Fundamental in Go Assignment

# Supermarket Item Locator

## Overview

The user input the item they are looking for and the program shows them the path to take to find the item they are looking for.

The program will load a supermarket map, from graph.txt where each node is a junction and will contain the x and y coordinates.

## Features

Staff:

* Add item and location
* Remove item and location
* Update item’s location
* Update item out of stock

User:

* Input an item they are looking for to find the path to take to get to the item

## Data Structures

BST was used to store the item name. Each node is compared based on the name of the item. This makes searching for the item by name faster O(log(n)).

Graph was used to store the map of the possible paths in the supermarket. Each node represents a turning point, and the edges are possible paths. A figure of the layout is attached below (Sample Supermarket Map). The map was represented as a path, so the shortest path can be found using the dijkstra algorithm.

Priority Queue was used to store the next nodes to check to do breath first search on the graph to find the shortest path.

Stack was used to store the nodes to get from starting to ending node.

Queue was used to store the path to get to the item from the starting location to the item the user is looking for.

## Data and Format

There are three .txt files which store the graph, item and user information.

The item file stores the item, their x and y coordinates and the stock amount. The graph stores the nodes and edges that represents the map below (Sample Supermarket Map).

Sample Supermarket Map:



The white area represents the movable area and the gray area represents shelves.

## Error Handling and Concurrency

### Error Handling

To do error handling, most functions will return an error variable. This is cascaded between functions that call other functions so when an error occurs, in an inner function, the error will be passed outwards and the cause of the problem will be given.

For the BST:

For functions like, insert node, remove node, search for a node, an error handling is done to check that the node exists and provide an error accordingly.

For the graph:

When adding edges an error will be given if one of the nodes it is connecting does not exists yet.

When adding node an error will be given if the node already exists.

For the function to get node from ID, an error will be given is no such node exists.

For the stacks, queue and priority queue:

When dequeuing or popping, and error is given if the stack or queue is empty.

For the user:

When logging in, an error is given when the incorrect password is given.

Certain functions that require a staff action will give an error when a non-staff user tries to execute the function.

For the algorithm functions:

The errors are cascaded from previous functions, there are also checks to ensure that certain conditions are met (i.e. angle input is correct)

For example, the checkPointOnEdge function will try to find an edge that the point lies on, but if it cannot find any such edge it will return an error to communicate this.

### Concurrency

The functions called in main are run as go routines. To be able to get the user input from the main function to the goroutines, without interruption, a string channel is used.

Once the goroutines have completed requesting for input and will continue to call other functions or execute other lines, a reply is given in the channel “q” to let the main function know that that goroutine has completed requesting for information from user and a new action can be started.

The main function uses a select function to listen for a reply. Otherwise it will execute the default, which is to either start a new function or send the user input into the string channel for the other goroutines to pick up.

When the main gets the “q” input from the channel, it will return to the original prompt.

The mutex lock is used to lock whenever a staff is editing the items in the BST and when the BST is being saved to a .txt file, to avoid a race condition.

## Instruction on how to run the application

Run ./supermarketfinder found in the execute file.

This is the main function.

There are three possible inputs: “login”, “find item”, “staff action”.

If an input that is not one of the above is given for the original prompt, the input will not be accepted and the prompt will be given again.

After each of these three inputs, some instructions will be given, once there are no more instructions, press enter again to return to the original prompt to start a new action.

Based on the input there will be different functions:

1. login

The command line will ask for the username and password.

The possible username and password:

Staff:

George password123

Rachel password456

---

Member:

Joe lookhere

Jeremy lookthere

---

Only after a staff user has logged in, will staff actions be possible.

After the action is complete press enter to return to the original prompt.

1. find item

A list of all items will be displayed. The user can then select the item name they are looking for from the list, and the location they are at.

The valid input location is:

X: 0 to 4

Y: 0 to 2

If the item is available, the path from the input location to the item location will be given as directions.

If the item is not available, or out of stock, a message will be given.

After the action is complete press enter to return to original prompt.

1. staff action

There are four possible action: “add item”, “remove item‘, “update stock”, “update item location”

Based on the input there will be different functions:

1. add item

The item name, and location will be requested.

The valid input location is:

X: 0 to 4

Y: 0 to 2

If the X is not a whole number, Y has to be a whole number and vice versa.

1. remove item

The item name will be requested

1. update stock

The item name and stock amount will be requested.

1. update item location

The item name, and new location will be requested

The valid input location is:

X: 0 to 4

Y: 0 to 2

If the X is not a whole number, Y has to be a whole number and vice versa.