链滴

字符串与哈希题集

作者：DoctorLo
原文链接：https://ld246.com/article/1569642306625
来源网站：链滴

我校OJ：测试地址

图书配套OJ：测试地址

A. Oulipo

Description

给出两个字符串s1,s2（只有大写字母），求s1在s2中出现多少次。例如：s1="ABA",s2="ABAABA",案为2。

解题思路：

KMP算法的模板，也可以用拓展KMP解决。

代码示例：

```
#include<cstring>
#include<cstdio>
const int N = 1e6+10;
int nex[N];        //nex[i] = T[i,n-1]与T[0,n-1]最长公共前缀
char S[N],T[N]; //S为目标串,T为模式串
int extend[N];

void getNex(char* str){
    /*计算nex数组*/
    int len = strlen(str) ,i = 0, j, p0 ;
    nex[0] = len;
    while(i+1 < len && str[i] == str[i+1]) i++;
    nex[1] = i; p0 = 1;
    for(int i = 2;i < len;i++){
        if(nex[i-p0]+i < nex[p0]+p0) nex[i] = nex[i-p0];
        else {
            j = nex[p0]+p0-i;
            if(j < 0) j = 0;
            while(i+j < len && str[j] == str[j+i]) j++;
            nex[i] = j; p0 = i;
        }
    }
}
void exKMP(char* str1,char *str2){
    /*计算str2与str1的所有后缀的最长公共前缀长度,存放在extend数组中*/
    int i = 0,j,p0,l1 = strlen(S), l2 = strlen(T);
    getNex(str2);
    while(i < l1 && i < l2 && str1[i] == str2[i]) i++;
    extend[0] = i;p0 = 0;
    for(int i = 1;i < l1;i++){
        if(nex[i-p0]+i < extend[p0]+p0) extend[i] = nex[i-p0];
        else{
            j = extend[p0]+p0-i;
            if(j < 0) j = 0;
            while(i+j < l1 && j < l2 && str1[j+i] == str2[j]) j++;
            extend[i] = j; p0 = i;
        }
    }
```

```
}
int solve(){
    /*计算并输出答案*/
    int l1 = strlen(S), l2 = strlen(T), ans = 0;
    exKMP(S,T); int len = strlen(T);
    for(int i = 0;i < l1;i++)
        if(extend[i] == len) ans++;
    return ans;
}
int t;
int main(){
    scanf("%d",&t);
    while(t--){
        scanf("%s%s",T,S);
        printf("%d\n",solve());
    }
    return 0;
}
```

B. 图书管理

Description

图书管理是一件十分繁杂的工作，在一个图书馆中每天都会有许多新书加入。为了更方便的管理图书以便于帮助想要借书的客人快速查找他们是否有他们所需要的书），我们需要设计一个图书查找系统该系统需要支持 2 种操作：

1. add(s) 表示新加入一本书名为 s 的图书。

2. find(s) 表示查询是否存在一本书名为 s 的图书。

解题思路：

该题本意是用hash解决，但是我偷懒直接用map了。。。

代码示例：

```
#include<bits/stdc++.h>
using namespace std;
map<string ,int> mp;
string str;
int n;
void add(string s){
    string tmp = "";
    for(int i = 4;s[i];i++) tmp += s[i];
    mp[tmp] = 1;
}
void Find(string s){
    string tmp = "";
    for(int i = 5;s[i];i++) tmp += s[i];
    if(mp.count(tmp)) cout << "yes" << endl;
    else cout << "no" << endl;
}
int main(){
    scanf("%d",&n);
    getchar();
```

```
    for(int i = 1;i <= n;i++){
        getline(cin,str);
        if(str[0] == 'a') add(str);
        else Find(str);
    }
    return 0;
}
```

C. Power Strings

Description

给定若干个长度 ≤10^6的字符串，询问每个字符串最多是由多少个相同的子字符串重复连接而成的
如：ababab 则最多有 3 个 ab 连接而成。

解题思路：

本题是循环节问题，有个结论：若字符串 s 是由某个长度为 x 的子串循环构成，那么必定有s[1,n-x] = s[x, n]。而根据nex数组的定义，nex[n] 就是 s[x, n] 的长度n-x。

因此如果该题有解，则n % (n-nex[n])为0，答案就是n/(n-nex[n])。

代码示例：

```
#include<cstdio>
#include<iostream>
#include<cstring>
using namespace std;
const int N = 1e6+10;
char str[N];
int nex[N];
void getNex(char* str){
    memset(nex,0,sizeof nex);
    for(int i = 2,j = 0;str[i];i++){
        while(j > 0 && str[i] != str[j+1]) j = nex[j];
        if(str[i] == str[j+1]) j++;
        nex[i] = j;
    }
}
void solve(){
    /*计算答案并输出*/
    getNex(str);
    int n = strlen(str)-1;
    if(n%(n-nex[n])) puts("1");
    else printf("%d\n",n/(n-nex[n]));
}
int main(){
    while(scanf("%s",str+1) && str[1] != '.'){
        str[0] = '*';
        solve();
    }
    return 0;
}
```

D. Seek the Name, Seek the Fame

Description

给定若干字符串（这些字符串总长 ≤4×10^5 ），在每个字符串中求出所有既是前缀又是后缀的子串度。例如：ababcababababcabab，既是前缀又是后缀的：ab，abab，ababcabab，ababcababab bcabab。

解题思路：

依然是利用KMP算法中的nex数组求解。首先该串本身是最长的“既是前缀又是后缀”的子串，下一满足条件的子串是s[1,nex[n]]，下下个是s[1, nex[nex[n]]]，... 。

道理很简单，就是想起来有点绕；如果存在一个长度为 x 的子串 son“既是前缀又是后缀”，那么有s 1,x] = s[n-x,n] = son，那么它的长度不还是不超过 nex[n] 吗？所以上述方法可以从大到小遍历所有能的长度（不是所有子串！）。

代码示例：

```
#include<cstdio>
#include<set>
#include<cstring>
using namespace std;
const int N = 4e5+10;
char str[N];
int nex[N];
void getNex(char *s){
    memset(nex,0,sizeof nex);
    int len = strlen(s);
    for(int i = 2,j = 0;i < len;i++){
        while(j > 0 && s[i] != s[j+1]) j = nex[j];
        if(s[i] == s[j+1]) j++;
        nex[i] = j;
    }
}
int ans[N];
void solve(){
    getNex(str); int n = strlen(str)-1;
    int cnt = 0; ans[++cnt] = n;
    while(nex[n]){
        ans[++cnt] = nex[n];
        n = nex[n];
    }
    for(int i = cnt;i > 1;i--) printf("%d ",ans[i]);
    printf("%d\n",ans[1]);
}
int main(){
    //freopen("123.txt","r",stdin);
    while(~scanf("%s",str+1)){
        str[0] = '#';
        solve();
    }
    return 0;
}
```

E. friends

Description

有三个好朋友喜欢在一起玩游戏，A 君写下一个字符串 S，B 君将其复制一遍得到 T，C 君在 T 的任位置（包括首尾）插入一个字符得到 U。现在你得到了 U，请你找出 S。

解题思路：

本题如果用nex或者hash求解，细节有点多，当然细心点应该是能写出来的。我的解题思路很简单，ON)复杂度，主要基于如下基本推论：

1. 如果n为偶数，无解。
2. 如果前n/2+1个字符通过删去一个，可以等同于后n/2个字符，则匹配成功。
3. 如果后n/2+1个字符通过删去一个，可以等同于前n/2个字符，则匹配成功。
4. 如果 2 和 3 都匹配成功，且两个得到的原串不同，则说明解不唯一。

代码示例：

```
#include<cstdio>
#include<cstring>
const int N = 2e6+10;
char str[N],tmp[N],s1[N],s2[N];
int n,nex[N],tot;
char ans1[N],ans2[N];
bool check(char *s1,char *s2){
    tot = 0; bool flag = false;
    int i = 1,j = 1;
    while(i <= n/2 && j <= n/2+1){
        if(s1[i] != s2[j]){
            if(flag) return false;
            flag = true; j++;
            if(s1[i] != s2[j]) return false;
        }
        tmp[tot++] = s1[i];
        i++ , j++;
    }
    tmp[tot] = '\0';
    return true;
}
void solve(){
    if(n%2 == 0){
        puts("NOT POSSIBLE"); return;
    }
    for(int i = 1;i <= n/2;i++) s1[i] = str[i];
    for(int i = 1;i <= n/2+1;i++) s2[i] = str[i+n/2];
    bool flag1,flag2;
    flag1 = check(s1,s2);
    for(int i = 0;i <= n/2;i++) ans1[i] = tmp[i];
    for(int i = 1;i <= n/2+1;i++) s1[i] = str[i];
    for(int i = 1;i <= n/2;i++) s2[i] = str[i+n/2+1];
    flag2 = check(s2,s1);
    for(int i = 0;i <= n/2;i++) ans2[i] = tmp[i];
    if(flag1 && flag2){
        bool f = true;
        for(int i = 1;i <= n/2;i++)
```

```
            if(ans1[i] != ans2[i]) f = false;
        if(!f){
            puts("NOT UNIQUE"); return;
        }
    }
    if(flag1 || flag2){
        if(flag1) puts(ans1);
        else puts(ans2);
        return;
    }
    puts("NOT POSSIBLE");
}
int main(){
    scanf("%d%s",&n,str+1);
    str[0] = '#'; solve();
    return 0;
}
```

F. A Horrible Poem

Description

给出一个由小写英文字母组成的字符串 S，再给出 q 个询问，要求回答 S 某个子串的最短循环节。如字符串 B 是字符串 A 的循环节，那么 A 可以由 B 重复若干次得到。

解题思路：

这个是最短循环节问题，另外本题卡常十分严格。为了快速求解，本题还用到了线性筛、滚动哈希优等策略。

首先是基础的字符串循环节知识。如果字符串 s 是由某个子串循环得到，|s| = n，那么

循环节的长度一定是 len 的约数，包括最短循环节。因此一个策略就是对于 n 的所有约数，挨个判断否为最短循环节的长度，而判断原理如下：

- 如果字符串 s 的最短循环节长度为 x ，那么必然有 s[1, x] = s[n-x, n] 。

我们可以通过滚动哈希技巧来在O(1)时间完成判断。那么现在主要花费的时间在于找寻 n 的约数上，般做法$O(\sqrt n)$，于是这种做法总复杂度为$O(q \sqrt n)$。

但是这题卡常很严重，这样还是会超时。于是就要利用质因数分解定理在$O(log_2^n)$时间内完成约分解，总时间复杂度为$O(q log_2^n)$。

代码示例：

```
#include<cstring>
#include<algorithm>
#include<cstdio>
using namespace std;
const int N = 5e5+10;
const int Q = 2e6+10;
char str[N];
int n,q;
typedef unsigned long long ull;
ull hsh[N],bse[N] , b = 31; //采用无符号长整形,通过自然溢出省去取模
```

```
bool check(int l,int r,int x){
    /*判断x是否为子串s[l,r]的最短循环节长度*/
    ull h1 = hsh[r-x] - hsh[l-1]*bse[r-x+1-l];
    ull h2 = hsh[r] - hsh[l-1+x]*bse[r-x+1-l];
    return h1 == h2;
}
int v[N],primes[N];
void getPri(){    //线性筛
    int cnt = 0;
    for(int i = 2;i <= n;i++){
        if(!v[i]){
            primes[cnt++] = i;
            v[i] = i;
        }
        for(int j = 0;j < cnt;j++){
            if(primes[j] > v[i] || primes[j]*i > N)
                break;
            v[i*primes[j]] = primes[j];
        }
    }
}
void ask(int l,int r){
    /*回答子串s[l,r]的最短循环节长度*/
    int len = r-l+1, ans = len, d = len;
    while(d != 1){
        int tmp = v[d];
        while(d%tmp == 0 && check(l,r,ans/tmp)) d /= tmp,ans /= tmp;
        while(d%tmp == 0) d /= tmp;
    }
    printf("%d\n",ans);
}
void solve(){
    /*预处理出hash数组,v数组*/
    getPri(); bse[0] = 1;
    for(int i = 1;i <= n;i++){
        hsh[i] = hsh[i-1]*b + str[i]-'a';
        bse[i] = bse[i-1]*b;
    }
}
inline int read() {
    int x=0,f=1; char c=getchar();
    while(c<'0'||c>'9') { if(c=='-') f=-1; c=getchar(); }
    while(c>='0'&&c<='9') { x=x*10+c-'0'; c=getchar(); }
    return x*f;
}
int main(){

    n = read();
    scanf("%s",str+1); str[0] = '#';
    q = read();
    solve();
    for(int i = 1,l,r;i <= q;i++){
        l = read(); r = read();
```

```
        ask(l,r);
    }
    return 0;
}
```

G. Beads

Description

Zxl有一次决定制造一条项链，她以非常便宜的价格买了一长条鲜艳的珊瑚珠子，她现在也有一个机，能把这条珠子切成很多块（子串），每块有k（k>0）个珠子，如果这条珠子的长度不是k的倍数，后一块小于k的就不要拉（nc真浪费），保证珠子的长度为正整数。 Zxl喜欢多样的项链，为她应该怎选择数字k来尽可能得到更多的不同的子串感到好奇，子串都是可以反转的，换句话说，子串（1，2，）和（3,2,1）是一样的。写一个程序，为Zxl决定最适合的k从而获得最多不同的子串。例如：这一串子是： (1,1,1,2,2,2,3,3,3,1,2,3,3,1,2,2,1,3,3,2,1)。

k=1的时候，我们得到3个不同的子串：(1),(2),(3)

k=2的时候，我们得到6个不同的子串： (1,1),(1,2),(2,2),(3,3),(3,1),(2,3)

k=3的时候，我们得到5个不同的子串： (1,1,1),(2,2,2),(3,3,3),(1,2,3),(3,1,2)

k=4的时候，我们得到5个不同的子串： (1,1,1,2),(2,2,3,3),(3,1,2,3),(3,1,2,2),(1,3,3,2)

解题思路：

刚开始时间复杂度算错了，n + n/2 + n/3 + n/4 + ... + 1是调和级数，时间复杂度并不高，因此两循环加上 滚动哈希 O(1)的判断是可以很快通过的。

代码示例：

```
#include<bits/stdc++.h>
using namespace std;
const int N = 2e5+10;
typedef unsigned long long ull;
int col[N] , n;
ull hsh[N],hsh2[N],bse[N] , B = 1e9+7;
map<ull,int> vis;
vector<int> ans;
int tc , mx , k;
void solve(){
    bse[0] = 1; int cnt = 0;
    for(int i = 1;i <= n;i++)
        hsh[i] = hsh[i-1]*B+col[i], bse[i] = bse[i-1]*B;
    for(int i = n;i >= 1;i--)
        hsh2[i] = hsh2[i+1]*B+col[i];
    for(int i = 1;i <= n;i++){
        vis.clear(); cnt = 0;
        for(int j = i;j <= n;j += i){
            ull h1 = hsh[j] - hsh[j-i]*bse[i];
            ull h2 = hsh2[j-i+1] - hsh2[j+1]*bse[i];
            if(vis.count(h1*h2) == 0){
                vis[h1*h2] = 1; cnt++;
            }
            //printf("%llu %llu\n",h1,h2);
        }
        //printf("%d %d\n",cnt,mx);
```

```
        if(cnt == mx) ans.push_back(i);
        else if(cnt > mx) {
            ans.clear(); ans.push_back(i); mx = cnt;
        }
    }
    printf("%d %d\n",mx,ans.size());
    for(int i = 0;i < ans.size();i++) printf("%d ",ans[i]);
}
int main(){
    scanf("%d",&n);
    for(int i = 1;i <= n;i++) scanf("%d",col+i);
    solve();
    return 0;
}
```

H. Antisymmetry

Description

对于一个01字符串，如果将这个字符串0和1取反后，再将整个串反过来和原串一样，就称作“反对称字符串。比如00001111和010101就是反对称的，1001就不是。

现在给出一个长度为N的01字符串，求它有多少个子串是反对称的。

解题思路：

01串的题目真是变化多端啊。要想解决本题首先要得出几个推论：

1. “反对称”的串长度一定是偶数。
2. 如果一个子串是“反对称”的，那么它一定关于中轴线，左右01对应（如0101）。
3. 如果一个串是“反对称”的，那么和它共中轴线的子串也是“反对称”的。

然后就发现和回文串的性质有些相似，仅有 2 点不同，就是回文串长度可以为奇，以及回文串是关于轴对称而不是相反。

那么我们依然可以利用Manacher算法，只不过将判等 改为判“反”即可。当然也要用通配符“#”填充，因为所有“反对称” 的串长度必须为偶，所以我们在“#”上进行计算，以此来保证长度为偶。

代码示例：

```
#include<bits/stdc++.h>
using namespace std;
const int N = 5e5+10;
char str[N],s2[2*N];
int n,len[2*N];
bool check(char a,char b){
    if(a == '#' && b == '#') return true;
    if(a == '1' && b == '0') return true;
    if(a == '0' && b == '1') return true;
    return false;
}
void solve(){
    s2[0] = '$',s2[1] = '#';
```

```
    for(int i = 1;i <= n;i++){
        s2[i<<1] = str[i-1];
        s2[(i<<1)+1] = '#';
    }
    s2[n*2+2] = '\0';
    int mx = 0, mid;
    for(int i = 1;i < 2*n+2;i += 2){
        if(i < mx) len[i] = min(len[2*mid-i],mx-i);
        else len[i] = 1;
        while( check(s2[i-len[i]], s2[i+len[i]]) ) len[i]++;
        if(len[i]+i > mx){
            mx = len[i] + i;
            mid = i;
        }
    }
    long long ans = 0;
    for(int i = 1;i < 2*n+2;i++) ans += len[i]>>1;
    printf("%lld\n",ans);
}
int main(){
    //freopen("123.txt","r",stdin);
    scanf("%d",&n);
    scanf("%s",str);
    solve();
    return 0;
}
```

I. 门票(tickets)

Description

已知数列a_n, a_0 = 1, a_{i+1} = (a_i * A + a_i \:mod \:B) \:mod \:C，请问给定A，B和C，该数列第几项第一次出现重复?

解题思路:

map太慢了啊，我测试了读入输出，测试了乘法和取模，就是没有怀疑map，浪费很多时间。

手写hash函数，利用邻接表来消除冲突，如此一来快了很多。

代码示例:

```
#include<cstdio>
#include<cmath>
typedef long long ll;
const int N = 2181271;
ll a,b,c;
double A;
struct Node{
    ll dat;
    Node* nex;
    Node(ll dat):dat(dat){nex = NULL;}
    Node(){ nex = NULL; }
}head[N];
bool check(ll x){
```

```
    int h = x%N;
    //printf("%d %lld\n",h,x);
    Node* p = head[h].nex;
    while(p != NULL){
        //printf("%lld\n",p->dat);
        if(p->dat == x) return true;
        p = p->nex;
    }
    return false;
}
void Add(ll x){
    int h = x%N;
//    printf("%d %lld\n",h,x);
    Node* p = head[h].nex;
    Node *q = &head[h];
    while(p != NULL) q = p,p = p->nex;
    q->nex = new Node(x);
    //printf("%lld\n",p->dat);
}
void solve(){
    ll res = 1; Add(res);
    for(int i = 1;i < 2e6;i++){
        res = (res*a+res%b)%c;
        if(check(res)){
            printf("%d\n",i); return;
        }
        Add(res);
    }
    puts("-1");
    return;
}
int main(){
    //freopen("123.txt","r",stdin);
    scanf("%lld%lld%lld",&a,&b,&c);
    solve();
    return 0;
}
```

J. 收集雪花(snowflakes)

Description

不同的雪花往往有不同的形状。在北方的同学想将雪花收集起来，作为礼物送给在南方的同学们。一有 n 个时刻，给出每个时刻下落雪花的形状，用不同的整数表示不同的形状。在收集的过程中，同学不希望有重复的雪花。你可以从任意 a 时刻开始，在 b 时刻停止。a 到 b 时刻中间的雪花也都将被集。他们希望收集的雪花最多。

解题思路：

这个用hash有点难写，难在重置哈希表。我用的是离散化+双指针，只需要维护左边界即可。如果当雪花第二次出现，那么知道它上一次出现位置是v[p]，所以先用当前区间长度更新答案，再将左边界新为 v[p] + 1。

正确性证明就不写了，很显然。

代码示例：

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 1e6+10;
int a[N], b[N], n, v[N];
int main(){
    //freopen("123.txt","r",stdin);
    scanf("%d",&n);
    for(int i = 1;i <= n;i++) scanf("%d",a+i),b[i] = a[i];
    sort(a+1,a+1+n);
    int tot = unique(a+1,a+1+n)-a-1;
    int ans = 0,l = 1;
    for(int i = 1;i <= n;i++){
        int p = lower_bound(a+1,a+tot+1,b[i])-a;
        if(v[p] >= l){
            ans = max(ans,i-l);
            l = v[p]+1;
        }
        v[p] = i;
        if(i == n) ans = max(ans,i-l+1);
    }
    printf("%d\n",ans);
    return 0;
}
```