COP 5536 Spring 2018

Programming Project

Due Date: Mar 30th , 2018, 11:55 pm EST

1. General

Problem description

ABC company is developing a new operating system. As a software developer in the company you are required to implement a job scheduler for this new operating system. It has been decided that when the processor becomes free, the scheduler will assign to it a job that has been run for the least amount of time so far. This job will run for the smaller of 5ms and the amount of remaining time it needs to complete. In case the job does not complete in 5ms it becomes a candidate for the next scheduling round. A job has the following fields:
jobID: Unique ID for each job.
executed_time: The amount of time for which the job has been scheduled so far.
total_time: Total time required to complete the job. So, the remaining time to complete is total_time-executed_time.

All times are in milliseconds (ms). The requirements of the scheduler are
1. Dispatch a job with the lowest executed_time to the processor. Ties may be broken arbitrarily.
2. Print (jobID,executed_time,total_time) triplet for a given jobID.
3. Print (jobID,executed_time,total_time) triplets for all jobIDs in the range [low,high].
4. NextJob(jobID,executed_time,total_time) prints the triplet for the job with smallest ID greater than a given jobID. Print (0,0,0) if there is no such job.
5. PreviousJob(jobID,executed_time,total_time) prints the triplet of the job with the greatest jobID that is less than to a given jobID. Print (0,0,0) if there is no such job.
6. Insert (jobID,total_time) insert a new job into the heap and RBT.

In order to complete the scheduler you must use a min heap and a Red Black Tree (RBT).
The key for the min heap is executed_time and that for the RBT is jobID. You are required to maintain pointers between correspondent nodes in the min heap and the RBT. The max size of the min heap in unknown, you should use array doubling whenever the current capacity is exceeded.

When a job is dispatched for processing, you should remove it from both data structures if the job will complete during this processing. Otherwise, you should increase executed_time by 5ms.

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 **jobscheduler**.

2. Input and Output Requirements

Your program should execute using the following
For c/c++:
$ ./ jobscheduler file_name
For java:
$ java jobscheduler 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(jobID,total_time)
PrintJob(jobID)
PrintJob(jobID1,jobID2)
NextJob(jobID)
PreviousJob(jobID)

You cannot insert a job to the data structures unless global time is equals to the arrival time of the job. All the time data are given in milliseconds.

Following is an example input.

0: Insert(5,25)
2: Insert(9,30)
7: Insert(30,3)
9: PrintJob(30)
10: Insert(1345,12)
13: PrintJob(10,300)
14: Insert(3455,14)
17: NextJob(30)
31: PreviousJob(345)
39: Insert(4455,14)

The number in 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 0 when your program starts to keep track of the current system time. 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. This method is used to simulate an actual scenario in an operating system where commands can appear interactively. End of file can be used to terminate the program. The dispatcher dispatches jobs between commands. When the end of file is encountered, the dispatcher schedules the remaining jobs and terminates.

PrintJob(jobID), NextJob(jobID), PreviousJob(jobID) queries should be executed in O(log(n)) and printJob(jobID1,jobID2) should be executed in O(log(n)+S) where S is the reported nodes.

In order to achieve O(log(n)+S) time you may move to a subtree of a node only if the node is contained in the range. Insert and delete from RBT is also O(log(n)).

Output Format

You should insert the records in to min heap and RBT for Insert(jobID,executed_time,total_time) and you should not produce any output.
PrintJob(jobID) will output the (jobID,executed_time,total_time) triplet if the jobID exists. If not print (0,0,0).
printJob(jobID1,jobID2) will output all (jobID,executed_time,total_time) triplets separated by commas in a single line including jobID1 and jobID2 if they exists. If nob job is found print (0,0,0). You should not print an additional comma at the end of the line.
NextJob(jobID) will output (jobID,executed_time,total_time) triplet of the of the lowest jobID that is greater than to a given jobID. Print (0,0,0) if there is no next job.
PreviousJob(jobID) will output (jobID,executed_time,total_time) triplet of the greatest jobID that is less than to a given jobID. Print (0,0,0) if there is no previous job.

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 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 Canvaas system. Please do not submit directly to a TA. All email submissions will be ignored without further notification. Please note that the due day 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. Please note that global time counter is only to keep track of the time in the simulator. Do not use sleep functions to sleep your actual program.

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 a 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

./ jobscheduler file_name

java jobscheduler 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 PrintJob query: -5 points

Any other input/output or submission requirement mentioned in the document: -3 points

In addition we may ask you to fix 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 primitive 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.