Scrabble Probability Project

MAT 258 — Summer 2021

Due Date: Monday, July 12

The Scrabble Probability Project should follow these guidelines:

Deliverables to be submitted in one zipped folder:

- C++ code files
- README.txt
- Document Explaining Logic
- Sample output

Students should write C++ command line code which computes the probability that a random selection of seven tiles from the scrabble bag will contain a seven letter word. The C++ code should not require any other special libraries or dll's. The README.txt file should indicate any compile guidelines, sample command line compile and run commands, or any other related explanations. The Document should give detailed explanation of the logic which is used to generate any intermediate files, and all steps in arriving at the final probability. Sample output should give enough information to demonstrate that the problem is solved.

Further Notes:

The 100 scrabble tiles and their frequency numbers are give as one text string on the website. There are two blanks, indicated by an underscore. The blank is a wild card. There is also a list of seven letter words on the website, which must be used to compute the probability.

If all the scrabble tiles are labelled (for instance: E1, E2, ..., E12, etc.) then all the tiles would be physically different, or distinguishable. Then the total number of possible draws of seven tiles would be the same as the number of subsets of size 7 taken from 100 distinct objects. This is the binomial coefficient "100 choose 7". Your goal is to count all of the times that such a draw results in at least one seven letter word. However, it is not efficient to loop through all of these. This would be the worst way to to try to solve the problem, even though it could be technically correct. For instance, even if we ignore the time spent creating the list of possible draws, and we reduced the computation time to look up each combination and see if it contains a seven letter word to one millionth of a second, this process would still take more than four hours to run.

A better approach is start instead with the list of seven letter words and follow as many shortcuts as possible. A first step could be to alphabetize each seven letter word and remove duplicates. If any draw from the scrabble bag is first alphabetized, then we only need to know whether it produces one seven letter word. For each resulting alphabetized string of length seven, we need to compute the number of labelled strings associated with this. There are two cases: 1) the string requires blanks, 2) the string does not require blanks. To decide which case a given string falls into, you need to answer the question: Can this string be achieved without blanks? For example, if the string contains more than one Z then it can only be achieved with at least one blank. This preprocessing step is important since it helps to avoid overcounting. If a string does not require blanks then counting the number of such labelled strings without blanks is straightforward, since it is just a product of numbers or binomial coefficients based on how many of each type of letter there are in the string and in the bag. Next, the number of such strings with blanks should also be computed. Here it is important to check what happens with one blank or two. For example, if a word contains three Z's then it is only possible to produce it with two blanks, but if a word contains exactly two Z's, then it is possible to produce it with one blank (and one Z) or two blanks.