# Houzhui

6194

#include<cstdio>

#include<cstring>

#include<algorithm>

#include<iostream>

#include<string>

using namespace std;

const int M = 2e+5+123;

int maxlen;

char s[M];

int sum[M];

int ranks[M],trank[M];

int sa[M],tsa[M];

int height[M];

int slen;

void sorting(int j)//基数排序

{

for(int i=0;i<=maxlen;i++) sum[i]=0;

for (int i=1; i<=slen; i++) sum[ ranks[i+j] ]++;

for (int i=1; i<=maxlen; i++) sum[i]+=sum[i-1];

for (int i=slen; i>0; i--)

tsa[ sum[ ranks[i+j] ]-- ]=i;

//对第二关键字计数排序，tsa代替sa为排名为i的后缀是tsa[i]

for(int i=0;i<=maxlen;i++) sum[i]=0;

for (int i=1; i<=slen; i++) sum[ ranks[i] ]++;

for (int i=1; i<=maxlen; i++) sum[i]+=sum[i-1];

for (int i=slen; i>0; i--)

sa[ sum[ ranks[ tsa[i] ] ]-- ]= tsa[i];

//对第一关键字计数排序,构造互逆关系

}

void get\_sa()

{

int p;

for (int i=0; i<slen; i++) trank[i+1]=s[i];

for (int i=1; i<=slen; i++) sum[ trank[i] ]++;

for (int i=1; i<=maxlen; i++) sum[i]+=sum[i-1];

for (int i=slen; i>0; i--)

sa[ sum[ trank[i] ]-- ]=i;

ranks[ sa[1] ]=1;

for (int i=2,p=1; i<=slen; i++)

{

if (trank[ sa[i] ]!=trank[ sa[i-1] ]) p++;

ranks[ sa[i] ]=p;

}//第一次的sa与ranks构造完成

for (int j=1; j<=slen; j\*=2)

{

sorting(j);

trank[ sa[1] ]=1; p=1; //用trank代替ranks

for (int i=2; i<=slen; i++)

{

if ((ranks[ sa[i] ]!=ranks[ sa[i-1] ]) || (ranks[ sa[i]+j ]!=ranks[ sa[i-1]+j ])) p++;

trank[ sa[i] ]=p;//空间要开大一点，至少2倍

}

for (int i=1; i<=slen; i++) ranks[i]=trank[i];

}

}

void get\_height(){

for (int i=1,j=0; i<=slen; i++)//用j代替上面的h数组

{

if (ranks[i]==1) continue;

for (; s[i+j-1]==s[ sa[ ranks[i]-1 ]+j-1 ]; ) j++;//注意越界之类的问题

height[ ranks[i] ]=j;

if (j>0) j--;

}

}

int tree[M\*4];

int insert(int num,int l,int r,int pos,int val){

if(l==r) {

tree[num]=val;

return 0;

}

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

if(pos<=mid) insert(num<<1,l,mid,pos,val);

else insert((num<<1)|1,mid+1,r,pos,val);

tree[num]=min(tree[num<<1],tree[(num<<1)|1]);

return 0;

}

int ans;

int query(int num,int l,int r,int left,int right){

if(l>=left&&r<=right){

ans=min(ans,tree[num]);

return 0;

}

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

if(left<=mid) query(num<<1,l,mid,left,right);

if(right>mid) query((num<<1)|1,mid+1,r,left,right);

return 0;

}

int clear(){

for(int i=0;i<=maxlen;i++) ranks[i]=0;

for(int i=0;i<=maxlen;i++) trank[i]=0;

for(int i=0;i<=maxlen;i++) sum[i]=0;

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

for(int i=0;i<maxlen;i++) height[i]=0;

return 0;

}

/\*

int prin(int j){

for(int i=j;i<=slen;i++){

printf("%c",s[i-1]);

}

printf("\n");

return 0;

}

int check(){

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

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

for(int j=sa[i];j<=slen;j++)

printf("%c",s[j-1]);

printf("\n");

}

return 0;

}

\*/

int near[M];

int main(){

//freopen("A.in","r",stdin);

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

int T;

scanf("%d",&T);

while(T--){

int K;

scanf("%d",&K);

scanf("%s",s);

slen=strlen(s);

maxlen=slen\*2+1;

get\_sa();

get\_height();

int n=slen;

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

insert(1,1,n,i,height[i]);

}

height[n+1]=0;

long long sum1=0;

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

ans=n-sa[i]+1;

if(K>=2)query(1,1,n,i-K+2,i);

sum1=sum1+max(0,ans-height[i+1]);

}

K++;

long long sum2=0;

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

ans=n-sa[i]+1;

if(K>=2)query(1,1,n,i-K+2,i);

sum2=sum2+max(0,ans-height[i+1]);

}

printf("%lld\n",sum1-sum2);

clear();

}

return 0;

}