CSE201 Advanced Programming
Assignment 04
IIIT-Delhi. 27th Nov 2019. Due by 11:59pm on 30th Nov 2019

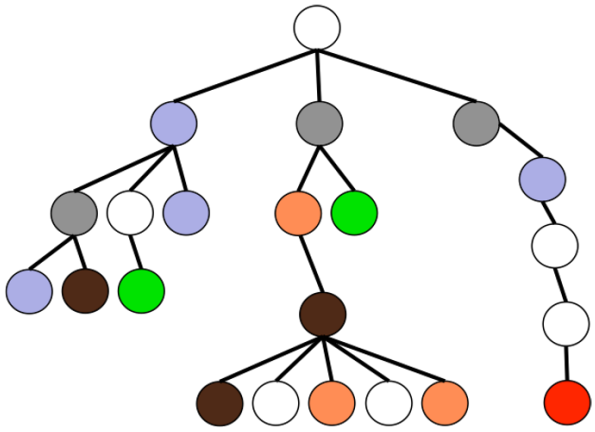
Multithreading and Design Patterns

This is a take-home lab assignment. No extensions whatsoever will be provided. Any submission after the deadline will not be evaluated. If you see any ambiguity or inconsistency in a question, please seek clarification from the teaching staff. Please read the entire text below very carefully before starting your implementation.

Plagiarism: All submitted lab assignments are expected to be the result of your individual effort. You should never misrepresent someone else's work as your own. In case any plagiarism case is detected, it will be dealt as per IIITD plagiarism policy and without any relaxations: https://www.iiitd.ac.in/sites/default/files/docs/education/AcademicDishonesty.pdf

Please note that you are not allowed to discuss the design/solution of the lab assignment (e.g. classroom page discussions etc.). Anyone who is found doing this will be treated as a plagiarism case. No excuses!

In this assignment, you will be focussing on the concepts of explicit multithreading, ForkJoinPool, and design patterns in Java. You are going to implement a **parallel recursive tree traversal program** by using both explicit multithreading and ForkJoinPool. This assignment will help you understand the productivity and performance of different styles for parallel programming. You are allowed to use Java Collection Framework in this assignment. (Edit: 27/11, 09:00 p.m) This assignment also has a bonus component as described below.



The program description is as follows.

Step-1:

- The main program would take input from the user for how many nodes are to be created in the tree. Let this value be **"N"**.
- Write a method that sequentially (and recursively) constructs an unsorted and unbalanced tree **"N"** number of nodes (see a sample tree in picture above), each with a random degree within the values 2, 3, 4, and 5. Each node will store a unique random integer within the range 1 to 1000000.
- Print the total time taken to construct this tree and the height of this tree.

Step-2:

- The main program would then ask the user to: a) input the set of nodes (i.e. numbers within the specified range mentioned above) that the user would like to check if they exist in the constructed tree, b) specify the technique the user would like to use for the above query processing, i.e., explicit multithreading or ForkJoinPool, and c) total number of threads that should be used in parallel processing.
- Based on the above inputs from the user, the program should be able to: a) traverse the tree recursively in parallel using the technique specified by the user and by using the specified number of threads (range: 1-4), b) specify if all those nodes exist in the tree and at what depths, c) print the total time taken for parallel processing, d) calculate and print the speedup over the sequential execution along with the parallel efficiency. You must traverse the tree **recursively** in both implementations, instead of simply storing the nodes in an array and then searching it over there.

Note that for implementing the ForkJoinPool implementation, you are not allowed to use the "**join()**" method. Instead of that you should use the "**helpQuiesce()".** Method. You can look online on Oracle documentation regarding the usage for helpQuiesce(). You should **NOT** use the producer-consumer model/design-pattern in your implementation. There is no strict requirement on the number of design patterns that you should implement. Although, you must have at least one design pattern in your implementation (**the one that is most relevant and can affect the execution**).

There will be some marks for extremely efficient parallel implementation.

There is no need for a sample test case as the user interaction is very limited, and is clearly mentioned in the above description.