ADVANCED DATA STRUCTURES (COP5536)
SPRING 2018
PROJECT REPORT

Submitted By:
SHRIVINAYAK BHAT
47971819
shrivinayak.bhat@ufl.edu

This project implements Job scheduler which is a simplified version of CFS ie Completely Fair Scheduler of Linux OS.
The data structures used here include Min Heap and Red Black Trees.The implementation language is C++.

The source code has 3 classes which provide the required abstraction.

1) Class Scheduler -

This is the top level class which provides the scheduler functionality as described in the problem statement. The methods that belong to this class are.

- void printjob(int JobID); prints the matching job id
- void printjob(int JobIDlow,int JobIDhigh); prints the range of jobIDs
- void nextjob(int JobID); prints the next greatest job wrt inorder traversal
- void prevjob(int JobID); prints last highest job id wrt inorder traversal
- void insert(int JobID, int time); inserts new job
- scheduler(); constructor that initializes the counter and creates objects of class heap & rbt
- void syncTime(int time); calls dispatch method until counter matches the timestamp of current comand
- int dispatch(int); schedules jobs
- int ifjob(); checks if there are any jobs in the queue.

2) Class heap

This class facilitates the Min Heap data structure.

The methods are

- heapnode* insert(int, int, int,rbtnode*); insert a new node into the heap.
- struct heapnode* removeMin(); remove the item from top of the heap ie minimum executed time.
- void swapJob(struct heapnode* a,struct heapnode* b); swap the positions of two nodes
- void heapify(); fix heap properties after a remove min
- void updateMin(int exec_time) // updates the root and re arranges the heap
- heap(int) // constructor
- void execute(int) // function to execute a job

---

3) Class rbt

This class is responsible for all features of a red black tree data structure.

The methods are

- void rotateleft(rbtnode *&, rbtnode *&); do a left rotation
- void rotateright(rbtnode *&, rbtnode *&); do a right rotation
- void fixtree(rbtnode *&, rbtnode *&); fix rbt properties after insert operation
- rbt(); constructor to initalize the tree
- rbtnode* insert(const int &n); insert new node into the tree
- rbtnode* findnode(int jobid); find node based on job id
- void nextnode(int jobid); find next node in inorder traversal
- void prevnode(int jobid); frind prev node in inorder traversal
- void inorder(int, int); print all values in inorder travel in the range low-high
- void deletenode(rbtnode*); - delete a node from the tree
- void fixviolation(rbtnode*); fix violation caused by deletion

The flow of control is shown in the next page

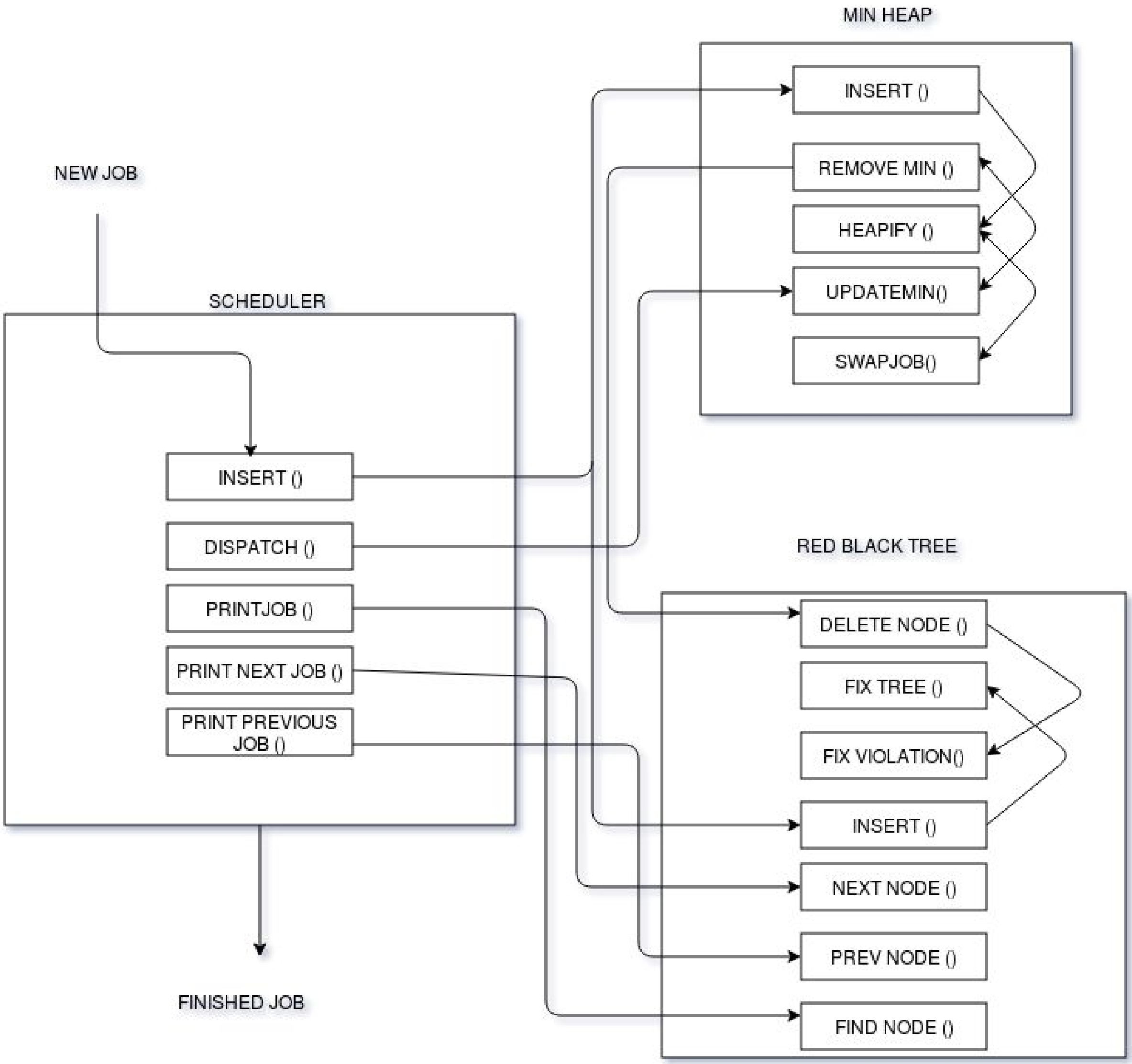

Figure showing the control flow of a scheduler.