Z函数

> O(n)

```
// C++ Version
vector<int> z_function(string s) {
  int n = (int)s.length();
  vector<int> z(n);
  for (int i = 1, l = 0, r = 0; i < n; ++i) {
    if (i <= r && z[i - l] < r - i + 1) {
      z[i] = z[i - l];
    } else {
      z[i] = max(0, r - i + 1);
      while (i + z[i] < n && s[z[i]] == s[i + z[i]]) ++z[i];
    }
    if (i + z[i] - 1 > r) l = i, r = i + z[i] - 1;
  }
  return z;
}
```

> O(n^2)

lcp以i和j开头的最长公共前缀

```
int lcp[n + 1][n + 1];
std::memset(lcp, 0, sizeof(lcp));

for (int i = n - 1; i >= 0; i--) {
  for (int j = n - 1; j >= 0; j--) {
    lcp[i][j] = s[i] == s[j] ? 1 + lcp[i + 1][j + 1] : 0;
  }
}
```

KMP查找

```
#include<iostream>
#include<algorithm>
#include<string>
#define ll long long
using namespace std;
void GetFailure(const string &pat,ll failure[])
{
    ll j=0,k=-1,len=pat.length();
    failure[0]=-1;
    while(j<len-1)
    {
```

```
        if(k==-1||pat[j]==pat[k])
        {
            failure[++j]=++k;
        }
        else
        {
            k=failure[k];
        }
    }
}
ll KMP_find(const string &ob,const string &pat,ll p=0)
//ob为主串，pat为与之匹配的模式串
{
    ll *failure=new ll [pat.length()];
    GetFailure(pat,failure);            //生成Failure数组
    ll oblen=ob.length(),patlen=pat.length(),i=p,j=0;
    while(i<oblen && j<patlen && patlen-j <= oblen-i)
    {
        if(j==-1 || pat[j] == ob[i] )
        {
            i++;j++;
        }
        else
        {
            j=failure[j];
        }
    }
    delete [] failure;
    if(j<patlen)
    {
        return -1;
    }
    else return i-j;
}
int main()
{
    string ob,pat;
    cin>>ob>>pat;
    cout<<KMP_find(ob, pat,0);
}
```

> KMP优化https://www.cnblogs.com/cherryljr/p/6519748.html

优化代码

```
#include<iostream>
#include<algorithm>
#include<string>
```

```
#define ll long long
using namespace std;
void GetFailure(const string &pat,ll failure[])
{
    ll j=0,k=-1,len=pat.length();
    failure[0]=-1;
    while(j<len-1)
    {
        if(k==-1||pat[j]==pat[k])
        {
            j++;k++;
            if(pat[j]!=pat[k])
              failure[j]=k;
            else  failure[j]=failure[k];    //此句为优化内容
        }
        else
        {
            k=failure[k];
        }
    }
}
ll KMP_find(const string &ob,const string &pat,ll p=0)
//ob为主串，pat为与之匹配的模式串
{
    ll *failure=new ll [pat.length()];
    GetFailure(pat,failure);                        //生成Failure数组
    ll oblen=ob.length(),patlen=pat.length(),i=p,j=0;
    while(i<oblen && j<patlen && patlen-j <= oblen-i)
    {
        if(j==-1 || pat[j] == ob[i] )
        {
            i++;j++;
        }
        else
        {
            j=failure[j];
        }
    }
    delete [] failure;
    if(j<patlen)
    {
        return -1;
    }
    else return i-j;
}
int main()
{
    string ob,pat;
    cin>>ob>>pat;
```

```
    cout<<KMP_find(ob, pat,0);
}
```

hash

```
ull bs[maxn];
ull h[maxn];

void init()
{
    bs[0]=1;
    for(int i=1;i<maxn;i++)
        bs[i]=bs[i-1]*base;
}

void build(string &t,ull* h,int len)  //O(n)
{
    h[0]=0;
    for(int i=1;i<len;i++)
    {
        h[i]=h[i-1]*base+(t[i]-'a'+1);
    }
}

ull get(int l,int r,ull *h) //查询h[maxn]中的子串hash值
{
    return h[r]-h[l-1]*bs[r-l+1];
}

ull hashh(string &t,int len)  //O(n)
{
    ull res=0;
    for(int i=1;i<len;i++)
        {
            res=res*base+(t[i]-'a'+1);
        }
    return res;
}
```

例题（回文串相关）

链接

```
#include <iostream>
#include<string>
```

```cpp
using namespace std;
#define ull unsigned long long
const int N = 2e6 + 7;
const int base = 27;
ull h[N], rh[N], bs[N];
string s;
bool isO[N];  //判断0～n的前缀是否是回文串

void init() //生成base数组，即每位的权重
{
    bs[0] = 1;
    for (int i = 1; i < N; i++)
        bs[i] = bs[i - 1] * base;
}

ull get(ull *h, int l, int r) //获取子串的hash值
{
    return (h[r] - h[l - 1] * bs[r - l + 1]);
}

void solve(string &s)
{
    s = ' ' + s;    //会在string前加个空格，这个s由于是引用，因此若之后还需使用s记得删除引用符号
    h[0] = rh[0] = 0;
    int len = s.length() - 1, ans = 0;
    for (int i = 1; i <= len + 1; i++) //求S原串及逆串的hash值
    {
        h[i] = h[i - 1] * base + s[i] - 'a' + 1;
        rh[i] = rh[i - 1] * base + s[len - i + 1] - 'a' + 1;
    }

    for (int i = 1; i <= len; i++)
    {
        ull l = get(h, 1, i);
        ull r = get(rh, len - i + 1, len);

        if (l == r)
        {
            isO[i] = 1;
        }
    }
}

int main()
{
    init();
    cin>>s;
    solve(s);
    int ans=0;
```

```
    int len=s.length();
    for(int i=1;i<len;i++)
    {
        if(isO[i]&&isO[(i+1)>>1])
        {
            ans++;
        }
    }
    cout<<ans;
}
```

ac自动机

```
struct trie{
    struct node{
        int num;
        int next[26];
    }nodes[MAXN];
    int fail[MAXN];
    int cnt;
    void init(){
        memset(nodes,0,sizeof(nodes));
        memset(fail,0,sizeof(fail));
        cnt=0;
    }
    void insert(string& s){
        int p=0,len=s.length();
        for(int i=0;i<len;i++){
            int c=s[i]-'a';
            if(!nodes[p].next[c]){
                nodes[p].next[c]=++cnt;
            }
            p=nodes[p].next[c];
        }
        nodes[p].num++;
    }
    void insert(char* s){
        int p=0,len=strlen(s);
        for(int i=0;i<len;i++){
            int c=s[i]-'a';
            if(!nodes[p].next[c]){
                nodes[p].next[c]=++cnt;
            }
            p=nodes[p].next[c];
        }
        nodes[p].num++;
    }
```

```
    bool find(string& s){
        int p=0,len=s.length();
        for(int i=0;i<len;i++){
            int c=s[i]-'a';
            if(!nodes[p].next[c]){
                return false;
            }
            p=nodes[p].next[c];
        }
        return nodes[p].num;
    }
    bool find(char* s){
        int p=0,len=strlen(s);
        for(int i=0;i<len;i++){
            int c=s[i]-'a';
            if(!nodes[p].next[c]){
                return false;
            }
            p=nodes[p].next[c];
        }
        return nodes[p].num;
    }
    void build(){
        queue<int> que;
        for(int i=0;i<26;i++){
            if(nodes[0].next[i]){
                que.push(nodes[0].next[i]);
            }
        }
        while(!que.empty()){
            int temp=que.front();
            que.pop();
            for(int i=0;i<26;i++){
                if(nodes[temp].next[i]){
                    fail[nodes[temp].next[i]]=nodes[fail[temp]].next[i];
                    que.push(nodes[temp].next[i]);
                }
                else{
                    nodes[temp].next[i]=nodes[fail[temp]].next[i];
                }
            }
        }
    }
    int query(string s){
        int now=0,res=0;
        int len=s.length();
        for(int i=0;i<len;i++){
            now=nodes[now].next[s[i]-'a'];
            for(int j=now;j&&nodes[j].num!=-1;j=fail[j]){
```

```
                res+=nodes[j].num;
                nodes[j].num=-1;
            }
        }
        return res;
    }
}tree;
```

模板题

```
#include<iostream>
#include<cmath>
#include<algorithm>
#include<cstdio>
#include<cstring>
#include<queue>
using namespace std;
const int maxn =1e6+10;
int trie[maxn][30];
int fail[maxn];
int cnt[maxn];
int tot=0;
inline int read()
{
  int x=0,f=1;char ch=getchar();
  while (!isdigit(ch)){if (ch=='-') f=-1;ch=getchar();}
    while (isdigit(ch)){x=x*10+ch-48;ch=getchar();}
    return x*f;
}
char str[maxn];
void init()
{
  memset(trie, 0, sizeof trie);
  memset(fail, 0, sizeof fail);
  memset(cnt, 0 , sizeof cnt);
  tot=0;
}
void insert(char *s)
{
  int len=strlen(s);
  int root=0;
  for(int i=0;i<len;i++)
    {
      int id=s[i]-'a';
      if(!trie[root][id]) trie[root][id]=++tot;
      root=trie[root][id];
    }
  cnt[root]++;
```

```
}
void getfail()
{
  queue<int> q;
  for(int i=0;i<26;i++)
    {
      if(trie[0][i])
        {
          q.push(trie[0][i]);
        }
    }
  while(!q.empty())
    {
      int root=q.front();
      q.pop();
      for(int i=0;i<26;i++)
        {
          if(trie[root][i])
            {
              fail[trie[root][i]]=trie[fail[root]][i];
              q.push(trie[root][i]);
            }
          else {
            trie[root][i]=trie[fail[root]][i];
          }
        }
    }
}
int query(char *s)
{
  int len=strlen(s);
  int root=0,ans=0;
  for(int i=0;i<len;i++)
    {
      int id=s[i]-'a';
      root=trie[root][id];
      for(int j=root;j&&cnt[j]!=-1;j=fail[j])
        {
          ans+=cnt[j];
          cnt[j]=-1;
        }
    }
  return ans;
}
int main(){
  int n=read();
  for(int i=0;i<n;i++)
    {
      scanf("%s",str);
```

```
      insert(str);
    }
  getfail();
  scanf("%s",str);
  printf("%d\n",query(str));
}
```