CS418 – P01 Puzzle Uninformed Search

Description:

- This project consists of writing two separate, but closely related programs.

- Each program must find a sequence of moves ("moving" blank space down, left, up, right) for solving the N x N tile puzzle given any legal starting configuration and any specified goal configuration. (Read the tile puzzle problem description in your textbook.)

- One program must implement a **depth-first-based** algorithm and the other program must implement a **breadth-first-based** algorithm.
    - Use the depth-first and breadth-first algorithms in your textbook as the basis for your implementation.
    - In both programs apply the applicable operator as follows: (1) down (2) left (3) up (4) right.
    - For the depth-first search, **you need only find the first solution**; you neither need to find all solutions nor the optimal (least number of moves) solution.

- Each program should print the solution as a series of states and the moves used.
    - The moves must be printed in between each state.
    - The moves names are **down, left, up,** and **right**; representing the blank tile movement from one state to the next. Of course, one cannot move a non-existent blank tile. This simply makes it easier to implement in code.
    - One could add an interface to indicate which tile (by number) is shifted.
    - These moves should be tried from each state in this order: **1) down; 2) left; 3) up and 4) right.** Some are not applicable and therefore a next state (child state) is not generated. In terms of search trees (graphs) the children (adjacent nodes) generated left to right respectively correspond to (leftmost) down, left, up and (rightmost) right.

- Do not wait until a couple of days before the due date to work on this project. Even if you successfully solve the problem, the required task of testing will take a great deal of time. Some problem instances can require complete CPU utilization (> 99% usage by the process representing the execution of your program) for several hours (>5 hours).

CS418 – P01 Puzzle Uninformed Search

Input-Output Specifications:

- You must acquire:
  1) the size, N of the N x N tile puzzle;
  2) the start state (configuration)
  3) the goal state (configuration).

- The start and goal states should be entered in row-major order: the user enters the N x N tile puzzle row by row.

- Use the numbers between 1 and N-1 for labeling the tiles and use 0 for the blank space in the tile and the constraint that the numbers (1..N-1) are all used once and only once must be enforced.

- Your program interface specification for input is as follows.
  - **Boldface** prompts are what your program should print to the console.
  - *Italicized* represents integer literals typed by the user of your program.

  Enter size of tile puzzle (integer greater that 0): *n*
  Enter start state row by row (numbers delimited by white space):
  X00 X01 … X0(n-1)
  X10 X11 … X1(n-1)
  …
  X(n-1)0 X(n-1)1 … X(n-1)(n-1)

  Enter goal state row by row (numbers delimited by white space):
  Z00 Z01 … Z0(n-1)
  Z10 Z11 … Z1(n-1)
  …
  Z(n-1)0 Z(n-1)1 … Z(n-1)(n-1)

- Your program interface specification for output must print the string Solution: on the console And then print the series of puzzle configurations with the operator (left, right, up, or down) used in between configurations with the goal configuration last.

  Solution:

  x00 x01 … x0(n-1)
  x10 x11 … x1(n-1)
  …
  x(n-1)0 x(n-1)1 … x(n-1)(n-1)

  operator name

  y00 y01 … y0(n-1)
  y10 y11 … y1(n-1)
  …
  y(n-1)0 y(n-1)1 … y(n-1)(n-1)

  operator name

  etc. until goal is found.

Problem Instance Example:

- Suppose that the start and goal states of a 3 x 3 tile puzzle problem instance is as follows:

Start State:

1	3	5
4	2	▨
7	8	6

Goal State

1	2	3
4	5	6
7	8	▨

- For this problem instance see ***BFS-DLUR Sample Run*** document.

What To Turn In

1) Create a zip file for your Depth-First Search solution and create a separate zip file for your Breadth-First Search solution. Name these two zip files as follows:
   Depth-First solution: ***P01DFyourLastName.zip***
   Breadth-First solution: ***P01BFyourLastName.zip***

   Make sure you the above zip files contain the entire project – not just the source code. Include Word document (.docx) showing console screen captures of sample problem instances, including the ones specified in this document.

2) After creating above files. Zip all files into one file. Name this file: ***P01yourLastName.zip.***

3) Submit the single zip file in the previous step to Blackboard.

4) A hard-copy of well-documented source code. The code must include proper use and implementation of appropriate data structures.