**WGUPS Routing Program Planning**

Farzin Shifat

Western Governors University

Data Structures and Algorithms II — C950: Task 1

Professor Mark Denchy

October 24, 2025

## 

## **A. Self-Adjusting Algorithm Identification**

I will use a **greedy nearest-neighbor (NN)** routing heuristic as my self-adjusting algorithm (Western Governors University, 2025b). NN is appropriate for WGUPS because after **each** delivery it recomputes the “nearest next stop” from the truck’s **current** position and time, so the route **self-adjusts** dynamically as conditions change (e.g., 09:05 delayed items becoming available or the 10:20 address correction for Package #9).

In my final approach, **deadline adherence is enforced at load/dispatch time**, not as a per-hop penalty: I seed earlier trucks with earlier-deadline groups and respect time gates (09:05 delays; 10:20 address fix) so that the hop-by-hop choice can remain purely nearest-neighbor while still meeting all deadlines (Western Governors University, 2025b, 2025c).

## **B. Self-Adjusting Data Structure Identification**

I will store package data in a **custom chained hash table implemented with Python lists only** (no built-in dictionaries). The table uses separate chaining and resizes around a ~0.75 load factor to preserve **O(1)** average-time **insert**, **lookup**, and **update** operations (Western Governors University, 2025b).

### **B1. How the Data Structure Accounts for Relationships**

Each hash-table entry encapsulates the relationships among the required fields:

* **Package → Address** (for routing),
* **Package → Deadline/Notes** (for grouping at load time and for time gates),
* **Package → Status/Time** (for real-time tracking and delivery timestamps).

For example, **before 10:20 a.m.** Package #9 remains **“on hold—incorrect address.”** When the correction arrives at **10:20**, I update its entry and it becomes **eligible for loading on the next departing truck** (Western Governors University, 2025b, 2025c).

## **C. Program Overview**

### **C1. Algorithm Logic (Pseudocode)**

The logic mirrors the course’s core algorithm overview (Western Governors University, n.d.), tailored to WGUPS:

**CONSTANTS**

TRUCK\_CAPACITY = 16

TRUCK\_SPEED = 18 mph

HUB\_DEPARTURE = 08:00

DELAYED\_GATE = 09:05 // delayed-on-flight items available at/after this time

ADDRESS\_FIX\_9 = 10:20 // package #9 becomes deliverable at/after this time

**INPUT**

distance[a][b] // symmetric miles between locations

packages in custom chained hash table keyed by Package ID:

{id, address, city, state, zip, deadline, notes, weight,

status, truck, depart\_time, deliver\_time}

**STEP 1:** LOAD DATA

read WGUPS Distance Table and WGUPS Package File

for each package: set status = "AT HUB"; clear times

**STEP 2:** APPLY CONSTRAINTS & SEED TRUCKS

group special cases:

- early deadlines (≠ EOD)

- “Truck 2 only”

- “must be delivered with …” groups

- “delayed on flight” (gate ≥ DELAYED\_GATE)

- package #9 held until ADDRESS\_FIX\_9

seed Truck 1 (earliest wave), Truck 3 (≥ DELAYED\_GATE), Truck 2 (waits

for earliest truck return) while respecting capacity 16

**STEP 3:** DISPATCH WAVES

Truck 1 departs at 08:00

Truck 3 departs at max(08:00, DELAYED\_GATE) = 09:05

Truck 2 departs at min(return(Truck 1), return(Truck 3))

**STEP 4:** ROUTE WITH PURE NEAREST-NEIGHBOR PER TRUCK

while cargo not empty:

from current\_location, choose nearest address among remaining cargo

travel; update truck.time += miles/TRUCK\_SPEED; truck.miles += miles

mark all packages for that address DELIVERED with timestamp

(Return to hub only when required for the next wave.)

**STEP 5:** REPORT

print total miles across all trucks (target ≤ 140)

support status queries (AT HUB / EN ROUTE / DELIVERED with times) by ID and time

### **C2. Programming Environment**

* **Language/Runtime:** Python 3.x
* **Data structures:** custom list-based chained hash table + lists (no Python dict in the task solution)
* **IDE/Tools:** VS Code or PyCharm; Git for version control
* **OS/Hardware:** macOS; multi-core CPU; ≥8 GB RAM  
   This environment reads the CSV exports of the WGUPS package file and distance table and prints a simple CLI for status/mileage (Western Governors University, 2025a, 2025d).

### **C3. Space–Time Complexity (Big-O)**

| **Segment** | **Time** | **Space** | **Notes** |
| --- | --- | --- | --- |
| Load packages into hash table (n≈40) | **O(n)** | **O(n)** | One pass insert into chained table |
| Constraint scan & seeding | **O(n)** | **O(n)** | Build small groups (must-with, truck-2-only, delayed, #9) |
| NN routing within a truck (k ≤ 16) | **O(k²)** | **O(k)** | Greedy recomputation after each stop |
| Status queries & report | **O(n)** | **O(n)** | Linear pass for “all at time T” |
| **Overall** | **O(n²)** | **O(n)** | NN dominates; acceptable for n≈40 |

### 

### **C4. Scalability and Adaptability**

* **Volume:** Hash-table **O(1)** average operations scale to larger daily volumes.
* **Geography:** The algorithm is **city-agnostic**; swapping in a new distance table reuses the same core.
* **Live constraints:** Time gates (09:05 delays, 10:20 correction) naturally integrate—when a gate time is reached, packages become eligible, and the next wave uses NN from the current state (Western Governors University, 2025c, 2025d).

### **C5. Software Efficiency and Maintainability**

* **Modular design:** Separate loaders, hash-table core, routing, and CLI.
* **Configurable constants:** Speed, capacity, and gate times are single-point changes.
* **Traceability:** Each package holds status, truck, depart, and delivery times for audit.
* **Reusability:** The distance and package data are external CSVs; logic is reusable across cities (Western Governors University, n.d., 2025a).

### **C6. Strengths and Weaknesses of the Data Structure**

**Strengths.**

* **O(1)** average insert/lookup/update for fast dispatcher queries.
* Simple memory model; easy to serialize/debug.
* Encapsulates all rubric-required fields per package entry.

**Weaknesses.**

* No inherent ordering (handled by routing/printing logic).
* Pathological collision patterns can degrade to **O(n)**, though resizing and chaining mitigate this.
* Not a “self-adjusting tree”; its “self-adjustment” here refers to **routing**, not rotations.

### **C7. Key Choice for Efficient Delivery Management**

I key the hash table by **Package ID** because it is unique, stable across address or deadline updates, and directly answers operational queries (“Where is Package 17 right now?”). The other candidates (address, city, ZIP, deadline, weight, status) are non-unique or volatile, which would complicate constant-time access (Western Governors University, 2025b, 2025c).

## 

References

Western Governors University. (n.d.). *Sample core algorithm overview* [Internal course handout for C950].  
 Western Governors University. (2025a). *Salt Lake City downtown map* [Internal project document for WGUPS routing].  
 Western Governors University. (2025b). *NHP3 Task 1: WGUPS routing program planning* (Data Structures and Algorithms II — C950) [Internal task prompt].  
 Western Governors University. (2025c). *WGUPS package file* [Microsoft Excel/CSV].  
 Western Governors University. (2025d). *WGUPS distance table* [Microsoft Excel/CSV].