二维字符串哈希

Hint:两次需要使用不同的进制数

1.对$n*m$的矩阵进行二维哈希

```
base1=1331,base2=131;
for(int i=1;i<=n;i++)
    for(int j=1;j<=m;j++)
    {
        char x;
        cin>>x;
        h[i][j]=h[i][j-1]*base1+x;
    }
for(int i=1;i<=n;i++)
    for(int j=1;j<=m;j++)
        h[i][j]=h[i][j]+h[i-1][j]*base2;
```

2.对区域$[x1,y1,x2,y2]$求哈希

```
ull get(int x1,int y1,int x2,int y2)
{
    return h[x2][y2]-h[x2][y1-1]*p1[y2-y1+1]-h[x1-1][y2]*p2[x2-x1+1]
      +h[x1-1][y1-1]*p1[y2-y1+1]*p2[x2-x1+1];
}
```

PAM

```
#include <cstdio>
#define N 300010
inline long long mymax(long long x, long long y) {return x > y ? x :
y;}
struct Palindromic_Tree {
    int nxt[N][26]; //i节点对应的回文串在两边各加一个字符后变成的节点编号
    int fail[N]; //fail[i]表示的串是i的最长后缀回文串
    int cnt[N]; //表示这个串出现过几次
    int num[N]; //节点i这个串的后缀有多少是回文的
    int len[N]; //节点i表示的回文串的长度
    int S[N]; //S[i]是第i次添加的字符
    int last; //以n结束的最长回文串所在的节点
    int n; //目前添加的字符个数
    int p; //下一个新建节点的标号
    inline int newnode(int l) {
        //新建节点
        for(int i = 0; i < 26; ++i) nxt[p][i] = 0;
        cnt[p] = num[p] = 0;
        len[p] = l;
        return p++;
    }
    inline void init() {
        p = 0;
        newnode(0);
        newnode(-1);
        last = n = 0;
        S[n] = -1;
        fail[0] = 1;
    }
    inline int get_fail(int x) {
        while(S[n - len[x] - 1] != S[n]) x = fail[x];
        return x;
    }
    inline void add(int c) {
        S[++n] = c;
        int cur = get_fail(last); //通过上一个回文串找这个串的匹配位置
```

```
        if(!nxt[cur][c]) {
            //这个回文串从未出现过
            int now = newnode(len[cur] + 2);
            fail[now] = nxt[get_fail(fail[cur])][c]; //和AC自动机简直不要
太类似
            nxt[cur][c] = now;
            num[now] = num[fail[now]] + 1;
        }
        last = nxt[cur][c];
        ++cnt[last];
    }
    inline void dp() {
        for(int i = p - 1; i >= 0; --i) cnt[fail[i]]+= cnt[i];
    }
    inline void work() {
        init();
        char c = getchar();
        while(c >= 'a' && c <= 'z') {
            add(c - 'a');
            c = getchar();
        }
        dp();
        long long ans = 0;
        for(int i = p - 1; i >= 0; --i) ans = mymax(ans, 1ll * cnt[i]
* len[i]);
        printf("%lld", ans);
    }
}pdtree;
int main() {
    pdtree.work();
    return 0;
}
```