Assignment Report 1 and 2: 8 puzzle problem and Breadth-First Search

CSE-0408 Summer 2021

Taskir Rahman Tasin
Department of Computer Science and Engineering
State University of Bangladesh (SUB)
Dhaka, Bangladesh
taskir.rahman72@gmail.com

Assignment Report 1 Name : 8 puzzel prolem
***Abstract*—This paper introduced for solving 8-puzzle problem using Heuristic functions.The heuristic function is a way to inform the search about the direction to a goal.**

***Index Terms*—8-puzzle, heuristic function,Anaconda Prompt(anaconda3)**

I. Introduction

The 8-puzzle problem is a puzzle invented and popularized by Noyes Palmer Chapman in the 1870s. It is played on a 3-by-3 grid with 8 square blocks labeled 1 through 8 and a blank square. Your goal is to rearrange the blocks so that they are in order. You are permitted to slide blocks horizontally or vertically into the blank square. The following shows a sequence of legal moves from an initial board position (left) to the goal position (right).

II. Literature Review

In 2012 Farhad S. et. al. [4] proposed new resolution for solving N-queens by using combination of DFS (Depth First Search) and BFS (Breadth First Search) techniques. The proposed algorithm act based on placing queens on chess board directly. The results report the performance and run time of this approach.

III. Proposed Methodology

The heuristic function h(N) 0 estimates the cost to go from STATE(N) to a goal state. A puzzle:current state [[1, 2, 4], [3, 0, 6], [7, 8, 5]] goal state: [[1, 2, 3], [4, 5, 6], [7, 8, 0]] In order for the game to be able to solve itself, we need to approach the solution with an idea of how to differentiate between a good move and a bad one. This is called a heuristic. Our heuristic was to order the moves based on the number of tiles out of place. We also took into consideration how far away the out of place tiles was from their correct position (called Manhattan distances). This entire process determines the heuristic value (H). We can then take H at any given time and successively find the path to a solution. But this will not provide us with the shortest solution.

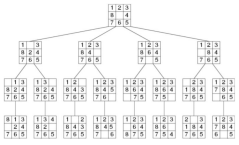

Fig. 1. Example of 8 puzzel problem.

IV. Rules For Solving The Puzzle

Instead of moving the tiles in the empty space we can visualize moving the empty space in place of the tile, basically swapping the tile with the empty space. The empty space can only move in four directions viz.
, 1. Up
2.Down
3. Right or
4. Left
The empty space cannot move diagonally and can take only one step at a time (i.e. move the empty space one position at a time)

V. Advantages

1.It is the best one from other techniques.
2.It is optimally efficient.

VI. output

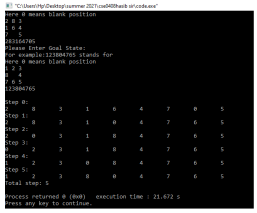

Fig. 2. Output of 8 puzzel problem.

VII. Explanation

This problem can be solved by searching for a solution, which is a sequence of actions (tile moves) that leads from the initial state to the goal state. Two possible states of the 8-puzzle. The state is a typical goal state. The state is a configuration that represents a worst case: transforming this state into the goal state requires some actions, which is the diameter of the search space. For search algorithms the problem is often to find the shortest solution, that is, one which consists of the least number of tile moves.

Acknowledgment

I would like to thank my honourable**Khan Md. Hasib Sir** for his time, generosity and critical insights into this project.

References

[1] iltaver, R., Lustrek, M., / Gams, M. (2012). The pathology of heuristic ˇ search in the 8-puzzle. Journal of Experimental Theoretical Artificial Intelligence, 24(1), 65-94.

End assignment 1

Assignment Report 2 Name: breath first search

***Abstract*—Solving problem and learn about breadth first search algorithm using c++ language.**

***Index Terms*—Nodes,Edges,source,graph,queue,visited, IDE is codeblocks.**

VIII. Introduction

Breadth First Traversal or Breadth First Search is a recursive algorithm for searching all the vertices of a graph or tree data structure. Traversal means visiting all the nodes of a graph.The breadth-first algorithm is a particular graph-search algorithm that can be applied to solve a variety of problems such as finding all the vertices reachable from a given vertex, finding if an undirected graph is connected, finding (in an unweighted graph) the shortest path from a given vertex to all other vertices, determining if a graph is bipartite, bounding the diameter of an undirected graph, partitioning graphs, and as a subroutine for finding the maximum flow in a flow network (using Ford-Fulkerson's algorithm). As with the other graph searches, BFS can be applied to both directed and undirected graphs.

IX. Literature Review

Breadth-First Search is an important kernel used by many graph-processing applications. In many of these emerging applications of BFS, such as analyzing social networks, the input graphs are low-diameter and scale-free. We propose a hybrid approach that is advantageous for low-diameter graphs, which combines a conventional top-down algorithm along with a novel bottom-up algorithm. The bottom-up algorithm can dramatically reduce the number of edges examined, which in turn accelerates the search as a whole. On a multi-socket server, our hybrid approach demonstrates speedups of 3.3 – 7.8 on a range of standard synthetic graphs and speedups of 2.4 – 4.6 on graphs from real social networks when compared to a strong baseline. We also typically double the performance of prior leading shared memory (multicore and GPU) implementations.

X. Proposed Methodology

A standard BFS implementation puts each vertex of the graph into one of two categories:
1.Visited
2.Not Visited
The purpose of the algorithm is to mark each vertex as visited while avoiding cycles. The algorithm works as follows:
1.Start by putting any one of the graph's vertices at the back of a queue.
2.Take the front item of the queue and add it to the visited list.
3.Create a list of that vertex's adjacent nodes. Add the ones which aren't in the visited list to the back of the queue.
4.Keep repeating steps 2 and 3 until the queue is empty.
The graph might have two different disconnected parts so to make sure that we cover every vertex, we can also run the BFS algorithm on every node.

A. Input

Firstly input the number of edges and number of nodes use in the graph.Then inputs are a graph(directed or undirected) G =(u, v) and a source vertex 1, where s is an element of u. The adjacency list representation of a graph is used in this analysis.

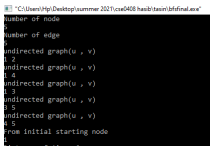

Fig. 3. Example of an input.

B. Output

The outputs are a predecessor graph, which represents the paths travelled in the BFS traversal, and a collection of distances, which represent the distance of each of the vertices from the source vertex.

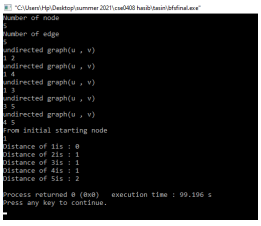

Fig. 4. Example of an output.

Fig. 5. code

XI. Advantages of Breadth First Search

1.Used to find the shortest path between vertices
2.Always finds optimal solutions.
3.There is nothing like useless path in BFS,since it searches level by level.
4.Finds the closest goal in less time

XII. Disadvantages of BFS:

1.All of the connected vertices must be stored in memory. So consumes more memory.

XIII. Explanation

Breadth first search is a general technique of traversing a graph. Breadth first search may use more memory but will always find the shortest path first. In this type of search the state space is represented in form of a tree. The solution is obtained by traversing through the tree. The nodes of the tree represent the start value or starting state, various intermediate states and the final state. In this search a queue data structure is used and it is level by level traversal. Breadth first search expands nodes in order of their distance from the root. It is a path finding algorithm that is capable of always finding the solution if one exists. The solution which is found is always the optional solution. This task is completed in a very memory intensive manner. Each node in the search tree is expanded in a breadth wise at each level.

Acknowledgment

I would like to thank my honourable**Khan Md. Hasib Sir** for his time, generosity and critical insights into this project.

References

[1] S. Beamer, K. Asanovic and D. Patterson, ”Direction-optimizing Breadth-First Search,” SC ’12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, 2012, pp. 1-10, doi: 10.1109/SC.2012.50.

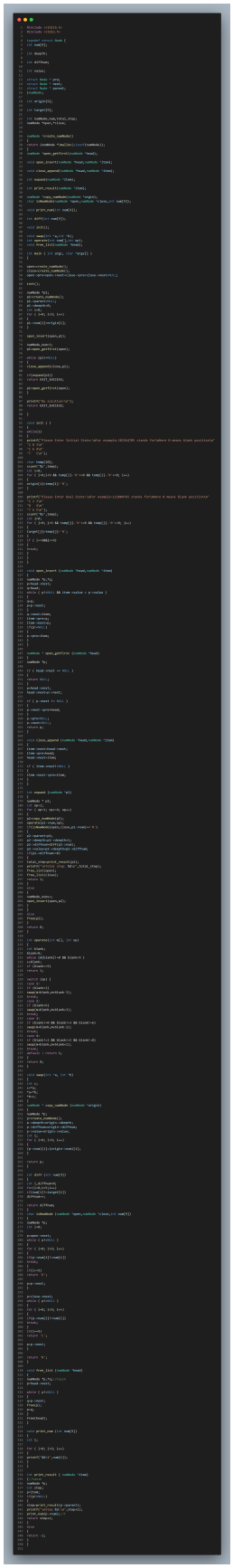

Fig. 6. code of 8 puzzel problem.