CSE 2046.1 – ANALYSIS OF ALGORITHMS
THIRD PROJECT

Author:

Erhan Yalnız - 150117905

EXPLANATION OF ALGORITHM AND IMPLEMENTATION DETAILS

First of all, I must point out that since I completed my project without completing the twelfth chapter, I did not use a greedy algorithm. After seeing the greedy algorithms in our lessons, I did not prefer to use them again because I had already completed my project. Although I don't use a greedy algorithm in my project, I think my program is fast enough. For now, briefly, I have designed a brute force algorithm that results by taking optimization parameters, denoting limited combinations, and dividing the question into pieces. I completed my program using 3 classes named as Problem, City and Path.

1) **Problem Class**

At the beginning of the program, I defined two variable that named as "p" which will keep the last answer found and "cities" to hold the cities as a list read from file with x and y positions on 2D grid. In this class, I have seven methods named as main, parseFile, tspTourCalculate, d, getSortedQueue, tryPath, printPath.

- Main Medhod
  In main method, I used a try-catch statement. In try block, there is a caller for parseFile method which used for parsing files to get cities and their positions on 2D grid. After this, I defined "p" to hold the results of calculations and I put a caller for tspTourCalculate method. This part defined to solve TSP problem with given cities and divide the whole problem to TSP problems with at most 1000 cities. Furthermore, I limited queue iteration on each part by 10 and limited the call (recursion) by 2 times of part size. After the program completed the necessary procedures, I called the printPath method to write the results. In catch block, I assigned an assignment to show the error and exit the program, if any error occurs.

- parseFile Medhod
  This method parses file to get cities and their positions on 2D grid. It reads whole file and appends to cities list.

- tspTourCalculate Medhod
  This method calculates the TSP problem with nearest neighbor algorithm approach and it gets three parameters. These parameters are called as partSize, queueLimit, callLimitMultiplier. partSize parameter is used to divide problem to more little problems to make it easier and faster to solve and other two parameters are used to optimize each and every step of calculating parts of whole TSP problem. Firstly, it gets the starting time to calculate how much time trying each tour takes and it starts to perform the divide and conquer method. trySolve function already calculates small number of tours with up to 1000 cities but to solve TSP problems with bigger sets of cities, there is another way to approach it: divide whole problem to TSP problems with more manageable sized sets of cities so divide them to be TSP problems with 1000 or less cities. Tour should contain exactly the same number of elements to visit every vertex once and with return to starting element we get one more element. Furthermore, I used a nested loop structure in this method and I defined a queue in it so that it represents all the cities that are needed to be visited and I added the cities that need to be visited to queue. After it, I defined "subresult" which type is Path Class to a system and I assigned it with calling tryPath method to calculate every and each path using a recursive function and get the best path. As finally, by using"concat" method which is a concatenate method and used to merge two paths and add passed path as a trail to this path in Path Class. After this for statement and required calculations are

completed I got the ending time which used to get an information about how good optimization variable can be to calculate how much time trying each tour takes. Then, I return the best path calculated.

- d Method
  As stated in the project assignment, distance between two cities is defined as the Euclidian distance rounded to the nearest integer and I used this method for this. It gets two parameter as city and calculates the distance between these two cities.

- getSortedQueue Method
  I created this method to do bubble-sort the queue according to minimal distance with current city and after this operation is completed I returned new queue.

- tryPath Method
  While I was talking about the structures that call and use this method above, I also mentioned this method. Now I will explain in more detail. This method is a recursive function to calculate tours starting from nearest neighbor path and it gets six parameters, queue, lastCityId, pathTaken, bestPath, queueLimit, callLimit. Queue represents all the cities that are not yet visited. (just id's of them). Last City Id is the city algorithm is currently on which will make decisions to further choose a next city to travel. Path Taken is the tour we are creating on this iteration of algorithm. Best Path is a complete tour of all cities with known minimum distance (known best solution for TSP) and Queue Limit is optimizing this algorithm by just allowing branching combinations when there is less cities than this limit to choose from queue. This method has required if statements to control of steps.
  When all cities are visited, it returns the resulting travel path and distance.
  If the path is longer than bestPath it returns maximum possible distance as a result to abort the path.
  If call limit is reached, it returns bestPath to abort to more recursive calls.
  It sorts queue by minimal distances to current city algorithm. It gets the number of remaining cities to visit and because of queue size is big, there is ton of combinations so limits the number of possible next cities to visit by "queueLimit". I defined remaining required implementations below to the program by using a for loop and condition statements in it;
  Getting the next selected city to visit.
  Creating the new queue for next sub-path.
  Creating path for next city by copying old path and adding next city to it.
  Getting a minimal tour distance of each combination of sub-path that could be chosen from queue and assigning it to a temp variable.
  I defined a property to be applied an optimization by using condition statements. This optimization should only be applied if "callLimit" is positive. It decreases call limit by each time the function was recursed and this can be calculated by difference between number of items between the path that was calculated and path that algorithm is currently have taken. Furthermore there is a condition statement again to check if current calculated sub-path is the new minimum sub-path from current city. If a new minimum distance is found, it is updated. Finally, after all operations and checkings are completed, it returns the minimum known completed tour as bestPath.
- printPath Method
  In this method, there are printing operations. Through this method, all info about path is printed to file.

2) City Class

This class is a simple City Class just used to represent cities in TSP problem.

3) Path Class

I created this class to describe a tour around cities and at the beginning of it, I defined a "dist" variable for sum of distance and I defined an array list variable which named as "cities" for cities travelled in order. Then, I placed the constructors given below, respectively;
Default constructor to initialize everything to empty.
Constructor to initialize everything to empty. It constructs with given distance.
Constructor to initialize a path with starting city.
Constructor to initialize a new path with a existing one by copying it.
Furthermore, in this class we have 5 methods that named as add, removeLast, concat, returnStart, d.

- add Method
  This method is used to add a new city to path. I should also point out that delta distance is the distance needed to reach destination city.

- removeLast Method
  This method is used to remove last city from path. I should also point out that delta distance is the distance needed to reach city from new last city.

- concat Method
  This method is used to merge two paths and add passed path as a trail to this path.

- returnStart Method
  As the name suggests, this method is used to return start. This method allows us to return to the starting city.

- d Method
  As stated in the project assignment, distance between two cities is defined as the Euclidian distance rounded to the nearest integer and I used this method for this. It gets two parameter as city and calculates the distance between these two cities.