优先队列

```
struct cmp1{
	bool operator ()(int &a,int &b){
		return a>b;//最小值优先 写
	}
};
struct cmp2{
	bool operator ()(int &a,int &b){
		return a<b;//最大值优先 默认
	}
};
int main(){
	priority_queue<int,vector<int>,cmp2> q;
}
```

Prime 表

```
bool prime[5000010];
void Primelist(int n){
	prime[1] = 1;
	for(int i = 2;i <= n;i ++){
		if(!prime[i])
			for(int j = 2;j * i <= n;j ++)
				prime[i * j] = 1;
	}
}
```

二分

```
do{
	mid = (l + r) >> 1;
	if(check(mid)) l = mid + 1,ans = mid;
	else r = mid - 1;
}while(l <= r);
```

约瑟夫环

```
int arr[30010];
int main(){
	int n,m;
	cin >> n >> m;
	int loc = 1;
	for(int i = 1;i <= n;i ++)
		arr[i] = i;
	for(int i = n;i > 0;i --){
		loc = (loc + m - 1) % i;
		if(loc == 0)	loc = i;
		cout << arr[loc] << " ";
		for(int j = loc;j < i;j ++)
			arr[j] = arr[j + 1];
	}
	return 0;
}
```

KMP 算法

```
int next[1000010];
int ans;
void Next(string pat){
    int loc,k,len = pat.length();
    for (loc = 1,k = 0;loc < len;loc ++){
        while(k > 0 && pat[loc] != pat[k])
            k = next[k - 1];
        if (pat[loc] == pat[k])
            k ++;
        next[loc] = k;
    }
}
void kmp(string str,string pat){
    int loc,len_str = str.length(),len_pat = pat.length();
    Next(pat);
    for (int i = 0,loc = 0; i < len_str;i ++){
        while(loc > 0 && pat[loc] != str[i])
            loc = next[loc - 1];
        if (pat[loc] == str[i]) loc ++;
        if (loc == len_pat)
  //printf("Pattern occurs with shift:%d\n",(i - len_pat + 1));
            ans ++;
    }
}
```

N 皇后

```
int arr[20];
int n,ans;
bool Judge(int a,int b){
    for(int i = 1;i < a;i ++)
        if(arr[i] == b || fabs(arr[i] - b) == fabs(a - i))
            return 0;
    return 1;
}
void dfs(int dep){
    if(dep == n + 1){
        ans ++;return;
    }
    for(int i = 1;i <= n;i ++){
        arr[dep] = i;
        if(!Judge(dep,i)){
            arr[dep] = 0;continue;
        }
        dfs(dep + 1);arr[dep] = 0;
    }
}
int main(){
    cin >> n;dfs(1);
    cout << ans << endl;
    return 0;
}
```

快速幂

```
long long Qpow(long long m,long long n,long long k){
    long long res = 1;
    while(n > 0){
        if(n & 1)
            res = res * m % k;
        n = n >> 1;
        m = m * m % k;
    }
    return res;
}
```

String 字符串替代

```
string& replace_all(string& str,const string& old_value,const
string& new_value)//replace_all(str,pat,sub);串 模板 替换
串
{
    while(true){
        string::size_type pos(0);
        if((pos=str.find(old_value))!=string::npos)

str.replace(pos,old_value.length(),new_value);
        else break;
    }
    return str;
}
```

多重背包

```
void ZeroOne(int vi,int w){
    for(int i = v;i >= vi;i --)
        F[i] = max(F[i],F[i - vi] + w);
}
void Multiple(int v,int w,int amount){
    int k = 1;
    while(amount > k){
        ZeroOne(k * v,k * w);
        amount = amount - k;
        k = k * k;
    }
    ZeroOne(amount * v,amount * w);
}
```

组合数

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

卡特兰数

```
const int N =
long long K[N + 10];
void katelan(){
    K[1] = 1,K[0] = 1;
    for(int i = 2;i <= N;i ++)
        K[i] = ((4 * i - 2) / (double)(i + 1)) * K[i - 1];
}
```

贝尔数

```
const int mod =
const int N =
long long T[N + 10],B[N + 10];
void Bell(){
    B[0] = 1,B[1] = 1,T[0] = 1;
    for(int i = 2;i < N;i ++){
        T[i - 1] = B[i - 1];
        for(int j = i - 2;j >= 0;j --)
            T[j] = (T[j] + T[j + 1]) % mod;
        B[i] = T[0];
    }
}
```

广义 Fibonacci

```
//六个变量分别是 Fn = p * Fn-1 + q * Fn-2      F1 = a1     F2 = a2     第 n 项     mod

const int MAX = 2;
typedef struct{
    long long arr[2][2];
}Matrix;
long long mod;
Matrix multi(Matrix a,Matrix b){
    Matrix c;
    for(int i = 0;i < MAX;i ++)
        for(int j = 0;j < MAX;j ++){
            c.arr[i][j] = 0;
            for(int k = 0;k < MAX;k ++)
                c.arr[i][j] = (c.arr[i][j] + a.arr[i][k] *
                b.arr[k][j]) % mod;
        }
    return c;
}
Matrix fast_mod(Matrix base,long long n){
    Matrix res = {1,0,0,1};
    while(n > 0){
        if(n & 1)
            res = multi(res,base);
        n >>= 1;
        base = multi(base,base);
    }
    return res;
}
int main(){
    long long p,q,a1,a2,n;
    cin >> p >> q >> a1 >> a2 >> n >> mod;
    Matrix base = {p,q,1,0};
    Matrix ans = fast_mod(base,n - 2);
    if(n == 1)
        cout << a1 << endl;
    else if(n == 2)
        cout << a2 << endl;
    else
        cout << (ans.arr[0][0] * a2 % mod + ans.arr[0][1] *
a1 % mod) % mod << endl;
    return 0;
}
```

最短路径 Bellman

```
typedef struct{
    int from,to,cost;
}EDGE;
EDGE es[MAX_E];
int dis[MAX_D];
int V,E;
void shortest_path(int s){
    for(int i = 0;i < V;i ++) dis[i] = INF;
    dis[s] = 0;
    while(1){
        bool update = false;
        for(int i = 0;i < E;i ++){
            edge e = es[i];
            if(dis[e.from] != INF && dis[e.to] > dis[e.from] +
            e.cost){
                dis[e.to] = dis[e.from] + e.cost;
                update = true;
            }
        }
        if(!update) break;
    }
}
```

```
bool find_negative_loop(){
    memset(dis,0,sizeof(dis));
    for(int i = 0;i < V;i ++){
        for(int j = 0;j < E;j ++){
            edge e = es[j];
            if(dis[e.to] > dis[e.from] + e.cost){
                d[e.to] = d[e.from] + e.cost;
                if(i == V - 1) return true;
            }
        }
    }
    return false;
}
```

Dijkstra

```
const int INF = 1e9;
const int V = MAX_V;
int cost[V + 1][V + 1];
int dis[V + 1];
bool used[V + 1];
void dijkstra(int s){
    for(int i = 1;i <= V;i ++)
        dis[i] = INF,used[i] = 0;
    dis[s] = 0;
    while(1){
        int v = -1;
        for(int u = 1;u <= V;u ++)
            if(!used[u] && (v == -1 || dis[u] < dis[v]))
                v = u;
        if(v == -1) break;
        used[v] = 1;
        for(int u = 1;u <= V;u ++)
            dis[u] = min(dis[u],dis[v] + cost[v][u]);
    }
}
```

Dijkstra 堆优化

```
typedef struct{
    int to,cost;
}edge;
typedef pair<int,int> P;
const int V = 52;const int INF = 1e9;
vector<edge> G[V + 1];
int dis[V + 1];
void dijkstra(int s){
    priority_queue<P,vector<P>,greater<P> > q;
    for(int i = 1;i <= V;i ++)
        dis[i] = INF;
    dis[s] = 0;
    q.push(P(0,s));
    while(!q.empty()){
        P p = q.top();q.pop();
        int v = p.second;
        if(dis[v] < p.first) continue;
        for(int i = 0;i < G[v].size();i ++){
            edge e = G[v][i];
            if(dis[e.to] > dis[v] + e.cost){
                dis[e.to] = dis[v] + e.cost;
                q.push(P(dis[e.to],e.to));
            }
        }
    }
}
```

Floyd

```
int dis[V + 1][V + 1];
void floyd(){
    for(int k = 1;k <= V;k ++)
        for(int i = 1;i <= V;i ++)
            for(int j = 1;j <= V;j ++)
                dis[i][j] = min(dis[i][j],dis[i][k] + dis[k][j]);
}
```

Gcd 和 Lcn

```
long long gcd(long long n,long long m){
    long long r;
    while(1){
        r = n % m;
        if(r == 0) return m;
        n = m; m = r;
    }
}
long long lcn(long long n,long long m){
    return (m * n / gcd(n,m));
}
```

Prim

```
int V;
const int MAX_V = 100;
const int INF = 1e9;
int cost[MAX_V + 1][MAX_V + 1];
int mincost[MAX_V    + 1];
bool used[MAX_V    + 1];
int prim(){
    for(int i = 1;i <= V;i ++)
        mincost[i] = INF,used[i] = 0;
    mincost[1] = 0;
    int res = 0;
    while(1){
        int v = -1;
        for(int u = 1;u <= V;u ++)
            if(!used[u] && (v == -1 || mincost[u] <
            mincost[v]))    v = u;
        if(v == -1) break;
        used[v] = 1;
        res += mincost[v];
        for(int u = 1;u <= V;u ++)
            mincost[u] = min(mincost[u],cost[v][u]);
    }
    return res;
}
```

Kruskal

```
int par[MAX_V];
int rank[MAX_V];
void init(int n){
    for(int i = 0;i < n;i ++)
        par[i] = i,rank[i] = 0;
}
int find(int x){
    if(par[x] == x) return x;
    else return par[x] = find(par[x]);
}
void unite(int x,int y){
    x = find(x),y = find(y);
    if(x == y) return;
    if(rank[x] < rank[y]) par[x] = y;
    else par[y] = x;
    if(rank[x] == rank[y]) rank[x] ++;
}
bool same(int x,int y){
    return find(x) == find(y);
}
```

```
typedef struct{
    int u,v,cost;
}edge;
edge es[MAX_E];
int V,E;
bool cmp(edge e1,edge e2){
    return e1.cost < e2.cost;
}
int kruskal(){
    sort(es + 1,es + E + 1,cmp);
    init(V + 1);
    int res = 0;
    for(int i = 1;i <= E;i ++){
        edge e = es[i];
        if(!same(e.u,e.v))
            unite(e.u,e.v),res += e.cost;
    }
    return res;
}
```

SCC 强连通分量

```
int V;
vector<int> G[MAX_V + 10];
vector<int> rG[MAX_V + 10];
vector<int> vs;//后续遍历顶点列表
bool vis[MAX_V + 10];
int cmp[MAX_V + 10];//所属强连通分量的拓排序
void add(int from,int to){
    G[from].push_back(to);
    rG[to].push_back(from);
}
void dfs(int v){
    vis[v] = true;
    for(int i = 0;i < G[v].size();i ++)
        if(!vis[G[v][i]]) dfs(G[v][i]);
    vs.push_back(v);
}
void rdfs(int v,int k){
    vis[v] = true;
    cmp[v] = k;
    for(int i = 0;i < rG[v].size();i ++)
        if(!vis[rG[v][i]]) rdfs(rG[v][i],k);
}
int scc(){
    memset(vis,0,sizeof(vis));
    vs.clear();
    for(int v = 0;v < V;v ++)
        if(!vis[v]) dfs(v);
    memset(vis,0,sizeof(vis));
    int k = 0;
    for(int i = vs.size() - 1;i >= 0;i --){
        if(!vis[vs[i]]) rdfs(vs[i],k ++);
    }
    return k;
}
```

```
//x1,y1 为最高点
double x1, y1, x2, y2, x3, y3;
scanf("%lf%lf%lf%lf%lf%lf", &x1, &y1, &x2, &y2, &x3, &y3);
printf("%.2f\n",
(y3-y1)*(x2-x3)*(x2-x3)*(x2-x3)/6.0/(x1-x3)/(x1-x3));
```

```
//错排原理
f[1] = 0,f[2] = 1,f[3] = 2;
f[i] = (i - 1) * (f[i - 1] + f[i - 2]);
```

LCA 最小公共祖先 二分

```
vector<int> G[MAX_V];
int root;
int parent[MAX_LOG_V][MAX_V];
int depth[MAX_V];
void dfs(int v,int p,int d){
    parent[0][v] = p;
    depth[v] = d;
    for(int i = 0;i < G[v].size();i ++)
        if(G[v][i] != p) dfs(G[v][i],v,d + 1);
}
void init(int V){
    dfs(root,-1,0);
    for(int k = 0;k + 1 < MAX_LOG_V;k ++){
        for(int v = 0;v < V;v ++){
            if(parent[k][v] < 0) parent[k + 1][v] = -1;
            else parent[k + 1][v] = parent[k][parent[k][v]];
        }
    }
}
```

```
int lca(int u,int v){
    if(depth[u] > depth[v]) swap(u,v);
    for(int k = 0;k < MAX_LOG_V;k ++){
        if((depth[v] - depth[u]) >> k & 1)
            v = parent[k][v];
    }
    if(u == v) return u;
    for(int k = MAX_LOG_V - 1;k >= 0;k --){
        if(parent[k][u] != parent[k][v]){
            u = parent[k][u];
            v = parent[k][v];
        }
    }
    return parent[0][u];
}
```

并查集

```
int par[MAX_N];
int rankk[MAX_N];
void init(int n){
    for(int i = 0;i < n;i ++)
        par[i] = i,rankk[i] = 0;
}
int find(int x){
    if(par[x] == x) return x;
    else return par[x] = find(par[x]);
}
void unite(int x,int y){
    x = find(x),y = find(y);
    if(x == y) return;
    if(rankk[x] < rankk[y]) par[x] = y;
    else par[y] = x;
    if(rankk[x] == rankk[y]) rankk[x] ++;
}
bool same(int x,int y){
    return find(x) == find(y);
}
```

最长序列

```
int dp[10000]
int max_length_dp;
int numbers[10000];
int dping(int loc){
    int max = 0;
    for(int i = 0;i < loc;i ++)
        if(numbers[loc] >= numbers[i] && dp[i] > max)
            max = dp[i];
    dp[loc] = max + 1;
    return dp[loc];
}
int main(){
    int n;    cin >> n;
    numbers = new int[n];
    dp = new int[n];
    for(int i = 0;i < n;i ++)
        cin >> numbers[i];
    for(int i = 0;i < n;i ++)
        if(dping(i) > max_length_dp)
            max_length_dp = dping(i);
    cout << max_length_dp;
    return 0;
 }
```

树状数组 1、2

```
int tree[100010],n,m;
void add(int k,int num){
    while(k <= n)
        tree[k] += num,k += k & -k;
}
int read(int k){
    int sum = 0;
    while(k)
        sum += tree[k],k -= k & -k;
    return sum;
}
int main(){
    cin >> n;
    int ele;
    for(int i = 1;i <= n;i ++)
        cin >> ele,add(i,ele);
    cin >> m;
    while(m --){
        int a,b,c;
        cin >> a >> b >> c;
        if(a == 1)    add(b,c);
        else
            cout << read(c) - read(b - 1) << endl;
    }
    return 0;
}
```

```
int arr[1000010],crr[1000010];
int n,m;
void add(int k,int num){
    while(k)    crr[k] += num,k -= k & -k;
}
int read(int k){
    int sum = arr[k];
    while(k <= n)    sum += crr[k],k += k & -k;
    return sum;
}
int main(){
    cin >> n;int ele;
    for(int i = 1;i <= n;i ++)
        cin >> arr[i];
    cin >> m;
    while(m --){
        int opr;cin >> opr;
        if(opr == 1){
            int a,b,c;
            cin >> a >> b >> c;
            add(b,c);add(a - 1,-c);
        }
        else{
            int a;    cin >> a;
            cout << read(a) << endl;
        }
    }    return 0;
}
```

DP 题

石子归并

```
int main(){
    int n,w[106],f[106][106],st;
    cin >> n;
    memset(f,0x3f3f3f3f,sizeof(f));
    for(int i = 1;i <= n;i ++){
        cin >> st;
        w[i] = w[i - 1] + st;
        f[i][i] = 0;
    }
    for(int j = 2;j <= n;j ++)
        for(int i = j - 1;i >= 1;i --)
            for(int k = i;k < j;k ++)
                f[i][j] = min(f[i][j],f[i][k] + f[k + 1][j] + w[j] -
                w[i - 1]);
    cout << f[1][n] << endl;
    return 0;
}
```

乘积最大

```
long long arr[50],sum[50][50],f[50][50];
int main()
{
    int n,m;
    string str;
    cin >> n >> m;
    cin >> str;
    for(int i = 1;i <= n;i ++)
        arr[i] = str[i - 1] - '0',sum[i][i] = arr[i];
    for(int i = 1;i < n;i ++)
        for(int j = i + 1;j <= n;j ++)
            sum[i][j] = sum[i][j - 1] * 10 + arr[j];
    for(int i = 1;i <= n;i ++)
        f[i][0] = sum[1][i];
    for(int j = 1;j <= m;j ++)
        for(int i = j + 1;i <= n;i ++)
            for(int k = j;k <= i - 1;k ++)
                f[i][j] = max(f[i][j],f[k][j - 1] * sum[k + 1][i]);
    cout << f[n][m] << endl;
    return 0;
}
```

数的划分

```
int dp[210][210];
int main(){
    int n,m;
    cin >> n >> m;
    dp[0][0] = 1;
    for(int i = 1;i <= m;i ++)
        for(int j = 0;j <= n;j ++){
            if(j - i >= 0)
                dp[i][j] = dp[i - 1][j] + dp[i][j - i];
            else
                dp[i][j] = dp[i - 1][j];
        }
    cout << dp[m][n] - dp[m - 1][n] << endl;//dp[m][n]表示
    n 个数分成不超过 m 份方法数
    return 0;
}
```

线段覆盖

```
typedef pair<int,int> Moran;
Moran moran[110];
bool cmp(Moran a,Moran b){
    return a.second < b.second ;
}
int main() {
    int n;
    cin >> n;
    for(int i = 0;i < n;i ++){
        cin >> moran[i].first >> moran[i].second ;
        if(moran[i].first > moran[i].second )
            swap(moran[i].first ,moran[i].second );
    }
    sort(moran,moran + n,cmp);
    int st = moran[0].second ;
    int cnt = 1;
    for(int i = 1;i < n;i ++)
        if(moran[i].first >= st)
            cnt ++,st = moran[i].second ;
    cout << cnt << endl;
    return 0;
}
```

最长公共序列

```
const int N = ;
int longest[N][N];
int LCS(string s1,string s2){
    int i,j,len1,len2;
    len1 = s1.length();
    len2 = s2.length();
    longest[0][0] = 0;
    for(i = 1;i <= len1;i ++) longest[i][0] = 0;
    for(i = 1;i <= len1;i ++) longest[0][i] = 0;
    for (i = 1;i <= len1;i ++){
        for (j = 1;j <= len2;j ++){
            if(s1[i - 1] == s2[j - 1])
                longest[i][j] = longest[i - 1][j - 1] + 1;
            else if(longest[i - 1][j] > longest[i][j - 1])
                longest[i][j] = longest[i - 1][j];
            else
                longest[i][j] = longest[i][j - 1];
        }
    }
    return longest[len1][len2];
}
```

最长上升子序列 **nlogn**

```
int arr[maxn],b[maxn];
int n;
int LIS(){
    int len = 1,j;
    b[1] = arr[1];
    for(int i = 2;i <= n;i ++){
        if(arr[i] > b[len]) j = ++len;
        else j = lower_bound(b + 1,b + 1 + len,arr[i]) - b;
        b[j] = arr[i];
    }
    return len;
}
```

Java 大数

```
import java.util.*;
import java.math.*;
public class Main{
    public static void main(String []args){
        Scanner cin=new Scanner(System.in);
        BigDecimal x,y;
        String a,b;
        while(cin.hasNext()){
            a=cin.next();
            b=cin.next();
            x=new BigDecimal(a);
            y=new BigDecimal(b);
          System.out.println(x.add(y).stripTrailingZeros().
          toPlainString());
        }
    }
}
```

```
BigInteger mod = new BigInteger("23333333333");
while(Inx.hasNext()){
        n=Inx.nextBigInteger();
        m = Inx.nextBigInteger();
        BigInteger yu = m.remainder(BigInteger.valueOf(3));
        BigInteger end = m.divide(BigInteger.valueOf(3));
        end = end.add(n);
        BigInteger temp = new BigInteger("0");
        temp =temp.add(n);
        temp = temp.add(BigInteger.valueOf(1));
        temp = temp.add(end);
        temp = temp.multiply(end.subtract(n));
        temp = temp.divide(BigInteger.valueOf(2));
        temp = temp.multiply(BigInteger.valueOf(3));
        int r = yu.intValue();
        for(int i = 0;i < r;i ++){
                temp = temp.add(BigInteger.valueOf(i));
                 temp = temp.add(end);
         }
         temp = temp.remainder(mod);
         System.out.println(temp);
}
```

DP 硬币 max 和 min

```
int dpmax(int S) {
    if(vis[S]) return d[S];
    vis[S] = 1;
    int &ans = d[S];
    ans = -1 << 30;
    for(int i = 1; i <= n; ++i) {
        if(S >= V[i]) ans = max(ans, dpmax(S - V[i]) + 1);
    }
    return ans;
}
int dpmin(int S) {
    if(vis[S]) return d[S];
    vis[S] = 1;
    int &ans = d[S];
    ans = -1 >> 30;
    for(int i = 1; i <= n; ++i) {
        if(S >= V[i]) ans = min(ans, dpmin(S - V[i]) + 1);
    }
    return ans;
}
```

逆序对

```
long long arr[100010],sum[100010],temp[100010],cnt;
void merge(int l,int mid,int r){
    int i = l,j = mid + 1,loc = l;
    while(i <= mid && j <= r){
        if(sum[i] > sum[j]){
            temp[loc ++] = sum[j ++];
            cnt += mid - i + 1;
        }
        else temp[loc ++] = sum[i ++];
    }
    while(i <= mid) temp[loc ++] = sum[i ++];
    while(j <= r) temp[loc ++] = sum[j ++];
    for(int i = l;i <= r;i ++) sum[i] = temp[i];
}
void mergesort(int l,int r){
    if(l < r){
        int mid = (l + r) >> 1;
        mergesort(l,mid);
        mergesort(mid + 1,r);
        merge(l,mid,r);
    }
}
```

凸包面积

```
typedef struct{
    double x,y;
}POINT;
POINT pot[200];
double Cross(POINT a,POINT b){
    return a.x * b.y - a.y * b.x;
}
int main(){
    int n;
    cin >> n;
    double sum = 0;
    for(int i = 1;i <= n;i ++){
        cin >> pot[i].x >> pot[i].y;
    }
    for(int i = 1;i < n;i ++)
        sum += fabs(Cross(pot[i],pot[i + 1]));
    sum += fabs(Cross(pot[1],pot[n]));
    cout << sum / 2 << endl;
    return 0;
}
```

三角形外心

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

三角形有向面积两倍

```
double area(double x0,double y0,double x1,double
y1,double x2,double,y2){
        return x0*y1+x2*y0+x1*y2-x0*y2-x1*y0-x2*y1;
}
```

组合博弈

1、巴什博弈 n = (m + 1) * r + s

2、威佐夫博弈

```
int main(){
    int a,b,k,a_k;
    while(scanf("%d%d",&a,&b)!=EOF){
        k = abs(a-b);
        a = a < b? a : b;
        a_k = floor(k*(1.0 + sqrt(5.0))/2);
        printf("%d\n",a!=a_k);
        //输出为 0，说明该点为必败点，1 为必胜点
    }
    return 0;
}
```

3、尼姆博弈

全部亦或 为 0 则为 T 不为 0 为 S

取火柴 1：取完胜 先手必胜为 S 必败为 T

取火柴 2：取完败 先手必胜态为 T0 S2 S1 必败态为 S0 T2

取火柴 1 时候 第一步选择数

```
int arr[5000];
int main(){
    int n;
    while(cin >> n){
        if(n == 0) break;
        int temp = 0;
        for(int i = 1;i <= n;i ++){
            cin >> arr[i];
            temp = temp ^ arr[i];
        }
        int cnt = 0;
        for(int i = 1;i <= n;i ++){
            int st = arr[i] ^ temp;
            if(st < arr[i])
                cnt ++;
        }
        cout << cnt << endl;
    }
    return 0;
}
```

匈牙利算法

```
const int maxn = ; bool vis[maxn],f[maxn][maxn];
int n,m,linkk[maxn];
bool find(int x){
    for(int i = 1;i <= n;i ++){
        if(f[x][i] && !vis[i]){
            vis[i] = 1;
            if(linkk[i] == -1 || find(linkk[i])){
                linkk[i] = x;
                return 1;
            }
        }
    }
    return 0;
}
int maxmatch(){
    int num = 0;
    memset(linkk,-1,sizeof(linkk));
    for(int i = 1;i <= n;i ++){
        memset(vis,0,sizeof(vis));
        if(find(i)) num ++;
    }
    return num;
}
```