Ali Nawaz Maan
S4468881

PHONE NETWORK FAULT FINDING (DATA STRUCTURES & ALGORITHMS)

The problem identifies as fault finding within a phone network. The phone network is a collection of routers and switches and phone diallers and receivers where there is a connection path between these. Each connection path is represented by a line in the file with dialler & receiver phone, switches and timestamp information.

The choice of data structures and algorithms are based on the fact of retaining the data in memory and perform efficient searches on the same.

CUSTOM LIST

The main data structure which helps in efficient O(n) storage of any arbitrary objects is the implementation of Array List like data structure with generic type settings. This allows me to have a collection of objects with linear memory efficiency and some efficient insert and locate methods for objects.

In the context of this particular problem, the data from the network files have to be stored in some collection in order to perform efficient accession and mutation. The list is implemented as generic type setting because data will be in lot many different type of objects and this approach helps reduce the work.

The custom list interface is implemented by OwnList class. Moreover, OwnList is extended by SortedList class to perform additional operations on the data. SortedList is also of generic type and contain sorting and searching methods.

Quick sort is implemented with arbitrary pivot index to sort the data. The data is compared using a comparator which the caller of the method will provide based on their stored object types. This algorithm allows for sorting the data in O(nlogn) time. (Goodrich, Tamassia, & Goldwasser, 2014) The purpose of sorting the data is to perform efficient binary search later. Once sorted, accessing the data and searching through an element becomes O(logn) time. SortedList also implements Binary search to aid in efficient searching through the collection. Searching also takes comparator as method argument in order to compare elements. (Goodrich, Tamassia, & Goldwasser, 2014)

GRAPH

The network consists of phones and switches. Now, when one phone is dialling a call to other phone, it has to go through different switches to reach the receiver. This raises an interesting idea of having a sparse disconnected graph. The nodes of graph are called PhoneNode here and the edges are the actual call records between two phone nodes. The call record contains the information about dialler phone, receiver phone, connection path, timestamp and the fault checking mechanism as well. ExtendedCallRecord object takes all this necassy information to form a call record. After extracting the information, it checks for faults in the connection path and stores the faulty data if any. PhoneNode object contains the phone identifier and the list for dialled extended call records and received extended call records.

The graph or network contains a list of PhoneNode objects. This helps a lot in finding faults and getting the receiving phone numbers and dialling ones.

When searching for a particular phone node from the list, a comparator based on phone identifier is used to search through it. When the phone node is found, it already contains information about connection paths that this phone number has dialled, received and faulty connection paths all with timestamp to aid in searching within a specific period of time.

This allows to perform efficient searching within a large dataset without going through the collection more than once.

SwitchToDateMap

SwitchToDatemap object contains mappings of a particular switch from the switches list to the relative timestamps in the call records data. While reading the data, SwitchToDateMap objects are created with switch identifiers and empty list of timestamps.

After creating the list of these objects, the collection is sorted based on comparator on switch identifiers. Now, while reading through the call records data, every time there is a switch in the connection path, it searches through the switches list and adds the record timestamp to the SwitchToDateMap list of timestamps. The searching through switches list is performed O(logn) time because the data is sorted and we can do binary search.

While searching for the minimum and maximum connection switch, the size of timestamp list is compared for all switches within the collection. This happens in O(n) time. Of course, searching through the switch connections within a specific time period is just applying filters for timestamps.

Bibliography

Goodrich, M. T., Tamassia, R., & Goldwasser, M. H. (2014). *Data Structures & Algorithms in Java* (6th ed.). USA: John Wiley & Sons.