目录

[动态规划dp 4](#_Toc68293310)

[数位dp 4](#_Toc68293311)

[数据结构 6](#_Toc68293312)

[树状数组[区间修改单点查询] 6](#_Toc68293313)

[线段树[单点修改区间查询] 7](#_Toc68293314)

[主席树 10](#_Toc68293315)

[主席树前k小的和 12](#_Toc68293316)

[RBtree 14](#_Toc68293317)

[splay 16](#_Toc68293318)

[treap比x大的数有多少个 18](#_Toc68293319)

[trie思想建树 22](#_Toc68293320)

[数学 24](#_Toc68293321)

[埃筛 24](#_Toc68293322)

[大素数判定+泼辣的肉 25](#_Toc68293323)

[第几个质数 28](#_Toc68293324)

[费马小定理 29](#_Toc68293325)

[高精度 30](#_Toc68293326)

[高精度除法 33](#_Toc68293327)

[高斯-约旦消元 34](#_Toc68293328)

[矩阵快速幂 36](#_Toc68293329)

[扩展欧几里得 38](#_Toc68293330)

[欧拉函数 39](#_Toc68293331)

[欧拉筛 39](#_Toc68293332)

[线性基 40](#_Toc68293333)

[圆和矩形的面积交 42](#_Toc68293334)

[Min25 46](#_Toc68293335)

[Zeller Formula 48](#_Toc68293336)

[图论 49](#_Toc68293337)

[网络流 49](#_Toc68293338)

[二分图最大流 49](#_Toc68293339)

[Dinic（Node版本） 51](#_Toc68293340)

[次小生成树 52](#_Toc68293341)

[二分图匹配-匈牙利算法 59](#_Toc68293342)

[dijkstra 61](#_Toc68293343)

[LCA 63](#_Toc68293344)

[LCA-倍增 63](#_Toc68293345)

[LCA-tarjan 65](#_Toc68293346)

[tarjan求割点 67](#_Toc68293347)

[tarjan缩点 69](#_Toc68293348)

[字符串 72](#_Toc68293349)

[KMP 72](#_Toc68293350)

[Manachar 73](#_Toc68293351)

[最大字典序子串 75](#_Toc68293352)

[最大最小表示法 75](#_Toc68293353)

[DFS 76](#_Toc68293354)

[DSU（树上启发式合并） 76](#_Toc68293355)

[STL&杂项 79](#_Toc68293356)

[二分（标准） 79](#_Toc68293357)

[优先队列 80](#_Toc68293358)

[exmu 82](#_Toc68293359)

[highbit 83](#_Toc68293360)

[LIS（最长上升子序列） 84](#_Toc68293361)

[Tarjan 86](#_Toc68293362)

[对拍.bat 87](#_Toc68293363)

[乱七八糟的性质 88](#_Toc68293364)

[图论的一些概念及性质 88](#_Toc68293365)

[位运算 92](#_Toc68293366)

[组合数学 92](#_Toc68293367)

# 动态规划dp

## 数位dp

/\*   
 LOJ 10163   
 ACWing, 1081 度的数量   
\* @ author: dragon\_bra   
\* @ email: tommy514@foxmail.com   
\* @ date: 2021-03-10 16:31   
\*/   
   
#include <bits/stdc++.h>   
#define fastio ios::sync\_with\_stdio(false); cin.tie(0);   
using namespace std;   
   
typedef long long ll;   
const int N = 35 + 10;   
   
void redirect() {   
#ifdef LOCAL   
 freopen("in.txt", "r", stdin);   
 freopen("out.txt", "w", stdout);   
#endif   
}   
   
int B, K;   
int f[N][N] = {0};   
   
void init() {   
 for (int i = 0; i < N; i++) {   
 for (int j = 0; j <= i; j++) {   
 if (j == 0)   
 f[i][j] = 1;   
 else   
 f[i][j] = f[i - 1][j - 1] + f[i - 1][j];   
 }   
 }   
}   
   
int dp(int n) {   
 if (!n)   
 return 0;   
   
 vector<int> nums;   
   
 while (n)   
 nums.push\_back(n % B), n /= B;   
   
 int res = 0;   
 int last = 0; // 当前已有1的个数   
   
 for (int i = nums.size() - 1; i >= 0; i--) {   
 int x = nums[i];   
   
 if (x) {   
 res += f[i][K - last];   
   
 if (x > 1) {   
 if (K - last - 1 >= 0)   
 res += f[i][K - last - 1];   
   
 break;   
 } else {   
 last ++;   
   
 if (last > K)   
 break;   
 }   
 }   
   
 if (i == 0 && last == K)   
 res ++;   
 }   
   
 return res;   
}   
   
int main() {   
 redirect();   
 init();   
   
 int l, r;   
 cin >> l >> r >> K >> B;   
   
 cout << dp(r) - dp(l - 1) << endl;   
   
 return 0;   
}

# 数据结构

## 树状数组[区间修改单点查询]

int n,m;   
int a[50005] = {0},c[50005]; //对应原数组和树状数组   
   
int lowbit(int x){   
 return x&(-x);   
}   
   
void updata(int i,int k){ //在i位置加上k   
 while(i <= n){   
 c[i] += k;   
 i += lowbit(i);   
 }   
}   
   
int getSum(int i){ //求D[1 - i]的和，即A[i]值   
 int res = 0;   
 while(i > 0){   
 res += c[i];   
 i -= lowbit(i);   
 }   
 return res;   
}   
   
int main(){   
 cin>>n;27 for(int i = 1; i <= n; i++){   
 cin>>a[i];   
 updata(i,a[i] - a[i-1]); //输入初值的时候，也相当于更新了值   
 }   
   
 //[x,y]区间内加上k   
 updata(x,k); //A[x] - A[x-1]增加k   
 updata(y+1,-k); //A[y+1] - A[y]减少k   
   
 //查询i位置的值   
 int sum = getsum(i);   
   
 return 0;   
}

## 线段树[单点修改区间查询]

#include <cstdio>  
#include <iostream>  
#include <algorithm>  
#include <cmath>  
#include <cstring>  
#include <map>  
#include <set>  
#include <queue>  
#include <string>  
#include <vector>  
using namespace std;  
typedef long long ll;  
typedef unsigned long long ull;  
const int INF = 0x7fffffff;  
const int mod = 1e9+7;  
const double eps = 1e-5;  
const int N = 1e5+10;  
  
void redirect(){  
 #ifdef LOCAL  
 freopen("test.txt","r",stdin);  
 #endif  
}  
inline ll read(){  
 ll f=1,x=0;char ch;  
 do{ch=getchar();if(ch=='-')f=-1;}while(ch<'0'||ch>'9');  
 do{x=x\*10+ch-'0';ch=getchar();}while(ch>='0'&&ch<='9');  
 return x\*f;  
}  
  
int n,k;  
int pos[N];int a[N];  
  
struct NOOD {  
 int l, r, add, Max;  
}tree[N \* 4 + 5];  
void Build(int L, int R, int x) {  
 tree[x].l = L, tree[x].r = R, tree[x].Max = 0;  
 if(L == R) {  
 tree[x].Max = a[L];  
 return ;  
 }  
 int mid = (L + R) / 2;  
 Build(L, mid, x \* 2);  
 Build(mid + 1, R, x \* 2 + 1);  
 tree[x].Max = max(tree[x \* 2].Max, tree[x \* 2 + 1].Max);  
}  
void PushDown(int x) {  
 if(tree[x].add) {  
 tree[x \* 2].Max = tree[x].add;  
 tree[x \* 2 + 1].Max = tree[x].add;  
 tree[x \* 2].add = tree[x].add;  
 tree[x \* 2 + 1].add = tree[x].add;  
 tree[x].add = 0;  
 }  
}  
void Update(int L, int R, int add, int x) {  
 if(L <= tree[x].l && tree[x].r <= R) {  
 tree[x].add = add;  
 tree[x].Max = add;  
 return ;  
 }  
 PushDown(x);  
 int mid = (tree[x].l + tree[x].r) / 2;  
 if(L <= mid)Update(L, R, add, x \* 2);  
 if(R > mid)Update(L, R, add, x \* 2 + 1);  
 tree[x].Max = max(tree[x \* 2].Max, tree[x \* 2 + 1].Max);  
}  
  
int Query(int L, int R, int x) {  
 if(L <= tree[x].l && tree[x].r <= R)return tree[x].Max;  
 PushDown(x);  
 int mid = (tree[x].l + tree[x].r) / 2;  
 int res = 0;  
 if(L <= mid) res = max(res, Query(L, R, x \* 2));  
 if(R > mid) res = max(res, Query(L, R, x \* 2 + 1));  
 return res;  
}  
  
int nxt[N];int ans[N];  
  
int dfs(int i){  
 if(nxt[i]==0||ans[i]!=1) return ans[i];  
 else return ans[i]=dfs(nxt[i])+1;  
}  
  
int main(){  
 redirect();  
 int T;scanf("%d",&T);  
 while(T--){  
 scanf("%d%d",&n,&k);  
 memset(nxt,0,sizeof(nxt));memset(tree, 0, sizeof(tree));  
 for(int i=1;i<=n;i++){  
 scanf("%d",&a[i]);pos[a[i]]=i;ans[i]=1;  
 }  
 Build(1, n, 1);  
 for(int i=n;i>=1;i--){  
 Update(pos[i], pos[i] , 0, 1);  
 int big = Query(max(pos[i]-k,1), min(pos[i]+k,n), 1);  
 if(big!=0) nxt[i]=big;  
 }  
  
 for(int i=1;i<=n;i++){  
 int ans = dfs(i);printf("%d%c",ans,i==n?'\n':' ');  
 }  
  
 }  
 return 0;  
}  
  
/\*  
---linux compile---  
g++ aa.cpp -o aa  
./ aa  
-------------------  
author:dragon\_bra  
\*/

## 主席树

#include<iostream>  
#include<algorithm>  
#include<cstdio>  
#include<cstring>  
using namespace std;  
const int N = 200500;  
  
void redirect() {  
 #ifdef LOCAL  
 freopen("in.txt","r",stdin);  
 freopen("out.txt","w",stdout);  
 #endif  
}  
  
struct node{  
 int l, r, sum;  
 #define l(x) tree[x].l  
 #define r(x) tree[x].r  
 #define sum(x) tree[x].sum  
}tree[N<<5];  
  
int n, m, a[N], b[N];  
int q, cnt, t[N];  
int build(int l, int r) {  
 int rt = ++cnt;  
 sum(rt) = 0;  
 int mid = (l + r) >> 1;  
 if (l < r) {  
 l(rt) = build(l, mid);  
 r(rt) = build(mid + 1, r);  
 }  
 return rt;  
}  
inline int update(int pre,int l,int r,int x) {  
 int rt = ++cnt;  
 l(rt) = l(pre), r(rt) = r(pre);  
 sum(rt) = sum(pre) + 1;  
 int mid = (l + r) >> 1;  
 if (l < r) {  
 if (x <= mid) l(rt) = update(l(pre), l, mid, x);  
 else r(rt) = update(r(pre), mid + 1, r, x);  
 }  
 return rt;  
}  
inline int query(int u,int v,int l,int r,int k) {  
 if (l >= r) return l;  
 int x = sum(l(v)) - sum(l(u));  
 int mid = (l + r) >> 1;  
 if (x >= k) return query(l(u), l(v), l, mid, k);  
 else return query(r(u), r(v), mid + 1, r, k - x);  
}  
int main() {  
 redirect();  
 cin >> n >> q;  
 for (int i = 1;i <= n; i++) {  
 cin >> a[i]; b[i] = a[i];  
 }  
 sort(b + 1,b + n + 1);  
 m = unique(b + 1,b + n + 1) - b - 1;  
  
 t[0] = build(1, m);  
 for (int i = 1;i <= n; i++) {  
 int T = lower\_bound(b + 1,b + m + 1, a[i]) - b;  
 t[i] = update(t[i-1], 1, m, T);  
 }  
  
 while (q--) {  
 int l, r, k;  
 cin >> l >> r >> k;  
 printf ("%d\n", b[query(t[l-1], t[r], 1, m, k)]);  
 }  
 return 0;  
}

## 主席树前k小的和

#include<bits/stdc++.h>  
using namespace std;  
const int MAXN=100010;  
const int M=MAXN\*30;  
int n,q,m,tot;  
int a[MAXN],t[MAXN];  
int T[MAXN],lson[M],rson[M],c[M];  
long long sum[M];  
void Init\_hash(){  
 for(int i=1;i<=n;i++){  
 t[i] = a[i];  
 }  
 sort(t+1,t+1+n);  
 m=unique(t+1,t+1+n)-t-1;  
}  
int build(int l,int r){  
 int root=tot++;  
 c[root]=0; sum[root] = 0;  
 if(l!=r){  
 int mid=(l+r)>>1;  
 lson[root] = build(l,mid);  
 rson[root] = build(mid+1,r);  
 }  
 return root;  
}  
int Hash(int x){  
 return lower\_bound(t+1,t+1+m,x)-t;  
}  
int update(int root,int pos, int val){  
 int newroot = tot++,tmp = newroot;  
 c[newroot] = c[root] + val;  
 sum[newroot] = sum[root] + t[pos];  
 int l=1,r=m;  
 while(l<r){  
 int mid = (l+r)>>1;  
 if(pos <= mid){  
 lson[newroot]= tot++; rson[newroot] = rson[root];  
 newroot = lson[newroot];root = lson[root];  
 r = mid;  
 }  
 else{  
 rson[newroot] = tot++; lson[newroot] = lson[root];  
 newroot = rson[newroot]; root = rson[root];  
 l = mid+1;  
 }  
 c[newroot] = c[root] + val;  
 sum[newroot] = sum[root] + t[pos];  
 }  
 return tmp;  
}  
int query(int left\_root,int right\_root,int k){  
 int l=1,r=m;  
 long long res = 0;  
 while( l < r ){  
 int mid = (l+r)>>1;  
 if(c[lson[left\_root]]-c[lson[right\_root]]>=k){  
 r = mid;  
 left\_root = lson[left\_root];  
 right\_root = lson[right\_root];  
 }  
 else{  
 l = mid + 1;  
 k -= c[lson[left\_root]]-c[lson[right\_root]];  
 res += sum[lson[left\_root]] - sum[lson[right\_root]];  
 left\_root = rson[left\_root];  
 right\_root = rson[right\_root];  
 }  
 }  
 return res;  
}  
int main(){  
 #ifdef LOCAL  
 freopen("in.txt","r",stdin);  
 freopen("out.txt","w",stdout);  
 #endif  
 while(scanf("%d%d",&n,&q) == 2){  
 tot = 0;  
 for(int i = 1; i <= n;i++){  
 scanf("%d",&a[i]);  
 }  
 Init\_hash();  
 T[n+1] = build(1,m);  
 for(int i = n;i ;i--){  
 int pos = Hash(a[i]);  
 T[i] = update(T[i+1], pos ,1);  
 }  
 while(q--){  
 int l,r,k;  
 scanf("%d%d%d",&l,&r,&k);  
 k = (r-l+1 + 1) - k; // 第k小变成第k大  
 printf("%d\n",query(T[l],T[r+1],k));  
 }  
 }  
}

## RBtree

template<class T>  
struct RBtree{  
 #define l \_M\_left  
 #define r \_M\_right  
 #define p \_M\_parent  
 #define node \_Rb\_tree\_node\_base  
#if \_\_cplusplus<=199711L  
 #define key \_M\_value\_field.first  
 #define size \_M\_value\_field.second  
#else //c++11  
 #define key \_M\_storage.\_M\_ptr()->first  
 #define size \_M\_storage.\_M\_ptr()->second  
#endif  
 typedef \_Rb\_tree\_node<pair<const T,int> > Node; map<T,int> M;  
 void fix\_size(node \*it){  
 int &it\_size=static\_cast<Node\*>(it)->size;it\_size=1;  
 if (it->l)it\_size+=static\_cast<Node\*>(it->l)->size;  
 if (it->r)it\_size+=static\_cast<Node\*>(it->r)->size;  
 }  
 void fix\_all(node \*it,node \*end){  
 for (;;it=it->p){  
 if (it->l)fix\_size(it->l);if (it->r)fix\_size(it->r);  
 if (it->p==end){fix\_size(it);break;}  
 }  
 }  
 void insert(const T &x){  
 pair<typename map<T,int>::iterator,bool> it=M.insert(make\_pair(x,0));  
 if (!it.second)return;  
 fix\_all(it.first.\_M\_node,M.end().\_M\_node);  
 }  
 int select(int k){  
 node \*p=get\_root();  
 while (k){  
 int sizel=p->l?static\_cast<Node\*>(p->l)->size:0;  
 if (k==sizel+1)break;  
 if (k<=sizel)p=p->l;  
 else k-=sizel+1,p=p->r;  
 }  
 return static\_cast<Node\*>(p)->key;  
 }  
 int rank(int x){  
 node \*p=get\_root(); int res=0;  
 while (p){  
 int y=static\_cast<Node\*>(p)->key;  
 int s=p->l?static\_cast<Node\*>(p->l)->size:0;  
 if (y<=x)res+=s+1,p=p->r;  
 else p=p->l;  
 }  
 return res;  
 }  
 node \*get\_root(){  
 node \*it=M.begin().\_M\_node;  
 while (it->p!=M.end().\_M\_node)it=it->p;  
 return it;  
 }  
 void print(){print\_node(get\_root(),"");}  
 void print\_node(const node \*it,string str){  
 if (!it){cout<<str<<"nil (0)"<<endl;return;}  
 cout<<str<<static\_cast<const Node\*>(it)->key;  
 cout<<"("<<static\_cast<const Node\*>(it)->size<<")"<<endl;  
 print\_node(it->l,str+" "); print\_node(it->r,str+" ");  
 }  
 #undef l  
 #undef r  
 #undef p  
 #undef node  
 #undef key  
 #undef size  
};  
RBtree<int> a;

## splay

#include <bits/stdc++.h>   
using namespace std;   
typedef long long ll;   
const int N = 2e5+10;   
   
struct node{   
 int data;   
}\_a[N];   
   
bool operator < (node const &\_a,node const &\_b){   
 return \_a.data<\_b.data;   
}   
bool operator > (node const &\_a,node const &\_b){   
 return \_a.data>\_b.data;   
}   
bool operator == (node const &\_a,node const &\_b){   
 return \_a.data<\_b.data;   
}   
bool operator != (node const &\_a,node const &\_b){   
 return \_a.data<\_b.data;   
}   
   
int n,t,\_root,\_sz;   
int \_fa[N],\_s[N][2],\_cnt[N],\_size[N];ll \_sum[N];   
   
inline int ws(int x){return \_s[\_fa[x]][1]==x;}//which son   
void setson(int son,int f,int w){//0-left,С;1-right,��;   
 if(son!=0) \_fa[son]=f;   
 if(f!=0) \_s[f][w]=son;   
}   
void maintain(int x){   
 \_size[x]=\_size[\_s[x][0]]+\_size[\_s[x][1]] + \_cnt[x];   
 \_sum[x]=\_sum[\_s[x][0]] + \_sum[\_s[x][1]] + (ll)\_cnt[x]\*\_a[x].data;   
}   
void rot(int x){   
 int f=\_fa[x]; int ff=\_fa[x]; int w=ws(x); int wf=ws(f);   
 int p=\_s[x][!w];   
 setson(p,f,w);   
 setson(x,ff,wf);   
 setson(f,x,!w);//!w   
 maintain(f);   
 maintain(x);   
}   
void splay(int x){   
 for(;\_fa[x];rot(x)) if(\_fa[\_fa[x]]&&ws(\_fa[x])==ws(x)) rot(\_fa[x]);//zig-zag or zig-zig   
 \_root=x;   
}   
void insert(int now,node p){   
 if(\_root==0){   
 \_root=++\_sz;   
 \_a[\_sz]=p;   
 \_size[\_sz]=\_cnt[\_sz]=1;   
 return;   
 }   
 while(\_a[now]!=p){   
 \_size[now]++;   
 if(p>\_a[now]){   
 if(\_s[now][1]==0){   
 \_a[++\_sz]=p;   
 setson(\_sz,now,1);   
 }   
 now=\_s[now][1];   
 }   
 else{   
 if(\_s[now][0]==0){   
 \_a[++\_sz]=p;   
 setson(\_sz,now,0);   
 }   
 now=\_s[now][0];   
 }   
 }   
 \_size[now]++; \_cnt[now]++;   
 splay(now);   
}

## treap比x大的数有多少个

#include<bits/stdc++.h>   
using namespace std;   
typedef long long ll;   
   
#define fastio ios::sync\_with\_stdio(false); cin.tie(0);   
const int N = 2500 + 5;   
   
struct Point{   
 int x,y;   
} p[N];   
   
bool cmp1 (Point a, Point b) {   
 return a.y < b.y;   
}   
   
bool cmp2 (Point a, Point b) {   
 return a.x < b.x;   
}   
   
void redirect() {   
 #ifdef LOCAL   
 freopen("in.txt","r",stdin);   
 freopen("out.txt","w",stdout);   
 #endif   
}   
   
struct treap {   
 int l[N], r[N], val[N], rnd[N], size[N], w[N];   
 int sz, ans, rt;   
 inline void pushup(int x) { size[x] = size[l[x]] + size[r[x]] + w[x]; }   
 void lrotate(int &k) {   
 int t = r[k];   
 r[k] = l[t];   
 l[t] = k;   
 size[t] = size[k];   
 pushup(k);   
 k = t;   
 }   
 void rrotate(int &k) {   
 int t = l[k];   
 l[k] = r[t];   
 r[t] = k;   
 size[t] = size[k];   
 pushup(k);   
 k = t;   
 }   
 void insert(int &k, int x) {   
 if (!k) {   
 sz++;   
 k = sz;   
 size[k] = 1;   
 w[k] = 1;   
 val[k] = x;   
 rnd[k] = rand();   
 return;   
 }   
 size[k]++;   
 if (val[k] == x) {   
 w[k]++;   
 } else if (val[k] < x) {   
 insert(r[k], x);   
 if (rnd[r[k]] < rnd[k]) lrotate(k);   
 } else {   
 insert(l[k], x);   
 if (rnd[l[k]] < rnd[k]) rrotate(k);   
 }   
 }   
   
 void del(int &k, int x) {   
 if (!k) return;   
 if (val[k] == x) {   
 if (w[k] > 1) {   
 w[k]--;   
 size[k]--;   
 return;   
 }   
 if (l[k] == 0 || r[k] == 0)   
 k = l[k] + r[k];   
 else if (rnd[l[k]] < rnd[r[k]]) {   
 rrotate(k);   
 del(k, x);   
 } else {   
 lrotate(k);   
 del(k, x);   
 }   
 } else if (val[k] < x) {   
 size[k]--;   
 del(r[k], x);   
 } else {   
 size[k]--;   
 del(l[k], x);   
 }   
 }   
   
 int queryrank(int k, int x) {   
 if (!k) return 0;   
 if (val[k] == x)   
 return size[l[k]] + 1;   
 else if (x > val[k]) {   
 return size[l[k]] + w[k] + queryrank(r[k], x);   
 } else   
 return queryrank(l[k], x);   
 }   
   
 int querynum(int k, int x) {   
 if (!k) return 0;   
 if (x <= size[l[k]])   
 return querynum(l[k], x);   
 else if (x > size[l[k]] + w[k])   
 return querynum(r[k], x - size[l[k]] - w[k]);   
 else   
 return val[k];   
 }   
   
 void querypre(int k, int x) {   
 if (!k) return;   
 if (val[k] < x)   
 ans = k, querypre(r[k], x);   
 else   
 querypre(l[k], x);   
 }   
   
 void querysub(int k, int x) {   
 if (!k) return;   
 if (val[k] > x)   
 ans = k, querysub(l[k], x);   
 else   
 querysub(r[k], x);   
 }   
} T[N];   
   
map<int, int> mpx;   
map<int, int> mpy;   
   
ll check(int i,int j){   
 int l = min(p[i].y,p[j].y), r = max(p[i].y,p[j].y);   
 T[i].insert(T[i].rt, p[j].y);   
 ll lcnt = T[i].queryrank(T[i].rt, l), rcnt = (j - i + 1) - T[i].queryrank(T[i].rt, r) + 1;   
 return lcnt\*rcnt;   
}   
   
int main(){   
 fastio;   
 redirect();   
 srand(unsigned(time(NULL)));   
 ll ans=0;   
 int n; cin >> n;   
 for(int i=0;i<n;i++){   
 cin>>p[i].x>>p[i].y;   
 }   
 sort (p, p + n, cmp1);   
 for (int i=0; i<n; i++) mpy[p[i].y] = i;   
   
 sort (p, p + n, cmp2);   
 for (int i=0; i<n; i++) mpx[p[i].x] = i;   
   
 for (int i=0; i<n; i++) {   
 p[i].x = mpx[p[i].x];   
 p[i].y = mpy[p[i].y];   
 }   
   
 for(int i=0;i<n;i++){   
 for(int j=i; j<n; j++){   
 ans += check(i,j);   
 }   
 }   
 cout<<ans+1<<endl;   
}

## trie思想建树

#include <bits/stdc++.h>   
// codeforces 1416C XOR Inverse   
   
#define mp make\_pair   
#define pb push\_back   
#define f first   
#define s second   
#define ll long long   
#define forn(i, a, b) for(int i = (a); i <= (b); ++i)   
#define forev(i, b, a) for(int i = (b); i >= (a); --i)   
#define VAR(v, i) \_\_typeof( i) v=(i)   
#define forit(i, c) for(VAR(i, (c).begin()); i != (c).end(); ++i)   
#define all(x) (x).begin(), (x).end()   
#define sz(x) ((int)(x).size())   
#define file(s) freopen(s".in","r",stdin); freopen(s".out","w",stdout);   
   
using namespace std;   
   
const int maxn = (int)5e6 + 100;   
const int maxm = (int)1e6 + 100;   
const int mod = (int)1e9 + 7;   
const int P = (int) 1e6 + 7;   
const double pi = acos(-1.0);   
   
#define inf mod   
   
typedef long double ld;   
typedef pair<int, int> pii;   
typedef pair<ll, ll> pll;   
typedef vector<int> vi;   
typedef vector<ll> Vll;   
typedef vector<pair<int, int> > vpii;   
typedef vector<pair<ll, ll> > vpll;   
   
int n, t[2][maxn], id = 1;   
ll dp[2][30];   
vi g[maxn];   
   
void add(int x, int pos){   
 int v = 0;   
 forev(i, 29, 0){   
 int bit = ((x >> i) & 1);   
 if(!t[bit][v]) t[bit][v] = id++;   
 v = t[bit][v];   
 g[v].pb(pos);   
 }   
}   
void go(int v, int b = 29){   
 int l = t[0][v], r = t[1][v];   
 if(l) go(l, b - 1);   
 if(r) go(r, b - 1);   
 if(!l || !r) return;   
 ll res = 0;   
 int ptr = 0;   
 for(auto x : g[l]){   
 while(ptr < sz(g[r]) && g[r][ptr] < x) ptr++;   
 res += ptr;   
 }   
 dp[0][b] += res;   
 dp[1][b] += sz(g[l]) \* 1ll \* sz(g[r]) - res;   
}   
void solve(){   
 scanf("%d", &n);   
 forn(i, 1, n){   
 int x;   
 scanf("%d", &x);   
 add(x, i);   
 }   
 go(0);   
 ll inv = 0;   
 int res = 0;   
 forn(i, 0, 29){   
 inv += min(dp[0][i], dp[1][i]);   
 if(dp[1][i] < dp[0][i])   
 res += (1 << i);   
 }   
 printf("%lld %d", inv, res);   
}   
   
int main () {   
 int t = 1;   
 //scanf("%d", &t);   
 while(t--) solve();   
}

# 数学

## 埃筛

//埃氏筛法   
#define N 10000   
int flag[N+1],p[N+1],pnum;   
/\*   
flag[n] 表示n是否是素数，1是素数，0不是   
prime 中是所有的素数按从小到大排列、   
pnum 表示素数的个数   
\*/   
void CreatePrime(){   
 pnum=0;//初始化没有素数   
 //先将所有数看做素数，然后开始筛选   
 for(int i=0; i<=N; i++){   
 flag[i]=1;   
 }   
 //遍历筛去所有最大因数是i的合数   
 for(int i=2; i<=N; i++){   
 if(flag[i]==1){   
 //把素数记录下来   
 p[pnum++]=i;   
 }   
 //遍历已知素数表中比i的最小素因数小的素数，并筛去合数   
 for(int j=0; j<pnum && p[j]\*i<=N; j++){   
 //筛去合数   
 flag[p[j]\*i]=0;   
 if(i%p[j]==0)   
 //找到i的最小素因数   
 break;   
 }   
 }   
}

## 大素数判定+泼辣的肉

#include<iostream>  
#include<cstdio>  
#include<cstring>  
#include<algorithm>  
#include<cstdlib>  
using namespace std;  
typedef long long ll;  
   
const int S=20;  
  
long long mult\_mod(long long a,long long b,long long c)  
{  
 a%=c;  
 b%=c;  
 long long ret=0;  
 while(b)  
 {  
 if(b&1){ret+=a;ret%=c;}  
 a<<=1;  
 if(a>=c)a%=c;  
 b>>=1;  
 }  
 return ret;  
}  
  
long long pow\_mod(long long x,long long n,long long mod)  
{  
 if(n==1)return x%mod;  
 x%=mod;  
 long long tmp=x;  
 long long ret=1;  
 while(n)  
 {  
 if(n&1) ret=mult\_mod(ret,tmp,mod);  
 tmp=mult\_mod(tmp,tmp,mod);  
 n>>=1;  
 }  
 return ret;  
}  
  
bool check(long long a,long long n,long long x,long long t)  
{  
 long long ret=pow\_mod(a,x,n);  
 long long last=ret;  
 for(int i=1;i<=t;i++)  
 {  
 ret=mult\_mod(ret,ret,n);  
 if(ret==1&&last!=1&&last!=n-1) return true;//合数  
 last=ret;  
 }  
 if(ret!=1) return true;  
 return false;  
}  
   
  
   
bool Miller\_Rabin(long long n)  
{  
 if(n<2)return false;  
 if(n==2)return true;  
 if((n&1)==0) return false;  
 long long x=n-1;  
 long long t=0;  
 while((x&1)==0){x>>=1;t++;}  
 for(int i=0;i<S;i++)  
 {  
 long long a=rand()%(n-1)+1;  
 if(check(a,n,x,t))  
 return false;  
 }  
 return true;  
}  
   
long long factor[100];  
int tol;  
   
long long gcd(long long a,long long b)  
{  
 if(a==0)return 1;//???????  
 if(a<0) return gcd(-a,b);  
 while(b)  
 {  
 long long t=a%b;  
 a=b;  
 b=t;  
 }  
 return a;  
}  
   
long long Pollard\_rho(long long x,long long c)  
{  
 long long i=1,k=2;  
 long long x0=rand()%x;  
 long long y=x0;  
 while(1)  
 {  
 i++;  
 x0=(mult\_mod(x0,x0,x)+c)%x;  
 long long d=gcd(y-x0,x);  
 if(d!=1&&d!=x) return d;  
 if(y==x0) return x;  
 if(i==k){y=x0;k+=k;}  
 }  
}  
  
void findfac(long long n)  
{  
 if(Miller\_Rabin(n))  
 {  
 factor[tol++]=n;  
 return;  
 }  
 long long p=n;  
 while(p>=n){  
 if (Pollard\_rho(p, rand()%(n-1)+1)!=0) p=Pollard\_rho(p,rand()%(n-1)+1);  
 }  
 findfac(p);  
 findfac(n/p);  
}  
   
int main(void)  
{  
 int t;  
 cin >> t;  
 while(t--)  
 {  
 ll n;  
 scanf("%lld", &n);  
 if(Miller\_Rabin(n)) printf("%lld\n", n);  
 else  
 {  
 tol = 0;  
 findfac(n);  
 ll ans = factor[0];  
 for(int i = 1; i < tol; i++)  
 ans = min(ans, factor[i]);  
 printf("%lld\n", ans);  
 }  
 }  
 return 0;  
}

## 第几个质数

//G++ 1560ms 6544k  
#include <bits/stdc++.h>  
#define ll long long  
using namespace std;  
ll f[340000],g[340000],n;  
void init(){  
 ll i,j,m;  
 for(m=1;m\*m<=n;++m)f[m]=n/m-1;  
 for(i=1;i<=m;++i)g[i]=i-1;  
 for(i=2;i<=m;++i){  
 if(g[i]==g[i-1])continue;  
 for(j=1;j<=min(m-1,n/i/i);++j){  
 if(i\*j<m)f[j]-=f[i\*j]-g[i-1];  
 else f[j]-=g[n/i/j]-g[i-1];  
 }  
 for(j=m;j>=i\*i;--j)g[j]-=g[j/i]-g[i-1];  
 }  
}  
int main(){  
 while(scanf("%I64d",&n)!=EOF){  
 init();  
 cout<<f[1]<<endl;  
 }  
 return 0;  
}  
/\*  
  
O(n^3/4) 筛一个大质数是第几个质数  
疑似 Meisell-Lehmer算法  
  
\*/

## 费马小定理

## 高精度

#include<iostream>  
#include<string>  
#include<cstring>  
#include<cstdio>  
using namespace std;  
const int N = 1005;  
struct bign  
{  
 int len,s[N];  
 bign() { memset(s,0,sizeof(s)); len=1; }  
 bign(int num) { \*this=num; }  
 bign(char \*num) { \*this=num; }  
 bign operator =(int num)  
 {  
 char c[N];  
 sprintf(c,"%d",num);  
 \*this=c;  
 return \*this;  
 }  
 bign operator =(const char \*num)  
 {  
 len=strlen(num);  
 for (int i=0;i<len;i++) s[i]=num[len-1-i]-'0';  
 return \*this;  
 }  
 string str()  
 {  
 string res="";  
 for (int i=0;i<len;i++) res=(char)(s[i]+'0')+res;  
 return res;  
 }  
 void clean()  
 {  
 while (len>1&&!s[len-1]) len--;  
 }  
 bign operator +(const bign &b)  
 {  
 bign c;   
 c.len=0;  
 for (int i=0,g=0;g||i<len||i<b.len;i++)  
 {  
 int x=g;  
 if (i<len) x+=s[i];  
 if (i<b.len) x+=b.s[i];  
 c.s[c.len++]=x%10;  
 g=x/10;  
 }  
 return c;  
 }  
 bign operator -(const bign &b)  
 {  
 bign c;  
 c.len=0;  
 int x;   
 for (int i=0,g=0;i<len;i++)  
 {  
 x=s[i]-g;  
 if (i<b.len) x-=b.s[i];  
 if (x>=0) g=0;  
 else{   
 x+=10;  
 g=1;  
 };  
 c.s[c.len++]=x;  
 }  
 c.clean();  
 return c;  
 }  
 bign operator \*(const bign &b)  
 {  
 bign c;  
 c.len=len+b.len;  
 for (int i=0;i<len;i++) for (int j=0;j<b.len;j++) c.s[i+j]+=s[i]\*b.s[j];  
 for (int i=0;i<c.len-1;i++) { c.s[i+1]+=c.s[i]/10; c.s[i]%=10; }  
 c.clean();  
 return c;   
 }  
 bool operator <(const bign &b)  
 {  
 if (len!=b.len) return len<b.len;  
 for (int i=len-1;i>=0;i--)  
 if (s[i]!=b.s[i]) return s[i]<b.s[i];  
 return false;  
 }  
 bign operator +=(const bign &b)  
 {  
 \*this=\*this+b;  
 return \*this;  
 }  
 bign operator -=(const bign &b)  
 {  
 \*this=\*this-b;  
 return \*this;  
 }   
};  
istream& operator >>(istream &in,bign &x)  
{  
 string s;  
 in>>s;  
 x=s.c\_str();  
 return in;  
}  
ostream& operator <<(ostream &out,bign &x)  
{  
 out<<x.str();  
 return out;  
}  
int main(){  
 bign a,b,c;  
 ios::sync\_with\_stdio(false);  
 cin>>a>>b;  
// cout<<a<<endl;  
// cout<<b<<endl;  
 c=a+b;  
 cout<<c<<endl;  
 return 0;  
}

## 高精度除法

#include<iostream>  
#include<algorithm>  
using namespace std;  
string div(string a,int b)//高精度a除以单精度b  
{  
 string r,ans;  
 int d=0;  
 if(a=="0") return a;//特判  
 for(int i=0;i<a.size();i++)  
 {  
 r+=(d\*10+a[i]-'0')/b+'0';//求出商  
 d=(d\*10+(a[i]-'0'))%b;//求出余数  
 }  
 int p=0;  
 for(int i=0;i<r.size();i++)  
 if(r[i]!='0') {p=i;break;}  
 return r.substr(p);  
}  
int main()  
{  
 string a;  
 int b;  
 while(cin>>a>>b)  
 {  
 cout<<div(a,b)<<endl;  
 }  
 return 0;  
}

## 高斯-约旦消元

int n;  
double matrix[N][N];  
double ans[N];  
  
bool Gauss() {  
 for (int i=1; i<=n; ++i) {   
 //枚举列（项）   
 int mx=i;  
 for (int j=i+1; j<=n; ++j) {  
 //选出该列最大系数   
 if ( fabs(matrix[j][i]) > fabs(matrix[mx][i]) ) {  
 //fabs是取浮点数的绝对值的函数  
 mx = j;  
 }  
 }  
 for (int j=1; j<=n+1; ++j) {  
 //交换  
 swap( matrix[i][j], matrix[mx][j] );  
 }  
  
 if (!matrix[i][i]) {  
 //最大值等于0则说明该列都为0，肯定无解   
 // puts("No Solution");  
 return false;  
 }  
  
 for(int j=1; j<=n; ++j) {  
 //每一项都减去一个数（就是小学加减消元）  
 if(j != i) {  
 double temp = matrix[j][i] / matrix[i][i];  
 for(int k=i+1;k<=n+1;++k) {  
 matrix[j][k] -= matrix[i][k]\*temp;  
 }  
 }  
 }  
 }  
 //上述操作结束后，矩阵会变成这样  
 /\*  
 k1\*a=e1  
 k2\*b=e2  
 k3\*c=e3  
 k4\*d=e4  
 \*/  
 //所以输出的结果要记得除以该项系数，消去常数  
 for(int i=1;i<=n;++i) {  
 ans[i] = matrix[i][n+1] / matrix[i][i];  
 if ( fabs(ans[i] - 0) < eps ) ans[i] = 0;  
 // printf("%.2lf\n",matrix[i][n+1]/matrix[i][i]);  
 }  
  
 return true;  
}

## 矩阵快速幂

#include <bits/stdc++.h>  
using namespace std;  
  
long long T,a,b,c,pp,mod;  
long long n;  
  
struct mat{  
 long long m[4][4];  
};  
  
mat mul(mat a,mat b){  
 mat ans;int i,j,k;  
 for(i=1;i<=3;i++)  
 for(j=1;j<=3;j++)  
 ans.m[i][j]=0;  
 for(i=1;i<=3;i++)  
 for(j=1;j<=3;j++)  
 for(k=1;k<=3;k++)  
 ans.m[i][j]=(ans.m[i][j]+a.m[i][k]\*b.m[k][j])%mod;  
 return ans;  
}  
  
mat matqp(mat t,long long p)  
{  
 mat ans;  
 int i,j;  
 for(i=1;i<=3;i++)  
 for(j=1;j<=3;j++)  
 if(i==j)ans.m[i][j]=1;  
 else ans.m[i][j]=0;  
 while(p)  
 {  
 if(p&1)  
 ans=mul(ans,t);  
 t=mul(t,t);  
 p=p>>1;  
 }  
 return ans;  
}  
  
long long qp(long long a,long long p)  
{  
 long long ans=1;  
 while(p){  
 if(p&1) {ans\*=a;ans%=pp;}  
 a=a\*a; a%=pp;  
 p=p>>1;  
 }  
 return ans;  
}  
  
int main(){  
 //scanf("%d",&T);  
 cin>>T;  
 while(T--)  
 {  
 //scanf("%I64d %d %d %d %d",&n,&a,&b,&c,&pp);  
 cin>>n>>a>>b>>c>>pp;  
 ///\*  
 mod=pp-1;  
 //\*/  
 mat base;  
 for(int i=1;i<=3;i++)  
 for(int j=1;j<=3;j++)  
 base.m[i][j]=0;  
 base.m[1][1]=c;base.m[1][2]=1;base.m[1][3]=1;base.m[2][1]=1;base.m[3][3]=1;  
 if(n==1){  
 cout<<1<<endl;  
 }  
 else{  
 mat out = matqp(base,n-2);  
 long long res = out.m[1][1]\*b%mod + out.m[1][3]\*b%mod;  
 //cout<<res<<endl;  
 long long ans = qp(a,res);  
 cout<<ans<<endl;  
 }  
 }  
  
 return 0;  
}

## 扩展欧几里得

int extend\_gcd( int a, int b, int &x, int &y ) {  
 if(b==0){  
 x=1;y=0;  
 return a;  
 }else{  
 int r = extend\_gcd(b,a%b,y,x);  
 y-=x\*(a/b);  
 return r;  
 }  
}

## 欧拉函数

int phi(int x)  
{  
 int 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;  
}

## 欧拉筛

void init() {  
 phi[1] = 1;  
 for (int i = 2; i < MAXN; ++i) {  
 if (!vis[i]) {  
 phi[i] = i - 1;  
 pri[cnt++] = i;  
 }  
 for (int j = 0; j < cnt; ++j) {  
 if (1ll \* i \* pri[j] >= MAXN) break;  
 vis[i \* pri[j]] = 1;  
 if (i % pri[j]) {  
 phi[i \* pri[j]] = phi[i] \* (pri[j] - 1);  
 } else {  
 phi[i \* pri[j]] = phi[i] \* pri[j];  
 break;  
 }  
 }  
 }  
}

## 线性基

#include <bits/stdc++.h>  
#define N 51  
#define ll long long  
using namespace std;  
  
//给n个数，输出n个数里异或和的最大值  
  
int n;  
ll ans;  
ll a[N], p[101];  
  
inline ll read()  
{  
 char ch = getchar();  
 ll x = 0, f = 1;  
 while(ch > '9' || ch < '0')  
 {  
 if(ch == '-')  
 f = -1;  
 ch = getchar();  
 }  
 while(ch >= '0' && ch <= '9')  
 {  
 x = x \* 10 + ch - '0';  
 ch = getchar();  
 }  
 return x \* f;  
}  
  
void Get\_LB(ll x)  
{  
 for(int i = 62; i >= 0; i--)  
 {  
 if(!(x >> (ll)i))  
 continue;  
 if(!p[i])  
 {  
 p[i] = x;  
 break;  
 }  
 x ^= p[i];  
 }  
}  
  
int main()  
{  
 n = read();  
 for(int i = 1; i <= n; i++)  
 Get\_LB(a[i] = read());  
 for(int i = 62; i >= 0; i--)  
 if((ans ^ p[i]) > ans)  
 ans ^= p[i];  
 cout << ans;  
   
 return 0;  
}

## 圆和矩形的面积交

#include<bits/stdc++.h>  
using namespace std;   
#define INF 0x3f3f3f3f   
#define eps 1e-17   
#define pi acos(-1.0)   
typedef long long ll;   
  
void redirect() {  
 #ifdef LOCAL  
 freopen("1.in","r",stdin);  
 freopen("1.out","w",stdout);  
 #endif  
}  
  
int dcmp(double x){   
 if(fabs(x)<eps)return 0;   
 return x>0?1:-1;   
}   
struct Point{   
 double x,y;   
 Point(double \_x=0,double \_y=0){   
 x=\_x;y=\_y;   
 }   
};   
Point operator + (const Point &a,const Point &b){   
 return Point(a.x+b.x,a.y+b.y);   
}   
Point operator - (const Point &a,const Point &b){   
 return Point(a.x-b.x,a.y-b.y);   
}   
Point operator \* (const Point &a,const double &p){   
 return Point(a.x\*p,a.y\*p);   
}   
Point operator / (const Point &a,const double &p){   
 return Point(a.x/p,a.y/p);   
}   
bool operator < (const Point &a,const Point &b){   
 return a.x<b.x||(dcmp(a.x-b.x)==0&&a.y<b.y);   
}   
bool operator == (const Point &a,const Point &b){   
 return dcmp(a.x-b.x)==0&&dcmp(a.y-b.y)==0;   
}   
double Dot(Point a,Point b){   
 return a.x\*b.x+a.y\*b.y;   
}   
double Length(Point a){   
 return sqrt(Dot(a,a));   
}   
double Angle(Point a,Point b){   
 return acos(Dot(a,b)/Length(a)/Length(b));   
}   
double angle(Point a){   
 return atan2(a.y,a.x);   
}   
double Cross(Point a,Point b){   
 return a.x\*b.y-a.y\*b.x;   
}   
Point vecunit(Point a){   
 return a/Length(a);   
}   
Point Normal(Point a){   
 return Point(-a.y,a.x)/Length(a);   
}   
Point Rotate(Point a,double rad){   
 return Point(a.x\*cos(rad)-a.y\*sin(rad),a.x\*sin(rad)+a.y\*cos(rad));   
}   
double Area2(Point a,Point b,Point c){   
 return Length(Cross(b-a,c-a));   
}   
bool OnSegment(Point p,Point a1,Point a2){   
 return dcmp(Cross(a1-p,a2-p))==0&&dcmp(Dot(a1-p,a2-p))<=0;   
}   
struct Line{   
 Point p,v;   
 double ang;   
 Line(){};   
 Line(Point p,Point v):p(p),v(v){   
 ang=atan2(v.y,v.x);   
 }   
 bool operator < (const Line &L) const {   
 return ang<L.ang;   
 }   
 Point point(double d){   
 return p+(v\*d);   
 }   
};   
bool OnLeft(const Line &L,const Point &p){   
 return Cross(L.v,p-L.p)>=0;   
}   
Point GetLineIntersection(Point p,Point v,Point q,Point w){   
 Point u=p-q;   
 double t=Cross(w,u)/Cross(v,w);   
 return p+v\*t;   
}   
Point GetLineIntersection(Line a,Line b){   
 return GetLineIntersection(a.p,a.v,b.p,b.v);   
}   
double PolyArea(vector<Point> p){   
 int n=p.size();   
 double ans=0;   
 for(int i=1;i<n-1;i++)   
 ans+=Cross(p[i]-p[0],p[i+1]-p[0]);   
 return fabs(ans)/2;   
}   
struct Circle{   
 Point c;   
 double r;   
 Circle(){}   
 Circle(Point c, double r):c(c), r(r){}   
 Point point(double a) {//����Բ�Ľ��������   
 return Point(c.x+cos(a)\*r, c.y+sin(a)\*r);   
 }   
};   
   
bool InCircle(Point x,Circle c){   
 return dcmp(c.r-Length(c.c-x))>=0;   
}   
bool OnCircle(Point x,Circle c){   
 return dcmp(c.r-Length(c.c-x))==0;   
}   
int getSegCircleIntersection(Line L,Circle C,Point \*sol){   
 Point nor=Normal(L.v);   
 Line p1=Line(C.c,nor);   
 Point ip=GetLineIntersection(p1,L);   
 double dis=Length(ip-C.c);   
 if(dcmp(dis-C.r)>0)return 0;   
 Point dxy=vecunit(L.v)\*sqrt(C.r\*C.r-dis\*dis);   
 int ret=0;   
 sol[ret]=ip+dxy;   
 if(OnSegment(sol[ret],L.p,L.point(1)))ret++;   
 sol[ret]=ip-dxy;   
 if(OnSegment(sol[ret],L.p,L.point(1)))ret++;   
 return ret;   
}   
double SegCircleArea(Circle C,Point a,Point b){   
 double a1=angle(a-C.c);   
 double a2=angle(b-C.c);   
 double da=fabs(a1-a2);   
 if(da>pi)da=pi\*2-da;   
 return dcmp(Cross(b-C.c,a-C.c))\*da\*C.r\*C.r/2.0;   
}   
double PolyCircleArea(Circle C,Point \*p,int n){   
 double ret=0;   
 Point sol[2];   
 p[n]=p[0];   
 for(int i=0;i<n;i++){   
 double t1,t2;   
 int cnt=getSegCircleIntersection(Line(p[i],p[i+1]-p[i]),C,sol); //�ж��߶���Բ�м������㣬   
 if(cnt==0){ //0�����㣬�ж��߶��ڶ�����ڲ������ⲿ��   
 if(!InCircle(p[i],C)||!InCircle(p[i+1],C))ret+=SegCircleArea(C,p[i],p[i+1]); //�ⲿֱ�Ӽ���Բ�����   
 else ret+=Cross(p[i+1]-C.c,p[i]-C.c)/2; //�ڲ����������������   
 }   
 if(cnt==1){   
 if(InCircle(p[i],C)&&(!InCircle(p[i+1],C)||OnCircle(p[i+1],C)))ret+=Cross(sol[0]-C.c,p[i]-C.c)/2,ret+=SegCircleArea(C,sol[0],p[i+1]);//,cout<<"jj-1"<<endl;   
 else ret+=SegCircleArea(C,p[i],sol[0]),ret+=Cross(p[i+1]-C.c,sol[0]-C.c)/2;//,cout<<"jj-2"<<endl;   
 }   
 if(cnt==2){  
 if((p[i]<p[i+1])^(sol[0]<sol[1]))swap(sol[0],sol[1]);   
 ret+=SegCircleArea(C,p[i],sol[0]);   
 ret+=Cross(sol[1]-C.c,sol[0]-C.c)/2;   
 ret+=SegCircleArea(C,sol[1],p[i+1]);   
 }   
 }   
 return fabs(ret);   
}  
Point p[5];   
int main(){  
 redirect();  
 double R,x1,y1,x2,y2,x3,y3;   
 cin>>x1>>y1>>R>>x2>>y2>>x3>>y3;  
  
 Circle C=Circle(Point(x1,y1),R);   
 if(x2>x3)swap(x2,x3);   
 if(y2>y3)swap(y2,y3);   
 p[0]=Point(x2,y2);   
 p[2]=Point(x3,y3);   
 p[1]=Point(x3,y2);   
 p[3]=Point(x2,y3);   
 double ans=PolyCircleArea(C,p,4);   
 if(ans < -eps) ans = -ans;  
 printf("%.4lf\n",ans);   
   
 return 0;   
}

## Min25

/\*  
\* @ author: dragon\_bra  
\* @ email: tommy514@foxmail.com  
\* @ data: 2020-09-20 13:59  
\*/  
// n以内素数和  
#include <algorithm>  
#include <cmath>  
#include <cstdio>  
#include <cstdlib>  
#include <cstring>  
#include <iostream>  
#include <sstream>  
#include <map>  
#include <set>  
#include <queue>  
#include <vector>  
using namespace std;  
  
const int N = 2e5 + 10;  
  
typedef long long ll;  
  
void redirect() {  
 #ifdef LOCAL  
 freopen("in.txt","r",stdin);  
 freopen("out.txt","w",stdout);  
 #endif  
}  
  
int T; ll n, K;  
  
namespace Min25 {  
  
 ll prime[N], id1[N], id2[N], flag[N], ncnt, m;  
  
 ll g[N], sum[N], a[N], T, n;  
  
 inline int ID(ll x) {  
 return x <= T ? id1[x] : id2[n / x];  
 }  
  
 inline ll calc(ll x) {  
 if (x % 2) return (x+1)/2 % K \* x % K;  
 else return x/2 % K \* (x+1) % K;  
 // return x \* (x + 1) / 2 - 1;  
 }  
  
 inline ll f(ll x) {  
 return x;  
 }  
  
 inline void init() {  
 T = sqrt(n + 0.5);  
 ncnt = 0; m = 0;  
 memset(flag, 0, sizeof flag);  
 memset(sum, 0, sizeof sum);  
 memset(prime, 0, sizeof prime);  
 memset(a, 0, sizeof a);  
 for (int i = 2; i <= T; i++) {  
 if (!flag[i]) prime[++ncnt] = i, sum[ncnt] = (sum[ncnt - 1] + i)%K;  
 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]) % K;  
 }  
 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] + K) % K + K) % K;  
 }  
 }  
  
 inline ll solve(ll x) {  
 if (x <= 1) return x;  
 return n = x, init(), g[ID(n)];  
 }  
  
}  
  
int main() {  
 redirect();  
  
 scanf("%d", &T);  
 while (T--) {  
 scanf("%lld %lld", &n, &K);  
 n = n+1;  
 ll ans = 0;  
 if (n%2) {  
 ans = (n+1)/2 % K \* n % K;  
 } else {  
 ans = n/2 % K \* (n+1) % K;  
 }  
 ans += Min25::solve(n) - 5;  
 ans %= K;  
 printf("%lld\n", ans);  
 }  
}

## Zeller Formula

int Day(int year, int month, int day){  
 int ret = 0;  
 int c, y, m, d;  
 if(month <= 2){  
 c = ( year - 1 ) / 100;  
 y = ( year - 1 ) % 100;  
 m = month + 12;  
 d = day;  
 }  
 else{  
 c = year / 100;  
 y = year % 100;  
 m = month;  
 d = day;  
 }  
 ret = y + y / 4 + c / 4 - 2 \* c + 26 \* ( m + 1 ) / 10 + d - 1;  
 ret = ret >= 0 ? ( ret % 7 ) : ( ret % 7 + 7 );  
 return ret;  
}

# 图论

## 网络流

### 二分图最大流

const int maxn = 200005;  
const int INF = 0x3f3f3f3f;  
  
struct Edge  
{  
 int from, to, flow, cap;  
 Edge(int x, int y, int f, int c) : from(x), to(y), flow(f), cap(c) {}  
};  
  
vector<Edge> edges;  
vector<int> G[maxn];  
int cur[maxn], d[maxn];  
int S,T;  
int cnt;  
  
inline void addedge(int from, int to, int cap)  
{  
 edges.push\_back(Edge(from, to, 0, cap));  
 edges.push\_back(Edge(to, from, 0, 0));  
 int m = edges.size();  
 G[from].push\_back(m - 2);  
 G[to].push\_back(m - 1);  
}  
  
int dfs(int u, int a)  
{  
 if (u == T || a == 0)  
 {  
 return a;  
 }  
 int flow = 0, f;  
 for (int &i = cur[u]; i < G[u].size(); i++)  
 {  
 Edge &e = edges[G[u][i]];  
 if (d[e.to] > d[u] && (f = dfs(e.to, min(a, e.cap - e.flow))) > 0)  
 {  
 flow += f;  
 e.flow += f;  
 edges[G[u][i] ^ 1].flow -= f;  
 a -= f;  
 if (a == 0)  
 {  
 break;  
 }  
 }  
 }  
 if (a)  
 {  
 d[u] = -1;  
 }  
 return flow;  
}  
  
bool bfs()  
{  
 memset(d, -1, (T + 1) \* sizeof(int));  
 queue<int> q;  
 q.push(S);  
 d[S] = 0;  
 while (!q.empty())  
 {  
 int u = q.front();  
 q.pop();  
 for (int i = 0; i < G[u].size(); i++)  
 {  
 Edge &e = edges[G[u][i]];  
 if (d[e.to] == -1 && e.cap > e.flow)  
 {  
 d[e.to] = d[u] + 1;  
 q.push(e.to);  
 }  
 }  
 }  
 return d[T] != -1;  
}  
  
int max\_flow()  
{  
 int ans = 0;  
 while (bfs())  
 {  
 memset(cur, 0, (T+1)\*sizeof(int));  
 ans += dfs(S, INF);  
 }  
 return ans;  
}

### Dinic（Node版本）

//以下是网络流模板  
struct Edge{  
 int to,nxt,w;  
}e[M<<1];  
int head[N],ecnt;  
void AddEdge(int u,int v,int w) {  
 e[ecnt]=(Edge){v,head[u],w};  
 head[u]=ecnt++;  
}  
void Link(int u,int v,int w){ AddEdge(u,v,w),AddEdge(v,u,0); }  
#define erep(u,i) for(int i=head[u];~i;i=e[i].nxt)  
   
int dis[N];  
int Bfs(){  
 static queue <int> que;  
 rep(i,1,vc) dis[i]=INF;  
 que.push(S),dis[S]=0;  
 while(!que.empty()) {  
 int u=que.front(); que.pop();  
 erep(u,i) {  
 int v=e[i].to,w=e[i].w;  
 if(!w || dis[v]<=dis[u]+1) continue;  
 dis[v]=dis[u]+1,que.push(v);  
 }  
 }  
 return dis[T]<INF;  
}  
   
int Dfs(int u,int flowin) {  
 if(u==T) return flowin;  
 int flowout=0;  
 erep(u,i) {  
 int v=e[i].to,w=e[i].w;  
 if(dis[v]!=dis[u]+1 || !w) continue;  
 int t=Dfs(v,min(flowin-flowout,w));  
 flowout+=t,e[i].w-=t,e[i^1].w+=t;  
 if(flowin==flowout) break;  
 }  
 if(!flowout) dis[u]=0;  
 return flowout;  
}  
   
int Dinic(){  
 int ans=0;  
 while(Bfs()) ans+=Dfs(S,INF);  
 return ans;  
}

## 次小生成树

//AcWing 356. 次小生成树   
#include <bits/stdc++.h>   
using namespace std;   
   
typedef long long LL;   
   
const int N = 100010, M = 300010, INF = 0x3f3f3f3f;   
   
int n, m;   
struct Edge {   
 int a, b, w;   
 bool used;   
 bool operator< (const Edge &t) const {   
 return w < t.w;   
 }   
} edge[M];   
int p[N];   
int h[N], e[M], w[M], ne[M], idx;   
int depth[N], fa[N][17], d1[N][17], d2[N][17];   
int q[N];   
   
void add(int a, int b, int c) {   
 e[idx] = b, w[idx] = c, ne[idx] = h[a], h[a] = idx ++;   
}   
   
int find(int x) {   
 return p[x] == x ? x : p[x] = find(p[x]);   
}   
   
LL kruskal() {   
 for (int i = 1; i <= n; i ++ ) p[i] = i;   
 sort (edge, edge + m);   
   
 LL res = 0;   
 for (int i = 0; i < m; i ++ ) {   
 int a = find(edge[i].a), b = find(edge[i].b), w = edge[i].w;   
 if (a != b) {   
 p[a] = b;   
 res += w;   
 edge[i].used = true;   
 }   
 }   
   
 return res;   
}   
   
void build() {   
 memset(h, -1, sizeof h);   
 for (int i = 0; i < m; i ++ ) {   
 if (edge[i].used) {   
 int a = edge[i].a, b = edge[i].b, w = edge[i].w;   
 add(a, b, w); add(b, a, w);   
 }   
 }   
}   
   
void bfs() {   
 memset(depth, 0x3f, sizeof depth);   
 depth[0] = 0, depth[1] = 1;   
 q[0] = 1;   
 int hh = 0, tt = 0;   
 while (hh <= tt) {   
 int t = q[hh ++ ];   
 for (int i = h[t]; ~i; i = ne[i]) {   
 int j = e[i];   
 if (depth[j] > depth[t] + 1) {   
 depth[j] = depth[t] + 1;   
 q[ ++ tt] = j;   
 fa[j][0] = t;   
 d1[j][0] = w[i], d2[j][0] = -INF;   
 for (int k = 1; k <= 16; k ++ ) {   
 int anc = fa[j][k - 1];   
 fa[j][k] = fa[fa[j][k - 1]][k - 1];   
 int distance[4] = {d1[j][k - 1], d2[j][k - 1], d1[anc][k - 1], d2[anc][k - 1]};   
 d1[j][k] = d2[j][k] = -INF;   
 for (int u = 0; u < 4; u ++ ) {   
 int d = distance[u];   
 if (d > d1[j][k]) d2[j][k] = d2[j][k], d1[j][k] = d;   
 else if (d != d1[j][k] && d > d2[j][k]) d2[j][k] = d;   
 }   
 }   
 }   
 }   
 }   
}   
   
int lca(int a, int b, int w) {   
 static int distance[N \* 2];   
 int cnt = 0;   
 if (depth[a] < depth[b]) swap(a, b);   
 for (int k = 16; k >= 0; k -- ) {   
 if (depth[fa[a][k]] >= depth[b]) {   
 distance[cnt ++ ] = d1[a][k];   
 distance[cnt ++ ] = d2[a][k];   
 a = fa[a][k];   
 }   
 }   
 if (a != b) {   
 for (int k = 16; k >= 0; k -- ) {   
 if (fa[a][k] != fa[b][k]) {   
 distance[cnt ++ ] = d1[a][k];   
 distance[cnt ++ ] = d2[a][k];   
 distance[cnt ++ ] = d1[b][k];   
 distance[cnt ++ ] = d2[b][k];   
 a = fa[a][k], b = fa[b][k];   
 }   
 }   
 distance[cnt ++ ] = d1[a][0];   
 distance[cnt ++ ] = d1[b][0];   
 }   
   
 int dist1 = -INF, dist2 = -INF;   
 for (int i = 0; i < cnt; i ++ ) {   
 int d = distance[i];   
 if (d > dist1) dist2 = dist1, dist1 = d;   
 else if (d != dist1 && d > dist2) dist2 = d;   
 }   
   
 if (w > dist1) return w - dist1;   
 if (w > dist2) return w - dist2;   
 return INF;   
}   
   
int main() {   
 cin >> n >> m;   
 for (int i = 0; i < m; i ++ ) {   
 int a, b, c;   
 cin >> a >> b >> c;   
 edge[i] = {a, b, c};   
 }   
   
 LL sum = kruskal();   
 build();   
   
 bfs(); // 倍增初始化部分   
   
 LL res = 1e18 + 10;   
 for (int i = 0; i < m; i ++ ) {   
 if (!edge[i].used) {   
 int a = edge[i].a, b = edge[i].b, w = edge[i].w;   
 res = min(res, sum + lca(a, b, w));   
 }   
 }   
   
 cout << res << "\n";   
}//AcWing 356. 次小生成树   
#include <bits/stdc++.h>   
using namespace std;   
   
typedef long long LL;   
   
const int N = 100010, M = 300010, INF = 0x3f3f3f3f;   
   
int n, m;   
struct Edge {   
 int a, b, w;   
 bool used;   
 bool operator< (const Edge &t) const {   
 return w < t.w;   
 }   
} edge[M];   
int p[N];   
int h[N], e[M], w[M], ne[M], idx;   
int depth[N], fa[N][17], d1[N][17], d2[N][17];   
int q[N];   
   
void add(int a, int b, int c) {   
 e[idx] = b, w[idx] = c, ne[idx] = h[a], h[a] = idx ++;   
}   
   
int find(int x) {   
 return p[x] == x ? x : p[x] = find(p[x]);   
}   
   
LL kruskal() {   
 for (int i = 1; i <= n; i ++ ) p[i] = i;   
 sort (edge, edge + m);   
   
 LL res = 0;   
 for (int i = 0; i < m; i ++ ) {   
 int a = find(edge[i].a), b = find(edge[i].b), w = edge[i].w;   
 if (a != b) {   
 p[a] = b;   
 res += w;   
 edge[i].used = true;   
 }   
 }   
   
 return res;   
}   
   
void build() {   
 memset(h, -1, sizeof h);   
 for (int i = 0; i < m; i ++ ) {   
 if (edge[i].used) {   
 int a = edge[i].a, b = edge[i].b, w = edge[i].w;   
 add(a, b, w); add(b, a, w);   
 }   
 }   
}   
   
void bfs() {   
 memset(depth, 0x3f, sizeof depth);   
 depth[0] = 0, depth[1] = 1;   
 q[0] = 1;   
 int hh = 0, tt = 0;   
 while (hh <= tt) {   
 int t = q[hh ++ ];   
 for (int i = h[t]; ~i; i = ne[i]) {   
 int j = e[i];   
 if (depth[j] > depth[t] + 1) {   
 depth[j] = depth[t] + 1;   
 q[ ++ tt] = j;   
 fa[j][0] = t;   
 d1[j][0] = w[i], d2[j][0] = -INF;   
 for (int k = 1; k <= 16; k ++ ) {   
 int anc = fa[j][k - 1];   
 fa[j][k] = fa[fa[j][k - 1]][k - 1];   
 int distance[4] = {d1[j][k - 1], d2[j][k - 1], d1[anc][k - 1], d2[anc][k - 1]};   
 d1[j][k] = d2[j][k] = -INF;   
 for (int u = 0; u < 4; u ++ ) {   
 int d = distance[u];   
 if (d > d1[j][k]) d2[j][k] = d2[j][k], d1[j][k] = d;   
 else if (d != d1[j][k] && d > d2[j][k]) d2[j][k] = d;   
 }   
 }   
 }   
 }   
 }   
}   
   
int lca(int a, int b, int w) {   
 static int distance[N \* 2];   
 int cnt = 0;   
 if (depth[a] < depth[b]) swap(a, b);   
 for (int k = 16; k >= 0; k -- ) {   
 if (depth[fa[a][k]] >= depth[b]) {   
 distance[cnt ++ ] = d1[a][k];   
 distance[cnt ++ ] = d2[a][k];   
 a = fa[a][k];   
 }   
 }   
 if (a != b) {   
 for (int k = 16; k >= 0; k -- ) {   
 if (fa[a][k] != fa[b][k]) {   
 distance[cnt ++ ] = d1[a][k];   
 distance[cnt ++ ] = d2[a][k];   
 distance[cnt ++ ] = d1[b][k];   
 distance[cnt ++ ] = d2[b][k];   
 a = fa[a][k], b = fa[b][k];   
 }   
 }   
 distance[cnt ++ ] = d1[a][0];   
 distance[cnt ++ ] = d1[b][0];   
 }   
   
 int dist1 = -INF, dist2 = -INF;   
 for (int i = 0; i < cnt; i ++ ) {   
 int d = distance[i];   
 if (d > dist1) dist2 = dist1, dist1 = d;   
 else if (d != dist1 && d > dist2) dist2 = d;   
 }   
   
 if (w > dist1) return w - dist1;   
 if (w > dist2) return w - dist2;   
 return INF;   
}   
   
int main() {   
 cin >> n >> m;   
 for (int i = 0; i < m; i ++ ) {   
 int a, b, c;   
 cin >> a >> b >> c;   
 edge[i] = {a, b, c};   
 }   
   
 LL sum = kruskal();   
 build();   
   
 bfs(); // 倍增初始化部分   
   
 LL res = 1e18 + 10;   
 for (int i = 0; i < m; i ++ ) {   
 if (!edge[i].used) {   
 int a = edge[i].a, b = edge[i].b, w = edge[i].w;   
 res = min(res, sum + lca(a, b, w));   
 }   
 }   
   
 cout << res << "\n";   
}

## 二分图匹配-匈牙利算法

/\*   
Problem: HDU 2063 过山车 匈牙利算法-二分图匹配模板题   
\* @ author: dragon\_bra   
\* @ email: tommy514@foxmail.com   
\* @ date: 2021-01-26 22:11   
\*/   
   
#include <bits/stdc++.h>   
#define fastio ios::sync\_with\_stdio(false); cin.tie(0);   
using namespace std;   
   
typedef long long ll;   
const int N = 500 + 10;   
   
void redirect() {   
 #ifdef LOCAL   
 freopen("in.txt","r",stdin);   
 freopen("out.txt","w",stdout);   
 #endif   
}   
   
int k, m, n;   
int line[N][N], used[N], nxt[N];   
   
bool Find(int x) {   
 for (int i=1; i<=m; i++) {   
 if (line[x][i] && !used[i]) {   
 used[i] = 1;   
 if (nxt[i] == 0 || Find(nxt[i])) {   
 nxt[i] = x;   
 return true;   
 }   
 }   
 }   
 return false;   
}   
   
int match() {   
 int sum = 0;   
 for (int i=1; i<=n; i++) {   
 memset(used, 0, sizeof(used));   
 if (Find(i)) sum ++;   
 }   
 return sum;   
}   
   
int main() {   
 redirect();   
   
 while (cin >> k && k) {   
 memset(line, 0, sizeof(line));   
 memset(nxt, 0, sizeof(nxt));   
 cin >> n >> m;   
 for (int i=1; i<=k; i++) {   
 int u, v;   
 cin >> u >> v;   
 line[u][v] = true;   
 }   
 cout << match() << "\n";   
 }   
   
 return 0;   
}

## dijkstra

// Problem: C. Dijkstra?   
// Contest: Codeforces - Codeforces Alpha Round #20 (Codeforces format)   
// URL: https://codeforces.com/problemset/problem/20/C   
// Memory Limit: 64 MB   
// Time Limit: 1000 ms   
// Powered by CP Editor (https://github.com/cpeditor/cpeditor)   
   
/\*   
 @ author: dragon\_bra   
 @ QQ: 1277037638   
 @ email: tommy514@foxmail.com   
\*/   
   
#include <bits/stdc++.h>   
#define fastio ios\_base::sync\_with\_stdio(false); cin.tie(0);   
using namespace std;   
   
typedef long long ll;   
const ll INF = 1e18;   
const int N = 2e5 + 10;   
   
int n, m;   
struct edge {   
 int v; ll w;   
 edge(int v, ll w):v(v), w(w){}   
};   
vector<edge> G[N];   
struct node {   
 int u; ll dis;   
 node(int u, ll dis):u(u), dis(dis){}   
 friend bool operator<(node a, node b) {   
 return a.dis > b.dis;   
 }   
};   
ll dis[N];   
ll f[N];   
bool vis[N];   
int ans[N];   
   
void init() {   
 for (int i=1; i<=n; i++) {   
 dis[i] = INF;   
 vis[i] = false;   
 }   
}   
   
int main() {   
   
 fastio;   
 cin >> n >> m;   
   
 init();   
   
 for (int i=1; i<=m; i++) {   
 int u, v; ll w;   
 cin >> u >> v >> w;   
 G[u].push\_back(edge(v, w));   
 G[v].push\_back(edge(u, w));   
 }   
   
 priority\_queue<node> Q; Q.push(node(1, 0)); dis[1] = 0;   
 while (!Q.empty()) {   
 node now = Q.top(); Q.pop();   
 int u = now.u; ll d = now.dis;   
 if (vis[u]) continue;   
 vis[u] = true;   
 // cout << u << ' ' << d << endl;   
 for (auto nxt: G[u]) {   
 int v = nxt.v; ll w = nxt.w;   
 if (d + w < dis[v]) {   
 dis[v] = d + w;   
 f[v] = u;   
 Q.push(node(v, dis[v]));   
 }   
 }   
 }   
   
 int cnt = 0; int x = n;   
 while (x != 1) {   
 if (f[x] == 0) break;   
 ans[++cnt] = x;   
 x = f[x];   
 }   
 if (cnt == 0) {   
 puts("-1");   
 } else {   
 ans[++cnt] = 1;   
 for (int i=cnt; i>=1; i--) {   
 cout << ans[i] << ' ';   
 }   
 }   
   
 return 0;   
}

## LCA

### LCA-倍增

/\*   
 洛谷P3379，LCA模板   
\*/   
#include <bits/stdc++.h>   
using namespace std;   
   
const int N = 5e5 + 10, M = N \* 2;   
const int LOG = 30 + 1;   
   
int n, m;   
int h[N], e[M], ne[M], idx;   
int depth[N], fa[N][LOG];   
int q[N];   
   
void add(int a, int b) {   
 e[idx] = b, ne[idx] = h[a], h[a] = idx ++;   
}   
   
void bfs(int root) {   
 memset(depth, 0x3f3f3f3f, sizeof depth);   
 depth[0] = 0, depth[root] = 1;   
 int hh = 0, tt = 0;   
 q[0] = root;   
 while (hh <= tt) {   
 int t = q[hh ++ ];   
 for (int i = h[t]; ~i; i = ne[i] ) {   
 int j = e[i];   
 if (depth[j] > depth[t] + 1) {   
 depth[j] = depth[t] + 1;   
 q[ ++ tt] = j;   
 fa[j][0] = t;   
 for (int k = 1; k < LOG; k ++ )   
 fa[j][k] = fa[fa[j][k - 1]][k - 1];   
 }   
 }   
 }   
}   
   
int lca(int a, int b) {   
 if (depth[a] < depth[b]) swap(a, b);   
 for (int k = LOG - 1; k >= 0; k -- ) {   
 if (depth[fa[a][k]] >= depth[b]) // 哨兵解决depth['0'] = '0' 满足不成立的条件   
 a = fa[a][k];   
 }   
   
 if (a == b) return a;   
 for (int k = LOG - 1; k >= 0; k -- ) {   
 if (fa[a][k] != fa[b][k]) { // 哨兵解决跳出去后   
 a = fa[a][k];   
 b = fa[b][k];   
 }   
 }   
 return fa[a][0];   
}   
   
int main() {   
 #ifdef LOCAL   
 freopen("in.txt","r",stdin);   
 freopen("out.txt","w",stdout);   
 #endif   
 int root = 0;   
 cin >> n >> m >> root;   
 memset(h, -1, sizeof h);   
   
 for (int i = 1; i < n; i ++ ) {   
 int a, b;   
 scanf("%d%d", &a, &b);   
 add(a, b), add(b, a);   
 }   
   
 bfs(root);   
   
 while (m -- ) {   
 int a, b;   
 scanf("%d%d", &a, &b);   
 int p = lca(a, b);   
 printf("%d\n", p);   
 }   
   
}

### LCA-tarjan

//AcWing 1171. 距离   
#include <bits/stdc++.h>   
using namespace std;   
   
typedef pair<int, int> PII;   
   
const int N = 2e4 + 10, M = N \* 2;   
   
int n, m;   
int h[N], e[M], w[M], ne[M], idx;   
int dist[N];   
int p[N];   
int st[N];   
int res[N];   
vector<PII> query[N]; // first存查询的另外一个点，second存查询编号   
   
void add(int a, int b, int c) {   
 e[idx] = b, w[idx] = c, ne[idx] = h[a], h[a] = idx ++;   
}   
   
void dfs(int u, int fa) {   
 for (int i = h[u]; ~i; i = ne[i]) {   
 int j = e[i];   
 if (j == fa) continue;   
 dist[j] = dist[u] + w[i];   
 dfs(j, u);   
 }   
}   
   
int find(int x) {   
 return p[x] == x ? x : p[x] = find(p[x]);   
}   
   
void tarjan(int u) {   
 st[u] = 1;   
 for (int i = h[u]; ~i; i = ne[i]) {   
 int j = e[i];   
 if (!st[j]) {   
 tarjan(j);   
 p[j] = u;   
 }   
 }   
   
 for (auto item : query[u]) {   
 int y = item.first, id = item.second;   
 if (st[y] == 2) {   
 int anc = find(y);   
 res[id] = dist[u] + dist[y] - 2 \* dist[anc];   
 }   
 }   
   
 st[u] = 2;   
}   
   
int main() {   
 cin >> n >> m;   
 memset(h, -1, sizeof h);   
 for (int i = 1; i < n; i ++ ) {   
 int a, b, c;   
 cin >> a >> b >> c;   
 add(a, b, c); add(b, a, c);   
 }   
   
 for (int i = 1; i <= m; i ++ ) {   
 int a, b;   
 cin >> a >> b;   
 if (a != b) {   
 query[a].push\_back({b, i});   
 query[b].push\_back({a, i});   
 }   
 }   
   
 for (int i = 1; i <= n; i ++ ) p[i] = i;   
   
 dfs(1, -1);   
 tarjan(1);   
   
 for (int i = 1; i <= m; i ++ ) cout << res[i] << "\n";   
}

## tarjan求割点

// Problem: P3388 【模板】割点（割顶）   
// Contest: Luogu   
// URL: https://www.luogu.com.cn/problem/P3388   
// Memory Limit: 125 MB   
// Time Limit: 1000 ms   
// Powered by CP Editor (https://github.com/cpeditor/cpeditor)   
   
/\*   
 @ author: dragon\_bra   
 @ QQ: 1277037638   
 @ email: tommy514@foxmail.com   
\*/   
   
#include <bits/stdc++.h>   
#define fastio ios\_base::sync\_with\_stdio(false); cin.tie(0);   
using namespace std;   
   
const int N = 2e5 + 10;   
   
int n,m;   
struct edge {   
 int next,to;   
} p[N];   
   
int head[N], num; // num stands for edge number   
   
void addEdge(int x,int y) {   
 p[++num].next=head[x];   
 p[num].to=y;   
 head[x]=num;   
}   
int dfn[N], low[N], tim, cut[N];   
// tim 代表入栈的顺序是第几个   
// cut[i]代表该点是否是割点   
   
void tag (int x,int zx) {   
 // zx 代表最早出现的祖先   
 int kid = 0;   
 dfn[x] = low[x] = ++tim;   
   
 for(int i=head[x]; i; i=p[i].next) {   
 int v = p[i].to;   
   
 if(!dfn[v]) {   
 tag(v, zx);   
 low[x] = min(low[v], low[x]);   
 if(low[v] >= dfn[x] && x!=zx) cut[x]=1;   
 if(x==zx) kid++;   
 }   
   
 low[x] = min(low[x], dfn[v]);   
 }   
 if(kid>1 && x==zx) cut[x]=1;   
 // 如果有两个及以上的儿子，则也是割点   
}   
   
int ans;   
   
int main() {   
 fastio;   
 cin >> n >> m;   
 for (int i=1; i<=m; i++) {   
 int u, v;   
 cin >> u >> v;   
 addEdge(u, v);   
 addEdge(v, u);   
 }   
   
 for(int i=1;i<=n;i++) if(!dfn[i]) tag(i,i);   
   
 for(int i=1;i<=n;i++) ans += cut[i];   
 printf("%d\n",ans);   
 for(int i=1;i<=n;i++)   
 if(cut[i]) printf("%d ",i);   
   
 return 0;   
}

## tarjan缩点

// Problem: P3387 【模板】缩点   
// Contest: Luogu   
// URL: https://www.luogu.com.cn/problem/P3387   
// Memory Limit: 125 MB   
// Time Limit: 1000 ms   
// Powered by CP Editor (https://github.com/cpeditor/cpeditor)   
   
/\*   
 @ author: dragon\_bra   
 @ QQ: 1277037638   
 @ email: tommy514@foxmail.com   
\*/   
   
#include <bits/stdc++.h>   
#define fastio ios\_base::sync\_with\_stdio(false); cin.tie(0);   
using namespace std;   
   
const int N = 10000+15;   
int n, m;   
vector<int> G[N];   
vector<int> G2[N];   
int tim, top;   
int p[N], belong[N], dfn[N], low[N];   
//DFN(u)为节点u搜索被搜索到时的次序编号(时间戳)，Low(u)为u或u的子树能够追溯到的最早的栈中节点的次序号   
int stac[N], vis[N];   
//栈只为了表示此时是否有父子关系   
int in[N], dist[N];   
   
void tarjan(int x) {   
 // tarjan 缩点核心代码   
 low[x]=dfn[x]=++tim;   
 stac[++top]=x;vis[x]=1;   
 for (int v:G[x]) {   
 if (!dfn[v]) {   
 tarjan(v);   
 low[x] = min(low[x], low[v]);   
 } else if (vis[v]) {   
 low[x] = min(low[x], low[v]);   
 }   
 }   
 if (dfn[x]==low[x]) {   
 int y;   
 while (y=stac[top--]) {   
 belong[y] = x;   
 vis[y] = 0;   
 if (x==y) break;   
 p[x] += p[y]; // 增加点权，本题有效   
 }   
 }   
}   
   
int topo() {   
 queue <int> Q;   
 for (int i=1; i<=n; i++) {   
 if (belong[i]==i && !in[i]) {   
 Q.push(i);   
 dist[i] = p[i];   
 }   
 }   
   
 while (!Q.empty()) {   
 int now = Q.front(); Q.pop();   
 for (int v:G2[now]) {   
 dist[v] = max(dist[v], dist[now] + p[v]);   
 in[v] --;   
 if (in[v]==0) Q.push(v);   
 }   
 }   
   
 int ans = 0;   
 for (int i=1;i<=n;i++) ans = max(ans, dist[i]);   
   
 return ans;   
}   
   
int main() {   
 fastio;   
 cin >> n >> m;   
 for (int i=1;i<=n;i++) cin >> p[i];   
   
 for (int i=1; i<=m; i++) {   
 int u, v; cin >> u >> v;   
 G[u].push\_back(v);   
 }   
   
 for (int i=1; i<=n; i++)   
 if (!dfn[i]) tarjan(i);   
   
 for (int i=1; i<=n; i++) {   
 for (int v:G[i]) {   
 if (belong[i] == belong[v]) continue;   
 G2[belong[i]].push\_back(belong[v]);   
 in[belong[v]] ++;   
 }   
 }   
   
 printf("%d",topo());   
   
 return 0;   
}

# 字符串

## KMP

void makeNext(string s) {  
 int i = 0, k = -1;  
 next[0] = -1;  
 int len = strlen(s);  
 while (i < len-1) {  
 while (k >= 0 && s[i] != s[k]) k = next[k];  
 i ++; k ++;  
 if (s[i] == s[k]) next[i] = next[k];  
 else next[i] = k;  
 }  
}  
  
int kmpMatch(string t, string p) {  
 int i = 0, j = 0;  
 int len\_1 = strlen(t), len2 = strlen(p);  
 while (i < len\_1 && j < len\_2) {  
 if (i == -1 || p[i] == c[j]) {  
 i ++; j ++;  
 } else {  
 i = next[i];  
 }  
 }  
 if (i >= len\_1) return j - len\_1 + 1;  
 else return 0;  
}

## Manachar

/\*  
\* @ author: dragon\_bra  
\* @ email: tommy514@foxmail.com  
\* @ data: 2020-05-16 15:19  
\*/  
  
#include <algorithm>  
#include <cmath>  
#include <cstdio>  
#include <cstdlib>  
#include <cstring>  
#include <iostream>  
#include <sstream>  
#include <map>  
#include <set>  
#include <queue>  
#include <vector>  
  
using namespace std;  
  
typedef long long ll;  
const int INF = 0x3f3f3f3f;  
const int mod = 1e9+7;  
const double eps = 1e-5;  
const int N = 2e5 + 10;  
  
void redirect() {  
 #ifdef LOCAL  
 freopen("in.txt","r",stdin);  
 freopen("out.txt","w",stdout);  
 #endif  
}  
  
int p[N\*2];  
char str[N\*2],t[N\*2];  
  
int Manacher(char \*str,int len){  
 // 初始化部分  
 t[0] = '$';t[1] = '#';  
 int tot = 2;  
 for(int i=0; i<len; i++){  
 t[tot++]=str[i];  
 t[tot++]='#';  
 }  
  
 int mx = 0,id = 0,reslen = 0,resCenter = 0;  
 for(int i=0; i<tot; i++){  
 if(i<mx) p[i] = min(p[2\*id - i] , mx - i); // 2\*id - i = id - (i-id); j和i关于id对称;  
 else p[i] = 1; // i比mx大了，也就是当前最大的回文串够不着它了  
  
 while( t[i+p[i]] == t[i-p[i]] ) p[i] ++; // 计算i为中心大时候，最大的回文字串有多大  
 if(p[i]+i > mx){  
 mx = i + p[i];  
 id = i;  
 }  
  
 if(reslen < p[i]) {  
 reslen = p[i], resCenter = i;  
 }  
  
 }  
 return reslen;  
}  
  
int main(){  
 while(~scanf("%s", str)){  
 int len = strlen(str);  
 printf("%d\n",Manacher(str,len)-1);  
 }  
 return 0;  
}

## 最大字典序子串

string lastSubstring(string s) {   
 int left=0;   
 int right=left+1;   
 int step=0;   
 while(right + step <s.size()){   
 if(s[left+step]<s[right+step]){   
 left=right;   
 right=left+1;   
 step=0;   
 }   
 else if(s[left+step]==s[right+step]){   
 step++;   
 }   
 else{ // s[left+step]>s[right+step]   
 right+=step+1;   
 step=0;   
 }   
 }   
 return s.substr(left, s.size()-left);   
}

## 最大最小表示法

int min\_max\_express(bool flag) // flag=true的时候为字典序最小，=false的时候为字典序最大   
{   
 int i = 0, j = 1, k = 0, t;   
 while(i < len && j < len && k < len)   
 {   
 t = str[(i + k) % len] - str[(j + k) % len];   
 if(!t) k++;   
 else   
 {   
 if(flag)   
 {   
 if(t > 0) i = i + k + 1;   
 else j = j + k + 1;   
 }   
 else   
 {   
 if(t > 0) j = j + k + 1;   
 else i = i + k + 1;   
 }   
   
 if(j == i) j++;   
 k = 0;   
 }   
 }   
   
 return i < j ? i : j;   
}

# DFS

## DSU（树上启发式合并）

/\*  
  
DSU-on-tree  
树上启发式合并  
重点：{  
 dfs1()：找出所有节点的重儿子，记录每个节点的子树大小  
 dfs2()：搜索下去更新答案，  
 如果是重儿子，  
 将兄弟所有的集合合并到重儿子，并将重儿子的答案合并到父亲节点  
 else 如果是轻儿子  
 寻找他的重儿子并先把答案合并到自己  
}  
  
\*/  
#include <bits/stdc++.h>  
using namespace std;  
  
typedef long long ll;  
const int N = 1e5 + 5;  
  
void redirect() {  
 #ifdef LOCAL  
 freopen("1.in","r",stdin);  
 freopen("1.out","w",stdout);  
 #endif  
}  
  
int n,f[N];  
int son[N], size[N];  
ll ans[N], rans[N];  
  
vector<int> G[N];  
set<ll> S[N];  
  
void merge(int a,int b) {  
 while(!S[b].empty()){  
 ll t = \*( S[b].begin() ); S[b].erase( t );  
  
 ll up=0, low=0;  
  
 if( S[a].upper\_bound(t) == S[a].begin() ) {  
 up = \*S[a].begin();  
 ans[a] += ( up - t ) \* ( up - t );  
 } else if( S[a].upper\_bound(t) == S[a].end() ) {  
 low = \* ( --S[a].lower\_bound(t) );  
 ans[a] += ( t - low ) \* ( t - low );  
 } else {  
 up = \* ( S[a].upper\_bound(t) ); low = \* ( --S[a].lower\_bound(t) ) ;  
 ans[a] -= ( up - low ) \* ( up - low ); ans[a] += ( up - t ) \* ( up - t ); ans[a] += ( t - low ) \* ( t - low );  
 }  
  
 S[a].insert(t);  
 }  
}  
  
void dfs1(ll u, ll fa) {//记录了所有子树的size 和 每个节点的重儿子  
 size[u] = 1;  
 for ( auto v:G[u] ) {  
 dfs1(v, u);  
 size[u] += size[v];  
 if ( size[v] > size[son[u]] ) son[u] = v;  
 }  
}  
  
void dfs2(ll u,ll fa,bool keep,bool isson){  
 for( auto v:G[u] ) {  
 if( v!=son[u] ){  
 dfs2(v,u,0,0);  
 }  
 }  
  
 if( son[u] ) {  
 dfs2(son[u],u,1,1);  
 }  
  
 if( keep ) {  
 for( auto v:G[fa] ) {  
 if( u==v ) continue;  
 merge( u, v );  
 }  
  
 if( S[fa].size() < S[u].size() ) S[fa].swap(S[u]), swap(ans[fa],ans[u]);  
 merge( fa, u );   
 rans[fa] = ans[fa];  
 }  
}  
  
int main() {  
 redirect();  
  
 scanf("%d",&n); f[1] = 1; S[1].insert(1);  
 for(ll i=2;i<=n;i++){  
 scanf("%d",&f[i]);  
 G[ f[i] ].push\_back(i); S[i].insert(i);  
 }  
  
 dfs1(1,1);  
 dfs2(1,1,0,0);  
  
 for(ll i=1;i<=n;i++) {  
 printf("%lld\n",rans[ i ]);  
 }  
  
 return 0;  
}  
  
/\*  
-----------------  
author:dragon\_bra  
-----------------  
\*/

# STL&杂项

## 二分（标准）

/\*\*  
 \* struct Interval {  
 \* int start;  
 \* int end;  
 \* Interval(int s, int e) : start(start), end(e) {}  
 \* };  
 \*/  
  
class Solution {  
public:  
 /\*\*  
 \* 代码中的类名、方法名、参数名已经指定，请勿修改，直接返回方法规定的值即可  
 \*   
 \* @param n int整型 玩偶数  
 \* @param m int整型 区间数  
 \* @param intervals Interval类vector 表示区间  
 \* @return int整型  
 \*/  
 static bool cmp(Interval a, Interval b) {  
 return a.start < b.start;  
 }  
   
 int doll(int n, int m, vector<Interval>& intervals) {  
 // write code here  
 long long l = 1, r = n;  
 while (l <= r) {  
 mid = (l+r) / 2;  
 // check code here  
 if (flag) {  
 ans = mid; l = mid + 1;  
 }  
 else r = mid - 1;  
 }  
   
 return ans;  
 }  
};

## 优先队列

#include<iostream>  
#include<vector>  
#include<queue>  
using namespace std;  
int tmp[100];  
struct cmp1{  
 bool operator()(int x,int y)  
 {  
 return x>y;//小的优先级高 ,从小到大排   
 }  
};   
struct cmp2{  
 bool operator()(const int x,const int y)  
 {  
 return tmp[x]>tmp[y];  
 }  
};   
struct node{  
 int x,y;  
 friend bool operator<(node a,node b)  
 {  
 return a.x>b.x;//按x从小到大排   
 }  
};  
priority\_queue<int>q1;  
priority\_queue<int,vector<int>,cmp1>q2;  
priority\_queue<int,vector<int>,cmp2>q3;  
priority\_queue<node>q4;  
int main()  
{  
 int i,j,k,m,n;  
 int x,y;  
 node a;  
 while(cin>>n)  
 {  
 for(int i=0;i<n;i++)  
 {  
 cin>>a.y>>a.x;  
 q4.push(a);  
 }  
 cout<<endl;  
 while(!q4.empty())  
 {  
 cout<<q4.top().y<<" "<<q4.top().x<<" "<<endl;  
 q4.pop();  
 }  
 cout<<endl;  
   
 int t;  
 for(i=0;i<n;i++)  
 {  
 cin>>t;  
 q2.push(t);  
 }  
 while(!q2.empty())  
 {  
 cout<<q2.top()<<endl;  
 q2.pop();  
 }  
 cout<<endl;  
 }  
 return 0;  
}

## exmu

#include <cstdio>   
#include <iostream>   
#include <algorithm>   
#include <cmath>   
#include <cstring>   
#include <map>   
#include <set>   
#include <queue>   
#include <string>   
#include <vector>   
using namespace std;   
typedef long long ll;   
typedef unsigned long long ull;   
const int INF = 0x7fffffff;   
const int mod = 1e9+7;   
const double eps = 1e-5;   
const int N = 1e5+10;   
   
void redirect() {   
 #ifdef LOCAL   
 freopen("test.txt","r",stdin);   
 //freopen("out.txt","w",stdout);   
 #endif   
}   
inline ll read() {   
 ll f=1,x=0;char ch;   
 do {ch=getchar(); if(ch=='-') f=-1;} while (ch<'0'||ch>'9');   
 do {x=x\*10+ch-'0'; ch=getchar(); } while (ch>='0'&&ch<='9');   
 return x\*f;   
}   
   
int main() {   
 //redirect();   
 cout<<"Hello world."<<endl;   
}   
   
/\*   
-----------------   
author:dragon\_bra   
-----------------   
\*/

## highbit

int highbit(int x) {  
 // leftest digit of 1  
 // nearly O(1)  
 union { double a; int b[2]; };  
 a = x;  
 return (b[1] >> 20) - 1023;  
}  
  
{ // 我爱发明  
 vector<long long> p(32);  
  
 void init() {  
 p[0] = 1;  
 for (int i=1; i<=31; i++) p[i] = p[i-1] \* 2;  
 }  
  
 int highbit(int x) {  
 return upper\_bound(p.begin(), p.end(), x) - p.begin() - 1;  
 }  
}

## LIS（最长上升子序列）

/\*  
\* @ author: dragon\_bra  
\* @ email: tommy514@foxmail.com  
\* @ data: 2020-07-25 12:12  
\*/  
  
#include <algorithm>  
#include <cmath>  
#include <cstdio>  
#include <cstdlib>  
#include <cstring>  
#include <iostream>  
#include <sstream>  
#include <map>  
#include <set>  
#include <queue>  
#include <vector>  
  
using namespace std;  
  
typedef long long ll;  
const int INF = 0x3f3f3f3f;  
const int mod = 1e9+7;  
const double eps = 1e-5;  
const int N = 1e3 + 10;  
  
void redirect() {  
 #ifdef LOCAL  
 freopen("in.txt","r",stdin);  
 freopen("out.txt","w",stdout);  
 #endif  
}  
  
int n, a[N];  
int f[N];  
  
int lis(int x) {  
 f[0]=-INF;  
 int s=0, t;  
 for(int i=1;i<=n;i++) {  
 t = a[i+x-1];  
 if(t > f[s]) f[++s]=t;  
 else {  
 int l=1, r=s, m;  
 while(l<=r) {  
 m=(l+r)/2;  
 if(t>f[m]) l=m+1;  
 else r=m-1;  
 }  
 f[l]=t;  
 }  
 }  
 return s;  
}  
  
int main() {  
 redirect();  
  
 cin>>n;  
 for (int i=1; i<=n; i++) {  
 cin >> a[i];  
 a[i+n] = a[i];  
 }  
  
 int mx = 0;  
 for (int i=1; i<=n; i++) {  
 mx = max(mx, lis(i));  
 }  
   
 cout << n - mx << endl;  
}

## Tarjan

void tarjan(int i) {   
 int j;   
 DFN[i]=LOW[i]=++Dindex;   
 instack[i]=true;   
 Stap[++Stop]=i;   
 for (edge \*e=V[i];e;e=e->next)   
 {   
 j=e->t;   
 if (!DFN[j])   
 {   
 tarjan(j);   
 if (LOW[j]<LOW[i])   
 LOW[i]=LOW[j];   
 }   
 else if (instack[j] && DFN[j]<LOW[i])   
 LOW[i]=DFN[j];   
 }   
 if (DFN[i]==LOW[i])   
 {   
 Bcnt++;   
 do   
 {   
 j=Stap[Stop--];   
 instack[j]=false;   
 Belong[j]=Bcnt;   
 }   
 while (j!=i);   
 }   
}   
void solve()   
{   
 int i;   
 Stop=Bcnt=Dindex=0;   
 memset(DFN,0,sizeof(DFN));   
 for (i=1;i<=N;i++)   
 if (!DFN[i])   
 tarjan(i);   
}

## 对拍.bat

:loop   
   
rand.exe   
A.exe   
A2.exe   
   
fc 1.out baoli.cout   
if errorlevel==1 pause   
   
goto loop

# 乱七八糟的性质

## 图论的一些概念及性质

图论的一些知识点，先mark一下

网上的东西，也不知道谁是原创。  
§1图论点、边集和二分图的相关概念和性质  
点覆盖、最小点覆盖  
点覆盖集即一个点集，使得所有边至少有一个端点在集合里。或者说是“点” 覆盖了所有“边”。。极小点覆盖(minimal vertex covering)：本身为点覆盖，其真子集都不是。最小点覆盖(minimum vertex covering)：点最少的点覆盖。点覆盖数(vertex covering number)：最小点覆盖的点数。  
边覆盖、极小边覆盖  
边覆盖集即一个边集，使得所有点都与集合里的边邻接。或者说是“边” 覆盖了所有“点”。极小边覆盖(minimal edge covering)：本身是边覆盖，其真子集都不是。最小边覆盖(minimum edge covering)：边最少的边覆盖。边覆盖数(edge covering number)：最小边覆盖的边数。  
独立集、极大独立集  
独立集即一个点集，集合中任两个结点不相邻，则称V为独立集。或者说是导出的子图是零图（没有边）的点集。极大独立集(maximal independent set)：本身为独立集，再加入任何点都不是。最大独立集(maximum independent set)：点最多的独立集。独立数(independent number)：最大独立集的点。  
团  
团即一个点集，集合中任两个结点相邻。或者说是导出的子图是完全图的点集。极大团(maximal clique)：本身为团，再加入任何点都不是。最大团(maximum clique)：点最多的团。团数(clique number)：最大团的点数。  
边独立集、极大边独立集  
边独立集即一个边集，满足边集中的任两边不邻接。极大边独立集(maximal edge independent set)：本身为边独立集，再加入任何边都不是。最大边独立集(maximum edge independent set)：边最多的边独立集。边独立数(edge independent number)：最大边独立集的边数。  
边独立集又称匹配(matching)，相应的有极大匹配(maximal matching)，最大匹配(maximum matching)，匹配数(matching number)。  
支配集、极小支配集  
支配集即一个点集，使得所有其他点至少有一个相邻点在集合里。或者说是一部分的“点”支配了所有“点”。极小支配集(minimal dominating set)：本身为支配集，其真子集都不是。最小支配集(minimum dominating set)：点最少的支配集。支配数(dominating number)：最小支配集的点数。  
边支配集、极小边支配集  
边支配集即一个边集，使得所有边至少有一条邻接边在集合里。或者说是一部分的“边”支配了所有“边”。极小边支配集(minimal edge dominating set)：本身是边支配集，其真子集都不是。最小边支配集(minimum edge dominating set)：边最少的边支配集。边支配数(edge dominating number)：最小边支配集的边数。  
最小路径覆盖  
最小路径覆盖(path covering)：是“路径” 覆盖“点”，即用尽量少的不相交简单路径覆盖有向无环图G的所有顶点，即每个顶点严格属于一条路径。路径的长度可能为0(单个点)。  
最小路径覆盖数＝G的点数－最小路径覆盖中的边数。应该使得最小路径覆盖中的边数尽量多，但是又不能让两条边在同一个顶点相交。拆点：将每一个顶点i拆成两个顶点Xi和Yi。然后根据原图中边的信息，从X部往Y部引边。所有边的方向都是由X部到Y部。因此，所转化出的二分图的最大匹配数则是原图G中最小路径覆盖上的边数。因此由最小路径覆盖数＝原图G的顶点数－二分图的最大匹配数便可以得解。  
匹配  
匹配(matching)是一个边集，满足边集中的边两两不邻接。匹配又称边独立集(edge independent set)。  
在匹配中的点称为匹配点(matched vertex)或饱和点；反之，称为未匹配点(unmatched vertex)或未饱和点。  
交错轨(alternating path)是图的一条简单路径，满足任意相邻的两条边，一条在匹配内，一条不在匹配内。  
增广轨(augmenting path)：是一个始点与终点都为未匹配点的交错轨。  
最大匹配(maximum matching)是具有最多边的匹配。  
匹配数(matching number)是最大匹配的大小。  
完美匹配(perfect matching)是匹配了所有点的匹配。  
完备匹配(complete matching)是匹配了二分图较小集合（二分图X，Y中小的那个）的所有点的匹配。  
增广轨定理：一个匹配是最大匹配当且仅当没有增广轨。  
所有匹配算法都是基于增广轨定理：一个匹配是最大匹配当且仅当没有增广轨。这个定理适用于任意图。  
二分图的性质  
二分图中，点覆盖数是匹配数。  
(1) 二分图的最大匹配数等于最小覆盖数，即求最少的点使得每条边都至少和其中的一个点相关联，很显然直接取最大匹配的一段节点即可。  
(2) 二分图的独立数等于顶点数减去最大匹配数，很显然的把最大匹配两端的点都从顶点集中去掉这个时候剩余的点是独立集，这是|V|-2\*|M|，同时必然可以从每条匹配边的两端取一个点加入独立集并且保持其独立集性质。  
(3) DAG的最小路径覆盖，将每个点拆点后作最大匹配，结果为n-m，求具体路径的时候顺着匹配边走就可以，匹配边i→j’,j→k’,k→l’….构成一条有向路径。  
（4）最大匹配数=左边匹配点+右边未匹配点。因为在最大匹配集中的任意一条边，如果他的左边没标记，右边被标记了，那么我们就可找到一条新的增广路，所以每一条边都至少被一个点覆盖。  
（5）最小边覆盖=图中点的个数-最大匹配数=最大独立集。  
二分图的判定  
二分图是这样一个图： 有两顶点集且图中每条边的的两个顶点分别位于两个顶点集中，每个顶点集中没有边直接相连接！  
　无向图G为二分图的充分必要条件是，G至少有两个顶点,且其所有回路的长度均为偶数。  
　判断二分图的常见方法是染色法： 开始对任意一未染色的顶点染色，之后判断其相邻的顶点中，若未染色则将其染上和相邻顶点不同的颜色， 若已经染色且颜色和相邻顶点的颜色相同则说明不是二分图，若颜色不同则继续判断，bfs和dfs可以搞定！  
易知：任何无回路的的图均是二分图。  
§2 另外一些姿势  
定理1：最大独立集S 与 最小覆盖集T 互补

对于无向图:

最小点覆盖+最大独立集=顶点个数

最大团=补图的最大独立子集

关系1：给定图G = (V,E)无孤立点，则G的极大点独立集都是G的极小支配集。  
关系2：G的点覆盖数 a与点独立集数 b满足： a + b = n。  
关系3：G的边覆盖数 a与边独立集数 b满足： a + b = n。（边独立集数即匹配数）  
关系3：给定图G = (V,E)无孤立点，|V | = n。M是G的匹配，W是G的边覆盖，则|M|≤|W|，等号成立时M是G的完美匹配而W是G的最小边覆盖。  
对于二部图:

最小点覆盖数 = 最大匹配数

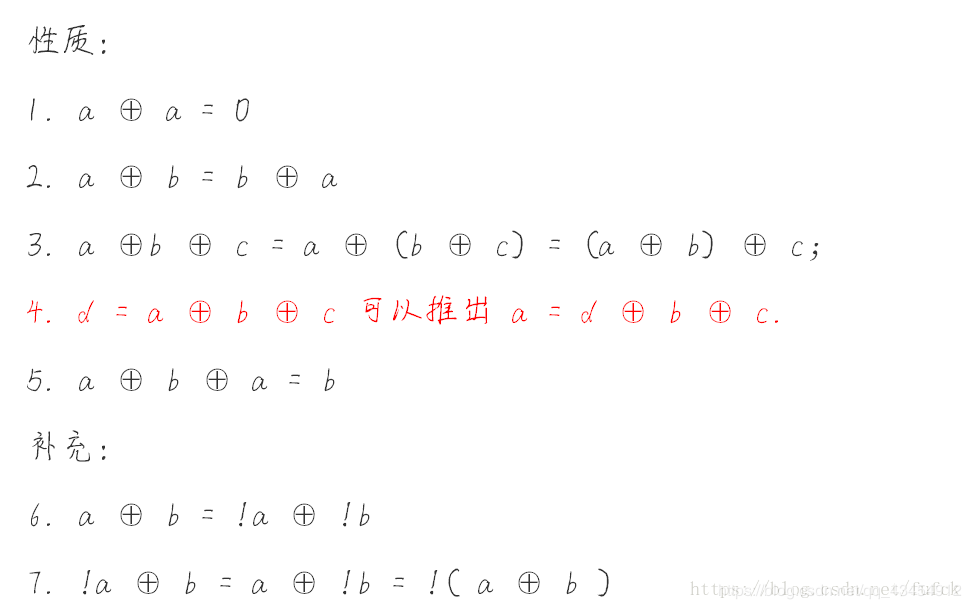
最小路径覆盖 = 顶点数 – 最大（二分）匹配数

最小边覆盖与最小路径覆盖的区别：

边覆盖集：通俗地讲，所谓边覆盖集，就是G中所有的顶点都是E\*中某条边的邻接顶点（边覆盖顶点），一条边只能覆盖2个顶点。  
注意：在无向图中存在用尽量少的边去“覆盖”住所有的顶点，所以边覆盖集有极小与最小的区别。  
极小边覆盖：若边覆盖E\*中的任何真子集都不是边覆盖集，则称E\*是极小边覆盖集。  
最小边覆盖：边数最小的边覆盖称为最小边覆盖，通俗地讲，就是极小边覆盖中的最小的一个集合。  
最小边覆盖在二分图中的应用：  
最小边覆盖 = 最大独立集 = n - 最大匹配，这个是二分图上的一个性质。  
最小路径覆盖和最小边覆盖不同，不要求给的图是二分图，而是要求是ＰＸＰ的有向图，不能有环，然后根据原图构造二分图，构造方法是将点一分为二，如，i分为i1和i2然后如果i和j有边，那么就在i1和j2之间连一条边。由此构成二分图  
然后最小路径覆盖是n-m，n为原图的点的个数，m为新造二分图的最大匹配。

要使用上面的最小边覆盖的公式的前提是你已经将点集分成了左右两个点集，如果只是一个无向图的话那么就可以按照下面的做法来做：  
无向图G(V,E)边覆盖的求解步骤：  
1.将无向图拆点，即若在无向图中存在节点i，则将节点i拆为i1,i2分别位于二分图的X部和Y部.若存在边ij，则连接二分图的i1j2,i2j1。  
2.原无向图中的节点数为|V|所以在构造的二分图有2\*|V|个节点。在二分图中存在公式：  
2\*|V| = 2\*二分图的最大匹配数 + 二分图中未匹配的点。其中二分图的最大匹配数+二分图中未匹配的点即覆盖了二分图中所有的点，相对于原无向图，相当于覆盖了每个点两次，即原边覆盖的最小 值转化为二分图的最大匹配数+二分图中未匹配的点的最小值。又有公式2\*|V| = 2\*二分图的最大匹配数 + 二分图中未匹配的点，可得：  
二分图的最大匹配数+二分图中未匹配的点 = 2\*|V| - 二分图的最大匹配数，又此结果为覆盖了原图所有顶点两次，所以结果应该除以2.  
所以无向图的最小边覆盖 = |V| - 二分图的最大匹配数/2.

## 位运算



## 组合数学

