Morse Code Write-Up

David Ericson

Message: If you can read this the code is working. Well done and party on. (Without periods)

Algorithm Explanation

The algorithm starts by obtaining all of the possible first words. I needed to take the morse code string and use the getCodes() function of the dictionary for every single substring of the morse code. What I mean by obtaining a substring of the morse code is that I run through with the morse code of let’s say …--., the next time through it will be …--, then …-, etc. until the string is gone. I made a function for completing this because I knew I would have to do it quite a bit. This gives me all of the words possible for the first word. I then create a for loop which goes through all the possible first words and creates a list of ToGo words from it, using a substring of the original message so that it doesn’t start from the beginning. I then call a helper function that recurses through all of the possible ToGo words and creates a new set of ToGo words for the chosen word, causing fairly heavy branching. Every time the recursive function is called it checks to see if the last word added meets the required frequency to pass as something that might occur in the English language. The total frequency is also checked to see if it is above a small threshold. The total frequency is the multiplication of the frequencies of the first word with the second word, the second word with the third word, etc. divided by 10000. This eliminates fairly lengthy sentences with minimal frequency words. Once there are no more words in the ToGo list we add the entire SoFar list to the Successes list. This recursive function is called several times until all of the original starting words have been used. We then return the list. Finally, that list is sorted by total frequency (highest to lowest) and is cut down to the top twenty entries.

Help

I received help from Drew Hurdle a decent amount at the start of the project, as well as when it came to sort the array of values. He also helped me with the mathematical analysis below.

Mathematical Analysis

T(n) = 32n\*T(n-1)+c ; T(0) = d So basically if we take the upper bound of getWordsForCode(String morseCode) then we find that at worst there are 32 different words that share the same set of morse code. Since we run through this loop n times we get 32\*n. We then send that down our recursive function call which loops around once again. Since the number of calculations being made is simply a constant we don’t really need it, thus c. Also the base case is 0 since we end when there are no more messages left in the morse code string. Now we simply solve.

T(n-1) = 32(n-1)\*T(n-2)+c

T(n-2) = 32(n-2)\*T(n-3)+c

T(n) = 32n\*(32(n-1)\*(32(n-2)\*T(n-3)+c)+c)+c

T(n) = 32(n!-(n-k)!)\*T(n-k)+kc

k = n

T(n) = 32n!\*T(0)+nc

T(n) = 32n!\*d+nc

0(n!)

The equation is classified as factorial growth.

Empirical Analysis

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Real Message | Length | Time to run (millis) | Total Frequency (multiplied by 10,000) | Number of branches tried | Alternative Funny Message |
| I am your father | 34 | 259 | .209 | 10,016 | So your father |
| You can’t handle the truth | 53 | 772 | 2.51 | 61,824 | You can’t handle the cube |
| Life is like a box of chocolates | 76 | 976 | 1.86 | 51,537 | Life is like a bone to earth at me no legs |
| Show me the money | 34 | 300 | .17 | 19,624 | N/A (none of them were really that funny) |
| If you can read this the code is working. Well done and party on | 132 | 6,844 | 2.26\*10^-7 | 486,477 | N/A |