Project Report

Name:Ziyuan Guan

Student ID: 94722121

Mail:ziyuan.guan@ufl.edu

Submitted Data:03/30/2018

1. Introduction

The project asked us to implement the the MinHeap and Red-Black Tree an use them to design a job_scheduler program. The main point of the project is the understanding of the data structure principle and operations. The programs are written in Java.

2. The Architecture

The project contains four source files. MinHeap, RBTree, Process and jobscheduler. The jobscheduler is applied to read the input files and deliver the jobs. The MinHeap and the RBTree are two significant data structures.I used the node and the key to present the value and the position of the tree. Process is a class which provides a type of variables used to build up the ***ArrayList***.

(i)Jobscheduler

Step1: Using the ***String []args*** to read the name of the input file.

Step2: Using Scanner to read the contents of the input.

Step3: Create the original variables such as global_time,node_count.

Step4: Create two ArrayLists: ***processes*** and ***jobs***. ***Processes*** is used to store the ***execution_time(the time has used), total_time*** and ***jobs*** is used to store the ***Job_ID.*** With these two lists, we can build the tree and insert the node one by one.

Step5: Compare with the ***global_time*** and ***insert_time***. If

global_time <***insert_time***, then keep dispatching the jobs. We need to use the ***heap.top*** to get the job who has the least execution_time. And then compare the ***left_time*** with 5ms, if bigger, than we continue, update the heap and add the time with ***global_time*** and ***execution_time***. If not ,we will delete the node from the heap and ***JobID*** from the R-B-Tree and also upgrade the time variables.

Step6: In the outer loop, we need to judge whether the input file still has the next input. If yes, we will continue when the time is right. If it reaches the end of file. The program is stopped.

Step7:In the ***jobscheduler***, I also wrote several functions used to process the jobs. ***Switcher:*** It is used to switch the command.***Insert(In the Switcher):*** It is used to insert the job. I used the ***processes.add()*** and ***jobs.add()*** to add the element into the ***ArrayList. PrintJob:*** It is used to print the jobs. Firstly, I judged the print mode, single one or a range. Then used the ***RBTree.pop()*** to get the smallest ***JobID,RBTree.next()*** and ***RBTree.previous()*** to get the key from different position**.**If the left bounds is smaller than the minimum ***JobID,*** then we will use the minimum.If not, we will print the job until the ***JobID*** reaches the right bound.Here is a bug that we need to consider the selection of the ***JobID i***s based on the value of ID not the order of the insertion. Same as the ***NextJob*** and ***PreviousJob.***

(ii)MinHeap

It's easily to build the heap. The key is the ***execution_time*** of each job. The heap has several operations: ***push***(insert the new node),***remove***(delete the node),***top***(get the root node key),***fix***(fix the heap after inserting and removing).

(iii)RBTree

The Red Black Tree is a little bit difficult since we need to have more operations. Firstly, I defined RBTree Node to represent the node and each node has its own parent and two kinds of colorsThe key of RBTree is the ***JobID***. Then the basic operations are ***left_rotation***, ***right_rotation***,***insert***,***remove***,***pop(used to return the root)***,***fix***. And also, I had some specific operations like ***next***, ***previous***,***minimum*** and ***maximum***. The minimum is the root and the maximum is the rightest node. This is helpful in the next programming. Especially in the ***PrintJob***.

(iv)Process

The class of ***Process*** is used to create the ArrayList sincewe need to have a public structure to store all of the variables.

3. The Results

With the test of the sample_input3, the answer is completely correct. Makefile can be used to create the .class files.