Tandem Algorithm Module 2

Anh Huynh

Project: Adaptive Routing for Real-Time Logistics Disruptions

```
#include <iostream>
#include <queue>
#include <limits>
using namespace std;

// edge going to a node with base travel cost
struct Edge {
    string to;
    int baseWeight;
};

// tandem algorithm
unordered_map<string, int> tandem(
    unordered_map<string, vector<Edge>>& graph,
    const string& start,
    unordered_map<string, int>& thresholds
) {
    unordered_map<string, int> dist;

    // initiatlize all distances to infinity
    for (auto& node : graph) {
    dist[node.first] = INT_MAX;
    for (auto& edge : node.second) {
        dist[edge.to] = INT_MAX; // all destination nodes are added
    }
}

    // priority queue for the shortest path
    priority_queue<pair<int, string>, vector<pair<int, string>>, greater<>> pq;
    pq.push({0, start});

    while (!pq.empty()) {
        auto [cost, node] = pq.top(); pq.pop();
        if (cost > dist[node]) continue;

        for (auto& edge : graph[node]) {
            int adjusted = edge.baseWeight;

            // apply penalty if threshold is exceeded
            if (thresholds[edge.to] > 5) adjusted *= 3;
            if (cost + adjusted < dist[edge.to]) {
                dist[edge.to] = cost + adjusted;
                pq.push({dist[edge.to], edge.to});
            }
        }
    }
    return dist;
}
```

Early Validation:

Graph of A ---> B ---> C
Initial Weights:
A ---> B = 2
B ---> C = 3

Disruption:
B ---> C delay so increase weight
If the threshold for a node is greater than 5, apply a penalty by tripling the edge weight leading to that node.

Expected shortest path (A->B->C): 5
After disruption (B->C reweighted to 9): 11
Observed result: Matches expected since algorithm re-evaluated and avoided disrupted route in alternate scenarios.

Terminal:
Shortest cost to C: 11
Test passed!

Preliminary Runtime:

O(E log V)

Memory: similar to standard Dijkstra (O(V + E))

Next Steps:

- Expand to larger graph (maybe >= 50 nodes)
- create disruption types
  - ex: traffic/weather/priority overload (too many high priority packages on a path)/seasonal delays/spike in fuel cost)
- add time-window constraints for time sensitive deliveries and add penalties to paths that violate the constraint
- recalculate best route from scratch if shipment's route becomes too delayed or too expensive overall

- add support for multiple shipments at once and prioritize urgent ones (aka paid priority or time sensitive or emergency supply)