Advanced Data Structures

COP 5536

Spring 2018

Project report

Maogeng Qin

81782381

qinmaogeng@ufl.edu

1, Associated Files

test.h: including class definition of RBT, Min heap, and job. And the definition of the functions of each class;

Main.cpp : Including the definition of struct input and the function ReadData. It also contain the main function of the program. In the main program, the job will be managed and dispatched and the command get from the input file will be implemented in certain time according to the time stamp of each command,

2, Classes

(1) **Class job:** the class job contains the execute time, Job ID, total time and remaining time of a job;

(2) **Class RBTNode:** class that represents the Node structure of the RBT, it contains the pointer to the Node's left and right child and its parent. It also contains the color and key which is the jobID of one job. Plus, a the pointer to the class job is also contained in the RBTNode as the data of this node.

(3) **Class RBT:** class of the RBT, it contains the Root of the RBT and the function needed for the RBT for this program, which is:

RBTNode* RBT::search(RBTNode *x,int key):
recursively search a Node with a given key, return it
void RBT::LRotate(RBTNode* &root, RBTNode* x):
left or what we called in course RR rotate function
void RBT::RRotate(RBTNode* &root, RBTNode* y):
right or what we called in course LL rotate function
void RBT::insertFix(RBTNode* &root, RBTNode* node):
the function that modifies the RBT after one node is inserted:
void RBT::insert(RBTNode* &root, RBTNode* node):
insert function,first search the position where the node should be inserted and then fix the RBT
void RBT::removeFix(RBTNode* &root, RBTNode *node, RBTNode *parent):
remove fix function,do the color change and rotate according to cases
void RBT::remove(RBTNode* &root, RBTNode *node):
remove function,first remove the node and then fix the RBT
RBTNode* RBT::next(int key):
function of finding the node with the least ID bigger than the given ID
RBTNode* RBT::previous(int key):
function of finding the node with the largest ID less than the given ID
void RBT::findRange(RBTNode* root, int low, int high):
recursively search the node between the given two IDs

(4) **Class PHeap:** class of min heap, which is a pairng heap, it contains the leftist child, left and right sibling of each heap. It also contain a pointer pointing to a job as its data, it contains such operation needed for the job scheduler as shown below::

PHeap* PHeap::merge(PHeap *p, PHeap *q):

function of merging two heaps, set the heap with the smaller key to the leftest child of the larger heap

PHeap* PHeap::insert:

function of insert a new node, just merge it with the existed heap

PHeap* PHeap::combine(PHeap* p):

combine the heap node from left to right after deletemin or increase key

PHeap* PHeap::deleteMin(PHeap* p):

deleteMIn operation

PHeap* PHeap::increaseKey(int val, PHeap *p):

increasekey operation

3, Other struct and functions

Struct input:

the struct containing the information reading from every line of the input file

ReadData(input in, string line):

this function convert the string line read from the file to some value in the input struct

4, Program structure and main function logic

This program is a jobs scheduler, the main function will first read one line from the input file and then extract the information of each line to the command including the command name, time stamp and the value of the command. The scheduler will execute in a loop in order to simulate the real time, every loop the global time will increase by 1. When an insert command come, the scheduler will insert the jobID of a job to the RBT and the execute time of the job to the min heap, both of the RBT and min heap node will also be inserted a pointer pointing to the inserted job, in that way two data structure is connected. If one job is dispatched, its execute time will increase by one, it will do 5ms or its remaining if the remaining is less than 5ms. After running 5 ms in a dispatcher time, the heap will do an increase key by 5 which will update the job with the least executed time. If the file reach the EOF and there are no jobs in the scheduler which is indicated by jobCount == 0. The scheduler will terminates.

5, Running instruction

Using make to compile the source code, and then ./jobscheduler filename to get command from a data with the given filename.