=1;
10     for(; c<=n; ++c) {
11         int pos=0;
12         for(int i=r; i<=n; ++i)
13             if(a[i][c]!=0) pos=i;
14         if(pos==0) continue;
15         for(int i=c; i<=n+1; ++i) swap(a[r][i], a[pos][i]);
16         for(int i=n+1; i>=c; --i) a[r][i]/=a[r][c];
17         for(int i=1; i<=n; ++i) {
18             if(r==i) continue;
19             for(int j=n+1; j>=c; --j) a[i][j]-=a[i][c]*a[r][j];
20         }
21         r++;
22     }
23     if(r<=n) {
24         printf("No Solution\n");
25         return 0;
26     }
27     for(int i=1; i<=n; ++i) printf("%.2lf\n", a[i][n+1]);
28     return 0;
29 }
```

线性基

```
1  int n;
2  LL base[66];
3
4  void insert(LL *base, LL x)
5  {
6      for(int i=60; i>=0; --i)
7          if((x>>i)&1) {
8              if(!base[i]) {
9                  base[i]=x;
10                 return;
11             }
12             x^=base[i];
13         }
```

```

13     }
14 }
15
16 int main()
17 {
18     scanf("%d", &n);
19     for(int i=1; i<=n; ++i) {
20         LL temp;
21         scanf("%lld", &temp);
22         insert(base, temp);
23     }
24     return 0;
25 }

```

矩阵树定理

度数矩阵 $D(G)$ 邻接矩阵 $A(G)$

Kirchhoff 矩阵 $L(G) = D(G) - A(G)$

生成树个数 $t(G) = \det L^{n-1}(G)$

有向图中 $t^{\text{root}}(G, k) = \det L^{\text{out}}(G, k)$

$L^{n-1}(G)$ 为 $L(G)$ 的任意 $n - 1$ 阶主子式

LGV 引理

$\omega(P)$ 表示 P 这条路径上所有边的边权之积。（路径计数时，可以将边权都设为 1）（事实上，边权可以为生成函数）

$e(u, v)$ 表示 u 到 v 的 **每一条** 路径 P 的 $\omega(P)$ 之和，即 $e(u, v) = \sum_{P: u \rightarrow v} \omega(P)$ 。

起点集合 A ，是有向无环图点集的一个子集，大小为 n 。

终点集合 B ，也是有向无环图点集的一个子集，大小也为 n 。

一组 $A \rightarrow B$ 的不相交路径 S : S_i 是一条从 A_i 到 $B_{\sigma(S)_i}$ 的路径 ($\sigma(S)$ 是一个排列)，对于任何 $i \neq j$, S_i 和 S_j 没有公共顶点。

$N(\sigma)$ 表示排列 σ 的逆序对个数。

$$M = \begin{bmatrix} e(A_1, B_1) & e(A_1, B_2) & \cdots & e(A_1, B_n) \\ e(A_2, B_1) & e(A_2, B_2) & \cdots & e(A_2, B_n) \\ \vdots & \vdots & \ddots & \vdots \\ e(A_n, B_1) & e(A_n, B_2) & \cdots & e(A_n, B_n) \end{bmatrix}$$

$$\det(M) = \sum_{S:A \rightarrow B} (-1)^{N(\sigma(S))} \prod_{i=1}^n \omega(S_i)$$

BEST 定理

$ec(G)$ 不同欧拉回路总数 欧拉图中 $t^{root}(G) = t^{leaf}(G)$

$$ec(G) = t^{root}(G, k) \prod_{v \in V} (deg(v) - 1)!$$

组合数学

拉格朗日插值

```
1 // 给定 x 出点值, 求多项式 k 处点值 (k>=0 && k<P)
2 // x 连续时使用 interp2
3 int T, n, k, ifac[MAXN];
4
5 int interp1(vi x, vi y, int k)
6 {
7     int n=x.size(), ans=0;
8     for(int i=0; i<n; ++i) {
9         int up=y[i], down=1;
10        for(int j=0; j<n; ++j) {
11            if(i==j) continue;
12            up=1LL*up*sub(k, x[j])%P;
13            down=1LL*down*sub(x[i], x[j])%P;
14        }
15        ans=(ans+1LL*up*qpow(down))%P;
16    }
17    return ans;
18 }
19
20 int interp2(vi y, int k)
21 {
22     int n=y.size(), ans=0;
23     vi pre(n), suf(n);
24     pre[0]=suf[n-1]=1;
25     for(int i=0; i<n-1; ++i) pre[i+1]=1LL*pre[i]*sub(k, i)%P;
26     for(int i=n-1; i>=1; --i) suf[i-1]=1LL*suf[i]*sub(k, i)%P;
27     for(int i=0; i<n; ++i) {
28         int up=1LL*y[i]*pre[i]%P*suf[i]%P;
29         int down=1LL*ifac[i]*((n-i)&1?ifac[n-i-1]:P-ifac[n-i-1])%P;
30         ans=(ans+1LL*up*down)%P;
31     }
32     return ans;
33 }
34
35 int main()
36 {
37     init(2001);
38     scanf("%d", &T);
39     while(T--) {
40         scanf("%d%d", &n, &k);
41         vi x(n+1), y(n+1);
42         for(int i=0; i<=n; ++i) scanf("%d%d", &x[i], &y[i]);
43         printf("%d\n", interp1(x, y, k));
44     }
```

```

45     return 0;
46 }

```

快速傅里叶变换

```

1  void fft(vc<C> &a) {
2      int n = sz(a), L = 31 - __builtin_clz(n);
3      static vc<C> rt(2, 1);
4      static vc<complex<long double>> R(2, 1);
5      for (static int k = 2; k < n; k *= 2) {
6          R.resize(n), rt.resize(n);
7          auto x = polar(1.L, acos(-1.L) / k);
8          for (int i = k; i < k * 2; ++i)
9              rt[i] = R[i] = i & 1 ? R[i / 2] * x : R[i / 2];
10     }
11     vc<int> rev(n);
12     for (int i = 0; i < n; ++i) rev[i] = (rev[i / 2] | (i & 1) << L) / 2;
13     for (int i = 0; i < n; ++i) if (i < rev[i]) swap(a[i], a[rev[i]]);
14     for (int k = 1; k < n; k *= 2)
15         for (int i = 0; i < n; i += k * 2) {
16             auto it1 = &a[i], it2 = it1 + k;
17             for (int j = 0; j < k; ++j, ++it1, ++it2) {
18                 auto x = (double *)&rt[j + k], y = (double *)it2;
19                 C z(x[0] * y[0] - x[1] * y[1], x[0] * y[1] + x[1] * y[0]);
20                 *it2 = *it1 - z;
21                 *it1 += z;
22             }
23         }
24 }
25
26 vd conv(vd &a, vd&b) {
27     if (a.empty() || b.empty()) return vd();
28     vd res(sz(a) + sz(b) - 1);
29     int L = 32 - __builtin_clz(sz(res) - 1), n = 1 << L;
30     vc<C> in(n), out(n);
31     copy(all(a), begin(in));
32     for (int i = 0; i < sz(b); ++i) in[i].imag(b[i]);
33     fft(in);
34     for (C &x: in) x *= x;
35     for (int i = 0; i < n; ++i) out[i] = in[-i & (n - 1)] - conj(in[i]);
36     fft(out);
37     for (int i = 0; i < sz(res); ++i) res[i] = imag(out[i]) / (n * 4);
38     return res;
39 }
40
41 int n, m;
42
43 int main()
44 {

```



```

45     scanf("%d%d", &n, &m);
46     vd a(n+1), b(m+1);
47     for(int i=0; i<=n; ++i) scanf("%lf", &a[i]);
48     for(int i=0; i<=m; ++i) scanf("%lf", &b[i]);
49     a=conv(a, b);
50     for(double e: a) printf("%d ", int(e+0.5));
51 }

```

多项式运算

```

1  template<class T> using vc = vector<T>;
2  using ll = int64_t;
3  constexpr ll md = 998244353, root = 62, LIM = 1 << 18;
4  struct Mod {
5      ll x;
6      Mod(ll x = 0): x(x) {}
7      Mod operator+(Mod b) {ll y=x+b.x;return y<md ? y : y - md; }
8      Mod operator-(Mod b) { return x - b.x + (x < b.x ? md : 0); }
9      Mod operator*(Mod b) { return x * b.x % md; }
10     void operator += (Mod b) { x += b.x; x < md ? : x -= md; }
11     void operator *= (Mod b) { (x *= b.x) %= md; }
12     void operator -= (Mod b) { x -= b.x; -x < 0 ? : x += md; }
13 };
14 Mod qpow(Mod b, ll e) {
15     Mod res = 1;
16     for (; e; b *= b, e /= 2)
17         if (e & 1) res *= b;
18     return res;
19 }
20
21 vc<Mod> inv(LIM), fac(LIM), ifac(LIM);
22
23 void init()
24 {
25     inv[0]=inv[1]=ifac[0]=fac[0]=1;
26     for(int i=2; i<LIM; ++i) inv[i]=inv[md%i]*(md-md/i);
27     for(int i=1; i<LIM; ++i) {
28         ifac[i]=ifac[i-1]*inv[i];
29         fac[i]=fac[i-1]*i;
30     }
31 }
32
33 void ntt(vc<Mod> &a) {
34     int n = sz(a), L = 31 - __builtin_clz(n);
35     static vc<Mod> rt(2, 1);
36     for (static int k = 2, s = 2; k < n; k *= 2, ++s) {
37         rt.resize(n);
38         array<Mod, 2> z{1, qpow(root, md >> s)};
39         for (int i = k; i < k * 2; ++i)

```

```

40         rt[i] = rt[i / 2] * z[i & 1];
41     }
42     vc<int> rev(n);
43     for (int i = 0; i < n; ++i)
44         rev[i] = (rev[i / 2] | (i & 1) << L) / 2;
45     for (int i = 0; i < n; ++i)
46         if (i < rev[i]) swap(a[i], a[rev[i]]);
47     for (int k = 1; k < n; k *= 2)
48         for (int i = 0; i < n; i += k * 2) {
49             auto it1 = &a[i], it2 = it1 + k;
50             for (int j = 0; j < k; ++j, ++it1, ++it2) {
51                 Mod z = rt[j + k] * *it2;
52                 *it2 = *it1 - z, *it1 += z;
53             }
54         }
55 }
56
57 vc<Mod> conv(vc<Mod> a, vc<Mod> b) {
58     if (a.empty() || b.empty()) return {};
59     int s = sz(a) + sz(b) - 1, n = 1 << (32 - __builtin_clz(s - 1));
60     Mod iv = md - (md - 1) / n;
61     vc<Mod> out(n);
62     a.resize(n), b.resize(n);
63     ntt(a), ntt(b);
64     for (int i = 0; i < n; ++i)
65         out[-i & (n - 1)] = a[i] * b[i] * iv;
66     ntt(out);
67     return {out.begin(), out.begin() + s};
68 }
69
70 void invIter (vc<Mod> &a, vc<Mod> &in, vc<Mod> &b) {
71     int n = sz(in);
72     vc<Mod> out(n);
73     copy(a.begin(), a.begin() + min(sz(a), n), out.begin());
74     auto conv = [&] {
75         ntt(out);
76         for (int i = 0; i < n; ++i) out[i] *= in[i];
77         ntt(out), reverse(out.begin() + 1, out.end());
78     };
79     conv(), fill(out.begin(), out.begin() + sz(b), 0), conv();
80     b.resize(n);
81     Mod iv = md - (md - 1) / n; iv *= iv;
82     for (int i = n / 2; i < n; ++i)
83         b[i] = out[i].x ? iv * (md - out[i].x) : 0;
84 }
85
86 vc<Mod> polyInv (vc<Mod> a) {
87     if (a.empty()) return {};
88     vc<Mod> b{qpow(a[0], md - 2)};

```

```

89     b.reserve(sz(a));
90     while (sz(b) < sz(a)) {
91         vc<Mod> in(sz(b) * 2);
92         copy(all(b), in.begin()), ntt(in);
93         invIter(a, in, b);
94     }
95     return {b.begin(), b.begin() + sz(a)};
96 }
97
98 vc<Mod> polyMod (vc<Mod> a, vc<Mod> b) {
99     if (sz(a) < sz(b)) return a;
100     int n = sz(a) - sz(b) + 1;
101     vc<Mod> da(a.rbegin(), a.rend()), db(b.rbegin(), b.rend());
102     da.resize(n), db.resize(n);
103     da = conv(da, polyInv(db));
104     da.resize(n), reverse(all(da));
105     auto c = conv(da, b);
106     a.resize(sz(b) - 1);
107     for (int i = 0; i < sz(a); ++i) a[i] -= c[i];
108     return a;
109 }
110
111 vc<Mod> deri (vc<Mod> a) {
112     for (int i = 1; i < sz(a); ++i) a[i - 1] = a[i] * i;
113     a.pop_back();
114     return a;
115 }
116
117 // initialize array inv
118 vc<Mod> inte (vc<Mod> a) {
119     for (int i = sz(a) - 1; i >= 1; --i) a[i] = a[i - 1] * inv[i];
120     a[0] = 0;
121     return a;
122 }
123
124 vc<Mod> polyLn (vc<Mod> &a) {
125     if (a.empty()) return {};
126     int n = 1 << (32 - __builtin_clz(2 * sz(a) - 2));
127     Mod iv = md - (md - 1) / n;
128     vc<Mod> b = polyInv(a), c = deri(a);
129     b.resize(n), c.resize(n);
130     ntt(b), ntt(c);
131     for (int i = 0; i < n; ++i) b[i] = b[i] * c[i] * iv;
132     ntt(b), reverse(b.begin() + 1, b.end());
133     b = inte(b);
134     return {b.begin(), b.begin() + sz(a)};
135 }
136
137 vc<Mod> polyExp (vc<Mod> &a) {

```

```

138     if (a.empty()) return {};
139     vc<Mod> b{1}, ib{1};
140     b.reserve(sz(a)), ib.reserve(sz(a));
141     auto conv = [&](vc<Mod> &a, vc<Mod> &b) {
142         ntt(a);
143         for (int i = 0; i < sz(a); ++i) a[i] *= b[i];
144         ntt(a), reverse(a.begin() + 1, a.end());
145     };
146     while (sz(b) < sz(a)) {
147         int h = sz(b), n = h * 2;
148         Mod iv = md - (md - 1) / n;
149         vc<Mod> db(n), dib(n), A(deri(b)), B(n);
150         copy(all(ib), dib.begin(), ntt(dib));
151         copy(all(b), db.begin(), ntt(db));
152         A.resize(n), conv(A, dib);
153         for (int i = 0; i < n; ++i) B[i] = db[i] * dib[i];
154         ntt(B), reverse(B.begin() + 1, B.end());
155         fill(B.begin(), B.begin() + h, 0);
156         vc<Mod> da(deri(vc<Mod>(a.begin(), a.begin() + h)));
157         da.resize(n), ntt(da), conv(B, da);
158         for (int i = min(n, sz(a)) - 1; i >= h; --i)
159             A[i] = (A[i - 1] - B[i - 1] * iv) * inv[i] * iv - a[i];
160         fill(A.begin(), A.begin() + h, 0), conv(A, db);
161         b.resize(n);
162         for (int i = h; i < n; ++i)
163             b[i] = A[i].x ? iv * (md - A[i].x) : 0;
164         if (sz(b) < sz(a)) invIter(b, dib, ib);
165     }
166     return {b.begin(), b.begin() + sz(a)};
167 }
168
169 vc<Mod> polyPow (vc<Mod> &a, ll k) {
170     vc<Mod> b = polyLn(a);
171     for (Mod &e: b) e *= k;
172     return polyExp(b);
173 }

```

分治 FFT

```

1 // f[0]=1 且满足递推关系 f[n]=\sum_i^n g[i]*f[n-i]
2 int n;
3 vc<Mod> f, g;
4
5 void divide(int l, int r)
6 {
7     if(l==r) return;
8     int mid=(l+r)>>1;
9     divide(l, mid);
10    vc<Mod> a{f.begin()+l, f.begin()+mid+1};

```

```

11     vc<Mod> b{g.begin(), g.begin()+r-l+1};
12     a=conv(a, b);
13     for(int i=mid+1; i<=r; ++i) f[i]+=a[i-1];
14     divide(mid+1, r);
15 }

```

分治乘法

```

1  // 求 vec 内多所有项式的卷积
2  vc<poly> vec;
3
4  poly solve(int l, int r)
5  {
6      if(l==r) return vec[l];
7      /* 找重心分治
8      int mid=l, sum1=0, sum2=sz(vec[mid]);
9      for(int i=l; i<=r; ++i) sum1+=sz(vec[i]);
10     while(mid+1<r && sum2+sz(vec[mid+1])<=sum1/2) sum2+=sz(vec[++mid]);
11     */
12     int mid=(l+r)/2;
13     return conv(solve(l, mid), solve(mid+1, r));
14 }
15
16 int main()
17 {
18     // 读入个多项式
19     // 中点分治需随机打乱
20     vc<Mod> f = solve(0, vec.size()-1);
21 }

```

线性递推

```

1  // f[0]=P-1 且有递推关系 a[n]=\sum_i^k f[i]*a[n-i]
2  // 求 a[m]
3  int recurrence(vc<Mod> f, vc<Mod> a, ll m)
4  {
5      int k=a.size(), n=1<<(32-__builtin_clz(2*k-2));
6      vc<Mod> g=polyInv(vc<Mod>(f.begin(), f.begin()+k-1));
7      reverse(all(f));
8      g.resize(n), f.resize(n);
9      ntt(g), ntt(f);
10     auto combine = [&](vc<Mod> a, vc<Mod> b) -> vc<Mod> {
11         Mod iv = md - (md - 1) / n;
12         vc<Mod> c(n), d(n);
13         a=conv(a, b);
14         copy(a.rbegin(), a.rbegin()+k-1, c.begin());
15         ntt(c);
16         for (int i=0; i<n; ++i) d[-i&(n-1)]=c[i]*g[i]*iv;

```

```

17     ntt(d);
18     copy(d.rend()-k+1, d.rend(), c.begin());
19     fill(c.begin()+k-1, c.end(), 0);
20     ntt(c);
21     for (int i=0; i<n; ++i) d[-i&(n-1)]=c[i]*f[i]*iv;
22     ntt(d);
23     for(int i=0; i<k; ++i) a[i]-=d[i];
24     return {a.begin(), a.begin()+k};
25 };
26 vc<Mod> b(k), c(k);
27 b[0]=1, c[1]=1;
28 for(; m; m>>=1) {
29     if(m&1) b=combine(b, c);
30     c=combine(c, c);
31 }
32 Mod ans;
33 for(int i=0; i<k; ++i) ans+=a[i]*b[i];
34 return ans.x;
35 }

```

单位根反演

$$\omega_a^b = g^{(P-1) \cdot b/a}$$

$$[n|a] = \frac{1}{n} \sum_{k=0}^{n-1} \omega_n^{ak}$$

$$[a \equiv b \pmod n] = \frac{1}{n} \sum_{k=0}^{n-1} \omega_n^{ak} \omega_n^{-bk}$$

第 K 大反演

$$k^{th}max(s) = \sum_{T \subseteq S} (-1)^{|T|-k} \binom{|T|-1}{k-1} min(T)$$

Burnside 引理

l 等价类个数

G 置换群

$c(p)$ 置换 p 中不动点个数

$$l = \frac{1}{|G|} \sum_{p \in G} c(p)$$

数论

线性筛

```
1  int n, t, cnt, pri[MAXN], vis[MAXN];
2  int mu[MAXN], phi[MAXN]; //莫比乌斯函数和欧拉函数
3
4  void sieve(int lim)
5  {
6      //phi[i]=mu[i]=1;
7      for(int i=2; i<=lim; ++i) {
8          if(!vis[i]) pri[++cnt]=i; // phi[i]=i-1, mu[i]=-1;
9          for(int j=1; j<=cnt && pri[j]*i<=lim; ++j) {
10             vis[i*pri[j]]=1;
11             if(i%pri[j]==0) break;
12             /*
13             if(i%pri[j]==0) {
14                 mu[pri[j]*i]=0;
15                 phi[i*pri[j]]=pri[j]*phi[i];
16                 break;
17             } else {
18                 mu[pri[j]*i]=-mu[i];
19                 phi[i*pri[j]]=(pri[j]-1)*phi[i];
20             }
21             */
22         }
23     }
24 }
```

质因数分解

```
1  const LL base[]={2, 325, 9375, 28178, 450775, 9780504, 1795265022};
2
3  int T;
4  LL n;
5  map<LL, int> mp;
6
7  LL qpow(LL x, LL y, LL p)
8  {
9      LL num=1;
10     while(y) {
11         if(y&1) num=(I128)num*x%p;
12         y>>=1, x=(I128)x*x%p;
13     }
14     return num;
15 }
16
17 bool MR(LL x)
```

```

18 {
19     if(x<=2 || !(x%2)) return x==2;
20     LL d=x-1, r=0;
21     while(!(d%2)) d>>=1, r++;
22     for(LL e: base) {
23         LL v=qpow(e, d, x);
24         if(v<=1 || v==x-1) continue;
25         for(int i=0; i<r-1; ++i) {
26             v=(1128)v*v%x;
27             if(v==x-1 || v==1) break;
28         }
29         if(v!=x-1) return false;
30     }
31     return true;
32 }
33
34 LL PR(LL x)
35 {
36     LL l=0, r=0, val=2, tmp;
37     auto f=[x](LL y) {return ((1128)y*y+1)%x;};
38     for(int i=0; ; ++i) {
39         if(!(i%60) && __gcd(val, x)>1) break;
40         if(l==r) l=rand()%(x-1)+1, r=f(l);
41         if(tmp=(1128)val*abs(r-l)%x) val=tmp;
42         l=f(l), r=f(f(r));
43     }
44     return __gcd(val, x);
45 }
46
47 void find(LL x, int num)
48 {
49     if(x<=1) return;
50     if(MR(x)) {
51         mp[x]+=num;
52         return;
53     }
54     LL y=x, cnt=0;
55     while(y==x) y=PR(x);
56     while(x%y==0) x/=y, cnt++;
57     find(x, num), find(y, cnt*num);
58 }
59
60 int main()
61 {
62     srand(time(0));
63     scanf("%d", &T);
64     while(T--) {
65         mp.clear();
66         scanf("%lld", &n);

```



```

67         find(n, 1); //指数形式分解质因数
68         for(auto e: mp) printf("%lld %d\n", e.first, e.second);
69     }
70     return 0;
71 }

```

BSGS

```

1  int a, b, x, p;
2
3  unordered_map<int, int> mp;
4
5  int qpow(int x, int y)
6  {
7      int num=1;
8      while(y) {
9          if(y&1) num=1LL*num*x%p;
10         x=1LL*x*x%p; y>>=1;
11     }
12     return num;
13 }
14
15 int bsgs(int a, int b)
16 {
17     if(a%p==0) return (b==0)?1:-1;
18     mp.clear();
19     int siz=(int)(sqrt(p)+0.5), tmp=1LL*a*b%p;
20     for(int i=1; i<=siz; ++i) {
21         mp[tmp]=i;
22         tmp=1LL*tmp*a%p;
23     }
24     int base=qpow(a, siz);
25     tmp=1;
26     for(int i=1; i<=siz; ++i) {
27         tmp=1LL*tmp*base%p;
28         if(mp[tmp]) return i*siz-mp[tmp];
29     }
30     return -1;
31 }
32
33 int main()
34 {
35     scanf("%d%d%d", &a, &b, &p);
36     x=bsgs(a, b); //a^x = b mod p
37     if(x==-1) printf("No solution\n");
38     else printf("%d\n", x);
39 }

```

```

1  int T, n, tot, vis[MAXN], pri[MAXN];
2  LL mu[MAXN], phi[MAXN];
3
4  unordered_map<int, LL> muf, phif;
5
6  void init()
7  {
8      mu[1]=phi[1]=1;
9      for(int i=2; i<=5e6; ++i)
10     {
11         if(!vis[i]) pri[++tot]=i, phi[i]=i-1, mu[i]=-1;
12         for(int j=1; j<=tot && pri[j]*i<=5e6; ++j)
13         {
14             vis[pri[j]*i]=1;
15             if(i%pri[j]==0)
16             {
17                 phi[i*pri[j]]=phi[i]*pri[j];
18                 mu[i*pri[j]]=0;
19                 break;
20             }
21             phi[i*pri[j]]=phi[i]*(pri[j]-1);
22             mu[i*pri[j]]=-mu[i];
23         }
24     }
25     for(int i=1; i<=5e6; ++i) phi[i]+=phi[i-1], mu[i]+=mu[i-1];
26 }
27
28 LL mus(int x)
29 {
30     if(x<=5e6) return mu[x];
31     if(muf[x]) return muf[x];
32     LL sum=1;
33     for(int l=2, r; l<=x; l=r+1)
34     {
35         r=x/(x/l);
36         sum-=1LL*(r-l+1)*mus(x/l);
37     }
38     muf[x]=sum;
39     return sum;
40 }
41
42 LL phis(int x)
43 {
44     if(x<=5e6) return phi[x];
45     if(phif[x]) return phif[x];
46     LL sum=1LL*x*(x+1)/2;
47     for(int l=2, r; l<=x; l=r+1)
48     {
49         r=x/(x/l);

```

```

50     sum-=1LL*(r-l+1)*phis(x/l);
51 }
52 phif[x]=sum;
53 return sum;
54 }
55
56 int main()
57 {
58     init();
59     scanf("%d", &T);
60     while(T--)
61     {
62         scanf("%d", &n);
63         printf("%lld %lld\n", phis(n), mus(n));
64     }
65     return 0;
66 }

```

Min25 筛

积性函数 $f(x)$, $f(p^k) = p^k(p^k - 1)$, 求 $\sum_i^n f(i)$

```

1  int LIM, INV2=5000000004, INV6=1666666668;
2  int cnt1, cnt2, vis[MAXN], pri[MAXN], g1[MAXN], g2[MAXN], h1[MAXN], h2[MAXN],
   id1[MAXN], id2[MAXN];
3  ll n, w[MAXN];
4
5  void sieve(int lim)
6  {
7      for(int i=2; i<=lim; ++i) {
8          if(!vis[i]) {
9              pri[++cnt1]=i;
10             g1[cnt1]=(g1[cnt1-1]+i)%P;
11             h1[cnt1]=(h1[cnt1-1]+1LL*i*i)%P;
12         }
13         for(int j=1; j<=cnt1 && pri[j]*i<=LIM; ++j) {
14             vis[i*pri[j]]=1;
15             if(i%pri[j]==0) break;
16         }
17     }
18 }
19
20 int cal(ll x, int y)
21 {
22     if(pri[y]>=x) return 0;
23     int id=(x<=LIM?id1[x]:id2[n/x]);
24     int val=sub(sub(h2[id], g2[id]), sub(h1[y], g1[y]));
25     for(int i=y+1; i<=cnt1; ++i) {
26         ll tmp=pri[i];
27         if(tmp*tmp>x) break;

```

```

28     for(int k=1; tmp<=x; k++, tmp*=pri[i]) {
29         ll tmp1=tmp%P; tmp1=tmp1*(tmp1-1)%P;
30         val=(val+tmp1*(cal(x/tmp, i)+(k>1)))%P;
31     }
32 }
33 return val;
34 }
35
36 int main()
37 {
38     scanf("%lld", &n);
39     LIM=sqrt(n);
40     sieve(LIM);
41     for(ll l=1, r; l<=n; l=r+1) {
42         r=n/(n/l); w[++cnt2]=n/l;
43         ll tmp=(n/l)%P;
44         g2[cnt2]=sub(tmp*(tmp+1)%P*INV2%P, 1);
45         h2[cnt2]=sub(tmp*(tmp+1)%P*(2*tmp+1)%P*INV6%P, 1);
46         if(n/r<=LIM) id1[n/r]=cnt2;
47         else id2[r]=cnt2;
48     }
49     for(int i=1; i<=cnt1; ++i) {
50         ll tmp=pri[i];
51         for(int j=1; j<=cnt2 && tmp*tmp<=w[j]; ++j) {
52             ll id=w[j]/pri[i]; id=(id<=LIM?id1[id]:id2[n/id]);
53             g2[j]=sub(g2[j], tmp*sub(g2[id], g1[i-1])%P);
54             h2[j]=sub(h2[j], tmp*tmp%P*sub(h2[id], h1[i-1])%P);
55         }
56     }
57     printf("%d\n", add(cal(n, 0), 1));
58 }

```

数据结构

树状数组

```
1  int n, t[MAXN];
2
3  void add(int x, int y) {for(; x<=n; x+=(x&-x)) t[x]+=y;} //add(x, y) 位置 x 加 y
4
5  int sum(int x) {int y=0; for(; x; x--=(x&-x)) y+=t[x]; return y;} //sum(x) 1~x 区间和
```

线段树

```
1  int n, m, val[MAXN*4], tag[MAXN*4];
2
3  void up(int root)
4  {
5      val[root]=val[ls]+val[rs];
6  }
7
8  void down(int root, int l, int r)
9  {
10     if(!tag[root]) return;
11     tag[ls]+=tag[root];
12     tag[rs]+=tag[root];
13     val[ls]+=tag[root]*(mid-l+1);
14     val[rs]+=tag[root]*(r-mid);
15     tag[root]=0;
16 }
17
18 void build(int root, int l, int r)
19 {
20     if(l==r)
21     {
22         scanf("%d", &val[root]);
23         return ;
24     }
25     build(ls, l, mid);
26     build(rs, mid+1, r);
27     up(root);
28 }
29
30 void add(int root, int l, int r, int x, int y, int k)
31 {
32     if(x>r || y<l) return;
33     if(l>=x && r<=y)
34     {
35         val[root]+=k*(r-l+1);
36         tag[root]+=k;
```

```

37     return;
38 }
39 down(root, l, r);
40 add(ls, l, mid, x, y, k);
41 add(rs, mid+1, r, x, y, k);
42 up(root);
43 }
44
45 int query(int root, int l, int r, int x, int y)
46 {
47     if(l>y || r<x) return 0;
48     if(l>=x && r<=y) return val[root];
49     down(root, l, r);
50     return query(ls, l, mid, x, y)+query(rs, mid+1, r, x, y);
51 }

```

主席树

```

1  int n, m, tot, a[MAXN], root[MAXN];
2
3  struct Node {int ls, rs, val;} t[MAXN*40];
4
5  void update(int &rt1, int rt2, int l, int r, int x)
6  {
7      rt1=++tot;
8      t[rt1]=t[rt2], t[rt1].val++;
9      if(l==r) return;
10     if(x<=mid) update(t[rt1].ls, t[rt2].ls, l, mid, x);
11     else update(t[rt1].rs, t[rt2].rs, mid+1, r, x);
12 }
13
14 int query(int rt1, int rt2, int l, int r, int k)
15 {
16     if(l==r) return l;
17     int temp=t[t[rt2].ls].val-t[t[rt1].ls].val;
18     if(temp>=k) return query(t[rt1].ls, t[rt2].ls, l, mid, k);
19     else return query(t[rt1].rs, t[rt2].rs, mid+1, r, k-temp);
20 }
21
22 int main()
23 {
24     scanf("%d%d", &n, &m);
25     for(int i=1; i<=n; ++i)
26     {
27         scanf("%d", &a[i]);
28         update(root[i], root[i-1], 1, n, a[i]);
29     }
30     for(int i=1; i<=m; i++)
31     {

```

```

32         //query(root[x-1], root[y], 1, n, k) 区间 x~y 第k大
33     }
34 }

```

点分治

```

1  int n, m, tot, rt, f[MAXN], siz[MAXN], k[MAXN], ans[MAXN], vis[MAXN];
2
3  vector<pair<int,int>> g[MAXN];
4  vector<int> vec;
5
6  void find(int x, int fa)
7  {
8      siz[x]=1; f[x]=0;
9      for(auto [to, d]: g[x]) {
10         if(to==fa || vis[to]) continue;
11         find(to, x);
12         siz[x]+=siz[to];
13         f[x]=max(f[x], siz[to]);
14     }
15     f[x]=max(f[x], tot-siz[x]);
16     if(!rt || f[x]<f[rt]) rt=x;
17 }
18
19 void dfs(int x, int fa, int dis)
20 {
21     vec.push_back(dis);
22     for(auto [to, d]: g[x]) {
23         if(to==fa || vis[to]) continue;
24         dfs(to, x, dis+d);
25     }
26 }
27
28 void divide(int x)
29 {
30     vis[x]=1;
31     set<int> st; st.insert(0);
32     for(auto [to, d]: g[x]) {
33         if(vis[to]) continue;
34         vec.clear();
35         dfs(to, x, d);
36         for(int i=1; i<=m; ++i)
37             for(int e: vec)
38                 if(st.count(k[i]-e)) ans[i]=1;
39         for(int e: vec) st.insert(e);
40     }
41     for(auto [to, d]: g[x]) {
42         if(vis[to]) continue;
43         tot=siz[to], rt=0;

```

```

44     find(to, 0);
45     divide(rt);
46 }
47 }
48
49 int main()
50 {
51     scanf("%d%d", &n, &m);
52     for(int i=1; i<n; ++i) {
53         int x, y, z;
54         scanf("%d%d%d", &x, &y, &z);
55         g[x].push_back({y, z});
56         g[y].push_back({x, z});
57     }
58     for(int i=1; i<=m; ++i) scanf("%d", &k[i]);
59     tot=n; rt=0;
60     find(1, 0);
61     divide(rt);
62     for(int i=1; i<=m; ++i) {
63         if(ans[i]) printf("AYE\n");
64         else printf("NAY\n");
65     }
66 }

```

莫队

```

1  int n, q, ans[MAXN], bol[MAXN];
2
3  struct Q {int l, r, id;} a[MAXN];
4
5  bool CMP(Q x, Q y)
6  {
7      if(bol[x.l]==bol[y.l]) {
8          if(bol[x.l]&1) return x.r<y.r;
9          else return x.r>y.r;
10     }
11     return x.l<y.l;
12 }
13
14 int main()
15 {
16     scanf("%d", &n);
17     int l=1, r=0, siz=sqrt(n);
18     for(int i=1; i<=n; ++i) bol[i]=(i-1)/siz+1;
19     scanf("%d", &q);
20     for(int i=1; i<=q; ++i) {
21         scanf("%d%d", &a[i].l, &a[i].r);
22         a[i].id=i;
23     }

```



```

24     sort(a+1, a+q+1, CMP);
25     for(int i=1; i<=q; ++i) {
26         while(r<a[i].r) {
27             r++;
28             //update(r, 1);
29         }
30         while(r>a[i].r) {
31             //update(r, -1);
32             r--;
33         }
34         while(l<a[i].l) {
35             //update(l, -1);
36             l++;
37         }
38         while(l>a[i].l) {
39             l--;
40             //update(l, 1);
41         }
42         //ans[a[i].id]=query();
43     }
44     for(int i=1; i<=q; ++i) printf("%d\n", ans[i]);
45     return 0;
46 }

```

图论

最短路

```
1  int n, m, S, dis[MAXN], vis[MAXN];
2
3  struct Node
4  {
5      int id, dis;
6      bool friend operator < (Node x, Node y)
7      {
8          return x.dis>y.dis;
9      }
10 };
11
12 vector<int> g1[MAXN], g2[MAXN];
13 priority_queue<Node> q;
14
15 void dijkstra()
16 {
17     memset(dis, 0x3f, sizeof(dis));
18     dis[S]=0;
19     q.push(Node{S, 0});
20     while(!q.empty())
21     {
22         int x=q.top().id; q.pop();
23         if(vis[x]) continue;
24         vis[x]=1;
25         for(int i=0; i<g1[x].size(); ++i)
26         {
27             int to=g1[x][i];
28             if(dis[to]>dis[x]+g2[x][i])
29             {
30                 dis[to]=dis[x]+g2[x][i];
31                 q.push(Node{to, dis[to]});
32             }
33         }
34     }
35 }
36
37 int main()
38 {
39     scanf("%d%d%d", &n, &m, &S);
40     for(int i=1; i<=m; ++i)
41     {
42         int x, y, z;
43         scanf("%d%d%d", &x, &y, &z);
44         g1[x].pb(y), g2[x].pb(z);
```

```

45     }
46     dijkstra();
47     //dis[i] 为 s 到 i 的最短路距离
48     return 0;
49 }

```

最小生成树

```

1  int n, m, ans, f[MAXN];
2
3  struct Edge {int x, y, dis;} edge[MAXM];
4
5  bool CMP(Edge x, Edge y)
6  {
7      return x.dis<y.dis;
8  }
9
10 int find(int x)
11 {
12     if(f[x]!=x) f[x]=find(f[x]);
13     return f[x];
14 }
15
16 void kruskal()
17 {
18     for(int i=1; i<=n; ++i) f[i]=i;
19     sort(edge+1, edge+m+1, CMP);
20     for(int i=1; i<=m; ++i)
21     {
22         int fx=find(edge[i].x), fy=find(edge[i].y);
23         if(fx!=fy)
24         {
25             ans+=edge[i].dis;
26             f[fx]=fy;
27         }
28     }
29 }
30
31 int main()
32 {
33     scanf("%d%d", &n, &m); //n 个点 m 条边
34     for(int i=1; i<=m; ++i)
35         scanf("%d%d%d", &edge[i].x, &edge[i].y, &edge[i].dis);
36     kruskal();
37     //ans 为最小生成树边权之和
38     return 0;
39 }

```

最近公共祖先

```

1  int n, m, s, dep[MAXN], f[MAXN][20];
2
3  vector<int> g[MAXN];
4
5  void dfs(int x, int fa)
6  {
7      f[x][0]=fa, dep[x]=dep[fa]+1;
8      for(int i=0; i<g[x].size(); ++i)
9      {
10         int to=g[x][i];
11         if(to==fa) continue;
12         dep[to]=dep[x]+1;
13         dfs(to, x);
14     }
15 }
16
17 int lca(int x, int y)
18 {
19     if(dep[x]>dep[y]) swap(x, y);
20     for(int i=19; i>=0; --i)
21         if(dep[f[y][i]]>=dep[x]) y=f[y][i];
22     if(x==y) return x;
23     for(int i=19; i>=0; --i)
24         if(f[x][i]!=f[y][i]) x=f[x][i], y=f[y][i];
25     return f[x][0];
26 }
27
28 int main()
29 {
30     scanf("%d%d", &n, &m);
31     for(int i=1; i<n; ++i)
32     {
33         int x, y;
34         scanf("%d%d", &x, &y);
35         g[x].push_back(y);
36         g[y].push_back(x);
37     }
38     dfs(1, 0);
39     for(int i=1; i<=19; ++i)
40         for(int j=1; j<=n; ++j) f[j][i]=f[f[j][i-1]][i-1];
41     //lca(x,y) 为 x,y 的最近公共祖先
42     return 0;
43 }

```

缩点

```

1  int n, m, cnt, ans, tot, dfn[MAXN], low[MAXN], ins[MAXN], id[MAXN];
2

```

```

3  queue<int> q;
4  stack<int> sta;
5  vector<int> g1[MAXN], g2[MAXN];
6
7  void tarjan(int x)
8  {
9      dfn[x]=low[x]=++cnt;
10     sta.push(x);
11     ins[x]=1;
12     for(int i=0; i<g1[x].size(); ++i)
13     {
14         int to=g1[x][i];
15         if(!dfn[to])
16         {
17             tarjan(to);
18             low[x]=min(low[x], low[to]);
19         }
20         else if(ins[to])
21             low[x]=min(low[x], dfn[to]);
22     }
23     if(low[x]==dfn[x])
24     {
25         tot++;
26         while(!sta.empty() && dfn[sta.top()]>=dfn[x])
27         {
28             int top=sta.top(); sta.pop();
29             ins[top]=0;
30             id[top]=tot;
31         }
32     }
33 }
34
35 int main()
36 {
37     scanf("%d%d", &n, &m);
38     for(int i=1; i<=m; ++i)
39     {
40         int x, y;
41         scanf("%d%d", &x, &y);
42         g1[x].push_back(y);
43     }
44     for(int i=1; i<=n; ++i)
45         if(!dfn[i]) tarjan(i);
46     //id[x] 为 x 所在强连通分量的编号
47     return 0;
48 }

```

割点

```

1  int n, m, cnt, ans, dfn[MAXN], low[MAXN], val[MAXN];
2  vector<int> g[MAXN];
3
4  void tarjan(int x)
5  {
6      dfn[x]=low[x]=++cnt;
7      for(int i=0; i<g[x].size(); ++i)
8      {
9          int to=g[x][i];
10         if(!dfn[to])
11         {
12             tarjan(to);
13             low[x]=min(low[x], low[to]);
14             if(low[to]>=dfn[x]) val[x]++;
15         }
16         else low[x]=min(low[x], dfn[to]);
17     }
18 }
19
20 int main()
21 {
22     scanf("%d%d", &n, &m);
23     for(int i=1; i<=m; ++i)
24     {
25         int x, y;
26         scanf("%d%d", &x, &y);
27         g[x].push_back(y);
28         g[y].push_back(x);
29     }
30     for(int i=1; i<=n; ++i)
31         if(!dfn[i])
32         {
33             ans++;
34             tarjan(i);
35             if(val[i]) val[i]--;
36         }
37     //val[i]>0 代表 i 为割点
38     return 0;
39 }

```

Dinic

```

1  int n, m, S, T, cnt=1, head[MAXN], dis[MAXN];
2  LL maxflow;
3
4  struct Edge {
5      int next, to;
6      LL flow;

```

```

7   } edge[MAXM*2];
8
9   queue<int> q;
10
11  inline void addedge(int from, int to, int flow)
12  {
13      edge[++cnt].next=head[from];
14      edge[cnt].to=to;
15      edge[cnt].flow=flow;
16      head[from]=cnt;
17  }
18
19  bool bfs()
20  {
21      for(int i=1; i<=n; ++i) dis[i]=0;
22      dis[S]=1;
23      q.push(S);
24      while(!q.empty())
25      {
26          int x=q.front(); q.pop();
27          for(int i=head[x]; i; i=edge[i].next)
28          {
29              int to=edge[i].to;
30              if(dis[to] || !edge[i].flow) continue;
31              dis[to]=dis[x]+1;
32              q.push(to);
33          }
34      }
35      return dis[T]>0;
36  }
37
38  LL dfs(int x, LL flow)
39  {
40      if(x==T) return flow;
41      LL add=0;
42      for(int i=head[x]; i && flow; i=edge[i].next)
43      {
44          int to=edge[i].to;
45          if(dis[to]!=dis[x]+1 || !edge[i].flow) continue;
46          LL f=dfs(to, min(edge[i].flow, flow));
47          edge[i].flow-=f, edge[i^1].flow+=f;
48          add+=f; flow-=f;
49      }
50      if(!add) dis[x]=0;
51      return add;
52  }
53
54  int main()
55  {

```

```

56     scanf("%d%d%d%d", &n, &m, &S, &T);
57     for(int i=1; i<=m; i++)
58     {
59         int u, v, w;
60         scanf("%d%d%d", &u, &v, &w);
61         addedge(u, v, w);
62         addedge(v, u, 0);
63     }
64     while(bfs()) maxflow+=dfs(S, INF);
65     printf("%lld\n", maxflow);
66 }

```

EK费用流

```

1  int n, m, S, T, cnt=1, head[MAXN];
2  int maxflow, mincost, vis[MAXN], dis[MAXN];
3
4  struct Edge {int next, to, flow, cost;} edge[MAXM*2];
5  struct Pre {int id, from;} pre[MAXN];
6
7  queue<int> q;
8
9  void addedge (int from, int to, int flow, int cost)
10 {
11     edge[++cnt].next=head[from];
12     edge[cnt].cost=cost;
13     edge[cnt].flow=flow;
14     edge[cnt].to=to;
15     head[from]=cnt;
16 }
17
18 bool spfa()
19 {
20     for(int i=0; i<=n; ++i) vis[i]=0, dis[i]=INF;
21     vis[S]=1; dis[S]=0;
22     q.push(S);
23     while(!q.empty())
24     {
25         int x=q.front(); q.pop();
26         vis[x]=0;
27         for(int i=head[x]; i; i=edge[i].next)
28         {
29             int to=edge[i].to;
30             if(dis[to]>dis[x]+edge[i].cost && edge[i].flow)
31             {
32                 dis[to]=dis[x]+edge[i].cost;
33                 pre[to].from=x, pre[to].id=i;
34                 if(!vis[to])
35                     {

```



```

36         q.push(to);
37         vis[to]=1;
38     }
39 }
40 }
41 }
42 return dis[T]<dis[0];
43 }
44
45 int main()
46 {
47     scanf("%d%d%d%d", &n, &m, &S, &T);
48     for(int i=1; i<=m; i++)
49     {
50         int u, v, w, f;
51         scanf("%d%d%d%d", &u, &v, &w, &f);
52         addedge(u, v, w, f);
53         addedge(v, u, 0, -f);
54     }
55     maxflow=0, mincost=0;
56     while(spfa())
57     {
58         int flow=INF;
59         for(int i=T; i!=S; i=pre[i].from) flow=min(flow, edge[pre[i].id].flow);
60         for(int i=T; i!=S; i=pre[i].from)
61         {
62             edge[pre[i].id].flow-=flow;
63             edge[pre[i].id^1].flow+=flow;
64         }
65         maxflow+=flow;
66         mincost+=dis[T]*flow;
67     }
68     printf("%d %d\n", maxflow, mincost);
69     return 0;
70 }

```

集合运算

枚举子集

```
1  int n;
2
3  int main()
4  {
5      scanf("%d", &n);
6      // 预处理
7      for(int sta=1; sta<(1<<n); ++sta)
8          for(int sub=sta; sub; sub=(sub-1)&sta)
9              {
10                 // sub 为 sta 的子集
11             }
12     return 0;
13 }
```

SOS DP

```
1  int n, f[MAXN], g[MAXN];
2
3  int main()
4  {
5      scanf("%d", &n);
6      for(int i=0; i<1<<n; ++i) scanf("%d%d", &f[i], &g[i]);
7      for(int i=0; i<n; ++i)
8          for(int sta=0; sta<1<<n; ++sta) {
9              if((sta>>i)&1) f[sta]+=f[sta^(1<<i)]; //子集和
10             if(!((sta>>i)&1)) g[sta]+=g[sta^(1<<i)]; //母集和
11         }
12     return 0;
13 }
```

快速沃尔什变换

```
1  void or_fwt(vi &a, int op)
2  {
3      for(int n=sz(a), step=1; step<n; step*=2)
4          for(int i=0; i<n; i+=2*step) for(int j=i; j<i+step; ++j) {
5              int &u=a[j], &v=a[j+step];
6              tie(u, v)=op>0?MP(add(u, v), u):MP(v, sub(u, v));
7          }
8  }
9
10 void and_fwt(vi &a, int op)
11 {
```

```

12     for(int n=sz(a), step=1; step<n; step*=2)
13         for(int i=0; i<n; i+=2*step) for(int j=i; j<i+step; ++j) {
14             int &u=a[j], &v=a[j+step];
15             tie(u, v)=op>0?MP(v, add(u, v)):MP(sub(v, u), u);
16         }
17     }
18
19 void xor_fwt(vi &a, int op)
20 {
21     for(int n=sz(a), step=1; step<n; step*=2)
22         for(int i=0; i<n; i+=2*step) for(int j=i; j<i+step; ++j) {
23             int &u=a[j], &v=a[j+step];
24             tie(u, v)=MP(add(u, v), sub(u, v));
25         }
26     if(op<0) {
27         int inv=qpow(sz(a));
28         for(int i=0; i<sz(a); ++i) a[i]=1LL*a[i]*inv%P;
29     }
30 }
31
32 vi or_conv(vi a, vi b)
33 {
34     or_fwt(a, 1), or_fwt(b, 1);
35     for(int i=0; i<sz(a); ++i) a[i]=1LL*a[i]*b[i]%P;
36     or_fwt(a, -1);
37     return a;
38 }
39
40 vi and_conv(vi a, vi b)
41 {
42     and_fwt(a, 1), and_fwt(b, 1);
43     for(int i=0; i<sz(a); ++i) a[i]=1LL*a[i]*b[i]%P;
44     and_fwt(a, -1);
45     return a;
46 }
47
48 vi xor_conv(vi a, vi b)
49 {
50     xor_fwt(a, 1), xor_fwt(b, 1);
51     for(int i=0; i<sz(a); ++i) a[i]=1LL*a[i]*b[i]%P;
52     xor_fwt(a, -1);
53     return a;
54 }

```

子集卷积

```

1 // WC2018 州区划分
2 int n, m, p, vis[22], w[22], v[MAXN], iv[MAXN];
3 vi vec[22], f[22], g[22];

```

```

4
5 int dfs(int x, int sta)
6 {
7     int cnt=1, deg=0;
8     vis[x]=1;
9     for(int to: vec[x])
10         if((sta>>to)&1) {
11             deg++;
12             if(vis[to]) continue;
13             int tmp=dfs(to, sta);
14             if(tmp== -1) return -1;
15             else cnt+=tmp;
16         }
17     return deg%2==0?cnt:-1;
18 }
19
20 int main()
21 {
22     scanf("%d%d%d", &n, &m, &p);
23     for(int i=1; i<=m; ++i) {
24         int x, y;
25         scanf("%d%d", &x, &y);
26         vec[x-1].PB(y-1);
27         vec[y-1].PB(x-1);
28     }
29     for(int i=0; i<n; ++i) scanf("%d", &w[i]);
30     for(int i=0; i<=n; ++i) f[i].resize(1<<n), g[i].resize(1<<n);
31     for(int sta=1; sta<1<<n; ++sta) {
32         memset(vis, 0, sizeof(vis));
33         int cnt=0;
34         for(int i=0; i<n; ++i)
35             if((sta>>i)&1) cnt++, v[sta]+=w[i];
36         if(!p) v[sta]=1;
37         else if(p==2) v[sta]=1LL*v[sta]*v[sta]%P;
38         if(dfs(__builtin_ctz(sta), sta)==cnt) g[cnt][sta]=0;
39         else g[cnt][sta]=v[sta];
40         iv[sta]=qpow(v[sta]);
41     }
42     for(int i=0; i<=n; ++i) fwt(g[i], 0);
43     for(int i=0; i<=n; ++i) {
44         if(i==0) {
45             f[0][0]=1;
46             fwt(f[0], 0);
47             for(int sta=0; sta<1<<n; ++sta) f[2][sta]=1LL*f[0][sta]*g[2][sta]%P;
48             fwt(f[2], 1);
49         } else {
50             fwt(f[i], 1);
51             for(int sta=0; sta<1<<n; ++sta) {
52                 //printf("%d %d %d\n", i, sta, f[i][sta]);

```

```

53         if(__builtin_popcount(sta)==i) f[i][sta]=1LL*f[i][sta]*iv[sta]%P;
54         else f[i][sta]=0;
55     }
56     fwt(f[i], 0);
57 }
58 for(int j=0; j<=n-i; ++j)
59     for(int sta=0; sta<1<<n; ++sta)
60         f[i+j][sta]=(f[i+j][sta]+1LL*f[i][sta]*g[j][sta])%P;
61
62 }
63 fwt(f[n], 1);
64 printf("%d\n", f[n][(1<<n)-1]);
65 return 0;
66 }

```

其他

离散化

```
1  int n, cnt, a[MAXN], b[MAXN], temp[MAXN*2], suba[MAXN], subb[MAXN];
2
3  int main()
4  {
5      scanf("%d", &n);
6      for(int i=1; i<=n; ++i)
7      {
8          scanf("%d%d", &a[i], &b[i]);
9          temp[i*2-1]=a[i], temp[2*i]=b[i];
10     }
11     sort(temp+1, temp+2*n+1);
12     cnt=unique(temp+1, temp+2*n+1)-temp-1;
13     for(int i=1; i<=n; ++i)
14     {
15         suba[i]=lower_bound(temp+1, temp+cnt+1, a[i])-temp;
16         subb[i]=lower_bound(temp+1, temp+cnt+1, b[i])-temp;
17     }
18     // suba subb 离散化后数组
19     return 0;
20 }
```

大数运算

```
1  //只限两个非负整数相加
2  string add(string a, string b)
3  {
4      string ans;
5      int na[MAXL]={0}, nb[MAXL]={0};
6      int la=a.size(), lb=b.size();
7      for(int i=0; i<la; i++) na[la-1-i]=a[i]-'0';
8      for(int i=0; i<lb; i++) nb[lb-1-i]=b[i]-'0';
9      int lmax=la>lb?la:lb;
10     for(int i=0; i<lmax; i++) na[i]+=nb[i], na[i+1]+=na[i]/10, na[i]%=10;
11     if(na[lmax]) lmax++;
12     for(int i=lmax-1; i>=0; i--) ans+=na[i]+'0';
13     return ans;
14 }
15
16 //只限大的非负整数减小的非负整数
17 string sub(string a, string b)
18 {
19     string ans;
20     int na[MAXL]={0}, nb[MAXL]={0};
21     int la=a.size(), lb=b.size();
```

```

22     for(int i=0; i<la; i++) na[la-1-i]=a[i]-'0';
23     for(int i=0; i<lb; i++) nb[lb-1-i]=b[i]-'0';
24     int lmax=la>lb?la:lb;
25     for(int i=0; i<lmax; i++)
26     {
27         na[i]-=nb[i];
28         if(na[i]<0) na[i]+=10, na[i+1]--;
29     }
30     while(!na[--lmax]&&lmax>0) ;lmax++;
31     for(int i=lmax-1; i>=0; i--) ans+=na[i]+'0';
32     return ans;
33 }
34
35 //只限非负整数相乘
36 string mul(string a, string b)
37 {
38     string ans;
39     int na[MAXL]={0}, nb[MAXL]={0}, nc[MAXL]={0}, La=a.size(), Lb=b.size(); //na存
    储被乘数, nb存储乘数, nc存储积
40     for(int i=La-1; i>=0; i--) na[La-i]=a[i]-'0'; //将字符串表示的大整数数转成i整形数组表
    示的大整数数
41     for(int i=Lb-1; i>=0; i--) nb[Lb-i]=b[i]-'0';
42     for(int i=1; i<=La; i++)
43         for(int j=1; j<=Lb; j++)
44             nc[i+j-1]+=na[i]*nb[j]; //a的第i位乘以b的第j位为积的第i+j-1位 (先不考虑进位)
45     for(int i=1; i<=La+Lb; i++)
46         nc[i+1]+=nc[i]/10, nc[i]%=10; //统一处理进位
47     if(nc[La+Lb]) ans+=nc[La+Lb]+'0'; //判断第i+j位上的数字是不是0
48     for(int i=La+Lb-1; i>=1; i--) ans+=nc[i]+'0';
49     return ans;
50 }
51
52 //高精度整数除单精度整数
53 string div(string a, int b)
54 {
55     string r, ans;
56     int d=0;
57     for(int i=0; i<a.size(); i++)
58     {
59         r+=(d*10+a[i]-'0')/b+'0';//求出商
60         d=(d*10+(a[i]-'0'))%b;//求出余数
61     }
62     int p=0;
63     for(int i=0; i<r.size(); i++)
64         if(r[i]!='0') {p=i; break;}
65     return r.substr(p);
66 }

```

`__builtin_clz(x)` 返回 x 前导 0 的个数

`__builtin_ctz(x)` 返回 x 末尾 0 的个数

`__builtin_popcount(x)` 返回 x 中 1 的个数

bitset

`count()` 返回 1 的个数

`any()` 返回是否有 1

`set()` 全体/某位置 1

`reset()` 全体/某位置 0

`flip()` 全体/某位取反

`_Find_first()` 返回最低位 1 的位置

`_Find_next(p)` 返回第 p 位后第一个 1 的位置

string

`insert (size_t pos, const string& str)` 在位置 pos 上插入串 str

`erase (size_t pos, size_t len = npos)` 删除位置 pos 上长度为 len 的串

`find (const string& str, size_t pos = 0)` 返回位置 pos 后串 str 第一次出现的位置

`substr (size_t pos = 0, size_t len = npos)` 返回在位置 pos 上长度为 len 的子串

对拍

```
1  while true; do
2      ./gen > a.in
3      ./A < a.in > a.out
4      ./std < a.in > a.ans
5      if diff a.out a.ans; then
6          echo AC
7      else
8          echo WA
9          exit 0
10     fi
11 done
```