/\*

Given two words (beginWord and endWord), and a dictionary's word list, find the length of shortest transformation sequence from beginWord to endWord, such that:

Only one letter can be changed at a time.

Each transformed word must exist in the word list. Note that beginWord is not a transformed word.

For example,

Given:

beginWord = "hit"

endWord = "cog"

wordList = ["hot","dot","dog","lot","log","cog"]

As one shortest transformation is "hit" -> "hot" -> "dot" -> "dog" -> "cog",

return its length 5.

思路是 BFS，很巧妙

\*/

class Solution {

public:

int ladderLength(string beginWord, string endWord, vector<string>& wordList)

{

map<string,int> wordmap;

for(int i=0;i<wordList.size();i++)

wordmap[wordList[i]]++;

int stepnum=1;

stack<string> s1,s2;

s1.push(beginWord);

while(wordmap.size()>0 && !s1.empty())

{

while(!s1.empty())

{

string str=s1.top();

s1.pop();

for(int i=0;i<str.size();i++)

{

for(char j='a';j<='z';j++)

{

char temp=str[i];

str[i]=j;

if(str==endWord)

return stepnum+1;

map<string,int>::iterator it=wordmap.find(str);

if(it!=wordmap.end())

{

s2.push(str);

wordmap.erase(it);

}

str[i]=temp;

}

}

}

swap(s1,s2);

stepnum++;

}

return 0;

}

};