Design report

Group member:

DING Xiangyuan 4014815
XUE Yayun 4024860
ZHUMANAZAROV Magzhan 4165121

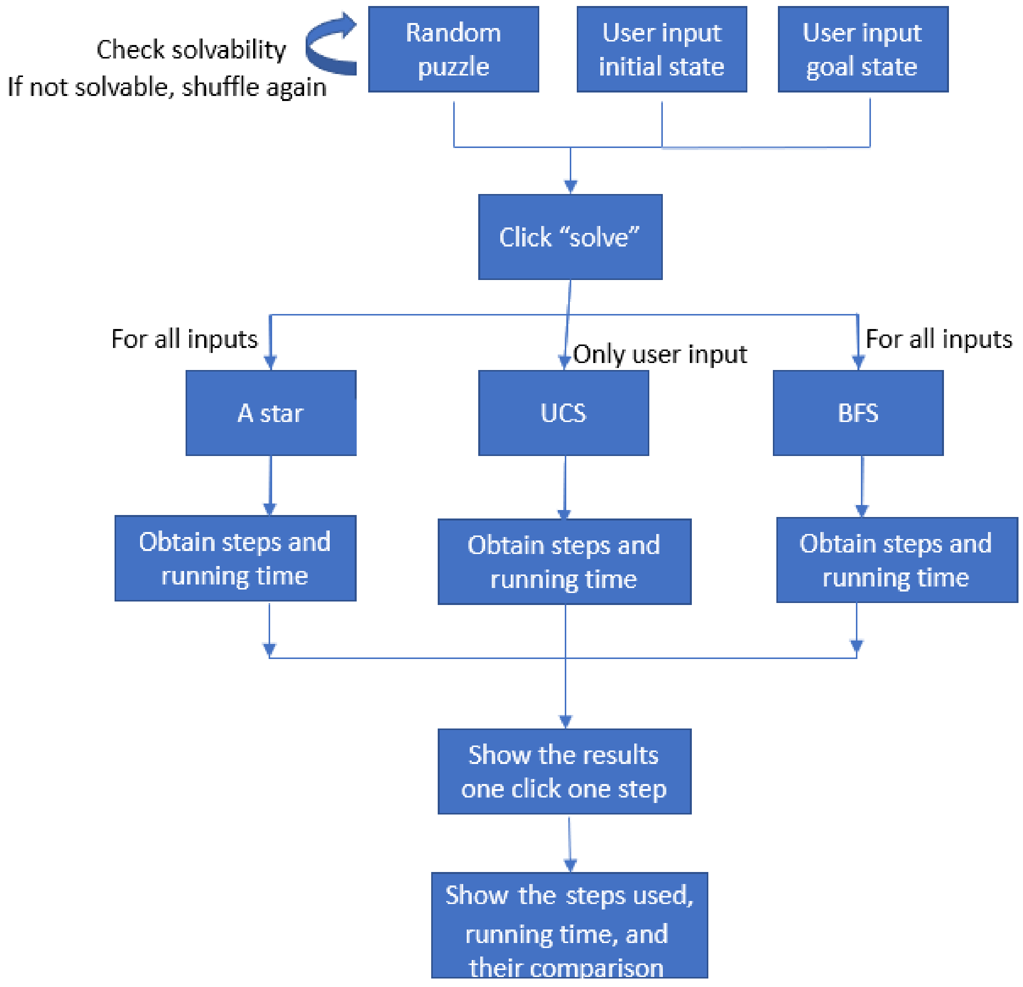
Task distribution:

DING Xiangyuan: Uniform cost search, generalization and the user interface for phase I, check solvability, final report, obtain running time and comparison

XUE Yayun: Best first search, generating random puzzle, reading puzzle from input

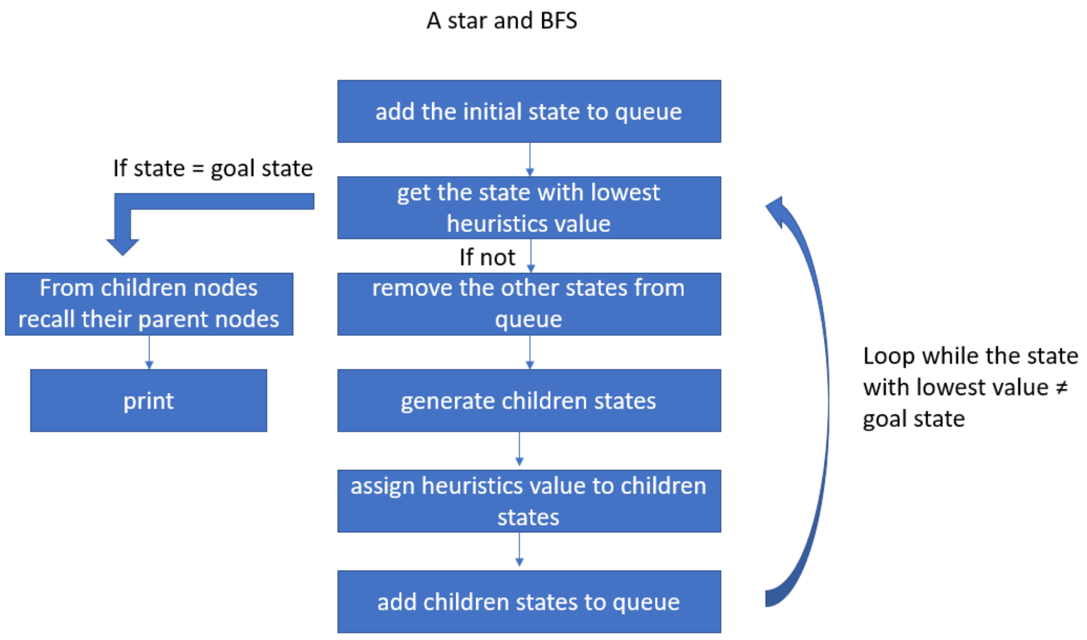
ZHUMANAZAROV Magzhan: A star search, generating random puzzle, reading puzzle from input, use GUI to output our result, generalize our program and adjust the programs, obtain running time.

Algorithm design report:

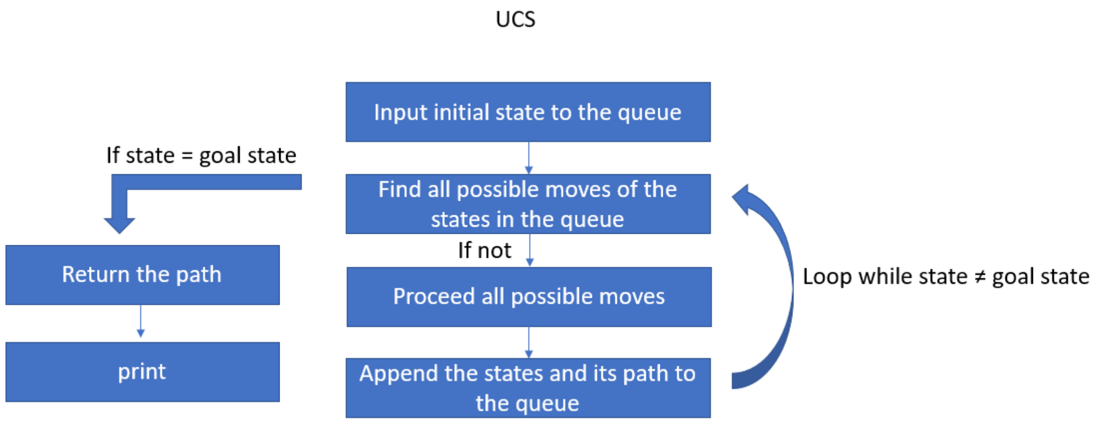

Check solvability
If not solvable, shuffle again
Random puzzle
User input initial state
User input goal state
Click "solve"
For all inputs
Only user input
For all inputs
A star
UCS
BFS
Obtain steps and running time
Obtain steps and running time
Obtain steps and running time
Show the results one click one step
Show the steps used, running time, and their comparison

The user can choose to input a puzzle or generate a random puzzle. The randomly generated puzzle will be checked to ensure that it is solvable. If it is not solvable, it will generate another one until it is solvable. The goal state must be input by users. Then, click "solve" and it will start. The BFS and A star search will run for both random puzzles and input puzzles, while the UCS will run only for user input puzzles since it may take a very long time to solve the random puzzle and may lead to not responding. After running the puzzles, the program will record the steps and running time taken. The results will not be displayed all at once. It will display one step at a time when you click the proceed button. The program will show how many steps and how much time that are taken and compare them.

This is the flow-chart of the A star search and BFS


A star and BFS
add the initial state to queue
get the state with lowest heuristics value
If state = goal state
From children nodes recall their parent nodes
print
If not
remove the other states from queue
generate children states
assign heuristics value to children states
add children states to queue
Loop while the state with lowest value ≠ goal state

This is the flow chart of UCS


UCS
Input initial state to the queue
If state = goal state
Find all possible moves of the states in the queue
Return the path
print
If not
Proceed all possible moves
Append the states and its path to the queue
Loop while state ≠ goal state

Data structures:

Here are some descriptions to our functions

test_sovability() is to test the solvability

generate_rand_puzzle() is to generate a random puzzle

detect_location() to detect the location of '#'

detect_move() to find how many possible directions can '#' move to

proceed_moves() to output a list of all possible moves given the initial state

uniform_cost_search() conduct the uniform cost search and output a list

print_ucs() is for printing the result of the uniform cost search
findnumber(): finds given number in a given state and returns its position
h(): heuristics function based on manhattan distance
solution_path(): finds the path from initial state to given state
a_star(): implements the A* star algorithm
showInfo() is to print the array in the correct form
getEmptyPos() is to detect the location of '#'
generateSubStates() to find possible directions can '#' move to
BFS() is the best first search
print_bfs() to conduct BFS and other function