学长的打工日记

数据结构

树状数组

```cpp
int lowbit(int x ){
    return x&(-x) ;
}
template <typename T>
class BIT{
    public :
        int size ;
        vector<T> t ;
    BIT( int _size) :size(_size), t(_size + 1, 0){   }
    BIT() : size(0) {}
    ~BIT( ) { t.clear() ;}
    BIT( const BIT &_t) {   size = _t.size ;  t = _t.t  ;}
    void add(int x,T d){
        for(int i =x ; i<=size ; i+= lowbit( i)) t[i] +=d ;
    }
    T sum( int x ) {
        T res = 0 ;
        for(int i = x; i ; i -=lowbit(i)) res +=t[i] ;
        return res;
    }
};
```

并查集

```cpp
template<typename T>
class DSU{
    public :
        int size ;
        vector<T> fa ;
        vector<T> siz ;
    DSU( ) :size(0) {}
    DSU( int _size) : size(_size) , siz(_size+1 ,1) ,fa(_size+1)    {
        for(int i = 0 ; i <=size; ++ i) fa[i] = i  ;
    }
    DSU( const DSU &_t) {   size = _t.size ;  fa = _t.fa  ; siz = _t.siz  ;
}
    ~DSU( ) { fa.clear() , siz.clear()  ; }
    int find( int x ) {
        if( fa[x] == x ) return x ;
        return fa[x] = find( fa[x]);
    }
    void merge( int a ,int b) {
        int pa = find( a) ,  pb=  find(b ) ;
        if( pa ==pb) return  ;
        siz[pa] += siz[pb] ;
        fa[pb] = pa ;
```

```
    }

    bool same(int a ,int b ){
        return find(a) == find(b ) ;
    }

};
```

带权并查集

```
template<typename T>
class DSU{
    public :
        int size ;
        vector<T> fa ;
        vector<T> siz ;
        vector<T> val ;
    DSU( ) :size(0) {}
    DSU( int _size) : size(_size) , siz(_size+1 ,1) ,fa(_size+1)
,val(_size+1 , 0)   {
        for(int i = 0 ; i <=size; ++ i) fa[i] = i  ;
    }
    DSU( const DSU &_t) {   size = _t.size ;  fa = _t.fa  ; siz = _t.siz  ;
val = _t.val ; }
    ~DSU( ) { fa.clear() , siz.clear() , val.clear() ; }
    int find( int x ) {
        if( fa[x] == x ) return x ;
        int  pa = find( fa[x]) ;
        val[x] += val[fa[x]] ;
        return fa[x] = pa;
    }
    void merge( int a ,int b , T w) {
        int pa = find( a) ,  pb=  find(b ) ;
        if( pa ==pb) return  ;
        siz[pa] += siz[pb] ;
        val[pb]  = val[a] + w - val[b] ;
        fa[pb] = pa ;
    }

    bool same(int a ,int b ){
        return find(a) == find(b ) ;
    }

};
```

倍增LCA

```
class LCA{
    public :
        int size ;
        int mx;
        vector<int> depth ;
        vector< vector<int> > fa ;
        vector< vector<int> > adj;
```

```
    LCA() : size(0) { }
    LCA(int _size): size(_size) , mx( log2(_size+1)+1 ) ,adj( _size+1) {
        fa.assign( _size +1, vector<int>( mx , 0 )) ;
        depth.assign(_size+1 , inf ) ;
    }
    LCA(const LCA &_t):size(_t.size) , depth( _t.depth) , mx(_t.mx ) , fa(_t.fa) ,adj(_t.adj) { }
    void addedges(int a, int b ) {
        adj[a].push_back( b) ;
    }
    void run( int u ) { bfs( u );}

    void bfs(int u  ) {
        depth[0] =   0 ;
        depth[u] = 1;  queue<int> q;
        q.push( u ) ;
        while(!q.empty()) {
            int t =q.front() ;  q.pop() ;
            for(auto u  : adj[t]) {
                if( depth[u] > depth[t ] + 1 ) {
                    depth[u] =  depth[t] + 1 ;
                    fa[u][0] = t;
                    for(int i = 1 ; i < mx; ++ i) {
                        fa[u][i] = fa[ fa[u][i-1]][i-1] ;
                    }
                    q.push( u ) ;
                }
            }
        }
    }

    int lca( int a,int b ) {
        if( a ==b) return a;
        if( depth[a] < depth[b]) swap(a,b ) ;
        for(int i =mx -1;i>=0; --i) if( depth[ fa[a][i]] >= depth[b] ) a = fa[a][i] ;
        if( a ==b ) return a;
        for(int i = mx -1; i>=0 ; --i) if( fa[a][i] != fa[b][i]) a=fa[a][i] , b = fa[b][i]  ;
        return fa[a][0] ;
    }

};
```

二维树状数组

```
int lowbit(int x ){
    return x&(-x) ;
}
template <typename T>
class BIT{
    public :
        int row , col ;
        vector< vector<T> > t ;
```

```
    BIT( int _row , int _col) :row(_row), col( _col) ,  t(_row + 1, vector<T>( col + 1 ,  0 )){   }
    BIT() : row(0), col( 0) {}
    ~BIT( ) { t.clear() ;}
    BIT( const BIT &_t) {   row = _t.row ; col = _t.col ;  t = _t.t  ;}
    void add(int x, int y , T d){
        if( !x || !y ) return ;
        for(int i =x ; i<= row ; i+= lowbit( i)) {
            for(int j = y ; j <=col; j += lowbit( j ))
                t[i][j] += d ;
        }
    }
    T sum( int x ,int y  ) {
        T res = 0 ;
        for(int i = x; i ; i -=lowbit(i)){
            for(int j  = y ; j ; j -=lowbit( j) )
                res += t[i][j] ;
        }
        return res;
    }
};
```

线段树

```
LL sum[N<<2] , tag[N<<2] , mx[N<<2] , mn[N<<2] ;
LL a[N] ;
void pushup(int x ) {
    sum[x] = sum[x<<1]  + sum[ x << 1 | 1 ] ;
    mx[x] =  max( mx[x<<1] , mx[x<<1 | 1 ]) ;
    mn[x] =  min( mn[x<<1] , mn[x << 1 | 1 ]) ;
}

void pushdown( int x ,int l ,int r  ) {
    if( tag[x]) {
        int mid = l + r >> 1 ;
        sum[x<<1] += 1ll *( mid - l + 1 )* tag[x] ;
        sum[x<<1|1] += 1ll *( r - mid ) * tag[x] ;
        mx[x<<1] += tag[x] ; mx[x<<1|1] += tag[x] ;
        mn[x<<1] += tag[x] ; mn[x<<1|1] += tag[x] ;
        tag[x<<1] += tag[x] ; tag[x<<1|1] += tag[x] ;
        tag[x] = 0 ;
    }
}


void add(int x, int l ,int r ,int L , int R , int d ) {
    if( L <= l && r <= R ) {
        sum[x] += 1ll * ( r - l + 1 ) * d ;
        mx[x] += d ; mn[x] += d ;tag[x] += d ;
        return ;
    }
    pushdown( x , l , r ) ;
    int mid = l + r >> 1 ;
    if( L <= mid ) add( x << 1 , l , mid , L , R , d ) ;
    if( R > mid ) add( x << 1 | 1 , mid + 1 ,r  , L , R , d ) ;
```

```
    pushup( x ) ;
}

LL query( int x, int l ,int r, int L , int R ) {
    if( L <= l && r <= R ) {
        return sum[x];
    }
    pushdown( x , l , r  ) ;
    int mid = l + r >> 1 ;
    LL res = 0;
    if( L <= mid ) res += query( x << 1 , l , mid , L , R  ) ;
    if( R > mid )  res += query( x << 1 | 1 , mid + 1 ,r  , L , R  ) ;
    return res ;
}
int querymx(int x, int l ,int r , int L ,int R , LL s) {
    if( l == r ) return mx[x] >= s ? l : inf ;
    int mid = l + r >> 1 ;
    pushdown( x , l , r ) ;
    if(  L <= l && r <= R ) {
        if( mx[x] < s ) return inf ;
        if( mx[x << 1 ] >= s  ) return querymx( x << 1 , l , mid , L , R , s ) ;
        return querymx( x << 1 | 1 , mid + 1 , r,  L , R , s ) ;
    }
    int res = inf ;
    if( L <= mid ) res = min( res ,querymx( x << 1 , l , mid , L , R , s )) ;
    if( R > mid ) res = min(res ,querymx( x << 1 | 1 , mid + 1 , r,  L , R , s )) ;
    return res ;
}

int querymn(int x, int l ,int r , int L ,int R , LL s) {
    if( l == r ) return mn[x] <= s ? l : inf ;
    int mid = l + r >> 1 ;
    pushdown( x , l , r ) ;
    if(  L <= l && r <= R ) {
        if( mn[x] > s ) return inf ;
        if( mn[x << 1 ] <= s  ) return querymn( x << 1 , l , mid , L , R , s ) ;
        return querymn( x << 1 | 1 , mid + 1 , r , L , R  , s ) ;
    }
    int res = inf ;
    if( L <= mid ) res = min(querymn( x << 1 , l , mid , L , R , s ) , res ) ;
    if( R > mid ) res = min(querymn( x << 1 | 1 , mid + 1 , r,  L , R , s ) ,res ) ;
    return res ;
}


void build( int x, int l , int r ) {
    if( l == r ) {
        sum[x] = mx[x] = mn[x] = a[l] ; return ;
    }
    int mid = l + r >> 1 ;
```

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

splay树

```
int a[N] ;
int stk[N] ;
struct Node {
    int s[2] , p , v;
    int siz ,rev ;
    LL sum ,val ,tag;
    void init(int _v , int _p) {
        v= _v , p = _p ; rev= 0 ;
        siz = 1 ; sum = val =  0 ;
    }
}tr[N];

int root , idx ;


void pushrev(int x ){
    swap( tr[x].s[0]  ,tr[x].s[1]) ;
    tr[x].rev ^=1 ;
}

void pushup(int x ) {
    auto &fa=  tr[x] , &left = tr[ tr[x].s[0]]  ,&right = tr[ tr[x].s[1]] ;
    fa.sum = left.sum + right.sum + fa.val ;
    fa.siz = left.siz + right.siz + 1 ;
}

void pushdown(int x ) {
    auto &fa=  tr[x] , &left = tr[ tr[x].s[0]]  ,&right = tr[ tr[x].s[1]] ;
    if( fa.tag) {
        left.sum += left.siz*fa.tag , right.sum += right.siz * fa.tag ;
        left.val += fa.tag ,right.val += fa.tag ;
        left.tag += fa.tag , right.tag += fa.tag;
        fa.tag = 0 ;
    }
    if( fa.rev) {
        pushrev( tr[x].s[0] ) ; pushrev( tr[x].s[1]) ;
        fa.rev =0 ;
    }
}

void rotate(int x ) {
    int y = tr[x].p ,z=tr[y].p ;
    int k=  tr[y].s[1] == x ;
    tr[z].s[  tr[z].s[1] == y] = x ; tr[x].p = z;
    tr[y].s[k] =   tr[x].s[k^1]  ; tr[  tr[x].s[k^1]].p = y ;
    tr[x].s[k^1] = y ; tr[y].p =x ;
    pushup( y ) ; pushup(x ) ;
}

```

```
50
51  void splay(int x ,int k  ) {
52      int top =0 , u = x;
53      stk[++top] =  u ;
54      while(  tr[u].p) stk[++top ] = u =  tr[u].p ;
55      while( top) pushdown( stk[top--] ) ;
56      while( tr[x].p !=k) {
57          int y  =tr[x].p , z= tr[y].p ;
58          if( z != k ) {
59              if( ( tr[y].s[1] == x) ^( tr[z].s[1] == y)) rotate(x) ;
60              else rotate( y ) ;
61          }
62          rotate( x) ;
63      }
64      if(!k  ) root = x ;
65  }
66
67  void insert(int v) {
68      int u = root ,p = 0 ;
69      while( u ) p = u , u = tr[u].s[ v > tr[u].v] ;
70      u = ++idx;
71      if( p ) tr[p].s[ v > tr[p].v ] = u ;
72      tr[u].init( v , p )  ;  tr[u].val = a[v] ;
73      splay( u , 0 ) ;
74  }
75
76  int get_k(int x ) {
77      int u = root ,res =0 ;
78      while( u ) {
79          pushdown( u) ;
80          if( tr[ tr[u].s[0]].siz + res == x   ) {
81              splay( u ,0) ;
82              return u ;
83          }
84          else  if( tr[ tr[u].s[0]].siz + res > x   ) u = tr[u].s[0] ;
85          else res += tr[ tr[u].s[0]].siz + 1 , u = tr[u].s[1];
86      }
87      return -1 ;
88  }
89
90
91  void update(int l ,int r ,LL d ) {
92      int pre= get_k( l-1 ) , suf = get_k( r +1 );
93      splay( pre,0 ) ; splay( suf , pre) ;
94      tr[tr[suf].s[0] ].tag += d ;
95      tr[tr[suf].s[0]].val += d ;
96      tr[tr[suf].s[0] ].sum += d*tr[tr[suf].s[0] ].siz ;
97      pushup(suf) , pushup( pre) ;
98  }
99
100 void reverse(int l , int r ) {
101     int pre= get_k( l-1 ) , suf = get_k( r +1 );
102     splay( pre,0 ) ; splay( suf , pre) ;
103     pushrev( tr[suf].s[0] ) ;
104 }
105
```

```
106
107 LL query(int l , int r ) {
108     int pre= get_k( l-1 ) , suf = get_k( r +1 );
109     splay( pre,0 ) ; splay( suf , pre) ;
110     return tr[tr[suf].s[0] ].sum ;
111 }
112
113 void solve( ) {
114     cin >> n  >> m ;
115     for(int i =1; i <=n ; ++ i ) cin >>a[i];
116     for(int i = 0 ; i <=n+ 1;  ++ i ) insert( i ) ;
117     int ans[N] ;
118     while(m--){
119         int op ; cin >> op;
120         if( op == 0 ) {
121             int x, y , k ;
122             cin >> x >> y >> k ;
123             update( x , y , k ) ;
124         }else if( op == 1){
125             int x, y ; cin >> x >> y ;
126             reverse( x , y );
127         }else {
128             int x, y ; cin >> x >> y ;
129             cout << query( x , y ) <<'\n';
130         }
131
132     }
133
134 }
```

区间修改主席树

```
1  struct tree{
2      int ls ,rs  ;
3      LL tag , sum  ;
4  }tr[N*75];
5
6  int root[N] ,tot ;
7
8
9  void build( int &u ,int l , int r ) {
10     u = ++tot ;
11     tr[u] = { 0 , 0 , 0 , 0  } ;
12     if( l == r ) return ;
13     int mid = l + r >> 1  ;
14     build( tr[u].ls , l , mid ) ; build( tr[u].rs , mid + 1 , r ) ;
15 }
16
17 void update(int &u , int v ,int l ,int r , int L ,int R ,LL add ) {
18     u = ++tot ;  tr[u] = tr[v] ;  tr[u].sum += add*( R- L  + 1 )  ;
19     if( l == L && r == R ) {
20         tr[u].tag  += add ; return ;
21     }
22     int mid = l+r >>1 ;
23     if( R <= mid ) update(tr[u].ls , tr[v].ls , l , mid , L , R , add ) ;
```

```
    else if( L > mid ) update( tr[u].rs , tr[v].rs, mid + 1, r , L , R , add ) ;
    else {
        if( L <= mid ) update( tr[u].ls , tr[v].ls , l , mid , L , mid , add ) ;
        if( R > mid ) update( tr[u].rs,  tr[v].rs , mid +1 , r , mid + 1 , R , add ) ;
    }
}

LL query(int u , int v , int l ,int r ,int L , int R  ,LL addv , LL addu ) {
    if( L <= l && R >= r ) return  tr[u].sum + addu*( r - l +  1) -tr[v].sum - addv*( r - l + 1)  ;
    int mid =  l  + r >> 1; LL res= 0 ;
    if( L  <= mid ) res += query( tr[u].ls , tr[v].ls , l , mid , L , R , addv + tr[v].tag ,  addu + tr[u].tag  ) ;
    if( R > mid )  res += query( tr[u].rs , tr[v].rs , mid + 1, r , L , R , addv + tr[v].tag ,addu + tr[u].tag ) ;
    return res ;
}
```

数学

复数

```
class Complex{
    public :
        double x, y ;
    Complex( ) :x( 0) ,y( 0 ) { }
    Complex(double _x , double _y) : x(_x) , y(_y){ }
    Complex(const Complex &c) : x( c.x ) , y(c.y) { }
    Complex operator+=(const Complex &c){
        x  += c.x ; y  += c.y ;
        return *this ;
    }
    Complex operator-=(const Complex &c){
        x  -= c.x ; y  -= c.y ;
        return *this ;
    }
    Complex operator*=(const Complex &com){
        double a = x, b = y ,c = com.x , d = com.y  ;
        x = a*c - b*d ; y = a*d + b*c ;
        return *this ;
    }
    Complex operator+( const Complex &c) const {
        Complex t(*this) ;
        return  t += c ;
    }
    Complex operator-( const Complex &c) const {
        Complex t(*this) ;
        return  t -= c ;
    }
    Complex operator*( const Complex &c) const {
        Complex t(*this) ;
```

```
        return  t *= c ;
    }
};
```

快速幂

```
LL qmi(  LL a, LL b) {
    LL res =1;
    while(b) {
        if( b & 1 )res= ( res *a) % mod ;
        b >>= 1;
        a= ( a*a ) %mod ;
    }
    return res;
}
```

扩展欧几里得

```
int gcd( int x,int y ){
    if( !y) return x;
    return gcd( y,x%y);
}


int exgcd( int a,int b , int &x, int &y){
    if( b== 0 ){
        x= 1; y = 0;

        return a;
    }
    int d = exgcd( b, a%b,y,x);
    y-= a/b*x;
    return d;
}
```

逆元组合数

```
LL qmi(LL a , LL b) {
    LL res = 1 ;
    while(b  ) {
        if( b & 1)   res= (res *a) %mod ;
        a = ( a*a) %mod ;
        b >>= 1 ;
    }
    return res ;
}
LL C(int a ,int b ) {
    return fab[a]*inv[b]%mod *inv[a-b] %mod ;
}
void init() {
    inv[0] =fab[0] = 1 ;
    for(int i =1; i <N ; ++ i ) {
        fab[i] = fab[i-1]*i %mod ;
```

```
        inv[i] = inv[i-1] *qmi( i ,mod-2) %mod ;
    }
}
```

欧拉函数

```
int n;
 n= 1;
while(n--){
    LL x;
    cin>>x;
    LL res = x;
    for(int i = 2; i<=x/i ; ++i){
        if( x%i == 0){
            res= res*(i-1)/i;
            while( x%i ==0 ) x/=i;
        }
    }
    if( x>1 ) res = res*(x-1)/x;
    cout<<res<<endl;
}
```

```
cin>>n;
for(int i=  2; i<= n ; ++i){
    if( !st[i]) {
        primes[cnt++] = i;
        phi[i] = i-1;
    }
    for(int j = 0 ; primes[j]<= n/i && j<cnt ; ++j ){
        st[ primes[j]*i ] = true;
        if( i% primes[j] == 0){
            phi[ i* primes[j]] = phi[i]*primes[j];
            break;
        }else phi[ i* primes[j] ] = phi[i]*(primes[j]-1);
    }
}
```

杨辉三角组合数

```
for(int i = 0; i< N ; ++i){
    for(int j = 0 ; j<=i ; ++j){
        if( j==0) c[i][j] = 1;
        else c[i][j] = ( c[i-1][j-1]+ c[i-1][j]) % mod;
    }
}
```

Miller Rabin

```

```

图论

匈牙利

```
bool find( int x){
    for(int i = h[x] ; i!=-1 ; i = ne[i]){
        int j = e[i];
        if( !st[j]){
            st[j] = true;
            if( match[j] == 0 || find( match[j]) ){
                match[j] = x;
                return true;
            }
        }
    }
    return false;
}
```

SPFA

```
int spfa(){
    memset( dist , 0x6f , sizeof dist);
    dist[ 1] = 0;
    queue<int> q;
    q.push(1);
    st[1] = true;
    while( q.size()){
        int t = q.front();
        q.pop();
        st[t ] = false;
        for(int i = h[t] ; i!= -1 ; i= ne[i]){
            int j =e[i];
            if( dist[j] > dist[t] + w[i]) {
                dist[j] = dist[t] + w[i];
                if( !st[j]){
                    q.push(j);
                    st[j] = true;
                }

            }
        }

    }
    if( dist[n] == 0x3f3f3f) return -1;
    return dist[n];
```

```
}
```

SCC

```
class Scc {
    public :
        int size ;
        vector< vector<int> > adj ;
        stack<int> stk ;
        vector<int>  dfn , low ,id  ;
        vector<bool >  st ;
        vector< vector<int> > scc ;
        int timestop ,cnt ;
    Scc( ):size( 0 ) { }
    Scc( int _size) : size(_size) , dfn(_size+1, 0 ) ,low(_size+1, 0 ) , id(_size+1 , 0) ,st( _size + 1, false) , adj(_size+1) ,scc(_size+1) ,timestop(0) ,cnt(0){}
    Scc( const Scc &_t) {
        size = _t.size ; stk = _t.stk ; dfn = _t.dfn ;low = _t.low;id = _t.id;
        st = _t.st ; timestop = _t.timestop ; cnt = _t.cnt ; scc =_t.scc ;
    }
    ~Scc( ) { dfn.clear( ) ; low.clear() ; id.clear() ; st.clear() ; scc.clear() ;while(!stk.empty()) stk.pop( ); }
    void addedges(int a, int b ) {
        adj[a].push_back( b) ;
    }
    void run( ) {
        for(int i=1; i <=size; ++ i) if(!dfn[i]) tarjan( i)  ;
    }

    void tarjan(int u  ) {
        stk.push( u ) ; st[u] = true ;
        dfn[u] = low[u] = ++ timestop ;
        for(auto son  : adj[u]) {
            if( !dfn[son]) {
                tarjan( son) ;
                low[u] = min( low[u] ,low[son]) ;
            }else if( st[son]) low[u] =min( low[u] ,dfn[son]) ;
        }
        if( low[u] == dfn[u] ) {
            int v ;
            ++cnt ;
            do{
                v = stk.top() ;  stk.pop() ;
                id[v] = cnt ;
                scc[cnt].push_back( v) ;
                st[v] = false ;
            }while( v !=u )  ;
        }
    }

    bool same( int a, int b ) {
        return id[a] == id[b] ;
    }
```

```
48
49      int getsize(int x ) { return scc[x].size()  ;}
50
51  };
```

割点

```
1   class Tarjan {
2       public :
3           int size ;
4           vector< vector<int> > adj ;
5           vector<int>  dfn , low  ;
6           vector<bool> st ;
7           vector<int> ans ;
8           vector<int> cut ;
9           int timestop ,cnt ;
10      Tarjan( ):size( 0 ) { }
11      Tarjan( int _size) : size(_size) , cut( _size + 1  , 0 ) , dfn(_size+1,
    0 ) ,low(_size+1, 0 )  , adj(_size+1) ,st(_size+1 , false) ,timestop(0)
    ,cnt(0){}
12      Tarjan( const Tarjan &_t) {
13          size = _t.size ;dfn = _t.dfn ;low = _t.low;
14          timestop = _t.timestop ; cnt = _t.cnt ; st = _t.st ; cut = _t.cut;
15      }
16      ~Tarjan( ) { dfn.clear( ) ; low.clear() ;  }
17      void addedges(int a, int b ) {
18          adj[a].push_back( b) ;
19      }
20      void run( ) {
21          for(int i=1; i <=size; ++ i) if(!dfn[i]) tarjan( i , i)  ;
22          for(int i =1 ;i <= size;  ++ i ) if(st[i]) ans.push_back( i ) ;
23      }
24
25      void tarjan(int u  ,int fa ) {
26          dfn[u] = low[u] = ++ timestop ;
27          int child = 0 ;
28          for(auto son  : adj[u]) {
29              if(!dfn[son]) {
30                  tarjan(  son , fa ) ;
31                  low[u] = min( low[u] , low[son]) ;
32                  if( low[son] >= dfn[u] && u != fa  ) st[u] = true ,cut[u] ++
    ;
33                  if( u == fa && child > 1  ) ++ cut[u]  ;
34                  if( u == fa) ++child ;
35              }else low[u] = min( low[u] , dfn[son]) ;
36          }
37          if( child >=2 && u == fa ) st[u] = true ;
38      }
39
40  };
```

割边

```
class Tarjan {
    public :
        int size ;
        vector< vector<int> > adj ;
        vector<int>  dfn , low  ;
        vector<bool> st ;
        vector< pair<int,int> > ans ;
        vector< int> id ;
        vector<vector<int>> Ecc ;
        int timestop ,cnt ;
        stack<int> stk ;
    Tarjan( ):size( 0 ) { }
    Tarjan( int _size) : size(_size) , id( _size + 1 ,0) ,dfn(_size+1, 0 ) ,low(_size+1, 0 )  , adj(_size+1) ,st(_size+1 , false) ,timestop(0) ,cnt(0) {}
    Tarjan( const Tarjan &_t) {
        size = _t.size ;dfn = _t.dfn ;low = _t.low;
        timestop = _t.timestop ; cnt = _t.cnt ; st = _t.st ;
    }
    ~Tarjan( ) { dfn.clear( ) ; low.clear() ;  }
    void addedges(int a, int b ) {
        adj[a].push_back( b) ;
    }
    void run( ) {
        for(int i=1; i <=size; ++ i) if(!dfn[i]) tarjan( i , i)  ;
    }

    void tarjan(int u  ,int fa ) {
        dfn[u] = low[u] = ++ timestop ;
        stk.push( u ) ;
        for(auto son  : adj[u]) {
            if( son == fa) continue ;
            if(!dfn[son]) {
                if( son == fa) continue ;
                tarjan(  son , u  ) ;
                low[u] = min( low[u] , low[son]) ;
                if( dfn[u] < low[son]) ans.push_back( { u , son }) ;
            }else low[u] = min( low[u] , dfn[son]) ;
        }
        if( dfn[u] == low[u] ) {
            int now = Ecc.size() ;
            vector<int> ecc ;
            while( stk.top() != u ) {
                id[ stk.top() ] =  now ;
                ecc.push_back( stk.top()) ; stk.pop( ) ;
            }
            id[u] = now ;
            stk.pop() ;
            Ecc.push_back(  ecc ) ;
        }
    }

```

```
};
```

点分治

```
void solve( ) {
    LL ans = 0 ;
    int k1, k2 ;
    cin >> n  >> k1 >> k2;
    vector< vector<int>> adj( n +1 ) ;
    vector<int> MX( n + 1) ,siz( n + 1 ) , dis( n + 1 )  ;
    vector<bool> st( n + 1 ) ;
    for(int i =1; i <n ; ++ i ) {
        int u , v ;cin >> u >> v  ;
        adj[u].push_back( v ) ;
        adj[v].push_back( u ) ;
    }
    MX[0] = inf ;
    auto get_root =[&](auto self , int u  , int fa, int tot ,int &wc )->void
{
        MX[u] = 0 ;siz[u] = 1 ;
        for(auto v :adj[u]) {
            if( v ==  fa || st[v]  ) continue ;
            self( self, v , u , tot , wc ) ;
            siz[u] += siz[v] ;
            MX[u] = max( MX[u] , siz[v] ) ;
        }
        MX[u] = max( MX[u] , tot - siz[u] ) ;
        if( MX[u] < MX[wc] ) wc= u;
    } ;

    auto get_dis =[&](auto self ,int u , int fa , vector<int> &d )->void {
        d.push_back(  dis[u] ) ;
        for(auto v :adj[u] ) {
            if( v ==fa || st[v]) continue ;
            dis[v]  = dis[u] + 1 ;
            self(self , v , u , d ) ;
        }
    };

    auto calc =[&](auto self, int u , int  len )->LL {
        vector<int> d ;
        dis[u] = len ;
        get_dis( get_dis ,  u , u  ,d  ) ;
        sort( d.begin() , d.end()) ;
        int l = 0 , r= (int)d.size() -1 ;
        LL res = 0 ;
        while( l <= r ) {
            if( d[l] + d[r] <= k )  res += r - l  ,  ++ l ;
            else --r ;
        }
        d.clear() ;
        return res;
    };


```

```
    auto dfs = [&]( auto self , int u  , int tot  ) ->void {
        int rt = 0 ;
        get_root(  get_root , u ,  0 , tot , rt  ) ;
        u = rt ;
        ans += calc( calc , u , 0 ) ;
        st[u] = true ;
        for(auto v :adj[u]) {
            if( st[v] ) continue ;
            ans -= calc( calc , v , 1 ) ;
        }
        for(auto v : adj[u]) {
            if( st[v]) continue ;
            self( self , v , siz[v]) ;
        }
    } ;
    k = k2 ;
    dfs( dfs , 1 , n ) ;
    LL res = ans ; ans = 0 ;
    if( k1 -1 ) st.assign( n + 1 ,false) ,  k = k1-1 , dfs(dfs ,1 ,n ) ;
    cout << res -ans <<'\n';


}
```

dinic

```
LL n , m  , k  ,S,T  ;
int h[N ] ,e[M] ,ne[M]  ,  idx ;
LL f[N] , cur[N] , d[N ] ;

void add(int a, int b ,LL c  ) {
    e[idx] =b  ,ne[idx] = h[a] , f[idx] = c , h[a] = idx ++ ;
    e[idx] =a ,ne[idx] = h[b] , f[idx] = 0 , h[b] = idx++ ;
}

bool bfs( ) {
    for(int i = 0; i <=n+ m + 1  ; ++ i ) d[i] = -1 ;
    d[S] = 0 ; cur[S] = h[S] ;
    queue<int> q; q.push( S) ;
    while(q.size() ) {
        int  u = q.front()  ;q.pop() ;
        for(int i =h[u]; ~i; i = ne[i]){
            int j = e[i] ;
            if(  d[j] == -1 && f[i]) {
                d[j] = d[u] + 1 ;
                cur[j] = h[j] ;
                if( j == T) return true ;
                q.push( j ) ;
            }
        }
    }
    return false ;
}

LL dfs(int u , LL limit) {
```

```
    if( u == T)  return  limit ;
    LL flow = 0 ;
    for(int i = cur[u] ;~i &&  flow < limit ; i = ne[i]) {
        cur[u] = i ;
        int j = e[i] ;
        if( d[j] == d[u] + 1 && f[i]) {
            LL  t = dfs( j ,  min(f[i] , limit - flow ))  ;
            if( !t ) d[j] = -1 ;
            f[i] -=t ; f[i^1] += t ; flow += t ;
        }
    }
    return flow ;
}


LL dinic( ) {
    LL res=  0 , flow = 0 ;
    while( bfs( )) while( flow = dfs( S , INF )) res += flow ;
    return res ;
}
```

最小费用最大流

```
LL n , m  , k  , S  , T  ;
int h[N] , e[N] , ne[N] , idx ;
LL f[N] , w[N] , d[N] , incf[N] ;
int pre[N] ;
bool st[N] ;
void add(int a, int b , int c , int d ) {
    e[idx] =b, ne[idx] = h[a] ,f[idx] = c, w[idx] = d ,h[a] = idx ++ ;
    e[idx] = a, ne[idx] = h[b] , f[idx] = 0 , w[idx] = -d , h[b] = idx++ ;
}

bool spfa( ) {
    queue<int> q;
    q.push( S) ;
    for(int i =1; i <=n ; ++ i ) d[i] = INF  ,incf[i] = 0  ;
    d[S] = 0 ;  incf[S] =INF ;
    st[S] =  true ;
    while(q.size( )) {
        int u = q.front( ) ;q.pop( ) ;
        st[u] = false ;
        for(int i = h[u] ;~i ; i =ne[ i]) {
            int j =e[i ] ;
            if( d[j] > d[u] + w[i] && f[i]) {
                d[j] = d[u] + w[i] ;
                incf[j] = min(f[i] , incf[u])  ;
                pre[j] = i  ;
                if( !st[j]) {
                    st[j] = true ;
                    q.push( j  ) ;
                }
            }
        }
    }
```

```
    return incf[T] > 0 ;
}


void EK( LL &flow ,LL &cost ) {
    flow = 0 , cost = 0 ;
    while( spfa( )) {
        LL t =  incf[T];
        flow += t , cost += t *d[T] ;
        for(int i = T ;  i!= S ; i = e[ pre[i] ^1 ]) {
            f[ pre[i] ] -= t ; f[ pre[i] ^ 1 ] +=t ;
        }
    }
}
```

字符串

KMP

```
void get(string s) {
    ne[0] = ne[1 ] = 0 ;
    for(int i =2 , j = 0 ; i < s.size() ; ++ i) {
        while( j && s[j+1] != s[i] ) j = ne[j] ;
        if( s[i] == s[j + 1 ]) ++ j ;
        ne[i] = j ;
    }
}
```

字符串哈希

```
#include <random>
#include <chrono>
std::mt19937
rng(std::chrono::steady_clock::now().time_since_epoch().count());

bool isprime(int n) {
    if (n <= 1) return false;
    for (int i = 2; i * i <= n; i++)
        if (n % i == 0)
            return false;
    return true;
}
int findPrime(int n) {
    while (!isprime(n))
        n++;
    return n;
}

template<int N>
struct StringHash {
    static array<int, N> mod;
    static array<int, N> base;
    vector<array<int, N>> p, h;
```

```
    StringHash() = default;
    StringHash(const string& s) {
        int n = s.size();
        p.resize(n);
        h.resize(n);
        fill(p[0].begin(), p[0].end(), 1);
        fill(h[0].begin(), h[0].end(), 1);
        for (int i = 0; i < n; i++)
            for (int j = 0; j < N; j++) {
                p[i][j] = 1ll * (i == 0 ? 1ll : p[i - 1][j]) * base[j] % mod[j];
                h[i][j] = (1ll * (i == 0 ? 0ll : h[i - 1][j]) * base[j] + s[i]) % mod[j];
            }
    }
    array<int, N> query(int l, int r) {
//        assert(r >= l - 1);
        array<int, N> ans{};
        if (l > r) return {0, 0};
        for (int i = 0; i < N; i++) {
            ans[i] = (h[r][i] - 1ll * (l == 0 ? 0ll : h[l - 1][i]) * (r - l + 1 == 0 ? 1ll : p[r - l][i]) % mod[i] + mod[i]) % mod[i];
        }
        return ans;
    }
};

constexpr int HN = 2;
template<>
array<int, 2> StringHash<HN>::mod =
        {findPrime(rng() % 900000000 + 100000000),
         findPrime(rng() % 900000000 + 100000000)};
template<>
array<int, 2> StringHash<HN>::base {13331, 131};
using Hashing = StringHash<HN>;

```

马拉车

```
string manacher( string  t) {
    string s ;  s += '@' ;
    for(auto ch : t )  s +='#' , s += ch ;
    s += "#$" ;
    int mid = 1 , r= 1  , len = 0  ,ans = 0  ;
    for(int i = 1  ; i < s.size() ; ++ i ) {
        if( i < r   )  d[i] = min( d[(mid<<1) - i]  , r - i  ) ;
        else d[i] = 1 ;
        while( s[ i - d[i]] == s[i + d[i]] ) ++d[i] ;
        if( i + d[i]  > r )  r = i + d[i]  , mid = i ;
        if( d[i] > len ) ans = i , len = d[i] ;
    }
    cout << len <<'\n';
    string res ;
    for(int i = ans - len + 1 ;  i <= ans + len -1 ; ++ i   ) {
        if( s[i] != '#') res += s[i] ;
```

```
    }
    return res ;
}
```

最小表示法

```
int get_min(  string s ) {
    int n = s.size() ;
    s = s + s;
    int i = 0 ,  j = 1, k = 0 ;
    while( i < n && j < n && k < n ) {
        if( s[i+k] == s[j+k] ) ++ k;
        else if( s[i+k] > s[j+k] ) i += k + 1 , k = 0 ;
        else j  += k + 1 , k = 0 ;
        if( i == j ) ++ j ;
    }
    return min( i , j ) ;
}
```