PE07: Programming Exercise

Instructions

- `dijkstra_main.py`
- `dijkstra.py`

Description

This assignment is to learn about the Dijkstra algorithm. Given a hash graph map from the dijkstra_main.py, complete the functions in the separate "DijkstraClass" class within dijkstra.py. The main object of the "DijkstraClass" is to find the lowest-cost path from the start (s) to finish (f). Note that "dijkstra_main.py" with the "main" method and starter file "dijkstra.py" with class has already been provided (download attachment).

As part of the assignment, describe how the Dijkstra algorithm works in your own words. Keep in mind to always comment and document your class and methods.

Documentation reference

Mertz, J. (n.d.). Documenting Python Code: A Complete Guide.

Expected result

1. `dijkstra_main.py`
    - This is the Main python file that is already provided and contains the main and test procedure, which calls methods implemented on "dijkstra.py" (this is already provided, but please include this file in your submission).
2. `dijkstra.py`
    - This class contains such methods as init, initial_costs_parents, find_shorted_path, find_lowest_cost_node, and print_path. Please keep in mind the following notes for each method during implementation:
    - `init()`: this method has already been provided.
    - `initial_costs_parents()`: this method initializes costs and parents global variables as to how the Dijkstra algorithm works.
    - `find_shorted_path()`: this method updates costs and parents global variables by following the Dijkstra algorithm.
    - `find_lowest_cost_node (costs)`: this method finds and returns the lowest cost node that hasn't been processed yet.
3. **Parameters**: *costs*
    - `print_path()`: if there is path, prints path from `s` to `f`

Summary

As part of the assignment, describe how the Dijkstra algorithm works in your own words.

Dijkstra's algorithm, developed by Edsger Dijkstra, is a method used to find the shortest path between nodes in a graph. The algorithm starts at a chosen `start node` and analyzes its immediate neighbors, calculating the tentative distances to them. It keeps track of the smallest value, representing the `shortest path`. The

algorithm then moves to the node with the smallest recorded distance, marks it as "**visited**", and reassesses its non-visited neighbors, recalculating their tentative distances. This process continues until all nodes have been visited, determining the shortest path from the start node to every other node.

It's important to note that Dijkstra's algorithm assumes all edge weights in the graph are non-negative. It cannot reliably find the shortest path if there are negative edges.

Screenshot