Execution Document

Data Structures used:

- **ArrayList**
- **PriorityQueue**

The A* star application I've done comes with 4 different Heuristic methods. The heuristics are partially from the link presented in the definition document.

I do believe that some of my heuristic methods aren't working as they should. They don't give a similar path as I would expect. They do still find the shortest path. I haven't had the time check out were the problem is (if there's any), but they ~almost~ work… ;)

Sadly I haven't separated all the logic from the graphic (the OpenGL stuff make the drawing a little bit ugly).

ArrayList

The ArrayList's implementation builds on an array that doubles in size when the maximum size is reached. (It's very poorly implemented for now, just the methods needed in this project are used.)

Because of the array structure, you can add at the last index by just accessing **Array.length - 1**.

	Add at end	Delete	Get	Find		
Wanted Time:	O(1)	O(n)	O(1)	O(n)		
Time:	**O(1)**	**O(n)**	**O(1)**	**O(n)**		
Wanted Space:	O(n)					
Space:	**O(n)**					

PriorityQueue

The priority queue is much similar to the ArrayList, with a array that doubles in size when the limit is reached.

The structure used is a binary heap (every node has at most two children) and the heap can be defined to be a max or min heap based on the Comparator provided during construction.

The left child of **P** can be found at index: **2 * Index(P) + 1**

The right child of **P** can be found at index: **2 * Index(P) + 2**

This way you can access all the elements of the heap from its own index in the array representation of the heap.

	Add	Delete	Get	Find	Poll	Peek
Wanted Time:	O(log(n))	O(log(n))	O(1)	N/A	O(log(n))	O(1)
Time:	**O(log(n))**	**O(log(n))**	**O(1)**	**O(n)**	**O(log(n))**	**O(1)**
Wanted Space:	O(n)					
Space:	**O(n)**					