COP 5536 Spring 2023

Programming Project

Due 11 Apr 2023, 11:59 pm EST

1. General

Problem description

GatorTaxi is an up-and-coming ride-sharing service. They get many ride requests every day and are planning to develop new software to keep track of their pending ride requests.

A ride is identified by the following triplet:

rideNumber: unique integer identifier for each ride.
rideCost: The estimated cost (in integer dollars) for the ride.
tripDuration: the total time (in integer minutes) needed to get from pickup to destination.

The needed operations are

1. **Print(rideNumber)** prints the triplet (rideNumber, rideCost, tripDuration).
2. **Print(rideNumber1, rideNumber2)** prints all triplets (r_x, rideCost, tripDuration) for which rideNumber1 <= r_x <= rideNumber2.
3. **Insert (rideNumber, rideCost, tripDuration)** where rideNumber differs from existing ride numbers.
4. **GetNextRide()** When this function is invoked, the ride with the lowest rideCost (ties are broken by selecting the ride with the lowest tripDuration) is output. This ride is then deleted from the data structure.
5. **CancelRide(rideNumber)** deletes the triplet (rideNumber, rideCost, tripDuration) from the data structures, can be ignored if an entry for rideNumber doesn't exist.
6. **UpdateTrip(rideNumber, new_tripDuration)** where the rider wishes to change the destination, in this case,
   a) if the new_tripDuration <= existing tripDuration, there would be no action needed.
   b) if the existing_tripDuration < new_tripDuration <= 2*(existing tripDuration), the driver will cancel the existing ride and a new ride request would be created with a penalty of 10 on existing rideCost . We update the entry in the data structure with (rideNumber, rideCost+10, new_tripDuration)
   c) if the new_tripDuration > 2*(existing tripDuration), the ride would be automatically declined and the ride would be removed from the data structure.

In order to complete the given task, you must use a min-heap and a Red-Black Tree (RBT). You must write your own code for the min heap and RBT. Also, you may assume that the number of active rides **will not exceed 2000**.

A min heap should be used to store (rideNumber, rideCost, tripDuration) triplets ordered by **rideCost**. If there are multiple triplets with the same **rideCost**, the one with the shortest **tripDuration** will be given higher priority (given all rideCost-tripDuration sets will be **unique**). An **RBT** should be used to store (rideNumber, rideCost, tripDuration) triplets ordered by **rideNumber**. You are required to maintain pointers between corresponding nodes in the min-heap and RBT.

GatorTaxi can handle only one ride at a time. When it is time to select a new ride request, the ride with the lowest rideCost(ties are broken by selecting the ride with the lowest tripDuration) is selected (Root node in min heap). When no rides remain, return a message "No active ride requests".

Programming Environment

You may use either Java or C++ for this project. Your program will be tested using the Java or g++ compiler on the thunder.cise.ufl.edu server. So, you should verify that it compiles and runs as expected on this server, which may be accessed via the Internet.

Your submission must include a makefile that creates an executable file named **gatorTaxi**.

2. Input and Output Requirements

Your program should execute using the following
For c/c++:
$ ./ gatorTaxi file_name
For java:
$ java gatorTaxi file_name

Where file_name is the name of the file that has the input test data.

Input Format

Input test data will be given in the following format.

Insert(rideNumber, rideCost, tripDuration)
Print(rideNumber)
Print (rideNumber1,rideNumber2)
UpdateTrip(rideNumber, newTripDuration)
GetNextRide()
CancelRide(rideNumber)

Example 1:

Insert(25,98,46)
GetNextRide()

GetNextRide()
Insert(42,17,89)
Insert(9,76,31)
Insert(53,97,22)
GetNextRide()
Insert(68,40,51)
GetNextRide()
Print(1,100)
UpdateTrip(53,15)
Insert(96,28,82)
Insert(73,28,56)
UpdateTrip(9,88)
GetNextRide()
Print(9)
Insert(20,49,59)
Insert(62,7,10)
CancelRide(20)
Insert(25,49,46)
UpdateTrip(62,15)
GetNextRide()
Print(1,100)
Insert(53,28,19)
Print(1,100)

The output for this would be:

(25,98,46)
No active ride requests
(42,17,89)
(68,40,51)
(9,76,31),(53,97,22)
(73,28,56)
(0,0,0)
(62,17,15)
(25,49,46),(53,97,15),(96,28,82)
Duplicate RideNumber

Print(rideNumber) query should be executed in O(log(n)) and Print(rideNumber1,rideNumber2) should be executed in O(log(n)+S) where n is the number of active rides and S is the number of triplets printed. For this, your search of the RBT should enter only those subtrees that may possibly have a ride in the specified range. All other operations should take O(log(n)) time.

Output Format

- Insert(rideNumber, rideCost, total_time) should produce no output unless rideNumber is a duplicate in which case you should output an error and stop.

- Print(rideNumber) will output the (rideNumber,rideCost,tripDuration) triplet if the rideNumber exists. If not print (0,0,0).

- Print(rideNumber1,rideNumber2) will output all (rideNumber,rideCost,tripDuration) triplets separated by commas in a single line including rideNumber1 and rideNumber2; if they exist. If there is no ride in the specified range, output (0,0,0). You should not print an additional comma at the end of the line. Other output includes a string message indicating that there are no rides.

- UpdateTrip(rideNumber,newTripDuration) will not print anything.

- CancelRide(rideNumber) will not print anything.

- GetNextRide() will output the ride with lowest rideCost and if there are multiple triplets with the same rideCost, the one with the shortest tripDuration will be printed.

- All output should go to a file named "**output_file.txt**".

3. Submission

Do not use nested directories. All your files must be in the first directory that appears after unzipping.

You must submit the following:

1.Makefile: You must design your makefile such that 'make' command compiles the source code and produces executable file. (For java class files that can be run with java command)

2. Source Program: Provide comments.

3. REPORT:
- The report should be in PDF format.
- The report should contain your basic info: **Name, UFID and UF Email** account
- Present function prototypes showing the structure of your programs. Include the structure of your program.

To submit, please compress all your files together using a zip utility and submit to the Canvas system. You should look for the Assignment Project for the submission.
Your submission should be named **LastName_FirstName.zip.**

Please make sure the name you provided is the same as the same that appears on the Canvas system. Please do not submit directly to a
TA. ***All email submissions will be ignored without further notification. Please note that the due date is a hard deadline. No late submission will be allowed. Any submission after the deadline will not be accepted.***

4. Grading Policy

Grading will be based on the correctness and efficiency of algorithms. Below are some details of the grading policy.
Correct implementation and execution: 60%
Important: Your program will be graded based on the produced output. You must make sure to produce the correct output to get points. There will be a threshold for the running time of your program. If your program runs slow, we will assume that you have not implemented the required data structures properly.

Comments and readability: 15%
Report: 25%

You will get **negative points** if you do not follow the input***/output or submission requirements above. Following is the clear guidance of how your marks will be deducted.***

Source files are not in a single directory after unzipping: -5 points
Incorrect output file name: -5 points Error in make file : -5 marks make file does produce an executable file that can be run with one of the following commands: -5

./ gatorTaxi file_name

java gatorTaxi file_name
Hard coded input file name instead of taking as an argument from the command prompt: -5 points
Additional comma at the end of the line of Print rideNumber query: -5 points
Any other input/output or submission requirement mentioned in the document: -3 points

Also, we may ask you to fix the above problems and demonstrate your projects.

5. Miscellaneous

- **Do not use complex data structures provided by programming languages**. You have to implement min-heap and RBT data structures on your own using fundamental data structures such as pointers. **You must not use any Map related libraries.**

- Your implementation should be your own. **You have to work by yourself for this assignment** (discussion is allowed). **Your submission will be checked for plagiarism.**