Data Structures and Algorithms II - C950

# F. Justify the package delivery algorithm used in the solution as written in the original program by doing the following:

# F1. Describe two or more strengths of the algorithm used in the solution.

**Easy Implementation**: The algorithm can be implemented without much complexity because of its straightforward logic.

**Quick Execution**: With fewer packages or addresses, the execution time is relatively fast, making it suitable for small-scale delivery tasks.

**Flexibility**: The algorithm does not commit to a predefined path for the entire journey, allowing it to adjust based on immediate circumstances.

**Resilience to Changes**: In dynamic real-world scenarios where sudden route changes may be required, this algorithm can accommodate those changes without completely overhauling the delivery path.

# F2. Verify that the algorithm used in the solution meets all requirements in the scenario.

1. Efficient Routing Requirement: The goal is to devise a route that guarantees the timely delivery of packages, adhering to their respective deadlines.
2. Package Selection: In the delivering\_packages method, the algorithm identifies the package nearest to the truck during each iteration. This is achieved by:
   1. Cycling through each package designated as 'on\_truck.'
   2. Using the get\_distance\_between\_addresses method to compute the distance between the truck's current location and each package's destination.
   3. Selecting the package with the minimal distance as the 'closest package.' This package is then set as the next delivery target. Once delivered, the truck's current location is updated to the address of this package.
3. Package Capacity: Each truck can accommodate a maximum of 16 packages simultaneously.
4. Speed Configuration: The speed is set to a fixed value of 18 mph within the Truck class. This speed is utilized to estimate the truck's mileage.
5. Hub Return Mechanism: The DeliveryManager factors the time and distance required for Truck 1 to revert to the hub before Truck 3 departs.
6. Address Correction: Package 9's delivery address is rectified appropriately to cater to the address amendment.
7. Distance Retrieval: The solution extracts distances using the approach self. distances[x][y]. If this does not yield a result, it defaults to self. distances[y][x], ensuring that the path's direction remains irrelevant and meets the stipulated requirement.
8. Delivery Completion: The delivering\_packages method persists in looping until the 'on\_truck' list (which holds the truck's packages) is depleted. This confirms that the algorithm runs until every package has been delivered.

# F3. Identify two other named algorithms that are different from the algorithm implemented in the solution and would meet all requirements in the scenario.

1. Bellman-Ford Algorithm
2. Floyd-Warshall Algorithm

# F3. A. Describe how both algorithms identified in part F3 are different from the algorithm used in the solution.

* The Nearest Neighbor algorithm is greedy and might yield a different route for delivery. It just picks the closest next destination without considering future steps.
* Floyd-Warshall and Bellman-Ford, on the other hand, work on considering all paths to determine the shortest path. They take a more holistic view.

# G. Describe what you would do differently, other than the two algorithms identified in part F3, if you did this project again, including details of the modifications that would be made.

1. The Truck class would a method to update the status of a package, such as "On Truck", "Delivered", or "Delayed".
2. I’d utilize a more robust hash function like DJB2 and add dynamic resizing to improve overall performance of the hash table.

# H. Verify that the data structure used in the solution meets all requirements in the scenario.

1. Store Packages:
   1. The HashTable class provides a mechanism to store packages. The table attribute is a list initialized to a specific capacity (default of 50), which will store the packages.
2. Insertion:
   1. The insert method in HashTable allows for the insertion of packages using a key and associated package data.
3. Search:
   1. The search method allows for searching packages using their keys.
4. Deletion:
   1. The remove method allows for the removal of packages using their keys.
5. Retrieve All Active Keys:
   1. The keys method lists all active package keys in the hash table.
6. Handle Collisions:
   1. The \_probe method in HashTable implements linear probing to handle hash collisions.
7. Load Data:
   1. The DataLoader class provides a static method load\_csv, which can load package data from a CSV file.

# H1. Identify two other data structures that could meet the same requirements in the scenario

1. Trie data structure
2. Balanced Binary Search Tree

# H1. A. Describe how each data structure identified in H1 is different from the data structure used in the solution.

1.Balanced Binary Search Tree

1. “BST does not require a load factor to be maintained as in Hash tables.
2. BST can support multiple keys with the same value, whereas Hash tables use the key to identify unique elements and cannot have multiple keys with the same value.” (GeeksforGeeks, 2023)
3. Collisions aren't an issue here since BST doesn't operate on the principle of hashing. However, it needs to handle balancing after operations.

2.Trie data structure

1. “For a hash table, we always compute the hash value for the whole input string whether the string exists in the hash table or not. For a trie, we can stop the search early when we don’t find a matching character link. Therefore, the lookup speed could be faster for the trie if the input string doesn’t exist in the trie.”
2. Each node in a trie can have a value associated with it, storing the package details.
3. Insertion: Inserting a string (key) into the trie would be efficient, especially if keys have shared prefixes

Work Cited page

1. Wu, G., & Wu, G. (2023). Hash Table vs. Trie (Prefix Tree) | Baeldung on Computer Science. *Baeldung on Computer Science*. <https://www.baeldung.com/cs/hash-table-vs-trie-prefix-tree#:~:text=For%20a%20hash%20table%2C%20we%20always%20compute%20the%20hash%20value,t%20exist%20in%20the%20trie>.
2. GeeksforGeeks. (2023). Advantages of BST over Hash Table. *GeeksforGeeks*. https://www.geeksforgeeks.org/advantages-of-bst-over-hash-table/