心态不能崩!!! 心态不能崩!!! 心态不能崩!!!

• 提交之前看一下数据范围，测一下边界，拒绝罚时• 上取整以及 GCD 小心负数 • mid 用 l+(r-l)/2 可以避免溢出和负数的问题

QFNU- Kindergarten Bus

好的开始

```
#include<bits/stdc++.h>
using namespace std;
#define _for(i,a,b) for(int i=(a);i<(b);i++)
#define _ref(i,a,b) for(int i=(a);i<=(b);i++)
#define ms(a, b) memset(a, b, sizeof(a))
#define fill(a,v,n) memset((a),(v),sizeof(a[0])*(n))
#define copy(a,b,n) memcpy((a),(b),sizeof(a[0])*(n))
#define sc(x) scanf("%d", &x)
#define pr(x) printf("%d", (x))
#define scl(x) scanf("%lld", &x)
#define prl(x) printf("%lld", (x))
#define sp putchar(' ')
#define en putchar('\n')
#define ALL(x) (x.begin(),x.end())
#define pb push_back
#define mp make_pair
#define fi first
#define se second
#define IOS ios::sync_with_stdio(0);cin.tie(0);
typedef double db;
typedef long long ll;
typedef unsigned long long ull;
typedef pair<int, int> pii;
typedef pair<ll, ll> pll;

#define MOD 1000000007
void _max(int &a, int b){a = max(a,b);}
void _min(int &a, int b){a = min(a,b);}

ll pw(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 gcd(ll a, ll b){ll t;while(b){t=a%b;a=b;b=t;}return a;}
ll lcm(ll a, ll b){return a/gcd(a,b)*b;}
int len(ll x){int k=0;while(x){x/=10;k++;}return k;}
int len(int x){int k=0;while(x){x/=10;k++;}return k;}

const int infi = 2147483647;
const ll infl = 9223372036854775807ll;
const db PI = 3.14159265358979323846;

inline void rd(int &x){x = 0;char c = getchar();while(c<'0'||c>'9')c=getchar();while(c>='0'&&c<='9')x=x*10+(c&15),c=getchar();}
inline void llread(ll &x){char ch = getchar(); x = 0;for (; ch < '0' || ch > '9'; ch = getchar());for (; ch >= '0' && ch <= '9'; ch = getchar()) x = x * 10 + ch - '0';}

#define LOCAL_JUDGE

int main()
{
#ifdef LOCAL_JUDGE
    freopen("Text.txt", "r", stdin);
#endif // LOCAL_JUDGE

#ifdef LOCAL_JUDGE
    fclose(stdin);
#endif // LOCAL_JUDGE
    return 0;
}
```

舞蹈链

精确覆盖

```
const int maxnode = 1001000;
const int maxn = 1010;
struct DLX{
    int D[maxnode], U[maxnode], R[maxnode],L[maxnode];
    int H[maxn], col[maxnode],row[maxnode];
    int ans[maxn];
    int size;
    int n, m;
    int ansed;
    void init(int _n, int _m){
        n = _n;
        m = _m;
        for(int i = 0; i <= m; ++i){
            D[i] = U[i] = i;
            R[i] = i + 1;
            L[i] = i - 1;
            col[i] = i;
            row[i] = 0;
        }
        L[0] = m;
        R[m] = 0;
        memset(ans, 0, sizeof(ans));
        memset(H, -1, sizeof(H));
        size = m;
        ansed = 0;
    }
    void push(int r, int c){
        size ++;
        D[size] = D[c];
        U[size] = c;
        U[D[c]] = size;
        D[c] = size;
        col[size] = c;
        row[size] = r;
        if(H[r] < 0){
            H[r] = R[size] = L[size] = size;
        }
        else{
            L[size] = H[r];
            L[R[H[r]]] = size;
            R[size] = R[H[r]];
            R[H[r]] = size;
        }
    }
    void del(int c){
        L[R[c]] = L[c];
        R[L[c]] = R[c];
        for(int i = D[c]; i != c; i = D[i]){
            for(int j = R[i]; j != i; j = R[j]){
                D[U[j]] = D[j];
                U[D[j]] = U[j];
            }
        }
    }
    void reback(int c){
        for(int i = U[c]; i != c; i = U[i]){
            for(int j = L[i]; j != i; j = L[j]){
                D[U[j]] = j;
                U[D[j]] = j;
            }
        }
        L[R[c]] = c;
        R[L[c]] = c;
    }
    bool dancing(int dep){
        if(R[0] == 0){
            ansed = dep;
            return true;
        }
        int c = R[0];
        del(c);
        for(int i = D[c]; i != c; i = D[i]){
            ans[dep] = row[i];
            for(int j = R[i]; j != i; j = R[j])
                del(col[j]);
            if(dancing(dep+1)){
                return true;
            }
            for(int j = L[i]; j != i; j = L[j]){
                reback(col[j]);
            }
        }
        return false;
    }
}dlx;
int main(){
    int n,m;
    scanf("%d %d", &n,&m);
    dlx.init(n,m);
    for(int i = 1; i <= n; ++i){
        int k;
        int c;
        scanf("%d", &k);
        for(int j = 0; j < k; ++j){
            scanf("%d",&c);
            dlx.push(i, c);
        }
    }
    if(dlx.dancing(0)){
        printf("%d", dlx.ansed);
        sort(dlx.ans,dlx.ans+dlx.ansed);
        for(int i = 0; i < dlx.ansed; ++i){
            printf(" %d", dlx.ans[i]);
        }
    }
    else
        printf("NO");
    return 0;
}
```

重复覆盖

```
const int NUM = 100 * 60;
struct SLX {
    int U[NUM], D[NUM], L[NUM], R[NUM];//上下左右的邻居
    int Col[NUM], Row[NUM];//每个元素对应的列行
    int col[NUM], row[NUM];//每一列的元素个数，行的行首
    int si;
    int ans[1000], ansd;
    void ini(int n, int m) {
        for (int i = 0; i <= m; i++) {
            R[i] = i + 1;
            L[i] = i - 1;
            D[i] = U[i] = i;
            Col[i] = i;
            Row[i] = 0;
            col[i] = 0;
        }
        R[m] = 0;
        L[0] = m;
        si = m;
        for (int i = 1; i <= n; i++) {
            row[i] = -1;
        }
    }
    void add(int r, int c) {
        si++;
        Col[si] = c;
        Row[si] = r;
        col[c]++;
        D[si] = D[c];
        U[D[c]] = si;
        U[si] = c;
        D[c] = si;
        if (row[r] < 0) {
            row[r] = R[si] = L[si] = si;
        } else {
            R[si] = R[row[r]];
            L[R[row[r]]] = si;
            L[si] = row[r];
            R[row[r]] = si;
        }
    }
    void remove(int c) {
        for (int i = D[c]; i != c; i = D[i]) {
            R[L[i]] = R[i];
            L[R[i]] = L[i];
        }
    }
    void resume(int c) {
        for (int i = D[c]; i != c; i = D[i]) {
            R[L[i]] = L[R[i]] = i;
        }
    }
    int geth() { //最少还要处理几列
        int ret = 0;
        bool vis[80];
        memset(vis, 0, sizeof(vis));
        for (int c = R[0]; c; c = R[c]) {
            if (!vis[c]) {
                ret++;
                for (int i = D[c]; i != c; i = D[i]) {
                    for (int j = R[i]; j != i; j = R[j]) {
                        vis[Col[j]] = true;
                    }
                }
            }
        }
        return ret;
    }
```

• 小心模板自带的意料之外的隐式类型转换 • 求最优解时不要忘记更新当前最优解 • 图论问题一定要注意图不连通、重边、无环问题

读完题!!! 读完题!!! 读完题!!!

```
        void dance(int k) {
            if (!R[0]) {
                ansd = min(ansd, k);
                return;
            } else if(k+geth()>=ansd)
                return;
            int c = R[0];
            for (int i = R[0]; i; i = R[i])
                if (col[i] < col[c])
                    c = i;
            for (int i = D[c]; i != c; i = D[i]) {
                remove(i);
                for (int j = R[i]; j != i; j = R[j]) {
                    remove(j);
                    col[Col[j]]--;
                }
                dance(k + 1);
                for (int j = L[i]; j != i; j = L[j])
                    resume(j),col[Col[j]]++;
                resume(i);
            }
        }
    } dlx;
```

单点替换、单点增减、区间求和、区间最值

```
#define lson l , m , rt << 1
#define rson m + 1 , r , rt << 1 | 1
const int maxn = 222222;

int MAX[maxn<<2];
int MIN[maxn<<2];
int SUM[maxn<<2];
int max(int a,int b) {
    if(a>b)return a;
    else return b;
}
int min(int a,int b) {
    if(a<b)return a;
    else return b;
}
void PushUP(int rt) {
    MAX[rt] = max(MAX[rt<<1], MAX[rt<<1|1]);
    MIN[rt] = min(MIN[rt<<1], MIN[rt<<1|1]);
    SUM[rt] = SUM[rt<<1] + SUM[rt<<1|1];
}
void build(int l,int r,int rt) {
    if (l == r) {
        scanf("%d",&MAX[rt]);
        MIN[rt] = MAX[rt];
        SUM[rt] = MAX[rt];
        return ;
    }
    int m = (l + r) >> 1;
    build(lson);
    build(rson);
    PushUP(rt);
}
void update(int p,int tihuan,int l,int r,int rt) {
    if (l == r) {
        MAX[rt] = tihuan;
        MIN[rt] = tihuan;
        SUM[rt] = tihuan;
        return ;
    }
    int m = (l + r) >> 1;
    if (p <= m) update(p, tihuan,lson);
    else update(p, tihuan, rson);
    PushUP(rt);
}
void update1(int p,int add,int l,int r,int rt) {
    if (l == r) {
        SUM[rt] = SUM[rt] + add;
        return ;
    }
    int m = (l + r) >> 1;
    if (p <= m) update1(p, add,lson);
    else update1(p, add, rson);
    PushUP(rt);
}
int query(int L,int R,int l,int r,int rt) {
    if (L <= l && r <= R) {
        return MAX[rt];
    }
    int m = (l + r) >> 1;
    int ret = -1;
    if (L <= m) ret = max(ret, query(L, R, lson));
    if (R > m)  ret =  max(ret, query(L, R, rson));
    return ret;
}
int query1(int L,int R,int l,int r,int rt) {
    if (L <= l && r <= R) {
        return MIN[rt];
    }
    int m = (l + r) >> 1;
    int ret = 99999;
    if (L <= m) ret = min(ret, query1(L, R, lson));
    if (R > m)  ret =  min(ret, query1(L, R, rson));
    return ret;
}
int queryhe(int L,int R,int l,int r,int rt) {
    if (L <= l && r <= R) {
        return SUM[rt];
    }
    int m = (l + r) >> 1;
    int ret = 0;
    if (L <= m) ret += queryhe(L, R, lson);
    if (R > m)  ret +=  queryhe(L, R, rson);
    return ret;
}
int main() {
    int n, m;
    while (~scanf("%d%d",&n,&m)) {
        build(1, n, 1);
        while (m --) {
            char op[2];
            int a, b;
            scanf("%s%d%d",op,&a,&b);
            if (op[0] == 'Q') { //区间求最大
                printf("%d\n",query(a, b, 1, n, 1));
            }
            else if(op[0]=='U') //单点替换
                update(a, b, 1, n, 1);
            else if(op[0]=='M') { //区间求最小
                printf("%d\n",query1(a, b, 1, n, 1));
            }
            else if(op[0]=='H') { //区间求和
                printf("%d\n",queryhe(a, b, 1, n, 1));
            }
            else if(op[0]=='S') { //单点增加
                scanf("%d%d",&a,&b);
                update1(a, b, 1, n, 1);
            }
            else if(op[0]=='E') { //单点减少
                scanf("%d%d",&a,&b);
                update1(a, -b, 1, n, 1);
            }
        }
    }
    return 0;
}
```

区间替换

```
#define max(a,b) (a>b)?a:b
#define min(a,b) (a>b)?b:a
#define lson l , m , rt << 1
#define rson m + 1 , r , rt << 1 | 1
#define LL long long
const int maxn = 100100;

int lazy[maxn<<2];
int sum[maxn<<2];

void PushUp(int rt) { //由左孩子、右孩子向上更新父节点
    sum[rt] = sum[rt<<1] + sum[rt<<1|1];
}
void PushDown(int rt,int m) { //向下更新
    if (lazy[rt]) { //懒惰标记
        lazy[rt<<1] = lazy[rt<<1|1] = lazy[rt];
        sum[rt<<1] = (m - (m >> 1)) * lazy[rt];
        sum[rt<<1|1] = ((m >> 1)) * lazy[rt];
        lazy[rt] = 0;
    }
}
void build(int l,int r,int rt) { //建树
    lazy[rt] = 0;
    if (l== r) {
        scanf("%d",&sum[rt]);
        return ;
    }
    int m = (l + r) >> 1;
    build(lson);
    build(rson);
    PushUp(rt);
}
void update(int L,int R,int c,int l,int r,int rt) { //更新
    if (L <= l && r <= R) {
        lazy[rt] = c;
        sum[rt] = c * (r - l + 1);
        return ;
    }
    PushDown(rt, r - l + 1);
    int m = (l + r) >> 1;
    if (L <= m) update(L, R, c, lson);
    if (R > m) update(L, R, c, rson);
    PushUp(rt);
}
LL query(int L,int R,int l,int r,int rt) {
    if (L <= l && r <= R) {
        return sum[rt];
    }
    PushDown(rt, r - l + 1);
    int m = (l + r) >> 1;
    LL ret = 0;
    if (L <= m) ret += query(L, R, lson);
    if (m < R) ret += query(L, R, rson);
    return ret;
}
int main() {
    int  n, m;
    char str[5];
    while(scanf("%d%d",&n,&m)) {
        build(1, n, 1);
        while (m--) {
            scanf("%s",str);
            int a, b, c;
            if(str[0]=='T') {
                scanf("%d%d%d",&a,&b,&c);
                update(a, b, c, 1, n, 1);
            } else if(str[0]=='Q') {
                scanf("%d%d",&a,&b);
                cout<<query(a,b,1,n,1)<<endl;
            }
        }
    }
    return 0;
}
```

区间增减

```
#define max(a,b) (a>b)?a:b
#define min(a,b) (a>b)?b:a
#define lson l , m , rt << 1
#define rson m + 1 , r , rt << 1 | 1
#define LL  int64
const int maxn = 100100;
using namespace std;
LL lazy[maxn<<2];
LL sum[maxn<<2];
void putup(int rt) {
    sum[rt] = sum[rt<<1] + sum[rt<<1|1];
}
void putdown(int rt,int m) {
    if (lazy[rt]) {
        lazy[rt<<1] += lazy[rt];
        lazy[rt<<1|1] += lazy[rt];
        sum[rt<<1] += lazy[rt] * (m - (m >> 1));
        sum[rt<<1|1] += lazy[rt] * (m >> 1);
        lazy[rt] = 0;
    }
}
void build(int l,int r,int rt) {
    lazy[rt] = 0;
    if (l == r) {
        scanf("%I64d",&sum[rt]);
        return ;
    }
    int m = (l + r) >> 1;
    build(lson);
    build(rson);
    putup(rt);
}
```

```
    }
    void update(int L,int R,int c,int l,int r,int rt) {
        if (L <= l && r <= R) {
            lazy[rt] += c;
            sum[rt] += (LL)c * (r - l + 1);
            return ;
        }
        putdown(rt, r - l + 1);
        int m = (l + r) >> 1;
        if (L <= m) update(L, R, c, lson);
        if (m < R) update(L, R, c, rson);
        putup(rt);
    }
    LL query(int L,int R,int l,int r,int rt) {
        if (L <= l && r <= R) {
            return sum[rt];
        }
        putdown(rt, r - l + 1);
        int m = (l + r) >> 1;
        LL ret = 0;
        if (L <= m) ret += query(L, R, lson);
        if (m < R) ret += query(L, R, rson);
        return ret;
    }
    int main() {
        int n, m;
        int a, b, c;
        char str[5];
        scanf("%d%d",&n,&m);
        build(1, n, 1);
        while (m--) {
            scanf("%s",str);
            if (str[0] == 'Q') {
                scanf("%d%d",&a,&b);
                printf("%I64d\n",query(a, b, 1, n, 1));
            } else if(str[0]=='C') {
                scanf("%d%d%d",&a,&b,&c);
                update(a, b, c, 1, n, 1);
            }
        }
        return 0;
    }
```

米勒拉宾

```
typedef long long LL;
LL mulmod( LL a, LL b , LL p ){
    LL  d =1;
    a = a%p;
    while( b>0 ){
        if(b&1)     d = (d*a)%p;
        a = (a*a)%p;
        b>>=1;
    }
    return d;
}
bool witness( LL a,LL n){
    LL d = n-1 ;
    if( n ==2 ) return true ;
    if( !(n&1) ) return false ;
    while(!(d&1)) d = d/2;
    LL t = mulmod(a,d,n);
    while((d!=n-1) && (t!=1)&&(t!=n-1)){
        t = mulmod( t ,2,n);
        d=d<<1;
    }
    return (t==n-1)||(d&1);
}

bool isprime( LL n){
    int a[3] = {2,7,61};
    for(int i=0;i<3;i++)
        if(!witness(a[i],n))
            return false;
    return true;
}
int main(){
    LL s;
    cin>>s;
    if(isprime(s))
        cout<<"YES";
    else
        cout<<"NO";
    return 0;
}
```

并查集

```
int par[maxn];
int Find(int x){
    if(par[x] == x)
        return x;
    return par[x] = Find(par[x]);
}
void Union(int x, int y){
    par[Find(x)] = Find(y);
}
void init(){
    for(int i = 0; i <= n; ++i)
        par[i] = i;
}
```

带权并查集

```
int par[maxn];
ll d[maxn];
int ans;

int Find(int x){
    if(x != par[x]){
        int k = par[x];
        par[x] = Find(par[x]);
        d[x] += d[k];
    }
    return par[x];
}
void init(int n){
    memset(d,0,sizeof(d));
    for(int i = 0; i <= n; i++)
        par[i] = i;
}
void Union(int x, int y, int v){
    int xx = Find(x);
    int yy = Find(y);
    if(xx == yy)
        if(d[x] + v != d[y])
            ans++;
    else{
        par[yy] = xx;
        d[yy] = d[x] - d[y]  + v;
    }
}
int main(){
    int n, m;
    while(scanf("%d%d", &n, &m) != EOF){
        ans = 0;
        init(n);
        int x, y, v;
        for(int i = 0; i < m; i++){
            scanf("%d%d%d", &x, &y, &v);
            x -= 1;
            Union(x, y, v);
        }
        printf("%d\n", ans);
    }
    return 0;
}
```

最短路

Dijkstra

```
const int inf = 0x3f3f3f3f;
const int maxn = 1010;
int G[maxn][maxn];
int vst[maxn];
int d[maxn];
int m, n;
void Dijkstra(int s){
    for(int i = 1; i <= n; ++i){
        d[i] = inf;
        vst[i] = 0;
    }
    d[s] = 0;
    vst[s] = 1;
    for(int i = 0; i < n-1; ++i){
        int Min = inf;
        int k = s;
        for(int j = 1; j <= n; ++j){
            if(!vst[j] && Min > d[j]){
                Min = d[j];
                k = j;
            }
        }
        vst[k] = 1;
        for(int j = 1; j <= n; ++j){
            if(G[k][j])
                if(!vst[j] && d[j] > d[k]+G[k][j]){
                    d[j] = d[k] + G[k][j];
                }
        }
    }
}
int main(){
    while(~scanf("%d %d", &m, &n)){
        memset(G, 0, sizeof(G));
        for(int i = 0; i < m; ++i){
            int x, y, v;
            scanf("%d%d%d", &x, &y, &v);
            if( G[x][y] == 0 || (G[x][y] && v < G[x][y]) ){
                G[x][y] = v;
                G[y][x] = v;
            }
        }
        Dijkstra(1);
        printf("%d\n", d[n]);
    }
    return 0;
}
```

Spfa

```
const int maxn = 1010;
const int inf = 0x3f3f3f3f;
int n, m;
int G[maxn][maxn];
int d[maxn];
int vst[maxn];
void spfa(int s){
   queue<int> q;
   for(int i = 0; i <= n; ++i){
       d[i] = inf;
   }
   d[s] = 0;
   memset(vst, 0, sizeof(vst));
   vst[s] = 1;
   q.push(s);
   while(!q.empty()){
       int k = q.front();
       vst[k] = 0;
       q.pop();
       for(int j = 1; j <= n; ++j){
           if(G[k][j])
           if(d[j] > d[k]+G[k][j]){
               d[j] = d[k] + G[k][j];
               if(!vst[j]){
                   vst[j] = 1;
                   q.push(j);
               }
           }
       }
   }
}
int main(){
    while(~scanf("%d %d", &m, &n)){
        memset(G, 0, sizeof(G));
        for(int i = 0; i < m; ++i){
            int x, y, v;
            scanf("%d %d %d", &x, &y, &v);
            if(!G[x][y] || (G[x][y] && G[x][y] > v) ){
                G[x][y] = G[y][x] = v;
            }
        }
        spfa(1);
        printf("%d\n", d[n]);
    }
    return 0;
}
```

SPFA

```
const int maxn = 210;
const int inf = 0x3f3f3f3f;

int n,m,s;
double v;
pair<double,double> G[maxn][maxn];
int vst[maxn];
double d[maxn];
int cot[maxn];//检测环路

int spfa(int s){
```

```
        for(int i = 0; i <= n; ++i){
            vst[i] = 0;
            d[i] = 0;
            cot[i] = 0;
        }
        vst[s] = 1;
        cot[s] = 1;
        d[s] = v;
        queue<int> q;
        q.push(s);
        while(!q.empty()){
            int k = q.front();
            q.pop();
            vst[k] = 0;
            for(int j = 1; j <= n; ++j){
                if(d[j]<(d[k]-G[k][j].second)*G[k][j].first){
                    d[j]=(d[k]-G[k][j].second)*G[k][j].first;
                    if(!vst[j]){
                        vst[j] = 1;
                        q.push(j);
                        cot[j] ++;
                        if(cot[j] > n)
                            return true;
                    }
                }
            }
        }
        return false;
    }
    int main(){
        while(~scanf("%d%d%d%lf", &n,&m,&s,&v)){
            memset(G, 0, sizeof(G));
            int x, y;
            double p,q;
            for(int i = 0; i < m; ++i){
                scanf("%d%d",&x,&y);
                scanf("%lf%lf",&p,&q);
                G[x][y] = make_pair(p,q);
                scanf("%lf%lf",&p,&q);
                G[y][x] = make_pair(p,q);
            }
            if(spfa(s)){
                printf("YES\n");
            }
            else{
                printf("NO\n");
            }
        }
        return 0;
    }
```

SPFA 链式向前星

```
typedef long long ll;
const int maxn = 1e6+10;
const int inf = 0x3f3f3f3f;

struct edge{
    int to, w, next;
};

edge e[maxn];
int head[maxn];
int vst[maxn];
ll d[maxn];
int t, n, m;
int x[maxn], y[maxn], v[maxn];

void add(int i, int u, int v, int w){
    e[i].to = v;
    e[i].w = w;
    e[i].next = head[u];
    head[u] = i;
}
void spfa(int s){
    for(int i = 0; i <= n; ++i){
        vst[i] = 0;
        d[i] = inf;
    }
    vst[s] = 1;
    d[s] = 0;
    queue<int> q;
    q.push(s);
    while(!q.empty()){
        int k = q.front();
        q.pop();
        vst[k] = 0;
        for(int j = head[k]; j != -1; j = e[j].next){
            if(d[e[j].to] > d[k] + e[j].w){
                d[e[j].to] = d[k] + e[j].w;
                if(!vst[e[j].to]){
                    vst[e[j].to] = 1;
                    q.push(e[j].to);
                }
            }
        }
    }
}
int main(){
    scanf("%d", &t);
    while(t--){
        scanf("%d%d", &n, &m);
        memset(head, -1, sizeof(head));
        memset(e, 0, sizeof(e));
        for(int i = 0; i < m; ++i){
            scanf("%d%d%d", &x[i], &y[i], &v[i]);
            add(i, x[i], y[i], v[i]);
        }
        ll ans = 0;
        spfa(1);
        for(int i = 1; i <= n; ++i){
            ans += d[i];
        }
        memset(head, -1, sizeof(head));
        for(int i = 0; i < m; ++i){
            add(i, y[i], x[i], v[i]);
        }
        spfa(1);
        for(int i = 1; i <= n; ++i){
            ans += d[i];
        }
        printf("%lld\n", ans);
    }
    return 0;
}
```

二分法

```
const int maxn=1e5+10;
int n,m,arr[maxn];
bool check(int x){
    int sum=0,cot=1;
    for(int i=0; i<n; ++i){
        sum+=arr[i];
        if(sum>x){
            cot++;
            sum=arr[i];
        }
    }
    return (cot<=m)?1:0;
}
int main(){
    cin>>n>>m;
    int l,r=l=0;
    for(int i=0; i<n; ++i){
        cin>>arr[i];
        r+=arr[i];
        l=max(l,arr[i]);
    }
    int ans;
    while(l<=r){
        int mid=(l+r)>>1;
        if(check(mid))
        {
            ans=mid;
            r=mid-1;
        }
        else    l=mid+1;
    }
    cout<<ans;
}
```

二分查找

```
int Find(int a[],int l,int r,int key){
    int m;
    while(l<=r){
        m=(l+r)>>1;
        if(a[m]==key)   return m;
        else if(key>a[m])   l=m+1;
        else    r=m-1;
    }
    return -1;
}
```

最小生成树
Kruskcal

```
const int verNum = 30;
const int arcNum = 2400;

typedef struct node{
    int val, start, end;
    node(int s,int e,int v){
        start = s;
        end = e;
        val = v;
    }
    node(){
        ;
    }
}Node;

int n, m;
Node v[arcNum];
int cot;
int par[arcNum];
void init(){
    for(int i = 0; i <= m; ++i)
        par[i] = i;
}
bool cmp(Node a, Node b){
    if(a.val < b.val)
        return true;
    return false;
}
int Find(int x){
    if(par[x] == x)
        return x;
    return par[x] = Find(par[x]);
}
void Union(int x, int y){
    par[Find(x)] = Find(y);
}
int Kruskcal(int s){
    int sum = 0;
    for(int i = 0; i < m; ++i){
        if(Find(v[i].start) != Find(v[i].end)){
            Union(v[i].start, v[i].end);
            sum += v[i].val;
        }
    }
    return sum;
}
int main(){
    while(~scanf("%d", &n) && n){
        m = 0;
        cot = 0;
        for(int j = 0; j < n-1; ++j){
            char ch[2];
            int pre, k, p, cost;
            scanf("%s %d",ch, &k);
            pre = ch[0] - 'A';
            m += k*2;
            for(int i = 0; i < k; ++i){
                scanf("%s %d",ch, &cost);
                p = ch[0] - 'A';
                v[cot] = node(pre,p,cost);
                cot ++;
                v[cot] = node(p, pre, cost);
                cot ++;
            }
        }
        init();
        sort(v, v+m, cmp);
        int ans = Kruskcal(0);
        printf("%d\n",ans);
    }
}
```

Prim

```
const int verNum = 30;
```

BFS

```
const int arcNum = 80;
const int inf = 0x3f3f3f3f;

int n;
int G[verNum][verNum];
int vst[verNum];
int d[verNum];
int sum;
void Prim(int s){
    memset(vst, 0, sizeof(vst));
    vst[s] = 1;
    for(int i = 0; i < n; ++i)
        d[i] = G[s][i];
    for(int i = 0; i < n-1; ++i){
        int Min = inf;
        int k = s;
        for(int j = 0; j < n; ++j){
            if(!vst[j] && Min > d[j]){
                Min = d[j];
                k = j;
            }
        }
        vst[k] = 1;
        sum += Min;
        for(int j = 0; j < n; ++j){
            if( d[j] > G[k][j])
                d[j] = G[k][j];
        }
    }
}
int main(){
    while(~scanf("%d", &n) && n){
        sum = 0;
        for(int i = 0; i < n; ++i){
            for(int j = 0; j < n; ++j)
                G[i][j] = inf;
        }
        for(int i = 0; i < n-1; ++i){
            char ch[2];
            int pre, p, cost;
            int k;
            scanf("%s %d", ch, &k);
            pre = ch[0] - 'A';
            for(int j = 0; j < k; ++j){
                scanf("%s %d", ch, &cost);
                p = ch[0] - 'A';
                G[pre][p] = G[p][pre] = cost;
            }
        }
        Prim(0);
        printf("%d\n", sum);
    }
    return 0;
}
```

```
const int maxn = 1e5+10;
int m, n;
// 1033 -8179 素数替换
bool prime[maxn];
void init(){
    memset(prime, 1, sizeof(prime));
    prime[1] = 0;
    for(int i = 2; i < maxn; ++i){
        for(int j = i*2; j < maxn; j += i)
            prime[j] = 0;
    }
}
int pre[maxn];
bool vis[maxn];
int d[maxn];
void bfs(){
    queue<int> q;
    q.push(n);
    vis[n] = 1;
    int p;
    while(!q.empty()){
        p = q.front();
        int t = p;
        q.pop();
        int a[4];
        if(p == m)
            return ;
        for(int i = 0; i < 4; ++i){
            a[i] = p%10;
            p /= 10;
        }
        for(int i = 0; i < 10; ++i){
            for(int k = 0; k < 4; ++k){
                int tmp = t - (a[k] - i)* pow(10,k);
                if( tmp < 10000 && tmp >= 1000 && i != a[k] && !vis[tmp] && prime[tmp]){
                    q.push(tmp);
                    d[tmp] = d[t] + 1;
                    vis[tmp] = 1;
//                  pre[tmp] = t;
                }
            }
        }
    }
}
//void Print(int x){
//    stack<int> s;
//    while(pre[x]){
//        s.push(x);
//        x = pre[x];
//    }
//    while(!s.empty()){
//        printf("%d  ", s.top());
//        s.pop();
//    }
//}
int main(){
    int t;
    init();
    scanf("%d", &t);
    while(t--){
        scanf("%d %d", &n, &m);
        memset(d, 0, sizeof(d));
        memset(vis, 0, sizeof(vis));
        bfs();
        cout<<d[m]<<endl;
//      Print(m);
    }
    return 0;
}
```

DFS

```
int m,n;
int b_x,b_y,e_x,e_y;
int row[100],col[100];
int k=1;
int step[][2]={{0,-1},{-1,0},{0,1},{1,0}};
int flag[100][100];
int mask=0;
int mp[100][100];
int OutBorder(int x,int y){
    if(x<1||x>m||y<1||y>n)  return 1;
    return 0;
}
void dfs(int x,int y){
    if(x==e_x && y==e_y){
        mask=1;
        for(int i=0;i<k;i++)
            printf(i==0?"(%d,%d)":"->(%d,%d)",row[i],col[i]);
        printf("\n");
        return ;
    }
    for(int i=0;i<4;i++){
        int cur_x=x+step[i][0];
        int cur_y=y+step[i][1];
        if( !OutBorder(cur_x,cur_y) && !flag[cur_x][cur_y] && mp[cur_x][cur_y] ){
            row[k]=cur_x;
            col[k]=cur_y;
            k++;
            flag[cur_x][cur_y]=1;
            dfs(cur_x,cur_y);
            flag[cur_x][cur_y]=0;
            k--;
        }
    }
}
int main()
{
    cin>>m>>n;
    for(int i=1;i<=m;i++){
        for(int j=1;j<=n;j++)
            scanf(" %d",&mp[i][j]);
    }
    memset(flag,0,sizeof(flag));
    cin>>b_x>>b_y;
    cin>>e_x>>e_y;
    flag[1][1]=1;
    row[0]=b_x;
    col[0]=b_y;
    dfs(b_x,b_y);
    if(mask==0)
        printf("-1\n");
}
```

树状数组

```
typedef long long ll;
const int maxn=1e6+5,N=1e5+5;
int Tree[maxn],a[N],n;
ll ans,s[N];
inline int lowbit(int x){
    return (x&-x);
}
void add(int x,int val){
    for(int i=x;i<=maxn;i+=lowbit(i))
        Tree[i]+=val;
}
int get(int x){
    int sum=0;
    for(int i=x;i;i-=lowbit(i))
        sum+=Tree[i];
    return sum;
}
int main(){
    for(int i = 1; i < N; ++i)
        s[i]=s[i-1]+i;
    scanf("%d",&n);
    for(int i = 1; i <= n; ++i){//下标不能取 0
        scanf("%d",&a[i]);
        add(i,a[i]);
    }
    for(int i = 1; i <= n; ++i)
        printf("%4d", get(i));
}
```

String KMP

```
int* getNextArray(string s){
    int *next=new int[s.length()];
    next[0] = -1;
    if(s.length()==1)
        return next;
    next[1]=0;
    int i=2,cn=0;
    while(i<s.length()){
        if(s[i-1]==s[cn])//cn 往前跳的值
            next[i++] = ++cn;
        else if(cn>0)//cn 继续往前跳
            cn = next[cn];
        else//cn 跳到头
            next[i++] = 0;
    }
    return next;
}
int kmp(string s,string m){
    if(s.length()<1||m.length()<1)
        return -1;
    int i=0,j=0;
    int *next=getNextArray(m);
    while(i<s.length() && j<m.length()){
        if(s[i]==m[j])
            i++,j++;
        else if(next[j]==-1)//开头配不上
            i++;
        else
            j=next[j];
    }
    return j==m.length()?i-j:-1;
}
int main(){
    string s,m;
    cin>>s>>m;
    cout<<kmp(s,m)<<endl;
}
```

Manacher

```
char *manacherString(string s){
    char *res = new char[s.length() * 2 + 1];
    int index = 0;
    for (int i = 0; i < s.length() * 2 + 1;i++)
        res[i] = (i & 1) ? s[index++] : '#';
    return res;
}
int manacher(string s){
    if (s.length() == 0)
        return 0;
    char *charArr = manacherString(s);
    int charArrLen = s.length() * 2 + 1;
    int *pArr = new int[charArrLen];
    int C = -1, R = -1;    //C 为中心  R 右为边界
    int Max = INT_MIN;
    for (int i = 0; i < charArrLen ; i++){
        pArr[i] = R > i ? min(pArr[2 * C - i] , R-
i) : 1;     //2、3 种情况  1、4(边界)种情况
        while (i + pArr[i] < charArrLen && i - pArr[i] > -1){   //
判定是否越界
            if (charArr[i + pArr[i]] == charArr[i - pArr[i]])  //
暴力扩
                pArr[i]++;
            else
                break;
        }
        if (i + pArr[i] > R){    //更新右边界
            R = i + pArr[i];
            C = i;
        }
        Max = max(Max , pArr[i]);
    }
    return Max - 1;//因增加#原因
}
int main(){
    string s;
    cin>>s;
    cout<<manacher(s)<<endl;
}
```

AC 自动机

```
// 求目标串中出现了几个模式串
struct Trie{
    int next[500010][26], fail[500010], end[500010];
    int root, L;
    int newnode(){
        for (int i = 0; i < 26; i++){
            next[L][i] = -1;
        }
        end[L++] = 0;
        return L - 1;
    }
    void init(){
        L = 0;
        root = newnode();
    }
    void insert(char buf[]){
        int len = (int)strlen(buf);
        int now = root;
        for (int i = 0; i < len; i++){
            if (next[now][buf[i] - 'a'] == -1){
                next[now][buf[i] - 'a'] = newnode();
            }
            now = next[now][buf[i] - 'a'];
        }
        end[now]++;
    }
    void build(){
        queue<int>Q;
        fail[root] = root;
        for (int i = 0; i < 26; i++){
            if (next[root][i] == -1){
                next[root][i] = root;
            }
            else{
                fail[next[root][i]] = root;
                Q.push(next[root][i]);
            }
        }
        while (!Q.empty()){
            int now = Q.front();
            Q.pop();
            for (int i = 0;i < 26;i++){
                if (next[now][i] == -1){
                    next[now][i] = next[fail[now]][i];
                }
                else{
                    fail[next[now][i]]=next[fail[now]][i];
                    Q.push(next[now][i]);
                }
            }
        }
    }
    int query(char buf[]){
        int len = (int)strlen(buf);
        int now = root;
        int res = 0;
        for (int i = 0; i < len; i++){
            now = next[now][buf[i] - 'a'];
            int temp = now;
            while (temp != root){
                res += end[temp];
                end[temp] = 0;
                temp = fail[temp];
            }
        }
        return res;
    }
};
char buf[1000010];
Trie ac;
int main(){
    int T;
    int n;
    scanf("%d", &T);
    while(T--){
        scanf("%d", &n);
        ac.init();
        for (int i = 0; i < n; i++){
            scanf("%s", buf);
            ac.insert(buf);
        }
        ac.build();
        scanf("%s", buf);
        printf("%d\n", ac.query(buf));
    }
    return 0;
}
```

主席树

```
typedef long long LL;
#define lson l, m
#define rson m+1, r
const int N=60005;
int a[N], Hash[N];
int T[N], L[N<<5], R[N<<5], sum[N<<5];
int S[N];
int n, m, tot;
struct node{
    int l, r, k;
    bool Q;
}op[10005];

int build(int l, int r){
    int rt=(++tot);
    sum[rt]=0;
    if(l!=r){
        int m=(l+r)>>1;
        L[rt]=build(lson);
        R[rt]=build(rson);
    }
    return rt;
}

int update(int pre, int l, int r, int x, int val){
    int rt=(++tot);
    L[rt]=L[pre], R[rt]=R[pre], sum[rt]=sum[pre]+val;
    if(l<r){
        int m=(l+r)>>1;
        if(x<=m)
            L[rt]=update(L[pre], lson, x, val);
        else
            R[rt]=update(R[pre], rson, x, val);
    }
    return rt;
}
int lowbit(int x){
    return x&(-x);
}
int use[N];
void add(int x, int pos, int val){
    while(x<=n){
        S[x]=update(S[x], 1, m, pos, val);
        x+=lowbit(x);
    }
}

int Sum(int x){
    int ret=0;
    while(x>0){
        ret+=sum[L[use[x]]];
        x-=lowbit(x);
    }
    return ret;
}
int query(int u, int v, int lr, int rr, int l, int r, int k){
    if(l>=r)
        return l;
    int m=(l+r)>>1;
    int tmp=Sum(v)-Sum(u)+sum[L[rr]]-sum[L[lr]];
    if(tmp>=k){
        for(int i=u;i;i-=lowbit(i))
            use[i]=L[use[i]];
        for(int i=v;i;i-=lowbit(i))
            use[i]=L[use[i]];
        return query(u, v, L[lr], L[rr], lson, k);
    }
    else{
        for(int i=u;i;i-=lowbit(i))
            use[i]=R[use[i]];
        for(int i=v;i;i-=lowbit(i))
            use[i]=R[use[i]];
        return query(u, v, R[lr], R[rr], rson, k-tmp);
    }
}
void modify(int x, int p, int d){
    while(x<=n){
        S[x]=update(S[x], 1, m, p, d);
        x+=lowbit(x);
    }
}
int main(){
    int t;
    scanf("%d", &t);
    while(t--){
        int q;//n 个数 q 次查询
        scanf("%d%d", &n, &q);
        tot=0;
        m=0;
        for(int i=1;i<=n;i++){
            scanf("%d", &a[i]);
            Hash[++m]=a[i];
        }
        for(int i=0;i<q;i++){
            char s[10];
            scanf("%s", s);
            if(s[0]=='Q'){//查询[l,r]第 k 小的数
                scanf("%d%d%d", &op[i].l, &op[i].r, &op[i].k);
                op[i].Q=1;
            }
            else{//更改第 i 个数的值为 r
                scanf("%d%d", &op[i].l, &op[i].r);
                op[i].Q=0;
                Hash[++m]=op[i].r;
            }
        }
        sort(Hash+1, Hash+1+m);
        int mm=unique(Hash+1, Hash+1+m)-Hash-1;
```

```
            m=mm;
            T[0]=build(1, m);
            for(int i=1;i<=n;i++)
                T[i]=update(T[i-1], 1, m, lower_bound(Hash+1, Hash+1+m, a[i])-Hash, 1);
            for(int i=1;i<=n;i++)
                S[i]=T[0];
            for(int i=0;i<q;i++){
                if(op[i].Q){
                    for(int j=op[i].l-1;j;j-=lowbit(j))
                        use[j]=S[j];
                    for(int j=op[i].r;j;j-=lowbit(j))
                        use[j]=S[j];
                    printf("%d\n", Hash[query(op[i].l-1, op[i].r, T[op[i].l-1], T[op[i].r], 1, m, op[i].k)]);
                }
                else{
                    modify(op[i].l, lower_bound(Hash+1, Hash+1+m, a[op[i].l])-Hash, -1);
                    modify(op[i].l, lower_bound(Hash+1, Hash+1+m, op[i].r)-Hash, 1);
                    a[op[i].l]=op[i].r;
                }
            }
        }
        return 0;
    }
```

ST 表

```
//预处理复杂度同为O(nlogn),查询时间上，ST 表为O(1),线段树为O(nlogn)
const int MAXN = 1e5 + 10;
int Max[MAXN][21];
int Min[MAXN][21];
int Query(int l,int r) {
    int k=log2(r-l+1);
    return max(Max[l][k],Max[r-(1<<k)+1][k]);//把拆出来的区间分别取最值
}
int lQuery(int l,int r) {
    int k=log2(r-l+1);
    return min(Min[l][k],Min[r-(1<<k)+1][k]);//把拆出来的区间分别取最值
}
int main() {
    int n, m;
    scanf("%d %d", &n, &m);
    for(int i = 1; i <= n; i++) {
        scanf("%d", &Max[i][0]);
        Min[i][0] = Max[i][0];
    }
    for(int j = 1; (1 << j) <= n; j++)
        for(int i = 1; i + (1 << j) - 1 <= n; i++) {
            Max[i][j] = max(Max[i][j-1], Max[i + (1 << (j - 1))][j - 1]);
            Min[i][j] = min(Min[i][j-1], Min[i + (1 << (j - 1))][j - 1]);
        }
    for(int i = 0; i < m; i++) {
        int l, r;
        scanf("%d %d", &l, &r);
        int maxItem = Query(l,r);
        int minItem = lQuery(l,r);
        printf("maxItem:");
        printf("%4d\n", maxItem);
        printf("minItem");
        printf("%4d\n", minItem);
    }
}
```

位运算

```
void swap(int& x , int& y){
    x ^= y;
    y ^= x;
    x ^= y;
}
```

矩阵快速幂

```
int n,Mod;
const int maxn=5, maxm=5;
struct Matrix{
    int n,m;
    int a[maxn][maxm];
    void clear(){
        n=m=0;
        memset(a,0,sizeof(a));
    }
    Matrix operator *(const Matrix &b) const{
        Matrix tmp;
        tmp.clear();
        tmp.n=n;tmp.m=b.m;
        for(int i=0;i<n;i++)
            for(int j=0;j<b.m;j++)
                for(int k=0;k<m;k++)
                    tmp.a[i][j]+=a[i][k]*b.a[k][j];
        return tmp;
    }
};
// [0, 1, 0, 0]    f[n-4]  f[n-3]
// [0, 0, 1, 0]    f[n-3]  f[n-2]
// [0, 0, 0, 1]    f[n-2]  f[n-1]
// [-1,1, 5, 1]    f[n-1]  f[n]
void Pow(int m){
    Matrix s;
    s.clear();
    s.n=s.m=4;
    s.a[3][3]=1;s.a[3][2]=5;
    s.a[3][1]=1;s.a[3][0]=-1;
    s.a[1][2]=1;s.a[2][3]=1;
    s.a[0][1]=1;
    Matrix ans;
    ans.clear();
    ans.n=4;
    ans.m=1;
    ans.a[0][0]=1;
    ans.a[1][0]=5;
    ans.a[2][0]=11;
    ans.a[3][0]=36;
    while(m){
        if(m&1)
            ans=s*ans;
        s=s*s;
        m>>=1;
    }
    cout<<ans.a[3][0]<<endl;
}
int main(){
    while(cin>>n>>Mod){
        if(!n)  return 0;
        if(n<4){
            switch(n){
            case 1:
                cout<<1%Mod<<endl;
                break;
            case 2:
                cout<<5%Mod<<endl;
                break;
            case 3:
                cout<<11%Mod<<endl;
                break;
            }
            continue;
        }
        Pow(n-4);
    }
    return 0;
}
```

快速幂

```
long ksm(long x, long n, long MOD){
    long ans = 1;
    while (n > 0){
        if ((n & 1) != 0)
            ans = ans * x % MOD;
        x = x * x % MOD;
        n >>= 1;
    }
    return ans;
}
```

欧拉函数 O(sqrt(n))

```
long euler(long n) {
    long res = n, a = n;
    for (int i = 2; i * i <= n; ++i) {
        if (a % i == 0) {
            res = res / i * (i - 1);
            while (a % i == 0)
                a /= i;
        }
    }
    if (a > 1)
        res = res / a * (a - 1);
    return res;
}
```

欧拉降幂公式(A^x)%p = (A^(x%phi(p)+phi))%p

```
long eulerDP(long A, BigInteger x, long p) {
    long phi = euler(p);
    long n = x.mod(BigInteger.valueOf(phi)).add(BigInteger.valueOf(phi)).longValue();
    return ksm(A, n, p);
}
```

递推求 1-n 欧拉函数 O(nlogn)

```
final static int maxn = 1000001;
static long[] euler = new long[maxn];
void Init() {
    euler[1] = 1;
    for (int i = 2; i < maxn; ++i)
        euler[i] = i;
    for (int i = 2; i < maxn; ++i)
        if (euler[i] == i)
            for (int j = i; j < maxn; j += i)
                euler[j] = euler[j] / i * (i - 1);
}
```

大整数快速幂

```
const int mod=1e9+7;
long long quick_mod(long long a,long long b){
    long long ans=1;
    while(b){
        if(b&1){
            ans=(ans*a)%mod;
            b--;
        }
        b/=2;
        a=a*a%mod;
    }
    return ans;
}//内部也用快速幂
long long quickmod(long long a,char *b,int len){
    long long ans=1;
    while(len>0){
```

```
            if(b[len-1]!='0'){
                int s=b[len-1]-'0';
                ans=ans*quick_mod(a,s)%mod;
            }
            a=quick_mod(a,10)%mod;
            len--;
        }
        return ans;
    }
    int main(){
        char s[100050];
        int a;          //求 a^s%mod
        while(~scanf("%d",&a)){
            scanf("%s",s);
            int len=strlen(s);
            printf("%I64d\n",quickmod(a,s,len));
        }
        return 0;
    }
```

Gxgcd

```
    //x，y 初始为任意值，最后变为一组特解
    int exgcd(int a, int b, int &x, int &y) {
    //对应最终情况，a=gcd(a,b),b=0,此时 x=1，y 为任意数
        if(b == 0) {
          x = 1;
          y = 0;
          return a;
        }
        //先递归到最终情况，再反推出初始情况
        int r = exgcd(b, a % b, x, y);
        int t = x; x = y; y = t - a / b * y;
        return r;       //gcd(a,b)
    }
```

扩展欧几里得求逆元

```
    int inv1(int a,int mod){
        int x,y;
        int d=exgcd(a,mod,x,y);
        if(d==1) return (x%mod+mod)%mod;
        return -1;
    }
```

费马小定理

```
    int inv2(int a,int mod){
        return ksm(a,mod-2,mod);
    }
```

线性递推求 1~mod 的所以逆元

```
    void inv3(int mod){
        inv[1]=1;
        for(int i=2;i<=mod-1;i++){
            inv[i]=(mod-mod/i)*inv[mod%i]%mod;
            cout<<inv[i]<<" ";
        }
    }
```

线性素数筛

```
    bool check[100005];
    int prime[100005];//储存第 i 个素数
    void getprime(int n){
        memset(check, 0, sizeof(check));
        int tot = 0;// tot 初始为 0，用来记录质数总个数
        for (int i = 2; i <= n; ++i){
            if (!check[i])
                prime[++tot] = i;
            for (int j = 1; j <= tot; ++j){
                if (i * prime[j] > n)
                    break;
                check[i * prime[j]] = 1;
                if (i % prime[j] == 0){
                    break;
                }
            }
        }
    }
```

逆序数

```
    const int N=100050;
    int n;
    long long c[N];//c[n]表示 a[1~n]的和，a 数组省略
    struct node{
        int val,pos;
    } a[100005];
    int lowbit(int x){//求 2^k
        return x & -x;
    }
    long long getsum(int n){//区间查询，求 a[1~n]的和
        long long res = 0;
        while(n>0){
            res+=c[n];
            n=n-lowbit(n);
        }
        return res;
    }
    int change(int x){//单点更新，将 c[x]的值加 1
        while(x<=n){
            c[x]++;
            x+=lowbit(x);
        }
    }
    bool cmp(node a,node b){//包含相同数
        if(a.val!=b.val)
            return a.val>b.val;
        return a.pos>b.pos;
    }
    int main(){
        while(cin>>n){
            memset(c,0,sizeof(c));
            for(int i=1; i<=n; i++){
                cin>>a[i].val;
                a[i].pos=i;
            }
            sort(a+1,a+n+1,cmp);
            long long cnt=0;
            for(int i=1; i<=n; i++){
                change(a[i].pos); //修改最大数位置的值
                cnt+=getsum(a[i].pos-1);
    //最大数位置之前的所有位置和，即区间求和，可知比当前数小的数有多少个
            }
            cout<<cnt<<endl;
        }
        return 0;
    }
```

求排列组合数

语法：result=P(long n,long m); / result=long C(long n,long m);
参数 m：排列组合的上系数 n：排列组合的下系数
返回值：排列组合数
注意：符合数学规则：m<=n

```
    long P(long n,long m){
        long p=1;
        while(m!=0){
            p*=n;
            n--;
            m--;
        }
        return p;
    }
    long C(long n,long m){
        long i,c=1;
        i=m;
        while(i!=0){
            c*=n;n--;i--;
        }
        while(m!=0){
            c/=m;m--;
        }
        return c;
    }
```

求某一天星期几
语法：result=weekday(int N,int M,int d)
参数：N,M,d：年月日，例如：2003,11,4
返回值：0：星期天，1 星期一……
注意：需要 math.h
适用于 1582 年 10 月 15 日之后，因为罗马教皇格里高利十三世在这一天启用新历法.

```
    int weekday(int N,int M,int d){
        int m,n,c,y,w;
        m=(M-2)%12;
        if (M>=3) n=N;else n=N-1;
        c=n/100;
        y=n%100;
        w=(int)(d+floor(13*m/5)+y+floor(y/4)+floor(c/4)-2*c)%7;
        while(w<0) w+=7;
        return w;
    }
```

任意进制转换
语法：conversion(char s1[],char s2[],char t[]);
参数：
s[]：转换前的数字 s2[]：转换后的数字 s2 高位位数越大
d1：原进制数 d2：需要转换到的进制数

```
    void conversion(char s[],char s2[],long d1,long d2){
        long i,j,t,num;
        char c;
        num=0;
        for (i=0; s[i]!='\0'; i++){
            if (s[i]<='9'&&s[i]>='0') t=s[i]-'0';
            else t=s[i]-'A'+10;
            num=num*d1+t;
        }
        i=0;
        while(1){
            t=num%d2;
            if (t<=9) s2[i]=t+'0';
            else s2[i]=t+'A'-10;
            num/=d2;
            if (num==0) break;
            i++;
        }
        for (j=0; j<i/2; j++){
            c=s2[j];
            s2[j]=s[i-j];
            s2[i-j]=c;
        }
        s2[i+1]='\0';
    }
```

Ronberg 算法计算积分
语法：result=integral(double a,double b);
参数：a：积分上限 b：积分下限
function f：积分函数
返回值：f 在（a,b）之间的积分值
注意：function f(x)需要自行修改，程序中用的是 sina(x)/x

需要 math.h 默认精度要求是 1e-5

```
double f(double x){
    return sin(x)/x; //在这里插入被积函数
}
double integral(double a,double b){
    double h=b-a;
    double t1=(1+f(b))*h/2.0;
    int k=1;
    double r1,r2,s1,s2,c1,c2,t2;
loop:
    double s=0.0;
    double x=a+h/2.0;
    while(x<b){
        s+=f(x);
        x+=h;
    }
    t2=(t1+h*s)/2.0;
    s2=t2+(t2-t1)/3.0;
    if(k==1){
        k++;
        h/=2.0;
        t1=t2;
        s1=s2;
        goto loop;
    }
    c2=s2+(s2-s1)/15.0;
    if(k==2){
        c1=c2;
        k++;
        h/=2.0;
        t1=t2;
        s1=s2;
        goto loop;
    }
    r2=c2+(c2-c1)/63.0;
    if(k==3){
        r1=r2;
        c1=c2;
        k++;
        h/=2.0;
        t1=t2;
        s1=s2;
        goto loop;
    }
    while(fabs(1-r1/r2)>1e-5){
        r1=r2;
        c1=c2;
        k++;
        h/=2.0;
        t1=t2;
        s1=s2;
        goto loop;
    }
    return r2;
}
```

行列式计算
语法：result=js(int s[][],int n)
参数：s[][]：行列式存储数组
n：行列式维数，递归用
返回值：行列式值
注意： 函数中常数 N 为行列式维度，需自行定

```
#define N 3
int js(int s[][N],int n){
    int z,j,k,r,total=0;
    int b[N][N];
    /*b[N][N]用于存放，在矩阵 s[N][N]中元素 s[0]的余子式*/
    if(n>2){
        for(z=0; z<n; z++){
            for(j=0; j<n-1; j++)
                for(k=0; k<n-1; k++)
                    if(k>=z) b[j][k]=s[j+1][k+1];
                    else
                        b[j][k]=s[j+1][k];
            if(z%2==0) r=s[0][z]*js(b,n-1); /*递归调用*/
            else r=(-1)*s[0][z]*js(b,n-1);
            total=total+r;
        }
    }
    else if(n==2)
        total=s[0][0]*s[1][1]-s[0][1]*s[1][0];
    return total;
}
```

杨辉三角
语法: void gen()
预置: const int Max;
注意: (Max 一般不能超过 22,否则要用高精度计算) int inta[Max][Max];
结果:inta 为杨辉三角序列.inta[1][1]=1;inta[2][1]=1;inta[2][2]=1…

```
const int Max = 22;
int inta[Max][Max];
void gen() {
    int i,j;
    for( i =1 ; i <= Max ; i++) {
        inta[i][1] = 1 ;
        inta[i][i] = 1 ;
    }
    inta[2][2] = 1   ;
    for( i = 3 ; i<=Max ; i++) {
        for(j =2 ; j<i ; j++ ) {
            inta[i][j] = inta[i-1][j-1] + inta[i-1][j] ;
        }
    }
}
```

字符串

数学
求三角形面积
语法：result=area3(float x1,float y1,float x2,float y2,float x3,float y3);
参数：x1～3：三角形 3 个顶点 x 坐标
y1～3：三角形 3 个顶点 y 坐标
返回值：三角形面积
注意： 需要 math.h

```
float area3(float x1,float y1,float x2,float y2,float x3,float y3)
 {
    float a,b,c,p,s;
    a=sqrt((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2));
    b=sqrt((x1-x3)*(x1-x3)+(y1-y3)*(y1-y3));
    c=sqrt((x3-x2)*(x3-x2)+(y3-y2)*(y3-y2));
    p=(a+b+c)/2;
    s=sqrt(p*(p-a)*(p-b)*(p-c));
    return s;
}
```

两矢量间角度
语法：result=angle(double x1, double y1, double x2, double y2);
参数：x/y1～2：两矢量的坐标
返回值：两的角度矢量
注意： 返回角度为弧度制，并且以逆时针方向为正方向 需要 math.h

```
#define PI 3.1415926
double angle(double x1, double y1, double x2, double y2) {
    double dtheta,theta1,theta2;
    theta1 = atan2(y1,x1);
    theta2 = atan2(y2,x2);
    dtheta = theta2 - theta1;
    while (dtheta > PI)
        dtheta -= PI*2;
    while (dtheta < -PI)
        dtheta += PI*2;
    return(dtheta);
}
```

语法：result=distance_2d(float x1,float x2,float y1,float y2);
参数：x/y/z1～2：各点的 x、y、z 坐标
返回值：两点之间的距离
注意： 需要 math.h

```
float distance_2d(float x1,float x2,float y1,float y2) {
    return(sqrt((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2)));
}

float distance_3d(float x1,float x2,float y1,float y2,float z1,float z2) {
    return(sqrt((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2)+(z1-z2)*(z1-z2)));
}
```

判断点是否在线段上
语法：result=Pointonline(Point p1,Point p2,Point p);
参数：p1、p2：线段的两个端点 p：被判断点
返回值：0：点在不在线段上；1：点在线段上
注意：若 p 线段端点上返回 1 需要 math.h

```
#define MIN(x,y) (x < y ? x : y)
#define MAX(x,y) (x > y ? x : y)
typedef struct {
    double x,y;
} Point;
int FC(double x1,double x2) {
    if (x1-x2<0.000002&&x1-x2>-0.000002) return 1;
    else return 0;
}
int Pointonline(Point p1,Point p2,Point p) {
    double x1,y1,x2,y2;
    x1=p.x-p1.x;
    x2=p2.x-p1.x;
    y1=p.y-p1.y;
    y2=p2.y-p1.y;
    if (FC(x1*y2-x2*y1,0)==0) return 0;
    if ((MIN(p1.x,p2.x)<=p.x&&p.x<=MAX(p1.x,p2.x))&&
                (MIN(p1.y,p2.y)<=p.y&&p.y<=MAX(p1.y,p2.y)))
        return 1;
    else return 0;
}
```

判断两线段是否相交
语法：result=lineintersect(Point p1,Point p2,Point p3,Point p4);
参数：p1～4：两条线段的四个端点
返回值：0：两线段不相交；1：两线段相交；2 两线段首尾相接
注意： p1!=p2;p3!=p4;

```
#define MIN(x,y) (x < y ? x : y)
#define MAX(x,y) (x > y ? x : y)
typedef struct {
    double x,y;
} Point;
int lineintersect(Point p1,Point p2,Point p3,Point p4) {
    Point tp1,tp2,tp3;
    if((p1.x==p3.x&&p1.y==p3.y)||(p1.x==p4.x&&p1.y==p4.y)||(p2.x==p3.x&&p2.y==p3.y)||(p2.x==p4.x&&p2.y==p4.y))
        return 2;
    //快速排斥试验
    if((MIN(p1.x,p2.x)<=p3.x&&p3.x<=MAX(p1.x,p2.x)&&MIN(p1.y,p2.y)<=p3.y&&p3.y<=MAX(p1.y,p2.y))||
        (MIN(p1.x,p2.x)<=p4.x&&p4.x<=MAX(p1.x,p2.x)&&MIN(p1.y,p2.y)<=p4.y&&p4.y<=MAX(p1.y,p2.y)))
        ;
    else return 0;
    //跨立试验
    tp1.x=p1.x-p3.x;
    tp1.y=p1.y-p3.y;
    tp2.x=p4.x-p3.x;
    tp2.y=p4.y-p3.y;
    tp3.x=p2.x-p3.x;
    tp3.y=p2.y-p3.y;
```

```
        if ((tp1.x*tp2.y-tp1.y*tp2.x)*(tp2.x*tp3.y-tp2.y*tp3.x)>=0) return 1;
        else return 0;
    }
```

判断线段与直线是否相交
语法：result=lineintersect(Point p1,Point p2,Point p3,Point p4);
参数：p1、p2：线段的两个端点　p3、p4：直线上的两个点
返回值：0：线段直线不相交；1：线段和直线相交
注意：　如线段在直线上，返回 1

```
typedef struct {
    double x,y;
} Point;
int lineintersect(Point p1,Point p2,Point p3,Point p4) {
    Point tp1,tp2,tp3;
    tp1.x=p1.x-p3.x;
    tp1.y=p1.y-p3.y;
    tp2.x=p4.x-p3.x;
    tp2.y=p4.y-p3.y;
    tp3.x=p2.x-p3.x;
    tp3.y=p2.y-p3.y;
    if ((tp1.x*tp2.y-tp1.y*tp2.x)*(tp2.x*tp3.y-tp2.y*tp3.x)>=0) return 1;
    else return 0;
}
```

点到线段最短距离
语法：result=mindistance(Point p1,Point p2,Point q);
参数：p1、p2：线段的两个端点 q：判断点
返回值：点 q 到线段 p1p2 的距离
注意：　需要 math.h

```
#define MIN(x,y) (x < y ? x : y)
#define MAX(x,y) (x > y ? x : y)
typedef struct {
    double x,y;
} Point;
double mindistance(Point p1,Point p2,Point q) {
    int flag=1;
    double k;
    Point s;
    if (p1.x==p2.x) {
        s.x=p1.x;
        s.y=q.y;
        flag=0;
    }
    if (p1.y==p2.y) {
        s.x=q.x;
        s.y=p1.y;
        flag=0;
    }
    if (flag) {
        k=(p2.y-p1.y)/(p2.x-p1.x);
        s.x=(k*k*p1.x+k*(q.y-p1.y)+q.x)/(k*k+1);
        s.y=k*(s.x-p1.x)+p1.y;
    }
    if (MIN(p1.x,p2.x)<=s.x&&s.x<=MAX(p1.x,p2.x))
        return sqrt((q.x-s.x)*(q.x-s.x)+(q.y-s.y)*(q.y-s.y));
    else
        return
            MIN(sqrt((q.x-p1.x)*(q.x-p1.x)+(q.y-p1.y)*(q.y-p1.y)),sqrt((q.x-p2.x)*(q.x-p2.x)+(q.y-p2.y)*(q.y-p2.y)));
}
```

求两直线的交点
语法：result=linecorss (Point p1,Point p2,Point q);
参数：p1～p4：直线上不相同的两点　*p：通过指针返回结果
返回值：1：两直线相交；2：两直线平行
注意：　如需要判断两线段交点，检验 k 和对应 k1（注释中）的值是否在 0～1 之间，用在 0～1 之间的那个求交点

```
typedef struct {
    double x,y;
} Point;
int linecorss(Point p1,Point p2,Point p3,Point p4,Point *p) {
    double k;
    if ((p4.y-p3.y)*(p2.x-p1.x)-(p4.x-p3.x)*(p2.y-p1.y)==0) return 0;
    if ((p4.x-p3.x)*(p1.y-p3.y)-(p4.y-p3.y)*(p1.x-p3.x)==0&&
            (p2.x-p1.x)*(p1.y-p3.y)-(p2.y-p1.y)*(p1.x-p3.x)==0) return 0;
    k=((p4.x-p3.x)*(p1.y-p3.y)-(p4.y-p3.y)*(p1.x-p3.x))/((p4.y-p3.y)*(p2.x-p1.x)-(p4.x-p3.x)*(p2.y-p1.y));
    (*p).x=p1.x+k*(p2.x-p1.x);
    (*p).y=p1.y+k*(p2.y-p1.y);
    return 1;
}
```

求两条线段的交点
语法：Result=IntersectPoint (Point p1,Point p2,Point p3,Point p4,Point &p);
参数：P1～P4：两条线断 4 个端点 P：线段交点
返回值：如果两条线段平行无交点，返回 0，否则返回 1

```
struct Point {
    float x,y;
};
int IntersectPoint (Point p1,Point p2,Point p3,Point p4,Point &p)
{
    float a,b,c,d,e,f;
    a=p2.y-p1.y;
    b=p1.x-p2.x;
    c=p1.y*(p2.x-p1.x)+p1.x*(p2.y-p1.y);
    d=p4.y-p3.y;
    e=p3.x-p4.x;
    f=p3.y*(p4.x-p3.x)+p1.x*(p4.y-p3.y);
    if (a*e==b*d)
        return 0;
    else {
        p.x=(e*c-b*f)/(b*d-a*e);
        p.y=(d*c-a*f)/(a*e-b*d);
        return 1;
    }
}
```

数论
x 的二进制长度
语法：result=BitLength(int x);
参数：x：测长的 x
返回值：x 的二进制长度

```
int BitLength(int x) {
    int d = 0;
    while (x > 0) {
        x >>= 1;
        d++;
    }
    return d;
}
```

返回 x 的二进制表示中从低到高的第 i 位
语法：result=BitAt(int x, int i);
参数：x：十进制 x　　i：要求二进制的第 i 位
返回值：返回 x 的二进制表示中从低到高的第 i 位
注意：　最低位为第一位

```
int BitAt(int x, int i) {
    return ( x & (1 << (i-1)) );
}
```

取幂运算
语法：result=Modular_Expoent(int a,int b,int n);
参数：a、b、n：a^b mod n 的对应参数
返回值：a^b mod n 的值
注意：　需要 BitLength 和 BitAt

```
int Modular_Expoent(int a,int b,int n) {
    int i, y=1;
    for (i = BitLength(b); i > 0; i--) {
        y = (y*y)%n;
        if (BitAt(b,i) > 0)
            y = (y*a)%n;
    }
    return y;
}
```

求解模线性方程
语法：result = modular_equation(int a,int b,int n);
参数：a、b、n：ax=b (mod n) 的对应参数
返回值：方程的解

```
int ext_euclid(int a,int b,int &x,int &y) { //求 gcd(a,b)=ax+by
    int t,d;
    if (b==0) {
        x=1;
        y=0;
        return a;
    }
    d=ext_euclid(b,a %b,x,y);
    t=x;
    x=y;
    y=t-a/b*y;
    return d;
}

void modular_equation(int a,int b,int n) {
    int e,i,d;
    int x,y;
    d=ext_euclid(a,n,x,y);
    if (b%d>0)
        printf("No answer!\n");
    else {
        e=(x*(b/d))%n;
        for (i=0; i<d; i++)
            printf("The %dth answer is : %ld\n",i+1,(e+i*(n/d))%n)
    ;
    }
}
```

求解模线性方程组(中国余数定理)
语法：result=Modular_Expoent(int a,int b,int n);
参数：B[]、W[]：a=B[] (mod W[]) 的对应参数
返回值：a 的值
注意：　其中 W[],B[]已知，W[i]>0 且 W[i]与 W[j]互质，求 a

```
int ext_euclid(int a,int b,int &x,int &y) { //求 gcd(a,b)=ax+by
    int t,d;
    if (b==0) {
        x=1;
        y=0;
        return a;
    }
    d=ext_euclid(b,a %b,x,y);
    t=x;
    x=y;
    y=t-a/b*y;
    return d;
}
int China(int B[],int W[],int k) {
    int i;
    int d,x,y,a=0,m,n=1;
    for (i=0; i<k; i++)
        n*=W[i];
    for (i=0; i<k; i++) {
        m=n/W[i];
```

```
            d=ext_euclid(W[i],m,x,y);
            a=(a+y*m*B[i])%n;
        }
        if (a>0) return a;
        else return(a+n);
    }
```

筛法素数产生器
语法：result=prime(int a[],int n);
参数：a[]：用于返回素数的数组
　　n：产生 n 以内的素数，按升序放入 a[]中
返回值：n 以内素数的个数

```
int prime(int a[],int n) {
    int i,j,k,x,num,*b;
    n++;
    n/=2;
    b=new int[(n+1)*2];
    a[0]=2;
    a[1]=3;
    num=2;
    for(i=1; i<=2*n; i++)
        b[i]=0;
    for(i=3; i<=n; i+=3)
        for(j=0; j<2; j++) {
            x=2*(i+j)-1;
            while(b[x]==0) {
                a[num++]=x;
                for(k=x; k<=2*n; k+=x)
                    b[k]=1;
            }
        }
    return num;
}
```

求一个数每一位相加之和
语法：result=digadd(int n)
参数：n：待求数字
返回值：各数字之和

```
int digadd(int n) {
    int i=0,k=0;
    while(i=n%10,n/=10)  k+=i;
    return k+i;
}
```

质因数分解
语法：result=int reduce(int prime[],int pn,int n,int rest[])
参数：prime[]：素数表，至少需要达到 sqrt(n)
pn：素数表的元素个数
n：待分解的数
rest：分解结果，按照升序排列

```
int reduce(int prime[],int pn,int n,int rest[]) {
    int i,k=0;
    for(i=0; i<pn; i++) {
        if (n==1) break;
        if (prime[i]*prime[i]>n) {
            rest[k++]=n;
            break;
        }
        while(n%prime[i]==0) {
            n/=prime[i];
            rest[k++]=prime[i];
        }
    }
    return k;
}
```

高斯消元法解线性方程组

```
const int MAXN=50;
int a[MAXN][MAXN];//增广矩阵
int x[MAXN];//解集
bool free_x[MAXN];//标记是否是不确定的变元
inline int gcd(int a,int b) {
    int t;
    while(b!=0) {
        t=b;
        b=a%b;
        a=t;
    }
    return a;
}
inline int lcm(int a,int b) {
    return a/gcd(a,b)*b;//先除后乘防溢出
}
// 高斯消元法解方程组(Gauss-Jordan elimination).(-2 表示有浮点数解，但无整数解，
//-1 表示无解，0 表示唯一解，大于 0 表示无穷解，并返回自由变元的个数)
//有 equ 个方程，var 个变元。增广矩阵行数为 equ,分别为 0 到 equ-1,列数为 var+1,分别为 0 到 var.
int Gauss(int equ,int var) {
    int i,j,k;
    int max_r;// 当前这列绝对值最大的行.
    int col;//当前处理的列
    int ta,tb;
    int LCM;
    int temp;
    int free_x_num;
    int free_index;
    for(int i=0; i<=var; i++) {
        x[i]=0;
        free_x[i]=true;
    }
    //转换为阶梯阵.
    col=0; // 当前处理的列
    for(k = 0; k < equ && col < var; k++,col++) {
        // 枚举当前处理的行.
        // 找到该 col 列元素绝对值最大的那行与第 k 行交换.(为了在除法时减小误差)
        max_r=k;
        for(i=k+1; i<equ; i++) {
            if(abs(a[i][col])>abs(a[max_r][col])) max_r=i;
        }
        if(max_r!=k) {// 与第 k 行交换.
            for(j=k; j<var+1; j++) swap(a[k][j],a[max_r][j]);
        }
        if(a[k][col]==0) {// 说明该 col 列第 k 行以下全是 0 了，则处理当前行的下一列.
            k--;
            continue;
        }
        for(i=k+1; i<equ; i++){// 枚举要删去的行.
            if(a[i][col]!=0) {
                LCM = lcm(abs(a[i][col]),abs(a[k][col]));
                ta = LCM/abs(a[i][col]);
                tb = LCM/abs(a[k][col]);
                if(a[i][col]*a[k][col]<0)tb=-tb;//异号的情况是相加
                for(j=col; j<var+1; j++) {
                    a[i][j] = a[i][j]*ta-a[k][j]*tb;
                }
            }
        }
    }
    // 1. 无解的情况：化简的增广阵中存在(0, 0, ..., a)这样的行(a != 0).
    for (i = k; i < equ; i++) {
        // 对于无穷解来说，如果要判断哪些是自由变元，那么初等行变换中的交换就会影响，则要记录交换.
        if (a[i][col] != 0) return -1;
    }
    // 2. 无穷解的情况：在 var * (var + 1)的增广阵中出现(0, 0, ..., 0)这样的行，即说明没有形成严格的上三角阵.
    // 且出现的行数即为自由变元的个数.
    if (k < var) {
        // 首先，自由变元有 var - k 个，即不确定的变元至少有 var - k 个.
        for (i = k - 1; i >= 0; i--) {
            // 第 i 行一定不会是(0, 0, ..., 0)的情况，因为这样的行是在第 k 行到第 equ 行.
            // 同样，第 i 行一定不会是(0, 0, ..., a)，a != 0 的情况，这样的无解的.
            free_x_num = 0; // 用于判断该行中的不确定的变元的个数，如果超过 1 个，则无法求解，它们仍然为不确定的变元.
            for (j = 0; j < var; j++) {
                if (a[i][j] != 0 && free_x[j]) free_x_num++, free_index = j;
            }
            if (free_x_num > 1) continue; // 无法求解出确定的变元.
            // 说明就只有一个不确定的变元 free_index，那么可以求解出该变元，且该变元是确定的.
            temp = a[i][var];
            for (j = 0; j < var; j++) {
                if (a[i][j] != 0 && j != free_index) temp -= a[i][j] * x[j];
            }
            x[free_index] = temp / a[i][free_index]; // 求出该变元.
            free_x[free_index] = 0; // 该变元是确定的.
        }
        return var - k; // 自由变元有 var - k 个.
    }
    // 3. 唯一解的情况：在 var * (var + 1)的增广阵中形成严格的上三角阵.
    // 计算出 Xn-1, Xn-2 ... X0.
    for (i = var - 1; i >= 0; i--) {
        temp = a[i][var];
        for (j = i + 1; j < var; j++) {
            if (a[i][j] != 0) temp -= a[i][j] * x[j];    //--因为 x[i]存的是 temp/a[i][i]的值，即是 a[i][i]=1 时 x[i]对应的值
        }
        if (temp % a[i][i] != 0) return -2; // 说明有浮点数解，但无整数解.
        x[i] = temp / a[i][i];
    }
    return 0;
}
int main(void) {
    int i, j;
    int equ,var;//equ * (var + 1)  参数
    while (scanf("%d %d", &equ, &var) != EOF) {
        memset(a, 0, sizeof(a));
        for (i = 0; i < equ; i++) {
            for (j = 0; j < var + 1; j++) {
                scanf("%d", &a[i][j]);
            }
        }
        int free_num = Gauss(equ,var);
        if (free_num == -1) printf("无解!\n");
        else if (free_num == -2) printf("有浮点数解，无整数解!\n");
        else if (free_num > 0) {
            printf("无穷多解! 自由变元个数为%d\n", free_num);
            for (i = 0; i < var; i++) {
                if (free_x[i]) printf("x%d 是不确定的\n", i + 1);
                else printf("x%d: %d\n", i + 1, x[i]);
            }
        } else {
            for (i = 0; i < var; i++) {
                printf("x%d: %d\n", i + 1, x[i]);
            }
        }
        printf("\n");
    }
    return 0;
}
```

dp
01 背包

```
int dp[maxn],v[maxn],w[maxn];
int main(){
    int T;cin>>T;
    while(T--){
        memset(dp,0,sizeof(dp));
        int n,W;cin>>n>>W;
        for(int i=0;i<n;i++) cin>>v[i];
        for(int i=0;i<n;i++) cin>>w[i];
        for(int i=0;i<n;i++)
            for(int j=W;j>=w[i];j--)
```

```
            dp[j]=max(dp[j],dp[j-w[i]]+v[i]);
        cout<<dp[W]<<endl;
    }
}
```

最长公共子序列

```
int LCS2(string s1,string s2){//s1 待匹配串 s2 匹配串
    int len1=s1.length();
    int len2=s2.length();

    int leftabove,left,above;
    int *dp=new int [len2+1];
    memset(dp,0,(len2+1)*sizeof(int));
    for(int i=1;i<=len1;i++){
        leftabove=left=dp[0];
        above=dp[1];
        for(int j=1;j<=len2;j++){
            if(s1[i-1]==s2[j-1])   dp[j]=leftabove+1;
            else    dp[j]=max(left,above);
            leftabove=above;
            above=dp[j+1];
            left=dp[j];
        }
    }
    cout<<dp[len2]<<endl;
}
int main() {
    string s1, s2;
    while(cin>>s1>>s2){
        LCS2(s1,s2);
    }
}
```

STL

```
vector<int> v;
v.begin();   //容器的起始位置
v.end();     //容器最后一个位置后的位置
v.front();v.back();   //返回第一个元素（最后一个元素，但不判断时候存在
v.empty();     //返回是否容器为空
v.clear();     //清空容器
v.erase(m);     //删除 m 位置的数据，并返回下一个数据的地址（m 是迭代器）
v.erase(m,n);      //删除 m 到 n 之间的数据，并返回下一个数据的地址
v2.assign(8,1);   // 重新给 vec2 赋值，8 个成员的初始值都为 1
v.push_back(element);     //压入一个元素到末端
v.pop_back();     //弹出最后一个元素
v.reserve(100);v.resize(101);     //resize 已经创建空间如果再 v.push_back();空间就会到 101，而 reserve 只是预留空间并没有真正创建，v.push_back();只是在第 1 位
v.size();v.capacity();        //size 表示的是已经创建的空间大小也可以表示元素个数可用 v[]的形式直接访问，capacity 容器容量，是预留空间并没有实际创建
swap(a,b);      //交换两个元素的位置如:swap(v[0],v[1]);
vector<int>  v(10);    //创建一个前十个元素为 int 的容器
vector<string> v(10,string("I"));  //使容器的前 10 个元素都为 string 型，并且都初始化为 I
vector<string> v1(v2);     //对于已经存在的 v2 创建一个 v1 副本
v.insert(place,element);
v.insert(place,n,element);      //在 place（迭代器）位插入 n 个元素
//注：对 vector 元素的访问可以用类似 c 语言的 v[],但是最好用 v.at(),它会检查是否越界更安全
v[0];   // A
v.at[0];  // B 这样越界的时候比较安全
```

reverse(vec.begin(),vec.end());

deque 双端队列

```
#include<queue>
deque <int> deq;//初始化对象为空
deque <int> deq1(10,6);//对象初始化有 10 个值为 6 的元素
deque <int>:: iterator it;
for(it = deq.begin(); it!=deq.end(); it++){
    cout<<*it<<endl;
}
deq.push_back(ele);//从队列尾部插入
deq.push_front(ele);//从队列头部插入
deq.insert(deq.begin()+1,3,9);//从队列中间插入三个 9
//和 vector 一样，双向队列也可以用下标的形式访问，也可以用 at
deq.at(n);//返回的是 n 这个下标的值
//也可以直接 deq[n]

deq1.at(1)=10;
deq1[2]=12;
//从 deq1 序列的前后各移去一个元素
deq1.pop_front();
deq1.pop_back();

deq.erase(deq.begin()+1);//清除 deq 的第二个元素

//对 deq2 赋值并显示
deq2.assign(8,1);
cout<<"deq2.assign(8,1):"<<endl;
put_deque(deq2,"deq2");
//erase(),assign()是大多数容器都有的操作
```

集合 set（特点是没有重复元素且元素时有序的）
注：set 是 STL 中一种标准关联容器（vector,list,string,deque 都是序列容器，而 set，multiset，map，multimap 是标准关联容器），它底层使用平衡的搜索树——红黑树实现，插入删除操作时仅仅需要指针操作节点即可完成，不涉及到内存移动和拷贝，所以效率比较高。

```
set<int> s;
s.insert(element);//插入元素
//遍历整个几何（不重复的有序序列）
set<int>:: iterator it;
for(it = s.begin(); it!=s.end(); it++){
    cout<<(*it)<<endl;
}
s.size();//返回元素个数
s.find(ele);//返回元素的下标，没找到的话返回 s.end();
//获得两个 set 的并
set<int> s1;
set<int> s2;
set<int> s3;//存结果
set_union(s1.begin(),s1.end(),s2.begin(),s2.end(),insert_iterator<set<int> >(s3,s3.begin()));
//输出也可以用下面的形式
copy(s3.begin(),s3.end(),ostream_iterator<int>(cout," "));
cout<<endl;
//获得两个 set 的交，注意进行集合操作之前接收结果的 set 要调用 clear()函数清空一下
s3.clear();
set_intersection(s1.begin(),s1.end(),s2.begin(),s2.end(),insert_iterator<set<int> >(s3.s3.begin()));
copy(s3.begin(),s3.end(),ostream_iterator<int>(cout," "));
cout<<endl;

//获得两个 set 的差
s3.clear();
set_difference(s1.begin(),s1.end(),s2.begin(),s2.end(),insert_iterator<set<int> >(s3,s3.begin()));
cout<<"Difference:";
copy(s3.begin(),s3.end(),ostream_iterator<int>(cout," "));
cout<<endl;
//获得两个 set 的对称差，也就是假设两个集合分别为 A 和 B 那么对称差为 AUB-AnB
   eg3.clear();
   set_symmetric_difference(s1.begin(),s1.end(),s2.begin(),s2.end(),insert_iterator<set<int> >(s3,s3.begin()));
   copy(s3.begin(),s3.end(),ostream_iterator<int>(cout," "));
   cout<<endl;
```

特别注意：常用的是将 set 升序排列

```
set<int,less<int> > set1;//降序排列
set<int,greater<int> > set1;//升序排列
```

map 和 Multimap 映射

```
#include<map>

map<int, string> mym;
//三种插入方法
mym.insert(pair<int,string>(120,"haha"));
mym.insert(map<int,string>::value_type(312,"gaga"));
mym[120] = "haha";
//find()函数返回一个迭代器指向键值为 key 的元素，如果没找到就返回指向 map 尾部的迭代器。
map<int ,string >::iterator l_it;
l_it=maplive.find(112);//返回的是一个指针
if(l_it==maplive.end())
    cout<<"we do not find112"<<endl;
else
    cout<<"wo find112"<<endl;

//也可以用这种方式来查找
map<int ,string> m;
m[1] = "haa";
m[44] = "xixi";
if(m[12]=="") puts("NO");

//删除一个元素
map<int,string> m;
map<int,string>::iterator it;
it = m.find(112);
if(it == m.end()){ exit;}
else m.erase(it);
//类似的也可以通过赋值为空来将一个元素删除

//Map 中的 swap 不是一个容器中的元素交换，而是两个容器交换

begin() //返回指向 map 头部的迭代器
clear()// 删除所有元素
count()// 返回指定元素出现的次数
empty() //如果 map 为空则返回 true
end() //返回指向 map 末尾的迭代器
equal_range()// 返回特殊条目的迭代器对
erase() //删除一个元素
find() //查找一个元素
insert() //插入元素
lower_bound() //返回键值>=给定元素的第一个位置
max_size() //返回可以容纳的最大元素个数
rbegin() //返回一个指向 map 尾部的逆向迭代器
rend() //返回一个指向 map 头部的逆向迭代器
size() //返回 map 中元素的个数
swap()// 交换两个 map
upper_bound()// 返回键值>给定元素的第一个位置
value_comp() //返回比较元素 value 的函数
```

```
//multimap 允许重复的键值插入容器
//        ************************************************************
**
//        * pair 只包含一对数
值:pair<int,char>                    *
//        * map 是一个集合类型，永远保持排好序
的，                                  *
//  pair  * map 每一个成员就是一个 pair,例如:
map<int,char>                        *
//        * map 的 insert()可以把一个 pair 对象作为 map 的参数,例如
map<p> *
//        ************************************************************
***
#pragma warning(disable:4786)
#include<map>
#include<iostream>
using namespace std;

int main(void)
{
 multimap<int,char*> m;
 //multimap 的插入只能用 insert()不能用数组
 m.insert(pair<int,char*>(1,"apple"));
```

```
        m.insert(pair<int,char*>(1,"pear"));//apple 和 pear 的价钱完全有可能是一样的
    m.insert(pair<int,char*>(2,"banana"));
    //multimap 的遍历只能用迭代器方式不能用数组
    cout<<"****************************************"<<endl;
    multimap<int,char*>::iterator i,iend;
    iend=m.end();
    for(i=m.begin();i!=iend;i++)
    {
     cout<<(*i).second<<"的价钱是"
       <<(*i).first<<"元/斤\n";
    }
    cout<<"****************************************"<<endl;
       //元素的反相遍历
    multimap<int,char*>::reverse_iterator j,jend;
    jend=m.rend();
    for(j=m.rbegin();j!=jend;j++)
    {
     cout<<(*j).second<<"的价钱是"
       <<(*j).first<<"元/斤\n";
    }
    cout<<"****************************************"<<endl;
    //元素的搜索
    find(),pair<iterator,iterator>equal_range(const key_type &k)const

       //和 multiset 的用法一样
    multimap<int,char*>::iterator s;
    s=m.find(1);//find()只要找到一个就行了，然后立即返回。
    cout<<(*s).second<<"     "
     <<(*s).first<<endl;
    cout<<"键值等于 1 的元素个数是："<<m.count(1)<<endl;
    cout<<"****************************************"<<endl;
    //删除 erase(),clear()
    m.erase(1);
       for(i=m.begin();i!=iend;i++)
    {
     cout<<(*i).second<<"的价钱是"
      <<(*i).first<<"元/斤\n";
    }
       return 0;
}
```

queue 与 priority_queue

```
#include<queue>
queue<int> q1;
q.push(x);//将元素插入到队列末尾
q.pop();//弹出队列的第一个元素（队首），并不会返回元素的值
q.front();//访问队首的元素
q.back();//访问队尾元素
q.size();//返回队列中元素的个数
```

```
priority_queue<int >q1;
priority_queue<pair<int,int> >q2;
priority_queue<int,vector<int>,greater<int> >q3;//定义小的先出队
```

stack 栈

```
#include<stack>
stack<int> s;
s.push();//压栈
s.pop();//出栈
s.top();//获取栈顶元素
s.empty();
s.size();
s.end();
s.begin();
```

下面介绍一下 stl 中的一些常用的算法

```
find(); count(); unique(); sort();
copy(v.begin(),v.end(),l.begin());
//将 v 中的元素复制到 1 中。
```

堆操作

```
make_heap(v.begin(),v.end());//创建堆
sort_heap(v.begin(),v.end());//堆排序
push_heap(v.begin(),v.end());//堆入队
pop_heap(v.begin(),v.end());//堆出队
```

卡特兰 (Catalan) 数列 原理
令 h(1)=1，catalan 数满足递归式:
h(n)= h(1)*h(n-1) + h(2)*h(n-2) + ... + h(n-1)h(1) (其中 n>=2)
该递推关系的解为：h(n)=c(2n-2,n-1)/n (n=1,2,3,...)
1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012, 742900, 2674440, 9694845, 35357670, 129644790, 477638700, 1767263190, 6564120420, 24466267020, 91482563640, 343059613650, 1289904147324, 4861946401452, …
1.括号化问题。
矩阵链乘： P=a1×a2×a3×……×an，依据乘法结合律，不改变其顺序，只用括号表示成对的乘积，试问有几种括号化的方案？(h(n)种)
2.出栈次序问题。
一个栈(无穷大)的进栈序列为 1,2,3,..n,有多少个不同的出栈序列?
类似：有 2n 个人排成一行进入剧场。入场费 5 元。其中只有 n 个人有一张 5 元钞票，另外 n 人只有 10 元钞票，剧院无其它钞票，问有多少中方法使得只要有 10 元的人买票，售票处就有 5 元的钞票找零？(将持 5 元者到达视作将 5 元入栈，持 10 元者到达视作使栈中某 5 元出栈)
3.将多边行划分为三角形问题。
将一个凸多边形区域分成三角形区域的方法数?
类似：一位大城市的律师在她住所以北 n 个街区和以东 n 个街区处工作。每天她走 2n 个街区去上班。如果他
从不穿越（但可以碰到）从家到办公室的对角线，那么有多少条可能的道路?
类似：在圆上选择 2n 个点,将这些点成对连接起来使得所得到的 n 条线段不相交的方法数?

博弈
1.巴什博奕
问题模型：只有一堆 n 个物品，两个人轮流从这堆物品中取物品，规定每次至少取一个，最多取 m 个，最后取光者得胜。)
结论:n%(m+1)==0 先手必败.否则先手必胜
变形:条件不变，改为最后取光的人输。 结论：(n-1) %(m+1)==0 先手必败,否则必胜

2. 威佐夫博奕
问题模型：有两堆各若干个物品，两个人轮流从某一堆或同时从两堆中取同样多的物品，规定每次至少取一个，多者不限，
最后取光者得胜。
结论: k=(b-a) (a>b) s = (double)(k * (sqrt(5.0) + 1) / 2); s==a 先手必败,否则必胜

3.尼姆博弈
问题模型：有三堆各若干个物品，两个人轮流从某一堆取任意多的物品，规定每次至少取一个，多者不限，最后取光者得胜。
结论：当石子堆数为 n 堆时，则推广为当对每堆的数目进行亦或之后值为零是必败态。

4.sg 函数

5.阶梯博弈
博弈在一列阶梯上进行...每个阶梯上放着自然数个点..两个人进行阶梯博弈...每一步则是将一个集体上的
若干个点(>=1)移到前面去..最后没有点可以移动的人输..
阶梯博弈也是可以转化成尼姆博弈的.
把所有奇数阶梯看成 N 堆石子..做 nim..把石子从奇数堆移动到偶数堆可以理解为拿走石子..就相当于几个
奇数堆的石子在做 Nim..

6.Chomp!博弈(巧克力游戏)
有一个 n*m 的棋盘，每次可以取走一个方格并拿掉它右边和上面的所有方格。拿到左下角的格子(1,1)者输，如下图是 8*3 的
棋盘中拿掉(6,2)和(2,3)后的状态。
结论：答案是除了 1*1 的棋盘，对于其他大小的棋盘，先手总能赢。
分析：有一个很巧妙的证明可以保证先手存在必胜策略，可惜这个证明不是构造性的，也就是说没有给出先手怎么下才能赢。

证明如下:
如果后手能赢，也就是说后手有必胜策略，使得无论先手第一次取哪个石子，后手都能获得最后的胜利。那么现在假设先手
取最右上角的石子(n,m)，接下来后手通过某种取法使得自己进入必胜的局面。但事实上，先手在第一次取的时候就可以和
后手这次取的一样，进入必胜局面了，与假设矛盾。

巧克力游戏的变形:
约数游戏：有 1~n 个数字，两个人轮流选择一个数字，并把它和它的约数擦去。擦去最后一个数的人赢，问谁会获胜。
分析：类似巧克力游戏，得到结论就是无论 n 是几，都是先手必胜。

翻棋子游戏:
题意：一个棋盘上每个格子有一个棋子，每次操作可以随便选一个朝上的棋子(x,y)，代表第 i 行第 j 列的棋子，选择一个形
如(x,b)或(a,y)(其中 b < y，a < x)的棋子，然后把它和(x,y)一起翻转，无法操作的人输。
分析：把坐标为(x,y)的棋子看成大小分别为 x 和 y 的两堆石子，则本题转化为了经典的 Nim 游戏，
如果难以把棋子看作石
子，可以先把 Nim 游戏中的一堆石子看成一个正整数，则 Nim 游戏中的每次操作是把其中一个正整数减小或者删除。
运行时间计算

```
注：vector 是顺序容器，没有 find 函数、#include<time.h>
clock_t start,finish;
start=clock();
finish=clock();
cout<< (double)(finish - start) / CLOCKS_PER_SEC;//运行时间:
```

待完善:
网络流
Dp