Aruj Jain (aj2933) and Benjamin Magid (bmm2179)

WordLadder

28th November 2022

TASK

Word Ladder is a word game where the player is given two words, and they are tasked with finding a sequence of words to connect the two, only changing one letter at a time. The result is a sequence of words -each valid according to a given dictionary- that is only one letter-modification different from the previous word in the sequence.

In homework 4, we implemented a BFS-style algorithm to solve Word Ladder with the constraint that the beginning and end words were of equal length. In order to make the problem more interesting, we aim to design a Word Ladder solver which can take two words of different lengths and produce the shortest sequence that connects them. This increases the branching factor of the solution from homework 4 as, given a word, we can modify any letter, add a letter, or delete a letter

> *Ex) Given `door` and `store`, we expect the following WordLadder:*
>
> *door, moor, moot, soot, sort, sore, store*

As in homework 4, we will only look at Word Ladders of length 20 or shorter. In other words, if we cannot find a sequence of dictionary words that connect the starting and ending words with 20 or fewer intermediaries, we will assume no such Word Ladder exists in the dictionary.

GOALS

1. Create a fast, serial implementation to solve this harder version of Word Ladder.
    a. Implement a BFS-style algorithm to generate all the valid words in a 20-rung ladder, using a set to avoid exploring already-explored words.
    b. Use ByteString (because we will only use ASCII characters), and can therefore represent strings more efficiently instead of lazily-evaluated strings.
    c. Consider bi-directional BFS searching to reduce the branching factor. This means running BFS both from the start and end words.
2. Create an even faster, parallel implementation that may parallelize
    a. The generation of the child nodes for a given word (all valid changes to that word by modifying, deleting, or adding a letter).
    b. The exploration of all nodes at a given level, checking all the potential n^{th} words in the sequence.
    c. The bi-directional BFS which runs the algorithm simultaneously from both the start and end words, with the goal of both BFS processes finding the same word allowing the search to stop.