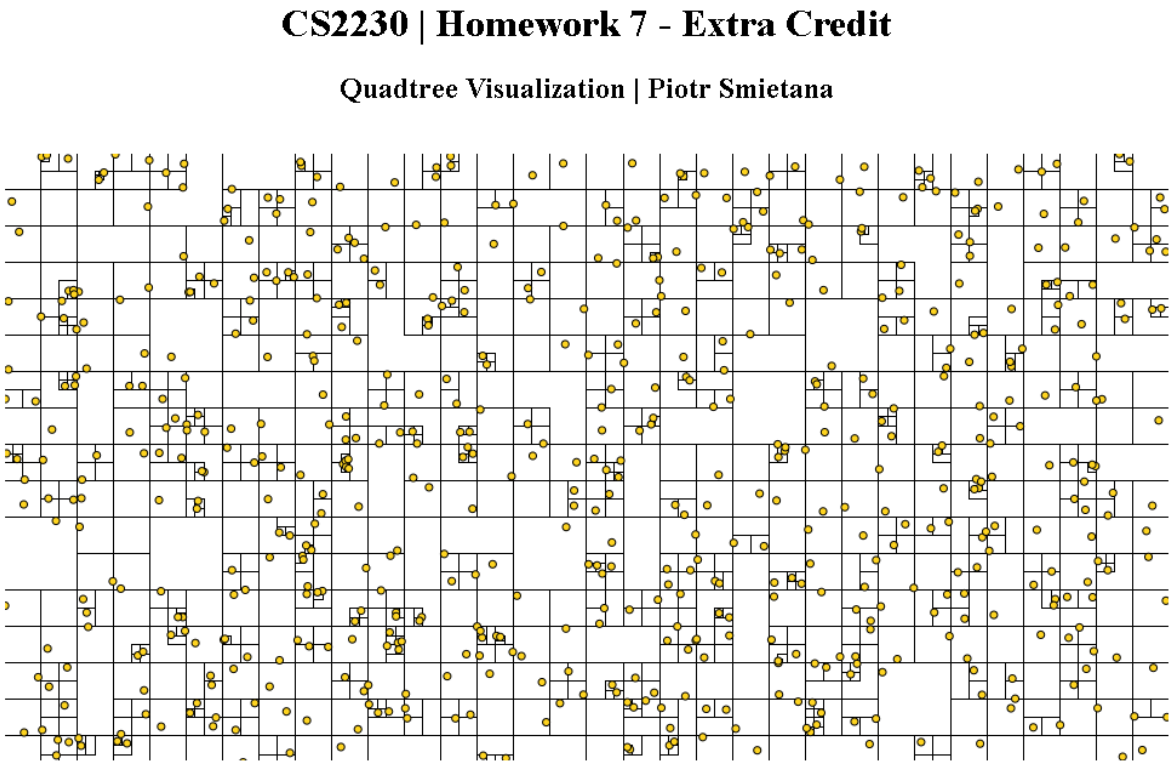
CS2230 Computer Science II: Data Structures

Homework 7 – Extra Credit

Quadtree Visualization | Piotr Smietana

As a part of extra credit for homework 7 I decided to use a drawing library to visualize a SpatialTreeMap, showing all the bisections and the data points.

I put all my code into a folder named EXTRA, so to see my work, please open index.html in a web browser. Here is a small preview of what should appear on the screen. EXTRA\index.html



I created a simple html website which consists of a d3[1] java script and some html/css code to define a QuadTree visualization using black and gold theme. To see my implementation, please open index.html in a notepad. All sections of the code should be commented.

[1] https://github.com/d3/d3/wiki

https://github.com/d3/d3 is an open JavaScript library for visualizing data using web standards. It helps to bring data to life using SVG, Canvas and HTML. The script uses APIs to communicate with different components. The script recursively breaks down canvas into smaller squares dividing each square into four equally-sized squares. Corresponding points are represented by a linked list.

To install the script, I used the following code

```
<script src="https://d3js.org/d3-quadtree.v1.min.js"></script>
<script>

var quadtree = d3.quadtree();

</script>
```

To communicate with d3 JavaScript I used application program interface (API). For example, to create an empty quadtree I called

```
var tree = d3.quadtree()
    .addAll(data);
```

Defining <head>, short description, <style> is quite simple. But here is my explanation of the algorithm I used in the <body> section:

1. Declaring variables - width/height/radius which set the size of the canvas and the size of data points
2. Randomly generating data from range using Math.random() and width/height
3. Constructor
4. Deciding in which format the visualization will be displayed (.svg)
5. Recursively defining squares and all intersection of data points based on nodes
6. Defining and adding data points to canvas as “circles” with previously defined radius
7. Finally defining API function nodes() which visits each node and adds it to our LinkedList