// SAM（原题来自洛谷SAM模板题）

const int MAXN=1e6+48;

char s[MAXN+48];int len;

vector<int> v[MAXN\*2];int sz[MAXN\*2];

//sz数组当前记录的是SAM的这个节点是否包含一个新的后缀

//在parent树上对SAM做一遍深搜，统计sz的子树和，可以得到每个节点的right集合大小

namespace SAM

{

int tot,nxt[MAXN\*2][27],par[MAXN\*2],maxn[MAXN\*2],root,last;

inline void init() {tot=1;root=last=1;}

inline void insert(char ch)

{

int p=last,np=++tot;;maxn[np]=maxn[p]+1;sz[np]=1;int w=ch-'a'+1;

while (p && !nxt[p][w]) nxt[p][w]=np,p=par[p];

if (!p) {par[np]=root;last=np;return;}

int q=nxt[p][w];

if (maxn[q]==maxn[p]+1) {par[np]=q;last=np;return;}

int nq=++tot;maxn[nq]=maxn[p]+1;

memcpy(nxt[nq],nxt[q],sizeof(nxt[q]));

par[nq]=par[q];par[np]=nq;par[q]=nq;

while (p && nxt[p][w]==q) nxt[p][w]=nq,p=par[p];

last=np;

}

}

// SPOJ LCS2

//提供了SAM上多串跑匹配的解决方案

struct node

{

int next[30];

int val,fail;

int maxn[15];

inline void init()

{

val=fail=0;

for (int i=1;i<=26;i++) next[i]=0;

for (int i=1;i<=10;i++) maxn[i]=0;

}

}SAM[800048];int tot=0;

int root,last;

vector<int> v[100048];

char s1[100048],s2[100048];

int len1,len2;

inline void extend(int w)

{

int p=last,np=++tot;SAM[np].init();

SAM[np].val=SAM[p].val+1;v[SAM[np].val].pb(np);

while (p && !SAM[p].next[w]) SAM[p].next[w]=np,p=SAM[p].fail;

if (!p)

SAM[np].fail=root;

else

{

int q=SAM[p].next[w];

if (SAM[p].val+1==SAM[q].val)

{SAM[np].fail=q;}

else

{

int nq=++tot;SAM[nq].init();

memcpy(SAM[nq].next,SAM[q].next,sizeof(SAM[q].next));

SAM[nq].val=SAM[p].val+1;v[SAM[nq].val].pb(nq);

SAM[nq].fail=SAM[q].fail;

SAM[q].fail=nq;SAM[np].fail=nq;

while (p && SAM[p].next[w]==q) SAM[p].next[w]=nq,p=SAM[p].fail;

}

}

last=np;

}

int main ()

{

int i,j;

scanf("%s",s1+1);len1=strlen(s1+1);

root=last=++tot;SAM[root].init();

for (i=1;i<=len1;i++)

extend(s1[i]-'a'+1);

int len,ans=0,cur,ind=0;

while (scanf("%s",s2+1)!=EOF)

{

++ind;

len2=strlen(s2+1);cur=root;len=0;

for (i=1;i<=len2;i++)

{

int w=s2[i]-'a'+1;

if (SAM[cur].next[w])

{

len++;

cur=SAM[cur].next[w];

}

else

{

while (cur && !SAM[cur].next[w]) cur=SAM[cur].fail;

if (!cur)

len=0,cur=root;

else

{

len=SAM[cur].val+1;

cur=SAM[cur].next[w];

}

}

SAM[cur].maxn[ind]=max(SAM[cur].maxn[ind],len);

}

}

// 子节点的信息沿着father边更新

for (len=len1;len>=1;len--)

for (i=0;i<int(v[len].size());i++)

for (j=1;j<=ind;j++)

if (SAM[v[len][i]].fail)

{ SAM[SAM[v[len][i]].fail].maxn[j]=max(SAM[SAM[v[len][i]].fail].maxn[j],SAM[v[len][i]].maxn[j]);

}

for (i=1;i<=tot;i++)

{

int res=INF;

for (j=1;j<=ind;j++)

res=min(res,SAM[i].maxn[j]);

res=min(res,SAM[i].val);

ans=max(ans,res);

}

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

return 0;

}