1 Structures

1.1 Big Integer

```
#include<cstdio>
#define lld "lld" //change this into "I64d" if using windows.
const int Size=100;
const long long D=1000000000ll;
char __tmp[(Size+5)*9];
class bigint
{
    void fix(){
        s[n]=0;
        for(int i=0;i<n;i++)
            s[i+1]+=s[i]/D,s[i]%=D;
        if(s[n])n++;
        s[n]=0;
    }
public:
    int n;
    long long s[Size];
    bigint(const char a[]){
        int i;
        for(i=0;a[i];i++);
        for(n=0;(i-=9)>=0;n++)
            sscanf(a+i,"%9"lld,s+n);
        sprintf(__tmp,"%%%d"lld,9+i);
        if(i+9>0)sscanf(a,__tmp,s+n++);
        s[n]=0;
    }
    bigint(const long long &a){s[1]=a/D,s[0]=a%D,s[2]=0,n=1+!!s[1];}
    bigint(const int &a){*this=0ll+a;}
    bigint(){*this=0;}
    const char *tostr()const{
        int i,j;
        char *t=__tmp;
        sprintf(t,"%lld",s[n-1]);
        for(i=0;t[i];i++);
        for(j=n-2;j>=0;j--,i+=9)
            sprintf(t+i,"%09lld",s[j]);
        return t;
    }
    bigint& operator+=(const bigint &t)
    {
        for(int i=n;i<=t.n;i++)s[i]=0;
        if(n<t.n)n=t.n;
        for(int i=0;i<t.n;i++)
            s[i]+=t.s[i];
        fix();
        return *this;
    }
    bigint& operator*=(const bigint &t)
    {
        bigint tmp=*this;
        const long long *a=tmp.s,*b=t.s;
        if(b==s)b=a;
        n+=t.n;
        for(int i=0;i<=n;i++)s[i]=0;
        for(int i=0;i<tmp.n;i++)
            for(int j=0;j<t.n;j++)
            {
                s[i+j]+=a[i]*b[j];
                s[i+j+1]+=s[i+j]/D,s[i+j]%=D;
            }
        while(n>1 && !s[n-1])n--;
        return *this;
    }
/*  inline void operator+=(const int &x)
    {
        s[0]+=x;
        fix();
    }
    inline void operator*=(const int &x)
    {
        for(int i=0;i<n;i++)
            s[i]*=x;
        fix();
    }*/
};
```

1.2 Fibonacci Heap

```
struct node
{
    int d,v,dg,mark;
    node *left,*right,*parent,*child;
    bool operator<(node a){return d<a.d;}
    void move(node *ptr)
    {
        left->right=right,right->left=left;
        left=ptr,right=ptr->right;
        left->right=this,right->left=this;
    }
    void link(node *p)
    {
        parent=p,p->dg++;
        if(!p->child)p->child=this;
        move(p->child);
    }
    void cut(node *&min)
    {
        if(!parent)return;
        parent->child=right;
        if(parent->child==this)parent->child=0;
        move(min);
        if(parent->mark)parent->cut(min);
        else parent->mark=1,parent->dg--;
        parent=0;
        if(*this<*min)min=this;
    }
    void decrease(int nd,node *&min)
    {
        d=nd;
        if(parent){if(*this<*parent)cut(min);}
        else if(*this<*min)min=this;
    }
};
class Fheap
{
    node *list[30];
    void bond()
    {
        int i,d;
        node *now,*next;
        for(i=0;i<30;i++)list[i]=0;
        min->parent=0,next=min->right;
        do{
            now=next,now->parent=0;
            next=now->right;
            d=now->dg;
            while(list[d])
            {
                if(*now<*list[d])list[d]->link(now);
                else now->link(list[d]),now=list[d];
                list[d++]=0;
            }
            list[d]=now;
        }while(now!=min && !min->parent);
        for(i=0;i<30;i++)
            if(list[i] && *list[i]<*min)
                min=list[i];
        while(min->parent)min=min->parent;
    }
public:
    int n;node *min;
    Fheap(){min=0,n=0;}
    node *insert(int d,int v)
    {
        node *ptr=new node;
        *ptr=(node){d,v,0,0,ptr,ptr,0,0};
        if(n++,min)ptr->move(min);
        else min=ptr;
        if(*ptr<*min)min=ptr;
        return ptr;
    }
    node extractmin()
    {
        node tmp=*min,*ptr=min;
        if(min->child)
        {
            min->right->left=min->child;
            min->child->right->left=min;
            ptr=min->right,min->right=min->child->right,
            min->child->right=ptr;
            min->child=0;
        }
        ptr=min,min=min->right;
        ptr->move(ptr);
        if(--n)bond();
        else min=0;
        delete ptr;
        return tmp;
    }
};
```

2 Graph

2.1 Minimum Spaning tree and Disjoint sets

```
#include<algorithm>
using namespace std;
const int V=1000;
int n,m,p[V],d[V],c[V][V];
int find(int v){
    if(p[v]==v)return v;
    return p[v]=find(p[v]);
}
void uni(int a,int b){
    a=find(a),b=find(b);
    if(d[a]<d[b])p[a]=b;
    else p[b]=a;
    if(d[a]==d[b])d[a]++;
}
void init(int n){
    while(n--)p[n]=n,d[n]=0;
}
struct pt{
    int x,y;
    int operator+(pt a){return (x-a.x)*(x-a.x)+(y-a.y)*(y-a.y);}
}pos[V];
struct ed{
    int s,t,c;
    bool operator<(ed x)const{return c<x.c;}
}s[V*V];
int kuskal(){
    std::sort(s,s+m);
    init(n);
    int i,sum;
    for(i=0;i<m;i++){
        if(find(s[i].s)==find(s[i].t))continue;
        uni(s[i].s,s[i].t);
        sum+=s[i].c;
    }
    return sum;
}
int prim(){
    const int inf=2147483647;
    int i,j,sum=0;
    for(i=0;i<n;i++)
        if(c[0][i]<inf)
            d[i]=c[0][i],p[i]=0;
        else d[i]=inf,p[i]=i;
    p[0]=-1;
    while(1){
        for(i=0;i<n;i++)
            if(d[i]>=0 && d[i]<inf)break;
        if(i>=n)break;
        for(j=i+1;j<n;j++)
            if(d[j]>=0 && d[j]<d[i])i=j;
        for(j=0;j<n;j++)
            if(c[i][j]<d[j])
                d[j]=c[i][j],p[j]=i;
        sum+=d[i];
        d[i]=-1;
    }
    return sum;
}
```

2.2 Matching

```
const int V=1005;
char p[V],c[V][V];
int t;
char go(int v){
    if(p[v])return 0;
    if(v==t)return 1;
    p[v]=1;
    for(int i=1;i<=t;i++)
        if(c[v][i]-- && go(i))
            return ++c[i][v];
        else c[v][i]++;
    return 0;
}
int flow(){
    int i,j=0;
    for(i=0;i<=t;i++)p[i]=0;
    while(go(0))
        for(j++,i=0;i<=t;i++)p[i]=0;
    return j;
}
```

2.3 Matching(Hopcroft-Karp algorithm)

```
const int V=1024,inf=2147483647;
int t,c[V][V],p[V],d[V],u[V],q[V];
int sp(){
    int i,j,k;
    for(i=0;i<=t;i++)
        d[i]=inf;
    d[q[0]=t]=0;
    for(i=0,j=1;d[0]>=inf && i<j;i++)
        for(k=0;k<=t;k++)
            if(c[k][q[i]] && d[k]>=inf)
                d[q[j++]=k]=d[q[i]]+1;
    return d[0];
}
int go(int v){
    if(v==t)return 1;
    int sum=0;
    for(;u[v]<=t;u[v]++)
        if(d[v]==d[u[v]]+1){
            if(c[v][u[v]]-- && go(u[v])){
                c[u[v]][v]++;
                if(v)return 1;
                sum++;
            }
            else c[v][u[v]]++;
        }
    return sum;
}
int flow(){
    int i,sum=0;
    while(sp()<inf){
        for(i=0;i<=t;i++)u[i]=0;
        sum+=go(0);
    }
    return sum;
}
```

2.4 Max-Flow

```
const int MaxV=200;
struct list{
    int v;
    list *prev,*next;
}*head;
int t,h[MaxV],e[MaxV],n[MaxV],c[MaxV][MaxV];
inline int min(int a,int b){
    if(a<b)return a;
    return b;
}
void makelist(){
    int i;
    list *ptr;
    head=new list;
    head->prev=head->next=NULL;
    ptr=head;
    for(i=1;i<t-1;i++,ptr=ptr->next){
        ptr->v=i;
        ptr->next=new list;
        ptr->next->prev=ptr;
    }
    ptr->v=t-1;
    ptr->next=NULL;
}
void cleanlist(list *&ptr=head){
    if(ptr){
        cleanlist(ptr->next);
        delete ptr;
        ptr=NULL;
    }
}
void tofront(list *v){
    v->prev->next=v->next;
    if(v->next!=NULL)
        v->next->prev=v->prev;
    v->next=head;
    head->prev=v;
    v->prev=NULL;
    head=v;
}
void push(int v,int u){
    int f=min(c[v][u],e[v]);
    e[v]-=f;
    e[u]+=f;
    c[v][u]-=f;
    c[u][v]+=f;
}
void relabel(int v){
    int i,min;
    for(i=0,min=2*t;i<=t;i++)
        if(c[v][i]>0 && h[i]+1<min)
            min=h[i]+1;
    h[v]=min;
}
void discharge(int v){
    while(e[v]>0){
        if(n[v]>t)
            relabel(v),n[v]=0;
        else if(c[v][n[v]]>0 && h[v]==h[n[v]]+1)
            push(v,n[v]);
        else n[v]++;
    }
}
int flow(){
    int i,v,lh;
    list *ptr;
    for(i=0;i<=t;i++)h[i]=n[i]=e[i]=0;
    for(i=0;i<=t;i++){
        if(c[0][i]==0)
            continue;
        e[i]=c[0][i];
        c[0][i]-=e[i];
        c[i][0]+=e[i];
    }
    h[0]=t+1;
    makelist();
    for(ptr=head;ptr!=NULL;ptr=ptr->next){
        v=ptr->v;
        lh=h[v];
        discharge(v);
        if(h[v]>lh && head!=ptr)
            tofront(ptr);
    }
    return e[t];
    cleanlist();
}
```

2.5 Min-cost Matching

```
const long long inf=9223372036854775807ll;
const int V=100;
char u[V];
int n,p[V],q[V*V],cap[V][V],cost[V][V];
long long d[V+1];
long long sp(){
    int i,j,k;
    for(i=0;i<n;i++)d[i]=inf,u[i]=1;
    q[0]=d[0]=u[0]=0;
    for(i=0,j=1;i<j;i++){
        u[q[i]]=1;
        for(k=0;k<n;k++){
            if(cap[q[i]][k]==2 && d[q[i]]-cost[k][q[i]]<d[k]){
                d[k]=d[p[k]=q[i]]-cost[k][q[i]];
                if(u[k])u[q[j++]=k]=0;
            }
            if(cap[q[i]][k]==1 && d[q[i]]+cost[q[i]][k]<d[k]){
                d[k]=d[p[k]=q[i]]+cost[q[i]][k];
                if(u[k])u[q[j++]=k]=0;
            }
        }
    }
    if(d[n-1]<inf)return d[n-1];
    return -1;
}
long long flow(int m){
    long long tmp,sum=0;
    while(m>0 && (tmp=sp())>=0){
        sum+=tmp;
        m--;
        for(int i=n-1;i>0;i=p[i])
            cap[p[i]][i]--,cap[i][p[i]]++;
    }
    return sum;
}
```

3 Search

3.1 Sticks

```
int y,n,m,min,max,use[51],num[51],able[10000];
void dfs(int len,int d,int i){
   if(y)return;
   if(!d){
       y=1;
       return;
   }
   if(!len){
       while(!use[i])i--;
       use[i]--;
       if(i==m)
           dfs(0,d-1,max);
       else if(i+min<=m)
           dfs(i,d,i);
       use[i]++;
       return;
   }
   for(;i>0 && !y;i--)
       if(use[i] && i+len<=m){
           use[i]--;
           if(len+i==m){
               dfs(0,d-1,max);
               use[i]++;
               return;
           }
           else if(i+min<=m)
               dfs(len+i,d,i);
           use[i]++;
       }
}
main(){
int i,k,sum,stick;
while(scanf("%d",&n) && n){
   for(min=51,max=i=sum=0;i<n;i++){
       scanf("%d",&stick);
       sum+=stick;
       num[stick]++;
       max>?=stick;
       min<?=stick;
   }
   for(y=0,i=max;!y && i<=50;i++)
       if(sum%i==0){
           m=i;
           for(k=1;k<=50;k++)use[k]=num[k];
           dfs(0,sum/i,max);
       }
   printf("%d\n",y?m:sum);
}
}
```

3.2 15 puzzle(IDA*)

```
#define vaild(x,y) (x>=0 && x<n && y>=0 && y<n)
const int n=4,dx[]={1,0,-1,0},dy[]={0,1,0,-1},rev[]={2,3,0,1};
char path[1000],dirt[]="DRUL";
int x,y,step,score,next,bound,s[n][n],num[n*n][n][n];
bool dfs(){
    if(step+score>bound){
        if(step+score<next)
            next=step+score;
        return false;
    }
    if(!score){
        return true;
        path[step]=0;
    }
    for(int i=0;i<4;i++)
        if(path[step-1]!=dirt[rev[i]] && vaild(x+dx[i],y+dy[i]))        {
            score-=num[s[x][y]][x][y];
            s[x+dx[i]][y+dy[i]]=s[x][y];
            s[x+=dx[i]][y+=dy[i]]=0;
            score+=num[s[x][y]][x][y];
            path[step++]=dirt[i];
            if(dfs())
                return true;
            step--;
            score-=num[s[x][y]][x][y];
            s[x-dx[i]][y-dy[i]]=s[x][y];
            s[x-=dx[i]][y-=dy[i]]=0;
            score+=num[s[x][y]][x][y];
        }
    return false;
}
main(){
    int i,j,k,l,t;
    scanf("%d",&t);
    for(k=n*n-2;k>=0;k--)
        for(i=0;i<n;i++)
            for(j=0;j<n;j++)
                num[k+1][i][j]=abs(k/n-i)+abs(k%n-j);
    while(t--){
        for(i=l=score=0;i<n;i++)
            for(j=0;j<n;j++){
                scanf("%d",&s[i][j]);
                if(s[i][j]==0){
                    x=i,y=j;
                    continue;
                }
                score+=num[s[i][j]][i][j];
                for(k=i*n+j;k>=0;k--)
                    if(s[k/n][k%n]>s[i][j])l++;
            }
        if((l+x+1)&1){
            puts("This puzzle is not solvable.");
            continue;
        }
        step=1,bound=score;
        while(1){
            next=1000;
            if(dfs())break;
            bound=next;
        }
        puts(path);
    }
}
```

4 Mathematics

4.1 Factorization and Primality Test

```
#include<cmath>
#include<cstdio>
#include<algorithm>
using namespace std;
long long Rand(){
    return rand()*(1ll<<48)+rand()*(1ll<<32)+rand()*(1ll<<16)+rand();
}
```

```
long long mul(long long a,long long b,long long m){
    long long i,res=0;
    for(i=1;i<=b;i*=2,(a*=2)%=m)
        if(b&i)(res+=a)%=m;
    return res;
}
long long pow(long long n,long long k,long long m){
    if(k==0)return 1;
    if(k%2==1)
        return mul(n,pow(n,k-1,m),m);
    n=pow(n,k/2,m);
    return mul(n,n,m);
}
bool witness(long long a,long long n){
    long long x,y,u,t;
    for(u=n-1,t=0;u%2==0;u/=2,t++);
    x=pow(a,u,n);
    while(t--){
        y=x;
        x=pow(x,2,n);
        if(x==1 && y!=1 && y!=n-1)
            return 1;
    }
    return x!=1;
}
bool mr(long long n,int s=25){
    if(n-1>=2 && witness(2,n))return 0;
    if(n-1>=3 && witness(3,n))return 0;
    if(n-1>=7 && witness(7,n))return 0;
    if(n-1>=61 && witness(61,n))return 0;
    if(n-1>=24251 && witness(24251,n))return 0;
    if(n==46856248255981ll)return 0;
    return 1;
}
long long gcd(long long a,long long b){
    while((a%=b)&&(b%=a));
    return a+b;
}
namespace g{long long abs(long long x){return x<0?-x:x;}}
int _c=1;
long long _f(long long x,long long n)
{return(mul(x,x,n)+_c)%n;}
long long go(long long n){
    long long x,y,d=1;
        x=y=Rand()%n;
        while(d==1){
            x=_f(x,n);
            y=_f(_f(y,n),n);
            d=gcd(g::abs(y-x),n);
        }
        if(d!=n)return d;
    return d;
}
void fa(long long n,int& fn,long long s[]){
    long long x;
    while(n>1 && n%2==0)
        s[fn++]=2,n/=2;
    while(!mr(n)){
        for(_c=1,x=n;x==n;_c=1+Rand()%(n-1))x=go(n);
        if(x<0)break;
        n/=x;
        fa(x,fn,s);
    }
    if(n>1)s[fn++]=n;
}
main(){
    int i,j,m;
    long long n,k,s[70];
    while(scanf("%I64d",&n)==1){
        m=0;
        while(n%2==0)n>>=1;
        fa(n,m,s);
        std::sort(s,s+m);
        for(i=0,k=1;i<m;i+=j){
            for(j=0;i+j<m && s[i+j]==s[i];j++);
            k*=j+1;
        }
        printf("%I64d\n",k);
    }
}
```

4.2 Permutation

```
#include<algorithm>
long long f[50];
long long go(int num[],int n,int t){
    int i,j,k;
    long long s1[25],s2[25];
    if(n<=1)return 1;
    for(j=0;j<=t;j++)s1[j]=0;
    for(j=0;j<=num[0] && j<=t;j++)
        s1[j]=f[t]/f[j];
    for(i=1;i<n;i++){
        for(j=0;j<=t;j++)s2[j]=0;
        for(j=0;j<=num[i] && j<=t;j++)
            for(k=0;j+k<=t;k++)
                s2[j+k]+=s1[k]/f[j];
        for(j=0;j<=t;j++)s1[j]=s2[j];
    }
    return s1[t];
}
int count(int s[],int n,int num[]){
    int i,j;
    std::sort(s,s+n);
    for(i=j=0;i<n;j++)
    {
        s[j]=s[i];
        for(num[j]=0;i<n && s[i]==s[j];i++)num[j]++;
    }
    return j;
}
void make(int s1[],int u1[],const int& n1,int s2[],int n2,long long k){
    int i,j;
    long long l;
    for(i=0;i<n2;i++)
        for(j=0;j<n1;j++)
        {
            if(!u1[j])continue;
            s2[i]=s1[j],u1[j]--;
            l=go(u1,n1,n2-i-1);
            if(k<=l)break;
            k-=l,u1[j]++;
        }
}
long long get(int s1[],int u1[],int n1,int s2[],int n2){
    int i,j;
    long long k=0;
    for(i=0;i<n2;i++)
        for(j=0;j<n1;j++)
        {
            u1[j]--;
            if(s2[i]==s1[j])break;
            k+=go(u1,n1,n2-i-1);
            u1[j]++;
        }
    return k;
}
```

4.3 Huge Mod

```
#include<cstdio>
struct sint{
    int n,m,k;
    sint(int a=0):n(a){}
    void operator*=(sint a){
        n*=a.n;
        if(n>k)n=k+(n-k)%m;
    }
    bool operator==(sint a){return n==a.n;}
}s[100];
int n,m[10],u[20000];
int go(int d){
    if(d>=n)return 1;
    int i,j;
    sint x=s[d];
    x.n=1,x*=s[d];
    for(j=0;j<=x.k+x.m;j++)u[j]=-1;
    for(i=1;;i++){
        if(u[x.n]>=0)break;
        u[x.n]=i;
        x*=s[d];
    }
    s[d+1].k=u[x.n];
    s[d+1].m=i-u[x.n];
    j=go(d+1);
    for(x.n=1;j--;x*=s[d]);
    return x.n;
}
main()
{
    int i,C=1;
    while(scanf("%d %d",&s[0].m,&n)==2){
        s[0].k=0;
        for(i=0;i<n;i++)
            scanf("%d",&s[i].n);
        printf("Case #%d: %d\n",C++,go(0));
    }
}
```

4.4 Recurrence

```
#include<stdio.h>
class martix{
public:
    int n,m,s[20][20];
    void clean(int d){
        int i,j;
        for(i=0;i<d;i++)
            for(j=0;j<d;j++)
                s[i][j]=0;
    }
    void operator*=(martix b){
        martix a=*this;
        int i,j,k;
        for(i=0;i<n;i++)
            for(j=0;j<n;j++){
                s[i][j]=0;
                for(k=0;k<n;k++){
                    s[i][j]+=(a.s[i][k]*b.s[k][j])%m;
                    s[i][j]%=m;
                }
            }
    }
    void print(){
        int i,j;
        for(i=0;i<n;i++){
            printf("%d",s[i][0]);
            for(j=1;j<n;j++)
                printf(" %d",s[i][j]);
            puts("");
        }
    }
}s,r;
main(){
    int i,j,d,n,m,a[20],f[20];
    while(scanf("%d %d %d",&d,&n,&m) && d+n+m){
        for(i=0;i<d;i++){
            scanf("%d",&a[i]);
            a[i]%=m;
        }
        for(i=0;i<d;i++){
            scanf("%d",&f[i]);
            f[i]%=m;
        }
        if(n<=d){
            printf("%d\n",f[n-1]);
            continue;
        }
        s.clean(d),r.clean(d);
        s.m=r.m=m;
        s.n=r.n=d;
        for(i=0;i<d;i++){
            s.s[i][i]=1;
            if(i)r.s[i-1][i]=1;
            r.s[d-1][d-i-1]=a[i];
        }
        for(i=1,n-=d;i<=n && i>0;i<<=1,r*=r)
            if(i&n)s*=r;
        for(i=n=0;i<d;i++)
            n=(n+s.s[d-1][i]*f[i])%m;
        printf("%d\n",n);
    }
}
```

5 Strings

5.1 Knuth-Morris-Pratt algorithm

```
#include<cstdio>
main(){
    int i,j,p[1001]={0,0};
    char s1[1001],s2[1001];
    while(scanf("%s %s",s1,s2)>0){
        for(i=0;s1[i];i++);
        s1[i++]='$',s1[i]=0;
        for(i=1,j=0;s1[i];i++){
            while(j && s1[i]!=s1[j])j=p[j];
            if(s1[i]==s1[j])j++;
            p[i+1]=j;
        }
        for(i=j=0;s2[i];i++){
            while(j && s1[j]!=s2[i])j=p[j];
            if(s1[j]==s2[i])j++;
        }
        printf("%d\n",j);
    }
}
```

5.2 Extended KMP

```
#include<cstdio>
#include<cstring>
struct node{
    node *sub,*c[26];
    int b;
}*root,*ptr1[100],*ptr2[100];
int to[256],now[1001],next[101],c[101][101];
char s[1000][1001],p[100][101];
void clear(node **ptr){
    if(*ptr){
        for(int i=0;i<26;i++)
            clear(&(*ptr)->c[i]);
        delete *ptr;
        *ptr=NULL;
    }
}
int build(node *&ptr,char *a,int b){
    if(!ptr){
        ptr=new node;
        for(int i=0;i<26;i++)
            ptr->c[i]=NULL;
        ptr->b=-1;
    }
    if(*a) return build(ptr->c[to[*a]],a+1,b);
    if(ptr->b>=0) return ptr->b;
    return(ptr->b=b);
}
main(){
    node *ptr;
    int i,j,n,m,t,sum;
    for(i='a';i<='z';i++)
        to[i]=i-'a';
    scanf("%d",&t);
    while(t--){
        clear(&root);
        scanf("%d %d",&n,&m);
        for(i=0;i<n;i++)
            scanf("%s",s[i]);
        scanf("%d %d",&m,&j);
        for(i=0;i<m;i++){
            next[i]=0;
            scanf("%s",p[i]);
            for(j=0;j<=i;j++)
                c[i][j]=c[j][i]=!strcmp(p[i],p[j]);
            j=build(root,p[i],i);
        }
        for(i=1,j=0;i<m;i++){
            while(j>0 && !c[i][j])
                j=next[j];
            j+=c[i][j];
            next[i+1]=j;
        }
        for(i=0;i<m;i++){
            ptr1[i]=root->c[to[p[i][0]]];
            ptr2[i]=root;
            ptr1[i]->sub=ptr2[i];
        }
        for(j=1;p[0][j];j++)
            for(i=0;i<m;i++){
                ptr1[i]=ptr1[i]->c[to[p[i][j]]];
                while(ptr2[i]!=root && !ptr2[i]->c[to[p[i][j]]])
                    ptr2[i]=ptr2[i]->sub;
                if(ptr2[i]->c[to[p[i][j]]])
                    ptr2[i]=ptr2[i]->c[to[p[i][j]]];
                ptr1[i]->sub=ptr2[i];
                if(ptr2[i]->b>0)
                    ptr1[i]->b=ptr2[i]->b;
            }
        for(j=0;s[0][j];j++)
            now[j]=0;
        for(i=sum=0;i<n;i++)
            for(j=0,ptr=root;s[i][j];j++){
                while(ptr!=root && !ptr->c[to[s[i][j]]])
                    ptr=ptr->sub;
                if(ptr->c[to[s[i][j]]])
                    ptr=ptr->c[to[s[i][j]]];
                if(ptr->b<0){
                    now[j]=0;
                    continue;
                }
                while(now[j]>0 && !c[ptr->b][now[j]])
                    now[j]=next[now[j]];
                if(c[ptr->b][now[j]])
                    now[j]++;
                if(now[j]==m)
                    sum++;
            }
        printf("%d\n",sum);
    }
}
```

5.3 Suffix Array

```
#include<cstdio>
const int maxn=200000;
inline bool leq(int a1,int a2,int b1,int b2) // lexicographic order
{return(a1<b1||a1==b1 && a2<=b2);} // for pairs
inline bool leq(int a1,int a2,int a3,int b1,int b2,int b3)
{return(a1<b1 || a1==b1 && leq(a2,a3, b2,b3));} // and triples
// stably sort a[0..n-1] to b[0..n-1] with keys in 0..K from r
int c[maxn+50];
```

```
static void radixPass(int a[],int b[],int r[],int n,int K)
{// count occurrences
    for(int i=0;i<=K;i++)c[i]=0; // reset counters
    for(int i=0;i<n;i++)c[r[a[i]]]++; // count occurrences
    for(int i=0,sum=0;i<=K;i++) // exclusive prefix sums
    {int t=c[i];c[i]=sum;sum+=t;}
    for(int i=0;i<n;i++)b[c[r[a[i]]]++]=a[i]; // sort
}
// find the suffix array SA of s[0..n-1] in {1..K}^n
// require s[n]=s[n+1]=s[n+2]=0, n>=2
void suffixArray(int s[],int SA[],int n,int K)
{
    int n0=(n+2)/3,n1=(n+1)/3,n2=n/3,n02=n0+n2;
    int *s12 =new int[n02+3];s12[n02]=s12[n02+1]=s12[n02+2]=0;
    int *SA12=new int[n02+3];SA12[n02]=SA12[n02+1]=SA12[n02+2]=0;
    int *s0  =new int[n0];
    int *SA0 =new int[n0];
// generate positions of mod 1 and mod 2 suffixes
// the "+(n0-n1)" adds a dummy mod 1 suffix if n%3 == 1
    for(int i=0,j=0;i<n+(n0-n1);i++)if(i%3!=0)s12[j++]=i;
// lsb radix sort the mod 1 and mod 2 triples
    radixPass(s12,SA12,s+2,n02,K);
    radixPass(SA12,s12,s+1,n02,K);
    radixPass(s12,SA12,s,n02,K);
// find lexicographic names of triples
    int name=0,c0=-1,c1=-1,c2=-1;
    for(int i=0;i<n02;i++){
        if(s[SA12[i]]!=c0 || s[SA12[i]+1]!=c1 || s[SA12[i]+2]!=c2)
           {name++;c0=s[SA12[i]];c1=s[SA12[i]+1];c2=s[SA12[i]+2];}
        if(SA12[i]%3==1)s12[SA12[i]/3]=name; // left half
        else s12[SA12[i]/3+n0]=name;  // right half
    }
// recurse if names are not yet unique
    if(name<n02){
        suffixArray(s12,SA12,n02,name);
// store unique names in s12 using the suffix array
        for(int i=0;i<n02;i++)s12[SA12[i]]=i+1;
    }else // generate the suffix array of s12 directly
        for(int i=0;i<n02;i++)SA12[s12[i]-1]=i;
// stably sort the mod 0 suffixes from SA12 by their first character
    for(int i=0,j=0;i<n02;i++)if(SA12[i]<n0)s0[j++]=3*SA12[i];
    radixPass(s0,SA0,s,n0,K);
// merge sorted SA0 suffixes and sorted SA12 suffixes
    for(int p=0,t=n0-n1,k=0;k<n;k++){
#define GetI() (SA12[t] < n0 ? SA12[t]*3+1: (SA12[t] - n0)*3+2)
        int i=GetI(); // pos of current offset 12 suffix
        int j=SA0[p]; // pos of current offset 0 suffix
        if(SA12[t]<n0? // different compares for mod 1 and mod 2 suffixes
            leq(s[i],s12[SA12[t]+n0],s[j],s12[j/3]):
            leq(s[i],s[i+1],s12[SA12[t]-n0+1],s[j],s[j+1],s12[j/3+n0]))
        {// suffix from SA12 is smaller
            SA[k]=i;t++;
            if(t==n02) // done --- only SA0 suffixes left
            for(k++;p<n0;p++,k++)SA[k]=SA0[p];
        }else{// suffix from SA0 is smaller
            SA[k]=j;p++;
            if(p==n0) // done --- only SA12 suffixes left
            for(k++;t<n02;t++,k++)SA[k]=GetI();
        }
    }
    delete[] s12; delete[] SA12; delete[] SA0; delete[] s0;
}

int n,s[maxn],t[maxn],d[maxn],x[maxn],ml[][maxn],mr[][maxn],ub[65537];
void go(int s[],int t[],int d[],int n){
    int i,j,k;
    for(i=0;i<n;i++)x[t[i]]=i;
    for(i=j=0;i<n;i++){
        if(x[i]==n-1)
        {j=d[x[i]]=0;continue;}
        k=t[x[i]+1];
        while(s[i+j]==s[k+j])
            j++;
        d[x[i]]=j;
        j-=(j>0);
    }
}
void prermq(int z[],int ml[][maxn],int mr[][maxn],int n){
    int i,j,k,p;
    for(i=1,p=0;i<=n;i*=2,p++){
        for(j=0;j<n;j+=i){
            ml[p][j]=z[j];
            for(k=j+1;k<j+i;k++)
                ml[p][k]=(z[k]<?ml[p][k-1]);
            mr[p][j+i-1]=z[j+i-1];
            for(k=j+i-2;k>=j;k--)
                mr[p][k]=(z[k]<?mr[p][k+1]);
        }
    }
}
char tmp[maxn];
int lcp(int i,int j){
    if(i>=n || j>=n || i<0 || j<0){
        printf("%d %d",i,j);
        scanf(" ");
        return 0;
    }
    i=x[i],j=x[j];
    if(i>j)i^=j^=i^=j;
    j--;
    int k=ub[i^j];
    return (mr[k][i])<?(ml[k][j]);
}
int find(int v,int l1,int l2){
    if(!lcp(v-l1,v))return v;
    int i,j,k=0,last=lcp(v-l1,v);
    for(i=0,j=(l2-1)<?(v-l1);i<=j;){
        k=(i+j)/2;
        if(lcp(v-l1-k,v-k)!=last+k)j=k-1;
        else if(v-l1-k-1>=0 && lcp(v-l1-k-1,v-k-1)==last+k+1)i=k+1;
        else break;
    }
    if(lcp(v-l1-k,v-k)<l2)
        return v;
    return v-k;
}
main()
{
    int i,j,k,l,g,max,T,C=1;
    for(i=1,k=0;i<65536;i*=2,k++)
        for(j=i*2-1;j>=i;j--)ub[j]=k;
    while(scanf("%d",&n)==1)
    {
        scanf("%d%c",&g,tmp);
        gets(tmp);
        for(n=0;tmp[n];n++)
            s[n]=tmp[n]-'a'+1;
        s[n]=s[n+1]=s[n+2]=0;
        suffixArray(s,t,n,26);
        go(s,t,d,n);
        for(i=j=0;i<n;i++)
             j>?=d[i];
        printf("%d\n",j);
prermq(d,ml,mr,n-1);
        for(i=max=0;i<n;i++)max>?=d[i];
        for(i=1+g,k=0;max+g>=i && i<n;i++)
            for(j=i;j<n;j+=i-g){
                j=find(j,i,i-g);
                l=lcp(j-i,j);
                if(l+g>=i)k+=l+g-i+1;
                if(l>0)j+=l-1;
            }
        printf("Case %d: %d\n",C++,k);*/
    }
}
```

6 Misc.

6.1 Date

```
int tonum(int y,int m,int d){
    static int day[]={0,31,28,31,30,31,30,31,31,30,31,30,31};
    y--;
    int num=y*365+y/4-y/100+y/400;
    y++;
    if((y%4==0 && y%100)|| y%400==0)day[2]=29;
    else day[2]=28;
    for(int i=1;i<m;i++)
        num+=day[i];
    return num+=d;
}
void todate(int num,int &y,int &m,int &d){
    static int y400=146097,y100=36524,y4=1461,
        day[]={0,31,28,31,30,31,30,31,31,30,31,30,31};;
    num--;
    y=num/y400*400;
    num%=y400;
    y+=num/y100*100;
    num%=y100;
    y+=num/y4*4;
    num%=y4;
    y+=num/365;
    num%=365;
    for(m=1;num>=day[m];num-=day[m++]);
    d=++num;
}
```