Artificial Intelligence Sessional

CSE-404

Group-B3

Group-B3

Sadia Bintay Mostafiz

202114084

Shahabuddin Akhon

202114092

Syed Nafees Kaiser

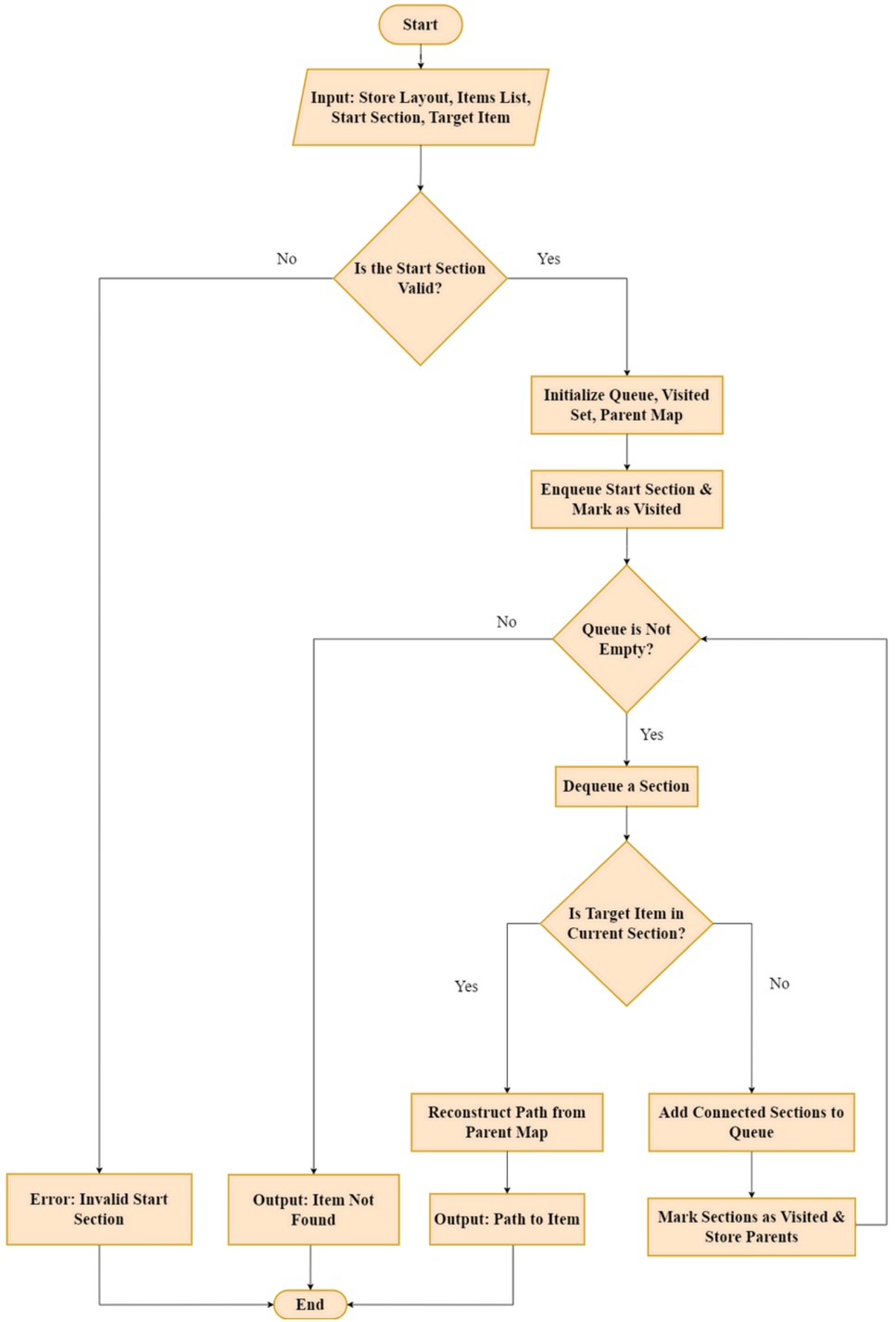
202114100

Finding a Grocery Item Using Breadth-First Search

Description

- In a grocery shop, Shoppers often struggle to determine the shortest or most efficient route to their desired items while finding them, which eventually contributes to spending unnecessary time because of the very complex or large layout of the shop.
- Therefore, inspired by the work of Dela Cruz et al. [1], an algorithm is developed using Breadth-First Search to determine the most efficient route to find the desired items.
- To achieve this, the store is modeled as a graph, where-
    1. Sections represent nodes and
    2. Paths between them represent edges.
- Using Breadth-First Search (BFS), an optimal path to the target item can be found by systematically exploring sections in increasing order of distance.

Flow chart


Start
Input: Store Layout, Items List, Start Section, Target Item
Is the Start Section Valid?
No
Yes
Initialize Queue, Visited Set, Parent Map
Enqueue Start Section & Mark as Visited
Queue is Not Empty?
No
Yes
Dequeue a Section
Is Target Item in Current Section?
Yes
No
Reconstruct Path from Parent Map
Add Connected Sections to Queue
Error: Invalid Start Section
Output: Item Not Found
Output: Path to Item
Mark Sections as Visited & Store Parents
End

Pseudo code

```
FUNCTION find_product_path(store_items, store_layout,
start_section, target_item):
    CREATE a queue (q)
    CREATE a set (visited) to track visited sections
    CREATE a map (parent) to store parent-child relationships

    ENQUEUE start_section into q
    ADD start_section to visited
    SET parent[start_section] = ""

    WHILE q is NOT EMPTY:
        section ← DEQUEUE from q

        IF section EXISTS in store_items:
            FOR each item in store_items[section]:
                IF item == target_item:
                    CREATE an empty list path
                    current ← section

                    WHILE current IS NOT EMPTY:
                        APPEND current to path
                        current ← parent[current]

                    PRINT "Path to target item:"
                    PRINT path in reverse order (from
start_section to found section)
                    RETURN

        FOR each neighbor in store_layout[section]:
            IF neighbor NOT in visited:
                ENQUEUE neighbor into q
                ADD neighbor to visited
                SET parent[neighbor] = section  // Store parent
section

    PRINT "Item not found"
```

Input and Constraints

Input

The user will input the item they are looking for.

Constraints

1. O(V+E) Search Complexity
2. Invalid Starting Position
3. Item Found in a Section
4. BFS Traversal Rules

Mapping and Searching Strategy

Graph Representation:

- The store is represented as an undirected graph.
  - Each section => node.
  - Paths => edges

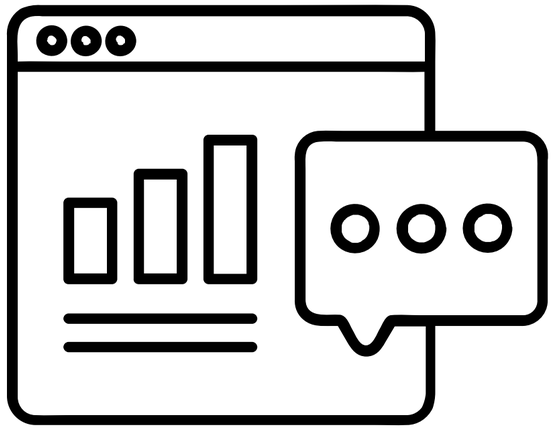
Algorithm Steps:

- Start BFS traversal from a given starting section.
- Use a queue to explore each section level by level.
- If an item is found in a section, print the path.
- If BFS completes without finding the item, print "Item not found".

Edge Cases Considered:

- The item is in the starting section.
- The item is in a distant section.
- The item does not exist in the store.
- Disconnected sections where BFS cannot reach the target.

Output



```
Enter the item you are looking for: Chocolate
Path to 'Chocolate': Entrance -> Bakery -> Snacks

Process returned 0 (0x0)   execution time : 35.715 s
Press any key to continue.
```

```
Enter the item you are looking for: Egg
Item 'Egg' not found in the store.

Process returned 0 (0x0)   execution time : 3.007 s
Press any key to continue.
```

Reference

[1] J. C. Dela Cruz, G. V. Magwili, J. P. E. Mundo, G. P. B. Gregorio, M. L. L. Lamoca and J. A. Villaseñor, "Items-mapping and route optimization in a grocery store using Dijkstra's, Bellman-Ford and Floyd-Warshall Algorithms," 2016 IEEE Region 10 Conference (TENCON), Singapore, 2016, pp. 243-246, doi: 10.1109/TENCON.2016.7847998.

Thank you