# Author：Tankche1

# 图论相关

## 2-sat

# include<stdio.h>

# include<string.h>

# include<algorithm>

using namespace std;

const int maxn=205<<1;

const int maxm=10005;

int n,m;

int a[maxm],b[maxm],c[maxm];

int first[maxn],nnext[maxm<<2],v[maxm<<2],bh[maxm<<2],edge;

void add\_edge(int a,int b,int c){

nnext[edge]=first[a];v[edge]=b;bh[edge]=c;

first[a]=edge++;

}

struct Twosat{

int dfn[maxn],low[maxn],st[maxn],tp;

bool vis[maxn];

int belong[maxn];

int need,idx,ans;

void tarjan(int u)

{

dfn[u] = low[u] = ++idx;

vis[u] = 1;

st[++tp] = u;

int vv ;

for(int i = first[u]; i != -1; i = nnext[i])

{

if(bh[i]>=need) continue;

vv=v[i];

if(!dfn[vv])

{

tarjan(vv) ;

low[u] = min(low[u],low[vv]);

}

else if(vis[vv])

low[u] = min(low[u],dfn[vv]);

}

if(dfn[u] == low[u])

{

ans++;

while(1)

{

vv = st[tp--];

vis[vv] = 0;

belong[vv] = ans;

if(vv == u)

break;

}

}

}

bool solve(int ned)

{

need=ned;

memset(vis,0,sizeof(vis));

memset(dfn,0,sizeof(dfn));

idx=tp = ans = 0;

for(int i = 0; i < 2\*n; i++)

if(!dfn[i])

tarjan(i) ;

for(int i = 0; i < n; i++)

if(belong[2\*i]==belong[(2\*i)^1])//矛盾

return false ;

return true;

}

}twosat;

int main(){

int tcase;

scanf("%d",&tcase);

while(tcase--){

scanf("%d%d",&n,&m);

memset(first,-1,sizeof(first));edge=0;

for(int i=0;i<m;i++){

scanf("%d%d%d",&a[i],&b[i],&c[i]);

if(c[i]==0){

add\_edge(2\*a[i],2\*b[i]+1,i);add\_edge(2\*b[i],2\*a[i]+1,i);

}

if(c[i]==1){

add\_edge(2\*a[i],2\*b[i],i);add\_edge(2\*a[i]+1,2\*b[i]+1,i);

add\_edge(2\*b[i],2\*a[i],i);add\_edge(2\*b[i]+1,2\*a[i]+1,i);

}

if(c[i]==2){

add\_edge(2\*a[i]+1,2\*b[i],i);add\_edge(2\*b[i]+1,2\*a[i],i);

}

}

int L=1,R=m,M;

while(L<=R){

M=(L+R)>>1;

if(twosat.solve(M)) L=M+1;

else R=M-1;

}

printf("%d\n",L-1);

}

return 0;

}

## DINIC

# include<stdio.h>

# include<string.h>

# include<queue>

# include<algorithm>

using namespace std;

const int maxn=210;

const int maxm=40010;

const int INF=~0U>>1;

struct Edge{

int from,to,flow,cap;

Edge(){}

Edge(int from,int to,int flow,int cap):from(from),to(to),flow(flow),cap(cap) {}

};

struct Dinic{

int S,T,edge,n;

Edge edges[maxm];

int first[maxn],next[maxm];

int cur[maxn],d[maxn];

void init(int nn){

n=nn+1;

memset(first,-1,sizeof(first));

edge=0;

}

void add\_edge(int from,int to,int cap){

next[edge]=first[from];

edges[edge]=Edge(from,to,0,cap);

first[from]=edge++;

next[edge]=first[to];

edges[edge]=Edge(to,from,0,0);

first[to]=edge++;

}

bool BFS(){

for(int i=0;i<n;i++) d[i]=INF;

d[S]=0;

queue<int> q;q.push(S);

while(!q.empty()){

int x=q.front();q.pop();

for(int e=first[x];e!=-1;e=next[e]){

if(edges[e].cap>edges[e].flow&&d[edges[e].to]>d[x]+1){

d[edges[e].to]=d[x]+1;

q.push(edges[e].to);

}

}

}

return d[T]!=INF;

}

int DFS(int x,int a){

if(x==T||a==0) return a;

int flow=0,f;

for(int &e=cur[x];e!=-1;e=next[e]){

if(d[edges[e].to]==d[x]+1&&(f=DFS(edges[e].to,min(edges[e].cap-edges[e].flow,a)))>0){

flow+=f;a-=f;

edges[e].flow+=f;edges[e^1].flow-=f;

if(a==0) break;

}

}

return flow;

}

int maxflow(int ss,int tt){

S=ss;T=tt;

int flow=0;

while(BFS()){

for(int i=0;i<n;i++) cur[i]=first[i];

flow+=DFS(S,INF);

}

return flow;

}

}dinic;

int n,m;

int main()

{

int S,T;

int x,y,c;

int tcase;

int t=0;

scanf("%d",&tcase);

while(tcase--){

printf("Case %d: ",++t);

scanf("%d%d",&n,&m);

S=1,T=n;

dinic.init(T+1);

for(int i=1;i<=m;i++){

scanf("%d%d%d",&x,&y,&c);

dinic.add\_edge(x,y,c);

}

printf("%d\n",dinic.maxflow(S,T));

}

return 0;

}

## 最小费用最大流

/\*

最小费用最大流

拆点 m工人变成m\*n个点 表示他修的倒数第i辆车 连车 费用为i\*cost

\*/

# include<stdio.h>

# include<string.h>

# include<queue>

# include<algorithm>

using namespace std;

const int maxn=10010;

const int maxm=30\*30\*2;

const int maxnode=80;

const int INF=~0U>>1;

struct Edge{

int from,to,flow,cap,cost;

Edge() {}

Edge(int from,int to,int flow,int cap,int cost):from(from),to(to),flow(flow),cap(cap),cost(cost) {}

};

Edge node;

struct Dinic{

int first[maxnode],next[maxm];

Edge edges[maxm];

int d[maxnode],a[maxnode],p[maxnode];

bool inq[maxnode];

int n,m;

void init(i nt a){

n=a;m=0;

memset(first,-1,sizeof(first));

}

void add\_edge(int from,int to,int cap,int cost){

next[m]=first[from];

first[from]=m;

edges[m++]=Edge(from,to,0,cap,cost);

next[m]=first[to];

first[to]=m;

edges[m++]=Edge(to,from,0,0,-cost);

}

bool BellmanFord(int s,int t,int &flow,int &cost){

queue<int> q;

q.push(s);

a[s]=INF;

p[s]=0;

memset(inq,false,sizeof(inq));

for(int i=0;i<n;i++) d[i]=INF;

d[s]=0;

while(!q.empty()){

int x=q.front();q.pop();inq[x]=false;

for(int e=first[x];e!=-1;e=next[e]){

int v=edges[e].to;

if(edges[e].cap>edges[e].flow&&d[v]>d[x]+edges[e].cost){

d[v]=d[x]+edges[e].cost;

a[v]=min(a[x],edges[e].cap-edges[e].flow);

p[v]=e;

if(!inq[v]) inq[v]=true,q.push(v);

}

}

}

if(d[t]==INF) return false;

flow+=a[t];

cost+=a[t]\*d[t];

int u=t;

while(u!=s){

edges[p[u]].flow+=a[t];

edges[p[u]^1].flow-=a[t];

u=edges[p[u]].from;

}

return true;

}

int Mincost(int s,int t){

int flow=0,cost=0;

while(BellmanFord(s,t,flow,cost));

return cost;

}

};

Dinic g;

# 数论

## Burnside:对于一个置换f，若着色方案s经过置换后不变，称s为f的不动点。不动点数目C(f),等价类数目为C(f)的平均值。

/\*

sgu 111

高精度开方

\*/

# include<stdio.h>

# include<string.h>

# include<algorithm>

using namespace std;

typedef long long LL;

const int bit=10000000;

const int bit\_num=7;

struct Big{

LL a[150];

int len;

void init() {len=1;memset(a,0,sizeof(a));}

void get(char \*s){

len=0;

int n=strlen(s),tmp;

for(int i=n-1;i>=0;i-=bit\_num){

tmp=0;

for(int j=max(0,i-bit\_num+1);j<=i;j++)

tmp=tmp\*10+s[j]-'0';

a[len++]=tmp;

}

}

void print(){

printf("%lld",a[len-1]);

for(int i=len-2;i>=0;i--)

printf("%07lld",a[i]);puts("");

}

};

Big C;

Big operator + (Big A, Big B){

int n=max(A.len,B.len);

C.init();

for(int i=0;i<n;i++)

C.a[i]=A.a[i]+B.a[i];

for(int i=0;i<n;i++)

if(C.a[i]>=bit) C.a[i]-=bit,C.a[i+1]++;

C.len=n;

if(C.a[C.len]) C.len++;

return C;

}

Big operator / (Big A,int b){

int n=A.len;LL tmp=0;

for(int i=n-1;i>=0;i--){

tmp=tmp\*bit+A.a[i];

C.a[i]=tmp/b;tmp=tmp-C.a[i]\*b;

}

for(C.len=n;C.len>1&&C.a[C.len-1]==0;C.len--);

return C;

}

Big operator \* (Big A,Big B){

C.init();

for(int i=0;i<A.len;i++)

for(int j=0;j<B.len;j++){

C.a[i+j]+=A.a[i]\*B.a[j];

}

for(int i=0;i<A.len+B.len-1;i++)

C.a[i+1]+=C.a[i]/bit,C.a[i]%=bit;

C.len=A.len+B.len-1;

while(C.a[C.len]>0){

C.a[C.len+1]+=C.a[C.len]/bit;C.a[C.len]%=bit;C.len++;

}

return C;

}

Big operator - (Big A,int b){

C.init();

C.len=A.len;for(int i=0;i<A.len;i++) C.a[i]=A.a[i];

int t;

C.a[0]-=b;

for(int i=0;C.a[i]<0;i++){

t=(-C.a[i]+bit-1)/bit;

C.a[i]+=t\*bit,C.a[i+1]-=t;

}

while(C.len>1&&C.a[C.len-1]==0) C.len--;

return C;

}

bool operator <= (Big A,Big B){

if(B.len>A.len) return true;

if(A.len>B.len) return false;

for(int i=A.len-1;i>=0;i--){

if(A.a[i]>B.a[i]) return false;

else if(A.a[i]<B.a[i]) return true;

}

return true;

}

Big L,R,M,num,T,one;

char s[2010];

int main(){

L.init();

scanf("%s",s);

num.get(s);//R.print();

L.a[0]=1;R=num;one.init();one.a[0]=1;

while(L<=R){

//printf("L:");L.print();printf("R:");R.print();

M=L+R;//printf("M:");M.print();

M=M/2;//printf("M:");M.print();

T=M\*M;//printf("T:");T.print();//printf("TR:");R.print();

if(T<=num) L=M+one;//printf("small L:");L.print();M.print();}

else //printf("RR:"),R.print(),

R=M-1;//,printf("big R:"),R.print(),M.print();

//int aa;scanf("%d",&aa);

}

L=L-1;

L.print();

return 0;

}

# 矩阵乘法

typedef long long LL;

const int maxl=2;

typedef LL Matrix[3][3];

typedef LL Vector[3];

void copy(Matrix A,Matrix B){

for(int i=1;i<=maxl;i++)

for(int j=1;j<=maxl;j++)

A[i][j]=B[i][j];

}

void Matrix\_mul(Matrix A,Matrix B,Matrix ans){

Matrix C;

for(int i=1;i<=maxl;i++)

for(int j=1;j<=maxl;j++){

C[i][j]=0;

for(int k=1;k<=maxl;k++)

C[i][j]+=A[i][k]\*B[k][j];

C[i][j]%=mod;

}

copy(ans,C);

}

void Matrix\_pow(Matrix A,int p,Matrix ans){

if(p==0) {copy(ans,mem);return;}

Matrix half;

Matrix\_pow(A,p>>1,half);

Matrix\_mul(half,half,half);

if(p&1) Matrix\_mul(half,A,half);

copy(ans,half);

return;

}

void Vector\_mul(Vector A,Matrix B,Vector ans){

Vector C;

for(int i=1;i<=maxl;i++){

C[i]=0;

for(int j=1;j<=maxl;j++)

C[i]+=A[j]\*B[j][i];

C[i]%=mod;

}

for(int i=1;i<=maxl;i++)

ans[i]=C[i];

}

## FFT

# include<stdio.h>

# include<math.h>

# include<string.h>

# include<algorithm>

using namespace std;

const double PI = acos(-1.0);

const int maxn = 100010 \*4;

const int mod=313;

struct complex

{

double r,i;

complex(double \_r = 0.0,double \_i = 0.0)

{

r = \_r; i = \_i;

}

complex operator +(const complex &b)

{

return complex(r+b.r,i+b.i);

}

complex operator -(const complex &b)

{

return complex(r-b.r,i-b.i);

}

complex operator \*(const complex &b)

{

return complex(r\*b.r-i\*b.i,r\*b.i+i\*b.r);

}

};

/\*

\* 进行FFT和IFFT前的反转变换。

\* 位置i和 （i二进制反转后位置）互换

\* len必须去2的幂

\*/

void change(complex y[],int len)

{

int i,j,k;

for(i = 1, j = len/2;i < len-1; i++)

{

if(i < j)swap(y[i],y[j]);

//交换互为小标反转的元素，i<j保证交换一次

//i做正常的+1，j左反转类型的+1,始终保持i和j是反转的

k = len/2;

while( j >= k)

{

j -= k;

k /= 2;

}

if(j < k) j += k;

}

}

/\*

\* 做FFT

\* len必须为2^k形式，

\* on==1时是DFT，on==-1时是IDFT

\*/

void fft(complex y[],int len,int on)

{

change(y,len);

for(int h = 2; h <= len; h <<= 1)

{

complex wn(cos(-on\*2\*PI/h),sin(-on\*2\*PI/h));

for(int j = 0;j < len;j+=h)

{

complex w(1,0);

for(int k = j;k < j+h/2;k++)

{

complex u = y[k];

complex t = w\*y[k+h/2];

y[k] = u+t;

y[k+h/2] = u-t;

w = w\*wn;

}

}

}

if(on == -1)

for(int i = 0;i < len;i++)

y[i].r /= len;

}

complex x1[maxn], x2[maxn];

void getfft(int arg1[], int len1, int arg2[], int len2, int fft\_out[])//模板：第一个数组，数组元素数量。 第二个数组，数组元素数量，输出数组

{

int len = 1;

while(len < len1\*2 || len < len2\*2)len<<=1;

for(int i = 0;i < len1;i++)

x1[i] = complex(arg1[i] , 0);

for(int i = len1;i < len;i++)

x1[i] = complex(0 , 0);

for(int i = 0;i < len2;i++)

x2[i] = complex(arg2[i] , 0);

for(int i = len2;i < len;i++)

x2[i] = complex(0, 0);

fft(x1,len,1);

fft(x2,len,1);

for(int i = 0;i < len;i++)

x1[i] = x1[i]\*x2[i];

fft(x1,len,-1);

for(int i = 0;i < len;i++)

fft\_out[i] = ((int)(x1[i].r+0.5));

}

int n;

int a[maxn];

int f[maxn];

int fft\_a[maxn],fft\_b[maxn],fft\_c[maxn];

void solve(int l,int r){

if(l==r){

f[l]=(f[l]+a[l])%mod;

return;

}

int mid=(l+r)>>1;

solve(l,mid);

int l1=mid-l+1,l2=r-l;

//for(int i=0;i<mid-l+1;i++) fft\_a[i]=f[i+mid+1];

//for(int i=0;i<l2;i++) fft\_b[i]=a[i+1];

getfft(&f[l],l1,&a[1],l2,fft\_c);

for(int i=mid+1;i<=r;i++){

f[i]=(f[i]+fft\_c[mid-l+i-(mid+1)])%mod;

}

solve(mid+1,r);

}

int main(){

//printf("%.1lf\n",1.56);

while(scanf("%d",&n)!=EOF){

if(n==0) break;

for(int i=1;i<=n;i++)scanf("%d",&a[i]),a[i]%=mod;

for(int i=1;i<=n;i++) f[i]=0;

solve(1,n);

printf("%d\n",f[n]);

}

return 0;

}

## NTT

#include <iostream>

#include <stdio.h>

#include <math.h>

#include <string.h>

#include <time.h>

#include <stdlib.h>

# include<algorithm>

using namespace std;

typedef long long LL;

const int maxn = 100086 \* 4, Mod = 998244353, G = 3;

int exp\_mod(int now, int p)

{

if (p == 0) return 1;

int tmp = exp\_mod(now, p / 2);

tmp = ((LL)tmp \* tmp) % Mod;

if (p & 1) tmp = ((LL)tmp \* now) % Mod;

return tmp;

}

struct T

{

int g[maxn], \_g[maxn], z1, z2, tmp, temp;

inline int reverse(int num, int n)

{

int u = 0;

for (int i = 0; i < n; i++)

u += ((num >> (n - i - 1)) & 1) << i;

return u;

}

void FFT(int z[], int w[], int n, int style)

{

int len=1 << n, d, step;

for (int i = 0; i < len; i++) z[i] = w[reverse(i,n)];

for (int i = 1; i <= n; i++)

{

d = 1 << i;

if (style == 1) step = g[i];

else step = \_g[i];

for (int j = 0; j < len; j += d)

{

temp = 1;

for (int k = j; k < j + d / 2; k++)

{

z1 = z[k];

z2 = ((LL)z[k + d / 2] \* temp) % Mod;

z[k] = (z1 + z2) % Mod;

z[k + d / 2] = (z1 - z2 + Mod) % Mod;

temp = ((LL)temp \* step) % Mod;

}

}

}

if (style == -1)

{

tmp = exp\_mod(len, Mod - 2);

for (int i = 0; i < len; i++)

z[i] = ((LL)z[i] \* tmp) % Mod;

}

}

void pre(int l)

{

g[0] = \_g[0] = 1;

for (int i = 1; i <= l; i++)

{

g[i] = exp\_mod(G, (Mod - 1) / (1 << i));

\_g[i] = exp\_mod(g[i], Mod - 2);

}

}

}F;

int n;

int N,M;

const int maxm=100010;

int A[maxm];

LL two[maxm],jie[maxm];

LL inv[maxm];

int t1[maxn],t2[maxn];

int a[maxn],b[maxn],c[maxn];

void prepare(){

two[0]=jie[0]=1;

for(int i=1;i<=100000;i++){

two[i]=two[i-1]\*2%Mod;

jie[i]=jie[i-1]\*i%Mod;

}

inv[100000]=exp\_mod(jie[100000],Mod-2);

for(int i=99999;i>=0;i--) inv[i]=inv[i+1]\*(i+1)%Mod;

}

LL ans[maxn];

bool cmp(int a,int b){

return a>b;

}

int main()

{

prepare();

/\*scanf("%d%d",&N,&M);

++N; ++M; int t=N+M-1;

for(int i=0;i<N;i++) scanf("%d",a+i);

for(int i=0;i<M;i++) scanf("%d",b+i);

fftinit(t); fft(a,0); fft(b,0);

for(int i=0;i<K;i++) c[i]=(ll)a[i]\*b[i]%MOD;

fft(c,1);

for(int i=0;i<t;i++) printf("%d ",c[i]);\*/

int T;

scanf("%d",&T);

for(int Tcase=1;Tcase<=T;Tcase++){

scanf("%d",&n);

for(int i=1;i<=n;i++) scanf("%d",&A[i]);

sort(A+1,A+n+1,cmp);

for(int i=0;i<n;i++) a[i]=(jie[i]\*two[n-(i+1)]%Mod)\*A[i+1]%Mod;

for(int i=0;i<=n;i++){

b[i]=inv[n-i];

}

b[0]=0;

N=n;M=n+1;

int t=N+M-1,l=0,len;

while((1<<l)<=t) l++;

F.pre(l);len=1<<l;

for(int i=N;i<len;i++) a[i]=0;

for(int i=M;i<len;i++) b[i]=0;

F.FFT(t1,a,l,1);

F.FFT(t2,b,l,1);

for(int i=0;i<len;i++) t1[i]=(LL)t1[i]\*t2[i]%Mod;

F.FFT(c,t1,l,-1);

ans[0]=0;

for(int i=1;i<=n;i++){

ans[i]=(ans[i-1]+(LL)c[i+n-1]\*inv[i-1])%Mod;

printf("%I64d",ans[i]);

printf(" ");

}puts("");

}

}

# 数据结构

## 后缀数组

/\*

后缀数组

\*/

#include<stdio.h>

#include<string.h>

#include<algorithm>

using namespace std;

const int maxn=500010<<1;

inline int lowbit(int x) {return x&(-x);}

char s[maxn];

int n;

int pre[maxn],next[maxn];

int C[maxn];

int query\_max(int x){

int res=0;

while(x) res=max(res,C[x]),x-=lowbit(x);

return res;

}

void insert\_max(int x,int val){

while(x<=n) C[x]=max(C[x],val),x+=lowbit(x);

}

int query\_min(int x){

int res=n;

while(x) res=min(res,C[x]),x-=lowbit(x);

return res;

}

void insert\_min(int x,int val){

while(x<=n) C[x]=min(C[x],val),x+=lowbit(x);

}

int sa[maxn],t1[maxn],t2[maxn],c[maxn],height[maxn],rank[maxn];

void get\_sa(int m){

int \*x=t1,\*y=t2;

for(int i=0;i<m;i++) c[i]=0;

for(int i=0;i<n;i++) c[x[i]=s[i]]++;

for(int i=1;i<m;i++) c[i]+=c[i-1];

for(int i=0;i<n;i++) sa[--c[x[i]]]=i;

for(int k=1;;k<<=1){

int p=0;

for(int i=n-1;i+k>=n;i--) y[p++]=i;

for(int i=0;i<n;i++) if(sa[i]>=k) y[p++]=sa[i]-k;

for(int i=0;i<m;i++) c[i]=0;

for(int i=0;i<n;i++) c[x[i]]++;

for(int i=1;i<m;i++) c[i]+=c[i-1];

for(int i=n-1;i>=0;i--) sa[--c[x[y[i]]]]=y[i];

swap(x,y);

x[sa[0]]=p=0;

for(int i=1;i<n;i++)

x[sa[i]]=y[sa[i]]==y[sa[i-1]]&&y[sa[i]+k]==y[sa[i-1]+k]?p:++p;

p++;

if(p>=n) break;

m=p;

}

//for(int i=0;i<n;i++)

//printf("sa[%d]:%d\n",i,sa[i]);

}

void get\_height(){

for(int i=0;i<n;i++) rank[sa[i]]=i;

int k=0;

for(int i=0;i<n;i++){

if(k) k--;

if(rank[i]==0) continue;

int j=sa[rank[i]-1];

while(s[i+k]==s[j+k]) k++;

height[rank[i]]=k;

}

//for(int i=1;i<n;i++)

//printf("height[%d]:%d\n",i,height[i]);

}

int same[maxn];

int main(){

scanf("%s",s);

n=strlen(s);

get\_sa('z'+1);

get\_height();

memset(C,0,sizeof(C));

for(int i=1;i<n;i++){

pre[i]=query\_max(height[i]+1);

//printf("pre[%d]:%d\n",i,pre[i]);

//if(height[pre[i]]==height[i]) same[i]=1;

insert\_max(height[i]+1,i);

}

for(int i=0;i<=n;i++) C[i]=n;

for(int i=n-1;i>=1;i--){

next[i]=query\_min(height[i]+1-1);

//printf("next[%d]:%d\n",i,next[i]);

insert\_min(height[i]+1,i);

}

long long ans=0;

ans+=(long long)(n+1)\*n/2\*(n-1);

for(int i=1;i<n;i++){

if(same[i]) continue;

ans-=(long long)(i-pre[i]+same[i])\*(next[i]-i)\*height[i]\*2;

//printf("%lld\n",(long long)(i-pre[i])\*(next[i]-i)\*height[i]\*2);

}

printf("%lld\n",ans);

return 0;

}

## 后缀自动机

//后缀自动机

//SAM

//Suffix Automation

struct State{

int len;

State \*par,\*go[26];

State(){

}

};

State \*root,\*last;

State seq[maxn\*2];int seq\_cnt;

State \*New(int \_len){

State \*o=&seq[seq\_cnt++];

memset(o->go,0,sizeof(o->go));o->par=0;o->len=\_len;

return o;

}

void extend(int w){

State \*p=last;

State \*np=New(p->len+1);

while(p&&p->go[w]==0) p->go[w]=np,p=p->par;

if(p==0) np->par=root;

else{

State \*q=p->go[w];

if(q->len==p->len+1){

np->par=q;

}

else{

State \*nq=New(p->len+1);

memcpy(nq->go,q->go,sizeof(q->go));

nq->par=q->par;

q->par=nq;

np->par=nq;

while(p&&p->go[w]==q)

p->go[w]=nq,p=p->par;

}

}

last=np;

}

void init(){

seq\_cnt=0;root=New(0);last=root;

}

## KMP

/\*

poj2406

2015.7.11

求最小循环节

\*/

# include<stdio.h>

# include<string.h>

# include<algorithm>

using namespace std;

const int maxn=1000010;

char s[maxn];

int next[maxn];

void getnext(char \*s,int n){

int i=0,j=-1;

next[0]=-1;

while(i<n){

if(s[i]==s[j]||j==-1){

i++;j++;

next[i]=j;

}

else j=next[j];

}

}

int main()

{

int n;

while(scanf("%s",s)!=EOF){

if(s[0]=='.') break;

n=strlen(s);

getnext(s,n);

int t=next[n];

if(n%(n-t)==0) printf("%d\n",n/(n-t));

else printf("1\n");

}

return 0;

}

## 扩展KMP

/\*

扩展kmp

poj3461

\*/

# include<stdio.h>

# include<string.h>

# include<algorithm>

using namespace std;

const int maxn=1000010;

char A[maxn],B[maxn];//A母串 B子串

int lenA,lenB;

void getExtendNext (char \*t, int \*next)

{

int lt = strlen(t);

for (int i = 1, j = -1, a, p; i < lt; i++, j--)

if (j < 0 || i + next[i - a] >= p)

{

if (j < 0) j = 0, p = i;

while (p < lt && t[j] == t[p]) j++, p++;

next[i] = j, a = i;

}

else next[i] = next[i - a];

}

void Getextand(int \*next,char \*str,char \*mode,int \*extand){//str母串 mode子串

int strLen=strlen(str);

int modeLen=strlen(mode);

int i,a,p,j=-1;

for(i=0;i<strLen;++i,--j){

if(j<0||i+next[i-a]>=p){

if(j<0) j=0,p=i;

while(p<strLen&&j<modeLen&&str[p]==mode[j]) ++p,++j;

extand[i]=j;a=i;

}

else extand[i]=next[i-a];

}

}

int exnext[maxn],extand[maxn];

int main()

{

int Tcase;

scanf("%d",&Tcase);

while(Tcase--){

scanf("%s%s",B,A);

lenA=strlen(A);lenB=strlen(B);

getExtendNext(B,exnext);

Getextand(exnext,A,B,extand);

int ans=0;

for(int i=0;i<lenA;i++)

if(extand[i]==lenB)

ans++;

printf("%d\n",ans);

}

return 0;

}

## 回文树

# include<stdio.h>

# include<string.h>

# include<algorithm>

using namespace std;

const int maxn=300010;

const int SIGMA=26;

typedef long long LL;

struct Palindromic\_Tree{

int ch[maxn][SIGMA];

int fail[maxn];

int cnt[maxn];

int num[maxn];

int len[maxn];

int S[maxn];

int last;

int n;

int p;

int newnode(int l){

for(int i=0;i<SIGMA;i++) ch[p][i]=0;

cnt[p]=num[p]=0;len[p]=l;

return p++;

}

void init(){

p=0;

newnode(0);

newnode(-1);

last=0;

n=0;

S[n]=-1;

fail[0]=1;

}

int get\_fail(int x){

while(S[n]!=S[n-len[x]-1]) x=fail[x];

return x;

}

void add(int c){

c-='a';

S[++n]=c;

int cur=get\_fail(last);

if(!ch[cur][c]){

int now=newnode(len[cur]+2);

fail[now]=ch[get\_fail(fail[cur])][c];

ch[cur][c]=now;

num[now]=num[fail[now]]+1;

}

last=ch[cur][c];

cnt[last]++;

}

void count(){

for(int i=p-1;i>=0;i--) cnt[fail[i]]+=cnt[i];

}

}Tree;

char s[maxn];

int main(){

scanf("%s",s);

int n=strlen(s);

Tree.init();

for(int i=0;i<n;i++){

Tree.add(s[i]);

}

Tree.count();

LL ans=0;

for(int i=0;i<Tree.p;i++){

ans=max(ans,(LL)Tree.cnt[i]\*Tree.len[i]);

}

printf("%lld\n",ans);

return 0;

}

## 树链剖分

树链剖分

//bzoj2243

# include<stdio.h>

# include<string.h>

# include<algorithm>

using namespace std;

const int maxn=100010;

const int INF=~0U>>1;

int n,m;

int weight[maxn];

int rcol,lcol;

struct Node{

int lc,rc,sum,lazy;

Node() {}

Node(int lc,int rc,int sum,int lazy=-1):lc(lc),rc(rc),sum(sum),lazy(lazy) {}

};

struct SeqmentTree{

Node s[maxn<<2];

void pushup(int o){

s[o]=Node(s[o<<1].lc,s[o<<1|1].rc,s[o<<1].sum+s[o<<1|1].sum-(s[o<<1].rc==s[o<<1|1].lc));

}

void pushdown(int o){

if(s[o].lazy!=-1){

s[o<<1].lazy=s[o<<1|1].lazy=s[o<<1].lc=s[o<<1].rc=s[o<<1|1].lc=s[o<<1|1].rc=s[o].lazy;

s[o<<1].sum=s[o<<1|1].sum=1;

s[o].lazy=-1;

}

}

void build(int o,int L,int R){

if(L==R){

s[o]=Node(weight[L],weight[L],1);return;

}

int M=(L+R)>>1;

build(o<<1,L,M);

build(o<<1|1,M+1,R);

pushup(o);

}

void change(int o,int L,int R,int qL,int qR,int col){

if(qL<=L&&qR>=R){

s[o].sum=1;s[o].lc=s[o].rc=col;s[o].lazy=col;return;

}

pushdown(o);

int M=(L+R)>>1;

if(qL<=M) change(o<<1,L,M,qL,qR,col);

if(qR>M) change(o<<1|1,M+1,R,qL,qR,col);

pushup(o);

}

int query(int o,int L,int R,int qL,int qR){

if(qL<=L&&qR>=R){

if(L==qL) lcol=s[o].lc;

if(R==qR) rcol=s[o].rc;

return s[o].sum;

}

pushdown(o);

int M=(L+R)>>1;

if(qR<=M) return query(o<<1,L,M,qL,qR);

else if(qL>M) return query(o<<1|1,M+1,R,qL,qR);

else{

return query(o<<1,L,M,qL,qR)+query(o<<1|1,M+1,R,qL,qR)-(s[o<<1].rc==s[o<<1|1].lc);

}

}

};

SeqmentTree seq;

int first[maxn],next[maxn<<1],v[maxn<<1],edge=0,st\_w[maxn];

void add\_edge(int a,int b){

next[edge]=first[a];v[edge]=b;first[a]=edge++;

}

int fa[maxn],son[maxn],size[maxn],dfn[maxn],dfs\_clock=0,top[maxn],dep[maxn];

void q\_change(int a,int b,int c){

int ta=top[a],tb=top[b];

while(ta!=tb){

if(dep[ta]<dep[tb]) swap(a,b),swap(ta,tb);

seq.change(1,1,n,dfn[ta],dfn[a],c);//printf("ta:%d a:%d\n",ta,a);

a=fa[ta];ta=top[a];

}

if(dep[a]>dep[b]) swap(a,b);

seq.change(1,1,n,dfn[a],dfn[b],c);//printf("a:%d b:%d\n",a,b);

}

void q\_ask(int a,int b){

int ta=top[a],tb=top[b],ca=-1,cb=-1,ans=0;

while(ta!=tb){

if(dep[ta]<dep[tb]) swap(a,b),swap(ta,tb),swap(ca,cb);

ans+=seq.query(1,1,n,dfn[ta],dfn[a]);ans-=rcol==ca;//printf("ta:%d a:%d\n",ta,a);

a=fa[ta];

ta=top[a];ca=lcol;

}

if(dep[a]>dep[b]) swap(a,b),swap(ca,cb);

ans+=seq.query(1,1,n,dfn[a],dfn[b]);ans-=rcol==cb;ans-=lcol==ca;//printf("a:%d b:%d\n",a,b);

printf("%d\n",ans);

}

void dfs\_size(int x){

size[x]=1;son[x]=0;

for(int e=first[x];e!=-1;e=next[e]){

if(v[e]==fa[x]) continue;

fa[v[e]]=x;//printf("x:%d v:%d\n",x,v[e]);

dfs\_size(v[e]);size[x]+=size[v[e]];

if(size[v[e]]>size[son[x]]) son[x]=v[e];

}

}

void dfs(int x){

dfn[x]=++dfs\_clock;

//printf("son[%d]:%d\n",x,son[x]);

if(son[x]) top[son[x]]=top[x],dep[son[x]]=dep[x]+1,dfs(son[x]);

for(int e=first[x];e!=-1;e=next[e]){

if(v[e]==fa[x]||v[e]==son[x]) continue;

top[v[e]]=v[e];dep[v[e]]=dep[x]+1;

dfs(v[e]);

}

}

int main(){

//freopen("input.txt","r",stdin);

int a,b,c;

char op[20];

scanf("%d",&n); scanf("%d",&m);

for(int i=1;i<=n;i++)

scanf("%d",&st\_w[i]);

memset(first,-1,sizeof(first));

for(int i=1;i<n;i++)

scanf("%d%d",&a,&b),add\_edge(a,b),add\_edge(b,a);size[0]=0;

fa[1]=0;dfs\_size(1);top[1]=1;dep[1]=1;

dfs(1);

for(int i=1;i<=n;i++)

weight[dfn[i]]=st\_w[i];

seq.build(1,1,n);

for(int mm=1;mm<=m;mm++){

scanf("%s",op);

if(op[0]=='C'){

scanf("%d%d%d",&a,&b,&c);

q\_change(a,b,c);

}

else{

scanf("%d%d",&a,&b);

q\_ask(a,b);

}

}

return 0;

}

## 点分治

//bzoj1468

# include<stdio.h>

# include<queue>

# include<string.h>

# include<algorithm>

using namespace std;

const int maxn=40010;

int n,m;

int first[maxn],next[maxn<<1],v[maxn<<1],w[maxn<<1],edge;

int fa[maxn],size[maxn];

bool used[maxn];

int num;

void add\_edge(int a,int b,int c){

next[edge]=first[a];

v[edge]=b;

w[edge]=c;

first[a]=edge++;

}

void dfs\_size(int x){

size[x]=1;

for(int e=first[x];e!=-1;e=next[e])

if(v[e]!=fa[x]) fa[v[e]]=x,dfs\_size(v[e]),size[x]+=size[v[e]];

}

void choose(int &x){

int mx=0,son;

for(int e=first[x];e!=-1;e=next[e]){

if(!used[v[e]]&&size[v[e]]>mx)

mx=size[v[e]],son=v[e];

}

if(mx>num/2){

fa[x]=son;size[x]-=size[son];size[son]+=size[x];fa[son]=0;

x=son;

choose(x);

}

}

int dist[maxn],cnt;

typedef pair<int,int> pi;

int cal(int root,int dis){

cnt=0;

queue<pi> Q;

Q.push(pi(root,dis));

while(!Q.empty()){

pi now=Q.front();Q.pop();dist[cnt++]=now.second;

for(int e=first[now.first];e!=-1;e=next[e]){

if(v[e]!=fa[now.first]&&!used[v[e]])

Q.push(pi(v[e],now.second+w[e]));

}

}

sort(dist,dist+cnt);

//printf("root:%d dis:%d\n",root,dis);for(int i=0;i<cnt;i++) printf("%d ",dist[i]);puts("");

int tail=cnt-1;

int res=0;

for(int i=0;i<tail;i++){

while(tail>i&&dist[tail]+dist[i]>m) tail--;

res+=tail-i;

}

//printf("res:%d\n",res);

return res;

}

int ans=0;

void solve(int x){

num=size[x];

if(num==1) return;

choose(x);//printf("x:%d\n",x);

ans+=cal(x,0);

for(int e=first[x];e!=-1;e=next[e])

if(!used[v[e]])

ans-=cal(v[e],w[e]);

used[x]=1;

for(int e=first[x];e!=-1;e=next[e]){

if(!used[v[e]]) solve(v[e]);

}

}

int main(){

int a,b,c;

scanf("%d",&n);

memset(first,-1,sizeof(first));edge=0;

for(int i=1;i<n;i++)

scanf("%d%d%d",&a,&b,&c),add\_edge(a,b,c),add\_edge(b,a,c);

scanf("%d",&m);

fa[1]=0;

dfs\_size(1);memset(used,false,sizeof(used));

solve(1);

printf("%d\n",ans);

return 0;

}

## LINK-CUT-TREE

struct Node{

Node \*ch[2],\*f,\*p;

int v,\_min,bh;

bool rev;

Node() {}

};

Node \*null;

Node seq[maxn];int seq\_cnt=0;

bool ok=true;

Node \*New(int bh,int val){

Node \*o=&seq[seq\_cnt++];

o->v=o->\_min=val;o->bh=bh;

o->ch[0]=o->ch[1]=o->f=null;

o->p=o;o->rev=false;

return o;

}

void maintain(Node \*o){

o->p=o;o->\_min=o->v;

if(o->ch[0]->\_min<o->\_min) o->\_min=o->ch[0]->\_min,o->p=o->ch[0]->p;

if(o->ch[1]->\_min<o->\_min) o->\_min=o->ch[1]->\_min,o->p=o->ch[1]->p;

}

void pushdown(Node \*o){

if(o->rev){

swap(o->ch[0],o->ch[1]);

o->ch[0]->rev^=1;o->ch[1]->rev^=1;

o->rev=false;

}

}

struct LCT{

Node \*tree[maxn];

bool havef(Node \*o){

return (o->f!=null)&&(o->f->ch[0]==o||o->f->ch[1]==o);

}

void Rotate(Node \*p){

int d=p->f->ch[1]==p?0:1;

Node \*o=p->f;

o->ch[!d]=p->ch[d];p->ch[d]->f=o;

if(havef(o)) o->f->ch[o->f->ch[1]==o]=p;

p->ch[d]=o;

p->f=o->f;

o->f=p;

maintain(o);maintain(p);

}

void splay(Node \*o){

Node \*p;

while(havef(o)){

p=o->f;

if(havef(p)){

pushdown(p->f);

pushdown(p);

pushdown(o);

p=p->f;

int d=p->ch[1]==o->f;

int dd=o->f->ch[1]==o;

if(d==dd)

Rotate(p->ch[d]),Rotate(o);

else Rotate(o),Rotate(o);

}

else pushdown(p),pushdown(o),Rotate(o);

}

}

void Access(Node \*x){

Node \*y=null;

while(x!=null){

splay(x);

pushdown(x);

x->ch[1]=y;

maintain(x);

y=x;

x=x->f;

}

}

/\*void print(Node \*x){

pushdown(x);

if(x->ch[0]!=null&&x->ch[0]->f==x) print(x->ch[0]);

if(x->ch[1]!=null&&x->ch[1]->f==x) print(x->ch[1]);

}\*/

void makeroot(Node \*x){

Access(x);

splay(x);

x->rev^=1;

}

void Link(Node \*x,Node \*y){

// if(ok) printf("Link: %d %d \n",x->v,y->v);

makeroot(x);

x->f=y;

Access(x);

}

void Cut(Node \*x,Node \*y){

//if(ok) printf("CUT %d %d\n",x->v,y->v);

Access(x);

splay(y);

if(y->f==x){

y->f=null;

}

else {

Access(y);

splay(x);

if(x->f==y){

x->f=null;

}

}

}

Node\* query(Node \*x,Node \*y){

makeroot(x);

Access(y);

splay(y);

return y->p;

}

Node \*getroot(Node \*x){

Access(x);splay(x);

while(x->ch[0]!=null) x=x->ch[0];

return x;

}

};

LCT lct;

# 几何

const double eps = 1e-6;

int dcmp(double x) {

if(fabs(x) < eps) return 0; else return x < 0 ? -1 : 1;

}

const double PI = acos(-1);

struct Point {

double x, y;

Point(double x=0, double y=0):x(x),y(y) { }

};

typedef Point Vector;

Vector operator + (Vector A, Vector B) { return Vector(A.x+B.x, A.y+B.y); }

Vector operator - (Point A, Point B) { return Vector(A.x-B.x, A.y-B.y); }

Vector operator \* (Vector A, double p) { return Vector(A.x\*p, A.y\*p); }

Vector operator / (Vector A, double p) { return Vector(A.x/p, A.y/p); }

bool operator < (const Point& a, const Point& b) {

return a.x < b.x || (a.x == b.x && 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(Vector A, Vector B) { return A.x\*B.x + A.y\*B.y; }

double Length(Vector A) { return sqrt(Dot(A, A)); }

double Angle(Vector A, Vector B) { return acos(Dot(A, B) / Length(A) / Length(B)); }

double Cross(Vector A, Vector B) { return A.x\*B.y - A.y\*B.x; }

Vector Rotate(Vector A, double rad) {

return Vector(A.x\*cos(rad)-A.y\*sin(rad), A.x\*sin(rad)+A.y\*cos(rad));

}

Vector Normal(Vector A) {

double L = Length(A);

return Vector(-A.y/L, A.x/L);

}

Point GetLineIntersection(Point P, Point v, Point Q, Point w) {

Vector u = P-Q;

double t = Cross(w, u) / Cross(v, w);

return P+v\*t;

}

Point GetLineProjection(Point P, Point A, Point B) {

Vector v = B-A;

return A+v\*(Dot(v, P-A) / Dot(v, v));

}

double DistanceToLine(Point P, Point A, Point B) {

Vector v1 = B - A, v2 = P - A;

return fabs(Cross(v1, v2)) / Length(v1); // 如果不取绝对值，得到的是有向距离

}

struct Line {

Point p;

Vector v;

Line(Point p, Vector v):p(p),v(v) { }

Point point(double t) {

return p + v\*t;

}

Line move(double d) {

return Line(p + Normal(v)\*d, v);

}

};

struct Circle {

Point c;

double r;

Circle(Point c, double r):c(c),r(r) {}

Point point(double a) {

return Point(c.x + cos(a)\*r, c.y + sin(a)\*r);

}

};

Point GetLineIntersection(Line a, Line b) {

return GetLineIntersection(a.p, a.v, b.p, b.v);

}

double angle(Vector v) {

return atan2(v.y, v.x);

}

int getLineCircleIntersection(Line L, Circle C, double& t1, double& t2, vector<Point>& sol){

double a = L.v.x, b = L.p.x - C.c.x, c = L.v.y, d = L.p.y - C.c.y;

double e = a\*a + c\*c, f = 2\*(a\*b + c\*d), g = b\*b + d\*d - C.r\*C.r;

double delta = f\*f - 4\*e\*g; // 判别式

if(dcmp(delta) < 0) return 0; // 相离

if(dcmp(delta) == 0) { // 相切

t1 = t2 = -f / (2 \* e); sol.push\_back(L.point(t1));

return 1;

}

// 相交

t1 = (-f - sqrt(delta)) / (2 \* e); sol.push\_back(L.point(t1));

t2 = (-f + sqrt(delta)) / (2 \* e); sol.push\_back(L.point(t2));

return 2;

}

int getCircleCircleIntersection(Circle C1, Circle C2, vector<Point>& sol) {

double d = Length(C1.c - C2.c);

if( dcmp(C1.r - C2.r)>0)

swap(C1,C2);

if(dcmp(d) == 0) {

if(dcmp(C1.r - C2.r) == 0) return -1; // 重合，无穷多交点

return 0;

}

if(dcmp(C1.r + C2.r - d) < 0) return 0;

if(dcmp(fabs(C1.r-C2.r) - d) > 0) return 0;

if(dcmp(C1.r + C2.r - d) == 0 || dcmp(fabs(C1.r - C2.r)-d) == 0){

Point p = C1.c-C2.c;

sol.push\_back(C2.c + p / Length(p) \* C2.r);

return 1;

}

double a = angle(C2.c - C1.c);

double da = acos((C1.r\*C1.r + d\*d - C2.r\*C2.r) / (2\*C1.r\*d));

Point p1 = C1.point(a-da), p2 = C1.point(a+da);

sol.push\_back(p1);

if(p1 == p2) return 1;

sol.push\_back(p2);

return 2;

}

## 向量旋转

Vector Rotate(Vector A, double rad) {

return Vector(A.x \* cos(rad) - A.y \* sin(rad), A.x \* sin(rad) + A.y \* cos(rad));

}

## 凸包

//凸包

# include<stdio.h>

# include<string.h>

# include<math.h>

# include<algorithm>

using namespace std;

const int maxn=100010;

int sgn(double a) {if(a<0) return -1;if(a>0) return 1;return 0;}

struct point{

double x,y;

int index;

point(double x=0,double y=0):x(x),y(y) {}

};

double operator ^(const point &a,const point &b){return a.x\*b.y-a.y\*b.x;}

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);}

bool operator < (const point& a,const point& b){if(a.y==b.y) return a.x<b.x;return a.y>b.y;}

int turn(const point &a,const point &b,const point &c){//1:left -1:right

return sgn((b-a)^(c-b));

}

int convex\_hull(point \*p,int n,int \*stk){//p是点集 n是点数 stk是凸包上的点 函数返回凸包上的点数

sort(p+1,p+n+1);

stk[1]=1;stk[2]=2;

int top=2;

for(int i=3;i<=n;i++){

while(top>=2&&turn(p[stk[top-1]],p[stk[top]],p[i])>=0) top--;

stk[++top]=i;

}

int tmp=top;

for(int i=n-1;i>=1;i--){

while(top>tmp&&turn(p[stk[top-1]],p[stk[top]],p[i])>=0) top--;

stk[++top]=i;

}

top--;

return top;

}