Partner: Nobody (for now)

No matter how many classes push it on me im not gonna learn LaTeXA math equations and formulas

Description automatically generated with medium confidence

I will show that if p\* = p1 p2, then p1 is a shortest path from a 🡪 c and p2 is a shortest path from c 🡪 b via contradiction.

First, let us consider another path q1, which travels from a 🡪 c with a cost of yi < y. Then the cost of this new path, py, is yi + z < y + z, but this contradicts the assumption that p\* is the shortest path from a 🡪 b.

Next, let us similarly consider an alternative path q2, which travels from c 🡪 b at a cost of zi < z. Then the cost of this new path, pz, is y + zi < y + z, which again contradicts the assumption that p\* is the shortest possible path from a 🡪 b.

In conclusion, if it is known that path p\* is the shortest path from a 🡪 b with intermediate paths a 🡪 c and c 🡪 b, then we know that there is no alternate intermediate path that can perform better than these shortest paths.

A black text on a white background

Description automatically generated

In the worst case using DFS, vertex b would be a leaf node furthest away from vertex a as possible and stemming from the last neighbor of a, meaning the algorithm would have to explore every edge, |E|, and vertex, |V|, possible before finding vertex b. In the best case, vertex b is the only neighbor of vertex a, meaning that the algorithm would only take a single computational step before finding and reaching b, looking at a total of 2 vertices and 1 edge (a, b, a 🡪 b).

A black and blue maze with white text

Description automatically generated

A text on a white background

Description automatically generated

#initializing caches and relevant structures

The idea behind my heuristic is to give all nodes without a pellet a weight of 1, and nodes with pellets a weight of -1. This way, the AI will be able to easily determine the shortest possible path to and from other pellets when it clears an area.

Current\_loc = current\_loc #just to have it for reference down here

Food\_remaining\_locs = food\_remaining\_locs

empty\_nodes = {!food\_remaining\_locs} #we don’t want to waste valuable resources revisiting a node when we don’t have to.

Heuristic\_fn(node): #this calculates the cost of moving to any specific node

Cost = 0

For node in path:

If node in empty\_nodes:

Cost ++

Else: #either the node is empty or has a pellet

Cost = cost

Return cost

Cost\_fn(node): #this returns the actual cost of moving to a specific node reflected by the question

Cost = 0

For node in path:

Cost ++

Return cost

\*With the way the heuristic and cost functions are set up, the smallest possible value determined by a total cost function would be 1, given the case that there’s a pellet right next to pacman. This also ensures that it is non-negative. In my heuristic function, I gave nodes with pellets a weight of 0 because I want the heuristic to view a 5-tile straight line filled with pellets as just as good as the option of moving a single square to a pellet. This way, the algorithm will be able to better select moving to a tile in the middle of a pellet-rich area as opposed to instead choosing to go for a single pellet in a dead zone in the opposite direction.\*

Find\_path(loc):

Cost\_matrix = {}

For node in map:

Cost\_matrix{node} = heuristic\_fn(node) + cost\_fn(node)

Return cost\_matrix

While food\_remaining\_locs != empty:

Taken\_path = min(find\_path(current\_loc)) #this is supposed to calculate the cost of every node and taken\_path will be the best possible pellet to travel to.

While executing taken\_path:

If node in food\_remaining\_locs:

Food\_remaining\_locs.pop(node)

Empty\_nodes.add(node)

#i want to update the tables as pac moves once it has made a decision

Return 0 #once there are no more pellets on the map, the algorithm will return 0 and exit.