Edwin Rodriguez, Jasmine Rodriguez, Amanda Shohdy

Algorithm 2

Defines a function greedyHamiltonian that takes three parameters: fuel city_distance , miles per hour, returns an integer representing the preferred starting city.

In the main function:
vector city_distances
vector fuel
int mpg (available fuel)

while(input does not equal 1) {

(until the user enters -1)
store distances in city_ditances

}

while (!=1) { (prompt user to enter available fuel at every city)

(until user enters -1)
}

(prompt user to enter miles per gallon)

call the greedyHamiltonian function with fuel, city_distances, and mpg, and output the preferred starting city returned by the function

int greedyHamiltonian(fuel, city_distances, mpg){

for (iterate through each city as a potential starting city){
fuel_available = 0
correct = true

for(iterate through all the cities to check if Hamiltonian path is possible){
(Compute the index of the current city considering the starting city)
(Update fuel_available by adding fuel multiples by mpg and subtracting the distance to the next city.

if(fuel_available < 0){
correct = false
}
}
If(correct is true after the loop){
return the current starting city as the preferred starting city
}

If no correct starting city is found, return -1.

Step Count Analysis

Time complexity would be O(n^2) because there is a function that iterates through each city to consider it as a potential starting city. It takes O(n) time. Inside the outer loop, another one iterates through all cities to check if a Hamiltonian path is possible. Traversing all the cities takes O(n) time which together comes out to O(n^2).