# Solution

I have written the solution using a very simple Map Class that creates a 64 x 64 square with no boundaries and a start and fixed location (being top left and bottom right)

The Application is made from a few classes.

* RTSGameMap – Stores the Map that is generated, including return types for whether the square is Occupied of Free.
* The Coords struct creates a simple definition for tracking x/y coordinates.
* The CreateMap – creates a map with a size, random squares that are occupied.
* The DijkstraSource - this uses a rough implementation of the Dijkstra algorithm by looking at adjacent squares and adding the steps to get to the square.
  + Step 1 of the code goes through and checks for adjacent tiles and adds in the step to get to the tile. Each tile that has been visited is then recorded.   
    Any Tile that is occupied is not returned in the array (i.e. you can only visit a tile once).  
    If the adjacent tile has a value that is in excess of the current value, then tile is set to the value of the current tile + 1 (This is simple as the steps are always 1)
  + Step 2 when the application reaches its goal, it goes backward and iterates over each tile’s adjacent value. As the jump from one tile to another is always 1, it just checks to see the jump is valid. (This could be modified if the jumps were not uniform in size).
  + If the jump is valid, the coordinates are added to the stack for reference.
* The final part of the application reads through the stack and plots the location of path on the map.
* As the application is finding the quickest path and then plotted a route, concurrency would be an issue. If the path was to take time, the location of item would need to be record, setting the location to be occupied. I have set the map array to be a byte to allow values other than 1/0, however I have only returned occupied or free in this example.