

Assignment 6

Due 11:59 PM Thursday, March 3, 2022

[15 “normal” credits + 15 extra credits]

1 Description

We want to devise algorithms 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 valid English words. For example, *theyouthevent* could be from “the you the vent”, “the youth event” or “they out he vent”. If the input is *theeaglehaslande*, then there’s no such way. Your task is to implement a dynamic programming algorithm and a recursive invocation algorithm. You will get the “normal” credits by implementing (any) one of the two ways, and you will get both the “normal” credits and the extra credits by implementing both of the two ways:

- iterative bottom-up version
- recursive memoized version (i.e., with memory)

Assume that the original sequence of words had no other punctuation (such as periods), no capital letters, and no proper names—all the words will be available in a dictionary file that will be provided to you.

Let the input string be $x = x_1x_2\dots x_n$. We define the subproblem `split(i)` as that of determining whether it is possible to correctly add spaces to $x_ix_{i+1}\dots x_n$. 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:

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

Obviously, `split(i)` only finds out whether there’s a sequence of valid words or not. **Your program must also find 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. Testing will be much simpler if you can submit your program as a single file (and not a zipped directory).

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 number: 1
theyouthevent
```

```
iterative attempt:
YES, can be split
the you the vent
```

```
memoized attempt:
YES, can be split
the you the vent
```

```
phrase number: 2
theeaglehaslande
```

```
iterative attempt:
NO, cannot be split
```

```
memoized attempt:
NO, cannot be split
```

```
phrase number: 3
lukelucklikeslakeslukeducklikeslakeslukelucklickslakesluckducklickslakes
```

```
iterative attempt:
YES, can be split
luke luck likes lakes luke duck likes lakes luke luck licks lakes luck duck licks lakes
```

```
memoized attempt:
YES, can be 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.