/\*

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(int 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;

int c[27][27];

int n,k,m;

char s[maxn],stu[maxn];

int main(){

int tcase;

char t;

scanf("%d",&tcase);

while(tcase--){

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

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

while((t=getchar())&&(t<'A'||t>'Z'));

s[i]=t;

}

int S=0,T=26+26+1;

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

g.init(T+1);

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

while((t=getchar())&&(t<'A'||t>'Z'));

stu[i]=t;

}

for(int i=1;i<=26;i++)

g.add\_edge(S,i,1,0),g.add\_edge(26+i,T,1,0);

memset(c,0,sizeof(c));

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

c[stu[i]-'A'+1][s[i]-'A'+1]++;

for(int i=1;i<=26;i++)

for(int j=1;j<=26;j++)

g.add\_edge(i,26+j,1,-c[i][j]);

int ans=-g.Mincost(S,T);

double output=ans/(n\*1.0);

printf("%.4lf\n",output);

}

}

return 0;

}

# include<stdio.h>

# include<string.h>

# include<math.h>

# include<algorithm>

using namespace std;

const int maxn=55;

const int maxt=(1<<10)+50;

const int INF=100000000;

int bit\_count[maxt],fact[10];

int n,m,k;

int D[maxn][maxn];

int dp[maxn][maxt];

struct tank{

int val,num;

tank(int val=0,int num=0):val(val),num(num) {}

}New[maxn];

bool cmp(tank a,tank b){return a.val<b.val;}

bool change[maxn];

void Floyed(){

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

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

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

D[i][j]=min(D[i][j],D[i][k]+D[k][j]);

}

void Steiner\_DP(){

int now,val;

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

for(int j=0;j<(1<<(k+k));j++)

dp[i][j]=INF;

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

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

dp[i][1<<(j-1)]=D[i][j];

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

dp[i][1<<(j-1)]=D[i][n-k+(j-k)];

}

for(int num=2;num<=k+k;num++){

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

if(bit\_count[j]!=num) continue;

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

for(int k=(j-1)&j;k;k=(k-1)&j){

dp[i][j]=min(dp[i][j],dp[i][k]+dp[i][j^k]);

}

New[i]=tank(dp[i][j],i);

}

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

memset(change,0,sizeof(change));

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

now=New[i].num;val=New[i].val;

if(change[now]) continue;

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

if(val+D[now][k]<dp[k][j]) dp[k][j]=val+D[now][k],change[k]=1;

}

}

}

return;

}

int p[10];

int Get(int s){

int ans=s;

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

if((s>>(i-1))&1) ans=ans^(1<<(p[i]-1));

return ans;

}

int f[maxt];

int main(){

for(int j=0;j<(1<<10);j++){

bit\_count[j]=0;

for(int i=0;i<10;i++)

bit\_count[j]+=(j>>i)&1;

}

fact[0]=1;for(int i=1;i<=5;i++) fact[i]=fact[i-1]\*i;

int tcase,a,b,c,ans;

scanf("%d",&tcase);

while(tcase--){

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

for(int i=1;i<=n;i++)for(int j=1;j<=n;j++) D[i][j]=INF;

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

scanf("%d%d%d",&a,&b,&c);D[a][b]=D[b][a]=min(D[a][b],c);

}

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

Floyed();

Steiner\_DP();

for(int i=1;i<=k;i++) p[i]=k+i;

ans=INF;

for(int tt=1;tt<=fact[k];tt++){

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

f[j]=INF;

int k=Get(j);

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

f[j]=min(f[j],dp[i][k]);

}

for(int num=2;num<=k;num++){

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

if(bit\_count[j]!=num) continue;

for(int k=(j-1)&j;k;k=(k-1)&j)

f[j]=min(f[j],f[k]+f[j^k]);

}

}

ans=min(ans,f[(1<<k)-1]);

next\_permutation(p+1,p+k+1);

}

if(ans>=INF) puts("No solution");

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

}

return 0;

}

/\*

sgu120 给正n边形两点 搞出n边形 方法是搞出中心然后向量旋转

\*/

# include<stdio.h>

# include<string.h>

# include<math.h>

# include<algorithm>

using namespace std;

const int N = 155;

const double pi = acos(-1.0);

int n, n1, n2;

struct Point {

double x, y;

int id;

Point() {}

Point(double x, double y) {

this->x = x;

this->y = y;

}

} p1, p2, ans[N];

bool cmpid(Point a, Point b) {

return a.id < b.id;

}

typedef Point Vector;

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

Vector operator - (Vector A, Vector 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); }

Vector Rotate(Vector A, double rad) {

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

}

double Dot(Vector A, Vector B) {return A.x \* B.x + A.y \* B.y;} //点积

double Length(Vector A) {return sqrt(Dot(A, A));} //向量的模

int main(){

int ok;

double d1,d2,d3;

Vector v1,v2,v3;

Point midpoint;

double length1,length2;

while(scanf("%d%d%d",&n,&n1,&n2)!=EOF){

scanf("%lf%lf%lf%lf",&p1.x,&p1.y,&p2.x,&p2.y);

if(n1>n2) swap(n1,n2),swap(p1,p2);

d1=2\*pi/n;d1=d1\*(n2-n1);ok=-1;

if(d1>=pi) d1=2\*pi-d1,ok=1;

//printf("d1:%.3lf d1:%d\n",d1,(int)(d1/pi\*180));

d2=(pi-d1)/2.0;

v1=(p2-p1)\*ok;

v2=Rotate(v1,d2);

length1=Length(p1-p2);

if(d1>pi||d1<pi)

length2=length1\*sin(d2)/sin(d1);

else length2=length1/2.0;

v2=v2\*(length2/length1);

if(ok==-1) midpoint=p2+v2;

else midpoint=p1+v2;

v3=p1-midpoint;d3=2\*pi/n;int now=n1;

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

ans[i]=midpoint+v3;

v3=Rotate(v3,d3);

ans[i].id=now;

if(--now==0) now=n;

}

sort(ans,ans+n,cmpid);

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

printf("%.6lf %.6lf\n",ans[i].x,ans[i].y);

}

return 0;

}

/\*

zoj1729

2015.7.11

找出一个字符串中字典序最小的开头

\*/

# include<stdio.h>

# include<string.h>

# include<algorithm>

using namespace std;

const int maxn=100010;

int n;

char s[maxn];

int MCP(int n,char \*s){

int i=0,j=1,count=0,x,y;

while(i<n&&j<n&&count<n){

x=i+count;y=j+count;

if(x>=n) x-=n;if(y>=n) y-=n;

if(s[x]==s[y]) {count++;continue;}

else if(s[x]>s[y]){

i=i+count+1;j=i+1;

}

else

j=j+count+1;

//if(i==j) j++;

count=0;

}

return i;

}

int main()

{

int Tcase;

scanf("%d",&Tcase);

while(Tcase--){

scanf("%d",&n);

scanf("%s",s);

printf("%d\n",MCP(n,s));

}

return 0;

}

# 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;

}

# 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;

}

point p[maxn];

int n,stk[maxn],num;

int main()

{

//freopen("G.out","w",stdout);

int Tcase,top;

scanf("%d",&Tcase);

while(Tcase--){

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

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

scanf("%lf%lf",&p[i].x,&p[i].y);p[i].index=i;

}

top=convex\_hull(p,n,stk);

printf("%d",num);

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

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

printf("%d %d\n",(int)p[stk[i]].x,(int)p[stk[i]].y);

}

}

return 0;

}

# 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;

}

/\*

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;

}

/\*

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;

}

/\*

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;

}