字符串

AC 自动机 - GY

```
const int maxc = 4;
class ACAnode {
  public:
    int go[maxc];
    int fail, acc;
    void clr() {
        memset (go, 0, sizeof (go) );
        fail = acc = 0;
    }
};
class ACA {
  public:
    int root, size;
    ACAnode a[1010];
    void pre() {
        root = size = 1;
        a[1].clr();
    }
    int init (int last, int x) {
        if (!a[last].go[x]) {
            a[last].go[x] = ++size;
            a[size].clr();
        }
        return a[last].go[x];
    }
    void build() {
        l = 1;
        r = 0;
        q[++r] = root;
        a[root].fail = root;
        for (; l <= r; l++) {
            int x = q[l];
            for (int j = 0; j < maxc; j++)
                if (a[x].go[j]) {
                    int y = a[x].go[j];
                    int last = a[x].fail;
                    while (last != root && !a[last].go[j]) last =
a[last].fail;
                    if (a[last].go[j] && a[last].go[j] != y)
a[y].fail = a[last].go[j];
```

```
                else a[y].fail = root;
                if (a[a[y].fail].acc) a[y].acc = 1;
                q[++r] = y;
            }
         for (int j = 0; j < maxc; j++)
            if (!a[x].go[j]) {
                if (x != root) a[x].go[j] = a[a[x].fail].go[j];
                else a[x].go[j] = root;
            }
      }
   }
} aca;
```

AC 自动机_LQY

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<vector>
using namespace std;

const int MaxN=100007;

typedef struct {int id,fa,ch[26],fail; bool isw;} NodeTp;

int n,m,l,root,tot,idx;
char str[MaxN]; int id[MaxN];
NodeTp trie[MaxN];
vector<int> iv[MaxN],qry[MaxN][2];
int fr,re,q[MaxN];
int indx,L[MaxN<<1],R[MaxN<<1],C[MaxN<<1];
int ans[MaxN];

void Add(int a,int b) {trie[a].fail=b; iv[b].push_back(a);
return;}

void MakeAC()
{
    int u,i;
    fr=re=1; q[re]=root; trie[root].fail=root;
    while(fr<=re)
    {
        u=q[fr++];
        for(i=0;i<26;i++)
```

```
            if(trie[u].ch[i])
            {
              if(u==root) Add(trie[u].ch[i],root);
              else Add(trie[u].ch[i],trie[trie[u].fail].ch[i]);
              q[++re]=trie[u].ch[i];
            }
            else
            {
               if(u==root) trie[u].ch[i]=root;
               else trie[u].ch[i]=trie[trie[u].fail].ch[i];
            }
    }
    return;
}

void DFS(int x)
{
    int i;
    L[x]=++indx;
    for(i=0;i<iv[x].size();i++)
       DFS(iv[x][i]);
    R[x]=++indx;
    return;
}

void Updata(int x,int d) {for(x;x&&x<=n;x+=x&(x^(x-1))) C[x]+=d;
return;}
int Calc(int x) {int s=0; for(x;x>0;x-=x&(x^(x-1))) s+=C[x];
return s;}

int main()
{
   freopen("type.in","r",stdin);
   freopen("type.out","w",stdout);
   int i,j,k,p,x,y;
   root=tot=1;
   scanf("%s",str+1); l=strlen(str+1);
   j=root;
   for(i=1;i<=l;i++)
   {
      if(str[i]=='B') j=trie[j].fa;
      else if(str[i]=='P') trie[j].isw=true,id[++idx]=j;
      else
      {
```

```
            if(!trie[j].ch[str[i]-'a'])
               trie[j].ch[str[i]-'a']=++tot,trie[tot].fa=j;
            j=trie[j].ch[str[i]-'a'];
        }
    }
    scanf("%d",&m);
    for(i=1;i<=m;i++)
    {

scanf("%d%d",&x,&y),x=id[x],y=id[y],qry[y][0].push_back(x),qry[y]
[1].push_back(i);
    }
    MakeAC();
    DFS(1);
    n=indx<<1; p=1;
    for(i=1;i<=l;i++)
    {
        if(str[i]=='B') Updata(L[p],-1),p=trie[p].fa;
        else if(str[i]=='P')
        {
            j=p;
            for(k=0;k<qry[j][0].size();k++)
            {
               ans[qry[j][1][k]]=Calc(R[qry[j][0][k]])-
Calc(L[qry[j][0][k]]-1);
            }
        }
        else p=trie[p].ch[str[i]-'a'],Updata(L[p],1);
    }
    for(i=1;i<=m;i++) printf("%d\n",ans[i]);
    return 0;
}
```

kmp & exkmp_GY

```
void get_kmp() {
    memset (fail, 0, sizeof (fail) );
    for (int i = 2; i <= N; i++) {
        int x = fail[i - 1];
        while (x && s1[x + 1] != s1[i]) x = fail[x];
        if (s1[x + 1] == s1[i]) fail[i] = x + 1;
        else fail[i] = 0;
    }
    memset (kmp, 0, sizeof (kmp) );
```

```
    for (int i = 1; i <= N; i++) {
        int j = kmp[i - 1];
        while (j && s1[j + 1] != s2[i]) j = fail[j];
        if (s1[j + 1] == s2[i]) kmp[i] = j + 1;
        else kmp[i] = 0;
    }
}

void exkmp() {
    memset (fail, 0, sizeof (fail) );
    fail[2] = (s[2] == s[1]);
    int k = 2, r = 1 + fail[2];
    for (int i = 3; i <= len; i++) {
        if (i <= r)
            fail[i] = min (r - i + 1, fail[i - k + 1]);
        while (i + fail[i] <= len && s[fail[i] + 1] == s[i +
fail[i]])
            fail[i]++;
        if (i + fail[i] - 1 > r) {
            k = i;
            r = i + fail[i] - 1;
        }
    }
}
```

manachar_GY

```
void manachar() {
    memset (length, 0, sizeof (length) );
    length[1] = 1;
    int k = 1, rr = k + length[k] - 1;
    for (int i = 2; i <= r; i++) {
        if (i <= rr) length[i] = min (rr - i + 1, length[2 * k -
i]);
        while (i + length[i] <= r && i - length[i] > 0 && ss[i +
length[i]] == ss[i - length[i]]) length[i]++;
        if (rr < i + length[i] - 1) {
            k = i;
            rr = i + length[i] - 1;
        }
    }
}
```

SAM - WP

```
struct node {
    node *f, *nex[26];
    int ml, size, first;
    node () {
        ml = size = first = 0;
    }
} pool[maxn], *tail, *init, *rank[maxn];
void add (int ch, int len) {
    node *p = tail, *np = &pool[++tot];
    np -> ml = len;
    for (; p && !p -> nex[ch]; p = p -> f) p -> nex[ch] = np;
    tail = np;
    if (!p) np -> f = init;
    else {
        if (p -> nex[ch] -> ml == p -> ml + 1) np -> f = p ->
nex[ch];
        else {
            node *q = p -> nex[ch], *just = &pool[++tot];
            *just = *q;
            just -> ml = p -> ml + 1;
            q -> f = np -> f = just;
            for (; p && p -> nex[ch] == q; p = p -> f) p -> nex[ch]
= just;
        }
    }
}
```

SA – GY

```
void getran (int x) {
    tempran[sa[1]] = 1;
    for (int i = 2; i <= N; i++)
        if ( (ran[sa[i]] == ran[sa[i - 1]]) && (ran[sa[i] + x] ==
ran[sa[i - 1] + x]) )
            tempran[sa[i]] = tempran[sa[i - 1]];
        else tempran[sa[i]] = tempran[sa[i - 1]] + 1;
    memcpy (ran, tempran, sizeof (ran) );
}

void work (int x) {
    memset (temp, 0, sizeof (temp) );
    for (int i = 1; i <= N; i++) temp[ran[i + x]]++;
    for (int i = 1; i < maxn; i++) temp[i] += temp[i - 1];
```

```
    for (int i = N; i; i--) tsa[temp[ran[i + x]]--] = i;
    memset (temp, 0, sizeof (temp) );
    for (int i = 1; i <= N; i++) temp[ran[i]]++;
    for (int i = 1; i < maxn; i++) temp[i] += temp[i - 1];
    for (int i = N; i; i--) sa[temp[ran[tsa[i]]]--] = tsa[i];
    getran (x);
}

void getheight() {
    int i, j, k = 0;
    for (i = 1; i <= N; height[ran[i++]] = k)
        for (k ? --k : 0, j = sa[ran[i] - 1]; s[i + k] == s[j + k];
k++);
}

void getST() {
    for (int i = 1; i <= N; i++) ST[0][i] = height[i];
    for (int step = 1, s = 1; (s << 1) <= N; step++, s <<= 1)
        for (int i = 1; i <= N; i++)
            if (i + s <= N) ST[step][i] = min (ST[step - 1][i],
ST[step - 1][i + s]);
            else ST[step][i] = 0;
}

//原串中 x、y 位置的后缀的 lcp
int lcp (int x, int y) {
    if (x == y) return N - x + 1;
    int l = ran[x], r = ran[y];
    if (l > r) swap (l, r);
    l++;
    int delta = r - l + 1;
    int t = log (delta) / log (2);
    return min (ST[t][l], ST[t][r - (1 << t) + 1]);
}
```

Suffix-Array - WP

```
void get_suffix() {
    int i, j;
    for (i = 1; i <= top; i++) rank[i] = s[i] - 'a' + 1;
    for (i = 1; i <= top; i++) wp[rank[i]]++;
    for (i = 1; i <= 30; i++) wp[i] += wp[i - 1];
    for (i = top; i; i--) SA[wp[rank[i]]--] = i;
    int u, q;
```

```
    for (q = 1; q < top; q <<= 1) {
        memset (wp, 0, sizeof (wp) );
        memcpy (SA_, SA, sizeof (SA) );
        for (i = 1; i <= top; i++) wp[rank[i]]++;
        for (i = 1; i <= top; i++) wp[i] += wp[i - 1];
        for (i = top; i; i--)
            if (SA_[i] > q) SA[wp[rank[SA_[i] - q]]--] = SA_[i] -
q;
        for (i = top - q + 1; i <= top; i++) SA[wp[rank[i]]--] = i;
        memcpy (rank_, rank, sizeof (rank) );
        rank[SA[1]] = u = 1;
        for (i = 2; i <= top; i++) {
            if (judge (rank_, SA[i], SA[i - 1], q) ) rank[SA[i]] =
u;
            else rank[SA[i]] = ++u;
        }
    }
}
void get_height() {
    for (i = 1; i <= top; i++) {
        if (rank[i] == 1) {
            H[rank[i]] = 0;
            continue;
        }
        H[rank[i]] = max (H[rank[i - 1]] - 1, 0);
        while (s[i + H[rank[i]]] == s[SA[rank[i] - 1] +
H[rank[i]]]) H[rank[i]]++;
    }
}
```

最小表示法 - WP

```
void zxbx (char *s, char *ans) {
    int L = strlen (s);
    int i, j, cot;
    for (i = 0, j = 1, cot = 0; cot < L && i < L && j < L; ) {
        if (s[ (i + cot) % L] == s[ (j + cot) % L]) cot++;
        else {
            if (s[ (i + cot) % L] > s[ (j + cot) % L]) i += cot +
1;
            else j += cot + 1;
            if (i == j) j = i + 1;
            cot = 0;
        }
```

```
    }
    int p = min (i, j);
    for (i = p; i < L; i++) ans[i p] = s[i];
    for (i = 0; i < p; i++) ans[L p + i] = s[i];
    return;
}
```