University College Dublin

COMP 20230 – Data Structures and Algorithms

Lab Project 1 Report

Mereta Degutyte – 09644831

STACK ADT Design

The Stack ADT required two key methods within the design of what we are trying to implement. These are the add and remove functions from the stack. It's also possible that we will require to get an element of get the head of the stack to get the last element added to the stack.

The stack should be capable of holding tuples or arrays of data. This will be the key element stored within the stack and will represent a set of coordinates or potentially other items.

To test the Stack ADT I will use the code provided. These files linkedList.Py and Linked_list.py contain the operations to implement a basic linked list. Inside each node I will assign array elements and will demonstrate each of the available functions (such as add, remove, head, get element, contains)

Pseudo Code

Robot is at square 0,0 and will travel to 7,7

Add current position to linked list as an array

Move one square in any direction

If this is a valid position, add this position to linked list

Continue iterating over blocks until not a valid position is reached.

If position is not valid, remove last node from linked list and backtrack through the path

Navigate back and try another direction from original square.

Note: Code for testing general linked list functionality is contained in the tests.py file.

Report

Introduction

The following report outlines the definition and implementation details of a project which required to have a robot move through a grid system. Certain aspects of the project required the use of recursion and Stack ADT.

Project Requirements

The Key requirements for this project were:

- Describe Stack ADT components required and test out some usecases and code before attempting to build the world and move the robot around it.
- The ability to read instructions in from a file and have a maze created based on these instructions
    - These instructions would include
        - Size of Maze
        - Robot starting position
        - Goal Position
        - Blocked off squares (a wall concept)
- The robot should can move from square to square within the grid
- We should be able to check if the robot can move to an allowed square
- We should be able to check if the robot has reached is goal
- We should be able to check where the robot is at any point in time
- We should have the ability for a robot to automatically navigate to his goal position by himself
    - We should use Stack ADT with Arrays to prove this out
    - The robot should find the shortest path while navigating
    - The robot should have the ability to “backtrack” if required

System Design

Stack ADT

The first requirement was to design the Stack ADT. I tried to understand how Stack ADT would be used within this project and wrote some pseudo code based on an approach which I thought would work. I was unsure of the backtrack functionality within this code due to the complexity of understanding how a recursive loop would work up front. I wrote the pseudo code based on what I thought should happen. I then implemented some tests in python tests.py file to further understand how I would use the linked list and add, remove, get elements and other general linked list functions.

Create World and General Functions

The next requirement was to create the world. My Approach is the following:

- Read world configuration from file
- Sanitize data
- Store data in structured arrays which had the following format (Item, Coord1, Coord2)
    - Eg (Goal, 7, 0) – goal position
    - Eg (r2d2, 0, 7) – robot position
    - Eg (Wall, 1 ,5) – Wall block position
- create the world based on this array of data
    - The world will be based on a integer 2d grid.
        - Open blocks will have 0 as their value
        - Robot will have 2 as its identifier
        - Walls will have 1 as their identifier
        - Goal will have 3 as its identifier
- Implement functions based on the robot in the world
    - Is_feasible
        - This function checks if the robot is trying to travel out of bounds of the grid or if the robot is trying to move to a wall block
    - Move_robot
        - This function calls the is feasible function. If it is a feasible move, this will move the robot to a specified block
    - Where_is_robot
        - This checks where in the grid the robot is
    - Goal_reached
        - This checks if the robots current position and the goal position are the same.
- Print the result of grid
- Implement a searchMaze function to allow the Robot to iterate through the maze
    - This function will kick off a recursive call within the maze. More specifically it should call the findPath function which will check a path which the robot can travel.
    - It should check paths and find the shortest path to the goal block.
    - This should use recursion to find the shortest path to the goal

Testing

I will create a test file which will call the methods in the robotApp file. This will validate that all functions work as expected. This file will be the main file to be run which will prove the results. It should print out results for every test run. This should prove the following functions work:

- Stack ADT
- Move Robot
- Is feasible
- Where is robot
- Goal reached
- Search maze

Challenges

Firstly, I had to understand Stack ADT and how this could be applied to my solution. While I wrote the pseudocode for this I was unable to implement it fully when implementing my final find shortest path functionality. This was down to the fact I failed to understand how the recursive function should flow in conjunction with the application.

However, when it came to building the world, it was very like the last project which we carried out for Software Engineering module. We had to read data from a file, sanitize it and do something with it. I wanted to put the data in a structured format after reading it from a file. This was achieved by using arrays. This allowed me to read this data easily when I wanted to use it without having to parse it each time.

Creating the world was easy, once the data was in a good format.

The functions associated with the world were again easy once the data was in a consistent format which could be passed over to these functions from the calling test function. I did get all of this working in my project and results can be seen by running the tests.py file.

The main challenge of this project for me was the Stack ADT with recursion and trying to have the robot reach its goal by traversing through the maze. I did a lot of research online to understand how others approached this problem. Below are some of the resources which I looked at as part of understanding different approaches to this problem. None of the resources online used a stack for iterating through a grid, but instead manipulated the grid itself. By reading and understanding these solutions, I came up with my own solution which tried to make use of the Stack ADT which we were tasked with using.

http://interactivepython.org/courselib/static/pythonds/Recursion/ExploringaMaze.html

http://research.cs.queensu.ca/~cisc121/2009f/slides/week08-4up.pdf

https://www.csee.umbc.edu/courses/201/fall14/P2-A-Maze-ing.doc

I did get a solution but it did not find the shortest path. My solution simply iterates through the grid and finds a path. This is not necessarily the shortest path. It then returns the number of steps which are recorded in the Stack ADT which it took to get to its current point.

Conclusion

Overall I feel that I achieved quite a lot on this project. I understood how to use recursion to iterate through a grid and remember each place which I have been. It has deepened my knowledge of recursion. I also feel that I understand the use of objects within lists to store data using Stack ADT. Conceptually, I like the idea of a list storing arrays which allows data to be passed around easily between functions. I have a working solution for each part of the application, while I may have not met all requirements I feel like have made progress and understand a lot more after this project.