CS6140: Advanced Programming Lab
Assignment 5

Lead TA: Sandeep N. K.

August 28, 2014

Goal of the assignment

The goal of this assignment is to implement an efficient dictionary using a Red Black Tree. The assignment will require you to make use of the graph visualization tool graphviz that was introduced in the last class. We will also plot the black height of the tree as a function of the number of nodes. For ease of visualization we will use another tool – gnuplot.

Requirement

Implement a dictionary using a Red Black Tree. The input file is a set of jobs where each job has a job-ID, priority, and duration. The input file format is the same as in the earlier assignments. Your dictionary is supposed to support insert and search. Note that you need not support deletions from the dictionary. In addition to insert and search, your program must support a print function which dumps the state of the Red Black Tree into a "dot" file. The "dot" file can be converted to an png/eps format using the appropriate "dot" commands.

You should consider reusing the Node class defined earlier for your Binary Search Tree assignment. A part of the public interface for the class is as given below:

```
class RBT{
public:
    RBT();
    ˜RBT();
    void insert(Node*);
    Node* search(int);
    printTree();
};
```

You may implement additional functions as needed. Split the class definition and the functions into RBT.h and RBT.cpp files respectively. Finally, have a main function (implemented in main.cpp) which supports one or more of the following command line options

- -i *datafile*: inserts the set of jobs in the datafile and creates a RBT.
- -t *testcases*: executes the set of test cases on the current RBT (possibly empty). Note that -t may be executed with -i option.
- -i *datafile* -h *filename*: inserts the set of jobs in the datafile and creates a RBT. In addition plots the black height of the tree at regular intervals. See below for details.

The file format for the testcases file is as below.

- I jobId duration priority
  Insert the above job in the RBT. In case of inserting a job with duplicate jobId, flag an error message: "Duplicate jobId. Insert unsuccessful."
- S jobId
  Search the job with id as jobId and print the details (duration and priority).
- P *filename*
  Print the state of the current RBT into a "dot" file name *filename*.dot. You should also execute the appropriate command from the C++ code to generate a *filename*.eps or *filename*.png file. One way to execute a shell command from the C++ program is by using the **system**("*command* ") function. You will need to include **cstdlib** to use the system command.

Plot Black Height as you build the tree

If your program is given the command line option of -i *datafile* -h *filename*, following needs to be done. Assume that the datafile contains n jobs to be inserted. We would like to see the black height of the tree as we build the tree from 0 to n nodes. To do this, compute the black height of the tree at about 10 different times as you build the tree. That is, compute the black height after you have inserted $\lceil n/10 \rceil$, $2\lceil n/10 \rceil$, $3\lceil n/10 \rceil$, $\ldots$, n nodes. For each of these values of number of nodes in the tree, plot the black height of the root at that time and also the function $2\log(\textit{number of nodes} + 1)$. Output these values in a file named *filename* which can be executed using gnuplot. Use a line graph to display both the curves as a function of the number of nodes.

Follow the submission guidlines as in the earlier assignments and submit a *roll-number*.tar.gz file.