COP 5536 Fall 2019

Programming Project

Due 25th Nov, 2019, 11:55 pm EST

1. General

Problem description

Wayne Enterprises is developing a new city. They are constructing many buildings and plan to use software to keep track of all buildings under construction in this new city. A building record has the following fields:
buildingNum: unique integer identifier for each building.
executed_time: total number of days spent so far on this building.
total_time: the total number of days needed to complete the construction of the building.

The needed operations are:
1. Print (buildingNum) prints the triplet buildingNume,executed_time,total_time.
2. Print (buildingNum1, buildingNum2) prints all triplets bn, executed_tims, total_time for which buildingNum1 <= bn <= buildingNum2.
3. Insert (buildingNum,total_time) where buildingNum is different from existing building numbers and executed_time = 0.

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 the min heap and RBT. Also, you may assume that the number of active buildings **will not exceed 2000**.

A min heap should be used to store (buildingNums,executed_time,total_time) triplets ordered by **executed_time**. You mwill need a suitable mechanism to handle duplicate executed_times in your min heap. An **RBT** should be used store (buildingNums,executed_time,total_time) triplets ordered by **buildingNum**. You are required to maintain pointers between corresponding nodes in the min-heap and RBT.

Wayne Construction works on one building at a time. When it is time to select a building to work on, the building with the lowest executed_time (ties are broken by selecting the building with the lowest buildingNum) is selected. The selected building is worked on until complete or for 5 days, whichever happens first. If the building completes during this period its number and day of completion is output and it is removed from the data structures. Otherwise, the building's executed_time is updated. In both cases, Wayne Construction selects the next building to work on using the selection rule. When no building remains, the completion date of the new city is output.

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 **risingCity**.

2. Input and Output Requirements

Your program should execute using the following
For c/c++:
```
$ ./ risingCity file_name
```
For java:
```
$ java risingCity 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(buildingNum, total_time)
Print(buildingNum)
Print (buildingNum1, buildingNum2)
```

You cannot insert a building for construction to the data structures unless global time equals to the arrival time of the construction. All the time data are given in days.

Following is an example of input.

```
0: Insert(5,25)
2: Insert(9,30)
7: Insert(30,3)
9: Print (30)
10: Insert(1345,12)
13: Print (10,100)
14: Insert(345,14)
39: Insert(4455,14)
```

The number at the beginning of each command is the global time that the command has appeared in the system. You must have a global time counter, which starts at 0. You can read the input command only when the global time matches the time in the input command. You can assume this time is an integer in increasing order. When no input remains, construction continues on the remaining buildings until all are complete.

PrintBuilding (buildingNum) query should be executed in O(log(n)) and PrintBuilding (buildingNum1 1, buildingNum2) should be executed in O(log(n)+S) where n is number of active buildings 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 building in the specified range. All other operations should take O(log(n)) time.

Output Format

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

· PrintBuilding (buildingNum) will output the (buildingNum,executed_time,total_time) triplet if the buildingNum exists. If not print (0,0,0).

· PrintBuilding (buildingNum1, buildingNum2) will output all (buildingNum,executed_time,total_time) triplets separated by commas in a single line including buildingNum1 and buildingNum2; if they exist. If there is no building in the specified range, output (0,0,0). You should not print an additional comma at the end of the line.

. Other oupt includes completion date of each building and completion date of city.

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

```
./ risingCity file_name
java risingCity 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 buildingNum 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.**