MAKING THE GRAPH

Floorplan information

Door information

Classes
Point class
Graph class
R-tree class

Making an R-tree datastructure for the walls. This is used to quickly find the nearest wall for any given node.

plotting the floorplan on a grid where each cell is 0.5x0.5.

Making a dictionary of rooms. Where each room is a key and the values are the lines making up the room.

Using the coordinates of the doors to make a pair of Point objects that are placed on each side of their respective walls.

Generating a Point object at the center of each room.

Using the networkx graph class to make a graph. The door points will be attributed as door nodes. The points coming from the dictionary of rooms will be attributed as room nodes. Besides this a node will be constructed at every cell of the floorplan with the attribution of grid nodes.

This means we have "room", "door" and "grid" nodes. All the nodes have a point associated with it. The point for each node is discretised to the nearest 0.5 x and y coordinate. Each node also has a distance to the nearest wall as an attribute. Lastly each node has an index associated with it.

Example:
Graph.add_node(index = 257, attributes = ("room", Point(2.5,3.0), dist = 0.7))

Furthermore each node is connected to the nodes up,down,left,right and diagonal to it.
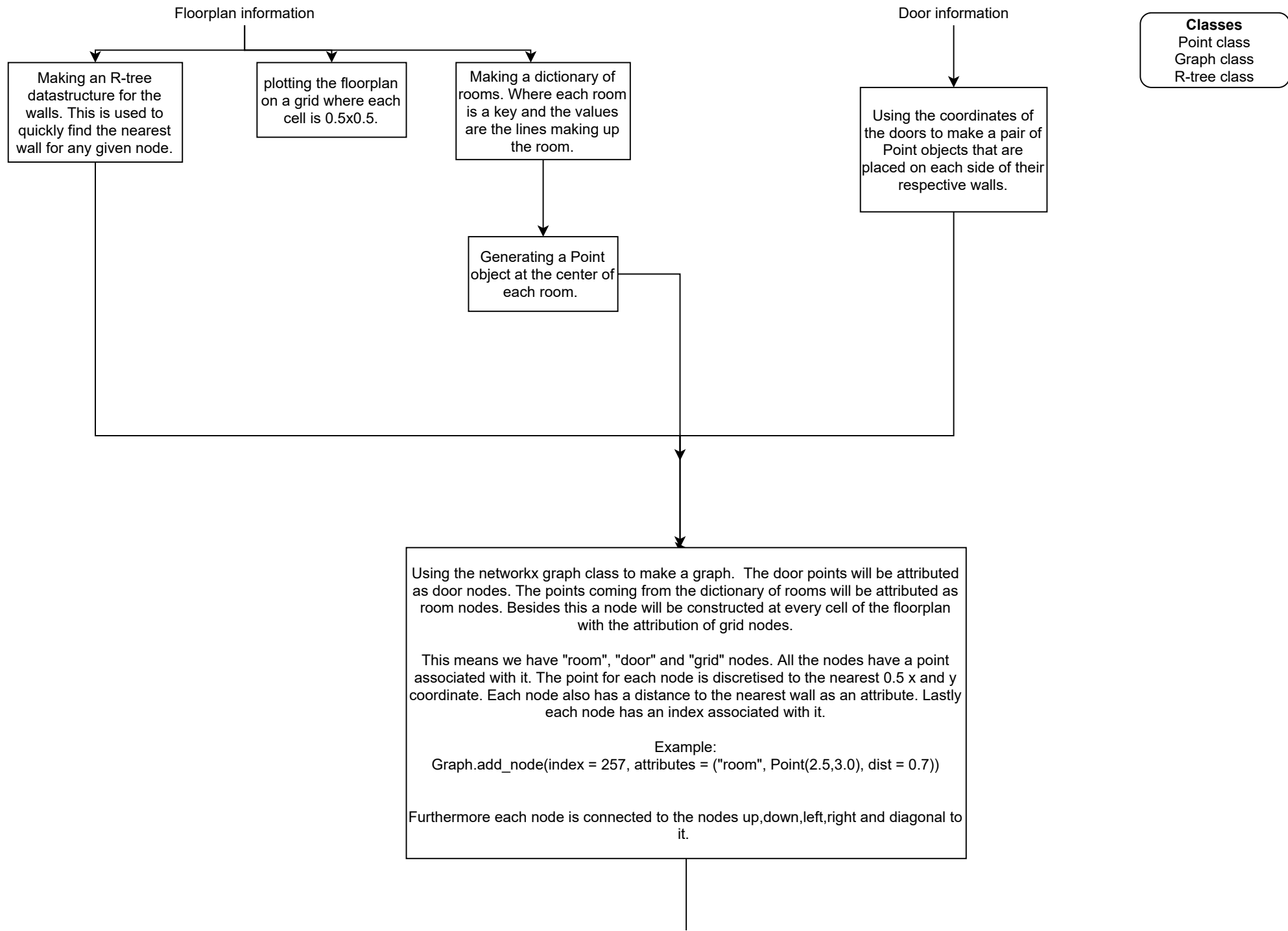
Removal of nodes that are within 0.5 distance of a wall.

This is to account for the dimensions of the robot.

The exceptions here are the door nodes, which are allowed to be arbitrarily close to the walls, since they will act as portal between rooms.

PATHFINDING

Using the A * algorithm to find the path between each room node. The output here is:

A dictionary of dictionaries where each room node is a key for the first dictionary and for each key we have another key that is the other room node and the values for the second key are the nodes that make up the path between the two nodes.

Example:
Dict = {room1:
{room2: (inbetween nodes of room1 and room2) = 7,5,4,9}
{room3: (inbetween nodes of room1 and room3)=15,9,24,54}
}

We now have the shortest path between each room node. We can now make a Minimum spanning tree that includes only the room nodes. This is useful when

making an approximate solution to the
traveling salesman problem.

Solving the Traveling salesman problem using Depth first
search approximation.

(see this video for more information on the precise
method https://www.youtube.com/watch?
v=M5UgglrAOME&t=243s&ab_channel=JoeJames
Watch from 2:50)

Plotting the path on the plot
made at the start. Making sure
to only plot the room nodes and
the door nodes. NOT the grid
nodes.