Advanced Data Structure Programming Project I
GatorTaxi

April 2023

1 Member Information

Haolan Xu
UFID:34326768
UF Email:haolanxu@ufl.edu

2 Implementation

In my GatorTaxi program, I use various data structures such as Red-Black Trees (RBT), Min Heaps, and standard C++ containers. The main components of the program include:

1.main.cpp
2.RideManager class (func.h, func.cpp)
3.Red-Black Tree class (rbt.h, rbt.cpp)
4.Min Heap class (mh.h, mh.cpp)
5.Ride struct (Ride.h)

2.1 Function Prototypes:

The following function prototypes provide an overview of the my program's structure:

main.cpp:

```
int main(int argc, char* argv[]);
```

RideManager class (func.h, func.cpp):

```
void labelDet(std::string op, std::string &label, std::string &rest);
void printRide(int rideNumber);
void printRides(int rideNumber1, int rideNumber2);
void insert(int rideNumber, int rideCost, int tripDuration);
void getNextRide();
void cancelRide(int rideNumber);
void updateTrip(int rideNumber, int new_tripDuration);
```

Red-Black Tree class (rbt.h, rbt.cpp):

```
void insert(int rideNumber, int rideCost, int tripDuration);
Ride* search(int rideNumber);
void printRide(int rideNumber);
void printRides(int rideNumber1, int rideNumber2);
Ride* getNextRide();
void cancelRide(int rideNumber);
void updateTrip(int rideNumber, int new_tripDuration);
```

Min Heap class (mh.h, mh.cpp):

```
void insert(Ride *ride);
void cancelRide(int rideNumber);
void updateTrip(Ride *ride, int new_tripDuration);
void upHeap(int i);
void downHeap(int i);
```

Ride struct (Ride.h):

```
struct Ride {
    int rideNumber, rideCost, tripDuration;
    Ride(int r, int c, int t);
    bool operator<(const Ride& rhs) const;
};
```

2.2 Time Complexity

Red-Black Tree Class Functions:

insert(): O(log N). The insertion operation in a Red-Black Tree has a time complexity of O(log N) since the tree is balanced.

search(): O(log N). Searching in a Red-Black Tree has a time complexity of O(log N) since the tree is balanced.

printRide(): O(log N). This function requires searching the tree for the ride, which takes O(log N) time.

printRides(): O(log N + M). In this case, M is the number of rides being printed. Searching for the start and end rides takes O(log N) time, and printing M rides takes O(M) time.

getNextRide(): O(1). Retrieving the next ride is an O(1) operation since the minimum element can be directly accessed using the ordered map.

cancelRide(): O(log N). Canceling a ride requires searching for the ride in the tree and then deleting it, both of which have O(log N) time complexity.

updateTrip(): O(log N). Updating a trip requires searching for the ride in the tree, which takes O(log N) time.

Min Heap Class Functions:

insert(): O(log N). Inserting an element into a Min Heap takes O(log N) time due to the upHeap() operation.

cancelRide(): O(N). Canceling a ride in a Min Heap requires finding the ride in the heap (O(N)) and then performing upHeap() or downHeap() operations (O(log N)). The overall time complexity is O(N).

updateTrip(): O(log N). Updating a trip in a Min Heap requires updating the ride's trip duration and then performing upHeap() or downHeap() operations. Since the ride is already found, the time complexity is O(log N).

upHeap(): O(log N). In the worst case, the upHeap() operation moves an element from the bottom to the top of the heap, which takes O(log N) time.

downHeap(): O(log N). In the worst case, the downHeap() operation moves an element from the top to the bottom of the heap, traversing the entire height of the balanced binary heap.

RideManager Class Functions:

printRide(): O(log N)

printRides(): O(log N + M)

insert(): O(log N)

getNextRide(): O(1)

cancelRide(): O(log N)

updateTrip(): O(log N)

All functions in RideManager Class call the related functions from the Red-Black Tree and Min Heap classes, so the time complexity is dependent on those functions.

2.3 Program Structure

The program begins execution in the main function, which handles file input/output and redirects the standard output to an output file. The main function also initializes a RideManager object that will manage the ride requests.

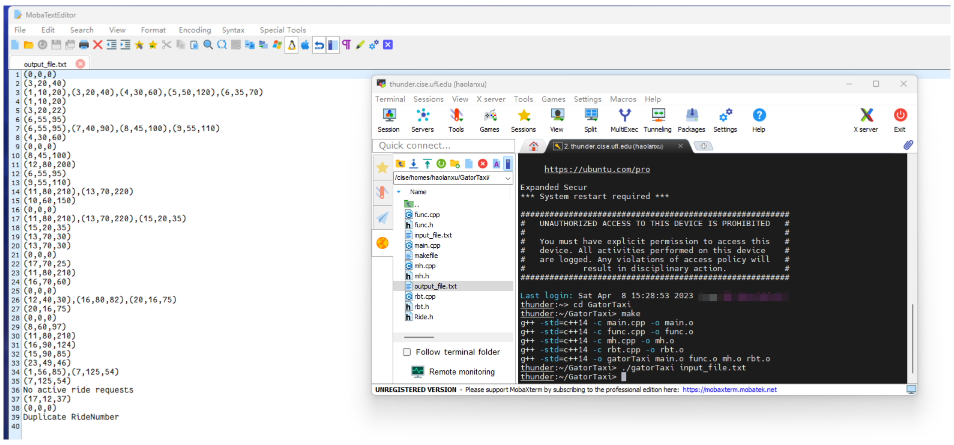
The RideManager class acts as the central hub for managing ride requests. It contains functions for determining operation labels, printing rides, inserting rides, retrieving the next ride, canceling rides, and updating trips. The RideManager class integrates the functionality of both the Red-Black Tree and Min Heap classes.

The Red-Black Tree class is used to store ride request data and perform operations such as insertion, searching, printing rides, getting the next ride, canceling rides, and updating trips. The tree is implemented using an ordered map (std::map) to store Ride objects.

The Min Heap class is used to efficiently retrieve the ride with the lowest ride cost and shortest trip duration. It supports operations such as insertion, canceling rides, and updating trips. The heap is implemented using a vector (std::vector) to store pointers to Ride objects.

The Ride struct defines the data structure for individual rides, including ride number, ride cost, and trip duration. It also includes an overloaded comparison operator to facilitate Min Heap operations.

2.4 Result



```
(0,0,0)
(3,20,40)
(1,10,20),(3,20,40),(4,30,60),(5,50,120),(6,35,70)
(1,10,20)
(3,20,22)
(6,55,95)
(6,55,95),(7,40,90),(8,45,100),(9,55,110)
(4,30,60)
(0,0,0)
(8,45,100)
(12,80,200)
(6,55,95)
(9,55,110)
(11,80,210),(13,70,220)
(10,60,150)
(0,0,0)
(11,80,210),(13,70,220),(15,20,35)
(15,20,35)
(13,70,30)
(13,70,30)
(0,0,0)
(17,70,25)
(11,80,210)
(16,70,60)
(0,0,0)
(12,40,30),(16,80,82),(20,16,75)
(20,16,75)
(0,0,0)
(8,60,97)
(11,80,210)
(16,90,124)
(15,90,85)
(23,49,46)
(1,56,85),(7,125,54)
(7,125,54)
No active ride requests
(17,12,37)
(0,0,0)
Duplicate RideNumber
```

```
     https://ubuntu.com/pro

Expanded Secur
*** System restart required ***

###############################################################
#   UNAUTHORIZED ACCESS TO THIS DEVICE IS PROHIBITED   #
#                                                               #
#   You must have explicit permission to access this   #
#   device. All activities performed on this device    #
#   are logged. Any violations of access policy will   #
#          result in disciplinary action.                       #
###############################################################

Last login: Sat Apr  8 15:28:53 2023 [redacted]
thunder:~> cd GatorTaxi
thunder:~/GatorTaxi> make
g++ -std=c++14 -c main.cpp -o main.o
g++ -std=c++14 -c func.cpp -o func.o
g++ -std=c++14 -c mh.cpp -o mh.o
g++ -std=c++14 -c rbt.cpp -o rbt.o
g++ -std=c++14 -o gatorTaxi main.o func.o mh.o rbt.o
thunder:~/GatorTaxi> ./gatorTaxi input_file.txt
thunder:~/GatorTaxi>
```

MobaTextEditor
File Edit Search View Format Encoding Syntax Special Tools
output_file.txt
thunder.cise.ufl.edu (haolanxu)
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X server Exit
Quick connect...
2. thunder.cise.ufl.edu (haolanxu)
/cise/homes/haolanxu/GatorTaxi/
Name
..
func.cpp
func.h
input_file.txt
main.cpp
makefile
mh.cpp
mh.h
output_file.txt
rbt.cpp
rbt.h
Ride.h
Follow terminal folder
Remote monitoring
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

Figure 1: Result in thunder

3 Conclusion

This report has presented the function prototypes and program structure for my project, illustrating the essential components and their relationships. The program that meets the project requirements efficiently manages ride requests and their associated data using a combination of Red-Black Trees, Min Heaps, and standard C++ containers.