Advanced Data Structures (ADS-MIRI): 4-Word Games

This assignment is quite different from the previous assignments, as its purpose is to develop a small application instead of conducting an experimental study of some data structure(s).

The program you will write will be able to play two games: *Word Challenge* and *Wordle*. Both games will use a "dictionary" of English words of length ≥ 3 as given by the file `dictionary.txt`. The file is sorted alphabetically. For each line we have a word and its frequency count (obtained from `kaggle.com`).

Word Challenge

In *Word Challenge* the user gives a (multi)set of up to 17 letters, and the program produces all words which can be written using a subset of the given letters. For example, if the user gives the letters {`E`,`T`,`F`,`H`,`R`,`R`,`E`,`O`,`E`} the program will write by increasing length and then in alphabetic order all the words that can be built using (some) of these letters, for example, `FOR`, `HER`, `ORE`, `THE`, . . . , `HERE`, . . . , `THEREFORE`. The dictionary does only contain words of length ≥ 3, hence if given $\ell \geq 3$ the list of results starts with words of length 3 and ends with words of length ℓ (or smaller, if there were no words of length ℓ using all given letters).

Besides being able to play interactively with the user, the program must give an "automatic mode" in which the program does the following repeatedly:

1. Picks a random word from the dictionary of given length ℓ
2. Rearranges its letters randomly
3. Supplies these letters to the function that generates the list of words which can be built from the given letters

In automatic mode the user gives the number of times that the game will be played, and the length ℓ, and it outputs the average number of words found and the average CPU time that the program takes to "solve" a word of length ℓ.

Wordle

In *Wordle* (`https://en.wikipedia.org/wiki/Wordle`) the user can choose to play as *keeper* or as *guesser*. The length ℓ of the secret word is defined at the beginning of the game. When playing as guesser, the program chooses a secret word of length ℓ and for each round the player writes a valid word of length ℓ and the computer outputs a string of ℓ digits with the following meaning: 0=the corresponding letter does not occur in the secret word, 1=the corresponding letter occurs in the secret word but not at that position, and 2=the corresponding letter occurs at that position. For example, if the secret word were `WORDS` and the current guess were `WHERE`, the program should output `20010`, the `W` was correctly guessed and the `R` is on the secret word, but it is not in the fourth position. The game ends when the secret word is correctly guessed or some `MAX_GUESSES` bound attained. When the player takes the rôle of the *keeper*, the computer will make the guesses and the user will give the answers (the strings of 0s, 1s, and 2's).

Finally the program should offer an automatic mode in which the computer plays both as keeper and guesser (without cheating, the guesser function does not have access to the secret word, only its length and the history so far of guesses and answers). For the automatic mode, the human user fixes a length ℓ and a (large) number of games to be played. For each game, the keeper chooses a random word of length ℓ and a game is played between the keeper and the guesser. At the end, the program outputs the average number of rounds needed by the guesser to guess each secret word and the average CPU time to do it.

Final considerations

Prepare a report that explains the algorithms and data structures behind both games. Give your explanations at a high level, avoid giving low level details. It is not necessary that you describe the parts of the program related to input/output and bookkeeping tasks. Things (data structures and/or algorithms) which are common to both games need to be described only once, for example, how to choose a random word of given length from the dictionary.

The report shall also give the results of the benchmarks, for example the performance of *WordChallenge* as a function of the length ℓ of the input, or the average number of guesses to find the secret word in *Wordle*.

We encourage you to use LaTeX to prepare your report. For the plots you can use any of the multiple packages that LaTeX has (in particular, the bundle TikZ+PGF) or use independent software such as gnuplot and then include the images/PDF plots thus generated into your document.

Submit your work using the FIB-Racó. It must consist of a zip or tar file containing all your source code, auxiliary files and your report in PDF format. Include a README file that briefly describes the contents of the zip/-tar file and gives instructions on how to produce an executable program and reproduce the experiments. The PDF file with your report must be called

`YourLastName_YourFirstName-4-wordgames.pdf`, and the zip/tar file must be called `YourLastName_YourFirstName-4-wordgames.zip` (or `.tar`).