Analysis of Algorithms

Programming Project 3 – Dynamic Programming

In this project, we are given a string, and we want to determine if this string can be "split" into one or more words such that each of the words appear in the book Alice in Wonderland. The words of the book are provided in this project in a text file. When splitting a word, you are not allowed to drop any characters or rearrange the order of the characters. You can split the word as many times as you need so that each of the words from your split are in Alice and Wonderland. If the input string is a word in the book, then you do not need to split it, you can accept the input string as a word itself. Your output should say how many words you can split it into, and print out the resulting words. If there are many ways to split it into words, you should output the minimum number of splits needed.

There are several options for storing the word list, but I'd recommend using hashing. If you are using Java, you can do this with a HashSet<String>. If using Python, you can use a dictionary. If using C, you could use your HashTable implementation from data structures. You could also use a binary search tree, but this would be a bit less efficient if you used a very large dictionary like a Scrabble word list.

Examples:

- Input: "aliceinwonderland".
  - Output: "aliceinwonderland can be split into 3 AiW words: alice in wonderland"
- Input: "suddenly".
  - Output: "suddenly can be split into 1 AiW word: suddenly"
- Input: "alicee"
  - Output: "alicee cannot be split into AiW words."

Note that in the last example, you can split the string into "alice e". "alice" is in the word list, but "e" is not, and so this is an invalid cutting. Note that in the first example, another solution would be "alice in wonder land" but that uses one more word than the optimal solution that was given above.

How I suggest you set up your dynamic programming table/subproblems:

- Let n be the length of the input string.
- c[i][j] is an nxn, two-dimensional array. Then c[i][j] stores the minimum number of cuts needed to cut the substring starting on the ith character and ending on the jth character into words. If it is not possible to cut, then assign it a value that is infeasible, (e.g., -1). Any time you see that value in the array, that would mean that you cannot split that substring into words.

- So in the first example above, c[5][12] would correspond to the substring “inwonder”. Then you want to set c[5][12] to be 2 because this can be split into “in wonder”.
- In the first example, c[6][10] would correspond to the substring “nwond”. This cannot be split into words in the word list, and therefore could set c[6][10] to be -1.
- The base cases could be c[i][j] = 1 if the substring corresponding to i and j is in the word list, and c[i][i] = 0 if the ith character is not a word in the word list (i.e., the subproblem has only one character in it, but that character is not a word in the word list).
- Then the trick is to figure out how to define c[i][j] as a function of its subproblems if the substring has more than 1 character in it and that substring is not in the word list itself.

If you do not implement a dynamic programming algorithm for this project, you will get a 0. Unlike the previous project, if you do not use dynamic programming then your solution is very unlikely to be correct for all possible inputs. You can implement your solution top-down or bottom-up, although I’d encourage you to try implementing it bottom-up as this tends to be a bit faster of an algorithm.

How to code this project:

For the most part, you can code this project in any language you want provided that the language is supported on the Fox servers at UTSA. We want you to use a language that you are very comfortable with so that implementation issues do not prevent you from accomplishing the project. That said, we will be compiling and running your code on the Fox servers, so you need to pick a language that is already installed on those machines. See the PDFs on Blackboard in the Programming Projects folder for more information on how to connect to the Fox servers remotely (also covered in the Programming Project 1 overview lecture).

Since everyone is coding in their own preferred language, we are asking you to provide a bash script named *project3.sh* that will act similarly to a makefile. I covered how bash scripts work in class in the Project 1 overview lecture, and I recorded a short follow-up to this here: https://youtu.be/CaIFJWiyU_U

In short, your bash script should contain the command to compile your code, and then on a different line, it should contain the line to execute your code. In this project, we will only be using the items.csv data file, so this file name can be hard coded inside your program. No command line arguments are needed for this project. So, the command to execute your code should look like this:

bash project3.sh

The output should say: “Optimal value: *solutionValue”* on one line (where solutionValue is the sum of the values of the items you are selecting), and then on each line you should print the name of the items that you selected. See the included output file for specifics.

Files provided in the project:

Since you are programming in different languages, we are providing no source files for your code. The word list is a simple text file with one word per line. Your output can just be written to standard output (e.g., printf(), System.out.println(), print(), etc).

Grading

We will grade according to the rubric provided on Blackboard. The majority of the points come from the following: did the student give a correct dynamic programming implementation that runs in O(n^2) time and returns the correct answers for all possible inputs? Proper documentation may help the grader understand your code and earn you partial credit in the event you have some mistakes in the code.

Violations of the UTSA Student Code of Conduct will be penalized harshly. In particular, be very careful about sending code to a student who asks how you accomplished a particular task. I've heard this story several times recently: "They said they just wanted to see how to perform part X of the project. I didn't think they would submit my exact code." If this happens, you will both be penalized for cheating. To protect yourself and to more properly help your fellow student, send pseudocode, and not actual compilable code.

Also we know about the online sites where people upload projects and have a third party complete the project for you. This is a particularly egregious form of cheating (it's in the best interest of your career to not tolerate this). If you use a solution from one of these sites or submit a minor modification (minor is at the disgression of the instructor) of a solution from one of these sites, you will receive a 0 and will be reported to the university for a viloation of the UTSA Student Code of Conduct.

Submitting

Zip up your project folder and submit on the dropbox on Blackboard by the due date.