Biological Warfare

CMU_xiaodao

Last build at June 4, 2019

Contents

Chapter 0

日常 (Daily Use)

0.1 表头 (Header)

```
/** Micro Mezz Macro Flation -- Overheated Economy ., Last Update: Nov. 7th 2013 **/ //{

/** Header .. **/ //{
#pragma comment(linker, ”/STACK:36777216”)
//#pragma GCC optimize (”O2”)
#define LOCAL
//#include ”testlib.h”
#include <functional>
#include <algorithm>
#include <iostream>
#include <fstream>
#include <sstream>
#include <iomanip>
#include <numeric>
#include <cstring>
#include <climits>
#include <cassert>
#include <complex>
#include <cstdio>
#include <string>
#include <vector>
#include <bitset>
#include <queue>
#include <stack>
#include <cmath>
#include <ctime>
#include <list>
#include <set>
#include <map>

//#include <tr1/unordered_set>
//#include <tr1/unordered_map>
//#include <array>

using namespace std;

#define REP(i, n) for (int i=0;i<n;++i)
#define FOR(i, a, b) for (int i=a;i<b;++i)
#define DWN(i, b, a) for (int i=b-1;i>=a;--i)
#define REP_1(i, n) for (int i=1;i<=n;++i)
#define FOR_1(i, a, b) for (int i=a;i<=b;++i)
#define DWN_1(i, b, a) for (int i=b;i>=a;--i)
#define REP_C(i, n) for (int n____=n,i=0;i<n____;++i)
#define FOR_C(i, a, b) for (int b____=b,i=a;i<b____;++i)
#define DWN_C(i, b, a) for (int a____=a,i=b-1;i>=a____;--i)
#define REP_N(i, n) for (i=0;i<n;++i)
#define FOR_N(i, a, b) for (i=a;i<b;++i)
```

```
#define DWN_N(i, b, a) for (i=b-1;i>=a;--i)
#define REP_1_C(i, n) for (int n____=n,i=1;i<=n____;++i)
#define FOR_1_C(i, a, b) for (int b____=b,i=a;i<=b____;++i)
#define DWN_1_C(i, b, a) for (int a____=a,i=b;i>=a____;--i)
#define REP_1_N(i, n) for (i=1;i<=n;++i)
#define FOR_1_N(i, a, b) for (i=a;i<=b;++i)
#define DWN_1_N(i, b, a) for (i=b;i>=a;--i)
#define REP_C_N(i, n) for (int n____=(i=0,n);i<n____;++i)
#define FOR_C_N(i, a, b) for (int b____=(i=0,b);i<b____;++i)
#define DWN_C_N(i, b, a) for (int a____=(i=b-1,a);i>=a____;--i)
#define REP_1_C_N(i, n) for (int n____=(i=1,n);i<=n____;++i)
#define FOR_1_C_N(i, a, b) for (int b____=(i=1,b);i<=b____;++i)
#define DWN_1_C_N(i, b, a) for (int a____=(i=b,a);i>=a____;--i)

#define ECH(it, A) for (__typeof(A.begin()) it=A.begin(); it != A.end(); ++it)
#define REP_S(i, str) for (char*i=str;*i;++i)
#define REP_L(i, hd, nxt) for (int i=hd;i;i=nxt[i])
#define REP_G(i, u) REP_L(i,hd[u],suc)
#define REP_SS(x, s) for (int x=s;x;x=(x-1)&s)
#define DO(n) for ( int ____n = n; ____n-->0; )
#define REP_2(i, j, n, m) REP(i, n) REP(j, m)
#define REP_2_1(i, j, n, m) REP_1(i, n) REP_1(j, m)
#define REP_3(i, j, k, n, m, l) REP(i, n) REP(j, m) REP(k, l)
#define REP_3_1(i, j, k, n, m, l) REP_1(i, n) REP_1(j, m) REP_1(k, l)
#define REP_4(i, j, k, ii, n, m, l, nn) REP(i, n) REP(j, m) REP(k, l) REP(ii, nn)
#define REP_4_1(i, j, k, ii, n, m, l, nn) REP_1(i, n) REP_1(j, m) REP_1(k, l) REP_1(ii, nn)

#define ALL(A) A.begin(), A.end()
#define LLA(A) A.rbegin(), A.rend()
#define CPY(A, B) memcpy(A, B, sizeof(A))
#define INS(A, P, B) A.insert(A.begin() + P, B)
#define ERS(A, P) A.erase(A.begin() + P)
#define LBD(A, x) (lower_bound(ALL(A), x) - A.begin())
#define UBD(A, x) (lower_bound(ALL(A), x) - A.begin())
#define CTN(T, x) (T.find(x) != T.end())
#define SZ(A) int((A).size())
#define PB push_back
#define MP(A, B) make_pair(A, B)
#define PTT pair<T, T>
#define Ts *this
#define rTs return Ts
#define fi first
#define se second
#define re real()
#define im imag()

#define Rush for(int ____T=RD(); ____T--;)
#define Display(A, n, m) { \
  REP(i, n){  \
        REP(j, m-1) cout << A[i][j] << ” ”; \
        cout << A[i][m-1] << endl;  \
    }      \
}
#define Display_1(A, n, m) { \
    REP_1(i, n){  \
        REP_1(j, m-1) cout << A[i][j] << ” ”; \
        cout << A[i][m] << endl;  \
    }      \
}

typedef long long LL;
//typedef long double DB;
typedef double DB;
typedef unsigned uint;
typedef unsigned long long uLL;

typedef vector<int> VI;
typedef vector<char> VC;
typedef vector<string> VS;
typedef vector<LL> VL;
typedef vector<DB> VF;
typedef set<int> SI;
typedef set<string> SS;
typedef map<int, int> MII;
typedef map<string, int> MSI;
typedef pair<int, int> PII;
typedef pair<LL, LL> PLL;
typedef vector<PII> VII;
typedef vector<VI> VVI;
typedef vector<VII> VVII;

template<class T> inline T& RD(T &);
template<class T> inline void OT(const T &);
//inline int RD(){int x; return RD(x);}
inline LL RD(){LL x; return RD(x);}
inline DB& RF(DB &);
inline DB RF(){DB x; return RF(x);}
inline char* RS(char *s);
inline char& RC(char &c);
inline char RC();
inline char& RC(char &c){scanf(” %c”, &c); return c;}
inline char RC(){char c; return RC(c);}
//inline char& RC(char &c){c = getchar(); return c;}
//inline char RC(){return getchar();}

template<class T> inline T& RDD(T &);
inline LL RDD(){LL x; return RDD(x);}

template<class T0, class T1> inline T0& RD(T0 &x0, T1 &x1){RD(x0), RD(x1); return x0;}
template<class T0, class T1, class T2> inline T0& RD(T0 &x0, T1 &x1, T2 &x2){RD(x0), RD(x1), RD(x2); return x0;}
template<class T0, class T1, class T2, class T3> inline T0& RD(T0 &x0, T1 &x1, T2 &x2, T3 &x3){RD(x0), RD(x1), RD(x2), RD(x3); return x0;}
template<class T0, class T1, class T2, class T3, class T4> inline T0& RD(T0 &x0, T1 &x1, T2 &x2, T3 &x3, T4 &x4){RD(x0), RD(x1), RD(x2), RD(x3), RD(x4); return x0;}
template<class T0, class T1, class T2, class T3, class T4, class T5> inline T0& RD(T0 &x0, T1 &x1, T2 &x2, T3 &x3, T4 &x4, T5 &x5){RD(x0), RD(x1), RD(x2), RD(x3), RD(x4), RD(x5); return x0;}
template<class T0, class T1, class T2, class T3, class T4, class T5, class T6> inline T0& RD(T0 &x0, T1 &x1, T2 &x2, T3 &x3, T4 &x4, T5 &x5, T6 &x6){RD(x0), RD(x1), RD(x2), RD(x3), RD(x4), RD(x5), RD(x6); return x0;}
template<class T0, class T1> inline void OT(const T0 &x0, const T1 &x1){OT(x0), OT(x1);}
template<class T0, class T1, class T2> inline void OT(const T0 &x0, const T1 &x1, const T2 &x2){OT(x0), OT(x1), OT(x2);}
template<class T0, class T1, class T2, class T3> inline void OT(const T0 &x0, const T1 &x1, const T2 &x2, const T3 &x3){OT(x0), OT(x1), OT(x2), OT(x3);}
template<class T0, class T1, class T2, class T3, class T4> inline void OT(const T0 &x0, const T1 &x1, const T2 &x2, const T3 &x3, const T4 &x4){OT(x0), OT(x1), OT(x2), OT(x3), OT(x4);}
template<class T0, class T1, class T2, class T3, class T4, class T5> inline void OT(const T0 &x0, const T1 &x1, const T2 &x2, const T3 &x3, const T4 &x4, const T5 &x5){OT(x0), OT(x1), OT(x2), OT(x3), OT(x4), OT(x5);}
template<class T0, class T1, class T2, class T3, class T4, class T5, class T6> inline void OT(const T0 &x0, const T1 &x1, const T2 &x2, const T3 &x3, const T4 &x4, const T5 &x5, const T6 &x6){OT(x0), OT(x1), OT(x2), OT(x3), OT(x4), OT(x5), OT(x6);}
inline char& RC(char &a, char &b){RC(a), RC(b); return a;}
inline char& RC(char &a, char &b, char &c){RC(a), RC(b), RC(c); return a;}
inline char& RC(char &a, char &b, char &c, char &d){RC(a), RC(b), RC(c), RC(d); return a;}
inline char& RC(char &a, char &b, char &c, char &d, char &e){RC(a), RC(b), RC(c), RC(d), RC(e); return a;}
inline char& RC(char &a, char &b, char &c, char &d, char &e, char &f){RC(a), RC(b), RC(c), RC(d), RC(e), RC(f); return a;}
inline char& RC(char &a, char &b, char &c, char &d, char &e, char &f, char &g){RC(a), RC(b), RC(c), RC(d), RC(e), RC(f), RC(g); return a;}
inline DB& RF(DB &a, DB &b){RF(a), RF(b); return a;}
inline DB& RF(DB &a, DB &b, DB &c){RF(a), RF(b), RF(c); return a;}
inline DB& RF(DB &a, DB &b, DB &c, DB &d){RF(a), RF(b), RF(c), RF(d); return a;}
inline DB& RF(DB &a, DB &b, DB &c, DB &d, DB &e){RF(a), RF(b), RF(c), RF(d), RF(e); return a;}
inline DB& RF(DB &a, DB &b, DB &c, DB &d, DB &e, DB &f){RF(a), RF(b), RF(c), RF(d), RF(e), RF(f); return a;}
inline DB& RF(DB &a, DB &b, DB &c, DB &d, DB &e, DB &f, DB &g){RF(a), RF(b), RF(c), RF(d), RF(e), RF(f), RF(g); return a;}
inline void RS(char *s1, char *s2){RS(s1), RS(s2);}
inline void RS(char *s1, char *s2, char *s3){RS(s1), RS(s2), RS(s3);}
template<class T0,class T1>inline void RDD(T0&a, T1&b){RDD(a),RDD(b);}
```

```
template<class T0,class T1,class T2>inline void RDD(T0&a, T1&b, T2&c){RDD(a),RDD(b),RDD(c);}

template<class T> inline void RST(T &A){memset(A, 0, sizeof(A));}
template<class T> inline void FLC(T &A, int x){memset(A, x, sizeof(A));}
template<class T> inline void CLR(T &A){A.clear();}

template<class T0, class T1> inline void RST(T0 &A0, T1 &A1){RST(A0), RST(A1);}
template<class T0, class T1, class T2> inline void RST(T0 &A0, T1 &A1, T2 &A2){RST(A0), RST(A1), RST(A2);}
template<class T0, class T1, class T2, class T3> inline void RST(T0 &A0, T1 &A1, T2 &A2, T3 &A3){RST(A0), RST(A1), RST(A2), RST(A3);}
template<class T0, class T1, class T2, class T3, class T4> inline void RST(T0 &A0, T1 &A1, T2 &A2, T3 &A3, T4 &A4){RST(A0), RST(A1), RST(A2), RST(A3), RST(A4);}
template<class T0, class T1, class T2, class T3, class T4, class T5> inline void RST(T0 &A0, T1 &A1, T2 &A2, T3 &A3, T4 &A4, T5 &A5){RST(A0), RST(A1), RST(A2), RST(A3), RST(A4), RST(A5);}
template<class T0, class T1, class T2, class T3, class T4, class T5, class T6> inline void RST(T0 &A0, T1 &A1, T2 &A2, T3 &A3, T4 &A4, T5 &A5, T6 &A6){RST(A0), RST(A1), RST(A2), RST(A3), RST(A4), RST(A5), RST(A6);}
template<class T0, class T1> inline void FLC(T0 &A0, T1 &A1, int x){FLC(A0, x), FLC(A1, x);}
template<class T0, class T1, class T2> inline void FLC(T0 &A0, T1 &A1, T2 &A2, int x){FLC(A0, x), FLC(A1, x), FLC(A2, x);}
template<class T0, class T1, class T2, class T3> inline void FLC(T0 &A0, T1 &A1, T2 &A2, T3 &A3, int x){FLC(A0, x), FLC(A1, x), FLC(A2, x), FLC(A3, x);}
template<class T0, class T1, class T2, class T3, class T4> inline void FLC(T0 &A0, T1 &A1, T2 &A2, T3 &A3, T4 &A4, int x){FLC(A0, x), FLC(A1, x), FLC(A2, x), FLC(A3, x), FLC(A4, x);}
template<class T0, class T1, class T2, class T3, class T4, class T5> inline void FLC(T0 &A0, T1 &A1, T2 &A2, T3 &A3, T4 &A4, T5 &A5, int x){FLC(A0, x), FLC(A1, x), FLC(A2, x), FLC(A3, x), FLC(A4, x), FLC(A5, x);}
template<class T0, class T1, class T2, class T3, class T4, class T5, class T6> inline void FLC(T0 &A0, T1 &A1, T2 &A2, T3 &A3, T4 &A4, T5 &A5, T6 &A6, int x){FLC(A0, x), FLC(A1, x), FLC(A2, x), FLC(A3, x), FLC(A4, x), FLC(A5, x), FLC(A6, x);}
template<class T> inline void CLR(priority_queue<T, vector<T>, less<T> > &Q){while (!Q.empty()) Q.pop();}
template<class T> inline void CLR(priority_queue<T, vector<T>, greater<T> > &Q){while (!Q.empty()) Q.pop();}
template<class T> inline void CLR(stack<T> &S){while (!S.empty()) S.pop();}

template<class T0, class T1> inline void CLR(T0 &A0, T1 &A1){CLR(A0), CLR(A1);}
template<class T0, class T1, class T2> inline void CLR(T0 &A0, T1 &A1, T2 &A2){CLR(A0), CLR(A1), CLR(A2);}
template<class T0, class T1, class T2, class T3> inline void CLR(T0 &A0, T1 &A1, T2 &A2, T3 &A3){CLR(A0), CLR(A1), CLR(A2), CLR(A3);}
template<class T0, class T1, class T2, class T3, class T4> inline void CLR(T0 &A0, T1 &A1, T2 &A2, T3 &A3, T4 &A4){CLR(A0), CLR(A1), CLR(A2), CLR(A3), CLR(A4);}
template<class T0, class T1, class T2, class T3, class T4, class T5> inline void CLR(T0 &A0, T1 &A1, T2 &A2, T3 &A3, T4 &A4, T5 &A5){CLR(A0), CLR(A1), CLR(A2), CLR(A3), CLR(A4), CLR(A5);}
template<class T0, class T1, class T2, class T3, class T4, class T5, class T6> inline void CLR(T0 &A0, T1 &A1, T2 &A2, T3 &A3, T4 &A4, T5 &A5, T6 &A6){CLR(A0), CLR(A1), CLR(A2), CLR(A3), CLR(A4), CLR(A5), CLR(A6);}
template<class T> inline void CLR(T &A, int n){REP(i, n) CLR(A[i]);}

template<class T> inline bool EPT(T &a){return a.empty();}
template<class T> inline T& SRT(T &A){sort(ALL(A)); return A;}
template<class T, class C> inline T& SRT(T &A, C B){sort(ALL(A), B); return A;}
template<class T> inline T& RVS(T &A){reverse(ALL(A)); return A;}
template<class T> inline T& UNQQ(T &A){A.resize(unique(ALL(A))-A.begin());return A;}
template<class T> inline T& UNQ(T &A){SRT(A);return UNQQ(A);}


//}

/** Constant List .. **/ //{

const int MOD = int(1e9) + 7;
//int MOD = 99990001;
const int INF = 0x3f3f3f3f;
const LL INFF = 0x3f3f3f3f3f3f3f3fLL;
const DB EPS = 1e-9;
const DB OO = 1e20;
const DB PI = acos(-1.0); //M_PI;

const int dx[] = {-1, 0, 1, 0};
const int dy[] = {0, 1, 0, -1};

//}

```

```
/** Add On .. **/ //{
// <<= '0. Nichi Joo ., //{

template<class T> inline T& checkMin(T &a,const T b){if (b<a) a=b;return a;}
template<class T> inline T& checkMax(T &a,const T b){if (a<b) a=b;return a;}
template<class T> inline T& checkMin(T &a, T &b, const T x){checkMin(a, x), checkMin(b, x);return a;}
template<class T> inline T& checkMax(T &a, T &b, const T x){checkMax(a, x), checkMax(b, x);return a;}
template <class T, class C> inline T& checkMin(T& a, const T b, C c){if (c(b,a)) a = b;return a;}
template <class T, class C> inline T& checkMax(T& a, const T b, C c){if (c(a,b)) a = b;return a;}
template<class T> inline T min(T a, T b, T c){return min(min(a, b), c);}
template<class T> inline T max(T a, T b, T c){return max(max(a, b), c);}
template<class T> inline T min(T a, T b, T c, T d){return min(min(a, b), min(c, d));}
template<class T> inline T max(T a, T b, T c, T d){return max(max(a, b), max(c, d));}
template<class T> inline T min(T a, T b, T c, T d, T e){return min(min(min(a,b),min(c,d)),e);}
template<class T> inline T max(T a, T b, T c, T d, T e){return max(max(max(a,b),max(c,d)),e);}
template<class T> inline T sqr(T a){return a*a;}
template<class T> inline T cub(T a){return a*a*a;}
template<class T> inline T ceil(T x, T y){return (x - 1) / y + 1;}
template<class T> T abs(T x){return x>0?x:-x;}
inline int sgn(DB x){return x < -EPS ? -1 : x > EPS;}
inline int sgn(DB x, DB y){return sgn(x - y);}

inline DB cos(DB a, DB b, DB c){return (sqr(a)+sqr(b)-sqr(c))/(2*a*b);}
inline DB cot(DB x){return 1./tan(x);};
inline DB sec(DB x){return 1./cos(x);};
inline DB csc(DB x){return 1./sin(x);};

//}
// <<= '1. Bitwise Operation ., //{
namespace BO{

inline bool _1(int x, int i){return bool(x&1<<i);}
inline bool _1(LL x, int i){return bool(x&1LL<<i);}
inline LL _1(int i){return 1LL<<i;}
inline LL _U(int i){return _1(i) - 1;};

inline int reverse_bits(int x){
    x = ((x >> 1) & 0x55555555) | ((x << 1) & 0xaaaaaaaa);
    x = ((x >> 2) & 0x33333333) | ((x << 2) & 0xcccccccc);
    x = ((x >> 4) & 0x0f0f0f0f) | ((x << 4) & 0xf0f0f0f0);
    x = ((x >> 8) & 0x00ff00ff) | ((x << 8) & 0xff00ff00);
    x = ((x >>16) & 0x0000ffff) | ((x <<16) & 0xffff0000);
    return x;
}

inline LL reverse_bits(LL x){
    x = ((x >> 1) & 0x5555555555555555LL) | ((x << 1) & 0xaaaaaaaaaaaaaaaaLL);
    x = ((x >> 2) & 0x3333333333333333LL) | ((x << 2) & 0xccccccccccccccccLL);
    x = ((x >> 4) & 0x0f0f0f0f0f0f0f0fLL) | ((x << 4) & 0xf0f0f0f0f0f0f0f0LL);
    x = ((x >> 8) & 0x00ff00ff00ff00ffLL) | ((x << 8) & 0xff00ff00ff00ff00LL);
    x = ((x >>16) & 0x0000ffff0000ffffLL) | ((x <<16) & 0xffff0000ffff0000LL);
    x = ((x >>32) & 0x00000000ffffffffLL) | ((x <<32) & 0xffffffff00000000LL);
    return x;
}

template<class T> inline bool odd(T x){return x&1;}
template<class T> inline bool even(T x){return !odd(x);}
template<class T> inline T low_bit(T x) {return x & -x;}
template<class T> inline T high_bit(T x) {T p = low_bit(x);while (p != x) x -= p, p = low_bit(x);return p;}
template<class T> inline T cover_bit(T x){T p = 1; while (p < x) p <<= 1;return p;}
template<class T> inline int cover_idx(T x){int p = 0; while (_1(p) < x ) ++p; return p;}

inline int clz(int x){return __builtin_clz(x);}
inline int clz(LL x){return __builtin_clzll(x);}
inline int ctz(int x){return __builtin_ctz(x);}
inline int ctz(LL x){return __builtin_ctzll(x);}
inline int lg2(int x){return !x ? -1 : 31 - clz(x);}
inline int lg2(LL x){return !x ? -1 : 63 - clz(x);}
inline int low_idx(int x){return !x ? -1 : ctz(x);}
inline int low_idx(LL x){return !x ? -1 : ctz(x);}
inline int high_idx(int x){return lg2(x);}
inline int high_idx(LL x){return lg2(x);}
inline int parity(int x){return __builtin_parity(x);}
inline int parity(LL x){return __builtin_parityll(x);}
inline int count_bits(int x){return __builtin_popcount(x);}
inline int count_bits(LL x){return __builtin_popcountll(x);}

} using namespace BO;//}
// <<= '9. Comutational Geometry .,//{
namespace CG{

#define cPo const Po&
#define cLine const Line&
#define cSeg const Seg&

inline DB dist2(DB x,DB y){return sqr(x)+sqr(y);}

struct Po{
    DB x,y;Po(DB x=0,DB y=0):x(x),y(y){}

    void in(){RF(x,y);}void out(){printf(”(%.2f,%.2f)”,x,y);}
    inline friend istream&operator>>(istream&i,Po&p){return i>>p.x>>p.y;}
    inline friend ostream&operator<<(ostream&o,Po p){return o<<”(”<<p.x<<”, ”<<p.y<< ”)”;}

    Po operator-()const{return Po(-x,-y);}
    Po&operator+=(cPo p){x+=p.x,y+=p.y;rTs;}Po&operator-=(cPo p){x-=p.x,y-=p.y;rTs;}
    Po&operator*=(DB k){x*=k,y*=k;rTs;}Po&operator/=(DB k){x/=k,y/=k;rTs;}
    Po&operator*=(cPo p){rTs=Ts*p;}Po&operator/=(cPo p){rTs=Ts/p;}
    Po operator+(cPo p)const{return Po(x+p.x,y+p.y);}Po operator-(cPo p)const{return Po(x-p.x,y-p.y);}
    Po operator*(DB k)const{return Po(x*k,y*k);}Po operator/(DB k)const{return Po(x/k,y/k);}
    Po operator*(cPo p)const{return Po(x*p.x-y*p.y,y*p.x+x*p.y);}Po operator/(cPo p)const{return Po(x*p.x+y*p.y,y*p.x-x*p.y)/p.len2();}

    bool operator==(cPo p)const{return!sgn(x,p.x)&&!sgn(y,p.y);};bool operator!=(cPo p)const{return sgn(x,p.x)||sgn(y,p.y);}
    bool operator<(cPo p)const{return sgn(x,p.x)<0||!sgn(x,p.x)&&sgn(y,p.y)<0;}bool operator<=(cPo p)const{return sgn(x,p.x)<0||!sgn(x,p.x)&&sgn(y,p.y)<=0;}
    bool operator>(cPo p)const{return!(Ts<=p);}bool operator >=(cPo p)const{return!(Ts<p);}

    DB len2()const{return dist2(x,y);}DB len()const{return sqrt(len2());}DB arg()const{return atan2(y,x);}
    Po&_1(){rTs/=len();}Po&conj(){y=-y;rTs;}Po&lt(){swap(x,y),x=-x;rTs;}Po&rt(){swap(x,y),y=-y;rTs;}
    Po&rot(DB a,cPo o=Po()){Ts-=o;Ts*=Po(cos(a),sin(a));rTs+=o;}
};

inline DB dot(DB x1,DB y1,DB x2,DB y2){return x1*x2+y1*y2;}
inline DB dot(cPo a,cPo b){return dot(a.x,a.y,b.x,b.y);}
inline DB dot(cPo p0,cPo p1,cPo p2){return dot(p1-p0,p2-p0);}
inline DB det(DB x1,DB y1,DB x2,DB y2){return x1*y2-x2*y1;}
inline DB det(cPo a,cPo b){return det(a.x,a.y,b.x,b.y);}
inline DB det(cPo p0,cPo p1,cPo p2){return det(p1-p0,p2-p0);}
inline DB ang(cPo p0,cPo p1){return acos(dot(p0,p1)/p0.len()/p1.len());}
inline DB ang(cPo p0,cPo p1,cPo p2){return ang(p1-p0,p2-p0);}
inline DB ang(cPo p0,cPo p1,cPo p2,cPo p3){return ang(p1-p0,p3-p2);}
inline DB dist2(const Po &a, const Po &b){return dist2(a.x-b.x, a.y-b.y);}
template<class T1, class T2> inline int dett(const T1 &x, const T2 &y){return sgn(det(x, y));}
template<class T1, class T2, class T3> inline int dett(const T1 &x, const T2 &y, const T3 &z){return sgn(det(x, y, z));}
template<class T1, class T2, class T3, class T4> inline int dett(const T1 &x, const T2 &y, const T3 &z, const T4 &w){return sgn(det(x, y, z, w));}
template<class T1, class T2> inline int dott(const T1 &x, const T2 &y){return sgn(dot(x, y));}
template<class T1, class T2, class T3> inline int dott(const T1 &x, const T2 &y, const T3 &z){return sgn(dot(x, y, z));}
template<class T1, class T2, class T3, class T4> inline int dott(const T1 &x, const T2 &y, const T3 &z, const T4 &w){return sgn(dot(x, y, z, w));}
template<class T1, class T2> inline DB arg(const T1 &x, const T2 &y){DB a=ang(x,y);return~dett(x,y)?a:2*PI-a;}
template<class T1, class T2, class T3> inline DB arg(const T1 &x, const T2 &y, const T3 &z){DB a=ang(x,y,z);return~dett(x,y,z)?a:2*PI-a;}
```

```
template<class T1, class T2, class T3, class T4> inline DB arg(const T1 &x, const T2 &y, const T3 &z, const T4 &w){DB a=ang(x,y,z,w);return~dett(x,y,z,w)?a:2*PI-a;}
template<class T1, class T2> inline DB dist(const T1 &x, const T2 &y){return sqrt(dist2(x, y));}
template<class T1, class T2, class T3> inline DB dist(const T1 &x, const T2 &y, const T3 &z){return sqrt(dist2(x, y, z));}
inline Po _1(Po p){return p._1();}inline Po conj(Po p){return p.conj();}
inline Po lt(Po p){return p.lt();}inline Po rt(Po p){return p.rt();}
inline Po rot(Po p,DB a,cPo o=Po()){return p.rot(a,o);}
inline Po operator *(DB k,cPo p){return p*k;}
inline Po operator /(DB k,cPo p){return conj(p)*k/p.len2();}

typedef vector<Po> VP;

struct Line{
    Po a,b;Line(cPo a=Po(),cPo b=Po()):a(a),b(b){}
    Line(DB x0,DB y0,DB x1,DB y1):a(Po(x0,y0)),b(Po(x1,y1)){}
    Line(cLine l):a(l.a),b(l.b){}

    //Ax+By+C=0
    Line(DB A,DB B,DB C){
        C=-C;if(!::sgn(A))a=Po(0,C/B),b=Po(1,C/B);
        else if(!::sgn(B))a=Po(C/A,0),b=Po(C/A,1);
        else a=Po(0,C/B),b=Po(1,(C-A)/B);
    }

    void in(){a.in(),b.in();}
    inline friend istream&operator>>(istream&i,Line& p){return i>>p.a>>p.b;}
    inline friend ostream&operator<<(ostream&o,Line p){return o<<p.a<<”-”<< p.b;}

    Line operator+(cPo x)const{return Line(a+x,b+x);}
    Line operator-(cPo x)const{return Line(a-x,b-x);}
    Line operator*(DB k)const{return Line(a*k,b*k);}
    Line operator/(DB k)const{return Line(a/k,b/k);}

    Po operator*(cLine)const;
    Po d()const{return b-a;}DB len2()const{return d().len2();}DB len()const{return d().len();}DB arg()const{return d().arg();}

    int sgn(cPo p)const{return dett(a, b, p);}
    int sgn(cLine)const;

    bool sameSgn(cPo p1,cPo p2)const{return sgn(p1)==sgn(p2);}
    void getEquation(DB&K,DB&B)const{
        K = ::sgn(a.x, b.x) ? (b.y-a.y)/(b.x-a.x) : OO;
        B = a.y - K*a.x;
    }
    void getEquation(DB&A,DB&B,DB&C)const{A=a.y-b.y,B=b.x-a.x,C=det(a, b);}

    Line&push(DB r){ // 正数右手螺旋向里
        Po v=d()._1().lt()*r;a+=v,b+=v; rTs;
    }
};

inline DB dot(cLine l1,cLine l2){return dot(l1.d(),l2.d());}
inline DB dot(cLine l,cPo p){return dot(l.a,l.b,p);}
inline DB dot(cPo p,cLine l){return dot(p,l.a,l.b);}
inline DB det(cLine l1,cLine l2){return det(l1.d(),l2.d());}
inline DB det(cLine l,cPo p){return det(l.a,l.b,p);}
inline DB det(cPo p,cLine l){return det(p,l.a,l.b);}
inline DB ang(cLine l0,cLine l1){return ang(l0.d(),l1.d());}
inline DB ang(cLine l,cPo p){return ang(l.a,l.b,p);}
inline DB ang(cPo p,cLine l){return ang(p,l.a,l.b);}

inline int Line::sgn(cLine l)const{return dett(Ts, l);}
inline Po Line::operator*(cLine l)const{return a+d()*det(a,l)/det(Ts,l);}
inline Po operator&(cPo p,cLine l){return l*Line(p,p+l.d().lt());}
inline Po operator%(cPo p,cLine l){return p&l*2-p;}
inline Line push(Line l, DB r){return l.push(r);}

```

```
struct Seg: public Line{
    Seg(cPo a=Po(),cPo b=Po()):Line(a,b){}
    Seg(DB x0,DB y0,DB x1,DB y1):Line(x0,y0,x1,y1){}
    Seg(cLine l):Line(l){}
    Seg(const Po &a,DB alpha):Line(a,alpha){}
    Seg(DB A,DB B,DB C):Line(A,B,C){}

    inline int sgn(cPo p)const;
    inline int sgn(cLine l)const;
    inline bool qrt(cSeg l)const;
    inline int sgn(cSeg l)const;
};

 // -1不相交 0相交（不规范） 1相交（规范）

inline int Seg::sgn(cPo p)const{return -dott(p,a,b);}
inline int Seg::sgn(cLine l)const{return sgn(Ts*l);}

// quick_rejection_test
inline bool Seg::qrt(cSeg l)const{
    return min(a.x,b.x)<=max(l.a.x,l.b.x)&&min(l.a.x,l.b.x)<=max(a.x,b.x)&&
        min(a.y,b.y)<=max(l.a.y,l.b.y)&&min(l.a.y,l.b.y)<=max(a.y,b.y);
}


inline int Seg::sgn(cSeg l)const{
    if (!qrt(l)) return -1;

    /*return
        (dett(a,b,l.a)*dett(a,b,l.b)<=0 &&
        dett(l.a,l.b,a)*dett(l.a,l.b,b)<=0)?1:-1;*/

    int d1=dett(a,b,l.a),d2=dett(a,b,l.b),d3=dett(l.a,l.b,a),d4=dett(l.a,l.b,b);
    if ((d1^d2)==-2&&(d3^d4)==-2)return 1;
    return ((!d1&&dott(l.a-a,l.a-b)<=0)||(!d2&&dott(l.b-a,l.b-b)<=0)||
            (!d3&&dott(a-l.a,a-l.b)<=0)||(!d4&&dott(b-l.a,b-l.b)<=0))?0:-1;
}

//inline DB dist2(cLine l,cPo p){return sqr(fabs(dot(lt(l.d()), p-l.a)))/l.len2();}
inline DB dist2(cLine l,cPo p){return sqr(fabs(det(l.d(), p-l.a)))/l.len2();}

inline DB dist2(cLine l1,cLine l2){return dett(l1,l2)?0:dist2(l1,l2.a);}

inline DB dist2(cSeg l,cPo p){
    Po pa = p - l.a, pb = p - l.b;
    if (dott(l.d(), pa) <= 0) return pa.len2();
    if (dott(l.d(), pb) >= 0) return pb.len2();
    return dist2(Line(l), p);
}


inline DB dist2(cSeg s,cLine l){
    Po v1=s.a-l.a,v2=s.b-l.a;DB d1=det(l.d(),v1),d2=det(l.d(),v2);
    return sgn(d1)!=sgn(d2) ? 0 : sqr(min(fabs(d1), fabs(d2)))/l.len2();
}
inline DB dist2(cSeg l1,cSeg l2){
    if (~l1.sgn(l2)) return 0;
    else return min(dist2(l2,l1.a), dist2(l2,l1.b), dist2(l1,l2.a), dist2(l1,l2.b));
}
template<class T1, class T2> inline DB dist2(const T1& a, const T2& b){
    return dist2(b, a);
}

} using namespace CG;//}
//}
```

```

/** I/O Accelerator Interface .. **/ //{
#define g (c=getchar())
#define d isdigit(g)
#define p x=x*10+c-’0’
#define n x=x*10+’0’-c
#define pp l/=10,p
#define nn l/=10,n
template<class T> inline T& RD(T &x){
    char c;while(!d);x=c-’0’;while(d)p;
    return x;
}
template<class T> inline T& RDD(T &x){
    char c;while(g,c!=’-’&&!isdigit(c));
    if (c==’-’){x=’0’-g;while(d)n;}
    else{x=c-’0’;while(d)p;}
    return x;
}
inline DB& RF(DB &x){
    //scanf(”%lf”, &x);
    char c;while(g,c!=’-’&&c!=’.’&&!isdigit(c));
    if(c==’-’)if(g==’.’){x=0;DB l=1;while(d)nn;x*=l;}
        else{x=’0’-c;while(d)n;if(c==’.’){DB l=1;while(d)nn;x*=l;}}
    else if(c==’.’){x=0;DB l=1;while(d)pp;x*=l;}
        else{x=c-’0’;while(d)p;if(c==’.’){DB l=1;while(d)pp;x*=l;}}
    return x;
}
#undef nn
#undef pp
#undef n
#undef p
#undef d
#undef g
inline char* RS(char *s){
    //gets(s);
    scanf(”%s”, s);
    return s;
}

LL last_ans; int Case; template<class T> inline void OT(const T &x){
    //printf(”Case #%d: ”, ++Case);
    //printf(”%lld\n”, x);
    //printf(”%.4f\n”, x);
    printf(”%d\n”, x);
    //cout << x << endl;
    //last_ans = x;
}
//}


//}/* ........................................................................................................................ */

int n;

int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    Rush{

    }
}
```

0.2 G++ 调栈

```
int __size__ = 256 << 20; // 256MB
char *__p__ = (char*)malloc(__size__) + __size__;
__asm__(”movl %0, %%esp\n” :: ”r”(__p__));
```

Part I

数据结构 (Data Structure)

Chapter 1

区间 kth

1.1 静态

1.1.1 算法一：主席树

```

const int N = int(1e5)+9, LV = 18, NN = N *(LV+9);


namespace FotileTree{

    VI P;int A[N],T[NN],lc[NN],rc[NN],cc[NN];LL ss[NN];
    int nn, n, m;

    #define rt 0,n-1
    #define lx lc[x]
    #define rx rc[x]
    #define ly lc[y]
    #define ry rc[y]
    #define ml (l+r>>1)
    #define mr (ml+1)

    int new_node(int x = 0){
        lc[nn]=lx,rc[nn]=rx,cc[nn]=cc[x],ss[nn]=ss[x];
        return nn++;
    }

    int Insert(int y,int l,int r,int p){
        int x = new_node(y); cc[x] += 1, ss[x] += P[p];
        if (l < r){
            if (p<mr) lx = Insert(ly,l,ml,p);
            else rx = Insert(ry,mr,r,p);
        }
        return x;
    }

    int Insert(int y,int p){
        return Insert(y,rt,p);
    }

    int Select(int x,int y,int l,int r,int k){
        if (l == r) return l;
        return k <= cc[lx]-cc[ly] ? Select(lx,ly,l,ml,k) : Select(rx,ry,mr,r,k-cc[lx]+cc[ly]);
    }

    int Select(int l,int r,int k){
        return P[Select(T[r],T[l-1],rt,k)];
    }

    template<class T> T Rank(int x,int y,int l,int r,int p,T cc[]){
        if (l == r) return cc[x]-cc[y];
        return p < P[mr] ? Rank(lx,ly,l,ml,p,cc) : cc[lx]-cc[ly]+Rank(rx,ry,mr,r,p,cc);
    }

    int Rank(int l,int r,int p){
        return P[Rank(T[r],T[l-1],rt,p,cc)];
    }

    LL Lsum(int l, int r, int p){
        return Rank(T[r],T[l-1],rt,p,ss);
    }

    void Init(){
        nn = 0; T[0] = new_node(); CLR(P); REP_1_C(i, n) P.PB(RDD(A[i])); UNQ(P);
        REP_1(i, n) T[i] = Insert(T[i-1], LBD(P,A[i]));
    }
} using namespace FotileTree;

int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    RD(n,m); Init(); DO(m){
        int a,b,k;RD(a,b,k);
        OT(Select(a,b,k));
    }
}
```

1.1.2 算法二：划分树

```

const int N = int(1e5)+9, LV = 18;
int A[N], T[LV][N];
int n, m;

#define rt 0,0,n-1
#define lvv (lv+1)
#define ml (l+r>>1)
#define mr (ml+1)
#define lc lvv,l,ml
#define rc lvv,mr,r
#define t T[lv][i]

void Build(int lv, int l, int r){
    if (l == r) return;
    int ll = l, rr = mr; FOR_1(i, l, r){
        if (t <= A[ml]) T[lvv][ll++] = t; else T[lvv][rr++] = t;
        t = ll-l;
    }
    Build(lc), Build(rc);
}

#define t (rr-ll)

int Select(int lv, int l, int r, int a, int b, int k){
    if (l == r) return A[a]; int ll = a == l ? 0 : T[lv][a-1], rr = T[lv][b];
    return t >= k ? Select(lc,l+ll,l+rr-1,k) : Select(rc,mr+a-l-ll,mr+b-l-rr,k-t);
}

int Rank(int lv, int l, int r, int a, int b, int v){
    if (l == r) return a==b; int ll = a == l ? 0 : T[lv][a-1], rr = T[lv][b];
    return v < A[mr] ? Rank(lc,l+ll,l+rr-1,v) : t+Rank(rc,mr+a-l-ll,mr+b-l-rr,v);
}

int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    RD(n, m); REP(i, n) T[0][i] = RDD(A[i]); sort(A,A+n);
    Build(rt);int a,b,k;DO(m){
        RD(a,b,k);--a,--b;
        OT(Select(rt,a,b,k));
    }
}
```

1.2 带修改

```


//}/* .............................................................................................................................. */

const int N = int(4e5) + 9, Z = 26, LV = 20;
int L[N],R[N],T[N];int n;

namespace SAM{

    int trans[N][Z], par[N], len[N], tot, tail;

#define v trans[u][c]
#define p par[u]
#define pp par[uu]

    inline int new_node(){
        RST(trans[tot]);
        return tot++;
    }

    inline int new_node(int u){
        CPY(trans[tot], trans[u]); par[tot] = par[u];
        return tot++;
    }

    inline int h(int u){
        return len[u] - len[p];
    }

    int Ext(int c){
        int u = tail, uu = new_node(); len[uu] = len[u] + 1;
        while (u && !v) v = uu, u = p;
        if (!u && !v) v = uu, pp = 0;
        else{
            if (len[v] == len[u] + 1) pp = v;
            else{
                int _v = v, vv = new_node(_v); len[vv] = len[u] + 1; par[_v] = pp = vv;
                while (u && v == _v) v = vv, u = p;
                if (!u && v == _v) v = vv;
            }
        }
        return tail = uu;
    }

    char str[N/2];int prefix[N/2];VI adj[N];int fa[LV][N],L[N],R[N],tt;

    int Find(int u, int l){
        u = prefix[u]; DWN(lv, LV, 0){
            if(len[fa[lv][u]] >= l) u = fa[lv][u];
        }
        return u;
    }

#undef v
#define v (*it)
    void dfs(int u = 0){
        L[u]=++tt; ECH(it, adj[u]){
            fa[0][v] = u;
            FOR(lv, 1, LV) fa[lv][v] = fa[lv-1][fa[lv-1][v]];
            dfs(v);
        }
        R[u]=tt;
    }

    void Init(){
        tail = tot = 0; new_node();
        RS(str); n = 0; REP_S(cur, str) prefix[n++] = Ext(*cur-’a’);
        REP(u, tot) adj[u].clear(); FOR(u, 1, tot) adj[p].PB(u),T[u]=0;tt=0,dfs();
    }

#undef v
#undef p
#undef pp
}

namespace SBT{
    const int NN = N*LV;
    int c[2][NN], sz[NN], ky[NN], tot;
#define lx l[x]
#define rx r[x]
#define l c[d]
#define r c[!d]
#define kx ky[x]
#define sx sz[x]
#define d 0
    int new_node(int v = 0){
        int x=++tot;lx=rx=0;
        sx=1;kx=v;
        return x;
    }

    void upd(int x){
        sx=sz[lx]+1+sz[rx];
    }
#undef d
    void rot(int &x,int d){
        int y=rx;rx=l[y];l[y]=x;
        upd(x),upd(y),x=y;
    }

    void fix(int &x,int d){
        if (sz[l[lx]] > sz[rx]) rot(x,!d);
        else{
            if (sz[r[lx]] > sz[rx]) rot(lx,d),rot(x,!d);
            else return;
        }
        d=0,fix(lx,0),fix(rx,1);
        fix(x,0),fix(x,1);
    }
#define d 0
    void Ins(int &x,int v){
        if(!x) x = new_node(v);
        else{
            ++sz[x]; Ins(c[v>kx][x],v);
            fix(x,v>=kx);
        }
    }

    int d_key; void Del(int &x,int v){
        --sx;if(kx==v||(v<kx&&!lx)||(v>kx&&!rx)){
            if(!lx||!rx) d_key = kx, x = lx | rx;
            else Del(lx,v+1), kx = d_key;
        }
        else Del(c[v>kx][x],v);
    }

    int Rank(int x,int v){
        int z=0;while(x){
            if(kx<v){
                z+=sz[lx]+1;
                x=rx;
            }
            else x=lx;
        }
        return z;
    }
    bool Find(int x,int v){
        if (!x) return 0;if (kx==v) return 1;
        return Find(c[v>kx][x],v);
    }

    void Init(){
        tot = 0;
    }

#undef d
#undef l
#undef r
#undef lx
#undef rx
#undef sx
#undef kx
};

namespace SGT{
#define rt 1, 1, n
#define lx (x<<1)
#define rx (lx|1)
#define ml (l+r>>1)
#define mr (ml+1)
#define lc lx, l, ml
#define rc rx, mr, r

    int T[N*4], p, v;

    void Build(int x, int l, int r){
        T[x]=0;if (l<r) Build(lc), Build(rc);
    }

    void Ins(int x, int l, int r){
        SBT::Ins(T[x], v);
        if (l < r){
            if (p < mr) Ins(lc); else Ins(rc);
        }
    }
    void Del(int x, int l, int r){
        SBT::Del(T[x], v);
        if (l < r){
            if (p < mr) Del(lc); else Del(rc);
        }
    }

    void Ins(int _p, int _v){
        p = _p, v = _v; Ins(rt);
    }
    void Del(int _p, int _v){
        p = _p, v = _v; Del(rt);
    }

    inline int Select(int x,int l,int r,int ll,int rr,int k){
#define cnt(x) (SBT::Rank(T[x],rr+1)-SBT::Rank(T[x],ll))
        while(l < r){
            if(cnt(lx)>=k){
                x = lx, r = ml;
            }
            else
            {
                k-=cnt(lx);
                x = rx, l = mr;
            }
        }
        return cnt(x)>=k ? l : -1;
    }

    void Init(){
        Build(rt);
    }
}

int main(){

#ifndef ONLINE_JUDGE
    //freopen(”in.txt”, ”r”, stdin);
    freopen(”1009.in”, ”r”, stdin);
    //freopen(”out2.txt”, ”w”, stdout);
#endif

//汇编调栈
int __size__ = 256 << 20; // 256MB
char *__p__ = (char*)malloc(__size__) + __size__;
__asm__(”movl %0, %%esp\n” :: ”r”(__p__));


    Rush{
        printf(”Case #%d:\n”,++Case);
        SAM::Init();SBT::Init();SGT::Init();

        Rush{
            int t,x,p,k;if(RD(t,x,p)==1){
                int u=SAM::Find(p,x);if(SBT::Find(T[u],x))continue;
                SGT::Ins(x,L[u]);SBT::Ins(T[u],x);
            }
            else if(t==2){
                int u=SAM::Find(p,x);if(!SBT::Find(T[u],x)) continue;
                SGT::Del(x,L[u]);SBT::Del(T[u],x);
            }
            else{
                int u=SAM::Find(p,x);RD(k)+=SBT::Rank(T[u],x);
                printf(”%d\n”, SGT::Select(rt,L[u],R[u],k));
        }
    }
}
}
```

1.3 带插入

1.4 例题 (E.g.)

Chapter 2

主席树 (Fotile Tree)

题目描述 (Brief description)

。。。n 个结点的带容量无向树，m 个询问。每个询问形如 (s, t, k, a, b)。。表示。。。。允许已 a 的代价修建一条单位容量的新边，b 的代价将一条旧边或新边增加单位流量。。。。预算为 k 时 s->t 的最大流。。

。。先考虑加边的情况。。如果要加边的话。。只会加在 s->t 上。。。。。如果 a <= b 。。那么狂加边就行了。。。否则的话。。只会添加一条边。。且扩容操作全部给这条边最优。

。。接下来考虑不加边的情况。。取出 s->t 路径上的所有边权。。在预算范围内尽可能让红线画的更高。。推更多的流。。。。显然这是树上区间 kth 问题。。。可以使用主席树。。。。

算法分析 (Algorithm analysis)

本来主席树求 kth 大是只带一个 logn 的。。。我比赛的时候搞着搞着又搞成二分那条红线了。又把第二个 logn 加回来艹。。。。。。。

。。言归正传。。。对于求 s->t 的初始 flow 的过程。。就是求这个路径上 rmq。。。。。现在反正有了主席树那么求初始流可以用 kth() 。。解决。（这里 k 固定为 1）。。。。。在预算范围至多还能推多少流的函数我们记作 kth2() 。。这里的”k” 表示预算。。在这个函数的末尾。。求出流量后我们立刻返回收益。。。。（注意。。主席树的值域我们只开到 10000.。所以可能返回收益的时刻还有没有花完的预算。。还可以继续加上。。

。。。。需要维护。。。。c[]：个数。。以及。d[]：和。。

```
const int N = 100009, M = 2 * N, LM = 18;

int hd[N], suc[M], to[M], wt[N];
int ST[LM][M], st[N], dep[N]; // Euler index ...
int n, tt; int T[N], Null;

const int NN = 20 * N;
int l[NN], r[NN], c[NN], d[NN], total;
// Chairman tree

#define lx l[x]
#define rx r[x]
#define ly l[y]
#define ry r[y]
#define cx c[x]
#define cy c[y]

#define ml (ll+rr>>1)
#define mr (ml+1)
#define lc lx, ll, ml
#define rc rx, mr, rr

#define lt lx = ++total, rx = ry, x = lx, y = ly, rr = ml
#define rt lx = ly, rx = ++total, x = rx, y = ry, ll = mr

int Tn;

int new_node(){
    ++total; l[total] = r[total] = c[total] = d[total] = 0;
    return total;
}
```

```

int Insert(int y, int p){

    int x = new_node(), root = x, ll = 0, rr = Tn;
    c[x] = c[y] + 1, d[x] = d[y] + p;

    while (ll < rr){
        if (p < mr) lt; else rt;
        c[x] = c[y] + 1, d[x] = d[y] + p;
    }

    return root;
}

inline bool elder(int a, int b){
    return dep[a] < dep[b];
}

inline int lca(int a, int b){
    int l = st[a], r = st[b];
    if (l > r) swap(l, r); ++r; int lv = lg2(r-l); //log2(r - l);
    return min(ST[lv][l], ST[lv][r-(1<<lv)], elder);
}

#define aa to[i^1]
#define bb to[i]
#define v bb
#define ww wt[i/2]

void dfs(int u = 1){
    ST[0][st[u] = ++tt] = u;
    REP_G(i, u) if (!st[v]){
        dep[v] = dep[u] + 1, T[v] = Insert(T[u], ww);
        dfs(v);
        ST[0][++tt] = u;
    }
}

int kth2(int x, int y, int k){

    int z = lca(x, y);
    x = T[x], y = T[y], z = T[z];
    int ll = 0, rr = Tn, t, cc = 0, dd = 0;
    int D = c[x] + c[y] - 2*c[z], tc, td;

    while (ll < rr){
        if (ml * (cc + (tc = c[lx] + c[ly] - 2*c[l[z]])) - (dd + (td = d[lx] + d[ly] - 2*d[l[z]])) >= k){
            x = l[x], y = l[y], z = l[z];
            rr = ml;
        }
        else {
            x = r[x], y = r[y], z = r[z];
            cc += tc, dd += td, ll = mr;
        }
    }

    if ((k-((cc*ll)-dd))<0) --ll;
    return ll + (k-((cc*ll)-dd))/D;
}

int kth(int x, int y, int k){

    int z = lca(x, y);
    x = T[x], y = T[y], z = T[z];
    int ll = 0, rr = Tn, t;

    while (ll < rr){
        if ((t = c[l[x]] + c[l[y]] - 2*c[l[z]]) >= k){
            x = l[x], y = l[y], z = l[z];
            rr = ml;
        }
        else {
            x = r[x], y = r[y], z = r[z];
            k -= t, ll = mr;
        }
    }

    return ll;
}

int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    freopen(”out2.txt”, ”w”, stdout);
#endif

    Rush{

        printf(”Case #%d:\n”, ++Case);

        int Q; RD(n, Q); fill(hd+1, hd+n+1, 0); fill(st+1, st+n+1, 0);
        Tn = 0; FOR_C(i, 2, n << 1){
            RD(to[i], to[i|1]); checkMax(Tn, RD(ww));
            suc[i] = hd[aa], hd[aa] = i++;
            suc[i] = hd[aa], hd[aa] = i;
        }

        total = 0, T[1] = new_node();
        tt = 0, dfs();

        for ( int lv = 1 ; _1(lv) <= tt ; lv ++ ){
            for ( int i = 1 ; i + _1(lv) <= tt + 1 ; i ++ )
                ST[lv][i] = min(ST[lv-1][i], ST[lv-1][i + _1(lv-1)], elder);
        }

        DO(Q){
            int s, t, k, a, b; RD(s, t, k, a, b);
            int flow = kth(s, t, 1), res = a <= b ? k/a + flow : max((k>=a?(k-a)/b+1:0) + flow, kth2(s, t, k/b));
            printf(”%d\n”, res);
        }
    }
}
```

2.0.1 DQUERY

简述 (Brief description)

分析 (Analysis)

```
离线 BIT

namespace BIT{
    const int N = int(3e4) + 9, M = int(2e5) + 9;
    int A[N], B[N], P[N], C[N], n;
    VII Q[N]; int ans[M], m;
    void Add(int x, int d){
        for (;x<=n;x+=low_bit(x)) C[x] += d;
    }
    int Sum(int x){
        int res = 0; for (;x;x^=low_bit(x)) res += C[x];
        return res;
    }
} using namespace BIT;

int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    REP_1_C(i, RD(n)) B[i] = RD(A[i]); sort(B+1, B+n+1), m = unique(B+1, B+n+1) - B;
    REP_1(i, n) A[i] = lower_bound(B+1, B+m, A[i]) - B; REP_C(i, RD(m)){
        int l, r; RD(l, r);
        Q[l].PB(MP(r, i));
    }

    DWN_1(i, n, 1){
        if (P[A[i]]) Add(P[A[i]], -1); Add(P[A[i]] = i, 1);
        ECH(it, Q[i]) ans[it->se] = Sum(it->fi);
    }

    REP(i, m) OT(ans[i]);
}
```

```
主席树

const int N = 30009;

int A[N], B[N], P[N];
int n, m;

namespace Fotile_Tree{

    #define lx l[x]
    #define rx r[x]
    #define ly l[y]
    #define ry r[y]
    #define cx c[x]
    #define cy c[y]
    #define mid ((ll+rr)>>1)

    const int NN = N * 18 + 9; // int(1e6) + 9;

    int l[NN], r[NN], c[NN], tot;
    int T[N];

    int Build(int ll, int rr){
        int x = ++tot; if (ll < rr) lx = Build(ll, mid), rx = Build(mid+1, rr);
        return x;
    }

    int Insert(int y, int p, int d){
        int x = ++tot, root = x;

        c[x] = c[y] + d; int ll = 1, rr = n;
        while (ll < rr){
            if (p <= mid){
                lx = ++tot, rx = ry;
                x = lx, y = ly, rr = mid;
            }
            else {
                lx = ly, rx = ++tot;
                x = rx, y = ry, ll = mid + 1;
            }
            c[x] = c[y] + d;
        }
        return root;
    }

    inline int lsum(int x, int p){
        int res = 0, ll = 1, rr = n;
        while (p != rr){
            if (p <= mid) x = lx, rr = mid;
            else res += c[lx], x = rx, ll = mid + 1;
        }
        return res + cx;
    }

    #undef lx
    #undef rx
    #undef ly
    #undef ry
    #undef cx
    #undef cy
    #undef mid

} using namespace Fotile_Tree;

int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
#endif

    REP_1_C(i, RD(n)) B[i] = RD(A[i]); sort(B+1, B+n+1), m = unique(B+1, B+n+1) - B;
    REP_1(i, n) A[i] = lower_bound(B+1, B+m, A[i]) - B;

    DWN_1(i, n, 1){
        T[i] = Insert(!P[A[i]] ? T[i+1] : Insert(T[i+1], P[A[i]], -1), i, 1);
        P[A[i]] = i;
    }

    Rush{
        int l, r; RD(l, r);
        OT(lsum(T[l], r));
    }
}
```

Chapter 3

可持久化树堆 (Treap)

```
// UVA 12538
const int N = int(1e7) + 9, SN = int(1e6) + 9, VN = int(5e4) + 9;

namespace Treap{

    int c[2][N], sz[N], ww[N], tot; char ch[N], str[SN];
    int T[VN], _T, tt;

#define l c[0]
#define r c[1]
#define lx l[x]
#define rx r[x]
#define ml (a + b >> 1)
#define mr (ml + 1)
#define lc a, ml
#define rc mr, b

    inline int update(int x){
        sz[x] = sz[lx] + 1 + sz[rx];
        return x;
    }

    inline int new_node(char chx){
        int x = ++tot;
        lx = rx = 0, ww[x] = rand(), sz[x] = 1, ch[x] = chx;
        return x;
    }

    inline int new_node(int xx){
        int x = ++tot;
        lx = l[xx], rx = r[xx], ww[x] = ww[xx], sz[x] = sz[xx], ch[x] = ch[xx];
        return x;
    }

    int merge(int a, int b){
        if(!a||!b) return a|b;

        if(ww[a] > ww[b]){
            a = new_node(a), r[a] = merge(r[a], b);
            return update(a);
        }
        else{
            b = new_node(b), l[b] = merge(a, l[b]);
            return update(b);
        }
    }

    void split(int x, int p, int &a, int &b){
        if(!p) a = 0, b = x; else if(sz[x] == p) a = x, b = 0;
        else{
            x = new_node(x);
            if(p <= sz[lx]) split(lx, p, a, b), lx = b, b = x;
            else split(rx, p-sz[lx]-1, a, b), rx = a, a = x;
            update(x);
        }
    }

    int build(int a = 0,int b = strlen(str)){
        if (a >= b) return 0;
        int x = new_node(str[ml]);
        lx = build(lc), rx = build(rc);
        update(x);
        return x;
    }

    void print(int x,int a,int b){
        if (!x) return;
        if (a <= sz[lx]) print(lx, a, b); a -= sz[lx]+1, b -= sz[lx]+1;
        if (a <= 0 && 0 < b) putchar(ch[x]), _T += ch[x] == ’c’;
        if (1 < b) print(rx, a, b);
    }
} using namespace Treap;

int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”print.txt”, ”w”, stdprint);
#endif

    int t, s, n, a, b, _; Rush switch(RD()){
        case 1:
            RD(s)-=_T, RS(str);
            split(T[tt], s, a, b);
            T[++tt] = merge(merge(a, build()), b);
            break;
        case 2:
            RD(s, n), s-=_T,n-=_T;
            split(T[tt], s-1, a, b), split(b, n, _, b);
            T[++tt] = merge(a, b);
            break;
        default:
            RD(t, s, n), t-=_T,s-=_T,n-=_T;
            print(T[t], s, s+n), puts(””);
    }
}
```

Chapter 4

替罪羊 (Scapegoat)

Chapter 5

KD-树 (KD-Tree)

5.0.1 区间合并

Chapter 6

动态 KD-树 (Dynamic KD-Tree)

6.0.1 区间合并

Chapter 7

伸展树 (Splay)

7.1 例题 (E.g.)

7.1.1 SPOJ SEQ2

```
const int N = 500009;

struct node{

    static node *NIL, *rt, *tp; node *c[2], *p;
    int sz, ky, ss, ls, rs, ms, bj;

#define NIL node::NIL
#define rt node::rt
#define l c[0]
#define r c[1]
#define lx x->l
#define rx x->r
#define px x->p
#define ly y->l
#define ry y->r
#define py y->p

    inline void reset(int v){l=r=p=NIL,ky=v,bj=0;}
    inline void rev(){bj^=1,swap(l,r),swap(ls,rs);}
    inline void sss(){bj=2,ss=sz*ky,ms=ls=rs=ky<0?ky:ss;}

    inline void upd(){
        assert(this != NIL);
        sz = l->sz + 1 + r->sz, ss = l->ss + ky + r->ss;
        ls = max(l->ls, l->ss + ky + max(0, r->ls));
        rs = max(r->rs, r->ss + ky + max(0, l->rs));
        ms = max(l->ms, max(0, l->rs) + ky + max(0, r->ls), r->ms);
    }
    inline void rls(){
        assert(this != NIL);
        if (bj){
            if (bj&1) l->rev(), r->rev();
            if (bj&2) l->ky = r->ky = ky, l->sss(), r->sss();
            bj = 0;
        }
    }
    inline int sgn(){return p->r==this;}
    inline void setc(int d,node*x){c[d]=x,px=this;}
    inline void setl(node*x){setc(0,x);}
    inline void setr(node*x){setc(1,x);}

    inline void rot(int d){
        node*y=p,*z=py;z->setc(y->sgn(),this);
        y->setc(d,c[!d]),setc(!d,y),y->upd();
    }
    inline void rot(){rot(sgn());}

    //inline void fix(){if (~sgn()) p->fix(); rls();}
/*
    inline node* splay(node*t){
        while (p!=t) rot(); upd();
        return this;
    }
*/
    inline node*splay(node*t){
        int a,b;while(p!=t){
            if (p->p==t){rot();break;}
            else a=sgn(),b=p->sgn(),(a^b?this:p)->rot(a),rot(b);
        }
        upd();if (t==NIL)rt=this;
        return this;
    }


    void rcc();

    void inorder(){
        if (this == NIL) return;
        rls(); l->inorder();
        printf(ky == -INF ? ”$ ” : ”%d ”, ky);
        r->inorder();
    }

} *NIL, *rt, TPool[N], *TStack[N];
int tp, ts;

#define mid (a + b >> 1)
#define lc a, mid-1
#define rc mid+1, b

node *select(int k, node*t=NIL){
    node *x = rt; while (x->rls(), lx->sz != k){
        if (k < lx->sz) x = lx;
        else k -= lx->sz+1, x = rx;
    }
    return x->splay(t);
}

node *select(int a, int b){
    return select(a-1, select(b+1))->r;
}

inline void node::rcc(){
    if (this == NIL) return;
    l->rcc(), r->rcc();
    TStack[++ts] = this;
}

inline node *new_node(int v){
    node *x = ts ? TStack[ts--] : &TPool[++tp];
    x->reset(v);
    return x;
}

int A[N], s, n, m; inline node *Build(int a = 1, int b = n){
    if (a > b) return NIL;
    node *x = new_node(A[mid]);
    x->setl(Build(lc)), x->setr(Build(rc)), x->upd();
    return x;
}

```

```
int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    NIL = &TPool[0], A[0] = -INF;

    Rush{

        tp = ts = 0; RD(n, m); REP_1(i, n) RDD(A[i]); A[n+1] = -INF; rt = Build(0, n+1);
        node *x, *y, *z; char cmd[10]; DO(m){

            switch(RS(cmd)[0]){
                case ’I’: // Insert ... .
                    RD(s, n); REP_1(i, n) RDD(A[i]); y = select(s, z = select(s+1)), x = Build();
                    y->setr(x), y->upd(), z->upd();
                    break;
                case ’D’: // Delete
                    RD(s, n); y = select(s-1, z = select(s+n)), x = ry;
                    x->rcc(), ry = NIL, y->upd(), z->upd();
                    break;
                case ’R’: // Reverse ..
                    RD(s, n); y = select(s-1, z = select(s+n)), x = ry;
                    x->rev(), y->upd(), z->upd();
                    break;
                case ’M’: // Make_Same // Max_Sum
                    if (cmd[2] == ’X’) {OT(rt->ms); break;}
                    RD(s, n); y = select(s-1, z = select(s+n)), x = ry;
                    RDD(x->ky), x->sss(), y->upd(), z->upd();
                    break;
                default: // Get_Sum ..
                    RD(s, n);
                    OT(select(s, s+n-1)->ss);
            }
        }
    }
}
```

Chapter 8

动态树 (Link-Cut Tree)

8.1 维护路劲信息 (Path)

8.1.1 定义

```
    static node *NIL; node *c[2], *p;
    int bj, sz, ky, ss, ls, rs, ms;

#define NIL node::NIL
#define l c[0]
#define r c[1]
#define lx x->l
#define rx x->r
#define px x->p
#define ly y->l
#define ry y->r
#define py y->p
```

8.1.2 标记

```
    inline void reset(int v=0){l=r=p=NIL,bj=0,ky=v;}
    inline node(int v=0){reset(v);}

    inline void rev(){bj^=1,swap(l, r),swap(ls, rs);}
    inline void sss(){bj|=2,ss=sz*ky,ms=ls=rs=max(0,ss);}

    inline void upd(){
        assert(this != NIL);
        sz = l->sz + 1 + r->sz, ss = l->ss + ky + r->ss;
        ls = max(l->ls, l->ss + ky + r->ls);
        rs = max(r->rs, r->ss + ky + l->rs);
        ms = max(l->ms, l->rs + ky + r->ls, r->ms);
    }

    inline void rls(){
        assert(this != NIL);
        if (bj){
            if (bj&1) l->rev(), r->rev();
            if (bj&2) l->ky = r->ky = ky, l->sss(), r->sss();
            bj = 0;
        }
    }
```

MSS
必须取满 k 段。。

```
    inline void sss(){bj|=2, ss = sz * ky, ms = ls = rs = ky < 0 ? ky : ss;}

    inline void upd(){
        sz = l->sz + 1 + r->sz, ss = l->ss + ky + r->ss;
        ls = max(l->ls, l->ss + ky + max(0, r->ls));
        rs = max(r->rs, r->ss + ky + max(0, l->rs));
        ms = max(l->ms, max(0, l->rs) + ky + max(0, r->ls), r->ms);
    }
```

不必须。。。

```
    inline void sss(){bj|=2,ss=sz*ky,ms=ls=rs=max(0,ss);}

    inline void upd(){
        assert(this != NIL);
        sz = l->sz + 1 + r->sz, ss = l->ss + ky + r->ss;
        ls = max(l->ls, l->ss + ky + r->ls);
        rs = max(r->rs, r->ss + ky + l->rs);
        ms = max(l->ms, l->rs + ky + r->ls, r->ms);
    }
```

LCIS

```
    inline void rev(){bj^=1,swap(l,r),swap(bd[0],bd[1]),swap(up[0],dn[1]),swap(up[1],dn[0]),swap(up[2],dn[2]);}

    inline void upd(){
        //assert(this != NIL);
        sz = l->sz + 1 + r->sz;
        bd[0] = l == NIL ? ky : l->bd[0];
        bd[1] = r == NIL ? ky : r->bd[1];

        up[0] = l->up[0]; if (l == NIL || up[0] == l->sz && l->bd[1] < ky)
            up[0] += 1 + (ky < r->bd[0] ? r->up[0] : 0);
        dn[0] = l->dn[0]; if (l == NIL || dn[0] == l->sz && l->bd[1] > ky)
            dn[0] += 1 + (ky > r->bd[0] ? r->dn[0] : 0);
        up[1] = r->up[1]; if (r == NIL || up[1] == r->sz && ky < r->bd[0])
            up[1] += 1 + (l->bd[1] < ky ? l->up[1] : 0);
        dn[1] = r->dn[1]; if (r == NIL || dn[1] == r->sz && ky > r->bd[0])
            dn[1] += 1 + (l->bd[1] > ky ? l->dn[1] : 0);

        up[2] = max(l->up[2], (l->bd[1] < ky ? l->up[1] : 0) + 1 + (ky < r->bd[0] ? r->up[0] : 0), r->up[2]);
        dn[2] = max(l->dn[2], (l->bd[1] > ky ? l->dn[1] : 0) + 1 + (ky > r->bd[0] ? r->dn[0] : 0), r->dn[2]);

    }
```

8.1.3 旋转

```
    inline int sgn(){return p->l==this?0:p->r==this?1:-1;}
    inline void setc(int d,node*x){c[d]=x,px=this;}

    inline void rot(int d){
        node*y=p,*z=py;if(~y->sgn())z->setc(y->sgn(),this);else p=z;
        y->setc(d,c[!d]),setc(!d,y),y->upd();
    }
    inline void rot(){rot(sgn());}
```

8.1.4 伸展

```
    inline void fix(){if (~sgn()) p->fix(); rls();}
/*
    inline node* splay(){
        fix();while (~sgn()) rot(); upd();
        return this;
    }
*/

    inline node*splay(){
        fix();int a,b;while(~(a=sgn())){
            if(~(b=(p->sgn())))(a^b?this:p)->rot(a),rot(b);
            else rot(a);
        }
        upd();
        return this;
    }
```

8.1.5 虚实切换

```
   inline node *acs(){
        node *x = this, *y = NIL; do{
            x->splay();
            rx = y, x->upd();
            y = x, x = px;
        } while (x != NIL);
        return splay();
    }
```

8.1.6 换根

```
   inline node* rt(){node* x; for (x = acs(); x->rls(), lx != NIL; x = lx); return x->splay();}
   inline node* ert(){acs()->rev();return this;}
```

8.1.7 动态 LCA

8.2 形态变换 (Link/Cut)

```
   void Link(node *x){
        if (rt() == x->rt()){
            puts(”-1”);
        }
        else {
            ert(), p = x;
        }
    }

    void Cut(){
        acs(); l->p = l = NIL;
    }

    void Cut(node* x){
        if (this == x || rt() != x->rt()){
            puts(”-1”);
        }
        else {
            ert(), x->Cut();
        }
    }
```

8.3 例题 (E.g.)

8.3.1 HDU 4010. Query on the trees

题目描述 (Brief description)

... 动态维护一组森林，要求支持以下操作：

Link(a, b) 如果 a，b 不在同一颗子树中，则通过在 a，b 之间连边的方式，连接这两棵子树。

Cut(a, b) 如果 a，b 在同一颗子树中、且 a != b，则将 a 视为这棵子树的根之后，切断 b 与其父亲结点的连接。

Modify(w, a, b) 如果 a, b 在同一颗子树中，则将 a, b 之间路径上所有的点权增加 w。

Query(a, b) 如果 a, b 在同一颗子树中，返回 a, b 之间路径上点权的最大值。

```
#include <iostream>
#include <cstdio>
#include <cstring>
#include <cassert>
using namespace std;
#define REP_1(i, n) for (int i=1;i<=n;++i)
#define Rush for(int ______T=RD(); ______T--;)

/** I/O Accelerator Interface .. **/ //{
#define g (c=getchar())
#define d isdigit(g)
#define p x=x*10+c-’0’
template<class T> inline T& RD(T &x){
    char c;while(!d);x=c-’0’;while(d)p;
    return x;
}
#undef p
#undef d
#undef g
inline int RD(){int x; return RD(x);}
inline char* RS(char *s){
    scanf(”%s”, s);
    return s;
}
template<class T> inline void OT(const T &x){
    printf(”%d\n”, x);
}
//}

const int N = int(3e5)+9, M = 2*N;
int c[2][N], p[N];
int w1[N], w2[N], d0[N]; bool r0[N];
#define l c[0]
#define r c[1]
void reset(int x){
    l[x]=r[x]=p[x]=0;
    d0[x]=r0[x]=0;
}
inline void rev(int x){
    r0[x]^=1,swap(l[x],r[x]);
}
inline void inc(int x,int d){
    if(!x)return;//!
    w1[x]+=d,w2[x]+=d,d0[x]+=d;
}
inline void upd(int x){
    w2[x]=max(max(w2[l[x]],w1[x]),w2[r[x]]);
}
inline void rls(int x){
    if (r0[x]){
        rev(l[x]),rev(r[x]);
        r0[x]=0;
    }
    if (d0[x]){
        inc(l[x],d0[x]),inc(r[x],d0[x]);
        d0[x]=0;
    }
}
inline int sgn(int x){return l[p[x]]==x?0:r[p[x]]==x?1:-1;}
inline void setc(int x, int d, int y){p[c[d][x]=y]=x;}
inline void rot(int x, int d){
    int y=p[x],z=p[y];if (~sgn(y))setc(z,sgn(y),x);else p[x]=z;
    setc(y,d,c[!d][x]),setc(x,!d,y),upd(y);
}
inline void fix(int x){if(~sgn(x))fix(p[x]);rls(x);}
inline int splay(int x){
    fix(x);int a,b,y;while (~(a=sgn(x))){
        if (~(b=sgn(y=p[x])))rot(a^b?x:y,a),rot(x,b);
        else rot(x,a);
    }
    upd(x);
    return x;
}
inline int acs(int __x){
    int x=__x,y=0;do{
        splay(x);
        r[x]=y,upd(x);
        y=x,x=p[x];
    }while(x);
    return splay(__x);
}

inline int lca(int y, int __x){
    acs(y);int x=__x,z;y=0;do{
        splay(x); if(!p[x])z=x;
        r[x]=y,upd(x);
        y=x,x=p[x];
    }while(x);
    splay(__x);
    return z;
}

inline int rt(int x){for (x=acs(x);rls(x),l[x];x=l[x]);return splay(x);}
inline int ert(int x){rev(acs(x));return x;}

void Link(int x, int y){
    if (rt(x)==rt(y))puts(”-1”);
    else ert(x),p[x]=y;
    //splay(x),p[x]=y;//有根树
}
void Cut(int x){
    p[l[acs(x)]]=0,l[x]=0;//!
}
void Cut(int x, int y){
    if (x==y||rt(x)^rt(y))puts(”-1”);
    else ert(x),Cut(y);
}
void Query(int x, int y){
    if (rt(x)^rt(y))puts(”-1”);
    else{ert(x),OT(w2[acs(y)]);}
}
void Modify(int x, int y, int d){
    if (rt(x)^rt(y))puts(”-1”);
    else{ert(x),inc(acs(y),d);}
}

int hd[N], suc[M], to[M];
int n;
#define aa to[i^1]
#define bb to[i]
#define v bb
inline void dfs(int u){
    for(int i=hd[u];i;i=suc[i])if (!p[v]){
        p[v]=u, dfs(v);
    }
}

int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    while (~scanf(”%d”, &n)){

        REP_1(i, n) reset(i);
        memset(hd+1, 0, sizeof(int)*n);

        for(int i=2;i<n<<1;){
            RD(aa),RD(bb);
            suc[i] = hd[aa], hd[aa] = i++;
            suc[i] = hd[aa], hd[aa] = i++;
        }

        REP_1(i, n) RD(w1[i]); p[1]=1,dfs(1),p[1]=0;

        REP_1(i, n) ert(i);

        int a, b, cmd;Rush{
            RD(cmd),RD(a),RD(b);if (cmd==1) Link(a,b);
            else if(cmd==2) Cut(a,b);
            else if(cmd==3) Modify(b,RD(),a);
            else Query(a,b);
        }
        puts(””);
    }

    /*RD(n); char cmd[9]; int a; Rush{
        RS(cmd); RD(a); if (cmd[0]==’c’) Cut(a);
        else if (cmd[1]==’i’) Link(a, RD());
        else OT(lca(a, RD()));
    }*/
}
```

```
const int N = int(3e5)+9, M = 2*N;

struct node{

    static node* NIL; node *c[2], *p;
    int w1, w2, d0; bool r0;

#define NIL node::NIL
#define l c[0]
#define r c[1]
#define lx x->l
#define rx x->r
#define px x->p
#define ly y->l
#define ry y->r
#define py y->p

```

```
    void reset(){
        l = r = p = NIL;
        w1 = w2 = d0 = r0 = 0;
    }

    inline node(){
        reset();
    }

    inline void rev(){
        r0 ^= 1, swap(l, r);
    }

    inline void inc(int d){
        if (this == NIL) return;
        w1 += d, w2 += d, d0 += d;
    }

    inline void upd(){
        w2 = max(l->w2, w1, r->w2);
    }

    inline void rls(){
        //if (this == NIL) return;
        if (r0){
            l->rev(), r->rev();
            r0 = 0;
        }
        if (d0){
            l->inc(d0), r->inc(d0);
            d0 = 0;
        }
    }

    // 旋转

    inline int sgn(){return p->l==this?0:p->r==this?1:-1;}
    inline void setc(int d,node*x){c[d]=x,px=this;}

    inline void rot(int d){
        node *y = p, *z = py; if (~y->sgn()) z->setc(y->sgn(), this); else p = z;
        y->setc(!d, c[d]), setc(d, y), y->upd();
    }

    inline void rot(){rot(!sgn());}
    inline void zag(){rot(0);}
    inline void zig(){rot(1);}

    // 伸展

    inline void fix(){if(~sgn()) p->fix(); rls();}

    //*
    inline node* splay(){
        fix(); while (~sgn()) rot(); upd();
        return this;
    }
    /*/
    inline node* splay(){
        fix(); while (sgn() != -1){
            node *y = p, *z = py; if (y->sgn() == -1){ rot(); break;}
            if (z->l == y){
                if (y->l == this) y->zig(), zig();
                else zag(), zig();
            }else{
                if (y->r == this) y->zag(), zag();
                else zig(), zag();
            }
        }
        upd();
        return this;
    } /*/

    inline node* acs(){
        node *x = this, *y = NIL; do{
            x->splay();
            rx = y, x->upd();
            y = x, x = px;
        } while (x != NIL);
        return splay();
    }

    node* rt(){node* x; for (x = acs(); x->rls(), lx != NIL; x = lx); return x->splay();}
    node* ert(){acs()->rev(); return this;}


    void Link(node *x){
        if (rt() == x->rt()){
            puts(”-1”);
        }
        else {
            ert(), p = x;
        }
    }

    void Cut(){
        acs(); l->p = l = NIL;
    }

    void Cut(node* x){
        if (this == x || rt() != x->rt()){
            puts(”-1”);
        }
        else {
            ert(), x->Cut();
        }
    }

    void Query(node* x){
        if (rt() != x->rt()){
            puts(”-1”);
        }
        else {

            x->ert(); OT(acs()->w2);

            /*acs(); node *y = NIL; do{
                x->splay(); if (px == NIL) OT(max(rx->w2, x->w1, y->w2));
                rx = y, x->upd();
                y = x, x = px;
            } while (x != NIL);*/
        }
    }

    void Modify(node *x, int d){
        if (rt() != x->rt()){
            puts(”-1”);
        }
        else {

            x->ert(); acs()->inc(d);

            /*acs(); node *y = NIL; do{
                x->splay(); if (px == NIL) rx->inc(d), x->w1 += d, y->inc(d);
                rx = y, x->upd();
                y = x, x = px;
            } while (x != NIL);*/
        }
    }
} *NIL, *T[N];

int hd[N], suc[M], to[M];
int n;
#define aa to[i^1]
#define bb to[i]
#define v bb
inline void dfs(int u){
    REP_G(i, u) if (T[v]->p == NIL){
        T[v]->p = T[u], dfs(v);
    }
}

} using namespace LCT;

int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    NIL = new node(); REP_1(i, N) T[i] = new node();

    while (~scanf(”%d”, &n)){

        REP_1(i, n) T[i]->reset();
        memset(hd+1, 0, sizeof(int)*n);

        // Initializing Phase
        FOR_C(i, 2, n << 1){
            RD(aa, bb);
            suc[i] = hd[aa], hd[aa] = i++;
            suc[i] = hd[aa], hd[aa] = i;
        }

        REP_1(i, n) RD(T[i]->w1);
        T[1]->p = T[1], dfs(1), T[1]->p = NIL;

        //Interaction Phase
        int a, b, cmd; Rush{
            RD(cmd, a, b); if (cmd == 1) T[a]->Link(T[b]);
            else if (cmd == 2) T[a]->Cut(T[b]);
            else if (cmd == 3) T[b]->Modify(T[RD()], a);
            else T[a]->Query(T[b]);
        }

        puts(””);
    }
}
```

8.3.2 SPOJ QTREE. Query on a tree

```
const int N = int(1e4) + 9, M = 2 * N;

struct node{

    static node *NIL; node *c[2], *p;
    int w0, w1;

#define NIL node::NIL
#define l c[0]
#define r c[1]
#define lx x->l
#define rx x->r
#define px x->p
#define ly y->l
#define ry y->r
#define py y->p

    void reset(int v = 0){
        l = r = p = NIL;
        w0 = w1 = v;
    }

    node(int v = 0){
        reset();
    }

    void upd(){
        w1 = max(l->w1, w0, r->w1);
    }

    int sgn(){return p->l==this?0:p->r==this?1:-1;}
    void setc(int d,node*x){c[d]=x,px=this;}

    void rot(int d){
        node*y=p,*z=py;if(~y->sgn())z->setc(y->sgn(),this);else p=z;
        y->setc(d,c[!d]),setc(!d,y),y->upd();
    }

    void rot(){rot(sgn());}

    node* splay(){
        int a,b;while(~(a=sgn())){
            if(~(b=(p->sgn())))(a^b?this:p)->rot(a),rot(b);
            else rot(a);
        }
        upd();
        return this;
    }

    node* acs(){
        node *x = this, *y = NIL; do{
            x->splay();
            rx = y, x->upd();
            y = x, x = px;
        } while (x != NIL);
        return splay();
    }

    void query(node *x){
        acs(); node *y = NIL; do{
            x->splay(); if (px == NIL) OT(max(y->w1, rx->w1));
            rx = y, x->upd();
            y = x, x = px;
        } while (x != NIL);
        splay();
    }
    void modify(int w){
        splay(); w0 = w;
    }
} *NIL, *T[N];

int hd[N], suc[M], to[M], ww[N], id[N], n;
#define aa to[i^1]
```

```
#define bb to[i]
#define w ww[i/2]
#define v bb

inline void dfs(int u){
    REP_G(i, u) if (T[v]->p == NIL){
        T[v]->w0 = w, id[i/2] = v, T[v]->p = T[u], dfs(v);
    }
}

int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    NIL = new node(); REP(i, N) T[i] = new node();

    Rush{

        RD(n); fill(hd+1, hd+n+1, 0);

        for (int i=2;i<n<<1;){
            RD(aa, bb, w);
            suc[i] = hd[aa], hd[aa] = i++;
            suc[i] = hd[aa], hd[aa] = i++;
        }

        REP_1(i, n) T[i]->reset();
        T[1]->p = T[0]; dfs(1); T[1]->p = NIL;

        char cmd[10]; int x, y; while (1){
            RS(cmd); if (cmd[0] == ’D’) break; RD(x, y);
            if (cmd[0] == ’Q’) T[x]->query(T[y]);
            else T[id[x]]->modify(y);
        }
    }
}
```

8.3.3 SPOJ QTREE4. Query on a tree IV

```
const int N = int(1e5) + 9, M = 2 * N;

int _2nd(multiset<int>& S){
    multiset<int>::reverse_iterator it = S.rbegin(); ++it;
    return *it;
}

namespace LCT{

struct node{

    static node *NIL; node *c[2], *p; multiset<int> s0, s1;
    int dd, d0, w0; int ls, rs, ms; bool r0;

#define NIL node::NIL
#define l c[0]
#define r c[1]
#define lx x->l
#define rx x->r
#define px x->p
#define ly y->l
#define ry y->r
```

```
#define py y->p

    void reset(int v = 0){
        l = r = p = NIL; d0 = dd = 0;
        w0 = v, ls = rs = ms = -INF; CLR(s0, s1); s0.insert(-INF); s0.insert(-INF); s1.insert(-INF);
        r0 = 0;
    }

    inline node(){
        reset();
    }

    inline void rev(){
        r0 ^= 1; swap(l, r); swap(ls, rs);
    }

#define w3 (*s1.rbegin())
#define w2 (*s0.rbegin() + __2nd(s0))
#define w1 (*s0.rbegin())

    inline void upd(){
        dd = l->dd + d0 + r->dd; int m0 = max(w0, w1), ml = max(m0, l->rs+d0), mr = max(m0, r->ls);
        ls = max(l->ls, l->dd+d0+mr), rs = max(ml+r->dd, r->rs);
        ms = max(l->ms, l->rs+d0+mr, max(w2, w3, w0?-INF:m0), ml+r->ls, r->ms);
    }

    inline void rls(){
        /*if (r0){
            l->rev(), r->rev();
            r0 = 0;
        }*/
    }

    inline int sgn(){return p->l==this?0:p->r==this?1:-1;}
    inline void setc(int d,node*x){c[d]=x,px=this;}

    inline void rot(int d){
        node*y=p,*z=py;if(~y->sgn())z->setc(y->sgn(),this);else p=z;
        y->setc(d,c[!d]),setc(!d,y),y->upd();
    }
    inline void rot(){rot(sgn());}

    inline void fix(){if (~sgn()) p->fix(); rls();}
/*
    inline node* splay(){
        fix();while (~sgn()) rot(); upd();
        return this;
    }
*/

    inline node*splay(){
        fix();int a,b;while(~(a=sgn())){
            if(~(b=(p->sgn())))(a^b?this:p)->rot(a),rot(b);
            else rot(a);
        }
        upd();
        return this;
    }

    inline node* acs(){
        node *x = this, *y = NIL; do{
            x->splay();
            if (y != NIL) x->s0.erase(x->s0.find(y->ls)), x->s1.erase(x->s1.find(y->ms));
            if (rx != NIL) x->s0.insert(rx->ls), x->s1.insert(rx->ms);
            rx = y, x->upd();
            y = x, x = px;
        } while (x != NIL);
        return splay();
    }

    inline node* rt(){node* x; for (x = acs(); x->rls(), lx != NIL; x = lx); return x->splay();}
    inline node* ert(){acs()->rev(); return this;}

    void link(node *x){
        acs(); p = x; x->s0.insert(ls), x->s1.insert(ms); //x->upd();
    }

    void cut(){
        acs(); l->p = NIL, l = NIL;
    }

    void cut(node* x){
        ert(), x->cut();
    }

    void tog(){
        acs(); w0 = w0 ? 0 : -INF; //upd();
    }

} *NIL, *T[N];

int hd[N], suc[M], to[M], ww[N], n;
#define aa to[i^1]
#define bb to[i]
#define w ww[i/2]
#define v bb

inline void dfs(int u){
    REP_G(i, u) if (T[v]->p == NIL){
        T[v]->p = T[u], T[v]->d0 = w, dfs(v);
        T[u]->s0.insert(T[v]->ls); T[u]->s1.insert(T[v]->ms);
    }
    T[u]->upd();
}

} using namespace LCT;

int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif


    NIL = new node();
    //REP(i, N) T[i] = new node();

    while (~scanf(”%d”, &n)){

        //REP_1(i, n) T[i]->reset();
        FOR_1(i, 0, n) T[i] = new node();

        //fill(hd+1, hd+n+1, 0);
        FOR_C(i, 2, n << 1){
            RD(aa, bb); RDD(w);
            suc[i] = hd[aa], hd[aa] = i++;
            suc[i] = hd[aa], hd[aa] = i;
        }

        T[1]->p = T[0]; T[1]->d0 = 0; dfs(1); T[1]->p = NIL;

        Rush{
            switch(RC()){
                case ’A’:
                    T[1]->splay();
                    if (T[1]->ms < 0) puts(”They have disappeared.”);
                    else OT(T[1]->ms);
                    break;
                default:
                    T[RD()]->tog();
            }
        }
    }
}
```

8.3.4 SPOJ QTREE5. Query on a tree V

```
...
    static node *NIL; node *c[2], *p; multiset<int> s;
    int sz, w0; int ls, rs;

[#define]

    void reset(){
        l = r = p = NIL; sz = 0;
        w0 = ls = rs = INF; CLR(s); s.insert(INF);
    }

    inline node(){
        reset();
    }

    inline void upd(){
        sz = l->sz + 1 + r->sz; int m0 = min(w0, *s.begin());
        ls = min(l->ls, l->sz+1+min(m0, r->ls));
        rs = min(r->rs, r->sz+min(m0, l->rs+1));
    }

[旋转/伸展]

    inline node* acs(){
...
            if (y != NIL) x->s.erase(x->s.find(y->ls));
            if (rx != NIL) x->s.insert(rx->ls);
...
    }

    void tog(){
        acs(); w0 = w0 ? 0 : INF;
    }

    int Query(){
        acs();
        return rs == INF ? -1 : rs;
    }
```

8.3.5 SPOJ QTREE6. Query on a tree VI

```
const int N = int(1e5) + 9, M = 2 * N;

struct node{

    static node* NIL; node* c[2],* p;
    bool r0; int d0, w0;

```

```
#define NIL node::NIL
#define l c[0]
#define r c[1]
#define lx x->l
#define rx x->r
#define px x->p
#define ly y->l
#define ry y->r
#define py y->p

    void reset(){
        l = r = p = NIL;
        d0 = r0 = 0, w0 = 1;
    }
    node(){
        reset();
    }

    void rev(){
        r0 ^= 1, swap(l, r);
    }

    void inc(int d){
        if (this == NIL) return;
        w0 += d, d0 += d;
    }

    void upd(){
    }

    void rls(){
        if (r0){
            l->rev(), r->rev();
            r0 = 0;
        }
        if (d0){
            l->inc(d0), r->inc(d0);
            d0 = 0;
        }
    }

    int sgn(){return p->l==this?0:p->r==this?1:-1;}
    void setc(int d,node*x){c[d]=x,px=this;}

    void rot(int d){
        node*y=p,*z=py;if(~y->sgn())z->setc(y->sgn(),this);else p=z;
        y->setc(d,c[!d]),setc(!d,y),y->upd();
    }

    void rot(){rot(sgn());}

    void fix(){if (~sgn()) p->fix(); rls();}

    node* splay(){
        fix();int a,b;while(~(a=sgn())){
            if(~(b=(p->sgn())))(a^b?this:p)->rot(a),rot(b);
            else rot(a);
        }
        upd();
        return this;
    }

    node* acs(){
        node *x = this, *y = NIL; do{
            x->splay();
            rx = y, x->upd();
            y = x, x = px;
        } while (x != NIL);
        return splay();
    }

    node* rt(){node* x; for (x = acs(); x->rls(), lx != NIL; x = lx); return x->splay();}
    node* ert(){acs()->rev(); return this;}

    void link(node *x){
        splay(); x->acs(); p = x; x->inc(w0);
    }

    void cut(){
        acs(); l->inc(-w0); l->p = NIL, l = NIL;
    }

    int query(){
        node *x = rt()->r; for(; x->rls(), lx != NIL; x = lx);
        return x->w0;
    }
} *NIL, *T[2][N]; int col[N], fa[N];

#define TT(u) T[col[u]][u]

void Toggle(int u){
    TT(u)->cut(); col[u] ^= 1;
    TT(u)->link(T[col[u]][fa[u]]);
}

int hd[N], suc[M], to[M], n;
#define aa to[i^1]
#define bb to[i]
#define v bb

inline void dfs(int u){
    REP_G(i, u) if (TT(v)->p == NIL){
        TT(v)->p = T[col[v]][fa[v] = u], dfs(v);
        T[col[v]][u]->w0 += TT(v)->w0;
    }
}


int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    NIL = new node();

    while (~scanf(”%d”, &n)){

        FOR_1(i, 0, n) T[0][i] = new node(), T[1][i] = new node();

        for (int i=2;i<n<<1;){
            RD(aa, bb);
            suc[i] = hd[aa], hd[aa] = i++;
            suc[i] = hd[aa], hd[aa] = i++;
        }

        TT(1)->p = T[col[1]][0]; dfs(1);

        int _, u; Rush{
            switch(RD(_, u)){
                case 0:
                    OT(TT(u)->query());
                    break;
                default:
                    Toggle(u);
            }
        }
    }
}
```

8.3.6 SPOJ QTREE7. Query on a tree VII

```
const int N = int(1e5) + 9, M = 2 * N;

namespace LCT{

struct node{

    static node *NIL; node *c[2], *p; multiset<int> s;
    bool r0; int w0, w1;

#define NIL node::NIL
#define l c[0]
#define r c[1]
#define lx x->l
#define rx x->r
#define px x->p
#define ly y->l
#define ry y->r
#define py y->p

    void reset(){
        l = r = p = NIL; CLR(s); s.insert(-INF);
        r0 = 0; w0 = w1 = -INF;
    }

    node(){
        reset();
    }

    void rev(){
        r0 ^= 1; swap(l, r);
    }

    void upd(){
        w1 = max(l->w1, w0, r->w1, *s.rbegin());
    }

    void rls(){
        if (r0){
            l->rev(), r->rev();
            r0 = 0;
        }
    }

    int sgn(){return p->l==this?0:p->r==this?1:-1;}
    void setc(int d,node*x){c[d]=x,px=this;}

    void rot(int d){
        node*y=p,*z=py;if(~y->sgn())z->setc(y->sgn(),this);else p=z;
        y->setc(d,c[!d]),setc(!d,y),y->upd();
    }
    void rot(){rot(sgn());}

    void fix(){if (~sgn()) p->fix(); rls();}

    node* splay(){
        fix();int a,b;while(~(a=sgn())){
            if(~(b=(p->sgn())))(a^b?this:p)->rot(a),rot(b);
            else rot(a);
        }
        upd();
        return this;
    }

    node* acs(){
        node *x = this, *y = NIL; do{
            x->splay();
            if (y != NIL) x->s.erase(x->s.find(y->w1));
            if (rx != NIL) x->s.insert(rx->w1);
            rx = y, x->upd();
            y = x, x = px;
        } while (x != NIL);
        return splay();
    }

    node* rt(){node* x; for (x = acs(); x->rls(), lx != NIL; x = lx); return x->splay();}
    node* ert(){acs()->rev(); return this;}

    int query(){
        node *x = rt()->r; //for(; x->rls(), lx != NIL; x = lx);
        return x->w1;
    }

    void modify(int w){
        acs(); w0 = w;
    }

    void cut(){
        acs(); l->p = NIL, l = NIL;
    }

    void link(node *x){
        splay(); x->acs();
        p = x; x->r = this;
    }

} *NIL, *T[2][N]; int col[N], fa[N];
#define TT(u) T[col[u]][u]

int hd[N], suc[M], to[M], n;
#define aa to[i^1]
#define bb to[i]
#define v bb

inline void dfs(int u){
    REP_G(i, u) if (TT(v)->p == NIL){
        TT(v)->p = T[col[v]][fa[v] = u], dfs(v);
        T[col[v]][u]->s.insert(TT(v)->w1);
    }
    TT(u)->upd();
}

inline void Toggle(int u){
    TT(u)->cut(); col[u] ^= 1;
    TT(u)->link(T[col[u]][fa[u]]);
}

inline void Modify(int u, int w){
    T[0][u]->modify(w), T[1][u]->modify(w);
}

} using namespace LCT;
```

```
int main(){

#ifndef ONLINE_JUDGE
    freopen("in.txt", "r", stdin);
    //freopen("out.txt", "w", stdout);
#endif

    NIL = new node();

    REP(i, N) T[0][i] = new node(), T[1][i] = new node();

    while (~scanf("%d", &n)){

        FOR_1(i, 0, n) T[0][i]->reset(), T[1][i]->reset();

        RST(hd); FOR_C(i, 2, n << 1){
            RD(aa, bb);
            suc[i] = hd[aa], hd[aa] = i++;
            suc[i] = hd[aa], hd[aa] = i;
        }

        REP_1(i, n) RD(col[i]);
        REP_1(i, n) T[1][i]->w0 = RDD(T[0][i]->w0);

        TT(1)->p = T[col[1]][0]; dfs(1);
        T[col[1]][0]->s.insert(TT(1)->w1);

         int _, u; Rush{
            switch(RD(_, u)){
                case 0:
                    OT(TT(u)->query());
                    break;
                case 1:
                    Toggle(u);
                    break;
                default:
                    Modify(u, RDD());
            }
        }
    }
}
```

8.4 例题 (E.g.)

8.4.1 kMSS

```
const int N = 1 << 17, TN = 1 << 18;
// Segment Tree
int A[N], n, a, b, k;

#define root 1, 1, n
#define lx (x << 1)
#define rx (lx | 1)
#define mid (l + r >> 1)
#define lc lx, l, mid
#define rc rx, mid+1, r

struct _Seg{
    int s, l, r;
    _Seg(int s=0, int l=0, int r=0):s(s),l(l),r(r){}
    _Seg operator +(const _Seg& rhs)const{
        if (!rhs.s) return *this;
        if (!s) return rhs;
        return _Seg(s+rhs.s, l, rhs.r);
    }
    bool operator <(const _Seg& r)const{
        return s < r.s;
    }
};

inline void apply_swap(_Seg &l, _Seg &r){
    swap(l, r);
}

struct Seg{
    _Seg S, maxL, maxR, maxS, minL, minR, minS; bool neg;
    Seg(int s=0, int l=0, int r=0){
        maxL = maxR = maxS = s > 0 ? _Seg(s, l, r) : _Seg();
        minL = minR = minS = s < 0 ? _Seg(-s, l, r) : _Seg();
    }
    void apply_negative(){
        S.s = -S.s, neg ^= 1;
        swap(maxL, minL);
        swap(maxR, minR);
        swap(maxS, minS);
    }
} T[TN];

inline void update(Seg &x, const Seg &l, const Seg &r){
    x.S = l.S + r.S;
    x.maxL = max(l.maxL, l.S + r.maxL);
    x.maxR = max(l.maxR + r.S, r.maxR);
    x.maxS = max(l.maxS, r.maxS, l.maxR + r.maxL);
    x.minL = min(l.minL, l.S + r.minL);
    x.minR = min(l.minR + r.S, r.minR);
    x.minS = min(l.minS, r.minS, l.minR + r.minL);
}

inline void update(int x){
    update(T[x], T[lx], T[rx]);
}

inline void release(int x){
    if (x < n && T[x].neg){
        T[lx].apply_negative(), T[rx].apply_negative();
        T[x].neg = 0;
    }
}

void Build(int x, int l, int r){

    if (l == r){
        T[x] = Seg(A[l], l, r);
    }
    else {
        Build(lc), Build(rc);
        update(x);
    }
}

Seg Query(int x, int l, int r){
    if (a <= l && r <= b) return T[x];
    else {
        release(x);
        if (b <= mid) return Query(lc);
        if (mid < a) return Query(rc);
        Seg res; update(res, Query(lc), Query(rc));
        return res;
    }
}

```

```
void Negate(int x, int l, int r){
    if (a <= l && r <= b){
        T[x].apply_negative();
    }
    else {
        release(x);
        if (a <= mid) Negate(lc);
        if (mid < b) Negate(rc);
        update(x);
    }
}

void Negate(int a, int b){
    int _a = ::a, _b = ::b; ::a = a, ::b = b;
    Negate(root), ::a = _a, ::b = _b;
}

void Modify(int x, int l, int r){
    if (l == r){
        T[x] = Seg(b, l, r);
    }
    else {
        release(x);
        if (a <= mid) Modify(lc);
        if (mid < a) Modify(rc);
        update(x);
    }
}

int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    REP_1_C(i, RD(n)) RD(A[i]); n = cover_bit(n); Build(root); Rush{
        if (RD()){ // Query .. .
            RD(a, b, k); VII op; int res = 0; DO(k){
                _Seg cur = Query(root).maxS;
                if (!cur.s) break;
                res += cur.s, op.PB(MP(cur.l, cur.r));

                Negate(cur.l, cur.r);
            }

            ECH(it, op) Negate(it->fi, it->se);
            OT(res);
        }
        else { // Modify .. .
            RD(a, b);
            Modify(root);
        }
    }
}
```

Part II

动态规划 (Dynamic Programing)

模型？问题表示、抽象状态？状态设计不合理？寻找不变量、同阶段、阶段间？重新设计状态。

进一步优化必要？从状态入手？去除冗（状态合并）余（记忆化搜索）状态从转移入手？改变规划方向？减少决策数（满足斜率条件？满足凸完全单调性？上单调队列）？减少单次决策的时间复杂度（部分和？上数据结构）？

Chapter 9

常见模型

9.1 背包问题 (Knacpack)

9.2 最长不降子序列 (LIS)

```
template<class T> int LIS(int n, T* a){
    VI b; b.PB(a[0]); FOR(i, 1, n){
        if (b.back() < a[i]) b.PB(a[i]);
        else {
            b[lower_bound(ALL(b), a[i]) - b.begin()] = a[i];
        }
    }
    return SZ(b);
}

//template<class T, class C = __typeof less<T>()> int LIS(int n, T* a, C cmp = less<T>()){
template<class T, class C> int LIS(int n, T* a, C cmp){
    vector<T> b; b.PB(a[0]); FOR(i, 1, n){
        if (cmp(b.back(), a[i])) b.PB(a[i]);
        else {
            b[lower_bound(ALL(b), a[i], cmp) - b.begin()] = a[i];
        }
    }
    return SZ(b);
}

template<class T> int LISS(int n, T* a, int* pre, int& lst){
    VI b; b.PB(0); pre[0] = -1; FOR(i, 1, n){
        if (a[b.back()] < a[i]) pre[i] = b.back(), b.PB(i);
        else {
            int l = 0, r = SZ(b); while (l < r){
                int m = l + r >> 1;
                if (a[b[m]] < a[i]) l = m + 1;
                else r = m;
            }
            pre[i] = !r ? -1 : b[r-1];
            b[r] = i;
        }
    }
    lst = b.back();
    return SZ(b);
}

//template<class T, class C = __typeof less<T>()> int LISS(int n, T* a, int* pre, int& lst, C cmp = less<T>()){
template<class T, class C> int LISS(int n, T* a, int* pre, int& lst, C cmp){
    VI b; b.PB(0); pre[0] = -1; FOR(i, 1, n){
        if (cmp(a[b.back()], a[i])) pre[i] = b.back(), b.PB(i);
        else {
            int l = 0, r = SZ(b); while (l < r){
                int m = l + r >> 1;
                if (cmp(a[b[m]], a[i])) l = m + 1;
                else r = m;
            }
            pre[i] = !r ? -1 : b[r-1];
            b[r] = i;
        }
    }

    lst = b.back();
    return SZ(b);
}
```

9.3　最长公共子序列 (LCS)

9.4　例题 (E.g.)

9.4.1　HDU 3919. Little Sheep

题目描述 (Brief description)

算法分析 (Algorithm Analysis)

。。首先题目中的图有一定误导性。。在中间的任何时刻。。当前位置打开羊圈有两边的羊可以跑出时。。都立即打开当前位置的羊圈。。而不会出现图中那种跳跃的情况。。。注意到初始位置固定。。。所以状态是。。dp[l][r][2]表示向左走了l步。。向右走了r步。。且当前在区间左/右端点处时的最优值。。。。。状态类似青蛙的烦恼。（参见黑书p133。

。。转移的时候需要中间未被访问的部分的一段rmq。。。因为是静态。。所以选择离线ST。。去掉多余部分。。并做一些位移。。可以一定程度上避免状态转移的时候不小心写疵。。

```
const int N = 2009, LN = 14;

int ST[LN][N], dp[N][N][2];
int A[N], n, k, nn;

inline int rmq(int l, int r){
    r=nn-r; int lv = lg2(r-l);
    return max(ST[lv][l], ST[lv][r-(1<<lv)]);
}

int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif
    while (~scanf(”%d%d”, &n, &k)){

        int s = 0; REP(i, n) s += RD(A[i]); nn = n-(2*k+1); ++k;

        REP(i, nn) ST[0][i] = A[k+i];

        for ( int lv = 1 ; _1(lv) < nn ; lv ++ ){
            for ( int i = 0 ; i + _1(lv) <= nn ; i ++ )
                ST[lv][i] = max(ST[lv-1][i], ST[lv-1][i + _1(lv-1)]);
        }

        FLC(dp, 0x3f), dp[0][0][0] = dp[0][0][1] = 0;

        FOR_1(len, 1, nn) FOR_1(l, 0, len){
            int r = len - l;
            if (l) dp[l][r][0] = min(dp[l-1][r][0] + rmq(l-1, r), dp[l-1][r][1] + len * rmq(l-1, r));
            if (r) dp[l][r][1] = min(dp[l][r-1][1] + rmq(l, r-1), dp[l][r-1][0] + len * rmq(l, r-1));
        }

        int f = INF; FOR_1(l, 0, nn){
            int r = nn - l;
            REP(t, 2) checkMin(f, dp[l][r][t]);
        }

        OT(s + f);
    }
}
```

9.4.2 环状最长公共子序列 (CLCS)

简述 (Brief description)

...

分析 (Analysis)

1. 首先认识 LCS 与 CLCS 的关系。。
。。CLCS 至少不会比 LCS 简单。。可以通过将两个串前面各补 n 个 '-'。。从而使得 CLCS 求出的就是对应的 LCS。。。
。。CLCS 不比 LCS 难太多。。显然可以通过枚举 offset 。。用 LCS 来求 CLCS。。进一步。。只要枚举一个串的 offset 就行了。。

2. 格点最短路
.. LCS 的本质是 Gnm(对应的格点图)上的最短路。。。(。。这也可以解释为什么当 A[i] == B[j] 时。。从 dp[i-1][j-1] + 1 转移上来最优。。。我们初
。。。显然中间有很多信息可以重复利用。。。究竟如何利用呢？

3. lowerest shortest path tree。。。
。。。为了删除第一行的时候。。对我们的影响尽可能小。。。我们保留最低最短路径树 。。
。。考虑删除一行。。此时最短路径树最多被割成两个部分。。。

4. reroot。。。
。。。只有这两个部分的边界点。。父亲的方向会发生变化。。。（从 变成 ←
。。solved。。

```
const int dx[] = {0, -1, -1};
const int dy[] = {-1, -1, 0};

const int N = 1509;
char A[2*N], B[N]; int dp[2*N][N], p[2*N][N];
int n, m;

int lcs(int o){
    int i = n + o, j = m, d, res = 0;
    while (i != o && j){
        if ((d = p[i][j]) == 1) ++res;
        i += dx[d], j += dy[d];
    }
    return res;
}

void reroot(int o){
    int i = o, j = 1;

    while(j <= m && !p[i][j]) ++j; if (j > m) return;
    p[i++][j] = 0;

    while (i <= 2*n && j < m){
        if (p[i][j] == 2){
            p[i++][j] = 0;
        }
        else if (p[i][j+1] == 1){
            p[i++][++j] = 0;
        }
        else{
            ++j;
        }
    }

    while (i <= 2*n && p[i][j] == 2) p[i++][j] = 0;
}

int clcs(){

    REP_2_1(i, j, 2*n, m){
        if (A[i] == B[j]) dp[i][j] = dp[i-1][j-1] + 1;
        else dp[i][j] = max(dp[i-1][j], dp[i][j-1]);
        if (dp[i][j] == dp[i][j-1]) p[i][j] = 0;
        else if (A[i] == B[j] && dp[i][j] == dp[i-1][j-1] + 1) p[i][j] = 1;
        else p[i][j] = 2;
    }

    int res = 0; REP(i, n){
        checkMax(res, lcs(i));
        reroot(i);
    }
    return res;
}


int main(){

#ifndef ONLINE_JUDGE
    freopen("in.txt", "r", stdin);
    //freopen("out.txt", "w", stdout);
#endif

    while (~scanf("%s %s", A+1, B+1)){
        n = strlen(A+1), m = strlen(B+1);
        REP_1(i, n) A[n+i] = A[i]; A[2*n+1] = 0;
        int res = clcs(); reverse(B+1, B+m+1); checkMax(res, clcs());
        OT(2*res);
    }
}
```

9.4.3 Codeforces Round #207 Problem D. Bags and Coins

题目描述

构造 n 个结点的森林，使得总的权值和为 s。。
并且以每个结点为根的子树的权值和恰为 ai。

算法分析

```
const int N = int(7e4) + 9;

uint dp[2][(N>>5)+9]; int p[N]; bool rt[N];
int A[N], n, s; PII B[N];

int main(){

#ifndef ONLINE_JUDGE
    freopen("in.txt", "r", stdin);
    //freopen("print.txt", "w", stdprint);
#endif

    RD(n, s); REP_1(i, n) B[i] = MP(RD(A[i]), i);
    sort(B+1, B+n+1, greater<PII>());
    s -= B[1].fi; if (s < 0){puts("-1"); exit(0);}

    int nn = s/32 + 1; uint *cur = dp[0], *prv = dp[1]; cur[0] = 1;

    FOR_1(i, 2, n){

        swap(cur, prv); memcpy(cur, prv, sizeof(int)*nn);

        int o1 = B[i].fi >> 5, o2 = B[i].fi & 31;

        REP(ii, nn-o1){
            cur[ii+o1] |= prv[ii] << o2;
            if (o2) cur[ii+o1+1] |= prv[ii] >> (32-o2);
        }

        REP(ii, nn){
            for (uint s=cur[ii]^prv[ii];s;s^=low_bit(s)){
                uint j = low_idx(s);
                p[(ii<<5)+j] = B[i].se;
            }
        }

        if (p[s]) break;
    }

    if (s && !p[s]){puts(”-1”);exit(0);} do{
        rt[p[s]] = 1;
    }while (s -= A[p[s]]);

    RST(p); int ii = B[1].se; FOR_1(i, 2, n) if (!rt[B[i].se]){
        p[ii] = B[i].se, ii = B[i].se;
    }

    REP_1(i, n){
        if (p[i]) printf(”%d 1 %d\n”, A[i]-A[p[i]], p[i]);
        else printf(”%d 0\n”, A[i]);
    }
}
```

Chapter 10

数位

10.1 例题 (E.g.)

10.1.1 HDU 4507. 吉哥系列故事——恨 7 不成妻

简述 (Brief description)

求[l , r] 中
如果一个整数符合下面3个条件之一，那么我们就说这个整数和7有关——
1、整数中某一位是7；
2、整数的每一位加起来的和是7的整数倍；
3、这个整数是7的整数倍；

现在问题来了：吉哥想知道在一定区间内和7无关的数字的平方和。

分析 (Analysis)

```
const int N = 20;
int F0[N][2][7][7], F1[N][2][7][7], F2[N][2][7][7]; /// 是否出现了 7， 数位和%7，本身%7。。。
int Pow10[N]; int a[N], n;

#define v0 n-1, _7||i==7, (s+i)%7, (m*10+i)%7, 0
#define v1 n-1, _7||i==7, (s+i)%7, (m*10+i)%7, 1

int f0(int n, bool _7, int s, int m, bool b){
    if (n<0) return _7 || !s || !m;
    if (b){
        int res = 0; int up = a[n], i;
        REP_N(i, up) INC(res, f0(v0));
        INC(res, f0(v1));
        return res;
    }
    else {
        int &res = F0[n][_7][s][m];
        if (res == -1){
            res = 0; int up = 10, i;
            REP_N(i, up) INC(res, f0(v0));
        }
        return res;
    }
}

#define x pdt(Pow10[n], i)

int f1(int n, bool _7, int s, int m, bool b){
    if (n<0) return 0;
    if (b){
        int res = 0; int up = a[n], i;
        REP_N(i, up) INC(res, sum(f1(v0), pdt(f0(v0), x)));
        INC(res, sum(f1(v1), pdt(f0(v1), x)));
        return res;
    }
    else {
        int &res = F1[n][_7][s][m];
        if (res == -1){
            res = 0; int up = 10, i;
            REP_N(i, up) INC(res, sum(f1(v0), pdt(f0(v0), x)));
        }
        return res;
    }
}

int f2(int n, bool _7, int s, int m, bool b){
    if (n<0) return 0;
    if (b){
        int res = 0; int up = a[n], i;
        REP_N(i, up) INC(res, sum(f2(v0), pdt(f1(v0), x, 2), pdt(f0(v0), x, x)));
        INC(res, sum(f2(v1), pdt(f1(v1), x, 2), pdt(f0(v1), x, x)));
        return res;
    }
    else {
        int &res = F2[n][_7][s][m];
        if (res == -1){
            res = 0; int up = 10, i;
            REP_N(i, up) INC(res, sum(f2(v0), pdt(f1(v0), x, 2), pdt(f0(v0), x, x)));
        }
        return res;
    }
}

#undef x

int s2(LL x){
    int a = x % MOD, b = (x+1) % MOD, c = (2*x+1) % MOD;
    return pdt(a,b,c,_I(6));
}

int f(LL x){
    if (!x) return 0;
    n = 0; int s = s2(x); while (x) a[n++] = x % 10, x /= 10;
    return dff(s, f2(n-1, 0, 0, 0, 1));
}

int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif


    Pow10[0] = 1; FOR(i, 1, N) Pow10[i] = pdt(Pow10[i-1], 10);

    FLC(F0, F1, F2, -1);

    Rush{
        LL l, r; RD(l, r);
        OT(dff(f(r), f(l-1)));
    }
}
```

10.1.2 Divisibility

简述 (Brief description)

... 给定一个n-维Grid，每个格点的数值，是其 “下方” 所有格点的数值和。源点的数值为0。
。。求一个子矩形内，不能被P整除的格点总数。
(n < 8, 1 < P < 20 ...)

分析 (Analysis)

... 把坐标的每一维分量看成一个P进制整数。。。。那么对应格点的数值不被P整除就是数位不等于0。。
。。。于是subtask可以用数位DP。。
。。。对每一个维度。枚举是否低于边框。。。外层暴力容斥原理。。
。。需要注意的是这里 “限制” 因为不只一个数。。需要状态压缩。。~。。。

状态f(c, r, s, b)表示：
。。。当前考察第c个数位，第r维分量。。模P的和为s。。限制状态集合为b时的方案数。。

```
const int N = 8, M = 70, PP = 20;

LL lo[N], hi[N]; int bound[N][M];
int flag[M][N][PP][1<<N], flags;
int memo[M][N][PP][1<<N];
int n, P, nn;

#define bb (b & ~(i<bound[r][c] ? _1(r) : 0))

int go(int c, int r, int s, int b){
    if (r == n){
        if (++c == nn) return 1;
        r = 0, s = 0;
    }

    int &res = memo[c][r][s][b];
    if (flag[c][r][s][b] != flags){
        flag[c][r][s][b] = flags, res = 0;
        REP(i, P-s) if (!_1(b, r) || i <= bound[r][c]){
            INC(res, go(c, r+1, s+i, bb));
        }
    }
    return res;
}

int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    Rush{
        RD(n, P); REP(i, n) RD(lo[i]);
        LL maxv = 0; REP(i, n){
            RD(hi[i]); if (lo[i] > hi[i]) swap(lo[i], hi[i]);
            checkMax(maxv, hi[i]);
        }
        nn = 2; for (long long tmp = maxv; tmp; tmp /= P) ++nn;

        int ans = 0; REP(mask, _1(n)){
            REP(i, n){
                LL x = _1(mask, i) ? lo[i] - 1 : hi[i];
                DWN(j, nn, 0){
                    bound[i][j] = x % P;
                    x /= P;
                }
            }
            ++flags;
            if (count_bits(mask)&1) DEC(ans, go(0, 0, 0, _U(n)));
            else INC(ans, go(0, 0, 0, _U(n)));
        }
        OT(ans);
    }
}
```

Chapter 11

状压

11.1 例题 (E.g.)

11.1.1 POJ 2411. Mondriaan’s Dream

简述 (Brief description)

给定一个 $n \times m$ 的矩阵，问存在多少种完美 1×2 多米诺覆盖。

```
const int hh = 11, ss = 1 << hh;
long long f[2][ss], d;
int h, w, up;
int i, p, q, s;

void dfs(int j, int ss){
    while (s & 1<<j) j++;

    if (j >= w)
        f[p][ss] += d;
    else {
        if (!(s & 3<<j)) dfs(j + 2, ss);
        dfs(j + 1, ss | 1<<j);
    }
}

int main(){
    while (scanf(”%d%d”, &h, &w)==2 && h!=0){
        if ((h*w)&1) printf(”0\n”);
        else {
            memset(f, 0, sizeof(f)), up = 1 << w;
            p = d = 1, s = up, dfs(0, 0);

            for (int i = 1; i < h; i ++){
                q = p, p = 1 - p;
                memset(f[p], 0, sizeof(f[p]));
                for (s = 0; s < up; s ++)
                    if (f[q][s]) d = f[q][s], s += up, dfs(0, 0), s -= up;
            }
            printf(”%lld\n”, f[p][0]);
        }
    }
}
```

11.1.2 Game with Strings

简述 (Brief description)

...

分析 (Analysis)

```
const int N = 20, C = 26;

int adj[2*N-1][C]; char str[N][N+1];
int dp[2*N-1][1<<N];
int n;

#define ss (s&adj[k][cc])

int f(int k, int s, int c){
    int &res = dp[k][s];
    if (res == INF){
        if (k == 2*(n-1)) res = 0;
        else {
            if (k&1){
                res = -INF; s |= s << 1;
                REP(cc, C) if (ss){
                    checkMax(res, f(k+1, ss, cc));
                }
            }
            else{
                res = INF; s |= s << 1;
                REP(cc, C) if (ss){
                    checkMin(res, f(k+1, ss, cc));
                }
            }
        }
        if (c == 0) ++res; else if (c == 1) --res;
        //cout << k << ” ” << s << ” ” << c << ” ” << res << endl;
    }
    return res;
}

int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    REP_C(i, RD(n)) RS(str[i]);

    REP_2(i, j, n, n) if (i || j){
        adj[i+j-1][str[i][j]-’a’] |= _1(i);
    }

    FLC(dp, 0x3f); int res = f(0, 1, str[0][0]-’a’);
    puts(res ? (res > 0 ? ”FIRST” : ”SECOND”) : ”DRAW”);
}
```

11.1.3 SRM 619 1000

简述 (Brief description)

A 串与 B 串 k-相似的条件是分别从两个串中删除至多 k 个元素之后得到的串相等。。。现给定 A，B 串的长度 n 和字符集的大小 m 。。问有多少对串 2-相似。。。

分析 (Analysis)

先弱化：考虑 A 串 B 串已经给定，判断是否可行。。。那么我们有 dp[N][N][3][3]。。最后 [3][3] 表示删除的次数。。。对于固定的一个串。。他最后 dp[N][N][3][3] 的结果是固定的。。。我们把那些合法方案相同的串用同一个状态表示。。。状态压缩。。。。也就是 s = 1«9 。。。。

```
const int N = 109, M = 9;
map<PII, Int> dp[2]; int _b[5], b[5], p, q;

void decode(int x){
    REP(i, 4) b[i] = x%4, x/=4;
}

int encode(){
    int x = 0; DWN(i, 4, 0) x*=4,x+=b[i];
    return x;
}

void recode(){
    MII H; int n = 0; FOR(i, 1, 5){
        if (!CTN(H, _b[i])) H[_b[i]] = n++;
        b[i-1] = H[_b[i]];
    }
}

class SimilarSequencesAnother {
public:
    int getCount(int n, int m){

        p = 0, q = 1; RST(b); CLR(dp[p]); dp[p][MP(1, encode())] = m;
        b[3] = 1; dp[p][MP(1, encode())] = Int(m)*(m-1);

#define u (it->se)
#define v dp[p][MP(ss, encode())]

        DO(n){
            swap(p, q); CLR(dp[p]); ECH(it, dp[q]) if (u){
                int s = it->fi.fi; decode(it->fi.se); CPY(_b, b); int up1 = *max_element(b, b+4)+1;
                FOR_1_C_N(_b[4], 0, up1){
                    Int c1 = _b[4] == up1 ? m-up1 : 1; recode();
                    int up2 = max(_b[4]+1, up1); FOR_1(a, 0, up2){
                        Int c2 = a == up2 ? m-up2 : 1; int ss = 0; REP(i, 9) if (_1(s, i)){
                            int x = i%3, y = i/3; REP(dx, 2) REP(dy, 3){
                                int xx = x+dx, yy = y+dy; if (xx > 2 || yy > 2) continue;
                                if (dx || a == _b[2-xx+yy]) ss |= _1(yy*3+xx);
                            }
                        }
                    if (ss) v += u*c1*c2;
                    }
                }
            }
        }

        Int res = 0; ECH(it, dp[p]){
            decode(it->fi.se); if (!b[2] && !b[3]) res += u;
        }
        return res;
    }
};
```

11.1.4 2013-2014 ACM-ICPC, NEERC, Moscow Subregional Contest Problem J. Jigsaw Puzzle

简述 (Brief description)

给定 $n \times m$ 的棋盘 (max(n, m) <= 6)。问有多少种对棋盘的裁剪方案，使得存在完美多米诺覆盖。

分析 (Analysis)

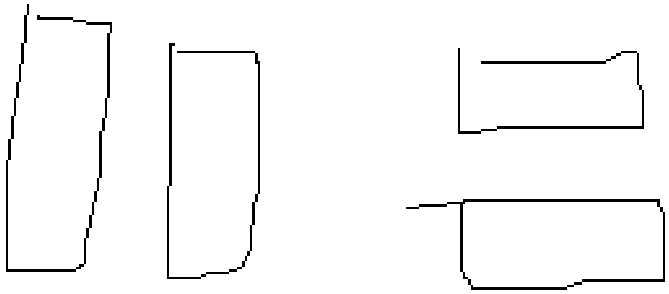
分析方法类似 SRM 619 的 1000，我们如法炮制。。
首先弱化：

考虑给定一个棋盘，问是否存在完美多米诺覆盖。除了二分图匹配之外，当然也可以状态压缩 DP。。。

考虑逐层 DP，s1 记录上一层是否有东西凸出来，s2 记录当前这一层的裁剪状态。。两个如果不冲突，并后取反，可以得到本层的 mask，（需要覆盖）。一共是 2^6。

回到原问题：初看我们状态一共是 2^{2^6} 种，考虑化简。

观察。。。



我们发现我们其实只需要计数，这两种情况并不需要加以区分，只需要保留兼容性更大的后一种即可。也就是那 2^6 状态里，只保留没有相邻 1 的状态，这样就只有 Fibnacci(m) 种了。

```
const int M = 6;

map<int, Int> dp[2]; VI adj[1<<M];
int p, q, n, m;

void dfs(int i, int s, VI &adj){
    if (i == M) adj.PB(s);
    else{
        if (i-1>=0 && _1(s, i) && _1(s, i-1)){
            dfs(i+1, s-_1(i)-_1(i-1), adj);
            if (i+1<m && _1(s, i+1)) dfs(i+1, s-_1(i)-_1(i+1), adj);
        }
        else{
            dfs(i+1, s, adj);
        }
    }
}

map<VI, int> H; VVI L; int h(VI &s){
    if (!CTN(H, s)) H[s] = SZ(L), L.PB(s);
    return H[s];
}

MII trans[1<<M];


int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    RD(n, m); if (n < m) swap(n, m); REP(s, _1(m)){
        dfs(0, s, adj[s]);
        //cout << s << ” ” << SZ(adj[s]) << ”: ” << endl;
        //ECH(it, adj[s]) cout << *it << ” ”; cout << endl;
    }

#define u (it->se)
#define v (dp[p][trans[s1][s0]])
#define s2 (*it)
#define s12 (_U(m)^(s1|s2))

```

```
    p = 0, q = 1; dp[p].clear(); VI s; s.PB(0); dp[p][h(s)] = 1; REP(i, n){
        swap(p, q); dp[p].clear(); ECH(it, dp[q]){
            int s0 = it->fi; REP(s1, _1(m)){
                if (!CTN(trans[s1], s0)){
                    VI s; ECH(it, L[s0]) if (!(s1&s2)) s.insert(s.end(), ALL(adj[s12])); UNQ(s);
                    trans[s1][s0] = h(s);
                }
                v += u;
            }
        }
    }

    Int z = 0; ECH(it, dp[p]) if (SZ(L[it->fi]) && !L[it->fi][0]) z += u; OT(z);
}
```

Chapter 12

组合

12.0.1 Facebook HackerCup 2013

简述 (Brief description)

给定一棵树，边的方向表示关联结点的大小关系。结点的标号是排列。。。。问有多少种合法的标号方案。。

分析 (Analysis)

$O(n^3)$，树状背包 + 组合 DP + 部分和

f[u][i]: 表示以 u 为根的子树中，小于 u 的点有 i 个的方案数。

。初始 f[u][0] = 1。。。我们枚举每一个孩子。。做树形分组背包。。
考虑新加入一组物品 v 。。。设 fu 缓存上一层 f[u] 的值。。。枚举背包容量 i 和物品容量 j 。

组合数的部分。。
对于容量 i 以内的物品。。有 j 个是从 v 处更新的。。(Binom(i, j)
对于容量 i 以外的物品。。有 sz[v] - j 是从 v 处更新的。。(Binom (sz[u]-1-i, sz[v]-j) 。。。
。。。另外 fu[i-j] 也是常量。。。把这部分记作一个转移因子。 b。。。

。。考虑 u < v ：
。。。则 f[v][j..sz[v]) 都可以提供容量为 j 的物品。。。。

若 u > v：
。。。。。 v 本身（包括所有小于等于 j 的数都）必然被添加进 u 中。。因此 j 至少是 1。。
。。此时只有 f[v][0..j)。。才可以提供容量为 j 的物品。。。

```
 //}/* .............................................................................................................. */

const int N = 1009;

Int Fact[N], Factt[N]; Int Binom(int n, int m){
    return Fact[n] * Factt[m] * Factt[n-m];
}

bool le[N][N]; VI adj[N];
Int f[N][N], fu[N]; int sz[N];
int n;

#define v (*it)
#define b (fu[i-j]*Binom(i,j)*Binom(sz[u]-1-i,sz[v]-j))

void dfs(int u, int p = -1) {

sz[u] = 1; ECH(it, adj[u]) if (v != p){
   dfs(v, u), sz[u] += sz[v];
}

```

```
REP(i, sz[u]) f[u][i] = fu[i] = 0; sz[u] = f[u][0] = 1;

    ECH(it, adj[u]) if (v != p){

        REP(i, sz[u]) fu[i] = f[u][i], f[u][i] = 0; sz[u] += sz[v];

        if (le[u][v]){
            REP(i, sz[u]) REP(j, min(sz[v], i+1))
                f[u][i] += (f[v][sz[v]] - f[v][j]) * b;
        }
        else{
            REP(i, sz[u]) REP_1(j, min(sz[v], i))
                f[u][i] += f[v][j] * b;
        }
    }

    DWN_1(i, sz[u], 1) f[u][i] = f[u][i-1]; f[u][0] = 0; REP_1(i, sz[u]) f[u][i] += f[u][i-1];
}

int main() {

    Fact[0] = 1; REP_1(i, N-1) Fact[i] = Fact[i-1] * i; Factt[N-1] = _I(Fact[N-1]); DWN(i, N, 1) Factt[i-1] = Factt[i] * i;

    //freopen(”in.txt”, ”r”, stdin);
freopen(”permutations.txt”,”r”,stdin);
freopen(”out2.txt”,”w”,stdout);

    Rush{

        REP_C(i, RD(n)) CLR(adj[i]);

        int x, y; char c; DO(n-1){
            RD(x), RC(c), RD(y);
            le[x][y] = c == ’<’;
            le[y][x] = c == ’>’;
            adj[x].PB(y), adj[y].PB(x);
        }

        dfs(0); OT(f[0][n]);
    }
}
```

Chapter 13

插头

Part III

状态空间搜索 (State Space Search)

13.1 补充

13.1.1 最大团

Part IV

图论 (Graph Theory)

13.1.2 HDU 3686. Traffic Real Time Query System

简述 (Brief description)

求出所有的边双连通分量，即缩点，然后计算缩点以后图度数为 1 个结点的个数 N，答案就是 (N+1)/2，可以证明不过简单的方法用一次 tarjan 就可以解决，代码如下：

分析 (Analysis)

```
//}/* .......................................................................................................... */

const int N = int(1e5) + 9, M = int(1e5) + 9, QN = int(1e4) + 9;

VI adj[N]; int cut[N], dep[N]; VII lca[N]; int ans[QN];
int dfn[N], low[N], tt, nn; stack<int> v_sta, e_sta;
int v_bj[N], e_bj[M];
int hd[N], prd[M*2], suc[M*2], to[M*2];
int n, m, q;

#define aa to[i^1]
#define bb to[i]
#define v bb

#define vis dfn

void del(int i){
    if (i == hd[aa]) prd[hd[aa] = suc[i]] = 0;
    else suc[prd[suc[i]] = prd[i]] = suc[i];
}

void add(int x, int y){
    adj[x].PB(y), adj[y].PB(x);
}

int new_node(int x = 0){
    cut[++nn] = x, CLR(adj[nn]);
    return nn;
}

void tarjan_bcc(int u){
    dfn[u] = low[u] = ++tt, v_sta.push(u); bool fb = 1;

    REP_G(i, u){
        e_sta.push(i/2), del(i^1);
        if (!vis[v]){

            tarjan_bcc(v);
            checkMin(low[u], low[v]);

#define uu v_bj[u]

            if (low[v] >= dfn[u]){

                if (fb) fb = 0, uu = new_node(1);
                add(uu, new_node());

                while (!v_sta.empty()){
                    int t = v_sta.top(), &tt = v_bj[t]; v_sta.pop();
                    if (tt) add(tt, nn); else tt = nn;
                    if (t == v) break;
                }

                while (!e_sta.empty()){
                    int t = e_sta.top(), &tt = e_bj[t]; e_sta.pop();
                    tt = nn;
                    if (t == i/2) break;
                }
            }
        }
        else {
            checkMin(low[u], dfn[v]);
        }
    }
}

#undef v
#define v (*it)

namespace DSU{
    int P[N];

    int Find(int x){
        return P[x] == x ? x : P[x] = Find(P[x]);
    }
    void Init(){
        REP_1(i, nn) P[i] = i;
    }
} using namespace DSU;


void tarjan_lca(int u, int p = -1){

    dep[u] += cut[u];

    ECH(it, adj[u]) if (v != p){
        dep[v] = dep[u], tarjan_lca(v, u);
        P[Find(v)] = u;
    }

    vis[u] = 1;

#undef v
#define v (*it).fi
#define id (*it).se

    ECH(it, lca[u]) if (vis[v]){
        int z = P[Find(v)];
        ans[id] = dep[u] + dep[v] - 2*dep[z]+cut[z];
    }
}

#undef id
#undef v
#define v bb

int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    while (RD(n, m)){

        RST(hd); FOR_1_C(i, 2, m << 1){
            RD(aa, bb);
            hd[aa] = prd[suc[i] = hd[aa]] = i, ++i;
            hd[aa] = prd[suc[i] = hd[aa]] = i;
        }

        RST(v_bj, vis), tt = nn = 0; REP_1(i, n) if (!vis[i]){
            CLR(v_sta, e_sta);
            tarjan_bcc(i);
        }

        REP_1(i, nn) CLR(lca[i]); REP_C(i, RD(q)){
            int x = e_bj[RD()], y = e_bj[RD()];
            if (x == y) ans[i] = 0;
            else {
                lca[x].PB(MP(y, i));
                lca[y].PB(MP(x, i));
            }
        }

        Init(); RST(vis, dep); REP_1(i, nn) if (!vis[i]) tarjan_lca(i);
        REP(i, q) OT(ans[i]);
    }
}
```

13.1.3 k-联通分量

```
 const int N = 109;
int C[N][N], CC[N][N], cut[N], prd[N], suc[N], tmp[N];
int n, m, nn, n0, s, t, tt; int cc, K;

int PP[N], P[N];

inline void Make(int x){P[x] = x;}
inline int Find(int x){return x == P[x] ? x : P[x] = Find(P[x]);}
inline void Unionn(int x, int y){P[x] = y;}
inline void Union(int x, int y){int xx = Find(x), yy = Find(y); Unionn(xx, yy);}

inline void del(int x){prd[suc[prd[x]] = suc[x]] = prd[x];}
inline void rsm(int x){prd[suc[suc[prd[x]] = x]] = x;}

#define hd suc[0]

void Extract(int &s){
    s = hd; REP_L(i, suc[s], suc) if (cut[i] > cut[s]) s = i;
}

void Prim(){
    REP_L(i, hd, suc) cut[i] = 0; tt = 0; DO(nn-1){
        Extract(s); del(s); tmp[tt++] = s;
        REP_L(i, hd, suc) cut[i] += CC[s][i];
    }
    Extract(t); DWN(i, tt, 0) rsm(tmp[i]); del(t);
}

bool Stoer_Wagner(VI& I, VI& A, VI& B){

    REP(i, SZ(I)) FOR(j, i+1, SZ(I)) CC[I[i]][I[j]] = CC[I[j]][I[i]] = C[I[i]][I[j]];
    nn = SZ(I); I.PB(0); REP(i, nn) suc[I[i]] = I[i+1], prd[I[i+1]] = I[i], Make(I[i]);
    prd[hd = I[0]] = 0; I.pop_back(); DO(nn-2){
        Prim(), --nn; if (cut[t] < K){
            REP(i, SZ(I)) if (Find(I[i]) == Find(t)) A.PB(I[i]); else B.PB(I[i]);
            return 1;
        }
        REP_L(i, hd, suc) CC[i][s] = CC[s][i] += CC[t][i]; Union(s, t);
    }

    Prim(); if (cut[t] < K){
        REP(i, SZ(I)) if (Find(I[i]) == Find(t)) A.PB(I[i]); else B.PB(I[i]);
        return 1;
    }
    return 0;
}
```

```


int keCC(VI &I){

    if (SZ(I) <= 1) return SZ(I);
    VI A, B; if (!Stoer_Wagner(I, A, B)) return 1;

    /*REP_2(i, j, SZ(A), SZ(B)){
        int x = A[i], y = B[j];
        C[x][y] = C[y][x] = 0;
    }*/

    return keCC(A) + keCC(B);
}

int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    while (~scanf(”%d%d%d”, &n, &m, &K)){

        RST(C); DO(m){
            int x, y; RD(x, y);
            ++C[x][y], ++C[y][x];
        }

        VI I; REP_1(i, n) I.PB(i);
        OT(keCC(I));
    }
}
```

13.2 例题 (e.g.)

13.2.1 圆桌骑士

题目描述 (Brief description)

给定一个无向图，问有多少结点不在任何一个简单奇圈上。

算法分析 (Algorithm Analysis)

简单圈上的所有结点必处在同一个双联通分量上。双联通分解, 忽略二分图。对于非二分图，尽管其中包含了一些奇圈，但如何判定一个节点恰好处在某个奇圈上呢？是否一个结点一定可以处在一个奇圈上呢？

答案是肯定的。

外部链接 (External Link)

13.2.2 Mining Your Own Business

题目描述 (Brief description)

给定一个无向图上，选择尽量少。。。

算法分析 (Algorithm Analysis)

。。。

外部链接 (External Link)

Chapter 14

最短路 ()

```
const int N = 100009, M = 1000009;
int bg[M], ed[M], to[M]; int D[N], P[N]; VI adj[N];
int n, m;

int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    while (~scanf(”%d”, &n)){

        m = 0; REP_1(u, n){
            D[u] = 0; DO(RD(P[u])){
                RD(bg[m]); bg[m] *= 60; bg[m] += RD();
                RD(ed[m]); ed[m] *= 60; ed[m] += RD();
                RD(to[m]); adj[u].PB(m++);
            }
            --P[u];
        }

#define arc adj[u][P[u]]
#define v to[arc]

        VII res; priority_queue<PII, VII, greater<PII> > Q; FLC(D, 0x3f); int Dn = INF; while (~P[1]){
            int u = 1; D[u] = bg[arc], Q.push(MP(D[u], u)); while (!Q.empty()){
                int u = Q.top().se; Q.pop();
                for (;~P[u]&&D[u]<=bg[arc];--P[u]) if (D[v] > ed[arc]){
                    D[v] = ed[arc];
                    Q.push(MP(D[v], v));
                }
            }

            if (D[n] != Dn){
                Dn = D[n];
                res.PB(MP(D[1], Dn));
            }
        }

        OT(SZ(RVS(res))); ECH(it, res)
            printf(”%02d:%02d %02d:%02d\n”, it->fi/60, it->fi%60, it->se/60, it->se%60);
    }
}
```

Chapter 15

生成树

15.1 最小生成树 (MST)

环切性质：树 $T = (V, E)$ 是 MST $\Leftrightarrow \forall e \in E, e' \notin E, w(e) \leq w(e')$。（贪心构造 / 检验）

15.1.1 Prim

邻接表（矩阵）、priority_queue<PII, VII, greater<PII> >、迭代。
适和于稠密图，时间复杂度是 $O(n^2)$、O(nlogn)（二叉堆优化）。

15.1.2 Kruskal

边表、重载比较函数、并查集、process()、sort、迭代。
适和于稀疏图，时间复杂度 $O(mlogm)$。最大边最小、且途中是最小生成森林。

15.2 次小

15.3 度限制

约定结点 1 为度限制的结点。。度限制为至多 K。
先不考虑这个结点，求生成森林。（Kruskal 的话还是一趟。）。之后从 1 开始向每个森林连边（dfs1）。则第一阶段结束后 1 。我们得到了一个度限制为 K0 的最小生成树。。（初始生成森林的连通块个数）要想进一步得到度限制为 K 的最小生成树，还需要进行 K - K0 次“差额最小添删操作” ...
... 其实就是贪心替换、、。。设 w1[u] 为 u 结点连到根结点的边的权值（可能输入的时候有重边。。保留最小的。）。从根结点 dfs2 下去。。得到每个结点到根结点路径删最大的边的权值和编号。mx[u], si[u]。。。则选择 u 结点进行替换的话。。产生的收益就是 mx[u] - w1[u] 。。。枚举每个结点。。找出收益最大的。。。迭代 K - K0 次即可。。（若某轮收益已经为 0。。则可直接 break 掉。。）
————————————————————————————实现过程中需要一个边表用来跑 Kruskal 。。一个支持 del 操作的手写邻接表、。。。

```

namespace DC_MST{

    const int N = 109, M = 10 * N * N;

    int hd[N], prd[M], suc[M], to[M], ww[M/2], m;
    int P[N], mst, n;

#define aa to[i^1]
#define bb to[i]

    inline void del(int i){
        if (!prd[i]) prd[hd[aa] = suc[i]] = 0;
        else suc[prd[suc[i]] = prd[i]] = suc[i];
    }

    inline void dell(int i){
        del(i), del(i^1);
    }

    inline void add(int x, int y, int w){
        ww[m>>1] = w, prd[hd[x]] = m, suc[m] = hd[x], hd[x] = m, to[m++] = y;
        swap(x, y), prd[hd[x]] = m, suc[m] = hd[x], hd[x] = m, to[m++] = y;
    }

    inline void Make(int x){P[x] = x;}
    inline int Find(int x){return P[x] == x ? x : P[x] = Find(P[x]);}
    inline void Unionn(int x, int y){P[y] = x;}

    struct Edge{
        int x, y, w; Edge(int x, int y, int w):x(x),y(y),w(w){}
        bool operator<(const Edge& r) const{return w < r.w;}
        void process(){
            int xx = Find(x), yy = Find(y); if (xx != yy){
                mst += w, Unionn(xx, yy);
                add(x, y, w);
            }
        }
    }; vector<Edge> E;

    void Kruskal(){
        m = 2, RST(hd); mst = 0; SRT(E); REP_1(i, n) Make(i);
        ECH(it, E) it->process();
    }

    bool vis[N]; int w1[N], mx[N], sw[N], si[N], uu;

#define v bb
#define w ww[i>>1]
    void dfs1(int u){
        vis[u] = 1; REP_G(i, u) if (!vis[v]){
            if (w1[v] < w1[uu]) uu = v;
            dfs1(v);
        }
    }
    void dfs2(int u = 1){
        vis[u] = 1; REP_G(i, u) if (!vis[v]){
            if (w > mx[u]) mx[v] = w, si[v] = i; else mx[v] = mx[u], si[v] = si[u];
            sw[v] = mx[v] - w1[v]; dfs2(v);
        }
    }
#undef w
#undef v

    int dc_mst(int K){
        Kruskal(); RST(vis); FOR_1(u, 2, n) if (!vis[u]){
            uu = u, dfs1(u), --K; add(1, uu, w1[uu]), mst += w1[uu];
        }

        DO(K){
            int dd = 0, ii, uu; RST(vis, mx); dfs2();
            FOR_1(u, 2, n) if (sw[u] > dd){
                dd = sw[u], ii = si[u], uu = u;
            }
            if (!dd) break;
            mst -= dd, dell(ii), add(1, uu, w1[uu]);
        }

        return mst;
    }


} using namespace DC_MST;

map<string, int> H;

```

```
inline int h(string s){
    if (!H[s]) H[s] = ++n;
    return H[s];
}

int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    int mm; while (~scanf(”%d”, &mm)){
        CLR(H, E); H[”Park”] = 1; n = 1; FLC(w1, 0x3f); DO(mm){
            string s1, s2; int w; cin >> s1 >> s2 >> w;
            int x = h(s1), y = h(s2); if (x > y) swap(x, y);
            if (x == 1) checkMin(w1[y], w); else E.PB(Edge(x, y, w));
        }

        printf(”Total miles driven: %d\n”, dc_mst(RD()));
    }
}
```

15.4 树形图

SRM 584 FoxTheLinguist.cpp

题目描述 (Brief description)

。。。有m种语言，开始都是0级。。有一些课程。。课程是形如。。。(Ai, Bj, w)的形式。。（如果A语言够i级。。那么支付w费用。。可以令B语言到达j级。。。。问所有语言都到达9级。。至少需要多少花费。。

算法分析 (Algorithm analysis)

。。注意到是树形图就行了。。。。建模是。。每个语言拆成[0, 9]十个点。。。。添加一个根结点。。连到所有语言的0级。。代价为0。。。。。。然后对每个语言的除了0以外的等级。。都向低一级连一条代价为0的边。。。。。（。如果边的形式是从一个语言集合到另一个语言还能做么。。

```
        const int N = 500 * 2 + 9;

int G[N][N], Cur[N], fa[N], ww[N], bj[N];
int n, nn, mst;

void Gen(){
    Cur[n++] = nn++;
}
void Del(int x){
    swap(Cur[x], Cur[--n]);
}

#define u Cur[i]
#define v Cur[j]
#define uu bj[u]
#define vv bj[v]

bool Find(){
    int _n = n, _nn = nn; RST(fa); FLC(ww, 0x3f);

    REP_2(i, j, n, n) if (i != j && G[u][v] < ww[v])
        fa[v] = u, ww[v] = G[u][v];

    //REP(i, n) cout << ww[u] << ” ”; cout << endl;

    bool found = 0; RST(bj), bj[0] = -1;

    FOR(i, 1, _n) if (!bj[u]){
        int x = u; do{bj[x] = u, x = fa[x];} while (!bj[x]);
        if (bj[x] == u){
            found = 1; do{bj[x] = nn, mst += ww[x = fa[x]];} while (bj[x] != nn);
            Gen();
        }
    }


    REP(i, _n) if (bj[u] < _nn) bj[u] = 0;
    return found;
}

void Melt(){

    REP(i, n) if (uu){ // Circle Canceling ...
        REP(j, n) if (vv != uu){
            if (vv) checkMin(G[uu][vv], G[u][v] - ww[v]);
            else checkMin(G[uu][v], G[u][v]);
        }
    }
    else {
        REP(j, n) if (vv){
            checkMin(G[u][vv], G[u][v] - ww[v]);
        }
    }

    REP(i, n) if (uu) Del(i--);
}

#undef vv
#undef uu
#undef u
#undef v

void dfs(int u = 0){
    bj[u] = 1; REP(v, n) if (G[u][v] != INF && !bj[v]) dfs(v);
}

int dMST(){
    RST(bj); dfs();
    FOR(i, 1, n) if (!bj[i]) return -1;

    REP(i, n) Cur[i] = i; nn = n, mst = 0;
    while (Find()) Melt();
    FOR(i, 1, n) mst += ww[Cur[i]];
    return mst;
}

class FoxTheLinguist {
public:
    int minimalHours(int _n, vector <string> courseInfo) {

        n = _n * 10 + 1; FLC(G, 0x3f); REP(i, _n){
            G[0][i*10+1] = 0; REP_1(j, 9) checkMin(G[i*10+j+1][i*10+j], 0);
        }

        string s; s = accumulate(ALL(courseInfo), s);
        REP(i, SZ(s)) if (s[i]==’-’ || s[i]==’>’ || s[i]==’:’) s[i] = ’ ’;
        istringstream iss(s); for (string s1, s2, s3; iss >> s1 >> s2 >> s3;){
            int u = (s1[0]-’A’) * 10 + (s1[1]-’0’) + 1;
            int v = (s2[0]-’A’) * 10 + (s2[1]-’0’) + 1;
            int w = s3[0] * 1000 + s3[1] * 100 + s3[2] * 10 + s3[3] - ’0’ * 1111;
            checkMin(G[u][v], w);
        }

        return dMST();
    }
};
```

15.5 例题

Chapter 16

匹配 (Match)

16.1 Hungary

```
VI adj[N]; bool vy[N]; int py[N];
int n;

#define y (*it)
#define vis vy[y]
#define p py[y]
bool dfs(int x){
    ECH(it, adj[x]) if (!vy[y]){
        vis = 1;
        if (!p || dfs(p)){
            p = x;
            return 1;
        }
    }
    return 0;
}
```

16.2 KM-1

```
const int N = 109;

DB W[N][N], lx[N], ly[N], slack[N], delta; int p[N]; bool vx[N], vy[N];
int n;

void init(){
    RD(n); REP_2(i, j, n, n) RD(W[i][j]);
}

#define w(x, y) (lx[x] + ly[y] - W[x][y])

bool dfs(int x){
    vx[x] = true; REP(y, n) if (!vy[y]){
        if (!sgn(w(x, y))){
            vy[y] = true; if(!~p[y]||dfs(p[y])){
                p[y] = x;
                return true;
            }
        }
        else {
            checkMin(slack[y], w(x, y));
        }
    }
    return false;
}

void KM(){

    FLC(p, -1); RST(lx, ly); REP_2(i, j, n, n) checkMax(lx[i], W[i][j]);

    REP(x, n){
        fill(slack, slack+n, OO); while (1){
            RST(vx, vy); if (dfs(x)) break;
            DB delta = OO; REP(i, n) if (!vy[i]) checkMin(delta, slack[i]);
            REP(i, n){
                if (vx[i]) lx[i] -= delta;
                if (vy[i]) ly[i] += delta; else slack[i] -= delta;
            }
        }
    }
}
```

16.3 KM-2

```
const int N = 109;

DB W[N][N], lx[N], ly[N], slack[N]; int px[N], py[N], pxx[N], Q[N], op, cz; bool vx[N], vy[N];
int n;

#define w(x, y) (lx[x] + ly[y] - W[x][y])

void add_to_tree(int x, int xx){
    vx[Q[op++] = x] = true, pxx[x] = xx;
    REP(y, n) checkMin(slack[y], w(x, y));
}

void KM(){

    FLC(px, py, -1), RST(lx, ly);
    REP_2(i, j, n, n) checkMax(lx[i], W[i][j]);

    REP(root, n){ // 1. Designate each exposed (unmatched) node in V as the root of a Hungarian tree.

        int x, y; while (1){

            RST(vx, vy), op = cz = 0;
            add_to_tree(x = root, -1);
            REP_N(y, n) slack[y] = w(x, y);

            while (cz < op){ // 2. Grow the Hungarian trees rooted at the exposed nodes in the equality subgraph.
                x = Q[cz++]; REP_N(y, n) if (!sgn(w(x, y)) && !vy[y]){
                    if (py[y] == -1) goto Augment;
                    vy[y] = true, add_to_tree(py[y], x);
                }
            }

            DB delta = OO; // 3. Modify the dual variables lx and ly as follows to add new edges to the equality subgraph.
            REP(i, n) if (!vy[i]) checkMin(delta, slack[i]);
            REP(i, n){
                if (vx[i]) lx[i] -= delta;
                if (vy[i]) ly[i] += delta; else slack[i] -= delta;
            }
        }

        assert(0); // !! Impossible Position !!.. No Perfect Matching found.

        Augment: for (int t;x!=-1;x=pxx[x],y=t) // 4. Augment the current matching by flipping matched and unmatched edges along the selected augmenting path.
```

```
            t = px[x], py[y] = x, px[x] = y;
    }
}
```

16.4 EBC

Edmonds Blossom-Contraction Algorithm 一般图最大匹配

```
const int dx[] = {-2, -2, -2, -2, -1, -1, -1, -1, -1, 0, 0, 1, 1, 1, 1, 1, 2, 2, 2, 2},
          dy[] = {-2, -1, 1, 2, -2, -1, 0, 1, 2, -1, 1, -2, -1, 0, 1, 2, -2, -1, 1, 2};

const int NN = 15, N = NN*NN;

int P[N], F[N], B[N], Q[N], cz, op;
bool G[N][N], InB[N], inQ[N]; int mark[N], tot;
int n, s, t;

#define Pi P[i]
#define pre F[Pi]

int lca(int u, int v){
    ++tot;
    for (int i=u;i;i=pre){ i=B[i]; mark[i]=tot; }
    for (int i=v;i;i=pre){ i=B[i]; if (mark[i]==tot) return i;}
}

void Bls(int u, int v){

    int z = lca(u, v); RST(InB);

    for (int i=u;B[i]!=z;i=pre){
        if (B[pre]!=z) F[pre]=Pi; //对于BFS树中的父边是匹配边的点，F向后跳
        InB[B[i]]=InB[B[Pi]]=1;
    }
    for (int i=v;B[i]!=z;i=pre){
        if (B[pre]!=z) F[pre]=Pi; //同理
        InB[B[i]]=InB[B[Pi]]=1;
    }

    if (B[u]!=z) F[u]=v; //注意不能从z这个奇环的关键点跳回来
    if (B[v]!=z) F[v]=u;

    REP_1(i, n) if (InB[B[i]]){
        B[i]=z; if (!inQ[i]){
            Q[op++]=i;
              inQ[i]=true; //要注意如果本来连向BFS树中父结点的边是非匹配边的点，可能是没有入队的
          }
      }
}

void Chg(){
    int x,y,z=t; while (z){
        y=F[z], x=P[y];
        P[y]=z, P[z]=y, z=x;
    }
}

bool bfs(){

    RST(F, inQ); REP_1(i, n) B[i] = i;
    Q[cz=0]=s, op=1, inQ[s]=1;

    while (cz < op){
        int u = Q[cz++];
        REP_1(v, n) if (G[u][v] && B[u]!=B[v] && P[u]!=v){
            if (s==v || P[v] && F[P[v]]) Bls(u, v);
            else if (!F[v]){
                F[v] = u; if (P[v]){
                    Q[op++] = P[v];
                    inQ[P[v]] = 1;
                }
                else{
                    t = v, Chg();
                    return 1;
                }
            }
        }
    }
    return 0;
}

int ebc(){
    int z=0; RST(P); REP_1_N(s, n) if (!P[s]) if (bfs()) ++z;
    return z;
}

char Map[NN][NN+1]; int nn, mm;

bool inGrid(int x, int y){
    return x >= 0 && y >= 0 && x < nn && y < mm;
}

int Id[NN][NN];

int id(int x, int y){
    //return x*mm+y+1;
    if (!Id[x][y]) Id[x][y] = ++n;
    return Id[x][y];
}

void init(){
    RD(nn, mm); REP(i, nn) RS(Map[i]); RST(Id, G); n = 0;

    REP_2(i, j, nn, mm) if (Map[i][j] != ’#’){
        REP(d, 20){
            int x = i + dx[d], y = j + dy[d];
            if (!inGrid(x, y) || Map[x][y] == ’#’) continue;
            int a = id(i, j), b = id(x, y);
            G[a][b] = 1;
        }
    }
}

bool ck(){
    int m1 = ebc(); REP_2(i, j, nn, mm) if (Map[i][j] == ’K’){
        int x = id(i, j); REP_1(y, n) G[x][y] = G[y][x] = 0;
    }
    int m2 = ebc();
    return m1 != m2;
}


int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
#endif

    Rush{
        init(); printf(”Case #%d: ”, ++Case);
        puts(ck() ? ”daizhenyang win” : ”daizhenyang lose”);
    }
}
```

Chapter 17

网络流 (Network FLow)

17.1 最大流/最小割

17.2 例题 (E.g.)

17.2.1 POJ Open 1036. Gugle Seating

简述 (Brief description)

...

分析 (Analysis)

S -> 电脑 -> 椅子 -> 电脑 -> T

```
const int N = int(5e5) + 9, M = 80*N;

int D[N], hd[N], suc[M], to[M], cap[M];
int n, m, s, t;

inline void add_edge(int x, int y, int c){
    suc[m] = hd[x], to[m] = y, cap[m] = c, hd[x] = m++;
    suc[m] = hd[y], to[m] = x, cap[m] = 0, hd[y] = m++;
}

inline void add_edgee(int x, int y, int c){
    suc[m] = hd[x], to[m] = y, cap[m] = c, hd[x] = m++;
    suc[m] = hd[y], to[m] = x, cap[m] = c, hd[y] = m++;
}

#define v to[i]
#define c cap[i]
#define f cap[i^1]

bool bfs(){
    static int Q[N]; int cz = 0, op = 1;
    fill(D, D+n, 0), D[Q[0] = s] = 1; while (cz < op){
        int u = Q[cz++]; REP_G(i, u) if (!D[v] && c){
            D[Q[op++] = v] = D[u] + 1;
            if (v == t) return 1;
        }
    }
    return 0;
}

LL Dinitz(){

    to[0] = s;
    LL max_flow = 0;

    while (bfs()){

        static int sta[N], cur[N]; int top = 0;
        sta[0] = 0, cur[s] = hd[s]; while (top != -1){

            int u = to[sta[top]], i; if (u == t){
                int d = INF; REP_1(ii, top) i = sta[ii], checkMin(d, c); max_flow += d;
                DWN_1(ii, top, 1){i = sta[ii], f += d, c -= d; if (!c) top = ii - 1;}
                u = to[sta[top]];
            }

            for (i=cur[u];i;i=suc[i])
                if (D[u] + 1 == D[v] && c) break;

            if (!i) D[u] = 0, --top;
            else {
                cur[u] = suc[i], cur[v] = hd[v];
                sta[++top] = i;
            }
        }
    }

    return max_flow;
}

const int NN = 509;
int A[NN][NN];
int nn, mm;

bool inGrid(int x, int y){
    return x >= 0 && y >= 0 && x < nn && y < mm;
}

int V(int lv, int x, int y){
    return lv*nn*mm + x*mm + y+1;
}

void Init(){
    REP_2(i, j, nn, mm) RD(A[i][j]);

    s = 0, t = 2*nn*mm+1, n = t+1, fill(hd, hd+n, 0), m = 2;
    REP_2(i, j, nn, mm) if (A[i][j]){
        if (A[i][j] == 2){
            if (j&1) add_edge(s,V(0,i,j),1);
            else add_edge(V(0,i,j),t,1);
        }
        else{
            add_edge(V(0,i,j),V(1,i,j),1); REP(d, 4){
                int x=i+dx[d],y=j+dy[d];
                if (!inGrid(x, y) || A[x][y] != 2) continue;
                if (y&1) add_edge(V(0,x,y),V(0,i,j),1);
                else add_edge(V(1,i,j),V(0,x,y),1);
            }
        }
    }
}

int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif
    while (~scanf(”%d%d”, &nn, &mm)){
        Init(); OT(Dinitz());
    }
}
```

17.2.2 上下界

17.2.3 无源汇

```

const int N = 509;
int C[N][N], cut[N], prd[N], suc[N], tmp[N];
int n, m, nn, s, t, tt;

inline void del(int x){prd[suc[prd[x]] = suc[x]] = prd[x];}
inline void rsm(int x){prd[suc[suc[prd[x]] = x]] = x;}

#define hd suc[0]

void Extract(int &s){
    s = hd; REP_L(i, suc[s], suc) if (cut[i] > cut[s]) s = i;
}

void Prim(){
    REP_L(i, hd, suc) cut[i] = 0; tt = 0; DO(nn-1){
        Extract(s); del(s); tmp[tt++] = s;
        REP_L(i, hd, suc) cut[i] += C[s][i];
    }
    Extract(t); DWN(i, tt, 0) rsm(tmp[i]); del(t);
}

int Stoer_Wagner(){

    REP_1(i, n) suc[i] = i+1, prd[i] = i-1;
    hd = 1, suc[n] = 0; nn = n;

    int res = INF; DO(n-2){
        Prim(), --nn, checkMin(res, cut[t]);
        REP_L(i, hd, suc) C[i][s] = C[s][i] += C[t][i];
    }

    Prim(), checkMin(res, cut[t]);
    return res;
}

int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    while (scanf(”%d%d”, &n, &m) != EOF){

        RST(C); DO(m){
            int x, y, w; RD(x, y, w); ++x, ++y;
            C[x][y] += w, C[y][x] += w;
        }

        OT(Stoer_Wagner());
    }
}
```

17.2.4 混合图欧拉回路

17.2.5 平面图最短路

17.3 最小费用最大流

17.4 例题 (E.g.)

17.4.1 SPOJ 371. Boxes

简述 (Brief description)

有 n 个盒子围成一个圆圈，我们把这些盒子顺时针从 1 到 n 编号。在这些盒子里面有一些球，球的总数不大于 n。我们现在要进行若干次操作，使得每个盒子里面最多存在一个球。对于一次操作，我们可以把一个球从它原先所在的盒子移到与它相邻的一个盒子里面。求最少需要多少次操作使得每个盒子中最多只有一个球。

```
const int N = 1099, M = int(1e6) + 9;

int D[N], hd[N], suc[M], to[M], cap[M], cst[M];
int n, m, s, t; LL flow, cost; int mask = 1023;

inline void add_edge(int x, int y, int c, int w){
    suc[m] = hd[x], to[m] = y, cap[m] = c, cst[m] = w, hd[x] = m++;
    suc[m] = hd[y], to[m] = x, cap[m] = 0, cst[m] = -w, hd[y] = m++;
}

inline void add_edgee(int x, int y, int c, int w){
    add_edge(x, y, c, w);
    add_edge(y, x, c, w);
}

int Q[N], F[N], pre[N], cz, op; bool inQ[N];

#define v to[i]
#define c cap[i]
#define f cap[i^1]
#define w cst[i]

bool spfa(){
    cz = 0, op = 1; fill(inQ, inQ+n, 0), fill(D, D+n, INF);
    D[Q[cz] = s] = 0, F[s] = INF; while (cz < op){
        int u = Q[cz++ & mask]; inQ[u] = 0;
        REP_G(i, u) if (c && D[v] > D[u] + w){
            D[v] = D[u] + w, F[v] = min(F[u], c); pre[v] = i;
            if (!inQ[v]) Q[op++ & mask] = v, inQ[v] = 1;
        }
    }
    return D[t] != INF;
}

#undef v

void add_path(){
    flow += F[t]; int u, v = t; do{
        int i = pre[v]; u = to[i^1], f += F[t], c -= F[t];
        cost += F[t]*w, v = u;
    } while (u != s);
}

pair<LL, LL> run(){
    to[m] = s, flow = 0, cost = 0; while (spfa()) add_path();
    return MP(cost, flow);
}

#undef c
#undef f
```

```
#undef w

int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    Rush{
        RD(n), m = 2, s = 0, t = n+1, fill(hd, hd+t+1, 0);

        REP_1(i, n){
            int Ai; if (!RD(Ai)) add_edge(i, t, 1, 0);
            else if (Ai-=1) add_edge(s, i, Ai, 0);
        }

        FOR_1(i, 2, n) add_edgee(i-1, i, INF, 1);
        add_edgee(1, n, INF, 1), n = t+1;

        OT(run().fi);
    }
}
```

17.5 网络单纯型

POJ 1273

```
// 最大流
// 考虑原图中有 n 个顶点，m 条边。。。
// 那么所对应的线性规划问题，有 m 个变量。
// m （边的容量约束） + （n-2）（中间结点的流量平衡约束） 个约束条件。。。

。。。。(注意考虑到我们是标准型。。流量平衡条件是等号。。所以要一分为 2.。。。)
(
注意下面的代码有问题。。。
理论上注释里的 3 种建模方式得到的结果都应该是一样的。。。。
。。。但是包括网上其他代码。。都只有最后一种可以 过。。。实在是意味不明。。。。


const int N = 609;

DB A[N][N]; int m, n;

void gao(){

    /*REP(i, m+1){
        REP(j, n+1){
            cout << A[i][j] << ” ”;
        }
        cout << endl;
    }*/

    for(;;){
        // Select ...
        int jj, ii; DB t = -OO; // jj 进基变量所在列，ii 出基变量所在行
        REP_1_N(jj, n) if (A[0][jj] > EPS) break; if (jj > n) break;
        REP_1(i, m) if (A[i][jj] < -EPS && A[i][0]/A[i][jj] > t) t = A[i][0]/A[i][jj], ii = i;
        t = -A[ii][jj], A[ii][jj] = -1; REP(j, n+1) A[ii][j] /= t;
        // Pivot ...
        REP(i, m+1) if (i != ii && A[i][jj] != 0){
            t = A[i][jj], A[i][jj] = 0;
            REP(j, n+1) A[i][j] += A[ii][j]*t;
        }
    }

    OT(A[0][0]); // 输出目标函数当前值
}

int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    int m, n; while(~scanf(”%d%d”, &m, &n)){

        RST(A); REP_1(i, m){
            int u, v; RD(u, v); A[i][0] = RF(); A[i][i] = -1;
            //if (u != 1) A[m+(u-1)*2-1][i] = 1, A[m+(u-1)*2][i] = -1; else A[0][i] = 1;
            //if (v != n) A[m+(v-1)*2-1][i] = -1, A[m+(v-1)*2][i] = 1;
            //if (u != 1) A[m+u-1][i] = 1, A[m+n+u-3][i] = -1; else A[0][i] = 1;
            //if (v != n) A[m+v-1][i] = -1, A[m+n+v-3][i] = 1;
            if (u != 1) A[m+u-1][i] = -1, A[m+n+u-3][i] = 1; else A[0][i] = 1;
            if (v != n) A[m+v-1][i] = 1, A[m+n+v-3][i] = -1;
        }

        ::m = m+2*(n-2), ::n = m, gao();
    }
}
```

17.6 例题

Part V

字符串 (Stringology)

Chapter 18

万金油 (Hash)

18.1 例题 (E.g.)

18.1.1 UVA 11996. Jewel Magic

伸展树维护 Hash。

```
const int N = int(4e5) + 9, C = 3;

uint P[N]; char str[N]; int n;

namespace Splay{

    int c[2][N], p[N], sz[N]; bool r0[N]; uint s[N], S[2][N];
    int rt, tot;

#define l c[0]
#define r c[1]
#define lx l[x]
#define rx r[x]
#define px p[x]
#define ly l[y]
#define ry r[y]
#define py p[y]

    void reset(int x, uint v){
        lx=rx=px=0;sz[x]=1;r0[x]=0;
        s[x]=v;
    }

    int new_node(uint v){
        ++tot; reset(tot, v);
        return tot;
    }

    void rev(int x){
        r0[x] ^= 1;
        swap(lx, rx), swap(S[0][x], S[1][x]);
    }

    void upd(int x){
        sz[x] = sz[lx] + 1 + sz[rx];
        S[0][x] = (S[0][lx]*C+s[x]) * P[sz[rx]] + S[0][rx];
        S[1][x] = (S[1][rx]*C+s[x]) * P[sz[lx]] + S[1][lx];
    }

    void rls(int x){
        if (r0[x]){
            rev(lx), rev(rx);
            r0[x] = 0;
        }
    }

    inline int sgn(int x){return r[px] == x;}
    inline void setc(int y, int d, int x){c[d][y] = x, px = y;}
    inline void setl(int y, int x){setc(y,0,x);}
    inline void setr(int y, int x){setc(y,1,x);}

    inline void rot(int x, int d){
        int y=px,z=py;setc(z,sgn(y),x);
        setc(y,d,c[!d][x]),setc(x,!d,y);upd(y);
    }

    inline void rot(int x){rot(x,sgn(x));}
    inline int splay(int x,int t=0){
        int a,b,y; while((y=px)!=t){
            if (py==t){rot(x);break;}
            else a=sgn(x),b=sgn(y),rot(a^b?x:y,a),rot(x,b);
        }
        upd(x);if(!t)rt=x;
        return x;
    }
    int Build(int a = 0, int b = n+1){
        if (a > b) return 0; int m = a + b >> 1, x = new_node(str[m]);
        setl(x, Build(a, m-1)), setr(x, Build(m+1, b)), upd(x);
        return x;
    }

    int Select(int k, int t=0){
        int x = rt; while (rls(x), sz[lx] != k){
            if (k<sz[lx]) x = lx;
            else k -= sz[lx]+1, x = rx;
        }
        return splay(x, t);
    }

    int Selectt(int a, int b){
        return r[Select(a-1, Select(b+1))];
    }

    int Lcp(int x, int y){
        int ll = -1, rr = n - max(x, y);
        while (ll < rr){
            int m = ll + rr + 1 >> 1;
            if (S[0][Selectt(x, x+m)] == S[0][Selectt(y, y+m)]) ll = m;
            else rr = m - 1;
        }
        return ll+1;
    }

    void Inorder(int x = rt){
        if (!x) return; rls(x);
        Inorder(lx);
        cout << char(s[x]+’0’-1);
        Inorder(rx);
    }

} using namespace Splay;

int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out2.txt”, ”w”, stdout);
#endif

    P[0] = 1; FOR(i, 1, N/2) P[i] = P[i-1] * C;

    int m; while (~scanf(”%d%d”, &n, &m)){

        n = strlen(RS(str+1)); REP_S(cur, str+1) *cur -= ’0’-1;

        tot = 0; rt = Build();

        int a, b, x, y, z; DO(m){
            switch(RD()){
                case 1: // Insert .. .
                    RD(a); y = Select(a, z = Select(a+1)); setr(y, x = new_node(RD()+1));
                    upd(x), upd(y), upd(z); ++n;
                    break;
                case 2: // Delete .. .
                    RD(a); y = Select(a-1, z = Select(a+1)); ry = 0;
                    upd(y), upd(z); --n;
                    break;
                case 3: // Reverse
                    RD(a, b); rev(Selectt(a, b));
                    break;
                default:
                    OT(Lcp(RD(), RD()));
            }

            //Inorder(); puts(””);
        }
    }
}
```

Chapter 19

在线算法 (Online)

19.1 KMP

1-offset

```
void get_pi(const char P[], int n, int pi[]){
    for (int i = 1, j = pi[1] = 0; i < n; pi[++i] = j){
        while (j && P[i] != P[j]) j = pi[j];
        if (P[i] == P[j]) ++j;
    }
}
```

0-offset

```
void get_pi(const char P[], int n, int pi[]){
    for (int i = 1, j = pi[1] = 0; i < n; pi[++i] = j){
        while (j && P[i] != P[j]) j = pi[j];
        if (P[i] == P[j]) ++j;
    }
}
```

19.2 Z

0-offset

```
void get_z(const char P[], int n, int z[]){

    int ex; z[0] = ex = n;

    for (int i = 1, l = 0, r = 0; i < n; z[i++] = ex){
        if (i > r){
            for (l = r = i; r < n && P[r] == P[r - l];) ++r;
            ex = r---l;
        }
        else {
            if (z[i - l] < r - i + 1) ex = z[i - l];
            else {
                for (l = i; r < n && P[r] == P[r - l];) ++r;
                ex = r---l;
            }
        }
    }
}
```

19.3 Manacher

```
// abab => $#a#b#c#d# .. .
// p[i]: 回文半径

int Manacher(char s[] = str+1){
    static char ss[2*N+2]; static int p[2*N+2];
    nn = 0; ss[nn++] = ’$’; REP(i, n) ss[nn++] =’#’, ss[nn++] = s[i]; ss[nn++] = ’#’; //ss[nn] = 0;

    int mx=0, id=0; FOR(i, 1, nn){
        p[i] = mx > i ? min(p[2*id-i], mx-i) : 1;
        while (ss[i+p[i]]==ss[i-p[i]]) ++p[i];
        if (i+p[i]>mx) mx=i+p[i], id=i;
    }

    int len = 0, pos; FOR(i, 1, nn) if (p[i] > len) len = p[i], pos = i;
    //FOR(i, 1, nn) cout << p[i] << ” ”; cout << endl;
    //for (int i = pos - --len + 1; i < pos + len; i += 2) putchar(ss[i]);
    //int st = pos/2 - (--len + !(pos&1))/2; FOR(i, st, st + len) putchar(s[i]); cout << endl;
    return --len;
}
```

19.4 最小表示

19.5 AC 自动机（Aho-Corasick Automaton））

19.6 例题 (E.g.)

19.6.1 HDU. 2222 Keywords Search

题目描述 (Brief description)

算法分析 (Algorithm Analysis)

外部链接 (External Link)

```
namespace ACM{
    const int N = int(5e5) + 9, Z = 26, L = int(1e6) + 9;
    int trans[N][Z], fail[N], cnt[N], Q[N], cz, op, tt;
    char str[L];

    int new_node(){
        RST(trans[tt]), fail[tt] = cnt[tt] = 0;
        return tt++;
    }

#define v trans[u][c]
#define f trans[fail[u]][c]

    void Build(){
        cz = op = 0; int u = 0; REP(c, Z) if (v) Q[op++] = v;
        while (cz < op){
            u = Q[cz++]; REP(c, Z){
                if (v) fail[Q[op++] = v] = f; // .. .
                else v = f;
            }
        }
    }

#define c (*cur - ’a’)

    void Insert(){
        RS(str); int u = 0; REP_S(cur, str){
            //if (cnt[v]) break; // .. .
            if (!v) v = new_node();
            u = v;
        }
        ++cnt[u];
    }

#define vis Q
    int Run(){
        RS(str); int res = 0; int t, u = 0; RST(vis);
        REP_S(cur, str){
            for (t=u=v;t&&!vis[t];t=fail[t]){
                res += cnt[t];
                vis[t] = 1;
            }
        }
        return res;
    }

    void Init(){
        tt = 0, new_node();
        Rush Insert(); Build();
    }

#undef vis
#undef c
#undef f
#undef v
} using namespace ACM;


int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    Rush{
        Init(); OT(Run());
    }
}
```

19.6.2 HDU. 3505 Writing Robot

题目描述 (Brief description)

算法分析 (Algorithm Analysis)

外部链接 (External Link)

```

const int PN = 159, PL = 59, TN = 159, TL = 1009;
char P[PN][PL], T[TL];
int li[PN], hi[PN], pn, tn;
int profit; VI match;

namespace ACM{
    const int N = PN*PL+9, Z = 26;
    int trans[N][Z], fail[N], cnt[N], id[N], Q[N], cz, op, tt;

    int new_node(){
        RST(trans[tt]), fail[tt] = cnt[tt] = id[tt] = 0;
        return tt++;
    }

#define v trans[u][c]
#define f trans[fail[u]][c]

    void Build(){
        cz = op = 0; int u = 0; REP(c, Z) if (v) Q[op++] = v;
        while (cz < op){
            u = Q[cz++]; REP(c, Z){
                if (v) fail[Q[op++] = v] = f, cnt[v] += cnt[f];
                else v = f;
            }
        }
    }

#define c (*cur - ’a’)

    void Insert(char str[], int idd){
        int u = 0; REP_S(cur, str){
            if (!v) v = new_node();
            u = v;
        }
        id[u] = idd, cnt[u] += li[idd];
    }

#define vis Q
    void Run(char str[]){
        profit = 0, CLR(match); int t, u = 0; RST(vis); REP_S(cur, str){
            for (profit += cnt[t=u=v];t&&!vis[t];t=fail[t]){
                if (id[t]) match.PB(id[t]);
                vis[t] = 1;
            }
        }
    }

    void Init(){
        tt = 0, new_node();
    }

#undef vis
#undef c
#undef f
#undef v
} //using namespace ACM;

namespace Network{

    const int N = PN + TN + 9, M = N*N+9;

    int D[N], hd[N], suc[M], to[M], cap[M];
    int n, m, s, t, total;

    inline void add_edge(int x, int y, int c){
        suc[m] = hd[x], to[m] = y, cap[m] = c, hd[x] = m++;
        suc[m] = hd[y], to[m] = x, cap[m] = 0, hd[y] = m++;
    }

#define v to[i]
#define c cap[i]
#define f cap[i^1]

    bool bfs(){
        static int Q[N]; int cz = 0, op = 1;
        fill(D, D+n, 0), D[Q[0] = s] = 1; while (cz < op){
            int u = Q[cz++]; REP_G(i, u) if (!D[v] && c){
                D[Q[op++] = v] = D[u] + 1;
                if (v == t) return 1;
            }
        }
        return 0;
    }

    LL Dinitz(){

        to[0] = s; LL max_flow = 0;

        while (bfs()){

            static int sta[N], cur[N]; int top = 0;
            sta[0] = 0, cur[s] = hd[s]; while (top != -1){

                int u = to[sta[top]], i; if (u == t){
                    int d = INF; REP_1(ii, top) i = sta[ii], checkMin(d, c); max_flow += d;
                    DWN_1(ii, top, 1){i = sta[ii], f += d, c -= d; if (!c) top = ii - 1;}
                    u = to[sta[top]];
                }

                for (i=cur[u];i;i=suc[i])
                    if (D[u] + 1 == D[v] && c) break;

                if (!i) D[u] = 0, --top;
                else {
                    cur[u] = suc[i], cur[v] = hd[v];
                    sta[++top] = i;
                }
            }
        }
        return max_flow;
    }

    int Solve(){

        RD(pn, tn), s = 0, t = pn + tn + 1, n = t + 1, m = 2, fill(hd, hd+n, 0);
        total = 0, ACM::Init();

        REP_1(i, pn){
            RD(li[i], hi[i]), ACM::Insert(RS(P[i]), i);
            add_edge(tn+i, t, hi[i]);
        }

        ACM::Build();

        REP_1(i, tn){
            RS(T); ACM::Run(T); total += profit, add_edge(s, i, profit);
            ECH(it, match) add_edge(i, tn+*it, INF);
            //cout << profit << ” ” << SZ(match) << endl;
        }

        return total - Dinitz();
    }

} //using namespace Max_Flow;


int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    Rush OT(Network::Solve());
}
```

Chapter 20

后缀三姐妹 (Indexed)

20.1 后缀数组 (Suffix Array)

20.1.1 子串计数

20.2 后缀自动机 (Suffix Automaton)

20.2.1 子串计数

20.2.2 出现次数向父亲传递

20.2.3 接收串数从孩子获取

```
int Q[N], C[N/2];
int adj[N][26];

void Init(){
    tot = 0, tail = new_node();
    RS(s); REP_S(c, s) Ext(*c - 'a');

    //REP(i, tot) C[i] = 0;
    REP(i, tot) ++C[len[i]];
    REP_1(i, len[tail]) C[i] += C[i-1];
    REP(i, tot) Q[--C[len[i]]] = i;

    DWN(i, tot, 0){
        int u = Q[i], t = 0; cnt[u] = 1;
        REP(c, Z) if (v){
            adj[u][t++] = c;
            cnt[u] += cnt[v];
        }
        //int u = Q[i]; cnt[p] += cnt[u];
    }
}
```

20.2.4 最小表示与循环同构

20.3 后缀树 (Suffix Tree)

20.3.1 子串计数

20.4 字典树 (On Trie)

20.5 动态 (On Dynamic)

20.5.1 push_back()

20.5.2 pop_back()

20.5.3 pop_front()

20.5.4 push_front()?

20.5.5 双向链表维护 SA

20.5.6 平衡树维护 SA

20.5.7 带删除标记的 SAM

20.5.8 LCT 维护 SAM

20.6 可持久化 (On Persistence)

20.6.1 LCT 维护 SAM

20.7 例题 (E.g.)

20.7.1 SPOJ SUBLEX. Lexicographical Substring Search

题目描述 (Brief description)

求字典序 k 大子串。

算法分析 (Algorithm analysis)

后缀自动机

```
        #define c adj[u][i]
void kth(int k){
    int u = 0; while (k){
        --k; REP(i, Z){
            if (k >= cnt[v]) k -= cnt[v];
            else{
                putchar(c+’a’);
                u = v; break;
            }
        }
    }
    puts(””);
}
```

后缀数组

```
int get_h(){
    ...
    a[0] = 0; REP_1(i, n) a[i] = a[i-1] + n-sa[i]-h[i];
}
```

20.7.2 BZOJ 3230. 相似子串

题目描述 (Brief description)

设两个字符串的最长公共前缀和后缀的长度分别为 a, b。则它们相似程度，定义为 $a^2 + b^2$。给定一个字符串，每次询问其字典序第 $k1 - th$ 大和第 $k2 - th$ 大的两个子串间的相似程度。

算法分析 (Algorithm analysis)

```
const int N = int(1e5)+9, M = 26+1, LV = 20;

LL a[N]; char s[N]; int n;
int C[N], key[N], t1[N], t2[N];

struct SA{

    int a[3*N], sa[3*N], rk[N], h[N];

    inline void rs(int*x,int*y,int*sa,int n,int m){
        REP(i, n)key[i]=i[y][x];
        memset(C, 0,sizeof(C[0])*m);
        REP(i, n) ++C[key[i]];
        FOR(i, 1, m) C[i] += C[i-1];
        DWN(i, n, 0) sa[--C[key[i]]] = y[i];
    }

    void da(int*a,int*sa,int n,int m){
        int *x = t1, *y = t2;
        memset(C,0,sizeof(C[0])*m);
        REP(i, n)++C[x[i]=a[i]];
        FOR(i, 1, m)C[i]+=C[i-1];
        DWN(i, n, 0)sa[--C[x[i]]]=i;
        for(int l=1,p=1;p<n;l<<=1,m=p){
            p=0; FOR(i, n-l, n) y[p++]=i;
            REP(i, n) if (sa[i]>=l) y[p++]=sa[i]-l;
            rs(x,y,sa,n,m),swap(x,y),x[sa[0]]=p=0;FOR(i, 1, n)
                x[sa[i]]=(y[sa[i]]==y[sa[i-1]]&&y[sa[i]+l]==y[sa[i-1]+l])?p:++p;
            ++p;
        }
    }

#define F(x) ((x)/3+((x)%3==1?0:tb))
#define G(x) ((x)<tb?(x)*3+1:((x)-tb)*3+2)
int c0(int*r,int a,int b)
{return r[a]==r[b]&&r[a+1]==r[b+1]&&r[a+2]==r[b+2];}
int c12(int k,int*r,int a,int b)
{if(k==2) return r[a]<r[b]||r[a]==r[b]&&c12(1,r,a+1,b+1);
 else return r[a]<r[b]||r[a]==r[b]&&key[a+1]<key[b+1];}

void dc3(int*a,int*sa,int n,int m){
    int i, j, *an=a+n, *san=sa+n, ta=0, tb=(n+1)/3, tbc=0, p;
    a[n] = a[n+1] = 0; REP(i, n) if (i%3) t1[tbc++]=i;

    rs(a+2,t1,t2,tbc,m),rs(a+1,t2,t1,tbc,m),rs(a,t1,t2,tbc,m);
    p=0,an[F(t2[0])]=0;FOR(i, 1, tbc)
        an[F(t2[i])]=c0(a,t2[i-1],t2[i])?p:++p;

    if (++p < tbc) dc3(an,san,tbc,p);
    else REP(i, tbc) san[an[i]] = i;

    REP(i, tbc) if(san[i] < tb) t2[ta++] = san[i] * 3;
    if (n%3==1) t2[ta++] = n-1; rs(a,t2,t1,ta,m);
    REP(i, tbc) key[t2[i]=G(san[i])] = i;

    for(i=0,j=0,p=0; i<ta && j<tbc; p++)
        sa[p]=c12(t2[j]%3,a,t1[i],t2[j]) ? t1[i++] : t2[j++];
    for(;i<ta;p++) sa[p]=t1[i++]; for(;j<tbc;p++) sa[p]=t2[j++];
}

void get_h(){
    REP_1(i, n) rk[sa[i]] = i;
    int k=0;for(int i=0;i<n;h[rk[i++]]=k){
        if (k)--k;for(int j=sa[rk[i]-1];a[i+k]==a[j+k];++k);
    }
}

int ST[LV][N];

#define cmp(a, b) (h[a]<h[b]?a:b)

inline int lcp(int l, int r){
    int lv = lg2(r - l); ++l, ++r;
    return min(h[ST[lv][l]], h[ST[lv][r-_1(lv)]]);
}

inline int lcpp(int l, int r){
    if (l == r) return n-l;
    l = rk[l], r = rk[r]; if (l > r) swap(l, r);
    return lcp(l, r);
}

void get_lcp(){
    REP_1(i, n) ST[0][i] = i;
    for (int lv = 1; _1(lv) <= n; ++lv){
        for (int i = 1; i + _1(lv) <= n + 1; ++i)
            ST[lv][i] = cmp(ST[lv-1][i], ST[lv-1][i+_1(lv-1)]);
    }
}

void bd(){
    dc3(a,sa,n+1,M),get_h(),get_lcp();
}
} A, B;

PII get(LL k){
    int r = lower_bound(a, a+n, k) - a; k -= a[r-1];
    return MP(A.sa[r]+1, A.h[r]+k);
}

LL f(LL x, LL y){
    if (x>a[n] || y>a[n]) return -1;
    PII a = get(x), b = get(y); int t = min(a.se, b.se);
    return sqr(LL(min(t,A.lcpp(a.fi-1, b.fi-1)))) + sqr(LL(min(t,B.lcpp(n-(a.fi+a.se-1), n-(b.fi+b.se-1)))));
}


int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    int m; RD(n, m); strlen(RS(s)); REP(i, n) B.a[n-i-1]=A.a[i]=s[i]-=’a’-1;
    A.bd(); B.bd(); REP_1(i, n) a[i]=a[i-1]+n-A.sa[i]-A.h[i];
    DO(m) OT(f(RD(), RD()));
}
```

20.7.3 CF 232D. Fence

题目描述 (Brief description)

...

算法分析 (Algorithm Analysis)

算法 1. 差分 + 后缀数组 + 二分 lcp 区间 + 可持久化线段树
http://hi.baidu.com/wyl8899/item/d1d5c406dc9e9611acdc7018

。。一道相当值得一做的后缀数组题。。题目没有上一题难。。但是比上一题细节更容易写错。。。

定义字符串 a 的子串 [l1, r1] 和其自身的一个子串 [l0, r0] 自匹配。。当且仅当。。
1. 长度相等 2. 没有自交
3. 对于所有的 i 。。a[l0 + i] + a[l1 + i] 均相等。。。

。。。显然第一步要差分。。（注意特判。。。
。。差分之后条件 3 就变成了相加 = 0。。再取反的话就可以归约到标准的自匹配问题了。。
。。。设 d[i] = a[i+1] - a[i] 。。且长度为 nn..
那么所要构建的就是 d[0], d[1], d[2] ... d[nn-1], OO, -d[0], -d[1], ... -d[nn-1] .。。这个 n = 2nn + 1 长的后缀数组了。。。

。。。后缀数组入手了之后。。对于每一个询问 x, y。。。（注意特判。。
。。设 len = y-x。。那么就是搞出最大的包含 x 的区间 [l, r]。。。
。。使得 lcp(l, r) >= len。。(。。和上一题里层的那个二分一模一样。。

。。。之后接一个可持久化线段树即可。。。

(。。具体说来就是一个二维平面上有一些点。。
(。。。询问 (l, r, a, b)。。表示询问横坐标在 [l, r] 之间。。纵坐标在 [a, b] 之间点的个数。。。
(。。。在这题中表现为。横坐标是在后缀数组中的位置。。纵坐标是在原数组中的位置。。。
(。。。然后这东西已然烂大街了。。参见。。。

以原数组为横轴。。

```
const int N = 200009, LN = 18;

int a[N], b[N], sa[N], sl[N], rankk[N], height[N], ST[LN][N];
int C[N], key[N], t1[N], t2[N]; int n, nn;

namespace Fotile_Tree{

    #define lx lc[x]
    #define rx rc[x]
    #define ly lc[y]
    #define ry rc[y]
    #define cx c[x]
    #define cy c[y]
    #define ml (l+r>>1)
    #define mr (ml+1)
    #define l0 1
    #define r0 nn
    #define lcc lx, l, ml
    #define rcc rx, mr, r

    const int NN = N*LN; //1000009

    int lc[NN], rc[NN], c[NN], tot, aa, bb;
    int T[N];

    inline int new_node(){
        return ++tot;
    }

    int Insert(int y, int p, int d){
        int x = new_node(), root = x; c[x] = c[y] + d;
        int l = l0, r = r0; while (l < r){
            if (p < mr){
                lx = new_node(), rx = ry;
                x = lx, y = ly, r = ml;
            }
            else {
                lx = ly, rx = new_node();
                x = rx, y = ry, l = mr;
            }
            c[x] = c[y] + d;
        }
        return root;
    }

    inline int sum(int x, int l = l0, int r = r0){
        if (r < aa || bb < l) return 0;
        if (aa <= l && r <= bb) return c[x];
        return sum(lcc) + sum(rcc);
    }

    inline int lsum(int x, int p){
        int res = 0, l = l0, r = r0; while (p != r){
            if (p < mr) x = lx, r = ml;
            else res += c[lx], x = rx, l = mr;
        }
        return res + cx;
    }

    #undef lx
    #undef rx
    #undef ly
    #undef ry
    #undef cx
    #undef cy
    #undef mid

} using namespace Fotile_Tree;


inline void rs(int *x, int *y, int *sa, int n, int m){
    REP(i, n) key[i] = i[y][x];
    memset(C, 0, sizeof(C[0]) * m);
    REP(i, n) ++C[key[i]];
    FOR(i, 1, m) C[i] += C[i-1];
    DWN(i, n, 0) sa[--C[key[i]]] = y[i];
}

void da(int a[], int sa[], int n, int m){
    int *x = t1, *y = t2;
    memset(C, 0, sizeof(C[0])*m);
    REP(i, n) ++C[x[i] = a[i]];
    FOR(i, 1, m) C[i] += C[i-1];
    DWN(i, n, 0) sa[--C[x[i]]] = i;
    for (int l = 1, p = 1; p < n; l <<= 1, m = p){
        p = 0; FOR(i, n-l, n) y[p++] = i;
        REP(i, n) if (sa[i] >= l) y[p++] = sa[i] - l;
        rs(x, y, sa, n, m), swap(x, y), x[sa[0]] = 0, p = 0; FOR(i, 1, n)
            x[sa[i]] = (y[sa[i]] == y[sa[i-1]] && y[sa[i]+l] == y[sa[i-1]+l]) ? p : ++p;
        ++p;
    }
}

void gh(int sa[], int rankk[], int height[], int n){
    REP_1(i, n) rankk[sa[i]] = i;
    int k = 0; for (int i = 0; i < n; height[rankk[i++]] = k){
        if (k) --k; for (int j = sa[rankk[i]-1]; a[i+k] == a[j+k]; ++k);
    }
}

inline bool shorter(int a, int b){
    return height[a] < height[b];
}

inline int lcp(int l, int r){
    if (l == r) return sl[sa[l]];
    int lv = lg2(r - l); ++l, ++r;
    return height[min(ST[lv][l], ST[lv][r-_1(lv)], shorter)];
}

inline int lcpp(int l, int r){
    l = rankk[l], r = rankk[r]; if (l > r) swap(l, r);
    return lcp(l, r);
}

void get_lcp(){
    REP_1(i, n) ST[0][i] = i;
    for (int lv = 1; _1(lv) <= n; ++lv){
        for (int i = 1; i + _1(lv) <= n + 1; ++i)
            ST[lv][i] = min(ST[lv-1][i], ST[lv-1][i+_1(lv-1)], shorter);
    }
}



int discretize(int a[], int n){
    int m = 1; VI A; REP(i, n) A.PB(a[i]); UNQ(A);
    REP(i, n) a[i] = lower_bound(ALL(A), a[i]) - A.begin() + 1; a[n] = 0;
    return SZ(A) + 1;
}


int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    REP_C(i, RD(n)) RD(a[i]);

    if (n == 1){
        Rush puts(”0”);
        return 0;
    }

    nn = n-1; REP(i, nn) a[i] = a[i+1] - a[i]; a[nn] = INF;
    REP(i, nn) a[i+n] = -a[i]; n=2*n-1;
    da(a, sa, n+1, discretize(a, n));
    gh(sa, rankk, height, n); get_lcp();

    T[0] = new_node(); REP_1(i, n){
        T[i] = sa[i] < nn ? T[i] = Insert(T[i-1], sa[i]+1, 1) : T[i-1];
    }

    Rush{
        int l, r; RD(l, r); if (l == r) OT(nn);
        else{
            int p = rankk[l+nn], len = r-l;
            aa = max(l0, l-len), bb = min(r0, l+len);

            l = 1, r = p; while (l < r){
                int m = l + r >> 1;
                if (lcp(m, p) >= len) r = m;
                else l = m + 1;
            }

            int ll = T[l-1];

            l = p, r = n; while (l < r){
                int m = l + r + 1 >> 1;
                if (lcp(p, m) >= len) l = m;
                else r = m - 1;
            }

            int rr = T[r];

            OT(c[rr]-c[ll]-(sum(rr)-sum(ll)));
        }
    }
}
```

20.7.4 CF 204E. Little Elephant and Strings

题目描述 (Brief description)

给定 n 个串。。问对于每一个字符串，有多少它的子串，可以至少匹配 k 个字符串。（包含自身）

算法分析 (Algorithm Analysis)

算法1. 二分 + 可持久化线段树

aaa$aba$aaa
首先当然还是要把所有串拼一起跑后缀数组。。。

考察第一个串。。记作 s0 = aaa 。。。我们枚举起始位 x。。
。。设 f(x, len)。。表示 。。。s0[x, x+len] 这个子串。。是否出现在了 k 个子串中。
。。。这个对 len 越长对满足这个性质越不利。。但是这里 len 值也就是对答案的贡献。。显然我们希望尽可能往→推。。。
。。于是这里可以二分 len。。。（外层的二分。。

$aaa
$aba$aaa
a
a$aaa
a$aba$aaa
aa
aa$aba$aaa
aaa
aaa$aba$aaa
aba$aaa
ba$aaa

。。由于后缀数组已经把我们要的东西全放在一起了。。对于任意一个 x 和 len。。。
。。我们所要做的就是找到一个最大的包含 x 的 [l, r] 区间。。使得该区间的 lcp(l, r) >= len。

（这里既是 SA 的一个性质。。然后对于任一个串 P，LCP(P, SA[i]) 是单峰的。

。。这里又是一个二分过程。。。
。。找到 [l, r] 区间后。。。就是 判断这个区间的后缀出现在了几个不同的字符串中。。。
。。。而这个是可持久化线段树的入门题了。。（参见 SPOJ Dquery 。。

。。这样这个题就得到了最傻逼的做法。。。复杂度 O(nlog^2n)
http://codeforces.com/contest/204/submission/4545205

算法 2. two-point
http://www.cppblog.com/hanfei19910905/archive/2012/07/26/185139.html

。。。上面的做法中。可持久化线段树。未免有点 overkill。。注意毕竟只是询问是否 >= k。。。
。而这个可以通过 two-point 离线搞出对于每个左端点。最早的合法位置。。。
具体做法是 prd[], suc[] 分别表示左右第一个与 b[i] 相同的位置。。

```
[cpp]
int last[N], prd[N], suc[N];
[/cpp]
```

两遍循环。。

```
[cpp]
    FOR_1(i, nn, n) prd[i] = last[bb(i)], last[bb(i)] = i;
    REP_1(i, nn) last[i] = n + 1;
    DWN_1(i, n, nn) suc[i] = last[bb(i)], last[bb(i)] = i;
[/cpp]
```

之后 two-point。。
（l 增大的时候。。如果 suc[l] > r 。。c -= 1
（r 增大的时候。。如果 prd[++r] < l。。则 c += 1

```
[cpp]
    for(int l=nn,r=nn-1,c;l<=n;c-=(suc[l++]>r)){
        if (r<l) r=l,c=1; while(c<k&&r<=n) if(prd[++r]<l)++c; if (c<k){n=l-1;break;}
        last[l] = r;
    }
[/cpp]
```

其他地方不变。。。。复杂度依旧是 O(nlog^2n)
http://codeforces.com/contest/204/submission/4544775

算法 3. 标记
。。为了杜绝掉二分的过程。。。我们注意到上面 two-point 得到一组最小的合法 (l, r, lcp) 的时候。。。
。。可以沿途打上事件标记。。。用扫描线的方法。弄一个平衡树。。每次取出最大的 lcp 好像就行了。。

http://hi.baidu.com/wyl8899/item/04772d462eeb6797823ae16d
..似乎已经做完了么...其实被坑了，样例2就可以把这个做法撸死。
究其原因，是存在某个k，他的真正可用的最大值并不能被上面所述的方法更新到。
这就坑爹了..因为能更新到k的最大值的那个区间，假设是[x,y']，会出现y'>y(使得rank为x..y的后缀分属K个串的最小y值)的情形。

。。然而这点也正是这题最精彩的地方。。。因为对于一组标记 (l, r, lcp) 来说。。
。。。r 之后并不代表这个标记就完全失效了。。而是以这个时刻开始。。。
。。随着时间的流逝。。产生衰减。。（说的神乎其神的。。具体来说就是每次 checkMin(delay, height[i])。。

因此。。我们每次除了要取出平衡树中的最大值以外。。还需要那些过了 "保质期" 的标记中的最大值。。
。。。然后从这两者之间。取最大值。。。。

。。 平衡树内的标记。。涉及增删操作。。最好的方法使用 multiset<int> 实现。。。
。。 平衡树以外的标记。。只需保留一个最大值即可（记作 delay）。。然后随着时间的推移。每次 checkMin(delay, height[i])。。
http://codeforces.com/contest/204/submission/4544928

。。除去不多的几次平衡树操作。。这个算法的复杂度已经很接近 O(n) 了。。而且非常好写。。
。。。。是一个优秀的算法。

算法 3

```
const int N = 200009, LN = 24;

int a[3*N], sa[3*N], rankk[N], height[N], ST[LN][N], b[N], sl[N];
int C[N], key[N], t1[N], t2[N]; char buf[N];
int nn, n, k;


inline void rs(int *x, int *y, int *sa, int n, int m){
    REP(i, n) key[i] = i[y][x];
    memset(C, 0, sizeof(C[0]) * m);
    REP(i, n) ++C[key[i]];
    FOR(i, 1, m) C[i] += C[i-1];
    DWN(i, n, 0) sa[--C[key[i]]] = y[i];
}

void da(int a[], int sa[], int n, int m){
    int *x = t1, *y = t2;
    memset(C, 0, sizeof(C[0])*m);
    REP(i, n) ++C[x[i] = a[i]];
    FOR(i, 1, m) C[i] += C[i-1];
    DWN(i, n, 0) sa[--C[x[i]]] = i;
    for (int l = 1, p = 1; p < n; l <<= 1, m = p){
        p = 0; FOR(i, n-l, n) y[p++] = i;
        REP(i, n) if (sa[i] >= l) y[p++] = sa[i] - l;
        rs(x, y, sa, n, m), swap(x, y), x[sa[0]] = 0, p = 0; FOR(i, 1, n)
            x[sa[i]] = (y[sa[i]] == y[sa[i-1]] && y[sa[i]+l] == y[sa[i-1]+l]) ? p : ++p;
        ++p;
    }
}

void gh(int sa[], int rankk[], int height[], int n){
    REP_1(i, n) rankk[sa[i]] = i;
    int k = 0; for (int i = 0; i < n; height[rankk[i++]] = k){
        if (k) --k; for (int j = sa[rankk[i]-1]; a[i+k] == a[j+k]; ++k);
    }
}


#define F(x) ((x)/3+((x)%3==1?0:tb))
#define G(x) ((x)<tb?(x)*3+1:((x)-tb)*3+2)
int c0(int *r,int a,int b)
{return r[a]==r[b]&&r[a+1]==r[b+1]&&r[a+2]==r[b+2];}
int c12(int k,int *r,int a,int b)
{if(k==2) return r[a]<r[b]||r[a]==r[b]&&c12(1,r,a+1,b+1);
 else return r[a]<r[b]||r[a]==r[b]&&key[a+1]<key[b+1];}

void dc(int a[], int *sa, int n, int m){
    int i, j, *an=a+n, *san=sa+n, ta=0, tb=(n+1)/3, tbc=0, p;
    a[n] = a[n+1] = 0; REP(i, n) if (i%3) t1[tbc++]=i;

    rs(a+2,t1,t2,tbc,m), rs(a+1,t2,t1,tbc,m), rs(a,t1,t2,tbc,m);
    p = 0, an[F(t2[0])] = 0; FOR(i, 1, tbc)
        an[F(t2[i])] = c0(a,t2[i-1],t2[i]) ? p : ++p;

    if (++p < tbc) dc(an,san,tbc,p);
    else REP(i, tbc) san[an[i]] = i;

    REP(i, tbc) if(san[i] < tb) t2[ta++] = san[i] * 3;
    if (n%3==1) t2[ta++] = n-1; rs(a,t2,t1,ta,m);
    REP(i, tbc) key[t2[i]=G(san[i])] = i;

    for(i=0,j=0,p=0; i<ta && j<tbc; p++)
        sa[p]=c12(t2[j]%3,a,t1[i],t2[j]) ? t1[i++] : t2[j++];
    for(;i<ta;p++) sa[p]=t1[i++]; for(;j<tbc;p++) sa[p]=t2[j++];
}


inline bool shorter(int a, int b){
    return height[a] < height[b];
}

inline int lcp(int l, int r){
    if (l == r) return sl[sa[l]];
    int lv = lg2(r - l); ++l, ++r;
    return height[min(ST[lv][l], ST[lv][r-_1(lv)], shorter)];
}

inline int lcpp(int l, int r){
    l = rankk[l], r = rankk[r]; if (l > r) swap(l, r);
    return lcp(l, r);
}

void get_lcp(){
    REP_1(i, n) ST[0][i] = i;
    for (int lv = 1; _1(lv) <= n; ++lv){
        for (int i = 1; i + _1(lv) <= n + 1; ++i)
            ST[lv][i] = min(ST[lv-1][i], ST[lv-1][i+_1(lv-1)], shorter);
    }
}

multiset<int> Q; int last[N], prd[N], suc[N];
int add[N]; VI sub[N]; LL ans[N];

int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    RD(nn, k); REP_1(ii, nn){
        int len = strlen(RS(buf)); REP(i, len) a[n] = a[n] = buf[i] - ’a’ + 2, sl[n] = len-i, b[n++] = ii;
        a[n++] = 1;
    }
    a[--n] = 0;

    dc(a, sa, n+1, 27+nn); gh(sa, rankk, height, n); get_lcp();

#define bb(i) b[sa[i]]

    FOR_1(i, nn, n) prd[i] = last[bb(i)], last[bb(i)] = i;
    REP_1(i, nn) last[i] = n + 1;
    DWN_1(i, n, nn) suc[i] = last[bb(i)], last[bb(i)] = i;

    for(int l=nn,r=nn-1,c;l<=n;c-=(suc[l++]>r)){
        if (r<l) r=l,c=1; while(c<k&&r<=n) if(prd[++r]<l)++c; if (c<k){n=l-1;break;}
        int w = lcp(l, r); add[l] = w; sub[r+1].PB(w);
    }

    int delay = 0; FOR_1(i, nn, n){
        ECH(it, sub[i]) Q.erase(*it), checkMax(delay, *it); Q.insert(add[i]); checkMin(delay, height[i]);
        ans[bb(i)] += min(max(*Q.rbegin(), delay), sl[sa[i]]);
    }

    REP_1(i, nn) OT(ans[i]);
}
```

```
const int N = 200009, LN = 24, Z = 26;

char buf[N/2]; LL ans[N/2];
int TtoM[N/2], MtoT[N], nn, K;

namespace Trie{
    int trans[N/2][Z]; VI b[N/2];
    int tot;

#define v trans[u][c]

    int new_node(){
        return tot++;
    }

    void Insert(int id){
        RS(buf); int u = 0; REP_S(cur, buf){
            int c = *cur - ’a’;
            if (!v) v = new_node();
            b[u = v].PB(id);
        }
    }

    void Init(){
        tot = 0, new_node();
        REP(i, nn) Insert(i);
    }
}

namespace SAM{

    int trans[N][Z], par[N], len[N], cnt[N], tot;
    VI adj[N]; int P[N], L[N/2];

#define p par[u]
#define pp par[uu]

    void Make(int x){P[x] = x;}
    int Find(int x){return x == P[x] ? x : P[x] = Find(P[x]);}
    void Union(int x, int y){P[y] = x;}

    inline int new_node(){
        RST(trans[tot]); //cnt[tot] = 0;
        return tot++;
    }

    inline int new_node(int u){
        CPY(trans[tot], trans[u]); par[tot] = par[u]; //cnt[tot] = cnt[u];
        return tot++;
    }

    inline int h(int u){
        return len[u] - len[p];
    }

    int Ext(int c, int tail){
        int u = tail, uu = new_node(); len[uu] = len[u] + 1;
        while (u && !v) v = uu, u = p; // 向上遍历没有 c-转移 的祖先 ..
        if (!u && !v) v = uu, pp = 0;
        else{
            if (len[v] == len[u] + 1) pp = v;
            else{
                int _v = v, vv = new_node(_v); len[vv] = len[u] + 1; par[_v] = pp = vv;
                while (u && v == _v) v = vv, u = p;
                if (!u && v == _v) v = vv;
            }
        }
        return uu;
    }

#undef v
#define v (*it)

    void tarjan(int u = 0){

        Make(u); ECH(it, adj[u]) tarjan(v), Union(u, v);

        if(MtoT[u]) ECH(it, Trie::b[MtoT[u]]){
            --cnt[Find(L[v])]; ++cnt[u], L[v] = u;
        }
    }

    void dfs1(int u = 0){
        ECH(it, adj[u]) dfs1(v), cnt[u] += cnt[v];
    }

    void dfs2(int u = 0){
        ECH(it, adj[u]) cnt[v] += cnt[u], dfs2(v);
        ECH(it, Trie::b[MtoT[u]]) ans[v] += cnt[u];
    }

#undef v
#define v Trie::trans[u][c]

    void Init(){
        tot = 0; queue<int> Q; Q.push(0); TtoM[0] = new_node();

        while(SZ(Q)){
            int u = Q.front(); Q.pop();
            REP(c, 26) if (v) Q.push(MtoT[TtoM[v] = Ext(c, TtoM[u])] = v);
        }

        FOR(u, 1, tot) adj[p].PB(u); fill(L, L+nn, 0); tarjan();
        dfs1(); FOR(u, 1, tot) cnt[u] = cnt[u] >= K ? h(u) : 0; dfs2();
    }
}


int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    RD(nn, K); Trie::Init(); SAM::Init();
    REP(i, nn) OT(ans[i]);
}
```

20.7.5 CF 316G3. Good Substrings string suffix structures

题目描述 (Brief description)

算法分析 (Algorithm Analysis)

20.7.6 CF 235C. Cyclical Quest

```

const int N = int(2e6) + 9, Z = 26;

```

```
namespace SAM{

    int trans[N][Z], par[N], len[N], cnt[N], tail, tot; char str[N/2];

    inline int new_node(){
        //RST(trans[tot]),
        tail = tot; cnt[tot] = 1;
        return tot++;
    }

    inline int new_node(int u){
        CPY(trans[tot], trans[u]), par[tot] = par[u]; //cnt[tot] = 0
        return tot++;
    }

#define v trans[u][c]
#define p par[u]
#define pp par[uu]

    void Ext(int c){
        int u = tail, uu = new_node(); len[uu] = len[u] + 1;
        while (u && !v) v = uu, u = p;
        if (!u && !v) v = uu, pp = 0;
        else{
            if (len[v] == len[u] + 1) pp = v;
            else{
                int _v = v, vv = new_node(_v); len[vv] = len[u] + 1;
                par[_v] = pp = vv;
                while (v == _v) v = vv, u = p;
            }
        }
    }

int Q[N], C[N/2], tt;
#define vis Q
#define c (*cur - ’a’)
    int Spell(){
        int ll = strlen(RS(str)); int res = 0, l = 0, u = 0; REP_S(cur, str){
            while (u && !v) l = len[u = p];
            if (u = v) ++l;
        }

        --tt; REP_S(cur, str){
            while (u && !v) l = len[u = p]; if (u = v) ++l;
            while (len[p] >= ll) l = len[u = p];

            if(l >= ll && vis[u] != tt){
                vis[u] = tt;
                res += cnt[u];
            }
        }
        return res;
    }

    void Init(){
        tot = 0, new_node(); RS(str); REP_S(cur, str) Ext(c);

        //REP(i, tot) C[i] = 0;
        REP(i, tot) ++C[len[i]];
        REP_1(i, len[tail]) C[i] += C[i-1];
        REP(i, tot) Q[--C[len[i]]] = i;

#undef c

        DWN(i, tot, 1){
            int u = Q[i];
            cnt[p] += cnt[u];
            //cout << u << ” ” << cnt[u] << endl;
        }
        tt = INF;
    }

} using namespace SAM;

int main(){
#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    Init(); Rush OT(Spell());
}
```

20.7.7 SPOJ NSUBSTR2. Substrings II

```
const int N = int(16e4) + 9, Z = 26;

int trans[N][Z], len[N], par[N], tail, tot; char s[N/2];

#define l c[0]
#define r c[1]
#define lx x->l
#define rx x->r
#define px x->p
#define ly y->l
#define ry y->r
#define py y->p

struct node{

    static node* NIL;
#define NIL node::NIL

    node *c[2], *p; //node* d[]

    int w0, delay; bool rev;

    inline void reset(int __w){
        p = c[0] = c[1] = NIL;
        w0 = __w, delay = rev = 0;
    }

    inline node(){
        //reset();
    }

    inline int sgn(){return p->l == this ? 0 : p->r == this ? 1 : -1;}
    inline void link(int d, node* x){c[d] = x; px = this;}

    inline void update(){
        //w1 = max(l->w1, w0, r->w1);
    }

    inline void inc(int d){
        if (this == NIL) return;
        w0 += d, delay += d;
    }

    inline void release(){
        //if (this == NIL) return;
        if (rev){
            swap(l, r);
            l->rev ^= 1, r->rev ^= 1;
            rev = 0;
        }
        if (delay){
            l->inc(delay), r->inc(delay);
            delay = 0;
        }
    }

    inline void _rot(int d){
        node *y = p, *z = py;
        if (y->sgn() != -1) z->link(y->sgn(), this); else p = z;
        y->link(d, c[d^1]), link(d^1, y);
        y->update();
    }

    inline void rot(){_rot(sgn());}
    inline void zig(){_rot(0);}
    inline void zag(){_rot(1);}

    inline void fix(){
        if(~sgn()) p->fix();
        release();
    }


/*
    inline node* splay(){
        fix(); while (sgn() != -1) rot();
        update();
        return this;
    }
*/
    inline node* splay(){
        fix(); while (sgn() != -1){
            node *y = p, *z = y->p; if (y->sgn() == -1){ rot(); break;}
            if (z->l == y){
                if (y->l == this) y->zig(), zig();
                else zag(), zig();
            }else{
                if (y->r == this) y->zag(), zag();
                else zig(), zag();
            }
        }
        update();
        return this;
    }

    inline node* access(){
        node *x = this, *y = NIL; do{
            x->splay();
            rx = y, x->update();
            y = x, x = px;
        } while (x != NIL);
        return y;
    }

    inline node* accesss(){
        access();
        return splay();
    }

    node* rt(){
        node* x; for (x = access(); x->release(), lx != NIL; x = lx);
        return x;
    }

    node* evert(){
        access()->rev ^=1;
        return this;
    }

#define evertt evert()->splay
    // Public ...

    void Link(node* x){
        //if (x == NIL) return;
        access(); p = x;
        p->access()->inc(w0);
    }

    void Cut(){
        accesss(); //if (l == NIL) return;
        l->inc(-w0), l->p = NIL, l = NIL;
    }

    int Query(){
        return accesss()->w0;
    }

} TPool[N], *T[N], *NIL;

#define v trans[u][c]
#define p par[u]
#define pp par[uu]

inline int new_node(){
    //RST(trans[tot);
    T[tot]->reset(1);
    return tot++;
}

inline int new_node(int u){
    CPY(trans[tot], trans[u]); par[tot] = par[u];
    T[tot]->reset(0); T[tot]->Link(T[par[u]]);
    return tot++;
}

void Ext(int c){
    int u = tail, uu = new_node();len[uu] = len[u] + 1;
    while (u && !v) v = uu, u = p; // 向上遍历没有 c-转移 的祖先 ..
    if (!u && !v) v = uu, pp = 0;
    else{
        if (len[v] == len[u] + 1) pp = v, T[uu]->Link(T[v]);
        else{
            int _v = v, vv = new_node(_v); len[vv] = len[u] + 1;
            T[_v]->Cut(), T[_v]->Link(T[vv]), T[uu]->Link(T[vv]);
            par[_v] = pp = vv;
            while (u && v == _v) v = vv, u = p;
            if (!u && v == _v) v = vv;
        }
    }
    // ...
    tail = uu;
}

#define c (*cc - 'a')
void Init(){
    tot = 0, tail = new_node();
    RS(s); REP_S(cc, s) Ext(c);
}

void Run(){
    int ans = 0, q, a, b; RD(q, a, b); DO(q){
        int u = 0; RS(s); REP_S(cc, s) if (!(u = v)) break;
        int ans = u ? T[u]->Query() : 0; OT(ans);
        Ext((a*ans+b)%26);
    }
}

int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif
    NIL = new node(); NIL->reset(0); REP(i, N) T[i] = &TPool[i];
    Init(), Run();
}
```

20.7.8 HDU 4641. K-string

```
const int N = int(5e5) + 9, Z = 26;

int trans[N][Z], len[N], par[N], tail, tot; char s[N/2];
LL ans; int n, m, K;

#define l c[0]
#define r c[1]
#define lx x->l
#define rx x->r
#define px x->p
#define ly y->l
#define ry y->r
#define py y->p

struct node{

    static node* NIL, *Deepest;
#define NIL node::NIL

    node *c[2], *p;
    int w0, w1, delay; bool rev;

    inline void reset(int _w){
        p = c[0] = c[1] = NIL;
        w0 = w1 = _w, delay = rev = 0;
    }

    inline node(){
        //reset();
    }

    inline int sgn(){return p->l == this ? 0 : p->r == this ? 1 : -1;}
    inline void link(int d, node* x){c[d] = x; px = this;}

    inline void update(){
        w1 = max(l->w1, w0, r->w1);
    }

    inline void inc(int d){
        if (this == NIL) return;
        w0 += d, w1 += d, delay += d;
    }

    inline void release(){
        //if (this == NIL) return;
        if (rev){
            swap(l, r);
            l->rev ^= 1, r->rev ^= 1;
            rev = 0;
        }
        if (delay){
            l->inc(delay), r->inc(delay);
            delay = 0;
        }
    }

    inline void _rot(int d){
        node *y = p, *z = py;
        if (y->sgn() != -1) z->link(y->sgn(), this); else p = z;
        y->link(d, c[d^1]), link(d^1, y);
        y->update();
    }

    inline void rot(){_rot(sgn());}
    inline void zig(){_rot(0);}
    inline void zag(){_rot(1);}

    inline void fix(){
        if(~sgn()) p->fix();
        release();
    }

/*
    inline node* splay(){
        fix(); while (sgn() != -1) rot();
        update();
        return this;
    }
*/
    inline node* splay(){
        fix(); while (sgn() != -1){
            node *y = p, *z = y->p; if (y->sgn() == -1){ rot(); break;}
            if (z->l == y){
                if (y->l == this) y->zig(), zig();
                else zag(), zig();
            }else{
                if (y->r == this) y->zag(), zag();
                else zig(), zag();
            }
        }
        update();
        return this;
    }

    inline node* access(){
        node *x = this, *y = NIL; do{
            x->splay();
            rx = y, x->update();
            y = x, x = px;
        } while (x != NIL);
        return y;
    }

    inline node* accesss(){
        access();
        return splay();
    }

    node* rt(){
        node* x; for (x = access(); x->release(), lx != NIL; x = lx);
        return x;
    }

    node* evert(){
        access()->rev ^=1;
        return this;
    }

#define evertt evert()->splay
    // Public ...

    void Link(node* x){
        //if (x == NIL) return;
        access(); p = x;
        p->access()->inc(w0);
    }

    void Cut(){
        accesss(); //if (l == NIL) return;
        l->inc(-w0), l->p = NIL, l = NIL;
    }

    int Query(){
        return accesss()->w0;
    }

    int h();

    void dfs(){
        if (this == NIL || w1 < K) return;
        Deepest = this; release(); if (w0 >= K) ans += h(), w0 -= INF;
        l->dfs(), r->dfs(), update();
    }

    void Stat(){
        Deepest = this, accesss()->dfs();
        Deepest->splay();
    }
} TPool[N], *T[N], *NIL, *node::Deepest;

#define v trans[u][c]
#define p par[u]
#define pp par[uu]

inline int new_node(){
    RST(trans[tot]);
    T[tot]->reset(1);
    return tot++;
}

inline int new_node(int u){
    CPY(trans[tot], trans[u]); par[tot] = par[u];
    T[tot]->reset(0),
    T[tot]->w1 = T[u]->w1;
    T[tot]->Link(T[par[u]]);
    return tot++;
}

inline int h(int u){
    return len[u] - len[p];
}

inline int node::h(){
    return ::h(this - TPool);
}

void Ext(int c){
    int u = tail, uu = new_node(); len[uu] = len[u] + 1;
    while (u && !v) v = uu, u = p; // 向上遍历没有 c-转移 的祖先 ..
    if (!u && !v) v = uu, pp = 0;
    else{
        if (len[v] == len[u] + 1) pp = v, T[uu]->Link(T[v]);
        else{
            int _v = v, vv = new_node(_v); len[vv] = len[u] + 1;
            T[_v]->Cut(), T[_v]->Link(T[vv]), T[uu]->Link(T[vv]);
            par[_v] = pp = vv;
            while (u && v == _v) v = vv, u = p;
            if (!u && v == _v) v = vv;
        }
    }
    T[uu]->Stat(); // .. .
    tail = uu;
}

void Init(){
    ans = tot = 0; tail = new_node();
    RS(s); REP_S(c, s) Ext(*c - ’a’);
}

int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    NIL = new node(); NIL->reset(-INF);
    REP(i, N) T[i] = &TPool[i];

    while (~scanf(”%*d%d%d”, &m, &K)){
        Init(); DO(m) if (RD() == 1) Ext(RC() - ’a’);
        else OT(ans);
    }
}
```

20.7.9 SPOJ AE5A2. quasi-template

```

const int N = int(4e5) + 9, C = 26;


namespace Splay{

struct node{

    static node*NIL;node*c[2],*p;
    int ll,ky,rr,dd;

#define NIL node::NIL
#define l c[0]
#define r c[1]
#define lx x->l
#define rx x->r

    void reset(int v = 0){
        l = r = p = NIL;
        dd = 0, ll = rr = ky = v;
    }

    node(int v = 0){
        reset(v);
    }

    void upd(){
        dd = 0;
        if (l == NIL) ll = ky; else ll = l->ll, checkMax(dd, max(l->dd, ky-l->rr));
        if (r == NIL) rr = ky; else rr = r->rr, checkMax(dd, max(r->dd, r->ll-ky));;
    }

    void setc(int d, node *x){c[d]=x,x->p=this;}
    int sgn(){return p->r==this;}

    void rot(int d){
        node*y=p,*z=y->p; z->setc(y->sgn(), this);
        y->setc(d, c[!d]), setc(!d, y), y->upd();
    }
    void rot(){rot(sgn());}

    node *splay(){
        int a, b; while(p!=NIL){
            //cout << ”!” << endl;
            if (p->p==NIL){rot();break;}
            else a=sgn(),b=p->sgn(),(a^b?this:p)->rot(a),rot(b);
        }
        upd();
        return this;
    }

    void insert(node *z){
        node *x=this,*y; while (x != NIL){
            y = x, x = x->c[z->ky>x->ky];
        }
        y->setc(z->ky>y->ky, z);
        z->splay();
    }

} *NIL, *T[N];

node*merge(node *y, node *x){
    if(x==NIL)return y;
    y = merge(y, lx), y = merge(y, rx);
    lx = rx = NIL, y->insert(x);
    return x;
}

#undef l
#undef r
#undef lx
#undef rx

} using namespace Splay;


namespace KMP{
    void get_pi(const char P[], int n, int pi[]){
        for (int i = 2, j = pi[1] = 0; i <= n; ++i){
            while (j && P[i] != P[j+1]) j = pi[j];
            if (P[i] == P[j+1]) ++j;
            pi[i] = j;
        }
    }
} using namespace KMP;


namespace SAM{

    int trans[N][C], fail[N], len[N], cnt[N], tail, tot;
    char str[N/2]; int n, pi[N], ll[N], rr[N], dd[N], ml[N];

    inline int new_node(){
        RST(trans[tot]); cnt[tot] = 1; tail = tot;
        return tot++;
    }

    inline int new_node(int u){
        CPY(trans[tot], trans[u]); fail[tot] = fail[u], cnt[tot] = 0;
        return tot++;
    }

#define v trans[u][c]
#define f fail[u]
#define ff fail[uu]

    void Ext(int c){
        int u = tail, uu = new_node(); len[uu] = len[u] + 1;
        while (u && !v) v = uu, u = f;
        if (!u && !v) v = uu, ff = 0;
        else{
            if (len[v] == len[u] + 1) ff = v;
            else{
                int _v = v, vv = new_node(_v); len[vv] = len[u] + 1;
                fail[_v] = ff = vv;
                while (v == _v) v = vv, u = f;
            }
        }
    }

    void Init(){
        tot = 0, tail = new_node();
    }

    int Q[N], CC[N/2];

    void Topo(int*key){
        memset(CC, 0, sizeof(int)*(len[tail]+1));
        REP(i, tot) ++CC[key[i]];
        REP_1(i, len[tail]) CC[i] += CC[i-1];
        REP(i, tot) Q[--CC[key[i]]] = i;
    }

    void Run(){

        REP(u, tot) T[u] = cnt[u] ? new node(len[u]) : NIL;

        Topo(len);

        FOR(i, 1, tot){
            int u = Q[i];
            pi[u] = cnt[fail[u]] ? len[fail[u]] : pi[fail[u]];
        }

        DWN(i, tot, 1){
            int u = Q[i]; if (!cnt[u]) continue;
            ll[u] = T[u]->ll; rr[u] = T[u]->rr; dd[u] = T[u]->dd;
            T[f] = cnt[f] > cnt[u] ? merge(T[f], T[u]) : merge(T[u], T[f]);
            cnt[f] += cnt[u];
        }

        Topo(rr);
    }

//#undef v
//#undef f
//#undef ff

} using namespace SAM;

```

```

namespace Segment_Tree{

    const int NN = 4 * N;

#define lx (x<<1)
#define rx (lx|1)
#define ml (l + r >> 1)
#define mr (ml + 1)
#define lc lx, l, ml
#define rc rx, mr, r
#define root 1, 0, n-1

    int T[NN], M[NN], a, b, cur, ss, mm; VI adj[N/2];

    inline void Build(int x, int l, int r){
        T[x] = M[x] = 0; if (l < r) Build(lc), Build(rc);
    }

    inline void Insert(int x, int l, int r){
        ++T[x], checkMax(M[x], a); if (l == r) return;
        if (a < mr) Insert(lc); else Insert(rc);
    }

    inline void Query(int x, int l, int r){
        if (b < l || r < a) return;
        if (a <= l && r <= b) ss += T[x], checkMax(mm, M[x]);
        else Query(lc), Query(rc);
    }

    void Insert(int _a){
        a = _a; Insert(root);
    }

    void Query(int _a, int _b){
        a = _a, b = _b, ss = 0, mm = 0;
        Query(root);
    }

    void Move(int tar){
        while (cur <= tar){
            ECH(it, adj[cur]) Insert(*it);
            ++cur;
        }
    }

    void STInit(){
        //REP(i, n) CLR(adj[i]);
        //cur = 0;
    }

#undef ml

} using namespace Segment_Tree;


namespace SHash{

    uLL S[N], P[N];
    LL ans; int minLen;

    uLL h(int a, int b){
        return S[b]-S[a-1]*P[b-a+1];
    }

    void init(){
        P[0] = 1, S[0] = 0; REP_1(i, n) P[i] = P[i-1] * (C+1), S[i] = S[i-1] * (C+1) + (str[i]-'a'+1);
        ans = 0, minLen = n;
    }

    void jud(int &p1, int p2){
        int l = 0, r = minLen; while(l<r){
            int m = l+r>>1;
            if (h(p1,p1+m)==h(p2,p2+m)) l = m+1;
            else r = m;
        }
        if (str[p2+l]<str[p1+l]) p1 = p2;
    }

} using namespace SHash;


int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out2.txt”, ”w”, stdout);
#endif

    NIL = new node(); NIL->reset();

    n = strlen(RS(str+1)); reverse(str+1, str+n+1); get_pi(str, n, pi); reverse(str+1, str+n+1);
    REP_1(i, n) adj[n-pi[i]].PB(n-i); Init(); REP_1(i, n) Ext(str[i]-’a’); Run();

    init(); FOR(i, 1, tot){

        int u = Q[i], L = max(ll[u]-pi[u],dd[u],len[f]+1), R=len[u]; if(L>R) continue;

        if(rr[u] == n){
            ans += R - L + 1, ml[u] = L;
        }
        else{
            Move(rr[u]); int l = rr[u]-R, r = rr[u]-L; Query(l, r); if (!ss) continue;
            ans += ss, ml[u] = rr[u] - mm;
        }

        checkMin(minLen, ml[u]);
    }

    OT(ans);

    int st, u; FOR_N(u, 1, tot) if (ml[u] == minLen){st = ll[u]-minLen+1; break;}
    FOR_N(u, u+1, tot) if (ml[u] == minLen) jud(st, ll[u]-minLen+1);
    FOR(i, st, st+minLen) putchar(str[i]); puts(””);
}
```

Part VI

数学 (Math)

Chapter 21

数论 (Number Theory)

21.1 表头

```
 //基础包。。
// <<= ’2. Number Theory .,//{
namespace NT{
#define gcd __gcd
inline LL lcm(LL a, LL b){return a*b/gcd(a,b);}

inline void INC(int &a, int b){a += b; if (a >= MOD) a -= MOD;}
inline int sum(int a, int b){a += b; if (a >= MOD) a -= MOD; return a;}
/* 模数两倍刚好超 int 时。
inline int sum(uint a, int b){a += b; a %= MOD;if (a < 0) a += MOD; return a;}
inline void INC(int &a, int b){a = sum(a, b);}
*/

inline void DEC(int &a, int b){a -= b; if (a < 0) a += MOD;}
inline int dff(int a, int b){a -= b; if (a < 0) a += MOD; return a;}
inline void MUL(int &a, int b){a = (LL)a * b % MOD;}
inline int pdt(int a, int b){return (LL)a * b % MOD;}

inline int gcd(int m, int n, int &x, int &y){

    x = 1, y = 0; int xx = 0, yy = 1, q;

    while (1){
        q = m / n, m %= n;
        if (!m){x = xx, y = yy; return n;}
        DEC(x, pdt(q, xx)), DEC(y, pdt(q, yy));
        q = n / m, n %= m;
        if (!n) return m;
        DEC(xx, pdt(q, x)), DEC(yy, pdt(q, y));
    }
}

inline int sum(int a, int b, int c){return sum(a, sum(b, c));}
inline int sum(int a, int b, int c, int d){return sum(sum(a, b), sum(c, d));}
inline int pdt(int a, int b, int c){return pdt(a, pdt(b, c));}
inline int pdt(int a, int b, int c, int d){return pdt(pdt(a, b), pdt(c, d));}

inline int pow(int a, LL b){
    int c(1); while (b){
        if (b&1) MUL(c, a);
        MUL(a, a), b >>= 1;
    }
    return c;
}

template<class T> inline T pow(T a, LL b){
    T c(1); while (b){
        if (b&1) c *= a;
        a *= a, b >>= 1;
    }
    return c;
}

template<class T> inline T pow(T a, int b){
    return pow(a, (LL)b);
}

inline int _I(int b){
    int a = MOD, x1 = 0, x2 = 1, q; while (1){
        q = a / b, a %= b;
        if (!a) return x2;
        DEC(x1, pdt(q, x2));

        q = b / a, b %= a;
        if (!b) return x1;
        DEC(x2, pdt(q, x1));
    }
}

inline void DIV(int &a, int b){MUL(a, _I(b));}
inline int qtt(int a, int b){return pdt(a, _I(b));}

} using namespace NT;//}

//。。自带取模的环类。。
struct Int{
    int val;

    operator int() const{return val;}

    Int(int val = 0):val(val){
        val %= MOD; if (val < 0) val += MOD;
    }
    Int(LL _val){
        _val %= MOD; if (_val < 0) _val += MOD;
        val = _val;
    }
    Int& operator +=(const int& rhs){INC(val, rhs);rTs;}
    Int operator +(const int& rhs) const{return sum(val, rhs);}
    Int& operator -=(const int& rhs){DEC(val, rhs);rTs;}
    Int operator -(const int& rhs) const{return dff(val, rhs);}
    Int& operator *=(const int& rhs){MUL(val, rhs);rTs;}
    Int operator *(const int& rhs) const{return pdt(val, rhs);}
    Int& operator /=(const int& rhs){DIV(val, rhs);rTs;}
    Int operator /(const int& rhs) const{return qtt(val, rhs);}
    Int operator-()const{return MOD-*this;}
};


//线性素筛。。
const int PMAX = 46341;
VI P; bitset<PMAX> isC;
#define ii (i*P[j])
void sieve(){
    FOR(i, 2, PMAX){
        if (!isC[i]) P.PB(i);
        for (int j=0;j<SZ(P)&&ii<PMAX;++j){
            isC[ii]=1; if (!(i%P[j])) break;
        }
    }
}
#undef ii

//因数分解。。
```

```
VII fac; void fact(int x){
    int z = x; fac.clear(); ECH(it, P) if (!(x%*it)){
        int c=1; x/=*it; while (!(x%*it)) x/=*it, ++c;
        fac.PB(MP(*it, c));
    }
    if (x!=1) fac.PB(MP(x, 1));
}

//最小素因子。。
const int PMAX = 46341;
VI P; bitset<PMAX> isC; int p[PMAX];
#define ii (i*P[j])
void sieve(){
    FOR(i, 2, PMAX){
        if (!isC[i]) P.PB(i),p[i]=i;
        for (int j=0;j<SZ(P)&&ii<PMAX;++j){
            isC[ii]=1; p[ii]=P[j]; if (!(i%P[j])) break;
        }
    }
}
#undef ii

//欧拉 phi 函数
const int PMAX = 46341;
VI P; bitset<PMAX> isC; int phi[PMAX];
#define ii (i*P[j])
void sieve(){
    phi[1] = 1; FOR(i, 2, PMAX){
        if (!isC[i]) P.PB(i),phi[i]=i-1;
        for (int j=0;j<SZ(P)&&ii<PMAX;++j){
            isC[ii]=1;if (!(i%P[j])){
                phi[ii] = phi[i]*P[j];
                break;
            }
            else{
                phi[ii] = phi[i]*P[j]-phi[i];
            }
        }
    }
}
#undef ii


//莫比乌斯 mu 函数
const int PMAX = 46341;
VI P; bitset<PMAX> isC; int mu[PMAX];
#define ii (i*P[j])
void sieve(){
    mu[1] = 1; FOR(i, 2, PMAX){
        if (!isC[i]) P.PB(i),mu[i]=-1;
        for (int j=0;j<SZ(P)&&ii<PMAX;++j){
            isC[ii]=1;if (!(i%P[j])){
                mu[ii] = 0;
                break;
            }
            else{
                mu[ii] = -mu[i];
            }
        }
    }
}
#undef ii

//原根
int getPrimitive(int p){
    --p; VI d; for (int i=2;i*i<=p;++i) if (!(p%i)) d.PB(i), d.PB(p/i);
    UNQ(d); MOD = ++p; FOR(i, 2, p){
        int j = 0; REP_N(j, SZ(d)) if (pow(i, d[j]) == 1) break;
        if (j == SZ(d)) return i;
    }
    assert(0);
    return -1; //!
}

// 离散对数


struct HashTable{

    int index[PMAX], head[PMAX], next[PMAX], sz;
    int key[PMAX];

    inline void clear() {
        sz = 0;
        memset(head, -1, sizeof(head));
    }
    inline void insert(int id, int val) {
        int x = val % PMAX;
        index[sz] = id, key[sz] = val;
        next[sz] = head[x];
        head[x] = sz++;
    }
    inline int search(int val){
        int x = val % PMAX;
        for ( int it = head[x] ; it != -1 ; it = next[it] )
            if ( key[it] == val ) return index[it];
        return -1;
    }
} H;

#define p MOD

int Dlog(int a, int b){

    a %= p, b %= p;

    // Baby-Step ...
    int m = ceil(sqrt(p));

    static Int A[PMAX]; A[0] = 1%p; REP_1(i, m){
        A[i] = A[i-1] * a;
        if (A[i] == b) return i;
    }

    H.clear(); REP(i, m){
        H.insert(i, A[i]);
    }

    // Giant-Step
    Int bb = b, am = _I(A[m]); FOR(i, 1, m){
        bb *= am; int j = H.search(bb);
        if (~j) return i*m+j;
    }
    return -1;
}

const int LN = 32;
int o;

int exDlog(int a, int b){

    a %= p, b %= p;

    Int aa = 1%p; int x0 = -1; REP(i, LN){
        if (aa == b){x0 = i; break;}
        aa *= a;
    }

    aa = 1; o = 0; for (int d = gcd(a, p); d != 1; d = gcd(a, p)){
        ++o; if (b % d) return ~x0 ? x0 : -1;
        b /= d, p /= d, aa *= a/d;
    }

    if (~x0) return x0;

    int z = Dlog(a, qtt(b, aa)); if (~z) z += o;
    return z;
}

#undef p


// 128位乘

#define m _mod
LL _mod;
inline void Incc(LL&x,LL y){
    x += y; if (x >= m) x -= m;
}

inline LL pdtt(LL x,LL y){
    x%=m,y%=m;
    LL p=sqrt(m)+0.5,q=p*p-m,a=x/p,b=x%p,c=y/p,d=y%p,e=a*c/p*q,f=a*c%p*q;
    LL t=((a*d+b*c)%m+e)%m;x=t/p*q,y=t%p*p;
    x = (((b*d+f)%m+x)%m+y)%m; if (x<0) x += m;
    return x;
}

inline LL pdtt1(LL x,LL y){
    x%=m,y%=m;
    x =(x*y-(LL)(((long double)x*y+0.5)/(long double)m)*m)%m; if (x<0) x += m;
    return x;
}

inline LL pdtt2(LL a,LL b){
    a%=m,b%=m;
    LL c=0; for (;b;b>>=1,Incc(a,a))
        if (b&1) Incc(c,a);
    return c;
}
#undef m


//Rho && Miller_Rabin
map<LL, int> fac;

inline LL pdtt(LL x,LL y,LL z){
    return (x*y-(LL)(((long double)x*y+0.5)/z)*z+z)%z;
}

inline LL poww(LL a, LL b, LL z){
    LL c = 1; for (;b;a=pdtt(a, a, z),b>>=1)
        if (b&1) c = pdtt(c, a, z);
    return c;
}

inline bool Miller_Rabin(int p,LL n){
    int t=0; LL u=n-1; while(!(u&1)) ++t,u>>=1;
    LL x=poww(p,u,n); if(x==1) return 1; DO(t){
        if(x==n-1) return 1;
        x=pdtt(x,x,n);
    }
    return 0;
}

inline bool isPrime(LL n){
    static const int P[]={2,3,5,7,11,13,17,19,23}, Pn = 9;
    if(find(P,P+Pn,n)!=P+Pn) return 1; if(n==1||!(n&1)) return 0;
    REP(i, Pn) if(!Miller_Rabin(P[i],n)) return 0;
    return 1;
}

void rho(LL n){

    if(isPrime(n)){
        fac[n]++;
        return;
    }

    int c=1; while(1){

        LL x1=1,x2=1; int i=1,k=2; while(1){

            x1=pdtt(x1,x1,n)+c;
            LL d=gcd(abs(x1-x2),n);
            if(d!=1&&d!=n){
                rho(d),rho(n/d);
                return;
            }
            if(x1==x2) break;
            if(++i==k) k<<=1,x2=x1;
        }
        ++c;
    }
}

//
```

21.1.1

```
/*
Int Fact[N], Factt[N]; Int Binom(int n, int m){
    return Fact[n] * Factt[m] * Factt[n-m];
}
*/

/*
    Fact[0] = 1; REP_1(i, N-1) Fact[i] = Fact[i-1] * i;
    Factt[N-1] = _I(Fact[N-1]); DWN(i, N, 1) Factt[i-1] = Factt[i] * i;
*/

/*
    int Binom[N][N];
    REP(i, N){Binom[i][0] = 1; REP_1(j, i) Binom[i][j] = Binom[i-1][j-1] + Binom[i-1][j];}
*/


/*
const int PMAX = 1;
VI P; bitset<PMAX> isP;
void sieve(){
    FOR(i, 2, PMAX){
        if (!isP[i]) P.PB(i);
        for (int j=0;j<SZ(P)&&i*P[j]<PMAX;++j){
            isP[i*P[j]]=1; if (!(i%P[j])) break;
        }
    }
}
*/

/*
inline int phi(int n){
    int res = n; for (int i=2;sqr(i)<=n;++i) if (!(n%i)){
        DEC(res, qtt(res, i));
        do{n /= i;} while(!(n%i));
    }
    if (n != 1)
        DEC(res, qtt(res, n));
    return res;
}
*/

/*LL d, x, y; void exGcd(LL a, LL b){
    if(!b) x = 1, y = 0, d = a;
    else{
        exGcd(b, a%b); LL t = y;
        y = x - (a/b)*y, x = t;
    }
}*/

} using namespace NT;//}
```

21.2 莫比乌斯反演

约数形式

$$g(n) = \sum_{d \mid n} f(d) \quad \text{for every integer } n \geq 1$$

$$f(n) = \sum_{d \mid n} \mu(d) g(n/d) \quad \text{for every integer } n \geq 1$$

倍数形式

$$g(d) = \sum_{d \mid n} f(n) \quad \text{for every integer } n \geq 1$$

$$f(d) = \sum_{d \mid n} \mu(n) g(n) \quad \text{for every integer } n \geq 1$$

```


线性素筛。。

const int PMAX = 46341; //ceil(sqrt(_U(31)));
VI P; bitset<PMAX> isC;
#define ii (i*P[j])
void sieve(){
    FOR(i, 2, PMAX){
        if (!isC[i]) P.PB(i);
        for (int j=0;j<SZ(P)&&ii<PMAX;++j){
            isC[ii]=1; if (!(i%P[j])) break;
        }
    }
}
#undef ii

因数分解。。

VII fac; void fact(int x){
    int z = x; fac.clear(); ECH(it, P) if (!(x%*it)){
        int c=1; x/=*it; while (!(x%*it)) x/=*it, ++c;
        fac.PB(MP(*it, c));
    }
    if (x!=1) fac.PB(MP(x, 1));
}


最小素因子。。

const int PMAX = 46341; //ceil(sqrt(_U(31)));
VI P; bitset<PMAX> isC; int p[PMAX];
#define ii (i*P[j])
void sieve(){
    FOR(i, 2, PMAX){
        if (!isC[i]) P.PB(i),p[i]=i;
        for (int j=0;j<SZ(P)&&ii<PMAX;++j){
            isC[ii]=1; p[ii]=P[j]; if (!(i%P[j])) break;
        }
    }
}
#undef ii

欧拉 phi 函数

const int PMAX = 46341; //ceil(sqrt(_U(31)));
VI P; bitset<PMAX> isC; int phi[PMAX];
#define ii (i*P[j])
void sieve(){
    phi[1] = 1; FOR(i, 2, PMAX){
        if (!isC[i]) P.PB(i),phi[i]=i-1;
        for (int j=0;j<SZ(P)&&ii<PMAX;++j){
            isC[ii]=1;if (!(i%P[j])){
                phi[ii] = phi[i]*P[j];
                break;
            }
            else{
                phi[ii] = phi[i]*P[j]-phi[i];
            }
        }
    }
}
#undef ii


莫比乌斯 mu 函数 。。。

const int PMAX = 46341; //ceil(sqrt(_U(31)));
VI P; bitset<PMAX> isC; int mu[PMAX];
#define ii (i*P[j])
void sieve(){
    mu[1] = 1; FOR(i, 2, PMAX){
        if (!isC[i]) P.PB(i),mu[i]=-1;
        for (int j=0;j<SZ(P)&&ii<PMAX;++j){
            isC[ii]=1;if (!(i%P[j])){
                mu[ii] = 0;
                break;
            }
            else{
                mu[ii] = -mu[i];
            }
        }
    }
}
#undef ii
---------
```

```
const int PMAX = int(1e5) + 9;
VI P; bitset<PMAX> isP; int mu[PMAX];
void sieve(){
    mu[1] = 1; FOR(i, 2, PMAX){
        if (!isP[i]) P.PB(i), mu[i] = -1;
        for (int j=0;j<SZ(P)&&i*P[j]<PMAX;++j){
            isP[i*P[j]]=1; if (!(i%P[j])){
                mu[i*P[j]] = 0;
                break;
            } else{
                mu[i*P[j]] = -mu[i];
            }
        }
    }
}

const int N = 109;
int A[N], B[N];
int n;

void r(){

    REP_1(i, n){
        B[i] = 0;
        REP_1(d, i) if (i%d == 0) B[i] += A[d];
    }

    REP(i, n) A[i] = B[i];
}

void l(){

    REP_1(i, n){
        B[i] = 0;
        REP_1(d, i) if (i%d == 0) B[i] += mu[d]*A[i/d];
    }

    REP(i, n) A[i] = B[i];
}


int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    sieve();

    n = 10; REP_1(i, n) A[i] = i*i;

    r(); r(); r();


    REP(i, n) printf(”%d ”, A[i]); puts(””);
}


https://en.wikipedia.org/wiki/M%C3%B6bius_inversion_formula#Repeated_transformations

莫比乌斯变换 ↑
莫比乌斯逆变换 ↓
```

```
Sigma_0
-2：
Mobius Function：
http://oeis.org/A008683
-1：
单位函数
1,0,0,0,0,0...
0：
常函数
1,1,1,1,1,1
1：
http://oeis.org/A000005
d0：0阶约数函数。。。约数的个数。。
tau2：拆成两个因子的顺序方案数。（顺序相关）。。。（不涉及相关其他数列的时候不强调下标.。。）
2:
???
http://oeis.org/A007425
tau3：拆成三个因子的方案。（顺序相关）。以下为 tau 4 tau 5.。。
3:
http://oeis.org/A007426 ;
4:
http://oeis.org/A061200



Sigma_1
-2：
http://oeis.org/A007431
-1：
http://oeis.org/A000010 ;
Euler totient function phi(n)
0：
1,2,3,4,5,6。。。
算数序列
1：
sigma_1
http://oeis.org/A000203
2：
http://oeis.org/A007429




Sigma_2:
查不到。1,2,7,9,23,14,47,36,64
Jordan_totient_Function .... // https://en.wikipedia.org/wiki/Jordan's_totient_function
平方序列
sigma_2
http://oeis.org/A007433





(前 n 项 phi 的和。。。
http://oeis.org/A002088 ??)
http://acm.hdu.edu.cn/showproblem.php?pid=1695


暴力容斥原理 （2000ms）

const int PMAX = int(1e5) + 9;
VI P; bitset<PMAX> isP; int p[PMAX];
void sieve(){
    p[1]=1;FOR(i, 2, PMAX){
        if (!isP[i]) P.PB(i),p[i]=i;
        for (int j=0;j<SZ(P)&&i*P[j]<PMAX;++j){
            isP[i*P[j]]=1,p[i*P[j]]=P[j]; if (!(i%P[j])) break;
        }
    }
}

int a, b, k;

int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    sieve(); Rush{
        int _; RD(_, a, _, b, k); if (!k) {OT(0); continue;}; a /= k, b /= k;
        LL z = 0; if (a > b) swap(a, b); REP_1(bb, b){
            int aa = min(bb, a); VI D; int x = bb; while (x != 1) D.PB(p[x]), x /= p[x]; UNQQ(D);
            REP(s, _1(SZ(D))){
                int c = 0, d = 1; REP(i, SZ(D)) if (_1(s, i)) ++c, d*=D[i];
                if (c&1) z -= aa/d; else z += aa/d;
            }
        }
        OT(z);
    }
}
```

莫比乌斯反演

```
const int PMAX = int(1e5) + 9;
VI P; bitset<PMAX> isP; int mu[PMAX];
void sieve(){
    mu[1] = 1; FOR(i, 2, PMAX){
        if (!isP[i]) P.PB(i), mu[i] = -1;
        for (int j=0;j<SZ(P)&&i*P[j]<PMAX;++j){
            isP[i*P[j]]=1; if (!(i%P[j])){
                mu[i*P[j]] = 0;
                break;
            } else{
                mu[i*P[j]] = -mu[i];
            }
        }
    }
}

int a, b, k;

int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    sieve(); Rush{
        int _; RD(_, a, _, b, k); if (!k) {OT(0); continue;}; a /= k, b /= k;
        LL z = 0; if (a > b) swap(a, b);
        REP_1(i, a) z -= (LL)mu[i]*(a/i)*(a/i); z /= 2;
        REP_1(i, a) z += (LL)mu[i]*(a/i)*(b/i); OT(z);
    }
}
```

```
分拆
https://oeis.org/A000041

k部-分拆
https://oeis.org/A026820



A[i][j] 表示：
 i 分拆成最大数不超过 j 的方案数。。）
A[0][0] = 1
A[i][j] = A[i][j-1] + A[i-j][min(i-j, j)];
不拆出 j 的方案书 + 至少拆除一个 j 的方案数。。

根据共轭性也可以将 A[i][j] 理解成。。
。。。 i 分拆成不多于 j 项的方案数。
那么此时转移可以理解成。。小于 j 项的方案数 + 恰好含有 j 项的方案数。。。

const int N = 109;
int A[N][N], n;

int main(){
      n = 5; A[0][0] = 1; REP_1(i, n) REP_1(j, i)
            A[i][j] = A[i][j-1] + A[i-j][min(i-j, j)];

      REP_1(i, n){
            REP_1(j, i) printf(”%d ”, A[i][j]);
            puts(””);
      }
}

/*
1
1 2
1 2 3
1 3 4 5
1 3 5 6 7
*/


不相同 k-部分拆
https://oeis.org/A000009



。。。显然不相同 k 部分拆不满足共轭性质。。。。
也就是


最大项为 k != 项数为 k

1.
设 A[i][j]：i 的最大项不大于 j 的不重复 k-部分拆。。。
/*。。
至多含有一个 j 的 = 至多含有 j-1 的 + 恰好含有一个 j 的。
*/

const int N = 109;
int A[N][N], n;

int main(){

#ifndef ONLINE_JUDGE
```

```
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    n = 8; A[0][0] = 1; REP_1(i, n) REP_1(j, i)
        A[i][j] = A[i][j-1] + A[i-j][min(i-j, j-1)];
    REP_1(i, n){
        REP_1(j, i) printf(”%d ”, A[i][j]);
        puts(””);
    }
}


2.
设 A[i][j]: 项数恰好为 j 项的不重复 k 部分拆数。

最小数 >1： A[i-j][j] （相当于每个数+1、
最小数 =1：A[i-j][[j-1]（新增加一个 1，原先的 j-1 项再每个数+1，

    n = 10; A[0][0] = 1; REP_1(i, n) REP_1(j, i)
        A[i][j] = A[i-j][j-1] + A[i-j][j];
```

21.2.1 TYVJ 1858. xlkxc

简述 (Brief description)

给定 n, m, A, D, P，设 $f(n) = \sum_{i=1}^{n} i^m$，$g(n) = \sum_{i=1}^{n} f(i)$，
求 $\sum_{i=0}^{n} g(A + iD) \bmod P$
（$0 \leq n, A, D \leq 10^8$,，$1 \leq m \leq 10^2$，P 为素数。）

分析 (Analysis)

预处理 令 $++m$，$f_m(x) = \sum_{i=1}^{n} i^{m-1}$，$f_m(x)$ 可以表达成 x 的 m 阶多项式（常数项系数为 0）。

设 $f_m(x) = \sum_{i=1}^{m} a_{m,i} x^i$。

根据贝努利公式可以在 $O(m^2)$ 时间复杂度内预处理出 $a_{m,i}$。

$f_m(x)$ 的前缀和为 $m+1$ 阶多项式。

类似定义 $g_m(x) = \sum_{i=1}^{m+1} b_{m,i} x^i$，有

$$\begin{aligned} g_m(x) &= \sum_{x'=1}^{x} \sum_{i=1}^{m} a_{m,i} x'^i \\ &= \sum_{i=1}^{m} a_i \sum_{x'=1}^{x} x'^i \\ &= \sum_{i=1}^{m} a_i f_{i+1}(x) \\ &= \sum_{i=1}^{m} a_i \sum_{j=1}^{i+1} a_{i+1,j} x^j \end{aligned} \tag{21.1}$$

因此我们可以在 $O(m^2)$ 时间复杂度内预处理出 $b_{m,i}$。

考虑询问

$$
\begin{aligned}
\sum_{i=0}^{n} g_m(A+iD) &= \sum_{i=0}^{n}\sum_{j} b_{m,j}(A+iD)^j \\
&= \sum_{i=0}^{n}\sum_{j} b_j \sum_{k=0}^{j}\binom{j}{k}(iD)^k A^{j-k} \\
&= \sum_{j} b_{m,j} \sum_{k=0}^{j}\binom{j}{k} D^k A^{j-k} \sum_{i=0}^{n} i^k \\
&= \sum_{j} b_{m,j} \sum_{k=0}^{j}\binom{j}{k} D^k A^{j-k} \sum_{i=0}^{n} i^k
\end{aligned}
\tag{21.2}
$$

把最右边的等幂求和代入 $f_{k+1}(n)$ 求值即可（注意特判 $k=0$）。
每次询问的时间复杂度为 $O(m^2)$。

```
const int N = 150;
Int Binom[N+1][N+1], B[N], a[N][N], b[N][N], A, D;
int m, n;

void init(){

    REP(i, N+1){Binom[i][0] = 1; REP_1(j, i) Binom[i][j] = Binom[i-1][j-1] + Binom[i-1][j];}

    B[0] = 1; FOR(i, 1, N){
        REP(j, i) B[i] += B[j] * Binom[i+1][j];
        B[i] /= -Binom[i+1][i];
    }

    B[1] = 1; B[1] /= 2; FOR(i, 1, N){
        REP_1(j, i) a[i][j] = B[i-j]*Binom[i][i-j]/i;
    }

    FOR(i, 1, N){
        REP_2_1(j, k, i, j+1) b[i][k] += a[i][j]*a[j+1][k];
    }
}

Int f(int n, int m){
    Int z=0; DWN_1(i, m, 1) z += a[m][i], z *= n;
    if (m == 1) z += 1; //!
    return z;
}

Int g(int n, int m){
    Int z=0; DWN_1(i, m+1, 1){
        z += b[m][i], z *= n;
    }
    return z;
}

Int gg(){

    Int z=0; REP_1(i, m+1){
        Int zz=0; REP(j, i+1)
            zz += Binom[i][j]*pow(D, j)*pow(A, i-j)*f(n, j+1);
        z += zz*b[m][i];
    }

    /*Int z=0; REP(ii, n+1){
        REP_1(i, m+1){
            Int zz=0; REP(j, i+1)
                zz += Binom[i][j]*pow(D, j)*pow(A, i-j)*pow(ii, j); //f(n, i-j+1);
            z += zz*b[m][i];
        }
    }*/

    return z;
}


int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif
   //printf(”%I64d\n”,gcd(-6,12));

    init(); Rush{
        ++RD(m, A, n, D);
        OT(gg());

        /*REP_1(i, 10){cout << f(i, 1) << ” ”;}cout << endl;
        REP_1(i, 10){cout << g(i, 1) << ” ”;}cout << endl;

        Int ss = 0; REP(i, n){
            ss += g(A+i*D, m); cout << g(A+i*D, m) << ”+”;
        }
        ss += g(A+n*D, m); cout << g(A+n*D, m) << ”=”;
        cout << ss << endl;*/
    }
}
```

..

Part VII

计算几何 (Computational Geometry)

Chapter 22

2D-几何基础

```
#define Ts *this
#define rTs return Ts
          ..
typedef long long LL;
typedef double DB;
          ...
const DB EPS = 1e-9;
const DB OO = 1e20;
const DB PI = acos(-1.0); //M_PI;
          ...
inline int sgn(DB x){return x<-EPS?-1:x>EPS;}
inline int sgn(DB x, DB y){return sgn(x-y);}
          ..
```

22.1 点

```
// <<= ’9. Comutational Geometry .,//{
namespace CG{

#define cPo const Po&
#define cLine const Line&
#define cSeg const Seg&

inline DB dist2(DB x,DB y){return sqr(x)+sqr(y);}

struct Po{
    DB x,y;Po(DB x=0,DB y=0):x(x),y(y){}

    void in(){RF(x,y);}void out(){printf(”(%.2f,%.2f)”,x,y);}
    inline friend istream&operator>>(istream&i,Po&p){return i>>p.x>>p.y;}
    inline friend ostream&operator<<(ostream&o,Po p){return o<<”(”<<p.x<<”, ”<<p.y<< ”)”;}

    Po operator-()const{return Po(-x,-y);}
    Po&operator+=(cPo p){x+=p.x,y+=p.y;rTs;}Po&operator-=(cPo p){x-=p.x,y-=p.y;rTs;}
    Po&operator*=(DB k){x*=k,y*=k;rTs;}Po&operator/=(DB k){x/=k,y/=k;rTs;}
    Po&operator*=(cPo p){rTs=Ts*p;}Po&operator/=(cPo p){rTs=Ts/p;}
    Po operator+(cPo p)const{return Po(x+p.x,y+p.y);}Po operator-(cPo p)const{return Po(x-p.x,y-p.y);}
    Po operator*(DB k)const{return Po(x*k,y*k);}Po operator/(DB k)const{return Po(x/k,y/k);}
    Po operator*(cPo p)const{return Po(x*p.x-y*p.y,y*p.x+x*p.y);}Po operator/(cPo p)const{return Po(x*p.x+y*p.y,y*p.x-x*p.y)/p.len2();}

    bool operator==(cPo p)const{return!sgn(x,p.x)&&!sgn(y,p.y);};bool operator!=(cPo p)const{return sgn(x,p.x)||sgn(y,p.y);}
    bool operator<(cPo p)const{return sgn(x,p.x)<0||!sgn(x,p.x)&&sgn(y,p.y)<0;}bool operator<=(cPo p)const{return sgn(x,p.x)<0||!sgn(x,p.x)&&sgn(y,p.y)<=0;}
    bool operator>(cPo p)const{return!(Ts<=p);}bool operator >=(cPo p)const{return!(Ts<p);}

    DB len2()const{return dist2(x,y);}DB len()const{return sqrt(len2());}DB arg()const{return atan2(y,x);}
    Po&_1(){rTs/=len();}Po&conj(){y=-y;rTs;}Po&lt(){swap(x,y),x=-x;rTs;}Po&rt(){swap(x,y),y=-y;rTs;}
    Po&rot(DB a,cPo o=Po()){Ts-=o;Ts*=Po(cos(a),sin(a));rTs+=o;}
};

    ...

} using namespace CG;
```

22.2 点积 && 叉积

```
inline DB dot(DB x1,DB y1,DB x2,DB y2){return x1*x2+y1*y2;}
inline DB dot(cPo a,cPo b){return dot(a.x,a.y,b.x,b.y);}
inline DB dot(cPo p0,cPo p1,cPo p2){return dot(p1-p0,p2-p0);}
inline DB det(DB x1,DB y1,DB x2,DB y2){return x1*y2-x2*y1;}
inline DB det(cPo a,cPo b){return det(a.x,a.y,b.x,b.y);}
inline DB det(cPo p0,cPo p1,cPo p2){return det(p1-p0,p2-p0);}
inline DB ang(cPo p0,cPo p1){return acos(dot(p0,p1)/p0.len()/p1.len());}
inline DB ang(cPo p0,cPo p1,cPo p2){return ang(p1-p0,p2-p0);}
inline DB ang(cPo p0,cPo p1,cPo p2,cPo p3){return ang(p1-p0,p3-p2);}
inline DB dist2(const Po &a, const Po &b){return dist2(a.x-b.x, a.y-b.y);}
template<class T1, class T2> inline int dett(const T1 &x, const T2 &y){return sgn(det(x, y));}
template<class T1, class T2, class T3> inline int dett(const T1 &x, const T2 &y, const T3 &z){return sgn(det(x, y, z));}
template<class T1, class T2, class T3, class T4> inline int dett(const T1 &x, const T2 &y, const T3 &z, const T4 &w){return sgn(det(x, y, z, w));}
template<class T1, class T2> inline int dott(const T1 &x, const T2 &y){return sgn(dot(x, y));}
template<class T1, class T2, class T3> inline int dott(const T1 &x, const T2 &y, const T3 &z){return sgn(dot(x, y, z));}
template<class T1, class T2, class T3, class T4> inline int dott(const T1 &x, const T2 &y, const T3 &z, const T4 &w){return sgn(dot(x, y, z, w));}
template<class T1, class T2> inline DB arg(const T1 &x, const T2 &y){DB a=ang(x,y);return~dett(x,y)?a:2*PI-a;}
template<class T1, class T2, class T3> inline DB arg(const T1 &x, const T2 &y, const T3 &z){DB a=ang(x,y,z);return~dett(x,y,z)?a:2*PI-a;}
template<class T1, class T2, class T3, class T4> inline DB arg(const T1 &x, const T2 &y, const T3 &z, const T4 &w){DB a=ang(x,y,z,w);return~dett(x,y,z,w)?a:2*PI-a;}
template<class T1, class T2> inline DB dist(const T1 &x, const T2 &y){return sqrt(dist2(x, y));}
template<class T1, class T2, class T3> inline DB dist(const T1 &x, const T2 &y, const T3 &z){return sqrt(dist2(x, y, z));}
inline Po _1(Po p){return p._1();}inline Po conj(Po p){return p.conj();}
inline Po lt(Po p){return p.lt();}inline Po rt(Po p){return p.rt();}
inline Po rot(Po p,DB a,cPo o=Po()){return p.rot(a,o);}
inline Po operator *(DB k,cPo p){return p*k;}
inline Po operator /(DB k,cPo p){return conj(p)*k/p.len2();}

typedef vector<Po> VP;
```

22.3 直线

```
struct Line{
    Po a,b;Line(cPo a=Po(),cPo b=Po()):a(a),b(b){}
    Line(DB x0,DB y0,DB x1,DB y1):a(Po(x0,y0)),b(Po(x1,y1)){}
    Line(cLine l):a(l.a),b(l.b){}

    //Ax+By+C=0
    Line(DB A,DB B,DB C){
        C=-C;if(!::sgn(A))a=Po(0,C/B),b=Po(1,C/B);
        else if(!::sgn(B))a=Po(C/A,0),b=Po(C/A,1);
        else a=Po(0,C/B),b=Po(1,(C-A)/B);
    }

    void in(){a.in(),b.in();}
    inline friend istream&operator>>(istream&i,Line& p){return i>>p.a>>p.b;}
    inline friend ostream&operator<<(ostream&o,Line p){return o<<p.a<<”-”<< p.b;}

    Line operator+(cPo x)const{return Line(a+x,b+x);}
    Line operator-(cPo x)const{return Line(a-x,b-x);}
    Line operator*(DB k)const{return Line(a*k,b*k);}
    Line operator/(DB k)const{return Line(a/k,b/k);}

    Po operator*(cLine)const;
    Po d()const{return b-a;}DB len2()const{return d().len2();}DB len()const{return d().len();}DB arg()const{return d().arg();}

    int sgn(cPo p)const{return dett(a, b, p);}
    int sgn(cLine)const;

    bool sameSgn(cPo p1,cPo p2)const{return sgn(p1)==sgn(p2);}
    void getEquation(DB&K,DB&B)const{
        K = ::sgn(a.x, b.x) ? (b.y-a.y)/(b.x-a.x) : OO;
        B = a.y - K*a.x;
    }
    void getEquation(DB&A,DB&B,DB&C)const{A=a.y-b.y,B=b.x-a.x,C=det(a, b);}

    Line&push(DB r){ // 正数右手螺旋向里
        Po v=d()._1().lt()*r;a+=v,b+=v; rTs;
    }
};

inline DB dot(cLine l1,cLine l2){return dot(l1.d(),l2.d());}
inline DB dot(cLine l,cPo p){return dot(l.a,l.b,p);}
inline DB dot(cPo p,cLine l){return dot(p,l.a,l.b);}
inline DB det(cLine l1,cLine l2){return det(l1.d(),l2.d());}
inline DB det(cLine l,cPo p){return det(l.a,l.b,p);}
inline DB det(cPo p,cLine l){return det(p,l.a,l.b);}
inline DB ang(cLine l0,cLine l1){return ang(l0.d(),l1.d());}
inline DB ang(cLine l,cPo p){return ang(l.a,l.b,p);}
inline DB ang(cPo p,cLine l){return ang(p,l.a,l.b);}

inline int Line::sgn(cLine l)const{return dett(Ts, l);}
inline Po Line::operator*(cLine l)const{return a+d()*det(a,l)/det(Ts,l);}
inline Po operator&(cPo p,cLine l){return l*Line(p,p+l.d().lt());}
inline Po operator%(cPo p,cLine l){return p&l*2-p;}
inline Line push(Line l, DB r){return l.push(r);}
```

22.4 线段

```
struct Seg: public Line{
    Seg(cPo a=Po(),cPo b=Po()):Line(a,b){}
    Seg(DB x0,DB y0,DB x1,DB y1):Line(x0,y0,x1,y1){}
    Seg(cLine l):Line(l){}
    Seg(const Po &a,DB alpha):Line(a,alpha){}
    Seg(DB A,DB B,DB C):Line(A,B,C){}

    inline int sgn(cPo p)const;
    inline bool qrt(cSeg l)const;
    inline int sgn(cSeg l)const;
};

 // -1不相交 0相交（不规范） 1相交（规范）

inline int Seg::sgn(cPo p)const{return -dott(p,a,b);}

// quick_rejection_test
inline bool Seg::qrt(cSeg l)const{
    return min(a.x,b.x)<=max(l.a.x,l.b.x)&&min(l.a.x,l.b.x)<=max(a.x,b.x)&&
        min(a.y,b.y)<=max(l.a.y,l.b.y)&&min(l.a.y,l.b.y)<=max(a.y,b.y);
}
```

```


inline int Seg::sgn(cSeg l)const{
    if (!qrt(l)) return -1;

    /*return
        (dett(a,b,l.a)*dett(a,b,l.b)<=0 &&
        dett(l.a,l.b,a)*dett(l.a,l.b,b)<=0)?1:-1;*/

    int d1=dett(a,b,l.a),d2=dett(a,b,l.b),d3=dett(l.a,l.b,a),d4=dett(l.a,l.b,b);
    if ((d1^d2)==-2&&(d3^d4)==-2)return 1;
    return ((!d1&&dott(l.a-a,l.a-b)<=0)||(!d2&&dott(l.b-a,l.b-b)<=0)||
            (!d3&&dott(a-l.a,a-l.b)<=0)||(!d4&&dott(b-l.a,b-l.b)<=0))?0:-1;
}

//inline DB dist2(cLine l,cPo p){return sqr(fabs(dot(lt(l.d()), p-l.a)))/l.len2();}
inline DB dist2(cLine l,cPo p){return sqr(fabs(det(l.d(), p-l.a)))/l.len2();}

inline DB dist2(cLine l1,cLine l2){return dett(l1,l2)?0:dist2(l1,l2.a);}

inline DB dist2(cSeg l,cPo p){
    Po pa = p - l.a, pb = p - l.b;
    if (dott(l.d(), pa) <= 0) return pa.len2();
    if (dott(l.d(), pb) >= 0) return pb.len2();
    return dist2(Line(l), p);
}


inline DB dist2(cSeg s,cLine l){
    Po v1=s.a-l.a,v2=s.b-l.a;DB d1=det(l.d(),v1),d2=det(l.d(),v2);
    return sgn(d1)!=sgn(d2) ? 0 : sqr(min(fabs(d1), fabs(d2)))/l.len2();
}
inline DB dist2(cSeg l1,cSeg l2){
    if (~l1.sgn(l2)) return 0;
    else return min(dist2(l2,l1.a), dist2(l2,l1.b), dist2(l1,l2.a), dist2(l1,l2.b));
}
template<class T1, class T2> inline DB dist2(const T1& a, const T2& b){
    return dist2(b, a);
}
```

22.5 三角与圆

```
struct Triangle; struct Circle;
typedef const Triangle&cTriangle; typedef const Circle&cCircle;

const int Disjoint = -2, Exscribe = -1, Cross = 0, Inscribe = 1, Contain = 2;

Po getX3(cPo a, cPo b, cPo c){ // 外接圆圆心
    Po v0=b-a,v1=c-a;DB l0=v0.len2(),l1=v1.len2(),d=2*det(a,b,c);
    return Po(l0*v1.y-l1*v0.y,l1*v0.x-l0*v1.x)/d+a;
    //Po v0 = b-a, v1 = c-a, m0 = (a+b)/2, m1 = (a+c)/2;
    //return Line(m0,m0+v0.lt())*Line(m1,m1+v1.lt());
}

Po getX4(cPo a, cPo b, cPo c){ // 垂心
    return Line(a,a&Line(b,c))*Line(b,b&Line(a,c));
}

struct Circle{
    Po o; DB r; Circle(cPo o=Po(),DB r=0):o(o),r(r){}
    // 外接圆

    Circle(cPo a,cPo b){
        o = (a+b)/2, r = dist(a,b)/2;
    }
    Circle(cPo a,cPo b,cPo c){
        o = getX3(a,b,c), r = dist(o,a);
    }
    void in(){o.in(),RF(r);}
    void out(){
        printf(”%.2f %.2f %.2f\n”, o.x, o.y, r);
    }
    bool operator <(cCircle c)const{return r<c.r;}
    //-1相离 0圆上 1包含
    inline int sgn(cPo p)const{return ::sgn(r*r, dist2(o, p));}
    //-1相离 0相切 1包含
    inline int sgn(cLine l)const{return ::sgn(r*r, dist2(l, o));}
    // -2外离 -1外切 0相交 1内切 2包含
    inline int sgn(cCircle c)const{
        DB d=dist2(o,c.o);
        if (::sgn(sqr(r+c.r),d)<0) return Disjoint;
        if (!::sgn(sqr(r+c.r), d)) return Exscribe;
        if (!::sgn(sqr(r-c.r), d)) return Inscribe;
        if (::sgn(sqr(r-c.r), d)>0) return Contain;
        return Cross;
    }

    inline DB s(){return PI*sqr(r);}
    inline DB p(){return 2*PI*r;}

    inline Po operator^(cCircle c)const{return Po(det(Po(o.x,r),Po(c.o.x,c.r)),det(Po(o.y,r),Po(c.o.y,c.r)))/(c.r-r);}

    inline void getIntersect(cLine l,Po&p0,Po&p1)const{
        Po m = o&l, d = (l.b-l.a)._1() * sqrt(sqr(r)-dist2(l, o));
        p0 = m + d, p1 = m - d;
    }
    inline void getIntersect(cCircle c,Po&p0,Po&p1)const{
        Po v=(c.o-o)._1()*r;DB a=acos(cos(r,dist(o,c.o),c.r));
        p0=o+rot(v,a),p1=o+rot(v,-a);
    }

    inline VP operator*(cLine l)const{
        VP P; int t = sgn(l); if (t==-1) return P;
        Po p0, p1; getIntersect(l, p0, p1); P.PB(p0); if (t == 1) P.PB(p1);
        return P;
    }

    inline VP operator*(cSeg s)const{
        VP _P = Ts*Line(s), P; ECH(p, _P) if (~s.sgn(*p)) P.PB((*p));
        return P;
    }

    inline VP operator*(cCircle c)const{
        VP P; int t = abs(sgn(c)); if (t == 2) return P;
        Po p0, p1; getIntersect(c, p0, p1); P.PB(p0); if (!t) P.PB(p1);
        return P;
    }

    inline void getTangency(cPo p,Po&p0,Po&p1)const{
        DB d=dist(o,p),a=acos(r/d);Po v=(p-o)._1()*r;
        p0=o+rot(v,a),p1=o+rot(v,-a);
    }
};

struct Triangle{
    Po A,B,C; DB a,b,c; DB alpha,beta,theta;
    DB r,R; DB S,P; Po I,G,O,H;

    void init(){
        S=fabs(det(A,B,C))/2,a=dist(B,C),b=dist(A,C),c=dist(A,B);
        alpha=acos(cos(b,c,a)),beta=acos(cos(a,c,b)),theta=acos(cos(a,b,c));
        P=a+b+c,R=(a*b*c)/(4*S),r=2*S/P;
        I=Po(a*A.x+b*B.x+c*C.x,a*A.y+b*B.y+c*C.y)/P;
        G=(A+B+C)/3,O=getX3(A,B,C),H=getX4(A,B,C);
        //DB s=P/2; assert(!sgn(S, sqrt(s*(s-a)*(s-b)*(s-c)))); // 海伦公式
        //assert(!sgn(dist(I,O), sqrt(R*(R-2*r))));
        //assert(!sgn(dist(H,G), dist(O,H)*2/3));
    }

    void in(){
        A.in(),B.in(),C.in();init();
    }
};
```

22.5.1 最小覆盖圆

```
Circle getMinimalCoverCircle(VP& P){ //#
    random_shuffle(ALL(P)); int n = SZ(P);
    Circle C(P[0]); FOR(i, 1, n) if (!~C.sgn(P[i])){
        C = Circle(P[i]); REP(j, i) if (!~C.sgn(P[j])){
            C = Circle(P[i], P[j]); REP(k, j) if (!~C.sgn(P[k])){
                C = Circle(P[i], P[j], P[k]);
            }
        }
    }
    return C;
}
```

22.6 多边形

```
Po getPo(){Po p;p.in();return p;}
Line getLine(){Line l;l.in();return l;}

DB getArea(const VP& P){DB z=0;FOR(i,1,SZ(P))z+=det(P[i-1],P[i]);return z;}
DB getPeri(const VP& P){DB z=0;FOR(i,1,SZ(P))z+=dist(P[i-1],P[i]);return z;}
```

22.6.1 凸多边形面积并

```
struct Polygon{
    VP P;
    void input();
};

inline bool equal(const pair<DB, DB>& lhs, cSeg rhs){
    DB k, b; rhs.getEquation(k, b);
    return !sgn(k, lhs.fi) && !sgn(b, lhs.se);
}

DB getUnion(vector<Polygon>& P, vector<Seg>& S){

    vector<pair<DB,DB> > L; ECH(Si, S){
        DB k, b; Si->getEquation(k, b);
        L.PB(MP(k, b));
    }

    UNQ(L); DB res = 0; ECH(Li, L){

        vector<pair<DB, int> > I;
        Line l0(0,Li->se,1,Li->fi+Li->se);

        ECH(Pi, P){
            int i; FOR_N(i, 1, SZ(Pi->P)) if (equal(*Li, Seg(Pi->P[i-1], Pi->P[i]))) break;
            if (i != SZ(Pi->P)) continue;

            VP cut; FOR_N(i, 1, SZ(Pi->P)){
                Seg l1(Pi->P[i-1], Pi->P[i]); if (!dett(l0,l1)) continue;
                Po p=l0*l1; if (~l1.sgn(p)) cut.PB(p);
            }

            if (SZ(UNQ(cut)) == 2){
                I.PB(MP(cut[0].x, 1));
                I.PB(MP(cut[1].x, -1));
            }
        }

        ECH(Si, S) if (equal(*Li, *Si)){
            I.PB(MP(min(Si->a.x, Si->b.x), 2));
            I.PB(MP(max(Si->a.x, Si->b.x), -2));
        }
#define h (I[i].fi-I[i-1].fi)
#define y0 (Li->fi * I[i-1].fi + Li->se)
#define y1 (Li->fi * I[i].fi + Li->se)
        SRT(I); int c0 = 0, c1 = 0; REP(i, SZ(I)){
            if (!c0 && c1) res += (y0+y1)*h;
            if (abs(I[i].se)==1) c0 += I[i].se;
            else c1 += I[i].se;
        }
#undef h
#undef y0
#undef y1
    }

    return res;
}

DB getUnion(vector<Polygon>& P){
    vector<Seg> up, down; ECH(it, P){
        FOR(i, 1, SZ(it->P)){
            Seg s(it->P[i-1], it->P[i]); int t = sgn(s.a.x, s.b.x);
            if (t > 0) up.PB(s); else if (t < 0) down.PB(s);
        }
    }
    return getUnion(P, up) - getUnion(P, down);
}
```

Chapter 23

凸包

```
VP getCH(VP& P){ //逆时针，无共线

    int n=SZ(P); if(n<=3) return P.PB(P[0]),getArea(P)<0?RVS(P):P;

    SRT(P); VP C; C.resize(n+9); int nn = -1; REP(i, n){ //#
        while (nn > 0 && dett(C[nn-1], C[nn], P[i]) <= 0) --nn; //#
        C[++nn] = P[i];
    }

    int _nn = nn; DWN(i, n-1, 0){
        while (nn > _nn && dett(C[nn-1], C[nn], P[i]) <= 0) --nn; //#
        C[++nn] = P[i];
    }

    C.resize(nn+1);
    return C;
}
```

23.0.1 圆凸包

```
struct Triangle; struct Circle;
typedef const Triangle&cTriangle; typedef const Circle&cCircle;

const int Disjoint = -2, Exscribe = -1, Cross = 0, Inscribe = 1, Contain = 2;

Po getX3(cPo a, cPo b, cPo c){ // 外接圆圆心
    Po v0=b-a,v1=c-a;DB l0=v0.len2(),l1=v1.len2(),d=2*det(a,b,c);
    return Po(l0*v1.y-l1*v0.y,l1*v0.x-l0*v1.x)/d+a;
    //Po v0 = b-a, v1 = c-a, m0 = (a+b)/2, m1 = (a+c)/2;
    //return Line(m0,m0+v0.lt())*Line(m1,m1+v1.lt());
}

Po getX4(cPo a, cPo b, cPo c){ // 垂心
    return Line(a,a&Line(b,c))*Line(b,b&Line(a,c));
}

struct Circle{
    Po o; DB r; Circle(cPo o=Po(),DB r=0):o(o),r(r){}
    // 外接圆

    Circle(cPo a,cPo b){
        o = (a+b)/2, r = dist(a,b)/2;
    }
    Circle(cPo a,cPo b,cPo c){
        o = getX3(a,b,c), r = dist(o,a);
    }
    void in(){o.in(),RF(r);}
    void out(){
        printf(”%.2f %.2f %.2f\n”, o.x, o.y, r);
    }
    bool operator <(cCircle c)const{return r<c.r;}
    //-1相离 0圆上 1包含
    inline int sgn(cPo p)const{return ::sgn(r*r, dist2(o, p));}
    //-1相离 0相切 1包含
    inline int sgn(cLine l)const{return ::sgn(r*r, dist2(l, o));}
    // -2外离 -1外切 0相交 1内切 2包含
    inline int sgn(cCircle c)const{
        DB d=dist2(o,c.o);
        if (::sgn(sqr(r+c.r),d)<0) return Disjoint;
        if (!::sgn(sqr(r+c.r), d)) return Exscribe;
        if (!::sgn(sqr(r-c.r), d)) return Inscribe;
        if (::sgn(sqr(r-c.r), d)>0) return Contain;
        return Cross;
    }

    inline DB s(){return PI*sqr(r);}
    inline DB p(){return 2*PI*r;}

    inline Po operator^(cCircle c)const{return Po(det(Po(o.x,r),Po(c.o.x,c.r)),det(Po(o.y,r),Po(c.o.y,c.r)))/(c.r-r);}

    inline void getIntersect(cLine l,Po&p0,Po&p1)const{
        Po m = o&l, d = (l.b-l.a)._1() * sqrt(sqr(r)-dist2(l, o));
        p0 = m + d, p1 = m - d;
    }
    inline void getIntersect(cCircle c,Po&p0,Po&p1)const{
        Po v=(c.o-o)._1()*r;DB a=acos(cos(r,dist(o,c.o),c.r));
        p0=o+rot(v,a),p1=o+rot(v,-a);
    }

    inline VP operator*(cLine l)const{
        VP P; int t = sgn(l); if (t==-1) return P;
        Po p0, p1; getIntersect(l, p0, p1); P.PB(p0); if (t == 1) P.PB(p1);
        return P;
    }

    inline VP operator*(cSeg s)const{
        VP _P = Ts*Line(s), P; ECH(p, _P) if (~s.sgn(*p)) P.PB((*p));
        return P;
    }

    inline VP operator*(cCircle c)const{
        VP P; int t = abs(sgn(c)); if (t == 2) return P;
        Po p0, p1; getIntersect(c, p0, p1); P.PB(p0); if (!t) P.PB(p1);
        return P;
    }

    inline void getTangency(cPo p,Po&p0,Po&p1)const{
        DB a=acos(r/dist(o,p)); Po op=(p-o)._1()*r;
        p0=o+rot(op,a), p1=o+rot(op,-a);
    }
    inline void getTangency(cCircle c,Po&p0,Po&p1,Po&p2,Po&p3)const{
        if (!::sgn(r,c.r)){Po d=(o-c.o).rt()._1()*r;p0=o+d,p1=o-d,p2=c.o+d,p3=c.o-d;}
        else{Po p=(*this)^c; getTangency(p,p0,p1), c.getTangency(p,p2,p3);}
    }

    inline DB arc(cPo a,cPo b){
        //DB alpha = acos(cos(dist(a, o), dist(b, o), dist(a, b)));
        //if (det(o,a,b)<0) alpha = 2*PI - alpha;
        return arg(o,a,b) * r;
    }
};


struct Triangle{
    Po A,B,C; DB a,b,c; DB alpha,beta,theta;
    DB r,R; DB S,P; Po I,G,O,H;

    void init(){
        S=fabs(det(A,B,C))/2,a=dist(B,C),b=dist(A,C),c=dist(A,B);
        alpha=acos(cos(b,c,a)),beta=acos(cos(a,c,b)),theta=acos(cos(a,b,c));
        P=a+b+c,R=(a*b*c)/(4*S),r=2*S/P;
        I=Po(a*A.x+b*B.x+c*C.x,a*A.y+b*B.y+c*C.y)/P;
        G=(A+B+C)/3,O=getX3(A,B,C),H=getX4(A,B,C);
        //DB s=P/2; assert(!sgn(S, sqrt(s*(s-a)*(s-b)*(s-c)))); // 海伦公式
        //assert(!sgn(dist(I,O), sqrt(R*(R-2*r))));
        //assert(!sgn(dist(H,G), dist(O,H)*2/3));
    }

    void in(){
        A.in(),B.in(),C.in(); //init();
    }
};

Po getPo(){Po p;p.in();return p;}
Line getLine(){Line l;l.in();return l;}

DB getArea(const VP& P){DB z=0;FOR(i,1,SZ(P))z+=det(P[i-1],P[i]);return z;}
DB getPeri(const VP& P){DB z=0;FOR(i,1,SZ(P))z+=dist(P[i-1],P[i]);return z;}


VP getCH(VP& P){ //无共线

    int n=SZ(P); if(n<=3) return P.PB(P[0]),getArea(P)<0?RVS(P):P;

    SRT(P); VP C; C.resize(n+9); int nn = -1; REP(i, n){ //#
        while (nn > 0 && dett(C[nn-1], C[nn], P[i]) <= 0) --nn; //#
        C[++nn] = P[i];
    }

    int _nn = nn; DWN(i, n-1, 0){
        while (nn > _nn && dett(C[nn-1], C[nn], P[i]) <= 0) --nn; //#
        C[++nn] = P[i];
    }

    C.resize(nn+1);
    return C;
}
...
const int N = 109;

Circle C[N]; int Cn, Tn; VP P;

DB f(const VP& P){
    int n = SZ(P); VI id; id.resize(n, -1);

    REP_2(i, j, SZ(P), Cn) if (!C[j].sgn(P[i])){
        id[i] = j; break;
    }

    DB res = 0; REP(i, SZ(P)-1) res += (~id[i] && id[i] == id[i+1]) ? C[id[i]].arc(P[i], P[i+1]) : dist(P[i], P[i+1]);

    return res;
}

void add(const Po&p, const Circle&c){
    Po p0, p1; c.getTangency(p, p0, p1);
    P.PB(p0), P.PB(p1);
}

void add(const Circle&c0, const Circle&c1){
    Po p0, p1, p2, p3; c0.getTangency(c1, p0, p1, p2, p3);
    P.PB(p0), P.PB(p1), P.PB(p2), P.PB(p3);
}

int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    while(~scanf(”%d%d”, &Cn, &Tn)){

        CLR(P); REP(i, Cn) C[i].in();

        DO(Tn){
            Po a, b, c; a.in(), b.in(), c.in(); P.PB(a), P.PB(b), P.PB(c);
            REP(i, Cn) add(a, C[i]), add(b, C[i]), add(c, C[i]);
        }

        REP_2(j, i, Cn, j) add(C[i], C[j]);
        OT(SZ(P) ? f(getCH(P)) : C[0].p());
    }
}
```

23.0.2 线性动态凸包

```
const int N = 50;
Po P0[N]; DB A[N], B[N];
int n;

DB s(DB t, int &m){
    VP P; REP(i, n) P.PB( P0[i] + Po(t * A[i], t * B[i]));
    m = SZ(P);
    return fabs(getArea(getConvexHull(P)));
}
DB f(DB x, DB a, DB b, DB c){
    return a*x*x*x/3+b*x*x/2+c*x;
}
DB f(DB r, DB l, DB a, DB b, DB c){
    return a*(r*r*r-l*l*l)/3+b*(r+l)*(r-l)/2+c*(r-l);
}

DB s(DB l, DB r){

    if (r - l < EPS) return 0;

    DB m = (l + r) / 2; int nl, nm, nr;
    DB sl = s(l, nl), sm = s(m, nm), sr = s(r, nr);

    if (nl != nr){
        return s(l, m) + s(m, r);
    }

    DB a = sl / (l - r) / (l - m)
           + sm / (m - l) / (m - r)
           + sr / (r - l) / (r - m);
    DB b = sl * (-m-r) / (l - r) / (l - m)
           + sm * (-l-r) / (m - l) / (m - r)
           + sr * (-l-m) / (r - l) / (r - m);
    DB c = sl * (m*r) / (l - r) / (l - m)
           + sm * (l*r) / (m - l) / (m - r)
           + sr * (l*m) / (r - l) / (r - m);

    //cout << sl << ” ”<< sm << ” ” << sr << endl;

    //cout << f(r, a,b,c) - f(l, a,b,c) << endl;
    return f(r,a,b,c) - f(l,a,b,c);
    //return f(r,l, a,b,c);

    //return sm * (r - l);
}

int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    int T; while (~scanf(”%d%d”, &n, &T)){
        REP(i, n) P0[i].in(), RF(A[i], B[i]);

        int C0 = 3000;
        DB d = (DB)T/C0; DB res = 0, st = 0, ed = d;

        DO(C0){
            res += s(st, ed); //cout << st << ” ”<< ed << endl;
            st = ed, ed += d;
        }

        OT(res / 2 / T);
    }

}


const int N = 50;
Po P0[N]; DB A[N], B[N];
int n;

DB s(DB t){
    VP P; REP(i, n) P.PB(P0[i] + Po(t * A[i], t * B[i]));
    return fabs(getArea(getConvexHull(P)));
}

DB f(DB x, DB a, DB b, DB c){
    return a*x*x*x/3+b*x*x/2+c*x;
}
DB f(DB r, DB l, DB a, DB b, DB c){
    return a*(r*r*r-l*l*l)/3+b*(r+l)*(r-l)/2+c*(r-l);
}

DB _sl; DB s(DB l, DB r){

    DB m = (l + r) / 2, sl = _sl, sm = s(m), sr = s(r);

    DB a = sl / (l - r) / (l - m)
           + sm / (m - l) / (m - r)
           + sr / (r - l) / (r - m);
    DB b = sl * (-m-r) / (l - r) / (l - m)
           + sm * (-l-r) / (m - l) / (m - r)
           + sr * (-l-m) / (r - l) / (r - m);
    DB c = sl * (m*r) / (l - r) / (l - m)
           + sm * (l*r) / (m - l) / (m - r)
           + sr * (l*m) / (r - l) / (r - m);

    _sl = sr;
    return f(r, l,a,b,c) ;// - f(l, a,b,c);
}

int T;
```

```
#define P P0
void add(vector<DB> &I, int i, int j, int k){

    DB a = A[i]*B[j] + A[j]*B[k] + A[k]*B[i] - A[i]*B[k] - A[j]*B[i] - A[k]*B[j];
    DB b = (A[i]*P[j].y+B[j]*P[i].x) + (A[j]*P[k].y+B[k]*P[j].x) + (A[k]*P[i].y+B[i]*P[k].x)
        - (A[i]*P[k].y+B[k]*P[i].x) - (A[j]*P[i].y+B[i]*P[j].x) - (A[k]*P[j].y+B[j]*P[k].x);
    DB c = P[i].x*(P[j].y-P[k].y) + P[j].x*(P[k].y-P[i].y) + P[k].x*(P[i].y - P[j].y);

    if (!sgn(a)){
        if (!sgn(b)) return;
        DB x = -c/b; if (0 < x && x < T) I.PB(x);
        return;
    }

    DB d = b*b - 4*a*c; if (sgn(d) < 0) return; d = sqrt(d); a *= 2;
    DB x = (-b+d)/a; if (0 < x && x < T) I.PB(x);
    x = (-b-d)/a; if (0 < x && x < T) I.PB(x);
}
#undef P

int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    while (~scanf(”%d%d”, &n, &T)){
        REP(i, n) P0[i].in(), RF(A[i], B[i]);

        vector<DB> I; I.PB(0), I.PB(T);

        REP(i, n) FOR(j, i+1, n) FOR(k, j+1, n){
            add(I, i, j, k);
        }

        _sl = s(0); UNQ(I);

        //REP(i,SZ(I)) cout << I[i] << ” ”; cout << endl;

        DB res = 0; FOR(i, 1, SZ(I)){
            res += s(I[i-1], I[i]);
        }

        OT(res/T/2);
    }

}
```

Chapter 24

半平面交

```
const int HPI_N = 109;

bool cmpHPI(cLine l,cLine r){
    int t = sgn(l.arg(), r.arg()); if (!t) t = dett(r.a,l);
    return t < 0;
}

Line Q[HPI_N]; int cz, op;

void cut_b(cLine l){while(cz<op&&dett(l,Q[op]*Q[op-1])<0)--op;}
void cut_f(cLine l){while(cz<op&&dett(l,Q[cz]*Q[cz+1])<0)++cz;}
void cut(cLine l){cut_b(l),cut_f(l),Q[++op]=l;}

VP getHPI(vector<Line>&L){
    SRT(L, cmpHPI); int n = 1; FOR(i, 1, SZ(L)) if (sgn(L[i-1].arg(), L[i].arg())) L[n++] = L[i];
    VP P; cz = 0, op = 1, Q[0] = L[0], Q[1] = L[1]; FOR(i, 2, n){
        if (!dett(Q[op],Q[op-1])||!dett(Q[cz],Q[cz+1])) return P;
        cut(L[i]);
    }
    cut_b(Q[cz]);cut_f(Q[op]);

    if (op <= cz+1) return P;
    for (int i=cz;i<op;++i) P.PB(Q[i]*Q[i+1]);
    if (cz<op+1) P.PB(Q[cz]*Q[op]);
    UNQQ(P).PB(P[0]);
    return P;
}
```

Chapter 25

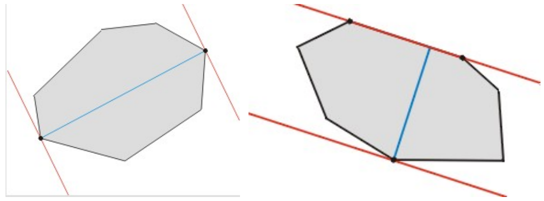
旋转卡壳

```
#define suc(x) (x+1==n?0:x+1)
```

为了减小精度误差，一般情况下我们返回所求距离（长度）的平方。

25.1 计算距离（Computing distances）

25.1.1 凸包的直径与宽度



```
DB rc(const VP& P){
    int n = SZ(P)-1, j = 1; DB d2 = 0; REP(i, n){
        while (dett(P[i+1]-P[i], P[j+1]-P[j])>0) j=suc(j);
        checkMax(d2, max(dist2(P[i], P[j]), dist2(P[i+1], P[j+1])));
    }
    return d2;
}
```

```
DB rc(const VP& P){
    int n = SZ(P)-1, j = 1; DB w2 = OO; REP(i, n){
        while (dett(P[i+1]-P[i], P[j+1]-P[j])>0) j=suc(j);
        checkMin(w2, dist2(Line(P[i], P[i+1]), P[j]));
    }
    return w2;
}
```

25.1.2 两个凸包间的距离（Distance between 2 convex polygons）

给定两组不相交的凸多边形 P、Q，我们的目标是最小化两对点之间的距离 $dist(p,q)$，满足 $p \in P$，$q \in Q$。
通常我们所说的凸包间距离指的是上述问题，
同时该问题也存在着诸多派生。例如最大化凸包间的距离，例如限定点只能处在凸包的顶点上（vertex distance），注意在后面这个派生中，P, Q 即使相交，问题仍然是有意义的。

```
DB rc(const VP& P1, const VP& P2){
    int n = SZ(P1)-1, m = SZ(P2)-1;
    int i=0, j=0; DB d2=OO;

    REP(k, n) if (P1[k].y > P1[i].y) i = k; //#
    REP(k, m) if (P2[k].y < P2[j].y) j = k;

    DO(n){
        Seg h(P1[i], P1[i+1]); while (dett(h.d(), P2[j+1]-P2[j])>0) j=suc(j,m);
        checkMin(d2, dist2(h, Seg(P2[j],P2[j+1])));
        i=suc(i,n);
    }
    return d2;
}
```

25.2 外接矩形（Enclosing rectangles）

25.2.1 最小外接矩形的面积与周长

```
#define suc(x) (x+1==n?0:x+1)
DB rc(const VP& P){

    int n=SZ(P)-1,l=1,r=1,u=1,ll=1,rr=1,uu=1; DB res=OO; REP(i, n){

        Line p(P[i], P[i+1]); p.b = p.a + p.d()._1();

        while (dott(p.d(), P[r+1]-P[r])>0) r=suc(r),++rr; if (uu<rr)u=r,uu=rr; //#
        while (dett(p.d(), P[u+1]-P[u])>0) u=suc(u),++uu; if (ll<uu)l=u,ll=uu;
        while (dott(p.d(), P[l+1]-P[l])<0) l=suc(l),++ll;

        DB w = //dist(Line(P[r], P[r]+p.d().lt()), Line(P[l], P[l]+p.d().lt())); //?
            dot(p, P[r]) - dot(p, P[l]);
        DB h = dist(p, P[u]);
        checkMin(res, w*h);
        //checkMin(res, w+h)
    }
    return res;
}
```

25.3 三角分解（Triangulations）

25.4 凸多边形性质（Properties of convex polygons）

25.5 例题 (E.g.)

25.5.1 HDU 3847. Trash Removal

简述 (Brief description)

略。）

分析 (Analysis)

凸包宽度。

Chapter 26

最近点对

```
const int N = int(1e5) + 9;
VP P; int n;

bool cmpy(cPo a, cPo b){return a.y < b.y;}

inline DB cp(VP &P, int l, int r){
    if (l >= r) return OO;

    int m = (l + r) >> 1; DB d = min(cp(P, l, m), cp(P, m+1, r)), mx = P[m].x;
    inplace_merge(P.begin()+l, P.begin()+m+1, P.begin()+r+1, cmpy);

    VP t; FOR_1(i, l, r) if (sgn(abs(P[i].x - mx), d)<0) t.PB(P[i]);
    REP(i, SZ(t)) FOR(j, i+1, min(SZ(t), i+9)) checkMin(d, dist2(t[i], t[j])); //#
    return d;
}

DB cp(VP& P){
    SRT(P); return cp(P, 0, n-1);
}

int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    DB pre = 0; REP_C(i, RD(n)) P.PB(Po(i, pre += RDD()));
    printf(”%.0f\n” , cp(P)+EPS);
}
```

Chapter 27

3D-几何基础

```

inline DB dist2(DB x,DB y,DB z){return dist2(x,y)+sqr(z);}

namespace D3{
    struct Po{
        DB x,y,z;Po(DB x=0,DB y=0,DB z=0):x(x),y(y),z(z){}
        void in(){RF(x,y,z);}

        Po operator-()const{return Po(-x,-y,-z);}
        Po&operator+=(cPo p){x+=p.x,y+=p.y,z+=p.z;rTs;}Po&operator-=(cPo p){x-=p.x,y-=p.y,z-=p.z;rTs;}
        Po&operator*=(DB k){x*=k,y*=k,z*=k;rTs;}Po&operator/=(DB k){x/=k,y/=k,z/=k;rTs;}
        Po operator+(cPo p)const{return Po(x+p.x,y+p.y,z+p.z);}Po operator-(cPo p)const{return Po(x-p.x,y-p.y,z-p.z);}
        Po operator*(DB k)const{return Po(x*k,y*k,z*k);}Po operator/(DB k)const{return Po(x/k,y/k,z/k);}

        DB len2()const{return dist2(x,y,z);}DB len()const{return sqrt(len2());}
        Po&_1(){rTs/=len();}
    };

    inline DB dot(DB x1,DB y1,DB z1,DB x2,DB y2,DB z2){return CG::dot(x1,y1,x2,y2)+z1*z2;}
    inline DB dot(cPo a,cPo b){return dot(a.x,a.y,a.z,b.x,b.y,b.z);}
    inline DB dot(cPo p0,cPo p1,cPo p2){return dot(p1-p0,p2-p0);}
    inline Po det(DB x1,DB y1,DB z1,DB x2,DB y2,DB z2){return Po(CG::det(y1,z1,y2,z2),CG::det(z1,x1,z2,x2),CG::det(x1,y1,x2,y2));}
    inline Po det(cPo a,cPo b){return det(a.x,a.y,a.z,b.x,b.y,b.z);}
    inline Po det(cPo p0,cPo p1,cPo p2){return det(p1-p0,p2-p0);}

    struct Line{
        Po a,b;
    };
};
```

Chapter 28

海岸线

Part VIII

补充 (More)

Chapter 29

倍增祖先

29.1 例题 (E.g.)

29.1.1 CC...

题目描述 (Brief description)

维护一颗有根树，不断添加叶子，询问直径的长度。

算法分析 (Algorithm Analysis)

。。。

```

const int N = int(1e5) + 9, LV = 20;

int dep[N], fa[LV][N];
int n;

int up(int x, int d){
    REP(lv, LV){
        if (d&1) x = fa[lv][x];
        d >>= 1;
    }
    return x;
}

int lca(int x, int y){
    if (dep[x] > dep[y]) x = up(x, dep[x]-dep[y]); else y = up(y, dep[y]-dep[x]);
    if (x == y) return x;
    else{
        DWN(lv, LV, 0) if (fa[lv][x] != fa[lv][y])
            x = fa[lv][x], y = fa[lv][y];
        return fa[0][x];
    }
}

int dist(int x, int y){
    return dep[x] + dep[y] - dep[lca(x, y)]*2;
}

int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    Rush{

        dep[1] = 0; FOR_1_C(i, 2, RD(n)) dep[i] = dep[RD(fa[0][i])] + 1;
        if (n <= 1) continue;

        REP_1(i, n) FOR(lv, 1, LV) fa[lv][i] = fa[lv-1][fa[lv-1][i]];

        int p1 = 1, p2 = 1, di = 0; FOR_1(i, 2, n){
            int l1 = dist(i, p1), l2 = dist(i, p2);
            if (l1 > di) di = l1, p2 = i;
            if (l2 > di) di = l2, p1 = i;
            OT(di);
        }
    }
}
```

Chapter 30

树链剖分

30.1 例题 (E.g.)

30.1.1 BZOJ 3083. 遥远的国度

题目描述 (Brief description)

动态维护一棵点权有根树，支持以下操作：

- 1 u: 换根。
- 2 x y v: 将 x->y 的路径上的点权全部修改为 v
- 3 u: 询问 u 为根的子树内的最小值。

算法分析 (Algorithm Analysis)

树链剖分后，一段路径在 DFS() 序列中被分割成不超过 $log(n)$ 段区间，只要先遍历重链所在方向即可。
考虑换根，查询 u 的时候，如果当前的根...

- 在 u 的上方 ⇒ 不变
- 就是 u 本身 ⇒ 整个树
- 在 u 的下方 ⇒ u 在根方向的孩子所在的区间的补。

```
const int N = 100009, M = 2 * N, LV = 18;
UINT A[N]; int L[N], R[N], sz[N], up[N], dep[N], fa[LV][N], n, m, nn, rt;
int hd[N], suc[M], to[M]; UINT T[4*N]; int bj[4*N], a, b; UINT c;

inline int move_up(int x, int t){
    REP(lv, LV){
        if (t&1) x = fa[lv][x];
        t >>= 1;
    }
    return x;
}

inline int lca(int a, int b){
    if (dep[a] > dep[b]) a = move_up(a, dep[a] - dep[b]);
    else b = move_up(b, dep[b] - dep[a]);

    if (a == b) return a;
    else {
        DWN(lv, LV, 0) if (fa[lv][a] != fa[lv][b])
            a = fa[lv][a], b = fa[lv][b];
        return fa[0][a];
    }
}

#define aa to[i^1]
```

```
#define bb to[i]
#define v bb

inline void dfs(int u = 1){
    sz[u] = 1; REP_G(i, u) if (v != fa[0][u]){
        dep[v] = dep[u] + 1, fa[0][v] = u;
        FOR(lv, 1, LV) if (!(fa[lv][v] = fa[lv-1][fa[lv-1][v]])) break;
        dfs(v), sz[u] += sz[v];
    }
}

inline void hld(int u = 1, int t = 1){
    L[u] = ++nn, up[u] = t;
    int h = 0; REP_G(i, u) if (v != fa[0][u] && sz[v] > sz[h])
        h = v;

    if (h){
        hld(h, t); REP_G(i, u) if (v != fa[0][u] && v != h)
            hld(v, v);
    }
    R[u] = nn;
}

#define root 1, 1, n
#define lx (x<<1)
#define rx (lx|1)
#define ml (l + r >> 1)
#define mr (ml + 1)
#define xx x, l, r
#define lc lx, l, ml
#define rc rx, mr, r

inline void Update(int x, int l, int r){
    T[x] = min(T[lx], T[rx]);
}

inline void Release(int x){
    if (bj[x]){
        T[lx] = T[rx] = T[x];
        bj[lx] = bj[rx] = 1;
        bj[x] = 0;
    }
}

inline void Build(int x, int l, int r){
    if (l == r){
        T[x] = A[l];
    }
    else {
        Build(lc), Build(rc);
        Update(xx);
    }
}

inline UINT Query(int x, int l, int r){
    if (r < a || b < l) return -1;
    if (a <= l && r <= b) return T[x];
    Release(x);
    return min(Query(lc), Query(rc));
}

inline void Modify(int x, int l, int r){
    if (r < a || b < l) return;
    if (a <= l && r <= b) T[x] = c, bj[x] = 1;
    else {
        Release(x);
        Modify(lc), Modify(rc);
        Update(xx);
    }
}

inline void Modifyy(int z, int x){
    while (up[z] != up[x]){
        a = L[up[x]], b = L[x], Modify(root);
        x = fa[0][up[x]];
    }
    a = L[z], b = L[x], Modify(root);
}

inline UINT Queryy(int l, int r){
    a = l, b = r;
    return Query(root);
}

int main(){

#ifndef ONLINE_JUDGE
    freopen("in.txt", "r", stdin);
    //freopen("out.txt", "w", stdout);
#endif

    RD(n, m); FOR_C(i, 2, n << 1){
        RD(aa, bb);
        suc[i] = hd[aa], hd[aa] = i, ++i;
        suc[i] = hd[aa], hd[aa] = i;
    }

    dfs(), hld(); REP_1(i, n) RD(A[L[i]]);
    RD(rt); Build(root); DO(m){
        int x, y, z; switch(RD()){
            case 1:
                RD(rt);
                break;
            case 2:
                RD(x, y, c); z = lca(x, y);
                Modifyy(z, x), Modifyy(z, y);
                break;
            case 3:
                if (rt == RD(x)){
                    OT(Queryy(1, n));
                }
                else if (L[x] <= L[rt] && L[rt] <= R[x]){
                    x = move_up(rt, (dep[rt] - dep[x] - 1));
                    OT(min(Queryy(1, L[x]-1), Queryy(R[x]+1, n)));
                }
                else {
                    OT(Queryy(L[x], R[x]));
                }
        }
    }
}
```

Chapter 31

一类算法的复合方法

31.1 例题 (E.g.)

31.1.1 SPOJ RECTANGLE

题目描述 (Brief description)

在一个平面上一组点集，问这个点集中可以组成多少矩形（边与坐标轴平行）。

算法分析 (Algorithm analysis)

对 “横’’、“纵” 形态的数据，各设计一个 $O(n2)$ 算法。合并起来得到一个 $O(n^{1.5})$ 的算法。

```
const int N = int(2.5e5) + 9;
PII P[N]; int PP[N];
int n, nn; int Q; LL ans;

inline bool cmp(int a, int b){
    return P[a].se < P[b].se || P[a].se == P[b].se && P[a].fi < P[b].fi;
}

inline LL C2(LL n){
    return n*(n-1)/2;
}

int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    Q = sqrt(RD(n)); REP(i, n) RDD(P[i].fi, P[i].se); P[n].fi = P[n].se = INF; sort(P, P+n);

    for (int i=0,ii;i<n;i=ii){

        ii = i+1; while (P[i].fi==P[ii].fi)++ii;

        if (ii-i>Q){
            set<int> H; int s=0; FOR(j, i, ii) H.insert(P[j].se);

            for(int j=ii;j<n;++j){
                if (CTN(H, P[j].se)) ++s;
                if (P[j].fi!=P[j+1].fi) ans += C2(s), s = 0;
            }
            for(int j=0;j<nn;++j){
                if (CTN(H, P[j].se)) ++s;
                if (P[PP[j]].fi!=P[PP[j+1]].fi) ans += C2(s), s = 0;
            }
        }
        else{
            FOR(j, i, ii) PP[nn++] = j; PP[nn] = n;
        }
    }

    sort(PP, PP+nn, cmp);

    for (int i=0,ii;i<nn;i=ii){
        ii = i+1; while(P[PP[i]].se==P[PP[ii]].se)++ii;
        map<int, int> H; FOR(j, i, ii){
            int t = P[PP[j]].fi;
            for(int jj=PP[j]-1;jj>=0&&P[jj].fi==t;--jj) ++H[P[jj].se];
        }
        ECH(it, H) ans += C2(it->se);
    }

    OT(ans);
}
```

Chapter 32

一类树上的构造问题

Chapter 33

培养皿问题

33.1 例题 (E.g.)

33.1.1 SGU 187

题目描述 (Brief description)

给一个 $n \times m$ 矩形，其中有些格子可以选，有些不能选。现在要求在可选的格子中选一些组成一个凸物，凸物要求所选的所有格子是相互联通的，并且如果同一行 (列) 选取了两个格子的话，和它们在同一行 (列)，并且在它们之间的所有格子都需要被选。问一共有多少种不同的选法。（$n, m \leq 100$）

算法分析 (Algorithm Analysis)

我们需要好好研究下这个所谓凸物的性质。首先，它是个联通体，然后，由于凸性，其每一行/每一列一定是连续的一段。想想一下这个图形，它的每一行我们都可以用一个三元组来表示 (r, a, b)，表示第 r 行选取的是从第 a 列到第 b 列的所有元素。我们选取的一定是连续的一些行，我们考察这些行各自的 a 和 b 所组成的 A 和 B 序列的性质。

A 数列是每一行最左边元素的列坐标组成的数组，B 则是最右边元素的列坐标数组。为了保证每一列也是连续的一段，A 数组必须是一个先非递增，再非递减的数组，而 B 数组也类似。根据这一点，一个唯一的凸物对应一对唯一的 A 和 B 数组。因此我们就可以来考虑 dp 解法来选取 A 和 B 数组了。dp[i][j][k][b1][b2] 表示最后一行为第 i 行的，第 i 行选取第 j 到第 k 列的方法数（b1 和 b2 用来表示 A 序列和 B 数列分别是否已经在非递减/非递增状态）。对于第 i 行，一种情况是整个凸物只有第 i 行的元素，这个是很 naive 的情况；另外一种是除了第 i 行，还有前 i-1 行的一些元素，这样我们就可以利用到 dp[i-1][?][?][?][?] 的值了。具体状态转移要根据后两维的不同来分别处理。实际上，在求 dp[i] 时，我们需要求的都是某一段 dp[i-1][j1 to j2][k1 to k2][?][?] 所有元素的和，这是一个很经典的二维数组求子矩阵和问题，可以 $O(n^2)$ 处理，之后 O(1) 计算。状态转移的时候还需要特别注意到，相等序列即符合非递增又符合非递减，所以 A 和 B 数组出现转折点的情况一定是出现了一个严格递增/严格递减的状态，这样可以保证每个状态被唯一表示，避免了重复计算。

dp[0/1][0/1][l][r]: 表示左右的增减状态为 b1, b2 ，当前区间为 [l, r] 时的状态。。。。状态的时候先从上一轮预处理二维部分和数组。。之后分四种情况讨论即可。。。Petr 的代码通过调整转移方向。。可以避免这步操作。。很优越。。)。。

。。实现的时候写了一个 Int 整数类。。自带取模。。代码尽量保持对称可避免敲错。。。。

```
const int N = 109;

struct Int{
    int val;

    operator int() const{return val;}

    Int(int val = 0):val(val){
        val %= MOD; if (val < 0) val += MOD;
    }
    inline Int& operator +=(const Int& rhs){
        INC(val, rhs);
        return *this;
    }
    inline Int operator +(const Int& rhs) const{
        return sum(val, rhs.val);
    }
    inline Int operator -(const Int& rhs) const{
        return dff(val, rhs);
    }
};

Int F[2][2][N][N], S[2][2][N][N]; bool A[N][N];
int n, m;

Int SS(int b1, int b2, int x1, int x2, int y1, int y2){
    return S[b1][b2][x2+1][y2+1] - S[b1][b2][x2+1][y1] - S[b1][b2][x1][y2+1] + S[b1][b2][x1][y1];
}

class AmoebaDivOne {
public:
    int count(vector <string> T) {

        n = SZ(T), m = SZ(T[0]); REP_2(i, j, n, m){
            int t = isdigit(T[i][j]) ? T[i][j] - '0' : T[i][j] - 'a' + 10;
            A[2*i][2*j] = t & 1, A[2*i][2*j+1] = t & 2;
            A[2*i+1][2*j] = t & 4, A[2*i+1][2*j+1] = t & 8;
        }

        n <<= 1, m <<= 1; RST(F); Int res; REP(i, n){

            RST(S); REP_4(b1, b2, l, r, 2, 2, m, m) S[b1][b2][l+1][r+1] = S[b1][b2][l][r+1] + S[b1][b2][l+1][r] - S[b1][b2][l][r] + F[b1][b2][l][r];

            RST(F); REP(l, m) FOR(r, l, m){

                if (A[i][r]) break;

                F[0][0][l][r] = SS(0, 0, l, r, l, r) + Int(1);
                F[0][1][l][r] = SS(0, 0, l, r, r+1, m-1) + SS(0, 1, l, r, r, m-1);
                F[1][0][l][r] = SS(0, 0, 0, l-1, l, r) + SS(1, 0, 0, l, l, r);
                F[1][1][l][r] = SS(0, 0, 0, l-1, r+1, m-1) + SS(0, 1, 0, l-1, r, m-1) + SS(1, 0, 0, l, r+1, m-1) + SS(1, 1, 0, l, r, m-1);

                REP_2(b1, b2, 2, 2) res += F[b1][b2][l][r];
            }
        }

        return res;
    }
};
```

33.1.2 SGU 187

题目描述 (Brief description)

最值问题。。需要打印方案。。

算法分析 (Algorithm Analysis)

```
const int N = 15;

struct rec{
    short b1, b2, l, r;
    rec(){}
    rec(int b1, int b2, int l, int r):b1(b1),b2(b2),l(l),r(r){}
    int len(){return r-l+1;}
    void output(int i){
        FOR_1(j, l, r) printf("%d %d\n", i, j+1);
    }
};

int dp[N+1][N*N+1][2][2][N][N]; rec pr[N+1][N*N+1][2][2][N][N];
int S[N+1], n, m, k;

```

```
void upd(int i, int k, int b1, int b2, int bb1, int bb2, int l, int r, int ll, int rr){
    if (dp[i][k-(r-l+1)][bb1][bb2][ll][rr] + S[r+1] - S[l] > dp[i+1][k][b1][b2][l][r]){
        dp[i+1][k][b1][b2][l][r] = dp[i][k-(r-l+1)][bb1][bb2][ll][rr] + S[r+1] - S[l];
        pr[i+1][k][b1][b2][l][r] = rec(bb1, bb2, ll, rr);
    }
}

void gao(int i, int k, rec& p){
    if (!k) return; p.output(i);
    gao(i-1, k-p.len(), pr[i][k][p.b1][p.b2][p.l][p.r]);
}

int main() {

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    RD(n, m, k); REP(i, n){

        REP(j, m) S[j+1] = S[j] + RD();

        REP_1(_k, k) REP(l, m) FOR(r, l, min(m, l+_k)){

            FOR_1(ll, l, r) FOR_1(rr, ll, r)
                upd(i, _k, 0, 0, 0, 0, l, r, ll, rr);

            FOR_1(ll, l, r) FOR(rr, r+1, m)
                upd(i, _k, 0, 1, 0, 0, l, r, ll, rr);
            FOR_1(ll, l, r) FOR(rr, r, m)
                upd(i, _k, 0, 1, 0, 1, l, r, ll, rr);

            FOR(ll, 0, l) FOR_1(rr, l, r)
                upd(i, _k, 1, 0, 0, 0, l, r, ll, rr);
            FOR_1(ll, 0, l) FOR_1(rr, l, r)
                upd(i, _k, 1, 0, 1, 0, l, r, ll, rr);

            FOR(ll, 0, l) FOR(rr, r+1, m)
                upd(i, _k, 1, 1, 0, 0, l, r, ll, rr);
            FOR(ll, 0, l) FOR(rr, r, m)
                upd(i, _k, 1, 1, 0, 1, l, r, ll, rr);
            FOR_1(ll, 0, l) FOR(rr, r+1, m)
                upd(i, _k, 1, 1, 1, 0, l, r, ll, rr);
            FOR_1(ll, 0, l) FOR(rr, r, m)
                upd(i, _k, 1, 1, 1, 1, l, r, ll, rr);

        }
    }

    int res = 0, ii; rec pp; REP_1(i, n) REP_4(b1, b2, r, l, 2, 2, m, r+1) if (dp[i][k][b1][b2][l][r] >= res){
        res = dp[i][k][b1][b2][l][r];
        ii = i, pp = rec(b1, b2, l, r);
    }

    cout << ”Oil : ” << res << endl;
    gao(ii, k, pp);
}
```

Chapter 34

临时

34.1 例题 (E.g.)

34.1.1 ABBYY Cup 3.0 G3. Good Substrings

简述 (Brief description)

给定一段 Text、以及 nn 个 Pattern。要求这个 Text 中合法的子串数目，使得对于第 ii 个 Pattern，恰好能够匹配 $[l_i i, r_i i]$ 之间次。（$nn \leq 10$）

分析 (Analysis)

SAM-DP。

我们把涉及到的所有字符串（Text && Patterns），依次插入到 SAM。（相邻的字符串之间，用彼此不同的分隔符隔开，以保证各字符串之间在 SAM 中不会相互干扰）

dp[ii][u] 表示：结点 u 所表示的子串集合，对于第 ii 个字符串的匹配次数。

```
namespace SAM{

    const int SN = int(5e4) + 1, NN = 11, N = 2*NN*SN + 9, Z = 26;

    int trans[N][Z+NN], fail[N], len[N], tail, tot; char str[SN];

    inline int new_node(){
        // RST(trans[tot);
        tail = tot;
        return tot++;
    }
    inline int new_node(int u){
        CPY(trans[tot], trans[u]), fail[tot] = fail[u];
        return tot++;
    }

#define v trans[u]1
#define f fail[u]
#define ff fail[uu]

    inline int h(int u){
        return len[u] - len[f];
    }

    void Ext(int c){
        int u = tail, uu = new_node(); len[uu] = len[u] + 1;
        while (u && !v) v = uu, u = f;
        if (!u && !v) v = uu, ff = 0;
        else{
            if (len[v] == len[u] + 1) ff = v;
            else{
                int _v = v, vv = new_node(_v); len[vv] = len[u] + 1;
                fail[_v] = ff = vv;
                while (v == _v) v = vv, u = f;
            }
        }
    }

    int dp[NN][N], l[NN], r[NN]; bool vis[N];
    int nn, ans;
#define c (*cur - ’a’)
    void Init(){
        tot = 0, new_node();
        gets(str); REP_S(cur, str) Ext(c); Ext(Z);
        REP_1_C(ii, RD(nn)){
            RS(str); RD(l[ii], r[ii]);
            REP_S(cur, str) Ext(c); Ext(Z+ii);
        }
    }
#undef c

    inline bool legal(int u){
        if (!u || !dp[0][u]) return 0;
        REP_1(ii, nn) if (dp[ii][u] < l[ii] || r[ii] < dp[ii][u]) return 0;
        return 1;
    }

    void dfs(int u = 0){
        if (vis[u]) return; vis[u] = 1;

        REP(ii, nn+1) if (trans[u][Z+ii]) dp[ii][u] = 1;

        REP(c, Z) if (v){
            dfs(v); REP(ii, nn+1) dp[ii][u] += dp[ii][v];
        }

        if (legal(u)) ans += h(u);
    }

} using namespace SAM;

int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    Init(); dfs(); OT(ans);
}
```

34.1.2 BZOJ 2806.

34.1.3 BZOJ 2806. [CTSC2012 Day2 T1] 熟悉的文章 (cheat)

简述 (Brief description)

给定一组句子，一个单词是可识别的，如果其是某个句子的子串。我们认为一篇文章是熟悉的，如果存在对该文章的一种分词，使得其中被识别出的单词的总长度，$\geq$ 文章总长度的 90%。

给定一篇文章，判断其是否是熟悉的，如果是，给出令最长的单词最短的分词方案。

分析 (Analysis)

二分答案，SAM-DP。

```

namespace SAM{

    const int N = int(2.2e6) + 9, Z = 3;
    int trans[N][Z], fail[N], len[N], tail, tot;
    char str[N/2];

    inline int new_node(){
        RST(trans[tot]), tail = tot;
        return tot++;
    }

    inline int new_node(int u){
        CPY(trans[tot], trans[u]), fail[tot] = fail[u];
        return tot++;
    }

#define v trans[u][c]
#define f fail[u]
#define ff fail[uu]

    void Ext(int c){
        int u = tail, uu = new_node(); len[uu] = len[u] + 1;
        while (u && !v) v = uu, u = f;
        if (!u && !v) v = uu, ff = 0;
        else{
            if (len[v] == len[u] + 1) ff = v;
            else{
                int _v = v, vv = new_node(_v); len[vv] = len[u] + 1;
                fail[_v] = ff = vv;
                while (v == _v) v = vv, u = f;
            }
        }
    }

    void Init(){
        tot = 0, tail = new_node();
    }

    int ll[N/2], n, n0;

    void Spell(){
        int u = 0, l = 0; REP_1(i, n){
            int c = str[i] - ’0’;
            while (u && !v) l = len[u = f];
            if (u = v) ++l; ll[i] = i - l;
        }
    }

    #undef f

    int f[N/2], g[N/2], q[N/2], cz, op, tt;

    bool check(int x){

        cz = 1, op = 0; REP_1(i, n){

            if (~(tt=i-x)){
                while (cz <= op && op[q][g] < g[tt]) --op;
                q[++op] = tt;
            }

            while (cz <= op && q[cz] < ll[i]) ++cz;

            f[i] = max(f[i-1], cz <= op ? cz[q][g] + i : 0);
            g[i] = f[i] - i;
        }

        return f[n] >= n0;
    }

```

```
} using namespace SAM;

int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    int Q, nn; RD(Q, nn); Init(); while (nn--){
        RS(str); REP_S(cur, str) Ext(*cur - ’0’);
        if (nn) Ext(Z-1);
    }

    DO(Q){

        n = strlen(RS(str+1)), n0 = ceil(n*0.9 - EPS); Spell();

        int l = 0, r = n; while (l < r){
            int m = l + r + 1 >> 1;
            if (check(m)) l = m; else r = m - 1;
        }

        OT(l);
    }
}
```

34.1.4 圆桌骑士

简述 (Brief description)

分析 (Analysis)

```

VP getCH(VP& P){ //逆时针，无共线

    int n=SZ(P); if(n<=3) return P.PB(P[0]),getArea(P)<0?RVS(P):P;

    SRT(P); VP C; C.resize(n+9); int nn = -1; REP(i, n){ //#
        while (nn > 0 && dett(C[nn-1], C[nn], P[i]) <= 0) --nn; //#
        C[++nn] = P[i];
    }

    int _nn = nn; DWN(i, n-1, 0){
        while (nn > _nn && dett(C[nn-1], C[nn], P[i]) <= 0) --nn; //#
        C[++nn] = P[i];
    }

    C.resize(nn+1);
    return C;
}

。要卡壳。。先凸包


。。。学习一般多边形直径受到巨大打击。。。。决定以退为进。。先学习下旋转卡壳。。


。。以下全程逆时针。。。各种约定遵守凸包算法。。。
。。三张图解释旋转卡壳的本质！。。
http://cgm.cs.mcgill.ca/~orm/app.html


http://acm.hust.edu.cn/vjudge/problem/viewProblem.action?id=15777
凸包直径。。
```

```
#define suc(x) (x+1==n?0:x+1)
DB rc(const VP& P){
    int n = SZ(P)-1, j = 1; DB d2 = 0; REP(i, n){
        while (dett(P[i+1]-P[i], P[j+1]-P[j])>0) j=suc(j);
        checkMax(d2, max(dist2(P[i], P[j]), dist2(P[i+1], P[j+1])));
    }
    return d2;
}
```

凸包宽度。。

```
#define suc(x) (x+1==n?0:x+1)
DB rc(const VP& P){
    int n = SZ(P)-1, j = 1; DB w2 = OO; REP(i, n){
        while (dett(P[i+1]-P[i], P[j+1]-P[j])>0) j=suc(j);
        checkMin(w2, dist2(Line(P[i], P[i+1]), P[j]));
    }
    return w2;
}
```

（！未找到题目测试。。。）

http://acm.hust.edu.cn/vjudge/problem/viewSource.action?id=1554076

。。两个凸包间的距离。。

```
#define suc(x, n) (x+1==n?0:x+1)
DB _rc(const VP& P1, const VP& P2){
    int n = SZ(P1)-1, m = SZ(P2)-1;
    int i=0, j=0; DB d2=OO;

    REP(k, n) if (P1[k].y > P1[i].y) i = k; //#
    REP(k, m) if (P2[k].y < P2[i].y) i = k;

    DO(n){
        Seg h(P1[i], P1[i+1]); while (dett(h.d(), P2[j+1]-P2[j])>0) j=suc(j,m);
        checkMin(d2, dist2(h, Seg(P2[j],P2[j+1])));
        i=suc(i,n);
    }
    return d2;
}
DB rc(const VP& P1, const VP& P2){
    return min(_rc(P1, P2), _rc(P2, P1));
}
```

（Another Method。。两边同时旋转。http://blog.csdn.net/zxy_snow/article/details/6540150

http://acm.hust.edu.cn/vjudge/problem/viewProblem.action?id=20288

。。。

。。最小覆盖矩形。。。

。。。。枚举其中一条边转。。其它三个边跟着转。。。

有一些恶心的边界情况要考虑。。

```
#define suc(x) (x+1==n?0:x+1)
DB rc(const VP& P){

    int n=SZ(P)-1,l=1,r=1,u=1,ll=1,rr=1,uu=1; DB res=OO; REP(i, n){

        Line p(P[i], P[i+1]); p.b = p.a + p.d()._1();

        while (dott(p.d(), P[r+1]-P[r])>0) r=suc(r),++rr; if (uu<rr)u=r,uu=rr; //#
        while (dett(p.d(), P[u+1]-P[u])>0) u=suc(u),++uu; if (ll<uu)l=u,ll=uu;
        while (dott(p.d(), P[l+1]-P[l])<0) l=suc(l),++ll;

        DB w = //dist(Line(P[r], P[r]+p.d().lt()), Line(P[l], P[l]+p.d().lt())); //?
            dot(p, P[r]) - dot(p, P[l]);
        DB h = dist(p, P[u]);
        //cout << w << ” ” << h << endl;
        checkMin(res, w*h);
    }
    //cout << res << endl;

    return res;
}


http://acm.hust.edu.cn/vjudge/problem/viewSource.action?id=1554091
http://acm.hust.edu.cn/vjudge/problem/viewSource.action?id=1555853
。。凸包内面积最大三角。（复杂度整个增加一维。。。似乎有优化的余地。。
#define suc(x) (x+1==n?0:x+1)
DB rc(const VP& P){
    int n = SZ(P)-1; DB res = 0; int j, k; REP(i, n){
        for (j=k=suc(i);j!=i;j=suc(j)){
            while (dett(P[j]-P[i], P[k+1]-P[k])>0) k=suc(k);
            checkMax(res, fabs(det(P[i], P[j], P[k])));
        }
    }
    return res/2;
}



        //}/* .............................................................................................................................. */

const int N = 1000009, M = 2 * N, LM = 21;
int hd[N], suc[M], to[M], wt[N];

int ST[LM][M], D[N], st[N], dep[N]; // Euler index ...
int n, tt;

inline bool elder(int a, int b){
    return dep[a] < dep[b];
}

inline int lca(int a, int b){
    int l = st[a], r = st[b];
    if (l > r) swap(l, r); ++r; int lv = lg2(r-l); //log2(r - l);
    return min(ST[lv][l], ST[lv][r-(1<<lv)], elder);
}

#define aa to[i^1]
#define bb to[i]
#define v bb
#define w wt[i/2]

void dfs(int u = 1){
    ST[0][st[u] = ++tt] = u;
    REP_G(i, u) if (!st[v]){
        dep[v] = dep[u] + 1, D[v] = D[u] + w; dfs(v);
        ST[0][++tt] = u;
    }
}

#undef v

const int MM = 1009;

struct Ants{
    int s, t, r; LL v;
    void in(){
        RD(s, t, v);
        r = lca(s, t);
    }
    void out(){
        cout << s << ” ” << t << ” ” << r <<endl;
    }

    int entry(int x){
        if (lca(r, x) != r) return r;
        int tt = lca(s, x); if (tt != r) return tt;
        return lca(t, x);
    }

    bool sgn(int x){ // 判断点是否在路径上。
        return lca(x, r) == r && (lca(s, x) == x || lca(t, x) == x);
    }

} ants[MM]; int m;

int d(int u, int v){
    return D[u] + D[v] - D[lca(u, v)]*2;
}

int check(Ants& a, Ants& b){

    int bg = a.entry(b.s); if (!b.sgn(bg)) return 0;
    int ed = a.entry(b.t); if (!b.sgn(ed)) return 0;

    bool b1 = 0, b2 = 0;

    /*DB all = (DB)d(a.s, bg) / a.v, arr = (DB)d(a.s, ed) / a.v; if (all > arr) swap(all, arr), b1 = 1;
    DB bll = (DB)d(b.s, bg) / b.v, brr = (DB)d(b.s, ed) / b.v; if (bll > brr) swap(bll, brr), b2 = 1;
    if (!sgn(all, bll)) return 1;
    if (all > bll) swap(all, bll), swap(arr, brr);
    return sgn(b1==b2?brr:bll, arr) <= 0;
    // 使用浮点数的话。。10-8 全挂。。10-9 260 分。。10-10 280 分。。之后 300 分。。。
    */

    LL al = d(a.s, bg), ar = d(a.s, ed); if (al > ar) swap(al, ar), b1 = 1;
    LL bl = d(b.s, bg), br = d(b.s, ed); if (bl > br) swap(bl, br), b2 = 1;
    LL av = a.v, bv = b.v;

    if (al*bv == bl*av) return 1;
    if (al*bv > bl*av) swap(al, bl), swap(ar, br), swap(av, bv);
    return (b1==b2?br:bl)*av <= ar*bv;
}


int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    Rush{

        FOR_C(i, 2, RD(n)<<1){
            RD(aa, bb, w);
            suc[i] = hd[aa], hd[aa] = i, ++i;
            suc[i] = hd[aa], hd[aa] = i;
        }

        dfs();

        for ( int lv = 1 ; (1 << lv) <= tt ; lv ++ ){
            for ( int i = 1 ; i + (1 << lv) <= tt + 1 ; i ++ )
                ST[lv][i] = min(ST[lv-1][i], ST[lv-1][i + (1<<(lv-1))], elder);

        }

        REP_C(i, RD(m)) ants[i].in();

        int res = 0; REP_2(j, i, m, j){
            res += check(ants[i], ants[j]);
        }

        OT(res);

        fill(hd+1, hd+n+1, 0),
        fill(st+1, st+n+1, 0),
        tt = 0;
    }
}



const int N = 5009, M = 2 * 30009;

int D[N], hd[N], suc[M], to[M], cap[M];
int n, m, s, t;

inline void add_edge(int x, int y, int c){
    suc[m] = hd[x], to[m] = y, cap[m] = c, hd[x] = m++;
    suc[m] = hd[y], to[m] = x, cap[m] = 0, hd[y] = m++;
}

inline void add_edgee(int x, int y, int c){
    suc[m] = hd[x], to[m] = y, cap[m] = c, hd[x] = m++;
    suc[m] = hd[y], to[m] = x, cap[m] = c, hd[y] = m++;
}

#define v to[i]
#define c cap[i]
#define f cap[i^1]

bool bfs(){
    static int Q[N]; int cz = 0, op = 1;
    fill(D, D+n, 0), D[Q[0] = s] = 1; while (cz < op){
        int u = Q[cz++]; REP_G(i, u) if (!D[v] && c){
            D[Q[op++] = v] = D[u] + 1;
            if (v == t) return 1;
        }
    }
    return 0;
}

LL Dinitz(){

    to[0] = s;
    LL max_flow = 0;

    while (bfs()){

        static int sta[N], cur[N]; int top = 0;
        sta[0] = 0, cur[s] = hd[s]; while (top != -1){

            int u = to[sta[top]], i; if (u == t){
                int d = INF; REP_1(ii, top) i = sta[ii], checkMin(d, c); max_flow += d;
                DWN_1(ii, top, 1){i = sta[ii], f += d, c -= d; if (!c) top = ii - 1;}
                u = to[sta[top]];
            }

            for (i=cur[u];i;i=suc[i])
                if (D[u] + 1 == D[v] && c) break;

            if (!i) D[u] = 0, --top;
            else {
                cur[u] = suc[i], cur[v] = hd[v];
                sta[++top] = i;
            }
        }
    }

    return max_flow;
}


inline int dist2(int x, int y){return sqr(x)+sqr(y);}

bool init(){

    //
    int W, L, R, N; RD(W, L, R, N); R*=2;
    if (R > W){
        puts(”-1”);
        return 0;
    }

    s = 0, t = 2*N+1, n = t + 1, m = 2; fill(hd, hd+n+1, 0);

    static int x[109], y[109]; REP_1(i, N){
        RD(x[i], y[i]); add_edge(i*2-1, i*2, RD());
        if (y[i] < R) add_edge(s, i*2-1, INF);
        else if (y[i] > W-R) add_edge(i*2, t, INF);
    }

    int RR = sqr(R); REP_2_1(j, i, N, j-1) if (dist2(x[i]-x[j], y[i]-y[j]) < RR){
        add_edge(i*2, j*2-1, INF);
        add_edge(j*2, i*2-1, INF);
    }

    return 1;
}

int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    Rush{
        if (init()) OT(Dinitz());
    }
}


//nlogn判断二维平面n条线段是否存在交点

const int N = 50009;
Seg L[N]; int n;

struct Event{
    DB x; int id, ty; Event(DB x, int id, int ty):x(x),id(id),ty(ty){}
    bool operator <(const Event &r)const {
        return sgn(x, r.x) < 0 || !sgn(x, r.x) && ty > r.ty;
    }
};

DB x; DB y(const Line& l){
    return sgn(l.d().x) ? l.a.y + (x - l.a.x)/l.d().x*l.d().y : l.a.y;
}

struct rec{
    int id; rec(int id):id(id){}
    bool operator<(const rec& r)const{
        return y(L[id]) < y(L[r.id]);
    }
};

#define s_it set<rec>::iterator
set<rec> S; s_it _S[N];
inline s_it preIt(s_it it){return it == S.begin() ? S.end() : --it;}
inline s_it sucIt(s_it it){return it == S.end() ? S.end() : ++it;}

void isIntersect(){

    vector<Event> I; REP(i, n){
        I.PB(Event(L[i].a.x, i, 1));
        I.PB(Event(L[i].b.x, i, -1));
    }

    SRT(I); ECH(i, I){

        x = i->x;

#define tryIntersect(a, b) if (~L[a].sgn(L[b])){ \
    printf(”YES\n%d %d\n”, a+1, b+1); return; \
}
        if (~i->ty){
            rec cur(i->id);
            s_it suc = S.lower_bound(cur), prd = preIt(suc);
            if (suc != S.end()) tryIntersect(cur.id, suc->id);
            if (prd != S.end()) tryIntersect(cur.id, prd->id);
            _S[i->id] = S.insert(suc, cur);
        }
        else {
            s_it &cur = _S[i->id], prd = preIt(cur), suc = sucIt(cur);
            if(prd != S.end() && suc != S.end()) tryIntersect(prd->id, suc->id);
            S.erase(cur);
        }
    }
    puts(”NO”);
}


int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    REP_C(i, RD(n)) L[i].in();
    isIntersect();
}




// Beiju ..

const int N = 109;

Po L[N], R[N];
int n, m;

DB x, s;
void g(cPo a, cPo b){
    DB t = det(a, b);
    x += t * (a.x + b.x);
    s += t;
}
Po gl(DB y){
    int i; REP_N(i, n){
        if (sgn(L[i+1].y, y) > 0) break;
        g(L[i], L[i+1]);
    }
    Po p = L[i] + (y-L[i].y)/(L[i+1].y-L[i].y)*(L[i+1]-L[i]);
    g(L[i], p);
    return p;
}
Po gr(DB y){
    int i; REP_N(i, m){
        if (sgn(R[i+1].y, y) > 0) break;
        g(R[i+1], R[i]);
    }
    Po p = R[i] + (y-R[i].y)/(R[i+1].y-R[i].y)*(R[i+1]-R[i]);
    g(p, R[i]);
    return p;
}
bool f(DB y){
    x = 0, s = 0, g(gl(y), gr(y)), x /= s*3;
    return sgn(L[0].x, x) <= 0 && sgn(x, R[0].x) <= 0;
}

int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    Rush{
        RD(n, m); vector<DB> Y; REP(i, n) L[i].in(), Y.PB(L[i].y); REP(i, m) R[i].in(), Y.PB(R[i].y);
        //assert(!sgn(L[n-1].y, R[m-1].y));
        DB top = min(L[n-1].y, R[m-1].y); UNQ(Y); int i; FOR_N(i, 1, SZ(Y)) if (!f(Y[i]) || !sgn(Y[i], top)) break;
        DB l = Y[i-1], r = Y[i]; DO(233){
            DB m = (l + r) / 2; if (f(m)) l = m; else r = m;
        }
        OT(l);
    }
}

//滚球兽
const int N = 109;

VP P, C; int n, nn;
Po T, O; DB alpha, res;

inline DB ang(VP &C, int i){
    return i ? ang(C[i-1],C[i],C[i],C[i+1]) : ang(C[nn-1],C[0],C[0],C[1]);
}

bool Roll(){
    DB beta = OO; REP(i, n){
        Seg l = Line(P[i], P[i+1]);
        if (~sgn(dist2(O,P[i]),dist2(O,P[i+1]))) swap(l.a, l.b);
        Po o = O&l; if (~sgn(dist2(O,o), dist2(O,T))) continue; //!
        Po t = o + l.d()._1() * sqrt(dist2(O,T) - dist2(O,o));
        if (~l.sgn(t)) checkMin(beta, arg(O,T,t));
    }
    return sgn(beta, alpha) <= 0 ? res += beta, 1 : 0;
}

int main() {

#ifndef ONLINE_JUDGE
```

```
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    while (scanf(”%d”, &n) != EOF){

        printf(”Case %d: ”, ++Case);

        P.resize(n); REP(i, n) P[i].in(); DB pm=getPeri(C=getConvexHull(P));
        P.PB(P[0]); T.in(); nn=SZ(C)-1;

        int i; REP_N(i, nn) if (!sgn(C[i].y)) break;
        int c=(T.x-C[i].x)/pm-1; res=2*PI*c, T.x-=pm*c;

        REP_N(c, 2*nn){ //!
            O = C[i], alpha = c ? ang(C, i) : (C[i+1]-C[i]).arg(); if (Roll()) break;
            T.rot(alpha, O), res += alpha; if (++i == nn) i = 0;
        }

        if (c == 2*nn) puts(”Impossible”); else OT(res);
    }
}
```

Problem F. Moles
Brief description:
... 动态维护一棵边权树, 支持单边修改。。（边权可以是负数）。。
。。以及询问两点之间的路径长度，询问某点到期子树内一点的路径长度的最大值。

```
const int N = 100009, M = 2 * N, LM = 24;
int hd[N], suc[M], to[M], wt[M];

int ST[LM][M], st[N], dep[N]; // Euler index ...;
int L[N], R[N], A[M], S[4*M], T[4*M];
int n, a, b, nn, tt;

inline bool elder(int a, int b){
    return dep[a] < dep[b];
}

inline int lca(int a, int b){
    int l = st[a], r = st[b];
    if (l > r) swap(l, r); ++r; int lv = log2(r - l);
    return min(ST[lv][l], ST[lv][r-(1<<lv)], elder);
}

#define aa to[i^1]
#define bb to[i]
#define v bb
#define w wt[i]

void dfs(int u = 1){
    L[u] = ++nn, ST[0][st[u] = ++tt] = u;
    REP_G(i, u) if (!L[v]){
        dep[v] = dep[u] + 1, dfs(v), A[L[v]] = w, A[R[v]] = -w;
        ST[0][++tt] = u;
    }
    R[u] = ++nn;
}

#define lx (x<<1)
#define rx (lx|1)
#define mid (l + r >> 1)
#define lc lx, l, mid
#define rc rx, mid+1, r
#define root 1, 1, nn
```

```

void Build(int x, int l, int r){
    if (l == r) T[x] = max(0, S[x] = A[l]);
    else {
        Build(lc), Build(rc);
        S[x] = S[lx] + S[rx];
        T[x] = max(T[lx], S[lx] + T[rx]);
    }
}

void Get(int x, int l, int r){ // get interval
    if (a <= l && r <= b) A[tt++] = x;
    else{
        if (a <= mid) Get(lc);
        if (mid < b) Get(rc);
    }
}

int Q1(int a, int b){
    ::a = a, ::b = b, tt = 0, Get(root);
    int res = 0; REP(i, tt) res += S[A[i]];
    return res;
}

int Q2(int a, int b){
    ::a = a, ::b = b, tt = 0, Get(root);
    int res = 0, sss = 0; REP(i, tt){
        int ai = A[i];
        checkMax(res, sss + T[ai]);
        sss += S[ai];
    }
    return res;
}

void Modify(int x, int l, int r){
    if (l == r) T[x] = max(0, S[x] = b);
    else {
        if (a <= mid) Modify(lc);
        else Modify(rc);
        S[x] = S[lx] + S[rx];
        T[x] = max(T[lx], S[lx] + T[rx]);
    }
}

int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    Rush{

        RST(hd, L, A, S, T), tt = nn = 0; FOR_C(i, 2, RD(n) << 1){
            RD(aa, bb); RDD(wt[i]), wt[i|1] = wt[i];
            suc[i] = hd[aa], hd[aa] = i, ++i;
            suc[i] = hd[aa], hd[aa] = i;
        }

        dfs();

        Build(root);

        for ( int lv = 1 ; (1 << lv) <= tt ; lv ++ ){
            for ( int i = 1 ; i + (1 << lv) <= tt + 1 ; i ++ )
                ST[lv][i] = min(ST[lv-1][i], ST[lv-1][i + (1<<(lv-1))], elder);
        }

        //REP_1(i, nn) cout << A[i] << ” ”; cout << endl;
        //REP_1(i, nn) cout << Sum(i, i) << ” ”; cout <<endl;

        int cmd; int x = 1, y, z; Rush{
            RD(cmd); if (cmd == 1){
                z = lca(x, RD(y));
                OT(Q1(L[z]+1, L[x]) + Q1(L[z]+1, L[y]));
                x = y;
            } else if (cmd == 2){
                OT(Q2(L[x]+1, R[x]-1));
            } else {
                int x; RD(x, y), RDD(z); //if (elder(y, x)) y = x;
                a = L[y], b = z, Modify(root);
                a = R[y], b = -z, Modify(root);
            }
        }
    }
}
```

34.1.5 blue-red hackenbush

简述 (Brief description)

分析 (Analysis)

```
#include <cstring>
#include <string>
#include <iostream>
#include <cstdio>
#include <vector>
#include <cassert>
#include <algorithm>

using namespace std;
#define MAXN 55
typedef long long int64;

/*
    Problem can be reduced to red-black hackenbush
    http://en.wikipedia.org/wiki/Hackenbush
    Each pile represent a hackenbush stalk
    Game value cooresponding to hackenbush stalk is easy to find.
    Please refer here : http://www.geometer.org/mathcircles/hackenbush.pdf.
    For hackebush games value of two disjoint game is equal to sum of individual game value.
    (http://www-math.mit.edu/~rstan/transparencies/games.pdf)

*/

int t,n,tcase;
int arr[MAXN];

int64 calculate(){
    int64 res = 0; int64 value = 1LL<<48;
    res = (arr[0]%2==0)?value:-value;
    bool is_changed = false;
    for(int i=1; i<n; ++i){
        assert(arr[i]!=arr[i-1]);
        if(arr[i]%2 != arr[i-1]%2){
            is_changed = true;
        }
        if(is_changed) value /= 2;
        res += (arr[i]%2==0)?value:-value;
    }
    return res;
}

int main(){
  for(scanf("%d",&tcase); tcase; tcase-=1){
    scanf("%d",&t);
    int64 res = 0;
    for(int i=0; i<t; ++i){
      scanf("%d",&n);
      for(int j=0; j<n; ++j) scanf("%d",&arr[j]);
      sort(arr,arr+n);
      res += calculate();
    }
    if(res > 0 ) printf("FIRST\n");
    else if(res < 0 ) printf("SECOND\n");
    else printf("DON'T PLAY\n");
  }
  return 0;
}
```

34.1.6 满足给定后缀数组的字符串数

简述 (Brief description)

分析 (Analysis)

Given a Suffix Array find number of strings
Let the suffix array be a1, a2, ... an. Let S be an arbitrary string corresponding to the given suffix array.
Let
S[i] = ith character of the string
Suffix[i] = suffix of S starting from ith character.

We must have

S[a1]   S[a2]   S[a3] ...   S[an]   - (1)

Also, by our definition of "string", if S[ai] < S[ai+1], then S[ai+1] = S[ai] + 1.

Therefore, if we replace each ' ' by exactly one out of '=' and '<', we will get a unique string.

Also, if each ' ' had full "freedom" of choosing exactly one out of '<' and '=', then number of strings for a given suffix array would be 2n-1.

However, world is not such a nice place. Any ' ' can always be replaced by '<', but it might not possible to replace it by '='. So lets try to see

Consider an arbitrary ' ', S[ai]   S[ai+1], and lets see what happens when we replace it by '='.

By suffix array condition, we have

Suffix[ai] < Suffix[ai+1].
But   S[ai] = S[ai+1]
So, we should have Suffix[ai+1] < Suffix[ai+1+1]

Therefore, the ' ' between S[ai] and S[ai+1] can be replaced by '=' iff ai+1 appears before ai+1+1 in the suffix array. The answer will be 2k
k = no of positions i such that ai+1 appears before ai+1+1 in the suffix array

To handle the case where ai = N or ai+1 = N, we can append(conceptually) '0' to the of string, and rewrite the suffix array as
n+1, a1, a2, ... an

Since the suffix array is very large, we will actually obtain it in the form:
(x1, y1), (x2, y2), ... (xk, yk)
Where each (xi, yi) represents an AP starting at xi, ending at yi, with common difference ±1.
It can be verified if both ai and ai+1 are not boundary of some contiguous segment(i.e. do not belong to the set { x1, y1, x2, y2, ... xk, yk })
It is non trivial to find the position of ai+1 's. However, to check the order of two arbitrary elements in suffix array, we only need to check

If they belong to same contiguous segment, then what is their order in the segment.
If they belong to different segments, then what is the order of those segments.
This can be done using STL's map data structure as m[xi] = m[yi] = i, and using lower_bound to locate the relative position of segments i

```
/*
Solution:

Using splay tree to creat the permutation.
Of course we cannot store all 10^9 numbers but we can know exactly where they are in the final array.

Each node in the splay tree will store a number of consecutive numbers from u to v (u may larger than v).
At the beginning our tree contains only single node which is (1, n).
For each operation at most two nodes will be splited an hence at most two new nodes are created.
Operation flip can be implemented with lazy update.

The final permutation will have the form (u_1, v_1) (u_2, v_2), ... (u_i, v_i).
Each (u_i, v_i) represent for a range of consecutive numbers.

When calculate the number of strings, we add an amazinal zero character at the end of the string and hence the suffix array will have an additional number - n + 1 at the end (the suffix n + 1 will be the smallest one).
Let's the suffix array is a_1, a_2, ..., a_n. Consider two adjacent elements a_i and a_(i + 1). We have two case:
1. the number (a_i + 1) is in the place before the number (a_(i + 1) + 1) in the array so S[a_i] <= S[a_(i + 1)];
2. the number (a_i + 1) is in the place after the number (a_(i + 1) + 1) in the array so S[a_i] < S[a_(i + 1)];
(This explain why we need the additional zero character at the end).

So when consider all adjacent numbers, we'll have that: S[a_1] <= or < S[a_2] <= or < S[a_3] ...
If S[a_i] < S[a_(i + 1)] then S[a_(i + 1)] = S[a_i] + 1 (to ensure that the number of different characters is equal to the maximal character).
If S[a_i] <= S[a_(i + 1)] then S[a_(i + 1)] = S[a_i] or S[a_i] + 1.
Besides, S[a_1] = 1.
So the result will be 2^(number of <=s).
Notice that our permutation is big so we need to think a little bit more to calculate the numer of <=s.

Test generation:
The core part of this problem is using special data structure so I just generate some random test.
As long as the number of consecutive groups in the final permutation is large (which will be sastified in random test) I think it's ok.

*/
#pragma comment(linker, "/STACK:16777216")
#include <cstdio>
#include <iostream>
#include <algorithm>
#include <vector>
#include <queue>
#include <stack>
#include <set>
#include <map>
#include <cstring>
#include <cstdlib>
#include <cmath>
#include <string>
#include <memory.h>
#include <sstream>
#include <complex>

#define REP(i,n) for(int i = 0, _n = (n); i < _n; i++)
#define REPD(i,n) for(int i = (n) - 1; i >= 0; i--)
#define FOR(i,a,b) for (int i = (a), _b = (b); i <= _b; i++)
#define FORD(i,a,b) for (int i = (a), _b = (b); i >= _b; i--)
#define DOWN(i,a,b) for (int i = (a), _b = (b); i >= _b; i--)
#define FOREACH(it,c) for (__typeof((c).begin()) it=(c).begin();it!=(c).end();it++)
#define RESET(c,x) memset (c, x, sizeof (c))

#define sqr(x) ((x) * (x))
#define PB push_back
```

```
#define MP make_pair
#define F first
#define S second
#define ALL(c) (c).begin(), (c).end()
#define SIZE(c) (c).size()

#define DEBUG(x) { cerr << #x << ” = ” << x << endl; }
#define PR(a,n) {cerr<<#a<<” = ”; FOR(_,1,n) cerr << a[_] << ' '; cerr <<endl;}
#define PR0(a,n) {cerr<<#a<<” = ”;REP(_,n) cerr << a[_] << ' '; cerr << endl;}

#define oo 2000111000
#define mod 1000000007
using namespace std;

struct Node {
    Node *left, *right, *parent;
    //Each node of the splay tree stores one segment [u..v] (u can be larger than v)
    //num is the total number of array’s elements in the sub-tree rooted at this node
    //flip = 1 when we need to flip this sub-tree but haven’t done it yet (lazy update)
    int u, v, num, flip;

};

Node *nullT;
int nNum, numU[1222222], numV[1222222], d[1222222], nD, n;

void initTree() {
    nullT = new Node;
    nullT -> left = nullT -> right = nullT -> parent = nullT;
    nullT -> u = nullT -> v = nullT -> num = nullT -> flip = 0;
}

//Splay tree’s stuffs
void setLink (Node *parent, Node *child, bool isLeft) {
    if (isLeft) parent -> left = child;
    else parent -> right = child;

    if (child != nullT) child -> parent = parent;
}

//If we need to flip the sub-tree rooted at root, we’ll do it now
void lazyUpdate(Node *root) {
     if (root == nullT) return;

     if (root -> flip) {
        root -> flip = 0;
        swap(root -> u, root -> v);
        swap(root -> left, root -> right);
        root -> left -> flip ^= 1;
        root -> right -> flip ^= 1;
    }
}

void update(Node * x) {
    if (x == nullT) return;
    x -> num = x -> right -> num + x -> left -> num + abs(x->u - x->v) + 1;
}

//find the node of the tree that contains the kth element
//We will split some node so that the there will be a node contains the segment [u..v] and v is the kth element
Node *findNode(Node * root, int kth) {
    lazyUpdate(root);

    if (root -> left -> num >= kth) return findNode(root -> left, kth);

    int len = abs(root -> u - root -> v) + 1;
    if ((root -> left -> num + len) >= kth) {
        if (root -> left -> num + len == kth) return root;

        int mid, pre;
        int dt = kth - root -> left -> num;

        if (root -> u < root -> v) {
            mid = root -> u + dt;
            pre = mid - 1;
        }
        else {
            mid = root -> u - dt;
            pre = mid + 1;
        }

        Node * tmp = new Node;
        tmp -> u = mid;
        tmp -> v = root -> v;
        tmp -> flip = 0;

        root -> v = pre;

        tmp -> left = nullT;
        setLink(tmp, root->right, 0);
        setLink(root, tmp, 0);

        update(tmp);
        update(root);

        return root;
    }

    return findNode(root -> right, kth - len - root -> left -> num);
}

void upTree(Node *x) {
    Node *y = x -> parent;
    Node *z = y -> parent;
    Node *tmp;
    if (y->right == x) {
        tmp = x -> left;
        setLink(x, y, true);
        setLink(y, tmp, false);
    }
    else {
        tmp = x -> right;
        setLink(x, y, false);
        setLink(y, tmp, true);
    }
    setLink(z, x, z->left == y);
    update(y); update(x);
}

void splay(Node *x) {
    while (1) {
        Node *y = x -> parent;
        if (y == nullT) return ;

        Node *z = y -> parent;

        if (z != nullT)
            if ((z->right == y) == (y->right == x)) upTree(y);
            else upTree(x);
        upTree(x);
    }
}

void printTree(Node *root) {
    lazyUpdate(root);
    if (root == nullT) return;
    printTree(root -> left);

    printf(”[%d %d]”, root -> u, root -> v);

    printTree(root -> right);
}

//Find the final array
void extract(Node *root) {
    lazyUpdate(root);
    if (root == nullT) return;
    extract(root -> left);

    numU[++nNum] = root->u;
    numV[nNum] = root->v;

    extract(root -> right);
}

//Split a tree into two sub-strees. a and b where a contains the first num elements
void split(Node *root, int num, Node * &a, Node * &b) {
    Node *tmp = findNode(root, num);
    splay(tmp);

    b = tmp -> right;
    b -> parent = nullT;

    tmp -> num -= tmp -> right -> num;
    tmp -> right = nullT;
    a = tmp;
}

void debugTree(Node * root) {
    printTree(root);
    printf(”\n”);
}

//Join two trees, a and b
Node * join(Node * a, Node * b) {
    Node * tmp = findNode(a, a->num);
    splay(tmp);


    setLink(tmp, b, 0);

    update(tmp);
    return tmp;
}

long long pow2(int n) {
    if (n == 0) return 1;

    long long res = pow2(n / 2);
    res = res * res % mod;

    if (n % 2) res = res * 2 % mod;

    return res;
}

vector <pair <int, int> > seg;
map <int, int> pos;

long long call() {
    //numV[nNum] always equal to n + 1
    //if numU[nNum] = n + 1 too, we ignore this segment, otherwise, we decrease numV[nNum] by 1
    if (numU[nNum] == n + 1) nNum--;
    else numV[nNum]--;

    //ignore the number 0 too
    int START = 1;
    if (numV[1] == 0) {
        START ++;
    }
    else numU[1] ++;

    FOR (i, START, nNum) {
        int d = 1;
        if (numU[i] > numV[i]) d = -1;
        int len = abs(numU[i] - numV[i]) + 1;
        if (len <= 4) {
            int u = numU[i];
            FOR (i, 1, len) {
                seg.PB(MP(u, u));
                u += d;
            }
        }
        else {
            int u = numU[i];
            seg.PB(MP(u, u));
            seg.PB(MP(u + d, u + d));

            int v = numV[i];
            seg.PB(MP(u + 2 * d, v - 2 * d));

            seg.PB(MP(v - d, v - d));
            seg.PB(MP(v, v));
        }
    }



    int pr = 0;
    int z = seg.size();

    FOR (i, 0, z - 1) {
        int len = abs(seg[i].first - seg[i].second) + 1;
        pos[seg[i].first] = pr + 1;
        pos[seg[i].second] = pr + len;
        pr += len;
    }
    pos[n + 1] = -1;

    int cnt = 0;

    //In the segment [u...v] (suppose u <= v)
    //We have that S[u] <= S[u + 1] <= S[u + 2] <= ... <= s[v - 1]
    //s[v + 1] may <= or < s[v] and this depends on the position of v + 1 and u - 1.
    //Similarly in the case u > v

    FOR (i, 0, z - 2) {
        int u = pos[seg[i].second + 1];
        if (u == 0) u = pos[seg[i].second] - 1;

        int v = pos[seg[i + 1].first + 1];
        if (v == 0) v = pos[seg[i + 1].first] + 1;

        if (u < v) cnt++;
    }


    FOR (i, 0, z - 1)
```

```
        cnt += abs(seg[i].first - seg[i].second);

    return pow2(cnt);
}

int main() {
    initTree();
    Node * root = new Node;
    int m;
    cin >> n >> m;

    //Initially the tree contain only one node: (0, n + 1)
    //adding 0 and n + 1 to avoid the cornner cases
    root -> u = 0;
    root -> v = n + 1;
    root -> num = n + 2;
    root -> flip = 0;

    setLink(root, nullT, 0);
    setLink(root, nullT, 1);

    root -> parent = nullT;

    while (m--) {
        int u, v, k;

        scanf(”%d%d%d”, &k, &u, &v);

        Node *a, *b, *c, *d, *e;

        if (k == 1) {
            //split the tree into 3 smaller trees a, b and c
            //b will contains only the elements from uth elements to vth elements
            split(root, u, a, b);
            split(b, v - u + 1, b, c);

            //flip b and join 3 trees
            b->flip |= 1;
            a = join(a, b);
            root = join(a, c);
        }
        else {
            if(u == 1) continue;
            //similarly split the tree into 3 sub-tree a, b, c
            split(root, 1, a, b);
            split(b, u - 1, b, c);
            split(c, v - u + 1, c, d);
            //we just need to change the order of a, b and c when joining them
            root = join(a, c);
            root = join(root, b);
            root = join(root, d);
        }
    }

    nNum = 0;
    extract(root);

    cout << call() << endl;
    return 0;
}
```

34.1.7 POJ 1741. Tree

简述 (Brief description)

... 求树中距离 <=k 的点对总数..

分析 (Analysis)

..

```
const int N = int(1e4) + 9, M = N * 2;

int hd[N], prd[M], suc[M], to[M], ww[N]; // adj ...
int dep[N], sz[N]; int L[N], Ln, ans;
int n, k, nn, c, cc;

#define a to[i^1]
#define b to[i]
#define w ww[i/2]
#define v b

inline void del(int i){
    if (i == hd[a]) prd[hd[a] = suc[i]] = 0;
    else prd[suc[i]] = prd[i], suc[prd[i]] = suc[i];
}

void dfs_c(int u, int p = 0){
    int ss = 0; sz[u] = 1;
    REP_G(i, u) if (v != p){
        dfs_c(v, u), sz[u] += sz[v];
        checkMax(ss, sz[v]);
    }
    checkMax(ss, nn - sz[u]);
    if (ss <= cc) cc = ss, c = u;
}

void dfs0(int u, int p = 0){
    L[Ln++] = dep[u], sz[u] = 1;
    REP_G(i, u) if (v != p){
        dep[v] = dep[u] + w;
        dfs0(v, u), sz[u] += sz[v];
    }
}

void dfs1(int u, int p = 0){
    L[Ln++] = dep[u];
    REP_G(i, u) if (v != p){
        dfs1(v, u);
    }
}

int f(){
    int res = 0, l = 0, r = Ln - 1;
    sort(L, L+Ln);
    while (l < r){
        if (L[l] + L[r] > k) --r;
        else res += r - l++;
    }
    Ln = 0;
    return res;
}

void gao(int u = 1){
    cc = INF, dfs_c(u), u = c;
    dep[u] = 0, dfs0(u), ans += f();

    REP_G(i, u){
        del(i^1), dfs1(v), ans -= f();
        nn = sz[v], gao(v);
    }
}

int main(){

```

```
#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif


    while (RD(n, k)){

        fill(hd+1, hd+n+1, 0), ans = 0;

        FOR_C(i, 2, n<<1){
            RD(a, b), RDD(w);
            suc[prd[hd[a]] = i] = hd[a], hd[a] = i++;
            suc[prd[hd[a]] = i] = hd[a], hd[a] = i;
        }

        nn = n, gao(), OT(ans);
    }

}
```

34.1.8 Hangzhou Generator

简述 (Brief description)

概率，AC 自动机

分析 (Analysis)

```
const int N = 20;

DB A[N][N];
int n, m;

namespace ACM{
    const int Z = 26, L = 20;
    int trans[N][Z], fail[N], cnt[N], Q[N], u, cz, op, tot;
    char str[L];

    inline int new_node(){
        fail[tot] = cnt[tot] = 0, RST(trans[tot]);
        return tot++;
    }

#define v trans[u][c]
#define f trans[fail[u]][c]

    inline void Build(){
        cz = op = u = 0; REP(c, Z) if (v) Q[op++] = v;
        while (cz < op){
            u = Q[cz++]; REP(c, Z){
                if (v) fail[Q[op++] = v] = f; // .. .
                else v = f;
            }
        }
    }

#define c (*cur - ’A’)
    inline void Insert(){
        RS(str), u = 0; REP_S(cur, str){
            if (!v) v = new_node();
            u = v;
        }
        n = strlen(str);
    }

    void Init(){
        tot = 0, new_node();
        Insert(); Build();
    }

#undef c
} using namespace ACM;

void Gauss(){
    REP(i, n){
        if (!sgn(A[i][i])){
                int j; FOR_N(j, i+1, n) if (sgn(A[j][i])) break;
                if (j == n){
                    // Warning;
                    assert(0);
                }
                FOR_1(k, 0, n) swap(A[i][k], A[j][k]);

        }
        DB t = A[i][i]; FOR_1(j, i, n) A[i][j] /= t;
        REP(j, n) if (i!=j&&sgn(A[j][i])){
            DB t = A[j][i]; FOR_1(k, i, n) A[j][k] -= A[i][k] * t;
        }
    }
}

int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    Rush{

        if (Case) puts(””);

        RD(m), Init();

        RST(A); REP(u, n){
            A[u][u] = A[u][n+1] = m;
            REP(c, m) A[u][v] -= 1;
        }

        A[n][n] = 1, ++n;
        //Display(A, n, n+1);
        Gauss();
        //Display(A, n, n+1);

        OT(A[0][n]);
    }

}
```

34.1.9 BZOJ 2154. Crash 的数字表格

简述 (Brief description)

...

分析 (Analysis)

..

```
const int PMAX = int(1e7) + 9;
VI P; int pp[PMAX]; Int G[PMAX];

void sieve(){
    G[1] = 1; FOR(i, 2, PMAX){
        if (!pp[i]) P.PB(i), pp[i] = i, G[i] = (LL)i*(1-i);
#define ii (i*P[j])
        for (int j=0;j<SZ(P)&&ii<PMAX;++j) if (i%P[j]){
            pp[ii] = P[j], G[ii] = G[i]*G[P[j]];
        } else{
            pp[ii] = pp[i]*P[j], G[ii] = pp[i]==i?(Int)ii*(1-P[j]):G[pp[i]*P[j]]*G[i/pp[i]];
            break;
        }
    }
#undef ii
    FOR(i, 1, PMAX) G[i] = G[i-1] + G[i];
}

Int f(int a, int b){
    if (a > b) swap(a, b); Int z=0; for(int i=1,ii;i<=a;i=ii+1){
        int aa = a/i, bb = b/i; ii = min(a/aa, b/bb);
        z += (G[ii]-G[i-1])*aa*bb*(aa+1)*(bb+1);
    }
    return z/4;
}

int main(){

#ifndef ONLINE_JUDGE
    freopen(”in.txt”, ”r”, stdin);
    //freopen(”out.txt”, ”w”, stdout);
#endif

    sieve(); OT(f(RD(), RD()));
}
```