字符串：

AC自动机：

题意：用n个字符串中最长的那个，来匹配剩余的所有字符串。若都能成功匹配，则输出最长的串，否则输出NO。

#include <iostream>

#include <cstdio>

#include <algorithm>

#include <cstring>

#include <string>

#include <queue>

#include <vector>

using namespace std;

const int N = 100005;

const int M = 26;

queue<int> q;

vector<string> vec;

bool vis[N];

struct Trie {

int trieN;

int ch[N][M], val[N], fail[N];

void init() {

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

trieN = -1;

newnode();

}

int newnode() {

memset(ch[++trieN], 0, sizeof(ch[0]));

val[trieN] = fail[trieN] = 0;

return trieN;

}

void insert(const string &str, int index) {

int cur = 0;

for (int i = 0;str[i];i++) {

int d = str[i] - 'a';

if (!ch[cur][d])

ch[cur][d] = newnode();

cur = ch[cur][d];

}

if (val[cur]) vis[index] = 1;

else val[cur] = index;

}

void build() {

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

if (ch[0][i])

q.push(ch[0][i]);

}

while (!q.empty()) {

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

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

int &next = ch[cur][i];

if (next) {

fail[next] = ch[fail[cur]][i];

q.push(next);

}

else next = ch[fail[cur]][i];

}

}

}

void query(const string &str) {

int cur = 0, tmp;

for (int i = 0;str[i];i++) {

int d = str[i] - 'a';

tmp = cur = ch[cur][d];

while (tmp && ~val[tmp]) {

if (val[tmp] != -1) vis[val[tmp]] = 1;

val[tmp] = -1;

tmp = fail[tmp];

}

}

}

}ac;

int main(){

ios::sync\_with\_stdio(false);

cin.tie(0);

int t;

cin >> t;

while(t--){

ac.init();

int n;

cin >> n;

vec.resize(n+1);

int maxlen = 0,maxi;

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

cin >> vec[i];

if (vec[i].size() > maxlen){

maxlen = vec[i].size();

maxi = i;

}

}

vis[maxi] = 1;

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

if (i != maxi)

ac.insert(vec[i],i);

}

ac.build();

ac.query(vec[maxi]);

bool flag = 1;

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

if (!vis[i]) flag = 0;

}

if (flag) cout << vec[maxi] << '\n';

else cout << "No\n";

}

return 0;

}

**KMP:**

#include<stdio.h>

#include<string.h>

const int N=1e6+5;

int lens,lent,cnt;

int nex[N];

char S[N],T[N];

void get\_next(){

int first=-1,last=0;

nex[last]=first;

while(last<lent){

if(first==-1 || T[first]==T[last]){

first++,last++;

if(T[last]==T[first]) nex[last]=nex[first];

else nex[last]=first;

}

else

first=nex[first];

}

}

void KMP(){

int first=0,last=0;

bool find=false;

while(last<lens){

if(first==-1 || S[last]==T[first]){

last++,first++;

if(first==lent){

cnt++; //匹配成功的次数

// printf("%d\n",last-lent);find=true; 统计第一次匹配成功的位置

// break;

}

}

else

first=nex[first];

}

//if(!find) 匹配不成功 输出－1

// printf("-1\n");

}

int main(){

int t;

scanf("%d",&t);

while(t--)

{

scanf(“%s%s",T,S);//T在S中第一次出现的位置和次数。

lent=strlen(T);

lens=strlen(S);

cnt=0;

get\_next();

KMP();

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

}

}

最长公共子列：

#include<iostream>

#include<cstring>

#include<algorithm>

#include<string>

#include<vector>

#include<cmath>

#include<cstdio>

#define N 500000

using namespace std;

int rk[N+50], sa[N+50],lcp[N+50];

int tmp[N + 50];

int k,n;

bool cmp\_sa(int i, int j)

{

if (rk[i] != rk[j]) return rk[i] < rk[j];

else {

int ri = i + k <= n ? rk[i + k] : -1;

int rj = j + k <= n ? rk[j + k] : -1;

return ri < rj;

}

}

void construct\_sa(char \*s, int \*sa)

{

n = strlen(s);

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

sa[i] = i;

rk[i] = i < n ? s[i] : -1;

}

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

{

sort(sa, sa + n + 1, cmp\_sa);

tmp[sa[0]] = 0;

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

{

tmp[sa[i]] = tmp[sa[i - 1]] + (cmp\_sa(sa[i - 1], sa[i]) ? 1 : 0);

}

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

rk[i] = tmp[i];

}

}

}

void construct\_lcp(char \*s, int \*sa, int \*lcp)

{

n = strlen(s);

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

int h = 0;

lcp[0] = 0;

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

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

for (h ? h-- : 0; j + h < n&&i + h < n&&s[j + h] == s[i + h]; h++);

lcp[rk[i] - 1] = h;

}

}

char str1[N + 50], str2[N + 50];

int main()

{

int T; cin >> T; getchar();

while (T--)

{

gets(str1);

gets(str2);

int len = strlen(str1);

str1[len] = '#';

str1[len + 1] = '\0';

strcat(str1, str2);

construct\_sa(str1 ,sa);

construct\_lcp(str1,sa,lcp);

int ans = 0;

for (int i = 0; i < strlen(str1); i++)

{

if ((sa[i] < len) != (sa[i + 1]) < len){

ans = max(ans, lcp[i]);

}

}

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

}

return 0;

}

最长回文字串：

#include <cstdio>

#include <cstring>

#include <algorithm>

using namespace std;

const int N = 2e6 + 5;

char s[N];

int p[N];

int n, id, mx,t=0;

void Manacher(){

t++;

id = mx = 0;

int ans = 0;

n = strlen(s);

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

s[2 \* i + 2] = s[i];

s[2 \* i + 1] = '#';

}

s[0] = '@';

p[0] = 1;

for(int i = 1; i < 2 \* n + 2; i ++){

if(mx > i)p[i] = min(p[2 \* id - i], mx - i);

else p[i] = 1;

while(s[i + p[i]] == s[i - p[i]]) p[i] += 1;

if(mx < p[i] + i){

id = i;

mx = p[i] + i;

}

ans = max(ans, p[i] - 1);

}

printf("Case %d: %d\n",t,ans);

}

int main(){

while(scanf("%s", s) == 1&&strcmp(s,"END")) Manacher();

return 0;

}

区间内不同字串的个数：

Hash：

const int HASH = 10007;

const int N = 2010;

struct HASHMAP {

int head[HASH], nxt[N], sz;

unsigned long long sta[N];

int f[N];

void init() {

sz = 0;

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

}

int insert(unsigned long long val, int \_id) {

int h = val % HASH;

for (int i = head[h]; i != -1; i = nxt[i])

if (val == sta[i]) {

int tmp = f[i];

f[i] = \_id;

return tmp;

}

f[sz] = \_id;

sta[sz] = val;

nxt[sz] = head[h];

head[h] = sz++;

return 0;

}

} H;

const int SEED = 11;

unsigned long long P[N];

unsigned long long S[N];

char str[N];

int ans[N][N];

int main() {

P[0] = 1;

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

P[i] = P[i - 1] \* SEED;

int T;

scanf("%d", &T);

while (T--) {

scanf("%s", str);

int n = strlen(str);

S[0] = 0;

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

S[i] = S[i - 1] \* SEED + str[i - 1];

memset(ans, 0, sizeof(ans));

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

H.init();

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

int l = H.insert(S[i + L - 1] - S[i - 1] \* P[L], i);

ans[i][i + L - 1]++;

ans[l][i + L - 1]--;

}

}

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

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

ans[i][j] += ans[i + 1][j] + ans[i][j - 1] - ans[i + 1][j - 1];

int m, u, v;

scanf("%d", &m);

while (m--) {

scanf("%d%d", &u, &v);

printf("%d\n", ans[u][v]);

}

}

return 0;

}

后缀数组：

DC3:

// 待排序的字符串放在 r 数组中, 从 r[0] 到 r[n-1], 长度为 n, 且最大值小于 m

// 约定除 r[n-1] 外所有的 r[i] 都大于 0, r[n-1]=0

// 函数结束后, 结果放在 sa 数组中, 从 sa[0] 到 sa[n-1]

#define maxn 100005

#define F(x)((x)/3+((x)%3==1?0:tb))

#define G(x)((x)<tb?(x)\*3+1:((x)-tb)\*3+2)

int wa[maxn], wb[maxn], wv[maxn], wss[maxn]; // 必须这么大

int s[maxn \* 3], sa[maxn \* 3];

int c0(int \*r, int a, int b) { return r[a] == r[b] && r[a + 1] == r[b + 1] && r[a + 2] == r[b + 2]; }

int c12(int k, int \*r, int a, int b) {

if (k == 2) return r[a] < r[b] || r[a] == r[b] && c12(1, r, a + 1, b + 1);

else return r[a] < r[b] || r[a] == r[b] && wv[a + 1] < wv[b + 1];

}

void sort(int \*r, int \*a, int \*b, int n, int m) {

int i;

for (i = 0; i < n; i++) wv[i] = r[a[i]];

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

for (i = 0; i < n; i++) wss[wv[i]]++;

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

for (i = n - 1; i >= 0; i--) b[--wss[wv[i]]] = a[i];

}

void dc3(int \*r, int \*sa, int n, int m) {

int i, j, \*rn = r + n, \*san = sa + n, ta = 0, tb = (n + 1) / 3, tbc = 0, p;

r[n] = r[n + 1] = 0;

for (i = 0; i < n; i++) if (i % 3 != 0) wa[tbc++] = i;

sort(r + 2, wa, wb, tbc, m);

sort(r + 1, wb, wa, tbc, m);

sort(r, wa, wb, tbc, m);

for (p = 1, rn[F(wb[0])] = 0, i = 1; i < tbc; i++)

rn[F(wb[i])] = c0(r, wb[i - 1], wb[i]) ? p - 1 : p++;

if (p < tbc) dc3(rn, san, tbc, p);

else for (i = 0; i < tbc; i++) san[rn[i]] = i;

for (i = 0; i < tbc; i++) if (san[i] < tb) wb[ta++] = san[i] \* 3;

if (n % 3 == 1) wb[ta++] = n - 1;

sort(r, wb, wa, ta, m);

for (i = 0; i < tbc; i++) wv[wb[i] = G(san[i])] = i;

for (i = 0, j = 0, p = 0; i < ta && j < tbc; p++)

sa[p] = c12(wb[j] % 3, r, wa[i], wb[j]) ? wa[i++] : wb[j++];

for (; i < ta; p++) sa[p] = wa[i++];

for (; j < tbc; p++) sa[p] = wb[j++];

}

int main() {

int n, m = 0;

scanf("%d", &n);

for (int i = 0; i < n; i++) scanf("%d", &s[i]), s[i]++, m = max(s[i] + 1, m);

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

s[n++] = 0;

dc3(s, sa, n, m);

for (int i = 0; i < n; i++) printf("%d ", sa[i]);

printf("\n");

}

倍增：

int wa[maxn], wb[maxn], wv[maxn], ws[maxn];

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

void da(int \*r, int \*sa, int n, int m) {

int i, j, p, \*x = wa, \*y = wb, \*t;

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

for (i = 0; i < n; i++) ws[x[i] = r[i]]++;

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

for (i = n - 1; i >= 0; i--) sa[--ws[x[i]]] = i;

for (j = 1, p = 1; p < n; j \*= 2, m = p) {

for (p = 0, 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 < n; i++) wv[i] = x[y[i]];

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

for (i = 0; i < n; i++) ws[wv[i]]++;

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

for (i = n - 1; i >= 0; i--) sa[--ws[wv[i]]] = y[i];

for (t = x, x = y, y = t, p = 1, x[sa[0]] = 0, i = 1; i < n; i++)

x[sa[i]] = cmp(y, sa[i - 1], sa[i], j) ? p - 1 : p++;

}

return;

}

int wa[maxn], wb[maxn], wv[maxn], ws[maxn];

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

void da(int \*r, int \*sa, int n, int m) {

int i, j, p, \*x = wa, \*y = wb, \*t;

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

for (i = 0; i < n; i++) ws[x[i] = r[i]]++;

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

for (i = n - 1; i >= 0; i--) sa[--ws[x[i]]] = i;

for (j = 1, p = 1; p < n; j \*= 2, m = p) {

for (p = 0, 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 < n; i++) wv[i] = x[y[i]];

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

for (i = 0; i < n; i++) ws[wv[i]]++;

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

for (i = n - 1; i >= 0; i--) sa[--ws[wv[i]]] = y[i];

for (t = x, x = y, y = t, p = 1, x[sa[0]] = 0, i = 1; i < n; i++)

x[sa[i]] = cmp(y, sa[i - 1], sa[i], j) ? p - 1 : p++;

}

return;

}

最长不重复字串：

#include<cstdio>

#include<cstring>

#include<algorithm>

using namespace std;

#define N 22222

#define INF (1<<30)

int wa[N],wb[N],wv[N],ws[N];

int cmp(int \*r,int a,int b,int l){

return r[a]==r[b] && r[a+l]==r[b+l];

}

int sa[N],rank[N],height[N];

void SA(int \*r,int n,int m){

int \*x=wa,\*y=wb;

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

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

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

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

int p=1;

for(int j=1; p<n; j<<=1,m=p){

p=0;

for(int i=n-j; i<n; ++i) y[p++]=i;

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

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

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

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

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

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

swap(x,y); x[sa[0]]=0; p=1;

for(int i=1; i<n; ++i) x[sa[i]]=cmp(y,sa[i-1],sa[i],j)?p-1:p++;

}

for(int i=1; i<n; ++i) rank[sa[i]]=i;

int k=0;

for(int i=0; i<n-1; height[rank[i++]]=k){

if(k) --k;

for(int j=sa[rank[i]-1]; r[i+k]==r[j+k]; ++k);

}

}

int n,a[N],r[N];

bool isok(int k){

bool flag=0;

int mx=-INF,mm=INF;

for(int i=2; i<=n; ++i){

if(height[i]>=k){

mm=min(mm,min(sa[i],sa[i-1]));

mx=max(mx,max(sa[i],sa[i-1]));

if(mx-mm>k) return 1;

}else{

mx=-INF,mm=INF;

}

}

return 0;

}

int main(){

while(~scanf("%d",&n) && n){

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

--n;

for(int i=0; i<n; ++i) r[i]=a[i+1]-a[i]+88;

r[n]=0;

SA(r,n+1,176);

int l=0,r=n>>1;

while(l<r){

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

if(isok(mid)) l=mid;

else r=mid-1;

}

if(l>=4) printf("%d\n",l+1);

else printf("%d\n",0);

}

return 0;

}

最长重复字串（不重叠）：

#include <iostream>

#include <cstdio>

#include <cstring>

#include <cstdlib>

#include <algorithm>

#define N 50050

using namespace std;

int r[N],wa[N],wc[N],wv[N],sa[N],wb[N],height[N],rank[N];

int n;

inline bool cmp(int \*r,int a,int b,int l)

{

return r[a]==r[b]&&r[a+l]==r[b+l];

}

inline void da(int \*r,int \*sa,int n,int m)

{

int i,j,p,\*x=wa,\*y=wb,\*t;

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

for(i=0;i<n;i++) wc[x[i]=r[i]]++;

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

for(i=n-1;i>=0;i--) sa[--wc[x[i]]]=i;

for(j=1,p=1;p<n;j<<=1,m=p)

{

for(i=n-j,p=0;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<n;i++) wv[i]=x[y[i]];

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

for(i=0;i<n;i++) wc[wv[i]]++;

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

for(i=n-1;i>=0;i--) sa[--wc[wv[i]]]=y[i];

for(t=x,x=y,y=t,p=1,x[sa[0]]=0,i=1;i<n;i++)

x[sa[i]]=cmp(y,sa[i-1],sa[i],j)?p-1:p++;

}

}

inline void getheight(int \*r,int \*sa,int n)

{

int i,j,k=0;

for(i=1;i<=n;i++) rank[sa[i]]=i;

for(i=0;i<n;height[rank[i++]]=k)

for(k?k--:0,j=sa[rank[i]-1];r[i+k]==r[j+k];k++);

}

inline bool check(int x)

{

int mx=sa[1],mn=sa[1];

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

{

if(height[i]<x) mx=mn=sa[i];

else

{

mx=max(mx,sa[i]);

mn=min(mn,sa[i]);

if(mx-mn>=x) return true;

}

}

return false;

}

inline void getans()

{

int l=4,r=(n>>1)+1,mid,res;

while(l<=r)

{

mid=(l+r)>>1;

if(check(mid)) res=mid,l=mid+1;

else r=mid-1;

}

if(res<4) puts("0");

else printf("%d\n",res+1);

}

inline void go()

{

n--;

for(int i=0;i<=n;i++) scanf("%d",&r[i]);

if(n<10) {puts("0");return;}

for(int i=0;i<n;i++) r[i]=r[i]-r[i+1]+90;

r[n]=0;

da(r,sa,n+1,200);

getheight(r,sa,n);

getans();

}

int main()

{

while(scanf("%d",&n),n) go();

return 0;

}

字典树：

//前缀出现次数

#include"bits/stdc++.h"

using namespace std;

int trie[400001][26],len,root,tot,sum[400001];

bool p;

int n,m;

char s[11];

void insert()

{

len=strlen(s);

root=0;

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

{

int id=s[i]-'a';

if(!trie[root][id]) trie[root][id]=++tot;

sum[trie[root][id]]++;//前缀后移一个位置保存

root=trie[root][id];

}

}

int search()

{

root=0;

len=strlen(s);

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

{

int id=s[i]-'a';

if(!trie[root][id]) return 0;

root=trie[root][id];

}//root经过此循环后变成前缀最后一个字母所在位置的后一个位置

return sum[root];//因为前缀后移了一个保存，所以此时的sum[root]就是要求的前缀出现的次数

}

int main()

{

bool ok=0;

while(gets(s)!=NULL)

{

n=strlen(s);

if(!n) {ok=1;continue;}

if(!ok) insert();

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

}

return 0;

}

字典树：

//查询单词或前缀是否出现过

/\*

trie[rt][x]=tot:rt是上个节点编号，x是字母，tot是下个节点编号

\*/

#include"bits/stdc++.h"

#define maxn 2000010

using namespace std;

int tot=1,n;

int trie[maxn][26];

//bool isw[maxn];查询整个单词用

void insert(char \*s,int rt)

{

for(int i=0;s[i];i++)

{

int x=s[i]-'a';

if(trie[rt][x]==0)//现在插入的字母在之前同一节点处未出现过

{

trie[rt][x]=++tot;//字母插入一个新的位置，否则不做处理

}

rt=trie[rt][x];//为下个字母的插入做准备

}

/\*isw[rt]=true;标志该单词末位字母的尾结点，在查询整个单词时用到\*/

}

bool find(char \*s,int rt)//前缀是否出现过

{

for(int i=0;s[i];i++)

{

int x=s[i]-'a';

if(trie[rt][x]==0)return false;//以rt为头结点的x字母不存在，返回0

rt=trie[rt][x];//为查询下个字母做准备

}

return true;

//查询整个单词时，应该return isw[rt]

}

char s[22];

int main()

{

tot=0;

int rt=1;

scanf("%d",&n);

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

{

cin>>s;

insert(s,rt);

}

scanf("%d",&n);

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

{

cin>>s;

if(find(s,rt))printf("YES\n");

else printf("NO\n");

}

return 0;

}

01字典树：

int tol; //节点个数

ll val[32\*N]; //点的值

int num[32\*N]; //每个节点被访问的次数

int ch[32\*N][2]; //边的值

void init()

{ //初始化

tol=1;

ch[0][0]=ch[0][1]=0;

}

void insert(ll x)

{ //往 01字典树中插入 x

int u=0;

for(int i=32;i>=0;i--)

{

int v=(x>>i)&1;

if(!ch[u][v])

{ //如果节点未被访问过

ch[tol][0]=ch[tol][1]=0; //将当前节点的边值初始化

val[tol]=0; //节点值为0，表示到此不是一个数

num[tol]=0; //每个节点被访问的次数+1

ch[u][v]=tol++; //边指向的节点编号

}

u=ch[u][v]; //下一节点

num[u]++;

}

val[u]=x; //节点值为 x，即到此是一个数

}

void update(ll x,int add)

{ //更新插入或删除 x后每个节点被访问的次数

int u=0;

for(int i=32;i>=0;i--)

{

int v=(x>>i)&1;

u=ch[u][v];

num[u]+=add;

}

}

ll query(ll x)

{ //查询所有数中和 x异或结果最大的数

int u=0;

for(int i=32;i>=0;i--)

{

int v=(x>>i)&1;

//利用贪心策略，优先寻找和当期位不同的数

if(ch[u][v^1]&&num[ch[u][v^1]]) u=ch[u][v^1];

else u=ch[u][v];

}

return x^val[u]; //返回结果

}

int main()

{

ci(t);

for(int ii=1;ii<=t;ii++){

init();

ci(n);

for(int i=0;i<n;i++) cl(a[i]),insert(a[i]);

ll ans=0;

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

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

update(a[i],-1);

update(a[j],-1);

ans=max(ans,query(a[i]+a[j]));

update(a[i],1);

update(a[j],1);

}

}

pl(ans);

}

return 0;

}