ACM/ICPC Notebook

November 3, 2018

Contents

0 Others

0.1 Fastreading

0.1.1 Fastreading

```
inline int Read() {
    char c=getchar(); int x=0,f=1;
    while(c<'0'||c>'9') {if(c=='-')f=-1;c=getchar();}
    while(c>='0'&&c<='9') {x=x*10+c-'0';c=getchar();}
    return x*f;
}

namespace IO {
    const int MX = 4e7; //1e7 □□□□□□□ 11000kb
    char buf[MX]; int c, sz;
    void Begin() {
        c = 0;
        sz = fread(buf, 1, MX, stdin);//□´□□□□²¿¶□□□
    }
    inline bool Read(int &t) {
        while (c < sz && buf[c] != '-' && (buf[c] < '0' || buf[c] > '9')) c++;
        if (c >= sz) return false;//□□□□□□□□¸□□□□□□□
        bool flag = 0; if(buf[c] == '-') flag = 1, c++;
        for(t = 0; c < sz && '0' <= buf[c] && buf[c] <= '9'; c++) t = t * 10 + buf[c] -
     '0';
        if(flag) t = -t;
        return true;
    }
}
IO::Begin();
IO::Read(x);
```

0.2 Lisanhua

0.2.1 lisanhua

```
vector<int> id;

int Getid(int x)    /// 1 ¿ª□±□□
{
    return lower_bound(id.begin(), id.end(), x) - id.begin() + 1;
}

void G()
{
    sort(id.begin(), id.end());
    id.erase(unique(id.begin(), id.end()), id.end());
}
```

0.3 Bignum

0.3.1 Bignum

```
struct bignum{
    int len;
    int a[LEN];

```

```
5       bignum(){
6           memset(a,0,sizeof a);
7           len=0;
8       }
9       int& operator[](int idx){
10          return a[idx];
11      }
12      const int& operator[](int idx) const{
13          return a[idx];
14      }
15
16      bignum& operator =(const bignum& b){
17          len=b.len;
18          for (int i=0; i<len; i++)
19              a[i]=b[i];
20          return *this;
21      }
22      bignum& operator =(char b[]){
23          len=strlen(b);
24          for (int i=0; i<len; i++)
25              a[i]=b[len-1-i]-'0';
26          while (len>0&&!a[len-1]) len--;
27          return *this;
28      }
29      bignum& operator =(const char b[]){
30          len=strlen(b);
31          for (int i=0; i<len; i++){
32              a[i]=b[len-i-1]-'0';
33          }
34          while (len>0&&!a[len-1]) len--;
35          return *this;
36      }
37      bignum& operator =(long long b){
38          len=0;
39          while (b){
40              a[len++]=b%10;
41              b/=10;
42          }
43          return *this;
44      }
45
46      bignum& operator +=(const bignum& b){
47          len=max(len,b.len);
48          for (int i=0; i<len; i++) a[i]+=b[i];
49          for (int i=0; i<len; i++){
50              a[i+1]+=a[i]/10;
51              a[i]%=10;
52          }
53          if (a[len]) len++;
54          return *this;
55      }
56      bignum& operator +=(int b){
57          a[0]+=b;
58          for (int i=0; i<LEN-1; i++){
59              a[i+1]+=a[i] / 10;
60              a[i]%=10;
61          }
62          a[LEN]%=10;
63          len=LEN;
```

```
64          while (len>0&&!a[len-1]) len--;
65          return *this;
66      }
67      bignum& operator +=(char b[]){
68          bignum bb;
69          bb=b;
70          return (*this)+=bb;
71      }
72      bignum& operator +=(const char b[]){
73          bignum bb;
74          bb=b;
75          return (*this)+=bb;
76      }
77
78      bignum& operator -=(const bignum& b){
79          len=max(len,b.len);
80          for (int i=0; i<len; i++) a[i]-=b[i];
81          for (int i=0; i<len; i++){
82              a[i+1]+=a[i]/10;
83              a[i]%=10;
84              if (a[i]<0){
85                  a[i]+=10;
86                  a[i+1]--;
87              }
88          }
89          while (len>0&&!a[len-1]) len--;
90          return *this;
91      }
92      bignum& operator -=(int b){
93          bignum bb;
94          bb=b;
95          return (*this-=bb);
96      }
97      bignum& operator -=(char b[]){
98          bignum bb;
99          bb=b;
100         return (*this)-=bb;
101     }
102     bignum& operator -=(const char b[]){
103         bignum bb;
104         bb=b;
105         return (*this)-=bb;
106     }
107
108     bignum& operator *=(const bignum& b){
109         bignum ans;
110         for (int i=0; i<len; i++)
111             for (int j=0; j<b.len; j++)
112                 ans[i+j]+=a[i]*b[j];
113         for (int i=0; i<len+b.len+1; i++){
114             ans[i+1]+=ans[i]/10;
115             ans[i]%=10;
116         }
117
118         for (int i=0; i<LEN; i++)
119             a[i]=ans[i];
120         len=LEN;
121         while (len>0&&!a[len-1]) len--;
122         return *this;
```

```
123     }
124     bignum& operator *=(const char* b){
125         bignum bb;
126         bb=b;
127         return (*this*=bb);
128     }
129     bignum& operator *=(char* b){
130         bignum bb;
131         bb=b;
132         return (*this*=bb);
133     }
134     bignum& operator *=(int b){
135         bignum bb;
136         bb=b;
137         return (*this*=bb);
138     }
139
140     void show(){
141         for (int i=len-1; i>=0; i--)
142             printf("%d",a[i]);
143         if (len==0) printf("0");
144     }
145     void get(){
146         char s[LEN];
147         scanf("%s",s);
148         *this=s;
149         while (len>0&&!a[len-1]) len--;
150     }
151 };
152 bignum operator + (bignum &a,bignum &b){
153     bignum ans;
154     ans=a;
155     ans+=b;
156     return ans;
157 }
158 bignum operator - (bignum &a,bignum &b){
159     bignum ans;
160     ans=a;
161     ans-=b;
162     return ans;
163 }
164 bignum operator * (bignum &a,bignum &b){
165     bignum ans;
166     ans=a;
167     ans*=b;
168     return ans;
169 }
170 bool operator < (bignum &a, bignum &b){
171     if (a.len<b.len) return 1;
172     if (a.len>b.len) return 0;
173
174     for (int i=a.len-1; i>=0; i--){
175         if (a[i]!=b[i]) return a[i]<b[i];
176     }
177 }
178
179 bool operator > (bignum &a, bignum &b){
180     if (a.len<b.len) return 0;
181     if (a.len>b.len) return 1;
```

```
182
183     for (int i=a.len-1; i>=0; i--){
184         if (a[i]!=b[i]) return a[i]>b[i];
185     }
186 }
187 bool operator > (bignum &a, long long b){
188     bignum bb;
189     bb=b;
190     return a>bb;
191 }
192
193 bool operator ==(bignum a,bignum b){
194     if (a.len!=b.len) return 0;
195     for (int i=0; i<a.len; i++){
196         if (a[i]!=b[i]) return 0;
197     }
198     return 1;
199 }
200 bool operator ==(bignum a,long long b){
201     bignum bb;
202     bb=b;
203     return a==bb;
204 }
205 bool operator ==(bignum a,char *b){
206     bignum bb;
207     bb=b;
208     return a==bb;
209 }
210 bool operator ==(bignum a,const char *b){
211     bignum bb;
212     bb=b;
213     return a==bb;
214 }
215 bool operator <= (bignum &a, bignum &b){
216     return a==b||a<b;
217 }
218 bool operator >= (bignum &a, bignum &b){
219     return a==b||a>b;
220 }
221 bool operator != (bignum &a, bignum &b){
222     return !(a==b);
223 }
224
225 bignum& operator /=(bignum &a,bignum &b){
226     bignum ans;
227     bignum bb;
228     ans.len=a.len;
229
230     for (int i=ans.len-1; i>=0; i--){
231         int l=0,r=9,mid;
232         while (l<r){
233             mid=(l+r)/2+1;
234             ans[i]=mid;
235             bb=b;
236             bb*=ans;
237
238             if (bb<=a) l=mid;
239             else r=mid-1;
240         }
```

```
241         ans[i]=l;
242     }
243
244     while (ans.len>0&&!ans[ans.len-1]) ans.len--;
245     a=ans;
246     return a;
247 }
248 bignum& operator /= (bignum& a,long long b){
249     bignum bb;
250     bb=b;
251     a/=bb;
252     return a;
253 }
254 bignum& operator /= (bignum& a,char *b){
255     bignum bb;
256     bb=b;
257     a/=bb;
258     return a;
259 }
260 bignum& operator /= (bignum& a,const char *b){
261     bignum bb;
262     bb=b;
263     a/=bb;
264     return a;
265 }
266 bignum operator / (bignum& a,bignum& b){
267     bignum ans;
268     ans=a;
269     ans/=b;
270     return ans;
271 }
272 bignum operator / (bignum& a,int b){
273     bignum ans,bb;
274     ans=a;
275     bb=b;
276     ans/=b;
277     return ans;
278 }
279 bignum operator / (bignum& a,char *b){
280     bignum ans,bb;
281     ans=a;
282     bb=b;
283     ans/=b;
284     return ans;
285 }
286 bignum operator / (bignum& a,const char *b){
287     bignum ans,bb;
288     ans=a;
289     bb=b;
290     ans/=b;
291     return ans;
292 }
293 bignum operator %= (bignum& a,bignum& b){
294     bignum d;
295     d=a/b;
296     a-=d*b;
297     return a;
298 }
299 long long operator %= (bignum& a,long long b){
```

```
300     long long ans=0,w=1;
301     for (int i=0; i<a.len; i++){
302         ans=(ans+a[i]*w%b)%b;
303         w=w*10%b;
304     }
305     a=ans;
306     return ans;
307 }
308 bignum operator %= (bignum& a,char *b){
309     bignum bb;
310     bb=b;
311     a%=bb;
312     return a;
313 }
314 bignum operator %= (bignum& a,const char *b){
315     bignum bb;
316     bb=b;
317     a%=bb;
318     return a;
319 }
320 bignum operator % (bignum& a,bignum& b){
321     bignum ans;
322     ans=a;
323     ans%=b;
324     return ans;
325 }
326 long long operator % (bignum& a,long long b){
327     long long ans=0,w=1;
328     for (int i=0; i<a.len; i++){
329         ans=(ans+a[i]*w%b)%b;
330         w=w*10%b;
331     }
332     return ans;
333 }
334 bignum operator % (bignum& a,char *b){
335     bignum bb;
336     bb=b;
337     return a%bb;
338 }
339 bignum operator % (bignum& a,const char *b){
340     bignum bb;
341     bb=b;
342     return a%bb;
343 }
344 bignum pow(bignum a,int x){
345     bignum ans;
346     ans=1;
347     while (x){
348         if (x%2) ans*=a;
349         a=a*a;
350         x>>=1;
351     }
352     return ans;
353 }
354 bignum groot(bignum &a,int x){
355     bignum ans;
356     ans.len=a.len/x+5;
357     bignum tmp;
358     for (int i=ans.len-1; i>=0; i--){
```

```
359         int l=0,r=9,mid;
360         while (l<r){
361             mid=(l+r)>>1;
362             mid++;
363             ans[i]=mid;
364             tmp=pow(ans,x);
365             if (tmp<=a) l=mid;
366             else r=mid-1;
367         }
368         ans[i]=l;
369     }
370 
371     while (ans.len>0&&!ans[ans.len-1]) ans.len--;
372     return ans;
373 }
374 bignum gcd(bignum a,bignum b){
375     return a%b==0LL?b:gcd(b,a%b);
376 }
377 int solve(bignum k){
378     if (k==1) return 1;
379     if (pow(groot(k,2),2)==k) return 2;
380     if (pow(groot(k,3),3)==k) return 3;
381     return 1;
382 }
383 
384 bool check[N+10];
385 long long prime[100000],tot=0;
386 map<long long,long long> MAP1,MAP2;
387 map<long long,long long>::iterator iter;
388 int main(){
389     memset(check,0,sizeof check);
390     for (int i=2; i<=N; i++){
391         if (!check[i]) prime[tot++]=i;
392         for (int j=0; j<tot; j++){
393             if (i*prime[j]>N) break;
394             check[i*prime[j]]=true;
395             if (i%prime[j]==0) break;
396         }
397     }
398 
399     int T;
400     scanf("%d",&T);
401     while (T--){
402         bignum a,b,d,tmp;
403         a.get();
404         b.get();
405         MAP1.clear();
406         MAP2.clear();
407         for (int i=0; i<tot; i++){
408             if (a%prime[i]==0){
409                 MAP1[prime[i]]=0;
410                 while (a%prime[i]==0){
411                     a/=prime[i];
412                     MAP1[prime[i]]++;
413                 }
414             }
415         }
416         for (int i=0; i<tot; i++){
417             if (b%prime[i]==0){
```

```
                MAP2[prime[i]]=0;
                while (b%prime[i]==0){
                    b/=prime[i];
                    MAP2[prime[i]]++;
                }
            }
        }
        long long ans1=1,ans2=1;
        for (iter=MAP1.begin(); iter!=MAP1.end(); iter++){
            ans1*=iter->second;
        }
        for (iter=MAP2.begin(); iter!=MAP2.end(); iter++){
            ans2*=iter->second;
        }
        if (a==1){
            printf("%lld %lld\n",ans1,ans2);
            continue;
        }
        d=gcd(a,b);
        long n = solve(d);
        bignum g=groot(d,n);
        long long c = 0;
        while (a%g==0LL){
            a/=g;
            c++;
        }
        ans1 *= c;
        ans1 *= solve(a);
        c=0;
        while (b%g==0LL){
            b/=g;
            c++;
        }
        ans2 *= c;
        ans2*=solve(b);
        printf("%lld %lld\n",ans1,ans2);
    }
    return 0;
}
```

0.4 Expression evaluation

0.4.1 Expression evaluation

```
/// No negative number
LL s[10000],b[10000],st[10000],top;
char S[10000];
LL Cal(LL a,LL b,LL c){
    LL result;
    if (c==-3){
        result=1;
        for (int i=1;i<=b;i++){
            result=result*a;
        }
    }
    else if (c==-4) {
        result=a*b;
    }
    else if (c==-5) {
```

```
16          result=(a+b);
17      }
18      else if (c==-6) {
19          result=(a-b);
20      }
21      return result;
22  }
23  LL Getans(const char *tmp){
24      LL len=0,i=0,total=0,top=-1;
25      LL n=strlen(tmp);
26      memset(s,0,sizeof s);
27      memset(b,0,sizeof b);
28      memset(st,0,sizeof st);
29      while (i<n){
30          if (tmp[i]==' '){
31              i++;
32              continue;
33          }
34          if (tmp[i]=='-') s[len++]=-6;
35          else if (tmp[i]=='+') s[len++]=-5;
36          else if (tmp[i]=='*') s[len++]=-4;
37          else if (tmp[i]=='^') s[len++]=-3;
38          else if (tmp[i]=='(') s[len++]=-2;
39          else if (tmp[i]==')') s[len++]=-1;
40          else {
41              LL ans=0;
42              while (tmp[i]>='0'&&tmp[i]<='9'){
43                  ans=ans*10+tmp[i]-'0';
44                  i++;
45              }
46              s[len++]=ans;
47              continue;
48          }
49          i++;
50      }
51      for (i=0;i<len;i++){
52          if (s[i]>=0){
53              b[total++]=s[i];
54          }
55          else {
56              if (s[i]==-1){
57                  while (st[top]!=-2){
58                      b[total++]=st[top--];
59                  }
60                  top--;
61              }
62              else if (s[i]==-2) st[++top]=s[i];
63              else {
64                  if (top==-1){
65                      st[++top]=s[i];
66                  }
67                  else if (st[top]==-2) st[++top]=s[i];
68                  else {
69                      if (s[i]==-3) {
70                          while (st[top]==s[i]){
71                              if (st[top]==-2) break;
72                              if (top<0) break;
73                              b[total++]=st[top--];
74                          }
```

```
                        st[++top]=s[i];
                    }
                    else if (s[i]==-4) {
                        while (st[top]>=s[i]){
                            if (st[top]==-2) break;
                            b[total++]=st[top--];
                            if (top<0) break;
                        }
                        st[++top]=s[i];
                    }
                    else if (s[i]==-5||s[i]==-6) {
                        while (1){
                            if (st[top]==-2) break;
                            if (top<0) break;
                            b[total++]=st[top--];
                        }
                        st[++top]=s[i];
                    }
                }
            }
        }
    }
    for (i=top;i>=0;i--)
        b[total++]=st[i];
    top=-1;
    for (int i=0;i<total;i++){
        if (b[i]>0) st[++top]=b[i];
        else {
            LL re=cal(st[top-1],st[top],b[i]);
            st[--top]=re;
        }
    }
    return st[0];
}
int main(){
    cout<<getans("-3");
    return 0;
}
```

0.5 Duipai

0.5.1 Windows

```
/// Windows£¿
int main()
{
    int t = 1000;
    while(t --){
        system("datamaker > a+b.in");
        system("truely < a+b.in > truely.out");
        system("ask < a+b.in > ask.out");
        if(system("fc ask.out truely.out"))   break;
        cout << 6 << endl;
    }
    system("pause");
    return 0;
}
```

0.5.2 Linux

```
int main()
{
    for(int i = 1; i <= 1000; i ++){
        system("./make");
        system("./ab1");
        system("./ab2");
        printf("%d : ",i);
        if(system("diff ab1.out ab2.out")){
            printf("WA\n");
            return 0;
        }
        else{
            printf("AC\n");
        }
    }

    return 0;
}
```

0.6 FFTbignummul

0.6.1 FFTbignummul

```
#define L(x) (1 << (x))
const double PI = acos(-1.0);
const int Maxn = 133015;
double ax[Maxn], ay[Maxn], bx[Maxn], by[Maxn];
char sa[Maxn/2],sb[Maxn/2];
int sum[Maxn];
int x1[Maxn],x2[Maxn];
int revv(int x, int bits){
    int ret = 0;
    for (int i = 0; i < bits; i++){
        ret <<= 1;
        ret |= x & 1;
        x >>= 1;
    }
    return ret;
}
void fft(double * a, double * b, int n, bool rev){
    int bits = 0;
    while (1 << bits < n) ++bits;
    for (int i = 0; i < n; i++){
        int j = revv(i, bits);
        if (i < j)
            swap(a[i], a[j]), swap(b[i], b[j]);
    }
    for (int len = 2; len <= n; len <<= 1){
        int half = len >> 1;
        double wmx = cos(2 * PI / len), wmy = sin(2 * PI / len);
        if (rev) wmy = -wmy;
        for (int i = 0; i < n; i += len){
            double wx = 1, wy = 0;
            for (int j = 0; j < half; j++){
                double cx = a[i + j], cy = b[i + j];
                double dx = a[i + j + half], dy = b[i + j + half];
                double ex = dx * wx - dy * wy, ey = dx * wy + dy * wx;
```

```
                a[i + j] = cx + ex, b[i + j] = cy + ey;
                a[i + j + half] = cx - ex, b[i + j + half] = cy - ey;
                double wnx = wx * wmx - wy * wmy, wny = wx * wmy + wy * wmx;
                wx = wnx, wy = wny;
            }
        }
    }
    if (rev){
        for (int i = 0; i < n; i++)
            a[i] /= n, b[i] /= n;
    }
}
int solve(int a[],int na,int b[],int nb,int ans[]){
    int len = max(na, nb), ln;
    for(ln=0; L(ln)<len; ++ln);
    len=L(++ln);
    for (int i = 0; i < len ; ++i){
        if (i >= na) ax[i] = 0, ay[i] =0;
        else ax[i] = a[i], ay[i] = 0;
    }
    fft(ax, ay, len, 0);
    for (int i = 0; i < len; ++i){
        if (i >= nb) bx[i] = 0, by[i] = 0;
        else bx[i] = b[i], by[i] = 0;
    }
    fft(bx, by, len, 0);
    for (int i = 0; i < len; ++i){
        double cx = ax[i] * bx[i] - ay[i] * by[i];
        double cy = ax[i] * by[i] + ay[i] * bx[i];
        ax[i] = cx, ay[i] = cy;
    }
    fft(ax, ay, len, 1);
    for (int i = 0; i < len; ++i)
        ans[i] = (int)(ax[i] + 0.5);
    return len;
}
string mul(string sa,string sb){
    int l1,l2,l;
    int i;
    string ans;
    memset(sum, 0, sizeof(sum));
    l1 = sa.size();
    l2 = sb.size();
    for(i = 0; i < l1; i++)
        x1[i] = sa[l1 - i - 1]-'0';
    for(i = 0; i < l2; i++)
        x2[i] = sb[l2-i-1]-'0';
    l = solve(x1, l1, x2, l2, sum);
    for(i = 0; i<l || sum[i] >= 10; i++){ // ½□□{
        sum[i + 1] += sum[i] / 10;
        sum[i] %= 10;
    }
    l = i;
    while(sum[l] <= 0 && l>0)    l--; // ¼□□□□□□□
    for(i = l; i >= 0; i--)    ans+=sum[i] + '0'; // µ¹□□□□□
    return ans;
}
```

0.7 Pbds

0.7.1 Pbds

```
/// ¿□¢¶□
#include <ext/pb_ds/priority_queue.hpp>
using namespace __gnu_pbds; ///
typedef __gnu_pbds::priority_queue <int, greater<int>, pairing_heap_tag> Heap;  ///
erase(iterator);    /// ¸□□¾□´□□□□³□□□□
modify(iterator, val);  /// ¸□□¾□´□□□□□□□
join(other);    /// □□□§□□□□¸□□□□□
/// □□□□§TL

/// ¿□□û¯□°□□□
#include <ext/rope>
using namespace __gnu_cxx;  ///
/// □±□□0¿ª□£¬²»¿□□□cin£¬¿□□□cout
/// □□□□rope□□□□□□§insert£¬erase£¬get¶¼□□lognµ□
reverse□□O(n)µ¿¬□□□□¹□□}¸□rope4□□
push_back(x);   ///    □□ĩ□□□□x
insert(pos,x);  ///   □□pos²□□□x
erase(pos,x);   ///    ´□pos¿ª□□³□x¸□
replace(pos,x); ///  ´□pos¿ª□»»³□x
substr(pos,x);  ///   □□□pos¿ª□x¸□
at(x)/[x];  ///    · □□□□x¸□□□□
rope<int>*his[maxn], his[i]=new rope<char>(*his[i-1]);  /// ¿□□û¯□□□□
```

1 Math

1.1 Montgomery modular multiplication

1.1.1 Montgomery modular multiplication

```
#include <bits/stdc++.h>
using namespace std;
#define rep(i,a,n) for (int i=a;i<n;i++)
#define per(i,a,n) for (int i=n-1;i>=a;i--)
#define pb push_back
#define mp make_pair
#define all(x) (x).begin(),(x).end()
#define fi first
#define se second
#define SZ(x) ((int)(x).size())
typedef vector<int> VI;
typedef long long ll;
typedef pair<int,int> PII;
const ll mod=1000000007;
ll powmod(ll a,ll b) {ll res=1;a%=mod; assert(b>=0); for(;b;b>>=1){if(b&1)res=res*a%mod
    ;a=a*a%mod;}return res;}
ll gcd(ll a,ll b) { return b?gcd(b,a%b):a;}
// head

typedef unsigned long long u64;
typedef __int128_t i128;
typedef __uint128_t u128;
int _,k;

struct Mod64 {
    Mod64():n_(0) {}
    Mod64(u64 n):n_(init(n)) {}
    static u64 init(u64 w) { return reduce(u128(w) * r2); }
    static void set_mod(u64 m) {
        mod=m; assert(mod&1);
        inv=m; rep(i,0,5) inv*=2-inv*m;
        r2=-u128(m)%m;
    }
    static u64 reduce(u128 x) {
        u64 y=u64(x>>64)-u64((u128(u64(x)*inv)*mod)>>64);
        return ll(y)<0?y+mod:y;
    }
    Mod64& operator += (Mod64 rhs) { n_+=rhs.n_-mod; if (ll(n_)<0) n_+=mod; return *
    this; }
    Mod64 operator + (Mod64 rhs) const { return Mod64(*this)+=rhs; }
    Mod64& operator -= (Mod64 rhs) { n_-=rhs.n_; if (ll(n_)<0) n_+=mod; return *this; }
    Mod64 operator - (Mod64 rhs) const { return Mod64(*this)-=rhs; }
    Mod64& operator *= (Mod64 rhs) { n_=reduce(u128(n_)*rhs.n_); return *this; }
    Mod64 operator * (Mod64 rhs) const { return Mod64(*this)*=rhs; }
    u64 get() const { return reduce(n_); }
    static u64 mod,inv,r2;
    u64 n_;
};
u64 Mod64::mod,Mod64::inv,Mod64::r2;

u64 pmod(u64 a,u64 b,u64 p) {
    u64 d=(u64)floor(a*(long double)b/p+0.5);
    ll ret=a*b-d*p;
```

```
52      if (ret<0) ret+=p;
53      return ret;
54  }
55
56
57  void bruteforce() {
58      u64 ans=1;
59      for (int i=0;i<=k;i++) {
60          ans=pmod(ans,A0,M);
61          u64 A2=pmod(M0,A1,M)+pmod(M1,A0,M)+C;
62          while (A2>=M) A2-=M;
63          A0=A1; A1=A2;
64      }
65      printf("%llu\n",ans);
66  }
67
68  int main()
69  {
70      u64 a, b, c;
71      Mod64::set_mod(M);
72
73      Mod64 a0(a), b0(b), ans(0);
74      scanf("%llu%llu", &a, &b);
75      ans = ans + a0 + b0;
76
77      printf("%llu\n",ans.get());
78  }
```

1.2 Screen

1.2.1 Prime

```
1   const int N = 1e6+7;
2   char v[N];
3   int pri[N];
4
5   int Init_pri(int n){
6       int len = 0;
7       memset(v,0,sizeof(v));
8       for(int i = 2; i <= n; i++){
9           if(v[i] == 0)
10              pri[len++] = i;
11          for(int j = 0; j<len && pri[j]*i<=n; j++){
12              v[i*pri[j]] = true;
13              if(i%pri[j]==0) break;
14          }
15      }
16      return len;
17  }
```

1.2.2 Mob

```
1   const int N = 1e6+7;
2   char v[N];
3   int pri[N];
4   int mu[N];
5
6   int Mobi(int x)/// ¼□□□ x μ□Ī±□□□□°¯□□
```

```
{
    int ans = 1;
    for(int i = 2,q = sqrt(x)+1; i < q; i++){
        if(x%i==0){
            x/=i;
            if(x%i)
                ans = -ans;
            else
                return 0;
            q = sqrt(x)+1;
        }
    }
    if(x!=1)
        ans = -ans;
    return ans;
}

void Init_mu(int n)
{
    memset(v,0,sizeof(v));
    int len = 0;
    mu[1] = 1;
    for(int i = 2; i <= n; i++)
    {
        if(v[i] == 0)
            mu[i] = -1,pri[len++] = i;
        for(int j = 0,k; j<len && (k=i*pri[j])<=n; j++){
            v[k] = 1;
            if(i%pri[j])
                mu[k]=-mu[i];
            else
            {
                mu[k]=0;
                break;
            }
        }
    }
}
```

1.2.3 Eul

```
const int N = 1e6+7;
int pri[N];
int phi[N];
void init_phi(int n)
{
    memset(phi,0,sizeof(phi));
    int len = 0;
    phi[1] = 1;
    for(int i = 2; i <= n; i++){
        if(phi[i] == 0)
            phi[i] = i-1,pri[len++] = i;
        for(int j = 0; j<len && (k=i*pri[j])<=n; j++)
            if(i%pri[j])
                phi[k]=phi[i]*phi[pri[j]];
            else{
                phi[k]=phi[i]*pri[j];
                break;
```

```
            }
        }
}

LL phi(LL x)//□□□ · μ»□□□□□□□□0□□□¹Ⓕ□□□□int£¬μ«³¬□²»¸°□□
{
    if(x<2)return x;
    LL ans = x;
    for(int i = 2,q = sqrt(x)+1; i < q; i++){
        if(x%i==0){
            ans-=ans/i;
            while(x%i==0)x/=i;
            q = sqrt(x)+1;
        }
    }
    if(x!=1)
        ans-=ans/x;
    return ans;
}
```

1.2.4 Inv

```
void Init_inv(int inv[],int n,int M)//□□□±£M□□□□□
{
    if(M<=n)
        n = M-1;
    inv[0] = 0;
    inv[1] = 1;
    for(int i = 2; i < n; i ++){
        inv[i] = (M-1LL*M/i*inv[M%i]%M)%M;
    }
}
```

1.2.5 Du

```
int m = 2e6,c[1000005],cnt,n;
long long phi[2000005],mu[2000005],p[100005],q[100005];
bool vis[100005];
void init()
{
    phi[1]=mu[1]=1;
    for (int i=2; i<=m; i++){
        if (!phi[i]){
            phi[i]=i-1;
            mu[i]=-1;
            c[++cnt]=i;
        }
        for (int j=1; j<=cnt && i*c[j]<=m; j++)
            if (i%c[j]){
                phi[i*c[j]]=phi[i]*(c[j]-1);
                mu[i*c[j]]=-mu[i];
            }
            else{
                phi[i*c[j]]=phi[i]*c[j];
                mu[i*c[j]]=0;
                break;
            }
    }
```

```
    for (int i=2; i<=m; i++)phi[i]+=phi[i-1],mu[i]+=mu[i-1];
}
void du_sift(long long x){
    if(vis[n/x])return;
    int l,r=2,k=n/x,t;
    vis[k]=1;
    p[k]=(x+1)*x>>1,q[k]=1;///¸□-□%□□□□□□¨³□□
    while(r<x){
        l=r,t=x/l,r=x/t+1;
        if(t<=m)p[k]-=phi[t]*(r-l),q[k]-=mu[t]*(r-l);
        else{
            du_sift(t);
            p[k]-=p[n/t],q[k]-=q[n/t];
        }
    }
}
int main()
{
    init();
    int t;
    scanf("%d",&t);
    while(t--){
        memset(vis,0,sizeof(vis));
        scanf("%d",&n);
        cnt=0;
        if(n<=m){
            printf("%lld %lld\n",phi[n],mu[n]);
            continue;
        }
        du_sift(n);
        printf("%lld %lld\n",p[1],q[1]);
    }
    return 0;
}
```

1.2.6 Min251

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 700005;
int p[N],v[N],s[N];
int np;
void init_pri(int n)
{
    np = 0;
    for(int i=2; i<=n; i++)
    {
        s[i]=s[i-1];
        if(!v[i])
            p[np++]=i,s[i]++;
        for(int j=0; j<np&&i*p[j]<=n; j++)
        {
            v[i*p[j]]=1;
            if(i%p[j]==0)
                break;
        }
    }
```

```
}
ll val[N],a[N],n;
int nq,cnt;
ll solve(ll n)
{
    cnt=0;
    nq=sqrt(n+1);
    for(ll i=1; i<=n; i=n/(n/i)+1)
        val[++cnt]=n/i;
    for(int i=1; i<=cnt; i++)
        a[i]=val[i]-1;
    for(int j=0; j<s[nq]; j++)
    {
        ll pp = (ll)p[j]*p[j];
        for(int i=1; pp<=val[i] && i<=cnt; i++)
        {
            ll d = val[i]/p[j];
            int k =(d<nq)?cnt-d+1:n/d;
            a[i]=a[i]-(a[k]-j);
        }
    }
    return a[1];
}
int main()
{
    init_pri(N/2);
    while(cin >> n)
        cout << solve(n) << endl;
    return 0;
}
```

1.2.7 Exera

```
typedef long long ll;
const int N = 700005;
int p[N],v[N],s[N];
int np;
void init_pri(int n)
{
    np = 0;
    for(int i=2; i<=n; i++)
    {
        s[i]=s[i-1];
        if(!v[i])
            p[np++]=i,s[i]++;
        for(int j=0; j<np&&i*p[j]<=n; j++)
        {
            v[i*p[j]]=1;
            if(i%p[j]==0)
                break;
        }
    }
}
long long sg(long long x){
    return x*(x+1)/2;
}
long long g(long long x)
{
```

```
26      return x;
27  }
28  long long fk(long long x,int k){
29      return x;
30  }
31  long long f(long long x)
32  {
33      return x;
34  }
35
36  ll val[N],a[N],n;
37  int nq,cnt;
38  ll nl_sift(ll n)
39  {
40      cnt=0;
41      nq=sqrt(n+1);
42      for(ll i=1; i<=n; i=n/(n/i)+1)
43          val[++cnt]=n/i;
44      for(int i=1; i<=cnt; i++)
45          a[i]=sg(val[i])-g(1);
46      ll sj = 0;
47      for(int j=0; j<s[nq]; sj+=g(p[j++]))
48      {
49          ll pp = (ll)p[j]*p[j];
50          for(int i=1; pp<=val[i] && i<=cnt; i++)
51          {
52              ll d = val[i]/p[j];
53              int k =(d<nq)?cnt-d+1:n/d;
54              a[i]=a[i]+g(p[j])*(sj-a[k]);
55          }
56      }
57      for(int j=s[nq]-1; ~j; sj-=g(p[j--]))
58      {
59          ll pp = (ll)p[j]*p[j];
60          for(int i=1; pp<=val[i] && i<=cnt; i++)
61          {
62              ll d = val[i]/p[j];
63              ll ff = p[j];
64              for(int l = 1; p[j]<=d; d/=p[j],l++){
65                  int k =(d<nq)?cnt-d+1:n/d;
66                  a[i] = a[i]+f(ff)*(a[k]-sj)+f(ff*=p[j]);///¿□□□□□ff
67                  //a[i] = a[i]+fk(p[j],l)*(a[k]-sj)+fk(p[j],l+1);///□□□□□□□ff
68              }
69          }
70      }
71      return a[1]+f(1);
72  }
73  int main()
74  {
75      init_pri(N/2);
76      while(cin >> n)
77          cout << nl_sift(n) << ' ' << sg(n) << endl;
78      return 0;
79  }
```

1.3 Fast Power

1.3.1 Numpower

```
LL quickmod(LL a, LL b, LL MOD)
{
    LL ans;

    ans = 1;
    while(b){
        if(b & 1){
            ans = (ans * a) % MOD;
        }
        b >>= 1;
        a = a * a % MOD;
    }

    return ans % MOD;
}
```

1.3.2 Marpower

```
const int N = 3;    /// ?????
const int MOD = 1e9 + 7;

LL a, b, c, d, e, n;

struct Met
{
    LL ma[3][3];
};

Met Mul(Met a, Met b)
{
    Met c;

    for(int i = 0; i < N; i ++){
        for(int j = 0; j < N; j ++){
            c.ma[i][j] = 0;
        }
    }
    for(int i = 0; i < N; i ++){
        for(int j = 0; j < N; j ++){
            LL t;

            t = 0;
            for(int k = 0; k < N; k ++){
                t = (t + a.ma[i][k] * b.ma[k][j] % MOD) % MOD;
            }
            c.ma[i][j] = t;
        }
    }

    return c;
}

Met Quick_mod(Met a, int n)
{
    Met t;
```

```
38
39      for(int i = 0; i < N; i ++){
40          for(int j = 0; j < N; j ++){
41              t.ma[i][j] = 0;
42          }
43      }
44      for(int i = 0; i < N; i ++){
45          t.ma[i][i] = 1;
46      }
47      while(n){
48          if(n & 1){
49              t = Mul(t, a);
50          }
51          n >>= 1;
52          a = Mul(a, a);
53      }
54
55      return t;
56  }
57
58  int main()
59  {
60      /**
61      dp[1] = A;
62      dp[2] = B;
63      dp[i] = C * dp[i - 2] + D * dp[i - 1] + E;
64      */
65
66      now.ma[0][0] = d % MOD;
67      now.ma[0][1] = c % MOD;
68      now.ma[1][0] = 1;
69      now.ma[2][2] = 1;
70      now.ma[0][2] = e;
71
72      return 0;
73  }
```

1.4 Transformation

1.4.1 FFT

```
1   const double PI = acos(-1.0); //PI
2   void FFT_d(complex<double> a[],int n,bool on=false) //³¤¶□□N (2μ□□□□),□□□□□±任£¬□□任¼□
        true
3   {
4       int r=0;
5       while((1<<++r)!=n);
6       for(int i=0; i<n; i++){
7           int tmp=0;
8           for(int j=0; j<r; j++)//°□μ□²□□□
9               if(i&(1<<j))
10                  tmp+=1<<(r-j-1);
11          if(i<tmp)
12              swap(a[i],a[tmp]);
13      }
14      for(int i=1; i<=r; i++){
15          int m=1<<i;
16          complex<double> wn(cos(2*PI/m),sin(2*PI/m));
17          for(int k=0; k<n; k+=m){
```

```
            complex<double> w(1,0);
            for(int j=0; j<(m>>1); j++){
                complex<double> t,u;
                t=w*(a[k+j+(m>>1)]);
                u=a[k+j];
                a[k+j]=(u+t);
                a[k+j+(m>>1)]=u-t;
                w=w*wn;
            }
        }
    }
    if(on){
        for(int i=1; i<n>>1; i++)
            swap(a[i],a[n-i]);
        complex<double> in(1.0/n);
        for(int i=0; i<n; i++)
            a[i]=a[i]*in;
    }
}
```

1.4.2 FWT

```
const int mod = 1e9+7;
int inv2 = 500000004;
void FWT(int a[], int n)//□□□±fn□2μ□□□□□´□□□
{
    for(int d = 1; d < n; d <<= 1){
        for(int m=d<<1,i=0;i<n;i+=m){
            for(int j=0;j<d;j++){
                int x=a[i+j],y=a[i+j+d];
                //xor_MOD:a[i+j]=(x+y)%mod,a[i+j+d]=(x-y+mod)%mod;
                //xor:a[i+j]=x+y,a[i+j+d]=x-y;
                //and:a[i+j]=x+y;
                //or:a[i+j+d]=x+y;
            }
        }
    }
}

void UFWT(int a[],int n)//□□□±fn□2μ□□□□□´□□□
{
    for(int d=1;d<n;d<<=1){
        for(int m=d<<1,i=0;i<n;i+=m){
            for(int j=0;j<d;j++){
                int x=a[i+j],y=a[i+j+d];
                //xor_MOD:a[i+j]=1LL*(x+y)*inv2%mod,a[i+j+d]=(1LL*(x-y)*inv2%mod+mod)%
    mod;
                //xor:a[i+j]=(x+y)/2,a[i+j+d]=(x-y)/2;
                //and:a[i+j]=x-y;
                //or:a[i+j+d]=y-x;
            }
        }
    }
}
```

1.4.3 NTT

```
const int mod = (479<<21)+1;
const int g = 3;  //□¸□
long long quick_mod(long long a,long long b)
{
    long long ans=1;
    while(b){
        if(b&1)
            ans=ans*a%mod;
        a=a*a%mod;
        b>>=1;
    }
    return ans;
}
void NTT(int n,long long a[],bool on=false) //³¤¶□□N (2μ□□□□),□□□□□±任£¬□□任¼□true
{
    int r=0;
    while((1<<++r)!=n);
    for(int i=0; i<n; i++){
        int tmp=0;
        for(int j=0; j<r; j++)//°□μ□²□□□
            if(i&(1<<j))
                tmp+=1<<(r-j-1);
        if(i<tmp)
            swap(a[i],a[tmp]);
    }
    for(int i=1; i<=r; i++){
        int m=1<<i;
        long long wn=quick_mod(g,(mod-1)/m);
        for(int k=0; k<n; k+=m){
            long long w=1;
            for(int j=0; j<(m>>1); j++){
                long long t,u;
                t=w*(a[k+j+(m>>1)]%mod)%mod;
                u=a[k+j]%mod;
                a[k+j]=(u+t)%mod;
                a[k+j+(m>>1)]=((u-t)%mod+mod)%mod;
                w=w*wn%mod;
            }
        }
    }
    if(on){
        for(int i=1; i<n>>1; i++)
            swap(a[i],a[n-i]);
        long long inv=quick_mod(n,mod-2);
        for(int i=0; i<n; i++)
            a[i]=a[i]%mod*inv%mod;
    }
}
```

1.5 Gcd

1.5.1 Exgcd

```
LL Exgcd(LL a, LL b, LL &x, LL &y, LL c = 1)
{
    LL gcd;

    if(!b){
        if(c % a){
```

```
            return 0;
        }
        x = c / a;
        y = 0;

        return a;
    }
    gcd = exgcd(b, a % b, y, x, c);
    y -= a / b * x;

    return gcd;
}
```

1.6 Miller–Rabin primality test

1.6.1 Miller–Rabin primality test

```
bool Rqui_prime(int x)//x²»□□int   /// shizhushuyes = true
{
    srand(time(0));
    int n = log(x)+7;
    if(quick_mod(7,x-1,x) != 1){
        return false;
    }
    for(int i = 0; i < n; i++){
        long long k = rand()%10000+7;
        if((k%x)&&(quick_mod(k,x-1,x)!=1)){ /// ¿□□□□□{
            return false;
        }
    }

    return true;
}
```

1.7 Equivalence sequence continuous XOR

1.7.1 Equivalence sequence continuous XOR

```
LL cal(LL a,LL b,LL c,LL n)
{
    LL re;

    re = 0;
    re += (a / c) * n * (n - 1) / 2 + (b / c) * n;
    b %= c;
    a %= c;
    if(a * n + b < c){
        return re;
    }
    else{
        return re+cal(c,(a*n+b)%c,a,(a*n+b)/c);
    }
}
int main()
{
    LL x,y,z;
    while(scanf("%lld%lld%lld",&x,&y,&z) == 3){
        LL ans=0;
```

```
        LL n=(y-x)/z+1;
        for(int i=0;i<32;i++){
            ans|=(cal(z,x,1LL<<i,n)&1)<<i;
        }
        printf("%lld\n",ans);
    }
    return 0;
}
```

1.8 Linear and Determinant

1.8.1 Linear basis

```
int Gauss()
{
    int cnt;

    cnt = 0;    ///%
    for(int i = 1; i <= n; i ++){
        for(int j = 62; j >= 0; j --){
            if((ma[i] >> j) & 1){
                if(!p[j]){
                    p[j] = ma[i];
                    break;
                }
                else{
                    ma[i] ^= p[j];
                }
            }
        }
    }
    for(int i = 0; i <= 62; i++){
        if(p[i]){
            cnt ++;
        }
    }

    return cnt;
}
```

1.8.2 Simplex

```
// ---
// 输入矩阵$a$描述线性规划的标准形式。\\\\
// $a$为$m+1$行$n+1$列，其中行$0 \sim m-1$为不等式，行$m$为目标函数（最大化）。\\\\
// 列$0 \sim n-1$为变量$0 \sim n-1$的系数，列$n$为常数项。\\\\
// 约束为$a_{i, 0}x_0 + a_{i, 1}x_1 + \cdots \le a_{i, n}$，目标为$\max(a_{m, 0}x_0 + a_{
    m, 1}x_1 + \cdots + a_{m, n - 1}x_{n - 1} - a_{m, n})$\\\\
// 注意：变量均有非负约束$x[i] \ge 0$
// ---
const int maxm = 500; // 约束数目上限
const int maxn = 500; // 变量数目上限
const double INF = 1e100;
const double eps = 1e-10;
struct Simplex
{
    int n;                  // 变量个数
    int m;                  // 约束个数
```

```
16      double a[maxm][maxn]; // 输入矩阵
17      int B[maxm], N[maxn]; // 算法辅助变量
18      void pivot(int r, int c)
19      {
20          swap(N[c], B[r]);
21          a[r][c] = 1 / a[r][c];
22          for (int j = 0; j <= n; j++)
23              if (j != c) a[r][j] *= a[r][c];
24          for (int i = 0; i <= m; i++)
25              if (i != r)
26              {
27                  for (int j = 0; j <= n; j++)
28                      if (j != c) a[i][j] -= a[i][c] * a[r][j];
29                  a[i][c] = -a[i][c] * a[r][c];
30              }
31      }
32      bool feasible()
33      {
34          for (;;)
35          {
36              int r, c;
37              double p = INF;
38              for (int i = 0; i < m; i++)
39                  if (a[i][n] < p) p = a[r = i][n];
40              if (p > -eps) return true;
41              p = 0;
42              for (int i = 0; i < n; i++)
43                  if (a[r][i] < p) p = a[r][c = i];
44              if (p > -eps) return false;
45              p = a[r][n] / a[r][c];
46              for (int i = r + 1; i < m; i++)
47                  if (a[i][c] > eps)
48                  {
49                      double v = a[i][n] / a[i][c];
50                      if (v < p) r = i, p = v;
51                  }
52              pivot(r, c);
53          }
54      }
55      // 解有界返回1，无解返回0，无界返回-1。b[i]为x[i]的值，ret为目标函数的值
56      int simplex(int n, int m, double x[maxn], double& ret)
57      {
58          this->n = n, this->m = m;
59          for (int i = 0; i < n; i++) N[i] = i;
60          for (int i = 0; i < m; i++) B[i] = n + i;
61          if (!feasible()) return 0;
62          for (;;)
63          {
64              int r, c;
65              double p = 0;
66              for (int i = 0; i < n; i++)
67                  if (a[m][i] > p) p = a[m][c = i];
68              if (p < eps)
69              {
70                  for (int i = 0; i < n; i++)
71                      if (N[i] < n) x[N[i]] = 0;
72                  for (int i = 0; i < m; i++)
73                      if (B[i] < n) x[B[i]] = a[i][n];
74                  ret = -a[m][n];
```

```
                return 1;
            }
            p = INF;
            for (int i = 0; i < m; i++)
                if (a[i][c] > eps)
                {
                    double v = a[i][n] / a[i][c];
                    if (v < p) r = i, p = v;
                }
            if (p == INF) return -1;
            pivot(r, c);
        }
    }
};
```

1.9 Original root

1.9.1 Original root

```
/*
□□□□□□□□pμ□□□□□□¸□£¬¸´□□□□□p*log^2(p)/phi(p-1)
□78494¸□□□□□(1~1000000)μ±□□¬□¸□μ□□%□□□488.£¬¿□□□□□□□³£□□¡£
□□□´□□□□□log^2(p)
*/
#define N 1000000
int top=0;
long long st[40];
void init(long long m){
    top=0;
    memset(st,0,sizeof st);
    for (long long i=2;i<=sqrt(m)+1;i++){
        if (m%i==0){
            st[top++]=i;
            while (m%i==0) m/=i;
        }
    }
    if (m>1) st[top++]=m;
}
long long solve(long long p){
    init(p-1);
    for (long long g=1;g<=p-1;g++){
        bool bb=true;
        for (int j=0;j<top;j++){
            long long mod=power(g,(p-1)/st[j],p);   /// ¿□□□□□
            if (mod==1) {
                bb=false;
                break;
            }
        }
        if (bb){
            return g;
        }
    }
}
int main(){
    long long p;
    cin>>p;
    cout<<solve(p)<<endl;
    return 0;
```

```
}
```

1.10 Lucas

1.10.1 Lucas

```
#define P 110119
using namespace std;
typedef pair<LL ,LL> PC;
LL fac[P],inv[P],facinv[P];
//□´¦□□□□□□□□□□□□Pμ□□□□□□□□μ□□□□£¬½□□□□°½□□□□□□□
void Init_lucas(){
    fac[0]=fac[1]=inv[0]=facinv[0]=inv[1]=facinv[1]=1;
    for(int i = 2; i < P; ++i) {
        fac[i] = fac[i - 1] * i % P;
        inv[i] = inv[P % i] * (P - P / i) % P;
//□□□□□□□¿□□□□□o(n)μ□´□□□□□□□□□□□□□□□μ□□□□
        facinv[i] = facinv[i - 1] * inv[i] % P;
    }
}
//n<=10^18,m<=10^18,p<=10^5£¬□□□□□□□
LL GetC(LL n,LL m,LL p){
    if (m>n) return 0;
    if (!n) return 1;
    LL ret=fac[n]*facinv[m]%p*facinv[n-m]%p;
    return ret;
}
/// lucas¶¨□□□□□□□□,n<=10^18,m<=10^18,p<=10^5
LL Pick(LL n,LL m,LL p){
    if (!m) return 1;
    LL a[100],b[100],i=-1;
    while (n){
        a[++i]=n%p;
        n/=p;
        b[i]=m%p;
        m/=p;
    }
    LL k=i;
    LL ret=1;
    for (int i=0;i<=k;i++){
        ret*=getC(a[i],b[i],p);
        ret%=p;
    }
    return ret;
}
int main(){
    init_lucas();
    cout<<pick(5,3,P)<<endl;
    return 0;
}
```

1.10.2 Ex$_L ucas$

```
LL exgcd(LL a, LL b, LL& x, LL& y) {
    if(!b) {
        x = 1;
        y = 0;
        return a;
```

```
6       }
7       LL res = exgcd(b, a%b, y, x);
8       y -= (a/b)*x;
9       return res;
10  }
11
12  LL reverse(LL a, LL p) {
13      LL x, y;
14      exgcd(a, p, x, y);
15      return (x % p + p)%p;
16  }
17
18  LL C(LL n, LL m, LL p) {
19      if(m>n) return 0;
20      LL res = 1, i, a, b;
21      for(i = 1; i <= m; i++) {
22          a = (n+1-i) % p;
23          b = reverse(i%p, p);
24          res = res*a%p*b%p;
25      }
26      return res;
27  }
28
29  LL Lucas(LL n, LL m, LL p) {
30      if(m == 0) return 1;
31      return Lucas(n/p, m/p, p)*C(n%p, m%p, p) % p;
32  }
33
34  LL cal(LL n, LL a, LL b, LL p) {
35      if(!n) return 1;
36      LL i, y = n/p, tmp = 1;
37      for(i = 1; i <= p; i++) if(i%a) tmp = tmp*i%p;
38      LL ans = pow(tmp, y, p);
39      for(i = y*p+1; i <= n; i++) if(i%a) ans = ans*i%p;
40      return ans * cal(n/a, a, b, p)%p;
41  }
42
43  LL multiLucas(LL n, LL m, LL a, LL b, LL p) {
44      LL i, t1, t2, t3, s = 0, tmp;
45      for(i = n; i; i/=a) s += i/a;
46      for(i = m; i; i/=a) s -= i/a;
47      for(i = n-m; i; i/=a) s -= i/a;
48      tmp = pow(a, s, p);
49      t1 = cal(n, a, b, p);
50      t2 = cal(m, a, b, p);
51      t3 = cal(n-m, a, b, p);
52      return tmp*t1%p*reverse(t2, p)%p*reverse(t3, p)%p;
53  }
54
55
56  LL exLucas(LL n, LL m, LL p) {
57      LL i, d, c, t, x, y, q[100], a[100], e = 0;
58      for(i = 2; i*i <= p; i++) {
59          if(p % i == 0) {
60              q[++e] = 1;
61              t = 0;
62              while(p%i==0) {
63                  p /= i;
64                  q[e] *= i;
```

```
                t++;
            }
            if(q[e] == i) a[e] = Lucas(n, m, q[e]);
            else a[e] = multiLucas(n, m, i, t, q[e]);
        }
    }
    if(p > 1) {
        q[++e] = p;
        a[e] = Lucas(n, m, p);
    }
    for(i = 2; i <= e; i++) {
        d = exgcd(q[1], q[i], x, y);
        c = a[i]-a[1];
        if(c%d) exit(-1);
        t = q[i]/d;
        x = (c/d*x%t+t)%t;
        a[1] = q[1]*x+a[1];
        q[1] = q[1]*q[i]/d;
    }
    return a[1];
}
```

1.11 Bsgs

1.11.1 Bsgs

```
/// □□□ a^x = b (mod p)
const int MAXINT=((1<<30)-1)*2+1;

int A,B,C;
struct Hashmap //¹□□±□□□□map
{
    static const int Ha=999917,maxe=46340;
    int E,lnk[Ha],son[maxe+5],nxt[maxe+5],w[maxe+5];
    int top,stk[maxe+5];
    void clear() {E=0;while (top) lnk[stk[top--]]=0;}
    void Add(int x,int y) {son[++E]=y;nxt[E]=lnk[x];w[E]=MAXINT;lnk[x]=E;}
    bool count(int y){
        int x=y%Ha;
        for (int j=lnk[x];j;j=nxt[j])
            if (y==son[j]) return true;
        return false;
    }
    int& operator [] (int y){
        int x=y%Ha;
        for (int j=lnk[x];j;j=nxt[j])
            if (y==son[j]) return w[j];
        Add(x,y);stk[++top]=x;return w[E];
    }
};
Hashmap f;
int exgcd(int a,int b,int &x,int &y)
{
}
int BSGS(int A,int B,int C)
{
    if (C==1) if (!B) return A!=1; else return -1;
    if (B==1) if (A) return 0; else return -1;
    if (A%C==0) if (!B) return 1; else return -1; //¼¸□□□□□
```

```
    int m=ceil(sqrt(C)),D=1,Base=1;f.clear();
    for (int i=0;i<=m-1;i++){ //□□□A^j´□□¹□□±□{
        f[Base]=min(f[Base],i);
        Base=((LL)Base*A)%C;
    }
    for (int i=0;i<=m-1;i++){
        int x,y,r=exgcd(D,C,x,y);
        x=((LL)x*B%C+C)%C; //)ÿ□□A^j
        if (f.count(x)) return i*m+f[x]; //□□%□□
        D=((LL)D*Base)%C;
    }
    return -1;
}
int main()
{
    while (~scanf("%d%d%d",&C,&A,&B)){
        int ans=BSGS(A,B,C);
        if (ans==-1) printf("no solution\n"); else
        printf("%d\n",ans);
    }
    return 0;
}
```

1.11.2 Exbsgs

```
/// □□□ a^x = b (mod p)
int t, a, b, m;
unordered_map<LL, int> mp;

LL EXBSGS(int A, int B, int C) {
    A %= C, B %= C;
    if(B == 1) return 0;
    int cnt = 0;
    LL t = 1;
    for(int g = __gcd(A, C); g != 1; g = __gcd(A, C)) {
        if(B % g) return -1;
        C /= g, B /= g;
        t = t * A / g % C;
        cnt++;
        if(B == t) return cnt;
    }
    mp.clear();
    int m = ceil(sqrt(1.0 * C));
    LL base = B;
    for(int i = 0; i < m; i++) {
       mp[base] = i;
       base = base * A % C;
    }
    base = Mod_Pow(A, m, C);
    LL nw = t;
    for(int i = 1; i <= m + 1; i++) {
        nw = base * nw % C;
        if(mp.count(nw)) {
            return i * m - mp[nw] + cnt;
        }
    }
    return -1;
}
```

```

LL exbsgs(LL a, LL b, LL p)
{
    if (b == 1LL) return 0;
    ll t, d = 1, k = 0;
    while ((t = gcd(a, p)) != 1)
    {
        if (b % t) return -1;
        ++k, b /= t, p /= t, d = d * (a / t) % p;
        if (b == d) return k;
    }
    map<LL, LL> dic;
    ll m = ceil(sqrt(p));
    ll a_m = Pow(a, m, p), mul = b;
    for (ll j = 1; j <= m; ++j) mul = mul * a % p, dic[mul] = j;
    for (ll i = 1; i <= m; ++i)
    {
        d = d * a_m % p;
        if (dic[d]) return i * m - dic[d] + k;
    }
    return -1;
}

int main() {
    scanf("%d", &t);
    while(t--) {
        scanf("%d%d%d", &a, &b, &m);
        LL ans = EXBSGS(a, b, m);
        printf("%lld\n", ans);
    }
    return 0;
}
```

1.12 Similar to Euclidean

1.12.1 F

```
/// $f(a, b, c, m) = \sum_{i = 0}^{n}[(ai+b)/c]$
/// □□□□□□□□
LL likegcd_f(long long a,long long b,long long c,long long n)///F(a,b,c,n)=¡□i(0~n) (ai+b)/c
{
    if(a==0) return((b/c)*(n+1));
    if(a>=c||b>=c) return likegcd_f(a%c,b%c,c,n)+(a/c)*n*(n+1)/2+(b/c)*(n+1);
    long long m=(a*n+b)/c;
    long long v=likegcd_f(c,c-b-1,a,m-1);
    return n*m-v;
}
```

1.13 Xsister

1.13.1 BM

```
const int MOD = 1000000007;

int inverse(int a) {
    return a == 1 ? 1 : (long long)(MOD - MOD / a) * inverse(MOD % a) % MOD;
}
```

```

// Berlekamp-Massey Algorithm
// Requirement: const MOD, inverse(int)
// Input: vector<int> the first elements of the sequence
// Output: vector<int> the recursive equation of the given sequence
// Example:  In: {1, 1, 2, 3}  Out: {1, 1000000006, 1000000006} (MOD = 1e9+7)

struct Poly {
    vector<int> a;

    Poly() { a.clear(); }

    Poly(vector<int> &a): a(a) {}

    int length() const { return a.size(); }

    Poly move(int d) {
        vector<int> na(d, 0);
        na.insert(na.end(), a.begin(), a.end());
        return Poly(na);
    }

    int calc(vector<int> &d, int pos) {
        int ret = 0;
        for (int i = 0; i < (int)a.size(); ++i) {
            if ((ret += (long long)d[pos - i] * a[i] % MOD) >= MOD) {
                ret -= MOD;
            }
        }
        return ret;
    }

    Poly operator - (const Poly &b) {
        vector<int> na(max(this->length(), b.length()));
        for (int i = 0; i < (int)na.size(); ++i) {
            int aa = i < this->length() ? this->a[i] : 0,
                bb = i < b.length() ? b.a[i] : 0;
            na[i] = (aa + MOD - bb) % MOD;
        }
        return Poly(na);
    }
};

Poly operator * (const int &c, const Poly &p) {
    vector<int> na(p.length());
    for (int i = 0; i < (int)na.size(); ++i) {
        na[i] = (long long)c * p.a[i] % MOD;
    }
    return na;
}

vector<int> solve(vector<int> a) {
    int n = a.size();
    Poly s, b;
    s.a.push_back(1), b.a.push_back(1);
    for (int i = 1, j = 0, ld = a[0]; i < n; ++i) {
        int d = s.calc(a, i);
        if (d) {
            if ((s.length() - 1) * 2 <= i) {
```

```
65                  Poly ob = b;
66                  b = s;
67                  s = s - (long long)d * inverse(ld) % MOD * ob.move(i - j);
68                  j = i;
69                  ld = d;
70              } else {
71                  s = s - (long long)d * inverse(ld) % MOD * b.move(i - j);
72              }
73          }
74      }
75      return s.a;
76  }
77
78  //end of template
79
80  int main() {
81      int T = 1000;
82      for (int i = 0; i < T; ++i) {
83          cout << "Test " << i + 1 << endl;
84          int n = rand() % 1000 + 1;
85          vector<int> s;
86          for (int i = 0; i < n; ++i) {
87              s.push_back(rand() % (MOD - 1) + 1);
88          }
89          vector<int> a;
90          for (int i = 0; i < n; ++i) {
91              a.push_back(rand() % MOD);
92          }
93          for (int i = 0; i < n; ++i) {
94              int na = 0;
95              for (int j = 0; j < n; ++j) {
96                  if ((na += (long long)a[n + i - 1 - j] * s[j] % MOD) >= MOD) {  /// a_{
    n + 1} = a_{n} * s{0} + a_{n - 1} * s{1} ……
97                      na -= MOD;
98                  }
99              }
100             a.push_back(na);
101         }
102         vector<int> ss = solve(a);
103
104         for (int i = 0; i < n; ++i) {
105             printf("%d%c", s[i], i == n - 1 ? '\n' : ' ');
106         }
107         cout << endl;
108         for (int i = 0; i < n; ++i) {
109             printf("%d%c", ss[i + 1], i == n - 1 ? '\n' : ' ');
110         }
111
112         assert((int)ss.size() == n + 1);
113         assert(ss[0] == 1);
114         for (int i = 0; i < n; ++i) {
115             assert((ss[i + 1] + s[i]) % MOD == 0);
116         }
117     }
118     cout << "All tests OK!!!" << endl;
119     return 0;
120 }
```

1.13.2 Blackbox1

```
const int LOG = 31, MOD = 1000000007;

// Calculating kth term of linear recurrence sequence
// Complexity: init O(n^2log) query O(n^2logk)
// Requirement: const LOG const MOD
// Input(constructor): vector<int> - first n terms
//                     vector<int> - transition function
// Output(calc(k)): int - the kth term mod MOD
// Example: In: {1, 1} {2, 1} an = 2an-1 + an-2
//          Out: calc(3) = 3, calc(10007) = 71480733 (MOD 1e9+7)

struct LinearRec {

    int n;
    vector<int> first, trans;
    vector<vector<int> > bin;

    vector<int> add(vector<int> &a, vector<int> &b) {
        vector<int> result(n * 2 + 1, 0);
        //You can apply constant optimization here to get a ~10x speedup
        for (int i = 0; i <= n; ++i) {
            for (int j = 0; j <= n; ++j) {
                if ((result[i + j] += (long long)a[i] * b[j] % MOD) >= MOD) {
                    result[i + j] -= MOD;
                }
            }
        }
        for (int i = 2 * n; i > n; --i) {
            for (int j = 0; j < n; ++j) {
                if ((result[i - 1 - j] += (long long)result[i] * trans[j] % MOD) >= MOD
    ) {
                    result[i - 1 - j] -= MOD;
                }
            }
            result[i] = 0;
        }
        result.erase(result.begin() + n + 1, result.end());
        return result;
    }

    LinearRec(vector<int> &first, vector<int> &trans):first(first), trans(trans) {
        n = first.size();
        vector<int> a(n + 1, 0);
        a[1] = 1;
        bin.push_back(a);
        for (int i = 1; i < LOG; ++i) {
            bin.push_back(add(bin[i - 1], bin[i - 1]));
        }
    }

    int calc(int k) {
        vector<int> a(n + 1, 0);
        a[0] = 1;
        for (int i = 0; i < LOG; ++i) {
            if (k >> i & 1) {
                a = add(a, bin[i]);
            }
```

```
        }
        int ret = 0;
        for (int i = 0; i < n; ++i) {
            if ((ret += (long long)a[i + 1] * first[i] % MOD) >= MOD) {
                ret -= MOD;
            }
        }
        return ret;
    }
};

//end of template

//test on http://tdpc.contest.atcoder.jp/tasks/tdpc_fibonacci

int n, k;

int main() {
    vector<int> a, b;
    scanf("%d%d", &n, &k);
    for(int i = 0; i < n; i ++){
        int x;

        scanf("%d", &x);
        a.push_back(x);
    }
    for(int i = 0; i < n; i ++){
        int x;

        scanf("%d", &x);
        b.push_back(x);
    }
    /// vector<int> a(n, 1);
    LinearRec f(a, b);
    printf("%d\n", f.calc(k));
    return 0;
}
```

1.13.3 Blackbox2

```
#define rep(i,a,n) for (int i=a;i<n;i++)
#define per(i,a,n) for (int i=n-1;i>=a;i--)
typedef pair<int,int> PII;
const ll mod=1000000007;
ll powmod(ll a,ll b) {ll res=1;a%=mod; assert(b>=0); for(;b;b>>=1){if(b&1)res=res*a%mod
    ;a=a*a%mod;}return res;}
// head

int _,n;
namespace linear_seq {
    const int N=10010;
    ll res[N],base[N],_c[N],_md[N];

    vector<int> Md;
    void mul(ll *a,ll *b,int k) {
        rep(i,0,k+k) _c[i]=0;
        rep(i,0,k) if (a[i]) rep(j,0,k) _c[i+j]=(_c[i+j]+a[i]*b[j])%mod;
        for (int i=k+k-1;i>=k;i--) if (_c[i])
```

```
18              rep(j,0,SZ(Md)) _c[i-k+Md[j]]=(_c[i-k+Md[j]]-_c[i]*_md[Md[j]])%mod;
19          rep(i,0,k) a[i]=_c[i];
20      }
21      int solve(ll n,VI a,VI b) { // a 系数 b 初值 b[n+1]=a[0]*b[n]+...
22  //        printf("%d\n",SZ(b));
23          ll ans=0,pnt=0;
24          int k=SZ(a);
25          assert(SZ(a)==SZ(b));
26          rep(i,0,k) _md[k-1-i]=-a[i];_md[k]=1;
27          Md.clear();
28          rep(i,0,k) if (_md[i]!=0) Md.push_back(i);
29          rep(i,0,k) res[i]=base[i]=0;
30          res[0]=1;
31          while ((1ll<<pnt)<=n) pnt++;
32          for (int p=pnt;p>=0;p--) {
33              mul(res,res,k);
34              if ((n>>p)&1) {
35                  for (int i=k-1;i>=0;i--) res[i+1]=res[i];res[0]=0;
36                  rep(j,0,SZ(Md)) res[Md[j]]=(res[Md[j]]-res[k]*_md[Md[j]])%mod;
37              }
38          }
39          rep(i,0,k) ans=(ans+res[i]*b[i])%mod;
40          if (ans<0) ans+=mod;
41          return ans;
42      }
43      VI BM(VI s) {
44          VI C(1,1),B(1,1);
45          int L=0,m=1,b=1;
46          rep(n,0,SZ(s)) {
47              ll d=0;
48              rep(i,0,L+1) d=(d+(ll)C[i]*s[n-i])%mod;
49              if (d==0) ++m;
50              else if (2*L<=n) {
51                  VI T=C;
52                  ll c=mod-d*powmod(b,mod-2)%mod;
53                  while (SZ(C)<SZ(B)+m) C.pb(0);
54                  rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
55                  L=n+1-L; B=T; b=d; m=1;
56              } else {
57                  ll c=mod-d*powmod(b,mod-2)%mod;
58                  while (SZ(C)<SZ(B)+m) C.pb(0);
59                  rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
60                  ++m;
61              }
62          }
63          return C;
64      }
65      int gao(VI a,ll n) {
66          VI c=BM(a);
67          c.erase(c.begin());
68          rep(i,0,SZ(c)) c[i]=(mod-c[i])%mod;
69          return solve(n,c,VI(a.begin(),a.begin()+SZ(c)));
70      }
71  };
72
73  int main() {
74      while (~scanf("%d",&n)) {
75          vector<int>v;
76          v.push_back(1);
```

```
        v.push_back(2);
        v.push_back(4);
        v.push_back(7);
        v.push_back(13);
        v.push_back(24);
        //VI{1,2,4,7,13,24}
        printf("%d\n",linear_seq::gao(v,n-1));
    }
}
```

1.13.4 MatrixMul

```
const int MOD = 1000000007, M = 52, LOG = 63;

int m; // dimension

struct Vector {
    int a[M];

    Vector() {
        memset(a, 0, sizeof(a));
    }

    int& operator[] (const int &i) { return a[i]; }
    const int operator[] (const int &i) const { return a[i]; }

    int operator * (const Vector &b) {
        int ret = 0;
        for (int i = 0; i < m; ++i) {
            if ((ret += (long long)a[i] * b[i] % MOD) >= MOD) {
                ret -= MOD;
            }
        }
        return ret;
    }

    Vector operator + (const Vector &b) {
        Vector ret;
        for (int i = 0; i < m; ++i) {
            if ((ret[i] = a[i] + b[i]) >= MOD) {
                ret[i] -= MOD;
            }
        }
        return ret;
    }

};

Vector operator * (int k, const Vector &b) {
    Vector ret;
    for (int i = 0; i < m; ++i) {
        ret[i] = (long long)k * b[i] % MOD;
    }
    return ret;
}

// Reimplement this structure to support sparse matrix
struct Matrix {
```

```
47      int a[M][M];
48
49      int* operator[] (const int &i) { return a[i]; }
50      const int* operator[] (const int &i) const { return a[i]; }
51
52      Vector operator * (const Vector &b) {
53          Vector ret;
54          for (int i = 0; i < m; ++i) {
55              for (int j = 0; j < m; ++j) {
56                  if ((ret[i] += (long long)a[i][j] * b[j] % MOD) >= MOD) {
57                      ret[i] -= MOD;
58                  }
59              }
60          }
61          return ret;
62      }
63  };
64
65  // Berlekamp-Massey Algorithm
66
67  int inverse(int a) {
68      return a == 1 ? 1 : (long long)(MOD - MOD / a) * inverse(MOD % a) % MOD;
69  }
70
71  struct Poly {
72      vector<int> a;
73
74      Poly() { a.clear(); }
75
76      Poly(vector<int> &a): a(a) {}
77
78      int length() const { return a.size(); }
79
80      Poly move(int d) {
81          vector<int> na(d, 0);
82          na.insert(na.end(), a.begin(), a.end());
83          return Poly(na);
84      }
85
86      int calc(vector<int> &d, int pos) {
87          int ret = 0;
88          for (int i = 0; i < (int)a.size(); ++i) {
89              if ((ret += (long long)d[pos - i] * a[i] % MOD) >= MOD) {
90                  ret -= MOD;
91              }
92          }
93          return ret;
94      }
95
96      Poly operator - (const Poly &b) {
97          vector<int> na(max(this->length(), b.length()));
98          for (int i = 0; i < (int)na.size(); ++i) {
99              int aa = i < this->length() ? this->a[i] : 0,
100                 bb = i < b.length() ? b.a[i] : 0;
101             na[i] = (aa + MOD - bb) % MOD;
102         }
103         return Poly(na);
104     }
105 };
```

```
106
107 Poly operator * (const int &c, const Poly &p) {
108     vector<int> na(p.length());
109     for (int i = 0; i < (int)na.size(); ++i) {
110         na[i] = (long long)c * p.a[i] % MOD;
111     }
112     return na;
113 }
114
115 vector<int> Berlekamp(vector<int> a) {
116     int n = a.size();
117     Poly s, b;
118     s.a.push_back(1), b.a.push_back(1);
119     for (int i = 1, j = 0, ld = a[0]; i < n; ++i) {
120         int d = s.calc(a, i);
121         if (d) {
122             if ((s.length() - 1) * 2 <= i) {
123                 Poly ob = b;
124                 b = s;
125                 s = s - (long long)d * inverse(ld) % MOD * ob.move(i - j);
126                 j = i;
127                 ld = d;
128             } else {
129                 s = s - (long long)d * inverse(ld) % MOD * b.move(i - j);
130             }
131         }
132     }
133     return s.a;
134 }
135
136 // Calculating kth term of linear recurrence sequence
137 // Modified for application
138
139 struct LinearRec {
140
141     int n;
142     vector<Vector> first;
143     vector<int> trans;
144     vector<vector<int> > bin;
145
146     vector<int> add(vector<int> &a, vector<int> &b) {
147         vector<int> result(n * 2 + 1, 0);
148         //You can apply constant optimization here to get a ~10x speedup
149         for (int i = 0; i <= n; ++i) {
150             for (int j = 0; j <= n; ++j) {
151                 if ((result[i + j] += (long long)a[i] * b[j] % MOD) >= MOD) {
152                     result[i + j] -= MOD;
153                 }
154             }
155         }
156         for (int i = 2 * n; i > n; --i) {
157             for (int j = 0; j < n; ++j) {
158                 if ((result[i - 1 - j] += (long long)result[i] * trans[j] % MOD) >= MOD
    ) {
159                     result[i - 1 - j] -= MOD;
160                 }
161             }
162             result[i] = 0;
163         }
```

```
164         result.erase(result.begin() + n + 1, result.end());
165         return result;
166     }
167 
168     LinearRec(vector<Vector> &first, vector<int> &trans):first(first), trans(trans) {
169         n = first.size();
170         vector<int> a(n + 1, 0);
171         a[1] = 1;
172         bin.push_back(a);
173         for (int i = 1; i < LOG; ++i) {
174             bin.push_back(add(bin[i - 1], bin[i - 1]));
175         }
176     }
177 
178     Vector calc(long long k) {
179         vector<int> a(n + 1, 0);
180         a[0] = 1;
181         for (int i = 0; i < LOG; ++i) {
182             if (k >> i & 1) {
183                 a = add(a, bin[i]);
184             }
185         }
186         Vector ret;
187         for (int i = 0; i < n; ++i) {
188             ret = ret + a[i + 1] * first[i];
189         }
190         return ret;
191     }
192 };
193 
194 Vector solve(Matrix &A, long long k, Vector &b) {
195     vector<Vector> vs;
196     vs.push_back(A * b);
197     for (int i = 1; i < m * 2; ++i) {
198         vs.push_back(A * vs.back());
199     }
200     if (k == 0) {
201         return b;
202     } else if (k <= m * 2) {
203         return vs[k - 1];
204     }
205     Vector x;
206     for (int i = 0; i < m; ++i) {
207         x[i] = rand() % MOD;
208     }
209     vector<int> a(m * 2);
210     for (int i = 0; i < m * 2; ++i) {
211         a[i] =  vs[i] * x;
212     }
213     vector<int> s = Berlekamp(a);
214     s.erase(s.begin());
215     for (int i = 0; i < s.size(); ++i) {
216         s[i] = (MOD - s[i]) % MOD;
217     }
218     vector<Vector> vf(vs.begin(), vs.begin() + s.size());
219     LinearRec f(vf, s);
220     return f.calc(k);
221 }
222 
```

```
223 //tested on CF 222E - Decoding Genome
224 int n;
225
226 long long k;
227
228 int main() {
229     scanf("%lld %d %d", &k, &m, &n);
230     Matrix A;
231     for (int i = 0; i < m; ++i) {
232         for (int j = 0; j < m; ++j) {
233             A[i][j] = 1;
234         }
235     }
236     for (int i = 0; i < n; ++i) {
237         char s[3];
238         scanf("%s", s);
239         int t1 = 'a' <= s[0] && s[0] <= 'z' ? t1 = s[0] - 'a' : t1 = s[0] - 'A' + 26;
240         int t2 = 'a' <= s[1] && s[1] <= 'z' ? t2 = s[1] - 'a' : t2 = s[1] - 'A' + 26;
241         A[t1][t2] = 0;
242     }
243     Vector b;
244     for (int i = 0; i < m; ++i) {
245         b[i] = 1;
246     }
247     int ans = solve(A, k - 1, b) * b;
248     printf("%d\n", ans);
249     return 0;
250 }
```

1.13.5 MatrixDeterminant

```
1  const int MOD = 1000000007, N = 10005, M = N * 11;
2
3  const int BAR = 10;
4
5  struct Vector {
6      int n, a[N];
7
8      Vector(int n):n(n) {
9          memset(a, 0, sizeof(a));
10     }
11
12     int& operator[] (const int &i) { return a[i]; }
13     const int operator[] (const int &i) const { return a[i]; }
14
15     int operator * (const Vector &b) {
16         unsigned long long ret = 0;
17         for (int i = 0; i < n; ++i) {
18             for (int j = 0; j < BAR && i < n; ++j, ++i) {
19                 ret = ret + (unsigned long long)a[i] * b[i];
20             }
21             --i;
22             ret %= MOD;
23         }
24         return ret;
25     }
26
27 };
```

```
28
29  struct Matrix {
30      int n, m;
31      int x[M], y[M], a[M];
32
33      Matrix(int n):n(n) {
34          m = 0;
35          memset(a, 0, sizeof(a));
36      }
37
38      void reshuffle() {
39          vector<pair<int, pair<int, int> > > v(m);
40          for (int i = 0; i < m; ++i) {
41              v[i].first = x[i], v[i].second.first = y[i], v[i].second.second = a[i];
42          }
43          sort(v.begin(), v.end());
44          for (int i = 0; i < m; ++i) {
45              x[i] = v[i].first, y[i] = v[i].second.first, a[i] = v[i].second.second;
46          }
47      }
48
49      Vector operator * (const Vector &b) const {
50          Vector ret(n);
51          for (int i = 0; i < m; ++i) {
52              if ((ret[x[i]] += (unsigned long long)a[i] * b[y[i]] % MOD) >= MOD) {
53                  ret[x[i]] -= MOD;
54              }
55          }
56          return ret;
57      }
58  };
59
60  unsigned long long buf[N];
61
62  void mul(const Matrix &A, Vector &b) { //to save memory
63      int n = A.n;
64      memset(buf, 0, sizeof(unsigned long long) * n);
65
66      for (int i = 0; i < A.m; ++i) {
67          buf[A.x[i]] += (unsigned long long)A.a[i] * b[A.y[i]];
68          if (i % BAR == 0) {
69              buf[A.x[i]] %= MOD;
70          }
71      }
72
73      for (int i = 0; i < A.n; ++i) {
74          b[i] = buf[i] % MOD;
75      }
76  }
77
78  // Berlekamp-Massey Algorithm
79
80  int inverse(int a) {
81      return a == 1 ? 1 : (long long)(MOD - MOD / a) * inverse(MOD % a) % MOD;
82  }
83
84  vector<int> na;
85
86  struct Poly {
```

```
87      vector<int> a;
88
89      Poly() { a.clear(); }
90
91      Poly(vector<int> &a): a(a) {}
92
93      int length() const { return a.size(); }
94
95      Poly move(int d) {
96          na.resize(d + a.size());
97          for (int i = 0; i < d + a.size(); ++i) {
98              na[i] = i < d ? 0 : a[i - d];
99          }
100         return na;
101     }
102
103     int calc(vector<int> &d, int pos) {
104         unsigned long long ret = 0;
105         for (int i = 0; i < (int)a.size(); ++i) {
106             for (int j = 0; j < BAR && i < (int)a.size(); ++j, ++i) {
107                 ret = ret + (unsigned long long)d[pos - i] * a[i];
108             }
109             --i;
110             ret %= MOD;
111         }
112         return ret;
113     }
114
115     Poly operator - (const Poly &b) {
116         na.resize(max(this->length(), b.length()));
117         for (int i = 0; i < (int)na.size(); ++i) {
118             int aa = i < this->length() ? this->a[i] : 0,
119                 bb = i < b.length() ? b.a[i] : 0;
120             na[i] = aa >= bb ? aa - bb : aa + MOD - bb;
121         }
122         return Poly(na);
123     }
124 };
125
126 Poly operator * (const int &c, const Poly &p) {
127     na.resize(p.length());
128     for (int i = 0; i < (int)na.size(); ++i) {
129         na[i] = (long long)c * p.a[i] % MOD;
130     }
131     return na;
132 }
133
134 vector<int> Berlekamp(vector<int> a) {
135     int n = a.size();
136     Poly s, b;
137     s.a.push_back(1), b.a.push_back(1);
138     for (int i = 1, j = 0, ld = a[0]; i < n; ++i) {
139         int d = s.calc(a, i);
140         if (d) {
141             if ((s.length() - 1) * 2 <= i) {
142                 Poly ob = b;
143                 b = s;
144                 s = s - (long long)d * inverse(ld) % MOD * ob.move(i - j);
145                 j = i;
```

```
146                 ld = d;
147             } else {
148                 s = s - (long long)d * inverse(ld) % MOD * b.move(i - j);
149             }
150         }
151     }
152     return s.a;
153 }
154
155 Vector getRandomVector(int n) {
156     Vector ret(n);
157     for (int i = 0; i < n; ++i) {
158         ret[i] = rand() % MOD;
159     }
160     return ret;
161 }
162
163 int solve(Matrix &A) {
164     Vector d = getRandomVector(A.n), x = getRandomVector(A.n), y = getRandomVector(A.n)
    ;
165     for (int i = 0; i < A.m; ++i) {
166         A.a[i] = (long long)A.a[i] * d[A.x[i]] % MOD;
167     }
168     vector<int> a;
169     for (int i = 0; i < A.n * 2 + 1; ++i) {
170         mul(A, x); //x = A * x;
171         a.push_back(x * y);
172     }
173     vector<int> s = Berlekamp(a);
174     int ret = s.back();
175     if (A.n & 1) {
176         ret = (MOD - ret) % MOD;
177     }
178     for (int i = 0; i < A.n; ++i) {
179         ret = (long long)ret * inverse(d[i]) % MOD;
180     }
181     return ret;
182 }
183
184 //tested on CF 348F - Little Artem and Graph
185
186 int n, k;
187
188 void initMatrix(Matrix &A) {
189     A.m = n - 1;
190     for (int i = 0; i < n - 1; ++i) {
191         A.x[i] = A.y[i] = i;
192         A.a[i] = 0;
193     }
194 }
195
196 void addEdge(Matrix &A, int u, int v) {
197     if (u < A.n && v < A.n) {
198         A.x[A.m] = u, A.y[A.m] = v, A.a[A.m] = MOD - 1, ++A.m;
199         A.x[A.m] = v, A.y[A.m] = u, A.a[A.m] = MOD - 1, ++A.m;
200     }
201     if (u < A.n) {
202         ++A.a[u];
203     }
```

```
    if (v < A.n) {
        ++A.a[v];
    }
}

int main() {
    scanf("%d%d", &n, &k);
    Matrix A(n - 1);
    initMatrix(A);
    for (int i = 0; i < k; ++i) {
        for (int j = i + 1; j < k; ++j) {
            addEdge(A, i, j);
        }
    }
    for (int i = k; i < n; ++i) {
        int u = i, v;
        for (int j = 0; j < k; ++j) {
            scanf("%d", &v);
            --v;
            addEdge(A, u, v);
        }
    }
    A.reshuffle();
    int ans = solve(A);
    printf("%d\n", ans);
    return 0;
}
```

1.14 Cal tree

1.14.1 Cal tree

```
/// ¸□¶¨□¸□□µ□□±□□µ□□£¨□□□□□□ · °□□¬£©□□□□□□□³□□□□□□¡£}¸□□□³□□□□±□□□□□□□õı□□□□
    Ų□»£¬»»¾仰□£¬□□□□□□□£
struct edge{
    int u, v, w, x;
    inline bool operator< (const edge &rhs) const{
        return x < rhs.x;
    }
}e[100005];
struct count{
    int l, r, use;
}g[100005];
int n, m, fa[50005], siz[50005];

int getfa(int x){
    return fa[x] == x ? x : getfa(fa[x]);
}

void link(int u, int v){
    if(siz[u] > siz[v]) fa[v] = u, siz[u] += siz[v];
    else fa[u] = v, siz[v] += siz[u];
}

bool Kruskal(){
    int cnt = 0, u, v;
    for(int i = 1; i <= m; ++i){
        u = getfa(e[i].u), v = getfa(e[i].v);
```

```
26          if(u != v){
27              link(u, v);
28              ++g[e[i].w].use;
29              if(++cnt == n - 1) return true;
30          }
31      }
32      return false;
33  }
34
35  int DFS(int w, int i, int k){
36      if(k == g[w].use) return 1;
37      if(i > g[w].r) return 0;
38      int ans = 0, u = getfa(e[i].u), v = getfa(e[i].v);
39      if(u != v){
40          link(u, v);
41          ans = DFS(w, i + 1, k + 1);
42          fa[u] = u, fa[v] = v;
43      }
44      return ans + DFS(w, i + 1, k);
45  }
46
47  int main(){
48      int u, v, w, ans;
49      cin >> n >> m;
50      for(int i = 1; i <= n; ++i)
51          fa[i] = i, siz[i] = 1;
52      for(int i = 1; i <= m; ++i){
53          cin >> u >> v >> w;
54         {u, v, 0, w};
55      }
56      sort(e + 1, e + m + 1);
57      w = 0;
58      for(int i = 1; i <= m; ++i)
59          if(e[i].x == e[i - 1].x) e[i].w = w;
60          else{
61              g[w].r = i - 1;
62              e[i].w = ++w;
63              g[w].l = i;
64          }
65      g[w].r = m;
66      ans = Kruskal();
67      for(int i = 1; i <= n; ++i)
68          fa[i] = i, siz[i] = 1;
69      for(int i = 1; i <= w; ++i){
70          ans = ans * DFS(i, g[i].l, 0) % 1000003;
71          for(int j = g[i].l; j <= g[i].r; ++j){
72              u = getfa(e[j].u), v = getfa(e[j].v);
73              if(u != v) link(u, v);
74          }
75      }
76      cout << ans << endl;
77      return 0;
78  }
```

1.14.2 Cal Mst

```
1  /// MST □£¬□□□½°¸µ□□□□□□□□
2  /// ans=sum(ÿ□□□□□*□□³□□□□□□□□□□³□□□□□□□□□□)
```

```
typedef long double ld;
const int N=10010,M=200010,K=70;
const ld eps=1e-9;
int n,m,i,j,k,f[N],g[N],cv,ce,POS,v[N],id[N],pool[K][K];ld ans,a[K][K];
struct E{int x,y,d,v;E(){}E(int _x,int _y,int _d){x=_x,y=_y,d=_d;}}e[M],w[K];
inline bool cmp(const E&a,const E&b){return a.d<b.d;}
inline bool cmpv(const E&a,const E&b){return a.v<b.v;}
int F(int x){return f[x]==x?x:f[x]=F(f[x]);}
int G(int x){return g[x]==x?x:g[x]=G(g[x]);}
inline int get(int x){
  if(v[x]<POS)v[x]=POS,id[x]=++cv;
  return id[x];
}
inline void add(int x,int y,int z){
  x=F(x),y=F(y);
  if(x==y)return;
  w[++ce]=E(get(x),get(y),z);
}
inline ld det(int n){
  ld ans=1;
  int i,j,k;
  for(i=1;i<=n;i++){
    for(j=i+1;j<=n;j++)if(fabs(a[j][i])>eps){
      ld t=a[i][i]/a[j][i];
      for(k=i;k<=n;k++)a[i][k]-=t*a[j][k];
      for(k=i;k<=n;k++)swap(a[i][k],a[j][k]);
      ans*=-1;
    }
    ans*=a[i][i];
    if(fabs(ans)<eps)return 0;
  }
  return ans;
}
inline ld cal(int l,int r,int o){
  int i,j,x,y;
  for(i=0;i<n;i++)for(j=0;j<n;j++)a[i][j]=0;
  for(i=l;i<r;i++)if(i!=o){
    x=id[w[i].x]-1,y=id[w[i].y]-1;
    a[x][x]++,a[y][y]++,a[x][y]--,a[y][x]--;
  }
  return det(n-1);
}
inline void work(){
  int i,j,k,x;
  for(i=1;i<=cv;i++)g[i]=i,pool[i][0]=0;
  for(i=1;i<=ce;i++)if(G(w[i].x)!=G(w[i].y))g[g[w[i].x]]=g[w[i].y];
  for(i=1;i<=cv;i++)pool[G(i)][++pool[G(i)][0]]=i;
  for(i=1;i<=ce;i++)w[i].v=G(w[i].x);
  sort(w+1,w+ce+1,cmpv);
  for(i=1;i<=ce;i=j){
    for(j=i;j<=ce&&w[i].v==w[j].v;j++);
    n=pool[x=w[i].v][0];
    for(k=1;k<=n;k++)id[pool[x][k]]=k;
    ld all=cal(i,j,0);
    for(k=i;k<j;k++)ans+=(all-cal(i,j,k))/all*w[k].d;
  }
}
int main(){
  scanf("%d%d",&n,&m);
```

```
  for(i=1;i<=m;i++)scanf("%d%d%d%d",&e[i].x,&e[i].y,&e[i].d,&e[i].v);
  sort(e+1,e+m+1,cmp);
  for(i=1;i<=n;i++)f[i]=i;
  for(i=1;i<=m;i=j){
    POS++,cv=ce=0;
    for(j=i;j<=m&&e[i].d==e[j].d;j++)add(e[j].x,e[j].y,e[j].v);
    work();
    for(k=i;k<j;k++)if(F(e[k].x)!=F(e[k].y))f[f[e[k].x]]=f[e[k].y];
  }
  printf("%.5f",(double)ans);
}
```

1.15 Adaptive Simpson

1.15.1 Adaptive Simpson

```
// ---
// $\int _a^b f(x)dx  \approx \frac{b-a}{6}[f(a)+4f(\frac{a+b}{2})+f(b)]$\\
// $|S(a, c) + S(c, b) - S(a, b)| / 15 < \epsilon$
// ---
double F(double x) {}
double simpson(double a, double b)
{ // □□μ□Simpson·¨
    double c = a + (b - a) / 2;
    return (F(a) + 4 * F(c) + F(b)) * (b - a) / 6;
}
double asr(double a, double b, double eps, double A)
{ //□□□□□Simpson¹«□£¨μ□□□³□©¡£□□□□¸□□□¼□[a,b]□□□□□μ□SimpsonA□
    double c = a + (b - a) / 2;
    double L = simpson(a, c), R = simpson(c, b);
    if (fabs(L + R - A) <= 15 * eps) return L + R + (L + R - A) / 15.0;
    return asr(a, c, eps / 2, L) + asr(c, b, eps / 2, R);
}
double asr(double a, double b, double eps) { return asr(a, b, eps, simpson(a, b)); }
```

1.16 LGLL

1.16.1 LGLL

```
/**
给定 n 个起点， n 个终点(一个终点对应一个起点)，解决 n □ n 条路径不相交的方案数
一个 n 阶的行列式，行代表起点，列代表终点，(i, j) 表示第 i 个起点到第 j 个终点的方案数，最后行列式的
    值就是n条不相交路径的方案数。
*/
```

1.17 Others

约瑟夫问题
n 个人围成一圈，从第一个开始报数，第 m 个将被杀掉

```
int josephus(int n, int m)
{
    int r = 0;
    for (int k = 1; k <= n; ++k) r = (r + m) % k;
    return r + 1;
}
```

n^n 最左边一位数

```
int leftmost(int n)
{
    double m = n * log10((double)n);
    double g = m - (ll)m;
    return (int)pow(10.0, g);
}
```

$n!$ 位数

```
int count(ll n)
{
    if (n == 1) return 1;
    return (int)ceil(0.5 * log10(2 * M_PI * n) + n * log10(n) - n * log10(M_E));
}
```

1.18 Formula

1. 约数定理：若 $n = \prod_{i=1}^{k} p_i^{a_i}$，则
   (a) 约数个数 $f(n) = \prod_{i=1}^{k}(a_i + 1)$
   (b) 约数和 $g(n) = \prod_{i=1}^{k}(\sum_{j=0}^{a_i} p_i^j)$
2. 小于 n 且互素的数之和为 $n\varphi(n)/2$
3. 若 $\gcd(n, i) = 1$，则 $\gcd(n, n-i) = 1 (1 \le i \le n)$
4. 错排公式：$D(n) = (n-1)(D(n-2) + D(n-1)) = \sum_{i=2}^{n} \frac{(-1)^k n!}{k!} = [\frac{n!}{e} + 0.5]$
5. 威尔逊定理：$p\ is\ prime\ \Rightarrow (p-1)! \equiv -1 \pmod p$
6. 欧拉定理：$\gcd(a, n) = 1 \Rightarrow a^{\varphi(n)} \equiv 1 \pmod n$
7. 欧拉定理推广：$\gcd(n, p) = 1 \Rightarrow a^n \equiv a^{n\%\varphi(p)} \pmod p$
8. 模的幂公式：$a^n \pmod m = \begin{cases} a^n \mod m & n < \varphi(m) \\ a^{n\%\varphi(m)+\varphi(m)} \mod m & n \ge \varphi(m) \end{cases}$
9. 素数定理：对于不大于 n 的素数个数 $\pi(n)$，$\lim_{n\to\infty} \pi(n) = \frac{n}{\ln n}$
10. 位数公式：正整数 x 的位数 $N = \log_{10}(n) + 1$
11. 斯特灵公式 $n! \approx \sqrt{2\pi n}(\frac{n}{e})^n$
12. 设 $a > 1, m, n > 0$，则 $\gcd(a^m - 1, a^n - 1) = a^{\gcd(m,n)} - 1$
13. 设 $a > b, \gcd(a, b) = 1$，则 $\gcd(a^m - b^m, a^n - b^n) = a^{\gcd(m,n)} - b^{\gcd(m,n)}$

$$G = \gcd(C_n^1, C_n^2, ..., C_n^{n-1}) = \begin{cases} n, & n \text{ is prime} \\ 1, & n \text{ has multy prime factors} \\ p, & n \text{ has single prime factor } p \end{cases}$$

$\gcd(Fib(m), Fib(n)) = Fib(\gcd(m, n))$

14. 若 $\gcd(m, n) = 1$，则：
   (a) 最大不能组合的数为 $m * n - m - n$
   (b) 不能组合数个数 $N = \frac{(m-1)(n-1)}{2}$
15. $(n+1)lcm(C_n^0, C_n^1, ..., C_n^{n-1}, C_n^n) = lcm(1, 2, ..., n+1)$
16. 若 p 为素数，则 $(x + y + ... + w)^p \equiv x^p + y^p + ... + w^p \pmod p$
17. 卡特兰数：1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012
    $h(0) = h(1) = 1, h(n) = \frac{(4n-2)h(n-1)}{n+1} = \frac{C_{2n}^n}{n+1} = C_{2n}^n - C_{2n}^{n-1}$

18. 伯努利数：$B_n = -\frac{1}{n+1}\sum_{i=0}^{n-1} C_{n+1}^{i} B_i$

$$\sum_{i=1}^{n} i^k = \frac{1}{k+1}\sum_{i=1}^{k+1} C_{k+1}^{i} B_{k+1-i}(n+1)^i$$

19. 二项式反演：

$$f_n = \sum_{i=0}^{n}(-1)^i \binom{n}{i} g_i \Leftrightarrow g_n = \sum_{i=0}^{n}(-1)^i \binom{n}{i} f_i$$

$$f_n = \sum_{i=0}^{n} \binom{n}{i} g_i \Leftrightarrow g_n = \sum_{i=0}^{n}(-1)^{n-i} \binom{n}{i} f_i$$

20. FFT 常用素数

$r\ 2^k + 1$	r	k	g
3	1	1	2
5	1	2	2
17	1	4	3
97	3	5	5
193	3	6	5
257	1	8	3
7681	15	9	17
12289	3	12	11
40961	5	13	3
65537	1	16	3
786433	3	18	10
5767169	11	19	3
7340033	7	20	3
23068673	11	21	3
104857601	25	22	3
167772161	5	25	3
469762049	7	26	3
998244353	119	23	3
1004535809	479	21	3
2013265921	15	27	31
2281701377	17	27	3
3221225473	3	30	5
75161927681	35	31	3
77309411329	9	33	7
206158430209	3	36	22
2061584302081	15	37	7
2748779069441	5	39	3
6597069766657	3	41	5
39582418599937	9	42	5
79164837199873	9	43	5
263882790666241	15	44	7
1231453023109121	35	45	3
1337006139375617	19	46	3
3799912185593857	27	47	5
4222124650659841	15	48	19
7881299347898369	7	50	6
31525197391593473	7	52	3
180143985094819841	5	55	6
1945555039024054273	27	56	5
4179340454199820289	29	57	3

2 String Processing

2.1 KMP

```
int next[N];
char S[N], T[N];
int slen, tlen;

void getNext()
{
    int j, k;
    j = 0; k = -1; next[0] = -1;
    while(j < tlen){
        if(k == -1 || T[j] == T[k])
            next[++j] = ++k;//表示T[j-1]和T[k-1]相匹配，当j处失配时，直接用next[j]处来匹配当前失配处
        else
            k = next[k];
    }
}

/*
返回模式串T在主串S中首次出现的位置
返回的位置是从0开始的。
*/

int KMP_Index()
{
    int i = 0, j = 0;
    getNext();

    while(i < slen && j < tlen){
        if(j == -1 || S[i] == T[j]){
            i++; j++;
        }
        else
            j = next[j];
    }
    if(j == tlen)
        return i - tlen;
    else
        return -1;
}

/*
返回模式串在主串S中出现的次数
*/

int KMP_Count()
{
    int ans = 0;
    int i, j = 0;

    if(slen == 1 && tlen == 1){
        if(S[0] == T[0])
            return 1;
        else
            return 0;
    }
```

```
55
56      getNext();
57      for(i = 0; i < slen; i++){
58          while(j > 0 && S[i] != T[j])
59              j = next[j];
60          if(S[i] == T[j])
61              j++;
62          if(j == tlen){
63              ans++;
64              j = next[j];
65          }
66      }
67
68      return ans;
69  }
70
71  int main()
72  {
73      int TT;
74      int i, cc;
75
76      data>>TT;
77      while(TT--){
78          data>>S>>T;
79          slen = strlen(S);
80          tlen = strlen(T);
81          getNext();
82          for(int i=0;i<=tlen;i++)
83              cout<<next[i]<<" ";
84          cout<<endl;
85          cout<<"模式串T在主串S中首次出现的位置是: "<<KMP_Index()<<endl;
86          cout<<"模式串T在主串S中出现的次数为: "<<KMP_Count()<<endl;
87      }
88      return 0;
89  }
90  /*
91  3
92  ababcabcacbabcac
93  abcac
94  */
```

2.2 ExKMP

```
1   /// 求 T 与 S 的每一个后缀  的最长公共前缀
2   const int MAX=100010; //字符串长度最大值
3   int Next[MAX],extend[MAX];
4
5   //预处理计算Next数组
6   void getNext(char str[])
7   {
8       int i=0,j,po,len=strlen(str);
9       next[0]=len; //初始化next[0]
10      while(str[i]==str[i+1] && i+1<len) i++; next[1]=i; //计算next[1]
11      po=1; //初始化po的位置
12      for(i=2;i<len;i++)
13      {
14          if(next[i-po]+i < next[po]+po) //第一种情况，可以直接得到next[i]的值
15              next[i]=next[i-po];
```

```
        else //第二种情况，要继续匹配才能得到next[i]的值
        {
            j = next[po]+po-i;
            if(j<0) j=0; //如果i>po+next[po],则要从头开始匹配
            while(i+j<len && str[j]==str[j+i]) j++; next[i]=j;
            po=i; //更新po的位置
        }
    }
}

//计算extend数组
void EXKMP(char s1[],char s2[])
{
    int i=0,j,po,len=strlen(s1),l2=strlen(s2);
    getNext(s2); //计算子串的next数组
    while(s1[i]==s2[i] && i<l2 && i<len) i++; extend[0]=i;
    po=0; //初始化po的位置
    for(i=1;i<len;i++)
    {
        if(next[i-po]+i < extend[po]+po) //第一种情况，直接可以得到extend[i]的值
            ex[i]=next[i-po];
        else //第二种情况，要继续匹配才能得到extend[i]的值
        {
            j = extend[po]+po-i;
            if(j<0) j=0; //如果i>extend[po]+po则要从头开始匹配
            while(i+j<len && j<l2 && s1[j+i]==s2[j]) j++; extend[i]=j;
            po=i; //更新po的位置
        }
    }
}
```

2.3 Manacher

```
string Manacher(string s) {
    // Insert '#'
    string t = "$#";
    for (int i = 0; i < s.size(); ++i) {
        t += s[i];
        t += "#";
    }
    // Process t
    vector<int> p(t.size(), 0);
    int mx = 0, id = 0, resLen = 0, resCenter = 0;
    for (int i = 1; i < t.size(); ++i) {
        p[i] = mx > i ? min(p[2 * id - i], mx - i) : 1;
        while (t[i + p[i]] == t[i - p[i]]) ++p[i];
        if (mx < i + p[i]) {
            mx = i + p[i];
            id = i;
        }
        if (resLen < p[i]) {
            resLen = p[i];
            resCenter = i;
        }
    }
    return s.substr((resCenter - resLen) / 2, resLen - 1);
}

```

```
int main() {
    string s1 = "12212";
    cout << Manacher(s1) << endl;
    string s2 = "122122";
    cout << Manacher(s2) << endl;
    string s = "waabwswfd";
    cout << Manacher(s) << endl;
}
```

2.4 StaticACA

```
#include<bits/stdc++.h>
using namespace std;
const int maxn = 300010;
const int dp_maxn = 1010;
const int INF = 0x3f3f3f3f;
const int type_N = 26;

int T, N, dp[dp_maxn][dp_maxn];
char in[60], text[1000010];
struct Trie
{
    int nxt[maxn][type_N],
        fail[maxn],
        cnt[maxn],
        size, root;
    bool finish[maxn];

    int newNode(){
        for(int i = 0; i < type_N; ++ i)
            nxt[size][i] = 0;
        fail[size] = 0;
        cnt[size] = 0;
        finish[size] = false;
        return size ++;
    }
    void init(){
        size = 1;
        root = newNode();
    }
    int idx(char ch){
        /*if(ch == 'A')return 0;
        if(ch == 'G')return 1;
        if(ch == 'C')return 2;
        return 3;*/
        return ch - 'a';
    }
    void insert(char *s){
        int now = root;
        while(*s){
            int index = idx(*s);
            if(!nxt[now][index])
                nxt[now][index] = newNode();
            now = nxt[now][index];
            ++ s;
        }
        ++ cnt[now];
        finish[now] = true;
```

```
48      }
49      void build(){
50          queue<int> que;
51          fail[root] = root;
52          for(int i = 0; i < type_N; ++ i){
53              if(nxt[root][i]){
54                  que.push(nxt[root][i]);
55                  fail[nxt[root][i]] = root;
56              }
57              else nxt[root][i] = root;
58          }
59          while(que.size()){
60              int now = que.front();
61              que.pop();
62              finish[now] = finish[now] || finish[fail[now]];
63              for(int i = 0; i < type_N; ++ i){
64                  if(nxt[now][i]){
65                      que.push(nxt[now][i]);
66                      fail[nxt[now][i]] = nxt[fail[now]][i];
67                  }
68                  else nxt[now][i] = nxt[fail[now]][i];
69              }
70          }
71      }
72      int Match(char *s){
73          int now = root, ans = 0;
74          while(*s){
75              int index = idx(*s);
76              while(now != root && nxt[now][index] == root){
77                  //printf("now = %d\n", now);
78                  now = fail[now];
79              }
80              now = nxt[now][index];
81              int match = now;
82              while(match != root){
83                  //printf("match = %d, now = %d\n", match, now);
84                  ans += cnt[match];
85                  cnt[match] = 0;///查找匹配单词个数则加上这句；查找模板串出现次数则删掉这句
86                  match = fail[match];
87              }
88               ++ s;
89          }
90          return ans;
91      }
92      int meta(char *s){///type_N = 4,
93          int len = strlen(s);
94          memset(dp, 0x3f, sizeof(dp));
95          dp[0][root] = 0;
96          for(int i = 0; i < len; ++ i){
97              for(int j = 0; j < size; ++ j){
98                  if(dp[i][j] < INF){
99                      for(int k = 0; k < 4; ++ k){
100                         int nstatus = nxt[j][k];
101                         if(finish[nstatus])continue;
102                         dp[i+1][nstatus] = min(dp[i+1][nstatus], dp[i][j] + (k != idx(s
    [i])));
103                     }
104                 }
105             }
```

```
106         }
107         int ans = INF;
108         for(int i = 0; i < size; ++ i)
109             ans = min(ans, dp[len][i]);
110         return ans == INF ? -1 : ans;
111     }
112 }ac;
113 int main(){
114     scanf("%d", &T);
115     while(T--){
116         ac.init();
117         scanf("%d", &N);
118         while(N--){
119             scanf("%s", in);
120             ac.insert(in);
121         }
122         ac.build();
123         scanf("%s", text);
124         printf("%d\n", ac.Match(text));
125     }
126     return 0;
127 }
```

2.5 DynamicACA

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  const int type_N = 26;///
4  const int maxn = 100010;///length of text
5
6  int idx(char ch){///Đ¡Đ´×ÖÄ¸idx()
7      return ch - 'a';
8  }
9
10 struct Trie{
11     Trie *nxt[type_N], *fail;
12     int finish;
13
14     void init(){
15         for(int i = 0; i < type_N; ++ i)
16             nxt[i] = NULL;
17         fail = NULL;
18         finish = 0;
19     }
20 }root;
21 int n;
22 char in[30], text[maxn];
23
24 void Insert(char *s){
25     int index;
26     Trie *now = &root, *tmp;
27     while(*s){
28         index = idx(*s);
29         if(!now->nxt[index]){
30             tmp = (Trie*) malloc(sizeof(Trie));
31             tmp->init();
32             now->nxt[index] = tmp;
33         }
```

```
34          now = now->nxt[index];
35          ++ s;
36      }
37      now->finish ++;
38  }
39
40  void Get_fail(){
41      queue<Trie*> Q;
42      Q.push(&root);
43      Trie *now, *anc;
44      while(!Q.empty()){
45          now = Q.front();
46          Q.pop();
47          for(int i = 0; i < type_N; ++ i){
48              if(now->nxt[i]){
49                  Q.push(now->nxt[i]);
50                  if(now == &root){
51                      now->nxt[i]->fail = &root;
52                      continue;
53                  }
54                  anc = now->fail;
55                  while(anc){
56                      if(anc->nxt[i]){
57                          now->nxt[i]->fail = anc->nxt[i];
58                          break;
59                      }
60                      anc = anc->fail;
61                  }
62                  if(!anc)
63                      now->nxt[i]->fail = &root;
64              }
65          }
66      }
67  }
68
69  int AC_Automaton(char *s){///
70      Trie *now = &root, *match;
71      int index, ret = 0;
72      while(*s){
73          index = idx(*s);
74          while(now != &root && !now->nxt[index])
75              now = now->fail;
76          now = now->nxt[index];
77          match = now;
78          if(!now)now = &root;
79          while(match){
80              ret += match->finish;
81              match = match->fail;
82          }
83          ++ s;
84      }
85      return ret;
86  }
87
88  void Del(Trie *now = &root){
89      for(int i = 0; i < type_N; ++ i){
90          if(now->nxt[i]){
91              Del(now->nxt[i]);
92              free(now->nxt[i]);
```

```
        }
    }
}

void init(){
    Del();
    root.init();
}

int main(){
    while(~scanf("%d", &n)){
        init();
        while(n--){
            scanf("%s", in);
            Insert(in);
        }
        Get_fail();
        while(~scanf("%s", text))///"ctrl+Z" to quit
            printf("%d\n", AC_Automaton(text));
    }
    return 0;
}
```

2.6 SAM

```
#define maxn 200010          ///两倍字符串长度
#define sigma 26             ///字符集大小
struct SuffixAutomaton{
    int root;                ///root = 0;
    int size, last;
    int go[maxn][sigma];     ///转移
    int maxlen[maxn];        ///节点i代表的子串的最长长度
    int par[maxn];           ///parent树中节点i的父节点
    int cnt[maxn];           ///节点i的right集合大小
    int who[maxn];           ///who[i]是按照maxlen从小到大排序后的第i个节点下标
    int a[maxn];             ///辅助拓扑排序的数组(计数排序)
    SuffixAutomaton():
        size(0){}

    int new_node(int len){
        memset(go[size], -1, sizeof(go[size]));
        maxlen[size] = len;
        par[size] = -1;
        return size ++;
    }
    int idx(char ch){
        return ch - 'a';
    }
    void extend(int ch){
        //printf("extending: %c\n", 'a' + ch);
        int p, q, np, nq;
        p = last;
        np = new_node(maxlen[p] + 1);
        while(p != -1 && go[p][ch] == -1){///在所有的last的祖先中，为所有没有标号为x的边的节点添加一条边指向节点np=ST(Tx)
            go[p][ch] = np;
            p = par[p];
        }
```

```
33          if(p == -1){///如果last的所有祖先没有标号为x的边，则状态np=trans(Tx)的parent为root
34              //printf("par[%d] = root\n", np);
35              par[np] = root;
36          }
37          else {///找到第一个有标号为x的边的状态p，令q=trans(p,x)
38              q = go[p][ch];
39              if(maxlen[p] + 1 == maxlen[q]){///如果maxlen[p] + 1 == maxlen[q]则直接令q的
    parent为np
40                  par[np] = q;
41              }
42              else {///否则新建节点nq，令maxlen[nq] = maxlen[p] + 1
43                  nq = new_node(maxlen[p] + 1);
44                  memcpy(go[nq], go[q], sizeof(go[q]));///nq的状态转移函数与q的完全相同
45                  par[nq] = par[q];
46                  par[q] = nq;        ///将parent树上的关系由q->par[q] 修改为 q->nq->par[q]
47                  par[np] = nq;       ///np=ST(Tx)的parent也是nq
48                  while(p != -1 && go[p][ch] == q){
49                      go[p][ch] = nq;
50                      p = par[p];
51                  }
52              }
53          }
54          last = np;
55          cnt[np] = 1;
56          //printf("cnt[%d] = %d\n", np, cnt[np]);
57      }
58      void count(){///拓扑排序并计数每个节点表示的子串出现次数
59          memset(a, 0, sizeof(a));
60          for(int i = 0; i < size; ++ i)a[maxlen[i]] ++;
61          for(int i = 1; i < size; ++ i)a[i] += a[i-1];
62          for(int i = size - 1; i >= 0; -- i)who[--a[maxlen[i]]] = i; ///计数排序
63          for(int i = size - 1; i >= 0; -- i){
64              if(par[who[i]] != -1){
65                  cnt[par[who[i]]] += cnt[who[i]];
66              }
67          }
68      }
69      void init(char *s){
70          size = 0;
71          last = root = new_node(0);
72          memset(cnt, 0, sizeof(cnt));
73          int len = strlen(s);
74          for(int i = 0; i < len; ++ i){
75              extend(idx(s[i]));
76          }
77          count();
78      }
79  }sam;
80  #undef sigma
81  #undef maxn
```

2.7 PAM

```
1   #define maxn 3000010
2   #define sigma 26
3   struct Palindromic_Tree{
4       int go[maxn][sigma], fail[maxn];
5       int depth[maxn], cnt[maxn];
```

```
6       int size, last, len;
7       char s[maxn];
8       int idx(char ch){
9           return ch - 'a';
10      }
11      int new_node(int length){
12          memset(go[size], 0, sizeof(go[size]));
13          fail[size] = 0;
14          depth[size] = length;
15          cnt[size] = 0;
16          return size++;
17      }
18      void Extend(int ch, int pos){
19          int p = last;
20          while (s[pos - depth[p] - 1] != s[pos]){
21              p = fail[p];
22          }
23          if (!go[p][ch]){
24              int now = new_node(depth[p] + 2), fa = fail[p];
25              while (s[pos - depth[fa] - 1] != s[pos])
26                  fa = fail[fa];
27              fail[now] = go[fa][ch];
28              if (fail[now] == 0) fail[now] = 1;
29              go[p][ch] = now;
30          }
31          last = go[p][ch];
32          cnt[last]++;
33      }
34      void init(char *x){
35          len = strlen(x + 1);
36          for (int i = 1; i <= len; i++)
37              s[i] = x[i];
38          size = 0;
39          new_node(-1);
40          new_node(0);
41          fail[0] = fail[1] = 0;
42          depth[0] = -1;
43          last = 1;
44          for (int i = 1; i <= len; i++)
45              Extend(idx(s[i]), i);
46      }
47  }pam;
48  #undef sigma
49  #undef maxn
```

2.8 StringHash

```
1   typedef unsigned long long ull;
2   const ull Seed_Pool[] = {146527, 19260817};
3   const ull Mod_Pool[] = {1000000009, 998244353};
4   struct Hash
5   {
6       ull SEED, MOD;
7       vector<ull> p, h;
8       Hash() {}
9       Hash(const string& s, const int& seed_index, const int& mod_index)
10      {
11          SEED = Seed_Pool[seed_index];
```

```
        MOD = Mod_Pool[mod_index];
        int n = s.length();
        p.resize(n + 1), h.resize(n + 1);
        p[0] = 1;
        for (int i = 1; i <= n; i++) p[i] = p[i - 1] * SEED % MOD;
        for (int i = 1; i <= n; i++) h[i] = (h[i - 1] * SEED % MOD + s[i - 1]) % MOD;
    }
    ull get(int l, int r) { return (h[r] - h[l] * p[r - l] % MOD + MOD) % MOD; }
    ull substr(int l, int m) { return get(l, l + m); }
};
```

2.9 Manacher

```
#define maxn 110010
//int rad[maxn<<1];
int manacher(char *x){      //@ret (int)length of the longest palindrome in string X
    int ret = -1;
    int len = strlen(x + 1) * 2 + 1;
    char s[maxn<<1] = {0};
    for(int i = 1; i <= len; ++ i){
        if(i & 1){
            s[i] = nons;
        }
        else {
            s[i] = x[i/2];
        }
    }
    int pos = 0, maxr = 0;
    for(int i = 1; i <= len; ++ i){
        if(i < maxr){
            rad[i] = min(rad[pos-i+pos], maxr - i);
        }
        else {
            rad[i] = 1;
        }
        while(i - rad[i] > 0 && i + rad[i] <= len && s[i-rad[i]] == s[i+rad[i]]){
            rad[i] ++;
        }
        if(i + rad[i] - 1 > maxr){
            maxr = i + rad[i] - 1;
            pos = i;
        }
        ret = max(ret, rad[i] - 1);
    }
    return ret;
}
#undef maxn
```

2.10 DC3

```
const int maxn = int(3e6)+10;
const int N = maxn;

    #define F(x) ((x)/3+((x)%3==1?0:tb))
    #define G(x) ((x)<tb?(x)*3+1:((x)-tb)*3+2)
    int wa[maxn],wb[maxn],wv[maxn],ws[maxn];
    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]&&wv[a+1]<wv[b+1];}
    void sort(int *r,int *a,int *b,int n,int m)
    {
        int i;
        for(i=0;i<n;i++) wv[i]=r[a[i]];
        for(i=0;i<m;i++) ws[i]=0;
        for(i=0;i<n;i++) ws[wv[i]]++;
        for(i=1;i<m;i++) ws[i]+=ws[i-1];
        for(i=n-1;i>=0;i--) b[--ws[wv[i]]]=a[i];
        return;
    }
    void dc3(int *r,int *sa,int n,int m) //°□□□□DA □□□
    {
        int i,j,*rn=r+n,*san=sa+n,ta=0,tb=(n+1)/3,tbc=0,p;
        r[n]=r[n+1]=0;
        for(i=0;i<n;i++) if(i%3!=0) wa[tbc++]=i;
        sort(r+2,wa,wb,tbc,m);
        sort(r+1,wb,wa,tbc,m);
        sort(r,wa,wb,tbc,m);
        for(p=1,rn[F(wb[0])]=0,i=1;i<tbc;i++)
        rn[F(wb[i])]=c0(r,wb[i-1],wb[i])?p-1:p++;
        if(p<tbc) dc3(rn,san,tbc,p);
        else for(i=0;i<tbc;i++) san[rn[i]]=i;
        for(i=0;i<tbc;i++) if(san[i]<tb) wb[ta++]=san[i]*3;
        if(n%3==1) wb[ta++]=n-1;
        sort(r,wb,wa,ta,m);
        for(i=0;i<tbc;i++) wv[wb[i]=G(san[i])]=i;
        for(i=0,j=0,p=0;i<ta && j<tbc;p++)
        sa[p]=c12(wb[j]%3,r,wa[i],wb[j])?wa[i++]:wb[j++];
        for(;i<ta;p++) sa[p]=wa[i++];
        for(;j<tbc;p++) sa[p]=wb[j++];
        return;
    }
```

2.11 DA

```
const int N = int(2e5)+10;
int cmp(int *r,int a,int b,int l){
    return (r[a]==r[b]) && (r[a+l]==r[b+l]);
}
// □□□□□□□□□¹□□□□□□□□□□¹□□□,
// ±□□□□□□□□□½□□,□´¦□□□□°□,r[n]=0(□□□□□□□□□□µ□□□□
int wa[N],wb[N],ws[N],wv[N];
int rank[N],height[N];
void DA(int *r,int *sa,int n,int m){ //´□¦N±□□□□□□□N□¶□1£¬□□□¤□□□□□¸□□□£¬□□□□□□□□CMP□□½□
    int i,j,p,*x=wa,*y=wb,*t;
    for(i=0;i<m;i++) ws[i]=0;
    for(i=0;i<n;i++) ws[x[i]=r[i]]++;
    for(i=1;i<m;i++) ws[i]+=ws[i-1];
    for(i=n-1;i>=0;i--) sa[--ws[x[i]]]=i; //□´¦□□¶□□1
    for(j=1,p=1;p<n;j*=2,m=p) //□□□□□□□□¤¶□Jµ□SA£¬4□□2*Jµ□SA
    {
        for(p=0,i=n-j;i<n;i++) y[p++]=i; // □□□□□□□û□□□□□¹□□□
        for(i=0;i<n;i++) if(sa[i]>=j) y[p++]=sa[i]-j; //□□□ó¤¶□Jµģ¬°´µ□□□¹□□□□□□□
```

```
20          for(i=0;i<n;i++) wv[i]=x[y[i]];
21          for(i=0;i<m;i++) ws[i]=0;
22          for(i=0;i<n;i++) ws[wv[i]]++;
23          for(i=1;i<m;i++) ws[i]+=ws[i-1];
24          for(i=n-1;i>=0;i--) sa[--ws[wv[i]]]=y[i];  //»□□□□□□□□□
25          for(t=x,x=y,y=t,p=1,x[sa[0]]=0,i=1;i<n;i++)
26              x[sa[i]]=cmp(y,sa[i-1],sa[i],j)?p-1:p++;  //¸□□□□□´□□□□□x[],□□□□□¨□□p□□
27      }
28  }
29
30  void calheight(int *r,int *sa,int n){ // ´□¦N□□¾□¤¶□
31      int i,j,k=0;        // height[]µÍ□¨·¶□□ 1-N, □□□□0□□□□¾□□□□□□
32      for(i=1;i<=n;i++) rank[sa[i]]=i;  // ¸□¾□SA□□RANK
33      for(i=0;i<n; height[rank[i++]] = k ) // ¶¨□壺h[i] = height[ rank[i] ]
34      for(k?k--:0,j=sa[rank[i]-1]; r[i+k]==r[j+k]; k++); //¸□¾□ h[i] >= h[i-1]-1 4□Ż¯¾□□□
        height¹□³□
35  }
36
37  char str[N];
38  int sa[N];
39  int main(){
40      char str[N];
41      scanf("%s",str);
42      int n = strlen(str);
43      str[n]=0;
44
45      da(str,sa,n+1,128);  //□□□□□·□¦□n+1,□□□□□□□□□¸□□□□□□□□□□±□□□
46      calheight(str,sa,n);
47  }
```

2.12 SA-IS.CPP

3 Data Structure

3.1 Lca

3.1.1 LcaTarjan

```
/****/
//qrec[]: 记录询问 (sorted pair<int, int>[])
//tot: 总询问数 (即qrec[]的大小)
#define maxn 100010
bool vis[maxn];
//memset(vis, false, sizeof(vis));
void tarjan(int now = root){        ///dfs(root)
    //printf("now = %d\n", now);
    fa[now] = now;                  ///union_find
    vis[now] = true;
    for(int i = head[now]; i != -1; i = node[i].ne){
        int nxt = node[i].to;
        if(!vis[nxt]){
            tarjan(nxt);
            //unite(nxt, now);        ///fa[nxt] = now;
            fa[nxt] = now;          ///unite(nxt, now)
        }
    }
    int id = get_first_query(now);  ///找到询问为(now,*)的第一个询问的下标为id (lower_bound)
    while(id < tot && qrec[id].first == now){   ///遍历所有以x开头的询问, 记录答案
        int query = qrec[id].second;
        if(vis[query]){
            ans[id] = find(query);
            //int loctn = loc(make_pair(query, now));
            //ans[loctn] = ans[id];
            ////printf("lca(%d, %d) = %d\n", now, query, ans[id]);
        }
        id ++;
    }
}
#undef maxn
/****/
```

3.1.2 Rmqlca

```
// ---
// DFS+ST在线算法\\
// 时间复杂度$O(nlogn+q)$
// ---
const int maxn = "Edit";
vector<int> G[maxn], sp;
int dep[maxn], dfn[maxn];
PII dp[21][maxn << 1];
void init(int n)
{
    for (int i = 0; i < n; i++) G[i].clear();
    sp.clear();
}
void dfs(int u, int fa)
{
    dep[u] = dep[fa] + 1;
    dfn[u] = sp.size();
    sp.push_back(u);
```

```
    for (auto& v : G[u])
    {
        if (v == fa) continue;
        dfs(v, u);
        sp.push_back(u);
    }
}
void initrmq()
{
    int n = sp.size();
    for (int i = 0; i < n; i++) dp[0][i] = {dfn[sp[i]], sp[i]};
    for (int i = 1; (1 << i) <= n; i++)
        for (int j = 0; j + (1 << i) - 1 < n; j++)
            dp[i][j] = min(dp[i - 1][j], dp[i - 1][j + (1 << (i - 1))]);
}
int lca(int u, int v)
{
    int l = dfn[u], r = dfn[v];
    if (l > r) swap(l, r);
    int k = 31 - __builtin_clz(r - l + 1);
    return min(dp[k][l], dp[k][r - (1 << k) + 1]).second;
}
```

3.2 Segment Tree

3.2.1 Area Combination

```
// 矩形面积并
map<double, int> Hash;
map<int, double> rHash;
struct line
{
    double l, r, h;
    int val;
    line(double l = 0, double r = 0, double h = 0, int val = 0) : l(l), r(r), h(h), val
    (val) {}
    bool operator<(const line& A) const { return h < A.h; }
};
struct Node
{
    int cover;
    double len;
};
const int maxn = 1000;
Node seg[maxn << 2];
void build(int rt, int l, int r)
{
    seg[rt].cover = seg[rt].len = 0;
    if (l == r) return;
    int mid = l + r >> 1;
    build(lson, l, mid);
    build(rson, mid + 1, r);
}
void pushup(int rt, int l, int r)
{
    if (seg[rt].cover > 0)
        seg[rt].len = rHash[r + 1] - rHash[l]; // [l,r)
    else if (l == r)
        seg[rt].len = 0;
```

```
    else
        seg[rt].len = seg[lson].len + seg[rson].len;
}
void update(int rt, int l, int r, int L, int R, int val)
{
    if (L <= l && R >= r)
    {
        seg[rt].cover += val;
        pushup(rt, l, r);
        return;
    }
    int mid = l + r >> 1;
    if (mid >= L) update(lson, l, mid, L, R, val);
    if (mid + 1 <= R) update(rson, mid + 1, r, L, R, val);
    pushup(rt, l, r);
}
int main()
{
    int n, kase = 0;
    while (~scanf("%d", &n))
    {
        if (!n) break;
        double x1, x2, y1, y2;
        vector<line> a;
        set<double> xval;
        for (int i = 0; i < n; i++)
        {
            scanf("%lf%lf%lf%lf", &x1, &y1, &x2, &y2);
            a.emplace_back(x1, x2, y1, 1);
            a.emplace_back(x1, x2, y2, -1);
            xval.insert(x1);
            xval.insert(x2);
        }
        // 离散化
        Hash.clear(), rHash.clear();
        int cnt = 0;
        for (auto& v : xval)
        {
            Hash[v] = ++cnt;
            rHash[cnt] = v;
        }
        sort(a.begin(), a.end());
        build(1, 1, cnt);
        double ans = 0;
        for (int i = 0; i < a.size() - 1; i++)
        {
            update(1, 1, cnt, Hash[a[i].l], Hash[a[i].r] - 1,
                   a[i].val); //[l,r)
            ans += (a[i + 1].h - a[i].h) * seg[1].len;
        }
        printf("Test case #%d\n", ++kase);
        printf("Total explored area: %.2lf\n\n", ans);
    }
}
```

3.2.2 Area Intersection

```
// 矩形面积交
```

```
map<double, int> Hash;
map<int, double> rHash;
struct Lines
{
    double l, r, h;
    int val;
    bool operator<(const Lines& A) const { return h < A.h; }
};
struct Node
{
    int cnt;     //  覆盖次数
    double len1; // 覆盖次数大于0的长度
    double len2; // 覆盖次数大于1的长度
};
Node seg[maxn << 2];
void build(int rt, int l, int r)
{
    seg[rt].cnt = seg[rt].len1 = seg[rt].len2 = 0;
    if (l == r) return;
    int mid = l + r >> 1;
    build(lson, l, mid);
    build(rson, mid + 1, r);
}
inline void pushup(int rt, int l, int r)
{
    if (seg[rt].cnt > 1)
        seg[rt].len1 = seg[rt].len2 = rHash[r + 1] - rHash[l];
    else if (seg[rt].cnt == 1)
    {
        seg[rt].len1 = rHash[r + 1] - rHash[l];
        if (l == r)
            seg[rt].len2 = 0;
        else
            seg[rt].len2 = seg[lson].len1 + seg[rson].len1;
    }
    else
    {
        if (l == r)
            seg[rt].len1 = seg[rt].len2 = 0;
        else
        {
            seg[rt].len1 = seg[lson].len1 + seg[rson].len1;
            seg[rt].len2 = seg[lson].len2 + seg[rson].len2;
        }
    }
}
void update(int rt, int l, int r, int L, int R, int val)
{
    if (L <= l && R >= r)
    {
        seg[rt].cnt += val;
        pushup(rt, l, r);
        return;
    }
    int mid = l + r >> 1;
    if (L <= mid) update(lson, l, mid, L, R, val);
    if (R >= mid + 1) update(rson, mid + 1, r, L, R, val);
    pushup(rt, l, r);
}
```

```
int main()
{
    int T;
    scanf("%d", &T);
    while (T--)
    {
        int n;
        scanf("%d", &n);
        double x1, x2, y1, y2;
        vector<Lines> line;
        set<double> X;
        for (int i = 1; i <= n; i++)
        {
            scanf("%lf%lf%lf%lf", &x1, &y1, &x2, &y2);
            line.push_back({x1, x2, y1, 1});
            line.push_back({x1, x2, y2, -1});
            X.insert(x1);
            X.insert(x2);
        }
        sort(line.begin(), line.end());
        int cnt = 0;
        Hash.clear();
        rHash.clear();
        for (auto& v : X) Hash[v] = ++cnt, rHash[cnt] = v;
        build(1, 1, cnt);
        double area = 0;
        for (int i = 0; i < line.size() - 1; i++)
        {
            update(1, 1, cnt, Hash[line[i].l], Hash[line[i].r] - 1, line[i].val);
            area += seg[1].len2 * (line[i + 1].h - line[i].h);
        }
        printf("%.2lf\n", area);
    }
}
```

3.2.3 Perimeter Combination

```
// 矩形周长并
int n, m[2];
int sum[maxn << 2], cnt[maxn << 2], all[2][maxn];
struct Seg
{
    int l, r, h, d;
    Seg() {}
    Seg(int l, int r, int h, int d) : l(l), r(r), h(h), d(d) {}
    bool operator<(const Seg& rhs) const { return h < rhs.h; }
} a[2][maxn];
#define lson l, m, rt << 1
#define rson m + 1, r, rt << 1 | 1
void pushup(int p, int l, int r, int rt)
{
    if (cnt[rt])
        sum[rt] = all[p][r + 1] - all[p][l];
    else if (l == r)
        sum[rt] = 0;
    else
        sum[rt] = sum[rt << 1] + sum[rt << 1 | 1];
}
```

```
void update(int p, int L, int R, int v, int l, int r, int rt)
{
    if (L <= l && r <= R)
    {
        cnt[rt] += v;
        pushup(p, l, r, rt);
        return;
    }
    int m = l + r >> 1;
    if (L <= m) update(p, L, R, v, lson);
    if (R > m) update(p, L, R, v, rson);
    pushup(p, l, r, rt);
}
int main()
{
    while (scanf("%d", &n) == 1)
    {
        for (int i = 1; i <= n; ++i)
        {
            int x1, y1, x2, y2;
            scanf("%d%d%d%d", &x1, &y1, &x2, &y2);
            all[0][i] = x1, all[0][i + n] = x2;
            all[1][i] = y1, all[1][i + n] = y2;
            a[0][i] = Seg(x1, x2, y1, 1);
            a[0][i + n] = Seg(x1, x2, y2, -1);
            a[1][i] = Seg(y1, y2, x1, 1);
            a[1][i + n] = Seg(y1, y2, x2, -1);
        }
        n <<= 1;
        sort(all[0] + 1, all[0] + 1 + n);
        m[0] = unique(all[0] + 1, all[0] + 1 + n) - all[0] - 1;
        sort(all[1] + 1, all[1] + 1 + n);
        m[1] = unique(all[1] + 1, all[1] + 1 + n) - all[1] - 1;
        sort(a[0] + 1, a[0] + 1 + n);
        sort(a[1] + 1, a[1] + 1 + n);
        int ans = 0;
        for (int i = 0; i < 2; ++i)
        {
            int t = 0, last = 0;
            memset(cnt, 0, sizeof cnt);
            memset(sum, 0, sizeof sum);
            for (int j = 1; j <= n; ++j)
            {
                int l = lower_bound(all[i] + 1, all[i] + 1 + m[i], a[i][j].l) - all[i];
                int r = lower_bound(all[i] + 1, all[i] + 1 + m[i], a[i][j].r) - all[i];
                if (l < r) update(i, l, r - 1, a[i][j].d, 1, m[i], 1);
                t += abs(sum[1] - last);
                last = sum[1];
            }
            ans += t;
        }
        printf("%d\n", ans);
    }
    return 0;
}
```

3.2.4 Chairtree(hjt)

```
const int N = 1000010;
```

```
2
3  struct Three
4  {
5      int l;
6      int r;
7      int v;
8      int son[2];
9  };
10
11 Three tree[N * 20];
12 int root[N];
13 int n, m;
14 int ma[N];
15 int tot;    /// 根节点个数
16
17 void Build(int &x, int l, int r)
18 {
19     tot ++;
20     x = tot;
21     tree[x].l = l;
22     tree[x].r = r;
23     //m printf("%d %d %d\n", x, tree[x].l, tree[x].r);
24     if(l == r){
25         tree[x].v = ma[l];
26     }
27     else{
28         int mid;
29
30         mid = l + ((r - l) >> 1);
31         Build(tree[x].son[0], l, mid);
32         Build(tree[x].son[1], mid + 1, r);
33     }
34 }
35
36 int Quary(int x, int pos)
37 {
38     int ll, rr;
39
40     ll = tree[x].l;
41     rr = tree[x].r;
42     // printf("%d %d %d %d\n", x, pos, ll, rr);
43     if(ll == rr && ll == pos){
44         return tree[x].v;
45     }
46     else{
47         int mid;
48
49         mid = ll + ((rr - ll) >> 1);
50         if(pos <= mid){
51             return Quary(tree[x].son[0], pos);
52         }
53         else{
54             return Quary(tree[x].son[1], pos);
55         }
56     }
57 }
58
59 void Update(int &x, int pre, int pos, int v)
60 {
```

```
    int ll, rr;

    ll = tree[pre].l;
    rr = tree[pre].r;
    tot ++;
    x = tot;
    tree[x] = tree[pre];
    if(ll == rr && ll == pos){
        tree[x].v = v;
    }
    else{
        int mid;

        mid = ll + ((rr - ll) >> 1);
        if(pos <= mid){
            Update(tree[x].son[0], tree[pre].son[0], pos, v);
        }
        else{
            Update(tree[x].son[1], tree[pre].son[1], pos, v);
        }
    }
}

int main(int argc, char const *argv[])
{
    while(scanf("%d%d", &n, &m) == 2){
        tot = 0;
        for(int i = 1; i <= n; i ++){
            scanf("%d", &ma[i]);
        }
        Build(root[0], 1, n);
        for(int i = 1; i <= m; i ++){
            int kind, ops, pos, v;

            scanf("%d%d", &kind, &ops);
            if(ops == 1){
                scanf("%d%d", &pos, &v);
                Update(root[i], root[kind], pos, v);
            }
            else{
                scanf("%d", &pos);
                root[i] = root[kind];   ///!
                printf("%d\n", Quary(root[i], pos));
            }
        }
    }

    return 0;
}
```

3.2.5 BIT

```
int bits[N];
int n;

int lowbit(int x)
{
    return x & -x;
```

```
}

int sum(int x)  ///区间求和，注意权值树状数组
{
    int ans = 0;

    while(x > 0){
        ans += bits[x];
        x -= lowbit(x);
    }

    return ans;
}

int add(int x, int v)
{
    while(x <= n){
        bits[x] += v;
        x += lowbit(x);
    }
}

void ini()
{
    memset(bits, 0, sizeof(bits));
}
```

3.2.6 SegTree

```
const int N = 100010;

struct Tree
{
    int l;
    int r;
    LL sum;
    LL lazy;
};

Tree tree[N * 4];
int ma[N];
int n, m;

void up(int x)
{
    tree[x].sum = tree[x << 1].sum + tree[x << 1 | 1].sum;
}

void down(int x)
{
    LL t = tree[x].lazy;

    if(t){
        tree[x].lazy = 0;
        tree[x << 1].sum += (tree[x << 1].r - tree[x << 1].l + 1) * t;
        tree[x << 1].lazy += t;
        tree[x << 1 | 1].sum += (tree[x << 1 | 1].r - tree[x << 1 | 1].l + 1) * t;
        tree[x << 1 | 1].lazy += t;
```

```
    }
}

void build(int x, int l, int r)
{
    tree[x].l = l;
    tree[x].r = r;
    tree[x].lazy = 0;
    if(l == r){
        tree[x].sum = ma[l];
    }
    else{
        int mid = l + ((r - l) >> 1);

        build(x << 1, l, mid);
        build(x << 1 | 1, mid + 1, r);
        up(x);
    }
}

void update(int x, int l, int r, int v)
{
    int ll = tree[x].l;
    int rr = tree[x].r;

    if(l > r){
        return ;
    }
    if(ll == l && rr == r){
        tree[x].sum += (tree[x].r - tree[x].l + 1) * v;
        tree[x].lazy += v;
    }
    else{
        int mid = ll + ((rr - ll) >> 1);

        down(x);
        update(x << 1, l, min(mid, r), v);
        update(x << 1 | 1, max(mid + 1, l), r, v);
        up(x);
    }
}

LL quary(int x, int l, int r)
{
    int ll = tree[x].l;
    int rr = tree[x].r;

    if(l > r){
        return 0;
    }
    if(ll == l && rr == r){
        return tree[x].sum;
    }
    else{
        int mid = ll + ((rr - ll) >> 1);
        LL t = 0;

        down(x);
        t += quary(x << 1, l, min(mid, r));
```

```
89          t += quary(x << 1 | 1, max(mid + 1, l), r);
90          up(x);
91
92          return t;
93      }
94  }
95
96  /// 区间乘加
97  struct Three
98  {
99      int l;
100     int r;
101     LL lazya;
102     LL lazym;
103     LL sum;
104 };
105
106 Three tree[N * 4];
107 int n, m, MOD;
108 int ma[N];
109
110 void Up(int x)
111 {
112     tree[x].sum = (tree[x << 1].sum + tree[x << 1 | 1].sum) % MOD;
113 }
114
115 void Down(int x)
116 {
117     tree[x << 1].sum = ((tree[x << 1].r - tree[x << 1].l + 1) * tree[x].lazya % MOD +
        tree[x << 1].sum * tree[x].lazym % MOD) % MOD;
118     tree[x << 1 | 1].sum = ((tree[x << 1 | 1].r - tree[x << 1 | 1].l + 1) * tree[x].
        lazya % MOD + tree[x << 1 | 1].sum * tree[x].lazym % MOD) % MOD;
119     tree[x << 1].lazya = (tree[x << 1].lazya * tree[x].lazym % MOD + tree[x].lazya) %
        MOD;
120     tree[x << 1 | 1].lazya = (tree[x << 1 | 1].lazya * tree[x].lazym % MOD + tree[x].
        lazya) % MOD;
121     tree[x << 1].lazym = (tree[x << 1].lazym * tree[x].lazym) % MOD;
122     tree[x << 1 | 1].lazym = (tree[x << 1 | 1].lazym * tree[x].lazym) % MOD;
123     tree[x].lazya = 0;
124     tree[x].lazym = 1;
125 }
126
127 void Build(int x, int l, int r)
128 {
129     tree[x].l = l;
130     tree[x].r = r;
131     tree[x].lazya = 0;
132     tree[x].lazym = 1;
133     if(l == r){
134         tree[x].sum = ma[l];
135     }
136     else{
137         int mid;
138
139         mid = l + ((r - l) >> 1);
140         Build(x << 1, l, mid);
141         Build(x << 1 | 1, mid + 1, r);
142         Up(x);
143     }
```

```
}

void Update(int x, int l, int r, int v, int kind)   /// 0 - mul   1 - add
{
    int ll, rr;

    ll = tree[x].l;
    rr = tree[x].r;
    if(l > r){
        return ;
    }
    if(ll == l && rr == r){
        if(kind){   /// Add
            tree[x].lazya = (tree[x].lazya + v) % MOD;
            tree[x].sum = (tree[x].sum + (rr - ll + 1) * v % MOD) % MOD;
        }
        else{
            tree[x].lazya = (tree[x].lazya * v) % MOD;
            tree[x].lazym = (tree[x].lazym * v) % MOD;
            tree[x].sum = (tree[x].sum * v) % MOD;
        }
    }
    else{
        int mid;

        mid = ll + ((rr - ll) >> 1);
        Down(x);
        Update(x << 1, l, min(mid, r), v, kind);
        Update(x << 1 | 1, max(mid + 1, l), r, v, kind);
        Up(x);
    }
}

LL Quary(int x, int l, int r)
{
    int ll, rr;

    if(l > r){
        return 0;
    }
    //printf("%d %d %d\n", x, l, r);
    ll = tree[x].l;
    rr = tree[x].r;
    //printf("%d %d\n", ll, rr);
    if(ll == l && rr == r){
        return tree[x].sum;
    }
    else{
        int mid;
        LL ans;

        ans = 0;
        mid = ll + ((rr - ll) >> 1);
        Down(x);
        ans = (ans + Quary(x << 1, l, min(mid, r))) % MOD;
        ans = (ans + Quary(x << 1 | 1, max(mid + 1, l), r)) % MOD;

        return ans % MOD;
    }
```

```
}
```

3.2.7 Double Area Combination

```
#include <bits/stdc++.h>
using namespace std;
const int MAXN=2020;
struct Node
{
    int l,r;
    int c;
    double lf,rf;
    double cnt;//¸²¸ÇÂ´¶¯¡¤¡¤¡¤¡¤¡¤¡¤¡¤¤¶¡
    double more;//¸²¸Ç¶}Â´¶¯¡¤¡¤¡¤¡¤¡¤¡¤¡¤¤¶¡
}segTree[MAXN*3];
struct Line
{
    double x,y1,y2;
    double f;
}line[MAXN];

double y[MAXN];

bool cmp(Line a,Line b)
{
    return a.x<b.x;
}

void Build(int i,int l,int r)
{
    segTree[i].l=l;
    segTree[i].r=r;
    segTree[i].cnt=0;
    segTree[i].more=0;
    segTree[i].lf=y[l];
    segTree[i].rf=y[r];
    if(l+1==r)return;
    int mid=(l+r)>>1;
    Build(i<<1,l,mid);
    Build((i << 1) | 1, mid,r);
}
void calen(int i)
{
    if(segTree[i].c>=2)
    {
        segTree[i].more=segTree[i].cnt=segTree[i].rf-segTree[i].lf;
        return;
    }
    else if(segTree[i].c==1)
    {
        segTree[i].cnt=segTree[i].rf-segTree[i].lf;
        if(segTree[i].l+1==segTree[i].r)segTree[i].more=0;
        else segTree[i].more=segTree[i<<1].cnt+segTree[(i<<1)|1].cnt;
    }
    else
    {
        if(segTree[i].l+1==segTree[i].r)
        {
```

```
55              segTree[i].cnt=segTree[i].more=0;
56          }
57          else
58          {
59              segTree[i].cnt=segTree[i<<1].cnt+segTree[(i<<1)|1].cnt;
60              segTree[i].more=segTree[i<<1].more+segTree[(i<<1)|1].more;
61          }
62      }
63  }
64  void update(int i,Line e)
65  {
66      if(e.y1==segTree[i].lf&&segTree[i].rf==e.y2)
67      {
68          segTree[i].c+=e.f;
69          calen(i);
70          return;
71      }
72      if(e.y2<=segTree[i<<1].rf) update(i<<1,e);
73      else if(e.y1>=segTree[(i<<1)|1].lf) update((i<<1)|1,e);
74      else
75      {
76          Line temp=e;
77          temp.y2=segTree[i<<1].rf;
78          update(i<<1,temp);
79          temp=e;
80          temp.y1=segTree[(i<<1)|1].lf;
81          update((i<<1)|1,temp);
82      }
83      calen(i);
84  }
85  int main()
86  {
87      //freopen("in.txt","r",stdin);
88      //freopen("out.txt","w",stdout);
89      int T;
90      int n;
91      double x1,y1,x2,y2;
92      scanf("%d",&T);
93      while(T--)
94      {
95          scanf("%d",&n);
96          int t=1;
97          for(int i=1;i<=n;i++)
98          {
99              scanf("%lf%lf%lf%lf",&x1,&y1,&x2,&y2);
100             line[t].x=x1;
101             //□□□□□□□L□□□□□□□□□□□□□□□%□□□□□□□□
102             line[t].y1=y1;
103             line[t].y2=y2;
104             line[t].x=x1;
105             line[t].f=1;
106             y[t]=y1;
107             t++;
108             line[t].y1=y1;
109             line[t].y2=y2;
110             line[t].x=x2;
111             line[t].f=-1;
112             y[t]=y2;
113             t++;
```

```
        }
        sort(line+1,line+t,cmp);
        sort(y+1,y+t);
        Build(1,1,t-1);
        update(1,line[1]);
        double ans=0;
        for(int i=2;i<t;i++)
        {
            ans+=segTree[1].more*(line[i].x-line[i-1].x);
            update(1,line[i]);
        }
        printf("%.2lf\n",ans);
    }
    return 0;
}
```

3.3 Splay Tree

3.3.1 IntervalSplay

```
/*
数据范围 100010
1. 插入x数
2. 删除x数(若有多个相同的数，因只删除一个)
3. 查询x数的排名(若有多个相同的数，因输出最小的排名)
4. 查询排名为x的数
5. 求x的前驱(前驱定义为小于x，且最大的数)
6. 求x的后继(后继定义为大于x，且最小的数)
*/
const int N = 100010;
const int INF = 0x7fffffff;

struct Tree
{
    int fa; ///
    int v;
    int siz;    /// 尺寸
    int son[2]; /// 0 - left
    int cnt;    /// 次数
};

Tree tree[N];
int n, m;
int root;
int ts; /// 树的大小

void Clean(int x)
{
    tree[x].son[0] = tree[x].son[1] = tree[x].v = tree[x].fa = tree[x].siz = tree[x].
    cnt = 0;
}

void Up(int x)
{
    if(x){
        tree[x].siz = tree[x].cnt;
        if(tree[x].son[0]){
            tree[x].siz += tree[tree[x].son[0]].siz;
        }
```

```
39          if(tree[x].son[1]){
40              tree[x].siz += tree[tree[x].son[1]].siz;
41          }
42      }
43  }
44
45  void Rotate(int x, int kind) /// 0 - zag    1 - zig
46  {
47      int y, z;
48
49      y = tree[x].fa;
50      z = tree[y].fa;
51      tree[y].son[!kind] = tree[x].son[kind];
52      tree[tree[x].son[kind]].fa = y;
53      tree[x].son[kind] = y;
54      tree[y].fa = x;
55      tree[z].son[tree[z].son[1] == y] = x;
56      tree[x].fa = z;
57      Up(y);
58  }
59
60  void Splay(int x, int goal)
61  {
62      if(x == goal){
63          return ;
64      }
65      while(tree[x].fa != goal){
66          int y, z;
67          bool rx, ry;
68
69          y = tree[x].fa;
70          z = tree[y].fa;
71  //        Down(z);
72  //        Down(y);
73  //        Down(x);
74          rx = (x == tree[y].son[0]);
75          ry = (y == tree[z].son[0]);
76          if(z == goal){
77              Rotate(x, rx);
78          }
79          else{
80              if(rx == ry){
81                  Rotate(y, ry);
82              }
83              else{
84                  Rotate(x, rx);
85              }
86              Rotate(x, ry);
87          }
88      }
89      Up(x);
90      if(!goal){
91          root = x;
92      }
93  }
94
95  int Findrank(int x)
96  {
97      int ans;
```

```
98      int now;
99
100     now = root;
101     ans = 0;
102     // printf("begin = %d\n", x);
103     while(true){
104         // cout << now << endl;
105         if(tree[now].v > x){
106             // printf("Turn left\n");
107             now = tree[now].son[0];
108         }
109         else{
110             // printf("Turn right\n");
111             if(tree[now].son[0]){
112                 ans += tree[tree[now].son[0]].siz;
113             }
114             // printf("now = %d = %d\n", tree[now].v, x);
115             if(tree[now].v == x){
116                 // cout << 998244353 << endl;
117                 Splay(now, 0);
118
119                 return ans + 1;
120             }
121             ans += tree[now].cnt; ///!
122             now = tree[now].son[1];
123         }
124     }
125 }
126
127 int Findnum(int x)
128 {
129     int now;
130
131     now = root;
132     while(true){
133         if(tree[now].son[0] && x <= tree[tree[now].son[0]].siz){
134             now = tree[now].son[0];
135         }
136         else{
137             int t;
138
139             t = tree[now].cnt;
140             if(tree[now].son[0]){
141                 t += tree[tree[now].son[0]].siz;
142             }
143             if(x <= t){
144                 Splay(now, 0);
145
146                 return tree[now].v;
147             }
148             x -= t;
149             now = tree[now].son[1];
150         }
151     }
152 }
153
154 int Findpre(int x)
155 {
156     int now;
```

```
157
158     Findrank(x);
159     // cout << "ENDDDDDDDDDDDDDD" << endl;
160     now = tree[root].son[0];
161     while(tree[now].son[1]){
162         now = tree[now].son[1];
163     }
164
165     return now;
166 }
167
168 int Findnet(int x)
169 {
170     int now;
171
172     Findrank(x);
173     now = tree[root].son[1];
174     while(tree[now].son[0]){
175         // cout << 666 << endl;
176         now = tree[now].son[0];
177     }
178
179     return now;
180 }
181
182 void Del(int x)
183 {
184     int t;
185     int l;
186     int now;
187
188     Findrank(x);
189     now = root;
190     if(tree[root].cnt > 1){
191         tree[root].cnt --;
192         tree[root].siz --;
193         return ;
194     }
195     if(!tree[root].son[0] && !tree[root].son[1]){   ///only one
196         Clean(root);
197         root = 0;
198
199         return ;
200     }
201     if(!tree[root].son[0]){
202         root = tree[root].son[1];
203         tree[root].fa = 0;
204         Clean(now);
205
206         return ;
207     }
208     if(!tree[root].son[1]){
209         root = tree[root].son[0];
210         tree[root].fa = 0;
211         Clean(now);
212
213         return ;
214     }
215     t = Findpre(tree[root].v);
```

```
216     Splay(t, 0);
217     tree[tree[now].son[1]].fa = root;
218     tree[root].son[1] = tree[now].son[1];
219     Clean(now);
220     Up(root);
221 }
222 
223 void Insert(int x)
224 {
225     if(!root){
226         // cout << 6566 << endl;
227         ts ++;
228         tree[ts].son[0] = tree[ts].son[1] = tree[ts].fa = 0;
229         tree[ts].v = x;
230         tree[ts].cnt = 1;
231         tree[ts].siz = 1;
232         root = ts;
233 
234         return;
235     }
236     else{
237         int now;
238         int t;
239 
240         t = 0;
241         now = root;
242         while(true){
243             if(tree[now].v == x){
244                 tree[now].cnt ++;
245                 Up(t);
246                 Splay(now, 0);
247                 break;
248             }
249             t = now;
250             now = tree[now].son[tree[now].v < x];
251             if(!now){
252                 ts ++;
253                 tree[ts].son[0] = tree[ts].son[1] = 0;
254                 tree[ts].fa = t;
255                 tree[ts].v = x;
256                 tree[ts].cnt = 1;
257                 tree[t].son[tree[t].v < x] = ts;
258                 Up(t);
259                 Splay(ts, 0);
260                 break;
261             }
262         }
263     }
264 }
265 
266 void Show()
267 {
268     printf("root = %d\n", root);
269     for(int i = 0; i <= ts; i ++){
270         printf("%d l = %d r = %d num = %d cnt = %d siz = %d fa = %d\n", i, tree[i].son
    [0], tree[i].son[1], tree[i].v, tree[i].cnt, tree[i].siz, tree[i].fa);
271     }
272 }
273 
```

```
274 void Ini()
275 {
276     root = ts = 0;
277 }
278
279 int main(int argc, char const *argv[])
280 {
281     while(scanf("%d", &n) == 1){
282         Ini();
283         for(int i = 0; i < n; i ++){
284             int ops, x;
285
286             scanf("%d%d", &ops, &x);
287             if(ops == 1){
288                 Insert(x);
289             }
290             else if(ops == 2){
291                 Del(x);
292             }
293             else if(ops == 3){
294                 printf("%d\n", Findrank(x));
295             }
296             else if(ops == 4){
297                 printf("%d\n", Findnum(x));
298             }
299             else if(ops == 5){
300                 Insert(x);
301                 // Show();
302                 printf("%d\n", tree[Findpre(x)].v);
303                 Del(x);
304             }
305             else{
306                 Insert(x);
307                 // Show();
308                 printf("%d\n", tree[Findnet(x)].v);
309                 Del(x);
310             }
311             // Show();
312         }
313     }
314
315     return 0;
316 }
```

3.3.2 IdxSplay

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  const int N = 100010;
5  const int INF = 0x7fffffff;
6
7  struct Tree
8  {
9      int fa; ///
10     int v;
11     int maxx;
12     int lazy;
```

```
13      int cnt;
14      int son[2]; /// 0 - left
15      bool rev;   /// 翻转标记
16  };
17
18  Tree tree[N];
19  int n, m;
20  int root;
21
22  void Ini(int x, int v)
23  {
24      tree[x].v = tree[x].maxx = v;
25      tree[x].cnt = 1;
26      tree[x].lazy = tree[x].rev = tree[x].son[0] = tree[x].son[1] = 0;
27  }
28
29  void Up(int x)
30  {
31      tree[x].maxx = tree[x].v;
32      tree[x].cnt = 1;
33      if(tree[x].son[0]){
34          tree[x].maxx = max(tree[x].maxx, tree[tree[x].son[0]].maxx);
35          tree[x].cnt += tree[tree[x].son[0]].cnt;
36      }
37      if(tree[x].son[1]){
38          tree[x].maxx = max(tree[x].maxx, tree[tree[x].son[1]].maxx);
39          tree[x].cnt += tree[tree[x].son[1]].cnt;
40      }
41  }
42
43  void Down(int x)
44  {
45      int t;
46
47      if(!x){
48          return ;
49      }
50      t = tree[x].lazy;
51      if(t){
52          if(tree[x].son[0]){
53              tree[tree[x].son[0]].v += t;
54              tree[tree[x].son[0]].maxx += t;
55              tree[tree[x].son[0]].lazy += t;
56          }
57          if(tree[x].son[1]){
58              tree[tree[x].son[1]].v += t;
59              tree[tree[x].son[1]].maxx += t;
60              tree[tree[x].son[1]].lazy += t;
61          }
62      }
63      tree[x].lazy = 0;
64      if(tree[x].rev){
65          tree[tree[x].son[0]].rev ^= 1;
66          tree[tree[x].son[1]].rev ^= 1;
67          swap(tree[x].son[0], tree[x].son[1]);
68          tree[x].rev = 0;
69      }
70  }
71  void Rotate(int x, int kind) /// 0 - zag    1 - zig
```

```
72  {
73      int y, z;
74
75      y = tree[x].fa;
76      z = tree[y].fa;
77      tree[y].son[!kind] = tree[x].son[kind];
78      tree[tree[x].son[kind]].fa = y;
79      tree[x].son[kind] = y;
80      tree[y].fa = x;
81      tree[z].son[tree[z].son[1] == y] = x;
82      tree[x].fa = z;
83      Up(y);
84  }
85
86  void Splay(int x, int goal)
87  {
88      if(x == goal){
89          return ;
90      }
91      while(tree[x].fa != goal){
92          int y, z;
93          bool rx, ry;
94
95          y = tree[x].fa;
96          z = tree[y].fa;
97          Down(z);
98          Down(y);
99          Down(x);
100         rx = (x == tree[y].son[0]);
101         ry = (y == tree[z].son[0]);
102         if(z == goal){
103             Rotate(x, rx);
104         }
105         else{
106             if(rx == ry){
107                 Rotate(y, ry);
108             }
109             else{
110                 Rotate(x, rx);
111             }
112             Rotate(x, ry);
113         }
114     }
115     Up(x);
116     if(!goal){
117         root = x;
118     }
119 }
120
121 int Find(int x)
122 {
123     int t;
124
125     t = root;
126     Down(t);
127     while(tree[tree[t].son[0]].cnt != x){
128         if(x < tree[tree[t].son[0]].cnt){
129             t = tree[t].son[0];
130         }
```

```
131         else{
132             x -= (tree[tree[t].son[0]].cnt + 1);
133             t = tree[t].son[1];
134         }
135         Down(t);
136     }
137 
138     return t;
139 }
140 
141 void Update(int l, int r, int v)
142 {
143     int x, y;
144 
145     x = Find(l - 1);
146     y = Find(r + 1);
147     Splay(x, 0);
148     Splay(y, x);
149     tree[tree[y].son[0]].maxx += v;
150     tree[tree[y].son[0]].lazy += v;
151     tree[tree[y].son[0]].v += v;
152 }
153 
154 void Reverse(int l, int r)
155 {
156     int x, y;
157 
158     x = Find(l - 1);
159     y = Find(r + 1);
160     Splay(x, 0);
161     Splay(y, x);
162     tree[tree[y].son[0]].rev ^= 1;
163 }
164 
165 int Quary(int l, int r)
166 {
167     int x, y;
168 
169     x = Find(l - 1);
170     y = Find(r + 1);
171     Splay(x, 0);
172     Splay(y, x);
173 
174     return tree[tree[y].son[0]].maxx;
175 }
176 
177 int Build(int l, int r)
178 {
179     int mid;
180     int ll, rr;
181 
182     if(l > r){
183         return 0;
184     }
185     else if(l == r){
186         return l;
187     }
188     mid = l + ((r - l) >> 1);
189     ll = Build(l, mid - 1);
```

```
190     rr = Build(mid + 1, r);
191     tree[mid].son[0] = ll;
192     tree[mid].son[1] = rr;
193     tree[ll].fa = tree[rr].fa = mid;
194     Up(mid);
195 
196     return mid;
197 }
198 
199 void Init(int n)
200 {
201     int li;
202 
203     Ini(0, -INF);
204     Ini(1, -INF);
205     Ini(n + 2, -INF);
206     li = n + 1;
207     for(int i = 2; i <= li; i ++){
208         Ini(i, 0);
209     }
210     root = Build(1, n + 2);
211     tree[root].fa = 0;
212     tree[0].fa = 0;
213     tree[0].son[1] = root;
214     tree[0].cnt = 0;
215 }
216 
217 int main(int argc, char const *argv[])
218 {
219     while(scanf("%d%d", &n, &m) == 2){
220         Init(n);
221         for(int i = 0; i < m; i ++){
222             int x;
223             int l, r;
224 
225             scanf("%d", &x);
226             if(x == 1){
227                 int v;
228 
229                 scanf("%d%d%d", &l, &r, &v);
230                 Update(l, r, v);
231             }
232             else if(x == 2){
233                 scanf("%d%d", &l, &r);
234                 Reverse(l, r);
235             }
236             else{
237                 scanf("%d%d", &l, &r);
238                 printf("%d\n", Quary(l, r));
239             }
240         }
241     }
242 
243     return 0;
244 }
```

3.3.3 Exotic$_t$*treeree*

```
1 /**
```

```
2  1.查询k在区间内的排名
3  2.查询区间内排名为k的值
4  3.修改某一位值上的数值
5  4.查询k在区间内的前驱(前驱定义为小于x，且最大的数)
6  5.查询k在区间内的后继(后继定义为大于x，且最小的数)
7  **/
8  int n,m,tot;
9  int num[50003],roots[2000003];
10 struct Splay
11 {
12     int fa,ch[2],siz,data,occ;  /// occ - cnt
13 }a[2000003];
14 int in()
15 {
16     int t=0,f=1;
17     char ch=getchar();
18     while (!pd(ch))
19     {
20         if (ch=='-') f=-1;
21         ch=getchar();
22     }
23     while (pd(ch)) t=(t<<3)+(t<<1)+ch-'0',ch=getchar();
24     return f*t;
25 }
26 void ct(int x)
27 {
28     a[x].siz=a[a[x].ch[0]].siz+a[a[x].ch[1]].siz+a[x].occ;
29 }
30 void made(int x,int id)
31 {
32     a[id].data=x,
33     a[id].occ=a[id].siz=1,
34     a[id].ch[0]=a[id].ch[1]=a[id].fa=0;
35 }
36 void rorate(int now,bool mk)
37 {
38     int pa=a[now].fa;
39     a[a[now].ch[mk]].fa=pa;
40     a[pa].ch[!mk]=a[now].ch[mk];
41     a[now].fa=a[pa].fa;
42     if (a[pa].fa)
43     {
44         if (a[a[pa].fa].ch[0]==pa) a[a[pa].fa].ch[0]=now;
45         else a[a[pa].fa].ch[1]=now;
46     }
47     a[now].ch[mk]=pa;
48     a[pa].fa=now;
49     ct(pa);ct(now);
50 }
51 void splay(int rt,int now,int goal)
52 {
53     int pa;
54     while (a[now].fa!=goal)
55     {
56         pa=a[now].fa;
57         if (a[pa].fa==goal)
58         {
59             if (a[pa].ch[0]==now) rorate(now,1);
60             else rorate(now,0);
```

```
61          }
62          else if (a[a[pa].fa].ch[0]==pa)
63          {
64              if (a[pa].ch[0]==now) rorate(pa,1);
65              else rorate(now,0);
66              rorate(now,1);
67          }
68          else
69          {
70              if (a[pa].ch[1]==now) rorate(pa,0);
71              else rorate(now,1);
72              rorate(now,0);
73          }
74      }
75      if (!goal) roots[rt]=now;
76  }
77  void insert(int rt,int x,int id)
78  {
79      if (!roots[rt]) {made(x,id);roots[rt]=id;return;}
80      int now=roots[rt];
81      while (now)
82      {
83          if (a[now].data==x) {a[now].occ++;a[now].siz++;splay(rt,now,0);return;}
84          if (a[now].data>x)
85          {
86              if (!a[now].ch[0]) {made(x,id);a[now].ch[0]=id;a[id].fa=now;break;}
87              else now=a[now].ch[0];
88          }
89          else
90          {
91              if (!a[now].ch[1]) {made(x,id);a[now].ch[1]=id;a[id].fa=now;break;}
92              else now=a[now].ch[1];
93          }
94      }
95      splay(rt,id,0);
96  }
97  int find(int root,int x)
98  {
99      int now=root;
100     while (now)
101     {
102         if (a[now].data==x) return now;
103         if (a[now].data>x) now=a[now].ch[0];
104         else now=a[now].ch[1];
105     }
106 }
107 int findmax(int now)
108 {
109     while (a[now].ch[1]) now=a[now].ch[1];
110     return now;
111 }
112 int find_next_min(int rt,int x)
113 {
114     int now=roots[rt],ans=-0x7fffffff;
115     while (now)
116     {
117         if (a[now].data<x)
118         {
119             if (ans<a[now].data)ans=a[now].data;
```

```
120             now=a[now].ch[1];
121         }
122         else now=a[now].ch[0];
123     }
124     return ans;
125 }
126 int find_next_max(int rt,int x)
127 {
128     int now=roots[rt],t=0,ans=0x7fffffff;
129     while (now)
130     {
131         if (a[now].data>x)
132         {
133             if (ans>a[now].data) ans=a[now].data,t=now;
134             now=a[now].ch[0];
135         }
136         else now=a[now].ch[1];
137     }
138     return ans;
139 }
140 void replace(int rt,int x,int k)
141 {
142     int now=find(roots[rt],x);
143     splay(rt,now,0);
144     if (a[now].occ>1) {a[now].occ--;a[now].siz--;}
145     //else if (a[now].siz==1) roots[rt]=0;
146     else if (!a[now].ch[0])
147     {
148         roots[rt]=a[now].ch[1];
149         a[a[now].ch[1]].fa=0;
150     }
151     else if (!a[now].ch[1])
152     {
153         roots[rt]=a[now].ch[0];
154         a[a[now].ch[0]].fa=0;
155     }
156     else
157     {
158         splay(rt,findmax(a[now].ch[0]),now);
159         a[a[now].ch[0]].ch[1]=a[now].ch[1];
160         a[a[now].ch[1]].fa=a[now].ch[0];
161         a[a[now].ch[0]].fa=0;
162         roots[rt]=a[now].ch[0];
163         ct(a[now].ch[0]);
164     }
165     if (!a[now].occ)insert(rt,k,now);
166     else insert(rt,k,++tot);
167 }
168 int find_rank(int rt,int x)//这里的findrank实际上是在splay里找比x小的数的数量
169 {
170     int now=roots[rt],ans=0;
171     while (now)
172     {
173         if (a[now].data>x) now=a[now].ch[0];
174         else if (a[now].data<x)
175             ans+=(a[now].occ+a[a[now].ch[0]].siz),
176             now=a[now].ch[1];
177         else {ans+=a[a[now].ch[0]].siz;break;}
178     }
```

```
179     return ans;
180 }
181 void build(int now,int begin,int end)
182 {
183     for (int i=begin;i<=end;i++) insert(now,num[i],++tot);
184     if (begin==end) return;
185     int mid=(begin+end)>>1;
186     build(now<<1,begin,mid);
187     build(now<<1|1,mid+1,end);
188 }
189 int solve1(int now,int begin,int end,int l,int r,int k)
190 {
191     if (l<=begin&&end<=r) return find_rank(now,k);
192     int mid=(begin+end)>>1,rank=0;
193     if (mid>=l) rank+=solve1(now<<1,begin,mid,l,r,k);
194     if (mid<r) rank+=solve1(now<<1|1,mid+1,end,l,r,k);
195     return rank;
196 }
197 int solve2(int l,int r,int k)
198 {
199     int begin=0,end=1e8+1,mid;
200     while (begin<end)
201     {
202         mid=(begin+end)>>1;
203         // printf("%d %d\n", mid, solve1(1,1,n,l,r,mid));
204         if (solve1(1,1,n,l,r,mid)<k)
205             begin=mid+1;
206         else end=mid;
207     }
208     return begin-1;
209 }
210 void solve3(int now,int begin,int end,int pos,int k)
211 {
212     replace(now,num[pos],k);
213     if (begin==end) {num[pos]=k;return;}
214     int mid=(begin+end)>>1;
215     if (mid>=pos) solve3(now<<1,begin,mid,pos,k);
216     else solve3(now<<1|1,mid+1,end,pos,k);
217 }
218 int solve4(int now,int begin,int end,int l,int r,int k)
219 {
220     if (l<=begin&&end<=r) return find_next_min(now,k);
221     int mid=(begin+end)>>1,ans=-0x7fffffff;
222     if (mid>=l) ans=max(ans,solve4(now<<1,begin,mid,l,r,k));
223     if (mid<r) ans=max(ans,solve4(now<<1|1,mid+1,end,l,r,k));
224     return ans;
225 }
226 int solve5(int now,int begin,int end,int l,int r,int k)
227 {
228     if (l<=begin&&end<=r) return find_next_max(now,k);
229     int mid=(begin+end)>>1,ans=0x7fffffff;
230     if (mid>=l) ans=min(ans,solve5(now<<1,begin,mid,l,r,k));
231     if (mid<r) ans=min(ans,solve5(now<<1|1,mid+1,end,l,r,k));
232     return ans;
233 }
234 int main()
235 {
236     n=in();m=in();
237     int opt,x,y,k;
```

```
    for (int i=1;i<=n;i++) num[i]=in();
    build(1,1,n);
    while (m--)
    {
        opt=in();
        if (opt!=3)x=in(),y=in(),k=in();
        else x=in(),y=in();
        if (opt==1) printf("%d\n",solve1(1,1,n,x,y,k)+1);
        else if (opt==2) printf("%d\n",solve2(x,y,k));
        else if (opt==3) solve3(1,1,n,x,y);
        else if (opt==4) printf("%d\n",solve4(1,1,n,x,y,k));
        else printf("%d\n",solve5(1,1,n,x,y,k));
    }
}
```

3.3.4 Ttreeree

```
const int N = 50050;
const int INF = 0x7fffffff;

struct Sblay
{
    int fa; ///
    int v;
    int siz;    /// 尺寸
    int son[2]; /// 0 - left
    int cnt;    /// 次数
};

Sblay splay[N * 40];
int n, m;
int root[N * 40];   /// 每颗平衡树的终点qwq
int ts; ///
int ma[N];

void Show()
{
    for(int i = 1; i <= 7; i ++){
        printf("root = %d ", root[i]);
    }
    printf("\n");
    for(int i = 0; i <= ts; i ++){
        printf("%d l = %d r = %d num = %d cnt = %d siz = %d fa = %d\n", i, splay[i].son
    [0], splay[i].son[1], splay[i].v, splay[i].cnt, splay[i].siz, splay[i].fa);
    }
}

void Clean(int x)
{
    splay[x].son[0] = splay[x].son[1] = splay[x].v = splay[x].fa = splay[x].siz = splay
    [x].cnt = 0;
}

void Up(int x)
{
    if(x){
        splay[x].siz = splay[x].cnt;
        if(splay[x].son[0]){
```

```
            splay[x].siz += splay[splay[x].son[0]].siz;
        }
        if(splay[x].son[1]){
            splay[x].siz += splay[splay[x].son[1]].siz;
        }
    }
}

void Rotate(int x, int kind) /// 0 - zag    1 - zig
{
    int y, z;

    y = splay[x].fa;
    z = splay[y].fa;
    splay[y].son[!kind] = splay[x].son[kind];
    splay[splay[x].son[kind]].fa = y;
    splay[x].son[kind] = y;
    splay[y].fa = x;
    splay[z].son[splay[z].son[1] == y] = x;
    splay[x].fa = z;
    Up(y);
}

void Splay(int num, int x, int goal)
{
    if(x == goal){
        return ;
    }
    while(splay[x].fa != goal){
        int y, z;
        bool rx, ry;

        y = splay[x].fa;
        z = splay[y].fa;
//        Down(z);
//        Down(y);
//        Down(x);
        rx = (x == splay[y].son[0]);
        ry = (y == splay[z].son[0]);
        if(z == goal){
            Rotate(x, rx);
        }
        else{
            if(rx == ry){
                Rotate(y, ry);
            }
            else{
                Rotate(x, rx);
            }
            Rotate(x, ry);
        }
    }
    Up(x);
    if(!goal){
        root[num] = x;
    }
}

int Findrank(int num, int x)
```

```
{
    int ans;
    int now;

    now = root[num];
    ans = 0;
    // printf("begin = %d %d\n", num, root[num]);
    while(now){ /// 原本是 true  但有可能不存在qwq
        // printf("now = %d v = %d\n", now, splay[now].v);
        if(splay[now].v > x){
            // printf("Turn left\n");
            now = splay[now].son[0];
        }
        else{
            // printf("Turn right\n");
            if(splay[now].son[0]){
                ans += splay[splay[now].son[0]].siz;
            }
            // printf("now = %d = %d\n", splay[now].v, x);
            if(splay[now].v == x){
                // cout << 998244353 << endl;
                Splay(num, now, 0);

                return ans; /// 因为不可合并所以不能 + 1
            }
            ans += splay[now].cnt; ///! /// 算上自己
            now = splay[now].son[1];
        }
    }

    return ans;
}

int Findpre(int num, int x)
{
    int now;

    // printf("pre %d %d\n", num, x);
    Findrank(num, x);
    // cout << "ENDDDDDDDDDDDDDD" << endl;
    now = splay[root[num]].son[0];
    if(!now){
        return -1;
    }
    while(splay[now].son[1]){
        // printf("now = %d\n", now);
        now = splay[now].son[1];
    }

    return now;
}

int Findnet(int num, int x)
{
    int now;

    Findrank(num, x);
    now = splay[root[num]].son[1];
    if(!now){
```

```
158         return -1;
159     }
160     while(splay[now].son[0]){
161        // cout << 666 << endl;
162        now = splay[now].son[0];
163     }
164 
165     return now;
166 }
167 
168 void Del(int num, int x)
169 {
170     int t;
171     int l;
172     int now;
173 
174     Findrank(num, x);
175     now = root[num];
176     if(splay[now].cnt > 1){
177        splay[now].cnt --;
178        splay[now].siz --;
179 
180        return ;
181     }
182     if(!splay[now].son[0] && !splay[now].son[1]){   ///only one
183        Clean(now);
184        root[num] = 0;
185 
186        return ;
187     }
188     if(!splay[root[num]].son[0]){
189        root[num] = splay[now].son[1];
190        splay[root[num]].fa = 0;
191        Clean(now);
192 
193        return ;
194     }
195     if(!splay[root[num]].son[1]){
196        root[num] = splay[root[num]].son[0];
197        splay[root[num]].fa = 0;
198        Clean(now);
199 
200        return ;
201     }
202     t = Findpre(num, splay[root[num]].v);
203     Splay(num, t, 0);
204     splay[splay[now].son[1]].fa = root[num];
205     splay[root[num]].son[1] = splay[now].son[1];
206     Clean(now);
207     Up(root[num]);
208 }
209 
210 void Insert(int num, int x)
211 {
212     if(!root[num]){
213         // cout << 6566 << endl;
214         ts ++;
215         splay[ts].son[0] = splay[ts].son[1] = splay[ts].fa = 0;
216         splay[ts].v = x;
```

```
217         splay[ts].cnt = 1;
218         splay[ts].siz = 1;
219         root[num] = ts;
220 
221         return;
222     }
223     else{
224         int now;
225         int t;
226 
227         t = 0;
228         now = root[num];
229         while(true){
230             if(splay[now].v == x){
231                 splay[now].cnt ++;
232                 Up(t);
233                 Splay(num, now, 0);
234                 break;
235             }
236             t = now;
237             now = splay[now].son[splay[now].v < x];
238             if(!now){
239                 ts ++;
240                 splay[ts].son[0] = splay[ts].son[1] = 0;
241                 splay[ts].fa = t;
242                 splay[ts].v = x;
243                 splay[ts].cnt = 1;
244                 splay[t].son[splay[t].v < x] = ts;
245                 Up(t);
246                 Splay(num, ts, 0);
247                 break;
248             }
249         }
250     }
251 }
252 
253 void Ini()
254 {
255     memset(root, 0, sizeof(root));
256     ts = 0;
257 }
258 
259 void Build(int x, int l, int r)
260 {
261     // printf("build %d %d %d\n", x, l, r);
262     for(int i = l; i <= r; i ++){
263         Insert(x, ma[i]);
264     }
265     // printf("root = %d\n", root[x]);
266     // Show();
267     // printf("%d\n", ts);
268     if(l == r){
269         return;
270     }
271     else{
272         int mid;
273 
274         mid = l + ((r - l) >> 1);
275         Build(x << 1, l, mid);
```

```
276         Build(x << 1 | 1, mid + 1, r);
277     }
278 }
279 
280 int Quary1(int x, int l, int r, int v, int ll, int rr)
281 {
282     if(l > r){
283         return 0;
284     }
285     // printf("l = %d r = %d ll = %d rr = %d\n", l, r, ll, rr);
286     if(l == ll && r == rr){
287         return Findrank(x, v);
288     }
289     else{
290         int mid;
291         int t;
292 
293         t = 0;
294         mid = ll + ((rr - ll) >> 1);
295         t += Quary1(x << 1, l, min(mid, r), v, ll, mid);
296         t += Quary1(x << 1 | 1, max(mid + 1, l), r, v, mid + 1, rr);
297 
298         return t;
299     }
300 }
301 
302 int Quary2(int l, int r, int x)
303 {
304     int dl, dr, dmid;
305     int ans;
306 
307     dl = 0;
308     ans = dl;
309     dr = 1e8 + 1;
310     while(dl <= dr){
311         dmid = dl + ((dr - dl) >> 1);
312         // printf("mid = %d  %d\n", dmid, Quary1(1, l, r, dmid, 1, n));
313         if(Quary1(1, l, r, dmid, 1, n) < x){
314             ans = dmid;
315             dl = dmid + 1;
316         }
317         else{
318             dr = dmid - 1;
319         }
320     }
321 
322     return ans;
323 }
324 
325 void Update(int x, int pos, int v, int ll, int rr)
326 {
327     // printf("**** %d %d %d\n", x, ll, rr);
328     // Show();
329     Del(x, ma[pos]);
330     // Show();
331     Insert(x, v);
332     // Show();
333     // printf("*****\n");
334     if(ll == rr){
```

```
335         ma[pos] = v;
336     }
337     else{
338         int mid;
339 
340         mid = ll + ((rr - ll) >> 1);
341         if(mid >= pos){
342             Update(x << 1, pos, v, ll, mid);
343         }
344         else{
345             Update(x << 1 | 1, pos, v, mid + 1, rr);
346         }
347     }
348 }
349 
350 int Quary3(int x, int l, int r, int v, int ll, int rr)
351 {
352     if(l > r){
353         return -INF;
354     }
355     if(ll == l && rr == r){
356         int ans;
357 
358         // Show();
359         Insert(x, v);
360         // Show();
361         ans = Findpre(x, v);
362         Del(x, v);
363         // Show();
364         // printf("range  %d %d %d %d %d %d\n", l, r, ll, rr, splay[ans].v, ans);
365         if(ans == -1){
366             return -INF;
367         }
368         else{
369             return splay[ans].v;
370         }
371     }
372     else{
373         int mid;
374         int ans;
375 
376         ans = -INF;
377 
378         mid = ll + ((rr - ll) >> 1);
379         ans = max(ans, Quary3(x << 1, l, min(mid, r), v, ll, mid));
380         ans = max(ans, Quary3(x << 1 | 1, max(mid + 1, l), r, v, mid + 1, rr));
381 
382         // printf("range  %d %d %d %d %d\n", l, r, ll, rr, ans);
383         return ans;
384     }
385 }
386 
387 int Quary4(int x, int l, int r, int v, int ll, int rr)
388 {
389     if(l > r){
390         return INF;
391     }
392     // printf("range  %d %d %d %d\n", l, r, ll, rr);
393     if(ll == l && rr == r){
```

```
394         int ans;
395 
396         Insert(x, v);
397         ans = Findnet(x, v);
398         Del(x, v);
399         // Show();
400         // printf("range  %d %d %d %d %d %d\n", l, r, ll, rr, splay[ans].v, ans);
401         if(ans == -1){
402             return INF;
403         }
404         else{
405             return splay[ans].v;
406         }
407     }
408     else{
409         int mid;
410         int ans;
411 
412         ans = INF;
413 
414         mid = ll + ((rr - ll) >> 1);
415         ans = min(ans, Quary4(x << 1, l, min(mid, r), v, ll, mid));
416         ans = min(ans, Quary4(x << 1 | 1, max(mid + 1, l), r, v, mid + 1, rr));
417         // printf("range  %d %d %d %d %d\n", l, r, ll, rr, ans);
418 
419         return ans;
420     }
421 }
422 
423 int main(int argc, char const *argv[])
424 {
425     while(scanf("%d%d", &n, &m) == 2){
426         Ini();
427         for(int i = 1; i <= n; i ++){
428             scanf("%d", &ma[i]);
429         }
430         Build(1, 1, n);
431         for(int i = 0; i < m; i ++){
432             int ops;
433             int l, r;
434 
435             scanf("%d", &ops);
436             if(ops == 1){
437                 int x;
438 
439                 scanf("%d%d%d", &l, &r, &x);
440                 printf("%d\n", Quary1(1, l, r, x, 1, n) + 1);   /// 这里 + 1
441             }
442             else if(ops == 2){
443                 int x;
444                 int t;
445 
446                 scanf("%d%d%d", &l, &r, &x);
447                 t = Quary2(l, r, x);
448                 if(!t){
449                     printf("-1\n");
450                 }
451                 else{
452                     printf("%d\n", Quary2(l, r, x));
```

```
453                 }
454             }
455             else if(ops == 3){
456                 int pos, x;
457 
458                 scanf("%d%d", &pos, &x);
459                 // Show();
460                 Update(1, pos, x, 1, n);
461                 /// printf("%d\n", ma[pos]);
462                 // printf("root = %d %d %d\n", root[7], root[3], root[1]);
463                 // Show();
464             }
465             else if(ops == 4){
466                 int x;
467 
468                 scanf("%d%d%d", &l, &r, &x);
469                 printf("%d\n", Quary3(1, l, r, x, 1, n));
470             }
471             else if(ops == 5){
472                 int x;
473 
474                 scanf("%d%d%d", &l, &r, &x);
475                 printf("%d\n", Quary4(1, l, r, x, 1, n));
476             }
477         }
478     }
479 
480     return 0;
481 }
```

3.4 KD Tree

3.4.1 $_{K}dTree$

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  
4  typedef long long LL;
5  
6  const LL maxn = 110000;
7  const LL INF = 1e18;
8  const LL dimension = 2;
9  LL D;
10 
11 struct Node {
12     LL d[dimension], maxpos[dimension], minpos[dimension], v, sum, lazy, cnt;
13     //以中心点d作为空间的代表，max和min分别是空间的边界
14     LL l, r;
15     bool operator < (const Node & b) const  {
16         return d[D] < b.d[D];
17     }
18     bool operator == (const Node & b) const {
19         bool ans = 1;
20         for (int i = 0; i < dimension; i++) {
21             ans &= d[i] == b.d[i];
22         }
23         return ans;
24     }
25 } p[maxn];
```

```
26
27  bool in(int x1,int y1,int x2,int y2,int X1,int Y1,int X2,int Y2) {
28      return x1<=X1&&X2<=x2&&y1<=Y1&&Y2<=y2;
29  }
30  bool out(int x1,int y1,int x2,int y2,int X1,int Y1,int X2,int Y2) {
31      return x1>X2||x2<X1||y1>Y2||y2<Y1;
32  }
33
34  struct KDT {
35      LL root, cnt, block;
36      Node tr[maxn], now;
37      //now,用来单点插入
38      void pushup(int rt) {
39          tr[rt].cnt = 1;
40          int l = tr[rt].l, r = tr[rt].r;
41          if (l) tr[rt].cnt += tr[l].cnt;
42          if (r) tr[rt].cnt += tr[r].cnt;
43          for (int i = 0; i < dimension; i++) {
44              tr[rt].maxpos[i] = tr[rt].minpos[i] = tr[rt].d[i];
45              if (l) {
46                  tr[rt].minpos[i] = min(tr[rt].minpos[i], tr[l].minpos[i]);
47                  tr[rt].maxpos[i] = max(tr[rt].maxpos[i], tr[l].maxpos[i]);
48              }
49              if (r) {
50                  tr[rt].minpos[i] = min(tr[rt].minpos[i], tr[r].minpos[i]);
51                  tr[rt].maxpos[i] = max(tr[rt].maxpos[i], tr[r].maxpos[i]);
52              }
53          }
54          tr[rt].sum = tr[l].sum + tr[r].sum + tr[rt].v;
55      }
56
57      void pushdown(int rt) {
58          if (tr[rt].lazy) {
59              int l = tr[rt].l, r = tr[rt].r;
60              if (l) {
61                  tr[l].lazy += tr[rt].lazy;
62                  tr[l].v += tr[rt].lazy;
63                  tr[l].sum += tr[l].cnt * tr[rt].lazy;
64              }
65              if (r) {
66                  tr[r].lazy += tr[rt].lazy;
67                  tr[r].v += tr[rt].lazy;
68                  tr[r].sum += tr[r].cnt * tr[rt].lazy;
69              }
70              tr[rt].lazy = 0;
71          }
72      }
73
74      LL rebuild(LL l, LL r, LL dep) { //重构
75          if (l > r) return 0;
76          D = dep; LL mid = (l + r) >> 1;
77          nth_element(p+l, p+mid, p+r+1);
78          tr[mid] = p[mid];
79          tr[mid].l = rebuild(l, mid-1, (dep+1)%dimension);
80          tr[mid].r = rebuild(mid+1, r, (dep+1)%dimension);
81
82          pushup(mid);
83          return mid;
84      }
```

```
85
86      void checkSize() {
87          if (cnt == block) {
88              for (int i = 1; i <= cnt; i++) p[i] = tr[i];
89              root = rebuild(1, cnt, 0);
90              block += 10000;
91          }
92      }
93
94      void ins(LL &rt, bool D) {//单点插入，如果没有就新开点
95          if (!rt) {
96              rt = ++cnt;
97              for (int i = 0; i < dimension; i++) tr[rt].d[i] = tr[rt].maxpos[i] = tr[rt
    ].minpos[i] = now.d[i];
98              tr[rt].v = tr[rt].sum = now.v;
99              return;
100         }
101         if (now == tr[rt]) {
102             tr[rt].v += now.v, tr[rt].sum += now.v;
103             return ;
104         }
105         pushdown(rt);
106         if (now.d[D] < tr[rt].d[D]) ins(tr[rt].l, D^1);
107         else ins(tr[rt].r, D^1);
108         pushup(rt);
109     }
110     /*
111     int x, y, p;
112     scanf("%d%d%d", &x, &y, &p);
113     Node &now = Tree.now;
114     now.d[0] = x, now.d[1] = y, now.v = p, now.sum = p;
115     Tree.ins(Tree.root, 0);
116     Tree.checkSize();
117     */
118     LL query(int rt, int x1, int y1, int x2, int y2) {
119         if(!rt) return 0;
120         LL res = 0;
121         if (out(x1, y1, x2, y2, tr[rt].minpos[0], tr[rt].minpos[1], tr[rt].maxpos[0],
    tr[rt].maxpos[1]))
122             return 0;
123         if (in(x1, y1, x2, y2, tr[rt].minpos[0], tr[rt].minpos[1], tr[rt].maxpos[0], tr
    [rt].maxpos[1]))
124             return tr[rt].sum;
125         pushdown(rt);
126         if (in(x1, y1, x2, y2, tr[rt].d[0], tr[rt].d[1], tr[rt].d[0], tr[rt].d[1]))
127             res += tr[rt].v;
128         res += query(tr[rt].l, x1, y1, x2, y2) + query(tr[rt].r, x1, y1, x2, y2);
129         pushup(rt);
130         return res;
131     }
132
133     void update(int rt, int x1, int y1, int x2, int y2, LL add) {
134         if(!rt) return ;
135         if (out(x1, y1, x2, y2, tr[rt].minpos[0], tr[rt].minpos[1], tr[rt].maxpos[0],
    tr[rt].maxpos[1]))
136             return ;
137         if (in(x1, y1, x2, y2, tr[rt].minpos[0], tr[rt].minpos[1], tr[rt].maxpos[0], tr
    [rt].maxpos[1])) {
138             tr[rt].lazy += add;
```

```
139             tr[rt].sum += add * tr[rt].cnt;
140             tr[rt].v += add;
141             return ;
142         }
143         pushdown(rt);
144         if (in(x1, y1, x2, y2, tr[rt].d[0], tr[rt].d[1], tr[rt].d[0], tr[rt].d[1])) {
145             tr[rt].v += add;
146         }
147         update(tr[rt].l, x1, y1, x2, y2, add);
148         update(tr[rt].r, x1, y1, x2, y2, add);
149         pushup(rt);
150     }
151 
152     void init() {
153         root = cnt = 0;
154         block = 10000;
155     }
156 
157     LL getdis(LL val[dimension], LL rt) { //估价函数，用来寻找区间
158         LL res = 0;
159         for(LL i = 0; i <dimension; i++) {
160             if(val[i] < tr[rt].minpos[i]) res += (tr[rt].minpos[i] - val[i]) * (tr[rt].
    minpos[i] - val[i]);
161             if(val[i] > tr[rt].maxpos[i]) res += (val[i] - tr[rt].maxpos[i]) * (val[i]
    - tr[rt].maxpos[i]);
162         }
163         return res;
164     }
165 
166     LL ans;
167 
168     void ask(LL val[dimension], LL rt) { //询问最近点
169         LL dis = 0;
170         for(LL i = 0; i < dimension; i++)
171             dis += ((tr[rt].d[i] - val[i]) * (tr[rt].d[i] - val[i]));
172         if(dis == 0) dis = INF;
173         if(dis < ans)
174             ans = dis;
175         LL dl = tr[rt].l? getdis(val, tr[rt].l) : INF;
176         LL dr = tr[rt].r? getdis(val, tr[rt].r) : INF;
177         if(dl < dr) {if(dl < ans) ask(val, tr[rt].l); if(dr < ans) ask(val, tr[rt].r);}
178         else {if(dr < ans) ask(val, tr[rt].r); if(dl < ans) ask(val, tr[rt].l);}
179     }
180 } Tree;
181 
182 int n, m;
183 int id;
184 int L[maxn], R[maxn];
185 int dep[maxn];
186 vector<int> G[maxn];
187 
188 void dfs(int u, int fa) {
189     dep[u] = dep[fa] + 1;
190     L[u] = ++id;
191     for (auto v:G[u]) {
192         if (v == fa) continue;
193         dfs(v, u);
194     }
195     R[u] = id;
```

```
}

int main() {
    while (~scanf("%d%d", &n, &m)) {
        Tree.init();
        id = 0;
        for (int i = 0; i < n - 1; i++) {
            int u, v;
            scanf("%d%d", &u, &v);
            G[u].push_back(v);
            G[v].push_back(u);
        }
        dep[0] = -1;
        dfs(1, 0);
        for (int i = 1; i <= n; i++) {
            Node &now = p[i];
            now.d[0] = L[i];
            now.d[1] = dep[i];
        }
        Tree.root = Tree.rebuild(1, n, 0);
        while (m--) {
            int tp, l, x;
            scanf("%d", &tp);
            if (tp == 1){
                scanf("%d%d", &l, &x);
                Tree.update(Tree.root, 1, l, n, l, x);
            } else {
                scanf("%d", &x);
                printf("%lld\n", Tree.query(Tree.root, L[x], 1, R[x], n));
            }
        }
    }
    return 0;
}
```

3.5 Sparse Table

```
const int maxn = "Edit";
int dp[maxn][20];
int a[maxn];
void init(int n)
{
    for (int i = 1; i <= n; i++) dp[i][0] = a[i];
    for (int j = 1; (1 << j) <= n; j++)
        for (int i = 1; i + (1 << j) - 1 <= n; i++)
            dp[i][j] = max(dp[i][j - 1], dp[i + (1 << (j - 1))][j - 1]);
}
// 返回[l,r]最大值
int rmq(int l, int r, int op)
{
    int k = 31 - __builtin_clz(r - l + 1);
    return max(dp[l][k], dp[r - (1 << k) + 1][k]);
}
```

二维 RMQ

```
void init(int n, int m)
{
    for (int i = 0; (1 << i) <= n; i++)
```

```
        for (int j = 0; (1 << j) <= m; j++)
        {
            if (i == 0 && j == 0) continue;
            for (int row = 1; row + (1 << i) - 1 <= n; row++)
                for (int col = 1; col + (1 << j) - 1 <= m; col++)
                    if (i)
                        dp[row][col][i][j] = max(dp[row][col][i - 1][j],
                                                 dp[row + (1 << (i - 1))][col][i - 1][j]);
                    else
                        dp[row][col][i][j] = max(dp[row][col][i][j - 1],
                                                 dp[row][col + (1 << (j - 1))][i][j - 1]);
        }
}
int rmq(int x1, int y1, int x2, int y2)
{
    int kx = 31 - __builtin_clz(x2 - x1 + 1);
    int ky = 31 - __builtin_clz(y2 - y1 + 1);
    int m1 = dp[x1][y1][kx][ky];
    int m2 = dp[x2 - (1 << kx) + 1][y1][kx][ky];
    int m3 = dp[x1][y2 - (1 << ky) + 1][kx][ky];
    int m4 = dp[x2 - (1 << kx) + 1][y2 - (1 << ky) + 1][kx][ky];
    return max({m1, m2, m3, m4});
}
```

3.6 Heavy-Light Decomposition

3.6.1 Hdu5044

```
#define maxn 100010
struct Node
{
    int to, ne;
    int fa, size, son, dep;     /**  dfs()
                                  *  fa: parent node
                                  *  size: cnt_node
                                  *  son: heavy
                                  *  dep: depth of current node
                                  */
    int top, pos;               /**  _dfs()
                                  *  top: top node
                                  *  pos: node's index in *array* seq[]
                                  */
}node[maxn];
int totw;                       /// = 1, for(int i = 1; i < totw; ++ i){}
int seq[maxn];
int head[maxn];
int top;

void add(int from, int to){
    node[top].to = to;
    node[top].ne = head[from];
    head[from] = top;
    top ++;
}

int dfs(int now, int prev = -1, int depth = 0){
    node[now].fa = prev;
    node[now].dep = depth;
    int ret = 1;
```

```
32      int cur_size = -1;
33      for(int i = head[now]; i != -1; i = node[i].ne){
34          int nxt = node[i].to;
35          if(nxt == prev)continue;
36
37          int son_size = dfs(nxt, now, depth + 1);
38          if(cur_size < son_size){
39              cur_size = son_size;
40              node[now].son = nxt;
41          }
42
43          ret += son_size;
44      }
45      node[now].size = ret;
46      return ret;
47  }
48
49  void _dfs(int now, int prev = -1){
50      node[now].pos = totw;
51      seq[totw ++] = now;
52      if(node[now].top == -1){
53          node[now].top = now;
54      }
55      if(node[now].son != -1){
56          node[node[now].son].top = node[now].top;
57          _dfs(node[now].son, now);
58      }
59      for(int i = head[now]; i != -1; i = node[i].ne){
60          int nxt = node[i].to;
61          if(nxt == prev || nxt == node[now].son)
62              continue;
63          _dfs(nxt, now);
64      }
65  }
66
67  void build(int root = 1){
68      dfs(root);
69      _dfs(root);
70  }
71
72  int lca(int x, int y){
73      while(true){
74          if(node[x].top == node[y].top){
75              return node[x].dep <= node[y].dep ? x : y;
76          }
77          if(node[node[x].top].dep >= node[node[y].top].dep){
78              x = node[node[x].top].fa;
79          }
80          else {
81              y = node[node[y].top].fa;
82          }
83      }
84  }
85
86  void update(){
87      if(/*update node value (node[l] to node[r], node.val += k) */){
88          while(1){
89              int f1 = node[l].top, f2 = node[r].top;
90              if(f1 == f2){
```

```
                if(node[l].pos > node[r].pos)swap(l, r);
                updateN(node[l].pos, node[r].pos, k);
                break;
            }
            if(node[f1].dep > node[f2].dep){
                updateN(node[f1].pos, node[l].pos, k);
                l = node[f1].fa;
            }
            else {
                updateN(node[f2].pos, node[r].pos, k);
                r = node[f2].fa;
            }
        }
    }
    else if(/*update edge value (edges between node[l] to node[r], edge.val += k)*/){
        while(1){
            int f1 = node[l].top, f2 = node[r].top;
            if(f1 == f2){
                if(l == r)break;
                if(node[l].pos > node[r].pos)swap(l, r);
                updateE(node[l].pos + 1, node[r].pos, k);
                break;
            }
            if(node[f1].dep > node[f2].dep){
                updateE(node[f1].pos, node[l].pos, k);
                l = node[f1].fa;
            }
            else {
                updateE(node[f2].pos, node[r].pos, k);
                r = node[f2].fa;
            }
        }
    }
}

void ini(){
    memset(head, -1, sizeof(head));
    top = 0;
    totw = 1;
}
#undef maxn
```

3.6.2 Slpfedge

```
#include <bits/stdc++.h>
using namespace std;
#define maxn 1000000

struct edage{
    int x,y,val;
}ed[maxn];

struct node{
    int fa;
    int dep;
    int siz;
    int son;//和该节点在同一重链上的节点
    int top;
```

```
    int id;
}no[maxn];
int num;
int val[maxn];

vector<int> v[maxn];
void dfs1(int u,int f,int dep){
    no[u].dep=dep;
    no[u].siz=1;
    no[u].son=0;
    no[u].fa=f;
    for(int i=0;i<v[u].size();i++){
        int ff=v[u][i];
        if(ff==f) continue;
        dfs1(ff,u,dep+1);
        no[u].siz+=no[ff].siz;
        if(no[no[u].son].siz<no[ff].siz) no[u].son=ff;
    }
}
void dfs2(int u,int tp){
    no[u].top=tp;
    no[u].id=++num;
    if(no[u].son) dfs2(no[u].son,tp);
    for(int i=0;i<v[u].size();i++){
        int ff=v[u][i];
        if(ff==no[u].fa||ff==no[u].son) continue;
        dfs2(ff,ff);
    }
}

struct node2{
    int l;
    int r;
    int val;
    int lazy;
}tree[4*maxn];

void pushup(int x){
    tree[x].val=max(tree[x<<1].val,tree[x<<1|1].val);
}

void build(int rt,int l,int r){
    tree[rt].l=l;
    tree[rt].r=r;
    if(l==r){
        tree[rt].val=val[l];
        return ;
    }
    int mid=(l+r)/2;
    build(rt<<1,l,mid);
    build(rt<<1|1,mid+1,r);
    pushup(rt);
}
void update(int rt,int pos,int va){
    int l=tree[rt].l;
    int r=tree[rt].r;
    if(l==r){
        tree[rt].val=va;
        return ;
```

```
74      }
75      int mid=(l+r)/2;
76      if(pos<=mid) update(rt<<1,pos,va);
77      else update(rt<<1|1,pos,va);
78      pushup(rt);
79  }
80
81  int query(int rt,int s,int t){
82      int l=tree[rt].l;
83      int r=tree[rt].r;
84      if(l==s&&t==r){
85          return tree[rt].val;
86      }
87      int mid=(l+r)/2;
88      if(t<=mid) return query(rt<<1,s,t);
89      else if(s>mid) return query(rt<<1|1,s,t);
90      return max(query(rt<<1,s,mid),query(rt<<1|1,mid+1,t));
91  }
92  int youth(int u,int v){
93      int tp1=no[u].top,tp2=no[v].top;
94      int ans=0;
95      while(tp1!=tp2){
96          if(no[tp1].dep<no[tp2].dep){
97              swap(tp1,tp2);
98              swap(u,v);
99          }
100         ans=max(query(1,no[tp1].id,no[u].id),ans);
101        // printf("1\n");
102         u=no[tp1].fa;
103         tp1=no[u].top;
104     }
105     if(u==v) return ans;
106     if(no[u].dep>no[v].dep) swap(u,v);
107     ans=max(query(1,no[no[u].son].id,no[v].id),ans);
108     //printf("2\n");
109     return ans;
110 }
111 void Clear(int n){
112     for(int i=1;i<=n;i++){
113         v[i].clear();
114     }
115 }
116
117 int main(){
118     int t;
119     scanf("%d",&t);
120     while(t--){
121         int n;
122         scanf("%d",&n);
123         for(int i=1;i<n;i++){
124
125             scanf("%d%d%d",&ed[i].x,&ed[i].y,&ed[i].val);
126             v[ed[i].x].push_back(ed[i].y);
127             v[ed[i].y].push_back(ed[i].x);
128         }
129         //printf("a");
130         dfs1(1,0,1);
131         dfs2(1,1);
132
```

```
        for(int i=1;i<n;i++){
            if(no[ed[i].x].dep<no[ed[i].y].dep) swap(ed[i].x,ed[i].y);
            val[no[ed[i].x].id]=ed[i].val;
        }

        build(1,1,num);

        char s[200];
        while(~scanf("%s",s)&&s[0]!='D'){
            int x,y;
            scanf("%d%d",&x,&y);
            if(s[0]=='Q'){
                printf("%d\n",youth(x,y));
            }
            if(s[0]=='C'){
                update(1,no[ed[x].x].id,y);
            }

        }
        Clear(n);
    }

    return 0;
}
```

3.6.3 Slpf

```
#include <bits/stdc++.h>
using namespace std;
const int N=50010;
int tot=1,num=0,n,m,point[N],next[N*10],pos[N]={0},siz[N]={0},v[N]={0};
int belong[N]={0},fa[N][20]={0},deep[N]={0},map[N];
bool use[N];
char ch[10];
struct S{
    int maxn,sum;
}tr[N*4];
struct C{
    int st,en;
}aa[N*10];
inline void add(int i,int j)
{
    tot+=1;next[tot]=point[i];point[i]=tot;
    aa[tot].st=i;aa[tot].en=j;
    tot+=1;next[tot]=point[j];point[j]=tot;
    aa[tot].st=j;aa[tot].en=i;
}
inline void dfs1(int x)
{
    int i;
    siz[x]=1;
    use[x]=false;
    for(i=1;i<=14;++i){
        if(deep[x]<(1<<i)) break;
        fa[x][i]=fa[fa[x][i-1]][i-1];
    }
    for(i=point[x];i;i=next[i])
      if(use[aa[i].en]){
```

```
32          fa[aa[i].en][0]=x;
33          deep[aa[i].en]=deep[x]+1;
34          dfs1(aa[i].en);
35          siz[x]+=siz[aa[i].en];
36        }
37  }
38  inline void dfs2(int x,int y)
39  {
40      int i,k=0;
41      num+=1;
42      pos[x]=num;
43      map[num]=x;
44      belong[x]=y;
45      for(i=point[x];i;i=next[i])
46        if(deep[aa[i].en]>deep[x]&&siz[k]<siz[aa[i].en])
47          k=aa[i].en;
48      if(k==0) return ;
49      dfs2(k,y);
50      for(i=point[x];i;i=next[i])
51        if(deep[aa[i].en]>deep[x]&&k!=aa[i].en)
52          dfs2(aa[i].en,aa[i].en);
53  }
54  inline int LCA(int x,int y)
55  {
56      int i;
57      if(deep[x]<deep[y])swap(x,y);
58      int t=deep[x]-deep[y];
59      for(i=0;i<=14;++i)
60          if(t&(1<<i))x=fa[x][i];
61      for(i=14;i>=0;--i)
62          if(fa[x][i]!=fa[y][i])
63          {x=fa[x][i];y=fa[y][i];}
64      if(x==y)return x;
65      else return fa[x][0];
66  }
67  #define mid (l+r)/2
68  #define L k<<1,l,mid
69  #define R k<<1|1,mid+1,r
70  inline void build(int k,int l,int r)
71  {
72      if(l==r){
73          tr[k].maxn=tr[k].sum=v[map[l]];
74          return ;
75      }
76      build(L);build(R);
77      tr[k].maxn=max(tr[k<<1].maxn,tr[k<<1|1].maxn);
78      tr[k].sum=tr[k<<1].sum+tr[k<<1|1].sum;
79  }
80  void insert(int k,int l,int r,int x,int y)
81  {
82      if(l==r&&l==x){
83          tr[k].sum=tr[k].maxn=y;
84          return ;
85      }
86      if(x<=mid) insert(L,x,y);
87      else insert(R,x,y);
88      tr[k].maxn=max(tr[k<<1].maxn,tr[k<<1|1].maxn);
89      tr[k].sum=tr[k<<1].sum+tr[k<<1|1].sum;
90  }
```

```
inline int qurry(int k,int l,int r,int x,int y,int kind)
{
    int sum=0,maxn=-210000000;
    if(x<=l&&y>=r) return kind==0?tr[k].maxn:tr[k].sum;
    if(x<=mid){
        if(kind==0) maxn=max(maxn,qurry(L,x,y,kind));
        else sum+=qurry(L,x,y,kind);
    }
    if(y>mid){
        if(kind==0) maxn=max(maxn,qurry(R,x,y,kind));
        else sum+=qurry(R,x,y,kind);
    }
    return kind==0?maxn:sum;
}
inline int ask(int x,int y,int kind)
{
    int sum=0,maxn=-210000000;
    while(belong[x]!=belong[y]){
        if(kind==0) maxn=max(maxn,qurry(1,1,n,pos[belong[x]],pos[x],kind));
        else sum+=qurry(1,1,n,pos[belong[x]],pos[x],kind);
        x=fa[belong[x]][0];
    }
    if(kind==0) maxn=max(maxn,qurry(1,1,n,pos[y],pos[x],kind));
    else sum+=qurry(1,1,n,pos[y],pos[x],kind);
    return kind==0?maxn:sum;
}
int main()
{
    int i,j,x,y;
    scanf("%d",&n);
    memset(use,1,sizeof(use));
    for(i=1;i<n;++i){
        scanf("%d%d",&x,&y);
        add(x,y);
    }
    for(i=1;i<=n;++i) scanf("%d",&v[i]);
    dfs1(1);
    dfs2(1,1);
    build(1,1,n);
    scanf("%d",&m);
    while(m--){
        scanf("%*c%s%d%d",&ch,&x,&y);
        if(ch[0]=='C') insert(1,1,n,pos[x],y),v[x]=y;
        else{
            int lca=LCA(x,y);
            if(ch[1]=='M') printf("%d\n",max(ask(x,lca,0),ask(y,lca,0)));
            else printf("%d\n",ask(x,lca,1)+ask(y,lca,1)-v[lca]);
        }
    }
}
```

3.7 Link-Cut Tree

3.7.1 LCT

动态维护一个森林

```
const int maxn = "Edit";
struct LCT
```

```
3  {
4      int val[maxn], sum[maxn]; // 基于点权
5      int rev[maxn], ch[maxn][2], fa[maxn];
6      int stk[maxn];
7      inline void init(int n)
8      { // 初始化点权
9          for (int i = 1; i <= n; i++) scanf("%d", val + i);
10         for (int i = 1; i <= n; i++)
11             fa[i] = ch[i][0] = ch[i][1] = rev[i] = 0;
12     }
13     inline bool isroot(int x) { return ch[fa[x]][0] != x && ch[fa[x]][1] != x; }
14     inline bool get(int x) { return ch[fa[x]][1] == x; }
15     void pushdown(int x)
16     {
17         if (!rev[x]) return;
18         swap(ch[x][0], ch[x][1]);
19         if (ch[x][0]) rev[ch[x][0]] ^= 1;
20         if (ch[x][1]) rev[ch[x][1]] ^= 1;
21         rev[x] ^= 1;
22     }
23     void pushup(int x) { sum[x] = val[x] + sum[ch[x][0]] + sum[ch[x][1]]; }
24     void rotate(int x)
25     {
26         int y = fa[x], z = fa[fa[x]], d = get(x);
27         if (!isroot(y)) ch[z][get(y)] = x;
28         fa[x] = z;
29         ch[y][d] = ch[x][d ^ 1], fa[ch[y][d]] = y;
30         ch[x][d ^ 1] = y, fa[y] = x;
31         pushup(y), pushup(x);
32     }
33     void splay(int x)
34     {
35         int top = 0;
36         stk[++top] = x;
37         for (int i = x; !isroot(i); i = fa[i]) stk[++top] = fa[i];
38         for (int i = top; i; i--) pushdown(stk[i]);
39         for (int f; !isroot(x); rotate(x))
40             if (!isroot(f = fa[x])) rotate(get(x) == get(f) ? f : x);
41     }
42     void access(int x)
43     {
44         for (int y = 0; x; y = x, x = fa[x]) splay(x), ch[x][1] = y, pushup(x);
45     }
46     int find(int x)
47     {
48         access(x), splay(x);
49         while (ch[x][0]) x = ch[x][0];
50         return x;
51     }
52     void makeroot(int x) { access(x), splay(x), rev[x] ^= 1; }
53     void link(int x, int y) { makeroot(x), fa[x] = y, splay(x); }
54     void cut(int x, int y) { makeroot(x), access(y), splay(y), fa[x] = ch[y][0] = 0; }
55     void update(int x, int v) { val[x] = v, access(x), splay(x); }
56     int query(int x, int y)
57     {
58         makeroot(y), access(x), splay(x);
59         return sum[x];
60     }
61 };
```

3.8 Virtual Tree

3.8.1 VTree

```
/**
- □□□□ÿ¸□□□□□Ⓕ²¢□□´□□dfs□□□¹□□□□□□□□
- ½«µ□1¸□□□µ½□µ±□□¬¿ª□¹¹½¨□□□□
- ö¾□½□□□¸□□u£¬¾□□□u□□□¶¥µ□vµĹ«¹²□□□□lca
- ¼□□□□□□□¶¥□ · ½µ□□□w£¨□□□□□□1□□¸□□□□□¹□□□□²½£©£¬□□wµ□□□□□□□□lca¾□□v□□wl□□□²¢¬□□□¯µ□v£¬□
  ´´□□½£¬ · □□□□½□□□²½
- □□lca²»□□±□µ□v£¬□□ô¾□□lca°□vl±□¬°□vµ¯³□□□lca³□□□¶¥□□°£□¨□□□□□□□□□□□□□□□□□¶¥û□□□□□lca
  □□ô¾□□□Ⓕ£¬ · □□□□□□□□□□□□
- □□□□u□□□□□□
- 3²½□«□□□ö¾□□¸□Ⓕµ½ö¾□□□□□□□□□□
- °□□¶¥v□□□¶¥□ · ½µ□□□wl±²¢¬□□□□vµ¯µ□□□ô□□□µ½□□□□□□¸□□
- □□□□□µ□□□□□□□□□□□□□□
*/
const int maxn = "Edit";
vector<int> vtree[maxn];
void build(vector<int>& vec)
{
    sort(vec.begin(), vec.end(), [&](int x, int y) { return dfn[x] < dfn[y]; });
    static int s[maxn];
    int top = 0;
    s[top] = 0;
    vtree[0].clear();
    for (auto& u : vec){
        int vlca = lca(u, s[top]);
        vtree[u].clear();
        if (vlca == s[top])
            s[++top] = u;
        else{
            while (top && dep[s[top - 1]] >= dep[vlca]){
                vtree[s[top - 1]].push_back(s[top]);
                top--;
            }
            if (s[top] != vlca){
                vtree[vlca].clear();
                vtree[vlca].push_back(s[top--]);
                s[++top] = vlca;
            }
            s[++top] = u;
        }
    }
    for (int i = 0; i < top; ++i) vtree[s[i]].push_back(s[i + 1]);
}
```

3.9 Cartesian Tree

```
const int maxn = "Edit";
int lson[maxn], rson[maxn], fa[maxn];
void build(int n)
{
    stack<int> s;
    for (int i = 0; i < n; i++)
    {
        int last = -1;
        while (!s.empty() && a[i] > a[s.top()]) last = s.top(), s.pop();
```

```
        if (!s.empty()) rson[s.top()] = i, fa[i] = s.top();
        lson[i] = last;
        if (~last) fa[last] = i;
        s.push(i);
    }
}
```

3.10 Block

3.10.1 Block

```
const int N = 100010;

int ma[N];
int bl[N];  /// 第 i 个元素属于的块的编号
int l[N];   /// 第 i 个块的左边界
int r[N];
int block;
int num;

void build()
{
    block = sqrt(n);
    num = n / block;
    if(n % block){
        num ++;
    }
    for(int i = 1; i <= num; i ++){
        l[i] = (i - 1) * block + 1;
        r[i] = i * block;
    }
    r[num] = n;
    for(int i = 1; i <= n; i ++){
        bl[i] = (i - 1) / block + 1;
    }
}
```

3.10.2 Mo

```
struct Quary
{
    int l;
    int r;
    int num;
//    bool operator < (const Quary node) const
//    {
//        return (pos[l] < pos[node.l])||(pos[l] == pos[node.l] && (pos[l] & 1 ? r < node.r : r > node.r));
//    }
};

Quary quary[N];
int pos[N];
int ans[N];
int block;
int n, m;
int l, r;

```

```
19 bool cmp(Quary a, Quary b)
20 {
21     if(pos[a.l] == pos[b.l]){
22         return a.r < b.r;
23     }
24     else{
25         return pos[a.l] < pos[b.l];
26     }
27 }
28
29 void add(int x)
30 {
31     ///
32 }
33
34 void del(int x)
35 {
36     ///
37 }
38
39 int main(int argc, char const *argv[])
40 {
41     while(scanf("%d%d", &n, &m) == 2){
42         memset(cnt, 0, sizeof(cnt));
43         block = sqrt(n);
44         for(int i = 1; i <= n; i ++){
45             scanf("%d", &ma[i]);
46             pos[i] = i / block;
47         }
48         for(int i = 1; i <= m; i ++){
49             int num, l, r;
50
51             scanf("%d%d%d%d", &quary[i].opt, &quary[i].l, &quary[i].r, &quary[i].x);
52             quary[i].num = i;
53         }
54         sort(quary + 1, quary + m + 1, cmp);
55         l = 1;
56         r = 0;
57         for(int i = 1; i <= m; i ++){
58             while(l > quary[i].l){
59                 l --;
60                 add(l);
61             }
62             while(r < quary[i].r){
63                 r ++;
64                 add(r);
65             }
66             while(l < quary[i].l){
67                 del(l);
68                 l ++;
69             }
70             while(r > quary[i].r){
71                 del(r);
72                 r --;
73             }
74             ans[quary[i].num] = ask(quary[i]);
75         }
76     }
77
```

```
78      return 0;
79  }
```

3.11 Kbaba

3.11.1 KthNumber

```
1   /// 不带修改区间第 K 大
2   /// 区间第 k 小 无不存在数据特判  节点范围跟随函数传参
3   const int N = 100010;
4
5   struct Three
6   {
7       int sum;
8       int son[2];
9   };
10
11  Three tree[N * 20];
12  int ma[N];
13  int n, m;
14  int root[N];
15  int tot;
16
17  void Up(int x)
18  {
19      tree[x].sum = tree[tree[x].son[0]].sum + tree[tree[x].son[1]].sum;
20  }
21
22  void Build(int &x, int l, int r)
23  {
24      tot ++;
25      x = tot;
26      if(l == r){
27          tree[x].sum = 0;
28      }
29      else{
30          int mid;
31
32          mid = l + ((r - l) >> 1);
33          Build(tree[x].son[0], l, mid);
34          Build(tree[x].son[1], mid + 1, r);
35          Up(x);
36      }
37  }
38
39  void Update(int &x, int pre, int pos, int ll, int rr)
40  {
41      tot ++;
42      x = tot;
43      tree[x] = tree[pre];
44      if(ll == rr && ll == pos){
45          tree[x].sum ++;
46      }
47      else{
48          int mid;
49
50          mid = ll + ((rr - ll) >> 1);
51          if(pos <= mid){
52              Update(tree[x].son[0], tree[pre].son[0], pos, ll, mid);
```

```
53          }
54          else{
55              Update(tree[x].son[1], tree[pre].son[1], pos, mid + 1, rr);
56          }
57          Up(x);
58      }
59  }
60
61  int Quary(int l, int r, int x, int ll, int rr)
62  {
63      // printf("%d %d %d\n", ll, rr, x);
64      if(ll == rr){
65          return ll;
66      }
67      else{
68          int mid;
69          int t;
70
71          t = tree[tree[r].son[0]].sum - tree[tree[l].son[0]].sum;
72          mid = ll + ((rr - ll) >> 1);
73          // printf(" %d - %d = %d\n", tree[tree[r].son[0]].sum, tree[tree[l].son[0]].sum
      , t);
74          if(t >= x){
75              // printf("gl\n");
76              return Quary(tree[l].son[0], tree[r].son[0], x, ll, mid);
77          }
78          else{
79              // printf("gr\n");
80              return Quary(tree[l].son[1], tree[r].son[1], x - t, mid + 1, rr);
81          }
82      }
83  }
84
85  int main(int argc, char const *argv[])
86  {
87      int ncase;
88
89      scanf("%d", &ncase);
90      while(ncase --){
91          tot = 0;
92          id.clear();
93          scanf("%d%d", &n, &m);
94          for(int i = 1; i <= n; i ++){
95              scanf("%d", &ma[i]);
96              id.pb(ma[i]);
97          }
98          sort(id.begin(), id.end());
99          id.erase(unique(id.begin(), id.end()), id.end());
100         Build(root[0], 1, n);
101         for(int i = 1; i <= n; i ++){
102             // printf("%d\n", Getid(ma[i]));
103             Update(root[i], root[i - 1], Getid(ma[i]), 1, n);
104         }
105         for(int i = 1; i <= m; i ++){
106             int l, r, x;
107
108             scanf("%d%d%d", &l, &r, &x);
109             printf("%d\n", id[Quary(root[l - 1], root[r], x, 1, n) - 1]);
110         }
```

```
111     }
112
113     return 0;
114 }
115
116 /// 整体二分
117 #include <bits/stdc++.h>
118 #define LL long long
119 using namespace std;
120
121 const int N = 200020;
122 const LL INF = 0x3f3f3f3f3f3f3f3f;
123
124 struct Node
125 {
126     int kind;
127     int pos;
128     int l;
129     int r;
130     int x;
131 };
132
133 Node node[N];
134 Node now[N];
135 int bits[N];
136 LL ans[N];
137 int n, m;
138 int cnt;
139
140 void Alldoubledive(int ql, int qr, LL l, LL r)
141 {
142     if(l == r){
143         for(int i = ql; i <= qr; i ++){
144             if(node[i].kind == 2){
145                 ans[node[i].pos] = l;
146             }
147         }
148     }
149     else{
150         int mid;
151         int now1, now2;
152
153         mid = l + ((r - l) >> 1);
154         now1 = ql;
155         now2 = qr;
156         for(int i = ql; i <= qr; i ++){
157             if(node[i].kind == 1){
158                 if(node[i].l <= mid){
159                     Add(node[i].pos, 1);
160                     now[now1] = node[i];
161                     now1 ++;
162                 }
163                 else{
164                     now[now2] = node[i];
165                     now2 --;
166                 }
167             }
168             else{
169                 int t;
```

```
170
171                 t = Sum(node[i].r) - Sum(node[i].l - 1);
172                 if(t >= node[i].x){
173                     now[now1] = node[i];
174                     now1 ++;
175                 }
176                 else{
177                     node[i].x -= t;
178                     now[now2] = node[i];
179                     now2 --;
180                 }
181             }
182         }
183         for(int i = ql; i < now1; i ++){
184             if(now[i].kind == 1){
185                 Add(now[i].pos, -1);
186             }
187         }
188         reverse(now + now2 + 1, now + qr + 1);  /// 顺序!
189         for(int i = ql; i <= qr; i ++){
190             node[i] = now[i];
191         }
192         if(now1 != ql){
193             Alldoubledive(ql, now1 - 1, l, mid);
194         }
195         if(now2 != qr){
196             Alldoubledive(now2 + 1, qr, mid + 1, r);
197         }
198     }
199 }
200
201 int main(int argc, char const *argv[])
202 {
203     int ncase;
204
205     scanf("%d", &ncase);
206     while(ncase --){
207         memset(bits, 0, sizeof(bits));
208         scanf("%d%d", &n, &m);
209         cnt = 1;
210         for(int i = 1; i <= n; i ++){
211             scanf("%d", &node[cnt].l);
212             node[cnt].pos = i;
213             node[cnt].kind = 1;
214             cnt ++;
215         }
216         for(int i = 1; i <= m; i ++){
217             scanf("%d%d%d", &node[cnt].l, &node[cnt].r, &node[cnt].x);
218             node[cnt].pos = i;
219             node[cnt].kind = 2;
220             cnt ++;
221         }
222         Alldoubledive(1, cnt - 1, -INF, INF);
223         for(int i = 1; i <= m; i ++){
224             if(ans[i] == INF){  /// k 大于区间长度
225                 ans[i] = 0;
226             }
227             printf("%lld\n", ans[i]);
228         }
```

```
229     }
230
231     return 0;
232 }
233
234
235 /// 带修改区间第 k 大
236 /// BZOJ 1901 整体二分
237 #include <bits/stdc++.h>
238 #define mid (((l) + (r)) / 2)
239 #define lowbit(x) (x & (-x))
240 using namespace std;
241
242 inline int ReadInt() {
243     static int ch, n;
244     ch = getchar(), n = 0;
245     while (!isdigit(ch)) ch = getchar();
246     while (isdigit(ch)) n = (n << 3) + (n << 1) + ch - '0', ch = getchar();
247     return n;
248 }
249
250 struct oper {
251     int type, idx, l, r, k;
252 }t;
253
254 const int maxn = 10000 + 3, maxm = 10000 + 3;
255 int c[maxn], a[maxn], ans[maxm], n, m, cnt = 0;
256
257 inline void add(int x, int v) {
258     x++;
259     while (x <= n) {
260         c[x] += v;
261         x += lowbit(x);
262     }
263 }
264 inline int sum(int x) {
265     x++; int ret = 0;
266     while (x > 0) {
267         ret += c[x];
268         x -= lowbit(x);
269     }
270     return ret;
271 }
272
273 //操作序列, 答案区间 [l, r)
274 void solve(queue<oper> &q, int l, int r) {
275     if (q.empty()) return;
276     if (l >= r) return;
277     if (r - l == 1) {
278         while (!q.empty()) {
279             t = q.front(); q.pop();
280             if (t.type == 3) ans[t.idx] = l;
281         }
282         return;
283     }
284     queue<oper> q1, q2;
285     queue<int> idp, idn;
286     while (!q.empty()) {
287         t = q.front(); q.pop();
```

```
288         if (t.type == 1) {
289             if (t.k < mid) add(t.idx, 1), idp.push(t.idx), q1.push(t);
290             else q2.push(t);
291         }else if (t.type == 2) {
292             if (t.k < mid) add(t.idx, -1), idn.push(t.idx), q1.push(t);
293             else q2.push(t);
294         } else {
295             int v = sum(t.r) - sum(t.l - 1);
296             if (v + 1 <= t.k) t.k -= v, q2.push(t);
297             else q1.push(t);
298         }
299     }
300     while (!idp.empty()) {
301         add(idp.front(), -1);
302         idp.pop();
303     }
304     while (!idn.empty()) {
305         add(idn.front(), 1);
306         idn.pop();
307     }
308     solve(q1, l, mid);
309     solve(q2, mid, r);
310 }
311
312 char str[2];
313 int main() {
314     queue<oper> q;
315     n = ReadInt(), m = ReadInt();
316     //type 1 +
317     //type 2 -
318     //type 3 Query
319     for (int i = 0; i < n; ++i) {
320         t.type = 1, a[i] = t.k = ReadInt();
321         t.idx = i;
322         q.push(t);
323     }
324     for (int i = 0; i < m; ++i) {
325         scanf("%s", str);
326         if (str[0] == 'Q') {
327             t.type = 3, t.l = ReadInt() - 1, t.r = ReadInt() - 1, t.k = ReadInt(), t.
    idx = cnt++;
328             q.push(t);
329         } else if (str[0] == 'C') {
330             t.type = 2, t.idx = ReadInt() - 1, t.k = a[t.idx];
331             q.push(t);
332             t.type = 1, a[t.idx] = t.k = ReadInt();
333             q.push(t);
334         }
335     }
336     memset(c, 0, sizeof c);
337     solve(q, 0, 1e9 + 3);
338     for (int i = 0; i < cnt; ++i)
339         printf("%d\n", ans[i]);
340     return 0;
341 }
342
343
344 /// 树上路径第 K 大
345 /// BZOJ 2588
```

```
/// 现在第 K 大跑到树上来了嘿嘿嘿
#include <bits/stdc++.h>
#define mid (((l) + (r)) / 2)
using namespace std;

inline int ReadInt() {
    static int n, ch;
    n = 0, ch = getchar();
    while (!isdigit(ch)) ch = getchar();
    while (isdigit(ch)) n = (n << 3) + (n << 1) + ch - '0', ch = getchar();
    return n;
}
typedef long long ll;

const int maxn = 100000 + 3;
struct SegNode *pit, *null;
struct SegNode {
    SegNode *ls, *rs;
    int s;
    inline void maintain() {
        s = ls->s + rs->s;
    }
    SegNode(): ls(null), rs(null), s(0) {}
}pool[maxn * 18], *root[maxn];

void init() {
    pit = pool;
    null = new SegNode();
    null->ls = null, null->rs = null;
}

SegNode* modify(const SegNode *o, int l, int r, int v) {
    if (l >= r) return null;
    SegNode *ne = pit++;
    *ne = *o;
    if (r - l == 1)
        ne->s++;
    else {
        if (v < mid) ne->ls = modify(ne->ls, l, mid, v);
        else ne->rs = modify(ne->rs, mid, r, v);
        ne->maintain();
    }
    return ne;
}

vector<int> G[maxn], Ws;
int n, m, w[maxn], ancestor[maxn][18], depth[maxn];

void compress() {
    for (int i = 0; i < n; ++i)
        Ws.push_back(w[i]);
    sort(Ws.begin(), Ws.end());
    Ws.erase(unique(Ws.begin(), Ws.end()), Ws.end());
    for (int i = 0; i < n; ++i)
        w[i] = lower_bound(Ws.begin(), Ws.end(), w[i]) - Ws.begin();
}

void process() {
    for (int w = 1; (1 << w) < n; ++w)
```

```
405         for (int i = 0; i < n; ++i) if (depth[i] - (1 << w) >= 0)
406             ancestor[i][w] = ancestor[ancestor[i][w - 1]][w - 1];
407 }
408
409 int LCA(int a, int b) {
410     if (depth[a] < depth[b]) swap(a, b);
411     int lim = log2(depth[a]);
412     for (int i = lim; i >= 0; --i)
413         if (depth[a] - (1 << i) >= depth[b])
414             a = ancestor[a][i];
415     if (a == b) return a;
416     for (int i = lim; i >= 0; --i)
417         if (depth[a] - (1 << i) >= 0 && ancestor[a][i] != ancestor[b][i]) {
418             a = ancestor[a][i];
419             b = ancestor[b][i];
420         }
421     return ancestor[a][0];
422 }
423
424 int query(const SegNode *a, const SegNode *b, const SegNode *c, const SegNode *d, int l
    , int r, int k) {
425     if (r - l == 1) return Ws[l];
426     int s = a->ls->s + b->ls->s - c->ls->s - d->ls->s;
427     if (k <= s) return query(a->ls, b->ls, c->ls, d->ls, l, mid, k);
428     else return query(a->rs, b->rs, c->rs, d->rs, mid, r, k - s);
429 }
430
431 void dfs(const SegNode *o, int u, int fa) {
432     ancestor[u][0] = fa, depth[u] = fa == -1 ? 0 : depth[fa] + 1;
433     root[u] = modify(o, 0, Ws.size(), w[u]);
434     for (int i = 0; i < (int)G[u].size(); ++i) {
435         int v = G[u][i];
436         if (v != fa) dfs(root[u], v, u);
437     }
438 }
439
440 int main() {
441     init();
442     n = ReadInt(), m = ReadInt();
443     for (int i = 0; i < n; ++i)
444         w[i] = ReadInt();
445     compress();
446     for (int i = 0; i < n - 1; ++i) {
447         int f = ReadInt() - 1, t = ReadInt() - 1;
448         G[f].push_back(t);
449         G[t].push_back(f);
450     }
451     dfs(null, 0, -1);
452     process();
453     int lastAns = 0;
454     while (m--) {
455         int u = (ReadInt() ^ lastAns) - 1, v = ReadInt() - 1, k = ReadInt();
456         int lca = LCA(u, v);
457         printf("%d", lastAns = query(root[u], root[v], root[lca], lca == 0 ? null :
    root[ancestor[lca][0]], 0, Ws.size(), k));
458         if (m) putchar('\n');
459     }
460     return 0;
461 }
```

```

/// BZOJ 1146
/// 主席树 带修改树上路径第 k 大
#include <bits/stdc++.h>
#define mid (((l) + (r)) / 2)
#define lowbit(x) ((x) & -(x))
using namespace std;

inline int ReadInt() {
    static int n, ch;
    n = 0, ch = getchar();
    while (!isdigit(ch)) ch = getchar();
    while (isdigit(ch)) n = (n << 3) + (n << 1) + ch - '0', ch = getchar();
    return n;
}
typedef long long ll;

const int maxn = 80000 + 3, maxq = 80000 + 3;
struct SegNode *pit, *null;
struct SegNode {
    SegNode *ls, *rs;
    int s;
    inline void maintain() {
        s = ls->s + rs->s;
    }
    SegNode(): ls(null), rs(null), s(0) {}
}pool[maxn * 85], *root[maxn], *Fen[maxn], *add[100], *dec[100];

void init() {
    pit = pool;
    null = new SegNode();
    null->ls = null, null->rs = null;
}

vector<int> G[maxn], Ws;
int n, m, w[maxn], ancestor[maxn][18], depth[maxn], k[maxn], a[maxn], b[maxn], id[maxn
    ], s[maxn], timestamp = 0;

void compress() {
    for (int i = 0; i < n; ++i)
        Ws.push_back(w[i]);
    for (int i = 0; i < m; ++i) {
        k[i] = ReadInt(), a[i] = ReadInt() - 1, b[i] = ReadInt() - (k[i] > 0);
        if (k[i] == 0) Ws.push_back(b[i]);
    }
    sort(Ws.begin(), Ws.end());
    Ws.erase(unique(Ws.begin(), Ws.end()), Ws.end());
    for (int i = 0; i < n; ++i)
        w[i] = lower_bound(Ws.begin(), Ws.end(), w[i]) - Ws.begin();
    for (int i = 0; i < m; ++i)
        if (k[i] == 0) b[i] = lower_bound(Ws.begin(), Ws.end(), b[i]) - Ws.begin();
}

void process() {
    for (int w = 1; (1 << w) < n; ++w)
        for (int i = 0; i < n; ++i) if (depth[i] - (1 << w) >= 0)
            ancestor[i][w] = ancestor[ancestor[i][w - 1]][w - 1];
}
```

```
520
521 int LCA(int a, int b) {
522     if (depth[a] < depth[b]) swap(a, b);
523     int lim = log2(depth[a]);
524     for (int i = lim; i >= 0; --i)
525         if (depth[a] - (1 << i) >= depth[b])
526             a = ancestor[a][i];
527     if (a == b) return a;
528     for (int i = lim; i >= 0; --i)
529         if (depth[a] - (1 << i) >= 0 && ancestor[a][i] != ancestor[b][i]) {
530             a = ancestor[a][i];
531             b = ancestor[b][i];
532         }
533     return ancestor[a][0];
534 }
535
536 SegNode* modify(const SegNode *o, int l, int r, int v, int op) {
537     if (l >= r) return null;
538     SegNode *ne = pit++;
539     *ne = *o;
540     if (r - l == 1)
541         ne->s += op;
542     else {
543         if (v < mid) ne->ls = modify(ne->ls, l, mid, v, op);
544         else ne->rs = modify(ne->rs, mid, r, v, op);
545         ne->maintain();
546     }
547     return ne;
548 }
549
550 int query(SegNode* add[], int addc, SegNode* dec[], int decc, int l, int r, int k) {
551     if (r - l == 1) return Ws[l];
552     int s = 0;
553     for (int i = 0; i < addc; ++i) s += add[i]->ls->s;
554     for (int i = 0; i < decc; ++i) s -= dec[i]->ls->s;
555     for (int i = 0; i < addc; ++i)
556         if (k <= s) add[i] = add[i]->ls;
557         else add[i] = add[i]->rs;
558     for (int i = 0; i < decc; ++i)
559         if (k <= s) dec[i] = dec[i]->ls;
560         else dec[i] = dec[i]->rs;
561     if (k <= s) {
562         return query(add, addc, dec, decc, l, mid, k);
563     }else return query(add, addc, dec, decc, mid, r, k - s);
564 }
565
566 int dfs(const SegNode *o, int u, int fa) {
567     id[u] = timestamp++, ancestor[u][0] = fa, depth[u] = fa == -1 ? 0 : depth[fa] + 1;
568     root[u] = modify(o, 0, Ws.size(), w[u], 1);
569     s[u] = 1;
570     for (int i = 0; i < (int)G[u].size(); ++i) {
571         int v = G[u][i];
572         if (v != fa) s[u] += dfs(root[u], v, u);
573     }
574     return s[u];
575 }
576
577 int main() {
578     init();
```

```
579     n = ReadInt(), m = ReadInt();
580     for (int i = 0; i < n; ++i)
581         w[i] = ReadInt();
582     for (int i = 0; i < n - 1; ++i) {
583         int f = ReadInt() - 1, t = ReadInt() - 1;
584         G[f].push_back(t);
585         G[t].push_back(f);
586     }
587     compress();
588     dfs(null, 0, -1);
589     process();
590     for (int i = 1; i <= n; ++i) Fen[i] = null;
591     for (int i = 0; i < m; ++i) {
592         if (k[i] == 0) {
593             for (int j = id[a[i]] + 1; j <= n; j += lowbit(j)) {
594                 Fen[j] = modify(Fen[j], 0, Ws.size(), w[a[i]], -1);
595                 Fen[j] = modify(Fen[j], 0, Ws.size(), b[i], 1);
596             }
597             for (int j = id[a[i]] + s[a[i]] + 1; j <= n; j += lowbit(j)) {
598                 Fen[j] = modify(Fen[j], 0, Ws.size(), w[a[i]], 1);
599                 Fen[j] = modify(Fen[j], 0, Ws.size(), b[i], -1);
600             }
601             w[a[i]] = b[i];
602         }else {
603             int lca = LCA(a[i], b[i]), length = depth[a[i]] + depth[b[i]] - 2 * depth[
    lca] + 1;
604             k[i] = length - k[i] + 1;
605             if (k[i] <= 0) puts("invalid request!");
606             else {
607                 int addc = 0, decc = 0;
608                 add[addc++] = root[a[i]], add[addc++] = root[b[i]], dec[decc++] = root[
    lca];
609                 if (lca) dec[decc++] = root[ancestor[lca][0]];
610                 for (int j = id[a[i]] + 1; j > 0; j -= lowbit(j)) add[addc++] = Fen[j];
611                 for (int j = id[b[i]] + 1; j > 0; j -= lowbit(j)) add[addc++] = Fen[j];
612                 for (int j = id[lca] + 1; j > 0; j -= lowbit(j)) dec[decc++] = Fen[j];
613                 if (lca) for (int j = id[ancestor[lca][0]] + 1; j > 0; j -= lowbit(j))
    dec[decc++] = Fen[j];
614                 printf("%d\n", query(add, addc, dec, decc, 0, Ws.size(), k[i]));
615             }
616         }
617     }
618     return 0;
619 }
```

4 Graph Theory

4.1 Buding

4.1.1 Buiding

```
struct Node
{
  int to;
  int w;
  int ne;
};

Node node[N];
int head[N];
int top;

void add(int from, int to, int w)
{
  node[top].to = to;
  node[top].w = w;
  node[top].ne = head[from];
  head[from] = top;
  top ++;
}

for(int i = head[u]; i != -1; i = node[i].ne){
  /// ±□□□
}

void ini()
{
  top = 0;
  memset(head, -1, sizeof(head));
}
```

4.2 Shortest Path

4.2.1 Dijkstra

```
Edge edge[N];
int n, m;
LL dis[N];

void Dij()
{
    priority_queue< pair<int, int> > Q;
    memset(dis, 0x3f3f3f3f, sizeof(dis));
    dis[1] = 0;
    Q.push(mp(0, 1));
    while(!Q.empty()){
        int w;
        int u;

        u = Q.top().second;
        Q.pop();
        for(int i = head[u]; i != -1; i = node[i].ne){
            int v;

```

```
            v = node[i].to;
            if(dis[v] > dis[u] + node[i].w){
                dis[v] = dis[u] + node[i].w
                Q.push(mp(-dis[v], node[i].w);
            }
        }
    }
}
```

4.2.2 SPFA

```
const int N = 1010;

Node node[N];
LL dis[N];
int n, m;

bool SPFA()
{
    queue<int> Q;
    bool inq[N];
    int cnt[N];

    memset(dis, 0x3f3f3f3f, sizeof(dis));
    memset(cnt, 0, sizeof(cnt));
    memset(inq, 0, sizeof(inq));
    dis[n] = 0;
    cnt[n] ++;
    inq[n] = 1;
    Q.push(n);
    while(!Q.empty()){
        int now;
        int li;

        now = Q.front();
        inq[now] = 0;
        Q.pop();
        for(int i = 0; i < li; i ++){
            if(dis[node[now].V[i]] > dis[now] + node[now].W[i]){
                dis[node[now].V[i]] = dis[now] + node[now].W[i];
                if(!inq[node[now].V[i]]){
                    inq[node[now].V[i]] = 1;
                    Q.push(node[now].V[i]);
                    cnt[node[now].V[i]] ++;
                    if(cnt[node[now].V[i]] > n + 1){
                        return false;
                    }
                }
            }
        }
    }

    return true;
}
```

4.2.3 Floyd

```
for(k = 1; k <= n; k ++)
    for(i = 1; i <= n; i ++)
        for(j = 1; j <= n; j ++)
            if(e[i][j] > e[i][k] + e[k][j])
                e[i][j]=e[i][k]+e[k][j];
```

4.2.4 K-th-Dijkstra

```
#include <iostream>
#include <cstring>
#include <queue>
#include <fstream>
using namespace std;

#define E 100005
#define V 1005
#define INF 1 << 30

int heads[V], r_heads[V];
int dists[V];
bool visits[V];

int nEdgeNum, nNodeNum, nEdgeCount;
int nEnd, nSrc, k;

struct Edge{
    int to_node;
    int next_edge;
    int edge_weight;
    int r_to_node;
    int r_next_edge;

    Edge(){}
    Edge(int from, int to, int weight){
        to_node = to;
        r_to_node = from;
        edge_weight = weight;
     }
}edges[E];

struct Node{
    int v;
    int src_to_v_dist;

    Node(){
        this->v = 0;
        this->src_to_v_dist = 0;
    }
    Node( int v, int d ){
        this->v = v;
        this->src_to_v_dist = d;
    }
    bool operator < ( const Node& other ) const{
        return src_to_v_dist + dists[v] > dists[other.v] + other.src_to_v_dist;
    }
};

void addEdge( int from, int to, int dist ){
```

```
51      edges[nEdgeCount] = Edge( from, to, dist );
52      edges[nEdgeCount].r_next_edge = r_heads[to];
53      edges[nEdgeCount].next_edge = heads[from];
54      heads[from] = nEdgeCount;
55      r_heads[to] = nEdgeCount;
56      nEdgeCount++;
57  }
58
59  void dijkstra( int src ){
60
61      priority_queue< Node > que;
62
63      for( int i = 1; i <= nNodeNum; i ++){
64          dists[i] = INF;
65      }
66      dists[src] = 0;
67      que.push(Node(src, 0));
68      while(!que.empty()){
69          Node cur = que.top();
70          que.pop();
71          if(visits[cur.v]){
72              continue;
73          }
74          visits[cur.v] = true;
75          for( int i = r_heads[cur.v]; ~i; i = edges[i].r_next_edge ){
76              if( dists[edges[i].r_to_node] > dists[cur.v] + edges[i].edge_weight ){
77                  dists[edges[i].r_to_node] = dists[cur.v] + edges[i].edge_weight;
78                  que.push(Node(edges[i].r_to_node, 0));
79              }
80          }
81      }
82  }
83
84  int AStar( int src ){
85      priority_queue< Node > que;
86
87      que.push(Node(src, 0));
88      while(!que.empty()){
89          Node cur = que.top();
90          que.pop();
91          if(cur.v == nEnd){
92              if(k > 1){
93                  k--;
94              }
95              else{
96                  return cur.src_to_v_dist;
97              }
98          }
99          for(int i = heads[cur.v]; ~i; i = edges[i].next_edge){
100             que.push(Node(edges[i].to_node, cur.src_to_v_dist + edges[i].edge_weight));
101         }
102     }
103
104     return -1;
105 }
106
107 void init(){
108     memset(visits, false, sizeof(visits ));
109     memset(heads, -1, sizeof(heads));
```

```
    memset(r_heads, -1, sizeof(r_heads));
    nEdgeCount = 0;
}

int main(){
    while(cin >> nNodeNum >> nEdgeNum){
        init();
        for(int i = 1; i <= nEdgeNum; i ++){
            int from, to, dist;

            cin >> from >> to >> dist;
            addEdge( from, to, dist );
        }
        cin >> nSrc >> nEnd >> k;
        dijkstra( nEnd );
        if(dists[nSrc] == INF){
            cout << "-1\n";

            continue;
        }

        if(nSrc == nEnd){
            k ++;
        }
        int ans = AStar(nSrc);
        cout << ans << endl;
    }

    return 0;
}
```

4.2.5 K-th-SPFA

```
#define INF 0xffffff
#define MAXN 100010
struct node
{
    int to;
    int val;
    int next;
};
struct node2
{
    int to;
    int g,f;
    bool operator<(const node2 &r ) const
    {
        if(r.f==f)
            return r.g<g;
        return r.f<f;
    }
};
node edge[MAXN],edge2[MAXN];
int n,m,s,t,k,cnt,cnt2,ans;
int dis[1010],visit[1010],head[1010],head2[1010];
void init()
{
    memset(head,-1,sizeof(head));
```

```
26      memset(head2,-1,sizeof(head2));
27      cnt=cnt2=1;
28  }
29  void addedge(int from,int to,int val)
30  {
31      edge[cnt].to=to;
32      edge[cnt].val=val;
33      edge[cnt].next=head[from];
34      head[from]=cnt++;
35  }
36  void addedge2(int from,int to,int val)
37  {
38      edge2[cnt2].to=to;
39      edge2[cnt2].val=val;
40      edge2[cnt2].next=head2[from];
41      head2[from]=cnt2++;
42  }
43  bool spfa(int s,int n,int head[],node edge[],int dist[])
44  {
45      queue<int>Q1;
46      int inq[1010];
47      for(int i=0;i<=n;i++)
48      {
49          dis[i]=INF;
50          inq[i]=0;
51      }
52      dis[s]=0;
53      Q1.push(s);
54      inq[s]++;
55      while(!Q1.empty())
56      {
57          int q=Q1.front();
58          Q1.pop();
59          inq[q]--;
60          if(inq[q]>n)
61              return false;
62          int k=head[q];
63          while(k>=0)
64          {
65              if(dist[edge[k].to]>dist[q]+edge[k].val)
66              {
67                  dist[edge[k].to]=edge[k].val+dist[q];
68                  if(!inq[edge[k].to])
69                  {
70                      inq[edge[k].to]++;
71                      Q1.push(edge[k].to);
72                  }
73              }
74              k=edge[k].next;
75          }
76      }
77      return true;
78  }
79  int A_star(int s,int t,int n,int k,int head[],node edge[],int dist[])
80  {
81      node2 e,ne;
82      int cnt=0;
83      priority_queue<node2>Q;
84      if(s==t)
```

```
85          k++;
86      if(dis[s]==INF)
87          return -1;
88      e.to=s;
89      e.g=0;
90      e.f=e.g+dis[e.to];
91      Q.push(e);
92
93      while(!Q.empty())
94      {
95          e=Q.top();
96          Q.pop();
97          if(e.to==t)//找到一条最短路径
98          {
99              cnt++;
100         }
101         if(cnt==k)//找到k短路
102         {
103             return e.g;
104         }
105         for(int i=head[e.to]; i!=-1; i=edge[i].next)
106         {
107             ne.to=edge[i].to;
108             ne.g=e.g+edge[i].val;
109             ne.f=ne.g+dis[ne.to];
110             Q.push(ne);
111         }
112     }
113     return -1;
114 }
115 int main()
116 {
117     while(~scanf("%d%d",&n,&m))
118     {
119         init();
120         for(int i=1;i<=m;i++)
121         {
122             int a,b,c;
123             scanf("%d%d%d",&a,&b,&c);
124             addedge(a,b,c);
125             addedge2(b,a,c);
126         }
127         scanf("%d%d%d",&s,&t,&k);
128         spfa(t,n,head2,edge2,dis);
129         ans=A_star(s,t,n,k,head,edge,dis);
130         printf("%d\n",ans);
131     }
132
133     return 0;
134 }
```

4.3 LCA

4.3.1 DFS+ST

DFS+ST 在线算法
时间复杂度 $O(nlogn+q)$

```
1 const int maxn = "Edit";
```

```
vector<int> G[maxn], sp;
int dep[maxn], dfn[maxn];
PII dp[21][maxn << 1];
void init(int n)
{
    for (int i = 0; i < n; i++) G[i].clear();
    sp.clear();
}
void dfs(int u, int fa)
{
    dep[u] = dep[fa] + 1;
    dfn[u] = sp.size();
    sp.push_back(u);
    for (auto& v : G[u])
    {
        if (v == fa) continue;
        dfs(v, u);
        sp.push_back(u);
    }
}
void initrmq()
{
    int n = sp.size();
    for (int i = 0; i < n; i++) dp[0][i] = {dfn[sp[i]], sp[i]};
    for (int i = 1; (1 << i) <= n; i++)
        for (int j = 0; j + (1 << i) - 1 < n; j++)
            dp[i][j] = min(dp[i - 1][j], dp[i - 1][j + (1 << (i - 1))]);
}
int lca(int u, int v)
{
    int l = dfn[u], r = dfn[v];
    if (l > r) swap(l, r);
    int k = 31 - __builtin_clz(r - l + 1);
    return min(dp[k][l], dp[k][r - (1 << k) + 1]).second;
}
```

4.3.2 Tarjan

Tarjan 离线算法
时间复杂度 $O(n+q)$

```
const int maxn = "Edit";
int par[maxn];           //并查集
int ans[maxn];           //存储答案
vector<int> G[maxn];     //邻接表
vector<PII> query[maxn]; //存储查询信息
bool vis[maxn];          //是否被遍历
inline void init(int n)
{
    for (int i = 1; i <= n; i++)
    {
        G[i].clear(), query[i].clear();
        par[i] = i, vis[i] = 0;
    }
}
inline void add_edge(int u, int v) { G[u].push_back(v); }
inline void add_query(int id, int u, int v)
{
```

```
    query[u].emplace_back(v, id);
    query[v].emplace_back(u, id);
}
void tarjan(int u)
{
    vis[u] = 1;
    for (auto& v : G[u])
    {
        if (vis[v]) continue;
        tarjan(v);
        unite(u, v);
    }
    for (auto& q : query[u])
    {
        int &v = q.X, &id = q.Y;
        if (!vis[v]) continue;
        ans[id] = find(v);
    }
}
```

4.4 Mst

4.4.1 Prim

```
const int N = 100010;

int ne[N];
int ma[100][100];
int n, m;
int cnt;

int Prim()
{
    int ans = 0;
    int num = 1;
    int minn;
    int v;
    int dis[N];

    memset(dis, 0x3f3f3f3f, sizeof(dis));
    for(int i = 1; i <= n; i ++){
        dis[i] = min(dis[i], ma[1][i]);
    }
    while(num < n){
        minn = 0x3f3f3f3f;
        for(int i = 1; i <= n; i ++){
            if(dis[i] < minn && dis[i]){
                minn = dis[i];
                v = i;
            }
        }
        ans += minn;
        for(int i = 1; i <= n; i ++){
            dis[i] = min(dis[i], ma[v][i]);
        }
        num ++;
    }

    return ans;
```

```
}
```

4.4.2 Kruskal

```
const int N = 100010;

struct Node
{
    int u;
    int v;
    int w;
};

bool cmp(Node a, Node b)
{
    return a.w < b.w;
}

Node node[N];
int ne[N];
int n, m;
int cnt;

void Ini()
{
    for(int i = 1; i <= n; i ++){
        ne[i] = i;
    }
}

int Find(int x)
{
    int t = x;

    while(t != ne[t]){
        t = ne[t];
    }
    while(x != t){
        int q;

        q = ne[x];
        ne[x] = t;
        x = q;
    }

    return t;
}

void Join(int x, int y)
{
    ne[x] = y;
}

int Kru()
{
    int ans;
    int num;

```

```
    ans = 0;
    num = 0;
    for(int i = 0; i < cnt; i ++){
        int u = node[i].u;
        int v = node[i].v;
        int w = node[i].w;

        u = Find(u);
        v = Find(v);
        if(u != v){
            Join(u, v);
            ans += w;
            num ++;
        }
        if(num == n - 1){
            break;
        }
    }

    return ans;
}
```

4.4.3 Zhu Liu

```
const int maxn = "Edit";
// 固定根的最小树型图，邻接矩阵写法
struct MDST
{
    int n;
    int w[maxn][maxn]; // 边权
    int vis[maxn];     // 访问标记，仅用来判断无解
    int ans;           // 计算答案
    int removed[maxn]; // 每个点是否被删除
    int cid[maxn];     // 所在圈编号
    int pre[maxn];     // 最小入边的起点
    int iw[maxn];      // 最小入边的权值
    int max_cid;       // 最大圈编号
    void init(int n)
    {
        this->n = n;
        for (int i = 0; i < n; i++)
            for (int j = 0; j < n; j++) w[i][j] = INF;
    }
    void AddEdge(int u, int v, int cost)
    {
        w[u][v] = min(w[u][v], cost); // 重边取权最小的
    }
    // 从s出发能到达多少个结点
    int dfs(int s)
    {
        vis[s] = 1;
        int ans = 1;
        for (int i = 0; i < n; i++)
            if (!vis[i] && w[s][i] < INF) ans += dfs(i);
        return ans;
    }
    // 从u出发沿着pre指针找圈
    bool cycle(int u)
```

```
35      {
36          max_cid++;
37          int v = u;
38          while (cid[v] != max_cid)
39          {
40              cid[v] = max_cid;
41              v = pre[v];
42          }
43          return v == u;
44      }
45      // 计算u的最小入弧，入弧起点不得在圈c中
46      void update(int u)
47      {
48          iw[u] = INF;
49          for (int i = 0; i < n; i++)
50              if (!removed[i] && w[i][u] < iw[u])
51              {
52                  iw[u] = w[i][u];
53                  pre[u] = i;
54              }
55      }
56      // 根结点为s，如果失败则返回false
57      bool solve(int s)
58      {
59          memset(vis, 0, sizeof(vis));
60          if (dfs(s) != n) return false;
61          memset(removed, 0, sizeof(removed));
62          memset(cid, 0, sizeof(cid));
63          for (int u = 0; u < n; u++) update(u);
64          pre[s] = s;
65          iw[s] = 0; // 根结点特殊处理
66          ans = max_cid = 0;
67          for (;;)
68          {
69              bool have_cycle = false;
70              for (int u = 0; u < n; u++)
71                  if (u != s && !removed[u] && cycle(u))
72                  {
73                      have_cycle = true;
74                      // 以下代码缩圈，圈上除了u之外的结点均删除
75                      int v = u;
76                      do
77                      {
78                          if (v != u) removed[v] = 1;
79                          ans += iw[v];
80                          // 对于圈外点i，把边i->v改成i->u（并调整权值）；v->i改为u->i
81                          // 注意圈上可能还有一个v'使得i->v'或者v'->i存在，
82                          // 因此只保留权值最小的i->u和u->i
83                          for (int i = 0; i < n; i++)
84                              if (cid[i] != cid[u] && !removed[i])
85                              {
86                                  if (w[i][v] < INF)
87                                      w[i][u] = min(w[i][u], w[i][v] - iw[v]);
88                                  w[u][i] = min(w[u][i], w[v][i]);
89                                  if (pre[i] == v) pre[i] = u;
90                              }
91                          v = pre[v];
92                      } while (v != u);
93                      update(u);
```

```
                    break;
                }
            if (!have_cycle) break;
        }
        for (int i = 0; i < n; i++)
            if (!removed[i]) ans += iw[i];
        return true;
    }
};
```

4.5 Depth-First Traversal

4.5.1 Biconnected-Component

```
//割顶的bccno无意义
const int maxn = "Edit";
int pre[maxn], iscut[maxn], bccno[maxn], dfs_clock, bcc_cnt;
vector<int> G[maxn], bcc[maxn];
stack<PII> s;
void init(int n)
{
    for (int i = 0; i < n; i++) G[i].clear();
}
inline void add_edge(int u, int v) { G[u].push_back(v), G[v].push_back(u); }
int dfs(int u, int fa)
{
    int lowu = pre[u] = ++dfs_clock;
    int child = 0;
    for (auto& v : G[u])
    {
        PII e = {u, v};
        if (!pre[v])
        {
            //没有访问过v
            s.push(e);
            child++;
            int lowv = dfs(v, u);
            lowu = min(lowu, lowv); //用后代的low函数更新自己
            if (lowv >= pre[u])
            {
                iscut[u] = true;
                bcc_cnt++;
                bcc[bcc_cnt].clear(); //注意! bcc从1开始编号
                for (;;)
                {
                    PII x = s.top();
                    s.pop();
                    if (bccno[x.first] != bcc_cnt)
                        bcc[bcc_cnt].push_back(x.first), bcc[x.first] = bcc_cnt;
                    if (bccno[x.second] != bcc_cnt)
                        bcc[bcc_cnt].push_back(x.second), bcc[x.second] = bcc_cnt;
                    if (x.first == u && x.second == v) break;
                }
            }
        }
        else if (pre[v] < pre[u] && v != fa)
        {
            s.push(e);
            lowu = min(lowu, pre[v]); //用反向边更新自己
```

```
46          }
47      }
48      if (fa < 0 && child == 1) iscut[u] = 0;
49      return lowu;
50  }
51  void find_bcc(int n)
52  {
53      //调用结束后S保证为空，所以不用清空
54      memset(pre, 0, sizeof(pre));
55      memset(iscut, 0, sizeof(iscut));
56      memset(bccno, 0, sizeof(bccno));
57      dfs_clock = bcc_cnt = 0;
58      for (int i = 0; i < n; i++)
59          if (!pre[i]) dfs(i, -1);
60  }
```

4.5.2 Strongly Connected Component

```
1   const int maxn = "Edit";
2   vector<int> G[maxn];
3   int pre[maxn], lowlink[maxn], sccno[maxn], dfs_clock, scc_cnt;
4   stack<int> S;
5   inline void init(int n)
6   {
7       for (int i = 0; i < n; i++) G[i].clear();
8   }
9   inline void add_edge(int u, int v) { G[u].push_back(v); }
10  void dfs(int u)
11  {
12      pre[u] = lowlink[u] = ++dfs_clock;
13      S.push(u);
14      for (auto& v : G[u])
15      {
16          if (!pre[v])
17          {
18              dfs(v);
19              lowlink[u] = min(lowlink[u], lowlink[v]);
20          }
21          else if (!sccno[v])
22              lowlink[u] = min(lowlink[u], pre[v]);
23      }
24      if (lowlink[u] == pre[u])
25      {
26          scc_cnt++;
27          for (;;)
28          {
29              int x = S.top();
30              S.pop();
31              sccno[x] = scc_cnt;
32              if (x == u) break;
33          }
34      }
35  }
36  void find_scc(int n)
37  {
38      dfs_clock = 0, scc_cnt = 0;
39      memset(sccno, 0, sizeof(sccno)), memset(pre, 0, sizeof(pre));
40      for (int i = 0; i < n; i++)
```

```
        if (!pre[i]) dfs(i);
}
```

4.5.3 2-SAT

```
const int maxn = "Edit";
struct TwoSAT
{
    int n;
    vector<int> G[maxn << 1];
    bool mark[maxn << 1];
    int S[maxn << 1], c;
    void init(int n)
    {
        this->n = n;
        for (int i = 0; i < (n << 1); i++) G[i].clear();
        memset(mark, 0, sizeof(mark));
    }
    bool dfs(int x)
    {
        if (mark[x ^ 1]) return false;
        if (mark[x]) return true;
        mark[x] = true;
        S[c++] = x;
        for (auto& y : G[x])
            if (!dfs(y)) return false;
        return true;
    }
    //x = xval or y = yval
    void add_clause(int x, int xval, int y, int yval)
    {
        x = (x << 1) + xval;
        y = (y << 1) + yval;
        G[x ^ 1].push_back(y);
        G[y ^ 1].push_back(x);
    }
    bool solve()
    {
        for (int i = 0; i < (n << 1); i += 2)
            if (!mark[i] && !mark[i + 1])
            {
                c = 0;
                if (!dfs(i))
                {
                    while (c > 0) mark[S[--c]] = false;
                    if (!dfs(i + 1)) return false;
                }
            }
        return true;
    }
};
```

4.5.4 Tarjan

```
const int N = 110;

struct Node
{
```

```
5       vector<int> V;
6   };
7
8   Node node[N];
9   int instack[N];
10  int low[N];
11  int dfn[N];
12  int St[N];
13  int bl[N];
14  int ine[N], oute[N];    /// □□□¬³□□
15  int index;
16  int cnt;
17  int top;
18  int ans1, ans2;
19  int n;
20
21  void ini()
22  {
23      for(int i = 1; i <= n; i ++){
24          node[i].V.clear();
25      }
26      memset(instack, 0, sizeof(instack));
27      memset(low, 0, sizeof(low));
28      memset(dfn, 0, sizeof(dfn));
29      memset(St, 0, sizeof(St));
30      memset(ine, 0, sizeof(ine));
31      memset(oute, 0, sizeof(oute));
32      ans1 = ans2 = index = top = cnt = 0;
33  }
34
35  void Tarjan(int u)
36  {
37      int li;
38
39      St[top] = u;
40      top ++;
41      index ++;
42      low[u] = dfn[u] = index;
43      instack[u] = 1;
44      li = node[u].V.size();
45      for(int i = 0; i < li; i ++){
46          int v;
47
48          v = node[u].V[i];
49          if(!dfn[v]){
50              Tarjan(v);
51              low[u] = min(low[u], low[v]);
52          }
53          else if(instack[v]){
54              low[u] = min(low[u], dfn[v]);
55          }
56      }
57      if(low[u] == dfn[u]){
58          int v;
59
60          v = -1;
61          while(v != u){
62              top --;
63              v = St[top];
```

```
64              instack[v] = 0;
65              bl[v] = cnt;
66          }
67          cnt ++;
68      }
69  }
70
71  int main(int argc, char const *argv[])
72  {
73      while(scanf("%d", &n) == 1){
74          ini();
75          for(int i = 1; i <= n; i ++){
76              while(true){
77                  int v;
78
79                  scanf("%d", &v);
80                  if(v){
81                      node[i].V.pb(v);
82                  }
83                  else{
84                      break;
85                  }
86              }
87          }
88          for(int i = 1; i <= n; i ++){
89              if(!dfn[i]){
90                  Tarjan(i);
91              }
92          }
93          for(int i = 1; i <= n; i ++){
94              int li;
95
96              li = node[i].V.size();
97              for(int j = 0; j < li; j ++){
98                  int v;
99
100                 v = node[i].V[j];
101                 if(bl[v] != bl[i]){
102                     oute[bl[i]] ++;
103                     ine[bl[v]] ++;
104                 }
105             }
106         }
107         for(int i = 0; i < cnt; i ++){
108             if(!ine[i]){
109                 ans1 ++;
110             }
111             else if(!oute[i]){
112                 ans2 ++;
113             }
114         }
115         printf("%d\n", ans1);
116         if(cnt == 1){
117             printf("0\n");
118         }
119         else{
120             printf("%d\n", max(ans1, ans2));
121         }
122     }
```

```
123
124     return 0;
125 }
```

4.6 Eular Path

- 基本概念:
  - 欧拉图: 能够没有重复地一次遍历所有边的图。(必须是连通图)
  - 欧拉路: 上述遍历的路径就是欧拉路。
  - 欧拉回路: 若欧拉路是闭合的(一个圈，从起点开始遍历最终又回到起点)，则为欧拉回路。
- 无向图 G 有欧拉路径的充要条件
  - G 是连通图
  - G 中奇顶点(连接边的数量为奇数)的数量等于 0 或 2.
- 无向图 G 有欧拉回路的充要条件
  - G 是连通图
  - G 中每个顶点都是偶顶点
- 有向图 G 有欧拉路径的充要条件
  - G 是连通图
  - u 的出度比入度大 1，v 的出度比入度小 1，其他所有点出度和入度相同。(u 为起点，v 为终点)
- 有向图 G 有欧拉回路的充要条件
  - G 是连通图
  - G 中每个顶点的出度等于入度

4.6.1 Fleury

若有两个点的度数是奇数，则此时这两个点只能作为欧拉路径的起点和终点。

```
1  const int maxn = "Edit";
2  int G[maxn][maxn];
3  int deg[maxn][maxn];
4  vector<int> ans;
5  inline void init() { memset(G, 0, sizeof(G)), memset(deg, 0, sizeof(deg)); }
6  inline void AddEdge(int u, int v) { deg[u]++, deg[v]++, G[u][v]++, G[v][u]++; }
7  void Fleury(int s)
8  {
9      for (int i = 0; i < n; i++)
10         if (G[s][i]){
11             G[s][i]--, G[i][s]--;
12             Fleury(i);
13         }
14     ans.push_back(s);
15 }
```

4.7 Network Flow

1. 一个二分图中的最大匹配数等于这个图中的最小点覆盖数
2. 最小路径覆盖 =|G|-最大匹配数
   在一个 $N \times N$ 的有向图中, 路径覆盖就是在图中找一些路经, 使之覆盖了图中的所有顶点, 且任何一个顶点有且只有一条路径与之关联;
   (如果把这些路径中的每条路径从它的起始点走到它的终点, 那么恰好可以经过图中的每个顶点一次且仅一次); 如果不考虑图中存在回路, 那么每每条路径就是一个弱连通子集.
   由上面可以得出:

(a) 一个单独的顶点是一条路径;

(b) 如果存在一路径 $p_1, p_2, ...p_k$, 其中 p_1 为起点,p_k 为终点, 那么在覆盖图中, 顶点 $p_1, p2, ...p_k$ 不再与其它的顶点之间存在有向边.

最小路径覆盖就是找出最小的路径条数, 使之成为 G 的一个路径覆盖.

路径覆盖与二分图匹配的关系: 最小路径覆盖 =$|G|$-最大匹配数;

3. 二分图最大独立集 = 顶点数-二分图最大匹配
   独立集: 图中任意两个顶点都不相连的顶点集合。
4. 最大权闭合子图 = 正权点和 - 新图最小割
5. 二分图最大边覆盖 = 点数 - 二分图最大匹配
6. 有向无环图的最小路径覆盖 = 原图点数 - 新图二分图最大匹配 (有向边 A->B，加边 Ax−>By)

4.7.1 Trick

建模技巧

二分图带权最大独立集。给出一个二分图，每个结点上有一个正权值。要求选出一些点，使得这些点之间没有边相连，且权值和最大。

解：在二分图的基础上添加源点 S 和汇点 T，然后从 S 向所有 X 集合中的点连一条边，所有 Y 集合中的点向 T 连一条边，容量均为该点的权值。X 结点与 Y 结点之间的边的容量均为无穷大。这样，对于图中的任意一个割，将割中的边对应的结点删掉就是一个符合要求的解，权和为所有权减去割的容量。因此，只需要求出最小割，就能求出最大权和。

公平分配问题。把 m 个任务分配给 n 个处理器。其中每个任务有两个候选处理器，可以任选一个分配。要求所有处理器中，任务数最多的那个处理器所分配的任务数尽量少。不同任务的候选处理器集 $\{p_1, p_2\}$ 保证不同。

解：本题有一个比较明显的二分图模型，即 X 结点是任务，Y 结点是处理器。二分答案 x，然后构图，首先从源点 S 出发向所有的任务结点引一条边，容量等于 1，然后从每个任务结点出发引两条边，分别到达它所能分配到的两个处理器结点，容量为 1，最后从每个处理器结点出发引一条边到汇点 T，容量为 x，表示选择该处理器的任务不能超过 x。这样网络中的每个单位流量都是从 S 流到一个任务结点，再到处理器结点，最后到汇点 T。只有当网络中的总流量等于 m 时才意味着所有任务都选择了一个处理器。这样，我们通过 $O(\log m)$ 次最大流便算出了答案。

区间 k 覆盖问题。数轴上有一些带权值的左闭右开区间。选出权和尽量大的一些区间，使得任意一个数最多被 k 个区间覆盖。

解：本题可以用最小费用流解决，构图方法是把每个数作为一个结点，然后对于权值为 w 的区间 $[u, v)$ 加边 $u \to v$，容量为 1，费用为 $-w$。再对所有相邻的点加边 $i \to i+1$，容量为 k，费用为 0。最后，求最左点到最右点的最小费用最大流即可，其中每个流量对应一组互不相交的区间。如果数值范围太大，可以先进行离散化。

最大闭合子图。给定带权图 G（权值可正可负），求一个权和最大的点集，使得起点在该点集中的任意弧，终点也在该点集中。

解：新增附加源 s 和附加汇 t，从 s 向所有正权点引一条边，容量为权值；从所有负权点向汇点引一条边，容量为权值的相反数。求出最小割以后，$S - \{s\}$ 就是最大闭合子图。

最大密度子图。给出一个无向图，找一个点集，使得这些点之间的边数除以点数的值（称为子图的密度）最大。

解：如果两个端点都选了，就必然要选边，这就是一种推导。如果把每个点和每条边都看成新图中的结点，可以把问题转化为最大闭合子图。

4.7.2 Dinic

```
1  const int N = 100010;
2  const int INF = 0x3f3f3f3f;
3
4  struct Node
5  {
6      int to;
```

```
7       int w;
8       int ne;
9   };
10
11  Node node[N];
12  int Q[N];   /// 队列  注意数组的大小
13  int head[N];
14  int cur[N]; /// 前向弧优化
15  int di[N];
16  int ma[N];
17  int ss, tt;
18  int top;
19  int n, m;
20
21  void ini()
22  {
23      top = 0;
24      memset(head, -1, sizeof(head));
25  }
26
27  void add_edge(int from, int to, int w)
28  {
29      node[top].to = to;
30      node[top].w = w;
31      node[top].ne = head[from];
32      head[from] = top;
33      top ++;
34  }
35
36  bool build()
37  {
38      int l, r;
39      int now;
40
41      memset(di, 0, sizeof(di));
42      di[ss] = 1;
43      l = r = 0;
44      Q[r] = ss;
45      r ++;
46      while(l != r){
47          now = Q[l];
48          l ++;
49          if(now == tt){
50              return true;
51          }
52          for(int i = head[now]; i != -1; i = node[i].ne){
53              int v, w;
54
55              v = node[i].to;
56              w = node[i].w;
57              if(w && !di[v]){
58                  di[v] = di[now] + 1;
59                  Q[r] = v;
60                  r ++;
61              }
62          }
63      }
64
65      return false;
```

```
66  }
67
68  int dfs(int u, int maxf)
69  {
70      int ans;
71
72      if(u == tt){
73          return maxf;
74      }
75      ans = 0;
76      for(int &i = cur[u]; i != -1; i = node[i].ne){
77          int v, w;
78
79          v = node[i].to;
80          w = node[i].w;
81          if(w && di[v] == di[u] + 1){
82              int t;
83
84              t = dfs(v, min(maxf - ans, w));
85              node[i].w -= t;
86              node[i ^ 1].w += t;
87              ans += t;
88              if(ans == maxf){
89                  return ans;
90              }
91          }
92      }
93      if(!ans){
94          di[u] = -2;
95      }
96
97      return ans;
98  }
99
100 int Dinic()
101 {
102     int ans;
103
104     ans = 0;
105     while(build()){
106         for(int i = 1; i <= tt; i ++){
107             cur[i] = head[i];
108         }
109         ans += dfs(ss, INF);
110     }
111
112     return ans;
113 }
114
115 int main(int argc, char const *argv[])
116 {
117     while(scanf("%d%d", &m, &n) == 2){
118         ss = 1;
119         tt = n;
120         ini();
121         for(int i = 1; i <= m; i ++){
122             int u, v, w;
123
124             scanf("%d%d%d", &u, &v, &w);
```

```
            add_edge(u, v, w);
            add_edge(v, u, 0);
        }
        printf("%d\n", Dinic());
    }

    return 0;
}
```

4.7.3 Hungary

```
const int N = 550;

int ma[N][N];
int used[N];    /// 另一边的使用情况
int ne[N];

bool Find(int x)
{
    for(int i = 1; i <= m; i ++){   /// 遍历另一边
        if(ma[x][i] && !used[i]){   /// 单向
            used[i] = 1;
            if(!ne[i] || Find(ne[i])){  /// 没有就直接上 || 如果有就腾开再上
                ne[i] = x;

                return true;
            }
        }
    }

    return false;
}

int match()
{
    int ans;

    ans = 0;
    memset(ne, 0, sizeof(ne));
    for(int i = 1; i <= n; i ++){   /// 遍历一边
        memset(used, 0, sizeof(used));
        if(Find(i)){
            ans ++;
        }
    }

    return ans;
}
```

4.7.4 Hungary bit

```
/// ¶□%□□□□Ż-
const int N = 550;
const int M = N / 32 + 1;   /// ¶□%□□□□□

int ma[N][M];
int b[M];               /// ¸¨□□ ne □□□□
int ne[N];
```

```
8  int n, m;
9  int tot;
10
11 inline void set1(int v[],int x){v[x>>5]|=1<<(x&31);}    /// μ□□□□ v[x] = 1
12 inline void flip(int v[],int x){v[x>>5]^=1<<(x&31);}    ///  °□□□□□□½□□□□□□□□□ · ´
13
14 bool Find(int x)
15 {
16     /// printf("%d %d\n", x, tot);
17     for(int i = 0; i <= tot; i ++){    /// □¶¨´□ 0 ¿ª□
18         while(true){
19             int t;
20             int q;
21
22             /// printf("%d %d %d %d\n", x, i, ma[x][i], b[i]);
23             t = ma[x][i] & b[i];
24             if(!t){
25                 break;
26             }
27             q = i << 5 | __builtin_ctz(t);  /// o ĩ□ 0 μ□□□□£¬x = 0 □½□□□¶¨□□
28             /// printf("%d %d\n", o, y);
29             flip(b, q);
30             if(!ne[q] || Find(ne[q])){
31                 ne[q] = x;
32                 return true;
33             }
34         }
35     }
36
37     return false;
38 }
39
40 int Match()
41 {
42     int ans;
43
44     ans = 0;
45     memset(ne, 0, sizeof(ne));
46     for(int i = 1; i <= n; i ++){   /// ±□□□□±□
47         for(int j = 1; j <= m/*(n???)*/; j ++){     /// ´□□¿¼
48             set1(b, j);
49         }
50         if(Find(i)){
51             ans ++;
52         }
53     }
54
55     return ans;
56 }
57
58 int main()
59 {
60     int ncase;
61
62     scanf("%d", &ncase);
63     while(ncase --){
64         memset(ma, 0, sizeof(ma));
65         scanf("%d%d", &m, &n);
66         tot = m >> 5;   /// □□°□□□³□ 32
```

```
67          for(int i = 1; i <= m; i ++){
68              int t;
69
70              scanf("%d", &t);
71              for(int j = 0; j < t; j ++){
72                  int x;
73
74                  scanf("%d", &x);
75                  set1(ma[x], i);
76                  /// ma[x][i] = 1;
77              }
78          }
79      }
80
81      return 0;
82  }
```

4.7.5 Isap

```
1   const int inf=0x7f7f7f7f;
2   const int maxn=1234; ////点数
3   const int maxm=123456;    ////边数
4   int n,m;
5   class Maxflow
6   {
7       int cnt;
8       int src,sink;           // 源点      汇点
9       int pre[maxn];          // 可增广路上的上一条弧的编号
10      int num[maxn];          // 和 t 的最短距离等于 i 的节点数量
11      int cur[maxm*2+10];          // 当前弧下标
12      int d[maxn];            // 残量网络中节点 i 到汇点 t 的最短距离
13      int Head[maxm*2+10];         //邻接表头
14      int visit[maxn];
15      bool visited[maxn];
16      struct Edge{
17          int from,to,val,nxt;
18      }edge[maxm*2+10];
19      // 预处理, 反向 BFS 构造 d 数组
20      public:
21      void addedge(int u,int v,int w)
22      {
23          cnt++;
24          edge[cnt].from=u;edge[cnt].to=v;edge[cnt].val=w;edge[cnt].nxt=Head[u];
25          //edge[cnt]=(Edge){u,v,w,Head[u]};
26          Head[u]=cnt;
27
28          cnt++;
29          edge[cnt].from=v;edge[cnt].to=u;edge[cnt].val=0;edge[cnt].nxt=Head[v];
30          //edge[cnt]=(Edge){v,u,0,Head[v]};
31          Head[v]=cnt;
32      }
33
34      void init()
35      {
36          memset(visit,0,sizeof(visit));
37          memset(Head,0,sizeof(Head));
38          cnt=1;src=1;sink=m;
39          for(int i=1;i<=n;i++)
```

```
40          {
41              int u,v,w;
42              scanf("%d%d%d",&u,&v,&w);
43              addedge(u,v,w);
44          }
45          ////加边
46      }
47
48      bool bfs()
49      {
50          memset(visited, 0, sizeof(visited));
51          queue<int> Q;
52          Q.push(sink);
53          visited[sink] = 1;
54          d[sink] = 0;
55          while (!Q.empty())
56          {
57              int u = Q.front();
58              Q.pop();
59              for (int i = Head[u]; i ; i=edge[i].nxt)
60              {
61                  Edge &e = edge[i^1]; ////引用反边
62
63                  if (!visited[e.from]) ////未访问&&有残量
64                  {
65                      visited[e.from] = true;
66                      d[e.from] = d[u] + 1;
67                      Q.push(e.from);
68                  }
69              }
70          }
71          return visited[src];
72      }
73
74      // 增广
75      int augment()
76      {
77          int u = sink, df = inf;
78          // 从汇点到源点通过 p 追踪增广路径, df 为一路上最小的残量
79          while (u != src)
80          {
81              df = min(df, edge[pre[u]].val);
82              u = edge[pre[u]].from;
83          }
84          u = sink;
85          // 从汇点到源点更新流量
86          while (u != src)
87          {
88              edge[pre[u]].val -= df;
89              edge[pre[u]^1].val += df;
90              u = edge[pre[u]].from;
91          }
92          return df;
93      }
94
95      int maxflow()
96      {
97          int flow = 0;
98          memset(num, 0, sizeof(num));
```

```
99          for(int i=0;i<=maxn;i++)cur[i]=Head[i];
100         bfs();
101         for (int i = 0; i < maxn; i++) num[d[i]]++; ////gap优化
102
103         int u = src;
104         while (d[src] < maxn)
105         {
106             if (u == sink)
107             {
108                 flow += augment();
109                 u = src;
110             }
111
112             bool advanced = false; ////判断是否增广成功
113             for (int i = cur[u]; i ; i=edge[i].nxt)
114             {
115                 Edge& e = edge[i];
116                 if (e.val && d[u] == d[e.to] + 1)
117                 {
118                     advanced = true;
119                     pre[e.to] = i;
120                     cur[u] = i; ////弧优化
121                     u = e.to;
122                     break;
123                 }
124             }
125             if (!advanced)
126             { // retreat
127                 int m = maxn - 1;
128                 for (int i = Head[u]; i ; i=edge[i].nxt)
129                 {
130                     if (edge[i].val)  ////如果有残量
131                     {
132                         m = min(m, d[edge[i].to]);
133                     }
134                 }
135
136                 if (--num[d[u]] == 0) break; // gap 优化
137                 num[d[u] = m+1]++;
138
139                 cur[u] = Head[u]; //弧优化部分
140                 if (u != src) u = edge[pre[u]].from;
141             }
142         }
143         return flow;
144     }
145 }TIM;
146 int main()
147 {
148     while(~scanf("%d%d",&n,&m))
149     {
150         TIM.init();
151         printf("%d\n",TIM.maxflow());
152     }
153 }
```

4.7.6 KM

```
1 const int N = 220;
```

```
const LL INF = 0x3f3f3f3f3f3f3f3f;

int n;
int ma[N][N];
LL lx[N],ly[N];
int linky[N];
LL pre[N];
bool vis[N];
bool visx[N];
bool visy[N];
LL slack[N];

void Find(int u)
{
    LL px, py;
    LL yy = 0;
    LL d;

    py = 0;
    yy = 0;
    memset(pre, 0, sizeof(pre));
    memset(slack, 0x3f3f3f3f, sizeof(slack));
    linky[py] = u;
    while(linky[py]){
        px = linky[py];
        d = INF;
        vis[py] = 1;
        for(int i = 1; i <= n; i++){
            if(!vis[i]){
                if(slack[i] > lx[px] + ly[i] - ma[px][i])
                    slack[i] = lx[px] + ly[i] -ma[px][i], pre[i]=py;
                if(slack[i]<d) d=slack[i],yy=i;
            }
        }
        for(int i = 0; i <= n; i++){
            if(vis[i]){
                lx[linky[i]] -= d;
                ly[i] += d;
            }
            else{
                slack[i] -= d;
            }
        }
        py = yy;
    }
    while(py){
        linky[py] = linky[pre[py]];
        py=pre[py];
    }
}

LL Km()
{
    LL ans;

    ans = 0;
    memset(lx, 0, sizeof(lx));
    memset(ly, 0, sizeof(ly));
    memset(linky, 0, sizeof(linky));
```

```
61      for(int i = 1; i <= n; i++){
62          memset(vis, 0, sizeof(vis));
63          Find(i);
64      }
65      for(int i = 1; i <= n; ++i){
66          ans += lx[i] + ly[i];
67      }
68
69      return ans;
70  }
71
72  int main(int argc, char const *argv[])
73  {
74      int ncase;
75      int nc;
76
77      nc = 1;
78      scanf("%d", &ncase);
79      while(ncase --){
80          scanf("%d", &n);
81          for(int i = 1; i <= n; i ++){
82              for(int j = 1; j <= n; j ++){
83                  scanf("%d", &ma[i][j]);
84                  ma[i][j] = -ma[i][j];
85              }
86          }
87          printf("Case #%d: %lld\n", nc, -Km());
88          nc ++;
89      }
90      return 0;
91  }
```

4.7.7 Upper-Lower Bound

上下界网络流建图方法

记号说明

- $f(u,v)$ 表示 $u \to v$ 的实际流量
- $b(u,v)$ 表示 $u \to v$ 的流量下界
- $c(u,v)$ 表示 $u \to v$ 的流量上界

无源汇可行流

建图

- 新建附加源点 S 和 T
- 原图中的边 $u \to v$，限制为 $[b,c]$，建边 $u \to v$，容量为 $c-b$
- 记 $d(i) = \sum b(u,i) - \sum b(i,v)$
- 若 $d(i) > 0$，建边 $S \to i$，流量为 $d(i)$
- 若 $d(i) < 0$，建边 $i \to T$，流量为 $-d(i)$

求解

- 跑 $S \to T$ 的最大流，如果满流，则原图存在可行流。
- 此时，原图中每一条边的流量为新图中对应边的流量加上这条边的下界。

有源汇可行流

建图

- 在原图中建边 $t \to s$，流量限制为 $[0, +\infty)$，这样就改造成了无源汇的网络流图。
- 之后就可以像求解无源汇可行流一样建图了。

求解 同无源汇可行流

有源汇最大流

建图 同有源汇可行流

求解

- 先跑一遍 $S \to T$ 的最大流，求出可行流
- 记此时 $\sum f(s, i) = sum_1$
- 将 $t \to s$ 这条边拆掉，在新图上跑 $s \to t$ 的最大流
- 记此时 $\sum f(s, i) = sum_2$
- 最终答案即为 $sum_1 + sum_2$

有源汇最小流

建图 同无源汇可行流

求解

- 求 $S \to T$ 最大流
- 建边 $t \to s$，容量为 $+\infty$
- 再跑一遍 $S \to T$ 的最大流，答案即为 $f(t, s)$

有源汇的最大流和最小流也可以通过二分答案求得，
即二分 $t \to s$ 的下界（最大流）和上界（最小流）复杂度多了个 $O(\log n)$ 这里不再赘述。

蓝书上的做法

- 先用无源汇可行流建图的方法求出可行流，然后用传统 $s - t$ 增广路算法即可得到最大流。把 t 看成源点，s 看成汇点后求出的 $t - s$ 最大流就是最小流。
- 注意：原先每条弧 $u \to v$ 的反向弧容量为 0，而在有容量下界的情形中，反向弧的容量应该等于流量下界。

有源汇费用流

建图

- 新建附加源点 S 和 T
- 原图中的边 $u \to v$，限制为 $[b, c]$，费用为 $cost$，建边 $u \to v$，容量为 $c - b$，费用为 $cost$
- 记 $d(i) = \sum b(u, i) - \sum b(i, v)$
- 若 $d(i) > 0$，建边 $S \to i$，流量为 $d(i)$，**费用为** 0
- 若 $d(i) < 0$，建边 $i \to T$，流量为 $-d(i)$，**费用为** 0
- 建边 $t \to s$，流量为 $+\infty$，费用为 0。

求解

- 跑 $S \to T$ 的最小费用最大流
- 答案为求出的费用加上原图中边的下界乘以边的费用

4.7.8 Upper and lower bound feasible flow

```
/// 若有源汇，先将汇点往源点流一条容量为 INF 的流(反向边正常建立)
int main(int argc, char const *argv[])
{
    int ncase;

    scanf("%d", &ncase);
    while(ncase --){
        scanf("%d%d", &n, &m);
        ini();
        ss = n + 1;
        tt = ss + 1;
        sum = 0;
        memset(cntf, 0, sizeof(cntf));
        for(int i = 1; i <= m; i ++){
            int from, to, upp; /// upp - 上界    low[i]  下界

            scanf("%d%d%d%d", &from, &to, &low[i], &upp);
            add_edge(from, to, upp - low[i], i);    ////
            add_edge(to, from, 0, i);
            cntf[from] -= low[i];
            cntf[to] += low[i];
        }
        for(int i = 1; i <= n; i ++){
            if(cntf[i] < 0){
                add_edge(i, tt, -cntf[i], 0);
                add_edge(tt, i, 0, 0);
            }
            else if(cntf[i] > 0){
                sum += cntf[i];
                add_edge(ss, i, cntf[i], 0);
                add_edge(i, ss, 0, 0);
            }
        }
        //cout << sum << endl;
        if(Dinic() == sum){
            printf("YES\n");
            for(int i = 1; i <= n; i ++){
                for(int j = head[i]; j != -1; j = node[j].ne){
                    if(!node[j].num || j % 2 == 0){
                        continue;
                    }
                    ans[node[j].num] = node[j].w + low[node[j].num];
                }
            }
            for(int i = 1; i <= m; i ++){
                printf("%d\n", ans[i]);
            }
        }
        else{
            printf("NO\n");
        }
    }

    return 0;
}
```

4.7.9 Mincostmaxflow

```
struct Node
{
    int from;
    int to;
    int ne;
    int flow;
    int v;
};

Node node[M * 10];
int n, m;
int ss, tt;
int ma[N];
int head[M];
int top;
/// SPFA
int dis[N];
int pre[N]; ///
int minn[N];
int mincost[N];
bool inq[N];
int Q[M * 10];
int ql, qr;

void Add_edge(int from, int to, int flow, int v)
{
    node[top].from = from;
    node[top].to = to;
    node[top].flow = flow;
    node[top].v = v;
    node[top].ne = head[from];
    head[from] = top;
    top ++;
}

void ini()
{
    top = 0;
    memset(head, -1, sizeof(head));
}

int Spfa()
{
    for(int i = 0; i <= tt; i ++){
        inq[i] = 0;
        dis[i] = INF;
        pre[i] = -1;
    }
    mincost[ss] = INF;
    mincost[tt] = 0;
    qr = ql = 0;
    dis[ss] = 0;
    minn[ss] = INF;
    inq[ss] = true;
    Q[qr] = ss;
    qr ++;
    while(ql != qr){
```

```
58          int now;
59
60          /// cout << 12312312312 << endl;
61          now = Q[ql];
62          /// cout << now << endl;
63          ql ++;
64          inq[now] = false;
65          for(int i = head[now]; i != -1; i = node[i].ne){
66              int v;
67
68              /// cout << 123123 << endl;
69              v = node[i].to;
70              if(node[i].flow > 0 && dis[v] > dis[now] + node[i].v){
71                  dis[v] = dis[now] + node[i].v;
72                  pre[v] = i; /// amazing  直接存边的编号
73                  /// pre[v] = u;
74                  mincost[v] = min(mincost[now], node[i].flow);
75                  if(!inq[v]){
76                      inq[v] = true;
77                      Q[qr] = v;
78                      qr ++;
79                  }
80              }
81          }
82      }
83
84      return mincost[tt];
85  }
86
87  int Mincostmaxflow()
88  {
89      int ans;
90      int t;
91
92      ans = 0;
93      while(true){
94          t = Spfa();
95          if(!t){
96              break;
97          }
98          /// cout << 666 << endl;
99          for(int i = pre[tt]; i != -1; i = pre[node[i].from]){
100             ans += t * node[i].v;
101             node[i].flow -= t;
102             node[i ^ 1].flow += t;
103         }
104     }
105
106     return ans;
107 }
```

4.7.10 Mincostmaxflow Dij

```
1  int he[N];
2
3  int Dij()
4  {
5      priority_queue< pair<int, int>, vector< pair<int, int> >, greater< pair<int, int> >
       > Q;
```

```
6       for(int i = 1; i <= tt; i ++){
7           dis[i] = INF;
8           pre[i] = -1;
9       }
10      dis[ss] = 0;
11      mincost[ss] = INF;
12      mincost[tt] = 0;
13      Q.push(mp(dis[ss], ss));
14      while(!Q.empty()){
15          int u, w;
16
17          u = Q.top().second;
18          w = Q.top().first;
19          Q.pop();
20          if(dis[u] < w){
21              continue;
22          }
23          for(int i = head[u]; i != -1; i = node[i].ne){
24              int v;
25
26              v = node[i].to;
27              if(node[i].flow > 0 && dis[v] > dis[u] + node[i].v + he[u] - he[v]){
28                  dis[v] = dis[u] + node[i].v + he[u] - he[v];
29                  pre[v] = i; /// amazing  ▯▯▯i▯▯▯
30                  /// pre[v] = u;
31                  mincost[v] = min(mincost[u], node[i].flow);
32                  Q.push(mp(dis[v], v));
33              }
34          }
35      }
36
37      return mincost[tt];
38  }
39
40  int Mincostmaxflow()
41  {
42      int ans;
43      int t;
44
45      ans = 0;
46      for(int i = 1; i <= tot; i ++){
47          he[i] = 0;
48      }
49      while(true){
50          t = Dij();
51          if(!t){
52              break;
53          }
54          for(int i = 1; i <= tot; i ++){
55              he[i] += dis[i];
56          }
57          maxflow += t;
58          /// cout << 666 << endl;
59          for(int i = pre[tt]; i != -1; i = pre[node[i].from]){
60              // printf("v = %d %d\n", node[i].v, t);
61              ans += t * node[i].v;
62              node[i].flow -= t;
63              node[i ^ 1].flow += t;
64          }
```

```
65      }
66
67      return ans;
68  }
```

4.8 Topolog

4.8.1 Topology

```
1   void Topology()
2   {
3       int li;
4       int Stop;
5
6       li = n * m;
7       Stop = 0;
8       memset(vis, 0, sizeof(vis));
9       for(int i = 0; i < li; i ++){
10          if(!inde[i]){
11              St[Stop] = i;
12              Stop ++;
13          }
14          else{
15              del[i] = 1;
16          }
17      }
18      while(Stop){
19          int now;
20
21          now = St[Stop - 1];
22          Stop --;
23          del[now] = 0;
24          for(int i = head[now]; i != -1; i = node[i].ne){
25              inde[node[i].to] --;
26              if(!inde[node[i].to]){
27                  St[Stop] = node[i].to;
28                  Stop ++;
29              }
30          }
31      }
32  }
```

5 Computational Geometry

5.1 Basic Function

```
#define zero(x) ((fabs(x) < eps ? 1 : 0))
#define sgn(x) (fabs(x) < eps ? 0 : ((x) < 0 ? -1 : 1))

struct point
{
    double x, y;
    point(double a = 0, double b = 0) { x = a, y = b; }
    point operator-(const point& b) const { return point(x - b.x, y - b.y); }
    point operator+(const point& b) const { return point(x + b.x, y + b.y); }
    // 两点是否重合
    bool operator==(point& b) { return zero(x - b.x) && zero(y - b.y); }
    // 点积(以原点为基准)
    double operator*(const point& b) const { return x * b.x + y * b.y; }
    // 叉积(以原点为基准)
    double operator^(const point& b) const { return x * b.y - y * b.x; }
    // 绕P点逆时针旋转a弧度后的点
    point rotate(point b, double a)
    {
        double dx, dy;
        (*this - b).split(dx, dy);
        double tx = dx * cos(a) - dy * sin(a);
        double ty = dx * sin(a) + dy * cos(a);
        return point(tx, ty) + b;
    }
    // 点坐标分别赋值到a和b
    void split(double& a, double& b) { a = x, b = y; }
};
struct line
{
    point s, e;
    line() {}
    line(point ss, point ee) { s = ss, e = ee; }
};
```

5.2 Position

5.2.1 Point-Point

```
double dist(point a, point b) { return sqrt((a - b) * (a - b)); }

//□□□□μ□□□
int dots_inline(point p1,point p2,point p3){
    return zero(xmult(p1,p2,p3));
}
int dots_inline(double x1,double y1,double x2,double y2,double x3,double y3){
    return zero(xmult(x1,y1,x2,y2,x3,y3));
}
```

5.2.2 Line-Line

```
// <0, *> 表示重合; <1, *> 表示平行; <2, P> 表示交点是P;
pair<int, point> spoint(line l1, line l2)
{
    point res = l1.s;
```

```
    if (sgn((l1.s - l1.e) ^ (l2.s - l2.e)) == 0)
        return {sgn((l1.s - l2.e) ^ (l2.s - l2.e)) != 0, res};
    double t = ((l1.s - l2.s) ^ (l2.s - l2.e)) / ((l1.s - l1.e) ^ (l2.s - l2.e));
    res.x += (l1.e.x - l1.s.x) * t;
    res.y += (l1.e.y - l1.s.y) * t;
    return {2, res};
}
```

5.2.3 Segment-Segment

```
bool segxseg(line l1, line l2)
{
    return
        max(l1.s.x, l1.e.x) >= min(l2.s.x, l2.e.x) &&
        max(l2.s.x, l2.e.x) >= min(l1.s.x, l1.e.x) &&
        max(l1.s.y, l1.e.y) >= min(l2.s.y, l2.e.y) &&
        max(l2.s.y, l2.e.y) >= min(l1.s.y, l1.e.y) &&
        sgn((l2.s - l1.e) ^ (l1.s - l1.e)) * sgn((l2.e-l1.e) ^ (l1.s - l1.e)) <= 0 &&
        sgn((l1.s - l2.e) ^ (l2.s - l2.e)) * sgn((l1.e-l2.e) ^ (l2.s - l2.e)) <= 0;
}

bool judge(Point &a,Point &b,Point &c,Point &d)
{
       /*
       ¿□□□□µ□
       }¸□□□□□¶□□□□□□□□ℓ□□□□¬□□□□□□}¸□□□□ú□□□□□µ□¿·⊞□□ô}□□□□□□□»¿□□□□□□□□µ□
       */
       if(!(min(a.x,b.x)<=max(c.x,d.x) && min(c.y,d.y)<=max(a.y,b.y)&&min(c.x,d.x)<=max
    (a.x,b.x) && min(a.y,b.y)<=max(c.y,d.y)))//□□□□□□□□□□¬□□□²½□□□□¨}¾□□□□□□□□□
       //1.□□□abµ□□□□□□□cdµ□□□□Ⓕ¿□□□□©□□ .2cdµ□□□□□□□□□abµ□□□□□□¨¿□□□□©□□
       //3.cdµ□□□□□□□□abµ□□□□Ⓕ¾□□□□□□1£¬}□□□□□□□□ ½□□□□□©□□ .4abµ□□□□□□□□□cdµ□□□□□□¨¾
    □□□□□□2£¬}□□□□□□ ·½□□□□□©□□
       //□□□□4¸□□□□£¬}□□□□□□□□ℓ□□□□□□□□□
       /*□□□□□□□□¸□□□□¬□□□□□¸□□□□□□□□□*/
       return false;
       /*
       ¿□b□□Ⓕ
       □□□}□□□□□□¬£□□ô±□□□□b£¬¾□□□□□□□□□□□□±□□£¬□□□□□□□□□}¶□□□¶¨□□□□□□□□□□}¶□
       □¾□□□□a b}µ□□□□□□cdµ□}¶□¬c d}µ□□□□□□abµ□}¶□
       */
    double u,v,w,z;// · □□¼}¸□□□-
    u=(c.x-a.x)*(b.y-a.y)-(b.x-a.x)*(c.y-a.y);
       v=(d.x-a.x)*(b.y-a.y)-(b.x-a.x)*(d.y-a.y);
       w=(a.x-c.x)*(d.y-c.y)-(d.x-c.x)*(a.y-c.y);
       z=(b.x-c.x)*(d.y-c.y)-(d.x-c.x)*(b.y-c.y);
       return (u*v<=0.00000001 && w*z<=0.00000001);
}
```

5.2.4 Line-Segment

```
//l1是直线,l2是线段
bool segxline(line l1, line l2)
{
    return sgn((l2.s - l1.e) ^ (l1.s - l1.e)) * sgn((l2.e - l1.e) ^ (l1.s - l1.e)) <=
    0;
}
```

5.2.5 Point-Line

```
double pointtoline(point p, line l)
{
    point res;
    double t = ((p - l.s) * (l.e - l.s)) / ((l.e - l.s) * (l.e - l.s));
    res.x = l.s.x + (l.e.x - l.s.x) * t, res.y = l.s.y + (l.e.y - l.s.y) * t;
    return dist(p, res);
}
```

5.2.6 Point-Segment

```
double pointtosegment(point p, line l)
{
    point res;
    double t = ((p - l.s) * (l.e - l.s)) / ((l.e - l.s) * (l.e - l.s));
    if (t >= 0 && t <= 1)
        res.x = l.s.x + (l.e.x - l.s.x) * t, res.y = l.s.y + (l.e.y - l.s.y) * t;
    else
        res = dist(p, l.s) < dist(p, l.e) ? l.s : l.e;
    return dist(p, res);
}
```

5.2.7 Point on Segment

```
bool PointOnSeg(point p, line l)
{
    return
        sgn((l.s - p) ^ (l.e-p)) == 0 &&
        sgn((p.x - l.s.x) * (p.x - l.e.x)) <= 0 &&
        sgn((p.y - l.s.y) * (p.y - l.e.y)) <= 0;
}
//□□□□□□□□□□□□□□□,°□(¶□□
int dot_online_in(point p,line l){
    return zero(xmult(p,l.a,l.b))&&(l.a.x-p.x)*(l.b.x-p.x)<eps&&(l.a.y-p.y)*(l.b.y-p.y)
    <eps;
}

//□□□□□□□□□□□□□□□,²»°□(¶□□
int dot_online_ex(point p,line l){
    return dot_online_in(p,l)&&(!zero(p.x-l.a.x)||!zero(p.y-l.a.y))&&(!zero(p.x-l.b.x)
    ||!zero(p.y-l.b.y));
}

//□□}µ□□□□□□□²□,µ□□□□□□□□□µ»□0
int same_side(point p1,point p2,line l){
    return xmult(l.a,p1,l.b)*xmult(l.a,p2,l.b)>eps;
}

//□□}µ□□□□□□□□□□,µ□□□□□□□□□µ»□0
int opposite_side(point p1,point p2,line l){
    return xmult(l.a,p1,l.b)*xmult(l.a,p2,l.b)<-eps;
}
```

5.3 Polygon

5.3.1 Area

```
double area(point p[], int n)
{
    double res = 0;
    for (int i = 0; i < n; i++) res += (p[i] ^ p[(i + 1) % n]) / 2;
    return fabs(res);
}
```

5.3.2 Point in Convex

```
// 点形成一个凸包，而且按逆时针排序(如果是顺时针把里面的<0改为>0)
// 点的编号 : [0,n)
// -1 : 点在凸多边形外
// 0  : 点在凸多边形边界上
// 1  : 点在凸多边形内
int PointInConvex(point a, point p[], int n)
{
    for (int i = 0; i < n; i++)
        if (sgn((p[i] - a) ^ (p[(i + 1) % n] - a)) < 0)
            return -1;
        else if (PointOnSeg(a, line(p[i], p[(i + 1) % n])))
            return 0;
    return 1;
}
```

5.3.3 Point in Polygon

```
// 射线法,poly[]的顶点数要大于等于3,点的编号0~n-1
// -1 : 点在凸多边形外
// 0  : 点在凸多边形边界上
// 1  : 点在凸多边形内
int PointInPoly(point p, point poly[], int n)
{
    int cnt;
    line ray, side;
    cnt = 0;
    ray.s = p;
    ray.e.y = p.y;
    ray.e.x = -100000000000.0; // -INF,注意取值防止越界
    for (int i = 0; i < n; i++)
    {
        side.s = poly[i], side.e = poly[(i + 1) % n];
        if (PointOnSeg(p, side)) return 0;
        //如果平行轴则不考虑
        if (sgn(side.s.y - side.e.y) == 0)
            continue;
        if (PointOnSeg(sid e.s, r ay))
            cnt += (sgn(side.s.y - side.e.y) > 0);
        else if (PointOnSeg(side.e, ray))
            cnt += (sgn(side.e.y - side.s.y) > 0);
        else if (segxseg(ray, side))
            cnt++;
    }
    return cnt % 2 == 1 ? 1 : -1;
}
```

5.3.4 Judge Convex

```
//点可以是顺时针给出也可以是逆时针给出
//点的编号1~n-1
bool isconvex(point poly[], int n)
{
    bool s[3];
    memset(s, 0, sizeof(s));
    for (int i = 0; i < n; i++)
    {
        s[sgn((poly[(i + 1) % n] - poly[i]) ^ (poly[(i + 2) % n] - poly[i])) + 1] = 1;
        if (s[0] && s[2]) return 0;
    }
    return 1;
}
```

5.3.5 Convex hull

```
const int maxn = 1e3 + 5;
struct Point {
    double x, y;
    Point(double x = 0, double y = 0):x(x),y(y){}
};
typedef Point Vector;
Point lst[maxn];
int stk[maxn], top;
Vector operator - (Point A, Point B){
    return Vector(A.x-B.x, A.y-B.y);
}
int sgn(double x){
    if(fabs(x) < eps)
        return 0;
    if(x < 0)
        return -1;
    return 1;
}
double Cross(Vector v0, Vector v1) {
    return v0.x*v1.y - v1.x*v0.y;
}
double Dis(Point p1, Point p2) { //¾□□□ p1p2µ□ ¾□□□
    return sqrt((p2.x-p1.x)*(p2.x-p1.x)+(p2.y-p1.y)*(p2.y-p1.y));
}
bool cmp(Point p1, Point p2) { //¾«½□□□□□□□ £¬½□□□□□□□□□□µ□□□□□□□
    int tmp = sgn(Cross(p1 - lst[0], p2 - lst[0]));
    if(tmp > 0)
        return true;
    if(tmp == 0 && Dis(lst[0], p1) < Dis(lst[0], p2))
        return true;
    return false;
}
//µ□ı□□0 ~ n - 1
// · µ»□□°□½□□stk[0 ~ top - 1]□□°□µı□□
void Graham(int n) {
    int k = 0;
    Point p0;
    p0.x = lst[0].x;
    p0.y = lst[0].y;
    for(int i = 1; i < n; ++i) {
        if( (p0.y > lst[i].y) || ((p0.y == lst[i].y) && (p0.x > lst[i].x)) ) {
            p0.x = lst[i].x;
```

```
43              p0.y = lst[i].y;
44              k = i;
45          }
46      }
47      lst[k] = lst[0];
48      lst[0] = p0;
49      sort(lst + 1, lst + n, cmp);
50      if(n == 1) {
51          top = 1;
52          stk[0] = 0;
53          return ;
54      }
55      if(n == 2) {
56          top = 2;
57          stk[0] = 0;
58          stk[1] = 1;
59          return ;
60      }
61      stk[0] = 0;
62      stk[1] = 1;
63      top = 2;
64      for(int i = 2; i < n; ++i) {
65          while(top > 1 && Cross(lst[stk[top - 1]] - lst[stk[top - 2]], lst[i] - lst[stk[
        top - 2]]) <= 0)
66              --top;
67          stk[top] = i;
68          ++top;
69      }
70      return ;
71  }
```

5.4 Integer Points

5.4.1 On Segment

```
1  int OnSegment(line l) { return __gcd(fabs(l.s.x - l.e.x), fabs(l.s.y - l.e.y)) + 1; }
```

5.4.2 On Polygon Edge

```
1  int OnEdge(point p[], int n)
2  {
3      int i, ret = 0;
4      for (i = 0; i < n; i++)
5          ret += __gcd(fabs(p[i].x - p[(i + 1) % n].x), fabs(p[i].y - p[(i + 1) % n].y));
6      return ret;
7  }
```

5.4.3 Inside Polygon

```
1  int InSide(point p[], int n)
2  {
3      int i, area = 0;
4      for (i = 0; i < n; i++)
5          area += p[(i + 1) % n].y * (p[i].x - p[(i + 2) % n].x);
6      return (fabs(area) - OnEdge(p, n)) / 2 + 1;
7  }
```

5.5 Circle

5.5.1 Circumcenter

```
point waixin(point a, point b, point c)
{
    double a1 = b.x - a.x, b1 = b.y - a.y, c1 = (a1 * a1 + b1 * b1) / 2;
    double a2 = c.x - a.x, b2 = c.y - a.y, c2 = (a2 * a2 + b2 * b2) / 2;
    double d = a1 * b2 - a2 * b1;
    return point(a.x + (c1 * b2 - c2 * b1) / d, a.y + (a1 * c2 - a2 * c1) / d);
}
```

5.5.2 Circle Line

```
inline NODE Vector(NODE A, NODE B);  //□□-AB
double cross(NODE A, NODE B, NODE P);
inline double dis2(NODE a, NODE b);
double disLine(NODE A, NODE B, NODE P);
double dot(NODE A, NODE B, NODE P);
NODE prxy(NODE A, NODE B, NODE O);
NODE Vbase(NODE A, NODE B, NODE O, int r);
int main()
{
    NODE A, B, O;
    double r;
    cin >> O.x >> O.y>>r;
    cin >> A.x >> A.y >> B.x >> B.y;
    NODE Base = Vbase(A, B, O, r);      //□□-base
    NODE pr = prxy(A, B, O);            //μ□pr
    NODE x1 = { Base.x + pr.x, Base.y + pr.y };
    NODE x2 = { pr.x - Base.x, pr.y - Base.y };
    if (disLine(A, B, O) > r)
        cout << "û□□»μ□" << endl;
    else
        cout << x1.x << ' ' << x1.y << ' ' << x2.x << ' ' << x2.y << endl;
    return 0;
}
NODE Vbase(NODE A,NODE B,NODE O,int r)    //□□□□-base
{
    double base = sqrt(r*r - disLine(A, B, O)*disLine(A, B, O));
    NODE AB = Vector(A, B);
    NODE e = { AB.x / sqrt(dis2(A, B)), AB.y / sqrt(dis2(A, B)) };
    return{ e.x*base, e.y*base };
}
NODE prxy(NODE A,NODE B,NODE O)   //□□□pr□□□□±□, □pr□□□□□O□½□□AB□¹□□
                                                                □
{
    NODE AO = Vector(A, O);
    NODE AB = Vector(A, B);
    double l = dot(A, B, O) / sqrt(dis2(A, B));
    NODE e = { AB.x / sqrt(dis2(A, B)), AB.y / sqrt(dis2(A, B)) };
    NODE Apr = { e.x*l, e.y*l };
    return{ A.x + Apr.x, A.y + Apr.y };
}
double disLine(NODE A, NODE B, NODE P)   //□P□½□□AB□l□□□□
                                                  □
{
    return fabs(cross(A,B,P))/sqrt(dis2(A,B));
}
double dot(NODE A, NODE B, NODE P)    //□□-AB,AP□□□□
```

```
{
    NODE AB = Vector(A, B);
    NODE AP = Vector(A, P);
    return AB.x*AP.x + AB.y*AP.y;
}
inline double dis2(NODE a, NODE b)            //µ□a,b¾□□□□□·½
{
    return (b.x - a.x)*(b.x - a.x) + (b.y - a.y)*(b.y - a.y);
}
double cross(NODE A, NODE B, NODE P)   //□□-AB□□□□-APµ□□□□
{
    NODE AB = Vector(A, B);
    NODE AP = Vector(A, P);
    return AB.x*AP.y - AB.y*AP.x;
}
inline NODE Vector(NODE A, NODE B)    //□□-AB
{
    return{ B.x - A.x, B.y - A.y };
}
```

5.5.3 Circle Circle

```
double arg(point p){return atan2(p.y,p.x);}
point polar(double a,double r)
{
    return point(cos(r)*a,sin(r)*a);
}

pair<point,point>getcrosspoints(point a1,double r1,point a2,double r2)
{
    double d=ABS(a2-a1);
    double a=acos((r1*r1+d*d-r2*r2)/(2*r1*d));
    double t=arg(a2-a1);
    return make_pair(a1+polar(r1,t+a),a1+polar(r1,t-a));
}
```

5.5.4 Min Curcke Cover

```
#define N 100005

const double pi=acos(-1.0);
const double eps=1e-9;
int dcmp(double x)
{
    if (x<=eps&&x>=-eps) return 0;
    return (x>0)?1:-1;
}
struct Vector
{
    double x,y;
    Vector(double X=0,double Y=0)
    {
        x=X,y=Y;
    }
};
typedef Vector Point;
Vector operator + (Vector a,Vector b) {return Vector(a.x+b.x,a.y+b.y);}
Vector operator - (Vector a,Vector b) {return Vector(a.x-b.x,a.y-b.y);}
```

```
Vector operator * (Vector a,double b) {return Vector(a.x*b,a.y*b);}
Vector operator / (Vector a,double b) {return Vector(a.x/b,a.y/b);}

int n;
double r;
Point c,p[N];

double Dot(Vector a,Vector b)
{
    return a.x*b.x+a.y*b.y;
}
double Cross(Vector a,Vector b)
{
    return a.x*b.y-a.y*b.x;
}
double Len(Vector a)
{
    return sqrt(Dot(a,a));
}
Vector rotate(Vector a,double rad)
{
    return Vector(a.x*cos(rad)-a.y*sin(rad),a.x*sin(rad)+a.y*cos(rad));
}
Point GLI(Point P,Vector v,Point Q,Vector w)
{
    Vector u=P-Q;
    double t=Cross(w,u)/Cross(v,w);
    return P+v*t;
}
Point TC(Point A,Point B,Point C)
{
    Point P=(A+B)/2,Q=(A+C)/2;
    Vector v=rotate(B-A,pi/2),w=rotate(C-A,pi/2);
    if (dcmp(Len(Cross(v,w)))==0)
    {
        if (dcmp(Len(A-B)+Len(B-C)-Len(A-C))==0)
            return (A+C)/2;
        if (dcmp(Len(A-C)+Len(B-C)+Len(A-B))==0)
            return (A+B)/2;
        if (dcmp(Len(A-B)+Len(A-C)-Len(B-C))==0)
            return (B+C)/2;
    }
    return GLI(P,v,Q,w);
}
void mcc()
{
    random_shuffle(p+1,p+n+1);
    c=p[1],r=0;
    for (int i=2;i<=n;++i)
        if (dcmp(Len(c-p[i])-r)>0)
        {
            c=p[i],r=0;
            for (int j=1;j<i;++j)
                if (dcmp(Len(c-p[j])-r)>0)
                {
                    c=(p[i]+p[j])/2,r=Len(c-p[i]);
                    for (int k=1;k<j;++k)
                        if (dcmp(Len(c-p[k])-r)>0)
                        {
```

```
                            c=TC(p[i],p[j],p[k]);
                            r=Len(c-p[i]);
                        }
                    }
                }
}
int main()
{
    scanf("%d",&n);
    for (int i=1;i<=n;++i) scanf("%lf%lf",&p[i].x,&p[i].y);
    mcc();
    printf("%.3lf\n",r);
}
```

5.6 RuJia Liu's

5.6.1 Point

```
struct Point
{
    double x, y;
    Point(double x = 0, double y = 0) : x(x), y(y) {}
};

typedef Point Vector;

//向量+向量=向量，点+向量=点
Vector operator+(Vector A, Vector B) { return Vector(A.x + B.x, A.y + B.y); }
//点-点=向量
Vector operator-(Point A, Point B) { return Vector(A.x - B.x, A.y - B.y); }
//向量*数=向量
Vector operator*(Vector A, double p) { return Vector(A.x * p, A.y * p); }
//向量/数=向量
Vector operator/(Vector A, double p) { return Vector(A.x / p, A.y / p); }

bool operator<(const Point& a, const Point& b)
{
    return a.x < b.x || (a.x == b.x && a.y < b.y);
}

const double eps = 1e-10;
double dcmp(double x)
{
    if (fabs(x) < eps)
        return 0;
    else
        return x < 0 ? -1 : 1;
}

bool operator==(const Point& a, const Point& b)
{
    return dcmp(a.x - b.x) == 0 && dcmp(a.y - b.y) == 0;
}

/*
 * 基本运算:
 * 点积
 * 叉积
 * 向量旋转
```

```
 */
double Dot(Vector A, Vector B) { return A.x * B.x + A.y * B.y; }
double Length(Vector A) { return sqrt(Dot(A, A)); }
double Angle(Vector A, Vector B) { return acos(Dot(A, B) / Length(A) / Length(B)); }

double Cross(Vector A, Vector B) { return A.x * B.y - A.y * B.x; }
double Area2(Point A, Point B, Point C) { return Cross(B - A, C - A); }

//rad是弧度
Vector Rotate(Vector A, double rad)
{
    return Vector(A.x * cos(rad) - A.y * sin(rad),
                  A.x * sin(rad) + A.y * cos(rad));
}

//调用前请确保A不是零向量
Vector Normal(Vector A)
{
    double L = Length(A);
    return Vector(-A.y / L, A.x / L);
}

/*
 * 点和直线:
 * 两直线交点
 * 点到直线的距离
 * 点到线段的距离
 * 点在直线上的投影
 * 线段相交判定
 * 点在线段上判定
 */

//调用前保证两条直线P+tv和Q+tw有唯一交点。当且仅当Cross(v, w)非0
Point GetLineIntersection(Point P, Vector v, Point Q, Vector w)
{
    Vector u = P - Q;
    double t = Cross(w, u) / Cross(v, w);
    return P + v * t;
}

double DistanceToLine(Point P, Point A, Point B)
{
    Vector v1 = B - A, v2 = P - A;
    return fabs(Cross(v1, v2)) / Length(v1); //如果不取绝对值，得到的是有向距离
}

double DistanceToSegment(Point P, Point A, Point B)
{
    if (A == B) return Length(P - A);
    Vector v1 = B - A, v2 = P - A, v3 = P - B;
    if (dcmp(Dot(v1, v2)) < 0) return Length(v2);
    if (dcmp(Dot(v1, v3)) > 0) return Length(v3);
    return fabs(Cross(v1, v2)) / Length(v1);
}

Point GetLineProjection(Point P, Point A, Point B)
{
    Vector v = B - A;
    return A + v * (Dot(v, P - A) / Dot(v, v));
```

```
}

bool SegmentProperIntersection(Point a1, Point a2, Point b1, Point b2)
{
    double c1 = Cross(a2 - a1, b1 - a1), c2 = Cross(a2 - a1, b2 - b1),
           c3 = Cross(b2 - b1, a1 - b1), c4 = Cross(b2 - b1, a2 - b1);
    return dcmp(c1) * dcmp(c2) < 0 && dcmp(c3) * dcmp(c4) < 0;
}

bool OnSegment(Point p, Point a1, Point a2)
{
    return dcmp(Cross(a1 - p, a2 - p)) == 0 && dcmp(Dot(a1 - p, a2 - p)) < 0;
}
```

5.6.2 Circle

```
struct Line
{
    Point p;    //直线上任意一点
    Vector v;   //方向向量。它的左边就是对应的半平面
    double ang; //极角。即从x正半轴旋转到向量v所需要的角（弧度）
    Line() {}
    Line(Point p, Vector v) : p(p), v(v) { ang = atan2(v.y, v.x); }
    bool operator<(const Line& L) const // 排序用的比较运算符
    {
        return ang < L.ang;
    }
    Point point(double t) { return p + v * t; }
};

struct Circle
{
    Point c;
    double r;
    Circle(Point c, double r) : c(c), r(r) {}
    Point point(double a) { return c.x + cos(a) * r, c.y + sin(a) * r; }
};

int getLineCircleIntersection(Line L, Circle C, double& t1, double& t2, vector<Point>&
    sol)
{
    double a = L.v.x, b = L.p.x - C.c.x, c = L.v.y, d = L.p.y - C.c.y;
    double e = a * a + c * c, f = 2 * (a * b + c * d), g = b * b + d * d - C.r * C.r;
    double delta = f * f - 4 * e * g; //判别式
    if (dcmp(delta) < 0) return 0;    //相离
    if (dcmp(delta) == 0)             //相切
    {
        t1 = t2 = -f / (2 * e);
        sol.push_back(L.point(t1));
        return 1;
    }
    //相交
    t1 = (-f - sqrt(delta)) / (2 * e);
    t2 = (-f + sqrt(delta)) / (2 * e);
    sol.push_back(t1);
    sol.push_back(t2);
    return 2;
}
```

```cpp

double angle(Vector v) { return atan2(v.y, v.x); }

int getCircleCircleIntersection(Circle C1, Circle C2, vector<Point>& sol)
{
    double d = Length(C1.c - C2.c);
    if (dcmp(d) == 0)
    {
        if (dcmp(C1.r - C2.r) == 0) return -1; //两圆重合
        return 0;
    }
    if (dcmp(C1.r + C2.r - d) < 0) return 0;       //内含
    if (dcmp(fabs(C1.r - C2.r) - d) > 0) return 0; //外离

    double a = angle(C2.c - C1.c); //向量C1C2的极角
    double da = acos((C1.r * C1.r + d * d - C2.r * C2.r) / (2 * C1.r * d));
    //C1C2到C1P1的角
    Point p1 = C1.point(a - da), p2 = C1.point(a + da);

    sol.push_back(p1);
    if (p1 == p2) return 1;
    sol.push_back(p2);
    return 2;
}

//过点p到圆C的切线，v[i]是第i条切线的向量，返回切线条数
int getTangents(Point p, Circle C, Vector* v)
{
    Vector u = C.c - p;
    double dist = Length(u);
    if (dist < C.r)
        return 0;
    else if (dcmp(dist - C.r) == 0)
    { //p在圆上，只有一条切线
        v[0] = Rotate(u, M_PI / 2);
        return 1;
    }
    else
    {
        double ang = asin(C.r / dist);
        v[0] = Rotate(u, -ang);
        v[1] = Rotate(u, +ang);
        return 2;
    }
}

//两圆的公切线
//返回切线的条数。-1表示无穷条切线。
//a[i]和b[i]分别是第i条切线在圆A和圆B上的切点
int getTangents(Circle A, Circle B, Point* a, Point* b)
{
    int cnt = 0;
    if (A.r < B.r)
    {
        swap(A, B);
        swap(a, b);
    }
    int d2 = (A.c.x - B.c.x) * (A.c.x - B.c.x) + (A.c.y - B.c.y) * (A.c.y - B.c.y);
    int rdiff = A.r - B.r;
```

```
    int rsum = A.r + B.r;
    if (d2 < rdiff * rdiff) return 0; //内含
    double base = atan2(B.c.y - A.c.y, B.c.x - A.c.x);
    if (d2 == 0 && A.r == B.r) return -1; //无限多条切线
    if (d2 == rdiff * rdiff)
    { //内切，一条切线
        a[cnt] = A.point(base);
        b[cnt] = B.point(base);
        cnt++;
        return 1;
    }
    //有外共切线
    double ang = acos(A.r - B.r) / sqrt(d2);
    a[cnt] = A.point(base + ang);
    b[cnt] = B.point(base + ang);
    cnt++;
    a[cnt] = A.point(base + ang);
    b[cnt] = B.point(base - ang);
    cnt++;
    if (d2 == rsum * rsum)
    {
        a[cnt] = A.point(base);
        b[cnt] = B.point(M_PI + base);
        cnt++;
    }
    else if (d2 > rsum * rsum)
    {
        double ang = acos((A.r + B.r) / sqrt(d2));
        a[cnt] = A.point(base + ang);
        b[cnt] = B.point(M_PI + base + ang);
        cnt++;
        a[cnt] = A.point(base - ang);
        b[cnt] = B.point(M_PI + base - ang);
        cnt++;
    }
    return cnt;
}

//三角形外接圆（三点保证不共线）
Circle CircumscribedCircle(Point p1, Point p2, Point p3)
{
    double Bx = p2.x - p1.x, By = p2.y - p1.y;
    double Cx = p3.x - p1.x, Cy = p3.y - p1.y;
    double D = 2 * (Bx * Cy - By * Cx);
    double cx = (Cy * (Bx * Bx + By * By) - By * (Cx * Cx + Cy * Cy)) / D + p1.x;
    double cy = (Bx * (Cx * Cx + Cy * Cy) - Cx * (Bx * Bx + By * By)) / D + p1.y;
    Point p = Point(cx, cy);
    return Circle(p, Length(p1 - p));
}

//三角形内切圆
Circle InscribedCircle(Point p1, Point p2, Point p3)
{
    double a = Length(p2 - p3);
    double b = Length(p3 - p1);
    double c = Length(p1 - p2);
    Point p = (p1 * a + p2 * b + p3 * c) / (a + b + c);
    return Circle(p, DistanceToLine(p, p1, p2));
}
```

5.6.3 Polygon

```
typedef vector<Point> Polygon;
//多边形的有向面积
double PolygonArea(Polygon po)
{
    int n = po.size();
    double area = 0.0;
    for (int i = 1; i < n - 1; i++)
        area += Cross(po[i] - po[0], po[i + 1] - po[0]);
    return area / 2;
}

//点在多边形内判定
int isPointInPolygon(Point p, Polygon poly)
{
    int wn = 0; //绕数
    int n = poly.size();
    for (int i = 0; i < n; i++)
    {
        if (OnSegment(p, poly[i], poly[(i + 1) % n])) return -1; //边界上
        int k = dcmp(Cross(poly[(i + 1) % n] - poly[i], p - poly[i]));
        int d1 = dcmp(poly[i].y - p.y);
        int d2 = dcmp(poly[(i + 1) % n].y - p.y);
        if (k > 0 && d1 <= 0 && d2 > 0) wn++;
        if (k < 0 && d2 <= 0 && d1 > 0) wn--;
    }
    if (wn != 0) return 1; //内部
    return 0;              //外部
}

//凸包(Andrew算法)
//如果不希望在凸包的边上有输入点，把两个 <= 改成 <
//如果不介意点集被修改，可以改成传递引用
Polygon ConvexHull(vector<Point> p)
{
    sort(p.begin(), p.end());
    p.erase(unique(p.begin(), p.end()), p.end());
    int n = p.size(), m = 0;
    Polygon res(n + 1);
    for (int i = 0; i < n; i++)
    {
        while (m > 1 && Cross(res[m - 1] - res[m - 2], p[i] - res[m - 2]) <= 0) m--;
        res[m++] = p[i];
    }
    int k = m;
    for (int i = n - 2; i >= 0; i--)
    {
        while (m > k && Cross(res[m - 1] - res[m - 2], p[i] - res[m - 2]) <= 0) m--;
        res[m++] = p[i];
    }
    m -= n > 1;
    res.resize(m);
    return res;
}

//半平面交
vector<Point> HalfplaneIntersection(vector<Line>& L)
{
```

```
58      int n = L.size();
59      sort(L.begin(), L.end()); // 按极角排序
60
61      int first, last;    // 双端队列的第一个元素和最后一个元素的下标
62      vector<Point> p(n); // p[i]为q[i]和q[i+1]的交点
63      vector<Line> q(n);  // 双端队列
64      vector<Point> ans;  // 结果
65
66      q[first = last = 0] = L[0]; // 双端队列初始化为只有一个半平面L[0]
67      for (int i = 1; i < n; i++)
68      {
69          while (first < last && !OnLeft(L[i], p[last - 1])) last--;
70          while (first < last && !OnLeft(L[i], p[first])) first++;
71          q[++last] = L[i];
72          if (fabs(Cross(q[last].v, q[last - 1].v)) < eps)
73          { // 两向量平行且同向，取内侧的一个
74              last--;
75              if (OnLeft(q[last], L[i].p)) q[last] = L[i];
76          }
77          if (first < last) p[last - 1] = GetLineIntersection(q[last - 1], q[last]);
78      }
79      while (first < last && !OnLeft(q[first], p[last - 1])) last--; // 删除无用平面
80      if (last - first <= 1) return vector<Point>();                 // 空集
81      p[last] = GetLineIntersection(q[last], q[first]);              // 计算首尾两个半平面的
        交点
82
83      return vector<Point>(q.begin() + first, q.begin() + last + 1);
84  }
```

5.7 XCQQ!

5.7.1 Minimum width

```
1   void fre() {  }
2   #define MS(x, y) memset(x, y, sizeof(x))
3   #define ls o<<1
4   #define rs o<<1|1
5   typedef long long LL;
6   typedef unsigned long long UL;
7   typedef unsigned int UI;
8   template <class T1, class T2>inline void gmax(T1 &a, T2 b) { if (b > a)a = b; }
9   template <class T1, class T2>inline void gmin(T1 &a, T2 b) { if (b < a)a = b; }
10  const int N = 2e5 + 10, M = 0, Z = 1e9 + 7, inf = 0x3f3f3f3f;
11  template <class T1, class T2>inline void gadd(T1 &a, T2 b) { a = (a + b) % Z; }
12  int casenum, casei;
13  LL sqr(LL x)
14  {
15      return x * x;
16  }
17  struct point
18  {
19      LL x, y;
20      point(){}
21      point(LL x, LL y) : x(x), y(y) {}
22      friend point operator + (const point &a, const point &b){
23          return point(a.x + b.x, a.y + b.y);
24      }
25      friend point operator - (const point &a, const point &b){
26          return point(a.x - b.x, a.y - b.y);
```

```
27      }
28      friend point operator * (const point &a, const double &b){
29          return point(a.x * b, a.y * b);
30      }
31      friend point operator / (const point &a, const double &b){
32          return point(a.x / b, a.y / b);
33      }
34      friend bool operator == (const point &a, const point &b){
35          return a.x == b.x && a.y == b.y;
36      }
37  };
38  LL det(point a, point b)
39  {
40      return a.x * b.y - a.y * b.x;
41  }
42  LL dot(point a, point b)
43  {
44      return a.x * b.x + a.y * b.y;
45  }
46  LL dist(point a, point b)
47  {
48      return sqr(a.x - b.x) + sqr(a.y - b.y);
49  }
50  struct polygon_convex
51  {
52      vector<point> p;
53      polygon_convex(int size = 0){
54          p.resize(size);
55      }
56  };
57  bool comp_less(const point &a, const point &b)
58  {
59      return a.x - b.x < 0 || a.x - b.x == 0 && a.y - b.y < 0;
60  }
61  polygon_convex convex_hull(vector<point> a)
62  {
63      polygon_convex res(2 * a.size() + 5);
64      sort(a.begin(), a.end(), comp_less);
65      a.erase(unique(a.begin(), a.end()), a.end());
66      int m = 0;
67      for(int i = 0; i < a.size(); i ++){
68          while(m > 1 && det(res.p[m - 1] - res.p[m - 2], a[i] - res.p[m - 2]) <= 0) -- m
    ;
69          res.p[m ++] = a[i];
70      }
71      int k = m;
72      for(int i = int(a.size()) - 2; i >= 0; i --){
73          while(m > k && det(res.p[m - 1] - res.p[m - 2], a[i] - res.p[m - 2]) <= 0) -- m
    ;
74          res.p[m ++] = a[i];
75      }
76      res.p.resize(m);
77      if(a.size() > 1) res.p.resize(m - 1);
78      return res;
79  }
80
81  LL cross(point a, point b, point c)
82  {
83      return (c.x - a.x) * (b.y - a.y) - (b.x - a.x) * (c.y - a.y);
```

```
}
int R;

double rotating_calipers(point *p, int n)
{
    int U = 1;
    double ans = 2.0 * R;
    p[n] = p[0];
    for(int i = 0; i < n; i ++){
        while(cross(p[i], p[i + 1], p[U + 1]) - cross(p[i], p[i + 1], p[U]) <= 0) U = (U + 1) % n;
        double d = sqrt(dist(p[i], p[i + 1]));
        double h = 1.0 * fabs(cross(p[i], p[i + 1], p[U])) / d;
        gmin(ans, h);
    }
    return ans;
}
int n;
point t, p[N];
polygon_convex a;
int main()
{
    scanf("%d%d", &n, &R);
    for(int i = 0; i < n; i ++){
        scanf("%lld%lld", &t.x, &t.y);
        a.p.push_back(t);
    }
    a = convex_hull(a.p);
    n = a.p.size();
    for(int i = 0; i < n; i ++){
        p[i] = a.p[i];
    }
    double ans;
    if(n <= 2){
        printf("0.000000000");
    }
    else {
        ans = rotating_calipers(p, n);
        printf("%.10f\n", ans);
    }
    return 0;
}
```

5.7.2 Minimum length

```
/// □□μ□□ □¶□□□□Ƶ¶
void Solve2(int num)
{
    int ymax=-1e5,ymin=1e5;
    int ymaxidx,yminidx;
    for(int i=1;i<=num;i++)
    {
        if(ch[i].y>ymax)
        {
            ymax=ch[i].y;
            ymaxidx=i;
        }
        if(ch[i].y<ymin)
```

```
14          {
15              ymin=ch[i].y;
16              yminidx=i;
17          }
18      }
19      int ans=dis2(ch[ymaxidx]-ch[yminidx]);
20      ch[num+1]=ch[1];
21      for(int t=1;t<=num;t++,yminidx=yminidx%num+1)
22      {
23          while(xmult(ch[yminidx+1],ch[ymaxidx+1],ch[yminidx])>xmult(ch[yminidx+1],ch[
        ymaxidx],ch[yminidx]))ymaxidx=ymaxidx%num+1;
24          ans=max(ans,dis2(ch[ymaxidx]-ch[yminidx]));
25          ans=max(ans,dis2(ch[ymaxidx]-ch[yminidx+1]));
26      }
27      printf("%d\n",ans);
28  }
```

5.8 Ohters

5.8.1 Polygon area sum

```
1   /*
2    * ¶□□□□□»£¬¶□□□□ı□□¶¨□□□°´□□□□Ⓕ□□□³□
3    * »¹□□□□□□□°□»¹□□¼°□£¬µ□□□□□□µÍ¯□□
4    * □□□²¢£□□°□□□□¼□□½»¼´¿□
5    */
6   const int maxn = //300;
7   const double eps = 1e-8;
8   int dcmp(double x)
9   {
10      if(x > eps) return 1;
11      return x < -eps ? -1 : 0;
12  }
13  struct Point
14  {
15      double x, y;
16  };
17  double cross(Point a,Point b,Point c) ///²□□
18  {
19      return (a.x-c.x)*(b.y-c.y)-(b.x-c.x)*(a.y-c.y);
20  }
21  Point intersection(Point a,Point b,Point c,Point d)
22  {
23      Point p = a;
24      double t =((a.x-c.x)*(c.y-d.y)-(a.y-c.y)*(c.x-d.x))/((a.x-b.x)*(c.y-d.y)-(a.y-b.y)
        *(c.x-d.x));
25      p.x +=(b.x-a.x)*t;
26      p.y +=(b.y-a.y)*t;
27      return p;
28  }
29  //¼□□□□□□□□□□
30  double PolygonArea(Point p[], int n)
31  {
32      if(n < 3) return 0.0;
33      double s = p[0].y * (p[n - 1].x - p[1].x);
34      p[n] = p[0];
35      for(int i = 1; i < n; ++ i)
36          s += p[i].y * (p[i - 1].x - p[i + 1].x);
```

```
37      return fabs(s * 0.5);
38  }
39  double CPIA(Point a[], Point b[], int na, int nb)//ConvexPolygonIntersectArea
40  {
41      Point p[20], tmp[20];
42      int tn, sflag, eflag;
43      a[na] = a[0], b[nb] = b[0];
44      memcpy(p,b,sizeof(Point)*(nb + 1));
45      for(int i = 0; i < na && nb > 2; i++)
46      {
47          sflag = dcmp(cross(a[i + 1], p[0],a[i]));
48          for(int j = tn = 0; j < nb; j++, sflag = eflag)
49          {
50              if(sflag>=0) tmp[tn++] = p[j];
51              eflag = dcmp(cross(a[i + 1], p[j + 1],a[i]));
52              if((sflag ^ eflag) == -2)
53                  tmp[tn++] = intersection(a[i], a[i + 1], p[j], p[j + 1]); ///□□□
54          }
55          memcpy(p, tmp, sizeof(Point) * tn);
56          nb = tn, p[nb] = p[0];
57      }
58      if(nb < 3) return 0.0;
59      return PolygonArea(p, nb);
60  }
61  double SPIA(Point a[], Point b[], int na, int nb)///SimplePolygonIntersectArea µ□□ô□¯□□
62  {
63      int i, j;
64      Point t1[4], t2[4];
65      double res = 0, num1, num2;
66      a[na] = t1[0] = a[0], b[nb] = t2[0] = b[0];
67      for(i = 2; i < na; i++)
68      {
69          t1[1] = a[i-1], t1[2] = a[i];
70          num1 = dcmp(cross(t1[1], t1[2],t1[0]));
71          if(num1 < 0) swap(t1[1], t1[2]);
72          for(j = 2; j < nb; j++)
73          {
74              t2[1] = b[j - 1], t2[2] = b[j];
75              num2 = dcmp(cross(t2[1], t2[2],t2[0]));
76              if(num2 < 0) swap(t2[1], t2[2]);
77              res += CPIA(t1, t2, 3, 3) * num1 * num2;
78          }
79      }
80      return res;
81  }
82  Point p1[maxn], p2[maxn];
83  int n1, n2;
84  int main()
85  {
86      while(cin>>n1>>n2)
87      {
88          for(int i = 0; i < n1; i++) scanf("%lf%lf", &p1[i].x, &p1[i].y);
89          for(int i = 0; i < n2; i++) scanf("%lf%lf", &p2[i].x, &p2[i].y);
90          double Area = SPIA(p1, p2, n1, n2);
91      }
92      return 0;
93  }
```

6 Others

6.1 Misc

6.1.1 Policy-Based Data Structures

红黑树

声明/头文件

```
#include <ext/pb_ds/tree_policy.hpp>
#include <ext/pb_ds/assoc_container.hpp>
using namespace __gnu_pbds;
typedef tree<pt, null_type, less<pt>, rb_tree_tag, tree_order_statistics_node_update>
    rbtree;
```

使用方法

```
pt                                  // 关键字类型
null_type                           // 无映射(低版本g++为null_mapped_type)
less<int>                           // 从小到大排序
rb_tree_tag                         // 红黑树 (splay_tree_tag)
tree_order_statistics_node_update   // 结点更新
T.insert(val);                      // 插入
T.erase(iterator);                  // 删除
T.order_of_key();                   // 查找有多少数比它小
T.find_by_order(k);                 // 有k个数比它小的数是多少
a.join(b);                          // b并入a 前提是两棵树的key的取值范围不相交
a.split(v, b);                      // key小于等于v的元素属于a, 其余的属于b
T.lower_bound(x);                   // >=x的min的迭代器
T.upper_bound((x);                  // >x的min的迭代器
```

6.1.2 Subset Enumeration

枚举真子集

```
for (int s = (S - 1) & S; s; s = (s - 1) & S)
```

枚举大小为 k 的子集

```
void subset(int k, int n)
{
    int t = (1 << k) - 1;
    while (t < (1 << n))
    {
        // do something
        int x = t & -t, y = t + x;
        t = ((t & ~y) / x >> 1) | y;
    }
}
```

6.1.3 Date Magic

```
string dayOfWeek[] = {"Mo", "Tu", "We", "Th", "Fr", "Sa", "Su"};

// converts Gregorian date to integer (Julian day number)
int DateToInt(int m, int d, int y)
{
    return 1461 * (y + 4800 + (m - 14) / 12) / 4
```

```
7             + 367 * (m - 2 - (m - 14) / 12 * 12) / 12
8             - 3 * ((y + 4900 + (m - 14) / 12) / 100) / 4
9             + d - 32075;
10 }
11
12 // converts integer (Julian day number) to Gregorian date: month/day/year
13 void IntToDate(int jd, int& m, int& d, int& y)
14 {
15     int x, n, i, j;
16     x = jd + 68569;
17     n = 4 * x / 146097;
18     x -= (146097 * n + 3) / 4;
19     i = (4000 * (x + 1)) / 1461001;
20     x -= 1461 * i / 4 - 31;
21     j = 80 * x / 2447;
22     d = x - 2447 * j / 80;
23     x = j / 11;
24     m = j + 2 - 12 * x;
25     y = 100 * (n - 49) + i + x;
26 }
27
28 // converts integer (Julian day number) to day of week
29 string IntToDay(int jd) { return dayOfWeek[jd % 7]; }
```