CONCEPTS OF PROGRAMMING LANGUAGES COURSE (CSEN 403)

SPRING SEMESTER 2022 REPORT

OF 1ST PROJECT

TEAM NUMBER 124

This Report Contains the description of the implementation of the AI module for a minesweeper robot in Haskell 4x4 grid, screenshots of two different grid configurations with the returned solutions and the bonus optimized implementation.

Kareem Ahmed Eladl, T-16, 52-5934

Mariam Mohamed Fawzy, T-16, 52-6892

Omar Mahmoud Awad, T-14, 52-9213

Youssef Osama Kamal, T-5, 52-1907

FUNCTIONS USED IN THE IMPLEMENTATION

- type Cell = (Int , Int) represents a position on the grid with the first coordinate in the pair representing a row number, and the second coordinate representing a column number.
- data MyState = Null | S Cell [Cell] String MyState represents the state of the robot. It is either Null or the data constructor S followed by a cell representing the robot's position, a list of cells representing the positions of the mines to be collected, a string representing the last action performed to reach this state, and the last state the robot was in before doing the last performed action.
- up (S (x , y) cells s mystate) function takes as input a state and returns the state resulting from moving up from the input state. For implementing this function we considered 2 case, If up will result in going out of the boundaries of the grid, Null should be returned. We did this by checking that the cell of the Robot does not have an X-coordinated = 0, Otherwise the Robot can move.
- down (S (x , y) cells s mystate) function takes as input a state and returns the state resulting from moving down from the input state. For implementing this function we considered 2 cases, If down result in going out of the boundaries of the grid, Null should be returned. We did this by checking that the cell of the Robot does not have an X-coordinated = 3, Otherwise the Robot can move.
- left (S (x , y) cells s mystate) The function takes as input a state and returns the state resulting from moving left from the input state. For implementing this function we considered 2 cases, If left will result in going out of the boundaries of the grid, Null should be returned. We did this by checking that the cell of the Robot does not have a Y-coordinated = 0, Otherwise the Robot can move.
- right (S (x , y) cells s mystate) The function takes as input a state and returns the state resulting from moving right from the input state. For implementing this function we considered 2 cases, If right will result in going out of the boundaries of the grid, Null should be returned. We did this by checking that the cell of the Robot does not have a Y-coordinated = 3, Otherwise the Robot can move.
- collect (S (x , y) (h : t) s mystate) function takes as input a state and returns the state resulting from collecting from the input state by removing the collected mine from the list of mines to be collected. To implement this function we used 3 helper functions collectH, checkAvailable, collectHL.
- collectH (x1,y1) (x2,y2) function takes as input 2 cells and checks if they are equal
- checkAvailable l (h : t) function takes as input cell representing the robot's position donated as l , a list of cells representing the positions of the mines to be collected donated as (h : t) and checks if cell representing the robot's position occurred in the list.
- collectHL l (h : t) function takes as input cell representing the robot's position donated as l, a list of cells representing the positions of the mines to be collected donated as (h : t) and if the cell representing robot's position occurs in the list, it deletes it from the list.

- search (h : t) function takes as input a list of states. It checks if the head of the input list is a goal state, if it is a goal, it returns the head. Otherwise, it gets the next states from the state at head of the input list, and calls itself recursively with the result of concatenating the tail of the input list with the resulting next states.
- constructSolution (S (x , y) l s mystate) function takes as input a state and returns a set of strings representing actions that the robot can follow to reach the input state from the initial state. We implemented this function by concatenating the string representing the direction to the list of the resulting list each time until all last state the robot was in before be Null
- isGoal (S (x,y) l s a) function takes as input a state, returns true if the input state has no more mines to collect (the list of mines is empty, and false list of cells representing the positions of the mines to be collected is not empty
- nextH mystate function that takes as input current state and returns a list which applies up, down, left, right and collect on the current state
- removeNull (h : t) function that takes a list of myStates and remove any occurrence of Null
- nextMyStates mystate function that takes current state and returns a list in which up, down, left, right, collect are applied on the current state and remove any of them if it returns Null
- solve cell cells The function takes as input a cell representing the starting position of the robot, a set of cells representing the positions of the mines, and returns a set of strings representing actions that the robot can follow to reach a goal state from the initial state.

DIFFERENT RUNS OF THE PROJECT

```
Main> solve (2,2) [(0,1),(2,3)]
["right","collect","up","up","left","left","collect"]
Main> solve (1,0) [(0,1),(2,3)]
["up","right","collect","down","down","right","right","collect"]
Main> solve (1,0) [(3,3),(3,1)]
["down","down","right","collect","right","right","collect"]
Main> solve (1,0) [(3,3),(2,1)]
["down","right","collect","down","right","right","collect"]
```

Kareem Ahmed Eladl, T-16, 52-5934
Mariam Mohamed Fawzy, T-16, 52-6892
Omar Mahmoud Awad, T-14, 52-9213
Youssef Osama Kamal, T-5, 52-1907

FOR OPTIMIZED PROJECT WE NEEDED THE FOLLOWING HELPER METHODS

- distance (S (x1,y1) cells s mystate) (x,y) function that gets distance between location of our robot and a pair
- distancePairs (x1,y1) (x2,y2) gets distance between 2 pairs on a grid using manhatten distance
- manhattenDistance (S (x,y) cells s mystate) a function that gets manhatten distance between current location of the robot and the closer mine
- removeInc (S (x,y) cells s mystate) a function that when we call nextMyStates it removes any state that makes the robot further from the mine
- nextMyStatesF (S (x,y) cells s mystate) a function that gets next state of our robot which makes it closer to the mines
- closer (x,y) (h:t) function that gets the closer pair in the list in the second parameter to the pair in the first parameter to decide which mine is closer to our position

DIFFERENT RUNS OF THE OPTIMIZED PROJECT

```
__   __ __  __  ____   ___      _________________________________________
||   || ||  || ||  || ||__      Hugs 98: Based on the Haskell 98 standard
||___|| ||__|| ||__||  __||     Copyright (c) 1994-2005
||---||         ___||           World Wide Web: http://haskell.org/hugs
||   ||                         Bugs: http://hackage.haskell.org/trac/hugs
||   || Version: Sep 2006       _________________________________________

Haskell 98 mode: Restart with command line option -98 to enable extensions

Type :? for help
Hugs> :load C:\Users\user\Desktop\OptimizedProject.hs
Main> solve (3,3) [(5,0) , (0,5)]
["up","up","up","right","right","collect","down","down","down","down","down","left","left","left","left","left","collect"]
Main> solve (2,1) [(5,0) , (0,5) , (4,1)]
["down","down","collect","down","left","collect","up","up","up","up","up","right","right","right","right","right","collect"]
Main> solve (1,3) [(3,0) , (0,5) , (4,1) , (2,2)]
["up","right","right","collect","down","down","left","left","left","collect","down","left","left","collect","down","right","collect"]
Main> |
```

Kareem Ahmed Eladl, T-16, 52-5934
Mariam Mohamed Fawzy, T-16, 52-6892
Omar Mahmoud Awad, T-14, 52-9213
Youssef Osama Kamal, T-5, 52-1907