Contents

1 Basic

1.1 default code

```cpp
#include <bits/stdc++.h>
#define PB push_back
#define MP make_pair
#define F first
#define S second
#define SZ(x) ((int)(x).size())
#define ALL(x) (x).begin(),(x).end()
#ifdef _DEBUG_
  #define debug(...) printf(__VA_ARGS__)
#else
  #define debug(...) (void)0
#endif
using namespace std;
typedef long long ll;
typedef pair<int,int> PII;
typedef vector<int> VI;

int main() {
  return 0;
}
```

1.2 .vimrc

```
color torte
syn on
set guifont=Consolas:h16: nu sc ai si ts=4 sm sts=4 sw=4

map <F9> <ESC>:w<CR>:!g++ % -o %< -O2 -Wall -Wno-unused-result -std=c++0x<CR>
map <S-F9> <ESC>:w<CR>:!g++ % -o %< -O2 -Wall -Wno-unused-result -D_DEBUG_ -std=c++0x<CR>
map <F5> <ESC>:!./%<<CR>
map <F6> <ESC>:w<CR>ggvG"+y
map <S-F5> <ESC>:!./%< < %<.in<CR>
imap <Home> <ESC>^i
com INPUT sp %<.in
```

2 math

2.1 ext gcd

```cpp
// find one solution (x,y) of ax+by=gcd(a,b)
void ext_gcd(int a,int b,int &g,int &x,int &y)
{
  if(!b){ g=a; x=1; y=0; }
  else{ ext_gcd(b, a%b, g, y, x); y -= x*(a/b); }
}
```

2.2 FFT other

```cpp
/* FFT code from shik in CodeForces*/
/* zj a577*/
#include <bits/stdc++.h>
using namespace std;
const int N=300000;

const double PI=acos(-1.0);
struct vir{
    double re,im;
    vir( double _re=0, double _im=0 ):re(_re),im(_im){}
};
vir operator +( vir a, vir b ) { return vir(a.re+b.re,a.im+b.im); }
vir operator -( vir a, vir b ) { return vir(a.re-b.re,a.im-b.im); }
vir operator *( vir a, vir b ) { return vir(a.re*b.re-a.im*b.im,a.re*b.im+a.im*b.re); }
vir x1[2*N],x2[2*N];

int rev( int x, int len ) {
    int r=0,i;
    for ( i=0; i<len; i++,x>>=1 ) r=(r<<1)+(x&1);
    return r;
}
void change( vir *x, int len, int loglen ) {
    for ( int i=0; i<len; i++ )
        if ( rev(i,loglen)<i ) swap(x[rev(i,loglen)],x[i]);
}
void fft( vir *x, int len, int loglen ) {
    change(x,len,loglen);
    int i,j,s,t=1;
    for ( i=0; i<loglen; i++,t<<=1 ) {
        for ( s=0; s<len; s+=t+t ) {
            vir a,b,wo(cos(PI/t),sin(PI/t)),wn(1,0);
            for ( j=s; j<s+t; j++ ) {
                a=x[j]; b=x[j+t]*wn;
                x[j]=a+b; x[j+t]=a-b;
                wn=wn*wo;
            }
        }
    }
}
void dit_fft( vir *x, int len, int loglen ) {
    int i,j,s,t=len>>1;
    for ( i=0; i<loglen; i++,t>>=1 ) {
        for ( s=0; s<len; s+=t+t ) {
            vir a,b,wn(1,0),wo(cos(PI/t),-sin(PI/t));
            for ( j=s; j<s+t; j++ ) {
                a=x[j]+x[j+t]; b=(x[j]-x[j+t])*wn;
                x[j]=a; x[j+t]=b;
                wn=wn*wo;
            }
        }
    }
    change(x,len,loglen);
    for ( i=0; i<len; i++ ) x[i].re/=len;
}

char a[N],b[N];
int ans[2*N];

int main()
{
  int na,nb,len=1,loglen=0;
  while(scanf("%s%s",a,b)==2)
  {
    for(int i=2*N-1;i>=0;i--)
      x1[i]=x2[i]=0.0;
    for(na=0;a[na];na++);
    for(nb=0;b[nb];nb++);
    for(int i=na-1;i>=0;i--)
      x1[i]=(double)(a[na-i-1]-'0');
    for(int i=nb-1;i>=0;i--)
      x2[i]=(double)(b[nb-i-1]-'0');
    while(len<=2*max(na,nb)+5)
    {
      len*=2;
      loglen++;
    }
    fft(x1,len,loglen);
    fft(x2,len,loglen);
    for(int i=0;i<len;i++)
      x1[i]=x1[i]*x2[i];
    dit_fft(x1,len,loglen);
    for(int i=len-1;i>=0;i--)
      ans[i]=(int)round(x1[i].re+0.01);
    for(int i=0;i<len;i++)
    {
      if(ans[i]>=10)
      {
        ans[i+1]+=ans[i]/10;
        ans[i]%=10;
      }
    }
    bool zero=0;
    for(int i=len-1;i>=0;i--)
    {
      //printf("%d\n",ans[i]);
      if(zero)
        printf("%d",ans[i]);
      else if(ans[i]>0)
      {
        printf("%d",ans[i]);
        zero=1;
      }
    }
    if(!zero)
      printf("0");
    puts("");
  }
  return 0;
}
```

2.3 MillerRabin other

```cpp
/* Miller Rabin code from ioicamp */
#include <bits/stdc++.h>
#define PB push_back
#define MP make_pair
```

```
#define F first
#define S second
#define SZ(x) ((int)(x).size())
#define ALL(x) (x).begin(),(x).end()
#ifdef _DEBUG_
  #define debug(...) printf(__VA_ARGS__)
#else
  #define debug(...) 0
#endif
using namespace std;
typedef long long ll;
typedef pair<int,int> PII;
typedef vector<int> VI;

ll mul(ll a, ll b, ll n) {
  ll r = 0;
  a %= n, b %= n;
  while(b) {
    if(b&1) r = (a+r>=n ? a+r-n : a+r);
    a = (a+a>=n ? a+a-n : a+a);
    b >>= 1;
  }
  return r;
}

ll bigmod(ll a, ll d, ll n) {
  if(d==0)  return 1LL;
  if(d==1) return a % n;
  return mul(bigmod(mul(a, a, n), d/2, n), d%2?a:1, n);
}

const bool PRIME = 1, COMPOSITE = 0;
bool miller_rabin(ll n, ll a) {
  if(__gcd(a, n) == n)  return PRIME;
  if(__gcd(a, n) != 1)  return COMPOSITE;
  ll d = n-1, r = 0, res;
  while(d%2==0) { ++r; d/=2; }
  res = bigmod(a, d, n);
  if(res == 1 || res == n-1)  return PRIME;
  while(r--) {
    res = mul(res, res, n);
    if(res == n-1)  return PRIME;
  }
  return COMPOSITE;
}

bool isprime(ll n) {
  if(n==1)
    return COMPOSITE;
  ll as[7] = {2, 325, 9375, 28178, 450775, 9780504, 1795265022};
  for(int i=0; i<7; i++)
    if(miller_rabin(n, as[i]) == COMPOSITE)
      return COMPOSITE;
  return PRIME;
}
```

2.4 Guass

```
// be care of the magic number 7 & 8
void guass() {
  for(int i = 0; i < 7; i++) {
    Frac tmp = mat[i][i]; // Frac -> the type of data
    for(int j = 0; j < 8; j++)
      mat[i][j] = mat[i][j] / tmp;
    for(int j = 0; j < 7; j++) {
      if(i == j)
        continue;
      Frac ratio = mat[j][i]; // Frac -> the type of data
      for(int k = 0; k < 8; k++)
        mat[j][k] = mat[j][k] - ratio * mat[i][k];
    }
  }
}
```

3 flow

3.1 dinic

```
#include <bits/stdc++.h>
#define PB push_back
#define MP make_pair
#define F first
#define S second
#define SZ(x) ((int)(x).size())
using namespace std;
typedef long long ll;
typedef pair<int,int> PII;
typedef vector<int> VI;

/*******************************/
// dinic
const int MAXV=300;
const int MAXE=10000;
const int INF=(int)1e9+10;

struct E{
  int to,co;//capacity
  E(int t=0,int c=0):to(t),co(c){}
}eg[2*MAXE];

// source:0  sink:n-1
struct Flow{
  VI e[MAXV];
  int ei,v;
  void init(int n) {
    v=n;
    ei=0;
    for(int i=0;i<n;i++)
      e[i]=VI();
  }
  void add(int a,int b,int c) { //a to b , maxflow=c
    eg[ei]=E(b,c);
    e[a].PB(ei);
    ei++;
    eg[ei]=E(a,0);
    e[b].PB(ei);
    ei++;
  }

```

```
  int d[MAXV],qu[MAXV],ql,qr;
  bool BFS() {
    memset(d,-1,v*sizeof(int));
    ql=qr=0;
    qu[qr++]=0;
    d[0]=0;
    while(ql<qr && d[v-1]==-1) {
      int n=qu[ql++];
      VI &v=e[n];
      for(int i=v.size()-1;i>=0;i--) {
        int u=v[i];
        if(d[eg[u].to]==-1 && eg[u].co>0) {
          d[eg[u].to]=d[n]+1;
          qu[qr++]=eg[u].to;
        }
      }
    }
    return d[v-1]!=-1;
  }
  int ptr[MAXV];
  int go(int n,int p) {
    if(n==v-1)
      return p;
    VI &u=e[n];
    int temp;
    for(int i=ptr[n];i<SZ(u);i++)
    {
      if(d[n]+1!=d[eg[u[i]].to] || eg[u[i
          ]].co==0)
        continue;
      if((temp=go(eg[u[i]].to,min(p,eg[u[i
          ]].co)))==0)
        continue;
      eg[u[i]].co-=temp;
      eg[u[i]^1].co+=temp;
      ptr[n]=i;
      return temp;
    }
    ptr[n]=SZ(u);
    return 0;
  }
  int max_flow() {
    int ans=0,temp;
    while(BFS()) {
      for(int i=0;i<v;i++)
        ptr[i]=0;
      while((temp=go(0,INF))>0)
        ans+=temp;
    }
    return ans;
  }
}flow;

int main() {

  return 0;
}
```

4 string

4.1 KMP

```
void KMP_build(const char *S,int *F) {
  int p=F[0]=-1;
  for(int i=1;S[i];i++) {
    while(p!=-1 && S[p+1]!=S[i])
      p=F[p];
    if(S[p+1]==S[i])
      p++;
    F[i]=p;
  }
}

VI KMP_match(const char *S,const int *F,
    const char *T) {
  VI ans;
  int p=-1;
  for(int i=0;T[i];i++) {
    while(p!=-1 && S[p+1]!=T[i])
      p=F[p];
    if(S[p+1]==T[i])
      p++;
    if(!S[p+1]) {
      ans.PB(i-p);
      p=F[p];
    }
  }
  return ans;
}
```

4.2 Z-value

```
void Z_build(const char *S,int *Z) {
  Z[0]=0;
  int bst=0;
  for(int i=1;S[i];i++) {
    if(Z[bst]+bst<i) Z[i]=0;
    else Z[i]=min(Z[bst]+bst-i,Z[i-bst]);
    while(S[Z[i]]==S[i+Z[i]]) Z[i]++;
    if(Z[i]+i>Z[bst]+bst) bst=i;
  }
}
```

4.3 Z-value-palindrome

```
// AC code of NTUJ1871
#include <bits/stdc++.h>
#define pb push_back
#define F first
#define S second
#define SZ(x) ((int)(x).size())
#define MP make_pair
using namespace std;
typedef long long ll;
typedef pair<int,int> PII;
typedef vector<int> VI;

char in[100100];
char s[200100];
int z[200100];

int main()
{
    while(gets(in))
    {
```

```
        int len=1;
        for(int i=0;in[i];i++)
        {
            s[len++]='*';
            s[len++]=in[i];
        }
        s[len]=0;
        z[0]=0;
        z[1]=0;
        int bst=1;
        for(int i=1;i<len;i++)
        {
            z[i]=min(bst+z[bst]-i,z[bst+bst
                -i]);
            while(s[i+z[i]+1]==s[i-z[i]-1])
                z[i]++;
            if(z[i]+i>bst+z[bst])
                bst=i;
        }
        /*for(int i=1;i<len;i++)
            putchar(s[i]);
        puts("");
        for(int i=1;i<len;i++)
            printf("%d",z[i]);
        puts("");*/
        bool yes=0;
        for(int i=3;i<len;i+=2)
            if(z[(i+1)/2]==i/2 && z[(i+len)
                /2]==(len-i-1)/2)
                yes=1;
        if(yes)
            puts("www");
        else
            puts("vvvvvv");
    }
    return 0;
}
```

4.4 Suffix Array($O(NlogN)$)

```
const int SASIZE=100020;  // >= (max length
    of string + 20)
struct SA{
  char S[SASIZE]; // put target string into
      S[0:(len-1)]
  // you can change the type of S into int
      if required
  // if the string is in int, please avoid
      number < 0
  int R[SASIZE*2],SA[SASIZE];
  int tR[SASIZE*2],tSA[SASIZE];
  int cnt[SASIZE],len;      // set len
      before calling build()
  int H[SASIZE];

  void build_SA() {
    int maxR=0;
    for(int i=0;i<len;i++)
      R[i]=S[i];
    for(int i=0;i<=len;i++)
      R[len+i]=-1;
    memset(cnt,0,sizeof(cnt));
    for(int i=0;i<len;i++)
      maxR=max(maxR,R[i]);
    for(int i=0;i<len;i++)
      cnt[R[i]+1]++;
    for(int i=1;i<=maxR;i++)
      cnt[i]+=cnt[i-1];
    for(int i=0;i<len;i++)
      SA[cnt[R[i]]++]=i;
    for(int i=1;i<len;i*=2)
    {
      memset(cnt,0,sizeof(int)*(maxR+10));
      memcpy(tSA,SA,sizeof(int)*(len+10));
      memcpy(tR,R,sizeof(int)*(len+i+10));
      for(int j=0;j<len;j++)
        cnt[R[j]+1]++;
      for(int j=1;j<=maxR;j++)
        cnt[j]+=cnt[j-1];
      for(int j=len-i;j<len;j++)
        SA[cnt[R[j]]++]=j;
      for(int j=0;j<len;j++)
      {
        int k=tSA[j]-i;
        if(k<0)
          continue;
        SA[cnt[R[k]]++]=k;
      }
      int num=0;
      maxR=0;
      R[SA[0]]=num;
      for(int j=1;j<len;j++)
      {
        if(tR[SA[j-1]]<tR[SA[j]] || tR[SA[j
            -1]+i]<tR[SA[j]+i])
          num++;
        R[SA[j]]=num;
        maxR=max(maxR,R[SA[j]]);
      }
    }
  }
  void build_H() {
    memset(H,0,sizeof(int)*(len+10));
    for(int i=0;i<len;i++)
    {
      if(R[i]==0)
        continue;
      int &t=H[R[i]];
      if(i>0)
        t=max(0,H[R[i-1]]-1);
      while(S[i+t]==S[SA[R[i]-1]+t]) t++;
    }
  }
}sa;
```

4.5 Aho-Corasick

```
// AC code of UVa 10679
#include <cstdio>
#include <cstring>
#include <new>

struct Trie {
  int c;
  bool fi=0;
  Trie *fail,*ch[52];
```

```
  Trie():c(0){memset(ch,0,sizeof(ch));}
}trie[1000100];

char m[1010],f[100100];
Trie *str[1010],*na,*root;

inline int c_i(char a) {
  return (a>='A' && a<='Z') ? a-'A' : a-'a'
      +26;
}

void insert(char *s,int num) {
  Trie *at=root;
  while(*s) {
    if(!at->ch[c_i(*s)])
      at->ch[c_i(*s)]=new (na++) Trie();
    at=at->ch[c_i(*s)],s++;
  }
  str[num]=at;
}

Trie *q[1000100];
int ql,qr;

void init() {
  ql=qr=-1;
  q[++qr]=root;
  root->fail=NULL;
  while(ql<qr) {
    Trie *n=q[++ql],*f;
    for(int i=0;i<52;i++) {
      if(!n->ch[i])
        continue;
      f=n->fail;
      while(f && !f->ch[i])
        f=f->fail;
      n->ch[i]->fail=f?f->ch[i]:root;
      q[++qr]=n->ch[i];
    }
  }
}

void go(char *s) {
  Trie*p=root;
  while(*s) {
    while(p && !p->ch[c_i(*s)])
      p=p->fail;
    p=p?p->ch[c_i(*s)]:root;
    p->fi=1;
    s++;
  }
}

void AC() {
  for(int i=qr;i>0;i--)
    q[i]->fail->c+=q[i]->c;
}

int main() {
  int T,q;
  scanf("%d",&T);
  while(T--) {
    na=trie;
    root=new (na++) Trie();
    scanf("%s",f);
    scanf("%d",&q);
    for(int i=0;i<q;i++) {
      scanf("%s",m);
      insert(m,i);
    }
    init();
    go(f);
    for(int i=0;i<q;i++)
      puts(str[i]->fi?"y":"n");
  }
  return 0;
}
```

4.6 Aho-Corasick-2016ioicamp

```
// AC code of 2016ioicamp 54
#include <bits/stdc++.h>
#define PB push_back
#define MP make_pair
#define F first
#define S second
#define SZ(x) ((int)(x).size())
#define ALL(x) (x).begin(),(x).end()
#ifdef _DEBUG_
  #define debug(...) printf(__VA_ARGS__)
#else
  #define debug(...) (void)0
#endif
using namespace std;
typedef long long ll;
typedef pair<int,int> PII;
typedef vector<int> VI;

const int MAXNM=100010;
int pp[MAXNM];

const int sizz=100010;
int nx[sizz][26],spt;
int fl[sizz],efl[sizz],ed[sizz];
int len[sizz];
int newnode(int len_=0) {
  for(int i=0;i<26;i++)nx[spt][i]=0;
  ed[spt]=0;
  len[spt]=len_;
  return spt++;
}
int add(char *s,int p) {
  int l=1;
  for(int i=0;s[i];i++) {
    int a=s[i]-'a';
    if(nx[p][a]==0) nx[p][a]=newnode(l);
    p=nx[p][a];
    l++;
  }
  ed[p]=1;
  return p;
}
int q[sizz],qs,qe;
void make_fl(int root) {
  fl[root]=efl[root]=0;
  qs=qe=0;
  q[qe++]=root;
  for(;qs!=qe;) {
```

```
    int p=q[qs++];
    for(int i=0;i<26;i++) {
      int t=nx[p][i];
      if(t==0) continue;
      int tmp=fl[p];
      for(;tmp&&nx[tmp][i]==0;) tmp=fl[tmp
          ];
      fl[t]=tmp?nx[tmp][i]:root;
      efl[t]=ed[fl[t]]?fl[t]:efl[fl[t]];
      q[qe++]=t;
    }
  }
}
char s[MAXNM];
char a[MAXNM];

int dp[MAXNM][4];

void mmax(int &a,int b) {
  a=max(a,b);
}

void match(int root) {
  int p=root;
  for(int i=1;s[i];i++) {
    int a=s[i]-'a';
    for(;p&&nx[p][a]==0;p=fl[p]);
    p=p?nx[p][a]:root;
    for(int j=1;j<=3;j++)
      dp[i][j]=dp[i-1][j];
    for(int t=p;t;t=efl[t]) {
      if(!ed[t])
        continue;
      for(int j=1;j<=3;j++)
        mmax(dp[i][j],dp[i-len[t]][j-1]+(pp
            [i]-pp[i-len[t]]));
    }
  }
}

int main() {
  int T;
  scanf("%d",&T);
  while(T--) {
    int n,m;
    scanf("%d%d",&n,&m);
    scanf("%s",s+1);
    for(int i=1;i<=n;i++)
      scanf("%d",pp+i);
    for(int i=1;i<=n;i++)
      pp[i]+=pp[i-1];
    spt=1;
    int root=newnode();
    for(int i=0;i<m;i++) {
      scanf("%s",a);
      add(a,root);
    }
    make_fl(root);
    for(int i=1;i<=n;i++)
      dp[i][1]=dp[i][2]=dp[i][3]=0;
    match(root);
    printf("%d\n",dp[n][3]);
  }
  return 0;
}
```

5 graph

5.1 Bipartite matching($O(N^3)$)

```
// NTUJ1263
#include <bits/stdc++.h>
#define pb push_back
#define F first
#define S second
#define SZ(x) ((int)(x).size())
#define MP make_pair
using namespace std;
typedef long long ll;
typedef pair<int,int> PII;
typedef vector<int> VI;

bool is(ll x)
{
  ll l=1,r=2000000,m;
  while(l<=r)
  {
    m=(l+r)/2;
    if(m*m==x)
      return 1;
    if(m*m<x)
      l=m+1;
    else
      r=m-1;
  }
  return 0;
}

VI odd,even;
int in[300];
VI e[300];
int match[300];
bool vis[300];

bool DFS(int x)
{
  vis[x]=1;
  for(int u:e[x])
  {
    if(match[u]==-1 || (!vis[match[u]]&&DFS
        (match[u])))
    {
      match[u]=x;
      match[x]=u;
      return 1;
    }
  }
  return 0;
}

int main()
{
  int N;
  while(scanf("%d",&N)==1)
  {
    odd.clear();
    even.clear();
    for(int i=0;i<N;i++)
      e[i].clear();
    for(int i=0;i<N;i++)
```

```
    {
      scanf("%d",in+i);
      if(in[i]%2==0)
        even.pb(i);
      else
        odd.pb(i);
    }
    for(int i:even)
      for(int j:odd)
        if(is(1ll*in[i]*in[i]+1ll*in[j]*in[j]) && __gcd(in[i],in[j])==1)
          e[i].pb(j), e[j].pb(i);
    int ans=0;
    fill(match,match+N,-1);
    for(int i=0;i<N;i++)
      if(match[i]==-1)
      {
        fill(vis,vis+N,0);
        if(DFS(i))
          ans++;
      }
    printf("%d\n",ans);
  }
  return 0;
}
```

5.2 Max clique(bcw)

```
class MaxClique {
public:
    static const int MV = 210;

    int V;
    int el[MV][MV/30+1];
    int dp[MV];
    int ans;
    int s[MV][MV/30+1];
    vector<int> sol;

    void init(int v) {
        V = v; ans = 0;
        FZ(el); FZ(dp);
    }

    /* Zero Base */
    void addEdge(int u, int v) {
        if(u > v) swap(u, v);
        if(u == v) return;
        el[u][v/32] |= (1<<(v%32));
    }

    bool dfs(int v, int k) {
        int c = 0, d = 0;
        for(int i=0; i<(V+31)/32; i++) {
            s[k][i] = el[v][i];
            if(k != 1) s[k][i] &= s[k-1][i];
            c += __builtin_popcount(s[k][i]);
        }
        if(c == 0) {
            if(k > ans) {
                ans = k;
                sol.clear();
                sol.push_back(v);
                return 1;
            }
            return 0;
        }
        for(int i=0; i<(V+31)/32; i++) {
            for(int a = s[k][i]; a ; d++) {
                if(k + (c-d) <= ans) return 0;
                int lb = a&(-a), lg = 0;
                a ^= lb;
                while(lb!=1) {
                    lb = (unsigned int)(lb) >> 1;
                    lg ++;
                }
                int u = i*32 + lg;
                if(k + dp[u] <= ans) return 0;
                if(dfs(u, k+1)) {
                    sol.push_back(v);
                    return 1;
                }
            }
        }
        return 0;
    }

    int solve() {
        for(int i=V-1; i>=0; i--) {
            dfs(i, 1);
            dp[i] = ans;
        }
        return ans;
    }
};
```

5.3 EdgeBCC

```
const int MAXN=1010;
const int MAXM=5010;
VI e[MAXN];
int low[MAXN],lvl[MAXN],bel[MAXN];
bool vis[MAXN];
int cnt;
VI st;
void DFS(int x,int l,int p) {
  st.PB(x);
  vis[x]=1;
  low[x]=lvl[x]=l;
  bool top=0;
  for(int u:e[x]) {
    if(u==p && !top) {
      top=1;
      continue;
    }
    if(!vis[u]) {
      DFS(u,l+1,x);
    }
    low[x]=min(low[x],low[u]);
  }
  if(x==1 || low[x]==l) {
```

```
    while(st.back()!=x) {
      bel[st.back()]=cnt;
      st.pop_back();
    }
    bel[st.back()]=cnt;
    st.pop_back();
    cnt++;
  }
}
int main() {
  int T;
  scanf("%d",&T);
  while(T--) {
    int N,M,a,b;
    scanf("%d%d",&N,&M);
    fill(vis,vis+N+1,0);
    for(int i=1;i<=N;i++)
      e[i].clear();
    while(M--) {
      scanf("%d%d",&a,&b);
      e[a].PB(b);
      e[b].PB(a);
    }
    cnt=0;
    DFS(1,0,-1);
    /*****/
  }
  return 0;
}
```

5.4 VerticeBCC

```
const int MAXN=10000;
const int MAXE=100000;

VI e[MAXN+10];
vector<PII> BCC[MAXE];
int bccnt;
vector<PII> st;
bool vis[MAXN+10];
int low[MAXN+10],level[MAXN+10];

void DFS(int x,int p,int l) {
  vis[x]=1;
  level[x]=low[x]=l;
  for(int u:e[x]) {
    if(u==p)
      continue;
    if(vis[u]) {
      if(level[u]<l) {
        st.PB(MP(x,u));
        low[x]=min(low[x],level[u]);
      }
    }
    else {
      st.PB(MP(x,u));
      DFS(u,x,l+1);
      if(low[u]>=l) {
        PII t=st.back();
        st.pop_back();
        while(t!=MP(x,u)) {
          BCC[bccnt].PB(t);
          t=st.back();
          st.pop_back();
        }
        BCC[bccnt].PB(t);
        bccnt++;
      }
      low[x]=min(low[x],low[u]);
    }
  }
}

int main() {
  int T,N,M;
  scanf("%d",&T);
  while(T--) {
    scanf("%d%d",&N,&M);
    for(int i=0;i<N;i++)
      e[i].clear();
    int cnt=0;
    while(1) {
      int x,y;
      scanf("%d%d",&x,&y);
      if(x==-1 && y==-1)
        break;
      cnt++;
      e[x].PB(y);
      e[y].PB(x);
    }
    for(int i=0;i<N;i++) { // no multi-edge
      sort(ALL(e[i]));
      e[i].erase(unique(ALL(e[i])),e[i].end
          ());
    }
    fill(vis,vis+N,0);
    while(bccnt)
      BCC[--bccnt].clear();
    DFS(0,-1,0);
    /***/
  }
  return 0;
}
```

5.5 Them.

```
1. Max (vertex) independent set = Max
   clique on Complement graph
2. Min vertex cover = |V| - Max independent
    set
3. On bipartite: Min vertex cover = Max
   Matching(edge independent)
4. Any graph with no isolated vertices: Min
    edge cover + Max Matching = |V|
```

6 data structure

6.1 Treap

```
#include <cstdlib>
#include <cstdio>
#include <algorithm>

using namespace std;
```

```
6
7  typedef long long ll;
8
9  const int N = 100000 + 10;
10
11 struct Treap {
12   static Treap mem[N], *pmem;
13
14   int sz, pri;
15   ll val, sum, add;
16   Treap *l, *r;
17
18   Treap() {}
19   Treap(ll _val):
20     l(NULL), r(NULL), sz(1), pri(rand()),
         val(_val), sum(_val), add(0) {}
21 } Treap::mem[N], *Treap::pmem = Treap::mem;
22
23 Treap* make(ll val) {
24   return new (Treap::pmem++) Treap(val);
25 }
26
27 inline int sz(Treap *t) {
28   return t ? t->sz : 0;
29 }
30
31 inline ll sum(Treap *t) {
32   return t ? t->sum + t->add * sz(t) : 0;
33 }
34
35 inline void add(Treap *t, ll x) {
36   t->add += x;
37 }
38
39 void push(Treap *t) {
40   t->val += t->add;
41   if(t->l)  t->l->add += t->add;
42   if(t->r)  t->r->add += t->add;
43   t->add = 0;
44 }
45
46 void pull(Treap *t) {
47   t->sum = sum(t->l) + sum(t->r) + t->val;
48   t->sz = sz(t->l) + sz(t->r) + 1;
49 }
50
51 Treap* merge(Treap *a, Treap *b) {
52   if(!a || !b)  return a ? a : b;
53   else if(a->pri > b->pri) {
54     push(a);
55     a->r = merge(a->r, b);
56     pull(a);
57     return a;
58   }
59   else {
60     push(b);
61     b->l = merge(a, b->l);
62     pull(b);
63     return b;
64   }
65 }
66
67 void split(Treap* t, int k, Treap *&a,
     Treap *&b) {
68   if(!t)  a = b = NULL;
69   else if(sz(t->l) < k) {
70     a = t;
71     push(a);
72     split(t->r, k - sz(t->l) - 1, a->r, b);
73     pull(a);
74   }
75   else {
76     b = t;
77     push(b);
78     split(t->l, k, a, b->l);
79     pull(b);
80   }
81 }
82
83 int main() {
84   srand(105105);
85
86   int n, q;
87   scanf("%d%d", &n, &q);
88
89   Treap *t = NULL;
90   for(int i = 0; i < n; i++) {
91     ll tmp;
92     scanf("%lld", &tmp);
93     t = merge(t, make(tmp));
94   }
95
96   while(q--) {
97     char c;
98     int l, r;
99     scanf("\n%c %d %d", &c, &l, &r);
100
101    Treap *tl = NULL, *tr = NULL;
102    if(c == 'Q') {
103      split(t, l - 1, tl, t);
104      split(t, r - l + 1, t, tr);
105      printf("%lld\n", sum(t));
106      t = merge(tl, merge(t, tr));
107    }
108    else {
109      ll x;
110      scanf("%lld", &x);
111      split(t, l - 1, tl, t);
112      split(t, r - l + 1, t, tr);
113      add(t, x);
114      t = merge(tl, merge(t, tr));
115    }
116  }
117
118  return 0;
119 }
```

6.2 copy on write treap

```
1 #include <cstdlib>
2 #include <cstdio>
3 #include <algorithm>
4 #include <climits>
5 #include <cstring>
6
7 using namespace std;
8
9 const int N = 1000000 + 10;
```

```
10
11 struct Treap {
12     char val;
13     int sz, refs;
14     Treap *l, *r;
15
16     Treap() {}
17     Treap(char _val):
18         val(_val), sz(1), refs(0), l(NULL),
               r(NULL) {}
19 };
20
21 Treap* make(Treap* t) {
22     return new Treap(*t);
23 }
24
25 Treap* make(char _val) {
26     return new Treap(_val);
27 }
28
29 void print_ref(Treap* t) {
30     if(!t)  return ;
31     print_ref(t->l);
32     printf("%d ", t->refs);
33     print_ref(t->r);
34 }
35
36 void print(Treap* t) {
37     if(!t)  return ;
38     print(t->l);
39     putchar(t->val);
40     print(t->r);
41 }
42
43 void takeRef(Treap* t) {
44     if(t)   t->refs++;
45 }
46
47 void dropRef(Treap* t) {
48     if(t) {
49         char c = t->val;
50         t->refs--;
51         if(t->refs <= 0) {
52             dropRef(t->l);
53             dropRef(t->r);
54             delete t;
55         }
56     }
57 }
58
59 int sz(Treap* t) {
60     return t ? t->sz : 0;
61 }
62
63 int rnd(int m) {
64     static int x = 851025;
65     return (x = (x*0xdefaced+1) & INT_MAX)
           % m;
66 }
67
68 void pull(Treap* t) {
69     t->sz = sz(t->l) + sz(t->r) + 1;
70 }
71
72 Treap* merge(Treap* a, Treap* b) {
73     if(!a || !b) {
74         Treap* t = a ? make(a) : make(b);
75         t->refs = 0;
76         takeRef(t->l);
77         takeRef(t->r);
78         return t;
79     }
80
81     Treap* t;
82     if( rnd(a->sz+b->sz) < a->sz) {
83         t = make(a);
84         t->refs = 0;
85         t->r = merge(a->r, b);
86         takeRef(t->l);
87         takeRef(t->r);
88     }
89     else {
90         t = make(b);
91         t->refs = 0;
92         t->l = merge(a, b->l);
93         takeRef(t->l);
94         takeRef(t->r);
95     }
96
97     pull(t);
98     return t;
99 }
100
101 void split(Treap* t, int k, Treap* &a,
       Treap* &b) {
102     if(!t)  a = b = NULL;
103     else if(sz(t->l) < k) {
104         a = make(t);
105         a->refs = 0;
106         split(a->r, k-sz(t->l)-1, a->r, b);
107         takeRef(a->l);
108         takeRef(a->r);
109         pull(a);
110     }
111     else {
112         b = make(t);
113         b->refs = 0;
114         split(b->l, k, a, b->l);
115         takeRef(b->l);
116         takeRef(b->r);
117         pull(b);
118     }
119 }
120
121 void print_inorder(Treap* t) {
122     if(!t)  return ;
123     putchar(t->val);
124     print_inorder(t->l);
125     print_inorder(t->r);
126 }
127
128 char s[N];
129
130 int main() {
131     int m;
132     scanf("%d", &m);
133     scanf("%s", s);
134     int n = strlen(s);
135     int q;
136     scanf("%d", &q);
```

```
137
138     Treap* t = NULL;
139     for(int i = 0; i < n; i++) {
140         Treap *a = t, *b = make(s[i]);
141         t = merge(a, b);
142         dropRef(a);
143         dropRef(b);
144     }
145
146     while(q--) {
147         int l, r, x;
148         scanf("%d%d%d", &l, &r, &x);
149         r++;
150
151         Treap *a, *b, *c, *d;
152         a = b = c = d = NULL;
153         split(t, l, a, b);
154         dropRef(a);
155         split(b, r-l, c, d);
156         dropRef(b);
157         dropRef(d);
158         split(t, x, a, b);
159         dropRef(t);
160         Treap* t2 = merge(c, b);
161         dropRef(b);
162         dropRef(c);
163         t = merge(a, t2);
164         dropRef(a);
165         dropRef(t2);
166
167         if(t->sz > m) {
168             Treap* t2 = NULL;
169             split(t, m, t2, a);
170             dropRef(a);
171             dropRef(t);
172             t = t2;
173         }
174     }
175
176     print(t);
177     putchar('\n');
178
179     return 0;
180 }
```

6.3 copy on write segment tree

```
1  #include <cstdlib>
2  #include <cstdio>
3  #include <algorithm>
4  #include <vector>
5
6  using namespace std;
7
8  const int N = 50000 + 10;
9  const int Q = 10000 + 10;
10
11 struct Seg {
12   static Seg mem[N*80], *pmem;
13
14   int val;
15   Seg *tl, *tr;
16
17   Seg() :
18     tl(NULL), tr(NULL), val(0) {}
19
20   Seg* init(int l, int r) {
21     Seg* t = new (pmem++) Seg();
22     if(l != r) {
23       int m = (l+r)/2;
24       t->tl = init(l, m);
25       t->tr = init(m+1, r);
26     }
27     return t;
28   }
29
30   Seg* add(int k, int l, int r) {
31     Seg* _t = new (pmem++) Seg(*this);
32     if(l==r) {
33       _t->val++;
34       return _t;
35     }
36
37     int m = (l+r)/2;
38     if(k <= m)  _t->tl = tl->add(k, l, m);
39     else    _t->tr = tr->add(k, m+1, r);
40
41     _t->val = _t->tl->val + _t->tr->val;
42     return _t;
43   }
44 } Seg::mem[N*80], *Seg::pmem = mem;
45
46 int query(Seg* ta, Seg* tb, int k, int l,
       int r) {
47   if(l == r)  return l;
48
49   int m = (l+r)/2;
50
51   int a = ta->tl->val;
52   int b = tb->tl->val;
53   if(b-a >= k)  return query(ta->tl, tb->tl
         , k, l, m);
54   else      return query(ta->tr, tb->tr, k
         -(b-a), m+1, r);
55 };
56
57 struct Query {
58   int op, l, r, k, c, v;
59
60   bool operator<(const Query b) const {
61     return c < b.c;
62   }
63 } qs[Q];
64 int arr[N];
65 Seg *t[N];
66 vector<int> vec2;
67
68 int main() {
69   int T;
70   scanf("%d", &T);
71
72   while(T--) {
73     int n, q;
74     scanf("%d%d", &n, &q);
75
76     for(int i = 1; i <= n; i++) {
77       scanf("%d", arr+i);
78       vec2.push_back(arr[i]);
```

```
    }
    for(int i = 0; i < q; i++) {
      scanf("%d", &qs[i].op);
      if(qs[i].op == 1) scanf("%d%d%d", &qs[i].l, &qs[i].r, &qs[i].k);
      else  scanf("%d%d", &qs[i].c, &qs[i].v);

      if(qs[i].op == 2) vec2.push_back(qs[i].v);
    }
    sort(vec2.begin(), vec2.end());
    vec2.resize(unique(vec2.begin(), vec2.end())-vec2.begin());
    for(int i = 1; i <= n; i++) arr[i] = lower_bound(vec2.begin(), vec2.end(), arr[i]) - vec2.begin();
    int mn = 0, mx = vec2.size()-1;

    for(int i = 0; i <= n; i++) t[i] = NULL;
    t[0] = new (Seg::pmem++) Seg();
    t[0] = t[0]->init(mn, mx);
    int ptr = 0;
    for(int i = 1; i <= n; i++) {
      t[i] = t[i-1]->add(arr[i], mn, mx);
    }

    for(int i = 0; i < q; i++) {
      int op = qs[i].op;
      if(op == 1) {
        int l = qs[i].l, r = qs[i].r, k = qs[i].k;
        printf("%d\n", vec2[query(t[l-1], t[r], k, mn, mx)]);
      }
      if(op == 2) {
        continue;
      }
      if(op == 3) puts("7122");
    }

    vec2.clear();
    Seg::pmem = Seg::mem;
  }

  return 0;
}
```

6.4 Treap+(HOJ 92)

```
#include <cstdlib>
#include <cstdio>
#include <algorithm>
#include <cstring>

using namespace std;

const int INF = 103456789;

struct Treap {
    int pri, sz, val, chg, rev, sum, lsum, rsum, mx_sum;
    Treap *l, *r;

    Treap() {}
    Treap(int _val) :
        pri(rand()), sz(1), val(_val), chg(INF), rev(0), sum(_val), lsum(_val), rsum(_val), mx_sum(_val), l(NULL), r(NULL) {}
};

int sz(Treap* t) {return t ? t->sz : 0;}
int sum(Treap* t) {
    if(!t)  return 0;
    if(t->chg == INF)   return t->sum;
    else    return t->chg*t->sz;
}
int lsum(Treap* t) {
    if(!t)  return -INF;
    if(t->chg != INF)   return max(t->chg, (t->chg)*(t->sz));
    if(t->rev)  return t->rsum;
    return t->lsum;
}
int rsum(Treap* t) {
    if(!t)  return -INF;
    if(t->chg != INF)   return max(t->chg, (t->chg)*(t->sz));
    if(t->rev)  return t->lsum;
    return t->rsum;
}
int mx_sum(Treap* t) {
    if(!t)  return -INF;
    if(t->chg != INF)   return max(t->chg, (t->chg)*(t->sz));
    return t->mx_sum;
}

void push(Treap* t) {
    if(t->chg != INF) {
        t->val = t->chg;
        t->sum = (t->sz) * (t->chg);
        t->lsum = t->rsum = t->mx_sum = max(t->sum, t->val);
        if(t->l)    t->l->chg = t->chg;
        if(t->r)    t->r->chg = t->chg;
        t->chg = INF;
    }
    if(t->rev) {
        swap(t->l, t->r);
        if(t->l)    t->l->rev ^= 1;
        if(t->r)    t->r->rev ^= 1;
        t->rev = 0;
    }
}

void pull(Treap* t) {
    t->sz = sz(t->l)+sz(t->r)+1;
    t->sum = sum(t->l)+sum(t->r)+t->val;
    t->lsum = max(lsum(t->l), sum(t->l)+max(0, lsum(t->r))+t->val);
    t->rsum = max(rsum(t->r), sum(t->r)+max(0, rsum(t->l))+t->val);
    t->mx_sum = max(max(mx_sum(t->l), mx_sum(t->r)), max(0, rsum(t->l))+max(0, lsum(t->r))+t->val);
```

```
}

Treap* merge(Treap* a, Treap* b) {
    if(!a || !b)    return a ? a : b;
    if(a->pri > b->pri) {
        push(a);
        a->r = merge(a->r, b);
        pull(a);
        return a;
    }
    else {
        push(b);
        b->l = merge(a, b->l);
        pull(b);
        return b;
    }
}

void split(Treap* t, int k, Treap* &a,
    Treap* &b) {
    if(!t) {
        a = b = NULL;
        return ;
    }
    push(t);
    if(sz(t->l) < k) {
        a = t;
        push(a);
        split(t->r, k-sz(t->l)-1, a->r, b);
        pull(a);
    }
    else {
        b = t;
        push(b);
        split(t->l, k, a, b->l);
        pull(b);
    }
}

void del(Treap* t) {
    if(!t)  return;
    del(t->l);
    del(t->r);
    delete t;
}

int main() {
    srand(7122);

    int n, m;
    scanf("%d%d", &n, &m);

    Treap* t = NULL;
    for(int i = 0; i < n; i++) {
        int x;
        scanf("%d", &x);
        t = merge(t, new Treap(x));
    }

    while(m--) {
        char s[15];
        scanf("%s", s);

        Treap *tl = NULL, *tr = NULL, *t2 =
            NULL;

        if(!strcmp(s, "INSERT")) {
            int p, k;
            scanf("%d%d", &p, &k);
            for(int i = 0; i < k; i++) {
                int x;
                scanf("%d", &x);
                t2 = merge(t2, new Treap(x)
                    );
            }
            split(t, p, tl, tr);
            t = merge(tl, merge(t2, tr));
        }

        if(!strcmp(s, "DELETE")) {
            int p, k;
            scanf("%d%d", &p, &k);
            split(t, p-1, tl, t);
            split(t, k, t, tr);
            del(t);
            t = merge(tl, tr);
        }

        if(!strcmp(s, "MAKE-SAME")) {
            int p, k, l;
            scanf("%d%d%d", &p, &k, &l);
            split(t, p-1, tl, t);
            split(t, k, t, tr);
            if(t)   t->chg = l;
            t = merge(tl, merge(t, tr));
        }

        if(!strcmp(s, "REVERSE")) {
            int p, k;
            scanf("%d%d", &p, &k);
            split(t, p-1, tl, t);
            split(t, k, t, tr);
            if(t)   t->rev ^= 1;
            t = merge(tl, merge(t, tr));
        }

        if(!strcmp(s, "GET-SUM")) {
            int p, k;
            scanf("%d%d", &p, &k);
            split(t, p-1, tl, t);
            split(t, k, t, tr);
            printf("%d\n", sum(t));
            t = merge(tl, merge(t, tr));
        }

        if(!strcmp(s, "MAX-SUM")) {
            printf("%d\n", mx_sum(t));
        }
    }

    return 0;
}
```

6.5 Leftist Tree

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

```

```
struct Left {
  Left *l,*r;
  int v,h;
  Left(int v_) : v(v_), h(1), l(0), r(0) {}
};

int height(Left *p) { return p ? p -> h : 0 ; }

Left* combine(Left *a,Left *b) {
  if(!a || !b) return a ? a : b ;
  Left *p ;
  if( a->v > b->v) {
    p = a;
    p -> r = combine( p -> r , b );
  }
  else {
    p = b;
    p -> r = combine( p -> r , a );
  }
  if( height( p->l ) < height( p->r ) )
    swap( p->l , p->r );
  p->h = min( height( p->l ) , height( p->r ) ) + 1;
  return p;
}
Left *root;

void push(int v) {
  Left *p = new Left(v);
  root = combine( root , p );
}
int top() { return root? root->v : -1; }
void pop() {
  if(!root) return;
  Left *a = root->l , *b = root->r ;
  delete root;
  root = combine( a , b );
}
void clear(Left* &p) {
  if(!p)
    return;
  if(p->l) clear(p->l);
  if(p->r) clear(p->r);
  delete p;
  p = 0 ;
}

int main() {
  int T,n,x,o,size;
  bool bst,bqu,bpq;
  scanf("%d",&T);
  while(T--) {
    bst=bqu=bpq=1;
    stack<int> st;
    queue<int> qu;
    clear(root);
    size=0;
    scanf("%d",&n);
    while(n--) {
      scanf("%d%d",&o,&x);
      if(o==1)
        st.push(x),qu.push(x),push(x),size++;
      else if(o==2) {
        size--;
        if(size<0)
          bst=bqu=bpq=0;
        if(bst) {
          if(st.top()!=x)
            bst=0;
          st.pop();
        }
        if(bqu) {
          if(qu.front()!=x)
            bqu=0;
          qu.pop();
        }
        if(bpq) {
        //  printf("(%d)\n",top());
          if(top()!=x)
            bpq=0;
          pop();
        }
      }
    }
    int count=0;
    if(bst)
      count++;
    if(bqu)
      count++;
    if(bpq)
      count++;

    if(count>1)
      puts("not sure");
    else if(count==0)
      puts("impossible");
    else if(bst)
      puts("stack");
    else if(bqu)
      puts("queue");
    else if(bpq)
      puts("priority queue");
  }
  return 0;
}
```

6.6 Link Cut Tree

```
#include <bits/stdc++.h>
#define PB push_back
#define MP make_pair
#define F first
#define S second
#define SZ(x) ((int)(x).size())
#define ALL(x) (x).begin(),(x).end()
#ifdef _DEBUG_
  #define debug(...) printf(__VA_ARGS__)
#else
  #define debug(...) (void)0
#endif
using namespace std;
typedef long long ll;
typedef pair<int,int> PII;
typedef vector<int> VI;

const int MAXN = 100000 + 10;
```

```
19
20 struct SplayTree {
21   int val, mx, ch[2], pa;
22   bool rev;
23   void init() {
24     val = mx = -1;
25     rev = false;
26     pa = ch[0] = ch[1] = 0;
27   }
28 } node[MAXN*2];
29
30 inline bool isroot(int x) {
31   return node[node[x].pa].ch[0]!=x && node[
       node[x].pa].ch[1]!=x;
32 }
33
34 inline void pull(int x) {
35   node[x].mx = max(node[x].val, max(node[
       node[x].ch[0]].mx, node[node[x].ch
       [1]].mx));
36 }
37
38 inline void push(int x) {
39   if(node[x].rev) {
40     node[node[x].ch[0]].rev ^= 1;
41     node[node[x].ch[1]].rev ^= 1;
42     swap(node[x].ch[0], node[x].ch[1]);
43     node[x].rev ^= 1;
44   }
45 }
46
47 void push_all(int x) {
48   if(!isroot(x))  push_all(node[x].pa);
49   push(x);
50 }
51
52 inline void rotate(int x) {
53   int y = node[x].pa, z = node[y].pa, d =
       node[y].ch[1]==x;
54   node[x].pa = z;
55   if(!isroot(y))  node[z].ch[node[z].ch
       [1]==y] = x;
56   node[y].ch[d] = node[x].ch[d^1];
57   node[node[x].ch[d^1]].pa = y;
58   node[x].ch[!d] = y;
59   node[y].pa = x;
60   pull(y);
61   pull(x);
62 }
63
64 void splay(int x) {
65   push_all(x);
66   while(!isroot(x)) {
67     int y = node[x].pa;
68     if(!isroot(y)) {
69       int z = node[y].pa;
70       if((node[z].ch[1]==y) ^ (node[y].ch
           [1]==x)) rotate(y);
71       else  rotate(x);
72     }
73     rotate(x);
74   }
75 }
76
77 inline int access(int x) {
78   int last = 0;
79   while(x) {
80     splay(x);
81     node[x].ch[1] = last;
82     pull(x);
83     last = x;
84     x = node[x].pa;
85   }
86   return last;
87 }
88
89 inline void make_root(int x) {
90   node[access(x)].rev ^= 1;
91   splay(x);
92 }
93
94 inline void link(int x, int y) {
95   make_root(x);
96   node[x].pa = y;
97 }
98
99 inline void cut(int x, int y) {
100   make_root(x);
101   access(y);
102   splay(y);
103   node[y].ch[0] = 0;
104   node[x].pa = 0;
105 }
106
107 inline void cut_parent(int x) {
108   x = access(x);
109   splay(x);
110   node[node[x].ch[0]].pa = 0;
111   node[x].ch[0] = 0;
112   pull(x);
113 }
114
115 inline int find_root(int x) {
116   x = access(x);
117   while(node[x].ch[0])  x = node[x].ch[0];
118   splay(x);
119   return x;
120 }
121
122 int find_mx(int x) {
123   if(node[x].val == node[x].mx) return x;
124   return node[node[x].ch[0]].mx==node[x].mx
        ? find_mx(node[x].ch[0]) : find_mx(
       node[x].ch[1]);
125 }
126
127 inline void change(int x,int b){
128     splay(x);
129     node[x].data=b;
130     up(x);
131 }
132 inline int query_lca(int u,int v){
133 /* □ □ □ □ □ □ □ □ □ □ □ sum □ □ □ □ □ □ □
     data □ □ □ □ □ □ */
134   access(u);
135   int lca=access(v);
136   splay(u);
137   if(u==lca){
138     return node[lca].data+node[node[lca].ch
         [1]].sum;
```

```
  }else{
    return node[lca].data+node[node[lca].ch
        [1]].sum+node[u].sum;
  }
}
```

6.7 Heavy Light Decomposition

```
#include <bits/stdc++.h>
#define PB push_back
#define MP make_pair
#define F first
#define S second
#define SZ(x) ((int)(x).size())
#define ALL(x) (x).begin(),(x).end()
#ifdef _DEBUG_
  #define debug(...) printf(__VA_ARGS__)
#else
  #define debug(...) (void)0
#endif
using namespace std;
typedef long long ll;
typedef pair<int,int> PII;
typedef vector<int> VI;

const int MAXN = 10000 + 10;

vector<PII> e[MAXN];
int val[MAXN];
int sz[MAXN], max_son[MAXN], p[MAXN], dep[
    MAXN];
int link[MAXN], link_top[MAXN], cnt;

void find_max_son(int u) {
  sz[u] = 1;
  max_son[u] = -1;
  for(int i=0; i<SZ(e[u]); i++) {
    PII tmp = e[u][i];
    int v = tmp.F;
    if(v == p[u]) continue;

    p[v] = u;
    dep[v] = dep[u]+1;
    val[v] = tmp.S;
    find_max_son(v);
    if(max_son[u]<0 || sz[v]>sz[ max_son[u]
        ])  max_son[u] = v;
    sz[u] += sz[v];
  }
}

void build_link(int u, int top) {
  link[u] = ++cnt;
  link_top[u] = top;
  if(max_son[u] > 0)  build_link(max_son[u
      ], top);
  for(int i=0; i<SZ(e[u]); i++) {
    PII tmp = e[u][i];
    int v = tmp.F;
    if(v==p[u] || v==max_son[u])  continue;

    build_link(v, v);
  }
}

int query(int a, int b) {
  int res = -1;
  int ta = link_top[a], tb = link_top[b];
  while(ta != tb) {
    if(dep[ta] < dep[tb]) {
      swap(a, b);
      swap(ta, tb);
    }

    res = max(res, seg->qry(link[ta], link[
        a], 1, cnt));
    ta = link_top[a=p[ta]];
  }

  if(a != b) {
    if(dep[a] > dep[b]) swap(a, b);
    a = max_son[a];
    res = max(res, seg->qry(link[a], link[b
        ], 1, cnt));
  }

  return res;
}
```

6.8 Disjoint Sets + offline skill

```
#include <bits/stdc++.h>
#define PB push_back
#define MP make_pair
#define F first
#define S second
#define SZ(x) ((int)(x).size())
#define ALL(x) (x).begin(),(x).end()
#ifdef _DEBUG_
  #define debug(...) printf(__VA_ARGS__)
#else
  #define debug(...) (void)0
#endif
using namespace std;
typedef long long ll;
typedef pair<int,int> PII;
typedef vector<int> VI;

const int MAXN = 300000 + 10;

bool q[MAXN];

struct DisJointSet {
  int p[MAXN], sz[MAXN], gps;
  vector<pair<int*, int> > h;
  VI sf;

  void init(int n) {
    for(int i=1; i<=n; i++) {
      p[i] = i;
      sz[i] = 1;
    }
    gps = n;
  }

  void assign(int *k, int v) {
```

```
    h.PB(MP(k, *k));
    *k = v;
  }

  void save() {
    sf.PB(SZ(h));
  }

  void load() {
    int last = sf.back(); sf.pop_back();
    while(SZ(h) != last) {
      auto x = h.back();  h.pop_back();
      *x.F = x.S;
    }
  }

  int find(int x) {
    return x==p[x] ? x : find(p[x]);
  }

  void uni(int x, int y) {
    x = find(x), y = find(y);
    if(x == y)  return ;
    if(sz[x] < sz[y]) swap(x, y);
    assign(&sz[x], sz[x]+sz[y]);
    assign(&p[y], x);
    assign(&gps, gps-1);
  }
} djs;

struct Seg {
  vector<PII> es;
  Seg *tl, *tr;

  Seg() {}
  Seg(int l, int r) {
    if(l == r)  tl = tr = NULL;
    else {
      int m = (l+r) / 2;
      tl = new Seg(l, m);
      tr = new Seg(m+1, r);
    }
  }

  void add(int a, int b, PII e, int l, int
      r) {
    if(a <= l && r <= b)  es.PB(e);
    else if(b < l || r < a) return ;
    else {
      int m = (l+r) / 2;
      tl->add(a, b, e, l, m);
      tr->add(a, b, e, m+1, r);
    }
  }

  void solve(int l, int r) {
    djs.save();
    for(auto p : es)  djs.uni(p.F, p.S);

    if(l == r) {
      if(q[l])  printf("%d\n", djs.gps);
    }
    else {
      int m = (l+r) / 2;
      tl->solve(l, m);
      tr->solve(m+1, r);
    }

    djs.load();
  }
};

map<PII, int> prv;

int main() {
  freopen("connect.in", "r", stdin);
  freopen("connect.out", "w", stdout);

  int n, k;
  scanf("%d%d\n", &n, &k);
  if(!k)  return 0;

  Seg *seg = new Seg(1, k);
  djs.init(n);
  for(int i=1; i<=k; i++) {
    char op = getchar();
    if(op == '?') {
      q[i] = true;
      op = getchar();
    }
    else {
      int u, v;
      scanf("%d%d\n", &u, &v);
      if(u > v) swap(u, v);
      PII eg = MP(u, v);
      int p = prv[eg];
      if(p) {
        seg->add(p, i, eg, 1, k);
        prv[eg] = 0;
      }
      else  prv[eg] = i;
    }
  }
  for(auto p : prv) {
    if(p.S) {
      seg->add(p.S, k, p.F, 1, k);
    }
  }

  seg->solve(1, k);

  return 0;
}
```

7 geometry

7.1 Basic

```
// correct code of NPSC2013 senior-final pF

#include <bits/stdc++.h>
#define pb push_back
#define F first
#define S second
#define SZ(x) ((int)(x).size())
#define MP make_pair
using namespace std;
```

```
typedef long long ll;
typedef pair<int,int> PII;
typedef vector<int> VI;

typedef double dou;
struct PT{
  dou x,y;
  PT(dou x_=0.0,dou y_=0.0): x(x_),y(y_) {}
  PT operator + (const PT &b) const {
     return PT(x+b.x,y+b.y); }
  PT operator - (const PT &b) const {
     return PT(x-b.x,y-b.y); }
  PT operator * (const dou &t) const {
     return PT(x*t,y*t); }
  dou operator * (const PT &b) const {
     return x*b.x+y*b.y; }
  dou operator % (const PT &b) const {
     return x*b.y-b.x*y; }
  dou len2() const { return x*x+y*y; }
  dou len() const { return sqrt(len2()); }
};

const dou INF=1e12;
const dou eps=1e-8;
PT inter(const PT &P1,const PT &T1,const PT
    &P2,const PT &T2) // intersection
{
  if(fabs(T1%T2)<eps)
    return PT(INF,INF);
  dou u=((P2-P1)%T2)/(T1%T2);
  return P1+T1*u;
}

PT conv[500],cat,to;

int main()
{
  int T,N,M;
  scanf("%d",&T);
  while(T--)
  {
    scanf("%d%d",&N,&M);
    for(int i=0;i<N;i++)
      scanf("%lf%lf",&conv[i].x,&conv[i].y)
         ;
    conv[N]=conv[0];
    dou ans=0.0;
    while(M--)
    {
      scanf("%lf%lf%lf%lf",&cat.x,&cat.y,&
         to.x,&to.y);
      for(int i=0;i<N;i++)
        if(fabs((conv[i]-conv[i+1])%to)>eps
           )
        {
        //  printf("M:%d i=%d\n",M,i);
          PT at=inter(conv[i],conv[i]-conv[
             i+1],cat,to);
          if((conv[i]-at)*(conv[i+1]-at)<
             eps && (at-cat)*to>-eps)
            ans=max(ans,(cat-at).len());
        }
    }
    printf("%.4f\n",ans);
  }
  return 0;
}
```

7.2 Smallist circle problem

```
#include <cstdlib>
#include <cstdio>
#include <algorithm>
#include <cmath>

using namespace std;

const int N = 1000000 + 10;

struct PT {
  double x, y;

  PT() {}
  PT(double x, double y):
    x(x), y(y) {}
  PT operator+(const PT &b) const {
    return (PT) {x+b.x, y+b.y};
  }
  PT operator-(const PT &b) const {
    return (PT) {x-b.x, y-b.y};
  }
  PT operator*(const double b) const {
    return (PT) {x*b, y*b};
  }
  PT operator/(const double b) const {
    return (PT) {x/b, y/b};
  }
  double operator%(const PT &b) const {
    return x*b.y - y*b.x;
  }

  double len() const {
    return sqrt(x*x + y*y);
  }
  PT T() const {
    return (PT) {-y, x};
  }
} p[N];

void update(PT a, PT b, PT c, PT &o, double
    &r) {
  if(c.x < 0.0) o = (a+b) / 2.0;
  else {
    PT p1 = (a+b)/2.0, p2 = p1 + (b-a).T();
    PT p3 = (a+c)/2.0, p4 = p3 + (c-a).T();
    double a123 = (p2-p1)%(p3-p1), a124 = (
       p2-p1)%(p4-p1);
    if(a123 * a124 > 0.0) a123 = -a123;
    else  a123 = abs(a123), a124 = abs(a124
       );
    o = (p4*a123 + p3*a124) / (a123 + a124)
       ;
  }
  r = (a-o).len();
}

int main() {
  srand(7122);
```

```
55
56    int m, n;
57    while(scanf("%d%d", &m, &n)) {
58      if(!n && !m)  return 0;
59
60      for(int i = 0; i < n; i++)  scanf("%lf%
          lf", &p[i].x, &p[i].y);
61
62      for(int i = 0; i < n; i++)
63        swap(p[i], p[rand() % (i+1)]);
64
65      PT a = p[0], b = p[1], c(-1.0, -1.0), o
          = (a+b) / 2.0;
66      double r = (a-o).len();
67      for(int i = 2; i < n; i++) {
68        if((p[i]-o).len() <= r) continue;
69
70        a = p[i];
71        b = p[0];
72        c = (PT) {-1.0, -1.0};
73        update(a, b, c, o, r);
74        for(int j = 1; j < i; j++) {
75          if((p[j]-o).len() <= r) continue;
76
77          b = p[j];
78          c = (PT) {-1.0, -1.0};
79          update(a, b, c, o, r);
80
81          for(int k = 0; k < j; k++) {
82            if((p[k]-o).len() <= r) continue;
83
84            c = p[k];
85            update(a, b, c, o, r);
86          }
87        }
88      }
89
90      printf("%.3f\n", r);
91    }
92  }
```

8 Others

8.1 Random

```
1  const int seed=1;
2
3  mt19937 rng(seed);
4  int randint(int lb,int ub) { // [lb, ub]
5    return uniform_int_distribution<int>(lb,
       ub)(rng);
6  }
```

8.2 Fraction

```
1  struct Frac {
2    ll a,b; //  a/b
3    void relax() {
4      ll g=__gcd(a,b);
5      if(g!=0 && g!=1)
6        a/=g, b/=g;
7      if(b<0)
8        a*=-1, b*=-1;
9    }
10   Frac(ll a_=0,ll b_=1): a(a_), b(b_) {
11     relax();
12   }
13   Frac operator + (Frac x) {
14     relax();
15     x.relax();
16     ll g=__gcd(b,x.b);
17     ll lcm=b/g*x.b;
18     return Frac(a*(lcm/b)+x.a*(lcm/x.b),lcm
         );
19   }
20   Frac operator - (Frac x) {
21     relax();
22     x.relax();
23     Frac t=x;
24     t.a*=-1;
25     return *this+t;
26   }
27   Frac operator * (Frac x) {
28     relax();
29     x.relax();
30     return Frac(a*x.a,b*x.b);
31   }
32   Frac operator / (Frac x) {
33     relax();
34     x.relax();
35     Frac t=Frac(x.b,x.a);
36     return (*this)*t;
37   }
38 };
```