Biological Warfare

CMU_xiaodao

Last build at April 16, 2014

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)
```

```
49  #define REP_1_C(i, n) for (int n_____=n,i=1;i<=n_____;++i)
50  #define FOR_1_C(i, a, b) for (int b_____=b,i=a;i<=b_____;++i)
51  #define DWN_1_C(i, b, a) for (int a_____=a,i=b;i>=a_____;--i)
52  #define REP_1_N(i, n) for (i=1;i<=n;++i)
53  #define FOR_1_N(i, a, b) for (i=a;i<=b;++i)
54  #define DWN_1_N(i, b, a) for (i=b;i>=a;--i)
55  #define REP_C_N(i, n) for (int n_____=(i=0,n);i<n_____;++i)
56  #define FOR_C_N(i, a, b) for (int b_____=(i=0,b);i<b_____;++i)
57  #define DWN_C_N(i, b, a) for (int a_____=(i=b-1,a);i>=a_____;--i)
58  #define REP_1_C_N(i, n) for (int n_____=(i=1,n);i<=n_____;++i)
59  #define FOR_1_C_N(i, a, b) for (int b_____=(i=1,b);i<=b_____;++i)
60  #define DWN_1_C_N(i, b, a) for (int a_____=(i=b,a);i>=a_____;--i)
61
62  #define ECH(it, A) for (__typeof(A.begin()) it=A.begin(); it != A.end(); ++it)
63  #define REP_S(i, str) for (char*i=str;*i;++i)
64  #define REP_L(i, hd, nxt) for (int i=hd;i;i=nxt[i])
65  #define REP_G(i, u) REP_L(i,hd[u],suc)
66  #define REP_SS(x, s) for (int x=s;x;x=(x-1)&s)
67  #define DO(n) for ( int _____n = n; _____n-->0; )
68  #define REP_2(i, j, n, m) REP(i, n) REP(j, m)
69  #define REP_2_1(i, j, n, m) REP_1(i, n) REP_1(j, m)
70  #define REP_3(i, j, k, n, m, l) REP(i, n) REP(j, m) REP(k, l)
71  #define REP_3_1(i, j, k, n, m, l) REP_1(i, n) REP_1(j, m) REP_1(k, l)
72  #define REP_4(i, j, k, ii, n, m, l, nn) REP(i, n) REP(j, m) REP(k, l) REP(ii, nn)
73  #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)
74
75  #define ALL(A) A.begin(), A.end()
76  #define LLA(A) A.rbegin(), A.rend()
77  #define CPY(A, B) memcpy(A, B, sizeof(A))
78  #define INS(A, P, B) A.insert(A.begin() + P, B)
79  #define ERS(A, P) A.erase(A.begin() + P)
80  #define LBD(A, x) (lower_bound(ALL(A), x) - A.begin())
81  #define UBD(A, x) (lower_bound(ALL(A), x) - A.begin())
82  #define CTN(T, x) (T.find(x) != T.end())
83  #define SZ(A) int((A).size())
84  #define PB push_back
85  #define MP(A, B) make_pair(A, B)
86  #define PTT pair<T, T>
87  #define Ts *this
88  #define rTs return Ts
89  #define fi first
90  #define se second
91  #define re real()
92  #define im imag()
93
94  #define Rush for(int _____T=RD(); _____T--;)
95  #define Display(A, n, m) { \
96    REP(i, n){  \
97          REP(j, m-1) cout << A[i][j] << " "; \
98          cout << A[i][m-1] << endl;  \
99      }     \
100 }
101 #define Display_1(A, n, m) { \
102     REP_1(i, n){  \
103         REP_1(j, m-1) cout << A[i][j] << " "; \
104         cout << A[i][m] << endl;  \
105     }     \
106 }
107
108 typedef long long LL;
109 //typedef long double DB;
110 typedef double DB;
111 typedef unsigned uint;
112 typedef unsigned long long uLL;
113
114 typedef vector<int> VI;
115 typedef vector<char> VC;
```

```
116 typedef vector<string> VS;
117 typedef vector<LL> VL;
118 typedef vector<DB> VF;
119 typedef set<int> SI;
120 typedef set<string> SS;
121 typedef map<int, int> MII;
122 typedef map<string, int> MSI;
123 typedef pair<int, int> PII;
124 typedef pair<LL, LL> PLL;
125 typedef vector<PII> VII;
126 typedef vector<VI> VVI;
127 typedef vector<VII> VVII;
128
129 template<class T> inline T& RD(T &);
130 template<class T> inline void OT(const T &);
131 //inline int RD(){int x; return RD(x);}
132 inline LL RD(){LL x; return RD(x);}
133 inline DB& RF(DB &);
134 inline DB RF(){DB x; return RF(x);}
135 inline char* RS(char *s);
136 inline char& RC(char &c);
137 inline char RC();
138 inline char& RC(char &c){scanf(" %c", &c); return c;}
139 inline char RC(){char c; return RC(c);}
140 //inline char& RC(char &c){c = getchar(); return c;}
141 //inline char RC(){return getchar();}
142
143 template<class T> inline T& RDD(T &);
144 inline LL RDD(){LL x; return RDD(x);}
145
146 template<class T0, class T1> inline T0& RD(T0 &x0, T1 &x1){RD(x0), RD(x1); return x0;}
147 template<class T0, class T1, class T2> inline T0& RD(T0 &x0, T1 &x1, T2 &x2){RD(x0), RD(x1), RD(x2); return x0;}
148 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;}
149 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;}
150 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;}
151 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;}
152 template<class T0, class T1> inline void OT(const T0 &x0, const T1 &x1){OT(x0), OT(x1);}
153 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);}
154 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);}
155 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);}
156 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);}
157 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);}
158 inline char& RC(char &a, char &b){RC(a), RC(b); return a;}
159 inline char& RC(char &a, char &b, char &c){RC(a), RC(b), RC(c); return a;}
160 inline char& RC(char &a, char &b, char &c, char &d){RC(a), RC(b), RC(c), RC(d); return a;}
161 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;}
162 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;}
163 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;}
164 inline DB& RF(DB &a, DB &b){RF(a), RF(b); return a;}
165 inline DB& RF(DB &a, DB &b, DB &c){RF(a), RF(b), RF(c); return a;}
166 inline DB& RF(DB &a, DB &b, DB &c, DB &d){RF(a), RF(b), RF(c), RF(d); return a;}
167 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;}
168 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;}
169 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;}
170 inline void RS(char *s1, char *s2){RS(s1), RS(s2);}
171 inline void RS(char *s1, char *s2, char *s3){RS(s1), RS(s2), RS(s3);}
172 template<class T0,class T1>inline void RDD(T0&a, T1&b){RDD(a),RDD(b);}
173 template<class T0,class T1,class T2>inline void RDD(T0&a, T1&b, T2&c){RDD(a),RDD(b),RDD(c);}
```

```
174
175 template<class T> inline void RST(T &A){memset(A, 0, sizeof(A));}
176 template<class T> inline void FLC(T &A, int x){memset(A, x, sizeof(A));}
177 template<class T> inline void CLR(T &A){A.clear();}
178
179 template<class T0, class T1> inline void RST(T0 &A0, T1 &A1){RST(A0), RST(A1);}
180 template<class T0, class T1, class T2> inline void RST(T0 &A0, T1 &A1, T2 &A2){RST(A0), RST(A1), RST(A2);}
181 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);}
182 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);}
183 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);}
184 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);}
185 template<class T0, class T1> inline void FLC(T0 &A0, T1 &A1, int x){FLC(A0, x), FLC(A1, x);}
186 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);}
187 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);}
188 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);}
189 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);}
190 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);}
191 template<class T> inline void CLR(priority_queue<T, vector<T>, less<T> > &Q){while (!Q.empty()) Q.pop();}
192 template<class T> inline void CLR(priority_queue<T, vector<T>, greater<T> > &Q){while (!Q.empty()) Q.pop();}
193 template<class T> inline void CLR(stack<T> &S){while (!S.empty()) S.pop();}
194
195 template<class T0, class T1> inline void CLR(T0 &A0, T1 &A1){CLR(A0), CLR(A1);}
196 template<class T0, class T1, class T2> inline void CLR(T0 &A0, T1 &A1, T2 &A2){CLR(A0), CLR(A1), CLR(A2);}
197 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);}
198 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);}
199 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);}
200 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);}
201 template<class T> inline void CLR(T &A, int n){REP(i, n) CLR(A[i]);}
202
203 template<class T> inline bool EPT(T &a){return a.empty();}
204 template<class T> inline T& SRT(T &A){sort(ALL(A)); return A;}
205 template<class T, class C> inline T& SRT(T &A, C B){sort(ALL(A), B); return A;}
206 template<class T> inline T& RVS(T &A){reverse(ALL(A)); return A;}
207 template<class T> inline T& UNQQ(T &A){A.resize(unique(ALL(A))-A.begin());return A;}
208 template<class T> inline T& UNQ(T &A){SRT(A);return UNQQ(A);}
209
210
211 //}
212
213 /** Constant List .. **/ //{
214
215 const int MOD = int(1e9) + 7;
216 //int MOD = 99990001;
217 const int INF = 0x3f3f3f3f;
218 const LL INFF = 0x3f3f3f3f3f3f3f3fLL;
219 const DB EPS = 1e-9;
220 const DB OO = 1e20;
221 const DB PI = acos(-1.0); //M_PI;
222
223 const int dx[] = {-1, 0, 1, 0};
224 const int dy[] = {0, 1, 0, -1};
225
226 //}
227
228 /** 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);}
```

```
296 inline int low_idx(int x){return !x ? -1 : ctz(x);}
297 inline int low_idx(LL x){return !x ? -1 : ctz(x);}
298 inline int high_idx(int x){return lg2(x);}
299 inline int high_idx(LL x){return lg2(x);}
300 inline int parity(int x){return __builtin_parity(x);}
301 inline int parity(LL x){return __builtin_parityll(x);}
302 inline int count_bits(int x){return __builtin_popcount(x);}
303 inline int count_bits(LL x){return __builtin_popcountll(x);}
304
305 } using namespace BO;//}
306 // <<= '9. Comutational Geometry .,//{
307 namespace CG{
308
309 #define cPo const Po&
310 #define cLine const Line&
311 #define cSeg const Seg&
312
313 inline DB dist2(DB x,DB y){return sqr(x)+sqr(y);}
314
315 struct Po{
316     DB x,y;Po(DB x=0,DB y=0):x(x),y(y){}
317
318     void in(){RF(x,y);}void out(){printf("(%.2f,%.2f)",x,y);}
319     inline friend istream&operator>>(istream&i,Po&p){return i>>p.x>>p.y;}
320     inline friend ostream&operator<<(ostream&o,Po p){return o<<"("<<p.x<<", "<<p.y<< ")";}
321
322     Po operator-()const{return Po(-x,-y);}
323     Po&operator+=(cPo p){x+=p.x,y+=p.y;rTs;}Po&operator-=(cPo p){x-=p.x,y-=p.y;rTs;}
324     Po&operator*=(DB k){x*=k,y*=k;rTs;}Po&operator/=(DB k){x/=k,y/=k;rTs;}
325     Po&operator*=(cPo p){rTs=Ts*p;}Po&operator/=(cPo p){rTs=Ts/p;}
326     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);}
327     Po operator*(DB k)const{return Po(x*k,y*k);}Po operator/(DB k)const{return Po(x/k,y/k);}
328     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();}
329
330     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);}
331     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;}
332     bool operator>(cPo p)const{return!(Ts<=p);}bool operator >=(cPo p)const{return!(Ts<p);}
333
334     DB len2()const{return dist2(x,y);}DB len()const{return sqrt(len2());}DB arg()const{return atan2(y,x);}
335     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;}
336     Po&rot(DB a,cPo o=Po()){Ts-=o;Ts*=Po(cos(a),sin(a));rTs+=o;}
337 };
338
339 inline DB dot(DB x1,DB y1,DB x2,DB y2){return x1*x2+y1*y2;}
340 inline DB dot(cPo a,cPo b){return dot(a.x,a.y,b.x,b.y);}
341 inline DB dot(cPo p0,cPo p1,cPo p2){return dot(p1-p0,p2-p0);}
342 inline DB det(DB x1,DB y1,DB x2,DB y2){return x1*y2-x2*y1;}
343 inline DB det(cPo a,cPo b){return det(a.x,a.y,b.x,b.y);}
344 inline DB det(cPo p0,cPo p1,cPo p2){return det(p1-p0,p2-p0);}
345 inline DB ang(cPo p0,cPo p1){return acos(dot(p0,p1)/p0.len()/p1.len());}
346 inline DB ang(cPo p0,cPo p1,cPo p2){return ang(p1-p0,p2-p0);}
347 inline DB ang(cPo p0,cPo p1,cPo p2,cPo p3){return ang(p1-p0,p3-p2);}
348 inline DB dist2(const Po &a, const Po &b){return dist2(a.x-b.x, a.y-b.y);}
349 template<class T1, class T2> inline int dett(const T1 &x, const T2 &y){return sgn(det(x, y));}
350 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));}
351 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));}
352 template<class T1, class T2> inline int dott(const T1 &x, const T2 &y){return sgn(dot(x, y));}
353 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));}
354 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));}
355 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;}
356 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;}
```

```
357 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;}
358 template<class T1, class T2> inline DB dist(const T1 &x, const T2 &y){return sqrt(dist2(x, y));}
359 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));}
360 inline Po _1(Po p){return p._1();}inline Po conj(Po p){return p.conj();}
361 inline Po lt(Po p){return p.lt();}inline Po rt(Po p){return p.rt();}
362 inline Po rot(Po p,DB a,cPo o=Po()){return p.rot(a,o);}
363 inline Po operator *(DB k,cPo p){return p*k;}
364 inline Po operator /(DB k,cPo p){return conj(p)*k/p.len2();}
365
366 typedef vector<Po> VP;
367
368 struct Line{
369     Po a,b;Line(cPo a=Po(),cPo b=Po()):a(a),b(b){}
370     Line(DB x0,DB y0,DB x1,DB y1):a(Po(x0,y0)),b(Po(x1,y1)){}
371     Line(cLine l):a(l.a),b(l.b){}
372
373     //Ax+By+C=0
374     Line(DB A,DB B,DB C){
375         C=-C;if(!::sgn(A))a=Po(0,C/B),b=Po(1,C/B);
376         else if(!::sgn(B))a=Po(C/A,0),b=Po(C/A,1);
377         else a=Po(0,C/B),b=Po(1,(C-A)/B);
378     }
379
380     void in(){a.in(),b.in();}
381     inline friend istream&operator>>(istream&i,Line& p){return i>>p.a>>p.b;}
382     inline friend ostream&operator<<(ostream&o,Line p){return o<<p.a<<"-"<< p.b;}
383
384     Line operator+(cPo x)const{return Line(a+x,b+x);}
385     Line operator-(cPo x)const{return Line(a-x,b-x);}
386     Line operator*(DB k)const{return Line(a*k,b*k);}
387     Line operator/(DB k)const{return Line(a/k,b/k);}
388
389     Po operator*(cLine)const;
390     Po d()const{return b-a;}DB len2()const{return d().len2();}DB len()const{return d().len();}DB arg()const{return d().arg();}
391
392     int sgn(cPo p)const{return dett(a, b, p);}
393     int sgn(cLine)const;
394
395     bool sameSgn(cPo p1,cPo p2)const{return sgn(p1)==sgn(p2);}
396     void getEquation(DB&K,DB&B)const{
397         K = ::sgn(a.x, b.x) ? (b.y-a.y)/(b.x-a.x) : OO;
398         B = a.y - K*a.x;
399     }
400     void getEquation(DB&A,DB&B,DB&C)const{A=a.y-b.y,B=b.x-a.x,C=det(a, b);}
401
402     Line&push(DB r){ // 正数右手螺旋向里
403         Po v=d()._1().lt()*r;a+=v,b+=v; rTs;
404     }
405 };
406
407 inline DB dot(cLine l1,cLine l2){return dot(l1.d(),l2.d());}
408 inline DB dot(cLine l,cPo p){return dot(l.a,l.b,p);}
409 inline DB dot(cPo p,cLine l){return dot(p,l.a,l.b);}
410 inline DB det(cLine l1,cLine l2){return det(l1.d(),l2.d());}
411 inline DB det(cLine l,cPo p){return det(l.a,l.b,p);}
412 inline DB det(cPo p,cLine l){return det(p,l.a,l.b);}
413 inline DB ang(cLine l0,cLine l1){return ang(l0.d(),l1.d());}
414 inline DB ang(cLine l,cPo p){return ang(l.a,l.b,p);}
415 inline DB ang(cPo p,cLine l){return ang(p,l.a,l.b);}
416
417 inline int Line::sgn(cLine l)const{return dett(Ts, l);}
418 inline Po Line::operator*(cLine l)const{return a+d()*det(a,l)/det(Ts,l);}
419 inline Po operator&(cPo p,cLine l){return l*Line(p,p+l.d().lt());}
420 inline Po operator%(cPo p,cLine l){return p&l*2-p;}
421 inline Line push(Line l, DB r){return l.push(r);}
422
```

```
423
424 struct Seg: public Line{
425     Seg(cPo a=Po(),cPo b=Po()):Line(a,b){}
426     Seg(DB x0,DB y0,DB x1,DB y1):Line(x0,y0,x1,y1){}
427     Seg(cLine l):Line(l){}
428     Seg(const Po &a,DB alpha):Line(a,alpha){}
429     Seg(DB A,DB B,DB C):Line(A,B,C){}
430
431     inline int sgn(cPo p)const;
432     inline int sgn(cLine l)const;
433     inline bool qrt(cSeg l)const;
434     inline int sgn(cSeg l)const;
435 };
436
437  // 不相交-1 相交（不规范）0 相交（规范）1
438
439 inline int Seg::sgn(cPo p)const{return -dott(p,a,b);}
440 inline int Seg::sgn(cLine l)const{return sgn(Ts*l);}
441
442 // quick_rejection_test
443 inline bool Seg::qrt(cSeg l)const{
444     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)&&
445         min(a.y,b.y)<=max(l.a.y,l.b.y)&&min(l.a.y,l.b.y)<=max(a.y,b.y);
446 }
447
448
449 inline int Seg::sgn(cSeg l)const{
450     if (!qrt(l)) return -1;
451
452     /*return
453         (dett(a,b,l.a)*dett(a,b,l.b)<=0 &&
454         dett(l.a,l.b,a)*dett(l.a,l.b,b)<=0)?1:-1;*/
455
456     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);
457     if ((d1^d2)==-2&&(d3^d4)==-2)return 1;
458     return ((!d1&&dott(l.a-a,l.a-b)<=0)||(!d2&&dott(l.b-a,l.b-b)<=0)||
459             (!d3&&dott(a-l.a,a-l.b)<=0)||(!d4&&dott(b-l.a,b-l.b)<=0))?0:-1;
460 }
461
462 //inline DB dist2(cLine l,cPo p){return sqr(fabs(dot(lt(l.d()), p-l.a)))/l.len2();}
463 inline DB dist2(cLine l,cPo p){return sqr(fabs(det(l.d(), p-l.a)))/l.len2();}
464
465 inline DB dist2(cLine l1,cLine l2){return dett(l1,l2)?0:dist2(l1,l2.a);}
466
467 inline DB dist2(cSeg l,cPo p){
468     Po pa = p - l.a, pb = p - l.b;
469     if (dott(l.d(), pa) <= 0) return pa.len2();
470     if (dott(l.d(), pb) >= 0) return pb.len2();
471     return dist2(Line(l), p);
472 }
473
474
475 inline DB dist2(cSeg s,cLine l){
476     Po v1=s.a-l.a,v2=s.b-l.a;DB d1=det(l.d(),v1),d2=det(l.d(),v2);
477     return sgn(d1)!=sgn(d2) ? 0 : sqr(min(fabs(d1), fabs(d2)))/l.len2();
478 }
479 inline DB dist2(cSeg l1,cSeg l2){
480     if (~l1.sgn(l2)) return 0;
481     else return min(dist2(l2,l1.a), dist2(l2,l1.b), dist2(l1,l2.a), dist2(l1,l2.b));
482 }
483 template<class T1, class T2> inline DB dist2(const T1& a, const T2& b){
484     return dist2(b, a);
485 }
486
487 } using namespace CG;//}
488 //}
489
```

```
490
491 /** I/O Accelerator Interface .. **/ //{
492 #define g (c=getchar())
493 #define d isdigit(g)
494 #define p x=x*10+c-'0'
495 #define n x=x*10+'0'-c
496 #define pp l/=10,p
497 #define nn l/=10,n
498 template<class T> inline T& RD(T &x){
499     char c;while(!d);x=c-'0';while(d)p;
500     return x;
501 }
502 template<class T> inline T& RDD(T &x){
503     char c;while(g,c!='-'&&!isdigit(c));
504     if (c=='-'){x='0'-g;while(d)n;}
505     else{x=c-'0';while(d)p;}
506     return x;
507 }
508 inline DB& RF(DB &x){
509     //scanf("%lf", &x);
510     char c;while(g,c!='-'&&c!='.'&&!isdigit(c));
511     if(c=='-')if(g=='.'){x=0;DB l=1;while(d)nn;x*=l;}
512         else{x='0'-c;while(d)n;if(c=='.'){DB l=1;while(d)nn;x*=l;}}
513     else if(c=='.'){x=0;DB l=1;while(d)pp;x*=l;}
514         else{x=c-'0';while(d)p;if(c=='.'){DB l=1;while(d)pp;x*=l;}}
515     return x;
516 }
517 #undef nn
518 #undef pp
519 #undef n
520 #undef p
521 #undef d
522 #undef g
523 inline char* RS(char *s){
524     //gets(s);
525     scanf("%s", s);
526     return s;
527 }
528
529 LL last_ans; int Case; template<class T> inline void OT(const T &x){
530     //printf("Case #%d: ", ++Case);
531     //printf("%lld\n", x);
532     //printf("%.4f\n", x);
533     printf("%d\n", x);
534     //cout << x << endl;
535     //last_ans = x;
536 }
537 //}
538
539
540 //}/* ..... */
541
542 int n;
543
544 int main(){
545
546 #ifndef ONLINE_JUDGE
547     freopen("in.txt", "r", stdin);
548     //freopen("out.txt", "w", stdout);
549 #endif
550
551     Rush{
552
553     }
554 }
```

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)

0.3 例题 (E.g.)

0.3.1 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;
    }
}
```

```
63
64  void Build(int x, int l, int r){
65
66      if (l == r){
67          T[x] = Seg(A[l], l, r);
68      }
69      else {
70          Build(lc), Build(rc);
71          update(x);
72      }
73  }
74
75  Seg Query(int x, int l, int r){
76      if (a <= l && r <= b) return T[x];
77      else {
78          release(x);
79          if (b <= mid) return Query(lc);
80          if (mid < a) return Query(rc);
81          Seg res; update(res, Query(lc), Query(rc));
82          return res;
83      }
84  }
85
86  void Negate(int x, int l, int r){
87      if (a <= l && r <= b){
88          T[x].apply_negative();
89      }
90      else {
91          release(x);
92          if (a <= mid) Negate(lc);
93          if (mid < b) Negate(rc);
94          update(x);
95      }
96  }
97
98  void Negate(int a, int b){
99      int _a = ::a, _b = ::b; ::a = a, ::b = b;
100     Negate(root), ::a = _a, ::b = _b;
101 }
102
103 void Modify(int x, int l, int r){
104     if (l == r){
105         T[x] = Seg(b, l, r);
106     }
107     else {
108         release(x);
109         if (a <= mid) Modify(lc);
110         if (mid < a) Modify(rc);
111         update(x);
112     }
113 }
114
115 int main(){
116
117 #ifndef ONLINE_JUDGE
118     freopen("in.txt", "r", stdin);
119     //freopen("out.txt", "w", stdout);
120 #endif
121
122     REP_1_C(i, RD(n)) RD(A[i]); n = cover_bit(n); Build(root); Rush{
123         if (RD()){ // Query .. .
124             RD(a, b, k); VII op; int res = 0; DO(k){
125                 _Seg cur = Query(root).maxS;
126                 if (!cur.s) break;
127                 res += cur.s, op.PB(MP(cur.l, cur.r));
128
129                 Negate(cur.l, cur.r);
```

```
130                 }
131
132                 ECH(it, op) Negate(it->fi, it->se);
133                 OT(res);
134             }
135             else { // Modify .. .
136                 RD(a, b);
137                 Modify(root);
138             }
139         }
140 }
```

Chapter 1

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。。

```
1  const int N = 100009, M = 2 * N, LM = 18;
2
3  int hd[N], suc[M], to[M], wt[N];
4  int ST[LM][M], st[N], dep[N]; // Euler index ...
5  int n, tt; int T[N], Null;
6
7  const int NN = 20 * N;
8  int l[NN], r[NN], c[NN], d[NN], total;
9  // Chairman tree
10
11 #define lx l[x]
12 #define rx r[x]
13 #define ly l[y]
14 #define ry r[y]
15 #define cx c[x]
16 #define cy c[y]
17
18 #define ml (ll+rr>>1)
19 #define mr (ml+1)
20 #define lc lx, ll, ml
21 #define rc rx, mr, rr
22
23 #define lt lx = ++total, rx = ry, x = lx, y = ly, rr = ml
24 #define rt lx = ly, rx = ++total, x = rx, y = ry, ll = mr
25
26 int Tn;
27
28 int new_node(){
29     ++total; l[total] = r[total] = c[total] = d[total] = 0;
30     return total;
31 }
```

```
32
33  int Insert(int y, int p){
34
35      int x = new_node(), root = x, ll = 0, rr = Tn;
36      c[x] = c[y] + 1, d[x] = d[y] + p;
37
38      while (ll < rr){
39          if (p < mr) lt; else rt;
40          c[x] = c[y] + 1, d[x] = d[y] + p;
41      }
42
43      return root;
44  }
45
46  inline bool elder(int a, int b){
47      return dep[a] < dep[b];
48  }
49
50  inline int lca(int a, int b){
51      int l = st[a], r = st[b];
52      if (l > r) swap(l, r); ++r; int lv = lg2(r-l); //log2(r - l);
53      return min(ST[lv][l], ST[lv][r-(1<<lv)], elder);
54  }
55
56  #define aa to[i^1]
57  #define bb to[i]
58  #define v bb
59  #define ww wt[i/2]
60
61  void dfs(int u = 1){
62      ST[0][st[u] = ++tt] = u;
63      REP_G(i, u) if (!st[v]){
64          dep[v] = dep[u] + 1, T[v] = Insert(T[u], ww);
65          dfs(v);
66          ST[0][++tt] = u;
67      }
68  }
69
70  int kth2(int x, int y, int k){
71
72      int z = lca(x, y);
73      x = T[x], y = T[y], z = T[z];
74      int ll = 0, rr = Tn, t, cc = 0, dd = 0;
75      int D = c[x] + c[y] - 2*c[z], tc, td;
76
77      while (ll < rr){
78          if (ml * (cc + (tc = c[lx] + c[ly] - 2*c[l[z]])) - (dd + (td = d[lx] + d[ly] - 2*d[l[z]])) >= k){
79              x = l[x], y = l[y], z = l[z];
80              rr = ml;
81          }
82          else {
83              x = r[x], y = r[y], z = r[z];
84              cc += tc, dd += td, ll = mr;
85          }
86      }
87
88      if ((k-((cc*ll)-dd))<0) --ll;
89      return ll + (k-((cc*ll)-dd))/D;
90  }
91
92  int kth(int x, int y, int k){
93
94      int z = lca(x, y);
95      x = T[x], y = T[y], z = T[z];
96      int ll = 0, rr = Tn, t;
97
98      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);
        }
    }
}
```

1.0.2 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;
        }
```

```
44          return root;
45      }
46
47      inline int lsum(int x, int p){
48          int res = 0, ll = 1, rr = n;
49          while (p != rr){
50              if (p <= mid) x = lx, rr = mid;
51              else res += c[lx], x = rx, ll = mid + 1;
52          }
53          return res + cx;
54      }
55
56      #undef lx
57      #undef rx
58      #undef ly
59      #undef ry
60      #undef cx
61      #undef cy
62      #undef mid
63
64  } using namespace Fotile_Tree;
65
66  int main(){
67
68  #ifndef ONLINE_JUDGE
69      freopen("in.txt", "r", stdin);
70  #endif
71
72      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;
73      REP_1(i, n) A[i] = lower_bound(B+1, B+m, A[i]) - B;
74
75      DWN_1(i, n, 1){
76          T[i] = Insert(!P[A[i]] ? T[i+1] : Insert(T[i+1], P[A[i]], -1), i, 1);
77          P[A[i]] = i;
78      }
79
80      Rush{
81          int l, r; RD(l, r);
82          OT(lsum(T[l], r));
83      }
84  }
```

Chapter 2

可持久化树堆 (Treap)

```
1  // UVA 12538
2  const int N = int(1e7) + 9, SN = int(1e6) + 9, VN = int(5e4) + 9;
3
4  namespace Treap{
5
6      int c[2][N], sz[N], ww[N], tot; char ch[N], str[SN];
7      int T[VN], _T, tt;
8
9  #define l c[0]
10 #define r c[1]
11 #define lx l[x]
12 #define rx r[x]
13 #define ml (a + b >> 1)
14 #define mr (ml + 1)
15 #define lc a, ml
16 #define rc mr, b
17
18     inline int update(int x){
19         sz[x] = sz[lx] + 1 + sz[rx];
20         return x;
21     }
22
23     inline int new_node(char chx){
24         int x = ++tot;
25         lx = rx = 0, ww[x] = rand(), sz[x] = 1, ch[x] = chx;
26         return x;
27     }
28
29     inline int new_node(int xx){
30         int x = ++tot;
31         lx = l[xx], rx = r[xx], ww[x] = ww[xx], sz[x] = sz[xx], ch[x] = ch[xx];
32         return x;
33     }
34
35     int merge(int a, int b){
36         if(!a||!b) return a|b;
37
38         if(ww[a] > ww[b]){
39             a = new_node(a), r[a] = merge(r[a], b);
40             return update(a);
41         }
42         else{
43             b = new_node(b), l[b] = merge(a, l[b]);
44             return update(b);
45         }
46     }
47
48     void split(int x, int p, int &a, int &b){
49         if(!p) a = 0, b = x; else if(sz[x] == p) a = x, b = 0;
50         else{
```

```
51              x = new_node(x);
52              if(p <= sz[lx]) split(lx, p, a, b), lx = b, b = x;
53              else split(rx, p-sz[lx]-1, a, b), rx = a, a = x;
54              update(x);
55          }
56      }
57
58      int build(int a = 0,int b = strlen(str)){
59          if (a >= b) return 0;
60          int x = new_node(str[ml]);
61          lx = build(lc), rx = build(rc);
62          update(x);
63          return x;
64      }
65
66      void print(int x,int a,int b){
67          if (!x) return;
68          if (a <= sz[lx]) print(lx, a, b); a -= sz[lx]+1, b -= sz[lx]+1;
69          if (a <= 0 && 0 < b) putchar(ch[x]), _T += ch[x] == 'c';
70          if (1 < b) print(rx, a, b);
71      }
72  } using namespace Treap;
73
74  int main(){
75
76  #ifndef ONLINE_JUDGE
77      freopen("in.txt", "r", stdin);
78      //freopen("print.txt", "w", stdprint);
79  #endif
80
81      int t, s, n, a, b, _; Rush switch(RD()){
82          case 1:
83              RD(s)-=_T, RS(str);
84              split(T[tt], s, a, b);
85              T[++tt] = merge(merge(a, build()), b);
86              break;
87          case 2:
88              RD(s, n), s-=_T,n-=_T;
89              split(T[tt], s-1, a, b), split(b, n, _, b);
90              T[++tt] = merge(a, b);
91              break;
92          default:
93              RD(t, s, n), t-=_T,s-=_T,n-=_T;
94              print(T[t], s, s+n), puts("");
95      }
96  }
```

Chapter 3

替罪羊 (Scapegoat)

Chapter 4

KD-树 (KD-Tree)

4.0.3 区间合并

Chapter 5

动态 KD-树 (Dynamic KD-Tree)

5.0.4 区间合并

Chapter 6

伸展树 (Splay)

6.1 例题 (E.g.)

6.1.1 SPOJ SEQ2

```
1  const int N = 500009;
2
3  struct node{
4
5      static node *NIL, *rt, *tp; node *c[2], *p;
6      int sz, ky, ss, ls, rs, ms, bj;
7
8  #define NIL node::NIL
9  #define rt node::rt
10 #define l c[0]
11 #define r c[1]
12 #define lx x->l
13 #define rx x->r
14 #define px x->p
15 #define ly y->l
16 #define ry y->r
17 #define py y->p
18
19     inline void reset(int v){l=r=p=NIL,ky=v,bj=0;}
20     inline void rev(){bj^=1,swap(l,r),swap(ls,rs);}
21     inline void sss(){bj=2,ss=sz*ky,ms=ls=rs=ky<0?ky:ss;}
22
23     inline void upd(){
24         assert(this != NIL);
25         sz = l->sz + 1 + r->sz, ss = l->ss + ky + r->ss;
26         ls = max(l->ls, l->ss + ky + max(0, r->ls));
27         rs = max(r->rs, r->ss + ky + max(0, l->rs));
28         ms = max(l->ms, max(0, l->rs) + ky + max(0, r->ls), r->ms);
29     }
30     inline void rls(){
31         assert(this != NIL);
32         if (bj){
33             if (bj&1) l->rev(), r->rev();
34             if (bj&2) l->ky = r->ky = ky, l->sss(), r->sss();
35             bj = 0;
36         }
37     }
38     inline int sgn(){return p->r==this;}
39     inline void setc(int d,node*x){c[d]=x,px=this;}
40     inline void setl(node*x){setc(0,x);}
41     inline void setr(node*x){setc(1,x);}
42
43     inline void rot(int d){
44         node*y=p,*z=py;z->setc(y->sgn(),this);
45         y->setc(d,c[!d]),setc(!d,y),y->upd();
46     }
```

```
47      inline void rot(){rot(sgn());}
48
49      //inline void fix(){if (~sgn()) p->fix(); rls();}
50  /*
51      inline node* splay(node*t){
52          while (p!=t) rot(); upd();
53          return this;
54      }
55  */
56      inline node*splay(node*t){
57          int a,b;while(p!=t){
58              if (p->p==t){rot();break;}
59              else a=sgn(),b=p->sgn(),(a^b?this:p)->rot(a),rot(b);
60          }
61          upd();if (t==NIL)rt=this;
62          return this;
63      }
64
65
66      void rcc();
67
68      void inorder(){
69          if (this == NIL) return;
70          rls(); l->inorder();
71          printf(ky == -INF ? "$ " : "%d ", ky);
72          r->inorder();
73      }
74
75  } *NIL, *rt, TPool[N], *TStack[N];
76  int tp, ts;
77
78  #define mid (a + b >> 1)
79  #define lc a, mid-1
80  #define rc mid+1, b
81
82  node *select(int k, node*t=NIL){
83      node *x = rt; while (x->rls(), lx->sz != k){
84          if (k < lx->sz) x = lx;
85          else k -= lx->sz+1, x = rx;
86      }
87      return x->splay(t);
88  }
89
90  node *select(int a, int b){
91      return select(a-1, select(b+1))->r;
92  }
93
94  inline void node::rcc(){
95      if (this == NIL) return;
96      l->rcc(), r->rcc();
97      TStack[++ts] = this;
98  }
99
100 inline node *new_node(int v){
101     node *x = ts ? TStack[ts--] : &TPool[++tp];
102     x->reset(v);
103     return x;
104 }
105
106 int A[N], s, n, m; inline node *Build(int a = 1, int b = n){
107     if (a > b) return NIL;
108     node *x = new_node(A[mid]);
109     x->setl(Build(lc)), x->setr(Build(rc)), x->upd();
110     return x;
111 }
112
113 int main(){
```

```
114
115 #ifndef ONLINE_JUDGE
116     freopen("in.txt", "r", stdin);
117     //freopen("out.txt", "w", stdout);
118 #endif
119
120     NIL = &TPool[0], A[0] = -INF;
121
122     Rush{
123
124         tp = ts = 0; RD(n, m); REP_1(i, n) RDD(A[i]); A[n+1] = -INF; rt = Build(0, n+1);
125         node *x, *y, *z; char cmd[10]; DO(m){
126
127             switch(RS(cmd)[0]){
128                 case 'I': // Insert ... .
129                     RD(s, n); REP_1(i, n) RDD(A[i]); y = select(s, z = select(s+1)), x = Build();
130                     y->setr(x), y->upd(), z->upd();
131                     break;
132                 case 'D': // Delete
133                     RD(s, n); y = select(s-1, z = select(s+n)), x = ry;
134                     x->rcc(), ry = NIL, y->upd(), z->upd();
135                     break;
136                 case 'R': // Reverse ..
137                     RD(s, n); y = select(s-1, z = select(s+n)), x = ry;
138                     x->rev(), y->upd(), z->upd();
139                     break;
140                 case 'M': // Make_Same // Max_Sum
141                     if (cmd[2] == 'X') {OT(rt->ms); break;}
142                     RD(s, n); y = select(s-1, z = select(s+n)), x = ry;
143                     RDD(x->ky), x->sss(), y->upd(), z->upd();
144                     break;
145                 default: // Get_Sum ..
146                     RD(s, n);
147                     OT(select(s, s+n-1)->ss);
148             }
149         }
150     }
151 }
```

Chapter 7

动态树 (Link-Cut Tree)

7.1 维护路劲信息 (Path)

7.1.1 定义

```
1     static node *NIL; node *c[2], *p;
2     int bj, sz, ky, ss, ls, rs, ms;
3
4  #define NIL node::NIL
5  #define l c[0]
6  #define r c[1]
7  #define lx x->l
8  #define rx x->r
9  #define px x->p
10 #define ly y->l
11 #define ry y->r
12 #define py y->p
```

7.1.2 标记

```
1     inline void reset(int v=0){l=r=p=NIL,bj=0,ky=v;}
2     inline node(int v=0){reset(v);}
3
4     inline void rev(){bj^=1,swap(l, r),swap(ls, rs);}
5     inline void sss(){bj|=2,ss=sz*ky,ms=ls=rs=max(0,ss);}
6
7     inline void upd(){
8         assert(this != NIL);
9         sz = l->sz + 1 + r->sz, ss = l->ss + ky + r->ss;
10        ls = max(l->ls, l->ss + ky + r->ls);
11        rs = max(r->rs, r->ss + ky + l->rs);
12        ms = max(l->ms, l->rs + ky + r->ls, r->ms);
13    }
14
15    inline void rls(){
16        assert(this != NIL);
17        if (bj){
18            if (bj&1) l->rev(), r->rev();
19            if (bj&2) l->ky = r->ky = ky, l->sss(), r->sss();
20            bj = 0;
21        }
22    }
```

MSS
必须取满 k 段。。

```
1     inline void sss(){bj|=2, ss = sz * ky, ms = ls = rs = ky < 0 ? ky : ss;}
2
3     inline void upd(){
4         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]);

    }
```

7.1.3 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());}
```

7.1.4 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;
    }
```

7.1.5 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();
}
```

7.1.6 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;}
```

7.1.7 动态 LCA

7.2 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();
    }
}
```

7.3 例题 (E.g.)

7.3.1 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。

```
1  #include <iostream>
2  #include <cstdio>
3  #include <cstring>
4  #include <cassert>
5  using namespace std;
6  #define REP_1(i, n) for (int i=1;i<=n;++i)
7  #define Rush for(int _____T=RD(); _____T--;)
8
9  /** I/O Accelerator Interface .. **/ //{
10 #define g (c=getchar())
11 #define d isdigit(g)
12 #define p x=x*10+c-'0'
13 template<class T> inline T& RD(T &x){
14     char c;while(!d);x=c-'0';while(d)p;
15     return x;
16 }
17 #undef p
18 #undef d
19 #undef g
20 inline int RD(){int x; return RD(x);}
21 inline char* RS(char *s){
22     scanf("%s", s);
23     return s;
24 }
25 template<class T> inline void OT(const T &x){
26     printf("%d\n", x);
27 }
28 //}
29
30 const int N = int(3e5)+9, M = 2*N;
31 int c[2][N], p[N];
32 int w1[N], w2[N], d0[N]; bool r0[N];
33 #define l c[0]
34 #define r c[1]
35 void reset(int x){
36     l[x]=r[x]=p[x]=0;
37     d0[x]=r0[x]=0;
38 }
39 inline void rev(int x){
40     r0[x]^=1,swap(l[x],r[x]);
41 }
42 inline void inc(int x,int d){
43     if(!x)return;//!
44     w1[x]+=d,w2[x]+=d,d0[x]+=d;
45 }
46 inline void upd(int x){
47     w2[x]=max(max(w2[l[x]],w1[x]),w2[r[x]]);
48 }
49 inline void rls(int x){
50     if (r0[x]){
51         rev(l[x]),rev(r[x]);
52         r0[x]=0;
53     }
54     if (d0[x]){
55         inc(l[x],d0[x]),inc(r[x],d0[x]);
56         d0[x]=0;
57     }
58 }
59 inline int sgn(int x){return l[p[x]]==x?0:r[p[x]]==x?1:-1;}
60 inline void setc(int x, int d, int y){p[c[d][x]=y]=x;}
61 inline void rot(int x, int d){
```

```
62      int y=p[x],z=p[y];if (~sgn(y))setc(z,sgn(y),x);else p[x]=z;
63      setc(y,d,c[!d][x]),setc(x,!d,y),upd(y);
64  }
65  inline void fix(int x){if(~sgn(x))fix(p[x]);rls(x);}
66  inline int splay(int x){
67      fix(x);int a,b,y;while (~(a=sgn(x))){
68          if (~(b=sgn(y=p[x])))rot(a^b?x:y,a),rot(x,b);
69          else rot(x,a);
70      }
71      upd(x);
72      return x;
73  }
74  inline int acs(int _x){
75      int x=_x,y=0;do{
76          splay(x);
77          r[x]=y,upd(x);
78          y=x,x=p[x];
79      }while(x);
80      return splay(_x);
81  }
82
83  inline int lca(int y, int _x){
84      acs(y);int x=_x,z;y=0;do{
85          splay(x); if(!p[x])z=x;
86          r[x]=y,upd(x);
87          y=x,x=p[x];
88      }while(x);
89      splay(_x);
90      return z;
91  }
92
93  inline int rt(int x){for (x=acs(x);rls(x),l[x];x=l[x]);return splay(x);}
94  inline int ert(int x){rev(acs(x));return x;}
95
96  void Link(int x, int y){
97      if (rt(x)==rt(y))puts("-1");
98      else ert(x),p[x]=y;
99      //splay(x),p[x]=y有根树;//
100 }
101 void Cut(int x){
102     p[l[acs(x)]]=0,l[x]=0;//!
103 }
104 void Cut(int x, int y){
105     if (x==y||rt(x)^rt(y))puts("-1");
106     else ert(x),Cut(y);
107 }
108 void Query(int x, int y){
109     if (rt(x)^rt(y))puts("-1");
110     else{ert(x),OT(w2[acs(y)]);}
111 }
112 void Modify(int x, int y, int d){
113     if (rt(x)^rt(y))puts("-1");
114     else{ert(x),inc(acs(y),d);}
115 }
116
117 int hd[N], suc[M], to[M];
118 int n;
119 #define aa to[i^1]
120 #define bb to[i]
121 #define v bb
122 inline void dfs(int u){
123     for(int i=hd[u];i;i=suc[i])if (!p[v]){
124         p[v]=u, dfs(v);
125     }
126 }
127
128 int main(){
```

```
129
130 #ifndef ONLINE_JUDGE
131     freopen("in.txt", "r", stdin);
132     //freopen("out.txt", "w", stdout);
133 #endif
134
135     while (~scanf("%d", &n)){
136
137         REP_1(i, n) reset(i);
138         memset(hd+1, 0, sizeof(int)*n);
139
140         for(int i=2;i<n<<1;){
141             RD(aa),RD(bb);
142             suc[i] = hd[aa], hd[aa] = i++;
143             suc[i] = hd[aa], hd[aa] = i++;
144         }
145
146         REP_1(i, n) RD(w1[i]); p[1]=1,dfs(1),p[1]=0;
147
148         REP_1(i, n) ert(i);
149
150         int a, b, cmd;Rush{
151             RD(cmd),RD(a),RD(b);if (cmd==1) Link(a,b);
152             else if(cmd==2) Cut(a,b);
153             else if(cmd==3) Modify(b,RD(),a);
154             else Query(a,b);
155         }
156         puts("");
157     }
158
159     /*RD(n); char cmd[9]; int a; Rush{
160         RS(cmd); RD(a); if (cmd[0]=='c') Cut(a);
161         else if (cmd[1]=='i') Link(a, RD());
162         else OT(lca(a, RD()));
163     }*/
164 }
```

```
1  const int N = int(3e5)+9, M = 2*N;
2
3  struct node{
4
5      static node* NIL; node *c[2], *p;
6      int w1, w2, d0; bool r0;
7
8  #define NIL node::NIL
9  #define l c[0]
10 #define r c[1]
11 #define lx x->l
12 #define rx x->r
13 #define px x->p
14 #define ly y->l
15 #define ry y->r
16 #define py y->p
17
18     void reset(){
19         l = r = p = NIL;
20         w1 = w2 = d0 = r0 = 0;
21     }
22
23     inline node(){
24         reset();
25     }
26
27     inline void rev(){
28         r0 ^= 1, swap(l, r);
29     }
30
```

```
31      inline void inc(int d){
32          if (this == NIL) return;
33          w1 += d, w2 += d, d0 += d;
34      }
35
36      inline void upd(){
37          w2 = max(l->w2, w1, r->w2);
38      }
39
40      inline void rls(){
41          //if (this == NIL) return;
42          if (r0){
43              l->rev(), r->rev();
44              r0 = 0;
45          }
46          if (d0){
47              l->inc(d0), r->inc(d0);
48              d0 = 0;
49          }
50      }
51
52      // 旋转
53
54      inline int sgn(){return p->l==this?0:p->r==this?1:-1;}
55      inline void setc(int d,node*x){c[d]=x,px=this;}
56
57      inline void rot(int d){
58          node *y = p, *z = py; if (~y->sgn()) z->setc(y->sgn(), this); else p = z;
59          y->setc(!d, c[d]), setc(d, y), y->upd();
60      }
61
62      inline void rot(){rot(!sgn());}
63      inline void zag(){rot(0);}
64      inline void zig(){rot(1);}
65
66      // 伸展
67
68      inline void fix(){if(~sgn()) p->fix(); rls();}
69
70      //*
71      inline node* splay(){
72          fix(); while (~sgn()) rot(); upd();
73          return this;
74      }
75      /*/
76      inline node* splay(){
77          fix(); while (sgn() != -1){
78              node *y = p, *z = py; if (y->sgn() == -1){ rot(); break;}
79              if (z->l == y){
80                  if (y->l == this) y->zig(), zig();
81                  else zag(), zig();
82              }else{
83                  if (y->r == this) y->zag(), zag();
84                  else zig(), zag();
85              }
86          }
87          upd();
88          return this;
89      } /*/
90
91      inline node* acs(){
92          node *x = this, *y = NIL; do{
93              x->splay();
94              rx = y, x->upd();
95              y = x, x = px;
96          } while (x != NIL);
97          return splay();
```

```
98      }
99
100     node* rt(){node* x; for (x = acs(); x->rls(), lx != NIL; x = lx); return x->splay();}
101     node* ert(){acs()->rev(); return this;}
102
103
104     void Link(node *x){
105         if (rt() == x->rt()){
106             puts("-1");
107         }
108         else {
109             ert(), p = x;
110         }
111     }
112
113     void Cut(){
114         acs(); l->p = l = NIL;
115     }
116
117     void Cut(node* x){
118         if (this == x || rt() != x->rt()){
119             puts("-1");
120         }
121         else {
122             ert(), x->Cut();
123         }
124     }
125
126     void Query(node* x){
127         if (rt() != x->rt()){
128             puts("-1");
129         }
130         else {
131
132             x->ert(); OT(acs()->w2);
133
134             /*acs(); node *y = NIL; do{
135                 x->splay(); if (px == NIL) OT(max(rx->w2, x->w1, y->w2));
136                 rx = y, x->upd();
137                 y = x, x = px;
138             } while (x != NIL);*/
139         }
140     }
141
142     void Modify(node *x, int d){
143         if (rt() != x->rt()){
144             puts("-1");
145         }
146         else {
147
148             x->ert(); acs()->inc(d);
149
150             /*acs(); node *y = NIL; do{
151                 x->splay(); if (px == NIL) rx->inc(d), x->w1 += d, y->inc(d);
152                 rx = y, x->upd();
153                 y = x, x = px;
154             } while (x != NIL);*/
155         }
156     }
157 } *NIL, *T[N];
158
159 int hd[N], suc[M], to[M];
160 int n;
161 #define aa to[i^1]
162 #define bb to[i]
163 #define v bb
164 inline void dfs(int u){
```

```
165     REP_G(i, u) if (T[v]->p == NIL){
166         T[v]->p = T[u], dfs(v);
167     }
168 }
169
170 } using namespace LCT;
171
172 int main(){
173
174 #ifndef ONLINE_JUDGE
175     freopen("in.txt", "r", stdin);
176     //freopen("out.txt", "w", stdout);
177 #endif
178
179     NIL = new node(); REP_1(i, N) T[i] = new node();
180
181     while (~scanf("%d", &n)){
182
183         REP_1(i, n) T[i]->reset();
184         memset(hd+1, 0, sizeof(int)*n);
185
186         // Initializing Phase
187         FOR_C(i, 2, n << 1){
188             RD(aa, bb);
189             suc[i] = hd[aa], hd[aa] = i++;
190             suc[i] = hd[aa], hd[aa] = i;
191         }
192
193         REP_1(i, n) RD(T[i]->w1);
194         T[1]->p = T[1], dfs(1), T[1]->p = NIL;
195
196         //Interaction Phase
197         int a, b, cmd; Rush{
198             RD(cmd, a, b); if (cmd == 1) T[a]->Link(T[b]);
199             else if (cmd == 2) T[a]->Cut(T[b]);
200             else if (cmd == 3) T[b]->Modify(T[RD()], a);
201             else T[a]->Query(T[b]);
202         }
203
204         puts("");
205     }
206 }
```

7.3.2 SPOJ QTREE. Query on a tree

```
1  const int N = int(1e4) + 9, M = 2 * N;
2
3  struct node{
4
5      static node *NIL; node *c[2], *p;
6      int w0, w1;
7
8  #define NIL node::NIL
9  #define l c[0]
10 #define r c[1]
11 #define lx x->l
12 #define rx x->r
13 #define px x->p
14 #define ly y->l
15 #define ry y->r
16 #define py y->p
17
18     void reset(int v = 0){
19         l = r = p = NIL;
20         w0 = w1 = v;
```

```
21      }
22
23      node(int v = 0){
24          reset();
25      }
26
27      void upd(){
28          w1 = max(l->w1, w0, r->w1);
29      }
30
31      int sgn(){return p->l==this?0:p->r==this?1:-1;}
32      void setc(int d,node*x){c[d]=x,px=this;}
33
34      void rot(int d){
35          node*y=p,*z=py;if(~y->sgn())z->setc(y->sgn(),this);else p=z;
36          y->setc(d,c[!d]),setc(!d,y),y->upd();
37      }
38
39      void rot(){rot(sgn());}
40
41      node* splay(){
42          int a,b;while(~(a=sgn())){
43              if(~(b=(p->sgn())))(a^b?this:p)->rot(a),rot(b);
44              else rot(a);
45          }
46          upd();
47          return this;
48      }
49
50      node* acs(){
51          node *x = this, *y = NIL; do{
52              x->splay();
53              rx = y, x->upd();
54              y = x, x = px;
55          } while (x != NIL);
56          return splay();
57      }
58
59      void query(node *x){
60          acs(); node *y = NIL; do{
61              x->splay(); if (px == NIL) OT(max(y->w1, rx->w1));
62              rx = y, x->upd();
63              y = x, x = px;
64          } while (x != NIL);
65          splay();
66      }
67      void modify(int w){
68          splay(); w0 = w;
69      }
70  } *NIL, *T[N];
71
72  int hd[N], suc[M], to[M], ww[N], id[N], n;
73  #define aa to[i^1]
74  #define bb to[i]
75  #define w ww[i/2]
76  #define v bb
77
78  inline void dfs(int u){
79      REP_G(i, u) if (T[v]->p == NIL){
80          T[v]->w0 = w, id[i/2] = v, T[v]->p = T[u], dfs(v);
81      }
82  }
83
84  int main(){
85
86  #ifndef ONLINE_JUDGE
87      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);
        }
    }
}
```

7.3.3 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);
    }
```

```
38
39  #define w3 (*s1.rbegin())
40  #define w2 (*s0.rbegin() + _2nd(s0))
41  #define w1 (*s0.rbegin())
42
43      inline void upd(){
44          dd = l->dd + d0 + r->dd; int m0 = max(w0, w1), ml = max(m0, l->rs+d0), mr = max(m0, r->ls);
45          ls = max(l->ls, l->dd+d0+mr), rs = max(ml+r->dd, r->rs);
46          ms = max(l->ms, l->rs+d0+mr, max(w2, w3, w0?-INF:m0), ml+r->ls, r->ms);
47      }
48
49      inline void rls(){
50          /*if (r0){
51              l->rev(), r->rev();
52              r0 = 0;
53          }*/
54      }
55
56      inline int sgn(){return p->l==this?0:p->r==this?1:-1;}
57      inline void setc(int d,node*x){c[d]=x,px=this;}
58
59      inline void rot(int d){
60          node*y=p,*z=py;if(~y->sgn())z->setc(y->sgn(),this);else p=z;
61          y->setc(d,c[!d]),setc(!d,y),y->upd();
62      }
63      inline void rot(){rot(sgn());}
64
65      inline void fix(){if (~sgn()) p->fix(); rls();}
66  /*
67      inline node* splay(){
68          fix();while (~sgn()) rot(); upd();
69          return this;
70      }
71  */
72
73      inline node*splay(){
74          fix();int a,b;while(~(a=sgn())){
75              if(~(b=(p->sgn())))(a^b?this:p)->rot(a),rot(b);
76              else rot(a);
77          }
78          upd();
79          return this;
80      }
81
82      inline node* acs(){
83          node *x = this, *y = NIL; do{
84              x->splay();
85              if (y != NIL) x->s0.erase(x->s0.find(y->ls)), x->s1.erase(x->s1.find(y->ms));
86              if (rx != NIL) x->s0.insert(rx->ls), x->s1.insert(rx->ms);
87              rx = y, x->upd();
88              y = x, x = px;
89          } while (x != NIL);
90          return splay();
91      }
92
93      inline node* rt(){node* x; for (x = acs(); x->rls(), lx != NIL; x = lx); return x->splay();}
94      inline node* ert(){acs()->rev(); return this;}
95
96      void link(node *x){
97          acs(); p = x; x->s0.insert(ls), x->s1.insert(ms); //x->upd();
98      }
99
100     void cut(){
101         acs(); l->p = NIL, l = NIL;
102     }
103
104     void cut(node* x){
```

```
105         ert(), x->cut();
106     }
107
108     void tog(){
109         acs(); w0 = w0 ? 0 : -INF; //upd();
110     }
111
112 } *NIL, *T[N];
113
114 int hd[N], suc[M], to[M], ww[N], n;
115 #define aa to[i^1]
116 #define bb to[i]
117 #define w ww[i/2]
118 #define v bb
119
120 inline void dfs(int u){
121     REP_G(i, u) if (T[v]->p == NIL){
122         T[v]->p = T[u], T[v]->d0 = w, dfs(v);
123         T[u]->s0.insert(T[v]->ls); T[u]->s1.insert(T[v]->ms);
124     }
125     T[u]->upd();
126 }
127
128 } using namespace LCT;
129
130 int main(){
131
132 #ifndef ONLINE_JUDGE
133     freopen("in.txt", "r", stdin);
134     //freopen("out.txt", "w", stdout);
135 #endif
136
137
138     NIL = new node();
139     //REP(i, N) T[i] = new node();
140
141     while (~scanf("%d", &n)){
142
143         //REP_1(i, n) T[i]->reset();
144         FOR_1(i, 0, n) T[i] = new node();
145
146         //fill(hd+1, hd+n+1, 0);
147         FOR_C(i, 2, n << 1){
148             RD(aa, bb); RDD(w);
149             suc[i] = hd[aa], hd[aa] = i++;
150             suc[i] = hd[aa], hd[aa] = i;
151         }
152
153         T[1]->p = T[0]; T[1]->d0 = 0; dfs(1); T[1]->p = NIL;
154
155         Rush{
156             switch(RC()){
157                 case 'A':
158                     T[1]->splay();
159                     if (T[1]->ms < 0) puts("They have disappeared.");
160                     else OT(T[1]->ms);
161                     break;
162                 default:
163                     T[RD()]->tog();
164             }
165         }
166     }
167 }
```

7.3.4 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;
    }
```

7.3.5 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();
    }
```

```
25
26      void rev(){
27          r0 ^= 1, swap(l, r);
28      }
29
30      void inc(int d){
31          if (this == NIL) return;
32          w0 += d, d0 += d;
33      }
34
35      void upd(){
36      }
37
38      void rls(){
39          if (r0){
40              l->rev(), r->rev();
41              r0 = 0;
42          }
43          if (d0){
44              l->inc(d0), r->inc(d0);
45              d0 = 0;
46          }
47      }
48
49      int sgn(){return p->l==this?0:p->r==this?1:-1;}
50      void setc(int d,node*x){c[d]=x,px=this;}
51
52      void rot(int d){
53          node*y=p,*z=py;if(~y->sgn())z->setc(y->sgn(),this);else p=z;
54          y->setc(d,c[!d]),setc(!d,y),y->upd();
55      }
56
57      void rot(){rot(sgn());}
58
59      void fix(){if (~sgn()) p->fix(); rls();}
60
61      node* splay(){
62          fix();int a,b;while(~(a=sgn())){
63              if(~(b=(p->sgn())))(a^b?this:p)->rot(a),rot(b);
64              else rot(a);
65          }
66          upd();
67          return this;
68      }
69
70      node* acs(){
71          node *x = this, *y = NIL; do{
72              x->splay();
73              rx = y, x->upd();
74              y = x, x = px;
75          } while (x != NIL);
76          return splay();
77      }
78
79      node* rt(){node* x; for (x = acs(); x->rls(), lx != NIL; x = lx); return x->splay();}
80      node* ert(){acs()->rev(); return this;}
81
82      void link(node *x){
83          splay(); x->acs(); p = x; x->inc(w0);
84      }
85
86      void cut(){
87          acs(); l->inc(-w0); l->p = NIL, l = NIL;
88      }
89
90      int query(){
91          node *x = rt()->r; for(; x->rls(), lx != NIL; x = lx);
```

```
92          return x->w0;
93      }
94  } *NIL, *T[2][N]; int col[N], fa[N];
95
96  #define TT(u) T[col[u]][u]
97
98  void Toggle(int u){
99      TT(u)->cut(); col[u] ^= 1;
100     TT(u)->link(T[col[u]][fa[u]]);
101 }
102
103 int hd[N], suc[M], to[M], n;
104 #define aa to[i^1]
105 #define bb to[i]
106 #define v bb
107
108 inline void dfs(int u){
109     REP_G(i, u) if (TT(v)->p == NIL){
110         TT(v)->p = T[col[v]][fa[v] = u], dfs(v);
111         T[col[v]][u]->w0 += TT(v)->w0;
112     }
113 }
114
115
116 int main(){
117
118 #ifndef ONLINE_JUDGE
119     freopen("in.txt", "r", stdin);
120     //freopen("out.txt", "w", stdout);
121 #endif
122
123     NIL = new node();
124
125     while (~scanf("%d", &n)){
126
127         FOR_1(i, 0, n) T[0][i] = new node(), T[1][i] = new node();
128
129         for (int i=2;i<n<<1;){
130             RD(aa, bb);
131             suc[i] = hd[aa], hd[aa] = i++;
132             suc[i] = hd[aa], hd[aa] = i++;
133         }
134
135         TT(1)->p = T[col[1]][0]; dfs(1);
136
137         int _, u; Rush{
138             switch(RD(_, u)){
139                 case 0:
140                     OT(TT(u)->query());
141                     break;
142                 default:
143                     Toggle(u);
144             }
145         }
146     }
147 }
```

7.3.6 SPOJ QTREE7. Query on a tree VII

```
1  const int N = int(1e5) + 9, M = 2 * N;
2
3  namespace LCT{
4
5  struct node{
6
```

```
7       static node *NIL; node *c[2], *p; multiset<int> s;
8       bool r0; int w0, w1;
9
10  #define NIL node::NIL
11  #define l c[0]
12  #define r c[1]
13  #define lx x->l
14  #define rx x->r
15  #define px x->p
16  #define ly y->l
17  #define ry y->r
18  #define py y->p
19
20      void reset(){
21          l = r = p = NIL; CLR(s); s.insert(-INF);
22          r0 = 0; w0 = w1 = -INF;
23      }
24
25      node(){
26          reset();
27      }
28
29      void rev(){
30          r0 ^= 1; swap(l, r);
31      }
32
33      void upd(){
34          w1 = max(l->w1, w0, r->w1, *s.rbegin());
35      }
36
37      void rls(){
38          if (r0){
39              l->rev(), r->rev();
40              r0 = 0;
41          }
42      }
43
44      int sgn(){return p->l==this?0:p->r==this?1:-1;}
45      void setc(int d,node*x){c[d]=x,px=this;}
46
47      void rot(int d){
48          node*y=p,*z=py;if(~y->sgn())z->setc(y->sgn(),this);else p=z;
49          y->setc(d,c[!d]),setc(!d,y),y->upd();
50      }
51      void rot(){rot(sgn());}
52
53      void fix(){if (~sgn()) p->fix(); rls();}
54
55      node* splay(){
56          fix();int a,b;while(~(a=sgn())){
57              if(~(b=(p->sgn())))(a^b?this:p)->rot(a),rot(b);
58              else rot(a);
59          }
60          upd();
61          return this;
62      }
63
64      node* acs(){
65          node *x = this, *y = NIL; do{
66              x->splay();
67              if (y != NIL) x->s.erase(x->s.find(y->w1));
68              if (rx != NIL) x->s.insert(rx->w1);
69              rx = y, x->upd();
70              y = x, x = px;
71          } while (x != NIL);
72          return splay();
73      }
```

```
74
75      node* rt(){node* x; for (x = acs(); x->rls(), lx != NIL; x = lx); return x->splay();}
76      node* ert(){acs()->rev(); return this;}
77
78      int query(){
79          node *x = rt()->r; //for(; x->rls(), lx != NIL; x = lx);
80          return x->w1;
81      }
82
83      void modify(int w){
84          acs(); w0 = w;
85      }
86
87      void cut(){
88          acs(); l->p = NIL, l = NIL;
89      }
90
91      void link(node *x){
92          splay(); x->acs();
93          p = x; x->r = this;
94      }
95
96  } *NIL, *T[2][N]; int col[N], fa[N];
97  #define TT(u) T[col[u]][u]
98
99  int hd[N], suc[M], to[M], n;
100 #define aa to[i^1]
101 #define bb to[i]
102 #define v bb
103
104 inline void dfs(int u){
105     REP_G(i, u) if (TT(v)->p == NIL){
106         TT(v)->p = T[col[v]][fa[v] = u], dfs(v);
107         T[col[v]][u]->s.insert(TT(v)->w1);
108     }
109     TT(u)->upd();
110 }
111
112 inline void Toggle(int u){
113     TT(u)->cut(); col[u] ^= 1;
114     TT(u)->link(T[col[u]][fa[u]]);
115 }
116
117 inline void Modify(int u, int w){
118     T[0][u]->modify(w), T[1][u]->modify(w);
119 }
120
121 } using namespace LCT;
122
123 int main(){
124
125 #ifndef ONLINE_JUDGE
126     freopen("in.txt", "r", stdin);
127     //freopen("out.txt", "w", stdout);
128 #endif
129
130     NIL = new node();
131
132     REP(i, N) T[0][i] = new node(), T[1][i] = new node();
133
134     while (~scanf("%d", &n)){
135
136         FOR_1(i, 0, n) T[0][i]->reset(), T[1][i]->reset();
137
138         RST(hd); FOR_C(i, 2, n << 1){
139             RD(aa, bb);
140             suc[i] = hd[aa], hd[aa] = i++;
```

```
141             suc[i] = hd[aa], hd[aa] = i;
142         }
143
144         REP_1(i, n) RD(col[i]);
145         REP_1(i, n) T[1][i]->w0 = RDD(T[0][i]->w0);
146
147         TT(1)->p = T[col[1]][0]; dfs(1);
148         T[col[1]][0]->s.insert(TT(1)->w1);
149
150          int _, u; Rush{
151             switch(RD(_, u)){
152                 case 0:
153                     OT(TT(u)->query());
154                     break;
155                 case 1:
156                     Toggle(u);
157                     break;
158                 default:
159                     Modify(u, RDD());
160             }
161         }
162     }
163 }
```

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 8

常见模型

8.1 背包问题 (Knacpack)

8.2 最长不降子序列 (LIS)

```
1   template<class T> int LIS(int n, T* a){
2       VI b; b.PB(a[0]); FOR(i, 1, n){
3           if (b.back() < a[i]) b.PB(a[i]);
4           else {
5               b[lower_bound(ALL(b), a[i]) - b.begin()] = a[i];
6           }
7       }
8       return SZ(b);
9   }
10
11  //template<class T, class C = ___typeof less<T>()> int LIS(int n, T* a, C cmp = less<T>()){
12  template<class T, class C> int LIS(int n, T* a, C cmp){
13      vector<T> b; b.PB(a[0]); FOR(i, 1, n){
14          if (cmp(b.back(), a[i])) b.PB(a[i]);
15          else {
16              b[lower_bound(ALL(b), a[i], cmp) - b.begin()] = a[i];
17          }
18      }
19      return SZ(b);
20  }
21
22  template<class T> int LISS(int n, T* a, int* pre, int& lst){
23      VI b; b.PB(0); pre[0] = -1; FOR(i, 1, n){
24          if (a[b.back()] < a[i]) pre[i] = b.back(), b.PB(i);
25          else {
26              int l = 0, r = SZ(b); while (l < r){
27                  int m = l + r >> 1;
28                  if (a[b[m]] < a[i]) l = m + 1;
29                  else r = m;
30              }
31              pre[i] = !r ? -1 : b[r-1];
32              b[r] = i;
33          }
34      }
35      lst = b.back();
36      return SZ(b);
37  }
38
39  //template<class T, class C = ___typeof less<T>()> int LISS(int n, T* a, int* pre, int& lst, C cmp = less<T>()){
40  template<class T, class C> int LISS(int n, T* a, int* pre, int& lst, C cmp){
41      VI b; b.PB(0); pre[0] = -1; FOR(i, 1, n){
42          if (cmp(a[b.back()], a[i])) pre[i] = b.back(), b.PB(i);
43          else {
44              int l = 0, r = SZ(b); while (l < r){
45                  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);
    }
```

8.3 最长公共子序列 (LCS)

8.4 例题 (E.g.)

8.4.1 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;
```

```
38              REP(t, 2) checkMin(f, dp[l][r][t]);
39          }
40
41          OT(s + f);
42      }
43  }
```

8.4.2 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。。。我们初始先求 G2nm。。。那么每次枚举 offset 。。就相当于删除该图的最上面一层。。
。。。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。。

```
1   const int dx[] = {0, -1, -1};
2   const int dy[] = {-1, -1, 0};
3
4   const int N = 1509;
5   char A[2*N], B[N]; int dp[2*N][N], p[2*N][N];
6   int n, m;
7
8   int lcs(int o){
9       int i = n + o, j = m, d, res = 0;
10      while (i != o && j){
11          if ((d = p[i][j]) == 1) ++res;
12          i += dx[d], j += dy[d];
13      }
14      return res;
15  }
16
17  void reroot(int o){
18      int i = o, j = 1;
19
20      while(j <= m && !p[i][j]) ++j; if (j > m) return;
21      p[i++][j] = 0;
22
23      while (i <= 2*n && j < m){
24          if (p[i][j] == 2){
25              p[i++][j] = 0;
26          }
27          else if (p[i][j+1] == 1){
28              p[i++][++j] = 0;
29          }
30          else{
31              ++j;
```

```
32          }
33      }
34
35      while (i <= 2*n && p[i][j] == 2) p[i++][j] = 0;
36  }
37
38  int clcs(){
39
40      REP_2_1(i, j, 2*n, m){
41          if (A[i] == B[j]) dp[i][j] = dp[i-1][j-1] + 1;
42          else dp[i][j] = max(dp[i-1][j], dp[i][j-1]);
43          if (dp[i][j] == dp[i][j-1]) p[i][j] = 0;
44          else if (A[i] == B[j] && dp[i][j] == dp[i-1][j-1] + 1) p[i][j] = 1;
45          else p[i][j] = 2;
46      }
47
48      int res = 0; REP(i, n){
49          checkMax(res, lcs(i));
50          reroot(i);
51      }
52      return res;
53  }
54
55
56  int main(){
57
58  #ifndef ONLINE_JUDGE
59      freopen("in.txt", "r", stdin);
60      //freopen("out.txt", "w", stdout);
61  #endif
62
63      while (~scanf("%s %s", A+1, B+1)){
64          n = strlen(A+1), m = strlen(B+1);
65          REP_1(i, n) A[n+i] = A[i]; A[2*n+1] = 0;
66          int res = clcs(); reverse(B+1, B+m+1); checkMax(res, clcs());
67          OT(2*res);
68      }
69  }
```

8.4.3 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

```
1   const int N = int(7e4) + 9;
2
3   uint dp[2][(N>>5)+9]; int p[N]; bool rt[N];
4   int A[N], n, s; PII B[N];
5
6   int main(){
7
8   #ifndef ONLINE_JUDGE
9       freopen("in.txt", "r", stdin);
10      //freopen("print.txt", "w", stdprint);
11  #endif
12
13      RD(n, s); REP_1(i, n) B[i] = MP(RD(A[i]), i);
14      sort(B+1, B+n+1, greater<PII>());
15      s -= B[1].fi; if (s < 0){puts("-1"); exit(0);}
16
17      int nn = s/32 + 1; uint *cur = dp[0], *prv = dp[1]; cur[0] = 1;
```

```
18
19      FOR_1(i, 2, n){
20
21          swap(cur, prv); memcpy(cur, prv, sizeof(int)*nn);
22
23          int o1 = B[i].fi >> 5, o2 = B[i].fi & 31;
24
25          REP(ii, nn-o1){
26              cur[ii+o1] |= prv[ii] << o2;
27              if (o2) cur[ii+o1+1] |= prv[ii] >> (32-o2);
28          }
29
30          REP(ii, nn){
31              for (uint s=cur[ii]^prv[ii];s;s^=low_bit(s)){
32                  uint j = low_idx(s);
33                  p[(ii<<5)+j] = B[i].se;
34              }
35          }
36
37          if (p[s]) break;
38      }
39
40      if (s && !p[s]){puts("-1");exit(0);} do{
41          rt[p[s]] = 1;
42      }while (s -= A[p[s]]);
43
44      RST(p); int ii = B[1].se; FOR_1(i, 2, n) if (!rt[B[i].se]){
45          p[ii] = B[i].se, ii = B[i].se;
46      }
47
48      REP_1(i, n){
49          if (p[i]) printf("%d 1 %d\n", A[i]-A[p[i]], p[i]);
50          else printf("%d 0\n", A[i]);
51      }
52  }
```

Chapter 9

数位

9.1 例题 (E.g.)

9.1.1 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)

```
1  const int N = 20;
2  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，本身。。。%7%7
3  int Pow10[N]; int a[N], n;
4
5  #define v0 n-1, _7||i==7, (s+i)%7, (m*10+i)%7, 0
6  #define v1 n-1, _7||i==7, (s+i)%7, (m*10+i)%7, 1
7
8  int f0(int n, bool _7, int s, int m, bool b){
9      if (n<0) return _7 || !s || !m;
10     if (b){
11         int res = 0; int up = a[n], i;
12         REP_N(i, up) INC(res, f0(v0));
13         INC(res, f0(v1));
14         return res;
15     }
16     else {
17         int &res = F0[n][_7][s][m];
18         if (res == -1){
19             res = 0; int up = 10, i;
20             REP_N(i, up) INC(res, f0(v0));
21         }
22         return res;
23     }
24 }
25
26 #define x pdt(Pow10[n], i)
27
28 int f1(int n, bool _7, int s, int m, bool b){
29     if (n<0) return 0;
30     if (b){
31         int res = 0; int up = a[n], i;
32         REP_N(i, up) INC(res, sum(f1(v0), pdt(f0(v0), x)));
33         INC(res, sum(f1(v1), pdt(f0(v1), x)));
34         return res;
```

```
35      }
36      else {
37          int &res = F1[n][_7][s][m];
38          if (res == -1){
39              res = 0; int up = 10, i;
40              REP_N(i, up) INC(res, sum(f1(v0), pdt(f0(v0), x)));
41          }
42          return res;
43      }
44  }
45
46  int f2(int n, bool _7, int s, int m, bool b){
47      if (n<0) return 0;
48      if (b){
49          int res = 0; int up = a[n], i;
50          REP_N(i, up) INC(res, sum(f2(v0), pdt(f1(v0), x, 2), pdt(f0(v0), x, x)));
51          INC(res, sum(f2(v1), pdt(f1(v1), x, 2), pdt(f0(v1), x, x)));
52          return res;
53      }
54      else {
55          int &res = F2[n][_7][s][m];
56          if (res == -1){
57              res = 0; int up = 10, i;
58              REP_N(i, up) INC(res, sum(f2(v0), pdt(f1(v0), x, 2), pdt(f0(v0), x, x)));
59          }
60          return res;
61      }
62  }
63
64  #undef x
65
66  int s2(LL x){
67      int a = x % MOD, b = (x+1) % MOD, c = (2*x+1) % MOD;
68      return pdt(a,b,c,_I(6));
69  }
70
71  int f(LL x){
72      if (!x) return 0;
73      n = 0; int s = s2(x); while (x) a[n++] = x % 10, x /= 10;
74      return dff(s, f2(n-1, 0, 0, 0, 1));
75  }
76
77  int main(){
78
79  #ifndef ONLINE_JUDGE
80      freopen("in.txt", "r", stdin);
81      //freopen("out.txt", "w", stdout);
82  #endif
83
84
85      Pow10[0] = 1; FOR(i, 1, N) Pow10[i] = pdt(Pow10[i-1], 10);
86
87      FLC(F0, F1, F2, -1);
88
89      Rush{
90          LL l, r; RD(l, r);
91          OT(dff(f(r), f(l-1)));
92      }
93  }
```

9.1.2 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);
    }
```

```
55 }
```

Chapter 10

状压

10.1 例题 (E.g.)

10.1.1 Game with Strings

简述 (Brief description)

...

分析 (Analysis)

```
1  const int N = 20, C = 26;
2
3  int adj[2*N-1][C]; char str[N][N+1];
4  int dp[2*N-1][1<<N];
5  int n;
6
7  #define ss (s&adj[k][cc])
8
9  int f(int k, int s, int c){
10     int &res = dp[k][s];
11     if (res == INF){
12         if (k == 2*(n-1)) res = 0;
13         else {
14             if (k&1){
15                 res = -INF; s |= s << 1;
16                 REP(cc, C) if (ss){
17                     checkMax(res, f(k+1, ss, cc));
18                 }
19             }
20             else{
21                 res = INF; s |= s << 1;
22                 REP(cc, C) if (ss){
23                     checkMin(res, f(k+1, ss, cc));
24                 }
25             }
26         }
27         if (c == 0) ++res; else if (c == 1) --res;
28         //cout << k << " " << s << " " << c << " " << res << endl;
29     }
30     return res;
31 }
32
33 int main(){
34
35 #ifndef ONLINE_JUDGE
36     freopen("in.txt", "r", stdin);
37     //freopen("out.txt", "w", stdout);
38 #endif
39
40     REP_C(i, RD(n)) RS(str[i]);
```

```
41
42      REP_2(i, j, n, n) if (i || j){
43          adj[i+j-1][str[i][j]-'a'] |= _1(i);
44      }
45
46      FLC(dp, 0x3f); int res = f(0, 1, str[0][0]-'a');
47      puts(res ? (res > 0 ? "FIRST" : "SECOND") : "DRAW");
48  }
```

Chapter 11

组合

11.0.2 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。。。

```
1
2
3
4
5
6
7   //}/* ..... */
8
9   const int N = 1009;
10
11  Int Fact[N], Factt[N]; Int Binom(int n, int m){
12      return Fact[n] * Factt[m] * Factt[n-m];
13  }
14
15  bool le[N][N]; VI adj[N];
16  Int f[N][N], fu[N]; int sz[N];
17  int n;
18
19  #define v (*it)
20  #define b (fu[i-j]*Binom(i,j)*Binom(sz[u]-1-i,sz[v]-j))
21
22  void dfs(int u, int p = -1) {
```

```
23
24  sz[u] = 1; ECH(it, adj[u]) if (v != p){
25      dfs(v, u), sz[u] += sz[v];
26  }
27
28  REP(i, sz[u]) f[u][i] = fu[i] = 0; sz[u] = f[u][0] = 1;
29
30      ECH(it, adj[u]) if (v != p){
31
32          REP(i, sz[u]) fu[i] = f[u][i], f[u][i] = 0; sz[u] += sz[v];
33
34          if (le[u][v]){
35              REP(i, sz[u]) REP(j, min(sz[v], i+1))
36                  f[u][i] += (f[v][sz[v]] - f[v][j]) * b;
37          }
38          else{
39              REP(i, sz[u]) REP_1(j, min(sz[v], i))
40                  f[u][i] += f[v][j] * b;
41          }
42      }
43
44      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];
45  }
46
47  int main() {
48
49      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;
50
51      //freopen("in.txt", "r", stdin);
52  freopen("permutations.txt","r",stdin);
53  freopen("out2.txt","w",stdout);
54
55      Rush{
56
57          REP_C(i, RD(n)) CLR(adj[i]);
58
59          int x, y; char c; DO(n-1){
60              RD(x), RC(c), RD(y);
61              le[x][y] = c == '<';
62              le[y][x] = c == '>';
63              adj[x].PB(y), adj[y].PB(x);
64          }
65
66          dfs(0); OT(f[0][n]);
67      }
68  }
```

Part III

状态空间搜索 (State Space Search)

11.1 补充

11.1.1 最大团

Part IV

图论 (Graph Theory)

11.1.2 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;
                }
```

```
59              }
60          }
61          else {
62              checkMin(low[u], dfn[v]);
63          }
64      }
65  }
66
67  #undef v
68  #define v (*it)
69
70  namespace DSU{
71      int P[N];
72
73      int Find(int x){
74          return P[x] == x ? x : P[x] = Find(P[x]);
75      }
76      void Init(){
77          REP_1(i, nn) P[i] = i;
78      }
79  } using namespace DSU;
80
81
82  void tarjan_lca(int u, int p = -1){
83
84      dep[u] += cut[u];
85
86      ECH(it, adj[u]) if (v != p){
87          dep[v] = dep[u], tarjan_lca(v, u);
88          P[Find(v)] = u;
89      }
90
91      vis[u] = 1;
92
93  #undef v
94  #define v (*it).fi
95  #define id (*it).se
96
97      ECH(it, lca[u]) if (vis[v]){
98          int z = P[Find(v)];
99          ans[id] = dep[u] + dep[v] - 2*dep[z]+cut[z];
100     }
101 }
102
103 #undef id
104 #undef v
105 #define v bb
106
107 int main(){
108
109 #ifndef ONLINE_JUDGE
110     freopen("in.txt", "r", stdin);
111     //freopen("out.txt", "w", stdout);
112 #endif
113
114     while (RD(n, m)){
115
116         RST(hd); FOR_1_C(i, 2, m << 1){
117             RD(aa, bb);
118             hd[aa] = prd[suc[i] = hd[aa]] = i, ++i;
119             hd[aa] = prd[suc[i] = hd[aa]] = i;
120         }
121
122         RST(v_bj, vis), tt = nn = 0; REP_1(i, n) if (!vis[i]){
123             CLR(v_sta, e_sta);
124             tarjan_bcc(i);
125         }
```

```
126
127         REP_1(i, nn) CLR(lca[i]); REP_C(i, RD(q)){
128             int x = e_bj[RD()], y = e_bj[RD()];
129             if (x == y) ans[i] = 0;
130             else {
131                 lca[x].PB(MP(y, i));
132                 lca[y].PB(MP(x, i));
133             }
134         }
135
136         Init(); RST(vis, dep); REP_1(i, nn) if (!vis[i]) tarjan_lca(i);
137         REP(i, q) OT(ans[i]);
138     }
139 }
```

11.1.3 k-联通分量

```
1   const int N = 109;
2  int C[N][N], CC[N][N], cut[N], prd[N], suc[N], tmp[N];
3  int n, m, nn, n0, s, t, tt; int cc, K;
4
5  int PP[N], P[N];
6
7  inline void Make(int x){P[x] = x;}
8  inline int Find(int x){return x == P[x] ? x : P[x] = Find(P[x]);}
9  inline void Unionn(int x, int y){P[x] = y;}
10 inline void Union(int x, int y){int xx = Find(x), yy = Find(y); Unionn(xx, yy);}
11
12 inline void del(int x){prd[suc[prd[x]] = suc[x]] = prd[x];}
13 inline void rsm(int x){prd[suc[suc[prd[x]] = x]] = x;}
14
15 #define hd suc[0]
16
17 void Extract(int &s){
18     s = hd; REP_L(i, suc[s], suc) if (cut[i] > cut[s]) s = i;
19 }
20
21 void Prim(){
22     REP_L(i, hd, suc) cut[i] = 0; tt = 0; DO(nn-1){
23         Extract(s); del(s); tmp[tt++] = s;
24         REP_L(i, hd, suc) cut[i] += CC[s][i];
25     }
26     Extract(t); DWN(i, tt, 0) rsm(tmp[i]); del(t);
27 }
28
29 bool Stoer_Wagner(VI& I, VI& A, VI& B){
30
31     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]];
32     nn = SZ(I); I.PB(0); REP(i, nn) suc[I[i]] = I[i+1], prd[I[i+1]] = I[i], Make(I[i]);
33     prd[hd = I[0]] = 0; I.pop_back(); DO(nn-2){
34         Prim(), --nn; if (cut[t] < K){
35             REP(i, SZ(I)) if (Find(I[i]) == Find(t)) A.PB(I[i]); else B.PB(I[i]);
36             return 1;
37         }
38         REP_L(i, hd, suc) CC[i][s] = CC[s][i] += CC[t][i]; Union(s, t);
39     }
40
41     Prim(); if (cut[t] < K){
42         REP(i, SZ(I)) if (Find(I[i]) == Find(t)) A.PB(I[i]); else B.PB(I[i]);
43         return 1;
44     }
45     return 0;
46 }
47
48
```

```
49  int keCC(VI &I){
50
51      if (SZ(I) <= 1) return SZ(I);
52      VI A, B; if (!Stoer_Wagner(I, A, B)) return 1;
53
54      /*REP_2(i, j, SZ(A), SZ(B)){
55          int x = A[i], y = B[j];
56          C[x][y] = C[y][x] = 0;
57      }*/
58
59      return keCC(A) + keCC(B);
60  }
61
62  int main(){
63
64  #ifndef ONLINE_JUDGE
65      freopen("in.txt", "r", stdin);
66      //freopen("out.txt", "w", stdout);
67  #endif
68
69      while (~scanf("%d%d%d", &n, &m, &K)){
70
71          RST(C); DO(m){
72              int x, y; RD(x, y);
73              ++C[x][y], ++C[y][x];
74          }
75
76          VI I; REP_1(i, n) I.PB(i);
77          OT(keCC(I));
78      }
79  }
```

11.2 例题 (e.g.)

11.2.1 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)

11.2.2 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 12

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 13

生成树

13.1 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）

13.1.1 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）。

13.1.2 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。

13.2 次小

13.3 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、。。。

```
1
2  namespace DC_MST{
3
4      const int N = 109, M = 10 * N * N;
5
6      int hd[N], prd[M], suc[M], to[M], ww[M/2], m;
7      int P[N], mst, n;
8
9  #define aa to[i^1]
10 #define bb to[i]
11
12     inline void del(int i){
13         if (!prd[i]) prd[hd[aa] = suc[i]] = 0;
14         else suc[prd[suc[i]] = prd[i]] = suc[i];
15     }
16
17     inline void dell(int i){
18         del(i), del(i^1);
19     }
```

```
20
21      inline void add(int x, int y, int w){
22          ww[m>>1] = w, prd[hd[x]] = m, suc[m] = hd[x], hd[x] = m, to[m++] = y;
23          swap(x, y), prd[hd[x]] = m, suc[m] = hd[x], hd[x] = m, to[m++] = y;
24      }
25
26      inline void Make(int x){P[x] = x;}
27      inline int Find(int x){return P[x] == x ? x : P[x] = Find(P[x]);}
28      inline void Unionn(int x, int y){P[y] = x;}
29
30      struct Edge{
31          int x, y, w; Edge(int x, int y, int w):x(x),y(y),w(w){}
32          bool operator<(const Edge& r) const{return w < r.w;}
33          void process(){
34              int xx = Find(x), yy = Find(y); if (xx != yy){
35                  mst += w, Unionn(xx, yy);
36                  add(x, y, w);
37              }
38          }
39      }; vector<Edge> E;
40
41      void Kruskal(){
42          m = 2, RST(hd); mst = 0; SRT(E); REP_1(i, n) Make(i);
43          ECH(it, E) it->process();
44      }
45
46      bool vis[N]; int w1[N], mx[N], sw[N], si[N], uu;
47
48  #define v bb
49  #define w ww[i>>1]
50      void dfs1(int u){
51          vis[u] = 1; REP_G(i, u) if (!vis[v]){
52              if (w1[v] < w1[uu]) uu = v;
53              dfs1(v);
54          }
55      }
56      void dfs2(int u = 1){
57          vis[u] = 1; REP_G(i, u) if (!vis[v]){
58              if (w > mx[u]) mx[v] = w, si[v] = i; else mx[v] = mx[u], si[v] = si[u];
59              sw[v] = mx[v] - w1[v]; dfs2(v);
60          }
61      }
62  #undef w
63  #undef v
64
65      int dc_mst(int K){
66          Kruskal(); RST(vis); FOR_1(u, 2, n) if (!vis[u]){
67              uu = u, dfs1(u), --K; add(1, uu, w1[uu]), mst += w1[uu];
68          }
69
70          DO(K){
71              int dd = 0, ii, uu; RST(vis, mx); dfs2();
72              FOR_1(u, 2, n) if (sw[u] > dd){
73                  dd = sw[u], ii = si[u], uu = u;
74              }
75              if (!dd) break;
76              mst -= dd, dell(ii), add(1, uu, w1[uu]);
77          }
78
79          return mst;
80      }
81
82
83  } using namespace DC_MST;
84
85  map<string, int> H;
86
```

```
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()));
    }
}
```

13.4 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;

```

```
28      FOR(i, 1, _n) if (!bj[u]){
29          int x = u; do{bj[x] = u, x = fa[x];} while (!bj[x]);
30          if (bj[x] == u){
31              found = 1; do{bj[x] = nn, mst += ww[x = fa[x]];} while (bj[x] != nn);
32              Gen();
33          }
34      }
35
36
37      REP(i, _n) if (bj[u] < _nn) bj[u] = 0;
38      return found;
39  }
40
41  void Melt(){
42
43      REP(i, n) if (uu){ // Circle Canceling ...
44          REP(j, n) if (vv != uu){
45              if (vv) checkMin(G[uu][vv], G[u][v] - ww[v]);
46              else checkMin(G[uu][v], G[u][v]);
47          }
48      }
49      else {
50          REP(j, n) if (vv){
51              checkMin(G[u][vv], G[u][v] - ww[v]);
52          }
53      }
54
55      REP(i, n) if (uu) Del(i--);
56  }
57
58  #undef vv
59  #undef uu
60  #undef u
61  #undef v
62
63  void dfs(int u = 0){
64      bj[u] = 1; REP(v, n) if (G[u][v] != INF && !bj[v]) dfs(v);
65  }
66
67  int dMST(){
68      RST(bj); dfs();
69      FOR(i, 1, n) if (!bj[i]) return -1;
70
71      REP(i, n) Cur[i] = i; nn = n, mst = 0;
72      while (Find()) Melt();
73      FOR(i, 1, n) mst += ww[Cur[i]];
74      return mst;
75  }
76
77  class FoxTheLinguist {
78  public:
79      int minimalHours(int _n, vector <string> courseInfo) {
80
81          n = _n * 10 + 1; FLC(G, 0x3f); REP(i, _n){
82              G[0][i*10+1] = 0; REP_1(j, 9) checkMin(G[i*10+j+1][i*10+j], 0);
83          }
84
85          string s; s = accumulate(ALL(courseInfo), s);
86          REP(i, SZ(s)) if (s[i]=='-' || s[i]=='>' || s[i]==':') s[i] = ' ';
87          istringstream iss(s); for (string s1, s2, s3; iss >> s1 >> s2 >> s3;){
88              int u = (s1[0]-'A') * 10 + (s1[1]-'0') + 1;
89              int v = (s2[0]-'A') * 10 + (s2[1]-'0') + 1;
90              int w = s3[0] * 1000 + s3[1] * 100 + s3[2] * 10 + s3[3] - '0' * 1111;
91              checkMin(G[u][v], w);
92          }
93
94          return dMST();
```

```
95      }
96  };
```

13.5 例题

Chapter 14

匹配 (Match)

14.1 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;
}
```

14.2 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;
            }
        }
    }
}
```

14.3 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;
    }
}
```

14.4 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; //对于树中的父边是匹配边的点，向后跳BFSF
        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; //注意不能从这个奇环的关键点跳回来z
    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树中父结点的边是非匹配边的点，可能是没有入队的BFS
          }
      }
}

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 15

网络流 (Network FLow)

15.1 最大流/最小割

15.2 例题 (E.g.)

15.2.1 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()){
```

```
37
38          static int sta[N], cur[N]; int top = 0;
39          sta[0] = 0, cur[s] = hd[s]; while (top != -1){
40
41              int u = to[sta[top]], i; if (u == t){
42                  int d = INF; REP_1(ii, top) i = sta[ii], checkMin(d, c); max_flow += d;
43                  DWN_1(ii, top, 1){i = sta[ii], f += d, c -= d; if (!c) top = ii - 1;}
44                  u = to[sta[top]];
45              }
46
47              for (i=cur[u];i;i=suc[i])
48                  if (D[u] + 1 == D[v] && c) break;
49
50              if (!i) D[u] = 0, --top;
51              else {
52                  cur[u] = suc[i], cur[v] = hd[v];
53                  sta[++top] = i;
54              }
55          }
56      }
57
58      return max_flow;
59  }
60
61  const int NN = 509;
62  int A[NN][NN];
63  int nn, mm;
64
65  bool inGrid(int x, int y){
66      return x >= 0 && y >= 0 && x < nn && y < mm;
67  }
68
69  int V(int lv, int x, int y){
70      return lv*nn*mm + x*mm + y+1;
71  }
72
73  void Init(){
74      REP_2(i, j, nn, mm) RD(A[i][j]);
75
76      s = 0, t = 2*nn*mm+1, n = t+1, fill(hd, hd+n, 0), m = 2;
77      REP_2(i, j, nn, mm) if (A[i][j]){
78          if (A[i][j] == 2){
79              if (j&1) add_edge(s,V(0,i,j),1);
80              else add_edge(V(0,i,j),t,1);
81          }
82          else{
83              add_edge(V(0,i,j),V(1,i,j),1); REP(d, 4){
84                  int x=i+dx[d],y=j+dy[d];
85                  if (!inGrid(x, y) || A[x][y] != 2) continue;
86                  if (y&1) add_edge(V(0,x,y),V(0,i,j),1);
87                  else add_edge(V(1,i,j),V(0,x,y),1);
88              }
89          }
90      }
91  }
92
93  int main(){
94
95  #ifndef ONLINE_JUDGE
96      freopen("in.txt", "r", stdin);
97      //freopen("out.txt", "w", stdout);
98  #endif
99      while (~scanf("%d%d", &nn, &mm)){
100         Init(); OT(Dinitz());
101     }
102 }
```

15.2.2 上下界

15.2.3 无源汇

```
1
2  const int N = 509;
3  int C[N][N], cut[N], prd[N], suc[N], tmp[N];
4  int n, m, nn, s, t, tt;
5
6  inline void del(int x){prd[suc[prd[x]] = suc[x]] = prd[x];}
7  inline void rsm(int x){prd[suc[suc[prd[x]] = x]] = x;}
8
9  #define hd suc[0]
10
11 void Extract(int &s){
12     s = hd; REP_L(i, suc[s], suc) if (cut[i] > cut[s]) s = i;
13 }
14
15 void Prim(){
16     REP_L(i, hd, suc) cut[i] = 0; tt = 0; DO(nn-1){
17         Extract(s); del(s); tmp[tt++] = s;
18         REP_L(i, hd, suc) cut[i] += C[s][i];
19     }
20     Extract(t); DWN(i, tt, 0) rsm(tmp[i]); del(t);
21 }
22
23 int Stoer_Wagner(){
24
25     REP_1(i, n) suc[i] = i+1, prd[i] = i-1;
26     hd = 1, suc[n] = 0; nn = n;
27
28     int res = INF; DO(n-2){
29         Prim(), --nn, checkMin(res, cut[t]);
30         REP_L(i, hd, suc) C[i][s] = C[s][i] += C[t][i];
31     }
32
33     Prim(), checkMin(res, cut[t]);
34     return res;
35 }
36
37 int main(){
38
39 #ifndef ONLINE_JUDGE
40     freopen("in.txt", "r", stdin);
41     //freopen("out.txt", "w", stdout);
42 #endif
43
44     while (scanf("%d%d", &n, &m) != EOF){
45
46         RST(C); DO(m){
47             int x, y, w; RD(x, y, w); ++x, ++y;
48             C[x][y] += w, C[y][x] += w;
49         }
50
51         OT(Stoer_Wagner());
52     }
53 }
```

Part V

字符串 (Stringology)

Chapter 16

万金油 (Hash)

16.1 例题 (E.g.)

16.1.1 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;
        }
    }
```

```
46
47      inline int sgn(int x){return r[px] == x;}
48      inline void setc(int y, int d, int x){c[d][y] = x, px = y;}
49      inline void setl(int y, int x){setc(y,0,x);}
50      inline void setr(int y, int x){setc(y,1,x);}
51
52      inline void rot(int x, int d){
53          int y=px,z=py;setc(z,sgn(y),x);
54          setc(y,d,c[!d][x]),setc(x,!d,y);upd(y);
55      }
56
57      inline void rot(int x){rot(x,sgn(x));}
58      inline int splay(int x,int t=0){
59          int a,b,y; while((y=px)!=t){
60              if (py==t){rot(x);break;}
61              else a=sgn(x),b=sgn(y),rot(a^b?x:y,a),rot(x,b);
62          }
63          upd(x);if(!t)rt=x;
64          return x;
65      }
66      int Build(int a = 0, int b = n+1){
67          if (a > b) return 0; int m = a + b >> 1, x = new_node(str[m]);
68          setl(x, Build(a, m-1)), setr(x, Build(m+1, b)), upd(x);
69          return x;
70      }
71
72      int Select(int k, int t=0){
73          int x = rt; while (rls(x), sz[lx] != k){
74              if (k<sz[lx]) x = lx;
75              else k -= sz[lx]+1, x = rx;
76          }
77          return splay(x, t);
78      }
79
80      int Selectt(int a, int b){
81          return r[Select(a-1, Select(b+1))];
82      }
83
84      int Lcp(int x, int y){
85          int ll = -1, rr = n - max(x, y);
86          while (ll < rr){
87              int m = ll + rr + 1 >> 1;
88              if (S[0][Selectt(x, x+m)] == S[0][Selectt(y, y+m)]) ll = m;
89              else rr = m - 1;
90          }
91          return ll+1;
92      }
93
94      void Inorder(int x = rt){
95          if (!x) return; rls(x);
96          Inorder(lx);
97          cout << char(s[x]+'0'-1);
98          Inorder(rx);
99      }
100
101 } using namespace Splay;
102
103 int main(){
104
105 #ifndef ONLINE_JUDGE
106     freopen("in.txt", "r", stdin);
107     //freopen("out2.txt", "w", stdout);
108 #endif
109
110     P[0] = 1; FOR(i, 1, N/2) P[i] = P[i-1] * C;
111
112     int m; while (~scanf("%d%d", &n, &m)){
```

```
113
114         n = strlen(RS(str+1)); REP_S(cur, str+1) *cur -= '0'-1;
115
116         tot = 0; rt = Build();
117
118         int a, b, x, y, z; DO(m){
119             switch(RD()){
120                 case 1: // Insert .. .
121                     RD(a); y = Select(a, z = Select(a+1)); setr(y, x = new_node(RD()+1));
122                     upd(x), upd(y), upd(z); ++n;
123                     break;
124                 case 2: // Delete .. .
125                     RD(a); y = Select(a-1, z = Select(a+1)); ry = 0;
126                     upd(y), upd(z); --n;
127                     break;
128                 case 3: // Reverse
129                     RD(a, b); rev(Selectt(a, b));
130                     break;
131                 default:
132                     OT(Lcp(RD(), RD()));
133             }
134
135             //Inorder(); puts("");
136         }
137     }
138 }
```

Chapter 17

在线算法 (Online)

17.1 KMP

1-offset

```
1  void get_pi(const char P[], int n, int pi[]){
2      for (int i = 1, j = pi[1] = 0; i < n; pi[++i] = j){
3          while (j && P[i] != P[j]) j = pi[j];
4          if (P[i] == P[j]) ++j;
5      }
6  }
```

0-offset

```
1  void get_pi(const char P[], int n, int pi[]){
2      for (int i = 1, j = pi[1] = 0; i < n; pi[++i] = j){
3          while (j && P[i] != P[j]) j = pi[j];
4          if (P[i] == P[j]) ++j;
5      }
6  }
```

17.2 Z

0-offset

```
1  void get_z(const char P[], int n, int z[]){
2
3      int ex; z[0] = ex = n;
4
5      for (int i = 1, l = 0, r = 0; i < n; z[i++] = ex){
6          if (i > r){
7              for (l = r = i; r < n && P[r] == P[r - l];) ++r;
8              ex = r---l;
9          }
10         else {
11             if (z[i - l] < r - i + 1) ex = z[i - l];
12             else {
13                 for (l = i; r < n && P[r] == P[r - l];) ++r;
14                 ex = r---l;
15             }
16         }
17     }
18 }
```

17.3 Manacher

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

```
3
4  int Manacher(char s[] = str+1){
5      static char ss[2*N+2]; static int p[2*N+2];
6      nn = 0; ss[nn++] = '$'; REP(i, n) ss[nn++] ='#', ss[nn++] = s[i]; ss[nn++] = '#'; //ss[nn] = 0;
7
8      int mx=0, id=0; FOR(i, 1, nn){
9          p[i] = mx > i ? min(p[2*id-i], mx-i) : 1;
10         while (ss[i+p[i]]==ss[i-p[i]]) ++p[i];
11         if (i+p[i]>mx) mx=i+p[i], id=i;
12     }
13
14     int len = 0, pos; FOR(i, 1, nn) if (p[i] > len) len = p[i], pos = i;
15     //FOR(i, 1, nn) cout << p[i] << " "; cout << endl;
16     //for (int i = pos - --len + 1; i < pos + len; i += 2) putchar(ss[i]);
17     //int st = pos/2 - (--len + !(pos&1))/2; FOR(i, st, st + len) putchar(s[i]); cout << endl;
18     return --len;
19 }
```

17.4 最小表示

17.5 AC-Automaton)

17.6 例题 (E.g.)

17.6.1 SPOJ RECTANGLE

题目描述 (Brief description)

算法分析 (Algorithm Analysis)

外部链接 (External Link)

17.6.2 SPOJ RECTANGLE

题目描述 (Brief description)

算法分析 (Algorithm Analysis)

外部链接 (External Link)

17.6.3 SPOJ RECTANGLE

题目描述 (Brief description)

算法分析 (Algorithm Analysis)

外部链接 (External Link)

Chapter 18

后缀三姐妹 (Indexed)

18.1 后缀数组 (Suffix Array)

18.1.1 子串计数

18.2 后缀自动机 (Suffix Automaton)

18.2.1 子串计数

18.2.2 出现次数向父亲传递

18.2.3 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];
    }
}
```

18.2.4 最小表示与循环同构

18.3 后缀树 (Suffix Tree)

18.3.1 子串计数

18.4 字典树 (On Trie)

18.5 动态 (On Dynamic)

18.5.1 push_back()

18.5.2 pop_back()

18.5.3 pop_front()

18.5.4 push_front()?

18.5.5 双向链表维护 SA

18.5.6 平衡树维护 SA

18.5.7 带删除标记的 SAM

18.5.8 LCT 维护 SAM

18.6 可持久化 (On Persistence)

18.6.1 LCT 维护 SAM

18.7 例题 (E.g.)

18.7.1 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

```
1         #define c adj[u][i]
2  void kth(int k){
3      int u = 0; while (k){
4          --k; REP(i, Z){
5              if (k >= cnt[v]) k -= cnt[v];
6              else{
7                  putchar(c+'a');
8                  u = v; break;
9              }
10         }
11     }
12     puts("");
13 }
```

后缀数组

```
1  int get_h(){
2      ...
3      a[0] = 0; REP_1(i, n) a[i] = a[i-1] + n-sa[i]-h[i];
4  }
```

18.7.2 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++];
```

```
59  }
60
61  void get_h(){
62      REP_1(i, n) rk[sa[i]] = i;
63      int k=0;for(int i=0;i<n;h[rk[i++]]=k){
64          if (k)--k;for(int j=sa[rk[i]-1];a[i+k]==a[j+k];++k);
65      }
66  }
67
68  int ST[LV][N];
69
70  #define cmp(a, b) (h[a]<h[b]?a:b)
71
72  inline int lcp(int l, int r){
73      int lv = lg2(r - l); ++l, ++r;
74      return min(h[ST[lv][l]], h[ST[lv][r-_1(lv)]]);
75  }
76
77  inline int lcpp(int l, int r){
78      if (l == r) return n-l;
79      l = rk[l], r = rk[r]; if (l > r) swap(l, r);
80      return lcp(l, r);
81  }
82
83  void get_lcp(){
84      REP_1(i, n) ST[0][i] = i;
85      for (int lv = 1; _1(lv) <= n; ++lv){
86          for (int i = 1; i + _1(lv) <= n + 1; ++i)
87              ST[lv][i] = cmp(ST[lv-1][i], ST[lv-1][i+_1(lv-1)]);
88      }
89  }
90
91  void bd(){
92      dc3(a,sa,n+1,M),get_h(),get_lcp();
93  }
94  } A, B;
95
96  PII get(LL k){
97      int r = lower_bound(a, a+n, k) - a; k -= a[r-1];
98      return MP(A.sa[r]+1, A.h[r]+k);
99  }
100
101 LL f(LL x, LL y){
102     if (x>a[n] || y>a[n]) return -1;
103     PII a = get(x), b = get(y); int t = min(a.se, b.se);
104     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)))));
105 }
106
107
108 int main(){
109
110 #ifndef ONLINE_JUDGE
111     freopen("in.txt", "r", stdin);
112     //freopen("out.txt", "w", stdout);
113 #endif
114
115     int m; RD(n, m); strlen(RS(s)); REP(i, n) B.a[n-i-1]=A.a[i]=s[i]-='a'-1;
116     A.bd(); B.bd(); REP_1(i, n) a[i]=a[i-1]+n-A.sa[i]-A.h[i];
117     DO(m) OT(f(RD(), RD()));
118 }
```

18.7.3 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。。

```
1  const int N = 200009, LN = 18;
2
3  int a[N], b[N], sa[N], sl[N], rankk[N], height[N], ST[LN][N];
4  int C[N], key[N], t1[N], t2[N]; int n, nn;
5
6  namespace Fotile_Tree{
7
8      #define lx lc[x]
9      #define rx rc[x]
10     #define ly lc[y]
11     #define ry rc[y]
12     #define cx c[x]
13     #define cy c[y]
14     #define ml (l+r>>1)
15     #define mr (ml+1)
16     #define l0 1
17     #define r0 nn
18     #define lcc lx, l, ml
19     #define rcc rx, mr, r
20
21     const int NN = N*LN; //1000009
22
23     int lc[NN], rc[NN], c[NN], tot, aa, bb;
24     int T[N];
25
26     inline int new_node(){
27         return ++tot;
28     }
29
30     int Insert(int y, int p, int d){
31         int x = new_node(), root = x; c[x] = c[y] + d;
32         int l = l0, r = r0; while (l < r){
33             if (p < mr){
34                 lx = new_node(), rx = ry;
35                 x = lx, y = ly, r = ml;
36             }
37             else {
```

```
38                  lx = ly, rx = new_node();
39                  x = rx, y = ry, l = mr;
40              }
41              c[x] = c[y] + d;
42          }
43          return root;
44      }
45
46      inline int sum(int x, int l = l0, int r = r0){
47          if (r < aa || bb < l) return 0;
48          if (aa <= l && r <= bb) return c[x];
49          return sum(lcc) + sum(rcc);
50      }
51
52      inline int lsum(int x, int p){
53          int res = 0, l = l0, r = r0; while (p != r){
54              if (p < mr) x = lx, r = ml;
55              else res += c[lx], x = rx, l = mr;
56          }
57          return res + cx;
58      }
59
60      #undef lx
61      #undef rx
62      #undef ly
63      #undef ry
64      #undef cx
65      #undef cy
66      #undef mid
67
68  } using namespace Fotile_Tree;
69
70
71  inline void rs(int *x, int *y, int *sa, int n, int m){
72      REP(i, n) key[i] = i[y][x];
73      memset(C, 0, sizeof(C[0]) * m);
74      REP(i, n) ++C[key[i]];
75      FOR(i, 1, m) C[i] += C[i-1];
76      DWN(i, n, 0) sa[--C[key[i]]] = y[i];
77  }
78
79  void da(int a[], int sa[], int n, int m){
80      int *x = t1, *y = t2;
81      memset(C, 0, sizeof(C[0])*m);
82      REP(i, n) ++C[x[i] = a[i]];
83      FOR(i, 1, m) C[i] += C[i-1];
84      DWN(i, n, 0) sa[--C[x[i]]] = i;
85      for (int l = 1, p = 1; p < n; l <<= 1, m = p){
86          p = 0; FOR(i, n-l, n) y[p++] = i;
87          REP(i, n) if (sa[i] >= l) y[p++] = sa[i] - l;
88          rs(x, y, sa, n, m), swap(x, y), x[sa[0]] = 0, p = 0; FOR(i, 1, n)
89              x[sa[i]] = (y[sa[i]] == y[sa[i-1]] && y[sa[i]+l] == y[sa[i-1]+l]) ? p : ++p;
90          ++p;
91      }
92  }
93
94  void gh(int sa[], int rankk[], int height[], int n){
95      REP_1(i, n) rankk[sa[i]] = i;
96      int k = 0; for (int i = 0; i < n; height[rankk[i++]] = k){
97          if (k) --k; for (int j = sa[rankk[i]-1]; a[i+k] == a[j+k]; ++k);
98      }
99  }
100
101 inline bool shorter(int a, int b){
102     return height[a] < height[b];
103 }
104
```

```
105 inline int lcp(int l, int r){
106     if (l == r) return sl[sa[l]];
107     int lv = lg2(r - l); ++l, ++r;
108     return height[min(ST[lv][l], ST[lv][r-_1(lv)], shorter)];
109 }
110
111 inline int lcpp(int l, int r){
112     l = rankk[l], r = rankk[r]; if (l > r) swap(l, r);
113     return lcp(l, r);
114 }
115
116 void get_lcp(){
117     REP_1(i, n) ST[0][i] = i;
118     for (int lv = 1; _1(lv) <= n; ++lv){
119         for (int i = 1; i + _1(lv) <= n + 1; ++i)
120             ST[lv][i] = min(ST[lv-1][i], ST[lv-1][i+_1(lv-1)], shorter);
121     }
122 }
123
124
125
126 int discretize(int a[], int n){
127     int m = 1; VI A; REP(i, n) A.PB(a[i]); UNQ(A);
128     REP(i, n) a[i] = lower_bound(ALL(A), a[i]) - A.begin() + 1; a[n] = 0;
129     return SZ(A) + 1;
130 }
131
132
133 int main(){
134
135 #ifndef ONLINE_JUDGE
136     freopen("in.txt", "r", stdin);
137     //freopen("out.txt", "w", stdout);
138 #endif
139
140     REP_C(i, RD(n)) RD(a[i]);
141
142     if (n == 1){
143         Rush puts("0");
144         return 0;
145     }
146
147     nn = n-1; REP(i, nn) a[i] = a[i+1] - a[i]; a[nn] = INF;
148     REP(i, nn) a[i+n] = -a[i]; n=2*n-1;
149     da(a, sa, n+1, discretize(a, n));
150     gh(sa, rankk, height, n); get_lcp();
151
152     T[0] = new_node(); REP_1(i, n){
153         T[i] = sa[i] < nn ? T[i] = Insert(T[i-1], sa[i]+1, 1) : T[i-1];
154     }
155
156     Rush{
157         int l, r; RD(l, r); if (l == r) OT(nn);
158         else{
159             int p = rankk[l+nn], len = r-l;
160             aa = max(l0, l-len), bb = min(r0, l+len);
161
162             l = 1, r = p; while (l < r){
163                 int m = l + r >> 1;
164                 if (lcp(m, p) >= len) r = m;
165                 else l = m + 1;
166             }
167
168             int ll = T[l-1];
169
170             l = p, r = n; while (l < r){
171                 int m = l + r + 1 >> 1;
```

```
172                 if (lcp(p, m) >= len) l = m;
173                 else r = m - 1;
174             }
175
176             int rr = T[r];
177
178             OT(c[rr]-c[ll]-(sum(rr)-sum(ll)));
179         }
180     }
181 }
```

18.7.4 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

aaaabaaaa
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。。

$aaa
abaaaa
a
a$aaa
aabaaaa
aa
aaabaaaa
aaa
aaaabaaaa
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 4. 后缀自动机

算法 3

```
1  const int N = 200009, LN = 24;
2
3  int a[3*N], sa[3*N], rankk[N], height[N], ST[LN][N], b[N], sl[N];
4  int C[N], key[N], t1[N], t2[N]; char buf[N];
5  int nn, n, k;
6
7
8  inline void rs(int *x, int *y, int *sa, int n, int m){
9      REP(i, n) key[i] = i[y][x];
10     memset(C, 0, sizeof(C[0]) * m);
11     REP(i, n) ++C[key[i]];
12     FOR(i, 1, m) C[i] += C[i-1];
13     DWN(i, n, 0) sa[--C[key[i]]] = y[i];
14 }
15
16 void da(int a[], int sa[], int n, int m){
17     int *x = t1, *y = t2;
18     memset(C, 0, sizeof(C[0])*m);
19     REP(i, n) ++C[x[i] = a[i]];
20     FOR(i, 1, m) C[i] += C[i-1];
21     DWN(i, n, 0) sa[--C[x[i]]] = i;
22     for (int l = 1, p = 1; p < n; l <<= 1, m = p){
23         p = 0; FOR(i, n-l, n) y[p++] = i;
24         REP(i, n) if (sa[i] >= l) y[p++] = sa[i] - l;
25         rs(x, y, sa, n, m), swap(x, y), x[sa[0]] = 0, p = 0; FOR(i, 1, n)
26             x[sa[i]] = (y[sa[i]] == y[sa[i-1]] && y[sa[i]+l] == y[sa[i-1]+l]) ? p : ++p;
27         ++p;
28     }
29 }
30
31 void gh(int sa[], int rankk[], int height[], int n){
32     REP_1(i, n) rankk[sa[i]] = i;
33     int k = 0; for (int i = 0; i < n; height[rankk[i++]] = k){
34         if (k) --k; for (int j = sa[rankk[i]-1]; a[i+k] == a[j+k]; ++k);
35     }
36 }
37
38
39 #define F(x) ((x)/3+((x)%3==1?0:tb))
40 #define G(x) ((x)<tb?(x)*3+1:((x)-tb)*3+2)
41 int c0(int *r,int a,int b)
42 {return r[a]==r[b]&&r[a+1]==r[b+1]&&r[a+2]==r[b+2];}
43 int c12(int k,int *r,int a,int b)
44 {if(k==2) return r[a]<r[b]||r[a]==r[b]&&c12(1,r,a+1,b+1);
45  else return r[a]<r[b]||r[a]==r[b]&&key[a+1]<key[b+1];}
46
47 void dc(int a[], int *sa, int n, int m){
48     int i, j, *an=a+n, *san=sa+n, ta=0, tb=(n+1)/3, tbc=0, p;
49     a[n] = a[n+1] = 0; REP(i, n) if (i%3) t1[tbc++]=i;
50
51     rs(a+2,t1,t2,tbc,m), rs(a+1,t2,t1,tbc,m), rs(a,t1,t2,tbc,m);
52     p = 0, an[F(t2[0])] = 0; FOR(i, 1, tbc)
53         an[F(t2[i])] = c0(a,t2[i-1],t2[i]) ? p : ++p;
54
55     if (++p < tbc) dc(an,san,tbc,p);
56     else REP(i, tbc) san[an[i]] = i;
57
58     REP(i, tbc) if(san[i] < tb) t2[ta++] = san[i] * 3;
59     if (n%3==1) t2[ta++] = n-1; rs(a,t2,t1,ta,m);
60     REP(i, tbc) key[t2[i]=G(san[i])] = i;
61
62     for(i=0,j=0,p=0; i<ta && j<tbc; p++)
63         sa[p]=c12(t2[j]%3,a,t1[i],t2[j]) ? t1[i++] : t2[j++];
64     for(;i<ta;p++) sa[p]=t1[i++]; for(;j<tbc;p++) sa[p]=t2[j++];
65 }
66
67
```

```
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]);
}
```

算法 4

```
const int N = 200009, LN = 24, Z = 26;

char buf[N/2]; LL ans[N/2];
int TtoM[N/2], MtoT[N], nn, K;

```

```
6  namespace Trie{
7      int trans[N/2][Z]; VI b[N/2];
8      int tot;
9
10 #define v trans[u][c]
11
12     int new_node(){
13         return tot++;
14     }
15
16     void Insert(int id){
17         RS(buf); int u = 0; REP_S(cur, buf){
18             int c = *cur - 'a';
19             if (!v) v = new_node();
20             b[u = v].PB(id);
21         }
22     }
23
24     void Init(){
25         tot = 0, new_node();
26         REP(i, nn) Insert(i);
27     }
28 }
29
30 namespace SAM{
31
32     int trans[N][Z], par[N], len[N], cnt[N], tot;
33     VI adj[N]; int P[N], L[N/2];
34
35 #define p par[u]
36 #define pp par[uu]
37
38     void Make(int x){P[x] = x;}
39     int Find(int x){return x == P[x] ? x : P[x] = Find(P[x]);}
40     void Union(int x, int y){P[y] = x;}
41
42     inline int new_node(){
43         RST(trans[tot]); //cnt[tot] = 0;
44         return tot++;
45     }
46
47     inline int new_node(int u){
48         CPY(trans[tot], trans[u]); par[tot] = par[u]; //cnt[tot] = cnt[u];
49         return tot++;
50     }
51
52     inline int h(int u){
53         return len[u] - len[p];
54     }
55
56     int Ext(int c, int tail){
57         int u = tail, uu = new_node(); len[uu] = len[u] + 1;
58         while (u && !v) v = uu, u = p; // 向上遍历没有c转移- 的祖先..
59         if (!u && !v) v = uu, pp = 0;
60         else{
61             if (len[v] == len[u] + 1) pp = v;
62             else{
63                 int _v = v, vv = new_node(_v); len[vv] = len[u] + 1; par[_v] = pp = vv;
64                 while (u && v == _v) v = vv, u = p;
65                 if (!u && v == _v) v = vv;
66             }
67         }
68         return uu;
69     }
70
71 #undef v
72 #define v (*it)
```

```
73
74      void tarjan(int u = 0){
75
76          Make(u); ECH(it, adj[u]) tarjan(v), Union(u, v);
77
78          if(MtoT[u]) ECH(it, Trie::b[MtoT[u]]){
79              --cnt[Find(L[v])]; ++cnt[u], L[v] = u;
80          }
81      }
82
83      void dfs1(int u = 0){
84          ECH(it, adj[u]) dfs1(v), cnt[u] += cnt[v];
85      }
86
87      void dfs2(int u = 0){
88          ECH(it, adj[u]) cnt[v] += cnt[u], dfs2(v);
89          ECH(it, Trie::b[MtoT[u]]) ans[v] += cnt[u];
90      }
91
92  #undef v
93  #define v Trie::trans[u][c]
94
95      void Init(){
96          tot = 0; queue<int> Q; Q.push(0); TtoM[0] = new_node();
97
98          while(SZ(Q)){
99              int u = Q.front(); Q.pop();
100             REP(c, 26) if (v) Q.push(MtoT[TtoM[v] = Ext(c, TtoM[u])] = v);
101         }
102
103         FOR(u, 1, tot) adj[p].PB(u); fill(L, L+nn, 0); tarjan();
104         dfs1(); FOR(u, 1, tot) cnt[u] = cnt[u] >= K ? h(u) : 0; dfs2();
105     }
106 }
107
108
109 int main(){
110
111 #ifndef ONLINE_JUDGE
112     freopen("in.txt", "r", stdin);
113     //freopen("out.txt", "w", stdout);
114 #endif
115
116     RD(nn, K); Trie::Init(); SAM::Init();
117     REP(i, nn) OT(ans[i]);
118 }
```

18.7.5 CF 316G3. Good Substrings string suffix structures

题目描述 (Brief description)

算法分析 (Algorithm Analysis)

18.7.6 CF 235C. Cyclical Quest

```
1
2   const int N = int(2e6) + 9, Z = 26;
3
4   namespace SAM{
5
6       int trans[N][Z], par[N], len[N], cnt[N], tail, tot; char str[N/2];
7
8       inline int new_node(){
9           //RST(trans[tot]),
10          tail = tot; cnt[tot] = 1;
11          return tot++;
```

```
12      }
13
14      inline int new_node(int u){
15          CPY(trans[tot], trans[u]), par[tot] = par[u]; //cnt[tot] = 0
16          return tot++;
17      }
18
19  #define v trans[u][c]
20  #define p par[u]
21  #define pp par[uu]
22
23      void Ext(int c){
24          int u = tail, uu = new_node(); len[uu] = len[u] + 1;
25          while (u && !v) v = uu, u = p;
26          if (!u && !v) v = uu, pp = 0;
27          else{
28              if (len[v] == len[u] + 1) pp = v;
29              else{
30                  int _v = v, vv = new_node(_v); len[vv] = len[u] + 1;
31                  par[_v] = pp = vv;
32                  while (v == _v) v = vv, u = p;
33              }
34          }
35      }
36
37  int Q[N], C[N/2], tt;
38  #define vis Q
39  #define c (*cur - 'a')
40      int Spell(){
41          int ll = strlen(RS(str)); int res = 0, l = 0, u = 0; REP_S(cur, str){
42              while (u && !v) l = len[u = p];
43              if (u = v) ++l;
44          }
45
46          --tt; REP_S(cur, str){
47              while (u && !v) l = len[u = p]; if (u = v) ++l;
48              while (len[p] >= ll) l = len[u = p];
49
50              if(l >= ll && vis[u] != tt){
51                  vis[u] = tt;
52                  res += cnt[u];
53              }
54          }
55          return res;
56      }
57
58      void Init(){
59          tot = 0, new_node(); RS(str); REP_S(cur, str) Ext(c);
60
61          //REP(i, tot) C[i] = 0;
62          REP(i, tot) ++C[len[i]];
63          REP_1(i, len[tail]) C[i] += C[i-1];
64          REP(i, tot) Q[--C[len[i]]] = i;
65
66  #undef c
67
68          DWN(i, tot, 1){
69              int u = Q[i];
70              cnt[p] += cnt[u];
71              //cout << u << " " << cnt[u] << endl;
72          }
73          tt = INF;
74      }
75
76  } using namespace SAM;
77
78  int main(){
```

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

    Init(); Rush OT(Spell());
}
```

18.7.7 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;
        }
    }
```

```
56
57      inline void _rot(int d){
58          node *y = p, *z = py;
59          if (y->sgn() != -1) z->link(y->sgn(), this); else p = z;
60          y->link(d, c[d^1]), link(d^1, y);
61          y->update();
62      }
63
64      inline void rot(){_rot(sgn());}
65      inline void zig(){_rot(0);}
66      inline void zag(){_rot(1);}
67
68      inline void fix(){
69          if(~sgn()) p->fix();
70          release();
71      }
72
73
74  /*
75      inline node* splay(){
76          fix(); while (sgn() != -1) rot();
77          update();
78          return this;
79      }
80  */
81      inline node* splay(){
82          fix(); while (sgn() != -1){
83              node *y = p, *z = y->p; if (y->sgn() == -1){ rot(); break;}
84              if (z->l == y){
85                  if (y->l == this) y->zig(), zig();
86                  else zag(), zig();
87              }else{
88                  if (y->r == this) y->zag(), zag();
89                  else zig(), zag();
90              }
91          }
92          update();
93          return this;
94      }
95
96      inline node* access(){
97          node *x = this, *y = NIL; do{
98              x->splay();
99              rx = y, x->update();
100             y = x, x = px;
101         } while (x != NIL);
102         return y;
103     }
104
105     inline node* accesss(){
106         access();
107         return splay();
108     }
109
110     node* rt(){
111         node* x; for (x = access(); x->release(), lx != NIL; x = lx);
112         return x;
113     }
114
115     node* evert(){
116         access()->rev ^=1;
117         return this;
118     }
119
120 #define evertt evert()->splay
121     // Public ...
122
```

```
123     void Link(node* x){
124         //if (x == NIL) return;
125         access(); p = x;
126         p->access()->inc(w0);
127     }
128 
129     void Cut(){
130         accesss(); //if (l == NIL) return;
131         l->inc(-w0), l->p = NIL, l = NIL;
132     }
133 
134     int Query(){
135         return accesss()->w0;
136     }
137 
138 } TPool[N], *T[N], *NIL;
139 
140 #define v trans[u][c]
141 #define p par[u]
142 #define pp par[uu]
143 
144 inline int new_node(){
145     //RST(trans[tot);
146     T[tot]->reset(1);
147     return tot++;
148 }
149 
150 inline int new_node(int u){
151     CPY(trans[tot], trans[u]); par[tot] = par[u];
152     T[tot]->reset(0); T[tot]->Link(T[par[u]]);
153     return tot++;
154 }
155 
156 void Ext(int c){
157     int u = tail, uu = new_node();len[uu] = len[u] + 1;
158     while (u && !v) v = uu, u = p; // 向上遍历没有c转移- 的祖先..
159     if (!u && !v) v = uu, pp = 0;
160     else{
161         if (len[v] == len[u] + 1) pp = v, T[uu]->Link(T[v]);
162         else{
163             int _v = v, vv = new_node(_v); len[vv] = len[u] + 1;
164             T[_v]->Cut(), T[_v]->Link(T[vv]), T[uu]->Link(T[vv]);
165             par[_v] = pp = vv;
166             while (u && v == _v) v = vv, u = p;
167             if (!u && v == _v) v = vv;
168         }
169     }
170     // ...
171     tail = uu;
172 }
173 
174 #define c (*cc - 'a')
175 void Init(){
176     tot = 0, tail = new_node();
177     RS(s); REP_S(cc, s) Ext(c);
178 }
179 
180 void Run(){
181     int ans = 0, q, a, b; RD(q, a, b); DO(q){
182         int u = 0; RS(s); REP_S(cc, s) if (!(u = v)) break;
183         int ans = u ? T[u]->Query() : 0; OT(ans);
184         Ext((a*ans+b)%26);
185     }
186 }
187 
188 int main(){
189 
```

```
#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();
}
```

18.7.8 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;
        }
    }
```

```
56
57      inline void _rot(int d){
58          node *y = p, *z = py;
59          if (y->sgn() != -1) z->link(y->sgn(), this); else p = z;
60          y->link(d, c[d^1]), link(d^1, y);
61          y->update();
62      }
63
64      inline void rot(){_rot(sgn());}
65      inline void zig(){_rot(0);}
66      inline void zag(){_rot(1);}
67
68      inline void fix(){
69          if(~sgn()) p->fix();
70          release();
71      }
72
73  /*
74      inline node* splay(){
75          fix(); while (sgn() != -1) rot();
76          update();
77          return this;
78      }
79  */
80      inline node* splay(){
81          fix(); while (sgn() != -1){
82              node *y = p, *z = y->p; if (y->sgn() == -1){ rot(); break;}
83              if (z->l == y){
84                  if (y->l == this) y->zig(), zig();
85                  else zag(), zig();
86              }else{
87                  if (y->r == this) y->zag(), zag();
88                  else zig(), zag();
89              }
90          }
91          update();
92          return this;
93      }
94
95      inline node* access(){
96          node *x = this, *y = NIL; do{
97              x->splay();
98              rx = y, x->update();
99              y = x, x = px;
100         } while (x != NIL);
101         return y;
102     }
103
104     inline node* accesss(){
105         access();
106         return splay();
107     }
108
109     node* rt(){
110         node* x; for (x = access(); x->release(), lx != NIL; x = lx);
111         return x;
112     }
113
114     node* evert(){
115         access()->rev ^=1;
116         return this;
117     }
118
119 #define evertt evert()->splay
120     // Public ...
121
122     void Link(node* x){
```

```
123         //if (x == NIL) return;
124         access(); p = x;
125         p->access()->inc(w0);
126     }
127
128     void Cut(){
129         accesss(); //if (l == NIL) return;
130         l->inc(-w0), l->p = NIL, l = NIL;
131     }
132
133     int Query(){
134         return accesss()->w0;
135     }
136
137     int h();
138
139     void dfs(){
140         if (this == NIL || w1 < K) return;
141         Deepest = this; release(); if (w0 >= K) ans += h(), w0 -= INF;
142         l->dfs(), r->dfs(), update();
143     }
144
145     void Stat(){
146         Deepest = this, accesss()->dfs();
147         Deepest->splay();
148     }
149 } TPool[N], *T[N], *NIL, *node::Deepest;
150
151 #define v trans[u][c]
152 #define p par[u]
153 #define pp par[uu]
154
155 inline int new_node(){
156     RST(trans[tot]);
157     T[tot]->reset(1);
158     return tot++;
159 }
160
161 inline int new_node(int u){
162     CPY(trans[tot], trans[u]); par[tot] = par[u];
163     T[tot]->reset(0),
164     T[tot]->w1 = T[u]->w1;
165     T[tot]->Link(T[par[u]]);
166     return tot++;
167 }
168
169 inline int h(int u){
170     return len[u] - len[p];
171 }
172
173 inline int node::h(){
174     return ::h(this - TPool);
175 }
176
177 void Ext(int c){
178     int u = tail, uu = new_node(); len[uu] = len[u] + 1;
179     while (u && !v) v = uu, u = p; // 向上遍历没有c转移- 的祖先..
180     if (!u && !v) v = uu, pp = 0;
181     else{
182         if (len[v] == len[u] + 1) pp = v, T[uu]->Link(T[v]);
183         else{
184             int _v = v, vv = new_node(_v); len[vv] = len[u] + 1;
185             T[_v]->Cut(), T[_v]->Link(T[vv]), T[uu]->Link(T[vv]);
186             par[_v] = pp = vv;
187             while (u && v == _v) v = vv, u = p;
188             if (!u && v == _v) v = vv;
189         }
```

```
190     }
191     T[uu]->Stat(); // .. .
192     tail = uu;
193 }
194
195 void Init(){
196     ans = tot = 0; tail = new_node();
197     RS(s); REP_S(c, s) Ext(*c - 'a');
198 }
199
200 int main(){
201
202 #ifndef ONLINE_JUDGE
203     freopen("in.txt", "r", stdin);
204     //freopen("out.txt", "w", stdout);
205 #endif
206
207     NIL = new node(); NIL->reset(-INF);
208     REP(i, N) T[i] = &TPool[i];
209
210     while (~scanf("%*d%d%d", &m, &K)){
211         Init(); DO(m) if (RD() == 1) Ext(RC() - 'a');
212         else OT(ans);
213     }
214 }
```

18.7.9 SPOJ AE5A2. quasi-template

```
1
2  const int N = int(4e5) + 9, C = 26;
3
4
5  namespace Splay{
6
7  struct node{
8
9      static node*NIL;node*c[2],*p;
10     int ll,ky,rr,dd;
11
12 #define NIL node::NIL
13 #define l c[0]
14 #define r c[1]
15 #define lx x->l
16 #define rx x->r
17
18     void reset(int v = 0){
19         l = r = p = NIL;
20         dd = 0, ll = rr = ky = v;
21     }
22
23     node(int v = 0){
24         reset(v);
25     }
26
27     void upd(){
28         dd = 0;
29         if (l == NIL) ll = ky; else ll = l->ll, checkMax(dd, max(l->dd, ky-l->rr));
30         if (r == NIL) rr = ky; else rr = r->rr, checkMax(dd, max(r->dd, r->ll-ky));;
31     }
32
33     void setc(int d, node *x){c[d]=x,x->p=this;}
34     int sgn(){return p->r==this;}
35
36     void rot(int d){
37         node*y=p,*z=y->p; z->setc(y->sgn(), this);
```

```
38          y->setc(d, c[!d]), setc(!d, y), y->upd();
39      }
40      void rot(){rot(sgn());}
41
42      node *splay(){
43          int a, b; while(p!=NIL){
44              //cout << "!" << endl;
45              if (p->p==NIL){rot();break;}
46              else a=sgn(),b=p->sgn(),(a^b?this:p)->rot(a),rot(b);
47          }
48          upd();
49          return this;
50      }
51
52      void insert(node *z){
53          node *x=this,*y; while (x != NIL){
54              y = x, x = x->c[z->ky>x->ky];
55          }
56          y->setc(z->ky>y->ky, z);
57          z->splay();
58      }
59
60  } *NIL, *T[N];
61
62  node*merge(node *y, node *x){
63      if(x==NIL)return y;
64      y = merge(y, lx), y = merge(y, rx);
65      lx = rx = NIL, y->insert(x);
66      return x;
67  }
68
69  #undef l
70  #undef r
71  #undef lx
72  #undef rx
73
74  } using namespace Splay;
75
76
77  namespace KMP{
78      void get_pi(const char P[], int n, int pi[]){
79          for (int i = 2, j = pi[1] = 0; i <= n; ++i){
80              while (j && P[i] != P[j+1]) j = pi[j];
81              if (P[i] == P[j+1]) ++j;
82              pi[i] = j;
83          }
84      }
85  } using namespace KMP;
86
87
88  namespace SAM{
89
90      int trans[N][C], fail[N], len[N], cnt[N], tail, tot;
91      char str[N/2]; int n, pi[N], ll[N], rr[N], dd[N], ml[N];
92
93      inline int new_node(){
94          RST(trans[tot]); cnt[tot] = 1; tail = tot;
95          return tot++;
96      }
97
98      inline int new_node(int u){
99          CPY(trans[tot], trans[u]); fail[tot] = fail[u], cnt[tot] = 0;
100         return tot++;
101     }
102
103 #define v trans[u][c]
104 #define f fail[u]
```

```
105 #define ff fail[uu]
106
107     void Ext(int c){
108         int u = tail, uu = new_node(); len[uu] = len[u] + 1;
109         while (u && !v) v = uu, u = f;
110         if (!u && !v) v = uu, ff = 0;
111         else{
112             if (len[v] == len[u] + 1) ff = v;
113             else{
114                 int _v = v, vv = new_node(_v); len[vv] = len[u] + 1;
115                 fail[_v] = ff = vv;
116                 while (v == _v) v = vv, u = f;
117             }
118         }
119     }
120
121     void Init(){
122         tot = 0, tail = new_node();
123     }
124
125     int Q[N], CC[N/2];
126
127     void Topo(int*key){
128         memset(CC, 0, sizeof(int)*(len[tail]+1));
129         REP(i, tot) ++CC[key[i]];
130         REP_1(i, len[tail]) CC[i] += CC[i-1];
131         REP(i, tot) Q[--CC[key[i]]] = i;
132     }
133
134     void Run(){
135
136         REP(u, tot) T[u] = cnt[u] ? new node(len[u]) : NIL;
137
138         Topo(len);
139
140         FOR(i, 1, tot){
141             int u = Q[i];
142             pi[u] = cnt[fail[u]] ? len[fail[u]] : pi[fail[u]];
143         }
144
145         DWN(i, tot, 1){
146             int u = Q[i]; if (!cnt[u]) continue;
147             ll[u] = T[u]->ll; rr[u] = T[u]->rr; dd[u] = T[u]->dd;
148             T[f] = cnt[f] > cnt[u] ? merge(T[f], T[u]) : merge(T[u], T[f]);
149             cnt[f] += cnt[u];
150         }
151
152         Topo(rr);
153     }
154
155 //#undef v
156 //#undef f
157 //#undef ff
158
159 } using namespace SAM;
160
161
162 namespace Segment_Tree{
163
164     const int NN = 4 * N;
165
166 #define lx (x<<1)
167 #define rx (lx|1)
168 #define ml (l + r >> 1)
169 #define mr (ml + 1)
170 #define lc lx, l, ml
171 #define rc rx, mr, r
```

```
172 #define root 1, 0, n-1
173
174     int T[NN], M[NN], a, b, cur, ss, mm; VI adj[N/2];
175
176     inline void Build(int x, int l, int r){
177         T[x] = M[x] = 0; if (l < r) Build(lc), Build(rc);
178     }
179
180     inline void Insert(int x, int l, int r){
181         ++T[x], checkMax(M[x], a); if (l == r) return;
182         if (a < mr) Insert(lc); else Insert(rc);
183     }
184
185     inline void Query(int x, int l, int r){
186         if (b < l || r < a) return;
187         if (a <= l && r <= b) ss += T[x], checkMax(mm, M[x]);
188         else Query(lc), Query(rc);
189     }
190
191     void Insert(int _a){
192         a = _a; Insert(root);
193     }
194
195     void Query(int _a, int _b){
196         a = _a, b = _b, ss = 0, mm = 0;
197         Query(root);
198     }
199
200     void Move(int tar){
201         while (cur <= tar){
202             ECH(it, adj[cur]) Insert(*it);
203             ++cur;
204         }
205     }
206
207     void STInit(){
208         //REP(i, n) CLR(adj[i]);
209         //cur = 0;
210     }
211
212 #undef ml
213
214 } using namespace Segment_Tree;
215
216
217 namespace SHash{
218
219     uLL S[N], P[N];
220     LL ans; int minLen;
221
222     uLL h(int a, int b){
223         return S[b]-S[a-1]*P[b-a+1];
224     }
225
226     void init(){
227         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);
228         ans = 0, minLen = n;
229     }
230
231     void jud(int &p1, int p2){
232         int l = 0, r = minLen; while(l<r){
233             int m = l+r>>1;
234             if (h(p1,p1+m)==h(p2,p2+m)) l = m+1;
235             else r = m;
236         }
237         if (str[p2+l]<str[p1+l]) p1 = p2;
238     }
```

```
239
240 } using namespace SHash;
241
242
243 int main(){
244
245 #ifndef ONLINE_JUDGE
246     freopen("in.txt", "r", stdin);
247     //freopen("out2.txt", "w", stdout);
248 #endif
249
250     NIL = new node(); NIL->reset();
251
252     n = strlen(RS(str+1)); reverse(str+1, str+n+1); get_pi(str, n, pi); reverse(str+1, str+n+1);
253     REP_1(i, n) adj[n-pi[i]].PB(n-i); Init(); REP_1(i, n) Ext(str[i]-'a'); Run();
254
255     init(); FOR(i, 1, tot){
256
257         int u = Q[i], L = max(ll[u]-pi[u],dd[u],len[f]+1), R=len[u]; if(L>R) continue;
258
259         if(rr[u] == n){
260             ans += R - L + 1, ml[u] = L;
261         }
262         else{
263             Move(rr[u]); int l = rr[u]-R, r = rr[u]-L; Query(l, r); if (!ss) continue;
264             ans += ss, ml[u] = rr[u] - mm;
265         }
266
267         checkMin(minLen, ml[u]);
268     }
269
270     OT(ans);
271
272     int st, u; FOR_N(u, 1, tot) if (ml[u] == minLen){st = ll[u]-minLen+1; break;}
273     FOR_N(u, u+1, tot) if (ml[u] == minLen) jud(st, ll[u]-minLen+1);
274     FOR(i, st, st+minLen) putchar(str[i]); puts("");
275 }
```

Part VI

数学 (Math)

```
1           // <<= '2. Number Theory .,//{
2   namespace NT{
3   inline LL gcd(LL a, LL b){return __gcd(a, b);}
4   inline LL lcm(LL a, LL b){return a*b/gcd(a,b);}
5
6   inline void INC(int &a, int b){a += b; if (a >= MOD) a -= MOD;}
7   inline int sum(int a, int b){a += b; if (a >= MOD) a -= MOD; return a;}
8   inline void DEC(int &a, int b){a -= b; if (a < 0) a += MOD;}
9   inline int dff(int a, int b){a -= b; if (a < 0) a += MOD; return a;}
10  inline void MUL(int &a, int b){a = (LL)a * b % MOD;}
11  inline int pdt(int a, int b){return (LL)a * b % MOD;}
12
13  inline int gcd(int m, int n, int &x, int &y){
14
15      x = 1, y = 0; int xx = 0, yy = 1, q;
16
17      while (1){
18          q = m / n, m %= n;
19          if (!m){x = xx, y = yy; return n;}
20          DEC(x, pdt(q, xx)), DEC(y, pdt(q, yy));
21          q = n / m, n %= m;
22          if (!n) return m;
23          DEC(xx, pdt(q, x)), DEC(yy, pdt(q, y));
24      }
25  }
26
27  inline int sum(int a, int b, int c){return sum(sum(a, b), c);}
28  inline int sum(int a, int b, int c, int d){return sum(sum(a, b), sum(c, d));}
29  inline int pdt(int a, int b, int c){return pdt(pdt(a, b), c);}
30  inline int pdt(int a, int b, int c, int d){return pdt(pdt(pdt(a, b), c), d);}
31
32  inline int pow(int a, LL b){
33      int c(1); while (b){
34          if (b&1) MUL(c, a);
35          MUL(a, a), b >>= 1;
36      }
37      return c;
38  }
39
40  template<class T> inline T pow(T a, LL b){
41      T c(1); while (b){
42          if (b&1) c *= a;
43          a *= a, b >>= 1;
44      }
45      return c;
46  }
47
48  template<class T> inline T pow(T a, int b){
49      return pow(a, (LL)b);
50  }
51
52  inline int _I(int b){
53      int a = MOD, x1 = 0, x2 = 1, q;
54      while (true){
55          q = a / b, a %= b;
56          if (!a) return (x2 + MOD) % MOD;
57          DEC(x1, pdt(q, x2));
58
59          q = b / a, b %= a;
60          if (!b) return (x1 + MOD) % MOD;
61          DEC(x2, pdt(q, x1));
62      }
63  }
64
65  inline void DIV(int &a, int b){MUL(a, _I(b));}
66  inline int qtt(int a, int b){return pdt(a, _I(b));}
```

```
67
68  struct Int{
69      int val;
70
71      operator int() const{return val;}
72
73      Int(int val = 0):val(val){
74          val %= MOD; if (val < 0) val += MOD;
75      }
76      Int(LL _val){
77          _val %= MOD; if (_val < 0) _val += MOD;
78          val = _val;
79      }
80      inline Int& operator +=(const int& rhs){
81          INC(val, rhs);
82          return *this;
83      }
84      inline Int operator +(const int& rhs) const{
85          return sum(val, rhs);
86      }
87      inline Int& operator -=(const int& rhs){
88          DEC(val, rhs);
89          return *this;
90      }
91      inline Int operator -(const int& rhs) const{
92          return dff(val, rhs);
93      }
94      inline Int& operator *=(const int& rhs){
95          MUL(val, rhs);
96          return *this;
97      }
98      inline Int operator *(const int& rhs) const{
99          return pdt(val, rhs);
100     }
101     inline Int& operator /=(const int& rhs){
102         DIV(val, rhs);
103         return *this;
104     }
105     inline Int operator /(const int& rhs) const{
106         return qtt(val, rhs);
107     }
108 };
109
110 /*
111 Int Fact[N], Factt[N]; Int Binom(int n, int m){
112     return Fact[n] * Factt[m] * Factt[n-m];
113 }
114 */
115
116 /*
117     Fact[0] = 1; REP_1(i, N-1) Fact[i] = Fact[i-1] * i;
118     Factt[N-1] = _I(Fact[N-1]); DWN(i, N, 1) Factt[i-1] = Factt[i] * i;
119 */
120
121 /*
122     int Binom[N][N];
123     REP(i, N){Binom[i][0] = 1; REP_1(j, i) Binom[i][j] = Binom[i-1][j-1] + Binom[i-1][j];}
124 */
125
126
127 /*
128 const int PMAX = 1;
129 VI P; bitset<PMAX> isP;
130 void sieve(){
131     FOR(i, 2, PMAX){
132         if (!isP[i]) P.PB(i);
133         for (int j=0;j<SZ(P)&&i*P[j]<PMAX;++j){
```

```
134                 isP[i*P[j]]=1; if (!(i%P[j])) break;
135             }
136         }
137 }
138 */
139
140 /*
141 inline int phi(int n){
142     int res = n; for (int i=2;sqr(i)<=n;++i) if (!(n%i)){
143         DEC(res, qtt(res, i));
144         do{n /= i;} while(!(n%i));
145     }
146     if (n != 1)
147         DEC(res, qtt(res, n));
148     return res;
149 }
150 */
151
152 /*LL d, x, y; void exGcd(LL a, LL b){
153     if(!b) x = 1, y = 0, d = a;
154     else{
155         exGcd(b, a%b); LL t = y;
156         y = x - (a/b)*y, x = t;
157     }
158 }*/
159
160 } using namespace NT;//}
```

..

Part VII

计算几何 (Computational Geometry)

Chapter 19

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);}
        ..
```

19.1 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;}
```

```
31      Po&rot(DB a,cPo o=Po()){Ts-=o;Ts*=Po(cos(a),sin(a));rTs+=o;}
32  };
33
34      ...
35
36  } using namespace CG;
```

19.2 点积 && 叉积

```
1   inline DB dot(DB x1,DB y1,DB x2,DB y2){return x1*x2+y1*y2;}
2   inline DB dot(cPo a,cPo b){return dot(a.x,a.y,b.x,b.y);}
3   inline DB dot(cPo p0,cPo p1,cPo p2){return dot(p1-p0,p2-p0);}
4   inline DB det(DB x1,DB y1,DB x2,DB y2){return x1*y2-x2*y1;}
5   inline DB det(cPo a,cPo b){return det(a.x,a.y,b.x,b.y);}
6   inline DB det(cPo p0,cPo p1,cPo p2){return det(p1-p0,p2-p0);}
7   inline DB ang(cPo p0,cPo p1){return acos(dot(p0,p1)/p0.len()/p1.len());}
8   inline DB ang(cPo p0,cPo p1,cPo p2){return ang(p1-p0,p2-p0);}
9   inline DB ang(cPo p0,cPo p1,cPo p2,cPo p3){return ang(p1-p0,p3-p2);}
10  inline DB dist2(const Po &a, const Po &b){return dist2(a.x-b.x, a.y-b.y);}
11  template<class T1, class T2> inline int dett(const T1 &x, const T2 &y){return sgn(det(x, y));}
12  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));}
13  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));}
14  template<class T1, class T2> inline int dott(const T1 &x, const T2 &y){return sgn(dot(x, y));}
15  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));}
16  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));}
17  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;}
18  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;}
19  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;}
20  template<class T1, class T2> inline DB dist(const T1 &x, const T2 &y){return sqrt(dist2(x, y));}
21  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));}
22  inline Po _1(Po p){return p._1();}inline Po conj(Po p){return p.conj();}
23  inline Po lt(Po p){return p.lt();}inline Po rt(Po p){return p.rt();}
24  inline Po rot(Po p,DB a,cPo o=Po()){return p.rot(a,o);}
25  inline Po operator *(DB k,cPo p){return p*k;}
26  inline Po operator /(DB k,cPo p){return conj(p)*k/p.len2();}
27
28  typedef vector<Po> VP;
```

19.3 直线

```
1   struct Line{
2       Po a,b;Line(cPo a=Po(),cPo b=Po()):a(a),b(b){}
3       Line(DB x0,DB y0,DB x1,DB y1):a(Po(x0,y0)),b(Po(x1,y1)){}
4       Line(cLine l):a(l.a),b(l.b){}
5
6       //Ax+By+C=0
7       Line(DB A,DB B,DB C){
8           C=-C;if(!::sgn(A))a=Po(0,C/B),b=Po(1,C/B);
9           else if(!::sgn(B))a=Po(C/A,0),b=Po(C/A,1);
10          else a=Po(0,C/B),b=Po(1,(C-A)/B);
11      }
12
13      void in(){a.in(),b.in();}
14      inline friend istream&operator>>(istream&i,Line& p){return i>>p.a>>p.b;}
15      inline friend ostream&operator<<(ostream&o,Line p){return o<<p.a<<"-"<< p.b;}
16
17      Line operator+(cPo x)const{return Line(a+x,b+x);}
18      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);}
```

19.4 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;
};

 // 不相交-1 相交（不规范）0 相交（规范）1

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);
}
```

19.5 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);}
```

```
28      void out(){
29          printf("%.2f %.2f %.2f\n", o.x, o.y, r);
30      }
31      bool operator <(cCircle c)const{return r<c.r;}
32      //相离-1 圆上0 包含1
33      inline int sgn(cPo p)const{return ::sgn(r*r, dist2(o, p));}
34      //相离-1 相切0 包含1
35      inline int sgn(cLine l)const{return ::sgn(r*r, dist2(l, o));}
36      // 外离-2 外切-1 相交0 内切1 包含2
37      inline int sgn(cCircle c)const{
38          DB d=dist2(o,c.o);
39          if (::sgn(sqr(r+c.r),d)<0) return Disjoint;
40          if (!::sgn(sqr(r+c.r), d)) return Exscribe;
41          if (!::sgn(sqr(r-c.r), d)) return Inscribe;
42          if (::sgn(sqr(r-c.r), d)>0) return Contain;
43          return Cross;
44      }
45
46      inline DB s(){return PI*sqr(r);}
47      inline DB p(){return 2*PI*r;}
48
49      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);}
50
51      inline void getIntersect(cLine l,Po&p0,Po&p1)const{
52          Po m = o&l, d = (l.b-l.a)._1() * sqrt(sqr(r)-dist2(l, o));
53          p0 = m + d, p1 = m - d;
54      }
55      inline void getIntersect(cCircle c,Po&p0,Po&p1)const{
56          Po v=(c.o-o)._1()*r;DB a=acos(cos(r,dist(o,c.o),c.r));
57          p0=o+rot(v,a),p1=o+rot(v,-a);
58      }
59
60      inline VP operator*(cLine l)const{
61          VP P; int t = sgn(l); if (t==-1) return P;
62          Po p0, p1; getIntersect(l, p0, p1); P.PB(p0); if (t == 1) P.PB(p1);
63          return P;
64      }
65
66      inline VP operator*(cSeg s)const{
67          VP _P = Ts*Line(s), P; ECH(p, _P) if (~s.sgn(*p)) P.PB((*p));
68          return P;
69      }
70
71      inline VP operator*(cCircle c)const{
72          VP P; int t = abs(sgn(c)); if (t == 2) return P;
73          Po p0, p1; getIntersect(c, p0, p1); P.PB(p0); if (!t) P.PB(p1);
74          return P;
75      }
76
77      inline void getTangency(cPo p,Po&p0,Po&p1)const{
78          DB d=dist(o,p),a=acos(r/d);Po v=(p-o)._1()*r;
79          p0=o+rot(v,a),p1=o+rot(v,-a);
80      }
81  };
82
83  struct Triangle{
84      Po A,B,C; DB a,b,c; DB alpha,beta,theta;
85      DB r,R; DB S,P; Po I,G,O,H;
86
87      void init(){
88          S=fabs(det(A,B,C))/2,a=dist(B,C),b=dist(A,C),c=dist(A,B);
89          alpha=acos(cos(b,c,a)),beta=acos(cos(a,c,b)),theta=acos(cos(a,b,c));
90          P=a+b+c,R=(a*b*c)/(4*S),r=2*S/P;
91          I=Po(a*A.x+b*B.x+c*C.x,a*A.y+b*B.y+c*C.y)/P;
92          G=(A+B+C)/3,O=getX3(A,B,C),H=getX4(A,B,C);
93          //DB s=P/2; assert(!sgn(S, sqrt(s*(s-a)*(s-b)*(s-c)))); // 海伦公式
94          //assert(!sgn(dist(I,O), sqrt(R*(R-2*r))));
```

```
95          //assert(!sgn(dist(H,G), dist(O,H)*2/3));
96      }
97
98      void in(){
99          A.in(),B.in(),C.in();init();
100     }
101 };
```

19.5.1 最小覆盖圆

```
1   Circle getMinimalCoverCircle(VP& P){ //#
2       random_shuffle(ALL(P)); int n = SZ(P);
3       Circle C(P[0]); FOR(i, 1, n) if (!~C.sgn(P[i])){
4           C = Circle(P[i]); REP(j, i) if (!~C.sgn(P[j])){
5               C = Circle(P[i], P[j]); REP(k, j) if (!~C.sgn(P[k])){
6                   C = Circle(P[i], P[j], P[k]);
7               }
8           }
9       }
10      return C;
11  }
```

19.6 多边形

```
1   Po getPo(){Po p;p.in();return p;}
2   Line getLine(){Line l;l.in();return l;}
3
4   DB getArea(const VP& P){DB z=0;FOR(i,1,SZ(P))z+=det(P[i-1],P[i]);return z;}
5   DB getPeri(const VP& P){DB z=0;FOR(i,1,SZ(P))z+=dist(P[i-1],P[i]);return z;}
```

19.6.1 凸多边形面积并

```
1   struct Polygon{
2       VP P;
3       void input();
4   };
5
6   inline bool equal(const pair<DB, DB>& lhs, cSeg rhs){
7       DB k, b; rhs.getEquation(k, b);
8       return !sgn(k, lhs.fi) && !sgn(b, lhs.se);
9   }
10
11  DB getUnion(vector<Polygon>& P, vector<Seg>& S){
12
13      vector<pair<DB,DB> > L; ECH(Si, S){
14          DB k, b; Si->getEquation(k, b);
15          L.PB(MP(k, b));
16      }
17
18      UNQ(L); DB res = 0; ECH(Li, L){
19
20          vector<pair<DB, int> > I;
21          Line l0(0,Li->se,1,Li->fi+Li->se);
22
23          ECH(Pi, P){
24              int i; FOR_N(i, 1, SZ(Pi->P)) if (equal(*Li, Seg(Pi->P[i-1], Pi->P[i]))) break;
25              if (i != SZ(Pi->P)) continue;
26
27              VP cut; FOR_N(i, 1, SZ(Pi->P)){
28                  Seg l1(Pi->P[i-1], Pi->P[i]); if (!dett(l0,l1)) continue;
29                  Po p=l0*l1; if (~l1.sgn(p)) cut.PB(p);
```

```
30              }
31
32              if (SZ(UNQ(cut)) == 2){
33                  I.PB(MP(cut[0].x, 1));
34                  I.PB(MP(cut[1].x, -1));
35              }
36          }
37
38          ECH(Si, S) if (equal(*Li, *Si)){
39              I.PB(MP(min(Si->a.x, Si->b.x), 2));
40              I.PB(MP(max(Si->a.x, Si->b.x), -2));
41          }
42  #define h (I[i].fi-I[i-1].fi)
43  #define y0 (Li->fi * I[i-1].fi + Li->se)
44  #define y1 (Li->fi * I[i].fi + Li->se)
45          SRT(I); int c0 = 0, c1 = 0; REP(i, SZ(I)){
46              if (!c0 && c1) res += (y0+y1)*h;
47              if (abs(I[i].se)==1) c0 += I[i].se;
48              else c1 += I[i].se;
49          }
50  #undef h
51  #undef y0
52  #undef y1
53      }
54
55      return res;
56  }
57
58  DB getUnion(vector<Polygon>& P){
59      vector<Seg> up, down; ECH(it, P){
60          FOR(i, 1, SZ(it->P)){
61              Seg s(it->P[i-1], it->P[i]); int t = sgn(s.a.x, s.b.x);
62              if (t > 0) up.PB(s); else if (t < 0) down.PB(s);
63          }
64      }
65      return getUnion(P, up) - getUnion(P, down);
66  }
```

Chapter 20

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;
}
```

20.0.2 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);
```

```
30      }
31      bool operator <(cCircle c)const{return r<c.r;}
32      //相离-1 圆上0 包含1
33      inline int sgn(cPo p)const{return ::sgn(r*r, dist2(o, p));}
34      //相离-1 相切0 包含1
35      inline int sgn(cLine l)const{return ::sgn(r*r, dist2(l, o));}
36      // 外离-2 外切-1 相交0 内切1 包含2
37      inline int sgn(cCircle c)const{
38          DB d=dist2(o,c.o);
39          if (::sgn(sqr(r+c.r),d)<0) return Disjoint;
40          if (!::sgn(sqr(r+c.r), d)) return Exscribe;
41          if (!::sgn(sqr(r-c.r), d)) return Inscribe;
42          if (::sgn(sqr(r-c.r), d)>0) return Contain;
43          return Cross;
44      }
45
46      inline DB s(){return PI*sqr(r);}
47      inline DB p(){return 2*PI*r;}
48
49      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);}
50
51      inline void getIntersect(cLine l,Po&p0,Po&p1)const{
52          Po m = o&l, d = (l.b-l.a)._1() * sqrt(sqr(r)-dist2(l, o));
53          p0 = m + d, p1 = m - d;
54      }
55      inline void getIntersect(cCircle c,Po&p0,Po&p1)const{
56          Po v=(c.o-o)._1()*r;DB a=acos(cos(r,dist(o,c.o),c.r));
57          p0=o+rot(v,a),p1=o+rot(v,-a);
58      }
59
60      inline VP operator*(cLine l)const{
61          VP P; int t = sgn(l); if (t==-1) return P;
62          Po p0, p1; getIntersect(l, p0, p1); P.PB(p0); if (t == 1) P.PB(p1);
63          return P;
64      }
65
66      inline VP operator*(cSeg s)const{
67          VP _P = Ts*Line(s), P; ECH(p, _P) if (~s.sgn(*p)) P.PB((*p));
68          return P;
69      }
70
71      inline VP operator*(cCircle c)const{
72          VP P; int t = abs(sgn(c)); if (t == 2) return P;
73          Po p0, p1; getIntersect(c, p0, p1); P.PB(p0); if (!t) P.PB(p1);
74          return P;
75      }
76
77      inline void getTangency(cPo p,Po&p0,Po&p1)const{
78          DB a=acos(r/dist(o,p)); Po op=(p-o)._1()*r;
79          p0=o+rot(op,a), p1=o+rot(op,-a);
80      }
81      inline void getTangency(cCircle c,Po&p0,Po&p1,Po&p2,Po&p3)const{
82          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;}
83          else{Po p=(*this)^c; getTangency(p,p0,p1), c.getTangency(p,p2,p3);}
84      }
85
86      inline DB arc(cPo a,cPo b){
87          //DB alpha = acos(cos(dist(a, o), dist(b, o), dist(a, b)));
88          //if (det(o,a,b)<0) alpha = 2*PI - alpha;
89          return arg(o,a,b) * r;
90      }
91  };
92
93
94  struct Triangle{
95      Po A,B,C; DB a,b,c; DB alpha,beta,theta;
96      DB r,R; DB S,P; Po I,G,O,H;
```

```
97
98      void init(){
99          S=fabs(det(A,B,C))/2,a=dist(B,C),b=dist(A,C),c=dist(A,B);
100         alpha=acos(cos(b,c,a)),beta=acos(cos(a,c,b)),theta=acos(cos(a,b,c));
101         P=a+b+c,R=(a*b*c)/(4*S),r=2*S/P;
102         I=Po(a*A.x+b*B.x+c*C.x,a*A.y+b*B.y+c*C.y)/P;
103         G=(A+B+C)/3,O=getX3(A,B,C),H=getX4(A,B,C);
104         //DB s=P/2; assert(!sgn(S, sqrt(s*(s-a)*(s-b)*(s-c)))); // 海伦公式
105         //assert(!sgn(dist(I,O), sqrt(R*(R-2*r))));
106         //assert(!sgn(dist(H,G), dist(O,H)*2/3));
107     }
108
109     void in(){
110         A.in(),B.in(),C.in(); //init();
111     }
112 };
113
114 Po getPo(){Po p;p.in();return p;}
115 Line getLine(){Line l;l.in();return l;}
116
117 DB getArea(const VP& P){DB z=0;FOR(i,1,SZ(P))z+=det(P[i-1],P[i]);return z;}
118 DB getPeri(const VP& P){DB z=0;FOR(i,1,SZ(P))z+=dist(P[i-1],P[i]);return z;}
119
120
121 VP getCH(VP& P){ //无共线
122
123     int n=SZ(P); if(n<=3) return P.PB(P[0]),getArea(P)<0?RVS(P):P;
124
125     SRT(P); VP C; C.resize(n+9); int nn = -1; REP(i, n){ //#
126         while (nn > 0 && dett(C[nn-1], C[nn], P[i]) <= 0) --nn; //#
127         C[++nn] = P[i];
128     }
129
130     int _nn = nn; DWN(i, n-1, 0){
131         while (nn > _nn && dett(C[nn-1], C[nn], P[i]) <= 0) --nn; //#
132         C[++nn] = P[i];
133     }
134
135     C.resize(nn+1);
136     return C;
137 }
138 ...
139 const int N = 109;
140
141 Circle C[N]; int Cn, Tn; VP P;
142
143 DB f(const VP& P){
144     int n = SZ(P); VI id; id.resize(n, -1);
145
146     REP_2(i, j, SZ(P), Cn) if (!C[j].sgn(P[i])){
147         id[i] = j; break;
148     }
149
150     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]);
151
152     return res;
153 }
154
155 void add(const Po&p, const Circle&c){
156     Po p0, p1; c.getTangency(p, p0, p1);
157     P.PB(p0), P.PB(p1);
158 }
159
160 void add(const Circle&c0, const Circle&c1){
161     Po p0, p1, p2, p3; c0.getTangency(c1, p0, p1, p2, p3);
162     P.PB(p0), P.PB(p1), P.PB(p2), P.PB(p3);
163 }
```

```
164
165 int main(){
166
167 #ifndef ONLINE_JUDGE
168     freopen("in.txt", "r", stdin);
169     //freopen("out.txt", "w", stdout);
170 #endif
171
172     while(~scanf("%d%d", &Cn, &Tn)){
173
174         CLR(P); REP(i, Cn) C[i].in();
175
176         DO(Tn){
177             Po a, b, c; a.in(), b.in(), c.in(); P.PB(a), P.PB(b), P.PB(c);
178             REP(i, Cn) add(a, C[i]), add(b, C[i]), add(c, C[i]);
179         }
180
181         REP_2(j, i, Cn, j) add(C[i], C[j]);
182         OT(SZ(P) ? f(getCH(P)) : C[0].p());
183     }
184 }
```

20.0.3 线性动态凸包

```
1  const int N = 50;
2  Po P0[N]; DB A[N], B[N];
3  int n;
4
5  DB s(DB t, int &m){
6      VP P; REP(i, n) P.PB( P0[i] + Po(t * A[i], t * B[i]));
7      m = SZ(P);
8      return fabs(getArea(getConvexHull(P)));
9  }
10 DB f(DB x, DB a, DB b, DB c){
11     return a*x*x*x/3+b*x*x/2+c*x;
12 }
13 DB f(DB r, DB l, DB a, DB b, DB c){
14     return a*(r*r*r-l*l*l)/3+b*(r+l)*(r-l)/2+c*(r-l);
15 }
16
17 DB s(DB l, DB r){
18
19     if (r - l < EPS) return 0;
20
21     DB m = (l + r) / 2; int nl, nm, nr;
22     DB sl = s(l, nl), sm = s(m, nm), sr = s(r, nr);
23
24     if (nl != nr){
25         return s(l, m) + s(m, r);
26     }
27
28     DB a = sl / (l - r) / (l - m)
29             + sm / (m - l) / (m - r)
30             + sr / (r - l) / (r - m);
31     DB b = sl * (-m-r) / (l - r) / (l - m)
32             + sm * (-l-r) / (m - l) / (m - r)
33             + sr * (-l-m) / (r - l) / (r - m);
34     DB c = sl * (m*r) / (l - r) / (l - m)
35             + sm * (l*r) / (m - l) / (m - r)
36             + sr * (l*m) / (r - l) / (r - m);
37
38     //cout << sl << " "<< sm << " " << sr << endl;
39
40     //cout << f(r, a,b,c) - f(l, a,b,c) << endl;
41     return f(r,a,b,c) - f(l,a,b,c);
```

```
42      //return f(r,l, a,b,c);
43
44      //return sm * (r - l);
45  }
46
47  int main(){
48
49  #ifndef ONLINE_JUDGE
50      freopen("in.txt", "r", stdin);
51      //freopen("out.txt", "w", stdout);
52  #endif
53
54      int T; while (~scanf("%d%d", &n, &T)){
55          REP(i, n) P0[i].in(), RF(A[i], B[i]);
56
57          int C0 = 3000;
58          DB d = (DB)T/C0; DB res = 0, st = 0, ed = d;
59
60          DO(C0){
61              res += s(st, ed); //cout << st << " "<< ed << endl;
62              st = ed, ed += d;
63          }
64
65          OT(res / 2 / T);
66      }
67
68  }
69
70
71  const int N = 50;
72  Po P0[N]; DB A[N], B[N];
73  int n;
74
75  DB s(DB t){
76      VP P; REP(i, n) P.PB(P0[i] + Po(t * A[i], t * B[i]));
77      return fabs(getArea(getConvexHull(P)));
78  }
79
80  DB f(DB x, DB a, DB b, DB c){
81      return a*x*x*x/3+b*x*x/2+c*x;
82  }
83  DB f(DB r, DB l, DB a, DB b, DB c){
84      return a*(r*r*r-l*l*l)/3+b*(r+l)*(r-l)/2+c*(r-l);
85  }
86
87  DB _sl; DB s(DB l, DB r){
88
89      DB m = (l + r) / 2, sl = _sl, sm = s(m), sr = s(r);
90
91      DB a = sl / (l - r) / (l - m)
92              + sm / (m - l) / (m - r)
93              + sr / (r - l) / (r - m);
94      DB b = sl * (-m-r) / (l - r) / (l - m)
95              + sm * (-l-r) / (m - l) / (m - r)
96              + sr * (-l-m) / (r - l) / (r - m);
97      DB c = sl * (m*r) / (l - r) / (l - m)
98              + sm * (l*r) / (m - l) / (m - r)
99              + sr * (l*m) / (r - l) / (r - m);
100
101     _sl = sr;
102     return f(r, l,a,b,c) ;// - f(l, a,b,c);
103 }
104
105 int T;
106 #define P P0
107 void add(vector<DB> &I, int i, int j, int k){
108
```

```
109     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];
110     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)
111         - (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);
112     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);
113
114     if (!sgn(a)){
115         if (!sgn(b)) return;
116         DB x = -c/b; if (0 < x && x < T) I.PB(x);
117         return;
118     }
119
120     DB d = b*b - 4*a*c; if (sgn(d) < 0) return; d = sqrt(d); a *= 2;
121     DB x = (-b+d)/a; if (0 < x && x < T) I.PB(x);
122     x = (-b-d)/a; if (0 < x && x < T) I.PB(x);
123 }
124 #undef P
125
126 int main(){
127
128 #ifndef ONLINE_JUDGE
129     freopen("in.txt", "r", stdin);
130     //freopen("out.txt", "w", stdout);
131 #endif
132
133     while (~scanf("%d%d", &n, &T)){
134         REP(i, n) P0[i].in(), RF(A[i], B[i]);
135
136         vector<DB> I; I.PB(0), I.PB(T);
137
138         REP(i, n) FOR(j, i+1, n) FOR(k, j+1, n){
139             add(I, i, j, k);
140         }
141
142         _sl = s(0); UNQ(I);
143
144         //REP(i,SZ(I)) cout << I[i] << " "; cout << endl;
145
146         DB res = 0; FOR(i, 1, SZ(I)){
147             res += s(I[i-1], I[i]);
148         }
149
150         OT(res/T/2);
151     }
152
153 }
```

Chapter 21

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 22

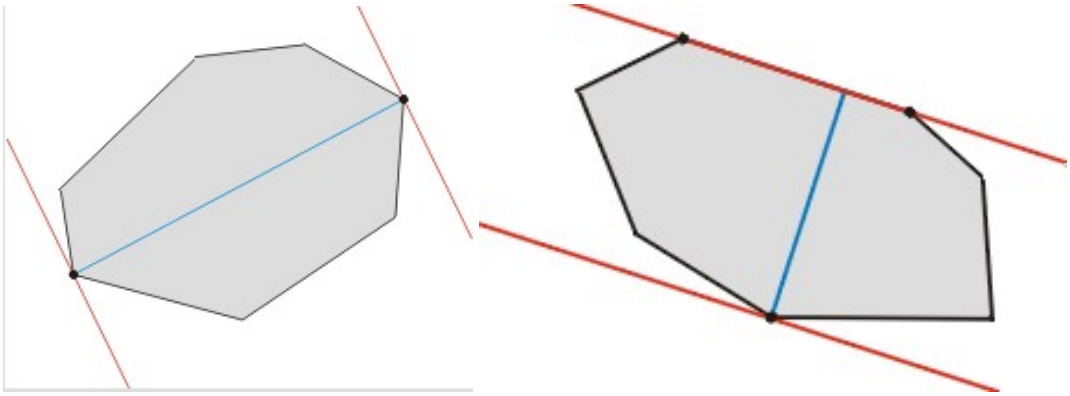
旋转卡壳

```
1  #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。

22.1 计算距离（Computing distances）

22.1.1 凸包的直径与宽度



```
1  DB rc(const VP& P){
2      int n = SZ(P)-1, j = 1; DB d2 = 0; REP(i, n){
3          while (dett(P[i+1]-P[i], P[j+1]-P[j])>0) j=suc(j);
4          checkMax(d2, max(dist2(P[i], P[j]), dist2(P[i+1], P[j+1])));
5      }
6      return d2;
7  }
```

```
1  DB rc(const VP& P){
2      int n = SZ(P)-1, j = 1; DB w2 = OO; REP(i, n){
3          while (dett(P[i+1]-P[i], P[j+1]-P[j])>0) j=suc(j);
4          checkMin(w2, dist2(Line(P[i], P[i+1]), P[j]));
5      }
6      return w2;
7  }
```

22.1.2 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，同时存在诸多派生。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，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。

最小距离

最大距离

22.2 外接矩形（Enclosing rectangles）

22.2.1 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;
}
```

22.3 三角分解（Triangulations）

22.4 凸多边形性质（Properties of convex polygons）

22.5 例题 (E.g.)

22.5.1 HDU 3847. Trash Removal

简述 (Brief description)

略。）

分析 (Analysis)

凸包宽度。

Chapter 23

3D-几何基础

```
1
2  inline DB dist2(DB x,DB y,DB z){return dist2(x,y)+sqr(z);}
3
4  namespace D3{
5      struct Po{
6          DB x,y,z;Po(DB x=0,DB y=0,DB z=0):x(x),y(y),z(z){}
7          void in(){RF(x,y,z);}
8
9          Po operator-()const{return Po(-x,-y,-z);}
10         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;}
11         Po&operator*=(DB k){x*=k,y*=k,z*=k;rTs;}Po&operator/=(DB k){x/=k,y/=k,z/=k;rTs;}
12         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);}
13         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);}
14
15         DB len2()const{return dist2(x,y,z);}DB len()const{return sqrt(len2());}
16         Po&_1(){rTs/=len();}
17     };
18
19     inline DB dot(DB x1,DB y1,DB z1,DB x2,DB y2,DB z2){return CG::dot(x1,y1,x2,y2)+z1*z2;}
20     inline DB dot(cPo a,cPo b){return dot(a.x,a.y,a.z,b.x,b.y,b.z);}
21     inline DB dot(cPo p0,cPo p1,cPo p2){return dot(p1-p0,p2-p0);}
22     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))
           ;}
23     inline Po det(cPo a,cPo b){return det(a.x,a.y,a.z,b.x,b.y,b.z);}
24     inline Po det(cPo p0,cPo p1,cPo p2){return det(p1-p0,p2-p0);}
25
26     struct Line{
27         Po a,b;
28     };
29 };
```

Chapter 24

海岸线

Part VIII

补充 (More)

Chapter 25

倍增祖先

25.1 例题 (E.g.)

25.1.1 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)

。。。

```
1
2  const int N = int(1e5) + 9, LV = 20;
3
4  int dep[N], fa[LV][N];
5  int n;
6
7  int up(int x, int d){
8      REP(lv, LV){
9          if (d&1) x = fa[lv][x];
10         d >>= 1;
11     }
12     return x;
13 }
14
15 int lca(int x, int y){
16     if (dep[x] > dep[y]) x = up(x, dep[x]-dep[y]); else y = up(y, dep[y]-dep[x]);
17     if (x == y) return x;
18     else{
19         DWN(lv, LV, 0) if (fa[lv][x] != fa[lv][y])
20             x = fa[lv][x], y = fa[lv][y];
21         return fa[0][x];
22     }
23 }
24
25 int dist(int x, int y){
26     return dep[x] + dep[y] - dep[lca(x, y)]*2;
27 }
28
29 int main(){
30
31 #ifndef ONLINE_JUDGE
32     freopen("in.txt", "r", stdin);
33     //freopen("out.txt", "w", stdout);
34 #endif
35
36     Rush{
37
38         dep[1] = 0; FOR_1_C(i, 2, RD(n)) dep[i] = dep[RD(fa[0][i])] + 1;
39         if (n <= 1) continue;
```

```
40
41          REP_1(i, n) FOR(lv, 1, LV) fa[lv][i] = fa[lv-1][fa[lv-1][i]];
42
43          int p1 = 1, p2 = 1, di = 0; FOR_1(i, 2, n){
44              int l1 = dist(i, p1), l2 = dist(i, p2);
45              if (l1 > di) di = l1, p2 = i;
46              if (l2 > di) di = l2, p1 = i;
47              OT(di);
48          }
49      }
50  }
```

Chapter 26

树链剖分

26.1 例题 (E.g.)

26.1.1 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。

```
1  const int N = 100009, M = 2 * N, LV = 18;
2  UINT A[N]; int L[N], R[N], sz[N], up[N], dep[N], fa[LV][N], n, m, nn, rt;
3  int hd[N], suc[M], to[M]; UINT T[4*N]; int bj[4*N], a, b; UINT c;
4
5  inline int move_up(int x, int t){
6      REP(lv, LV){
7          if (t&1) x = fa[lv][x];
8          t >>= 1;
9      }
10     return x;
11 }
12
13 inline int lca(int a, int b){
14     if (dep[a] > dep[b]) a = move_up(a, dep[a] - dep[b]);
15     else b = move_up(b, dep[b] - dep[a]);
16
17     if (a == b) return a;
18     else {
19         DWN(lv, LV, 0) if (fa[lv][a] != fa[lv][b])
20             a = fa[lv][a], b = fa[lv][b];
21         return fa[0][a];
22     }
23 }
24
25 #define aa to[i^1]
26 #define bb to[i]
```

```
27  #define v bb
28
29  inline void dfs(int u = 1){
30      sz[u] = 1; REP_G(i, u) if (v != fa[0][u]){
31          dep[v] = dep[u] + 1, fa[0][v] = u;
32          FOR(lv, 1, LV) if (!(fa[lv][v] = fa[lv-1][fa[lv-1][v]])) break;
33          dfs(v), sz[u] += sz[v];
34      }
35  }
36
37  inline void hld(int u = 1, int t = 1){
38      L[u] = ++nn, up[u] = t;
39      int h = 0; REP_G(i, u) if (v != fa[0][u] && sz[v] > sz[h])
40          h = v;
41
42      if (h){
43          hld(h, t); REP_G(i, u) if (v != fa[0][u] && v != h)
44              hld(v, v);
45      }
46      R[u] = nn;
47  }
48
49  #define root 1, 1, n
50  #define lx (x<<1)
51  #define rx (lx|1)
52  #define ml (l + r >> 1)
53  #define mr (ml + 1)
54  #define xx x, l, r
55  #define lc lx, l, ml
56  #define rc rx, mr, r
57
58  inline void Update(int x, int l, int r){
59      T[x] = min(T[lx], T[rx]);
60  }
61
62  inline void Release(int x){
63      if (bj[x]){
64          T[lx] = T[rx] = T[x];
65          bj[lx] = bj[rx] = 1;
66          bj[x] = 0;
67      }
68  }
69
70  inline void Build(int x, int l, int r){
71      if (l == r){
72          T[x] = A[l];
73      }
74      else {
75          Build(lc), Build(rc);
76          Update(xx);
77      }
78  }
79
80  inline UINT Query(int x, int l, int r){
81      if (r < a || b < l) return -1;
82      if (a <= l && r <= b) return T[x];
83      Release(x);
84      return min(Query(lc), Query(rc));
85  }
86
87  inline void Modify(int x, int l, int r){
88      if (r < a || b < l) return;
89      if (a <= l && r <= b) T[x] = c, bj[x] = 1;
90      else {
91          Release(x);
92          Modify(lc), Modify(rc);
93          Update(xx);
```

```
94      }
95  }
96
97  inline void Modifyy(int z, int x){
98      while (up[z] != up[x]){
99          a = L[up[x]], b = L[x], Modify(root);
100         x = fa[0][up[x]];
101     }
102     a = L[z], b = L[x], Modify(root);
103 }
104
105 inline UINT Queryy(int l, int r){
106     a = l, b = r;
107     return Query(root);
108 }
109
110 int main(){
111
112 #ifndef ONLINE_JUDGE
113     freopen("in.txt", "r", stdin);
114     //freopen("out.txt", "w", stdout);
115 #endif
116
117     RD(n, m); FOR_C(i, 2, n << 1){
118         RD(aa, bb);
119         suc[i] = hd[aa], hd[aa] = i, ++i;
120         suc[i] = hd[aa], hd[aa] = i;
121     }
122
123     dfs(), hld(); REP_1(i, n) RD(A[L[i]]);
124     RD(rt); Build(root); DO(m){
125         int x, y, z; switch(RD()){
126             case 1:
127                 RD(rt);
128                 break;
129             case 2:
130                 RD(x, y, c); z = lca(x, y);
131                 Modifyy(z, x), Modifyy(z, y);
132                 break;
133             case 3:
134                 if (rt == RD(x)){
135                     OT(Queryy(1, n));
136                 }
137                 else if (L[x] <= L[rt] && L[rt] <= R[x]){
138                     x = move_up(rt, (dep[rt] - dep[x] - 1));
139                     OT(min(Queryy(1, L[x]-1), Queryy(R[x]+1, n)));
140                 }
141                 else {
142                     OT(Queryy(L[x], R[x]));
143                 }
144         }
145     }
146 }
```

Chapter 27

一类算法的复合方法

27.1 例题 (E.g.)

27.1.1 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。

```
1  const int N = int(2.5e5) + 9;
2  PII P[N]; int PP[N];
3  int n, nn; int Q; LL ans;
4
5  inline bool cmp(int a, int b){
6      return P[a].se < P[b].se || P[a].se == P[b].se && P[a].fi < P[b].fi;
7  }
8
9  inline LL C2(LL n){
10     return n*(n-1)/2;
11 }
12
13 int main(){
14
15 #ifndef ONLINE_JUDGE
16     freopen("in.txt", "r", stdin);
17     //freopen("out.txt", "w", stdout);
18 #endif
19
20     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);
21
22     for (int i=0,ii;i<n;i=ii){
23
24         ii = i+1; while (P[i].fi==P[ii].fi)++ii;
25
26         if (ii-i>Q){
27             set<int> H; int s=0; FOR(j, i, ii) H.insert(P[j].se);
28
29             for(int j=ii;j<n;++j){
30                 if (CTN(H, P[j].se)) ++s;
31                 if (P[j].fi!=P[j+1].fi) ans += C2(s), s = 0;
32             }
33             for(int j=0;j<nn;++j){
34                 if (CTN(H, P[j].se)) ++s;
35                 if (P[PP[j]].fi!=P[PP[j+1]].fi) ans += C2(s), s = 0;
36             }
37         }
38         else{
39             FOR(j, i, ii) PP[nn++] = j; PP[nn] = n;
```

```
40          }
41      }
42
43      sort(PP, PP+nn, cmp);
44
45      for (int i=0,ii;i<nn;i=ii){
46          ii = i+1; while(P[PP[i]].se==P[PP[ii]].se)++ii;
47          map<int, int> H; FOR(j, i, ii){
48              int t = P[PP[j]].fi;
49              for(int jj=PP[j]-1;jj>=0&&P[jj].fi==t;--jj) ++H[P[jj].se];
50          }
51          ECH(it, H) ans += C2(it->se);
52      }
53
54      OT(ans);
55  }
```

Chapter 28

一类树上的构造问题

Chapter 29

培养皿问题

29.1 例题 (E.g.)

29.1.1 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。。。。

```
1  const int N = 109;
2
3  struct Int{
4      int val;
5
6      operator int() const{return val;}
7
8      Int(int val = 0):val(val){
9          val %= MOD; if (val < 0) val += MOD;
10     }
11     inline Int& operator +=(const Int& rhs){
12         INC(val, rhs);
13         return *this;
14     }
15     inline Int operator +(const Int& rhs) const{
16         return sum(val, rhs.val);
17     }
18     inline Int operator -(const Int& rhs) const{
19         return dff(val, rhs);
20     }
21 };
22
```

```
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;
    }
};
```

29.1.2 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);
```

```
20      }
21  }
22
23  void gao(int i, int k, rec& p){
24      if (!k) return; p.output(i);
25      gao(i-1, k-p.len(), pr[i][k][p.b1][p.b2][p.l][p.r]);
26  }
27
28  int main() {
29
30  #ifndef ONLINE_JUDGE
31      freopen("in.txt", "r", stdin);
32      //freopen("out.txt", "w", stdout);
33  #endif
34
35      RD(n, m, k); REP(i, n){
36
37          REP(j, m) S[j+1] = S[j] + RD();
38
39          REP_1(_k, k) REP(l, m) FOR(r, l, min(m, l+_k)){
40
41              FOR_1(ll, l, r) FOR_1(rr, ll, r)
42                  upd(i, _k, 0, 0, 0, 0, l, r, ll, rr);
43
44              FOR_1(ll, l, r) FOR(rr, r+1, m)
45                  upd(i, _k, 0, 1, 0, 0, l, r, ll, rr);
46              FOR_1(ll, l, r) FOR(rr, r, m)
47                  upd(i, _k, 0, 1, 0, 1, l, r, ll, rr);
48
49              FOR(ll, 0, l) FOR_1(rr, l, r)
50                  upd(i, _k, 1, 0, 0, 0, l, r, ll, rr);
51              FOR_1(ll, 0, l) FOR_1(rr, l, r)
52                  upd(i, _k, 1, 0, 1, 0, l, r, ll, rr);
53
54              FOR(ll, 0, l) FOR(rr, r+1, m)
55                  upd(i, _k, 1, 1, 0, 0, l, r, ll, rr);
56              FOR(ll, 0, l) FOR(rr, r, m)
57                  upd(i, _k, 1, 1, 0, 1, l, r, ll, rr);
58              FOR_1(ll, 0, l) FOR(rr, r+1, m)
59                  upd(i, _k, 1, 1, 1, 0, l, r, ll, rr);
60              FOR_1(ll, 0, l) FOR(rr, r, m)
61                  upd(i, _k, 1, 1, 1, 1, l, r, ll, rr);
62
63          }
64      }
65
66      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){
67          res = dp[i][k][b1][b2][l][r];
68          ii = i, pp = rec(b1, b2, l, r);
69      }
70
71      cout << "Oil : " << res << endl;
72      gao(ii, k, pp);
73  }
```

Chapter 30

临时

30.1 例题 (E.g.)

30.1.1 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。

```
1  namespace SAM{
2
3      const int SN = int(5e4) + 1, NN = 11, N = 2*NN*SN + 9, Z = 26;
4
5      int trans[N][Z+NN], fail[N], len[N], tail, tot; char str[SN];
6
7      inline int new_node(){
8          // RST(trans[tot);
9          tail = tot;
10         return tot++;
11     }
12     inline int new_node(int u){
13         CPY(trans[tot], trans[u]), fail[tot] = fail[u];
14         return tot++;
15     }
16
17 #define v trans[u]1
18 #define f fail[u]
19 #define ff fail[uu]
20
21     inline int h(int u){
22         return len[u] - len[f];
23     }
24
25     void Ext(int c){
26         int u = tail, uu = new_node(); len[uu] = len[u] + 1;
27         while (u && !v) v = uu, u = f;
28         if (!u && !v) v = uu, ff = 0;
29         else{
30             if (len[v] == len[u] + 1) ff = v;
31             else{
32                 int _v = v, vv = new_node(_v); len[vv] = len[u] + 1;
33                 fail[_v] = ff = vv;
34                 while (v == _v) v = vv, u = f;
35             }
```

```
        }
    }

    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);
}
```

30.1.2 BZOJ 2806.

30.1.3 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];
```

```
7
8       inline int new_node(){
9           RST(trans[tot]), tail = tot;
10          return tot++;
11      }
12
13      inline int new_node(int u){
14          CPY(trans[tot], trans[u]), fail[tot] = fail[u];
15          return tot++;
16      }
17
18  #define v trans[u][c]
19  #define f fail[u]
20  #define ff fail[uu]
21
22      void Ext(int c){
23          int u = tail, uu = new_node(); len[uu] = len[u] + 1;
24          while (u && !v) v = uu, u = f;
25          if (!u && !v) v = uu, ff = 0;
26          else{
27              if (len[v] == len[u] + 1) ff = v;
28              else{
29                  int _v = v, vv = new_node(_v); len[vv] = len[u] + 1;
30                  fail[_v] = ff = vv;
31                  while (v == _v) v = vv, u = f;
32              }
33          }
34      }
35
36      void Init(){
37          tot = 0, tail = new_node();
38      }
39
40      int ll[N/2], n, n0;
41
42      void Spell(){
43          int u = 0, l = 0; REP_1(i, n){
44              int c = str[i] - '0';
45              while (u && !v) l = len[u = f];
46              if (u = v) ++l; ll[i] = i - l;
47          }
48      }
49
50      #undef f
51
52      int f[N/2], g[N/2], q[N/2], cz, op, tt;
53
54      bool check(int x){
55
56          cz = 1, op = 0; REP_1(i, n){
57
58              if (~(tt=i-x)){
59                  while (cz <= op && op[q][g] < g[tt]) --op;
60                  q[++op] = tt;
61              }
62
63              while (cz <= op && q[cz] < ll[i]) ++cz;
64
65              f[i] = max(f[i-1], cz <= op ? cz[q][g] + i : 0);
66              g[i] = f[i] - i;
67          }
68
69          return f[n] >= n0;
70      }
71
72  } using namespace SAM;
73
```

```
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);
    }
}
```

30.1.4 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 。。两边同时旋转。Methodhttp://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(){
```

```
167         cout << s << " " << t << " " << r <<endl;
168     }
169
170     int entry(int x){
171         if (lca(r, x) != r) return r;
172         int tt = lca(s, x); if (tt != r) return tt;
173         return lca(t, x);
174     }
175
176     bool sgn(int x){ // 判断点是否在路径上。
177         return lca(x, r) == r && (lca(s, x) == x || lca(t, x) == x);
178     }
179
180 } ants[MM]; int m;
181
182 int d(int u, int v){
183     return D[u] + D[v] - D[lca(u, v)]*2;
184 }
185
186 int check(Ants& a, Ants& b){
187
188     int bg = a.entry(b.s); if (!b.sgn(bg)) return 0;
189     int ed = a.entry(b.t); if (!b.sgn(ed)) return 0;
190
191     bool b1 = 0, b2 = 0;
192
193     /*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;
194     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;
195     if (!sgn(all, bll)) return 1;
196     if (all > bll) swap(all, bll), swap(arr, brr);
197     return sgn(b1==b2?brr:bll, arr) <= 0;
198     // 使用浮点数的话。。10-8 全挂。。10-9 260 分。。10-10 280 分。。之后300 分。。。
199     */
200
201     LL al = d(a.s, bg), ar = d(a.s, ed); if (al > ar) swap(al, ar), b1 = 1;
202     LL bl = d(b.s, bg), br = d(b.s, ed); if (bl > br) swap(bl, br), b2 = 1;
203     LL av = a.v, bv = b.v;
204
205     if (al*bv == bl*av) return 1;
206     if (al*bv > bl*av) swap(al, bl), swap(ar, br), swap(av, bv);
207     return (b1==b2?br:bl)*av <= ar*bv;
208 }
209
210
211 int main(){
212
213 #ifndef ONLINE_JUDGE
214     freopen("in.txt", "r", stdin);
215     //freopen("out.txt", "w", stdout);
216 #endif
217
218     Rush{
219
220         FOR_C(i, 2, RD(n)<<1){
221             RD(aa, bb, w);
222             suc[i] = hd[aa], hd[aa] = i, ++i;
223             suc[i] = hd[aa], hd[aa] = i;
224         }
225
226         dfs();
227
228         for ( int lv = 1 ; (1 << lv) <= tt ; lv ++ ){
229             for ( int i = 1 ; i + (1 << lv) <= tt + 1 ; i ++ )
230                 ST[lv][i] = min(ST[lv-1][i], ST[lv-1][i + (1<<(lv-1))], elder);
231
232         }
233
```

```
234         REP_C(i, RD(m)) ants[i].in();
235
236         int res = 0; REP_2(j, i, m, j){
237             res += check(ants[i], ants[j]);
238         }
239
240         OT(res);
241
242         fill(hd+1, hd+n+1, 0),
243         fill(st+1, st+n+1, 0),
244         tt = 0;
245     }
246 }
247
248
249
250 const int N = 5009, M = 2 * 30009;
251
252 int D[N], hd[N], suc[M], to[M], cap[M];
253 int n, m, s, t;
254
255 inline void add_edge(int x, int y, int c){
256     suc[m] = hd[x], to[m] = y, cap[m] = c, hd[x] = m++;
257     suc[m] = hd[y], to[m] = x, cap[m] = 0, hd[y] = m++;
258 }
259
260 inline void add_edgee(int x, int y, int c){
261     suc[m] = hd[x], to[m] = y, cap[m] = c, hd[x] = m++;
262     suc[m] = hd[y], to[m] = x, cap[m] = c, hd[y] = m++;
263 }
264
265 #define v to[i]
266 #define c cap[i]
267 #define f cap[i^1]
268
269 bool bfs(){
270     static int Q[N]; int cz = 0, op = 1;
271     fill(D, D+n, 0), D[Q[0] = s] = 1; while (cz < op){
272         int u = Q[cz++]; REP_G(i, u) if (!D[v] && c){
273             D[Q[op++] = v] = D[u] + 1;
274             if (v == t) return 1;
275         }
276     }
277     return 0;
278 }
279
280 LL Dinitz(){
281
282     to[0] = s;
283     LL max_flow = 0;
284
285     while (bfs()){
286
287         static int sta[N], cur[N]; int top = 0;
288         sta[0] = 0, cur[s] = hd[s]; while (top != -1){
289
290             int u = to[sta[top]], i; if (u == t){
291                 int d = INF; REP_1(ii, top) i = sta[ii], checkMin(d, c); max_flow += d;
292                 DWN_1(ii, top, 1){i = sta[ii], f += d, c -= d; if (!c) top = ii - 1;}
293                 u = to[sta[top]];
294             }
295
296             for (i=cur[u];i;i=suc[i])
297                 if (D[u] + 1 == D[v] && c) break;
298
299             if (!i) D[u] = 0, --top;
300             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());
    }
}


//判断二维平面条线段是否存在交点nlognn

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{
```

```
368     int id; rec(int id):id(id){}
369     bool operator<(const rec& r)const{
370         return y(L[id]) < y(L[r.id]);
371     }
372 };
373
374 #define s_it set<rec>::iterator
375 set<rec> S; s_it _S[N];
376 inline s_it preIt(s_it it){return it == S.begin() ? S.end() : --it;}
377 inline s_it sucIt(s_it it){return it == S.end() ? S.end() : ++it;}
378
379 void isIntersect(){
380
381     vector<Event> I; REP(i, n){
382         I.PB(Event(L[i].a.x, i, 1));
383         I.PB(Event(L[i].b.x, i, -1));
384     }
385
386     SRT(I); ECH(i, I){
387
388         x = i->x;
389
390 #define tryIntersect(a, b) if (~L[a].sgn(L[b])){ \
391     printf("YES\n%d %d\n", a+1, b+1); return; \
392 }
393         if (~i->ty){
394             rec cur(i->id);
395             s_it suc = S.lower_bound(cur), prd = preIt(suc);
396             if (suc != S.end()) tryIntersect(cur.id, suc->id);
397             if (prd != S.end()) tryIntersect(cur.id, prd->id);
398             _S[i->id] = S.insert(suc, cur);
399         }
400         else {
401             s_it &cur = _S[i->id], prd = preIt(cur), suc = sucIt(cur);
402             if(prd != S.end() && suc != S.end()) tryIntersect(prd->id, suc->id);
403             S.erase(cur);
404         }
405     }
406     puts("NO");
407 }
408
409
410 int main(){
411
412 #ifndef ONLINE_JUDGE
413     freopen("in.txt", "r", stdin);
414     //freopen("out.txt", "w", stdout);
415 #endif
416
417     REP_C(i, RD(n)) L[i].in();
418     isIntersect();
419 }
420
421
422
423
424 // Beiju ..
425
426 const int N = 109;
427
428 Po L[N], R[N];
429 int n, m;
430
431 DB x, s;
432 void g(cPo a, cPo b){
433     DB t = det(a, b);
434     x += t * (a.x + b.x);
```

```
435     s += t;
436 }
437 Po gl(DB y){
438     int i; REP_N(i, n){
439         if (sgn(L[i+1].y, y) > 0) break;
440         g(L[i], L[i+1]);
441     }
442     Po p = L[i] + (y-L[i].y)/(L[i+1].y-L[i].y)*(L[i+1]-L[i]);
443     g(L[i], p);
444     return p;
445 }
446 Po gr(DB y){
447     int i; REP_N(i, m){
448         if (sgn(R[i+1].y, y) > 0) break;
449         g(R[i+1], R[i]);
450     }
451     Po p = R[i] + (y-R[i].y)/(R[i+1].y-R[i].y)*(R[i+1]-R[i]);
452     g(p, R[i]);
453     return p;
454 }
455 bool f(DB y){
456     x = 0, s = 0, g(gl(y), gr(y)), x /= s*3;
457     return sgn(L[0].x, x) <= 0 && sgn(x, R[0].x) <= 0;
458 }
459
460 int main(){
461
462 #ifndef ONLINE_JUDGE
463     freopen("in.txt", "r", stdin);
464     //freopen("out.txt", "w", stdout);
465 #endif
466
467     Rush{
468         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);
469         //assert(!sgn(L[n-1].y, R[m-1].y));
470         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;
471         DB l = Y[i-1], r = Y[i]; DO(233){
472             DB m = (l + r) / 2; if (f(m)) l = m; else r = m;
473         }
474         OT(l);
475     }
476 }
477
478 //滚球兽
479 const int N = 109;
480
481 VP P, C; int n, nn;
482 Po T, O; DB alpha, res;
483
484 inline DB ang(VP &C, int i){
485     return i ? ang(C[i-1],C[i],C[i],C[i+1]) : ang(C[nn-1],C[0],C[0],C[1]);
486 }
487
488 bool Roll(){
489     DB beta = OO; REP(i, n){
490         Seg l = Line(P[i], P[i+1]);
491         if (~sgn(dist2(O,P[i]),dist2(O,P[i+1]))) swap(l.a, l.b);
492         Po o = O&l; if (~sgn(dist2(O,o), dist2(O,T))) continue; //!
493         Po t = o + l.d()._1() * sqrt(dist2(O,T) - dist2(O,o));
494         if (~l.sgn(t)) checkMin(beta, arg(O,T,t));
495     }
496     return sgn(beta, alpha) <= 0 ? res += beta, 1 : 0;
497 }
498
499 int main() {
500
501 #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
```

```
568 #define root 1, 1, nn
569
570 void Build(int x, int l, int r){
571     if (l == r) T[x] = max(0, S[x] = A[l]);
572     else {
573         Build(lc), Build(rc);
574         S[x] = S[lx] + S[rx];
575         T[x] = max(T[lx], S[lx] + T[rx]);
576     }
577 }
578
579 void Get(int x, int l, int r){ // get interval
580     if (a <= l && r <= b) A[tt++] = x;
581     else{
582         if (a <= mid) Get(lc);
583         if (mid < b) Get(rc);
584     }
585 }
586
587 int Q1(int a, int b){
588     ::a = a, ::b = b, tt = 0, Get(root);
589     int res = 0; REP(i, tt) res += S[A[i]];
590     return res;
591 }
592
593 int Q2(int a, int b){
594     ::a = a, ::b = b, tt = 0, Get(root);
595     int res = 0, sss = 0; REP(i, tt){
596         int ai = A[i];
597         checkMax(res, sss + T[ai]);
598         sss += S[ai];
599     }
600     return res;
601 }
602
603 void Modify(int x, int l, int r){
604     if (l == r) T[x] = max(0, S[x] = b);
605     else {
606         if (a <= mid) Modify(lc);
607         else Modify(rc);
608         S[x] = S[lx] + S[rx];
609         T[x] = max(T[lx], S[lx] + T[rx]);
610     }
611 }
612
613 int main(){
614
615 #ifndef ONLINE_JUDGE
616     freopen("in.txt", "r", stdin);
617     //freopen("out.txt", "w", stdout);
618 #endif
619
620     Rush{
621
622         RST(hd, L, A, S, T), tt = nn = 0; FOR_C(i, 2, RD(n) << 1){
623             RD(aa, bb); RDD(wt[i]), wt[i|1] = wt[i];
624             suc[i] = hd[aa], hd[aa] = i, ++i;
625             suc[i] = hd[aa], hd[aa] = i;
626         }
627
628         dfs();
629
630         Build(root);
631
632         for ( int lv = 1 ; (1 << lv) <= tt ; lv ++ ){
633             for ( int i = 1 ; i + (1 << lv) <= tt + 1 ; i ++ )
634                 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);
            }
        }
    }
}
```

30.1.5 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;
```

```
40  }
41
42  int main(){
43    for(scanf("%d",&tcase); tcase; tcase-=1){
44      scanf("%d",&t);
45      int64 res = 0;
46      for(int i=0; i<t; ++i){
47        scanf("%d",&n);
48        for(int j=0; j<n; ++j) scanf("%d",&arr[j]);
49        sort(arr,arr+n);
50        res += calculate();
51      }
52      if(res > 0 ) printf("FIRST\n");
53      else if(res < 0 ) printf("SECOND\n");
54      else printf("DON'T PLAY\n");
55    }
56    return 0;
57  }
```

30.1.6 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 in

```
/*
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
```

```
62  #define S second
63  #define ALL(c) (c).begin(), (c).end()
64  #define SIZE(c) (c).size()
65
66  #define DEBUG(x) { cerr << #x << " = " << x << endl; }
67  #define PR(a,n) {cerr<<#a<<" = "; FOR(_,1,n) cerr << a[_] << ' '; cerr <<endl;}
68  #define PR0(a,n) {cerr<<#a<<" = ";REP(_,n) cerr << a[_] << ' '; cerr << endl;}
69
70  #define oo 2000111000
71  #define mod 1000000007
72  using namespace std;
73
74  struct Node {
75      Node *left, *right, *parent;
76      //Each node of the splay tree stores one segment [u..v] (u can be larger than v)
77      //num is the total number of array's elements in the sub-tree rooted at this node
78      //flip = 1 when we need to flip this sub-tree but haven't done it yet (lazy update)
79      int u, v, num, flip;
80
81  };
82
83  Node *nullT;
84  int nNum, numU[1222222], numV[1222222], d[1222222], nD, n;
85
86  void initTree() {
87      nullT = new Node;
88      nullT -> left = nullT -> right = nullT -> parent = nullT;
89      nullT -> u = nullT -> v = nullT -> num = nullT -> flip = 0;
90  }
91
92  //Splay tree's stuffs
93  void setLink (Node *parent, Node *child, bool isLeft) {
94      if (isLeft) parent -> left = child;
95      else parent -> right = child;
96
97      if (child != nullT) child -> parent = parent;
98  }
99
100 //If we need to flip the sub-tree rooted at root, we'll do it now
101 void lazyUpdate(Node *root) {
102      if (root == nullT) return;
103
104      if (root -> flip) {
105          root -> flip = 0;
106          swap(root -> u, root -> v);
107          swap(root -> left, root -> right);
108          root -> left -> flip ^= 1;
109          root -> right -> flip ^= 1;
110      }
111 }
112
113 void update(Node * x) {
114      if (x == nullT) return;
115      x -> num = x -> right -> num + x -> left -> num + abs(x->u - x->v) + 1;
116 }
117
118 //find the node of the tree that contains the kth element
119 //We will split some node so that the there will be a node contains the segment [u..v] and v is the kth element
120 Node *findNode(Node * root, int kth) {
121      lazyUpdate(root);
122
123      if (root -> left -> num >= kth) return findNode(root -> left, kth);
124
125      int len = abs(root -> u - root -> v) + 1;
126      if ((root -> left -> num + len) >= kth) {
127          if (root -> left -> num + len == kth) return root;
128
```

```
129         int mid, pre;
130         int dt = kth - root -> left -> num;
131 
132         if (root -> u < root -> v) {
133             mid = root -> u + dt;
134             pre = mid - 1;
135         }
136         else {
137             mid = root -> u - dt;
138             pre = mid + 1;
139         }
140 
141         Node * tmp = new Node;
142         tmp -> u = mid;
143         tmp -> v = root -> v;
144         tmp -> flip = 0;
145 
146         root -> v = pre;
147 
148         tmp -> left = nullT;
149         setLink(tmp, root->right, 0);
150         setLink(root, tmp, 0);
151 
152         update(tmp);
153         update(root);
154 
155         return root;
156     }
157 
158     return findNode(root -> right, kth - len - root -> left -> num);
159 }
160 
161 void upTree(Node *x) {
162     Node *y = x -> parent;
163     Node *z = y -> parent;
164     Node *tmp;
165     if (y->right == x) {
166         tmp = x -> left;
167         setLink(x, y, true);
168         setLink(y, tmp, false);
169     }
170     else {
171         tmp = x -> right;
172         setLink(x, y, false);
173         setLink(y, tmp, true);
174     }
175     setLink(z, x, z->left == y);
176     update(y); update(x);
177 }
178 
179 void splay(Node *x) {
180     while (1) {
181         Node *y = x -> parent;
182         if (y == nullT) return ;
183 
184         Node *z = y -> parent;
185 
186         if (z != nullT)
187             if ((z->right == y) == (y->right == x)) upTree(y);
188             else upTree(x);
189         upTree(x);
190     }
191 }
192 
193 void printTree(Node *root) {
194     lazyUpdate(root);
195     if (root == nullT) return;
```

```
196     printTree(root -> left);
197
198     printf("[%d %d]", root -> u, root -> v);
199
200     printTree(root -> right);
201 }
202
203 //Find the final array
204 void extract(Node *root) {
205     lazyUpdate(root);
206     if (root == nullT) return;
207     extract(root -> left);
208
209     numU[++nNum] = root->u;
210     numV[nNum] = root->v;
211
212     extract(root -> right);
213 }
214
215 //Split a tree into two sub-strees. a and b where a contains the first num elements
216 void split(Node *root, int num, Node * &a, Node * &b) {
217     Node *tmp = findNode(root, num);
218     splay(tmp);
219
220     b = tmp -> right;
221     b -> parent = nullT;
222
223     tmp -> num -= tmp -> right -> num;
224     tmp -> right = nullT;
225     a = tmp;
226 }
227
228 void debugTree(Node * root) {
229     printTree(root);
230     printf("\n");
231 }
232
233 //Join two trees, a and b
234 Node * join(Node * a, Node * b) {
235     Node * tmp = findNode(a, a->num);
236     splay(tmp);
237
238
239     setLink(tmp, b, 0);
240
241     update(tmp);
242     return tmp;
243 }
244
245 long long pow2(int n) {
246     if (n == 0) return 1;
247
248     long long res = pow2(n / 2);
249     res = res * res % mod;
250
251     if (n % 2) res = res * 2 % mod;
252
253     return res;
254 }
255
256 vector <pair <int, int> > seg;
257 map <int, int> pos;
258
259 long long call() {
260     //numV[nNum] always equal to n + 1
261     //if numU[nNum] = n + 1 too, we ignore this segment, otherwise, we decrease numV[nNum] by 1
262     if (numU[nNum] == n + 1) nNum--;
```

```
263     else numV[nNum]--;
264
265     //ignore the number 0 too
266     int START = 1;
267     if (numV[1] == 0) {
268         START ++;
269     }
270     else numU[1] ++;
271
272     FOR (i, START, nNum) {
273         int d = 1;
274         if (numU[i] > numV[i]) d = -1;
275         int len = abs(numU[i] - numV[i]) + 1;
276         if (len <= 4) {
277             int u = numU[i];
278             FOR (i, 1, len) {
279                 seg.PB(MP(u, u));
280                 u += d;
281             }
282         }
283         else {
284             int u = numU[i];
285             seg.PB(MP(u, u));
286             seg.PB(MP(u + d, u + d));
287
288             int v = numV[i];
289             seg.PB(MP(u + 2 * d, v - 2 * d));
290
291             seg.PB(MP(v - d, v - d));
292             seg.PB(MP(v, v));
293         }
294     }
295
296
297
298     int pr = 0;
299     int z = seg.size();
300
301     FOR (i, 0, z - 1) {
302         int len = abs(seg[i].first - seg[i].second) + 1;
303         pos[seg[i].first] = pr + 1;
304         pos[seg[i].second] = pr + len;
305         pr += len;
306     }
307     pos[n + 1] = -1;
308
309     int cnt = 0;
310
311     //In the segment [u...v] (suppose u <= v)
312     //We have that S[u] <= S[u + 1] <= S[u + 2] <= ... <= s[v - 1]
313     //s[v + 1] may <= or < s[v] and this depends on the position of v + 1 and u - 1.
314     //Similarly in the case u > v
315
316     FOR (i, 0, z - 2) {
317         int u = pos[seg[i].second + 1];
318         if (u == 0) u = pos[seg[i].second] - 1;
319
320         int v = pos[seg[i + 1].first + 1];
321         if (v == 0) v = pos[seg[i + 1].first] + 1;
322
323         if (u < v) cnt++;
324     }
325
326
327     FOR (i, 0, z - 1)
328         cnt += abs(seg[i].first - seg[i].second);
329
```

```
330     return pow2(cnt);
331 }
332
333 int main() {
334     initTree();
335     Node * root = new Node;
336     int m;
337     cin >> n >> m;
338
339     //Initially the tree contain only one node: (0, n + 1)
340     //adding 0 and n + 1 to avoid the cornner cases
341     root -> u = 0;
342     root -> v = n + 1;
343     root -> num = n + 2;
344     root -> flip = 0;
345
346     setLink(root, nullT, 0);
347     setLink(root, nullT, 1);
348
349     root -> parent = nullT;
350
351     while (m--) {
352         int u, v, k;
353
354         scanf("%d%d%d", &k, &u, &v);
355
356         Node *a, *b, *c, *d, *e;
357
358         if (k == 1) {
359             //split the tree into 3 smaller trees a, b and c
360             //b will contains only the elements from uth elements to vth elements
361             split(root, u, a, b);
362             split(b, v - u + 1, b, c);
363
364             //flip b and join 3 trees
365             b->flip |= 1;
366             a = join(a, b);
367             root = join(a, c);
368         }
369         else {
370             if(u == 1) continue;
371             //similarly split the tree into 3 sub-tree a, b, c
372             split(root, 1, a, b);
373             split(b, u - 1, b, c);
374             split(c, v - u + 1, c, d);
375             //we just need to change the order of a, b and c when joining them
376             root = join(a, c);
377             root = join(root, b);
378             root = join(root, d);
379         }
380     }
381
382     nNum = 0;
383     extract(root);
384
385     cout << call() << endl;
386     return 0;
387 }
```

30.1.7 POJ 1741. Tree

简述 (Brief description)

... 求树中距离 <=k 的点对总数..

..

```
1  const int N = int(1e4) + 9, M = N * 2;
2
3  int hd[N], prd[M], suc[M], to[M], ww[N]; // adj ...
4  int dep[N], sz[N]; int L[N], Ln, ans;
5  int n, k, nn, c, cc;
6
7  #define a to[i^1]
8  #define b to[i]
9  #define w ww[i/2]
10 #define v b
11
12 inline void del(int i){
13     if (i == hd[a]) prd[hd[a] = suc[i]] = 0;
14     else prd[suc[i]] = prd[i], suc[prd[i]] = suc[i];
15 }
16
17 void dfs_c(int u, int p = 0){
18     int ss = 0; sz[u] = 1;
19     REP_G(i, u) if (v != p){
20         dfs_c(v, u), sz[u] += sz[v];
21         checkMax(ss, sz[v]);
22     }
23     checkMax(ss, nn - sz[u]);
24     if (ss <= cc) cc = ss, c = u;
25 }
26
27 void dfs0(int u, int p = 0){
28     L[Ln++] = dep[u], sz[u] = 1;
29     REP_G(i, u) if (v != p){
30         dep[v] = dep[u] + w;
31         dfs0(v, u), sz[u] += sz[v];
32     }
33 }
34
35 void dfs1(int u, int p = 0){
36     L[Ln++] = dep[u];
37     REP_G(i, u) if (v != p){
38         dfs1(v, u);
39     }
40 }
41
42 int f(){
43     int res = 0, l = 0, r = Ln - 1;
44     sort(L, L+Ln);
45     while (l < r){
46         if (L[l] + L[r] > k) --r;
47         else res += r - l++;
48     }
49     Ln = 0;
50     return res;
51 }
52
53 void gao(int u = 1){
54     cc = INF, dfs_c(u), u = c;
55     dep[u] = 0, dfs0(u), ans += f();
56
57     REP_G(i, u){
58         del(i^1), dfs1(v), ans -= f();
59         nn = sz[v], gao(v);
60     }
61 }
62
63 int main(){
64
```

```
#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);
    }

}
```

30.1.8 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);
    }

```

```
38      void Init(){
39          tot = 0, new_node();
40          Insert(); Build();
41      }
42
43  #undef c
44  } using namespace ACM;
45
46  void Gauss(){
47      REP(i, n){
48          if (!sgn(A[i][i])){
49                  int j; FOR_N(j, i+1, n) if (sgn(A[j][i])) break;
50                  if (j == n){
51                      // Warning;
52                      assert(0);
53                  }
54                  FOR_1(k, 0, n) swap(A[i][k], A[j][k]);
55
56          }
57          DB t = A[i][i]; FOR_1(j, i, n) A[i][j] /= t;
58          REP(j, n) if (i!=j&&sgn(A[j][i])){
59              DB t = A[j][i]; FOR_1(k, i, n) A[j][k] -= A[i][k] * t;
60          }
61      }
62  }
63
64  int main(){
65
66  #ifndef ONLINE_JUDGE
67      freopen("in.txt", "r", stdin);
68      //freopen("out.txt", "w", stdout);
69  #endif
70
71      Rush{
72
73          if (Case) puts("");
74
75          RD(m), Init();
76
77          RST(A); REP(u, n){
78              A[u][u] = A[u][n+1] = m;
79              REP(c, m) A[u][v] -= 1;
80          }
81
82          A[n][n] = 1, ++n;
83          //Display(A, n, n+1);
84          Gauss();
85          //Display(A, n, n+1);
86
87          OT(A[0][n]);
88      }
89
90  }
```

30.1.9 BZOJ 2154. Crash 的数字表格

简述 (Brief description)

...

分析 (Analysis)

..

```
1  const int PMAX = int(1e7) + 9;
2  VI P; int pp[PMAX]; Int G[PMAX];
```

```
3
4  void sieve(){
5      G[1] = 1; FOR(i, 2, PMAX){
6          if (!pp[i]) P.PB(i), pp[i] = i, G[i] = (LL)i*(1-i);
7  #define ii (i*P[j])
8          for (int j=0;j<SZ(P)&&ii<PMAX;++j) if (i%P[j]){
9              pp[ii] = P[j], G[ii] = G[i]*G[P[j]];
10         } else{
11             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]];
12             break;
13         }
14     }
15 #undef ii
16     FOR(i, 1, PMAX) G[i] = G[i-1] + G[i];
17 }
18
19 Int f(int a, int b){
20     if (a > b) swap(a, b); Int z=0; for(int i=1,ii;i<=a;i=ii+1){
21         int aa = a/i, bb = b/i; ii = min(a/aa, b/bb);
22         z += (G[ii]-G[i-1])*aa*bb*(aa+1)*(bb+1);
23     }
24     return z/4;
25 }
26
27 int main(){
28
29 #ifndef ONLINE_JUDGE
30     freopen("in.txt", "r", stdin);
31     //freopen("out.txt", "w", stdout);
32 #endif
33
34     sieve(); OT(f(RD(), RD()));
35 }
```