126. Word Ladder II JAVA

Add to List
QuestionEditorial Solution
My Submissions

- Total Accepted: **56840**
- Total Submissions: **416610**
- Difficulty: **Hard**
- Contributors: **Admin**

Given two words (*beginWord* and *endWord*), and a dictionary's word list, find all shortest transformation sequence(s)

from *beginWord* to*endWord*, such that:

1. Only one letter can be changed at a time

2. Each intermediate word must exist in the word list

For example,

Given:

beginWord = "hit"

endWord = "cog"

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

Return

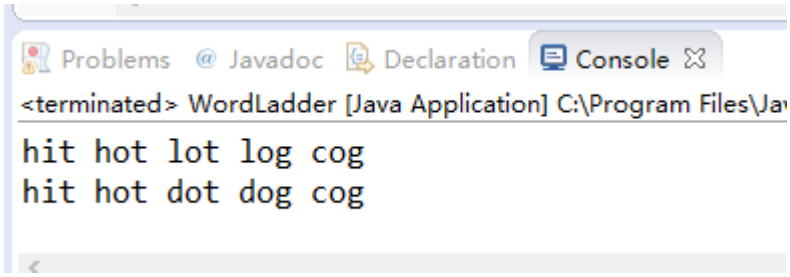
```
[
  ["hit","hot","dot","dog","cog"],
  ["hit","hot","lot","log","cog"]
]
```

```
import java.util.HashMap;
import java.util.HashSet;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import java.util.Set;
import java.util.Vector;

public class WordLadder {

static Vector<List<String>> res;
```

```
static List<String> path;
static HashMap<String, List<String>> myMap;
static Set<String> Worddict;

private static void getChildren(String s, Set<String> next,
Set<String> WordDict)
{
     for(int i=0;i<s.length();i++)
     {
          String temp = s;
          StringBuilder sb = new StringBuilder(temp);
          for(char a = 'a'; a<='z';a++)
          {
               sb.setCharAt(i, a);
               String tmp = sb.toString();
               if(WordDict.contains(tmp))
               {
                    next.add(tmp);
                    if(myMap.containsKey(tmp))
                    {
                         List<String> listt = myMap.get(tmp);
                         listt.add(s);
                         myMap.put(tmp, listt);
                    }else{
                        List<String> listt = new
LinkedList<String>();
                         listt.add(s);
                         myMap.put(tmp, listt);
                    }
               }
          }
     }
}

private static List<String> copyList(List<String> p)
{
     List<String> re = new LinkedList<>();
     for(int k=0;k<p.size();k++) re.add(p.get(k));
     return re;
}

private static void copySet(Set<String> a,Set<String> b)
{
     a.clear();
     Iterator<String> it = b.iterator();
     while(it.hasNext())
     {
          a.add(it.next());
```

```
    }
}

private static void printPath(String beginWord, String endWord)
{
   path.add(endWord);
   if(beginWord.equals(endWord))
   {
     reverse(path);
     List<String> ltmp = copyList(path);
     res.add(ltmp);
     reverse(path);
   }else{
     List<String> v = myMap.get(endWord);
          for(int k=0;k<v.size();k++)
          {
               printPath(beginWord, v.get(k));
          }
     }
   path.remove(endWord);
}

private static void reverse(List<String> path1)
{
     int size = path1.size();
     for(int i=0;i<size/2;i++)
     {
          String a = path1.get(i);
          String b = path1.get(size-i-1);
          path1.set(i, b);
          path1.set(size-i-1, a);
     }
}

public static void findLadders(String beginword, String endword,
Set<String> wordlist)
{
     wordlist.add(beginword);
     wordlist.add(endword);
     Set<String> next,current;
     next = new HashSet<String>();current = new HashSet<String>();
     current.add(beginword);
     while(current.size()>0)
     {

          if(current.contains(endword))
          {
               printPath(beginword, endword);
```

```
                return;
            }

        Iterator<String> it = current.iterator();
        while(it.hasNext())
        {
            String ss = it.next();
            if(wordlist.contains(ss)) wordlist.remove(ss);
        }
        it = current.iterator();
        while(it.hasNext())
        {
            String ss = it.next();
            getChildren(ss, next, wordlist);
        }
        current.clear();
        copySet(current,next);
        next.clear();
        }
}


public static void main(String[] args) {
// TODO Auto-generated method stub
res = new Vector<List<String>>();
myMap = new HashMap<String, List<String>>();
path = new LinkedList<String>();
//wordList = ["hot","dot","dog","lot","log"]
String[] wordarr = {"hot","dot","dog","lot","log"};
Worddict = new HashSet<String>();
for(int k=0;k<wordarr.length;k++) Worddict.add(wordarr[k]);
findLadders("hit","cog",Worddict);
System.out.println(res.size());
for(int i=0;i<res.size();i++)
{
    List<String> tp = res.get(i);
    for(int j=0;j<tp.size();j++)
        System.out.printf("%s ", tp.get(j));
System.out.printf("\n");
}
}

}
```


Problems
Javadoc
Declaration
Console
<terminated> WordLadder [Java Application] C:\Program Files\Ja
hit hot lot log cog
hit hot dot dog cog