CS 3353 Data Structure and Algorithm Analysis

Assignment 05: Full Marks 100
(Due Date: 11/22/2020, 11:59 PM CDT)

This assignment is to be completed individually and should be programmed using Java. You are not supposed to use any Collection frameworks and should be completed using AVL tree.

All these assignments must be completed on CSX machine. Logging on to the CSX account and other relevant information can be easily accessed in: http://www.cs.okstate.edu/loggingon.html

Brief Background:

We will use AVL tree is used to schedule processes in a fair way such that all processes will be able to use the processor to complete its task.

There are number of processes in the system waiting for the processor. Instead of maintaining a queue for these processes, use a AVL tree where each node stores the process related information. The idea behind the schedule is as follows:

1. Each process has a virtual time associated with it and the processes are arranged in the tree based on it.
2. Process with the minimum virtual time is executed first. As the process executes its virtual time is increased by 1.
3. Each process takes certain (but equal) amount of processor time. If the process does not complete its task within the given time, then it will be reinserted into the tree with the increased value of virtual time as given by step (2). So, for how much processor time each process would require depends on the associated burst time.
4. Once the step (2) for one process is completed, another processes with the smallest virtual time is selected from the AVL tree for execution and the process repeats again as long as there is any process left in the tree.
5. If the process completes its task within the given time, then it will be removed from the tree.

Problem Description:

There is an input file (*processesInformation.txt*) that contains the following three columns:

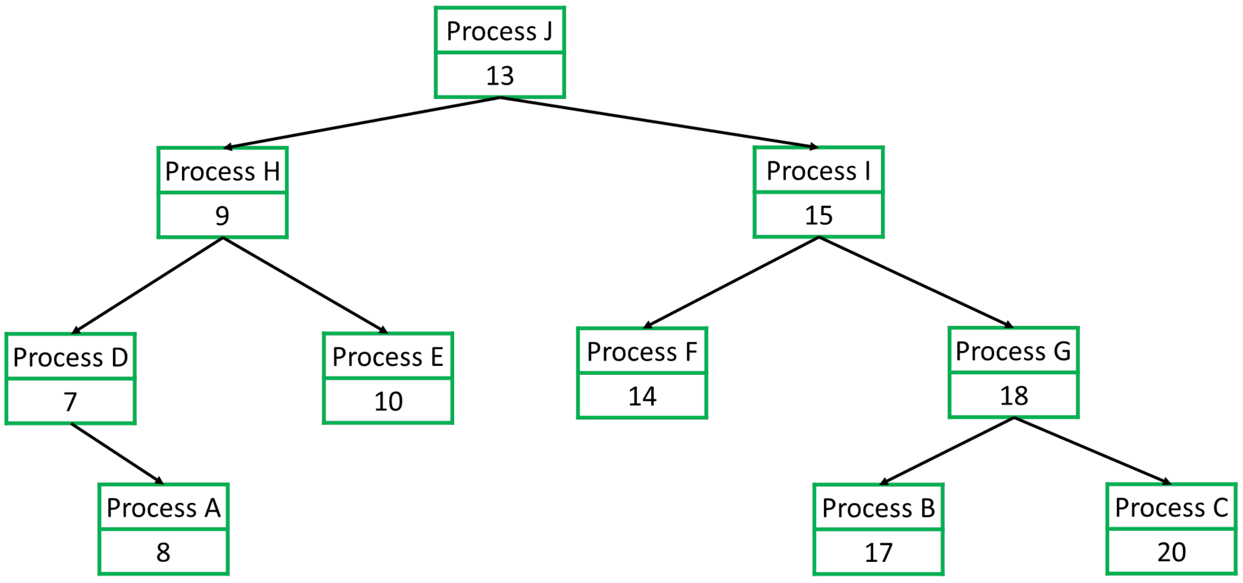
- **Process Name**: Name of the process
- **Burst Time**: Number of times the process will execute.
- **Virtual Runtime**: Processes are arranged in the tree based on the virtual runtime.

Consider for example, you have following information related to the process:

Process Name	Burst Time	Virtual Runtime
Process J	5	13

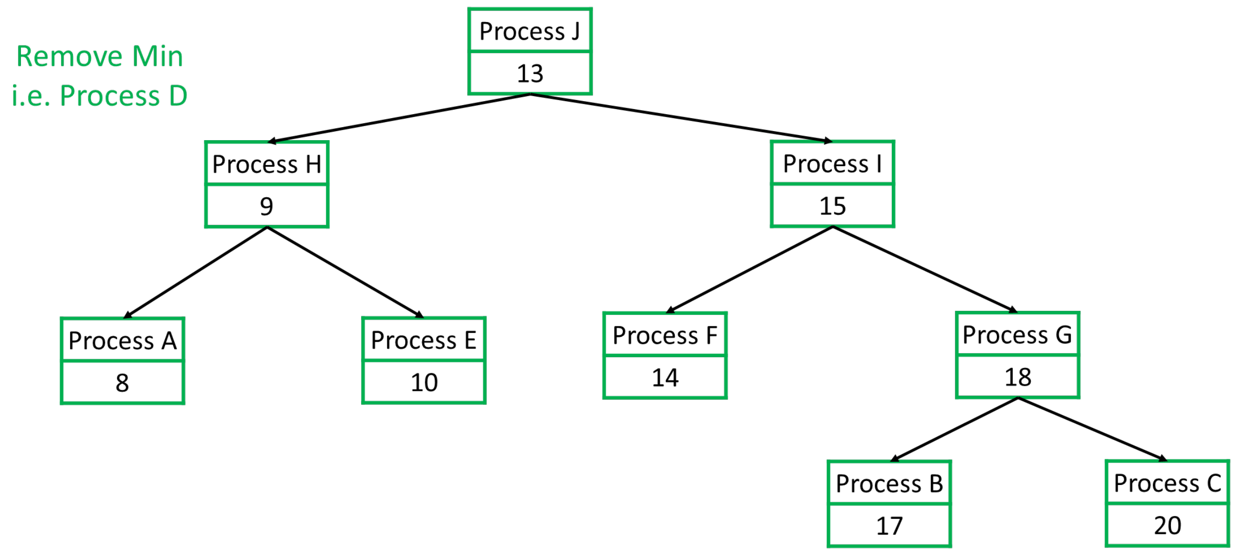
Process H	7	9
Process I	6	15
Process D	4	7
Process E	5	10
Process F	8	14
Process G	6	18
Process A	4	8
Process B	7	17
Process C	8	20

Read the file and insert the process in the AVL tree as follows:

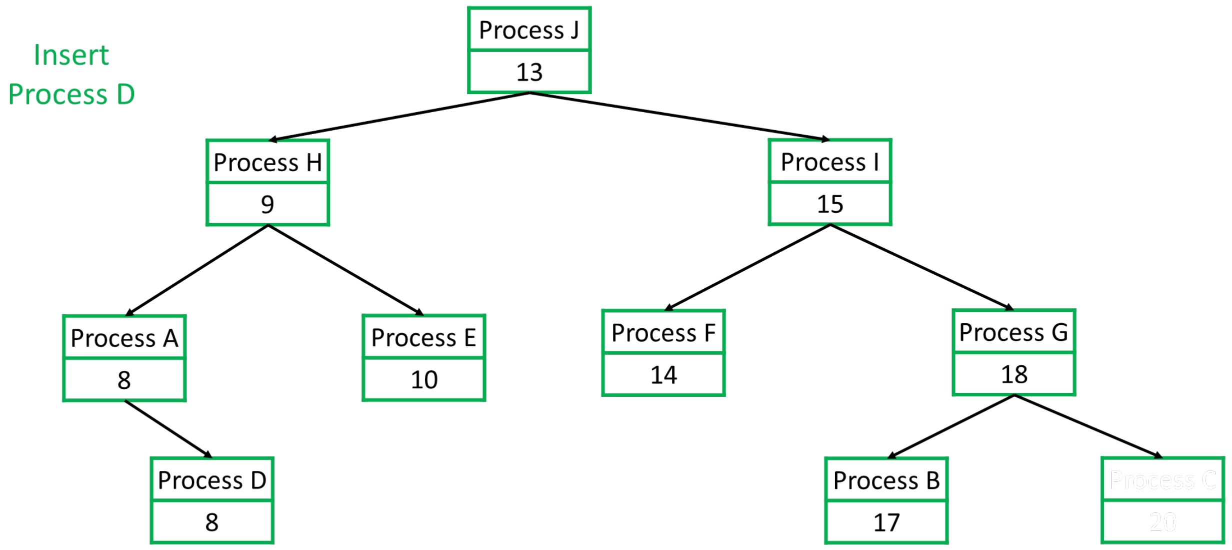


Refer to the lecture video for the AVL tree.

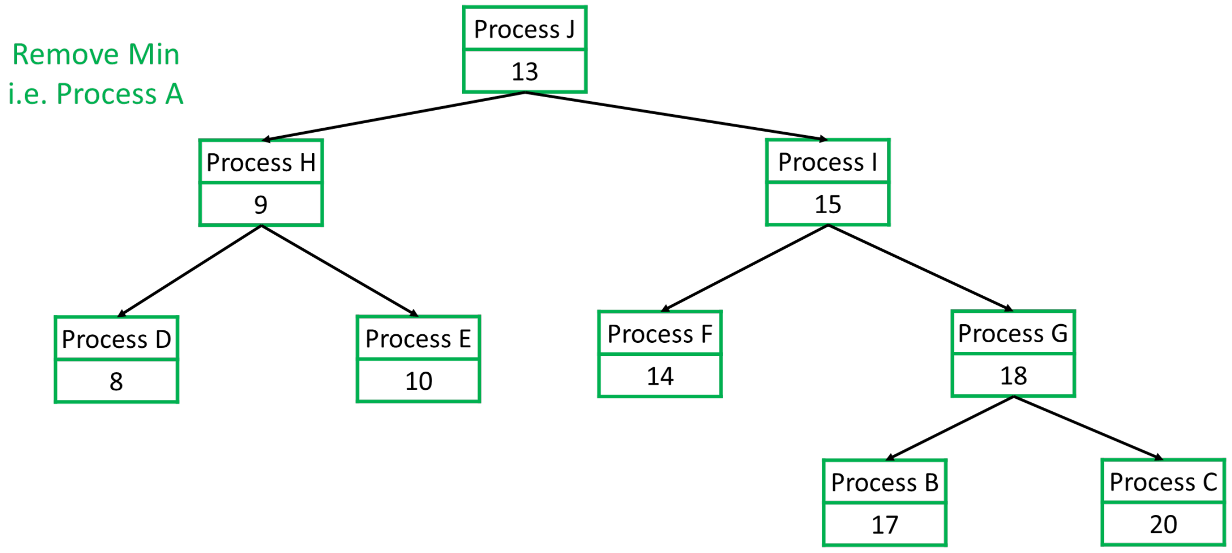
From the tree, you will choose the process with the minimum virtual runtime. In this case, the process D has the minimum virtual runtime. So, you will remove the process with the minimum virtual runtime and you will print that process information (process name and the burst time) on the screen and decrease the burst time by 1. The burst time for the process D will be 3. After this the tree looks like this:



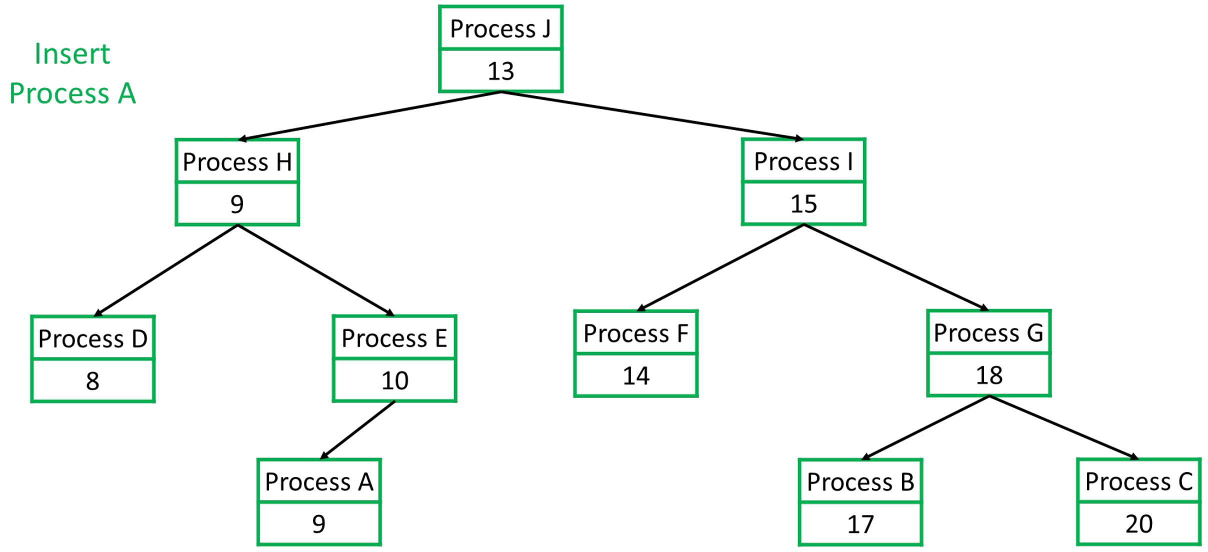
Since the process D does not have burst time of 0, it needs to be re-inserted into the tree again. But before inserting, you need to increase its virtual runtime by 1. The virtual time for the process D will be now 8. So, after inserting the process D into the tree, it looks like:



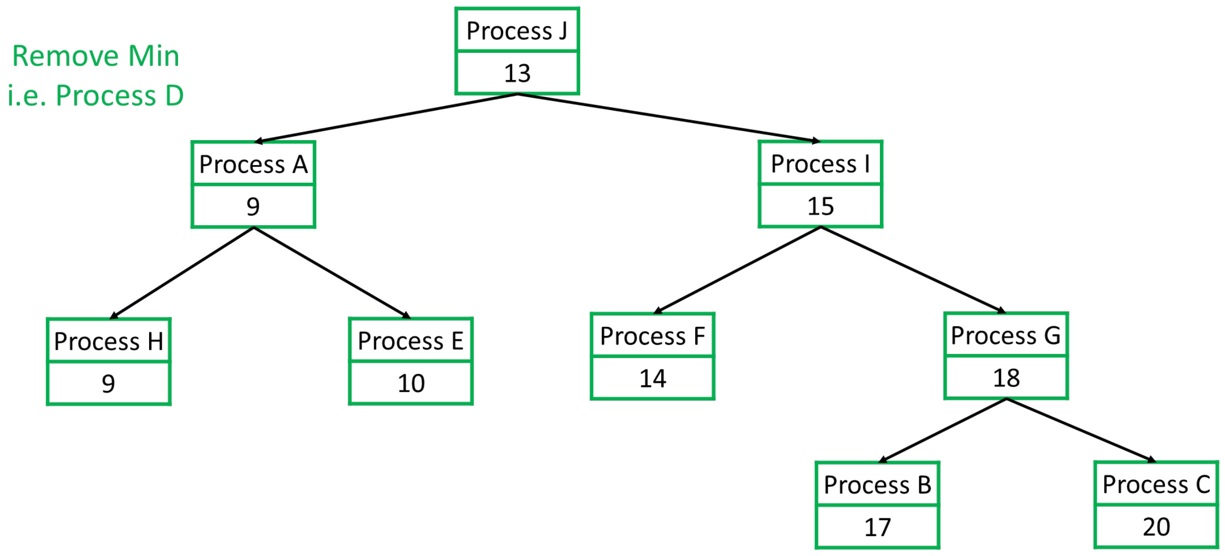
Now, from the tree, you will again choose the process with the minimum virtual runtime. In this case, the process A has the minimum virtual runtime i.e. 8. So, you will remove the process A from the tree, and you will print that process information (process name and the burst time) on the screen and decrease the burst time by 1. The burst time for the process A will be 3. After removing the process A from the tree, the tree looks like this:



Since the process A does not have burst time of 0, it needs to be re-inserted into the tree again. But before inserting, you need to increase its virtual runtime by 1. The virtual time for the process A will be now 9. So, after inserting the process A into the tree, it looks like:



From the tree, you will again choose the process with the minimum virtual runtime. In this case, the process D has the minimum virtual runtime. So, you will remove the process D from the tree and you will print that process name on the screen and decrease the burst time by 1. The burst time for the process D will be 2. After removing the process D, the tree looks like this:



The process should continue as long as you do not get an empty tree.

Program Interface:

Once you run the program, the program should read the file and perform the removeMin and insert operation depending upon the burst time. Accordingly, you need to display the process information like Process Name and the Burst Time.

Gradings would be as follows:

- Implementation of insertion operation in the tree. **[20 Points]**
- Implementation of rotation maintaining AVL tree property. **[40 Points]**
  - **LL Rotation** **[7 Points]**
  - **RR Rotation** **[7 Points]**
  - **LR Rotation** **[13 Points]**
  - **RL Rotation** **[13 Points]**
- Implementation of removeMin operation in the tree. **[20 Points]**

- Run time analysis:
  - Insertion operation **[5 Points]**
  - Rotation operation **[10 Points]**
  - Remove minimum operation **[5 Points]**

Failure to compile the program correctly will deduct 20 points automatically.
Additional points to consider:

- Your program will be evaluate based on the correctness, efficient programming.
- Your program should consider OOPs concepts.

- Appropriate points will be deducted if the instruction is not followed properly.

Submission Guidelines:

- The main driver file needs to have the following header information:
  - Author Name:
  - Email: <Use only official email address>
  - Date:
  - Program Description:

- Use comments throughout your program.

- Each class and the method should be properly commented:
  - Mention each argument type, purpose
  - Method description
  - Return type of the method

- Failure to follow standard programming practices will lead to points deduction, even if the program is running correctly. Some of the common places where you could lose points are:
  - Program not compiling successfully: -20 points
  - No comments on code (excluding class and method): -5 points
  - No comments on class and methods: -5 points
  - Not writing meaningful identifiers: -5 points
  - Failure to consider OOPs concepts: -10 points

- Efficient implementation of the code is very important. All of your list operations should be efficient i.e. it should not exceed the run time as we discussed in the lecture videos.