1a) What is the minimum cost of reaching location (m, n) (note that m, n ≥ 0) starting from location (0, 0) in the above city? Describe one possible path achieving the minimum cost. Is it unique (i.e., are there multiple paths that achieve the minimum cost)?

Given the cost function of the problem...

$$cost\big((x,y),a\big) = 1 + \max\,(x,0)$$

The minimum cost function is given by...

$$minimum\ cost = 1 + \max(m,0) + \max\,(n,0)$$

The path to reach the minimum point is not unique, and there are many that can achieve it. One of the possible paths is to move in the direction (+1,0) until reaching coordinate m, and then move in the direction (0,+1) until reaching coordinate n.

1b) How will Uniform Cost Search (UCS) behave on this problem? Mark the following as true or false:

i. UCS will never terminate because the number of states is infinite. F

ii. UCS will return the minimum cost path and explore only locations between (0, 0) and (m, n); that is, (x, y) such that $0 \leq x \leq m \; and \; 0 \leq y \leq n.$ T

iii. UCS will return the minimum cost path and explore only locations whose past costs are less than the minimum cost from (0, 0) to (m, n). T

1c) Now consider running UCS on an arbitrary graph. Mark the following as true or false:

i. If you add a connection between two locations, the minimum distance cannot go up. F

ii. If you make the cost of an action from some state small enough (possibly negative), that action will show up in the minimum cost path. V

iii. If you increase the cost of each action by 1, the minimum cost path does not change (even though its cost does). F

3b) If there are n locations and k waypoint tags, what is the maximum number of states that UCS could visit? What we expect: A mathematical expression that depends on n and k, with a brief explanation justifying it

The mathematical expression is $n2^k$, where this arises from the fact that for each of the n locations, there are two possible options (visited or not visited), and for each of the k waypoint labels, there are two possible options (included or not included).

4d) Recall that heuristics are most effective when they are equal to the future cost. Consider a n × n grid map (createGridMap) where the start and end are at the opposite corners; for n = 10, we would have startLocation = "0,0", endTag = "label=9,9".

Provide a concrete example of n waypointTags so that running A* with the NoWaypointsHeuristic results in thevsame running time complexity as solving the relaxed shortest path problem.

For the NoWaypointsHeuristic to have the same complexity as the relaxed shortest path problem, we must place a waypoint tag at each intermediate coordinate between the start and the end, excluding the corners. Since it will be forced to consider the cost of reaching each tag, essentially replicating the relaxed problem where only the number of moves to reach the goal matters, the points will be:

[1, 1]	[2, 1]	[3, 1]	[4, 1]	[5, 1]	[6, 1]	[7, 1]	[8, 1]	[9, 1]
[1, 2]	[2, 2]	[3, 2]	[4, 2]	[5, 2]	[6, 2]	[7, 2]	[8, 2]	[9, 2]
[1, 3]	[2, 3]	[3, 3]	[4, 3]	[5, 3]	[6, 3]	[7, 3]	[8, 3]	[9, 3]
[1, 4]	[2, 4]	[3, 4]	[4, 4]	[5, 4]	[6, 4]	[7, 4]	[8, 4]	[9, 4]
[1, 5]	[2, 5]	[3, 5]	[4, 5]	[5, 5]	[6, 5]	[7, 5]	[8, 5]	[9, 5]
[1, 6]	[2, 6]	[3, 6]	[4, 6]	[5, 6]	[6, 6]	[7, 6]	[8, 6]	[9, 6]
[1, 7]	[2, 7]	[3, 7]	[4, 7]	[5, 7]	[6, 7]	[7, 7]	[8, 7]	[9, 7]
[1, 8]	[2, 8]	[3, 8]	[4, 8]	[5, 8]	[6, 8]	[7, 8]	[8, 8]	[9, 8]