1. PQ, dict, list, tuple, array

PQ – time complexity O(logn), space complexity O(n) where n is the number of nodes

2D arrays – time complexity O(n\*m), space complexity O(n2)

first attempt – 2D arrays to store all nodes like PQ but takes up too much time and space so second attempt uses PriorityQueue (reference reblobgames)

**dict**

came\_from – store all nodes (values) to the node (key) that came before it {(0,0): (0,1)}

cost\_so\_far – {node: sum of cost since the last node}

**PQ**

frontier – initialise by adding start\_node given from input -> adds in next best node with the lowest code calculated using A\* search

**tuple**

t\_next\_node, node\_start, node\_goal – since data stores doesn’t need to be changed and needed to check if these are in dict

**2D array**

Blocks – read in from json and store cell coordinates of blue nodes

set value of came\_from to NULL, cost\_so\_far to 0 so that when finding shortest path these nodes will be marked as “visited”

**class**

used on content on PQ

1. Heuristics – straight line distance

Since hex cells are of same size, and cost of moving one cell to another are all 1, so can assume calculating the straight-line distance will give the shortest path (overall cost will just be 1 + straight-line distance)

cost of computing this heuristics - ?