Assignment 2: The Validator

Philip Boeken, Stein Heijjer and Koen Meijer

Model description:

The validator contains the following classes with its functions:

Class	**Description**
Instance	Contains all information of the instance The most important functions are: - Parsing an instance.txt file - Calculating the distance between two requests
Tool	Contains all information of one type of tool
Request	Contains all information of one request
Schedule	Is a schedule of what Requests (deliveries and pickups) are planned on what day, keeps track of this information via a dictionary of {day: ScheduleDay} pairs The most important functions are: - Checking its theoretical feasibility in terms of tool use: is for every day (deliveries-pickups) > (start depot for that day)? - Checking if the schedule of requests is valid. These are things such as if all requests are done on the right day, if all requests are handled etc.
ScheduleDay	Contains a list of deliveries (Requests) and a list of pickups (Requests) to be done on one day
Routing	Is a configuration of trips to be done every day, keeps track of this information via a dictionary of {day: RoutingDay} pairs. The most important functions are: - Calculating the costs (and all relevant information for calculating costs) of a configuration of Trips (vehicles) - Checking whether all RoutingDay elements are valid
RoutingDay	Contains a list of Trips to be done on that day The most important functions are: - Checking whether every Trip is valid - Checking whether the total tools used on all the trips is below the maximum tool use.
Trip	Contains a list of Request elements to be visited (by a single vehicle on a single day), where a negative requestID represents a pickup, a positive requestID represents a delivery and a ‘0’ element represents visiting the depot. The most important function is: - Checking whether it is valid in terms of total distance, and whether the capacity of the vehicle is not exceeded in every sub trip (the Trip between two visits of the depot)

Solution	Contains all relevant information for creating a solution file The most important functions are: - Parsing a Routing instance - Writing a solution.txt file
Validator	Reading the input arguments, creating an Instance and a Solution, possibly retrieving and printing errors

Despite the strong resemblance between the information contained by Schedule and Routing, we chose to implement both models because of their different functionality which provides convenience for the way we wanted to create a solution from scratch: first determining a configuration of what requests must be handled on what day (Schedule), and then given that Schedule creating a configuration of Trips (i.e. vehicles) which minimizes the total costs.

The validator works by checking two functions:

- Schedule.hasErrors():
  The schedule.hasErrors checks if all the requests are handled properly, i.e. checking whether all requests are handled, whether all requests are done on the right day etc.
- Routing.hasErrors():
  The routing.hasErrors checks whether all routes are valid in terms of vehicle capacity, tool use and vehicle distance. We calculate vehicle capacity by making a time series for all tools on all the requests on a trip, starting from a depot. When a delivery is done, the last item in the time series for the appropriate tool is subtracted by the amount and size of the request. For pickups we add the amount and size. At the end of a trip we check the minimum and maximum of the sum of the time series for all the tools. The difference between the minimum and maximum is the maximum capacity used during the trip.
  The same principle is done for the tool count. The lowest negative number in the time series for every tool is the amount of tools needed from the depot on that day. If the series only consists of positive numbers (pickups) tools used is set to 0.
  Furthermore this function also checks if depot visits are done properly and if the total vehicle distance is not exceeded.

Runtime analysis:
To be able to analyse the runtime of our validator we introduce the following notation for the instance and solution parameters:

- Number of days: D
- Number of tools: Ts
- Total number of requests: R
- Maximum number of request on a single trip: R_T
- Total number of trips: T
- Maximum number of trips on a single day: T_D
- The size of the instance: N

The validator calls the following functions with their total runtimes as shown below:

Instance(): O(R) *Create input instance*

Solution.parseFromTxt(): O(S) *Parse solution file*

Solution.writeRouting(): O(D x T_D) *Write routing instance*
Solution.parseTrip(): O(R_T)

Solution.writeSchedule(): O(R) *Write the schedule instance*

Schedule.hasErrors(): O(D x R_D) *Check if all requests are handled properly*

Routing.hasErrors(): O(D)
RoutingDay.hasErrors(): O(T_D)
Trip.hasErrors(): O(R_T) Check correct use of depots
Trip.hasErrors(): O(R_T) Check if vehicle distance is not too high
Trip.hasErrors(): O(1) Check vehicle capacity
Trip.capacityError(): O(R_T)

RoutingDay.hasErrors(): O(Ts) Check if tools used is not higher than tools available
RoutingDay.toolsNeeded: O(T_D)
Trip.totalToolsNeeded(): O(R_T)

Solution.calculateInfo(): O(1) *Calculate information of the solution*
Routing.maxNumberofVehicles(): O(D)
Routing.numberOfVehicleDays(): O(D)
Routing.distance(): O(D) *Calculate distance*
RoutingDay.distance(): O(T_D)
Trip.distance(): O(R_T)

Routing.cost(): O(1) *Calculate cost*
Routing.maxNumberofVehicles(): O(D)
Routing.numberOfVehicleDays(): O(D)
Routing.distance(): O(D)
RoutingDay.distance(): O(T_D)
Trip.distance(): O(R_T)

Routing.toolCount(): O(D) *Calculate tools used*
RoutingDay.toolsNeeded() O(T_D)
Trip.totalToolsNeeded(): O(R_T)

If a function is placed a tab to the right, it is part of a loop within the upper function, in which case runtimes should be multiplied. If a function is on the same tab runtimes have to be added. It can be seen that the maximum runtime of this validator is never higher than *O(D x T_D x R_T),* which translates to *O(R)*. This shows that our validator has a runtime of *O(N)*.