字符串模板

[一、 Hash 1](#_Toc401600432)

[一般字符串Hash 1](#_Toc401600433)

[字符串区间Hash 1](#_Toc401600434)

[二、 KMP 2](#_Toc401600435)

[函数版 2](#_Toc401600436)

[KMP类 3](#_Toc401600437)

[三、 拓展KMP 4](#_Toc401600438)

[函数版 4](#_Toc401600439)

[四、 Manacher 5](#_Toc401600440)

[Manacher模板 5](#_Toc401600441)

[五、 AC自动机 6](#_Toc401600442)

[普通匹配 6](#_Toc401600443)

[AC自动机+DP 7](#_Toc401600444)

[六、 后缀数组 9](#_Toc401600445)

[论文模板(O(nlogn)) 9](#_Toc401600446)

[七、 字符串最小表示法 10](#_Toc401600447)

[朴素最小表示法 10](#_Toc401600448)

[判定两串是否循环同构 10](#_Toc401600449)

## **Hash**

### 一般字符串Hash

LL strhash (char \*str) {

int i;

LL c, ret = 0;

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

c = str[i] - 'a';

ret = ((ret << 5) + ret) ^ c;

}

return ret;

}

17,37, 79, 163, 331,

673,1361,2729,5471,10949,

21911,43853, 87719, 175447, 350899,

701819,1403641,2807303,5614657, 11229331,

22458671, 44917381, 89834777, 179669557, 359339171,

718678369, 1437356741, 2147483647

### 字符串区间Hash

//prePow预处理MOD^x的值, 可以在程序最开始做!

prePow[0] = 1;

for(i = 1; i <= len; i++) {

prePow[i] = prePow[i - 1] \* MOD;

}

//存储在H数组中, 利用溢出进行Hash

void preHash(int len, char str[], LL H[]) {

int i;

H[0] = str[0];

for(i = 1; i < len; i++) {

H[i] = (H[i - 1] \* MOD + str[i]);

}

}

//求区间[l,r]的Hash值

LL HASH(int l, int r, LL H[], LL prePow[]) {[l,r], base 0

LL ret = 0;

if(l) ret = H[l - 1] \* prePow[r - l + 1];

return (H[r] - ret);

}

## KMP

### 函数版

char S[Maxn], T[Maxn];

int next[Maxn], is[Maxn];

void getnext(char T[], int LT, int next[]) {

int i, j;

next[0]=-1; next[1] = 0;

for (i = 1, j = 0; i < LT; ) {

while (j != -1 && T[i] != T[j]) j = next[j];

i++; j++;

next[i] = j;

}

}

void KMP (char S[], int LS, char T[], int LT, int next[]) {

int i, j;

for(i = 0; i < LS; i++) is[i] = 0;

for (i = 0, j = 0; i < LS; i++) {

while (j != -1 && S[i] != T[j]) j = next[j];

j++;

if (j == LT) {

is[i - LT + 1] = 1;

j = next[j];

}

}

}

### KMP类

struct KMP {//调用init(s(待匹配串),t(模式串));适用多次KMP

char S[Maxn], T[Maxn];

int next[Maxn], is[Maxn];

int LT, LS;

void init(char s[], char t[]) {

int i;

LT = strlen(t);

LS = strlen(s);

for(i = 0; i <= LT; i++) T[i] = t[i];

for(i = 0; i <= LS; i++) S[i] = s[i];

}

void getnext() {

int i, j;

next[0]=-1; next[1] = 0;

for (i = 1, j = 0; i < LT; ) {

while (j != -1 && T[i] != T[j]) j = next[j];

i++; j++;

next[i] = j;

}

}

void kmp() {

int i, j;

for(i = 0; i < LS; i++) is[i] = 0;

for (i = 0, j = 0; i < LS; i++) {

while (j != -1 && S[i] != T[j]) j = next[j];

j++;

if (j == LT) {

is[i - LT + 1] = 1;

j = next[j];

}

}

}

}kmp;

## 拓展KMP

### 函数版

char S[Maxn], T[Maxn];

int next[Maxn], B[Maxn];

void preExKmp(char T[], int LT, int next[]) {

int i, ind = 0, k = 1;

next[0] = LT;

while(ind + 1 < LT && T[ind + 1] == T[ind]) ind++;

next[1] = ind;

for(i = 2; i < LT; i++) {

if(i <= k + next[k] - 1 && next[i - k] + i < k + next[k])

next[i] = next[i - k];

else {

ind = max(0, k + next[k] - i);

while(ind + i < LT && T[ind + i] == T[ind]) ind++;

next[i] = ind; k = i;

}

}

}

void exKmp(char S[], int LS, char T[], int LT, int next[], int B[]) {

int i, ind = 0, k = 0;

preExKmp(T, LT, next);

while(ind < LS && ind < LT && T[ind] == S[ind]) ind++;

B[0] = ind;

for(i = 1; i < LS; i++) {

int p = k + B[k] - 1, L = next[i - k];

if((i - 1) + L < p)

B[i] = L;

else {

ind = max(0, p - i + 1);

while(ind + i < LS && ind < LT && S[ind + i] == T[ind]) ind++;

B[i] = ind;

k = i;

}

}

}

## Manacher

### Manacher模板

//HDU\_3068

//s为原串, str为插入$和#的串

//读入s后, 调用init(s, str, len),

//最后调用Manacher(str,p,len)

//求解遍历p数组求最大值, 注意输出ans-1

//最长回文子串对应原串T中的位置：l = (i - p[i])/2; r = (i + p[i])/2 - 2;

int len, p[Maxn];

char s[Maxn], str[Maxn];

void init(char s[], char str[], int& len) {

int i, j, k;

str[0] = '$';

str[1] = '#';

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

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

str[i \* 2 + 3] = '#';

}

len = len \* 2 + 2;

s[len] = 0;

}

void Manacher (char str[], int p[], int len) {

int i;

int mx = 0;

int id;

for (i = len; i < Maxn; i++)

str[i] = 0; //没有这一句有问题。。就过不了ural1297，比如数据：ababa aba

for (i = 1; i < len; i++) {

if ( mx > i )

p[i] = min ( p[2 \* id - i], p[id] + id - i );

else

p[i] = 1;

for (; str[i + p[i]] == str[i - p[i]]; p[i]++)

;

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

mx = p[i] + i;

id = i;

}

}

}

int main() {

int i, ans;

while (scanf ("%s", s) != EOF) {

len = strlen (s);

init(s, str, len);

Manacher(str, p, len);

ans = 0;

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

if (p[i] > ans)

ans = p[i];

printf ("%d\n", ans - 1);

}

return 0;

}

## AC自动机

### 普通匹配

struct node {

struct node \* fail;

struct node \* next[10];

bool is;

int lab;

}Tree[Maxn], \* root, \* que[Maxn];

int tot, head, tail;

node \* newNode() {

node \* p = &Tree[tot++];

p->is = false;

p->lab = 0;

p->fail = NULL;

for(int i = 0; i < 10; i++) p->next[i] = NULL;

return p;

}

void Insert(char str[], int n) {

node \* p = root;

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

k = str[i] - '0';

if(p->next[k] == NULL) {

p->next[k] = newNode();

}

p = p->next[k];

}

p->is = true;

p->lab = n;

}

void buildAC() {

head = tail = 0;

int i;

node \* p, \* q;

root->fail = root;

que[tail++] = root;

while(head < tail) {

p = que[head++]; q = p->fail;

for(i = 0; i < 10; i++) {

if(p->next[i] != NULL) {

if(p == root) p->next[i]->fail = root;

else {

p->next[i]->fail = q->next[i];

p->next[i]->is |= q->next[i]->is;

}

que[tail++] = p->next[i];

}

else {

if(p == root) p->next[i] = root;

else p->next[i] = q->next[i];

}

}

}

}

void query(char str[]) {

node \* p , \* q;

p = root;

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

k = str[i]-'0';

p = p->next[k];

if(p->is) {

q = p;

while(q->is) {

cnt[q->lab]++;

q = q->fail;

}

}

}

}

### AC自动机+DP

struct node {

struct node \* next[26];

struct node \* fail;

int is;

int v;

}A[Maxn], \*root, \*que[Maxn];

int head, tail;

char str[Maxn];

int tot, ans, n;

node \* newNode() {

node \* p = &A[tot++];

p->fail = NULL;

p->is = false;

p->v = -MOD;

for(int i = 0; i <26; i++) p->next[i] = NULL;

return p;

}

void Insert(node \* root, char str[], int l, int r) {

int i, k;

for(i = l; i < r; i++) {

k = str[i] - 'a';

if(root->next[k] == NULL) root->next[k] = newNode();

root = root->next[k];

}

root->is = 1;

}

void buildAC() {

int k;

head = tail = 0;

que[tail++] = root;

root->fail = root;

node \* p, \* q;

while(head < tail) {

p = que[head++]; q = p->fail;

for(k = 0; k < 26; k++) {

if(p->next[k] != NULL) {

if(p == root) p->next[k]->fail = root;

else {

p->next[k]->fail = q->next[k];

p->next[k]->is |= q->next[k]->is;

}

que[tail++] = p->next[k];

}

else {

if(p == root) p->next[k] = root;

else p->next[k] = q->next[k];

}

}

}

}

void query(int n, int l, int r, int v0) {

int i, k, v;

node \* p, \* q;

p = root;

v = v0;

for(i = l; i < r; i++) {

k = str[i] - 'a';

p = p->next[k];

if(p->is) {

q = p;

while(q->is) {

cmax(v, v0 + q->v);

q = q->fail;

}

}

}

cmax(p->v, v);

cmax(ans, v);

}

## 后缀数组

### 论文模板(O(nlogn))

论文模板, 使用时注意num[]有效位为0~n-1, 但是需要将num[n]=0, 否则RE;另外, 对于模板的处理将空串也处理了, 作为rank最小的串, 因此有效串为0~n共, n-1个, 在调用da()函数时, 需要调用da(num, n + 1, m); 对于sa[], rank[]和height[]数组都将空串考虑在内, 作为rank最小的后缀!

//O(nlogn)

//调用da(num, len+1, m);//m为字符个数略大

int len;

int num[Maxn]; //待处理的串

int sa[Maxn], rank[Maxn], height[Maxn]; //sa[1~n]value(0~n-1); rank[0..n-1]value(1..n); height[2..n]

int wa[Maxn], wb[Maxn], wv[Maxn], wd[Maxn];

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

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

}

void da(int \*r, int n, int m) { // 倍增算法 r为待匹配数组 n为总长度+1 m为字符范围

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

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

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

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

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

for(j = 1, p = 1; p < n; j <<= 1, 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++) wd[i] = 0;

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

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

for(i = n - 1; i >= 0; i--) sa[--wd[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++;

}

}

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

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

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

}

}

## 字符串最小表示法

### 朴素最小表示法

ZOJ2006 ZOJ1729 HDU 3374

int MinRep (char S[], int L) {

int i = 0, j = 1, k = 0, t;

while (i < L && j < L && k < L) { //找不到比它还小的或者完全匹配

t = S[ (i + k) % L] - S[ (j + k) % L];

// t = s[(i + k) >= L ? i + k - L : i + k]

// - s[(j + k) >= L ? j + k - L : j + k];

if (t == 0)

k++;//相等的话,检测长度加1

else {

//大于的话,s[i]为首的肯定不是最小表示,最大表示就改<

if (t > 0) i += k + 1;

else j += k + 1;

if (i == j) j++;

k = 0;

}

}

return min (i, j);

}

### 判定两串是否循环同构