```
// akshayjaggi

#include<bits/stdc++.h>

using namespace std;

//#define TRACE
#ifdef TRACE
#define trace(...) __f(#__VA_ARGS__, __VA_ARGS__)
template <typename Arg1>
void __f(const char* name, Arg1&& arg1){
    cerr << name << " : " << arg1 << std::endl;
}
template <typename Arg1, typename... Args>
void __f(const char* names, Arg1&& arg1, Args&&... args){
    const char* comma = strchr(names + 1, ',');cerr.write(names,
comma - names) << " : " << arg1<<" | ";__f(comma+1, args...);
}
#else
#define trace(...)
#endif

#define si(x) scanf("%d",&x)
#define F first
#define S second
#define PB push_back
#define MP make_pair


typedef long long LL;
typedef pair<int,int> PII;
typedef vector<int> VI;
typedef vector<PII> VPII;

#ifdef ONLINE_JUDGE
//FILE *fin = freopen("in","r",stdin);
//FILE *fout = freopen("out","w",stdout);
#endif

class SAutomata
{
        class SNode
        {
                public:
                        int link, len;
                        map<char,int> next;
                        SNode()
                        {
                                link = -1;
                                len = 0;
                        }
                        SNode(const SNode& other)
                        {
                                link = other.link;
                                len = other.len;
                                next = other.next;
                        }
        };
        vector<SNode> nodes;
        vector<long long int> dp;
        int last, cur;
        void dfs(int node)
        {
                if(dp[node]!=-1)
                        return;
                for(auto it: nodes[node].next)
                        dfs(it.S);
                long long int sum = 0;
                for(auto it: nodes[node].next)
                        sum += dp[it.S];
                dp[node] = 1 + sum;
        }
        long long int dfs2(int node)
        {
                if(dp[node]!=-1)
                        return 0;
                dp[node]=0;
                long long int ans = 0;
                for(auto it: nodes[node].next)
                        ans += dfs2(it.S);
                if(node!=0)
                        ans+=(nodes[node].len -
nodes[nodes[node].link].len);
                return ans;
        }


        public:

        SAutomata(string S)
                : nodes(2*S.size()),
                dp(2*S.size(), -1),
                last(0),
                cur(1)
        {
                for(int i=0; i<S.size(); i++)
                        add_char(S[i]);
        }
        void add_char(char A)
        {
                trace(A);
                int nw = cur++;
                nodes[nw].len = nodes[last].len+1;
                int p=last;
                for(; p!=-1 &&
nodes[p].next.find(A)==nodes[p].next.end(); p=nodes[p].link)
                {
                        nodes[p].next[A]=nw;
                }
                if(p==-1)
                {
                        nodes[nw].link = 0;
                }
                else if(nodes[nodes[p].next[A]].len == nodes[p].len
+ 1)
                {
                        nodes[nw].link = nodes[p].next[A];
                }
                else
                {
                        int nxt = nodes[p].next[A];
                        int clone = cur++;
                        nodes[clone] = nodes[nxt];
                        nodes[clone].len = nodes[p].len + 1;
                        nodes[nxt].link = clone;
                        nodes[nw].link = clone;
                        for(; p!=-1 && nodes[p].next.find(A)!
=nodes[p].next.end() && nodes[p].next[A]==nxt; p=nodes[p].link)
                        {
                                nodes[p].next[A]=clone;
                        }
                }
                last = nw;
        }
        long long int count_distinct()
        {
                return dfs2(0);
                dfs(0);
                for(int i=0; i<=last; i++)
                        trace(i,dp[i]);
                return dp[0] - 1 ;
        }
};

int main()
{
        ios_base::sync_with_stdio(false); cin.tie(0);
        int T;
        cin>>T;
        while(T--)
        {
                string S;
                cin>>S;
                auto mata = SAutomata(S);
                cout<<mata.count_distinct()<<endl;
        }
  return 0;
}
```