```
/*  SPOJ - APIO14_A - Palindromes
        Given a string S of lower-case Latin letters(a-z), |S| <= 300,000. Let us
        define substring's "occurrence value" as the number of the substring
        occurrences in the string multiplied by the length of the substring.
        For a given string find the largest occurrence value of palindromic substrings.
*/
#include<bits/stdc++.h>
using namespace std;
#define ll long long
const int MAXN = 300005;
struct Node{
    int nxt[26];
    int val;
    int length, suffixLink;
    int startPos, endPos;
};
struct PalTree{
    Node tree[MAXN];
    Node root1, root2;
    int ptr, curNode;
    char s[MAXN];

    void init(){
        root1.length = -1, root1.suffixLink = 1;
        root2.length =  0, root2.suffixLink = 1;
        tree[1] = root1, tree[2] = root2;
        ptr = curNode = 2;
    }

    void addLetter(int pos){
        int ch = s[pos]-'a';
        int cur = curNode;

        while(true){
            int curLength = tree[cur].length;
            if(pos-1-curLength >= 0 && s[pos-1-curLength] == s[pos])break;
            cur = tree[cur].suffixLink;
        }

        if(tree[cur].nxt[ch] != 0){
            curNode = tree[cur].nxt[ch];
            tree[curNode].val++;
            return;
        }

        ptr++;
        curNode = ptr;
        tree[cur].nxt[ch] = curNode;
        tree[curNode].length = tree[cur].length + 2;
        tree[curNode].startPos = pos - tree[curNode].length + 1;
        tree[curNode].endPos = pos;
```

```
52.
53.         if(tree[curNode].length == 1){
54.             tree[curNode].suffixLink = 2;
55.             tree[curNode].val = 1;
56.             return;
57.         }
58.
59.         while(true){
60.             cur = tree[cur].suffixLink;
61.             int curLength = tree[cur].length;
62.             if(pos-1-curLength >= 0 && s[pos-1-curLength] == s[pos]){
63.                 tree[curNode].suffixLink = tree[cur].nxt[ch];
64.                 break;
65.             }
66.         }
67.
68.         tree[curNode].suffixLink = tree[cur].nxt[ch];
69.         tree[curNode].val = 1;
70.         return;
71.     }
72.
73.     ll getResult(){
74.         ll ans = 0;
75.         for(int i=ptr; i>=3; i--){
76.             ll sum = (ll)tree[i].val * tree[i].length;
77.             ans = max(ans, sum);
78.             tree[tree[i].suffixLink].val += tree[i].val;
79.         }
80.         return ans;
81.     }
82.
83.     void Clear(){
84.         for(int i=0; i<=ptr; i++){
85.             memset(tree[i].nxt, 0, sizeof(tree[i].nxt));
86.         }
87.     }
88. };
89. PalTree Pt;
90. int main(){
91.     scanf("%s",&Pt.s);
92.     int n = strlen(Pt.s);
93.     Pt.init();
94.     for(int i=0; i<n; i++) Pt.addLetter(i);
95.     ll ans = Pt.getResult();
96.     printf("%lld\n",ans);
97.     return 0;
98. }
99.
100. Input                  Output
101. ababa                  6
102. abacaba                7
```