**Task 1:**

The data structure that we will be using for this project will be a graph. Our graph will be undirected, have no edge weights, and the internal structure of the graph will be an adjacency list. The graph will be sparsely connected, with each vertex having a maximum of eight edges associated with it.

Conceptually, we interpret this graph as being a two-dimensional, square grid. An edge stemming from one vertex will only be able to connect to one of its eight adjacent vertices. In comparison to a complete graph, out graph will have a much smaller space complexity.

The space complexity of any graph is defined as *O(V + E)*, where *V* is the number of vertices, and *E* is the number of edges. We will use *N* to represent the number of vertices in a graph, and *n* to represent the width of the grid our graph represents In a complete graph, the number of edges it has is given by . In our sparser graph, the number of edges is given as . Thus, the space complexity of a complete graph is:

which is on the order of *O(N2)*. For our sparser graph, the space complexity is:

which is on the order of *O(N)*. Thus, our graph will be much more space efficient for larger values of N. This is necessary, as our plan is to use , since our graph represents a 1000x1000 grid.

Initialization of our graph involves two *O(N)* operations. The first *O(N)* operation is necessary to initialize all vertices of the graph, and the second is to initialize all necessary edges. However, these operations will only occur once, as they are used to construct the map class that uses the graph. The other main operations are updating the map (graph) with a new set of probabilities, and the ray casting operation. Updating the map will involve iterating through each vertex, which is an *O(N)* operation. The ray casting method iterates through each vertex as well, but also needs to call the map update function in each iteration, meaning that ray casting is an *O(N2)* operation.

**Task 2:**

Yes.