CS 315 Intermediate Algorithms
Winter 2024

Assignment 5

Due 11:59 PM Thursday, Feb. 29, 2024

[10 credits]

1 Description

We want to devise a dynamic programming algorithm for the following problem: there is a string of characters which might have been a sequence of English words with all the spaces removed, and we want to find a way, if any, in which to insert spaces that separate English words. For example, *theyouthevent* could be "the you the vent", "the youth event" or "they out he vent". If the input is *theeaglehaslande*, then there's no such way; note that "the eagle has lande" does not count, because "lande" is not an English word. Your task is to implement a dynamic programming algorithm (in a bottom-up manner). Assume that the original sequence of characters have no other punctuation such as periods, no capital letters, and no proper names—that is, all the English words will be available in a dictionary file that will be provided to you.

Let the input string be $x = x_1 x_2 ... x_n$. We define the subproblem `split(i)` as that of determining whether it is possible to correctly add spaces to $x_i x_{i+1} ... x_n$. Let *dict(w)* be the function that will look up a provided word in the dictionary, and return *true* if and only if the word w is in it. A recurrence relation is given below:

$$split(i) = \begin{cases} \text{true} & \text{if } i = n+1 \\ \bigvee_{j=i}^{n} [dict(x_i x_{i+1} ... x_j) \wedge split(j+1)] & \text{otherwise} \end{cases}$$

Obviously, `split(i)` only finds out whether there's a sequence of valid English words or not. **Your program must also find and display at least one such sequence**.

The program will read a text file from standard input. For example, if you have a Java class named `dynProg`, the command `java dynProg < inSample.txt` is what you would use to run your program. The name of the dictionary file should be hardwired in the code. We will be testing your program on a file named "diction10k.txt", and your program will be tested in a directory containing that file. Please make sure to test your codes on your own before submission.

2 Sample Input

The first line of input is an integer C. This is followed by C lines, each containing a single string, representing a phrase to be tested.

```
3
```

```
theyouthevent
theeaglehaslande
lukelucklikeslakeslukeducklikeslakeslukelucklickslakesluckducklickslakes
```

3 Sample Output

```
phrase 1
theyouthevent

output 1
YES, can split.
the you the vent

phrase 2
theeaglehaslande

output 2
NO, cannot split.

phrase 3
lukelucklikeslakeslukeducklikeslakeslukelucklickslakesluckducklickslakes

output 3
YES, can split.
luke luck likes lakes luke duck likes lakes luke luck licks lakes luck duck licks lakes
```

4 Submission

Submit a copy of your Java, Python, C, or C++ program via Canvas. Please submit your work as a *single* source code file (i.e., not as a zipped file or zipped directory). Providing a simple instruction about how to run your code could be helpful, and if you choose to do so, please leave this instruction as a note with your Canvas submission. Note that you will find some test cases on Canvas, but we often do not publish *all* the test cases used for grading before finishing the grading.