CS 202 – Data Structures (Spring 2007)

Homework Assignment 5: Word Morphing

Assigned on 9 May, 2007 — Due by 23:00:00 on Thursday 31 May, 2007
Graded by Uraz Türker (urazc@. . .)

PLEASE NOTE:

- The deadline is hard: No exceptions will be made.
- Solutions must be your own: No cooperation is permitted.
- Late solutions will be penalised by 10 points for each day of delay, but solutions that are late by more than 2 days will get 0 points.
- Solutions must be submitted via the webct server (whose clock may differ from yours): No other method of submission will be accepted.

1 The Problem

Given a set of words, called the *lexicon*, and two words w and v, can one transform w into v by a sequence of one-character substitutions, one-character insertions, and one-character deletions, while always getting a word of the lexicon? If yes, what is a shortest such morphing sequence?

Example 1 In the English language, one can morph *bleed* into *blood* by going through a sequence of such transformations giving the sequence of words *bleed*, *blend*, *blond*, and *blood*.

Example 2 In the English language, one can morph *can* into *pale* by going through the sequence of words *can*, *man*, *mane*, *male*, and *pale*.

2 The Input and the Output

Read the lexicon from a text file called `words.txt`. The words will not be in any particular order and may contain special characters such as the hyphen (-) or apostrophe ('). Once this file is read and processed, enter the following loop:

1. Read two space-separated words w and v from standard input. For Example 2, this input would be the following:

```
can pale
```

2. If w starts with a star (*), then exit the loop and terminate the program.

3. If w or v is not in the lexicon, then write "`Word(s) not both in lexicon!`" to standard output and return to step 1.

4. If w cannot be morphed into v, then write "`Morphing impossible!`" to standard output and return to step 1.

5. Write a *shortest* sequence of transformations of w into v to standard output in the following way: first print w on one line, and then print each remaining word on one line along with its transformation from the previous word. For Example 2, the output would be the following, given a suitable English lexicon:

```
can
man (change c at position 1 to m)
mane (insert e after position 3)
male (change n at position 3 to l)
pale (change m at position 1 to l)
```

   For deletions, adapt the following transformation: "`(delete m at position 4)`".

6. Return to step 1.

Construct your own `words.txt` file to test your program.

3 Work To Be Done

First, design an algorithm that efficiently solves this problem using data structures and algorithms seen in this course, and implement it. We have only specified the requirements above. There are a lot of details that now have to be decided on. **Hints:** You may want to have a suitable data structure for the lexicon; a class for suitably representing and manipulating graphs; a function for determining whether a word can be morphed into another word using just *one* insertion, deletion, or substitution; and a graph representation of such transformations between the lexicon words.

Second, write a conclusion of 250 to 500 words on this homework, in an *unformatted text file* called `analysis.txt`, containing an analysis of the asymptotic runtime complexity of your overall algorithm. Use `n^p` for n^p, `Theta` for Θ, and `Omega` for Ω. Indicate the number of words of your analysis.

Above all, have fun!

What and How to Submit When?

Take our warning on plagiarism **very seriously**. We assume that by submitting your solution, you are certifying that you are submitting your own work. Take the following steps:

1. Name the folder containing your program and analysis files as *LastnameName-StudentID-hw5*. Do not use any special Turkish characters in the folder name. Remove any executables (debug or release), as they take up too much space.

2. Compress your folder, giving *MehmetogluAli-15432-hw5.zip* for example. *Make sure it decompresses properly, reproducing your folder exactly, and actually corresponds to assignment* 5.

3. Submit this compressed file via WebCT by the deadline given on the first page.

Why Submit? Grading Rules

Your solution will be graded in the following way:

- If your program is complete and compiles without error, then you get 30 points.

- We will run an unspecified number n of tests on your program with different `words.txt` files. Your outputs must be *exactly* in the format described above, without a single additional character: otherwise your program is considered to function incorrectly. Each fully correct test result will earn you $30/n$ points.

- The style of your own code will be graded on clarity, comments, etc. This will cover 20 points. You will however not get any of these points if your programs fail in *all* the tests we perform. A nice looking but useless program is just a useless program, no matter how nice it is.

- Your algorithm analysis will be graded on clarity and correctness. This will cover the remaining 20 points.

For late submissions, we will first grade your solution as indicated above and then discount accordingly. So, assuming you got 90 points, if you were late at least one second but at most one day, then your actual grade will be 80 points; if you were late more than one day but at most two days, then your actual grade will be 70 points; if you were late more than two days, then your actual grade will be 0 points and your solution will not be graded.

Bonus Competition: The fastest program to pass *all* our n tests and earn *at least* 10 points *each* on the style and algorithm analysis will earn a 25 point bonus! All other such programs that are within 5% of the runtime performance of the winning program will earn a 20 point bonus.