hust-yc-template

yincircle

2019 年 9 月 26 日

目录

1 字符串

1.1 最大最小表示法

```
int get_posmin(char *s){
    int len=strlen(s);
    int i=0,j=1,k=0;
    while(i<len&&j<len&&k<len){
        int t=s[(i+k)%len]-s[(j+k)%len];
        if(t==0)k++;
        else {
            if(t>0)
                i+=k+1;
                //max: j+=k+1;
            else
                j+=k+1;
                //max: i+=k+1;
            if(i==j)j++;
            k=0;
        }
    }
    return min(i,j);
}
```

1.2 KMP

1.2.1 前缀函数（每一个前缀的最长 border）

```
/*
length of min loop
if(len%(len-nex[len])==0) res=len/(len-nex[len]);
else res=1;
s->s[0..n-1]
*/
char s[maxn];
int nex[maxn];
void get_next(char *s,int *nex,int len){
    int i,j;
    i=0;
    j=nex[0]=-1;
    while(i<len)
        if(j==-1||s[i]==s[j]) nex[++i]=++j;
        else j=nex[j];
}
int KMP(char *a,char *b,int lena,int lenb){
    int i,j;
    get_next(b,nex,lenb);
    i=j=0;
    while(i<lena){
        if(j==-1||a[i]==b[j]){i++;j++;}
        else j=nex[j];
        if(j==lenb) break;//successful match
    }
    return j==-1?0:j;
}
```

1.2.2 Z 函数（每一个后缀和该字符串的 LCP 长度）

```
void get_z(int a[], char s[], int n) {
    int l = 0, r = 0; a[0] = n;
    for(int i=1;i<n;i++) {
        a[i] = i > r ? 0 : min(r - i + 1, a[i - l]);
        while (i + a[i] < n && s[a[i]] == s[i + a[i]]) ++a[i];
        if (i + a[i] - 1 > r) { l = i; r = i + a[i] - 1; }
    }
}
```

1.3 manacher

```
int ma[maxn<<1|1];
char s[maxn];
void manacher(char s[]){
    int n = strlen(s);
    int id=0, ub=0;
    for(int i=0; i<2*n-1; i++) {
        int p=i/2, q=(i+1)/2;
        int l = q<ub?min(ub-q, ma[id-i]):0;
        while(p-l>=0 && q+l<n && s[p-l]==s[q+l]) l++;
        if(ub < q+l) {
            ub = q+l;
            id = i*2;
        }
        ma[i] = l;
    }
    for(int i=0; i<2*n-1; i++)
        ma[i] = ma[i]*2-(!(i&1));
}
```

1.4 AC 自动机

```cpp
// HDU 6138
//给定若干字典串。
// query:strx stry 求最长的p,p为strx、stry子串，且p为某字典串的前缀
struct Aho_Corasick_Automaton{
    //basic
    int nxt[maxn*10][26],fail[maxn*10];
    int root,tot;
    //special
    int flag[maxn*10];
    int len[maxn*10];
    void clear(){
        memset(nxt[0],0,sizeof nxt[0]);
        root = tot=0;
    }
    int newnode(){
        tot++;
        memset(nxt[tot],0,sizeof nxt[tot]);
        flag[tot] = len[tot]=0;
        return tot;
    }
    void insert(char *s ){
        int now = root;
        while (*s){
            int id = *s-'a';
            if(!nxt[now][id])nxt[now][id] = newnode();
            len[nxt[now][id]] = len[now]+1;
            now = nxt[now][id];
        }
    }
    void insert(string str){
        int now = root;
        for (int i=0;i<str.size();i++){
            int id = str[i]-'a';
            if(!nxt[now][id])nxt[now][id] = newnode();
            len[nxt[now][id]] = len[now]+1;
            now = nxt[now][id];
        }
    }
    void build(){
        fail[root] = root;
        queue<int>Q;Q.push(root);
        while (!Q.empty()){
            int head = Q.front();Q.pop();
            for (int i=0;i<26;i++){
                if(!nxt[head][i])continue;
                int temp = nxt[head][i];
                fail[temp] = fail[head];
                while (fail[temp]&&!nxt[fail[temp]][i]){
                    fail[temp] = fail[fail[temp]];
                }
                if(head&&nxt[fail[temp]][i])fail[temp] = nxt[fail[temp]][i];
                Q.push(temp);
            }
        }
    }
    void search(string str,int QID);
    int query(string str,int QID);
}acam;
void Aho_Corasick_Automaton::search(string str,int QID) {
    int now = root;
    for (int i=0;i<str.size();i++){
        int id = str[i]-'a';
        now = nxt[now][id];int temp = now;
        while (temp!=root&&flag[temp]!=QID){
            flag[temp] = QID;
            temp = fail[temp];
        }
    }
}
int Aho_Corasick_Automaton::query(string str, int QID) {
    int ans =0;int now = root;
    for (int i=0;i<str.size();i++){
        int id = str[i]-'a';
        now = nxt[now][id];
        int temp = now;
        while (temp!=root){
            if(flag[temp]==QID){
                ans = max(ans,len[temp]);
                break;
            }
            temp = fail[temp];
        }
    }
    return ans;
}
string a[maxn];
```

```cpp
int m,n,qid;
int main(){
    int T;cin>>T;
    while (T--){
        acam.clear();cin>>n;
        for (int i=1;i<=n;i++){
            cin>>a[i];
            acam.insert(a[i]);
        }
        acam.build();cin>>m;
        for (int i=1;i<=m;i++){
            int x,y;cin>>x>>y;
            qid++;
            acam.search(a[x],qid);
            int ans = acam.query(a[y],qid);
            cout<<ans<<endl;
        }
    }
    return 0;
}
```

1.5 SAM

```cpp
struct SAM{
    int last,cnt,nxt[maxn*2][27],fa[maxn*2],l[maxn*2];
    void init(){
        last = cnt=1;
        memset(nxt[1],0,sizeof nxt[1]);
        fa[1]=0;ans=0;l[1]=0;
    }
    int inline newnode(){
        ++cnt;
        memset(nxt[cnt],0,sizeof nxt[cnt]);
        fa[cnt]=l[cnt]=0;
        return cnt;
    }
    void add(int c){
        int p = last;
        int np = newnode();
        last = np;
        l[np] = l[p]+1;
        while (p&&!nxt[p][c]){
            nxt[p][c]=np;
            p = fa[p];
        }
        if (!p){
            fa[np]=1;
        }else{
            int q = nxt[p][c];
            if (l[q]==l[p]+1){
                fa[np] = q;
            }else{
                int nq = newnode();
                memcpy(nxt[nq],nxt[q],sizeof nxt[q]);
                fa[nq] = fa[q];
                l[nq] = l[p]+1;
                fa[np]=fa[q]=nq;
                while (nxt[p][c]==q){
                    nxt[p][c]=nq;
                    p=fa[p];
                }
            }
        }
        ans+=l[last]-l[fa[last]];
    }
}sam;

//SPOJ substring
// calc ans_i=长度=i的所有子串，出现次数最多的一种出现了多少次。
#include<bits/stdc++.h>
#define RIGHT
//RIGHT: parent树的dfs序上主席树，求每个点的Right集合
using namespace std;
const int maxn = 25e4+100;
struct Node{int L,R,val;}Tree[maxn*40];
#ifdef RIGHT
struct Chairman_Tree{
    int cnt = 0;
    int root[maxn*2];
    void init(){
        memset(root,0,sizeof root);
        cnt =0;
    }
    /* 建T0空树 */
    int buildT0(int l, int r){
        int k = cnt++;
        Tree[k].val =0;
        if (l==r) return k;
        int mid = l+r >>1;
        Tree[k].L = buildT0(l, mid);Tree[k].R = buildT0(mid + 1, r);
        return k;
    }
```

```
    /* 上一个版本节点P, 【ppos】+=del 返回新版本节点*/
    int update (int P,int l,int r,int ppos,int del){
        assert(cnt < maxn*50);
        int k = cnt++;
        Tree[k].val = Tree[P].val +del;
        if (l==r) return k;
        int mid = l+r >>1;
        if (ppos<=mid){
            Tree[k].L = update(Tree[P].L,l,mid,ppos,
                del);
            Tree[k].R = Tree[P].R;
        }else{
            Tree[k].L = Tree[P].L;
            Tree[k].R = update(Tree[P].R,mid+1,r,ppos
                ,del);
        }
        return k;
    }
    int query(int PL,int PR,int l,int r,int L,int R)
        {
        if (l>R || L>r)return 0;
        if (L <= l && r <= R)return Tree[PR].val -
            Tree[PL].val;
        int mid = l + r >> 1;
        return query(Tree[PL].L,Tree[PR].L,l,mid,L,R)
             + query(Tree[PL].R,Tree[PR].R,mid+1,r,L,
            R);
    }
}tree;
#endif
char s[maxn];int n,ans[maxn];
/*注意需要按l将节点基数排序来拓扑更新parent树*/
struct Suffix_Automaton{
    //basic
    int nxt[maxn*2][26],fa[maxn*2],l[maxn*2];
    int last,cnt;
    //extension
    int cntA[maxn*2],A[maxn*2];/*辅助拓扑更新*/
    int num[maxn*2];/*每个节点代表的所有串的出现次数*/
    #ifdef RIGHT
    vector<int> E[maxn*2];
    int dfsl[maxn*2],dfsr[maxn*2],dfn;
    int pos[maxn*2];
    int end_pos[maxn*2];//1基
    #endif
    Suffix_Automaton(){ clear(); }
    void clear(){
        last =cnt=1;
        fa[1]=l[1]=0;
        memset(nxt[1],0,sizeof nxt[1]);
    }
    void init(char *s){
        while (*s){
            add(*s-'a');s++;
        }
    }
    void add(int c){
        int p = last;
        int np = ++cnt;
        memset(nxt[cnt],0,sizeof nxt[cnt]);
        l[np] = l[p]+1;last = np;
        while (p&&!nxt[p][c])nxt[p][c] = np,p = fa[p
            ];
        if (!p)fa[np]=1;
        else{
            int q = nxt[p][c];
            if (l[q]==l[p]+1)fa[np] =q;
            else{
                int nq = ++ cnt;
                l[nq] = l[p]+1;
                memcpy(nxt[nq],nxt[q],sizeof (nxt[q]))
                    ;
                fa[nq] =fa[q];fa[np] = fa[q] =nq;
                while (nxt[p][c]==q)nxt[p][c] =nq,p =
                    fa[p];
            }
        }
    }
    void build(){
        memset(cntA,0,sizeof cntA);
        memset(num,0,sizeof num);
        for (int i=1;i<=cnt;i++)cntA[l[i]]++;
        for (int i=1;i<=cnt;i++)cntA[i]+=cntA[i-1];
        for (int i=cnt;i>=1;i--)A[cntA[l[i]]--] =i;
        /*更行主串节点*/
        int temp=1;
        for (int i=0;i<n;i++){
            num[temp = nxt[temp][s[i]-'a'] ]=1;
        }
        /*拓扑更新*/
        for (int i=cnt;i>=1;i--){
            //basic
            int x = A[i];
            num[fa[x]]+=num[x];
            //special
            ans[l[x]] = max(ans[l[x]],num[x]);
```

```
        }
        //special
        for (int i=l[last];i>1;i--){
            ans[i-1] = max(ans[i-1],ans[i]);
        }
    }

    #ifdef RIGHT
    int get_right_between(int u,int l,int r){
        return tree.query(tree.root[dfsl[u] - 1],tree
            .root[dfsr[u]],1,::n,l,r);
    }
    void dfs(int u){
        dfsl[u] = ++ dfn;
        pos[dfn] = u;
        for (int v : E[u]){
            dfs(v);
        }
        dfsr[u] = dfn;
    }
    void extract_right(){
        int temp = 1;
        for (int i=0;i<n;i++){
            temp = nxt[temp][s[i] - 'a'];
            end_pos[temp] = i+1;
        }
        for (int i=2;i<=cnt;i++){
            E[fa[i]].push_back(i);
        }
        dfn = 0;
        dfs(1);
        tree.root[0] = tree.buildT0(1,n);
        for (int i=1;i<=cnt;i++){
            int u = pos[i];
            if (end_pos[u]){
                int idx = end_pos[u];
                tree.root[i] = tree.update(tree.root[i
                    -1],1,n,idx,1);
            }else{
                tree.root[i] = tree.root[i-1];
            }
        }
    }
    #endif
    void debug(){
        for (int i=cnt;i>=1;i--){
            printf("num[%d]=%d l[%d]=%d fa[%d]=%d\n",
                i,num[i],i,l[i],i,fa[i]);
            }
        }
}sam;
int main(){
    scanf("%s",s);
    /* calc n must before sam.init()*/
    n = strlen(s);
    sam.init(s);
    sam.build();
    for (int i=1;i<=n;i++){
        printf("%d\n",ans[i]);
    }
    return 0;
}
```

1.6 PAM

```
struct Palindromic_AutoMaton{
    //basic
    int s[maxn],now;
    int nxt[maxn][26],fail[maxn],l[maxn],last,tot;
    //extension
    int num[maxn];
    void clear(){
        //1节点：奇数长度root 0节点：偶数长度root
        s[0]=l[1]=-1;
        fail[0]=tot=now=1;
        last=l[0]=0;
        memset(nxt[0],0,sizeof(nxt[0]));
        memset(nxt[1],0,sizeof(nxt[1]));
    }
    Palindromic_AutoMaton(){clear();}
    int newnode(int x){
        tot++;
        memset(nxt[tot],0,sizeof(nxt[tot]));
        fail[tot]=num[tot]=0;
        l[tot]=x;
        return tot;
    }
    int get_fail(int x){
        while(s[now-l[x]-2]!=s[now-1])x=fail[x];
        return x;
    }
    void add(int ch){
        s[now++]=ch;
        int cur=get_fail(last);
        if(!nxt[cur][ch]){
```

```
            int tt=newnode(l[cur]+2);
            fail[tt]=nxt[get_fail(fail[cur])][ch];
            nxt[cur][ch]=tt;
        }
        last=nxt[cur][ch];num[last]++;
    }
    void build(){
        for(int i=tot;i>=2;i--){
            num[fail[i]]+=num[i];
        }
        num[0]=num[1]=0;
    }
    void init(char* ss){
        while(*ss){
            add(*ss-'a');ss++;
        }
    }
    void init(string str){
        for(int i=0;i<(int)str.size();i++){
        add(str[i]-'a');
        }
    }
}pam;
```

1.7 DA+LCA

```
int t1[maxn],t2[maxn],c[maxn];

bool cmp(int *r,int a,int b,int l){
    return r[a]==r[b]&&r[a+l]==r[b+l];
}

void da(int str[],int sa[],int rank[],int height[],
     int n,int m){
    n++;
    int i,j,p,*x=t1,*y=t2;
    for(i=0;i<m;i++)c[i]=0;
    for(i=0;i<n;i++)c[x[i]=str[i]]++;
    for(i=1;i<m;i++)c[i]+=c[i-1];
    for(i=n-1;i>=0;i--)sa[--c[x[i]]]=i;
    for(j=1;j<=n;j<<=1){
        p=0;
        for(i=n-j;i<n;i++)y[p++]=i;
        for(i=0;i<n;i++)if(sa[i]>=j)y[p++]=sa[i]-j;
        for(i=0;i<m;i++)c[i]=0;
        for(i=0;i<n;i++)c[x[y[i]]]++;
        for(i=1;i<m;i++)c[i]+=c[i-1];
        for(i=n-1;i>=0;i--)sa[--c[x[y[i]]]]=y[i];
        swap(x,y);
        p=1;
        x[sa[0]]=0;
        for(i=1;i<n;i++)
        x[sa[i]]=cmp(y,sa[i-1],sa[i],j)?p-1:p++;
        if(p>=n)break;
        m=p;
    }
    int k=0;
    n--;
    for(i=0;i<=n;i++)rank[sa[i]]=i;
    for(i=0;i<n;i++){
        if(k)k--;
        j=sa[rank[i]-1];
        while(str[i+k]==str[j+k])k++;
            height[rank[i]]=k;
    }
}

int rnk[maxn],height[maxn],r[maxn],sa[maxn];
int rmq[maxn];

int n,minnum[maxn][20];
void RMQ(){
    int i,j;
    int m=(int)(log(n*1.0)/log(2.0));
    for(i=1;i<=n;i++)
        minnum[i][0]=height[i];
    for(j=1;j<=m;j++)
        for(i=1;i+(1<<j)-1<=n;i++)
            minnum[i][j]=min(minnum[i][j-1],minnum[i
                +(1<<(j-1))][j-1]);
}
int askrmq(int a,int b){
    int k=int(log(b-a+1.0)/log(2.0));
    return min(minnum[a][k],minnum[b-(1<<k)+1][k]);
}
int lcp(int a,int b){
    a=rnk[a],b=rnk[b];
    if(a>b)
    swap(a,b);
    return askrmq(a+1,b);
}
```

1.8 HASH

```cpp
#include<bits/stdc++.h>
using namespace std;
typedef unsigned long long ULL;
const int maxn = 305*305;
/* 字符集大小 */
const int sigma = maxn;
/* hash次数 */
const int HASH_CNT = 2;
int n;
int s[maxn];
/* char* 1-bas
* sum[i] = s[i]+s[i-1]*Seed+s[i-2]*Seed^2+...+s[1]*Seed^(i-1)*/
ULL Prime_Pool[] = {1998585857ul,23333333333ul};
ULL Seed_Pool[]={911,146527,19260817,91815541};
ULL Mod_Pool[]={29123,998244353,1000000009,4294967291ull};
struct Hash_1D{
    ULL Seed,Mod;
    ULL bas[maxn];ULL sum[maxn];
    int perm[sigma];
    void init(int seedIndex,int modIndex){
        Seed = Seed_Pool[seedIndex];
        Mod = Mod_Pool[modIndex];
        bas[0]=1;
        for (int i=1;i<=n;i++){
            bas[i] = bas[i-1]*Seed%Mod;
        }
        for (int i=1;i<=n;i++){
            sum[i] = (sum[i-1]*Seed%Mod+s[i])%Mod;
        }
    }
    /*random_shuffle 离散化id，防止kill_hash*/
    void indexInit(int seedIndex,int modIndex){
        for (int i=1;i<n;i++){
            perm[i]=i;
        }
        random_shuffle(perm+1,perm+1+sigma);
        Seed = Seed_Pool[seedIndex];
        Mod = Mod_Pool[modIndex];
        bas[0]=1;
        for (int i=1;i<=n;i++){
            bas[i] = bas[i-1]*Seed%Mod;
        }
        for (int i=1;i<=n;i++){
            sum[i] = (sum[i-1]*Seed%Mod+perm[s[i]])%Mod;
        }
    }
    ULL getHash(int l,int r){
        return (sum[r]-sum[l-1]*bas[r-l+1]%Mod+Mod)%Mod;
    }
}hasher[HASH_CNT];
map<pair<pair<ULL,ULL>,int>,int>veid;int vecnt;
map<string,int>id;int idcnt;
vector<int> pos[maxn];
string a[maxn];
int sumL[maxn];
int main(){
    cin>>n;
    for (int i=1;i<=n;i++){
        cin>>a[i];
        if (!id[a[i]])id[a[i]] = ++idcnt;
        s[i] = id[a[i]];
        sumL[i] = sumL[i-1]+a[i].size();
    }
    for (int i=0;i<HASH_CNT;i++){
        hasher[i].indexInit(i,i);
    }
    int ans = sumL[n]+n-1;
    for (int i=1;i<=n;i++){
        for (int j=1;j<=n;j++){
            ULL hash1 = hasher[0].getHash(i,j);
            ULL hash2 = hasher[1].getHash(i,j);
            int len = j-i+1;
            pair<pair<ULL,ULL>,int> x = {{hash1,hash2},len};
            if (veid[x]==0)veid[x] = ++vecnt;
            pos[veid[x]].push_back(i);
        }
    }
    int maxDelta =0;
    for (auto x:veid){
        int len = x.first.second;
        int i = x.second;
        sort(pos[i].begin(),pos[i].end());
        int num =0;
        for (int j=0,last = -maxn;j<pos[i].size();j++){
            if (pos[i][j]>=last+len){
                last = pos[i][j];
                num++;
            }
        }
        if (num==1)continue;
```

```cpp
        int cost1 = sumL[pos[i][0]+len-1]-sumL[pos[i][0]-1]+len-1;
        int cost2 = len;
        int tempDelta = (cost1-cost2)*num;
        maxDelta = max(maxDelta,tempDelta);
    }
    cout<<ans-maxDelta<<endl;
    return 0;
}
```

2 数据结构

2.1 01Trie

```cpp
/*
数组大小(x+1)*MAX:插入的值的最大值<2^x<MAX
Trie.Insert(1,x,v);
Trie.Delete(1,x,v);
Trie.query(1,x,v);
Trie.clear(1,x);
*/
struct Trie
{
    int cnt[32*MAX],val[32*MAX];
    void Insert(int x,int pos,int v)
    {
        if(pos<0)
        {
            cnt[x]++;
            val[x]=v;
            return;
        }
        Insert((x<<1)|((v>>pos)&1),pos-1,v);
        cnt[x]=cnt[x<<1]+cnt[x<<1|1];
    }
    void Delete(int x,int pos,int v)
    {
        if(pos<0)
        {
            cnt[x]--;
            return;
        }
        Delete((x<<1)|((v>>pos)&1),pos-1,v);
        cnt[x]=cnt[x<<1]+cnt[x<<1|1];
    }
    void clear(int x,int pos)
    {
        cnt[x]=0;
        val[x]=0;
        if(pos<0) return;
        clear(x<<1,pos-1);
        clear(x<<1|1,pos-1);
    }
    int query(int x,int pos,int v)//查询与v异或的最大值 并返回
    {
        if(pos<0) return val[x];
        int temp=(v>>pos)&1;
        temp|=x<<1;
        if(cnt[temp^1]) return query(temp^1,pos-1,v);
        return query(temp,pos-1,v);
    }
}tr;
```

2.2 Trie

```cpp
struct Trie
{
    #define type int
    struct trie
    {
        int v;
        trie *next[26];
        trie()
        {
            v=0;
            for(int i=0;i<26;i++) next[i]=NULL;
        }
    }*root;
    void insert(trie *p,char *s)
    {
        int i=0,t;
        while(s[i])
        {
            t=s[i]-'a';
            if(p->next[t]==NULL) p->next[t]=new trie;
            p=p->next[t];
            p->v++;//按情况改
            i++;
        }
    }
    int find(trie *p,char *s)
    {
        int i=0,t;
```

```
        while(s[i])
        {
            t=s[i]-'a';
            if(p->next[t]==NULL) return 0;
            p=p->next[t];
            i++;
        }
        return p->v;//按情况改
    }
    //删除前缀为s的字符串
    void del(char *s)
    {
        int i=0,t,temp;
        trie *p,*pre;
        pre=p=root;
        while(s[i])
        {
            t=s[i]-'a';
            if(p->next[t]==NULL) return;
            if(!s[i+1])
            {
                temp=p->next[t]->v;
                p->next[t]=NULL;
                break;
            }
            pre=p;
            p=p->next[t];
            i++;
        }
        i=0;
        p=root;
        while(s[i])
        {
            t=s[i]-'a';
            if(p->next[t]==NULL) return;
            p=p->next[t];
            p->v-=temp;
            i++;
        }
    }
    #undef type
}tr;
```

2.3 线段树

```
struct Seg_Tree{
    int val[maxn<<2],lazy[maxn<<2];
    void init(){
        memset(val,0,sizeof(val));
        memset(lazy,0,sizeof(val));
    }
    inline void up(int rt){
        val[rt]=val[rt<<1]+val[rt<<1|1];
    }
    inline void down(int rt,int l,int r){
        int mid=l+r>>1;
        if(lazy[rt]){
            lazy[rt<<1]+=lazy[rt];
            lazy[rt<<1|1]+=lazy[rt];
            val[rt<<1]+=lazy[rt]*(mid-l+1);
            val[rt<<1|1]+=lazy[rt]*(r-mid);
            lazy[rt]=0;
        }
    }
    void build(int rt,int l,int r){
        if(l==r){
            val[rt]=a[l];
            return ;
        }
        int mid=l+r>>1;
        build(rt<<1,l,mid);
        build(rt<<1|1,mid+1,r);
        up(rt);
    }
    void update(int rt,int l,int r,int L,int R,int
         del){
        if(l>R||r<L)return ;
        if(L<=l&&r<=R){
            val[rt]+=del*(r-l+1);
            lazy[rt]+=del;
            return ;
        }
        int mid=l+r>>1;
        down(rt,l,r);
        update(rt<<1,l,mid,L,R,del);
        update(rt<<1|1,mid+1,r,L,R,del);
        up(rt);
    }
    void add(int rt,int l,int r,int L,int del){
        if(l==r){
            val[rt]+=del;
            return ;
        }
        down(rt,l,r);
        int mid=l+r>>1;
```

```cpp
        if(L<=mid)add(rt<<1,l,mid,L,del);
        else add(rt<<1|1,mid+1,r,L,del);
        up(rt);
    }
    int query_sum(int rt,int l,int r,int L,int R){
        if(L<=l&&r<=R)return val[rt];
        int mid=l+r>>1;
        down(rt,l,r);
        int res=0;
        if(L<=mid)res+=query_sum(rt<<1,l,mid,L,R);
        if(R>mid)res+=query_sum(rt<<1|1,mid+1,r,L,R);
        return res;
    }
}seg;
```

2.4 主席树

```cpp
#include<bits/stdc++.h>
using namespace std;
const int maxn=1e5+100;
int a[maxn];int rk[maxn];int pos[maxn];
int root[maxn];int cnt,m,n,T;
struct Chairman_Tree{
    struct Node{int L,R,val;}tree[maxn*500];
    void init(){
        memset(root,0,sizeof root);
        cnt =0;
    }
    /* 建T0空树 */
    int buildT0(int l, int r){
        int k = cnt++;
        tree[k].val =0;
        if (l==r) return k;
        int mid = l+r >>1;
        tree[k].L = buildT0(l, mid);tree[k].R = buildT0(mid + 1, r);
        return k;
    }
    /* 上一个版本节点P, 【ppos】+=del 返回新版本节点*/
    int update (int P,int l,int r,int ppos,int del){
        int k = cnt++;
        tree[k].val = tree[P].val +del;
        if (l==r) return k;
        int mid = l+r >>1;
        if (ppos<=mid){
            tree[k].L = update(tree[P].L,l,mid,ppos,del);
            tree[k].R = tree[P].R;
        }else{
            tree[k].L = tree[P].L;
            tree[k].R = update(tree[P].R,mid+1,r,ppos,del);
        }
        return k;
    }
    int query_kth(int lt,int rt,int l,int r,int k){
        if (l==r) return a[rk[l]];
        int mid = l+r >>1;
        if (tree[tree[rt].L].val-tree[tree[lt].L].val>=k) return query_kth(tree[lt].L,tree[rt].L,l,mid,k);
        else return query_kth(tree[lt].R,tree[rt].R,mid+1,r,k+tree[tree[lt].L].val-tree[tree[rt].L].val);
    }
}tree;
bool cmp(int x,int y){return a[x]<a[y];}
int main() {
    scanf("%d", &T);
    while (T--) {
        scanf("%d%d",&n,&m);
        for (int i=1;i<=n;i++){
            scanf("%d",&a[i]);
            rk[i]=i;
        }
        tree.init();
        sort(rk+1,rk+1+n,cmp);
        for (int i1=1;i1<=n;i1++){
            pos[rk[i1]] =i1;
        }
        root[0] = tree.buildT0(1, n);
        for (int i1=1;i1<=n;i1++){
            root[i1] = tree.update(root[i1-1],1,n,pos[i1],1);
        }
        while (m--){
            int l,r,k;scanf("%d%d%d",&l,&r,&k);
            printf("%d\n",tree.query_kth(root[l-1],root[r],1,n,k));
        }
    }
    return 0;
}
```

2.5 HLD

```
/*
size[]数组，以x为根的子树节点个数
top[]数组，当前节点的所在链的顶端节点
son[]数组，重儿子
deep[]数组，当前节点的深度
fa[]数组，当前节点的父亲
idx[]数组，树中每个节点剖分后的新编号
rnk[]数组，idx的逆，表示线段上中当前位置表示哪个节点
*/
struct HLD
{
    #define type int
    struct edge{int a,b;type v;edge(int _a,int _b,
        type _v=0):a(_a),b(_b),v(_v){}};
    struct node{int to;type w;node(){}node(int _to,
        type _w):to(_to),w(_w){}};
    vector<int> mp[MAX];
    vector<edge> e;
    int deep[MAX],fa[MAX],size[MAX],son[MAX];
    int rnk[MAX],top[MAX],idx[MAX],tot;
    int n,rt;
    void init(int _n)
    {
        n=_n;
        for(int i=0;i<=n;i++) mp[i].clear();
        e.clear();
        e.pb(edge(0,0));
    }
    void add_edge(int a,int b,type v=0)
    {
        e.pb(edge(a,b,v));
        mp[a].pb(b);
        mp[b].pb(a);
    }
    void dfs1(int x,int pre,int h)
    {
        int i,to;
        deep[x]=h;
        fa[x]=pre;
        size[x]=1;
        for(i=0;i<sz(mp[x]);i++)
        {
            to=mp[x][i];
            if(to==pre) continue;
            dfs1(to,x,h+1);
            size[x]+=size[to];
            if(son[x]==-1||size[to]>size[son[x]]) son
                [x]=to;
        }
    }
    void dfs2(int x,int tp)
    {
        int i,to;
        top[x]=tp;
        idx[x]=++tot;
        rnk[idx[x]]=x;
        if(son[x]==-1) return;
        dfs2(son[x],tp);
        for(i=0;i<sz(mp[x]);i++)
        {
            to=mp[x][i];
            if(to!=son[x]&&to!=fa[x]) dfs2(to,to);
        }
    }
    void work(int _rt)
    {
        int i;
        rt=_rt;
        mem(son,-1);
        tot=0;
        dfs1(rt,0,0);
        dfs2(rt,rt);
    }
    int LCA(int x,int y)
    {
        while(top[x]!=top[y])
        {
            if(deep[top[x]]<deep[top[y]]) swap(x,y);
            x=fa[top[x]];
        }
        if(deep[x]>deep[y]) swap(x,y);
        return x;
    }
    //node
    void init_node()
    {
        build(n);
    }
    void modify_node(int x,int y,type val)
    {
        while(top[x]!=top[y])
        {
            if(deep[top[x]]<deep[top[y]]) swap(x,y);
```

```
            update(idx[top[x]],idx[x],val);
            x=fa[top[x]];
        }
        if(deep[x]>deep[y]) swap(x,y);
        update(idx[x],idx[y],val);
    }
    type query_node(int x,int y)
    {
        type res=0;
        while(top[x]!=top[y])
        {
            if(deep[top[x]]<deep[top[y]]) swap(x,y);
            res+=query(idx[top[x]],idx[x]);
            x=fa[top[x]];
        }
        if(deep[x]>deep[y]) swap(x,y);
        res+=query(idx[x],idx[y]);
        return res;
    }
    //path
    void init_path()
    {
        v[idx[rt]]=0;
        for(int i=1;i<n;i++)
        {
            if(deep[e[i].a]<deep[e[i].b]) swap(e[i].a,e[i].b);
            v[idx[e[i].a]]=e[i].v;
        }
        build(n);
    }
    void modify_edge(int id,type val)
    {
        if(deep[e[id].a]>deep[e[id].b]) update(idx[e[id].a],idx[e[id].a],val);
        else update(idx[e[id].b],idx[e[id].b],val);
    }
    void modify_path(int x,int y,type val)
    {
        while(top[x]!=top[y])
        {
            if(deep[top[x]]<deep[top[y]]) swap(x,y);
            update(idx[top[x]],idx[x],val);
            x=fa[top[x]];
        }
        if(deep[x]>deep[y]) swap(x,y);
        if(x!=y) update(idx[x]+1,idx[y],val);
    }
    type query_path(int x,int y)
    {
        type res=0;
        while(top[x]!=top[y])
        {
            if(deep[top[x]]<deep[top[y]]) swap(x,y);
            res+=query(idx[top[x]],idx[x]);
            x=fa[top[x]];
        }
        if(deep[x]>deep[y]) swap(x,y);
        if(x!=y) res+=query(idx[x]+1,idx[y]);
        return res;
    }
    #undef type
}hld;
/**********attention!***********/
//hld.init(n)
//hld.add_edge(): undirected edge.
/*******************************/
```

3 数学

3.1 矩阵

```
struct Mat {
    static const LL M = 2;
    LL v[M][M];
    Mat() { memset(v, 0, sizeof v); }
    void eye() { FOR (i, 0, M) v[i][i] = 1; }
    LL* operator [] (LL x) { return v[x]; }
    const LL* operator [] (LL x) const { return v[x]; }
    Mat operator * (const Mat& B) {
        const Mat& A = *this;
        Mat ret;
        FOR (k, 0, M)
        FOR (i, 0, M) if (A[i][k])
        FOR (j, 0, M)
        ret[i][j] = (ret[i][j] + A[i][k] * B[k][j]) % MOD;
        return ret;
    }
    Mat pow(LL n) const {
        Mat A = *this, ret; ret.eye();
        for (; n; n >>= 1, A = A * A)
        if (n & 1) ret = ret * A;
        return ret;
```

```
        }
        Mat operator + (const Mat& B) {
            const Mat& A = *this;
            Mat ret;
            FOR (i, 0, M)
            FOR (j, 0, M)
            ret[i][j] = (A[i][j] + B[i][j]) % MOD;
            return ret;
        }
        void prt() const {
            FOR (i, 0, M)
            FOR (j, 0, M)
            printf("%lld%c", (*this)[i][j], j == M - 1 ?
                '\n' : ' ');
        }
    };
```

3.2 快速乘

```
LL mul(LL a, LL b, LL m) {
    LL ret = 0;
    while (b) {
        if (b & 1) {
            ret += a;
            if (ret >= m) ret -= m;
        }
        a += a;
        if (a >= m) a -= m;
        b >>= 1;
    }
    return ret;
}
```

3.3 快速幂

如果模数是素数，则可在函数体内加上 n

```
LL qpow(LL a,LL b,LL Mod){
    LL ret=1;
    while(b){
        if(b&1)ret=(ret*a)%Mod;
        a=(a*a)%Mod;
        b>>=1;
    }
    return ret;
}
```

3.4 筛

3.4.1 线性筛素数

```
const LL p_max=1e6;
LL prime[p_max+100],p_sz;//用vector存素数能优化时间
void get_prime(){
    bool vis[p_max+100];
    for(int i=2;i<=p_max;i++){
        if(!vis[i])prime[p_sz++]=i;
        for(int j=0;j<p_sz&&prime[j]*i<=p_max;j++){
            vis[prime[j]*i]=1;
            if(i%prime[j]==0)break;
        }
    }
}
```

3.4.2 线性筛欧拉函数

```
const LL p_max=1e6;
LL phi[p_max+100],prime[p_max+100],p_sz=0;
bool vis[p_max+100];
void get_phitable(){
    phi[1]=1;
    for(int i=2;i<=p_max;i++){
        if(!vis[i]){
        prime[p_sz++]=i;
        phi[i]=i-1;
        }
        LL d;
        for(int j=0;j<p_sz&&(d=prime[j]*i)<=p_max;j
            ++){
            vis[d]=1;
            if(i%prime[j]==0){
                phi[d]=phi[i]*prime[j];
                break;
            }
            else phi[d]=phi[i]*(prime[j]-1);
        }
    }
}
```

3.4.3 线性筛莫比乌斯函数

```
const LL p_max=1e6;
```

```
LL mu[p_max+100],prime[p_max+100],p_sz;
bool vis[p_max+100];
void get_mutable(){
    mu[1]=1;
    for(int i=2;i<=p_max;i++){
        if(!vis[i]){
            prime[p_sz++]=i;
            mu[i]=-1;
        }
    LL d;
    for(int j=0;j<p_sz&&(d=prime[j]*i)<=p_max;j++){
        vis[d]=1;
        if(i%prime[j]==0){
            mu[d]=0;
            break;
        }
        else mu[d]=-mu[i];
        }
    }
}
```

3.4.4 杜教筛

求 $S(n)=\sum_{i=1}^{n} f(i)$，其中 f 是一个积性函数。

构造一个积性函数 g，那么由 $(f*g)(n)=\sum_{d|n} f(d)g(\frac{n}{d})$，得到 $f(n)=(f*g)(n)-\sum_{d|n,d<n} f(d)g(\frac{n}{d})$。

$$
\begin{aligned}
g(1)S(n) &= \sum_{i=1}^{n}(f*g)(i)-\sum_{i=1}^{n}\sum_{d|i,d<i} f(d)g(\frac{n}{d}) && (1)\\
&\overset{t=\frac{i}{d}}{=} \sum_{i=1}^{n}(f*g)(i)-\sum_{t=2}^{n} g(t)S(\lfloor\frac{n}{t}\rfloor) && (2)
\end{aligned}
$$

当然，要能够由此计算 $S(n)$，会对 f,g 提出一些要求：

$f*g$ 要能够快速求前缀和。

g 要能够快速求分段和（前缀和）。

对于正常的积性函数 $g(1)=1$，所以不会有什么问题。

在预处理 $S(n)$ 前 $n^{\frac{2}{3}}$ 项的情况下复杂度是 $O(n^{\frac{2}{3}})$。

杜教筛筛欧拉函数

```
#include<iostream>
#include<unordered_map>
#include<vector>
using namespace std;
typedef long long ll;
const int maxn=5e6;
const int mod=1e9+7;
int Madd(int a,int b){
    return a+b<mod?a+b:a+b-mod;
}
int Msub(int a,int b){
    return a<b?mod+a-b:a-b;
}
int Mmul(int a,int b){
    return (ll)a*b%mod;
}
vector<int>p;
int vis[maxn+10];
int phi[maxn+10];
unordered_map<ll,ll>map;
ll qphi(ll n){
    if(n<maxn)return phi[n];
    auto it=map.find(n);
    if(it!=map.end())
        return it->second;
    ll res=n&1?Mmul((n+1)/2%mod,n%mod):Mmul(n/2%mod
        ,(n+1)%mod);
    for(ll i=2,last;i<=n;i=last+1){
        last=n/(n/i);
        res=Msub(res,Mmul((last-i+1)%mod,qphi(n/i)));
    }
    map.emplace(n,res);
    return res;
}
void init(){
    phi[1]=1;
    for(int i=2;i<maxn;i++){
        if(!vis[i]){
        p.push_back(i);
        phi[i]=i-1;
    }
    for(int j=0;j<(int)p.size()&&i*p[j]<maxn;j++){
        vis[i*p[j]]=1;
        if(i%p[j])
            phi[i*p[j]]=phi[i]*(p[j]-1);
        else{
            phi[i*p[j]]=phi[i]*p[j];
            break;
        }
    }
}
for(int i=2;i<maxn;i++)
    phi[i]=(phi[i]+phi[i-1])%mod;
}
int main(){
    ios_base::sync_with_stdio(false);
    cin.tie(0);
    init();
```

```cpp
    long long a;
    cin>>a;
    cout<<qphi(a)<<endl;
    return 0;
}
```

杜教筛筛莫比乌斯函数

```cpp
#include<iostream>
#include<unordered_map>
#include<vector>
using namespace std;
typedef long long ll;
const int maxn=5e6;

vector<int>p;
int vis[maxn+10];
int mu[maxn+10];
unordered_map<ll,ll>map;
ll qmu(ll n){
    if(n<maxn)return mu[n];
    auto it=map.find(n);
    if(it!=map.end())
        return it->second;
    ll res=1;
    for(ll i=2,last;i<=n;i=last+1){
        last=n/(n/i);
        res-=(ll)(last-i+1)*qmu(n/i);
    }
    map.emplace(n,res);
    return res;
}
void init(){
    mu[1]=1;
    for(int i=2;i<maxn;i++){
        if(!vis[i]){
        p.push_back(i);
        mu[i]=-1;
    }
    for(int j=0;j<(int)p.size()&&i*p[j]<maxn;j++){
        vis[i*p[j]]=1;
        if(i%p[j])
            mu[i*p[j]]=-mu[i];
        else
            break;
        }
    }
    for(int i=2;i<maxn;i++)
        mu[i]+=mu[i-1];
}
int main(){
    ios_base::sync_with_stdio(false);
    cin.tie(0);
    init();
    long long a,b;
    cin>>a>>b;
    cout<<qmu(b)-qmu(a-1)<<endl;
    return 0;
}
```

3.5 素数测试

3.5.1 素数判断

前置：快速乘、快速幂 int 范围内只需检查 2, 7, 61 long long 范围 2, 325, 9375, 28178, 450775, 9780504, 1795265022 3E15 内 2, 2570940, 880937, 610386380, 4130785767 4E13 内 2, 2570940, 211991001, 3749873356

```cpp
bool checkQ(LL a, LL n) {
    if (n == 2 || a >= n) return 1;
    if (n == 1 || !(n & 1)) return 0;
    LL d = n - 1;
    while (!(d & 1)) d >>= 1;
    LL t = bin(a, d, n); // 不一定需要快速乘
    while (d != n - 1 && t != 1 && t != n - 1) {
        t = mul(t, t, n);
        d <<= 1;
    }
    return t == n - 1 || d & 1;
}
bool primeQ(LL n) {
    static vector<LL> t = {2, 325, 9375, 28178,
        450775, 9780504, 1795265022};
    if (n <= 1) return false;
    for (LL k: t) if (!checkQ(k, n)) return false;
    return true;
}
```

3.5.2 Pollard-Rho

```cpp
mt19937 mt(time(0));
LL pollard_rho(LL n, LL c) {
    LL x = uniform_int_distribution<LL>(1, n - 1)(mt
        ), y = x;
    auto f = [&](LL v) { LL t = mul(v, v, n) + c;
        return t < n ? t : t - n; };
    while (1) {
```

```
        x = f(x); y = f(f(y));
        if (x == y) return n;
        LL d = gcd(abs(x - y), n);
        if (d != 1) return d;
    }
}

LL fac[100], fcnt;
void get_fac(LL n, LL cc = 19260817) {
    if (n == 4) { fac[fcnt++] = 2; fac[fcnt++] = 2; return; }
    if (primeQ(n)) { fac[fcnt++] = n; return; }
    LL p = n;
    while (p == n) p = pollard_rho(n, --cc);
    get_fac(p); get_fac(n / p);
}
```

3.6 BM 线性递推

```
typedef vector<int> VI;
namespace linear_seq
{
    #define rep(i,a,n) for (int i=a;i<n;i++)
    #define SZ(x) ((int)(x).size())
    const ll mod=1e9+7;
    ll powmod(ll a,ll b){ll res=1;a%=mod; assert(b>=0); for(;b;b>>=1){if(b&1)res=res*a%mod;a=a*a%mod;}return res;}
        const int N=10010;
        ll res[N],base[N],_c[N],_md[N];
        vector<int> Md;
        void mul(ll *a,ll *b,int k)
        {
            rep(i,0,k+k) _c[i]=0;
            rep(i,0,k) if (a[i]) rep(j,0,k) _c[i+j]=(_c[i+j]+a[i]*b[j])%mod;
            for (int i=k+k-1;i>=k;i--) if (_c[i])
            rep(j,0,SZ(Md)) _c[i-k+Md[j]]=(_c[i-k+Md[j]]-_c[i]*_md[Md[j]])%mod;
            rep(i,0,k) a[i]=_c[i];
        }
        int solve(ll n,VI a,VI b){
            ll ans=0,pnt=0;
            int k=SZ(a);
            assert(SZ(a)==SZ(b));
            rep(i,0,k) _md[k-1-i]=-a[i];_md[k]=1;
            Md.clear();
            rep(i,0,k) if (_md[i]!=0) Md.push_back(i);
            rep(i,0,k) res[i]=base[i]=0;
            res[0]=1;
            while ((1ll<<pnt)<=n) pnt++;
            for (int p=pnt;p>=0;p--) {
                mul(res,res,k);
                if ((n>>p)&1) {
                    for (int i=k-1;i>=0;i--) res[i+1]=res[i];res[0]=0;
                    rep(j,0,SZ(Md)) res[Md[j]]=(res[Md[j]]-res[k]*_md[Md[j]])%mod;
                }
            }
            rep(i,0,k) ans=(ans+res[i]*b[i])%mod;
            if (ans<0) ans+=mod;
            return ans;
        }
        VI BM(VI s){
            VI C(1,1),B(1,1);
            int L=0,m=1,b=1;
            rep(n,0,SZ(s)){
                ll d=0;
                rep(i,0,L+1) d=(d+(ll)C[i]*s[n-i])%mod;
                if(d==0) ++m;
                else if(2*L<=n){
                    VI T=C;
                    ll c=mod-d*powmod(b,mod-2)%mod;//ÄæÔª
                    while (SZ(C)<SZ(B)+m) C.pb(0);
                    rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
                    L=n+1-L; B=T; b=d; m=1;
                } else {
                    ll c=mod-d*powmod(b,mod-2)%mod;//ÄæÔª
                    while (SZ(C)<SZ(B)+m) C.pb(0);
                    rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
                    ++m;
                }
            }
            return C;
        }
        int gao(VI a,ll n)
        {
            VI c=BM(a);
```

```
            c.erase(c.begin());
            rep(i,0,SZ(c)) c[i]=(mod-c[i])%mod;
            return solve(n,c,VI(a.begin(),a.begin()+SZ(c)));
        }
    };//linear_seq::gao(VI{},n-1)
```

3.7 扩展欧几里德

```
/*
解xa+yb=gcd(a,b)
返回值为gcd(a,b)
其中一组解为x y
通解:
x1=x+b/gcd(a,b)*t
y1=y-a/gcd(a,b)*t
(t为任意整数)
*/
ll exgcd(ll a,ll b,ll &x,ll &y)
{
    if(b==0)
    {
        x=1;
        y=0;
        return a;
    }
    ll g,t;
    g=exgcd(b,a%b,x,y);
    t=x;
    x=y;
    y=t-a/b*y;
    return g;
}
//xa+yb=c 有解条件 c%gcd(a,b)==0
ll linear_equation(ll a,ll b,ll c,ll &x,ll &y)
{
    ll g,t;
    g=exgcd(a,b,x,y);
    if(!c) x=y=0;
    else if((!a&&!b&&c)||c%g) return -1;//no solution
    else if(!a&&b) x=1,y=c/b;
    else if(a&&!b) x=c/a,y=-c/a;
    else
    {
        a/=g,b/=g,c/=g;
        x*=c,y*=c;
        t=x;
        x%=b;
        if(x<=0) x+=b;//or x<0
        ll k=(t-x)/b;
        y+=k*a;
    }
    return g;
}
```

3.8 线性基

```
struct Base
{
    #define type ll
    #define mx 60
    type d[mx+3];
    int p[mx+3],cnt;
    void init()
    {
        memset(d,0,sizeof(d));
        cnt=0;
    }
    bool insert(type x,int pos=0)
    {
        int i;
        for(i=mx;~i;i--)
        {
            if(!(x&(1LL<<i))) continue;
            if(!d[i])
            {
                cnt++;
                d[i]=x;
                p[i]=pos;
                break;
            }
            if(p[i]<pos)
            {
                swap(d[i],x);
                swap(p[i],pos);
            }
            x^=d[i];
        }
        return x>0;
    }
    type query_max(int pos=-1)
    {
        int i;
```

```
        type res=0;
        for(i=mx;~i;i--)
        {
            if(p[i]>=pos)
            {
                if((res^d[i])>res) res^=d[i];
            }
        }
        return res;
    }
    type query_min(int pos=-1)
    {
        for(int i=0;i<=mx;i++)
        {
            if(d[i]&&p[i]>=pos) return d[i];
        }
        return 0;
    }
    void merge(Base x)
    {
        if(cnt<x.cnt)
        {
            swap(cnt,x.cnt);
            swap(d,x.d);
            swap(p,x.p);
        }
        for(int i=mx;~i;i--)
        {
            if(x.d[i]) insert(x.d[i]);
        }
    }
    //kth min
    //first use rebuild()
    type tp[mx+3];
    void rebuild()
    {
        int i,j;
        cnt=0;
        for(i=mx;~i;i--)
        {
            for(j=i-1;~j;j--)
            {
                if(d[i]&(1LL<<j)) d[i]^=d[j];
            }
        }
        for(i=0;i<=mx;i++)
        {
            if(d[i]) tp[cnt++]=d[i];
            }
        }
        type kth(type k)
        {
            type res=0;
            if(k>=(1LL<<cnt)) return -1;
            for(int i=mx;~i;i--)
            {
                if(k&(1LL<<i)) res^=tp[i];
            }
            return res;
        }
};
```

3.9 exBSGS

```
//a^x¡=b (mod c)
ll exBSGS(ll a,ll b,ll c)
{
    ll i,g,d,num,now,sq,t,x,y;
    if(c==1) return b?-1:(a!=1);
    if(b==1) return a?0:-1;
    if(a%c==0) return b?-1:1;
    num=0;
    d=1;
    while((g=__gcd(a,c))>1)
    {
        if(b%g) return -1;
        num++;
        b/=g;
        c/=g;
        d=(d*a/g)%c;
        if(d==b) return num;
    }
    mp.clear();
    sq=ceil(sqrt(c));
    t=1;
    for(i=0;i<sq;i++)
    {
        if(!mp.count(t)) mp[t]=i;
        else mp[t]=min(mp[t],i);
        t=t*a%c;
    }
    for(i=0;i<sq;i++)
    {
        exgcd(d,c,x,y);
        x=(x*b%c+c)%c;
```

```cpp
            if(mp.count(x)) return i*sq+mp[x]+num;
            d=d*t%c;
        }
        return -1;
    }
```

3.10 中国剩余定理

```cpp
//m是除数 r是余数 p是除数的LCM(也就是答案的循环节)
int CRT(int *m,int *r,int n)
{
    int p=m[0],res=r[0],x,y,g;
    for(int i=1;i<n;i++)
    {
        g=exgcd(p,m[i],x,y);
        if((r[i]-res)%g) return -1;//无解
        x=(r[i]-res)/g*x%(m[i]/g);
        res+=x*p;
        p=p/g*m[i];
        res%=p;
    }
    return res>0?res:res+p;
}
```

3.11 类欧几里德

$m=\lfloor\frac{an+b}{c}\rfloor$.

$f(a,b,c,n)=\sum_{i=0}^{n}\lfloor\frac{ai+b}{c}\rfloor$: 当 $a\geq c$ or $b\geq c$ 时, $f(a,b,c,n)=(\frac{a}{c})n(n+1)/2+(\frac{b}{c})(n+1)+f(a \bmod c,b \bmod c,c,n)$；否则 $f(a,b,c,n)=nm-f(c,c-b-1,a,m-1)$。

$g(a,b,c,n)=\sum_{i=0}^{n}i\lfloor\frac{ai+b}{c}\rfloor$: 当 $a\geq c$ or $b\geq c$ 时, $g(a,b,c,n)=(\frac{a}{c})n(n+1)(2n+1)/6+(\frac{b}{c})n(n+1)/2+g(a \bmod c,b \bmod c,c,n)$；否则 $g(a,b,c,n)=\frac{1}{2}(n(n+1)m-f(c,c-b-1,a,m-1)-h(c,c-b-1,a,m-1))$。

$h(a,b,c,n)=\sum_{i=0}^{n}\lfloor\frac{ai+b}{c}\rfloor^2$: 当 $a\geq c$ or $b\geq c$ 时, $h(a,b,c,n)=(\frac{a}{c})^2n(n+1)(2n+1)/6+(\frac{b}{c})^2(n+1)+(\frac{a}{c})(\frac{b}{c})n(n+1)+h(a \bmod c,b \bmod c,c,n)+2(\frac{a}{c})g(a \bmod c,b \bmod c,c,n)+2(\frac{b}{c})f(a \bmod c,b \bmod c,c,n)$；否则 $h(a,b,c,n)=nm(m+1)-2g(c,c-b-1,a,m-1)-2f(c,c-b-1,a,m-1)-f(a,b,c,n)$。

3.12 逆元

预处理 1 n 的逆元

```cpp
LL inv[N] = {-1, 1};
void inv_init(LL n, LL p) {
    inv[1] = 1;
    for(int i=2;i<=n;i++)
        inv[i] = (p - p / i) * inv[p % i] % p;
}
```

预处理阶乘及其逆元

```cpp
LL invf[M], fac[M] = {1};
void fac_inv_init(LL n, LL p) {
    for(int i = 2;i <= n ; i++)
        fac[i] = i * fac[i - 1] % p;
    invf[n - 1] = qpow(fac[n - 1], p - 2, p);
    for(int i = n-1;i >= 0 ; i--)
        invf[i] = invf[i + 1] * (i + 1) % p;
}
```

3.13 组合数

如果数较小，模较大时使用逆元前置模板：逆元-预处理阶乘及其逆元

```cpp
inline LL C(LL n, LL m) { // n >= m >= 0
    return n < m || m < 0 ? 0 : fac[n] * invf[m] %
        MOD * invf[n - m] % MOD;
}
```

如果模数较小，数字较大，使用 Lucas 定理

前置模板可选 1：求组合数（如果使用阶乘逆元，需

```cpp
fac_inv_init(MOD, MOD)
```

前置模板可选 2：模数不固定下使用，无法单独使用。

```cpp
LL C(LL n, LL m) { // m >= n >= 0
    if (m - n < n) n = m - n;
    if (n < 0) return 0;
    LL ret = 1;
    for(int i=1;i<=n;i++)
        ret = ret * (m - n + i) % MOD * qpow(i, MOD -
            2, MOD) % MOD;
    return ret;
}

LL Lucas(LL n, LL m) { // m >= n >= 0
    return m ? C(n % MOD, m % MOD) * Lucas(n / MOD,
        m / MOD) % MOD : 1;
}
```

3.14 公式

3.14.1 数论公式

当 $x\geq\phi(p)$ 时有 $a^x\equiv a^{x \bmod \phi(p)+\phi(p)} \pmod p$

$\mu^2(n)=\sum_{d^2|n}\mu(d)$

$\sum_{d|n}\varphi(d)=n$

$\sum_{d|n}2^{\omega(d)}=\sigma_0(n^2)$，其中 ω 是不同素因子个数

$\sum_{d|n}\mu^2(d)=2^{\omega(d)}$

3.14.2 一些数论函数求和的例子

$\sum_{i=1}^n i[gcd(i,n)=1]=\frac{n\varphi(n)+[n=1]}{2}$

$\sum_{i=1}^n\sum_{j=1}^m[gcd(i,j)=x]=\sum_d\mu(d)\lfloor\frac{n}{dx}\rfloor\lfloor\frac{m}{dx}\rfloor$

$\sum_{i=1}^n\sum_{j=1}^m gcd(i,j) = \sum_{i=1}^n\sum_{j=1}^m\sum_{d|gcd(i,j)}\varphi(d) = \sum_d\varphi(d)\lfloor\frac{n}{d}\rfloor\lfloor\frac{m}{d}\rfloor$

$S(n) = \sum_{i=1}^n\mu(i) = 1-\sum_{i=1}^n\sum_{d|i,d<i}\mu(d) \overset{t=\frac{i}{d}}{=} 1-\sum_{t=2}^n S(\lfloor\frac{n}{t}\rfloor)$

利用 $[n=1]=\sum_{d|n}\mu(d)$

$S(n)=\sum_{i=1}^n\varphi(i)=\sum_{i=1}^n i-\sum_{i=1}^n\sum_{d|i,d<i}\varphi(i) \overset{t=\frac{i}{d}}{=} \frac{i(i+1)}{2}-\sum_{t=2}^n S(\frac{n}{t})$

利用 $n=\sum_{d|n}\varphi(d)$

$\sum_{i=1}^n\mu^2(i)=\sum_{i=1}^n\sum_{d^2|n}\mu(d)=\sum_{d=1}^{\lfloor\sqrt{n}\rfloor}\mu(d)\lfloor\frac{n}{d^2}\rfloor$

$\sum_{i=1}^n\sum_{j=1}^n gcd^2(i,j)=\sum_d d^2\sum_t\mu(t)\lfloor\frac{n}{dt}\rfloor^2 \overset{x=dt}{=} \sum_x\lfloor\frac{n}{x}\rfloor^2\sum_{d|x}d^2\mu(\frac{t}{x})$

$\sum_{i=1}^n\varphi(i)=\frac{1}{2}\sum_{i=1}^n\sum_{j=1}^n[i\perp j]-1=\frac{1}{2}\sum_{i=1}^n\mu(i)\cdot\lfloor\frac{n}{i}\rfloor^2-1$

3.14.3 莫比乌斯反演

$g(n)=\sum_{d|n}f(d)\Leftrightarrow f(n)=\sum_{d|n}\mu(d)g(\frac{n}{d})$

$f(n)=\sum_{n|d}g(d)\Leftrightarrow g(n)=\sum_{n|d}\mu(\frac{d}{n})f(d)$

3.14.4 低阶等幂求和

$\sum_{i=1}^n i^1=\frac{n(n+1)}{2}=\frac{1}{2}n^2+\frac{1}{2}n$

$\sum_{i=1}^n i^2=\frac{n(n+1)(2n+1)}{6}=\frac{1}{3}n^3+\frac{1}{2}n^2+\frac{1}{6}n$

$\sum_{i=1}^n i^3=\left[\frac{n(n+1)}{2}\right]^2=\frac{1}{4}n^4+\frac{1}{2}n^3+\frac{1}{4}n^2$

$\sum_{i=1}^n i^4=\frac{n(n+1)(2n+1)(3n^2+3n-1)}{30}=\frac{1}{5}n^5+\frac{1}{2}n^4+\frac{1}{3}n^3-\frac{1}{30}n$

$\sum_{i=1}^n i^5=\frac{n^2(n+1)^2(2n^2+2n-1)}{12}=\frac{1}{6}n^6+\frac{1}{2}n^5+\frac{5}{12}n^4-\frac{1}{12}n^2$

3.14.5 一些组合公式

错排公式：$D_1=0,D_2=1,D_n=(n-1)(D_{n-1}+D_{n-2})=n!(\frac{1}{2!}-\frac{1}{3!}+\cdots+(-1)^n\frac{1}{n!})=\lfloor\frac{n!}{e}+0.5\rfloor$

卡塔兰数（n 对括号合法方案数，n 个结点二叉树个数，$n\times n$ 方格中对角线下方的单调路径数，凸 $n+2$ 边形的三角形划分数，n 个元素的合法出栈序列数）：$C_n=\frac{1}{n+1}\binom{2n}{n}=\frac{(2n)!}{(n+1)!n!}$

3.15 Theorem

费马平方和定理：奇素数能表示为两个平方数之和的充分必要条件是该素数被 4 除余 1。

4 图论

4.1 k 短路

POJ 2449

```
#include<bits/stdc++.h>
using namespace std;
const int maxN=10000;
const int INF=0x3f3f3f3f;
typedef pair<int,int>P;
int n,m,s,t,k;
int dist[maxN],tdist[maxN],cnt[maxN];
bool f[maxN];
vector<P>Adj[maxN];
vector<P>Rev[maxN];
struct edge{
    int to,len;
    edge(){}
    edge(int t,int l):to(t),len(l){}
};
priority_queue<edge> q;
bool operator<(const edge &a,const edge &b){
    return (a.len+dist[a.to])>(b.len+dist[b.to]);
}
void dijkstra(){
    memset(dist,0,sizeof(dist));
    fill(tdist,tdist+maxN,INF);
    tdist[t]=0;
    while(!q.empty())q.pop();
    q.push(edge(t,0));
    while(!q.empty()){
        int x=q.top().to;
        int d=q.top().len;
        q.pop();
        if(tdist[x]<d)continue;
        for(int i=0;i<(int)Rev[x].size();i++){
            int y=Rev[x][i].first;
            int len=Rev[x][i].second;
            if(d+len<tdist[y]){
                tdist[y]=d+len;
                q.push(edge(y,tdist[y]));
            }
        }
    }
    for(int i=1;i<=n;i++)
        dist[i]=tdist[i];
}
int aStar(){
```

```cpp
        if(dist[s]==INF)return -1;
        while(!q.empty())q.pop();
        q.push(edge(s,0));
        memset(cnt,0,sizeof(cnt));
        while(!q.empty()){
            int x=q.top().to;
            int d=q.top().len;
            q.pop();
            cnt[x]++;
            if(cnt[t]==k)return d;
            if(cnt[x]>k)continue;
            for(int i=0;i<(int)Adj[x].size();i++){
                int y=Adj[x][i].first;
                int len=Adj[x][i].second;
                q.push(edge(y,d+len));
                }
            }
        return -1;
    }
    int main(){
        scanf("%d%d",&n,&m);
        for(int i=0;i<m;i++){
            int st,ed,l;
            scanf("%d%d%d",&st,&ed,&l);
            Adj[st].push_back(make_pair(ed,l));
            Rev[ed].push_back(make_pair(st,l));
        }
        scanf("%d%d%d",&s,&t,&k);
        if(s==t)k++;
        dijkstra();
        printf("%d\n",aStar());
        return 0;
    }
```

4.2 最大流

```cpp
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int maxn = 11000;
const int maxm = 110000;
const int INF = 0x3f3f3f3f;
struct Max_Flow{
    int first[maxn],nxt[maxm*2],des[maxm*2],c[maxm*2],tot;
    int dep[maxn];int ss,tt;
    Max_Flow(){ clear(); }
    void clear(){
        memset(first,-1,sizeof first);tot =-1;
    }
    inline void addEdge(int u,int v,int w){
        tot++;
        des[tot] = v;c[tot] =w;
        nxt[tot] = first[u];first[u] = tot;
    }
    bool bfs(){
        memset(dep,-1,sizeof dep);
        dep[ss] =0;
        queue<int> Q;Q.push(ss);
        while (!Q.empty()){
            int q = Q.front();Q.pop();
            for (int t = first[q];t!=-1;t= nxt[t]){
            int v = des[t],cx = c[t];
            if (dep[v]==-1&&cx){
                dep[v] = dep[q]+1;
                    Q.push(v);
                }
            }
        }
        return dep[tt]!=-1;
    }
    int dfs(int node,int now){
        if (node==tt)return now;
        int res =0;
        for (int t = first[node];t!=-1&&res<now;t=nxt[t]){
            int v = des[t],cx = c[t];
            if (dep[v]==dep[node]+1&&cx){
                int x = min(cx,now-res);
                x = dfs(v,x);
                res+=x;c[t]-=x;c[t^1]+=x;
            }
        }
        if (!res) dep[node] = -2;
        return res;
    }
    // tuple<from,to,flow>
    void init(vector<tuple<int,int,int> > Edge){
        for (auto tp : Edge){
            int u,v,w;tie(u,v,w) = tp;
            addEdge(u,v,w);addEdge(v,u,0);
        }
    }
    // s->t max_flow
    ll max_flow(int s,int t){
```

```
        ss = s;tt = t;
        ll res =0,del =0;
        while (bfs()){while (del = dfs(ss,INF)){res
            += del;}}
        return res;
    }
}net;
int n,m,s,t;
vector<tuple<int,int,int> > E;
int main(){
    scanf("%d%d%d%d",&n,&m,&s,&t);
    for (int i=0;i<m;i++){
        int u,v,w;
        scanf("%d%d%d",&u,&v,&w);
        E.push_back(make_tuple(u,v,w));
    }
    net.init(E);
    printf("%lld\n",net.max_flow(s,t));
    return 0;
}
```

4.3 最小费用流

```
#include <bits/stdc++.h>
using namespace std;
const int maxn = 2000+50;
const int maxm = 20000+50;
const int INF = 0x3f3f3f3f;
int m,n,ss,tt,dis[maxn],pre[maxn];
int first[maxn],from[maxm*2],des[maxm*2],nxt[maxm
    *2],cost[maxm*2],flow[maxm*2],tot;
bool in[maxn];
inline void addE(int x,int y,int f,int c){
    tot++;
    from[tot] =x;des[tot] =y;
    flow[tot] =f;cost[tot] =c;
    nxt[tot] = first[x];first[x] = tot;
}
inline void addEdge(int x,int y,int f,int c){
    addE(x,y,f,c);addE(y,x,0,-c);
}
void input(){
    scanf("%d%d",&n,&m);
    tot =-1;
    memset(first,-1,sizeof first);
    for (int i=0;i<m;i++){
        int u,v,c;
        scanf("%d%d%d",&u,&v,&c);
        addEdge(u,v,1,c);addEdge(v,u,1,c);
    }
    addEdge(0,1,2,0);
}
bool spfa(){
    memset(in,0,sizeof in);
    memset(dis,INF,sizeof dis);
    memset(pre,-1,sizeof pre);
    dis[ss] =0;in[ss] =1;
    queue<int> Q;Q.push(ss);
    while (!Q.empty()){
        int q = Q.front();
        Q.pop();in[q] = 0;
        for (int t = first[q];t!=-1;t = nxt[t]){
            int v=des[t],len=cost[t],cx=flow[t];
            if (cx&&dis[v]>dis[q]+len){
                dis[v] = dis[q]+len;
                pre[v] = t;
                if (!in[v]){
                    Q.push(v);in[v] = 1;
                }
            }
        }
    }
    return pre[tt]!=-1;
}
void solve(){
    ss =0;tt=n;
    int totflow =0,totcost =0,nowflow =0,nowcost =0;
    while (spfa()){
        nowcost =0;nowflow = INF;
        int now =pre[tt];
        while (now!=-1){
            nowflow = min(nowflow,flow[now]);
            now = pre[from[now]];
        }
        now = pre[tt];
        while (now!=-1){
            flow[now] -= nowflow;
            flow[now^1] += nowflow;
            nowcost +=cost[now];
            now = pre[from[now]];
        }
        nowcost*=nowflow;
        totflow +=nowflow;
        totcost +=nowcost;
    }
```

```
    cout<<totcost<<endl;
}
int main(){
    input();
    solve();
    return 0;
}
```

4.4 点分治

```
//求树上长度小于等于k的有向路径数
#include<stdio.h>
#include<algorithm>
#include<cstring>
using namespace std;
const int MAX = 1e4+100;
const int INF = 0x3f3f3f3f;
int first [MAX*2]; int des[MAX*2];
int len[MAX*2]; int nxt[MAX*2];
int n,k,tot; int a[MAX]; int sum[MAX];
int dp[MAX]; int dis[MAX]; int num,ans;
bool vis[MAX]; int Sum,Min,Minid;
void init(){
    memset(first,0,sizeof first);
    tot =0; ans =0;
    memset(vis,0,sizeof vis);
}
inline void add(int x,int y,int z){
    tot++;
    des[tot]= y; len[tot] =z;
    nxt[tot] = first[x]; first[x] = tot;
}
void input(){
    for (int i=1;i<n;i++){
        int u,v,w;
        scanf("%d%d%d",&u,&v,&w);
        add(u,v,w); add(v,u,w);
    }
}
void dfs1(int node,int father){
    sum[node] = 1; dp[node] = 0;
    for (int t = first[node];t;t = nxt[t]){
        int v = des[t];
        if (v == father||vis[v]){
            continue;
        }
        dfs1(v,node);
        sum[node] += sum[v];
        dp[node] = max(dp[node],sum[v]);
    }
}
void dfs2(int node,int father){
    int temp = max(dp[node],Sum-sum[node]);
    if (temp<Min){
        Min = temp; Minid = node;
    }
    for (int t = first[node];t;t = nxt[t]){
        int v = des[t];
        if (v==father||vis[v]){ continue; }
        dfs2(v,node);
    }
}
int getRoot(int u){
    dfs1(u,0); Sum = sum[u];
    Min = INF; Minid = -1;
    dfs2(u,0);
    return Minid;
}
void getDist(int node,int father,int dist){
    dis[num++] = dist;
    for (int t = first[node];t;t = nxt[t]){
        int v =des[t];
        if (v == father||vis[v]){ continue; }
        getDist(v,node,dist+len[t]);
    }
}
int calc (int u,int val){
    num=0; int res =0;
    getDist(u,0,0);
    sort(dis,dis+num);
    int i=0;int j=num-1;
    while (i<j){
        if (dis[i]+dis[j]+2*val<=k){
            res+=j-i;
            i++;
        }else{ j--; }
    }
    return res;
}
void solve(int u){
    int root = getRoot(u);
    ans +=calc(root,0); vis[root] = true;
    for (int t = first[root];t;t = nxt[t]){
        int v = des[t];
        if (vis[v]){
```

```
            continue;
        }
        ans-=calc(v,len[t]);
        solve(v);
    }
}
int main(){
    while (scanf("%d%d",&n,&k)!=EOF&&n&&k){
    init();
    input();
    solve(1);
    printf("%d\n",ans);
}
return 0;
}
```

5 计算几何

```

#include <iostream>
#include <cstdio>
#include <cmath>
#include <algorithm>


using namespace std;
const double PI = acos(-1.0);
const double eps = 1e-10;

/***************常用函数***************/
//判断ta与tb的大小关系
int sgn( double ta, double tb)
{
    if(fabs(ta-tb)<eps)return 0;
    if(ta<tb) return -1;
    return 1;
}

//点
class Point
{
    public:

    double x, y;

    Point(){}
    Point( double tx, double ty){ x = tx, y = ty;}

    bool operator < (const Point &_se) const
    {
        return x<_se.x || (x==_se.x && y<_se.y);
    }
    friend Point operator + (const Point &_st,const
        Point &_se)
    {
        return Point(_st.x + _se.x, _st.y + _se.y);
    }
    friend Point operator - (const Point &_st,const
        Point &_se)
    {
        return Point(_st.x - _se.x, _st.y - _se.y);
    }
    double operator ^(const Point &b)const
    {
        return x*b.y - y*b.x;
    }
    //点位置相同(double类型)
    bool operator == (const Point &_off)const
    {
        return sgn(x, _off.x) == 0 && sgn(y, _off.y)
            == 0;
    }

};

/***************常用函数***************/
//点乘
double dot(const Point &po,const Point &ps,const
    Point &pe)
{
    return (ps.x - po.x) * (pe.x - po.x) + (ps.y -
        po.y) * (pe.y - po.y);
}
//叉乘
double xmult(const Point &po,const Point &ps,const
    Point &pe)
{
    return (ps.x - po.x) * (pe.y - po.y) - (pe.x -
        po.x) * (ps.y - po.y);
}
//两点间距离的平方
double getdis2(const Point &st,const Point &se)
{
    return (st.x - se.x) * (st.x - se.x) + (st.y -
```

```
        se.y) * (st.y - se.y);
    }
    //两点间距离
    double getdis(const Point &st,const Point &se)
    {
        return sqrt((st.x - se.x) * (st.x - se.x) + (st.y - se.y) * (st.y - se.y));
    }

    //两点表示的向量
    class Line
    {
        public:

        Point s, e;//两点表示，起点[s]，终点[e]
        double a, b, c;//一般式,ax+by+c=0
        double angle;//向量的角度，[-pi,pi]

        Line(){}
        Line( Point ts, Point te):s(ts),e(te){}//get_angle();}
        Line(double _a,double _b,double _c):a(_a),b(_b),c(_c){}

        //排序用
        bool operator < (const Line &ta)const
        {
            if(angle!=ta.angle) return angle<ta.angle;
            return ((s - ta.s)^(ta.e - ta.s)) < 0;
        }
        //向量与向量的叉乘
        friend double operator / ( const Line &_st, const Line &_se)
        {
            return (_st.e.x - _st.s.x) * (_se.e.y - _se.s.y) - (_st.e.y - _st.s.y) * (_se.e.x - _se.s.x);
        }
        //向量间的点乘
        friend double operator *( const Line &_st, const Line &_se)
        {
            return (_st.e.x - _st.s.x) * (_se.e.x - _se.s.x) - (_st.e.y - _st.s.y) * (_se.e.y - _se.s.y);
        }
        //从两点表示转换为一般表示
        //a=y2-y1,b=x1-x2,c=x2*y1-x1*y2
        bool pton()
        {
            a = e.y - s.y;
            b = s.x - e.x;
            c = e.x * s.y - e.y * s.x;
            return true;
        }
        //半平面交用
        //点在向量左边（右边的小于号改成大于号即可,在对应直线上则加上=号）
        friend bool operator < (const Point &_Off, const Line &_Ori)
        {
            return (_Ori.e.y - _Ori.s.y) * (_Off.x - _Ori.s.x)
            < (_Off.y - _Ori.s.y) * (_Ori.e.x - _Ori.s.x);
        }
        //求直线或向量的角度
        double get_angle( bool isVector = true)
        {
            angle = atan2( e.y - s.y, e.x - s.x);
            if(!isVector && angle < 0)
            angle += PI;
            return angle;
        }

        //点在线段或直线上 1:点在直线上 2点在s,e所在矩形内
        bool has(const Point &_Off, bool isSegment = false) const
        {
            bool ff = sgn( xmult( s, e, _Off), 0) == 0;
            if( !isSegment) return ff;
            return ff
            && sgn(_Off.x - min(s.x, e.x), 0) >= 0 && sgn(_Off.x - max(s.x, e.x), 0) <= 0
            && sgn(_Off.y - min(s.y, e.y), 0) >= 0 && sgn(_Off.y - max(s.y, e.y), 0) <= 0;
        }

        //点到直线/线段的距离
        double dis(const Point &_Off, bool isSegment = false)
        {
            ///化为一般式
            pton();
            //到直线垂足的距离
            double td = (a * _Off.x + b * _Off.y + c) /
```

```
            sqrt(a * a + b * b);
        //如果是线段判断垂足
        if(isSegment)
        {
            double xp = (b * b * _Off.x - a * b *
                _Off.y - a * c) / ( a * a + b * b);
            double yp = (-a * b * _Off.x + a * a *
                _Off.y - b * c) / (a * a + b * b);
            double xb = max(s.x, e.x);
            double yb = max(s.y, e.y);
            double xs = s.x + e.x - xb;
            double ys = s.y + e.y - yb;
            if(xp > xb + eps || xp < xs - eps || yp >
                 yb + eps || yp < ys - eps)
            td = min( getdis(_Off,s), getdis(_Off,e))
                ;
        }
        return fabs(td);
    }

    //关于直线对称的点
    Point mirror(const Point &_Off)
    {
        ///注意先转为一般式
        Point ret;
        double d = a * a + b * b;
        ret.x = (b * b * _Off.x - a * a * _Off.x - 2
             * a * b * _Off.y - 2 * a * c) / d;
        ret.y = (a * a * _Off.y - b * b * _Off.y - 2
             * a * b * _Off.x - 2 * b * c) / d;
        return ret;
    }
    //计算两点的中垂线
    static Line ppline(const Point &_a,const Point &
        _b)
    {
        Line ret;
        ret.s.x = (_a.x + _b.x) / 2;
        ret.s.y = (_a.y + _b.y) / 2;
        //一般式
        ret.a = _b.x - _a.x;
        ret.b = _b.y - _a.y;
        ret.c = (_a.y - _b.y) * ret.s.y + (_a.x - _b.
            x) * ret.s.x;
        //两点式
        if(fabs(ret.a) > eps)
        {
            ret.e.y = 0.0;
            ret.e.x = - ret.c / ret.a;
            if(ret.e == ret. s)
            {
                ret.e.y = 1e10;
                ret.e.x = - (ret.c - ret.b * ret.e.y)
                    / ret.a;
            }
        }
        else
        {
            ret.e.x = 0.0;
            ret.e.y = - ret.c / ret.b;
            if(ret.e == ret. s)
            {
                ret.e.x = 1e10;
                ret.e.y = - (ret.c - ret.a * ret.e.x)
                    / ret.b;
            }
        }
        return ret;
    }

    //------------直线和直线（向量）-------------
    //向量向左边平移t的距离
    Line& moveLine( double t)
    {
        Point of;
        of = Point( -( e.y - s.y), e.x - s.x);
        double dis = sqrt( of.x * of.x + of.y * of.y)
            ;
        of.x= of.x * t / dis, of.y = of.y * t / dis;
        s = s + of, e = e + of;
        return *this;
    }
    //直线重合
    static bool equal(const Line &_st,const Line &
        _se)
    {
        return _st.has( _se.e) && _se.has( _st.s);
    }
    //直线平行
    static bool parallel(const Line &_st,const Line
        &_se)
    {
        return sgn( _st / _se, 0) == 0;
    }
    //两直线（线段）交点
    //返回-1代表平行，0代表重合，1代表相交
```

```
    static bool crossLPt(const Line &_st,const Line &_se, Point &ret)
    {
        if(parallel(_st,_se))
        {
            if(Line::equal(_st,_se)) return 0;
            return -1;
        }
        ret = _st.s;
        double t = ( Line(_st.s,_se.s) / _se) / ( _st / _se);
        ret.x += (_st.e.x - _st.s.x) * t;
        ret.y += (_st.e.y - _st.s.y) * t;
        return 1;
    }
    //------------线段和直线（向量）----------
    //直线和线段相交
    //参数：直线[_st],线段[_se]
    friend bool crossSL( Line &_st, Line &_se)
    {
        return sgn( xmult( _st.s, _se.s, _st.e) * xmult( _st.s, _st.e, _se.e), 0) >= 0;
    }

    //判断线段是否相交(注意添加eps)
    static bool isCrossSS( const Line &_st, const Line &_se)
    {
        //1.快速排斥试验判断以两条线段为对角线的两个矩形是否相交
        //2.跨立试验（等于0时端点重合）
        return
        max(_st.s.x, _st.e.x) >= min(_se.s.x, _se.e.x) &&
        max(_se.s.x, _se.e.x) >= min(_st.s.x, _st.e.x) &&
        max(_st.s.y, _st.e.y) >= min(_se.s.y, _se.e.y) &&
        max(_se.s.y, _se.e.y) >= min(_st.s.y, _st.e.y) &&
        sgn( xmult( _se.s, _st.s, _se.e) * xmult( _se.s, _se.e, _st.s), 0) >= 0 &&
        sgn( xmult( _st.s, _se.s, _st.e) * xmult( _st.s, _st.e, _se.s), 0) >= 0;
    }
};

//寻找凸包的graham 扫描法所需的排序函数
Point gsort;
bool gcmp( const Point &ta, const Point &tb)/// 选取与最后一条确定边夹角最小的点，即余弦值最大者
{
double tmp = xmult( gsort, ta, tb);
if( fabs( tmp) < eps)
return getdis( gsort, ta) < getdis( gsort, tb);
else if( tmp > 0)
return 1;
return 0;
}

class Polygon
{
public:
const static int maxpn = 5e4+7;
Point pt[maxpn];//点（顺时针或逆时针）
Line dq[maxpn]; //求半平面交打开注释
int n;//点的个数


//求多边形面积，多边形内点必须顺时针或逆时针
double area()
{
    double ans = 0.0;
    for(int i = 0; i < n; i ++)
    {
        int nt = (i + 1) % n;
        ans += pt[i].x * pt[nt].y - pt[nt].x * pt[i].y;
    }
    return fabs( ans / 2.0);
}
//求多边形重心，多边形内点必须顺时针或逆时针
Point gravity()
{
    Point ans;
    ans.x = ans.y = 0.0;
    double area = 0.0;
    for(int i = 0; i < n; i ++)
    {
        int nt = (i + 1) % n;
        double tp = pt[i].x * pt[nt].y - pt[nt].x * pt[i].y;
        area += tp;
        ans.x += tp * (pt[i].x + pt[nt].x);
        ans.y += tp * (pt[i].y + pt[nt].y);
    }
```

```
    ans.x /= 3 * area;
    ans.y /= 3 * area;
    return ans;
}
//判断点是否在任意多边形内[射线法]，O(n)
bool ahas( Point &_Off)
{
    int ret = 0;
    double infv = 1e20;//坐标系最大范围
    Line l = Line( _Off, Point( -infv ,_Off.y));
    for(int i = 0; i < n; i ++)
    {
        Line ln = Line( pt[i], pt[(i + 1) % n]);
        if(fabs(ln.s.y - ln.e.y) > eps)
        {
            Point tp = (ln.s.y > ln.e.y)? ln.s: ln.e;
            if( ( fabs( tp.y - _Off.y) < eps && tp.x
                < _Off.x + eps) || Line::isCrossSS(
                ln, l))
            ret++;
        }
        else if( Line::isCrossSS( ln, l))
        ret++;
    }
    return ret&1;
}

//判断任意点是否在凸包内，O(logn)
bool bhas( Point & p)
{
    if( n < 3)
    return false;
    if( xmult( pt[0], p, pt[1]) > eps)
    return false;
    if( xmult( pt[0], p, pt[n-1]) < -eps)
    return false;
    int l = 2,r = n-1;
    int line = -1;
    while( l <= r)
    {
        int mid = ( l + r) >> 1;
        if( xmult( pt[0], p, pt[mid]) >= 0)
        line = mid,r = mid - 1;
        else l = mid + 1;
    }
    return xmult( pt[line-1], p, pt[line]) <= eps;
}
```

```


//凸多边形被直线分割
Polygon split( Line &_Off)
{
    //注意确保多边形能被分割
    Polygon ret;
    Point spt[2];
    double tp = 0.0, np;
    bool flag = true;
    int i, pn = 0, spn = 0;
    for(i = 0; i < n; i ++)
    {
        if(flag)
        pt[pn ++] = pt[i];
        else
        ret.pt[ret.n ++] = pt[i];
        np = xmult( _Off.s, _Off.e, pt[(i + 1) % n]);
        if(tp * np < -eps)
        {
            flag = !flag;
            Line::crossLPt( _Off, Line(pt[i], pt[(i +
                1) % n]), spt[spn++]);
        }
        tp = (fabs(np) > eps)?np: tp;
    }
    ret.pt[ret.n ++] = spt[0];
    ret.pt[ret.n ++] = spt[1];
    n = pn;
    return ret;
}


/** 卷包裹法求点集凸包，_p为输入点集，_n为点的数量 **/
void ConvexClosure( Point _p[], int _n)
{
    sort( _p, _p + _n);
    n = 0;
    for(int i = 0; i < _n; i++)
    {
        while( n > 1 && sgn( xmult( pt[n-2], pt[n-1],
            _p[i]), 0) <= 0)
        n--;
        pt[n++] = _p[i];
    }
    int _key = n;
    for(int i = _n - 2; i >= 0; i--)
    {
```

```
        while( n > _key && sgn( xmult( pt[n-2], pt[n
            -1], _p[i]), 0) <= 0)
        n--;
        pt[n++] = _p[i];
    }
    if(n>1) n--;//除去重复的点，该点已是凸包凸包起点
}
/****** 寻找凸包的graham 扫描法********************/
/****** _p为输入的点集,_n为点的数量****************/

void graham( Point _p[], int _n)
{
    int cur=0;
    for(int i = 1; i < _n; i++)
    if( sgn( _p[cur].y, _p[i].y) > 0 || ( sgn( _p[
        cur].y, _p[i].y) == 0 && sgn( _p[cur].x, _p[
        i].x) > 0) )
    cur = i;
    swap( _p[cur], _p[0]);
    n = 0, gsort = pt[n++] = _p[0];
    if( _n <= 1) return;
    sort( _p + 1, _p+_n ,gcmp);
    pt[n++] = _p[1];
    for(int i = 2; i < _n; i++)
    {
        while(n>1 && sgn( xmult( pt[n-2], pt[n-1], _p
            [i]), 0) <= 0)// 当凸包退化成直线时需特别
            注意n
        n--;
        pt[n++] = _p[i];
    }
}
//凸包旋转卡壳(注意点必须顺时针或逆时针排列)
//返回值凸包直径的平方（最远两点距离的平方）
pair<Point,Point> rotating_calipers()
{
    int i = 1 % n;
    double ret = 0.0;
    pt[n] = pt[0];
    pair<Point,Point>ans=make_pair(pt[0],pt[0]);
    for(int j = 0; j < n; j ++)
    {
        while( fabs( xmult( pt[i+1], pt[j], pt[j +
            1])) > fabs( xmult( pt[i], pt[j], pt[j +
            1])) + eps)
        i = (i + 1) % n;
        //pt[i]和pt[j],pt[i + 1]和pt[j + 1]可能是对踵
            点
        if(ret < getdis2(pt[i],pt[j])) ret = getdis2(
            pt[i],pt[j]), ans = make_pair(pt[i],pt[j
            ]);
        if(ret < getdis2(pt[i+1],pt[j+1])) ret =
            getdis(pt[i+1],pt[j+1]), ans = make_pair(
            pt[i+1],pt[j+1]);
    }
    return ans;
}

//凸包旋转卡壳(注意点必须逆时针排列)
//返回值两凸包的最短距离
double rotating_calipers( Polygon &_Off)
{
    int i = 0;
    double ret = 1e10;//inf
    pt[n] = pt[0];
    _Off.pt[_Off.n] = _Off.pt[0];
    //注意凸包必须逆时针排列且pt[0]是左下角点的位置
    while( _Off.pt[i + 1].y > _Off.pt[i].y)
    i = (i + 1) % _Off.n;
    for(int j = 0; j < n; j ++)
    {
        double tp;
        //逆时针时为 >,顺时针则相反
        while((tp = xmult(_Off.pt[i + 1],pt[j], pt[j
            + 1]) - xmult(_Off.pt[i], pt[j], pt[j +
            1])) > eps)
        i = (i + 1) % _Off.n;
        //(pt[i],pt[i+1])和(_Off.pt[j],_Off.pt[j +
            1])可能是最近线段
        ret = min(ret, Line(pt[j], pt[j + 1]).dis(
            _Off.pt[i], true));
        ret = min(ret, Line(_Off.pt[i], _Off.pt[i +
            1]).dis(pt[j + 1], true));
        if(tp > -eps)//如果不考虑TLE问题最好不要加这个
            判断
        {
            ret = min(ret, Line(pt[j], pt[j + 1]).dis
                (_Off.pt[i + 1], true));
            ret = min(ret, Line(_Off.pt[i], _Off.pt[i
                + 1]).dis(pt[j], true));
        }
    }
    return ret;
}

//-----------半平面交-------------
```

```
//复杂度:O(nlog2(n))
//获取半平面交的多边形（多边形的核）
//参数：向量集合[l]，向量数量[ln];(半平面方向在向量左
    边）
//函数运行后如果n[即返回多边形的点数量]为0则不存在半平
    面交的多边形（不存在区域或区域面积无穷大）
int judege( Line &_lx, Line &_ly, Line &_lz)
{
    Point tmp;
    Line::crossLPt(_lx,_ly,tmp);
    return sgn(xmult(_lz.s,tmp,_lz.e),0);
}
int halfPanelCross(Line L[], int ln)
{
    int i, tn, bot, top;
    for(int i = 0; i < ln; i++)
    L[i].get_angle();
    sort(L, L + ln);
    //平面在向量左边的筛选
    for(i = tn = 1; i < ln; i ++)
    if(fabs(L[i].angle - L[i - 1].angle) > eps)
    L[tn ++] = L[i];
    ln = tn, n = 0, bot = 0, top = 1;
    dq[0] = L[0], dq[1] = L[1];
    for(i = 2; i < ln; i ++)
    {
        while(bot < top && judege(dq[top],dq[top-1],L
            [i]) > 0)
        top --;
        while(bot < top && judege(dq[bot],dq[bot+1],L
            [i]) > 0)
        bot ++;
        dq[++ top] = L[i];
    }
    while(bot < top && judege(dq[top],dq[top-1],dq[
        bot]) > 0)
    top --;
    while(bot < top && judege(dq[bot],dq[bot+1],dq[
        top]) > 0)
    bot ++;
    //若半平面交退化为点或线
    // if(top <= bot + 1)
    // return 0;
    dq[++top] = dq[bot];
    for(i = bot; i < top; i ++)
    Line::crossLPt(dq[i],dq[i + 1],pt[n++]);
    return n;
}
};


class Circle
{
public:
Point c;//圆心
double r;//半径
double db, de;//圆弧度数起点， 圆弧度数终点(逆时针0
    -360)

//-------圆---------

//判断圆在多边形内
bool inside( Polygon &_Off)
{
    if(_Off.ahas(c) == false)
    return false;
    for(int i = 0; i < _Off.n; i ++)
    {
        Line l = Line(_Off.pt[i], _Off.pt[(i + 1) %
            _Off.n]);
        if(l.dis(c, true) < r - eps)
        return false;
    }
    return true;
}

//判断多边形在圆内（线段和折线类似）
bool has( Polygon &_Off)
{
    for(int i = 0; i < _Off.n; i ++)
    if( getdis2(_Off.pt[i],c) > r * r - eps)
    return false;
    return true;
}

//-------圆弧-------
//圆被其他圆截得的圆弧，参数：圆[_Off]
Circle operator-(Circle &_Off) const
{
    //注意圆必须相交，圆心不能重合
    double d2 = getdis2(c,_Off.c);
    double d = getdis(c,_Off.c);
    double ans = acos((d2 + r * r - _Off.r * _Off.r)
        / (2 * d * r));
    Point py = _Off.c - c;
    double oans = atan2(py.y, py.x);
```

```
    Circle res;
    res.c = c;
    res.r = r;
    res.db = oans + ans;
    res.de = oans - ans + 2 * PI;
    return res;
}
//圆被其他圆截得的圆弧，参数：圆[_Off]
Circle operator+(Circle &_Off) const
{
    //注意圆必须相交，圆心不能重合
    double d2 = getdis2(c,_Off.c);
    double d = getdis(c,_Off.c);
    double ans = acos((d2 + r * r - _Off.r * _Off.r)
         / (2 * d * r));
    Point py = _Off.c - c;
    double oans = atan2(py.y, py.x);
    Circle res;
    res.c = c;
    res.r = r;
    res.db = oans - ans;
    res.de = oans + ans;
    return res;
}

//过圆外一点的两条切线
//参数：点[_Off](必须在圆外),返回：两条切线(切线的s点
    为_Off,e点为切点)
pair<Line, Line> tangent( Point &_Off)
{
    double d = getdis(c,_Off);
    //计算角度偏移的方式
    double angp = acos(r / d), ango = atan2(_Off.y -
         c.y, _Off.x - c.x);
    Point pl = Point(c.x + r * cos(ango + angp), c.y
         + r * sin(ango + angp)),
    pr = Point(c.x + r * cos(ango - angp), c.y + r *
         sin(ango - angp));
    return make_pair(Line(_Off, pl), Line(_Off, pr))
        ;
}

//计算直线和圆的两个交点
//参数：直线[_Off](两点式)，返回两个交点，注意直线必须
    和圆有两个交点
pair<Point, Point> cross(Line _Off)
{
    _Off.pton();
    //到直线垂足的距离
    double td = fabs(_Off.a * c.x + _Off.b * c.y +
        _Off.c) / sqrt(_Off.a * _Off.a + _Off.b *
        _Off.b);

    //计算垂足坐标
    double xp = (_Off.b * _Off.b * c.x - _Off.a *
        _Off.b * c.y - _Off.a * _Off.c) / ( _Off.a *
         _Off.a + _Off.b * _Off.b);
    double yp = (- _Off.a * _Off.b * c.x + _Off.a *
        _Off.a * c.y - _Off.b * _Off.c) / (_Off.a *
        _Off.a + _Off.b * _Off.b);

    double ango = atan2(yp - c.y, xp - c.x);
    double angp = acos(td / r);

    return make_pair(Point(c.x + r * cos(ango + angp
        ), c.y + r * sin(ango + angp)),
    Point(c.x + r * cos(ango - angp), c.y + r * sin(
        ango - angp)));
}
};

class triangle
{
public:
Point a, b, c;//顶点
triangle(){}
triangle(Point a, Point b, Point c): a(a), b(b), c(c
    ){}

//计算三角形面积
double area()
{
    return fabs( xmult(a, b, c)) / 2.0;
}

//计算三角形外心
//返回：外接圆圆心
Point circumcenter()
{
    double pa = a.x * a.x + a.y * a.y;
    double pb = b.x * b.x + b.y * b.y;
    double pc = c.x * c.x + c.y * c.y;
    double ta = pa * ( b.y - c.y) - pb * ( a.y - c.y
        ) + pc * ( a.y - b.y);
    double tb = -pa * ( b.x - c.x) + pb * ( a.x - c.
        x) - pc * ( a.x - b.x);
```

```
    double tc = a.x * ( b.y - c.y) - b.x * ( a.y - c
        .y) + c.x * ( a.y - b.y);
    return Point( ta / 2.0 / tc, tb / 2.0 / tc);
}

//计算三角形内心
//返回：内接圆圆心
Point incenter()
{
    Line u, v;
    double m, n;
    u.s = a;
    m = atan2(b.y - a.y, b.x - a.x);
    n = atan2(c.y - a.y, c.x - a.x);
    u.e.x = u.s.x + cos((m + n) / 2);
    u.e.y = u.s.y + sin((m + n) / 2);
    v.s = b;
    m = atan2(a.y - b.y, a.x - b.x);
    n = atan2(c.y - b.y, c.x - b.x);
    v.e.x = v.s.x + cos((m + n) / 2);
    v.e.y = v.s.y + sin((m + n) / 2);
    Point ret;
    Line::crossLPt(u,v,ret);
    return ret;
}

//计算三角形垂心
//返回：高的交点
Point perpencenter()
{
    Line u,v;
    u.s = c;
    u.e.x = u.s.x - a.y + b.y;
    u.e.y = u.s.y + a.x - b.x;
    v.s = b;
    v.e.x = v.s.x - a.y + c.y;
    v.e.y = v.s.y + a.x - c.x;
    Point ret;
    Line::crossLPt(u,v,ret);
    return ret;
}

//计算三角形重心
//返回：重心
//到三角形三顶点距离的平方和最小的点
//三角形内到三边距离之积最大的点
Point barycenter()
{
    Line u,v;
    u.s.x = (a.x + b.x) / 2;
    u.s.y = (a.y + b.y) / 2;
    u.e = c;
    v.s.x = (a.x + c.x) / 2;
    v.s.y = (a.y + c.y) / 2;
    v.e = b;
    Point ret;
    Line::crossLPt(u,v,ret);
    return ret;
}

//计算三角形费马点
//返回：到三角形三顶点距离之和最小的点
Point fermentPoint()
{
    Point u, v;
    double step = fabs(a.x) + fabs(a.y) + fabs(b.x)
        + fabs(b.y) + fabs(c.x) + fabs(c.y);
    int i, j, k;
    u.x = (a.x + b.x + c.x) / 3;
    u.y = (a.y + b.y + c.y) / 3;
    while (step > eps)
    {
        for (k = 0; k < 10; step /= 2, k ++)
        {
            for (i = -1; i <= 1; i ++)
            {
                for (j =- 1; j <= 1; j ++)
                {
                    v.x = u.x + step * i;
                    v.y = u.y + step * j;
                    if (getdis(u,a) + getdis(u,b) +
                        getdis(u,c) > getdis(v,a) +
                        getdis(v,b) + getdis(v,c))
                    u = v;
                }
            }
        }
    }
    return u;
}
};

int main(void)
{

return 0;
```

```
}
```

6 杂项

6.1 define

```
#include <bits/stdc++.h>
using namespace std;
using LL = long long;
#define FOR(i, x, y) for (decay<decltype(y)>::type i = (x), _##i = (y); i < _##i; ++i)
#define FORD(i, x, y) for (decay<decltype(x)>::type i = (x), _##i = (y); i > _##i; --i)
```

6.2 数位 dp

```
LL dfs(LL base, LL pos, LL len, LL s, bool limit) {
    if (pos == -1) return s ? base : 1;
    if (!limit && dp[base][pos][len][s] != -1) return dp[base][pos][len][s];
    LL ret = 0;
    LL ed = limit ? a[pos] : base - 1;
    for(int i= 0;i<ed + 1;i++) {
        tmp[pos] = i;
        if (len == pos)
            ret += dfs(base, pos - 1, len - (i == 0), s, limit && i == a[pos]);
        else if (s &&pos < (len + 1) / 2)
            ret += dfs(base, pos - 1, len, tmp[len - pos] == i, limit && i == a[pos]);
        else
        ret += dfs(base, pos - 1, len, s, limit && i == a[pos]);
    }
    if (!limit) dp[base][pos][len][s] = ret;
    return ret;
}

LL solve(LL x, LL base) {
    LL sz = 0;
    while (x) {
        a[sz++] = x % base;
        x /= base;
    }
    return dfs(base, sz - 1, sz - 1, 1, true);
}
```

6.3 fastio

```
namespace fastIO{
    #define BUF_SIZE 100000
    #define OUT_SIZE 100000
```

```
    //fread->read
    bool IOerror=0;
    //inline char nc(){char ch=getchar();if(ch==-1)IOerror=1;return ch;}
    inline char nc(){
        static char buf[BUF_SIZE],*p1=buf+BUF_SIZE,*pend=buf+BUF_SIZE;
        if(p1==pend){
            p1=buf;pend=buf+fread(buf,1,BUF_SIZE,stdin);
            if(pend==p1){IOerror=1;return -1;}
        }
        return *p1++;
    }
    inline bool blank(char ch){return ch==' '||ch=='\n'||ch=='\r'||ch=='\t';}
    template<class T> inline bool read(T &x){
        bool sign=0;char ch=nc();x=0;
        for(;blank(ch);ch=nc());
        if(IOerror)return false;
        if(ch=='-')sign=1,ch=nc();
        for(;ch>='0'&&ch<='9';ch=nc())x=x*10+ch-'0';
        if(sign)x=-x;
        return true;
    }
    inline bool read(double &x){
        bool sign=0;char ch=nc();x=0;
        for(;blank(ch);ch=nc());
        if(IOerror)return false;
        if(ch=='-')sign=1,ch=nc();
        for(;ch>='0'&&ch<='9';ch=nc())x=x*10+ch-'0';
        if(ch=='.'){
            double tmp=1; ch=nc();
            for(;ch>='0'&&ch<='9';ch=nc())tmp/=10.0,x+=tmp*(ch-'0');
        }
        if(sign)x=-x;
        return true;
    }
    inline bool read(char *s){
        char ch=nc();
        for(;blank(ch);ch=nc());
        if(IOerror)return false;
        for(;!blank(ch)&&!IOerror;ch=nc())*s++=ch;
        *s=0;
        return true;
    }
    inline bool read(char &c){
        for(c=nc();blank(c);c=nc());
        if(IOerror){c=-1;return false;}
        return true;
    }
    template<class T,class... U>bool read(T& h,U&... t){return read(h)&&read(t...);}
```

```
    #undef OUT_SIZE
    #undef BUF_SIZE
};
using namespace fastIO;
```

6.4 date

```
/*
zeller返回星期几%7
*/
int zeller(int y,int m,int d) {
    if (m<=2) y--,m+=12; int c=y/100; y%=100;
    int w=((c>>2)-(c<<1)+y+(y>>2)+(13*(m+1)/5)+d-1)%7;
    if (w<0) w+=7; return(w);
}
/*
用于计算天数
*/
int getId(int y, int m, int d) {
    if (m < 3) {y --; m += 12;}
    return 365 * y + y / 4 - y / 100 + y / 400 + (153 * m + 2) / 5 + d;
}
```

6.5 常用概念

6.6 欧拉路径

欧拉回路：每条边恰走一次的回路
欧拉通路：每条边恰走一次的路径
欧拉图：存在欧拉回路的图
半欧拉图：存在欧拉通路的图
有向欧拉图：每个点入度 = 出度
无向欧拉图：每个点度数为偶数
有向半欧拉图：一个点入度 = 出度 +1，一个点入度 = 出度-1，其他点入度 = 出度
无向半欧拉图：两个点度数为奇数，其他点度数为偶数

6.7 映射

[injective] or [one-to-one] 函数值不重复
[surjective] or [onto] 值域都被取到
[bijective] or [one-to-one correspondence] 一一对应

6.8 反演

反演中心 O, 反演半径 r, 点 p 的反演点 p' 满足 $|OP||OP'| = r^2$
不经过反演中心的直线，反形为经过反演中心的圆
不经过反演中心的圆，反形为圆，反演中心为这两个互为反形的圆的位似中心

6.9 弦图

设 $next(v)$ 表示 $N(v)$ 中最前的点. 令 $w*$ 表示所有满足 $A \in B$ 的 w 中最后的一个点, 判断 $v \cup N(v)$ 是否为极大团, 只需判断是否存在一个 $w \in w*$, 满足 $Next(w) = v$ 且 $|N(v)| + 1 \leq |N(w)|$ 即可.

6.10 五边形数

$\prod_{n=1}^{\infty}(1-x^n) = \sum_{n=0}^{\infty}(-1)^n(1-x^{2n+1})x^{n(3n+1)/2}$

6.11 pick 定理

整多边形面积 $A=$ 内部格点数 $i+$ 边上格点数 $\frac{b}{2}-1$

6.12 重心

半径为 r , 圆心角为 θ 的扇形重心与圆心的距离为 $\frac{4r\sin(\theta/2)}{3\theta}$
半径为 r , 圆心角为 θ 的圆弧重心与圆心的距离为 $\frac{4r\sin^3(\theta/2)}{3(\theta-\sin(\theta))}$

6.13 曼哈顿距离与切比雪夫距离

曼哈顿距离:

$dis = |x1 - x2| + |y1 - y2|$

切比雪夫距离:

$dis = max(|x1 - x2|, |y1 - y2|)$

manhattan to chebyshev

$(x, y) \to (x + y, x - y)$

chebyshev to manhattan

$(x, y) \to (\frac{x+y}{2}, \frac{x-y}{2})$

6.14 第二类 Bernoulli number

$B_m = 1 - \sum_{k=0}^{m-1}\binom{m}{k}\frac{B_k}{m-k+1}$
$S_m(n) = \sum_{k=1}^{n}k^m = \frac{1}{m+1}\sum_{k=0}^{m}\binom{m+1}{k}B_k n^{m+1-k}$

6.15 Fibonacci 数

$F_n = \frac{\varphi^n - (-\varphi)^{-n}}{\sqrt{5}}, \varphi = \frac{1+\sqrt{5}}{2}$
$F_n = \lfloor\frac{\varphi^n}{\sqrt{5}} + \frac{1}{2}\rfloor$

6.16 Catalan 数

$C_{n+1} = \frac{2(2n+1)}{n+2}C_n$
$C_n = \frac{1}{n+1}\binom{2n}{n} = \frac{(2n)!}{(n+1)!n!}$
前 20 项:1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012, 742900, 2674440, 9694845, 35357670, 129644790, 477638700, 1767263190

所有的奇卡塔兰数 C_n 都满足 $n = 2^k - 1$。所有其他的卡塔兰数都是偶数

6.17 Lucas 定理

$C(n, m)modp = C(nmodp, mmodp) * C(n/p, m/p), p$ 是质数

6.18 扩展 Lucas 定理

若 p 不是质数，将 p 分解质因数后分别求解，再用中国剩余定理合并

6.19 BEST theorem

有向图中欧拉回路的数量 $\mathrm{ec}(G) = t_w(G)\prod_{v\in V}\big(\deg(v)-1\big)!$.

其中 $deg(v)$ 表示 v 的入度，$tw(G)$ 表示以 w 为根的外向树的数量，且在连通欧拉图中以任一点为根的外向树数量相同

若需要定起点，则答案乘上 $deg(s)$，表示对每一条欧拉回路，s 出现了 $deg(s)$ 次，选取一个点切开得到一条从 s 出发的欧拉回路

6.20 欧拉示性数定理

对平面图 $V-E+F=2$

6.21 最值反演 (MinMax 容斥)

$max\{S\} = \sum_{T\subseteq S}(-1)^{|T|-1}min\{T\}$

扩展到期望

$E[max\{S\}] = \sum_{T\subseteq S}(-1)^{|T|-1}E[min\{T\}]$

6.22 Polya 定理

设对 n 个对象用 m 种颜色：$b_1, b_2, \cdots, b_m$ 着色。

设 $m^{c(p_i)} = (b_1+b_2+\cdots+b_m)^{c_1(p_i)}(b_1^2+b_2^2+\cdots+b_m^2)^{c_2(p_i)}\cdots(b_1^n+b_2^n+\cdots+b_m^n)^{c_n(p_i)}$，其中 $c_j(p_i)$ 表示置换群中第 i 个置换循环长度为 j 的个数。

设 $S_k = (b_1^k+b_2^k+\cdots+b_m^k), k=1,2\cdots,n$，则波利亚计数定理的母函数形式为：$P(G) = \dfrac{1}{\mid G\mid}\sum_{j=1}^{g}\Pi_{k=1}^{n}S_k^{c_k(p_j)}$

6.23 Stirling 数

第一类:n 个元素的项目分作 k 个环排列的方法数目

$s(n,k) = (-1)^{n+k}|s(n,k)|$

$|s(n,0)| = 0$

$|s(1,1)| = 1$

$|s(n,k)| = |s(n-1,k-1)| + (n-1)*|s(n-1,k)|$

第二类:n 个元素的集定义 k 个等价类的方法数

$S(n,1) = S(n,n) = 1$

$S(n,k) = S(n-1,k-1) + k*S(n-1,k)$

6.24 常用排列组合公式

$\sum_{i=1}^{n} x_i = k, x_i \geq 0$ 的解数为 $C(n+k-1, n-1)$

$x_1 \geq 0, x_i \leq x_{i+1}, x_n \leq k-1$ 的解数等价于在 [0,k-1] 共 k 个数中可重复的取 n 个数的组合数，为 $C(n+k-1,n)$

6.25 三角公式

$\sin(a\pm b) = \sin a\cos b \pm \cos a \sin b$

$\cos(a\pm b) = \cos a\cos b \mp \sin a\sin b$

$\tan(a\pm b) = \frac{\tan(a)\pm\tan(b)}{1\mp\tan(a)\tan(b)}$

$\tan(a)\pm\tan(b) = \frac{\sin(a\pm b)}{\cos(a)\cos(b)}$

$\sin(a)+\sin(b) = 2\sin(\frac{a+b}{2})\cos(\frac{a-b}{2})$

$\sin(a) - \sin(b) = 2\cos(\frac{a+b}{2})\sin(\frac{a-b}{2})$
$\cos(a) + \cos(b) = 2\cos(\frac{a+b}{2})\cos(\frac{a-b}{2})$
$\cos(a) - \cos(b) = -2\sin(\frac{a+b}{2})\sin(\frac{a-b}{2})$
$\sin(na) = n\cos^{n-1} a \sin a - \binom{n}{3}\cos^{n-3} a \sin^3 a + \binom{n}{5}\cos^{n-5} a \sin^5 a - \dots$
$\cos(na) = \cos^n a - \binom{n}{2}\cos^{n-2} a \sin^2 a + \binom{n}{4}\cos^{n-4} a \sin^4 a - \dots$

6.26 积分表

== 含有 $ax + b$ 的积分 ==
$\int (ax+b)^n \mathrm{d}x = \frac{(ax+b)^{n+1}}{a(n+1)} + C$
$\int \frac{1}{ax+b}\mathrm{d}x = \frac{1}{a}\ln|ax+b| + C$
$\int \frac{x}{ax+b}\mathrm{d}x = \frac{1}{a^2}(ax+b-b\ln|ax+b|) + C$
$\int \frac{x^2}{ax+b}\mathrm{d}x = \frac{1}{2a^3}\left[(ax+b)^2 - 4b(ax+b) + 2b^2\ln|ax+b|\right] + C$
$\int \frac{1}{x(ax+b)}\mathrm{d}x = -\frac{1}{b}\ln\left|\frac{ax+b}{x}\right| + C$
$\int \frac{1}{x^2(ax+b)}\mathrm{d}x = \frac{a}{b^2}\ln\left|\frac{ax+b}{x}\right| - \frac{1}{bx} + C$
== 含有 $\sqrt{a+bx}$ 的积分 ==
$\int x\sqrt{a+bx}\mathrm{d}x = \frac{2}{15b^2}(3bx-2a)(a+bx)^{\frac{3}{2}} + C$
$\int x^2\sqrt{a+bx}\mathrm{d}x = \frac{2}{105b^3}(15b^2x^2 - 12abx + 8a^2)(a+bx)^{\frac{3}{2}} + C$
$\int x^n\sqrt{a+bx}\mathrm{d}x = \frac{2}{b(2n+3)}x^n(a+bx)^{\frac{3}{2}} - \frac{2na}{b(2n+3)}\int x^{n-1}\sqrt{a+bx}\mathrm{d}x$
$\int \frac{\sqrt{a+bx}}{x}\mathrm{d}x = 2\sqrt{a+bx} + a\int \frac{1}{x\sqrt{a+bx}}\mathrm{d}x$
$\int \frac{\sqrt{a+bx}}{x^n}\mathrm{d}x = \frac{-1}{a(n-1)}\frac{(a+bx)^{\frac{3}{2}}}{x^{n-1}} - \frac{(2n-5)b}{2a(n-1)}\int \frac{\sqrt{a+bx}}{x^{n-1}}\mathrm{d}x, n \neq 1$
$\int \frac{1}{x\sqrt{a+bx}}\mathrm{d}x = \frac{1}{\sqrt{a}}\ln\left(\frac{\sqrt{a+bx}-\sqrt{a}}{\sqrt{a+bx}+\sqrt{a}}\right) + C, a > 0 = \frac{2}{\sqrt{-a}}\arctan\sqrt{\frac{a+bx}{-a}} + C, a < 0$
$\int \frac{1}{x^n\sqrt{a+bx}}\mathrm{d}x = \frac{-1}{a(n-1)}\frac{\sqrt{a+bx}}{x^{n-1}} - \frac{(2n-3)b}{2a(n-1)}\int \frac{1}{x^{n-1}}\sqrt{a+bx}\mathrm{d}x, n \neq 1$
== 含有 $x^2 \pm \alpha^2$ 的积分 ==
$\int \frac{1}{x^2+\alpha^2}\mathrm{d}x = \frac{\arctan\dfrac{x}{\alpha}}{\alpha} + C$
$\int \frac{1}{\pm x^2 \mp \alpha^2}\mathrm{d}x = \frac{\ln\left(\dfrac{x\mp\alpha}{\pm x+\alpha}\right)}{2\alpha} + C$
== 含有 $ax^2 + b$ 的积分 ==
$\int \frac{1}{ax^2+b}\mathrm{d}x = \frac{1}{\sqrt{ab}}\arctan\frac{\sqrt{a}x}{\sqrt{b}} + C$
== 含有 $ax^2 + bx + c \quad (a > 0)$ 的积分 ==
$\int ax^2 + bx + c\mathrm{d}x = \frac{ax^3}{3} + \frac{bx^2}{2} + cx + C$
== 含有 $\sqrt{a^2 + x^2} \quad (a > 0)$ 的积分 ==
$\int \sqrt{a^2+x^2}\mathrm{d}x = \frac{1}{2}x\sqrt{a^2+x^2} + \frac{1}{2}a^2\ln\left(x+\sqrt{a^2+x^2}\right) + C$
$\int x^2\sqrt{a^2+x^2}\mathrm{d}x = \frac{1}{8}x(a^2+2x^2)\sqrt{a^2+x^2} - \frac{1}{8}a^4\ln\left(x+\sqrt{a^2+x^2}\right) + C$
$\int \frac{\sqrt{a^2+x^2}}{x}\mathrm{d}x = \sqrt{a^2+x^2} - a\ln\left(\frac{a+\sqrt{a^2+x^2}}{x}\right) + C$
$\int \frac{\sqrt{a^2+x^2}}{x^2}\mathrm{d}x = \ln\left(x+\sqrt{a^2+x^2}\right) - \frac{\sqrt{a^2+x^2}}{x} + C$
$\int \frac{1}{\sqrt{a^2+x^2}}\mathrm{d}x = \ln\left(x+\sqrt{a^2+x^2}\right) + C$
$\int \frac{x^2}{\sqrt{a^2+x^2}}\mathrm{d}x = \frac{1}{2}x\sqrt{a^2+x^2} - \frac{1}{2}a^2\ln\left(\sqrt{a^2+x^2}+x\right) + C$
$\int \frac{1}{x\sqrt{a^2+x^2}}\mathrm{d}x = \frac{1}{a}\ln\left(\frac{x}{a+\sqrt{a^2+x^2}}\right) + C$
$\int \frac{1}{x^2\sqrt{a^2+x^2}}\mathrm{d}x = -\frac{\sqrt{a^2+x^2}}{a^2x} + C$
== 含有 $\sqrt{x^2 - a^2} \quad (x^2 > a^2)$ 的积分 =
$\int \frac{1}{\sqrt{x^2-a^2}}\mathrm{d}x = \ln|x+\sqrt{x^2-a^2}| + C$
== 含有 $\sqrt{a^2 - x^2} \quad (a^2 > x^2)$ 的积分 ==
$\int \frac{1}{\sqrt{a^2-x^2}}\mathrm{d}x = \arcsin\frac{x}{a} + C = -\arccos\frac{x}{a} + C$
$\int \sqrt{a^2-x^2}\mathrm{d}x = \frac{1}{2}x\sqrt{a^2-x^2} + \frac{a^2}{2}\arcsin\frac{x}{a} + C$

$\int x^2\sqrt{a^2-x^2}\mathrm{d}x = \frac{1}{8}x(2x^2-a^2)\sqrt{a^2-x^2} + \frac{1}{8}a^4 \arcsin \frac{x}{a} + C$
$\int \frac{\sqrt{a^2-x^2}}{x}\mathrm{d}x = \sqrt{a^2-x^2} - a\ln\left(\frac{a+\sqrt{a^2-x^2}}{x}\right) + C$
$\int \frac{\sqrt{a^2-x^2}}{x^2}\mathrm{d}x = -\frac{\sqrt{a^2-x^2}}{x} - \arcsin\frac{x}{a} + C$
$\int \frac{1}{x\sqrt{a^2-x^2}}\mathrm{d}x = -\frac{1}{a}\ln\left(\frac{a+\sqrt{a^2-x^2}}{x}\right) + C$
$\int \frac{x^2}{\sqrt{a^2-x^2}}\mathrm{d}x = -\frac{1}{2}x\sqrt{a^2-x^2} + \frac{1}{2}a^2\arcsin\frac{x}{a} + C$
$\int \frac{1}{x^2\sqrt{a^2-x^2}}\mathrm{d}x = -\frac{\sqrt{a^2-x^2}}{a^2x} + C$
== 含有 $R = \sqrt{|a|x^2+bx+c} \qquad (a \neq 0)$ 的积分 ==
$\int \frac{\mathrm{d}x}{R} = \frac{1}{\sqrt{a}}\ln\left(2\sqrt{a}R + 2ax + b\right) \qquad (\text{for } a > 0)$
$\int \frac{\mathrm{d}x}{R} = \frac{1}{\sqrt{a}}\ \mathrm{arsinh}\,\frac{2ax+b}{\sqrt{4ac-b^2}} \qquad (\text{for } a > 0,\, 4ac - b^2 > 0)$
$\int \frac{\mathrm{d}x}{R} = \frac{1}{\sqrt{a}}\ln|2ax+b| \quad (\text{for } a > 0,\, 4ac - b^2 = 0)$
$\int \frac{\mathrm{d}x}{R} = -\frac{1}{\sqrt{-a}}\arcsin\frac{2ax+b}{\sqrt{b^2-4ac}} \qquad (\text{for } a < 0,\, 4ac - b^2 < 0,\ (2ax+b) < \sqrt{b^2-4ac})$
$\int \frac{\mathrm{d}x}{R^3} = \frac{4ax+2b}{(4ac-b^2)R}$
$\int \frac{\mathrm{d}x}{R^5} = \frac{4ax+2b}{3(4ac-b^2)R}\left(\frac{1}{R^2} + \frac{8a}{4ac-b^2}\right)$
$\int \frac{\mathrm{d}x}{R^{2n+1}} = \frac{2}{(2n-1)(4ac-b^2)}\left[\frac{2ax+b}{R^{2n-1}} + 4a(n-1)\int\frac{\mathrm{d}x}{R^{2n-1}}\right]$
$\int \frac{x}{R}\ \mathrm{d}x = \frac{R}{a} - \frac{b}{2a}\int\frac{\mathrm{d}x}{R}$
$\int \frac{x}{R^3}\ \mathrm{d}x = -\frac{2bx+4c}{(4ac-b^2)R}$
$\int \frac{x}{R^{2n+1}}\ \mathrm{d}x = -\frac{1}{(2n-1)aR^{2n-1}} - \frac{b}{2a}\int\frac{\mathrm{d}x}{R^{2n+1}}$
$\int \frac{\mathrm{d}x}{xR} = -\frac{1}{\sqrt{c}}\ln\left(\frac{2\sqrt{c}R+bx+2c}{x}\right)$
$\int \frac{\mathrm{d}x}{xR} = -\frac{1}{\sqrt{c}}\,\mathrm{arsinh}\left(\frac{bx+2c}{|x|\sqrt{4ac-b^2}}\right)$
== 含有三角函数的积分 ==
$\int \cos x\mathrm{d}x = \sin x + C$
$\int \sin x\mathrm{d}x = -\cos x + C$
$\int \sec^2 x\mathrm{d}x = \tan x + C$
$\int \csc^2 x\mathrm{d}x = -\cot x + C$
$\int \sec x\tan x\mathrm{d}x = \sec x + C$
$\int \csc x\cot x\mathrm{d}x = -\csc x + C$
$\int \tan x\mathrm{d}x = -\ln|\cos x| + C = \ln|\sec x| + C$
$\int \cot x\mathrm{d}x = \ln|\sin x| + C$
$\int \sec x\mathrm{d}x = \ln|\sec x + \tan x| + C$
$\int \csc x\mathrm{d}x = -\ln|\csc x + \cot x| + C = \ln\left|\frac{\tan x - \sin x}{\sin x\tan x}\right| + C$
$\int \sin^n x\mathrm{d}x = -\frac{1}{n}\sin^{n-1}x\cos x + \frac{n-1}{n}\int\sin^{n-2}x\mathrm{d}x + C \quad \forall n \geq 2$
$\int \sin^2 x\mathrm{d}x = \frac{x}{2} - \frac{\sin 2x}{4} + C$
$\int \cos^n x\mathrm{d}x = \frac{1}{n}\cos^{n-1}x\sin x + \frac{n-1}{n}\int\cos^{n-2}x\mathrm{d}x + C \quad \forall n \geq 2$
$\int \cos^2 x\mathrm{d}x = \frac{x}{2} + \frac{\sin 2x}{4} + C$
$\int \tan^n x\mathrm{d}x = \frac{1}{n-1}\tan^{n-1}x - \int\tan^{n-2}x\mathrm{d}x + C \quad \forall n \geq 2$
$\int \tan^2 x\mathrm{d}x = \tan x - x + C$
$\int \cot^n x\mathrm{d}x = \frac{1}{n-1}\cot^{n-1}x - \int\cot^{n-2}x\mathrm{d}x + C \quad \forall n \geq 2$
$\int \cot^2 x\mathrm{d}x = -\cot x - x + C$
$\int \sec^n x\mathrm{d}x = \frac{1}{n-1}\sec^{n-2}x\tan x + \frac{n-2}{n-1}\int\sec^{n-2}x\mathrm{d}x + C \quad \forall n \geq 2$
$\int \csc^n x\mathrm{d}x = -\frac{1}{n-1}\csc^{n-2}x\cot x + \frac{n-2}{n-1}\int\csc^{n-2}x\mathrm{d}x + C \quad \forall n \geq 2$
== 含有反三角函数的积分 ==
$\int \arcsin x\mathrm{d}x = x\arcsin x + \sqrt{1-x^2} + C$
$\int \arccos x\mathrm{d}x = x\arccos x - \sqrt{1-x^2} + C$
$\int \arctan x\mathrm{d}x = x\arctan x - \ln\sqrt{1+x^2} + C$

$\int arccot(x)\mathrm{d}x = x \times arccot(x) + \ln\sqrt{1+x^2} + C$

$\int arcsec(x)\mathrm{d}x = x \times arcsec(x) - sgn(x)\ln\left|x+\sqrt{x^2-1}\right| + C = x \times arcsec(x) + sgn(x)\ln\left|x-\sqrt{x^2-1}\right| + C$

$\int arccsc(x)\mathrm{d}x = x \times arccsc(x) + sgn(x)\ln\left|x+\sqrt{x^2-1}\right| + C = x \times arccsc(x) - sgn(x)\ln\left|x-\sqrt{x^2-1}\right| + C$

== 含有指数函数的积分 ==

$\int e^x \mathrm{d}x = e^x + C$

$\int \alpha^x \mathrm{d}x = \frac{\alpha^x}{\ln\alpha} + C$

$\int xe^{ax}\mathrm{d}x = \frac{1}{a^2}(ax-1)e^{ax} + C$

$\int x^n e^{ax}\mathrm{d}x = \frac{1}{a}x^n e^{ax} - \frac{n}{a}\int x^{n-1}e^{ax}\mathrm{d}x$

$\int e^{ax}\sin bx\mathrm{d}x = \frac{e^{ax}}{a^2+b^2}(a\sin bx - b\cos bx) + C$

$\int e^{ax}\cos bx\mathrm{d}x = \frac{e^{ax}}{a^2+b^2}(a\cos bx + b\sin bx) + C$

== 含有对数函数的积分 ==

$\int \ln x\mathrm{d}x = x\ln x - x + C$

$\int \log_\alpha x\mathrm{d}x = \frac{1}{\ln\alpha}\left(x\ln x - x\right) + C$

$\int x^n \ln x\mathrm{d}x = \frac{x^{n+1}}{(n+1)^2}[(n+1)\ln x - 1] + C$

$\int \frac{1}{x\ln x}\mathrm{d}x = \ln\left(\ln x\right) + C$

== 含有双曲函数的积分 ==

$\int \sinh x\mathrm{d}x = \cosh x + C$

$\int \cosh x\mathrm{d}x = \sinh x + C$

$\int \tanh x\mathrm{d}x = \ln\left(\cosh x\right) + C$

$\int \coth x\mathrm{d}x = \ln\left(\sinh x\right) + C$

$\int \mathrm{sech}\ x\mathrm{d}x = \arcsin\left(\tanh x\right) + C = \arctan\left(\sinh x\right) + C$

$\int \mathrm{csch}\ x\mathrm{d}x = \ln\left(\tanh\frac{x}{2}\right) + C$

== 定积分 ==

$\int_{-\infty}^{\infty} e^{-\alpha x^2}\mathrm{d}x = \sqrt{\frac{\pi}{\alpha}}$

$\int_0^{\frac{\pi}{2}} \sin^n x\mathrm{d}x = \int_0^{\frac{\pi}{2}} \cos^n x\mathrm{d}x =$

$$\begin{cases} \frac{n-1}{n}\cdot\frac{n-3}{n-2}\cdot\ldots\cdot\frac{4}{5}\cdot\frac{2}{3}, & \text{if } n > 1 \text{ 且} n \text{ 为奇数} \\ \frac{n-1}{n}\cdot\frac{n-3}{n-2}\cdot\ldots\cdot\frac{3}{4}\cdot\frac{1}{2}\cdot\frac{\pi}{2}, & \text{if } n > 0 \text{ 且} n \text{ 为偶数} \end{cases}$$