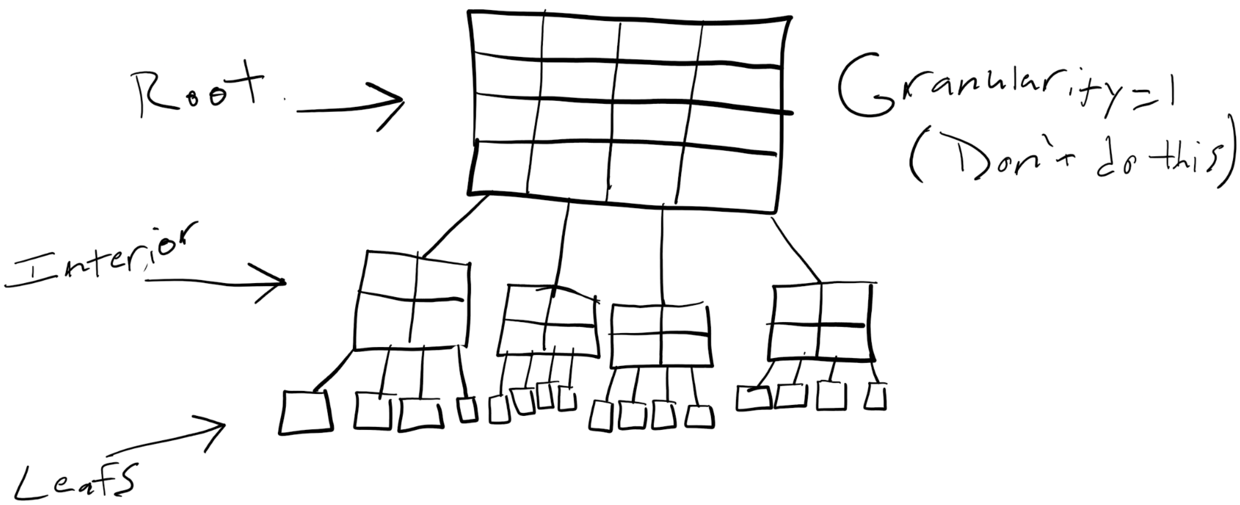
Project 3 Write-Up

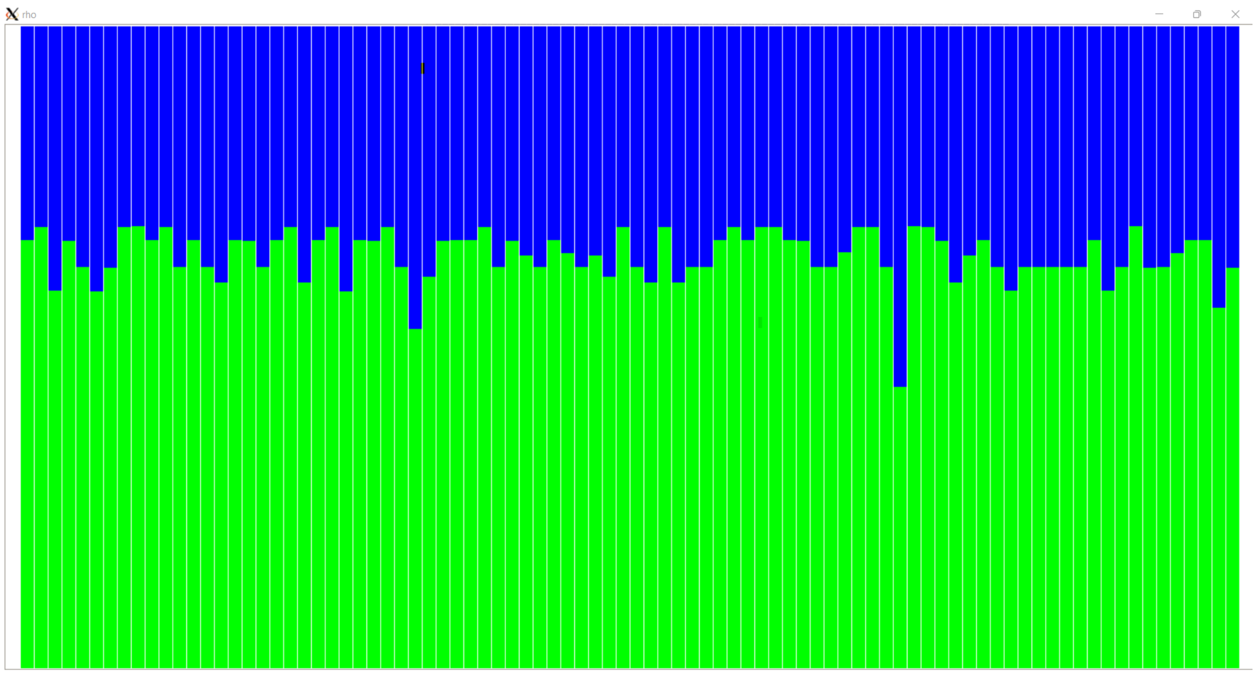
I first create the mesh by creating a matrix depending on user input. Each point on the matrix holds a region object that contains references of that coordinate's current temperature and it's metal proportions of the three metals making up the alloy for easy retrieval during computation of the region's new temperature. I store each of the heat constants in an interface for easy retrieval. Following the bootstrap provided closely, I break up the mesh into quarters each time until the subsequent quarter reaches a certain size, and they act as leaves in what becomes a tree structure composing the mesh. I then perform the summation calculation on the leaves and save a reference to the mesh after the calculations have been performed. This essentially creates a tree structure where each new division of the mesh becomes a node. Below is a diagram of how I break up the mesh.



I also keep a reference to whatever the maximum differences of each of the leaves are after the computations are performed. The largest value of these percolates up to the root node -the original mesh- and becomes the maximum difference of the entire mesh. The same summation will be performed on the leaves until the maximum difference of the mesh becomes less than or equal to .001. To compute the new values, each node on the tree extends the Recursive Action class, so I run a fork on each node that will run another fork on each of their children until they reach the leaves. I then join all of these forks until I get back to the root node and check and see if the maximum difference of the root node is less than or equal to .001. I use Java Swing to display the mesh. To do this, I related each temperature with an inverse HCB value to make a more dynamic image where the highest represented temperature is 255 degrees. Otherwise, every value above this temperature just gets represented with the same HCB color as 255 degrees. So, a temperature of 0 would have an HCB value of 255 and a temperature of 255 has an HCB value of 0.

Project 3 Write-Up

Following the bootstrap and making sure to understand the recursive action class made writing the concurrent structure of this project not so difficult, as all I had to think about was tree traversal and how forking onto each node of the tree would allow me to traverse and get to the leaves concurrently so I can run calculations. I likely would’ve had this project done a week ago if I had realized that my method for generating random metal percentages in each of the regions was flawed. It also runs very well concurrently, as every core on Rho gets used when running the program.



Granularity is important here as well. If the granularity is too fine, it could lead to less performance due to the overhead required to maintain the threads produced by forking. I had optimized my granularity enough to a point where the cores on Rho were mostly used up, but each core could certainly be more utilized if the granularity is further customized so that there isn’t excessive thread overhead. This is something I kept in mind for Project 4 so that each node in the cluster is at full efficiency and their granularities are customized according to how many cores they have.