Names: Samantha Rehome
Robert Olivares

Programming Project 1 Report

1. **Time Complexity**

   BFS Algorithm:

   - Loop for BFS algorithm:
     - Loop 'approximately the number of visited nodes' times:
       $O(1 + b + b^2 + \ldots + b^s) = O(b^s)$
   - Within the loop:
     - Check Visited Set for node: (Worst case w/unordered set):
       - $O(1 + b + b^2 + \ldots + b^s) = O(b^s)$
     - Check Frontier Queue for node: (Iterate through queue w/for loop)
       - $O(b^s)$
     - Total Complexity Within the Loop:
       - $O(b^s) + O(b^s) = 2O(b^s) = O(b^s)$
   - Average branching factor:
     - 1 Start corner:
       - 3
     - 4(n-2) Sides:
       - 2 if traveling along
     - 2 Corners (goal node not expanded):
       - 0 (node leading to corner already expanded surrounding nodes)
     - $(n-2)^2$ Middle Squares:
       - 3 if traveling horizontally/vertically
       - 5 if traveling diagonally
       - Average ~4
     - Avg ~= $\frac{1 \cdot 3 + 4(n-2) \cdot 2 + 2 \cdot 0 + (n-2)^2 \cdot 4}{(n^2 - 1)}$

       $\frac{3 + 8n - 16 + 4n^2 - 16n + 16}{(n^2 - 1)}$

       $\frac{4n^2 - 8n + 3}{(n^2 - 1)}$

       For our 8 x 8 grid: Average branching factor ~= 3.1
   - Algorithm Time Complexity:
     - $O(b^s) * O(b^s) = O((b^s)^2) =$ **$O(b^s) = O(3.1^s)$**

2. **Space Complexity**

   BFS ALgorithm: n x n grid

   - Color grid:
     - $O(n^2)$
   - Visited Set:
     - $O(1 + b + b^2 + \ldots + b^s) = O(b^s)$
   - Frontier Queue:
     - $O(b^s)$
   - Average branching factor:

1 Start corner:

3

4(n-2) Sides:

2 if traveling along

2 Corners (goal node not expanded):

0 (node leading to corner already expanded surrounding nodes)

$(n-2)^2$ Middle Squares:

3 if traveling horizontally/vertically

5 if traveling diagonally

Average ~4

Avg ~= $\frac{1 \cdot 3 + 4(n-2) \cdot 2 + 2 \cdot 0 + (n-2)^2 \cdot 4}{(n^2-1)}$

$\frac{3 + 8n - 16 + 4n^2 - 16n + 16}{(n^2-1)}$

$\frac{4n^2 - 8n + 3}{(n^2-1)}$

For our 8 x 8 grid: Average branching factor ~= 3.1

Algorithm Space Complexity:

$O(b^s) + O(b^s) = 2O(b^s) =$ **$O(b^s)$ ~= $O(3.1^s)$**

Program Space Complexity:

$O(n^2) + O(b^s) = O(n^2 + b^s) =$ **$O(b^s)$ ~= $O(3.1^s)$**

3. **Algorithms used and why**
   We technically tried both the DFS and BFS algorithms to find the shortest colored path in our grid, but ultimately, the BFS algorithm is the algorithm we choose to use for our search because the DFS algorithm finds the leftmost path to the goal, which is not always the shortest when several blocks of the same color are clustered together. The BFS algorithm is considered an optimal algorithm when all the path costs are uniform, and since the cost of moving from square to square is always considered to be equal to 1, BFS can be used to find the shortest path from start to goal for each color.

4. **State space representation of your application**
   a. **Start state**
   The start state is the upper right corner of the grid, which is (0,7) in the 8x8 grid.

   b. **Goal state**
   The goal state is any same color path from the start state to the bottom left corner of the grid, which is (7,0) in the 8x8 grid.

   c. **Actions**
   The path can move in 8 potential directions: up, up-right diagonally, right, down-right diagonally, down, down-left diagonally, left, or up-left diagonally.

   d. **State space**

All possible same color paths leading from the start node in the upper right corner of the grid.

5. **Link to your GitHub repo**
   https://github.com/srehome/shortest-colorpath-bfs-dfs

6. **Resume update: Add 2 points to your resume about this project. I am not expecting your resume to be submitted; rather, mention how you would present this as your experience developing an AI application for future employment prospects.**
   Python Color Grid PathFinder Using BFS/DFS
   a. Implemented uninformed search algorithm techniques to find multiple optimal paths in a grid.
   b. Implemented a UI using python's pygame library displaying the result of a search algorithm.