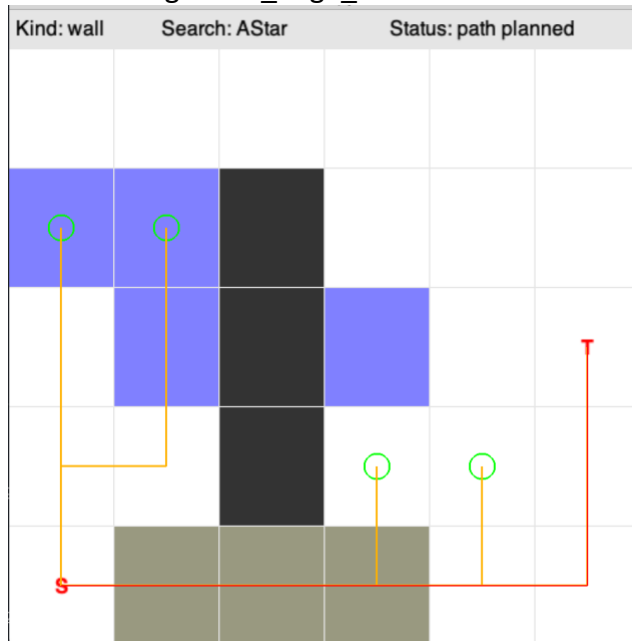


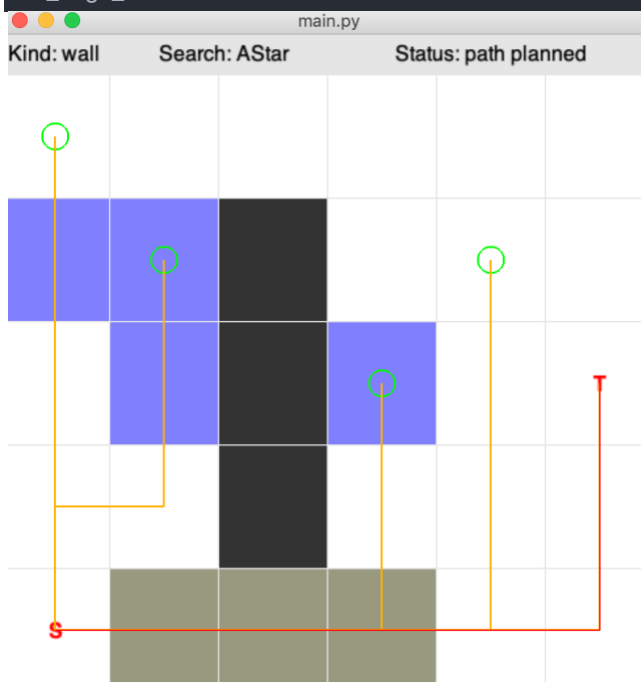
Before change: min\_edge\_cost = 10.0



```
Success! Done! Steps: 12 Cost: 15.0
Path (8)=[0, 1, 2, 3, 4, 5, 11, 17]
Open (4)=pq: [(33.0, 13, 10), (42.0, 11, 9), (67.0, 6, 19), (67.0, 7, 18)]
Closed (12)={0, 1, 2, 3, 4, 5, 6, 7, 11, 12, 13, 17}
Route (16)={0: 0, 1: 0, 6: 0, 7: 6, 12: 6, 13: 7, 19: 13, 18: 12, 2: 1, 3: 2, 4: 3, 9: 3, 5: 4, 10: 4, 11: 5, 17: 11}
```

After change:

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



```
Success! Done! Steps: 16 Cost: 15.0
Path (8)=[0, 1, 2, 3, 4, 5, 11, 17]
Open (4)=pq: [(17.0, 19, 22), (19.0, 10, 24), (19.0, 15, 15), (22.0, 8, 19)]
Closed (16)={0, 1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13, 16, 17, 18}
Route (20)={0: 0, 1: 0, 6: 0, 7: 6, 12: 6, 13: 7, 18: 12, 2: 1, 19: 13, 3: 2, 24: 18, 4: 3, 9: 3, 5: 4, 10: 4, 15: 9, 11: 5, 16: 10, 17: 11, 22: 16}
```

By changing the value to the min value, more routes are discovered. This means that the path finding algorithm 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.