# A.  Develop a hash table, without using any additional libraries or classes, that has an insertion function that takes the package ID as input and inserts each of the following data components into the hash table:

•   delivery address

•   delivery deadline

•   delivery city

•   delivery zip code

•   package weight

•   delivery status (i.e., at the hub, en route, or delivered), including the delivery time

Fulfilled by the hash\_table class.

B.  Develop a look-up function that takes the package ID as input and returns *each* of the following corresponding data components:

•   delivery address

•   delivery deadline

•   delivery city

•   delivery zip code

•   package weight

•   delivery status (i.e., at the hub, en route, or delivered), including the delivery time

Exemplified by the all\_packages variable and its uses. All\_packages is a hash\_table with package objects inserted using the package.id as a key. Any of these requested data can be retrieved from the package object by name.

C.  Write an original program that will deliver *all* packages and meet all requirements using the attached supporting documents “Salt Lake City Downtown Map,” “WGUPS Distance Table,” and “WGUPS Package File.”

1.  Create an identifying comment within the first line of a file named “main.py” that includes your student ID.

2.  Include comments in your code to explain both the process and the flow of the program.

D.  Provide an intuitive interface for the user to view the delivery status (including the delivery time) of any package at any time and the total mileage traveled by all trucks. (The delivery status should report the package as at the hub, en route, or delivered. Delivery status must include the time.)

1.  Provide screenshots to show the status of *all* packages loaded onto *each* truck at a time between 8:35 a.m. and 9:25 a.m. A screenshot of a computer

Description automatically generated

2.  Provide screenshots to show the status of *all* packages loaded onto *each* truck at a time between 9:35 a.m. and 10:25 a.m. A screenshot of a computer

Description automatically generated

3.  Provide screenshots to show the status of *all* packages loaded onto *each* truck at a time between 12:03 p.m. and 1:12 p.m.A screenshot of a computer

Description automatically generated

# E.  Provide screenshots showing successful completion of the code that includes the total mileage traveled by *all* trucks.

A screenshot of a message

Description automatically generated

# F.  Justify the package delivery algorithm used in the solution as written in the original program

1.  Describe **two or more** strengths of the algorithm used in the solution.

2.  Verify that the algorithm used in the solution meets *all* requirements in the scenario.

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

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

The algorithm used in this solution was a nearest neighbor algorithm with 2-Opt optimization. Some of the strengths of this algorithm include ease of implementation while providing a relatively good approximation of the optimal solution. The nearest neighbor algorithm is naïve in its approach but is augmented quite easily into a 2-Opt algorithm. 2-Opt runs in O(N^2) which provides an improved result over nearest neighbor alone for distance traveled at the cost of increasing compute time. 2-Opt expands upon nearest neighbor by checking if any 2 nodes in the algorithm could be swapped for one another within the walk. Another strength is that 2-Opt always maintains a viable walk so it could be timed out at any moment, if necessary, while still providing for a chance at greater optimization than nearest neighbor alone.

Another algorithm that would meet the requirements for this project would be a modified Dijkstra’s Algorithm. Dijkstra’s Algorithm works by selecting two points and finding the shortest path between them. Some strengths of using Dijkstra’s Algorithm over nearest neighbor would be for nodes that are furthest away from the hub it could find a path that would visit additional nodes along the route minimizing the overall distance traveled. The main difference between the algorithms is that for Dijkstra’s you must come up with a selection method to determine which node to use as your end node.

A second algorithm that could also be used for this task would be Christofieds algorithm which uses a minimum spanning tree of the graph and a set of the odd vertices to create the path. It runs in polynomial time like 2-Opt but is O(n^3) instead of O(n^2). The main benefit of Christofieds is that it guarantees solutions that are at least 3/2 of the optimal solution. (see Christofieds, Nicos (1976) retrieved from: https://web.archive.org/web/20190721172134/https://apps.dtic.mil/dtic/tr/fulltext/u2/a025602.pdf)

# 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.

If I were to do this project again, I would optimize how I completed a few of the tasks. For example, when I created the distance\_table matrix I wrote a quick script to essentially double the amount of data stored so that I could easily access any node using distance[node1][node2] but in hindsight I realized that it was unnecessary. Instead I could have used distance[largest\_node\_index][smallest\_node\_index] while maintaining the original matrix shape given in the distance table, reducing the overall memory requirement.

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

1.  Identify **two** other data structures that could meet the same requirements in the scenario.

a.  Describe how *each* data structure identified in H1 is different from the data structure used in the solution.

The data structure I used for this project was a hash table with simple chaining containing a package object which can readily provide any of the requested data in part B.

Another data structure that could be used to store the data for this project would be a simple linked list. It is different from a hash table because the data is all stored in a Node object which have a parent, data, and child attributes. The list is searched by navigating through each child node and checking for some value. Each node could contain the package object, and the search would just need to match the requested package id to package.id.

A third and final data structure that could be used would be a simple binary tree using the package id to balance the tree. A binary tree is similar to a linked list but instead of only having one child node each node has two children with the left node being less than the right node and additional balancing algorithms are ran to ensure maximum efficiency in access times.