**Project 2**

**Artificial Intelligence**

**CSCE 5210 – Fall 2021**

**Distributed: Tuesday, September 28**

**Due: Tuesday October 19**

*[Solutions to this assignment must be submitted via CANVAS prior to midnight on the due date. Submissions no more than one day late will not be penalized. Submissions up to one week late will be penalized 10 points. Submissions will not be accepted if more than one week later than the due date.]*

This project may be undertaken individually or in pairs. If working in a pair, please state clearly the names of the two people undertaking the project and the contributions that each has made. Only one submission should be made per pair.

The purpose is to adapt the Iterative Deepening Search (IDS) method learnt in class to a realistic problem that is of relevance to Industry.

The environment is the warehouse that was used in Project 1, except that it has 15 locations, instead of 10. The 15 locations now represent divisions in the warehouse that stock different types of items. For example, location 1 is a division that stocks Electronics items while division 2 holds Clothes, and so on.

For reasons of efficiency the lead designer in the AI team has decided to map the warehouse in the hope that it will optimize the robot navigation process. Also, it was observed in the trial run undertaken earlier that the sensors were not accurate enough to support a 1-step lookahead search and it was thus decided to use the sensors to perceive the current location only and not the surrounding neighborhood.

The map produced by the AI team is in the form of a binary tree as shown below:

20

20

20

30

10

40

30

10

20

30

20

20

20

20

Each link in the tree has a **weight that reflects the distance between a pair of divisions**. Navigation is done by moving from the current position in the map (say the robot is at division B) to other map locations (say D) in order to service customer orders.

**Part A**

In this part we are concerned with minimizing the amount of inter-divisional movement. A customer always orders items from only **one** division in the company. Answer the following two questions. Note that the answers to the questions do not need any programming effort.

Q1) From the map above what data structure needs to be derived to support the minimization of movement between divisions that is required to support consecutive orders that involve different divisions?

Answer:

* Data structure: Binary tree
* Each node has the following information:
* Its child nodes (left and right)
* It’s parent node
* It’s weight from the parent
* It’s division name / category

Q2) How will we populate the data structure that you proposed in Q1 above? Outline the procedure involved. You only need to describe the procedure, not implement it at this stage.

Answer:

* Assumption: Due to time constrain, the robot has to follow a specific order of nodes / divisions not to delay processing and delivery departments.
* By observation, we can tell that child node’s number are equal to (their parent’s \* 2) + 0 / + 1 and parent node’s of a node is equal to floor(the node / 2)
  + For example: node 5 has children at 5 \* 2 + 0 and 5 \* 2 + 1, which are 10 and 11. Similarly, node 12 or 13 has a parent at floor(12 / 2) or floor(13 / 2), which is 6.
* When a new order comes, the robot checks for the path by some caculation:
  + Check the current’s and next node’s division and convert them to numbers associated with the tree using the data structure of a dictionary of which key stores the division’s category and the value stores the number of the tree
  + Comparing the current and the next node’s number (1)
    - If they are equal, the next node is the current one
    - If current > next, check if floor(cur/2) == next
      * If yes, next is the child of current
      * If no, current goes up to its parents and the algorithm repeats from (1) until the path to the next node is complete
    - If current < next, check if floor(next/2) – current
      * If yes, current is the child of next
      * If no, next goes up to its parents and the algorithm repeats from (1) until the path to the next node is complete

**Part B**

In Part A above we only considered movement to divisions. In this part we will consider the effort involved in servicing a customer order by moving to the divisions required as well as navigating to the shelf location that contains the item ordered.

Customer orders are generated by a 3-step process. First, a random number is generated that specifies the division number that contains the items ordered. This will be thus a random number in the range 1 to 15. Next, a random number is generated that represents the number of items ordered k. We will assume that no customer orders more than 3 items. In the 3rd step, k random numbers, each in the range 1 to 63, are generated that represent the shelf numbers where the items are located. For example, a customer may have ordered two different items at shelf locations 17 and 61 respectively. Shelves across all divisions have the same numbering scheme, which is in the range 1 to 63.

Thus, within a division you can represent the shelf by a complete binary tree. Except for the lowest level each node k has two children numbered 2k and 2k+1 respectively. **Within any division you may assume a constant step cost of 1 to move from any parent node to any one of its child nodes.**

Customer orders are serviced one-by-one in the order that they were received. The robot is in **division 1 at the first order**. From there it may have to move to another division if the first order does not have items contained in the first order. Once the robot services the current order at a given division **it must retrace its path back to the entrance of that division which is represented by the root node of the binary tree that covers the shelf locations for that division.**

Q3) Implement Q2 above by writing a Python program that populates the data structure you proposed in Q1.

Q4 Implement Iterative Deepening Search (IDS) to locate shelf locations for servicing customer orders.

Q5) Test your program for this boundary case. Generate an order for division 6 and trace its path from division 1 to division 6 and then onwards to the only item, item 33, that was ordered. Check that its path length is what you expected. Print out both the path and its length.

Q6) Run your program for a certain number N of customer orders (say 100) and answer the following questions:

1. What is the average path length travelled across the N orders? **Take care to include in your path length both the paths travelled to get to the division that contains the order and the movement required within the division to get to the items required.**
2. Print out the length of the shortest and longest paths across the N orders that you generated.

Hand in:

1. A short report (between 1 and 2 pages in size) describing the approach that you took. In addition to the description of your approach you should answer the 6 questions given above.
2. Any assumptions that you made while undertaking this project.
3. The complete set of Python code that you used.

End of project specification.

P.S. A\* is coming in the next project.