This problem is solved by a solution that finds the minimum path cost on a grid where each cell has a specific height. We traverse from the starting source S to all designated in the set B. This problem solution includes elements of graph traversal, pathfinding, and dynamic programming. The dynamic programming problem is structured around a state space defined as (x,y,visited) where x and y are coordinates and visited is the bitmask that represents which special stations have been visited thus far. The recursive relation is that for updating the cost to reach any cell for the new position (x’ , y’) we calculate new_cost = cost + travel_time(x,y,x’,y’) where travel_time is defined as travel_time(xi,yi,xf,yf) = max(-1 , 1 + grid_heights[xf][yf] = grid_heights[xi][yi])).

In terms of traversal, this code employs Dijkstra’s algorithm using a priority queue. This algorithm is suited for this problem uniquely since it takes into account negative weights, this is because the negative weight will never go below -1 since the max function for travel_time takes care of this. This algorithm operates with a complexity of O(VlogV+E), where V is the number of vertices and E is the number of edges. V corresponds to the number of cells in the grid times the number of possible states of the visited set. E is roughly 4V since each vertex in the grid has at most 4 neighbors. Thus the overall complexity can be estimated as O(VlogV) considering the dense nature of the graph and that E is a variation of V (4V) leading to a simplification. This algorithm roughly calculates the minimum path cost while ensuring that all special stations are visited at least once, leveraging Dijkstra’s algorithm without being compromised by a negative value which is handled inside the travel_time implementation. Then, the result is written to a file (not the path, just the cost).