The Word Ladders Game

MARCH 30TH, 2015

The Word Ladders Game

- This is a word game invented by Lewis Carroll in 1877.
- You are given two words on the same length, e.g., “cold” and “warm.”
- Your task is to find a chain of words from the starting word (“cold”) to the ending word (“warm”) so that each successive word differs from the previous in exactly one letter.
- **Example:** “cold”, “cord”, “card”, “ward”, “warm.”

The 5-letter version

- Donald Knuth, a Turing award winning computer scientist created a file of 5,757 5-letter words. These were all valid 5-letter English words at the time at which the file was created.
- I have posted this file: `words.dat`
- Our problem is to write a program that reads two 5-letter words and plays the word ladders game.

Example Run

Enter the first five-letter word for the Ladders game (0 to quit): about
Enter the second five-letter word for the Ladders game (0 to quit): house
Here is the word chain from about to house:
about
abort
aport
sport
spurt
spurs
sours
tours
touts
routs
route
rouse
house

High-level Plan

- **Step 1:**
  - We will build a *word network*. This is the collection of 5-letter words with pairs of words that differ in exactly letter, connected by an "edge."
- **Step 2:**
  - Given a pair of words `word1` and `word2`, we will "explore" the word network, starting at `word1` and try to find a path in the network from `word1` to `word2`.
- We will work on Step 2 later. We will first solve Step 1 in a couple of ways. First using lists and then using *dictionaries*.

Word Network Examples

- The word “house” is connected by an edge to each of these seven 5-letter words:
  - douse, horse, louse, mouse, rouse, souse, youse.
- The word “fails” is connected by an edge to each of these fourteen 5-letter words:
  - bails, fairs, falls, foils, hails, jails, mails, nails, pails, rails, sails, tails, vails, wails.

Building the Word Network: More Details

1. We will write a boolean function called `areNeighbors` that takes two words and determines if they are "neighbors" (i.e., differ in exactly one letter).

2. We will write code to read from the file of 5-letter words (called `words.dat`) and store the words in a word list.

3. We will then build the word network.

Building the Word Network: More Details

- We will use a list called `wordList` to store the list of words (in the order in which we read them from `words.dat`).

- We will create an additional data structure – a list of lists, called `neighborsList` such that if a word `w` occurs in `wordList` in position *i* then all its neighbors are stored as a list, in position *i*, in `neighborsList`.

- For example, "fails" appears in position 1,622 in `wordList`. So `neighborsList[1622]` equals `['bails', 'fairs', 'falls', 'foils', 'hails', 'jails', 'mails', 'nails', 'pails', 'rails', 'sails', 'tails', 'vails', 'wails']`

The function areNeighbors

```
# Two words are neighbors if they differ in exactly on letter.
# This function returns True if a given pair of words are neighbors
# It is assumed that the two words have the same length.
def areNeighbors(w1, w2):
    count = 0
    for i in range(len(w1)):
        if w1[i] != w2[i]:
            count = count + 1

    return count == 1
```

Building wordList

```
# Main program
fin = open("words.dat", "r")

# Loop to read words from the and to insert them in a list
wordList = []
for word in fin:
    newWord = word.strip("\n")
    wordList.append(newWord)

fin.close()
```

Building neighborsList

```
neighborList = [ [nbr
                    for nbr in wordList if areNeighbors(word, nbr)
                  ]
                   for word in wordList
                ]
```