MODEL BY VOIDJACKLEE

2020-10-13

ZCMU

目录

Header

```
#include <stdio.h>
#include <stdarg.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <time.h>
#include <limits.h>
#include <vector>
#include <list>
#include <set>
#include <utility> // pair
#include <map>
#include <iostream>
#include <sstream>
#include <algorithm> // sort
#include <functional>
#include <string>
#include <stack>
#include <queue>
#include <fstream>
#include <bitset>

//#include <unordered_map>
//#include <unordered_set>

using namespace std;

#define ll long long
#define lll __int128
#define uchar unsigned char
#define ushort unsigned short
#define uint unsigned int
#define ulong unsigned long
#define ull unsigned long long

#define INT_INF 0x7fffffff

#define pi acos(-1)

#define mem(a,b) memset(a,b,sizeof(a))
#define memn(a,b,c,n) memset(a,b,sizeof(c)*(n))
#define fre(a) freopen(a,"r",stdin)

#define cio ios::sync_with_stdio(false); // Do not use it with "scanf" and other c input!
#define pb push_back
#define mpair make_pair
```

```
#define rep(i,a,b) for(int i=a;i<=b;i++)
#define pre(i,a,b) for(int i=a;i>=b;i--)
#define REP(i,a,b) for(int i=a;i<b;i++)

#define reada(a,s,n) rep(i,s,n)scanf("%d",a + i);
#define READa(a,s,n) REP(i,s,n)scanf("%d",a + i);

#define readall(a,s,n) rep(i,s,n)scanf("%lld",a + i);
#define READall(a,s,n) REP(i,s,n)scanf("%lld",a + i);

//template <typename _Tp> inline void read(_Tp&x) {
//    char ch;bool flag=0;x=0;
//    ch=getchar();
//    while(!isdigit(ch)){if(ch=='-')flag=1;ch=getchar();}
//    while(isdigit(ch))x=x*10+ch-'0',ch=getchar();
//    if(flag)x=-x;
//}
//inline void print_lll(lll x) {
//    if(x<0) {x=-x;putchar('-');}
//    if(x>9) print_lll(x/10);
//    putchar(x%10+'0');
//}

#define _T_(T) int T;scanf("%d",&T);while(T--)
#define __T _T_(TTESTCASES)
#define _E_(T) while(~T)

#define _C(a) cout << a << endl;

#define dsci(a) int a;scanf("%d",&a)
#define dscii(a,b) int a,b;scanf("%d%d",&a,&b)
#define dsciii(a,b,c) int a,b,c;scanf("%d%d%d",&a,&b,&c)
#define dscl(a) ll a;scanf("%lld",&a)
#define dscll(a,b) ll a,b;scanf("%lld%lld",&a,&b)
#define dsclll(a,b,c) ll a,b,c;scanf("%lld%lld%lld",&a,&b,&c)
#define dscd(a) double a;scanf("%lf",&a)
#define dscdd(a,b) double a,b;scanf("%lf%lf",&a,&b)
#define dscddd(a,b,c) double a,b,c;scanf("%lf%lf%lf",&a,&b,&c)

#define sci(a) scanf("%d",&a)
#define scii(a,b) scanf("%d%d",&a,&b)
#define sciii(a,b,c) scanf("%d%d%d",&a,&b,&c)
#define scl(a) scanf("%lld",&a)
#define scll(a,b) scanf("%lld%lld",&a,&b)
#define sclll(a,b,c) scanf("%lld%lld%lld",&a,&b,&c)
#define scd(a) scanf("%lf",&a)
#define scdd(a,b) scanf("%lf%lf",&a,&b)
#define scddd(a,b,c) scanf("%lf%lf%lf",&a,&b,&c)
```

```
#define lowbit(x) ((x)&(-(x)))

#define Tprint(a,s,e) REP(i,s,e){if(i!=s)printf(" ");printf("%lld",a
[i]);}

#define endl '\n'

#define itn int
#define iny int
#define nit int
#define inr int
#define mian main
#define iman main
#define mina main
#define mian main
#define ednl endl
#define fro for
#define fir for
#define reutrn return
#define retunr return
#define reutnr return
#define re0 return 0
#define re1 return 1
```

差分 & 前缀和

差分

```
int p[1000] = {0};
int a[1000];
int n;

void pls(int l,int r,int k)
{
    p[l] += k;
    p[r + 1] -= k;
}

void init()
{
    REP(i,0,n) {
        pls(i,i,a[i]);
    }
}

int main()
```

```
{
    scanf("%d",&n);

    READ(a,0,n);
    init();

    Tprint(p,0,n);
    printf("\n");

    int q;
    scanf("%d",&q);
    int l,r;
    while (q --) {
        scanf("%d%d",&l,&r);
        pls(l,r,1);
    }

    int s[1000];

    s[0] = p[0];
    rep(i,1,n - 1) {
        s[i] = s[i - 1] + p[i];
    }

    rep(i,0,n - 1) printf("%d ",s[i]);
    return 0;
}
```

二维差分

```
int p[1000][1000] = {0};
int a[1000][1000];

void pls(int a,int b,int x,int y,int k)
{
    p[a][b] += k;
    p[x + 1][y + 1] += k;
    p[x][y + 1] -= k;
    p[x + 1][y] -= k;
}

int main()
{
    int n,m;
    scanf("%d%d",&n,&m);
    rep(i,1,n) {
        rep(j,1,m) {
            scanf("%d",&a[i][j]);
        }
    }

    rep(i,1,n) {
        rep(j,1,m) {
            pls(i,j,i,j,a[i][j]);
        }
    }

    rep(i,1,n) {
        rep(j,1,m) {
            printf("%d ",p[i][j]);
        }
        printf("\n");
    }
    printf("\n");

    int sum[1000][1000] = {0};
    sum[1][1] = p[1][1];

    rep(i,1,n) {
        rep(j,1,m) {
            sum[i][j] = sum[i - 1][j] + sum[i][j - 1] - sum[i - 1][j -
1] + p[i][j];
        }
    }

    rep(i,1,n) {
        rep(j,1,m) {
            printf("%d ",sum[i][j]);
        }
        printf("\n");
    }


    return 0;
}
```

二维前缀和

```
int main()
{
    int arr[1000][1000] = {0};
    int sum[1000][1000] = {0};
    int n,m;
    scanf("%d%d",&n,&m);
    rep(i,1,n) {
        rep(j,1,m) {
            scanf("%d",&arr[i][j]);
        }
    }

    rep(i,1,n) {
        rep(j,1,m) {
            printf("%d ",arr[i][j]);
        }
        printf("\n");
    }

    puts("");

    sum[1][1] = arr[1][1];
//    rep(i,1,m) sum[0][i] = sum[0][i - 1] + arr[0][i];
//    rep(i,1,n) sum[i][0] = sum[i - 1][0] + arr[i][0];

    rep(i,1,n) {
        rep(j,1,m) {
            sum[i][j] = sum[i - 1][j] + sum[i][j - 1] - sum[i - 1][j -
1] + arr[i][j];
        }
    }

    // sum up
    rep(i,1,n) {
        rep(j,1,m) {
            printf("%d ",sum[i][j]);
        }
        printf("\n");
    }

    // obtain
    int q;
    scanf("%d",&q);
    int a,b,x,y;
    while (q --) {
        scanf("%d%d%d%d",&a,&b,&x,&y);
        printf("%d\n",sum[x][y] - sum[a - 1][y] - sum[x][b - 1] + sum[a
 - 1][b - 1]);
    }

    return 0;
}
```

树状数组

单点修改 & 区间查询

```
// 树状数组：单点修改，区间和查询
const int MAXN = 5e5 + 10;

ll a[MAXN];
ll c[MAXN];

int n,m;

void modify(int i,ll x)
{
    // 从叶子结点一路向上更新
    for (;i <= n;i += lowbit(i)) {
        c[i] += x;
    }
}

ll sum(int i)
{
    // 查询：  由于每个c结点相当于一小段前缀和，因此全+起来，最后求得的便是总共的前缀和
    ll ans = 0;
    for (;i > 0;i -= lowbit(i))
    {
        ans += c[i];
    }
    return ans;
}

int main()
{
    scii(n,m);
    rep(i,1,n) scl(a[i]);
    rep(i,1,n) modify(i, a[i]);
    int t,x,y;
    while (m --) {
        sciii(t,x,y);
        if (t == 1) {
            // modify
            modify(x, y);
        } else if (t == 2) {
            // query
            printf("%lld\n",sum(y) - sum(x - 1));
        }
    }
    re0;
}
```

(+差分) 区间修改 & 单点查询

```
// 树状数组+差分： 区间修改+单点查询
const int MAXN = 5e5 + 10;

int a[MAXN];
int c[MAXN];

int n,m;

void modify(int idx,ll x)
{
    // 从叶子结点一路向上更新
    for (int i = idx;i <= n;i += lowbit(i))
    {
        c[i] += x;
    }
}

ll sum(int idx)
{
    // 查询：由于每个 c 结点相当于一小段前缀和，因此全+起来，最后求得的便是总
共的前缀和
    ll ans = 0;
    for (int i = idx;i > 0;i -= lowbit(i))
    {
        ans += c[i];
    }
    return ans;
}

void pls(int l,int r,int k)
{
    modify(l, k);
    modify(r + 1, -k);
}

void init()
{
    rep(i,1,n) {
        pls(i,i,a[i]);
    }
}
```

```
int main()
{
    scii(n,m);
    rep(i,1,n) sci(a[i]);
    init();

    int t;
    int x,y,k;

    while (m --) {
        sci(t);
        if (t == 1) {
            sciii(x,y,k);
            // modify
            pls(x,y,k);
        } else if (t == 2) {
            sci(x);
            // get_ans: sum_up
            printf("%lld\n",sum(x));
        }
    }
    return 0;
}
```

求最值

```
int n;

const int MAXN = 1e5 + 10;

int a[MAXN]; // 原数组
ll c[MAXN]; // 求和树状数组

void modify(int i,ll x)
{
    // 从叶子结点一路向上更新
    for (;i <= n;i += lowbit(i)) {
        c[i] += x;
    }
}

ll sum(int i)
{
    // 查询： 由于每个c结点相当于一小段前缀和，因此全+起来，最后求得的便是总
共的前缀和
    ll ans = 0;
    for (;i > 0;i -= lowbit(i))
    {
        ans += c[i];
    }
    return ans;
}

int mx[MAXN]; // 最大值数状数组

void modify_m(int i,int x)
{
    int low;
    a[i] = x; // 会直接修改数组的值
    for (;i <= n;i += lowbit(i)) {
        mx[i] = a[i];
        low = lowbit(i);
        for (int j = 1;j < low;j <<= 1) {
            mx[i] = max(mx[i], mx[i - j]);
        }
    }
}

int query_max(int l,int r)
{
    int ans = max(a[l],a[r]);
    while(true)
    {
        ans = max(ans, a[r]);
        if (l == r) break;
        r --;
        for (;r - l > lowbit(r);r -= lowbit(r))
        {
            ans = max(ans,mx[r]);
        }
    }
    return ans;
}

int main()
{
    int m;
    scii(n,m);
    rep(i,1,n) {
        sci(a[i]);
        modify_m(i, a[i]); // 修改最大值
        modify(i, a[i]);
    }
    int t,x,y;
    while (m --) {
        sciii(t,x,y);
        if (t == 1) {
            modify(x, y - a[x]);
            modify_m(x, y);
        } else if (t == 2) {
            printf("%lld\n",sum(y) - sum(x - 1));
        } else printf("%d\n",query_max(x, y));
    }
    return 0;
}
```

求逆序数

```
// 树状数组求逆序数
const int MAXN = 5e4 + 10;

int a[MAXN];
int b[MAXN];
ll c[MAXN];

int n,m;

void modify(int idx,ll x)
{
    // 从叶子结点一路向上更新
    for (int i = idx;i <= n;i += lowbit(i)) {
        c[i] += x;
    }
}

ll sum(int idx)
{
    // 查询： 由于每个c结点相当于一小段前缀和，因此全+起来，最后求得的便是总共的前缀和
    ll ans = 0;
    for (int i = idx;i > 0;i -= lowbit(i))
    {
        ans += c[i];
    }
    return ans;
}

int main()
{
    int *x;
    ll ans;
    int idx;
    while (~sci(n)) {

        rep(i,1,n) {
            sci(a[i]);
            b[i] = a[i];
            c[i] = 0;
        }
        sort(a + 1, a + 1 + n);
        x = unique(a + 1, a + 1 + n);
//        for (int *i = a + 1;i != x;i ++) printf("%d ",*i);
//        puts("");
        ans = 0;
        pre(i,n,1) {
            idx = (int) (lower_bound(a + 1, x, b[i]) - a);

            ans += sum(idx);
            modify(idx, 1);
        }
        printf("%lld\n",ans);
    }
    re0;
}
```

线段树

```
// 线段树 - 二叉树，节点存的是一个 l，r，区间的内容n
const int MAXN = 1e5 + 10;

struct Node {
    int l,r;
    ll mx;
    ll mn;
    ll sum;
    int lazy;
    ll lzn;
} tree[MAXN << 2];
ll a[MAXN];

void push_up(int i)
{
    tree[i].sum = tree[i << 1].sum + tree[i << 1 | 1].sum;
    tree[i].mn = min(tree[i << 1].mn,tree[i << 1 | 1].mn);
    tree[i].mx = max(tree[i << 1].mx,tree[i << 1 | 1].mx);
}

void push_down(int i) //下推标记
{
    if (tree[i].lazy == 1) {
        tree[i << 1].sum += (tree[i << 1].r - tree[i << 1].l + 1) * tre
e[i].lzn;
        tree[i << 1 | 1].sum += (tree[i << 1 | 1].r - tree[i << 1 | 1].
l + 1) * tree[i].lzn;

        tree[i << 1].mx += tree[i].lzn;
        tree[i << 1 | 1].mx += tree[i].lzn;

        tree[i << 1].mn += tree[i].lzn;
        tree[i << 1 | 1].mn += tree[i].lzn;

        tree[i << 1].lzn += tree[i].lzn;
        tree[i << 1 | 1].lzn += tree[i].lzn;

        tree[i << 1].lazy = tree[i].lazy;
        tree[i << 1 | 1].lazy = tree[i].lazy;

        tree[i].lazy = 0;
        tree[i].lzn = 0;
    } else if (tree[i].lazy == 2) {
        tree[i << 1].sum = (tree[i << 1].r - tree[i << 1].l + 1) * tree
[i].lzn;
        tree[i << 1 | 1].sum = (tree[i << 1 | 1].r - tree[i << 1 | 1].l
 + 1) * tree[i].lzn;

        tree[i << 1].mx = tree[i].lzn;
        tree[i << 1 | 1].mx = tree[i].lzn;

        tree[i << 1].mn = tree[i].lzn;
        tree[i << 1 | 1].mn = tree[i].lzn;

        tree[i << 1].lzn = tree[i].lzn;
        tree[i << 1 | 1].lzn = tree[i].lzn;

        tree[i << 1].lazy = tree[i].lazy;
        tree[i << 1 | 1].lazy = tree[i].lazy;

        tree[i].lazy = 0;
        tree[i].lzn = 0;
    }
}

// i - 二叉树节点编号，调用时取1
// l，r 区间左右端下标，调用的时候取最大范围即可 build(1,n,1);
void build(int l,int r,int i)
{
    tree[i].l = l;
    tree[i].r = r;
    tree[i].lazy = 0;
    tree[i].lzn = 0;
    if (l == r) {
        tree[i].sum = a[l]; // a 原数组，把原来的数值给叶子结点
        tree[i].mx = a[l];
        tree[i].mn = a[l];
        return;
    }
    int m = (l + r) >> 1;
    build(l,m,i << 1);
    build(m + 1,r,i << 1 | 1);
    push_up(i);
}

void add(int l,int r,ll x,int i) // 将区间[l,r]整个加上x，调用(l,r,x,1)
{
    if (l <= tree[i].l && r >= tree[i].r) {
        tree[i].sum += (tree[i].r - tree[i].l + 1) * x;
        tree[i].mx += x;
        tree[i].mn += x;
        tree[i].lzn += x;

        tree[i].lazy = 1;
        return;
    }
    push_down(i);
    int m = (tree[i].l + tree[i].r) >> 1;
    if (l <= m) add(l,r,x,i << 1);
    if (r > m) add(l,r,x,i << 1 | 1);
    push_up(i);
}

void modify(int l,int r,ll x,int i) // 将区间[l,r]直接变成x，调用(l,r,x,1)
{
    if (l <= tree[i].l && r >= tree[i].r) {
        tree[i].sum = (tree[i].r - tree[i].l + 1) * x;
        tree[i].mx = x;
        tree[i].mn = x;
        tree[i].lzn = x;

        tree[i].lazy = 2;
        return;
    }
    push_down(i);
    int m = (tree[i].l + tree[i].r) >> 1;
    if (l <= m) modify(l,r,x,i << 1);
    if (r > m) modify(l,r,x,i << 1 | 1);
    push_up(i);
}

ll query(int l,int r,int i) //查询
```

```
{
    if (l <= tree[i].l && r >= tree[i].r){
        return tree[i].sum;
//        return tree[i].mx;
//        return tree[i].mn;
    }
    push_down(i);
    int m = (tree[i].l + tree[i].r) >> 1;
    ll sum = 0;
//    ll mx = 0;
//    ll mn = INT_INF;
    if (l <= m) {
         sum += query(l,r,i << 1);
//         mx = max(mx,query(l,r,i << 1));
//         mn = min(mn,query(l,r,i << 1));
    }
    if (r > m) {
        sum += query(l,r,i << 1 | 1);
//        mx = max(mx,query(l,r,i << 1 | 1));
//        mn = min(mn,query(l,r,i << 1 | 1));
    }
    return sum;
//    return mx;
//    return mn;
}

int main()
{
    dscii(n,m);
    rep(i,1,n) scl(a[i]);
    build(1, n, 1);
    int k,a,b;
    ll c;
    while (m --) {
        sciii(k,a,b);
        if (k == 1) {
            printf("%lld\n",query(a, b, 1));
        } else if (k == 2) {
            scl(c);
            modify(a, b, c, 1);
            rep(i,1,n) printf("a[%d]=%lld\n",i,query(i, i, 1));
        }
    }
    re0;
}
```

Utils

gcd

```
ll gcd(ll a,ll b)
{
    if (a % b == 0) return b;
    return gcd(b, a % b);
}
```

exgcd

```
/// 欧几里得扩展exgcd
/// ax + by = gcd(a,b)
ll exgcd(ll a, ll b, ll &x, ll &y)
{
    if (b == 0)
    {
        x = 1;
        y = 0;
        return a;
    }
    ll r = exgcd(b, a % b, x, y);
    ll t = y;
    y = x - (a / b) * y;
    x = t;
    return r;
}
```

ax + by = gcd(a,b)的通解为：

$$\begin{cases} x = x_0 - c\dfrac{b}{gcd(a,b)} \\ y = y_0 + c\dfrac{a}{gcd(a,b)} \end{cases} c\text{是任意常数}$$

quickpow

- 可以求 a 的逆元:quickpow(a % mod,mod - 2)，注意 a 的范围，它也要%mod过才不会爆精度。

```
const int mod = 1000000007;

ll quickpow(ll a, ll b)
{
    ll ans = 1;
    while (b)
    {
        if (b & 1) ans = a * ans % mod;
        a = a * a % mod;
        b >>= 1;
    }
    return ans;
}
```

dfs 序

可以将**树**线段化。

```
const int MAXN = 1e5 + 10;

int dfn = 0;
int in[MAXN],out[MAXN];

void dfs(int k,int f)
{
    in[k] = ++ dfn;
    for (int i = g[k];~i;i = e[i].nxt)
    {
        if (e[i].to != f) {
            dfs(e[i].to,k);
        }
    }
    out[k] = dfn;
}
```

Euler

Euler Function

$$\varphi(x) = x(1 - \frac{1}{p_1})(1 - \frac{1}{p_2})\ldots(1 - \frac{1}{p_n})$$

$p_1, p_2, \ldots, p_n$是 x 的所有质因子。

```
ll eular(ll x)
{
    ll ans = x;
    for (int i = 2;i * i <= x;i ++)
    {
        if (x % i == 0)
        {
            ans = ans / i * (i - 1);
            while (x % i == 0) x /= i;
        }
    }
    if (x > 1) ans = ans / x * (x - 1);
    return ans;
}
```

欧拉降幂(指数取余)

$$a^p \equiv a^{pmod\varphi(m)+\varphi(m)}(mod\ m)$$

```
const int mod = 1000000007;
ll e = eular(mod);
ll ans = quickpow(a % mod,b % e + e);
```

Matrix

```
/// 矩阵模版 快速幂
const int mod = 1e9 + 7;

class Matrix
{
    const static int MAXN = 10;

    ll a[MAXN][MAXN];

public:
    int w;
    int h;

    Matrix(int h,int w):w(w),h(h)
    {
        mem(a,0);
    }

    Matrix(const char format[], ...) {

        va_list args;
        w = 0;
        h = 0;

        char buf[1000];

        va_start(args, format);
        vsprintf(buf, format, args);
        va_end(args);
        stringstream ss(buf);
        stringstream num;
        ll n;
        string line;
        while (getline(ss, line)) {
            num.clear();
            num << line;
            w = 0;
            while (num >> n) {
                a[h + 1][++ w] = n;
            }
            h ++;
        }
    }

    void E()
    {
        if (w == h) {
            mem(a,0);
            rep(i,1,w) a[i][i] = 1;
        }
    }

    void print()
    {
        int f = 1;
        rep(i,1,h) {
            f = 1;
            rep(j,1,w) {
                if (f) f = 0;
                else printf(" ");
                printf("%lld",a[i][j]);
            }
            puts("");
        }
    }

    void read_in()
    {
        rep(i,1,h) {
            rep(j,1,w) {
                scanf("%lld",&a[i][j]);
            }
        }
    }

    Matrix operator* (const Matrix &B) const
    {
        if (w != B.h) return Matrix(0,0); // invalid

        Matrix ans(h,B.w);
        rep(i,1,h) {
            rep(j,1,B.w) {
                rep(k,1,w) {
//                      ans[i][j] = (ans[i][j] + a[i][k] * B[k][j]); // N
ot Moduled
                    ans[i][j] = (ans[i][j] + a[i][k] * B[k][j] % mod) %
 mod; // Moduled
                }
            }
        }
        return ans;
    }

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

    ll* operator[] (int i) {
        return a[i];
    }
};

Matrix quickpow(Matrix a, ll b)
{
    if (a.h != a.w) return Matrix(0,0); // invalid

    Matrix ans(a.h,a.w);
    rep(i,1,ans.h) ans[i][i] = 1; // Set ans matrix to E
    while (b)
    {
        if (b & 1) ans = a * ans;
        a = a * a;
        b >>= 1;
    }
    return ans;
}
```

sample of use:

https://www.luogu.com.cn/problem/P1939

$$a_x = \begin{cases} 1, x = 1,2,3 \\ a_x = a_{x-1} + a_{x-3}, x \geq 4 \end{cases}$$

```
/// Sample of build a Matrix to solve num sequent.

int main()
```

```
{
    Matrix p("1 0 1\n1 0 0\n0 1 0"); // Construct a matrix
    Matrix base("1\n1\n1");
    int n;
    __T {
        scanf("%d",&n);
        if (n <= 3) {
            printf("1\n");
            continue;
        }
        printf("%lld\n",(quickpow(p, n - 3) * base)[1][1]);
    }
    return 0;
}
```

Forward Star

No weight

```
const int MAXN = 1e5 + 10;

struct Edge {
    int to;
    int nxt;
} e[MAXN * 2];
int g[MAXN]; // Please call init() to memset it to -1!
int cnt = 0;

void init(int n)
{
    cnt = 0;
    memn(g,-1,int,n);
}

void add_edge(int u,int v)
{
    e[cnt] = {v,g[u]};
    g[u] = cnt ++;
}
```

With weight

```
const int MAXN = 1e5 + 10;

struct Edge {
    int to;
    ll w;
    int nxt;
} e[MAXN * 2];
int g[MAXN]; // Please call init() to memset it to -1!
int cnt = 0;

void init(int n)
{
    cnt = 0;
    memn(g,-1,int,n);
}

void add_edge(int u,int v,ll w)
{
    e[cnt] = {v,w,g[u]};
    g[u] = cnt ++;
}
```

Dijkstra

Build Graphic from Vector

```
const int MAXN = 1e5 + 10;

int dis[MAXN];
int vis[MAXN];

struct ST {
    int n;
    int w;

    bool operator< (const ST &other) const {
        return w > other.w;
    }
};

vector<ST> g[MAXN];

void dij(int s) {
    mem(dis,-1);
    mem(vis,0);

    priority_queue<ST> q;
    q.push({s,dis[s] = 0});

    ST current;
    int k;
    while (!q.empty()) {
        current = q.top();
        q.pop();
        if (vis[current.n]) continue;
        vis[current.n] = 1;

        for (auto to : g[current.n]) {
            k = current.w + to.w;
            if (dis[to.n] == -1 || dis[to.n] > k) {
                q.push({to.n,dis[to.n] = k});
            }
        }
    }
}
```

Build Graphic from Forward Star

```
const int MAXN = 1e5 + 10;

struct Edge {
    int to;
    int w;
    int nxt;
} e[MAXN * 2];
int g[MAXN]; // Please call init() to memset it to -1!
int cnt = 0;

void init(int n)
{
    cnt = 0;
    memn(g,-1,int,n);
}

void add_edge(int u,int v,int w)
{
    e[cnt] = {v,w,g[u]};
    g[u] = cnt ++;
}


int dis[MAXN];
int vis[MAXN];

struct ST {
    int n;
    int w;

    bool operator< (const ST &other) const {
        return w > other.w;
    }
};

void dij(int s) {
    mem(dis,-1);
    mem(vis,0);

    priority_queue<ST> q;
    q.push({s,dis[s] = 0});

    ST current;
    int k,to;
    while (!q.empty()) {
        current = q.top();
        q.pop();
        if (vis[current.n]) continue;
        vis[current.n] = 1;

        for (int i = g[current.n];~i;i = e[i].nxt) {
            to = e[i].to;
            k = dis[current.n] + e[i].w;
            if (dis[to] == -1 || dis[to] > k) {
                q.push({to,dis[to] = k});
            }
        }
    }
}

int main()
{
    int n,m;
    int u,v,w;
    int s,t;
    __T {
        scanf("%d%d",&n,&m);
        init(n + 5);
        while (m --) {
            scanf("%d%d%d",&u,&v,&w);
            add_edge(u,v,w);
            add_edge(v,u,w);
        }
        scanf("%d%d",&s,&t);
        dij(s);
        printf("%d\n",dis[t]);
    }
    return 0;
}
```

dij 变式，多状态

```
const int MAXN = 1e5 + 10;

struct ST {
    int n;
    double w;
    int s;

    bool operator< (const ST &other) const {
        return w > other.w;
    }
};

double dis[MAXN][3];
int vis[MAXN][3];
vector<ST> g[MAXN];

void dij(int s) {
    rep(i,1,MAXN) dis[i][0] = dis[i][1] = dis[i][2] = -1;
    mem(vis,0);


    priority_queue<ST> q;
    q.push({s,dis[s][0] = 0,0});

    inr step;

    ST current;
    double k,cost;
    while (!q.empty()) {
        current = q.top();
        q.pop();

        if (vis[current.n][current.s]) continue;
        vis[current.n][current.s] = 1;

        for (auto to : g[current.n]) {

            cost = to.w;

            // update costs from step(mode)
            //
            // REP(i,0,current.s % 3) {
            //    cost = 1 / (1 - cost);
            // }
            // cost = fabs(cost);
```

```
                // update step
                // step = (current.s + 1) % 3;

                k = dis[current.n][current.s] + cost;
                if (dis[to.n][step] == -1 || dis[to.n][step] > k) {
                    q.push({to.n,dis[to.n][step] = k,step});
                }
            }
        }
    }
```

并查集

路径压缩

```
const int MAXN = 1e5 + 10;

int find_set[MAXN];

int find(int a)
{
    if (find_set[a] == a) return a;
    return find_set[a] = find(find_set[a]);
}

inline void bind(int a,int b)
{
    find_set[find(a)] = find(b);
}

void init(int n)
{
    rep(i,0,n) {
        find_set[i] = i;  // 每个种类初始状态只有自己一个点
    }
}
```

路径压缩+按秩合并

```
const int MAXN = 1e5 + 10;

int find_set[MAXN];
int depth[MAXN];

int find(int a)
{
    if (find_set[a] == a) return a;
    return find_set[a] = find(find_set[a]);
}

inline void bind(int a,int b)
{
    int x = find(a), y = find(b);
    if (depth[x] >= depth[y]) { // 如果a的 根的子树深度 比b的 根的子树深度 大，那a的根继续做根
        find_set[y] = x; // 改变b节点的根的根为a的根
        if (depth[x] == depth[y]) { // 俩根深度一样
            if (x != y) depth[x] ++; // 作为a的根，自然子树的深度++
        }
    } else find_set[x] = y;
}

void init(int n)
{
    rep(i,0,n) {
        find_set[i] = i;  // 每个种类初始状态只有自己一个点
        depth[i] = 1;  // 初始化秩
    }
}
```

ST 表

```
const int MAXN = 100010;

int st[MAXN][20];
int a[MAXN];

int n,m;

inline int read_int()
{
    int x = 0,f = 1;
    char ch = getchar();
    while (!isdigit(ch)) {
        if (ch == '-') f = -1;
        ch = getchar();
    }
    while (isdigit(ch))
    {
        x = x * 10 + ch - 48;
        ch = getchar();
    }
    return x * f;
}

void init() {
    // 定义 st[i][j] 是从i开始，到i + 2^j这一段，即[i,i + 2^j]这一段中的最大/小值
    rep(i,1,n) st[i][0] = a[i];

    for (int j = 1;(1 << j) <= n;j ++) { // 遍历所有的j，j是一个很小的数字，最大值=log2(n)
        rep(i,1,n - (1 << j) + 1) { // 在[1,n]区间范围内，确定j的情况下，把所有的i都遍历求值一遍
            st[i][j] = max(st[i][j - 1], st[i + (1 << (j - 1))][j - 1]); // 套公式
        }
    }
}

int query(int l, int r)
{
    int x = log2(r - l + 1);
    return max(st[l][x],st[r - (1 << x) + 1][x]);
}

int main()
{
    scanf("%d%d",&n,&m);
    rep(i,1,n) {
        a[i] = read_int();
    }
    init();
    int l,r;
    while (m --) {
        scanf("%d%d",&l,&r);
        printf("%d\n",query(l, r));
    }
    return 0;
}
```

求树上一条边两边点的个数

```
const int MAXN = 1e5 + 10;

int all = 0;
int nn;

ll ch[MAXN];
ll a[MAXN];
```

```
struct Edge {
    int to;
    int nxt;
} e[MAXN * 2];
int g[MAXN]; // Please memset it to -1!!!!!!!!
int cnt = 0;

void init(int n)
{
    cnt = 0;
    memn(g,-1,int,n);
}

void add_edge(int u,int v)
{
    e[cnt] = {v,g[u]};
    g[u] = cnt ++;
}


// 方法：记录每个节点的子节点数，在用总结点数减去该边子节点数即可
void dfs(int n, int f)
{
    ch[n] ++;

    int to;
    for (int i = g[n];~i;i = e[i].nxt) {
        to = e[i].to;
        if (to == f) continue;
        dfs(to, n);
        ch[n] += ch[to];
    }

    for (int i = g[n];~i;i = e[i].nxt) {
        to = e[i].to;
        if (to == f) continue;
        a[all ++] = ch[to] * (nn - ch[to]);
    }

}

int main()
{
    int u,v;
    scanf("%d",&nn);
    init(nn + 5);
    rep(i,1,nn - 1) {
        scanf("%d%d",&u,&v);
        add_edge(u,v);
        add_edge(v,u);
    }
    mem(ch,0);
    dfs(1,-1);
    sort(a, a + all);
    nn --;
    ll sum = 0;
    int i = 0;
    while (nn) {
        sum += (nn --) * a[i ++];
    }
    printf("%lld\n",sum);
    return 0;
}
```

Net Flow

Edmond-Karp

```
/// 网络流-最大流模板 EK 算法 前向星
/// https://www.luogu.com.cn/problem/P3376
const int MAXN = 5000 + 10;

struct Edge {
    int to;
    ll w;
    int nxt;
} e[MAXN * 2];
int g[MAXN];
int cnt = 0;

void init(int n)
{
    cnt = 0;
    memn(g,-1,int,n);
}

void add_edge(int u,int v,ll w)
{
    e[cnt] = {v,w,g[u]};
    g[u] = cnt ++;
}


int pre[MAXN]; // record the previous node index and indicating a node
whether has been visited.
```

```
ll flow[MAXN];
int n,m;

ll getAgtPath(int s,int t) // method to find an augmented path
{
    // bfs
    queue<int> q;
    q.push(s);

    int current;
    memn(pre,-1,int,n + 5);

    flow[s] = INT_INF;
    pre[s] = 0; // sourse node has been visited

    int to;
    while (!q.empty()) {
        current = q.front();
        q.pop();

        if (current == t) break;

        for (int i = g[current];~i;i = e[i].nxt) {
            to = e[i].to;
            if (pre[to] != -1 || e[i].w <= 0) continue;

            pre[to] = i; // record the index of current edge

            flow[to] = min(flow[current],e[i].w);
            q.push(to);
        }
    }

    if (pre[t] != -1) return flow[t]; // return the terminal's flow
    return -1; // 404 not found
}

ll EKmaxFlow(int s,int t) // source -> terminal
{
    if (s == t) return INT_INF;


    ll flow = 0;
    ll agt_flow;
    int cur;
    int edge_idx;

    while ((agt_flow = getAgtPath(s, t)) != -1) { // find augmented pat
h until all flows are gone.

        // modify the w of path
        cur = t;
        while (cur != s) {
            edge_idx = pre[cur];

            e[edge_idx].w -= agt_flow;
            e[edge_idx ^ 1].w += agt_flow; // xor can get the reverse e
dge swiftly

            cur = e[edge_idx ^ 1].to;
        }

        flow += agt_flow; // add to ans
    }

    return flow;
}

int vis[210][210];

int main()
{
    int s,t;
    scanf("%d%d%d%d",&n,&m,&s,&t);
    init(n + 5);
    mem(vis,-1);
    int u,v,w;
    rep(i,1,m) {
        scanf("%d%d%d",&u,&v,&w);
        if (vis[u][v] == -1) { // record the vised edge
            add_edge(u, v, w);
            add_edge(v, u, 0); // add reversed edge, which w=0;
            vis[u][v] = cnt - 2;
        }
        else {
            e[vis[u][v]].w += w; // increase the weight
        }
    }
    printf("%lld\n",EKmaxFlow(s, t));
    return 0;
}
```

Dinic

```
/// 网络流-最大流模板 Dinic 算法 前向星
/// https://www.luogu.com.cn/problem/P3376
const int MAXN = 5010;
```

```cpp
struct Edge {
    int to;
    ll w;
    int nxt;
} e[MAXN * 2];
int g[MAXN]; // Please call init() to memset it to -1!
int cnt = 0;

void init(int n)
{
    cnt = 0;
    memn(g,-1,int,n);
}

void add_edge(int u,int v,ll w)
{
    e[cnt] = {v,w,g[u]};
    g[u] = cnt ++;
}

int s,t,n,m;
int dis[MAXN];
int cur[MAXN]; // 替代g数组，记住上次dfs最后跑到的地方，优化，减少dfs的跑的次数

int bfs()
{
    memn(dis,-1,int,n + 5);

    queue<int> q;
    q.push(s);
    dis[s] = 0;

    int to,current,k;

    while (!q.empty()) {
        current = q.front();
        q.pop();

        for (int i = g[current];~i;i = e[i].nxt) {
            to = e[i].to;
            k = dis[current] + 1;
            if (dis[to] == -1 && e[i].w > 0) { // 只有没有访问过的，且该通路可以走(w > 0)
                dis[to] = k;
                if (to == t) return 1;
                q.push(to);
            }
        }
    }

    return 0;
}

ll dfs(int node,ll flow)
{
    if (node == t) return flow;
    int to;
    ll d;
    for (int &i = cur[node];~i;i = e[i].nxt) { // 改变 i 的同时，cur[node]的值也会被改变
        to = e[i].to;
        if (dis[node] + 1 == dis[to] && e[i].w > 0) {
            d = dfs(to,min(e[i].w,flow));
            if (d > 0) {
                e[i].w -= d;
                e[i ^ 1].w += d;
                return d;
            }
        }
    }
    return 0;
}

ll dinic()
{
    ll ans = 0;
    ll d;
    while (bfs()) {
        rep(i,1,n) cur[i] = g[i];
        while ((d = dfs(s,INT_INF)))
            ans += d;
    }
    return ans;
}

int main()
{
    scanf("%d%d%d%d",&n,&m,&s,&t);
    int u,v;
    ll w;
    init(n + 5);
    while (m --) {
        scanf("%d%d%lld",&u,&v,&w);
        add_edge(u, v, w);
        add_edge(v, u, 0);
    }
    printf("%lld\n",dinic());
    return 0;
}
```

Manacher

最长回文串

```
string Manacher(string &s)
{
    //改造字符串
    int n = (int) s.size();
    string res = "$#";
    for (int i = 0;i < n;i ++)
    {
        res += s[i];
        res += "#";
    }

    //数组
    n = (int) res.size();
    vector<int> P(n,0);
    int mi = 0,right = 0; //mi 为最大回文串对应的中心点，right 为该回文串能达到的最右端的值
    int maxLen = 0,maxPoint = 0; //maxLen 为最大回文串的长度，maxPoint 为记录中心点
    for (int i = 1;i < n;i ++)
    {
        P[i] = right > i ? min(P[2 * mi - i],right - i) : 1; //关键句，文中对这句以详细讲解
        while (res[i + P[i]] == res[i - P[i]]) {
            P[i] ++;
        }
        if (right < i + P[i]) //超过之前的最右端，则改变中心点和对应的最右端
        {
            right = i + P[i];
            mi = i;
        }
        if (maxLen < P[i]) //更新最大回文串的长度，并记下此时的点
        {
            maxLen = P[i];
            maxPoint = i;
        }
    }
    return s.substr((maxPoint - maxLen) / 2,maxLen - 1);
}
```

回文串个数

```
ll Manacher_n(string &s)
{
    //改造字符串
    int n = (int) s.size();
    string res = "$#";
    for (int i = 0;i < n;i ++)
    {
        res += s[i];
        res += "#";
    }

    //数组
    n = (int) res.size();
    vector<ll> P(n,0);
    ll mi = 0,right = 0; //mi 为最大回文串对应的中心点，right 为该回文串能达
到的最右端的值
    ll maxLen = 0,maxPoint = 0; //maxLen 为最大回文串的长度，maxPoint 为记
录中心点

    ll ans = 0;
    for (ll i = 1;i < n;i ++)
    {
        P[i] = right > i ? min(P[2 * mi - i],right - i) : 1; //关键句，
文中对这句以详细讲解
        while (res[i + P[i]] == res[i - P[i]]) {
            P[i] ++;
        }
        if (right < i + P[i]) //超过之前的最右端，则改变中心点和对应的最右
端
        {
            right = i + P[i];
            mi = i;
        }
        if (maxLen < P[i]) //更新最大回文串的长度，并记下此时的点
        {
            maxLen = P[i];
            maxPoint = i;
        }
        ans += P[i] / 2;
    }
    return ans;
}
```

Min25

快速求$[1, 10^{10}]$的质数和

```
const int MAXN = 1000010;

namespace Min25 {
    int prime[MAXN], id1[MAXN], id2[MAXN], flag[MAXN], ncnt, m;

    ll g[MAXN], sum[MAXN], a[MAXN], T, n;

    inline int ID(ll x) {
        return x <= T ? id1[x] : id2[n / x];
    }

    inline ll calc(ll x) {
        return x * (x + 1) / 2 - 1;
    }

    inline ll f(ll x) {
        return x;
    }

    inline void init() {

        rep(i,0,MAXN - 1) {
            prime[i] = id1[i] = id2[i] = flag[i] = 0;
            g[i] = sum[i] = a[i] = 0;
        }

        ncnt = 0;
        m = 0;
        T = sqrt(n + 0.5);
        for (int i = 2; i <= T; i++) {
            if (!flag[i]) {
                prime[++ncnt] = i;
                sum[ncnt] = sum[ncnt - 1] + i;
            }
            for (int j = 1; j <= ncnt && i * prime[j] <= T; j++) {
                flag[i * prime[j]] = 1;
                if (i % prime[j] == 0) break;
            }
        }
        for (ll l = 1; l <= n; l = n / (n / l) + 1) {
            a[++m] = n / l;
            if (a[m] <= T) id1[a[m]] = m; else id2[n / a[m]] = m;
            g[m] = calc(a[m]);
        }
        for (int i = 1; i <= ncnt; i++)
            for (int j = 1; j <= m && (ll)prime[i] * prime[i] <= a[j];
j++)
                g[j] = g[j] - (ll)prime[i] * (g[ID(a[j] / prime[i])] -
sum[i - 1]);
    }

    inline ll solve(ll x) {
        if (x <= 1) return x;
        n = x;
        init();
        return g[ID(n)];
    }

}
```

rope

奇淫怪巧：rope是一种类似块状链表操作的东西，速度很快，底层是可持续化平衡树实现。

只有gnu编译器可以使用，clang不行。

头文件加入

```
#include <ext/rope>
using namespace __gnu_cxx;
```

操作

```
push_back(x); //在末尾添加x

insert(pos,x); //在pos插入x，自然支持整个char数组的一次插入

erase(pos,x); //从pos开始删除x个

copy(pos,len,x); //从pos开始到pos+len为止用x代替

replace(pos,x); //从pos开始换成x

substr(pos,x); //提取pos开始x个

at(x); //访问第x个元素

[x] //访问第x个元素
```