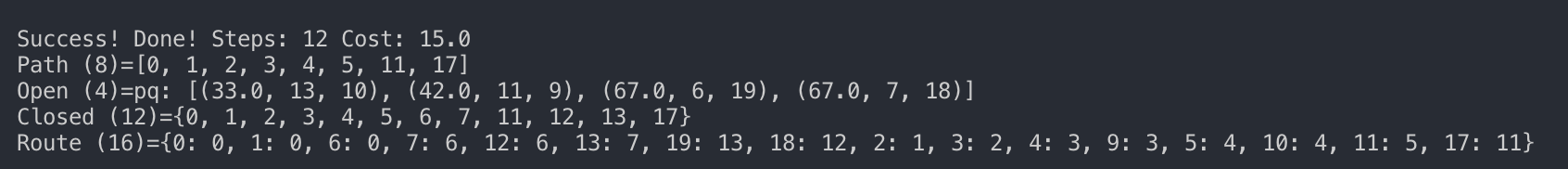
Before change: min\_edge\_cost = 10.0

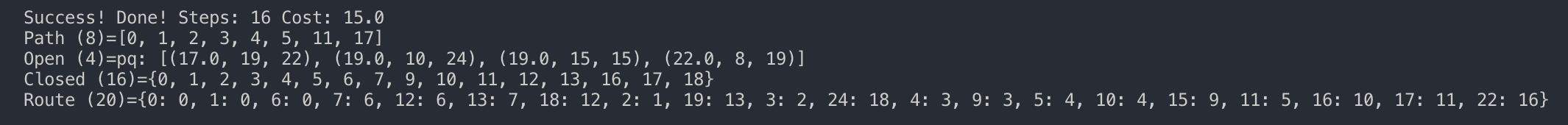
A screenshot of a cell phone

Description automatically generated 

After change:

min\_edge\_cost = 1.0 *# must be min value for heuristic cost to work*

A screenshot of a cell phone

Description automatically generated 

By changing the value to the min value, more routes are discovered. This means that the path finding algorith is more optimal to find more available solutions.

def reset\_navgraph(self):

self.path = None *# invalid so remove if present*

self.graph = SparseGraph()

*# Set a heuristic cost function for the search to use*

self.graph.cost\_h = self.\_manhattan

self.graph.cost\_h = self.\_hypot

self.graph.cost\_h = self.\_max

nx, ny = self.nx, self.ny

*# add all the nodes required*

for i, box in enumerate(self.boxes):

box.pos = (i % nx, i // nx) *#tuple position*

box.node = self.graph.add\_node(Node(idx=i))

*# build all the edges required for this world*

for i, box in enumerate(self.boxes):

*# four sided N-S-E-W connections*

if box.kind in no\_edge:

continue

*# UP (i + nx)*

if (i+nx) < len(self.boxes):

self.\_add\_edge(i, i+nx)

*# DOWN (i - nx)*

if (i-nx) >= 0:

self.\_add\_edge(i, i-nx)

*# RIGHT (i + 1)*

if (i%nx + 1) < nx:

self.\_add\_edge(i, i+1)

*# LEFT (i - 1)*

if (i%nx - 1) >= 0:

self.\_add\_edge(i, i-1)

*# # Diagonal connections*

*# UP LEFT(i + nx - 1)*

j = i + nx

if (j-1) < len(self.boxes) and (j%nx - 1) >= 0:

self.\_add\_edge(i, j-1, 1.4142) *# sqrt(1+1)*

*# UP RIGHT (i + nx + 1)*

j = i + nx

if (j+1) < len(self.boxes) and (j%nx + 1) < nx:

self.\_add\_edge(i, j+1, 1.4142)

*# DOWN LEFT(i - nx - 1)*

j = i - nx

if (j-1) >= 0 and (j%nx - 1) >= 0:

print( i, j, j%nx)

self.\_add\_edge(i, j-1, 1.4142)

*# DOWN RIGHT (i - nx + 1)*

j = i - nx

if (j+1) >= 0 and (j%nx +1) < nx:

self.\_add\_edge(i, j+1, 1.4142)

Code needed to add diagonal edges is uncommented and fixed.

The use of Manhattan distance calculation is altered.