Bidirectional BFS

Eckel, TJHSST AI1, Fall 2022

Background & Explanation

This assignment modifies BFS into Bidirectional BFS, an algorithm that conceptually isn't that different but in some situations makes an enormous difference in runtime. You'll apply BiBFS to both Sliding Puzzles and Word Ladders, and identify when a situation is a good fit for applying this algorithm.

Bidirectional BFS

The basic idea is to add another algorithm that applies BFS principles from both the source *and* the goal. As usual, the submission link is on the course website.

Bidirectional BFS is not a terribly complex idea, in theory. Your goal is to write a new method that runs a BFS search from both sides simultaneously, creating a visited set and fringe from the source state and a different visited set and fringe from the goal state, alternating adding to each. As you add to each fringe, see if each state is in the visited set from the other direction. If a state you generate from the start side is in the visited set from the goal side, or vice versa, you've found the shortest path!

As you might expect, though, there are a number of subtleties here, particularly when you have to keep track of the path. Think carefully about how all this will work before coding.

Required Tasks

You're going to apply BiBFS to both Sliding Puzzles and Word Ladders, but you're only going to submit code to run Sliding Puzzles. The rest will be in a report to me on Mattermost. Here's what you need to do:

1) Return to the sample run shown in Sliding Puzzles: BFS. Recreate this sample run on `slide_puzzle_tests.txt`, except that each puzzle should produce two lines of output – one showing a BFS search, and another showing a Bidirectional BFS search. Be sure to time them both so you can see the difference. This is the code you will submit, but I'd like you to send me a message about it anyway; what speed gains are you seeing?
2) Return to the benchmark you did for question 5 on Sliding Puzzles: BFS using the `15_puzzles.txt` file. Repeat the exercise using Bidirectional BFS. What's the last puzzle you can solve in less than a minute? How does this compare to your BFS results?
3) Return Word Ladders, and this time find all of the ladders using Bidirectional BFS instead of BFS. Find the total runtime for the whole sample puzzle file using each strategy. What speed gains did you get?
4) How did you store the word ladder during BiBFS, not just the length? Send me code & an explanation.
5) Finally, from your results in 1, 2, and 3, and your own judgment & common sense, explain where Bidirectional BFS is most useful as an alternative to BFS.

Specification

Submit a single Python script to the link on the course website.

This assignment is **complete** if:

- You follow the instructions on the submission form to format your submission properly.
- You sent your answers to questions 1-5 above and I OKed them.
- Your code meets the full spec for Sliding Puzzles: BFS, but adds a second line of output for each puzzle as described in question 1 above.