Template

GummyBear_

October 10, 2019

目录

1 Math **1**
1.1 FT_fft_2D 1
1.2 FT_fft_2D_使用说明 1
1.3 FT_fft 基础版本 2
1.4 NTT_2D 2
1.5 Poly_多项式相关 3
1.6 Poly_扩展版 7
1.7 Poly_标准版 10
1.8 一维 FFT_单模式串_模式串带通配符匹配 . . 11
1.9 二维 FFT_单模式串_模式串带通配符匹配 . . 12
1.10 扩展卢卡斯 13
1.11 拟阵交 14
1.12 拟阵交_带权 16

2 String **19**
2.1 PAM_优化转移_偶回文切割方案数 19
2.2 PAM_优化转移_最小回文切割段数 19
2.3 PAM_单串_支持双端插入 20
2.4 PAM_单串_支持撤销_单加 log 21
2.5 PAM_单串_支持撤销_单加可退化 21
2.6 PAM_多串模式 21
2.7 PAM_标准版本 22
2.8 PAM_空间压缩_单链表 22
2.9 SAM_多串模式_串运行_树上合并_map . . . 22
2.10 SAM_广义_trie 树 23
2.11 SAM_标准版 24

1 Math

1.1 FT_fft_2D

```
const int _M = 2050, _N = N;
template <class V>
struct FT {
   struct cp { double x, y; } tmp[_M * 2 + 5]; cp aa[_M][_M], bb[_M][_M];
   friend cp operator + (cp &a, cp &b) { return cp{ a.x + b.x,a.y + b.y }; }
   friend cp operator - (cp &a, cp &b) { return cp{ a.x - b.x,a.y - b.y }; }
   friend cp operator * (cp &a, cp &b) { return cp{ a.x*b.x - a.y*b.y,a.x*b.y + a.y*b.x }; }
   cp get(double x) { return cp{ cos(x),sin(x) }; }
   void FFT(cp *a, int n, int op) {
      for (int i = (n >> 1), j = 1; j < n; j++) {
         if (i < j) swap(a[i], a[j]);
         int k; for (k = (n >> 1); k&i; i ^= k, k >>= 1); i ^= k;
      }
      for (int m = 2; m <= n; m <<= 1) {
         cp w = get(2 * PI*op / m); tmp[0] = cp{ 1,0 };
         for (int j = 1; j < (m >> 1); j++) tmp[j] = tmp[j - 1] * w;
         for (int i = 0; i < n; i += m)
            for (int j = i; j < i + (m >> 1); j++) {
               cp u = a[j], v = a[j + (m >> 1)] * tmp[j - i];
               a[j] = u + v, a[j + (m >> 1)] = u - v;
            }
      }
      if (op == -1) rep(i, 0, n) a[i] = cp{ a[i].x / n,a[i].y / n };
   }
   void FFT(cp a[][_M], int n, int op) { rep(i, 0, n) FFT(a[i], n, op); }
   template <class T>
   void Transpose(T a[][_M], int n) {
      rep(i, 0, n) rep(j, 0, i) swap(a[i][j], a[j][i]);
   }
   void Reverse(V a[][_M], int n, int m) {
      rep(i, 0, (n - 1 >> 1) + 1) rep(j, 0, m) swap(a[i][j], a[n - 1 - i][j]);
      rep(i, 0, n) rep(j, 0, (m - 1 >> 1) + 1) swap(a[i][j], a[i][m - 1 - j]);
   }
   void Shift(V a[][_M], int n, int m, int p, int q) {
      rep(i, n, n + p) rep(j, m, m + q) a[i - n][j - m] = a[i][j];
   }
   void In(cp p[][_M], int len, V a[][_M], int n, int m) {
      rep(i, 0, len) rep(j, 0, len) p[i][j] = cp{ i < n&&j < m ? (double)a[i][j] : 0,0 };
   }
   void Out(V a[][_M], int n, int m, cp p[][_M], int len) {
      rep(i, 0, n) rep(j, 0, m) a[i][j] = (V)(p[i][j].x + 0.5) % _p;
   }
   void Multiply(V A[][_M], int n, V B[][_M], int m, V C[][_M], int &len, int op = 0) {
      if (op) Reverse(A, n, n);
      len = 1; while (len < n + m - 1) len <<= 1;
      In(aa, len, A, n, n), In(bb, len, B, m, m), FFT(aa, len, 1), FFT(bb, len, 1);
      Transpose(aa, len), Transpose(bb, len), FFT(aa, len, 1), FFT(bb, len, 1);
      rep(i, 0, len) rep(j, 0, len) aa[i][j] = aa[i][j] * bb[i][j];
      FFT(aa, len, -1), Transpose(aa, len), FFT(aa, len, -1), Out(C, len, len, aa, len);
      if (op) Shift(C, n - 1, n - 1, m, m), len = m, Reverse(A, n, n);
   }
};

inline void Random(int a[][_M], int n) {
   rep(i, 0, n) rep(j, 0, n) a[i][j] = rand();
}
```

1.2 FT_fft_2D_使用说明

```
/*
* FT_fft_2D 使用说明
*
* 【接口说明】
*
* cp get(double x) ： 获取一个辐角为 x 的复数
*
* void FFT(cp *a,int n,int op) ： 变换接口，注意： op=1 为正卷积， op=-1 为逆卷积
*
* void FFT(cp a[ ][_M],int n,int op) ： 行变换接口，逐行进行正 / 逆变换
*
* void Transpose(T a[ ][_M],int n) ： 转置接口
*
* void Reverse(V a[ ][_M],int n,int m) ： 翻转接口
*
* void Shift(V a[ ][_M],int n,int m,int p,int q) ： 移位接口，将矩阵 a 的 (n,m) 整体移位到 (0,0) 长度保留 p 和 q ( 长和宽 )
*
* void In(cp p[][_M],int len,V a[][_M],int n,int m) ： 数据填充接口
```

```
*
* void Out(V a[][_M],int n,int m,cp p[][_M],int len) : 数据提取接口
*
* void Multiply(V A[][_M],int n,V B[][_M],int m,V C[][_M],int &len,int op=0) :
*
* 乘法接口，表示 n * n 的矩阵 A 乘上 m*m 的矩阵 B ，结果放到 C 中，规模为 len * len 且计算并返回到 len 中， op=1 为差卷积
*/
```

1.3 FT_fft 基础版本

```
const int _M = N, _N = N;
template <class V>
struct FT {
   struct cp { double x, y; } tmp[_M * 2 + 5];
   friend cp operator + (cp &a, cp &b) { return cp{ a.x + b.x,a.y + b.y }; }
   friend cp operator − (cp &a, cp &b) { return cp{ a.x − b.x,a.y − b.y }; }
   friend cp operator * (cp &a, cp &b) { return cp{ a.x*b.x − a.y*b.y,a.x*b.y + a.y*b.x }; }
   cp get(double x) { return cp{ cos(x),sin(x) }; }
   vector <cp> aa, bb;
   void FFT(vector<cp> &a, int n, int op) {
      for (int i = (n >> 1), j = 1; j < n; j++) {
         if (i < j) swap(a[i], a[j]);
         int k; for (k = (n >> 1); k&i; i ^= k, k >>= 1); i ^= k;
      }
      for (int m = 2; m <= n; m <<= 1) {
         cp w = get(2 * PI*op / m); tmp[0] = cp{ 1,0 };
         for (int j = 1; j < (m >> 1); j++) tmp[j] = tmp[j − 1] * w;
         for (int i = 0; i < n; i += m)
            for (int j = i; j < i + (m >> 1); j++) {
               cp u = a[j], v = a[j + (m >> 1)] * tmp[j − i];
               a[j] = u + v, a[j + (m >> 1)] = u − v;
            }
      }
      if (op == −1) rep(i, 0, n) a[i] = cp{ a[i].x / n,a[i].y / n };
   }
   vector<V> multiply(vector<V> A, vector<V> B, int op = 0) {
      if (op) reverse(all(A));
      int lena = A.size(), lenb = B.size(), len = 1;
      while (len < lena + lenb) len <<= 1;
      aa = vector<cp>(len), bb = vector<cp>(len);
      rep(i, 0, lena) aa[i] = cp{ (double)A[i],0 };
      rep(i, 0, lenb) bb[i] = cp{ (double)B[i],0 };
      FFT(aa, len, 1), FFT(bb, len, 1);
      rep(i, 0, len) aa[i] = aa[i] * bb[i];
      FFT(aa, len, −1); A.clear();
      if (!op) rep(i, 0, len) A.pb((ll)(aa[i].x + 0.5)); else
         rep(i, lena − 1, lena + lenb − 2 + 1) A.pb((ll)(aa[i].x + 0.5));
      return A;
   }
};
```

1.4 NTT_2D

```
typedef vector<vector<int>> vii;

const int _M = N, _N = N;
template <class V>
struct FT {
   vector <V> aa, bb; int _p, K, _m, N; V w[2][_M * 2 + 5], rev[_M * 2 + 5], tmp, w0;
   inline void Init(int _K, int p) { K = _K, _p = p; }
   ll Pow(ll x, ll k, ll _p) { ll ans = 1; for (; k; k >>= 1, x = x*x%_p) if (k & 1) (ans *= x) %= _p; return ans; }
   inline void get_len(int a, int b, int &len, int &L) { len = 1, L = 0; while (len < a + b) len <<= 1, ++L; }
   inline void init_w(int m) {
      N = 1 << m, w0 = Pow(3, (_p − 1) / N, _p); w[0][0] = w[1][0] = 1;
      rep(i, 1, N) w[0][i] = w[1][N − i] = (ll)w[0][i − 1] * w0%_p;
      rep(i, 1, N) rev[i] = (rev[i >> 1] >> 1) | (i & 1) << m − 1;
   }
   inline void FFT(vector<V>& A, int m, int op) {
      if (m != _m) init_w(_m = m);
      rep(i, 0, N) if (i < rev[i]) swap(A[i], A[rev[i]]);
      for (int i = 1; i < N; i <<= 1)
         for (int j = 0, t = N / (i << 1); j < N; j += i << 1)
            for (int k = j, l = 0; k < j + i; k++, l += t) {
               V x = A[k], y = (ll)w[op][l] * A[k + i] % _p;
               A[k] = (x + y) % _p, A[k + i] = (x − y + _p) % _p;
            }
      if (op) { tmp = Pow(N, _p − 2, _p); rep(i, 0, N) A[i] = 1ll * A[i] * tmp%_p; }
   }
   inline void multiply(const vector<V>& A, const vector<V>& B, vector<V> *C) {
      int lena = A.size(), lenb = B.size(), len = 1, L = 0; aa = A, bb = B;
      get_len(lena, lenb, len, L), aa.resize(len), bb.resize(len);
      FFT(aa, L, 0), FFT(bb, L, 0), (*C).resize(len);
```

```
      rep(i, 0, len) (*C)[i] = (ll)aa[i] * bb[i] % _p;
      FFT(*C, L, 1); if (K < len - 1) (*C).resize(K + 1);
   }
};

struct Matrix {
   int n, m; vii a;
   inline void Set_m(int _m, int x = 0) { m = _m; rep(i, 0, n) a[i].resize(m, x); }
   inline void Set_n(int _n) { n = _n, a.resize(n); }
   inline void Set(int _n, int _m, int x = 0) { Set_n(_n), Set_m(_m, x); }
   Matrix(int n = 0, int m = 0, int x = 0) :n(n), m(m) {
      a.clear(), a.resize(n); Set_m(m, x);
   }
   inline void Transpose() {
      Matrix t = Matrix(m, n, 0);
      rep(i, 0, n) rep(j, 0, m) t.a[j][i] = a[i][j];
      *this = t;
   }
   inline void Reverse() {
      Matrix t = Matrix(n, m, 0);
      rep(i, 0, n) rep(j, 0, m) t.a[n - 1 - i][m - 1 - j] = a[i][j];
      *this = t;
   }
   inline void Shift(int x, int y) {
      rep(i, x, n - 1 + x + 1) rep(j, y, m - 1 + y + 1) a[i - x][j - y] = (i < n&&j < m) ? a[i][j] : 0;
   }
   inline void FFT(FT<int> &T, int len, int op) {
      if (!op) Set_m(1 << len, 0);
      rep(i, 0, n) T.FFT(a[i], len, op);
   }
   inline void print() const;
   inline void Normalize(int _p);
   inline void Random();
};

inline void Matrix::print() const {
   printf("\n\n\n\n\nn => %d     m => %d\n", n, m);
   debug_arr2(a, n - 1, m - 1);
}

inline void Matrix::Normalize(int _p) {
   rep(i, 0, n) rep(j, 0, m) if (a[i][j] < 0) a[i][j] += _p;
}

inline void Matrix::Random() {
   rep(i, 0, n) rep(j, 0, m) a[i][j] = (rand() << 15) + rand();
}

inline bool operator==(const Matrix &A, const Matrix &B) {
   if (A.n != B.n || A.m != B.m) return 0;
   rep(i, 0, A.n) rep(j, 0, A.m) if (A.a[i][j] != B.a[i][j]) return 0;
   return 1;
}

struct Calculator {
   Matrix aa, bb, cc; FT<int> T; int len, L, _p;
   inline void Init(int p) { _p = p; }
   inline void Multiply(const Matrix &A, const Matrix &B, Matrix &C, int op = 0) {
      aa = A, bb = B; if (op) aa.Reverse(); T.get_len(A.m, B.m, len, L), T.Init(cc.m = A.n + B.n - 1, _p);
      aa.FFT(T, L, 0), bb.FFT(T, L, 0), aa.Transpose(), bb.Transpose(), cc.Set_n(aa.n);
      rep(i, 0, aa.n) T.multiply(aa.a[i], bb.a[i], &cc.a[i]);
      cc.Transpose(), cc.FFT(T, L, 1), cc.Set_m(A.m + B.m - 1), C = cc;
      if (op) C.Shift(A.n - 1, A.m - 1);
   }
   inline void add(int &x, int y) { x += y, x %= _p; }
   inline int mul(int x, int y) { return (ll)x*y%_p; }
   inline void Multiply_B(const Matrix &A, const Matrix &B, Matrix &C) {
      C.Set(A.n + B.n - 1, A.m + B.m - 1);
      rep(xa, 0, A.n) rep(ya, 0, A.m) rep(xb, 0, B.n) rep(yb, 0, B.m)
         add(C.a[xa + xb][ya + yb], mul(A.a[xa][ya], B.a[xb][yb]));
   }
   inline void Multiply_B_sub(const Matrix &A, const Matrix &B, Matrix &C) {
      C.Set(A.n + B.n - 1, A.m + B.m - 1);
      rep(xa, 0, A.n) rep(ya, 0, A.m) rep(xb, xa, B.n) rep(yb, ya, B.m)
         add(C.a[xb - xa][yb - ya], mul(A.a[xa][ya], B.a[xb][yb]));
   }
};
```

1.5 Poly_多项式相关

```
#include<bits/stdc++.h>
using namespace std;
```

```
#define pb push_back
#define mp make_pair
#define rep(i, a, b) for(int i=(a); i<(b); i++)
#define per(i, a, b) for(int i=(b)-1; i>=(a); i--)
#define fi first
#define se second
#define fe first
#define FI(x) freopen(#x".in","r",stdin)
#define FO(x) freopen(#x".out","w",stdout)
typedef pair<int,int> pii;
typedef long long ll;
typedef double ld;
typedef vector<int> vi;

const int P=998244353;
#define SZ 666666
ll w[2][SZ],rev[SZ];
inline ll qp(ll a,ll b) {
   ll ans=1;
   while(b) {
      if(b&1) ans=ans*a%P;
      a=a*a%P;
      b>>=1;
   }
   return ans;
}
int K;
inline void fftinit(int n) {
   for(K=1; K<n; K<<=1);
   w[0][0]=w[0][K]=1;
   ll g=qp(3,(P-1)/K);
   for(int i=1; i<K; i++) w[0][i]=w[0][i-1]*g%P;
   for(int i=0; i<=K; i++) w[1][i]=w[0][K-i];
}
inline void fft(int* x,int v) {
   for(int i=0; i<K; i++) x[i]=(x[i]%P+P)%P;
   for(int i=0,j=0; i<K; i++) {
      if(i>j) swap(x[i],x[j]);
      for(int l=K>>1; (j^=l)<l; l>>=1);
   }
   for(int i=2; i<=K; i<<=1)
      for(int l=0; l<i>>1; l++) {
         register int W=w[v][K/i*l],*p=x+l+(i>>1),*q=x+l,t;
         for(register int j=0; j<K; j+=i) {
            p[j]=(q[j]-(t=(ll)p[j]*W%P)<0)?(q[j]-t+P):(q[j]-t);
            q[j]=(q[j]+t-P>=0)?(q[j]+t-P):(q[j]+t);
         }
      }
   if(!v) return;
   ll rv=qp(K,P-2);
   for(int i=0; i<K; i++) x[i]=x[i]*rv%P;
}
struct poly {
   vector<int> ps;
   inline int cs() {
      return ps.size()-1;
   }
   inline int& operator [] (int x) {
      return ps[x];   //ps.at(x)
   }
   inline void sc(int x) {
      ps.resize(x+1);
   }
   inline void dbg() {
      bool fi=0;
      for(int i=cs(); i>=0; i--) {
         ps[i]=(ps[i]%P+P)%P;
         if(!ps[i]) continue;
         if(ps[i]>P/2) ps[i]-=P;
         if(fi) {
            if(i==0) printf("%+d",ps[i]);
            else if(ps[i]==1) printf("+");
            else if(ps[i]==-1) printf("-");
            else printf("%+d",ps[i]);
         } else {
            if(i==0) printf("%d",ps[i]);
            else if(ps[i]==1);
            else if(ps[i]==-1) printf("-");
            else printf("%d",ps[i]);
         }
         if(i>1) printf("x^%d",i);
         else if(i==1) printf("x");
```

```
        fi=1;
      }
      if(!fi) printf("0");
      putchar(10);
   }
   inline void clr() {
      int p=cs()+1;
      while(p&&!ps[p-1]) --p;
      sc(p-1);
   }
};
namespace PolyMul { int ta[SZ], tb[SZ], tc[SZ];}
inline poly operator * (poly a, poly b) {
   using namespace PolyMul;
   if(a.cs()<180||b.cs()<180) {
      poly g;
      g.sc(a.cs()+b.cs());
      int*G=&g[0],*A=&a[0],*B=&b[0];
      for(int i=0; i<=a.cs(); i++) {
         register int*h=G+i,j=0;
         register ll x=A[i];
         for(; j<=b.cs(); ++j) h[j]=(h[j]+x*(ll)B[j])%P;
      }
      return g;
   }
   poly c;
   int t=a.cs()+b.cs();
   c.sc(t);
   fftinit(t+1);
   memset(ta,0,sizeof(int)*K);
   memset(tb,0,sizeof(int)*K);
   memset(tc,0,sizeof(int)*K);
   for(int i=a.cs(); i>=0; i--) ta[i]=a[i];
   for(int i=b.cs(); i>=0; i--) tb[i]=b[i];
   fft(ta,0);
   fft(tb,0);
   for(int i=0; i<K; i++) tc[i]=(ll)ta[i]*tb[i]%P;
   fft(tc,1);
   for(int i=t; i>=0; i--) c[i]=tc[i];
   c.clr();
   return c;
}
namespace PolyInv {
   int ay[SZ],a0[SZ],tmp[SZ];
}
inline void ginv(int t) {
   using namespace PolyInv;
   if(t==1) {
      a0[0]=qp(ay[0],P-2);
      return;
   }
   ginv((t+1)>>1);
   fftinit(t+t+3);
   memset(tmp,0,sizeof(int)*K);
   for(int i=t; i<K; i++) tmp[i]=a0[i]=0;
   for(int i=0; i<t; i++) tmp[i]=ay[i];
   fft(tmp,0);
   fft(a0,0);
   for(int i=0; i<K; i++) a0[i]=(2-(ll)tmp[i]*a0[i])%P*a0[i]%P;
   fft(a0,1);
   for(int i=t; i<K; i++) a0[i]=0;
}
inline poly inv(poly x) {
   using namespace PolyInv;
   poly y;
   y.sc(x.cs());
   for(int i=x.cs(); i>=0; i--) ay[i]=x[i];
   ginv(x.cs()+1);
   for(int i=x.cs(); i>=0; i--) y[i]=a0[i];
   y.clr();
   return y;
}
inline poly operator + (poly a,poly b) {
   poly w;
   w.sc(max(a.cs(),b.cs()));
   for(int i=a.cs(); i>=0; i--) w[i]=a[i];
   for(int i=b.cs(); i>=0; i--) (w[i]+=b[i])%=P;
   return w;
}
inline poly operator - (poly a,poly b) {
   poly w;
   w.sc(max(a.cs(),b.cs()));
```

```
   for(int i=a.cs(); i>=0; i--) w[i]=a[i];
   for(int i=b.cs(); i>=0; i--) (w[i]-=b[i])%=P;
   w.clr();
   return w;
}
inline void div(poly a,poly b,poly& d,poly& r) {
   int n=a.cs(),m=b.cs();
   if(n<m) {
      d.sc(0);
      d[0]=0;
      r=a;
      return;
   }
   fftinit(2*n);
   poly aa=a;
   reverse(aa.ps.begin(),aa.ps.end());
   poly bb=b;
   reverse(bb.ps.begin(),bb.ps.end());
   bb.sc(n-m);
   bb=inv(bb);
   d=aa*bb;
   d.sc(n-m);
   reverse(d.ps.begin(),d.ps.end());
   r=a-b*d;
   r.clr();
}
inline poly operator / (poly a,poly b) {
   poly d,r;
   div(a,b,d,r);
   return d;
}
inline poly operator % (poly a,poly b) {
   a.clr();
   b.clr();
   if(a.cs()<b.cs()) return a;
   poly d,r;
   div(a,b,d,r);
   return r;
}
inline poly dev(poly x) {
   for(int i=1; i<=x.cs(); i++) x[i-1]=(ll)x[i]*i%P;
   x.sc(x.cs()-1);
   return x;
}
inline poly inte(poly x) {
   x.sc(x.cs()+1);
   for(int i=x.cs(); i>=1; i--) x[i]=x[i-1];
   x[0]=0;
   for(int i=x.cs(); i>=1; i--) x[i]=(ll)x[i]*rev[i]%P;
   return x;
}
inline ll qz(poly& a,ll x) {
   ll ans=0;
   for(int i=a.cs(); i>=0; i--) ans=(ans*x+a[i])%P;
   return ans;
}
poly vvs[SZ];
inline void gvs(int m,int* x,int id) {
   if(m==1) {
      vvs[id].sc(1), vvs[id][1]=1, vvs[id][0]=-*x;
      return;
   }
   int hf=m>>1;
   gvs(hf,x,id*2);
   gvs(m-hf,x+hf,id*2+1);
   vvs[id]=vvs[id*2]*vvs[id*2+1];
}
namespace PolyGetv {
   int xs[SZ],anss[SZ];
};
inline void gv(poly f,int m,int* x,int* ans,int id) {
   if(f.cs()<=1000) {
      int c=f.cs(),*F=&f.ps[0];
      for(int i=0; i<m; i++) {
         register ll t=0;
         register int v=x[i];
         for(register int j=c; ~j; --j) t=(t*v+F[j])%P;
         ans[i]=t;
      }
      return;
   }
   int hf=m>>1;
```

```
   gv(f%vvs[id*2],hf,x,ans,id*2);
   gv(f%vvs[id*2+1],m-hf,x+hf,ans+hf,id*2+1);
}
inline vector<int> getv(poly a,vector<int> x) {
   using namespace PolyGetv;
   a.clr();
   if(!x.size()) return vector<int>();
   int m=x.size();
   for(int i=0; i<m; i++) xs[i]=x[i];
   gvs(m,xs,1);
   gv(a%vvs[1],m,xs,anss,1);
   vector<int> ans;
   ans.resize(m);
   for(int i=0; i<m; i++) ans[i]=anss[i];
   return ans;
}
namespace PolyIntp {
   int xs[SZ],vs[SZ];
};
inline poly comb(int m,int*v,int id) {
   if(m==1) {
      poly s;
      s.sc(0);
      s[0]=*v;
      return s;
   }
   int hf=m>>1;
   return comb(hf,v,id*2)*vvs[id*2+1]
         +comb(m-hf,v+hf,id*2+1)*vvs[id*2];
}
inline poly intp(vector<int> x,vector<int> y) {
   using namespace PolyIntp;
   int m=x.size();
   for(int i=0; i<m; i++) xs[i]=x[i];
   gvs(m,xs,1);
   gv(dev(vvs[1]),m,xs,vs,1);
   for(int i=0; i<m; i++)
      vs[i]=y[i]*qp(vs[i],P-2)%P;
   return comb(m,vs,1);
}

int n, m, x, y;

int main() {
   ios :: sync_with_stdio(0);
   cin.tie(0);
   rev[1]=1;
   for(int i=2; i<SZ; ++i) rev[i]=-rev[P%i]*(ll)(P/i)%P;
   cin >> n;
   vi xx, yy, zz;
   /*for(int i=0,x,y;i<n;++i) {
       cin >> x >> y;
      xx.pb(x),yy.pb(y);
   }
   poly g = intp(xx, yy);
   g.dbg(); */
   poly g;
   g.sc(n);
   rep(i, 0, n+1) cin >> x, g[i] = x;

   for(int i = 0, z; i < m; ++i) cin >> z, zz.pb(z);
   vector<int> vs = getv(g, zz);
   for(int i=0; i<m; ++i)
      cout << ((vs[i]%P+P)%P) << "\n";
}
```

1.6 Poly_扩展版

```
const int _N=200005; ll inv[_N<<2],fac[_N<<2],fac_inv[_N<<2];
inline ll add(ll x,ll y) { x+=y; return x%P; }
inline ll mul(ll x,ll y) { return (ll)x*y%P; }
inline ll Pow(ll x,ll k) { ll ans=1; for (;k;k>>=1,x=x*x%P) if (k&1) (ans*=x)%=P; return ans; }
inline void init_inv(int n) { inv[1]=1; rep(i,2,n+1) inv[i]=mul(P-P/i,inv[P%i]); }
inline void init_fac(int n) {
   fac[0]=fac_inv[0]=1;
   rep(i,1,n+1) fac[i]=mul(fac[i-1],i),fac_inv[i]=mul(fac_inv[i-1],inv[i]);
}

template <class V>
struct FT{
   int n,nn; V w[2][_N<<2],rev[_N<<2],tmp;
   inline int init_len(int _n) { for (n=1; n<=_n; n<<=1); return n; }
```

```
   inline int Init(int _n) {
      init_len(_n); if (n==nn) return n; nn=n;
      V w0=Pow(3,(P-1)/n); w[0][0]=w[1][0]=1;
      rep(i,1,n) w[0][i]=w[1][n-i]=mul(w[0][i-1],w0);
      rep(i,0,n) rev[i]=(rev[i>>1]>>1)|((i&1)*(n>>1)); return n;
   }
   void FFT(V A[],int op){
      rep(i,0,n) if (i<rev[i]) swap(A[i],A[rev[i]]);
      for (int i=1; i<n; i<<=1)
         for (int j=0,t=n/(i<<1); j<n; j+=i<<1)
            for (int k=j,l=0; k<j+i; k++,l+=t) {
               V x=A[k],y=mul(w[op][l],A[k+i]);
               A[k]=add(x,y),A[k+i]=add(x-y,P);
            }
      if (op) { tmp=inv[n]; rep(i,0,n) A[i]=mul(A[i],tmp); }
   }
};

template <class V>
struct Calculator{
   FT<V> T; V X[_N<<2],Y[_N<<2],A[_N<<2],B[_N<<2],C[_N<<2];
   inline void Fill(V a[],V b[],int n,int len) {
      if (a!=b) memcpy(a,b,sizeof(V)*n); fill(a+n,a+len,0);
   }
   inline void Add(V a[],int n,V b[],int m,V c[],int t=1) {
      n=max(n,m); rep(i,0,n) c[i]=add(a[i],t*b[i]);
   }
   inline void Dot_Mul(V a[],V b[],int len,V c[]) {
      rep(i,0,len) c[i]=mul(a[i],b[i]);
   }
   inline void Dot_Mul(V a[],int len,V v,V c[]) {
      rep(i,0,len) c[i]=mul(a[i],v);
   }
   inline void Mul(V a[],int n,V b[],int m,V c[]) {
      int len=T.Init(n+m-1); Fill(X,a,n,len),Fill(Y,b,m,len);
      T.FFT(X,0),T.FFT(Y,0),Dot_Mul(X,Y,len,c),T.FFT(c,1);
   }
   inline void Int(V a[],int n,V b[]) {
      per(i,0,n) b[i+1]=mul(a[i],inv[i+1]); b[0]=0;
   }
   inline void Der(V a[],int n,V b[]) {
      rep(i,1,n) b[i-1]=mul(a[i],i); b[n-1]=0;
   }
   inline void Inv(V a[],int n,V b[]) {
      if (n==1) { b[0]=Pow(a[0],P-2),b[1]=0; return; }
      Inv(a,(n+1)>>1,b); int len=T.Init(2*n-1);
      Fill(X,a,n,len),Fill(b,b,n,len),T.FFT(X,0),T.FFT(b,0);
      rep(i,0,len) b[i]=mul(b[i],2-mul(b[i],X[i]));
      T.FFT(b,1),Fill(b,b,n,len);
   }
   inline void Log(V a[],int n,V b[]) {
      static V A[_N<<2],B[_N<<2];
      Der(a,n,A),Inv(a,n,B),Mul(A,n,B,n,b);
      Int(b,n,b),Fill(b,b,n,T.n);
   }
   inline void Exp(V a[],int n,V b[]) {
      if (n==1) { b[0]=exp(a[0]),b[1]=0; return; }
      Exp(a,(n+1)>>1,A),Log(A,n,B),Add(a,n,B,n,B,-1);
      (B[0]+=1)%=P,Mul(A,n,B,n,b),Fill(b,b,n,T.n);
   }
   inline void Sqrt(V a[],int n,V b[]) {
      if (n==1) { b[0]=sqrt(a[0]),b[1]=0; return; }
      Sqrt(a,(n+1)>>1,b),Inv(b,n,B),Mul(a,n,B,n,B);
      Add(b,n,B,n,b),Dot_Mul(b,n,inv[2],b),Fill(b,b,n,T.n);
   }
   inline void Power(V a[],int n,ll k,V b[]) {
      Log(a,n,C),Dot_Mul(C,n,k,C),Exp(C,n,b),Fill(b,b,n,T.n);
   }
   inline V Lagrange(V a[],int n,int k) {
      Inv(a,n,A),Power(A,n,k,B); return mul(B[k-1],inv[k]);
   }
   inline void Div(V a[],int n,V b[],int m,V d[],V r[]) {
      int len=T.init_len(2*n-1); Fill(A,a,n,len),Fill(B,b,m,len);
      reverse(A,A+n),reverse(B,B+m),Inv(B,n-m+1,Y);
      Mul(A,n,Y,n,d),Fill(d,d,n-m+1,len),reverse(d,d+n-m+1);
      reverse(B,B+m),Fill(A,d,n-m+1,len);
      Mul(A,n,B,n,r),Add(a,n,r,n,r,-1),Fill(r,r,n,len);
   }
   inline void Sinh(V a[],int n,V b[]) {
      Exp(a,n,b); for (int i=0; i<n; i+=2) b[i]=0;
   }
   inline void Cosh(V a[],int n,V b[]) {
```

```
   Exp(a,n,b); for (int i=1; i<n; i+=2) b[i]=0;
}
inline void Dirichlet_Mul(V a[],int n,V b[],int m,V c[],int L) {
   int len=min((ll)n*m,L); Fill(c,c,0,L+1);
   rep(i,1,n+1) for (int j=1; j<=m && (ll)i*j<=len; j++)
      c[i*j]=add(c[i*j],mul(a[i],b[j]));
}
inline void Der_k(V a[],int n,int k,V b[]) {
   Der(a,n,b); rep(i,1,k) Der(b,n,b);
}
inline void Int_k(V a[],int n,int k,V b[]) {
   Int(a,n,b); rep(i,1,k) Int(b,n,b);
}
inline void Grow(V a[],int n,V b[]) {
   rep(i,0,n) b[i]=mul(a[i],i);
}
inline void Grow_k(V a[],int n,int k,V b[]) {
   rep(i,0,n) b[i]=mul(a[i],Pow(i,k));
}
inline void Shl(V a[],int n,int k,V b[]) {
   rep(i,k,n) b[i-k]=a[i]; Fill(b,b,n-k,n);
}
inline void Shr(V a[],int n,int k,V b[]) {
   per(i,k,n) b[i]=a[i-k]; Fill(b,b,0,k);
}
inline void To_egf(V a[],int n,V b[]) { Dot_Mul(a,fac,n,b); }
inline void To_ogf(V a[],int n,V b[]) { Dot_Mul(a,fac_inv,n,b); }
inline void Bin_Mul(V a[],int n,V b[],int m,V c[]) {
   static V A[_N<<2],B[_N<<2];
   To_ogf(a,n,A),To_ogf(b,m,B),Mul(A,n,B,m,c),To_egf(c,T.n,c);
}
inline void POW(V a[],int n,ll k,V b[],int t=0) {
   if (k*t>=n || !k) { Fill(b,b,0,n),b[0]=!k; return; }
   if (t) Shl(a,n,t,a); Power(a,n-t,k,b),Shr(b,n,k*t,b);
}
inline void Reverse(V a[],int n,V b[]) { reverse_copy(a,a+n,b); }
inline void Init_Com_Num_H_B(V a[],int n,ll k) {
   a[0]=1; rep(i,1,n) a[i]=mul(a[i-1],inv[i]*(k-i+1)%P);
}
inline void Init_Com_Num_L_B(V a[],int n,ll k) {
   a[0]=1; rep(i,1,n) a[i]=mul(a[i-1],inv[i]*(k+i)%P);
}
inline void Pre_Sum(V a[],int n,V b[]) {
   b[0]=a[0]; rep(i,1,n) b[i]=add(b[i-1],a[i]);
}
inline void Pre_Sum_k(V a[],int n,ll k,V b[]) {
   Init_Com_Num_L_B(b,n,k-1),Mul(a,n,b,n,b);
}
inline void Fly(V a[],int n,ll k,V b[]) {
   k%=P; for (int i=0,t=1; i<n; ++i,t=mul(t,k)) b[i]=mul(a[i],t);
}
inline void Crossify(V a[],int n) { Fly(a,n,-1,a); }
inline void Diff(V a[],int n,V b[]) {
   rep(i,0,n-1) b[i]=a[i+1]-a[i]; b[n-1]=-a[n-1];
}
inline void Diff_k(V a[],int n,int k,V b[]) {
   Init_Com_Num_H_B(b,k+1,k),Crossify(b,k+1),Mul(a,n,b,k+1,b),Shl(b,n+k,k,b);
}
inline void Get_all_one(V a[],int n) { rep(i,0,n) a[i]=1; }
inline void Get_exp_x(V a[],int n) { Fill(a,fac_inv,n,n); }
inline void Get_log_1_add_x(V a[],int n) {
   a[0]=0; int t=1; rep(i,1,n) a[i]=t*inv[i],t=-t;
}
inline void Init_Bell_Num(V a[],int n) {
   Get_exp_x(C,n),C[0]=0,Exp(C,n,a),To_egf(a,n,a);
}
inline void Init_Bernoulli_Num(V a[],int n) {
   Get_exp_x(C,n+1),Shl(C,n+1,1,C),Inv(C,n,a),To_egf(a,n,a);
}
inline V Get_Num_Power_Sum(ll n,int k) {
   n%=P; V ans=0; static V C[_N<<2];
   Init_Com_Num_H_B(C,k+2,k+1),Init_Bernoulli_Num(B,k+1);
   for (int i=1,t=n*(k&1?-1:1); i<=k+1; ++i,t=mul(t,-n))
      ans=add(ans,mul(C[i],B[k+1-i])*t%P);
   ans=mul(ans,inv[k+1]); return ans;
}
inline void Init_Stiriling_Num_2_H_B(V a[],int n,ll k) {
   k%=P-1; static V A[_N<<2],B[_N<<2];
   rep(i,0,n) A[i]=(i&1)?-1:1,B[i]=Pow(i,k);
   Bin_Mul(A,n,B,n,a),To_ogf(a,n,a);
}
inline void Init_Stiriling_Num_2_L(V a[],int n,int k) {
```

```
      static V A[_N<<2]; Get_exp_x(A,n+k),A[0]=0,POW(A,n+k,k,a,1);
      Dot_Mul(a,n+k,fac_inv[k],a),To_egf(a,n+k,a),Shl(a,n+k,k,a);
   }
   inline void Init_Stiriling_Num_1_L(V a[],int n,int k) {
      static V A[_N<<2]; Get_log_1_add_x(A,n+k),POW(A,n+k,k,a,1);
      Dot_Mul(a,n+k,((k&1)?-1:1)*fac_inv[k],a);
      To_egf(a,n+k,a),Crossify(a,n+k),Shl(a,n+k,k,a);
   }
   inline void Mod_p(V a[],int n) {
      rep(i,0,n) a[i]=(a[i]%P+P)%P;
   }
};
```

1.7 Poly_标准版

```
const int _N=200005; ll inv[_N<<2];
inline ll add(ll x,ll y) { x+=y; return x%P; }
inline ll mul(ll x,ll y) { return (ll)x*y%P; }
inline ll Pow(ll x,ll k) { ll ans=1; for (;k;k>>=1,x=x*x%P) if (k&1) (ans*=x)%=P; return ans; }
inline void init_inv(int n) { inv[1]=1; rep(i,2,n+1) inv[i]=mul(P-P/i,inv[P%i]); }

template <class V>
struct FT{
   int n,nn; V w[2][_N<<2],rev[_N<<2],tmp;
   inline int init_len(int _n) { for (n=1; n<=_n; n<<=1); return n; }
   inline int Init(int _n) {
      init_len(_n); if (n==nn) return n; nn=n;
      V w0=Pow(3,(P-1)/n); w[0][0]=w[1][0]=1;
      rep(i,1,n) w[0][i]=w[1][n-i]=mul(w[0][i-1],w0);
      rep(i,0,n) rev[i]=(rev[i>>1]>>1)|((i&1)*(n>>1)); return n;
   }
   void FFT(V A[],int op){
      rep(i,0,n) if (i<rev[i]) swap(A[i],A[rev[i]]);
      for (int i=1; i<n; i<<=1)
         for (int j=0,t=n/(i<<1); j<n; j+=i<<1)
            for (int k=j,l=0; k<j+i; k++,l+=t) {
               V x=A[k],y=mul(w[op][l],A[k+i]);
               A[k]=add(x,y),A[k+i]=add(x-y,P);
            }
      if (op) { tmp=inv[n]; rep(i,0,n) A[i]=mul(A[i],tmp); }
   }
};

template <class V>
struct Calculator{
   FT<V> T; V X[_N<<2],Y[_N<<2],A[_N<<2],B[_N<<2],C[_N<<2];
   inline void Fill(V a[],V b[],int n,int len) {
      if (a!=b) memcpy(a,b,sizeof(V)*n); fill(a+n,a+len,0);
   }
   inline void Add(V a[],int n,V b[],int m,V c[],int t=1) {
      n=max(n,m); rep(i,0,n) c[i]=add(a[i],t*b[i]);
   }
   inline void Dot_Mul(V a[],V b[],int len,V c[]) {
      rep(i,0,len) c[i]=mul(a[i],b[i]);
   }
   inline void Dot_Mul(V a[],int len,V v,V c[]) {
      rep(i,0,len) c[i]=mul(a[i],v);
   }
   inline void Mul(V a[],int n,V b[],int m,V c[]) {
      int len=T.Init(n+m-1); Fill(X,a,n,len),Fill(Y,b,m,len);
      T.FFT(X,0),T.FFT(Y,0),Dot_Mul(X,Y,len,c),T.FFT(c,1);
   }
   inline void Int(V a[],int n,V b[]) {
      per(i,0,n) b[i+1]=mul(a[i],inv[i+1]); b[0]=0;
   }
   inline void Der(V a[],int n,V b[]) {
      rep(i,1,n) b[i-1]=mul(a[i],i); b[n-1]=0;
   }
   inline void Inv(V a[],int n,V b[]) {
      if (n==1) { b[0]=Pow(a[0],P-2),b[1]=0; return; }
      Inv(a,(n+1)>>1,b); int len=T.Init(2*n-1);
      Fill(X,a,n,len),Fill(b,b,n,len),T.FFT(X,0),T.FFT(b,0);
      rep(i,0,len) b[i]=mul(b[i],2-mul(b[i],X[i]));
      T.FFT(b,1),Fill(b,b,n,len);
   }
   inline void Log(V a[],int n,V b[]) {
      static V A[_N<<2],B[_N<<2];
      Der(a,n,A),Inv(a,n,B),Mul(A,n,B,n,b);
      Int(b,n,b),Fill(b,b,n,T.n);
   }
   inline void Exp(V a[],int n,V b[]) {
      if (n==1) { b[0]=exp(a[0]),b[1]=0; return; }
```

```
      Exp(a,(n+1)>>1,A),Log(A,n,B),Add(a,n,B,n,B,-1);
      (B[0]+=1)%=P,Mul(A,n,B,n,b),Fill(b,b,n,T.n);
   }
   inline void Sqrt(V a[],int n,V b[]) {
      if (n==1) { b[0]=sqrt(a[0]),b[1]=0; return; }
      Sqrt(a,(n+1)>>1,b),Inv(b,n,B),Mul(a,n,B,n,B);
      Add(b,n,B,n,b),Dot_Mul(b,n,inv[2],b),Fill(b,b,n,T.n);
   }
   inline void Power(V a[],int n,ll k,V b[]) {
      Log(a,n,C),Dot_Mul(C,n,k,C),Exp(C,n,b),Fill(b,b,n,T.n);
   }
   inline V Lagrange(V a[],int n,int k) {
      Inv(a,n,A),Power(A,n,k,B); return mul(B[k-1],inv[k]);
   }
   inline void Div(V a[],int n,V b[],int m,V d[],V r[]) {
      int len=T.init_len(2*n-1); Fill(A,a,n,len),Fill(B,b,m,len);
      reverse(A,A+n),reverse(B,B+m),Inv(B,n-m+1,Y);
      Mul(A,n,Y,n,d),Fill(d,d,n-m+1,len),reverse(d,d+n-m+1);
      reverse(B,B+m),Fill(A,d,n-m+1,len);
      Mul(A,n,B,n,r),Add(a,n,r,n,r,-1),Fill(r,r,n,len);
   }
   inline void Sinh(V a[],int n,V b[]) {
      Exp(a,n,b); for (int i=0; i<n; i+=2) b[i]=0;
   }
   inline void Cosh(V a[],int n,V b[]) {
      Exp(a,n,b); for (int i=1; i<n; i+=2) b[i]=0;
   }
   inline void Dirichlet_Mul(V a[],int n,V b[],int m,V c[],int L) {
      int len=min((ll)n*m,L); Fill(c,c,0,L+1);
      rep(i,1,n+1) for (int j=1; j<=m && (ll)i*j<=len; j++)
         c[i*j]=add(c[i*i],mul(a[i],b[j]));
   }
};
```

1.8 一维 FFT_单模式串_模式串带通配符匹配

```
struct cp { double x, y; };
inline cp operator + (cp &a, cp &b) { return cp{ a.x + b.x,a.y + b.y }; }
inline cp operator - (cp &a, cp &b) { return cp{ a.x - b.x,a.y - b.y }; }
inline cp operator * (cp &a, cp &b) { return cp{ a.x*b.x - a.y*b.y,a.x*b.y + a.y*b.x }; }
inline cp get(double x) { return cp{ cos(x),sin(x) }; }
inline ostream& operator<<(ostream &out, const cp &t) { out << "(" << t.x << "," << t.y << ")"; return out; }

const int _M = 1 << 18, _N = N;
struct FT {
   cp tmp[_M * 2 + 5], aa[_M], bb[_M];
   void FFT(cp *a, int n, int op) {
      for (int i = (n >> 1), j = 1; j < n; j++) {
         if (i < j) swap(a[i], a[j]);
         int k; for (k = (n >> 1); k&i; i ^= k, k >>= 1); i ^= k;
      }
      for (int m = 2; m <= n; m <<= 1) {
         cp w = get(2 * PI*op / m); tmp[0] = cp{ 1,0 };
         for (int j = 1; j < (m >> 1); j++) tmp[j] = tmp[j - 1] * w;
         for (int i = 0; i < n; i += m)
            for (int j = i; j < i + (m >> 1); j++) {
               cp u = a[j], v = a[j + (m >> 1)] * tmp[j - i];
               a[j] = u + v, a[j + (m >> 1)] = u - v;
            }
      }
      if (op == -1) rep(i, 0, n) a[i] = cp{ a[i].x / n,a[i].y / n };
   }
   void In(cp p[], int len, cp a[], int n) {
      rep(i, 0, len) p[i] = i < n ? a[i] : cp{ 0,0 };
   }
   void Out(int a[], int n, cp p[], int len) {
      rep(i, 0, n) a[i] = (int)(p[i].x + eps);
   }
   void Shift(int a[], int n, int p) { rep(i, n, n + p) a[i - n] = a[i]; }
   void Multiply(cp A[], int n, cp B[], int m, int C[], int &len, int op = 0) {
      if (op) reverse(A, A + n);
      len = 1; while (len < n + m - 1) len <<= 1;
      In(aa, len, A, n), In(bb, len, B, m), FFT(aa, len, 1), FFT(bb, len, 1);
      rep(i, 0, len) aa[i] = aa[i] * bb[i];
      FFT(aa, len, -1), Out(C, n + m - 1, aa, len);
      if (op) Shift(C, n - 1, m), len = m, reverse(A, A + n);
   }
};

void Build(cp A[], int n, char s[], int M, int op, int cc = 'a') {
   rep(i, 0, n) A[i] = (s[i] == '?') ? cp{ 0,0 } : get(2 * PI / M*(s[i] - cc)*op);
}
```

```
int n, m, len, tot = 0, tt; char s[N], t[N]; FT T; cp A[_M], B[_M]; int C[_M]; vi ans;

int main() {
   //file_put();

   scanf("%s%s", s, t), n = strlen(s), m = strlen(t);
   rep(i, 0, m) tot += (t[i] != '?');
   Build(A, n, s, 26, 1), Build(B, m, t, 26, -1);
   T.Multiply(B, m, A, n, C, len, 1);
   //debug_arr(C,len-1);
   rep(i, 0, n - m + 1) if (C[i] >= tot) ans.pb(i);
   printf("%d\n", tt = ans.size());
   rep(i, 0, tt) printf("%d\n", ans[i]);

   return 0;
}
```

1.9 二维 FFT_单模式串_模式串带通配符匹配

```
struct cp { double x, y; };
inline cp operator + (cp &a, cp &b) { return cp{ a.x + b.x,a.y + b.y }; }
inline cp operator - (cp &a, cp &b) { return cp{ a.x - b.x,a.y - b.y }; }
inline cp operator * (cp &a, cp &b) { return cp{ a.x*b.x - a.y*b.y,a.x*b.y + a.y*b.x }; }
inline cp get(double x) { return cp{ cos(x),sin(x) }; }
inline ostream& operator<<(ostream &out, const cp &t) { out << "(" << t.x << "," << t.y << ")"; return out; }

const int _M = 2048, _N = N;
template <class V>
struct FT {
   cp tmp[_M * 2 + 5], aa[_M][_M], bb[_M][_M];
   void FFT(cp *a, int n, int op) {
      for (int i = (n >> 1), j = 1; j < n; j++) {
         if (i < j) swap(a[i], a[j]);
         int k; for (k = (n >> 1); k&i; i ^= k, k >>= 1); i ^= k;
      }
      for (int m = 2; m <= n; m <<= 1) {
         cp w = get(2 * PI*op / m); tmp[0] = cp{ 1,0 };
         for (int j = 1; j < (m >> 1); j++) tmp[j] = tmp[j - 1] * w;
         for (int i = 0; i < n; i += m)
            for (int j = i; j < i + (m >> 1); j++) {
               cp u = a[j], v = a[j + (m >> 1)] * tmp[j - i];
               a[j] = u + v, a[j + (m >> 1)] = u - v;
            }
      }
      if (op == -1) rep(i, 0, n) a[i] = cp{ a[i].x / n,a[i].y / n };
   }
   void FFT(cp a[][_M], int n, int op) { rep(i, 0, n) FFT(a[i], n, op); }
   template <class T>
   void Transpose(T a[][_M], int n) {
      rep(i, 0, n) rep(j, 0, i) swap(a[i][j], a[j][i]);
   }
   void Reverse(V a[][_M], int n, int m) {
      rep(i, 0, (n - 1 >> 1) + 1) rep(j, 0, m) swap(a[i][j], a[n - 1 - i][j]);
      rep(i, 0, n) rep(j, 0, (m - 1 >> 1) + 1) swap(a[i][j], a[i][m - 1 - j]);
   }
   void Shift(int a[][_M], int n, int m, int p, int q) {
      rep(i, n, n + p) rep(j, m, m + q) a[i - n][j - m] = a[i][j];
   }
   void In(cp p[][_M], int len, V a[][_M], int n, int m) {
      rep(i, 0, len) rep(j, 0, len) p[i][j] = i < n&&j < m ? a[i][j] : cp{ 0,0 };
   }
   void Out(int a[][_M], int n, int m, cp p[][_M], int len) {
      rep(i, 0, n) rep(j, 0, m) a[i][j] = (int)(p[i][j].x + eps);
   }
   void Multiply(V A[][_M], int n, V B[][_M], int m, int C[][_M], int &len, int op = 0) {
      if (op) Reverse(A, n, n);
      len = 1; while (len < n + m - 1) len <<= 1;
      In(aa, len, A, n, n), In(bb, len, B, m, m), FFT(aa, len, 1), FFT(bb, len, 1);
      Transpose(aa, len), Transpose(bb, len), FFT(aa, len, 1), FFT(bb, len, 1);
      rep(i, 0, len) rep(j, 0, len) aa[i][j] = aa[i][j] * bb[i][j];
      FFT(aa, len, -1), Transpose(aa, len), FFT(aa, len, -1), Out(C, len, len, aa, len);
      if (op) Shift(C, n - 1, n - 1, m, m), len = m, Reverse(A, n, n);
   }
};

void Build(cp A[][_M], int n, int m, char s[][405], int M, int op, int cc = 'a') {
   rep(i, 0, n) rep(j, 0, m) A[i][j] = (s[i][j] == '?') ? cp{ 0,0 } : get(2 * PI / M*(s[i][j] - cc)*op);
}

int n1, n2, m1, m2, nn, mm, len, tot = 0; char s[405][405], t[405][405]; FT<cp> T; cp A[_M][_M], B[_M][_M]; int C[_M][_M];
```

```
int main() {
   //file_put();

   scanf("%d%d", &n1, &m1);
   rep(i, 0, n1) scanf("%s", s[i]);
   scanf("%d%d", &n2, &m2), nn = n1 + n2, mm = m1 + m2;
   rep(i, 0, n2) scanf("%s", t[i]);
   rep(i, 0, n2) rep(j, 0, m2) tot += (t[i][j] != '?');
   Build(A, n1, m1, s, 26, 1), Build(B, n2, m2, t, 26, -1);
   rep(i, 0, nn) rep(j, 0, mm) {
      if (i < n1 && j < m1) continue;
      A[i][j] = A[i%n1][j%m1];
   }
   //debug_arr2(A,nn-1,mm-1);
   //debug_arr2(B,n2-1,m2-1);
   T.Multiply(B, max(n2, m2), A, max(nn, mm), C, len, 1);
   //debug_arr2(C,len-1,len-1);
   rep(i, 0, n1) {
      rep(j, 0, m1) printf("%c", "01"[C[i][j] >= tot]);
      printf("\n");
   }

   return 0;
}
```

1.10 扩展卢卡斯

```
namespace exlucas {
   const int N = 1e6;
   typedef long long ll;
   ll n, m, p;
   inline ll power(ll a, ll b, const ll p = LLONG_MAX) {
      ll ans = 1;
      while (b) {
         if (b & 1)
            ans = ans * a % p;
         a = a * a % p;
         b >>= 1;
      }
      return ans;
   }
   ll fac(const ll n, const ll p, const ll pk) {
      if (!n)
         return 1;
      ll ans = 1;
      for (int i = 1; i < pk; i++)
         if (i % p)
            ans = ans * i % pk;
      ans = power(ans, n / pk, pk);
      for (int i = 1; i <= n % pk; i++)
         if (i % p)
            ans = ans * i % pk;
      return ans * fac(n / p, p, pk) % pk;
   }
   ll exgcd(const ll a, const ll b, ll &x, ll &y) {
      if (!b) {
         x = 1, y = 0;
         return a;
      }
      ll xx, yy, g = exgcd(b, a % b, xx, yy);
      x = yy;
      y = xx - a / b * yy;
      return g;
   }
   ll inv(const ll a, const ll p) {
      ll x, y;
      exgcd(a, p, x, y);
      return (x % p + p) % p;
   }
   ll C(const ll n, const ll m, const ll p, const ll pk) {
      if (n < m)
         return 0;
      ll f1 = fac(n, p, pk), f2 = fac(m, p, pk), f3 = fac(n - m, p, pk), cnt = 0;
      for (ll i = n; i; i /= p)
         cnt += i / p;
      for (ll i = m; i; i /= p)
         cnt -= i / p;
      for (ll i = n - m; i; i /= p)
         cnt -= i / p;
      return f1 * inv(f2, pk) % pk * inv(f3, pk) % pk * power(p, cnt, pk) % pk;
   }
   ll a[N], c[N];
```

```
   int cnt;
   inline ll CRT() {
      ll M = 1, ans = 0;
      for (int i = 0; i < cnt; i++)
         M *= c[i];
      for (int i = 0; i < cnt; i++)
         ans = (ans + a[i] * (M / c[i]) % M * inv(M / c[i], c[i]) % M) % M;
      return ans;
   }
   ll exlucas(const ll n, const ll m, ll p) {
      ll tmp = sqrt(p);
      for (int i = 2; p > 1 && i <= tmp; i++) {
         ll tmp = 1;
         while (p % i == 0)
            p /= i, tmp *= i;
         if (tmp > 1)
            a[cnt] = C(n, m, i, tmp), c[cnt++] = tmp;
      }
      if (p > 1)
         a[cnt] = C(n, m, p, p), c[cnt++] = p;
      return CRT();
   }
   int work() {
      ios::sync_with_stdio(false);
      cin >> n >> m >> p;
      cout << exlucas(n, m, p);
      return 0;
   }
}
```

1.11 拟阵交

```
#include<bits/stdc++.h>
#define MAXN 65
#define MAXM 6005
#define INF 1000000000
#define MOD 1000000007
#define F first
#define S second
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
int color[MAXM];
ll val[MAXM];
int n,m,tot,tot2;
struct LinearMatroid{
    ll basis[62];
    void clear() { memset(basis,0,sizeof(basis));}
    void add(ll x) {
        for(int j=60;j>=0;j--) {
            if(!(x&(1LL<<j))) continue;
            if(!basis[j]) {
                basis[j]=x;
                return;
            }
            else x^=basis[j];
        }
    }
    bool test(ll x) {
        for(int j=60;j>=0;j--) {
            if(!(x&(1LL<<j))) continue;
            if(!basis[j]) return true; else x^=basis[j];
        }
        return false;
    }
};

struct ColorfulMatroid {
    int cnt[125];
    void clear() { memset(cnt,0,sizeof(cnt)); }
    void add(int x) { cnt[x]++; }
    bool test(int x) { return (cnt[x]==0); }
};

template <typename MT1, typename MT2>
struct MatroidIntersection {
    int n;
    MatroidIntersection(int _n):n(_n){}
    int pre[MAXM],id[MAXM];
    bool vis[MAXM],sink[MAXM],has[MAXM];
    queue<int> que;
    void clear_all() {
```

```
        memset(vis,false,sizeof(vis));
        memset(sink,false,sizeof(sink));
        memset(pre,0,sizeof(pre));
        while(que.size()) que.pop();
    }
    vector<int> getcur() {
        vector<int> ret;
        for(int i=1;i<=n;i++) if(has[i]) ret.push_back(i);
        return ret;
    }
    void enqueue(int v,int p) {
        vis[v]=true; pre[v]=p;
        que.push(v);
    }
    vector<int> run() {
        MT1 mt1;  MT2 mt2;
        memset(has,false,sizeof(has));
        while(true) {
            vector<int> cur=getcur();
            int cnt=0;
            for(int i=1;i<=n;i++) if(has[i]) id[i]=cnt++;
            MT1 allmt1; MT2 allmt2; allmt1.clear(); allmt2.clear();
            vector<MT1> vmt1(cur.size()); vector<MT2> vmt2(cur.size());
            for(auto &x:vmt1) x.clear(); for(auto &x:vmt2) x.clear();
            clear_all();
            for(auto x:cur) allmt1.add(val[x]),allmt2.add(color[x]);
            for(int i=0;i<(int)cur.size();i++)
                for(int j=0;j<(int)cur.size();j++) {
                    if(i==j) continue;
                    vmt1[i].add(val[cur[j]]);
                    vmt2[i].add(color[cur[j]]);
                }
            for(int i=1;i<=n;i++) {
                if(has[i]) continue;
                if(allmt1.test(val[i])) {que.push(i); vis[i]=true;}
            }
            for(int i=1;i<=n;i++) {
                if(has[i]) continue;
                if(allmt2.test(color[i])) sink[i]=true;
            }
            int last=-1;
            while(que.size()) {
                int v=que.front(); que.pop();
                if(sink[v]) {last=v; break;}
                for(int i=1;i<=n;i++) {
                    if(vis[i]) continue;
                    if(has[i]==has[v]) continue;
                    if(has[v]) {
                        if(vmt1[id[v]].test(val[i])) enqueue(i,v);
                    }
                    else {
                        if(vmt2[id[i]].test(color[v])) enqueue(i,v);
                    }
                }
            }
            if(last==-1) return cur;
            while(last) {
                has[last]^=1;
                last=pre[last];
            }
        }
    }
};
//Pick Your Own Nim
//In real cases, Linear Matroid Need Optimization to Pass
int main() {
    scanf("%d",&n);
    for(int i=0;i<n;i++) {
        ll x;
        scanf("%lld",&x);
        val[++tot]=x; color[tot]=++tot2;
    }
    scanf("%d",&m);
    for(int i=0;i<m;i++) {
        int k;
        scanf("%d",&k);
        tot2++;
        for(int j=0;j<k;j++) {
            ll x;
            scanf("%lld",&x);
            val[++tot]=x; color[tot]=tot2;
        }
```

```
    }
    MatroidIntersection<LinearMatroid,ColorfulMatroid> matint(tot);
    vector<int> res=matint.run();
    if(res.size()<n+m) {puts("-1"); return 0;}
    else {
        vector<ll> ans;
        int last=n;
        for(auto x:res) {
            if(color[x]>last) {
                ans.push_back(val[x]);
                last=color[x];
            }
        }
        for(auto x:ans) printf("%lld\n",x);
    }
    return 0;
}
```

1.12 拟阵交_带权

```
#pragma GCC optimize(3)
#include<bits/stdc++.h>
#define MAXN 85
#define MAXM 205
#define INF 1000000000
#define MOD 1000000007
#define F first
#define S second
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
int c[MAXN],k[MAXN],color[MAXM],u[MAXM],v[MAXM],w[MAXM],cost[MAXM];
ll val[MAXM];
int T,n,m,tot,tot2;
struct LinearMatroid
{
    ll basis[62];
    void clear()
    {
        memset(basis,0,sizeof(basis));
    }
    void add(ll x)
    {
        for(int j=60;j>=0;j--)
        {
            if(!(x&(1LL<<j))) continue;
            if(!basis[j])
            {
                basis[j]=x;
                return;
            }
            else x^=basis[j];
        }
    }
    bool test(ll x)
    {
        for(int j=60;j>=0;j--)
        {
            if(!(x&(1LL<<j))) continue;
            if(!basis[j]) return true; else x^=basis[j];
        }
        return false;
    }
};
struct ColorfulMatroid
{
    int cnt[125];
    void clear()
    {
        memset(cnt,0,sizeof(cnt));
    }
    void add(int x)
    {
        cnt[x]++;
    }
    bool test(int x)
    {
        return (cnt[x]==0);
    }
};

struct GraphMatroid
```

```
{
    vector<int> G[MAXN];
    bool vis[MAXN];
    bool exist[MAXN];
    void dfs(int v)
    {
        vis[v]=true;
        for(auto to:G[v]) if(!vis[to]) dfs(to);
    }
    bool test(vector<int> &vec)
    {
        for(int i=1;i<=n+1;i++) G[i].clear();
        memset(vis,false,sizeof(vis));
        memset(exist,true,sizeof(exist));
        for(auto x:vec) exist[x]=false;
        for(int i=1;i<=tot;i++)
        {
            if(exist[i])
            {
                G[u[i]].push_back(v[i]);
                G[v[i]].push_back(u[i]);
            }
        }
        dfs(1);
        for(int i=1;i<=n+1;i++) if(!vis[i]) return false;
        return true;
    }
};

struct PartitionMatroid
{
    int cnt[125];
    bool test(vector<int> &vec)
    {
        memset(cnt,0,sizeof(cnt));
        for(auto x:vec) cnt[color[x]]++;
        for(int i=1;i<=m;i++) if(cnt[i]>c[i]-k[i]) return false;
        return true;
    }
};


template <typename MT1, typename MT2>
struct MatroidIntersection
{
    int n,S,T;
    MatroidIntersection(int _n):n(_n){}
    int pre[MAXM],id[MAXM],d[MAXM];
    bool inque[MAXM],sink[MAXM],has[MAXM];
    vector<int> g[MAXN];
    queue<int> que;
    void clear_all()
    {
        for(int i=1;i<=n+2;i++)
        {
            inque[i]=false;
            sink[i]=false;
            pre[i]=0;
            d[i]=-INF;
            if(has[i]) cost[i]=w[i]; else cost[i]=-w[i];
            g[i].clear();
        }
        while(que.size()) que.pop();
    }
    void add_edge(int u,int v)
    {
        g[u].push_back(v);
    }
    vector<int> getcur()
    {
        vector<int> ret;
        for(int i=1;i<=n;i++) if(has[i]) ret.push_back(i);
        return ret;
    }
    void enqueue(int v,int p)
    {
        pre[v]=p;
        if(!inque[v])
        {
            inque[v]=true;
            que.push(v);
        }
```

```
    }
    pair<vector<int>,ll> run()
    {
        ll ans=0;
        MT1 mt1;  MT2 mt2;
        memset(has,false,sizeof(has));
        S=n+1; T=n+2;
        while(true)
        {
            clear_all();
            for(int i=1;i<=n;i++)
            {
                if(!has[i])
                {
                    cost[i]=w[i];
                    has[i]^=1;
                    vector<int> tmp=getcur();
                    if(mt1.test(tmp)) add_edge(S,i);
                    if(mt2.test(tmp)) add_edge(i,T);
                    has[i]^=1;
                }
                else cost[i]=-w[i];
            }
            for(int i=1;i<=n;i++)
            {
                if(!has[i])
                {
                    for(int j=1;j<=n;j++)
                    {
                        if(has[j])
                        {
                            has[i]^=1; has[j]^=1;
                            vector<int> tmp=getcur();
                            if(mt1.test(tmp)) add_edge(j,i);
                            if(mt2.test(tmp)) add_edge(i,j);
                            has[i]^=1; has[j]^=1;
                        }
                    }
                }
            }
            d[S]=0; que.push(S); inque[S]=true;
            cost[S]=cost[T]=0;
            int counter=0;
            while(que.size())
            {
                counter++;
                int u=que.front(); que.pop();
                for(auto to:g[u])
                    if(d[to]<d[u]+cost[to])
                    {
                        d[to]=d[u]+cost[to];
                        enqueue(to,u);
                    }
                inque[u]=false;
            }
            if(!pre[T]) return make_pair(getcur(),ans);
            ans+=d[T];
            int last=pre[T];
            while(last!=S)
            {
                has[last]^=1;
                last=pre[last];
            }
        }
    }
};
//hdu 6636 Milk Candy
int main()
{
    scanf("%d",&T);
    while(T--)
    {
        tot=0;
        scanf("%d%d",&n,&m);
        int sum=0;
        ll ans=0;
        for(int i=1;i<=m;i++)
        {
            scanf("%d%d",&c[i],&k[i]);
            sum+=c[i]-k[i];
            for(int j=1;j<=c[i];j++)
            {
```

```
                int l,r,cost;
                scanf("%d%d%d",&l,&r,&cost);
                color[++tot]=i; u[tot]=l; v[tot]=r+1; w[tot]=cost;
                ans+=cost;
            }
        }
        MatroidIntersection<GraphMatroid,PartitionMatroid> matint(tot);
        auto res=matint.run();
        GraphMatroid gm; PartitionMatroid pm;
        if((int)res.F.size()!=sum||!gm.test(res.F)||!pm.test(res.F)) puts("-1"); else printf("%lld\n",ans-res.S);
    }
    return 0;
}
```

2 String

2.1 PAM_优化转移_偶回文切割方案数

```
const int M = 26;
struct PAM {
   int s[N], len[N], next[N][M], fail[N], cnt[N], dep[N], id[N], no[N], last, n, p, cur, now; int df[N], slink[N], pre[N]; ll
    ans, dp[N], res[N];
   inline int new_node(int _l) { mem(next[p], 0); cnt[p] = dep[p] = 0, len[p] = _l; return p++; }
   inline void Init() { new_node(p = 0), new_node(s[0] = -1), fail[last = n = 0] = 1; }
   inline int get_fail(int x) { for (; s[n - len[x] - 1] != s[n]; x = fail[x]); return x; }
   inline void I(int c) {
      c -= 'a', s[++n] = c, cur = get_fail(last);
      if (!next[cur][c]) {
         now = new_node(len[cur] + 2);
         fail[now] = next[get_fail(fail[cur])][c];
         next[cur][c] = now;
         dep[now] = dep[fail[now]] + 1; //...
      }
      last = next[cur][c], cnt[last]++; id[n] = last, no[last] = n;
      df[last] = len[last] - len[fail[last]];
      slink[last] = (df[last] == df[fail[last]]) ? slink[fail[last]] : fail[last]; //...
   }
   inline void Insert(char s[], int op = 0, int _n = 0) {
      if (!_n) _n = strlen(s);  if (!op) rep(i, 0, _n) I(s[i]); else per(i, 0, _n) I(s[i]);
   }
   inline void count() { per(i, 0, p) cnt[fail[i]] += cnt[i]; }
   inline void Q() {
      rep(i, 0, n + 2) dp[i] = 0; dp[0] = 1;
      rep(i, 1, n + 1) for (int t = id[i]; len[t] > 0; t = slink[t]) {
         res[t] = dp[i - len[slink[t]] - df[t]];
         if (df[t] == df[fail[t]]) (res[t] += res[fail[t]]) %= P;
         if (!odd(i)) (dp[i] += res[t]) %= P;
      }
      printf("%I64d\n", dp[n]);
   }
};

/*【题目】

codeforces 932G 求原串偶数段的段式回文切割的方案数目首尾间隔取字母构成新串，转化为求偶回文切割方案数（每段长度为偶数）


*/
```

2.2 PAM_优化转移_最小回文切割段数

```
const int M = 26;
struct PAM {
   int s[N], len[N], next[N][M], fail[N], cnt[N], dep[N], id[N], no[N], last, n, p, cur, now; int df[N], slink[N], dp[N], res
   [N]; ll ans;
   inline int new_node(int _l) { mem(next[p], 0); cnt[p] = dep[p] = 0, len[p] = _l; return p++; }
   inline void Init() { new_node(p = 0), new_node(s[0] = -1), fail[last = n = 0] = 1; }
   inline int get_fail(int x) { for (; s[n - len[x] - 1] != s[n]; x = fail[x]); return x; }
   inline void I(int c) {
      c -= 'a', s[++n] = c, cur = get_fail(last);
      if (!next[cur][c]) {
         now = new_node(len[cur] + 2);
         fail[now] = next[get_fail(fail[cur])][c];
         next[cur][c] = now;
         dep[now] = dep[fail[now]] + 1; //...
      }
      last = next[cur][c], cnt[last]++; id[n] = last, no[last] = n;
      df[last] = len[last] - len[fail[last]];
      slink[last] = (df[last] == df[fail[last]]) ? slink[fail[last]] : fail[last]; //...
   }
   inline void Insert(char s[], int op = 0, int _n = 0) {
```

```
      if (!_n) _n = strlen(s);  if (!op) rep(i, 0, _n) I(s[i]); else per(i, 0, _n) I(s[i]);
   }
   inline void count() { per(i, 0, p) cnt[fail[i]] += cnt[i]; }
   inline int Q() {
      rep(i, 0, n + 2) dp[i] = oo; dp[0] = 0;
      rep(i, 1, n + 1) {
         for (int t = id[i]; len[t] > 0; t = slink[t]) {
            res[t] = dp[i - len[slink[t]] - df[t]];
            if (df[t] == df[fail[t]]) res[t] = min(res[t], res[fail[t]]);
            dp[i] = min(dp[i], res[t] + 1);
         }
      }
      return dp[n];
   }
};

/*【题目】求给定串的最小回文切割段数【注意】如果要求段的长度为奇数或偶数，在


1.dp赋值语句前限制，请不要乱改自动机[]如果要求段数为奇数或偶数，
2.和开两维dpresdp[i表示段数为偶数的答案][0]

*/
```

2.3 PAM__单串__支持双端插入

```
const int M = 26;
struct PAM {
   int ss[2 * N], *s, len[N], next[N][M], fail[N], cnt[N], dep[N], id[N], no[N], last[2], n[2], p, cur, now, t; ll sum_cnt,
   ans;
   inline int new_node(int _l) { mem(next[p], 0); cnt[p] = dep[p] = 0, len[p] = _l; return p++; }
   inline void Init() { s = ss + N; new_node(p = 0), new_node(s[0] = s[1] = -1), fail[last[0] = last[1] = n[1] = 0] = n[0] =
   1; /*...*/ sum_cnt = 0; }
   inline int get_fail(int x, int op) { int t = op * 2 - 1; for (; s[n[op] - t*(len[x] + 1)] != s[n[op]]; x = fail[x]);
   return x; }
   inline void I(int c, int op) {
      c -= 'a', t = op * 2 - 1, s[n[op] += t] = c, s[n[op] + t] = -1, cur = get_fail(last[op], op);
      if (!next[cur][c]) {
         now = new_node(len[cur] + 2);
         fail[now] = next[get_fail(fail[cur], op)][c];
         next[cur][c] = now;
         dep[now] = dep[fail[now]] + 1; //...
      }
      last[op] = next[cur][c], cnt[last[op]]++;
      if (len[last[op]] == n[1] - n[0] + 1) last[op ^ 1] = last[op];
      id[n[op]] = last[op]; no[last[op]] = n[op] + (len[last[op]] - 1) * !op; //...
      sum_cnt += dep[last[op]];
   }
   inline void Insert(char s[], int back = 1, int op = 0, int _n = 0) {
      if (!_n) _n = strlen(s);  if (!op) rep(i, 0, _n) I(s[i], back); else per(i, 0, _n) I(s[i], back);
   }
   inline void count() { per(i, 0, p) cnt[fail[i]] += cnt[i]; }
   inline void Q() { /*count();*/ }
};

/*注：

 1) 支持在线维护本质不同回文串个数 p-2 , 所有回文串个数 sum_cnt , 每个回文串出现的一次起点下标 ( 不保证最左最右 )

2) 注意 id[x] 是不准确的

*/

/*

6
1 a
1 b
2 a
2 c
3
4
8
1 a
2 a
2 a
1 a
3
1 b
3
4
```

```
4
5
4
5
11

1 左 2 右
3 p-2
4 sum_cnt

*/
```

2.4 PAM_单串_支持撤销_单加 log

```
const int M = 26;
struct PAM {
   int s[N], len[N], next[N][M], fail[N], cnt[N], id[N], no[N], pre[N], qlink[N], dep[N], last, n, p, cur, now; ll ans;
   inline int new_node(int _l) { mem(next[p], 0); cnt[p] = dep[p] = 0, len[p] = _l; qlink[p] = 0; return p++; }
   inline void Init() { new_node(p = 0), new_node(s[0] = -1), fail[last = n = 0] = 1; /* ... */ }
   inline bool ok(int x, int y, int d = 0) { return s[n - len[x] - d] == s[n - len[y]]; }
   inline int get_fail(int x) { for (; !ok(x, 0, 1); x = qlink[x]) if (ok(fail[x], 0, 1)) return fail[x]; return x; }
   inline void I(int c) {
      c -= 'a', s[++n] = c, cur = get_fail(last);
      if (!next[cur][c]) {
         now = new_node(len[cur] + 2);
         fail[now] = next[get_fail(fail[cur])][c];
         next[cur][c] = now; pre[now] = cur;
         dep[now] = dep[fail[now]] + 1; //...
         if (len[now] > 1) qlink[now] = ok(fail[now], fail[fail[now]]) ? qlink[fail[now]] : fail[fail[now]];
      }
      last = next[cur][c], cnt[last]++; id[n] = last, no[last] = n; //...
   }
   inline void D() { if (p <= 1) return; if (!(--cnt[last])) next[pre[last]][s[n]] = 0, --p; last = id[--n]; }
   inline void Insert(char s[], int op = 0, int _n = 0) {
      if (!_n) _n = strlen(s);  if (!op) rep(i, 0, _n) I(s[i]); else per(i, 0, _n) I(s[i]);
   }
   inline void count() { per(i, 0, p) cnt[fail[i]] += cnt[i]; }
   inline void Q() { /*count();*/ }
};

/*注：



1) 此模板如果不使用回退操作，总复杂度仍为线性，单次插入不超过 log(n) ，回退操作总是常数

2) 若有回退操作，总复杂度退化；使用可持久化线段树优化的 dLink 链接，单次插入可降为 log 字符集

*/
```

2.5 PAM_单串_支持撤销_单加可退化

```
const int M = 26;
struct PAM {
   int s[N], len[N], next[N][M], fail[N], cnt[N], dep[N], id[N], no[N], pre[N], last, n, p, cur, now; ll ans;
   inline int new_node(int _l) { mem(next[p], 0); cnt[p] = dep[p] = 0, len[p] = _l; return p++; }
   inline void Init() { new_node(p = 0), new_node(s[0] = -1), fail[last = n = 0] = 1; }
   inline int get_fail(int x) { for (; s[n - len[x] - 1] != s[n]; x = fail[x]); return x; }
   inline void I(int c) {
      c -= 'a', s[++n] = c, cur = get_fail(last);
      if (!next[cur][c]) {
         now = new_node(len[cur] + 2);
         fail[now] = next[get_fail(fail[cur])][c];
         next[cur][c] = now; pre[now] = cur;
         dep[now] = dep[fail[now]] + 1; //...
      }
      last = next[cur][c], cnt[last]++; id[n] = last, no[last] = n; //...
   }
   inline void D() { if (p <= 1) return; if (!(--cnt[last])) next[pre[last]][s[n]] = 0, --p; last = id[--n]; }
   inline void Insert(char s[], int op = 0, int _n = 0) {
      if (!_n) _n = strlen(s);  if (!op) rep(i, 0, _n) I(s[i]); else per(i, 0, _n) I(s[i]);
   }
   inline void count() { per(i, 0, p) cnt[fail[i]] += cnt[i]; }
   inline void Q() { /*count();*/ }
};
```

2.6 PAM_多串模式

```
const int M = 26;
struct PAM {
   int s[N], len[N], next[N][M], fail[N], cnt[N][11], dep[N], id[N], no[N], last, n, p, cur, now, str_cnt, d0; ll ans;
```

```
   inline int new_node(int _l) { mem(next[p], 0); mem(cnt[p], 0); dep[p] = 0, len[p] = _l; return p++; }
   inline void Init() { new_node(p = 0), new_node(s[0] = -1), fail[last = n = 0] = 1; str_cnt = 0; d0 = -1; }
   inline int get_fail(int x) { for (; s[n - len[x] - 1] != s[n]; x = fail[x]); return x; }
   inline void I(int c) {
      if (c < 0) { s[++n] = c; last = 1; return; }
      c -= 'a', s[++n] = c, cur = get_fail(last);
      if (!next[cur][c]) {
         now = new_node(len[cur] + 2);
         fail[now] = next[get_fail(fail[cur])][c];
         next[cur][c] = now;
         dep[now] = dep[fail[now]] + 1; //...
      }
      last = next[cur][c], cnt[last][str_cnt]++; id[n] = last, no[last] = n; //...
   }
   inline void Insert(char s[], int op = 0, int _n = 0) {
      if (str_cnt) I(--d0); if (!_n) _n = strlen(s);
      if (!op) rep(i, 0, _n) I(s[i]); else per(i, 0, _n) I(s[i]); ++str_cnt;
   }
   inline void count() { per(i, 0, p) rep(j, 0, str_cnt) cnt[fail[i]][j] += cnt[i][j]; }
   inline ll Q() { count(); /* ... */ }
};
```

2.7 PAM_标准版本

```
const int M = 26;
struct PAM {
   int s[N], len[N], next[N][M], fail[N], cnt[N], dep[N], id[N], no[N], last, n, p, cur, now; ll ans;
   inline int new_node(int _l) { mem(next[p], 0); cnt[p] = dep[p] = 0, len[p] = _l; return p++; }
   inline void Init() { new_node(p = 0), new_node(s[0] = -1), fail[last = n = 0] = 1; }
   inline int get_fail(int x) { for (; s[n - len[x] - 1] != s[n]; x = fail[x]); return x; }
   inline void I(int c) {
      c -= 'a', s[++n] = c, cur = get_fail(last);
      if (!next[cur][c]) {
         now = new_node(len[cur] + 2);
         fail[now] = next[get_fail(fail[cur])][c];
         next[cur][c] = now;
         dep[now] = dep[fail[now]] + 1; //...
      }
      last = next[cur][c], cnt[last]++; id[n] = last, no[last] = n; //...
   }
   inline void Insert(char s[], int op = 0, int _n = 0) {
      if (!_n) _n = strlen(s);  if (!op) rep(i, 0, _n) I(s[i]); else per(i, 0, _n) I(s[i]);
   }
   inline void count() { per(i, 0, p) cnt[fail[i]] += cnt[i]; }
   inline void Q() { /*count();*/ }
};
```

2.8 PAM_空间压缩_单链表

```
struct Link {
   struct node { int id, x; node *nxt; }; node *head, *p;  Link() { head = NULL; }
   void I(int id, int x) { head = new node{ id,x,head }; }
   int F(int id) { for (p = head; p; p = p->nxt) if (p->id == id) return p->x; return 0; }
};

struct PAM {
   int s[N], len[N], fail[N], cnt[N][3], dep[N], id[N], no[N], last, n, p, cur, now, str_cnt, d0, tt; Link next[N]; ll ans;
   inline int new_node(int _l) { mem(cnt[p], 0); dep[p] = 0, len[p] = _l; return p++; }
   inline void Init() { new_node(p = 0), new_node(s[0] = -1), fail[last = n = 0] = 1; str_cnt = 0; d0 = -1; }
   inline int get_fail(int x) { for (; s[n - len[x] - 1] != s[n]; x = fail[x]); return x; }
   inline void I(int c) {
      if (c < 0) { s[++n] = c; last = 1; return; }
      c -= 'a', s[++n] = c, cur = get_fail(last); tt = now = 0;
      if (!(tt = next[cur].F(c))) {
         now = new_node(len[cur] + 2);
         fail[now] = next[get_fail(fail[cur])].F(c);
         next[cur].I(c, now);
         dep[now] = dep[fail[now]] + 1; //...
      }
      last = tt + now, cnt[last][str_cnt]++; id[n] = last, no[last] = n; //...
   }
   inline void Insert(char s[], int op = 0, int _n = 0) {
      if (str_cnt) I(--d0); if (!_n) _n = strlen(s);
      if (!op) rep(i, 0, _n) I(s[i]); else per(i, 0, _n) I(s[i]); ++str_cnt;
   }
   inline void count() { per(i, 0, p) rep(j, 0, str_cnt) cnt[fail[i]][j] += cnt[i][j]; }
   inline ll Q() { count(); /* ... */ }
};
```

2.9 SAM_多串模式_串运行_树上合并_map

```
struct SAM {
   static const int M = 26; ll dp[N << 1], ans; map<int, int> SS[N << 1];
   int go[N << 1][M], pre[N << 1], step[N << 1], rr[N << 1], temp[N << 1], toop[N << 1], id[N << 1], dep[N << 1], str_cnt,
   cnt, S, T, n, p, q, nq;
   inline int h(int c) { return c - 'a'; }
   inline int new_node(int _s, int c) { step[++cnt] = _s, pre[cnt] = dep[cnt] = 0, rr[cnt] = c; /* */mem(go[cnt], 0); return
   cnt; }
   inline void Init() { n = cnt = str_cnt = 0, S = T = new_node(0, 0); }
   inline void I(int c) {
      ++n, c = h(c), p = T; if (!go[p][c]) T = new_node(step[T] + 1, c);
      for (; p && !go[p][c]; p = pre[p]) go[p][c] = T;
      if (!p) pre[T] = S; else {
         q = go[p][c]; int &X = (p == T ? T : pre[T]);
         if (step[p] + 1 == step[q]) X = q; else {
            nq = new_node(step[p] + 1, c);
            rep(j, 0, M) go[nq][j] = go[q][j];
            for (; p && go[p][c] == q; p = pre[p]) go[p][c] = nq;
            pre[nq] = pre[q], X = pre[q] = nq;
         }
      }
      id[n] = T; SS[T][str_cnt]++; //...
   }
   inline void Insert(const char s[], int _n = 0, int op = 0) {
      if (!_n) _n = strlen(s); ++str_cnt; T = S;
      if (!op) rep(i, 0, _n) I(s[i]); else per(i, 0, _n) I(s[i]);
   }
   inline void Go(int &p, int c) { while (p && !go[p][c]) p = pre[p]; if (p) p = go[p][c], ans += dp[p]; }
   inline int Run(const char s[], int _n = 0, int op = 0) {
      if (!_n) _n = strlen(s); int pos = S, p = op*(_n - 1), q = (op ? -1 : _n); op = 1 - 2 * op; ans = 0;
      for (int i = p; i != q; i += op) if (pos) Go(pos, h(s[i])); else break; return pos;
   }
   inline int get_dep(int x) { return (!x || dep[x]) ? dep[x] : dep[x] = get_dep(pre[x]) + 1; }
   inline void Toop() {
      rep(i, 0, n + 1) temp[i] = 0; rep(i, 1, cnt + 1) temp[get_dep(i)]++;
      rep(i, 1, n + 1) temp[i] += temp[i - 1]; per(i, 1, cnt + 1) toop[temp[dep[i]]--] = i;
   }
   inline void Merge(int x, int y) { rep_it(it, SS[y]) SS[x][it->fir] += it->sec; }
   inline void Count() { per(i, 1, cnt + 1) Merge(pre[toop[i]], toop[i]); }
   inline void Q() {
      Toop(), Count(); rep(i, 1, cnt + 1) if (SS[i].size() >= k) dp[i] = step[i] - step[pre[i]]; else dp[i] = 0;
      rep(i, 1, cnt + 1) dp[toop[i]] += dp[pre[toop[i]]]; rep(i, 1, str_cnt + 1) Run(st[i].c_str()), printf("%lld ", ans);
   }//...
};
```

2.10 SAM_广义_trie 树

```
struct SAM {
   static const int M = 26; int go[N << 1][M], pre[N << 1], step[N << 1], rr[N << 1], temp[N << 1], toop[N << 1], num[N <<
   1], dep[N << 1], cnt, S, n, p, q, nq; ll dp[N << 1], ans = 0;
   inline int h(int c) { return c - 'a'; }
   inline int new_node(int _s, int c) { step[++cnt] = _s, pre[cnt] = num[cnt] = 0, rr[cnt] = c; /* */mem(go[cnt], 0); return
   cnt; }
   inline void Init() { n = cnt = 0, S = new_node(0, 0); }
   inline int I(int T, int c) {
      if (go[T][c = h(c)] && step[go[T][c]] == step[T] + 1) return go[T][c];
      ++n, p = T, T = new_node(step[T] + 1, c);
      for (; p && !go[p][c]; p = pre[p]) go[p][c] = T;
      if (!p) pre[T] = S; else {
         q = go[p][c];
         if (step[p] + 1 == step[q]) pre[T] = q; else {
            nq = new_node(step[p] + 1, c);
            rep(j, 0, M) go[nq][j] = go[q][j];
            for (; p && go[p][c] == q; p = pre[p]) go[p][c] = nq;
            pre[nq] = pre[q], pre[T] = pre[q] = nq;
         }
      }
      num[T]++; return T; //...
   }
   inline void Go(int &p, int c) { while (p && !go[p][c]) p = pre[p]; if (p) p = go[p][c], ans += dp[p]; }
   inline int Run(const char s[], int _n = 0, int op = 0) {
      if (!_n) _n = strlen(s); int pos = S, p = op*(_n - 1), q = (op ? -1 : _n); op = 1 - 2 * op; ans = 0;
      for (int i = p; i != q; i += op) if (pos) Go(pos, h(s[i])); else break; return pos;
   }
   inline int get_dep(int x) { return (!x || dep[x]) ? dep[x] : dep[x] = get_dep(pre[x]) + 1; }
   inline void Toop() {
      rep(i, 0, n + 1) temp[i] = 0; rep(i, 1, cnt + 1) temp[get_dep(i)]++;
      rep(i, 1, n + 1) temp[i] += temp[i - 1]; per(i, 1, cnt + 1) toop[temp[dep[i]]--] = i;
   }
   inline void Count() { per(i, 1, cnt + 1) num[pre[toop[i]]] += num[toop[i]]; }
   inline ll Q() {
      /* Toop(); Count(); */ll ans = 0; rep(i, 1, cnt + 1) ans += step[i] - step[pre[i]]; return ans; //...
```

```
   }
};

const int _N = 1e5 + 5, _M = 1e5 + 5;
struct Tu {
   int head[_N], nxt[_M * 2], e[_M * 2], id[_N], n, tot; ll v[_N], w[_M * 2]; SAM M;
   inline void Init(int _n) { n = _n, mem(head, 0), tot = 0; M.Clear(); id[0] = M.S; }
   inline void I(int x, int y, ll _w = 0) { e[++tot] = y, w[tot] = _w; nxt[tot] = head[x], head[x] = tot; }
   void dfs(int x, int f) {
      id[x] = M.I(id[f], v[x]);
      for (int i = head[x]; i; i = nxt[i]) if (e[i] != f) dfs(e[i], x);
   }
   inline void Q() { /* */ }
};

// 此版本注意： N=2*M*V( 结点数 )
```

2.11 SAM_标准版

```
struct SAM {
   static const int M = 26; int go[N << 1][M], pre[N << 1], step[N << 1], rr[N << 1], temp[N << 1], toop[N << 1], num[N <<
   1], id[N << 1], cnt, S, T, n, p, q, nq;
   inline int h(int c) { return c - 'a'; }
   inline int new_node(int _s, int c) { step[++cnt] = _s, pre[cnt] = num[cnt] = 0, rr[cnt] = c; /* */mem(go[cnt], 0); return
   cnt; }
   inline void Init() { n = cnt = 0, S = T = new_node(0, 0); }
   inline void I(int c) {
      ++n, c = h(c), p = T, T = new_node(step[T] + 1, c);
      for (; p && !go[p][c]; p = pre[p]) go[p][c] = T;
      if (!p) pre[T] = S; else {
         q = go[p][c];
         if (step[p] + 1 == step[q]) pre[T] = q; else {
            nq = new_node(step[p] + 1, c);
            rep(j, 0, M) go[nq][j] = go[q][j];
            for (; p && go[p][c] == q; p = pre[p]) go[p][c] = nq;
            pre[nq] = pre[q], pre[T] = pre[q] = nq;
         }
      }
      num[id[n] = T]++; //...
   }
   inline void Insert(char s[], int _n = 0, int op = 0) {
      if (!_n) _n = strlen(s);
      if (!op) rep(i, 0, _n) I(s[i]); else per(i, 0, _n) I(s[i]);
   }
   inline void Toop() {
      rep(i, 0, n + 1) temp[i] = 0; rep(i, 1, cnt + 1) temp[step[i]]++;
      rep(i, 1, n + 1) temp[i] += temp[i - 1]; rep(i, 1, cnt + 1) toop[temp[step[i]]--] = i;
   }
   inline void Count() { per(i, 1, cnt + 1) num[pre[toop[i]]] += num[toop[i]]; }
   inline int Q() { Toop(); return 0; }//...
};

// rr[] 表示 right 集合 , 用来维护需要量
```