COMP3506 Homework 1

Daniel Nathan

August 4, 2019

For simplicity, the width and height of the ArrayGrid will be refered to as w and h respectviely. The ArrayGrid implementation has memory complexity $O(hw)$. The class stores two integers (width and height), and a two dimensional array with these dimensions. Letting x be the memory complexity of an integer, and y be the memory complexity of the data type being stored, the total memory required for an instance of ArrayGrid is therefore $2x + hwy$. Choosing $c = 3y, n_0 = \frac{x}{y} \leq hw$, the definition of big-O is satisfied for $f(h, w)$:

$$\begin{aligned} \frac{x}{y} &\leq hw \\ \implies 2x &\leq 2hwy \\ \implies 2x + hwy &\leq 3hwy \end{aligned}$$

This is not a memory efficient implementation, as the memory used upscales with the size of the grid, rather than the amount of data stored. This means a large ArrayGrid with most cells empty can have much greater memory requirement than the sum of its elements' memory requirements.

An alternative way of storing data in a two dimensional grid is using a point quadtree. The first data point added divides the grid into quadrants with vertical and horizontal lines passing through it, then each subsequent point partitions the region it is in into four similarly. Each node is connected to the node that defines its region using a tree structure. Nodes are only created for positions where there is data, meaning its memory complexity is $O(n)$, where n is the number of data points being stored. This is much more efficient than $O(hw)$ for cases where the grid is large but sparsely populated. Due to the tree structure, the clear and resize methods will also have $O(n)$ time complexity instead of $O(hw)$, as they would no longer need to iterate over every position in the grid. However, due to the tree structure needed to traverse the quadtree, the access time for a particular data point in the add, get and remove methods is $O(\log n)$ (or $O(n)$ for the worst case scenario where the tree has only one branch), rather than $O(1)$ for ArrayGrid. The remove method may also have to restructure the tree if the deleted node has children, making the algorithm take longer.