回文树

假设我们有一个串S，S下标从0开始，则回文树能做到如下几点：

1.求串S前缀0~i内本质不同回文串的个数（两个串长度不同或者长度相同且至少有一个字符不同便是本质不同）

2.求串S内每一个本质不同回文串出现的次数

3.求串S内回文串的个数（其实就是1和2结合起来）

4.求以下标i结尾的回文串的个数

那么我们该如何构造回文树？

首先我们定义一些变量。

1.len[i]表示编号为i的节点表示的回文串的长度（一个节点表示一个回文串）

2.next[i][c]表示编号为i的节点表示的回文串在两边添加字符c以后变成的回文串的编号（和字典树类似）。

3.fail[i]表示节点i失配以后跳转不等于自身的节点i表示的回文串的最长后缀回文串（和AC自动机类似）。

4.cnt[i]表示节点i表示的本质不同的串的个数（建树时求出的不是完全的，最后count()函数跑一遍以后才是正确的）

5.num[i]表示以节点i表示的最长回文串的最右端点为回文串结尾的回文串个数。

6.last指向新添加一个字母后所形成的最长回文串表示的节点。

7.S[i]表示第i次添加的字符（一开始设S[0] = -1（可以是任意一个在串S中不会出现的字符））。

8.p表示添加的节点个数。

9.n表示添加的字符个数。

|  |
| --- |
| const int MAXN = 100005 ;  const int N = 26 ;  struct Palindromic\_Tree {  int next[MAXN][N] ;//next指针，next指针和字典树类似，指向的串为当前串两端加上同一个字符构成  int fail[MAXN] ;//fail指针，失配后跳转到fail指针指向的节点  int cnt[MAXN] ;  int num[MAXN] ;  int len[MAXN] ;//len[i]表示节点i表示的回文串的长度  int S[MAXN] ;//存放添加的字符  int last ;//指向上一个字符所在的节点，方便下一次add  int n ;//字符数组指针  int p ;//节点指针  int newnode ( int l ) {//新建节点  for ( int i = 0 ; i < N ; ++ i ) next[p][i] = 0 ;  cnt[p] = 0 ;  num[p] = 0 ;  len[p] = l ;  return p ++ ;  }  void init () {//初始化  p = 0 ;  newnode ( 0 ) ;  newnode ( -1 ) ;  last = 0 ;  n = 0 ;  S[n] = -1 ;//开头放一个字符集中没有的字符，减少特判  fail[0] = 1 ;  }  int get\_fail ( int x ) {//和KMP一样，失配后找一个尽量最长的  while ( S[n - len[x] - 1] != S[n] ) x = fail[x] ;  return x ;  }  void add ( int c ) {  c -= 'a' ;  S[++ n] = c ;  int cur = get\_fail ( last ) ;//通过上一个回文串找这个回文串的匹配位置  if ( !next[cur][c] ) {//如果这个回文串没有出现过，说明出现了一个新的本质不同的回文串  int now = newnode ( len[cur] + 2 ) ;//新建节点  fail[now] = next[get\_fail ( fail[cur] )][c] ;//和AC自动机一样建立fail指针，以便失配后跳转  next[cur][c] = now ;  num[now] = num[fail[now]] + 1 ;  }  last = next[cur][c] ;  cnt[last] ++ ;  }  void count () {  for ( int i = p - 1 ; i >= 0 ; -- i ) cnt[fail[i]] += cnt[i] ;  //父亲累加儿子的cnt，因为如果fail[v]=u，则u一定是v的子回文串！  }  } pali\_Tree; |

Timus OJ1960. Palindromes and Super Abilities

求前i个字符串的回文子串数。

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  const int maxn = 100010;  struct PalindromicTree{  int fail[maxn],len[maxn],son[maxn][26];  int tot,last,n;  char s[maxn];  int newnode(int slen = 0){  memset(son[tot],0,sizeof son[tot]);  len[tot] = slen;  return tot++;  }  void init(){  n = tot = last = 0;  newnode(0);  newnode(-1);  fail[1] = fail[0] = 1;  s[n] = -1;  }  int getFail(int x){  while(s[n - len[x] - 1] != s[n]) x = fail[x];  return x;  }  void extend(int c){  s[++n] = c;  int cur = getFail(last);  if(!son[cur][c]){  int x = newnode(len[cur] + 2);  fail[x] = son[getFail(fail[cur])][c];  son[cur][c] = x;  }  last = son[cur][c];  }  }pt;  char str[maxn];  int main(){  while(~scanf("%s",str)){  pt.init();  bool flag = false;  for(int i = 0; str[i]; ++i){  pt.extend(str[i] - 'a');  if(flag) putchar(' ');  printf("%d",pt.tot - 2);  flag = true;  }  putchar('\n');  }  return 0;  } |

[【Tsinsen】A1280. 最长双回文串](http://www.tsinsen.com/ViewGProblem.page?gpid=A1280" \t "_blank)

|  |
| --- |
| #include <cstdio>  #include <cstring>  #include <algorithm>  using namespace std ;  typedef long long LL ;  #define rep( i , a , b ) for ( int i = ( a ) ; i < ( b ) ; ++ i )  #define For( i , a , b ) for ( int i = ( a ) ; i <= ( b ) ; ++ i )  #define rev( i , a , b ) for ( int i = ( a ) ; i >= ( b ) ; -- i )  #define clr( a , x ) memset ( a , x , sizeof a )  const int MAXN = 100005 ;  const int N = 26 ;  struct Palindromic\_Tree {  int next[MAXN][N] ;  int fail[MAXN] ;  int cnt[MAXN] ;  int len[MAXN] ;  int S[MAXN] ;  int last ;  int n ;  int p ;  int newnode ( int l ) {  for ( int i = 0 ; i < N ; ++ i ) next[p][i] = 0 ;  cnt[p] = 0 ;  len[p] = l ;  return p ++ ;  }  void init () {  p = 0 ;  newnode ( 0 ) ;  newnode ( -1 ) ;  last = 0 ;  n = 0 ;  S[n] = -1 ;  fail[0] = 1 ;  }  int get\_fail ( int x ) {  while ( S[n - len[x] - 1] != S[n] ) x = fail[x] ;  return x ;  }  int add ( int c ) {  c -= 'a' ;  S[++ n] = c ;  int cur = get\_fail ( last ) ;  if ( !next[cur][c] ) {  int now = newnode ( len[cur] + 2 ) ;  fail[now] = next[get\_fail ( fail[cur] )][c] ;  next[cur][c] = now ;  }  last = next[cur][c] ;  cnt[last] ++ ;  return len[last] ;  }  void count () {  for ( int i = p - 1 ; i >= 0 ; -- i ) cnt[fail[i]] += cnt[i] ;  }  } ;  Palindromic\_Tree T ;  int n ;  int len[MAXN] ;  char s[MAXN] ;  void solve () {  int ans = 0 ;  n = strlen ( s ) ;  T.init () ;  for ( int i = n - 1 ; i >= 0 ; -- i ) {  len[i] = T.add ( s[i] ) ;  }  T.init () ;  for ( int i = 0 ; i < n - 1 ; ++ i ) {  ans = max ( ans , T.add ( s[i] ) + len[i + 1] ) ;  }  printf ( "%d\n" , ans ) ;  }  int main () {  while ( ~scanf ( "%s" , s ) ) solve () ;  return 0 ;  } |