COMP 4752
Computational Intelligence

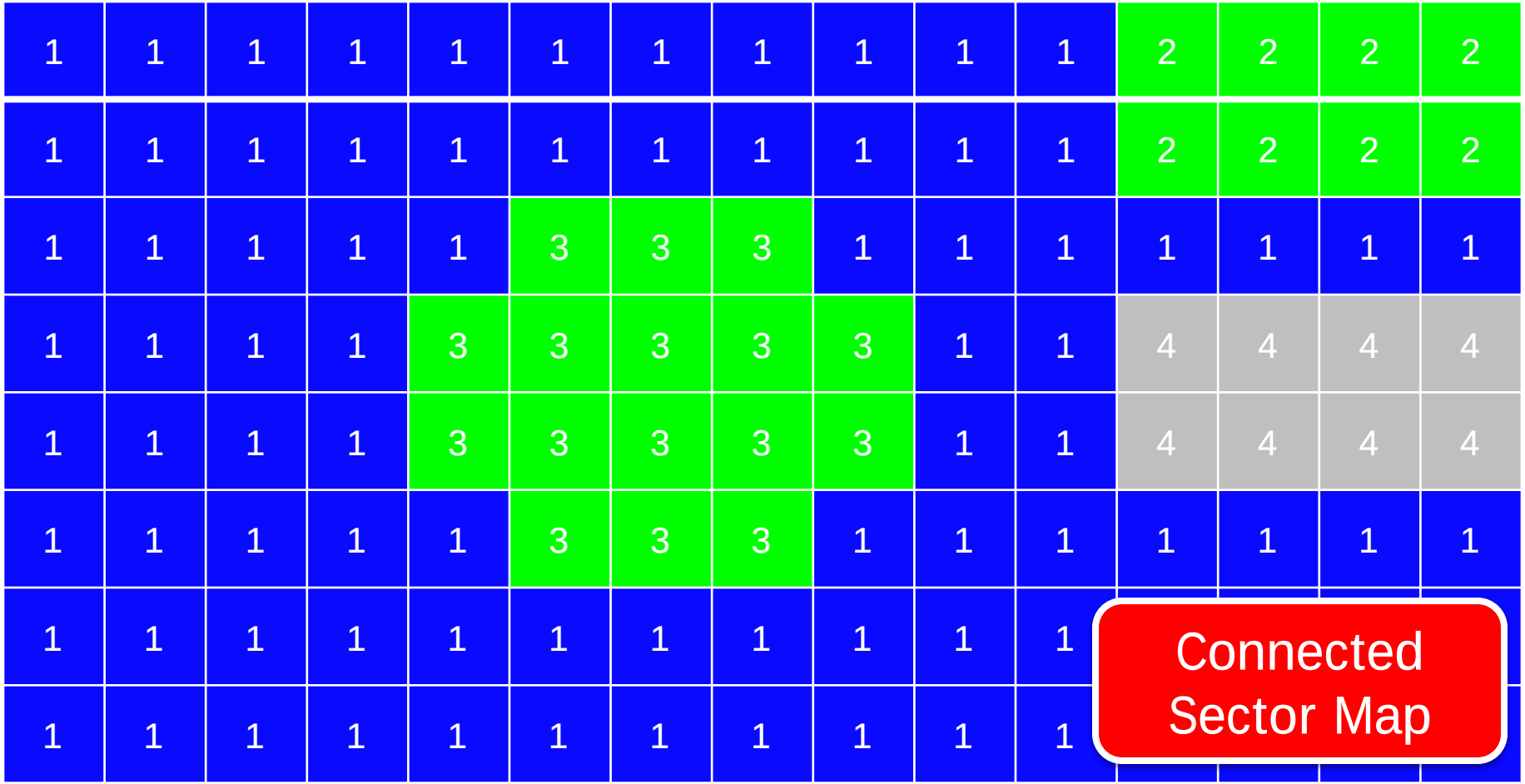
Lecture 8

2D Grid Connectedness

A* Search Example

Determining Connectedness

- is_connected(start, goal, size)
  - Returns whether a path exists from start to goal
  - Can be calculated via 4-directional BFS (flood fill)
- We could run this BFS every query, but that is slow, and lots of repeated computation
- Instead, we will pre-compute connectedness when we construct the Grid object
- Adds start-up time to make query time faster

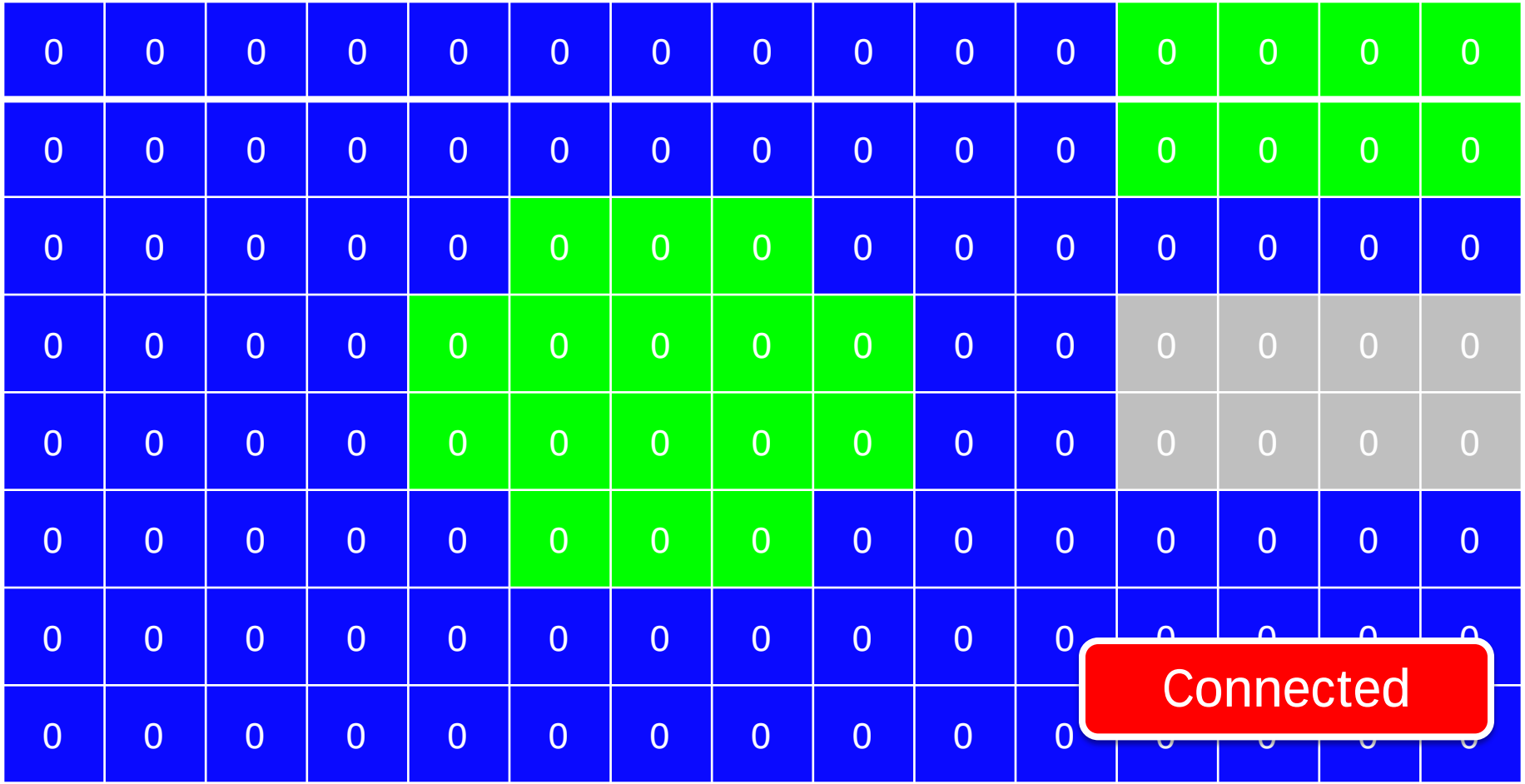

1 1 1 1 1 1 1 1 1 1 1 2 2 2 2
1 1 1 1 1 1 1 1 1 1 1 2 2 2 2
1 1 1 1 1 3 3 3 1 1 1 1 1 1 1
1 1 1 1 3 3 3 3 3 1 1 4 4 4 4
1 1 1 1 3 3 3 3 3 1 1 4 4 4 4
1 1 1 1 1 3 3 3 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1
Connected
Sector Map

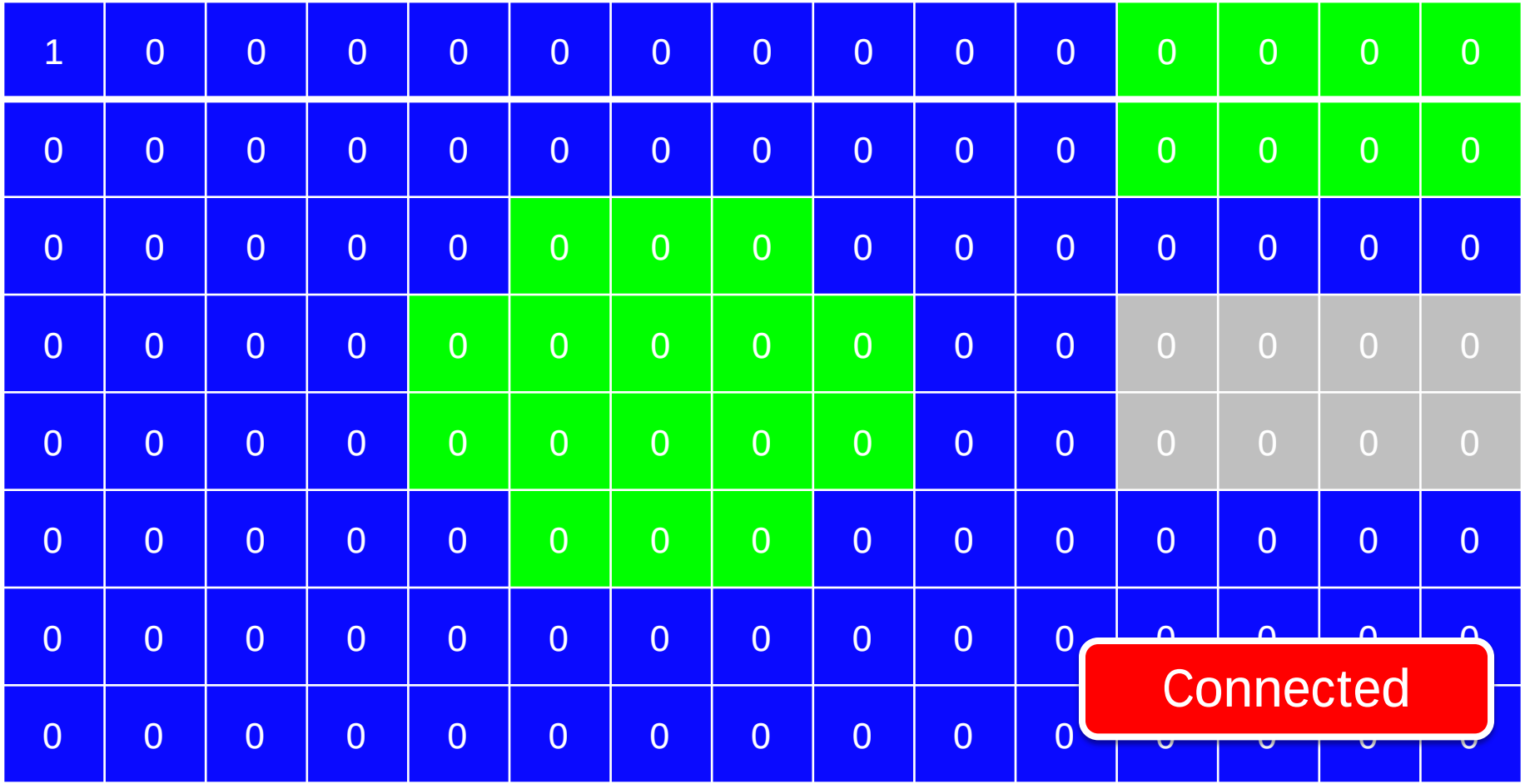
Connected Sectors

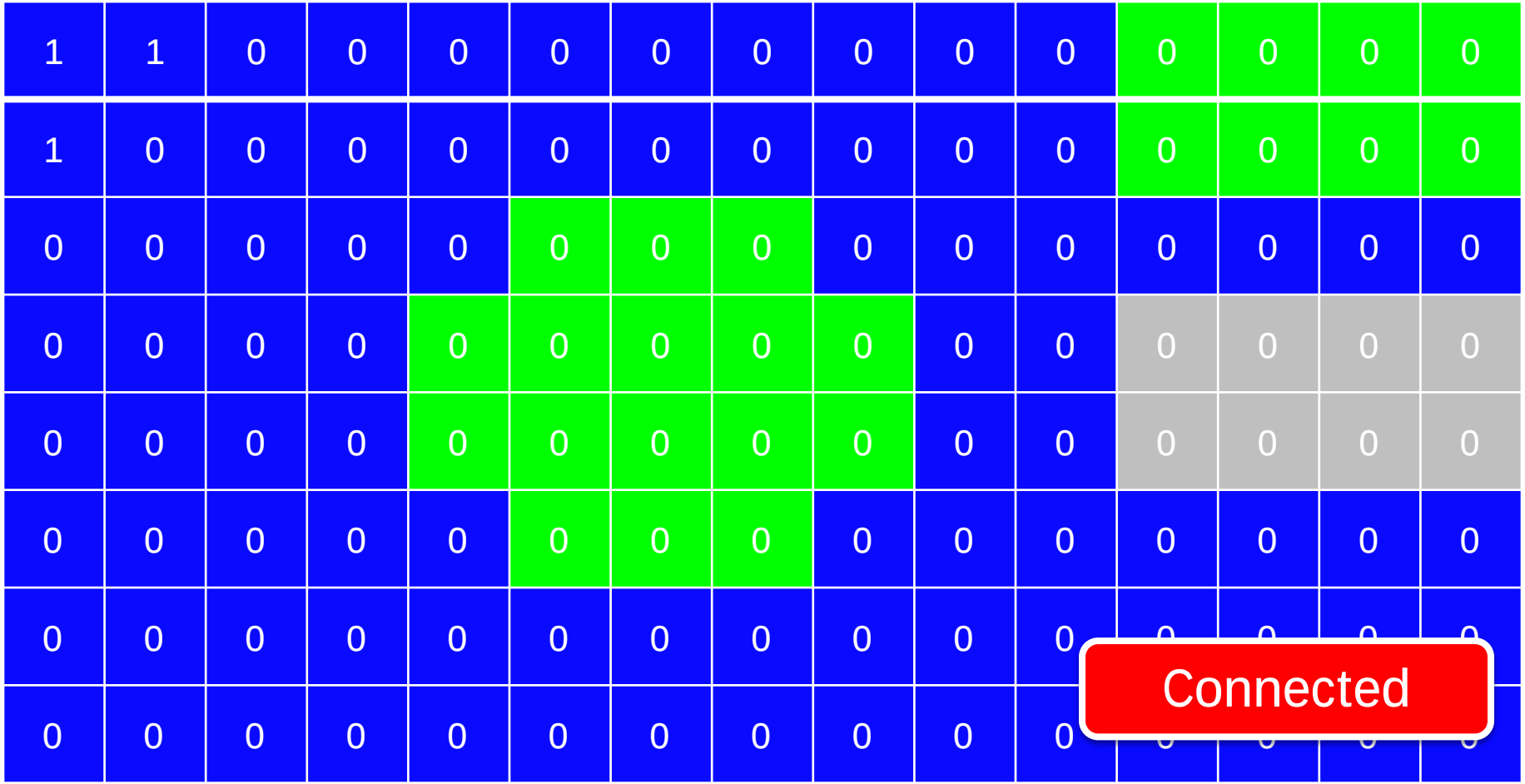
- sectors[size][x][y]
- Stores integers = connected sectors
- Same size as our 2D grid map
- Two tiles (x1,y1) (x2,y2) are connected if sectors[s][x1][y1] == sectors[s][x2][y2]
- Not walkable if sectors[s][x1][y1] == 0

Computing Connected Sectors

```
1.  Function ComputeSectors(tiles, max_size)
2.      sectors = zeros[size][x][y]
3.      sector_num = 1
4.      for s in range(1, max_size+1):
5.          for (x,y) in tiles:
6.              if (sectors[x][y] != 0) continue
7.              flood_fill(x, y, size, sector_num)
8.              sector_num += 1
```


0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
Connected


1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Connected

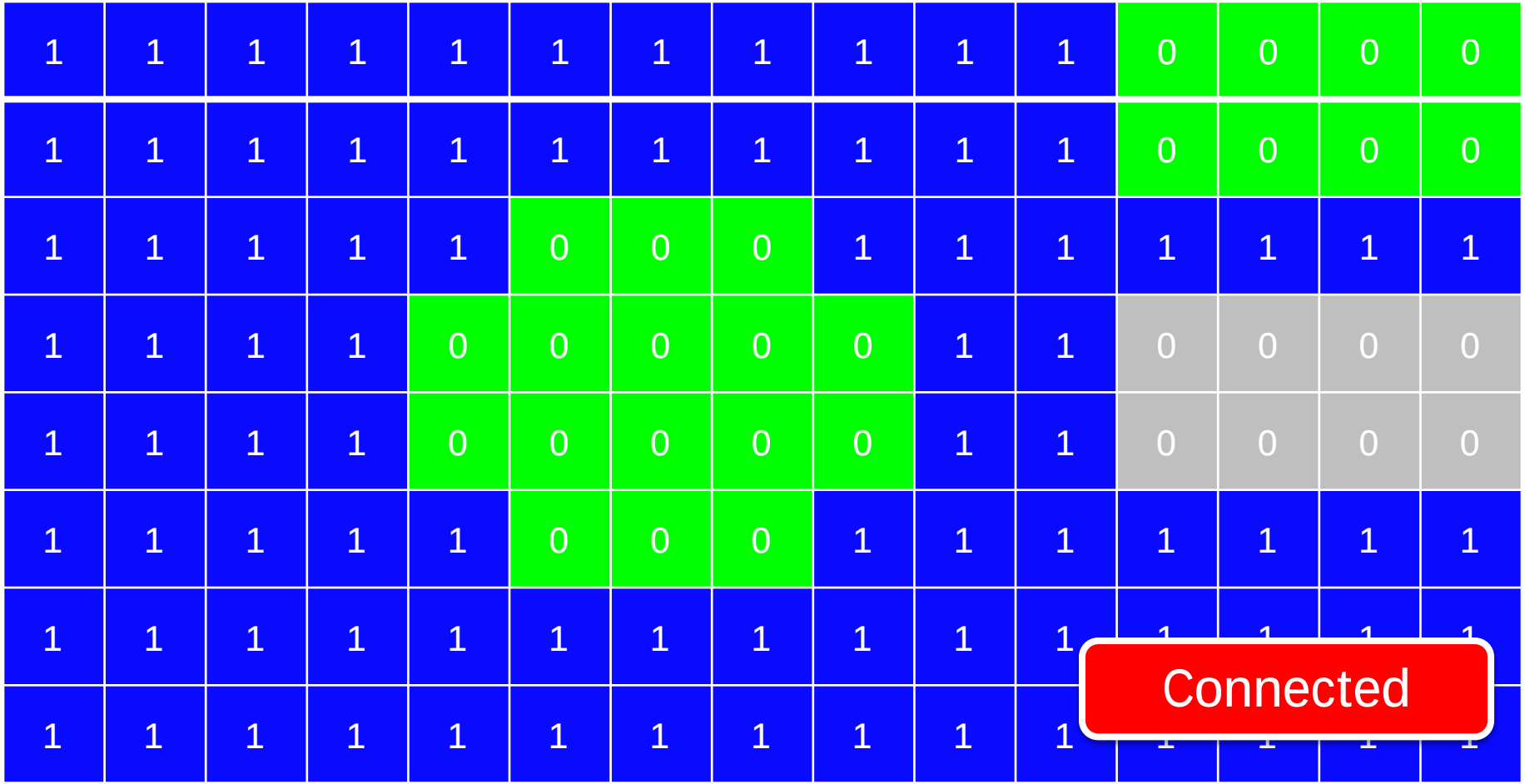

1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Connected

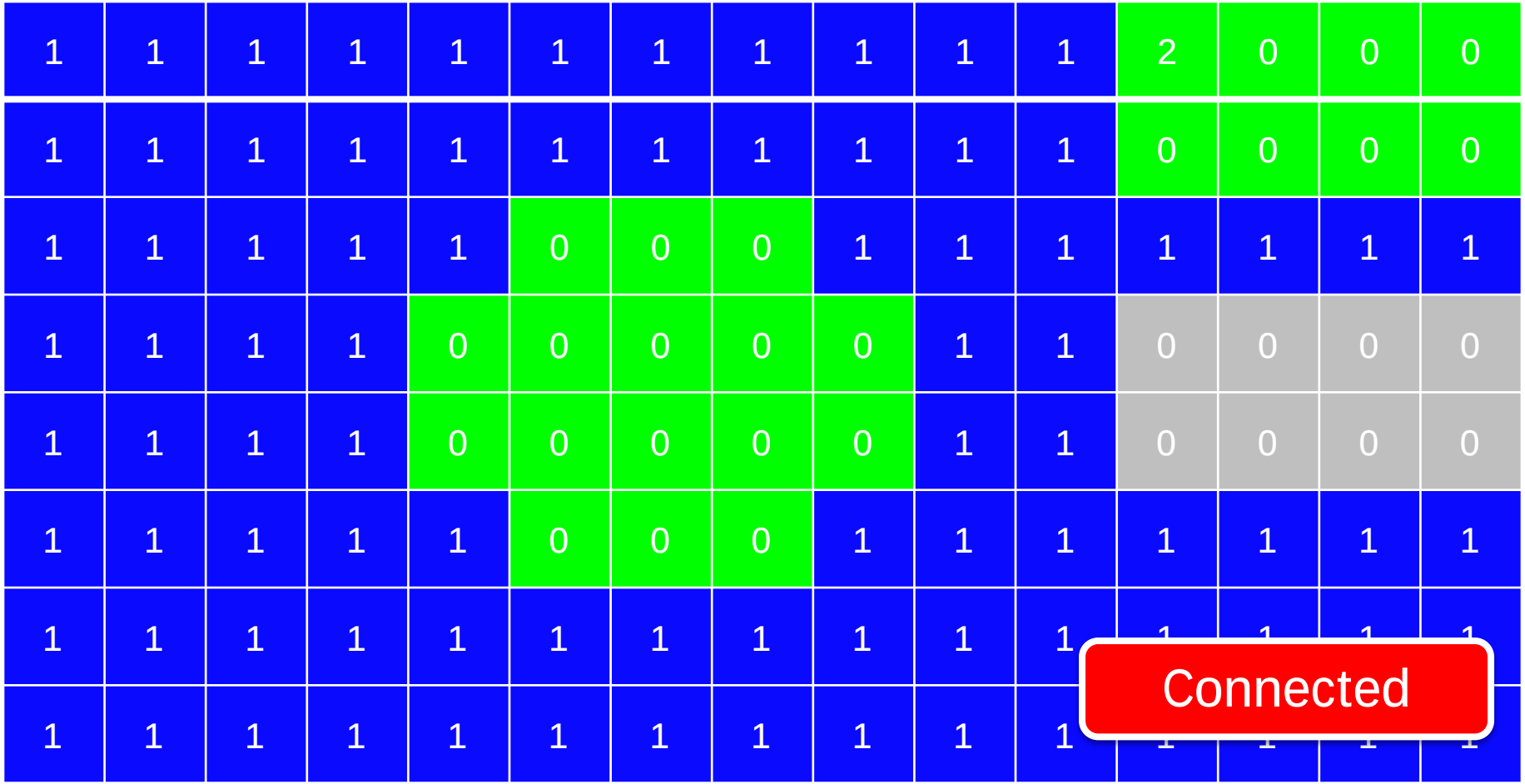
1	1	1	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0

Connected

1	1	1	1	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0

Connected


1 1 1 1 1 1 1 1 1 1 1 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 0 0 0 0
1 1 1 1 1 0 0 0 1 1 1 1 1 1 1
1 1 1 1 0 0 0 0 0 1 1 0 0 0 0
1 1 1 1 0 0 0 0 0 1 1 0 0 0 0
1 1 1 1 1 0 0 0 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1
Connected


1 1 1 1 1 1 1 1 1 1 1 2 0 0 0
1 1 1 1 1 1 1 1 1 1 1 0 0 0 0
1 1 1 1 1 0 0 0 1 1 1 1 1 1 1
1 1 1 1 0 0 0 0 0 1 1 0 0 0 0
1 1 1 1 0 0 0 0 0 1 1 0 0 0 0
1 1 1 1 1 0 0 0 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1
Connected

1	1	1	1	1	1	1	1	1	1	1	2	2	0	0
1	1	1	1	1	1	1	1	1	1	1	2	0	0	0
1	1	1	1	1	0	0	0	1	1	1	1	1	1	1
1	1	1	1	0	0	0	0	0	1	1	0	0	0	0
1	1	1	1	0	0	0	0	0	1	1	0	0	0	0
1	1	1	1	1	0	0	0	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Connected

1	1	1	1	1	1	1	1	1	1	1	2	2	2	0
1	1	1	1	1	1	1	1	1	1	1	2	2	0	0
1	1	1	1	1	0	0	0	1	1	1	1	1	1	1
1	1	1	1	0	0	0	0	0	1	1	0	0	0	0
1	1	1	1	0	0	0	0	0	1	1	0	0	0	0
1	1	1	1	1	0	0	0	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Connected

1	1	1	1	1	1	1	1	1	1	1	2	2	2	2
1	1	1	1	1	1	1	1	1	1	1	2	2	2	0
1	1	1	1	1	0	0	0	1	1	1	1	1	1	1
1	1	1	1	0	0	0	0	0	1	1	0	0	0	0
1	1	1	1	0	0	0	0	0	1	1	0	0	0	0
1	1	1	1	1	0	0	0	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Connected

1	1	1	1	1	1	1	1	1	1	1	2	2	2	2
1	1	1	1	1	1	1	1	1	1	1	2	2	2	2
1	1	1	1	1	0	0	0	1	1	1	1	1	1	1
1	1	1	1	0	0	0	0	0	1	1	0	0	0	0
1	1	1	1	0	0	0	0	0	1	1	0	0	0	0
1	1	1	1	1	0	0	0	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Connected

1	1	1	1	1	1	1	1	1	1	1	2	2	2	2
1	1	1	1	1	1	1	1	1	1	1	2	2	2	2
1	1	1	1	1	3	0	0	1	1	1	1	1	1	1
1	1	1	1	0	0	0	0	0	1	1	0	0	0	0
1	1	1	1	0	0	0	0	0	1	1	0	0	0	0
1	1	1	1	1	0	0	0	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

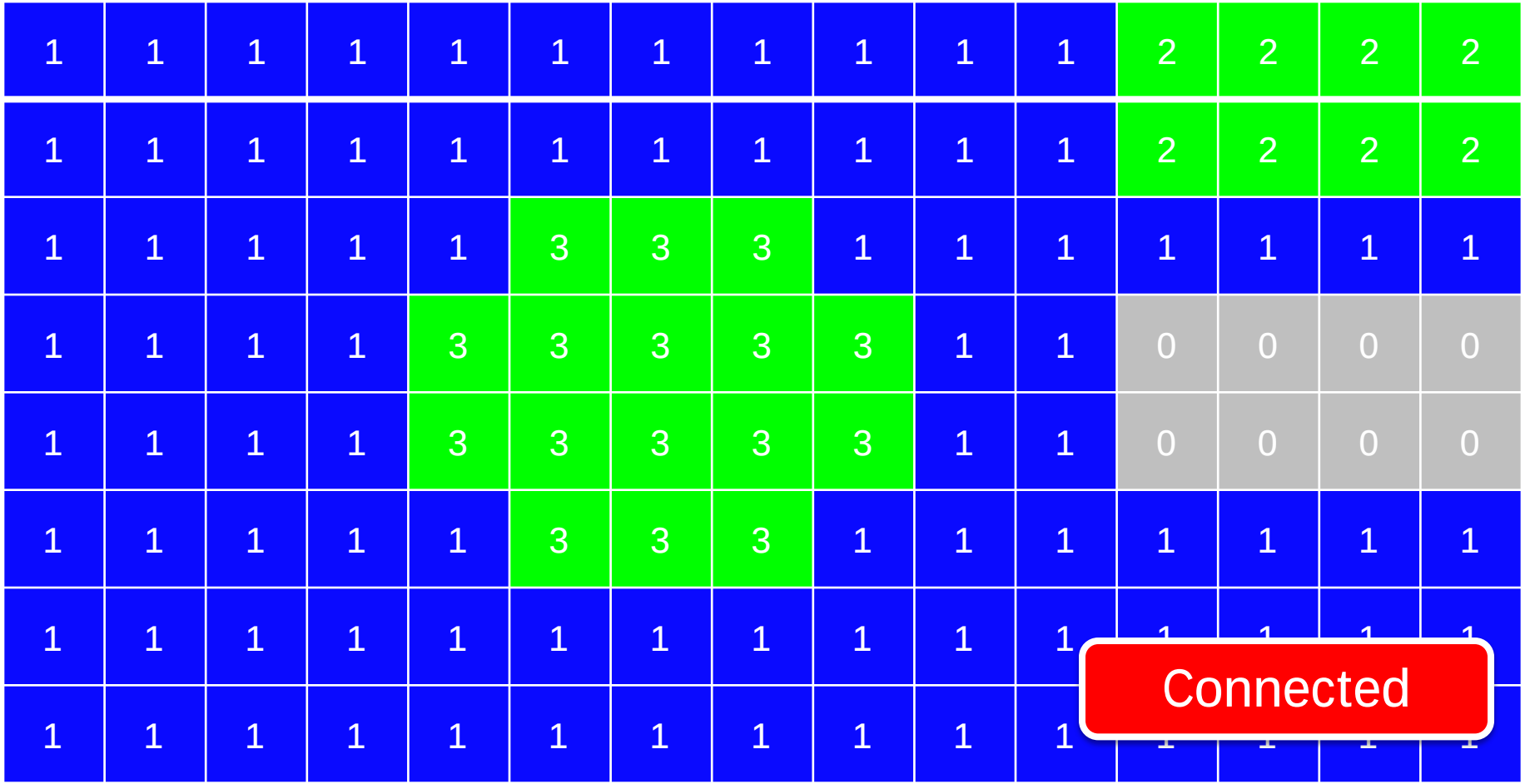
Connected

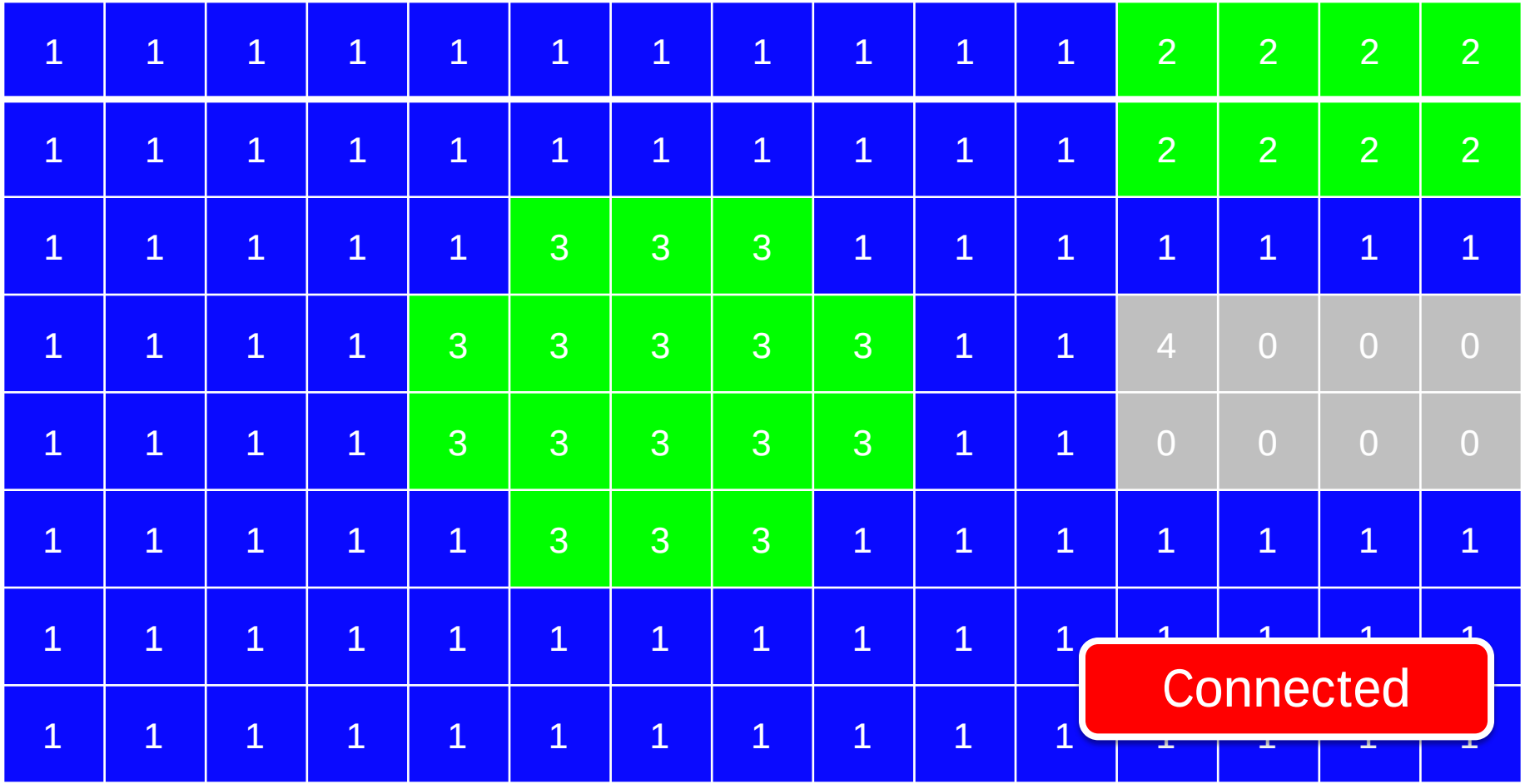
1	1	1	1	1	1	1	1	1	1	1	2	2	2	2
1	1	1	1	1	1	1	1	1	1	1	2	2	2	2
1	1	1	1	1	3	3	0	1	1	1	1	1	1	1
1	1	1	1	0	3	0	0	0	1	1	0	0	0	0
1	1	1	1	0	0	0	0	0	1	1	0	0	0	0
1	1	1	1	1	0	0	0	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

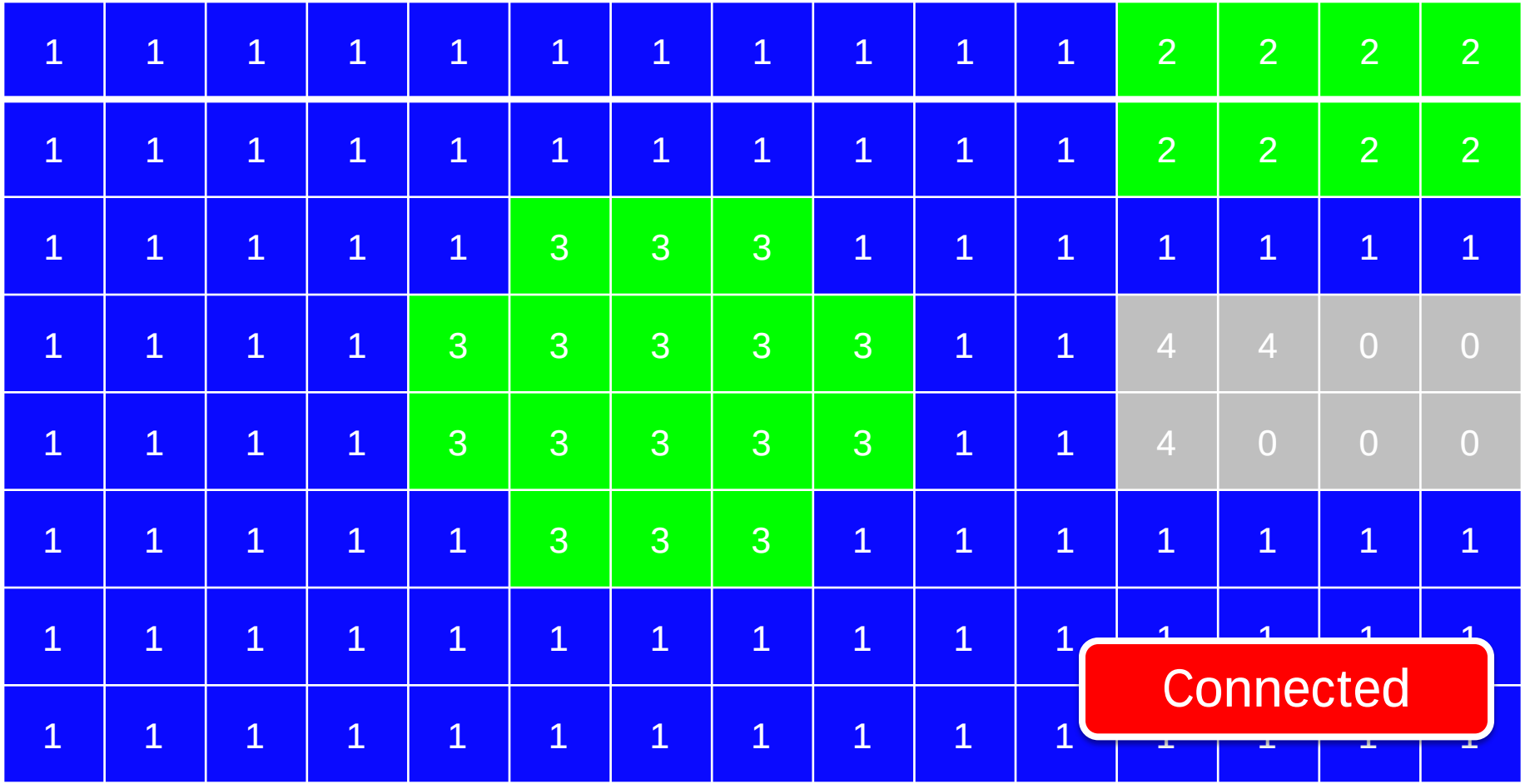
Connected

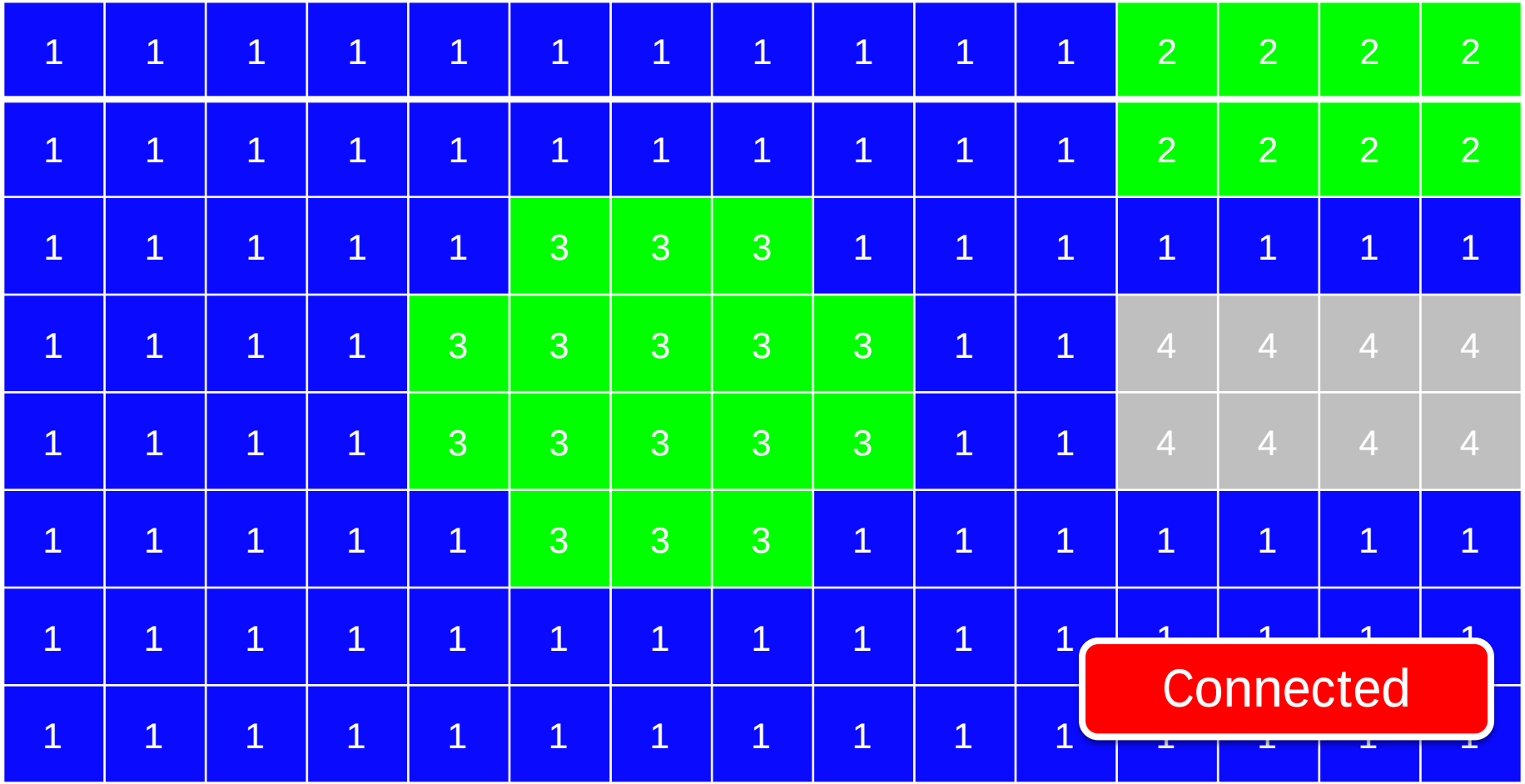
1	1	1	1	1	1	1	1	1	1	1	2	2	2	2
1	1	1	1	1	1	1	1	1	1	1	2	2	2	2
1	1	1	1	1	3	3	3	1	1	1	1	1	1	1
1	1	1	1	3	3	3	0	0	1	1	0	0	0	0
1	1	1	1	0	3	0	0	0	1	1	0	0	0	0
1	1	1	1	1	0	0	0	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

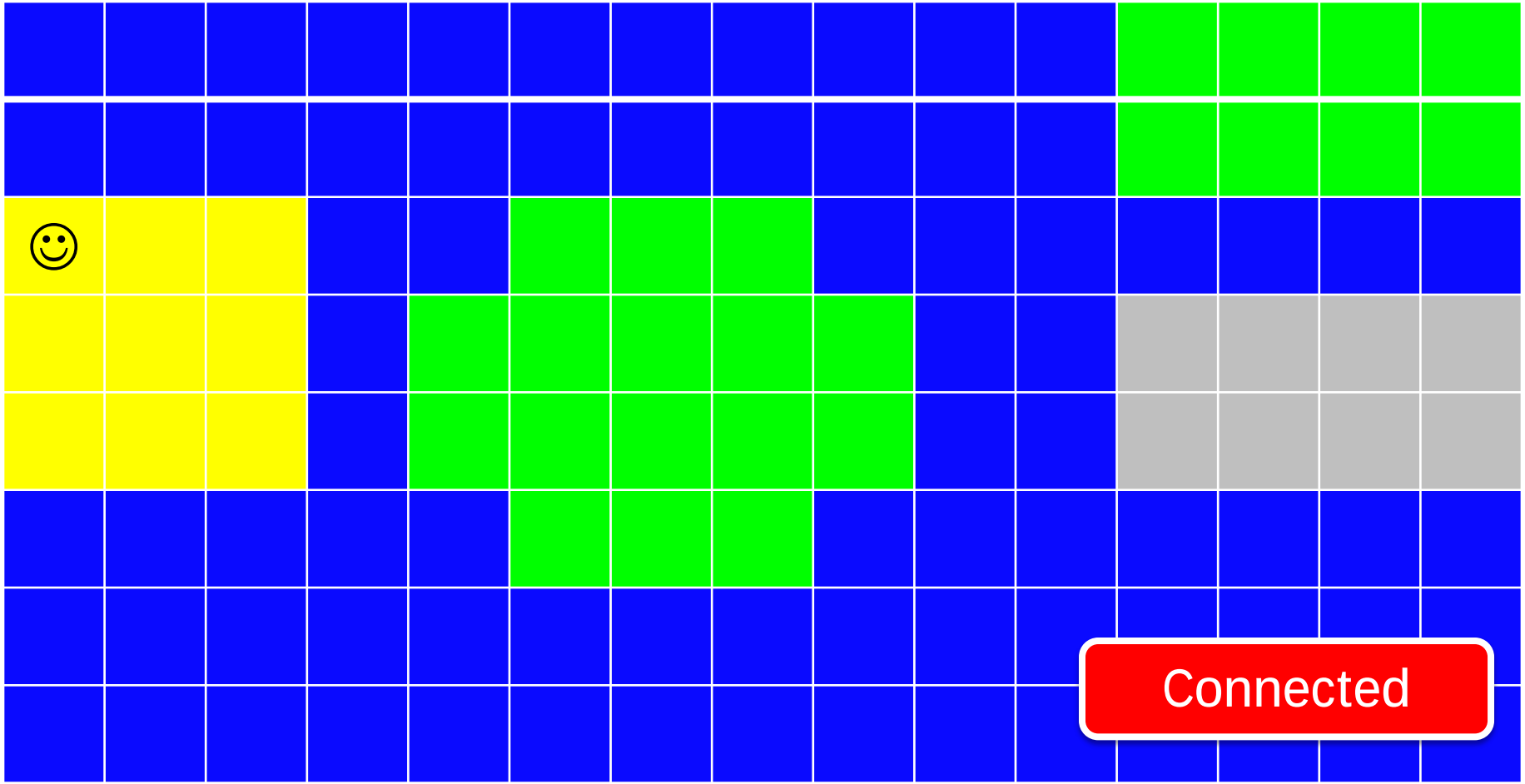
Connected

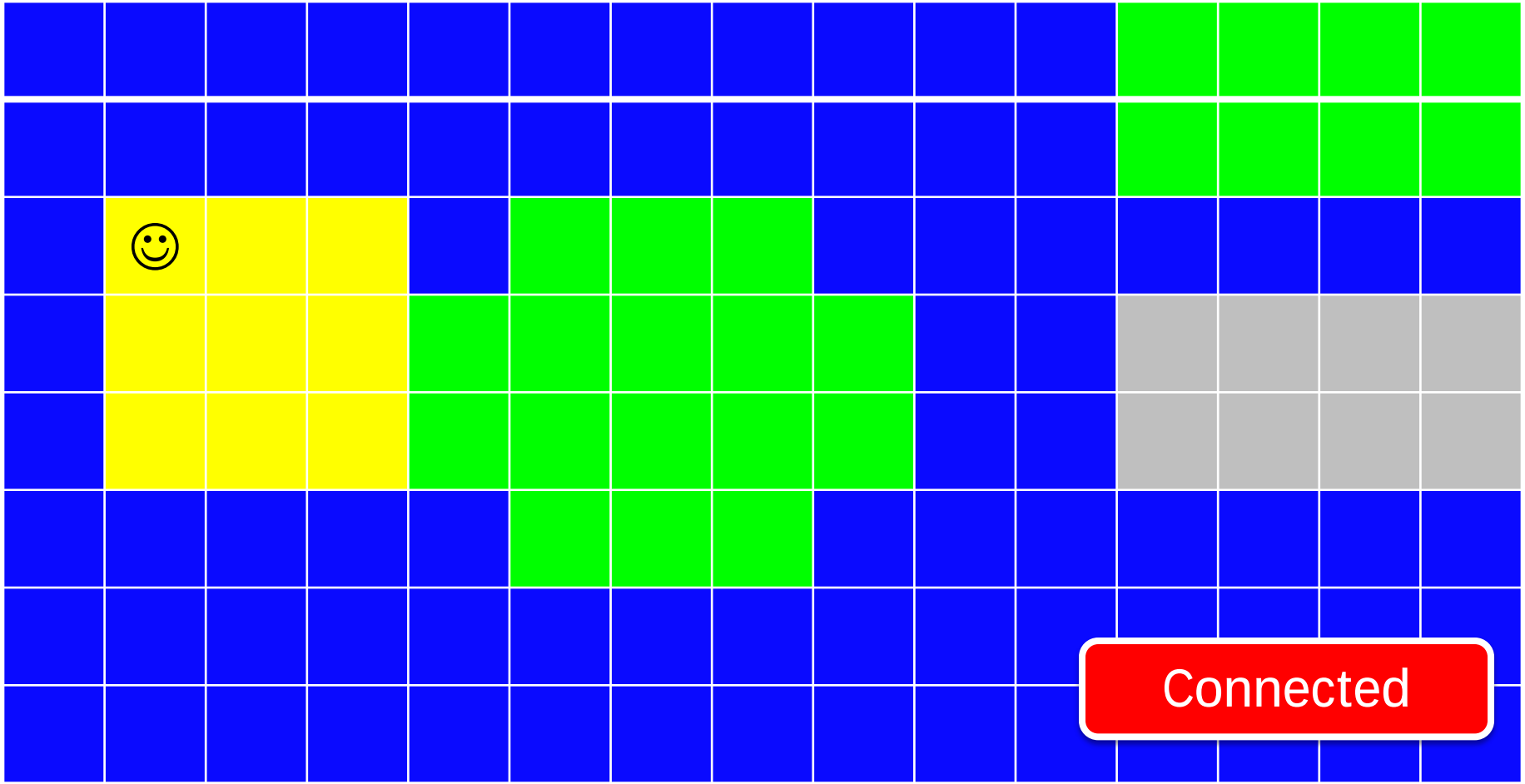

1 1 1 1 1 1 1 1 1 1 1 2 2 2 2
1 1 1 1 1 1 1 1 1 1 1 2 2 2 2
1 1 1 1 1 3 3 3 1 1 1 1 1 1 1
1 1 1 1 3 3 3 3 3 1 1 0 0 0 0
1 1 1 1 3 3 3 3 3 1 1 0 0 0 0
1 1 1 1 1 3 3 3 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1
Connected

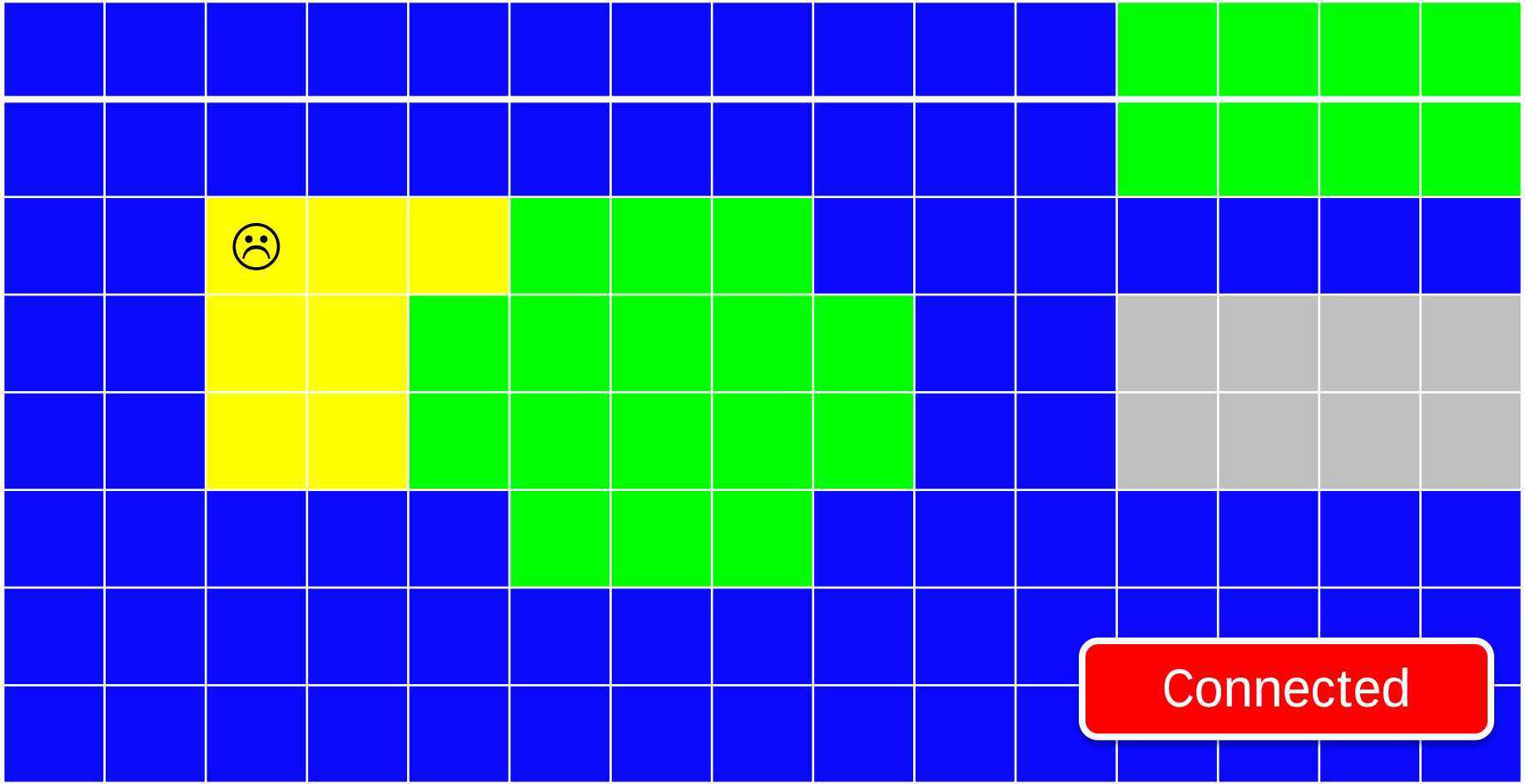

1 1 1 1 1 1 1 1 1 1 1 2 2 2 2
1 1 1 1 1 1 1 1 1 1 1 2 2 2 2
1 1 1 1 1 3 3 3 1 1 1 1 1 1 1
1 1 1 1 3 3 3 3 3 1 1 4 0 0 0
1 1 1 1 3 3 3 3 3 1 1 0 0 0 0
1 1 1 1 1 3 3 3 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1
Connected

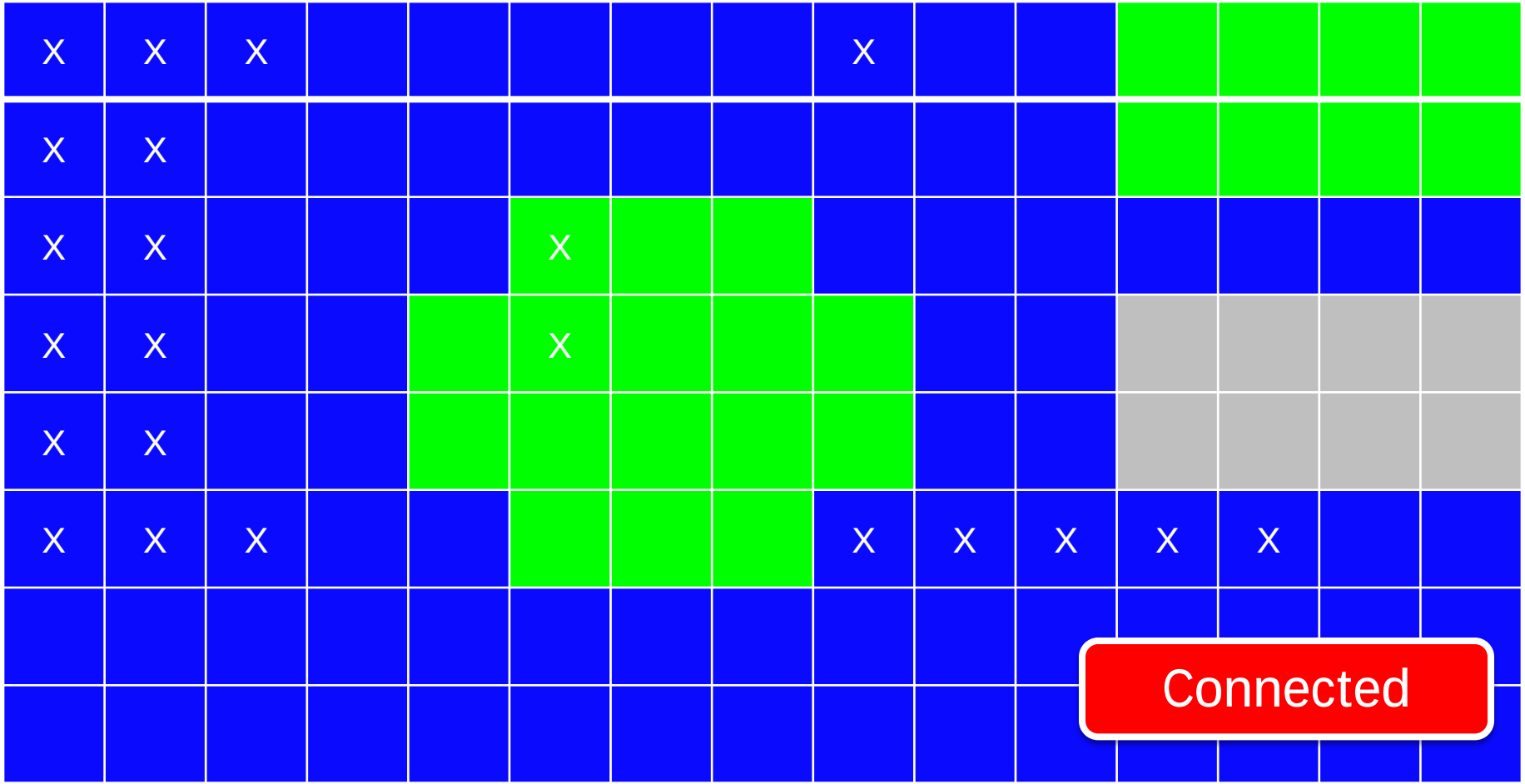

1 1 1 1 1 1 1 1 1 1 1 2 2 2 2
1 1 1 1 1 1 1 1 1 1 1 2 2 2 2
1 1 1 1 1 3 3 3 1 1 1 1 1 1 1
1 1 1 1 3 3 3 3 3 1 1 4 4 0 0
1 1 1 1 3 3 3 3 3 1 1 4 0 0 0
1 1 1 1 1 3 3 3 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1
Connected

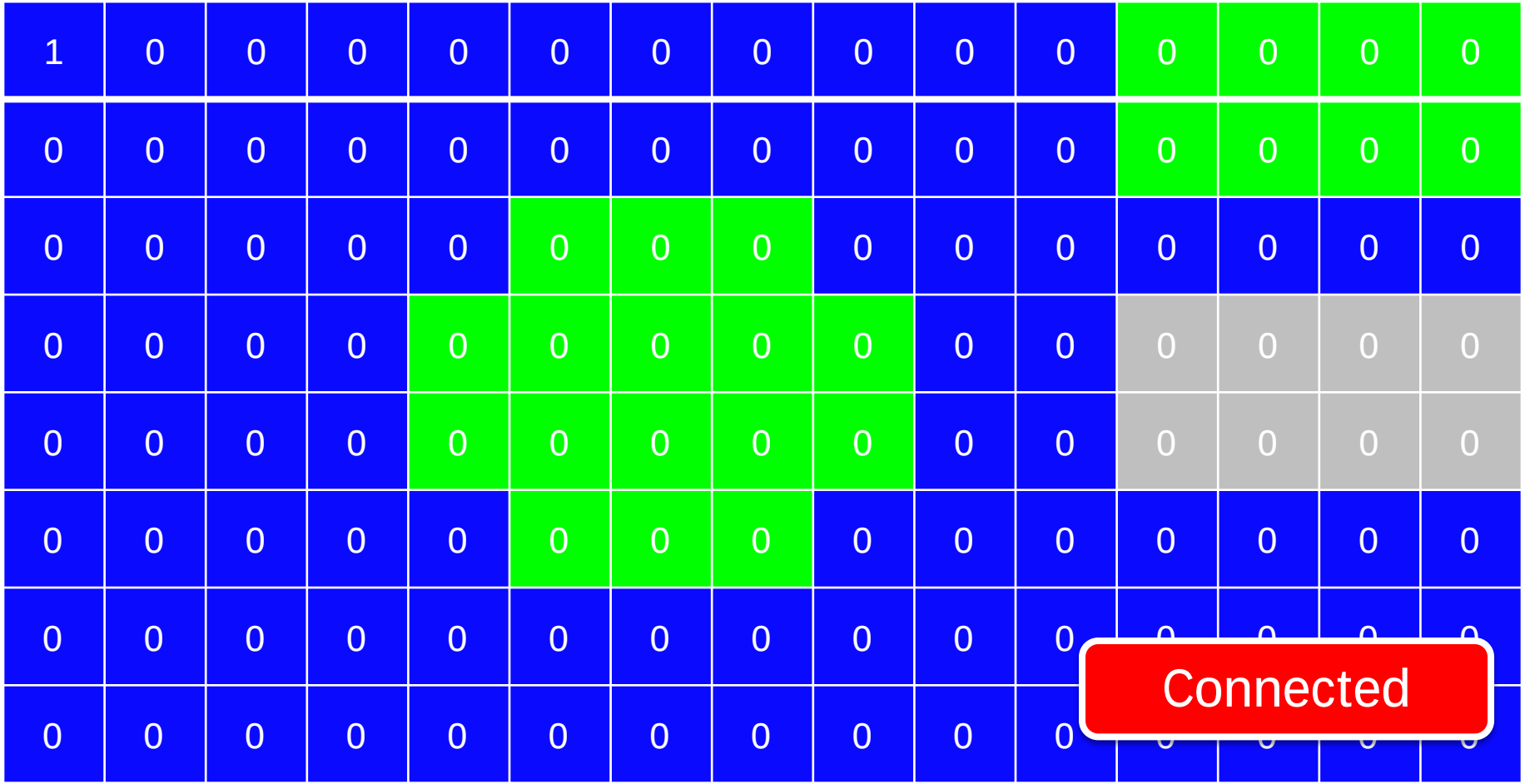

1 1 1 1 1 1 1 1 1 1 1 2 2 2 2
1 1 1 1 1 1 1 1 1 1 1 2 2 2 2
1 1 1 1 1 3 3 3 1 1 1 1 1 1 1
1 1 1 1 3 3 3 3 3 1 1 4 4 4 4
1 1 1 1 3 3 3 3 3 1 1 4 4 4 4
1 1 1 1 1 3 3 3 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1
Connected

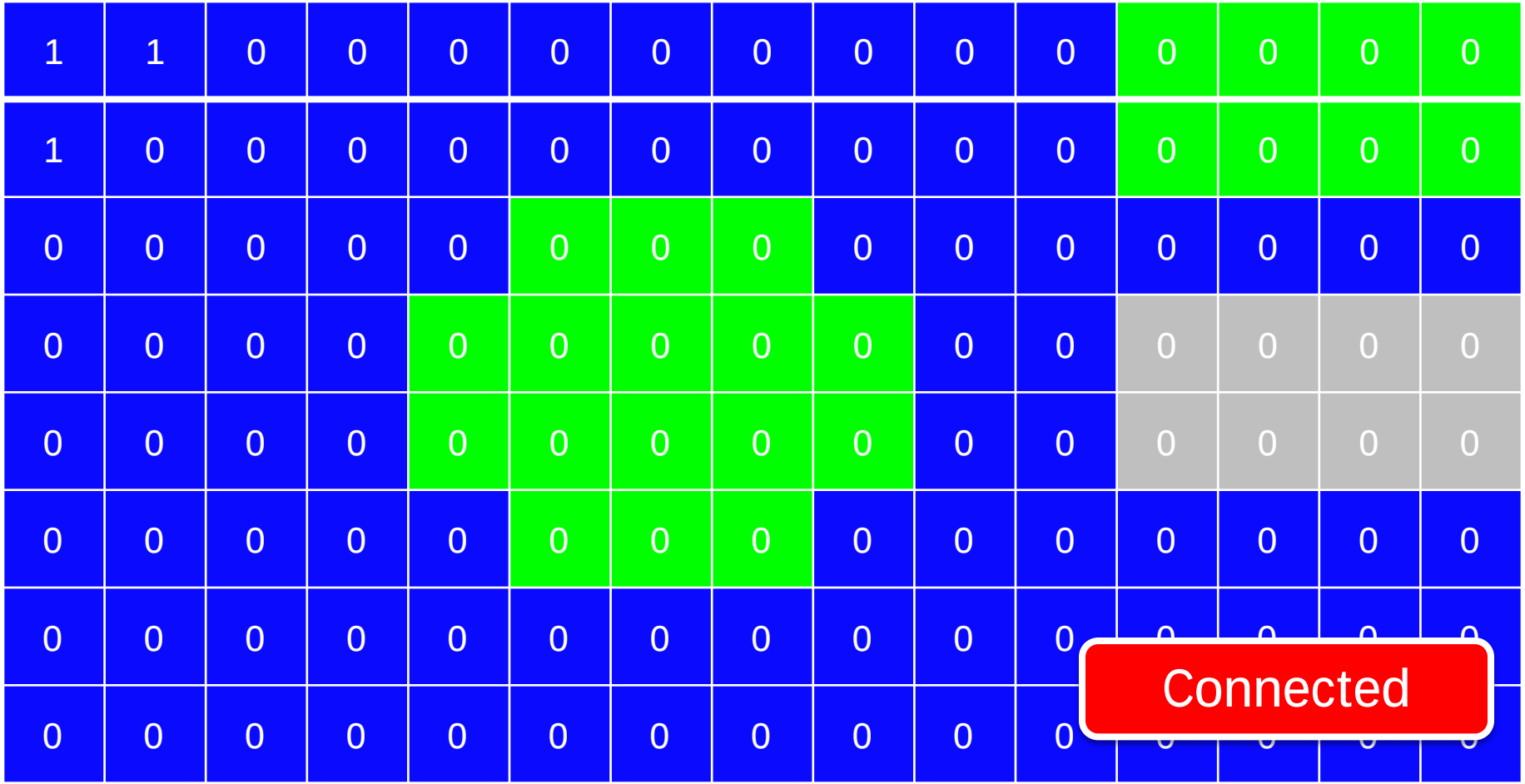

Connected


Connected


Connected

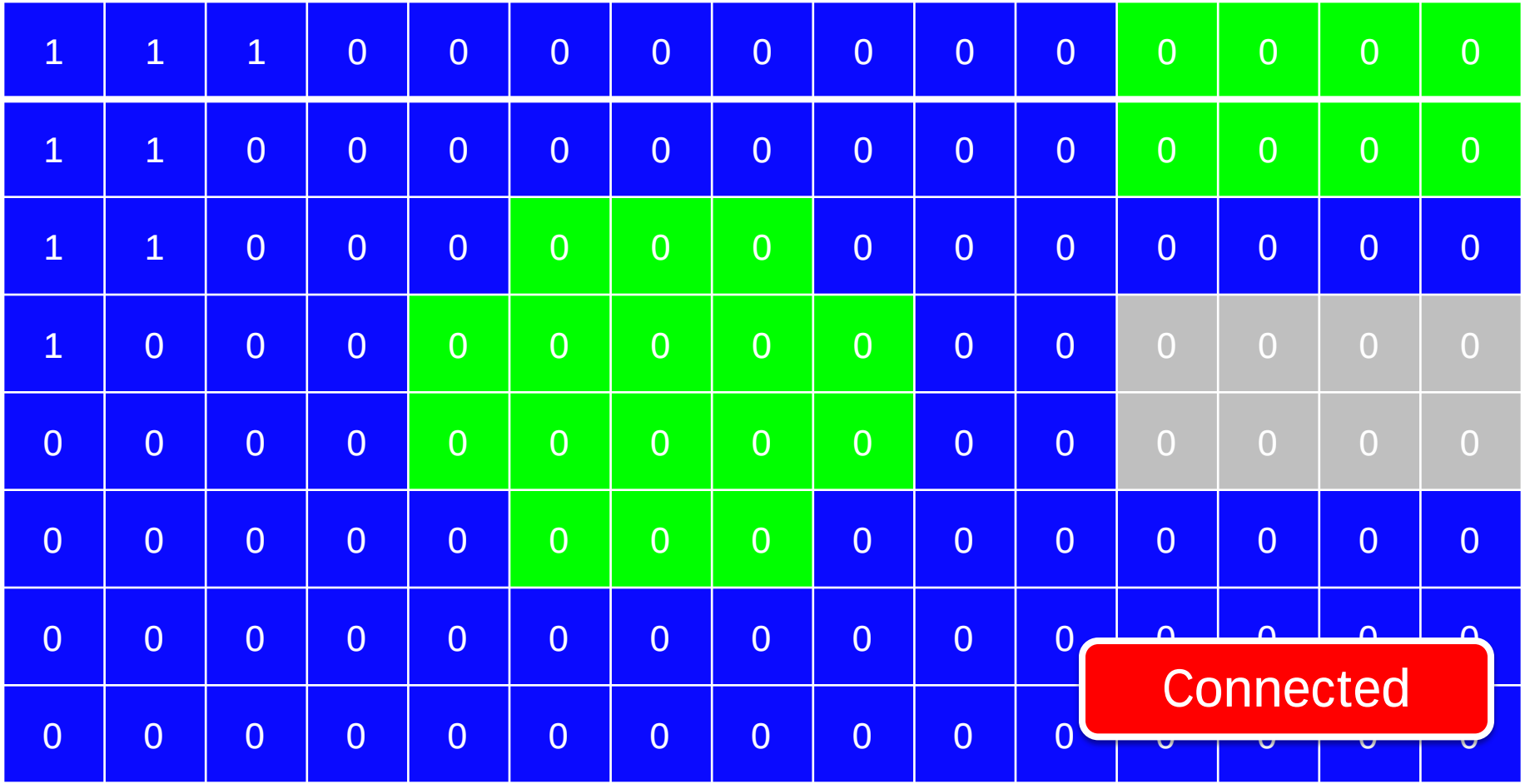

Connected


1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Connected


1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Connected

1	1	1	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0

Connected


1 1 1 0 0 0 0 0 0 0 0 0 0 0
1 1 0 0 0 0 0 0 0 0 0 0 0 0
1 1 0 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
Connected

1	1	1	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0

Connected

1	1	1	0	0	0	0	0	2	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	3	0	0	0	0	0	0	0	0	0
1	1	0	0	0	3	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	4	4	4	4	4	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Connected

A* Graph-Search (Add All)

```
1.  Function AStar(problem, h(n))
2.      closed = empty set                      # stores states (tile locations)
3.      open = [(Node(problem.initial_state)]   # stores Nodes
4.      while (true)
5.          if (open.empty) return fail
6.          node = remove_min_f_node(open)
7.          if (node.state is goal) return solution
8.          closed.add(node.state)
9.          for child in Expand(node, problem)
10.             if (child.state in closed) continue
11.             child.f = child.g + h(child)
12.             open.add(child)
```

A* Graph-Search (Skip if Worse)

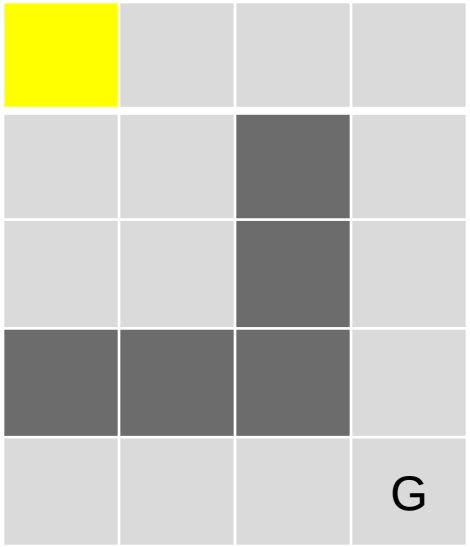
```
1.  Function AStar(problem, h(n))
2.      closed = empty set                          # stores states (tile locations)
3.      open = [(Node(problem.initial_state)]       # stores Nodes
4.      while (true)
5.          if (open.empty) return fail
6.          node = remove_min_f_node(open)
7.          if (node.state is goal) return solution
8.          closed.add(node.state)
9.          for child in Expand(node, problem)
10.             if (child.state in closed) continue
11.             child.f = child.g + h(child)
12.             if (any node in open with state = child.state and g < child.g) continue
13.             open.add(child)
```

A* Graph-Search (Replace Existing Worse)

```
1.  Function AStar(problem, h(n))
2.      closed = empty set                          # stores states (tile locations)
3.      open = [(Node(problem.initial_state)]       # stores Nodes
4.      while (true)
5.          if (open.empty) return fail
6.          node = remove_min_f_node(open)
7.          if (node.state is goal) return solution
8.          closed.add(node.state)
9.          for child in Expand(node, problem)
10.             if (child.state in closed) continue
11.             child.f = child.g + h(child)
12.             if (any node in open with state = child.state and g <= child.g) continue
13.             if (any node in open with state = child.state and g > child.g)
14.                 update_in_open_list(node_in_open, child)
15.             else open.add(child)
```

Node Implementation

```
class Node:
    def __init__(self, tile):
        self.state = tile
        self.g, self.h, self.f = 0, 0, 0
        self.action = (0, 0)
        self.parent = None
```




```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

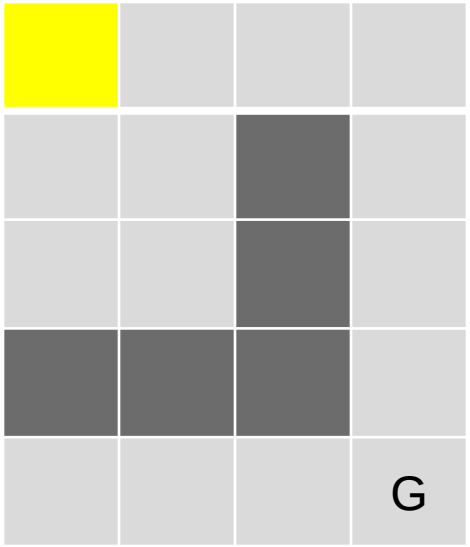
Open List (Nodes)

ID	S	G	H	F	A	P
0	0,0	0	423	423	-1,-1	

Closed

State

Action Costs:
Cardinal: 100
Diagonal: 141



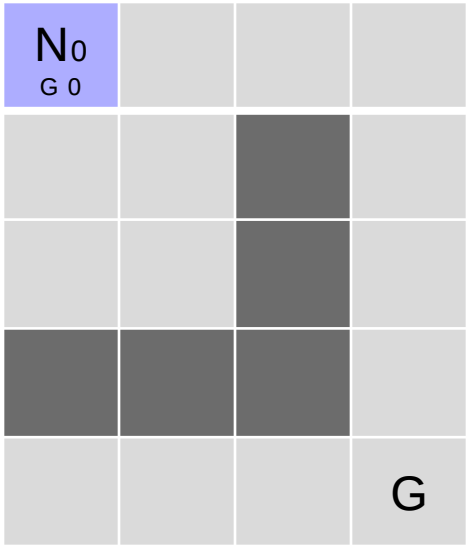

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
0	0,0	0	423	423	-1,-1	

Closed

State



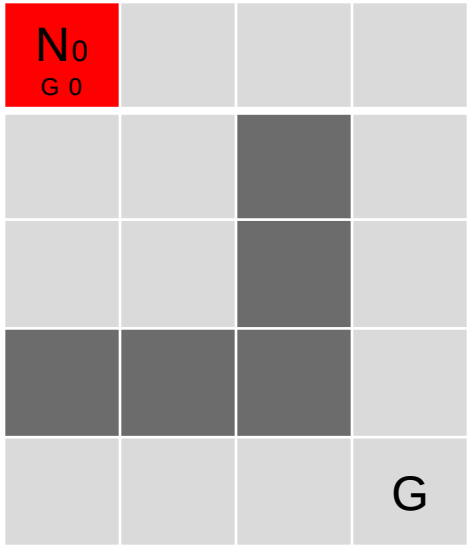

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
0	0,0	0	423	423	-1,-1	

Closed

State



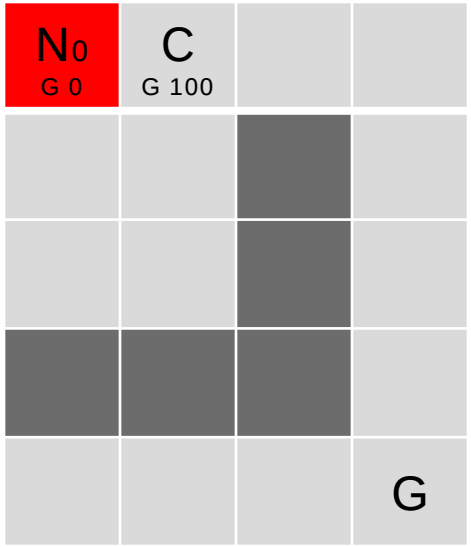

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P

Closed

State
0, 0



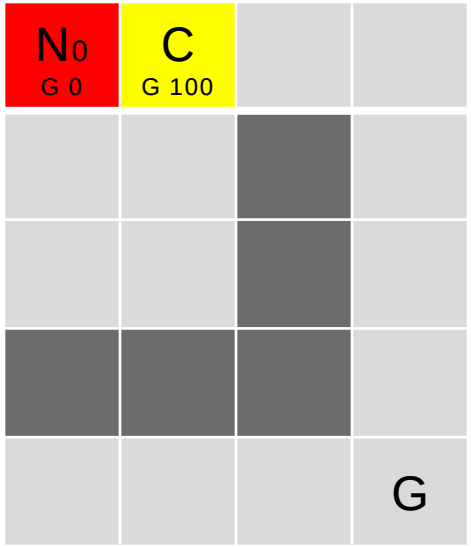

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P

Closed

State
0, 0



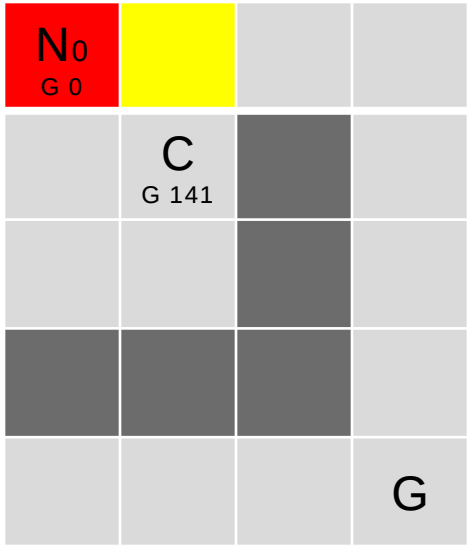

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
1	1,0	100	482	582	1,0	0

Closed

State
0, 0



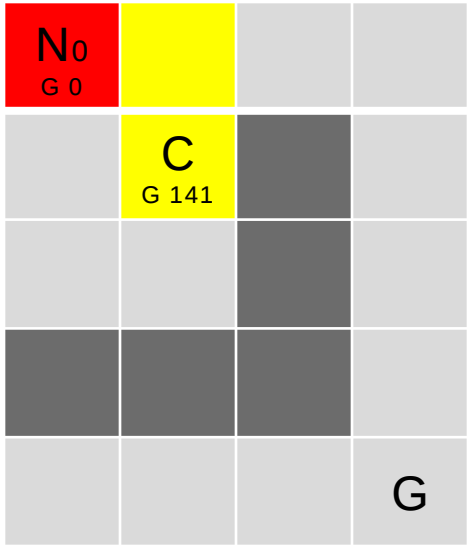

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
1	1,0	100	482	582	1,0	0

Closed

State
0, 0



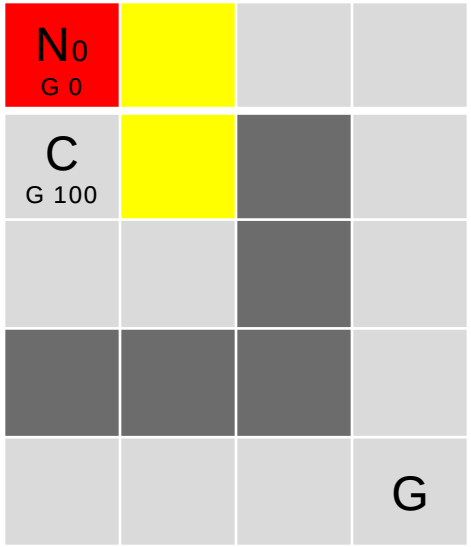

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
1	1,0	100	482	582	1,0	0
2	1,1	141	382	523	1,1	0

Closed

State
0, 0



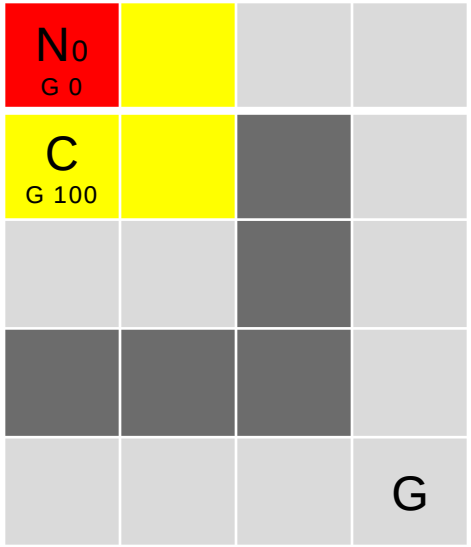

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
1	1,0	100	482	582	1,0	0
2	1,1	141	382	523	1,1	0

Closed

State
0, 0



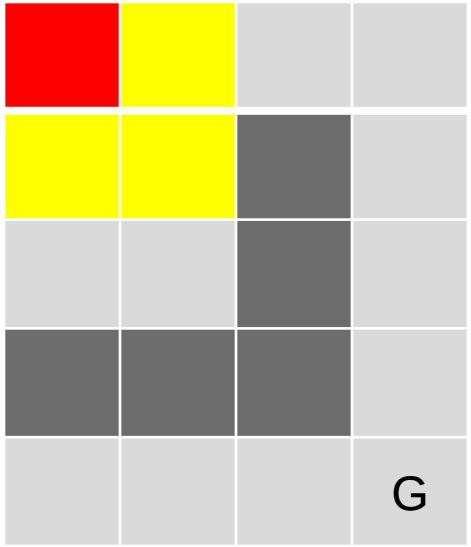

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
1	1,0	100	482	582	1,0	0
2	1,1	141	382	523	1,1	0
3	0,1	100	423	523	0,1	0

Closed

State
0, 0



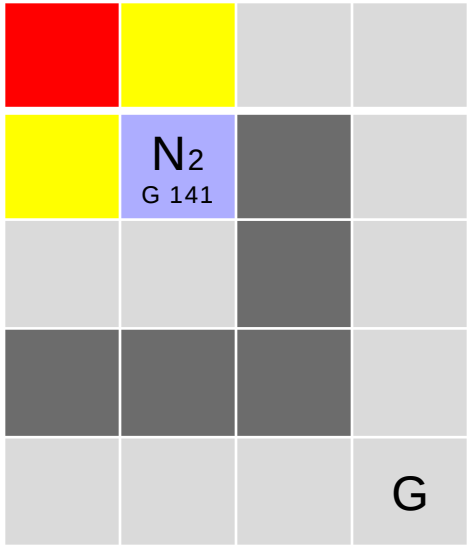

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
1	1,0	100	482	582	1,0	0
2	1,1	141	382	523	1,1	0
3	0,1	100	423	523	0,1	0

Closed

State
0, 0



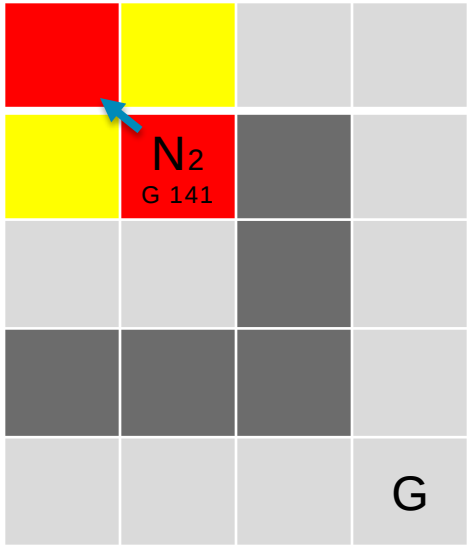

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
1	1,0	100	482	582	1,0	0
2	1,1	141	382	523	1,1	0
3	0,1	100	423	523	0,1	0

Closed

State
0, 0



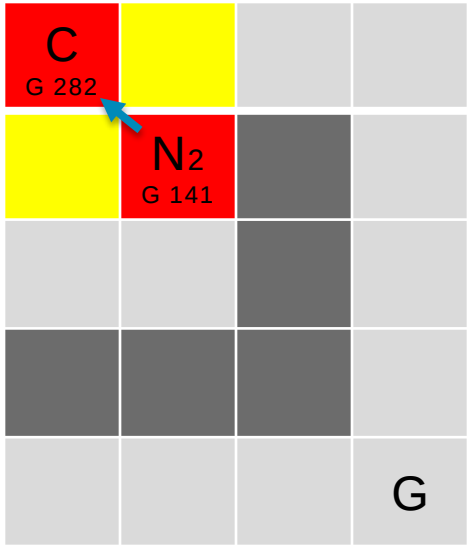

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
1	1,0	100	482	582	1,0	0
3	0,1	100	423	523	0,1	0

Closed

State
0, 0
1, 1



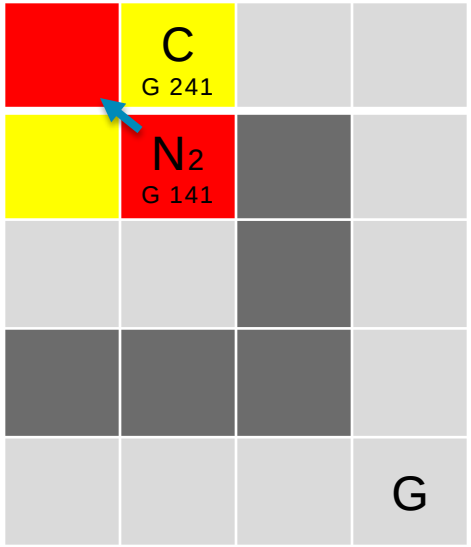

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
1	1,0	100	482	582	1,0	0
3	0,1	100	423	523	0,1	0

Closed

State
0, 0
1, 1



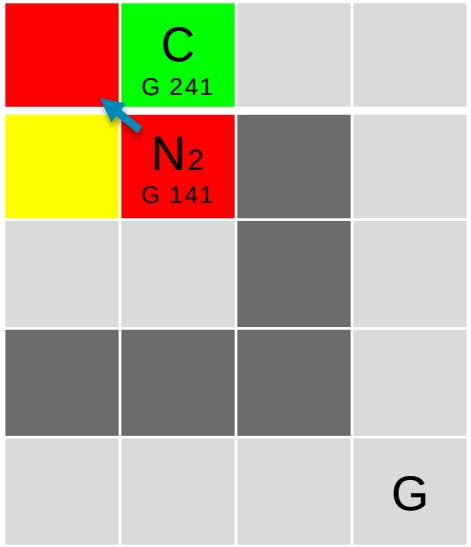

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
1	1,0	100	482	582	1,0	0
3	0,1	100	423	523	0,1	0

Closed

State
0, 0
1, 1



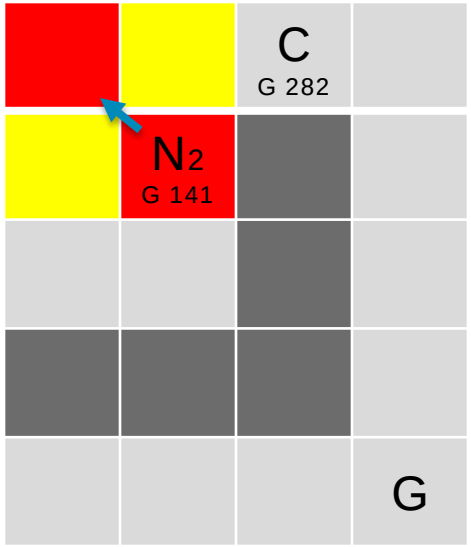

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
1	1,0	100	482	582	1,0	0
3	0,1	100	423	523	0,1	0

Closed

State
0, 0
1, 1



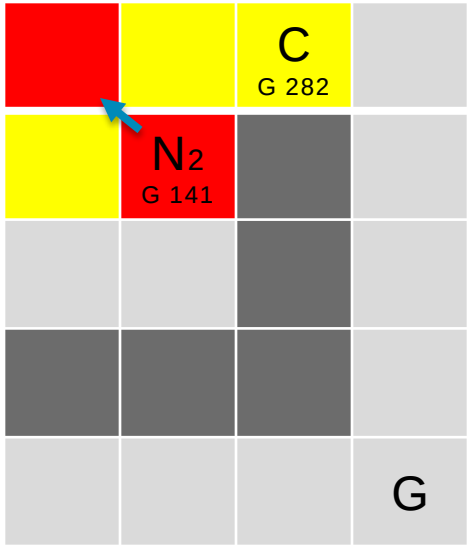

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
1	1,0	100	482	582	1,0	0
3	0,1	100	423	523	0,1	0

Closed

State
0, 0
1, 1



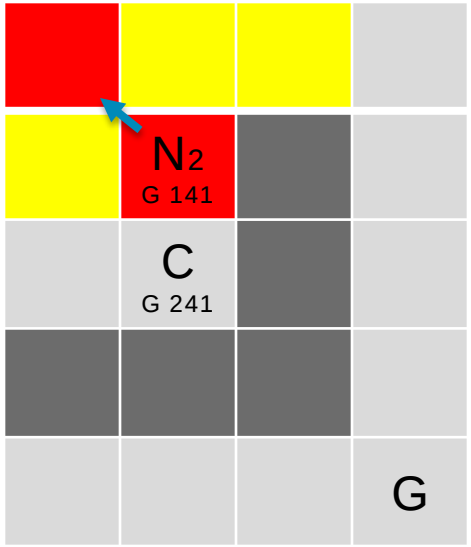

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
1	1,0	100	482	582	1,0	0
3	0,1	100	423	523	0,1	0
4	2,0	282	441	723	1,-1	2

Closed

State
0, 0
1, 1



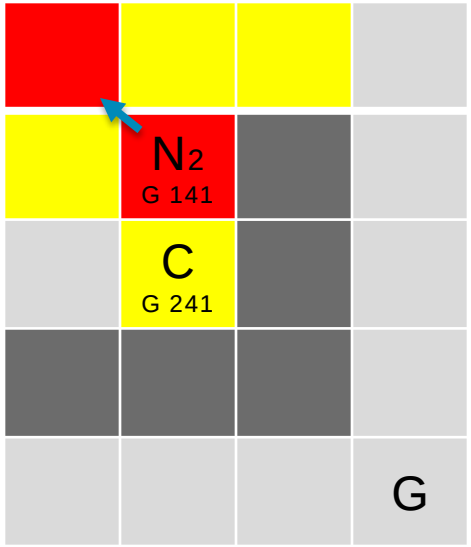

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
1	1,0	100	482	582	1,0	0
3	0,1	100	423	523	0,1	0
4	2,0	282	441	723	1,-1	2

Closed

State
0, 0
1, 1



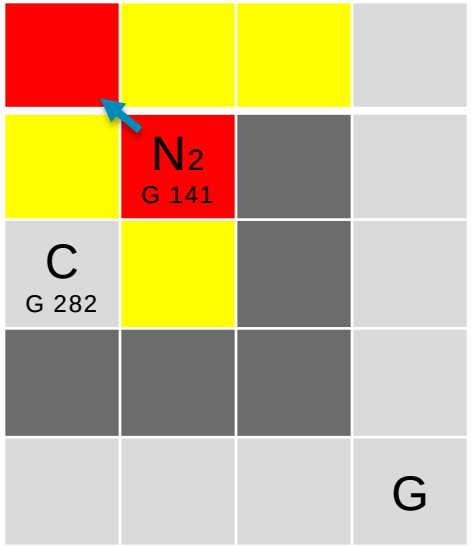

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
1	1,0	100	482	582	1,0	0
3	0,1	100	423	523	0,1	0
4	2,0	282	441	723	1,-1	2
5	1,2	241	282	523	0,1	2

Closed

State
0, 0
1, 1



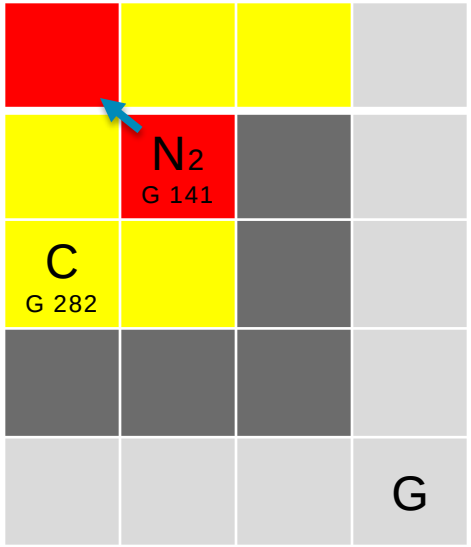

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
1	1,0	100	482	582	1,0	0
3	0,1	100	423	523	0,1	0
4	2,0	282	441	723	1,-1	2
5	1,2	241	282	523	0,1	2

Closed

State
0, 0
1, 1



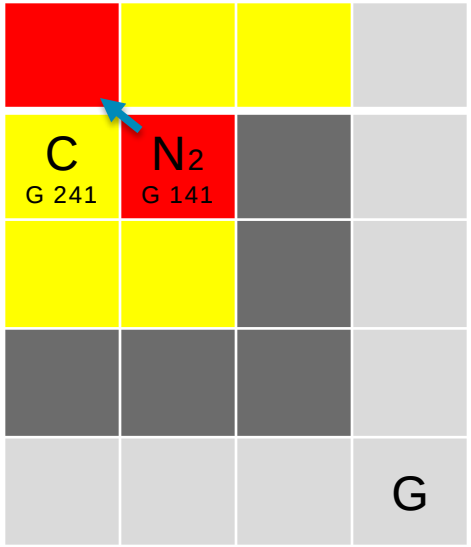

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
1	1,0	100	482	582	1,0	0
3	0,1	100	423	523	0,1	0
4	2,0	282	441	723	1,-1	2
5	1,2	241	282	523	0,1	2
6	0,2	282	382	664	-1,1	2

Closed

State
0, 0
1, 1



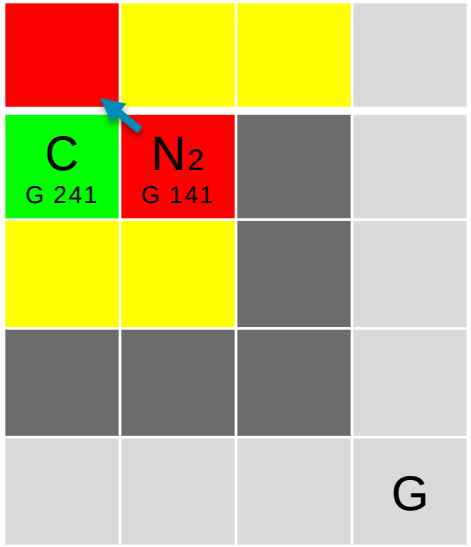

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
1	1,0	100	482	582	1,0	0
3	0,1	100	423	523	0,1	0
4	2,0	282	441	723	1,-1	2
5	1,2	241	282	523	0,1	2
6	0,2	282	382	664	-1,1	2

Closed

State
0, 0
1, 1



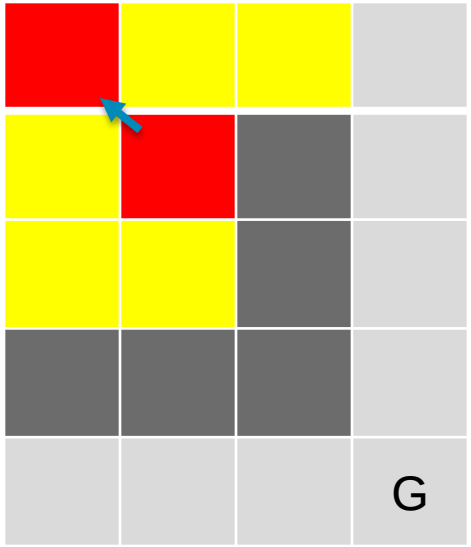

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
1	1,0	100	482	582	1,0	0
3	0,1	100	423	523	0,1	0
4	2,0	282	441	723	1,-1	2
5	1,2	241	282	523	0,1	2
6	0,2	282	382	664	-1,1	2

Closed

State
0, 0
1, 1



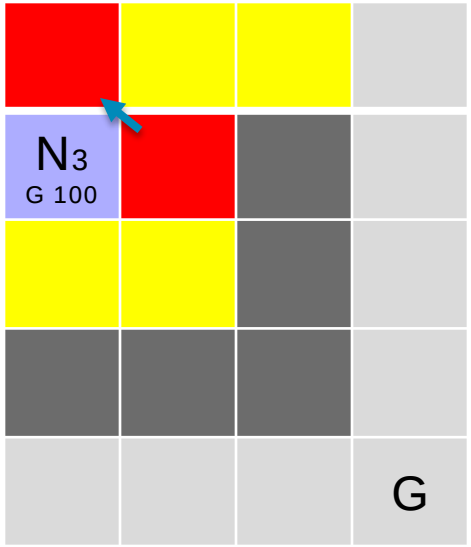

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
1	1,0	100	482	582	1,0	0
3	0,1	100	423	523	0,1	0
4	2,0	282	441	723	1,-1	2
5	1,2	241	282	523	0,1	2
6	0,2	282	382	664	-1,1	2

Closed

State
0, 0
1, 1



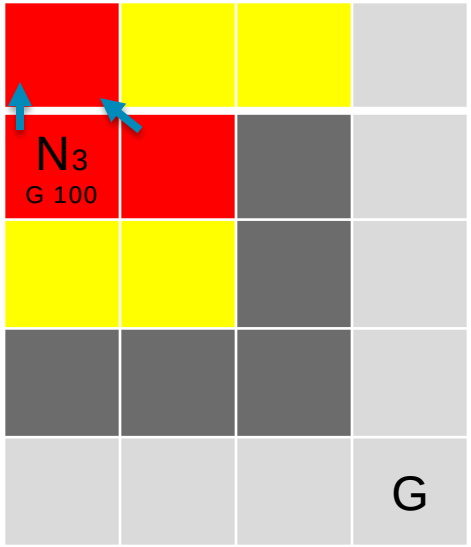

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
1	1,0	100	482	582	1,0	0
3	0,1	100	423	523	0,1	0
4	2,0	282	441	723	1,-1	2
5	1,2	241	282	523	0,1	2
6	0,2	282	382	664	-1,1	2

Closed

State
0, 0
1, 1



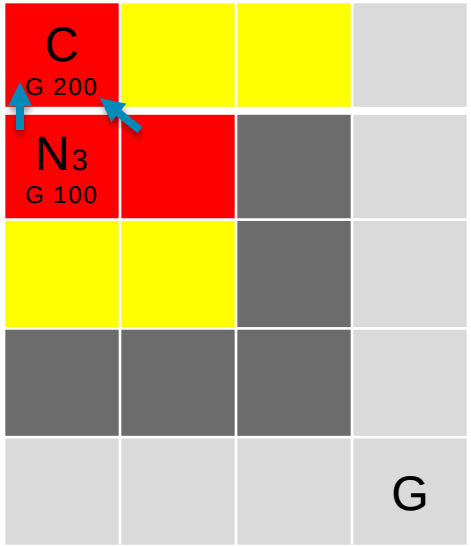

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
1	1,0	100	482	582	1,0	0
4	2,0	282	441	723	1,-1	2
5	1,2	241	282	523	0,1	2
6	0,2	282	382	664	-1,1	2

Closed

State
0, 0
1, 1
0, 1



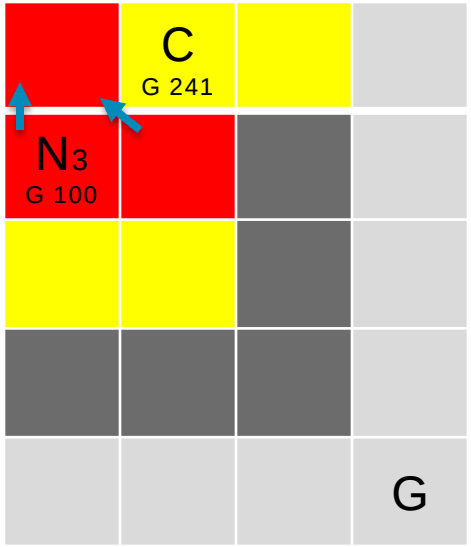

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
1	1,0	100	482	582	1,0	0
4	2,0	282	441	723	1,-1	2
5	1,2	241	282	523	0,1	2
6	0,2	282	382	664	-1,1	2

Closed

State
0, 0
1, 1
0, 1



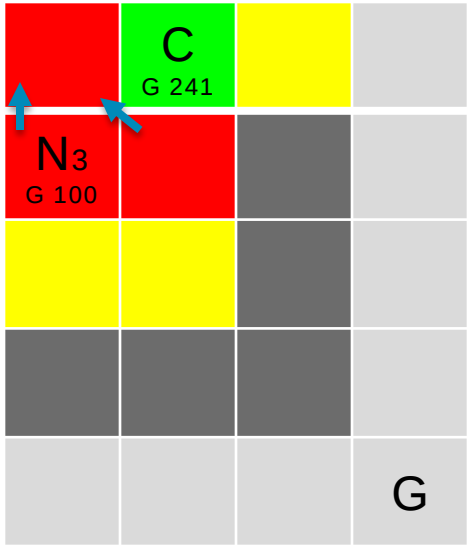

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
1	1,0	100	482	582	1,0	0
4	2,0	282	441	723	1,-1	2
5	1,2	241	282	523	0,1	2
6	0,2	282	382	664	-1,1	2

Closed

State
0, 0
1, 1
0, 1



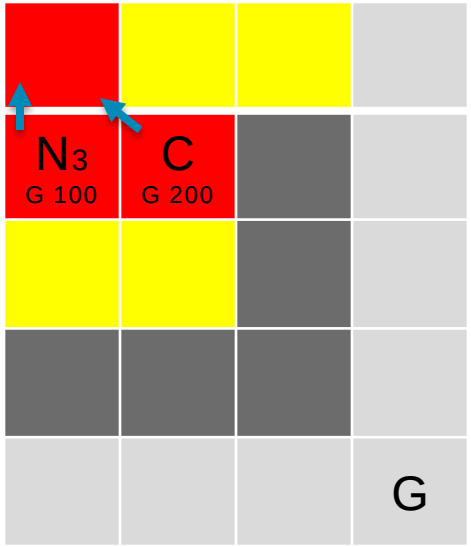

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
1	1,0	100	482	582	1,0	0
4	2,0	282	441	723	1,-1	2
5	1,2	241	282	523	0,1	2
6	0,2	282	382	664	-1,1	2

Closed

State
0, 0
1, 1
0, 1



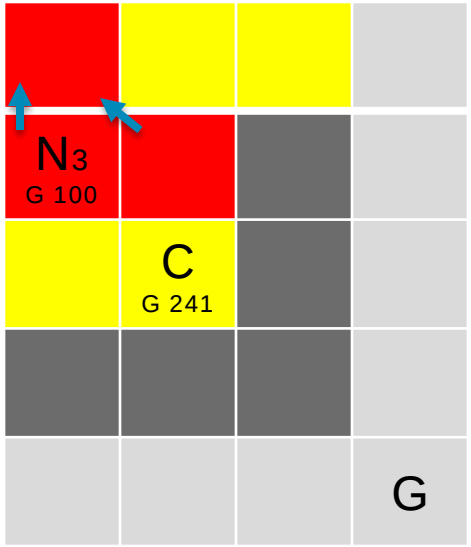

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
1	1,0	100	482	582	1,0	0
4	2,0	282	441	723	1,-1	2
5	1,2	241	282	523	0,1	2
6	0,2	282	382	664	-1,1	2

Closed

State
0, 0
1, 1
0, 1



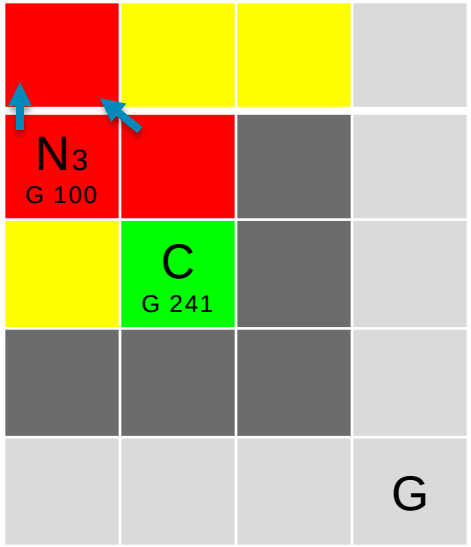

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
1	1,0	100	482	582	1,0	0
4	2,0	282	441	723	1,-1	2
5	1,2	241	282	523	0,1	2
6	0,2	282	382	664	-1,1	2

Closed

State
0, 0
1, 1
0, 1



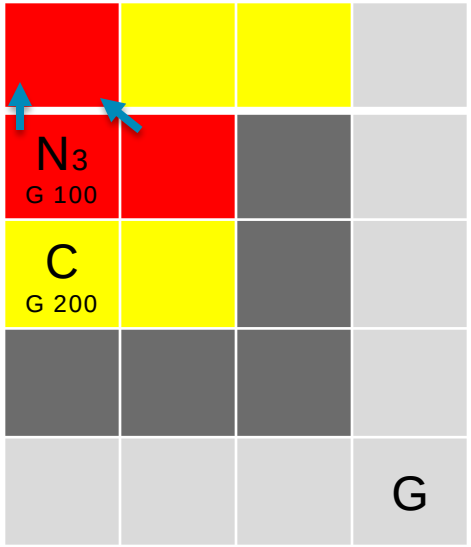

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
1	1,0	100	482	582	1,0	0
4	2,0	282	441	723	1,-1	2
5	1,2	241	282	523	0,1	2
6	0,2	282	382	664	-1,1	2

Closed

State
0, 0
1, 1
0, 1



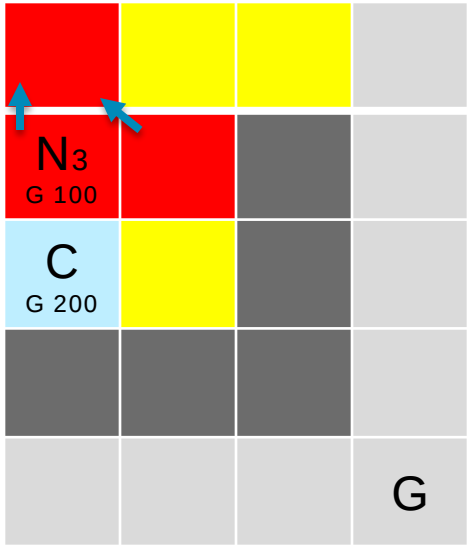

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
1	1,0	100	482	582	1,0	0
4	2,0	282	441	723	1,-1	2
5	1,2	241	282	523	0,1	2
6	0,2	282	382	664	-1,1	2
7	0,2	200	382	582	0,1	3

Closed

State
0, 0
1, 1
0, 1




```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

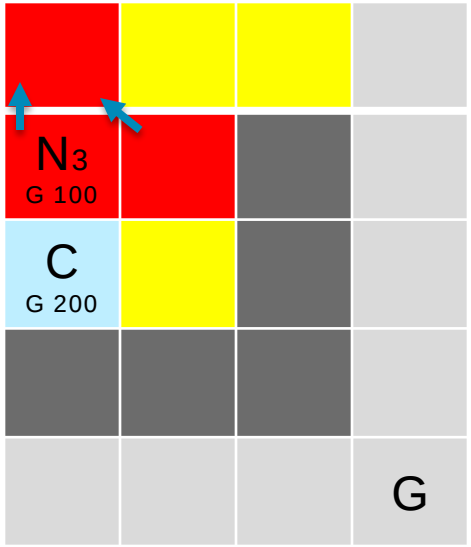
ID	S					P
1	1					0
4	2					2
5	1,2	241	282	523	0,1	2
6	0,2	282	382	664	-1,1	2
7	0,2	200	382	582	0,1	3

Optimization:
Remove Node 6
It Will Never Expand

Even Better:
Don't Add / Remove
Replace 6 with 7

Closed

State
0, 0
1, 1
0, 1



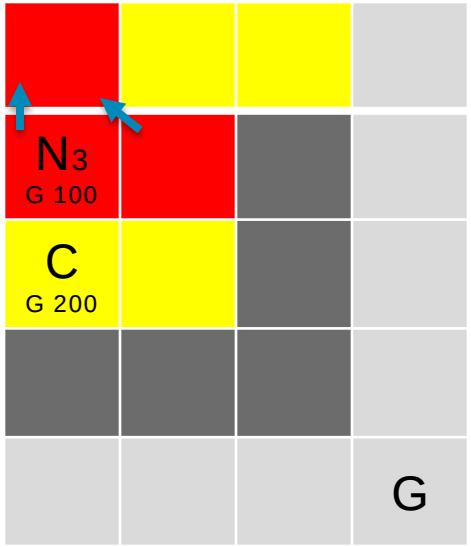

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
1	1,0	100	482	582	1,0	0
4	2,0	282	441	723	1,-1	2
5	1,2	241	282	523	0,1	2
6	0,2	282	382	664	-1,1	2
7	0,2	200	382	582	0,1	3

Closed

State
0, 0
1, 1
0, 1



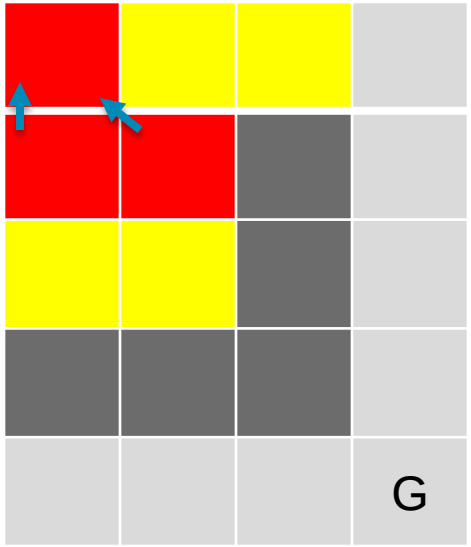

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
1	1,0	100	482	582	1,0	0
4	2,0	282	441	723	1,-1	2
5	1,2	241	282	523	0,1	2
7	0,2	200	382	582	0,1	3

Closed

State
0, 0
1, 1
0, 1



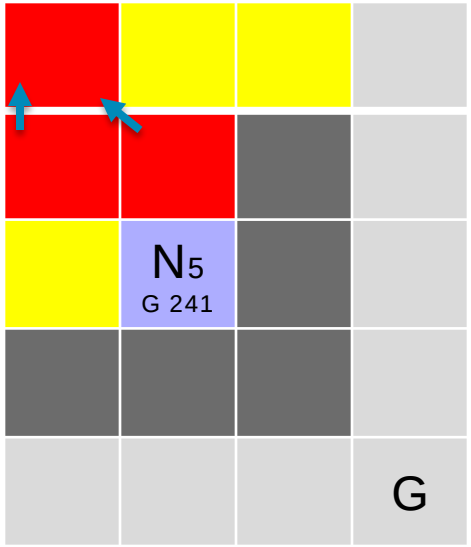

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
1	1,0	100	482	582	1,0	0
4	2,0	282	441	723	1,-1	2
5	1,2	241	282	523	0,1	2
7	0,2	200	382	582	0,1	3

Closed

State
0, 0
1, 1
0, 1



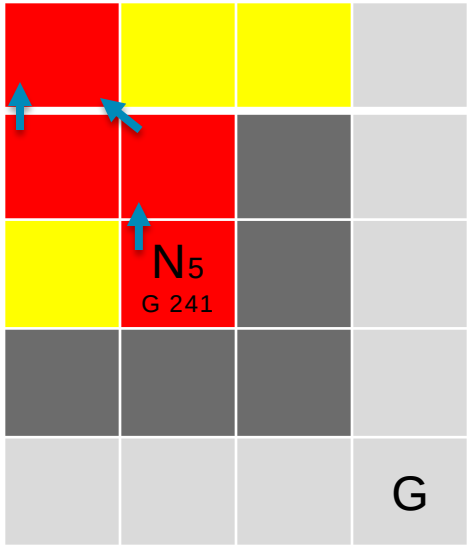

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
1	1,0	100	482	582	1,0	0
4	2,0	282	441	723	1,-1	2
5	1,2	241	282	523	0,1	2
7	0,2	200	382	582	0,1	3

Closed

State
0, 0
1, 1
0, 1



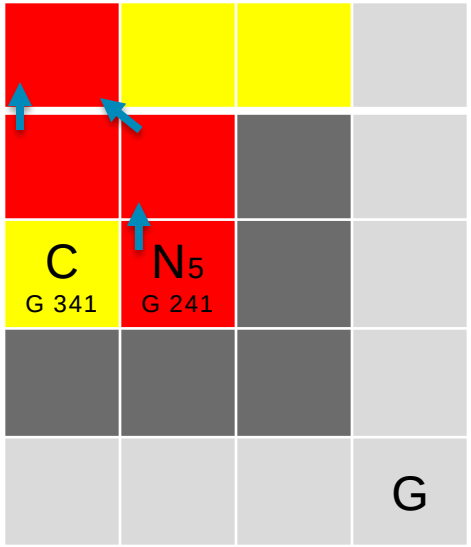

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
1	1,0	100	482	582	1,0	0
4	2,0	282	441	723	1,-1	2
7	0,2	200	382	582	0,1	3

Closed

State
0, 0
1, 1
0, 1
1, 2



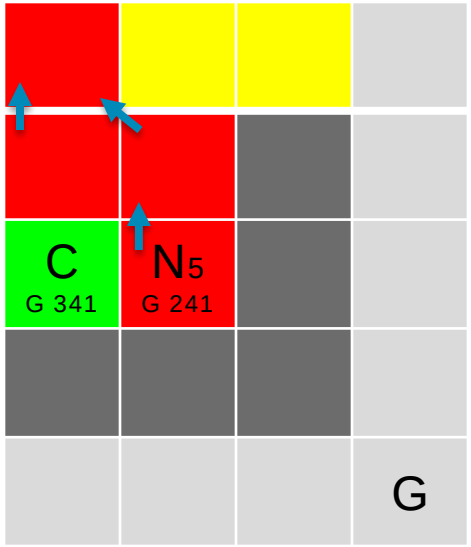

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
1	1,0	100	482	582	1,0	0
4	2,0	282	441	723	1,-1	2
7	0,2	200	382	582	0,1	3

Closed

State
0, 0
1, 1
0, 1
1, 2



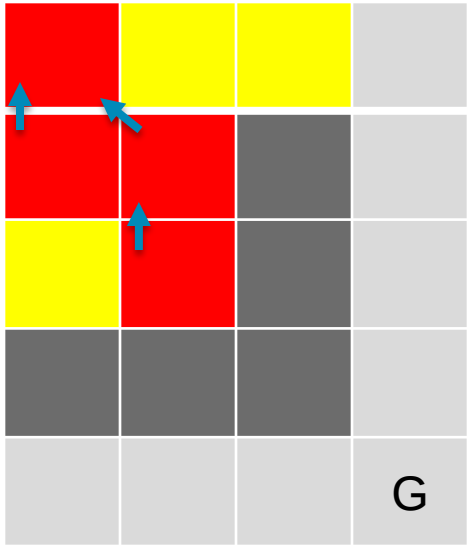

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
1	1,0	100	482	582	1,0	0
4	2,0	282	441	723	1,-1	2
7	0,2	200	382	582	0,1	3

Closed

State
0, 0
1, 1
0, 1
1, 2



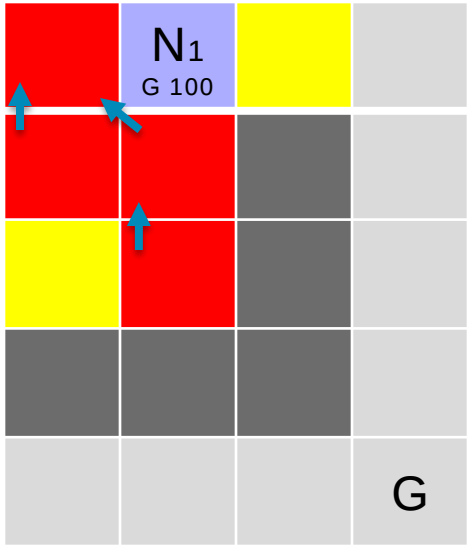

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
1	1,0	100	482	582	1,0	0
4	2,0	282	441	723	1,-1	2
7	0,2	200	382	582	0,1	3

Closed

State
0, 0
1, 1
0, 1
1, 2



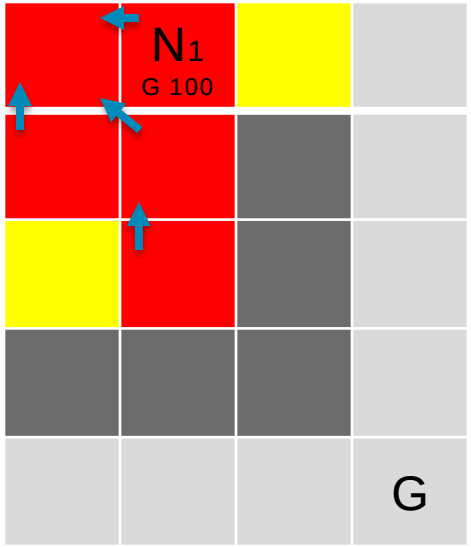

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
1	1,0	100	482	582	1,0	0
4	2,0	282	441	723	1,-1	2
7	0,2	200	382	582	0,1	3

Closed

State
0, 0
1, 1
0, 1
1, 2



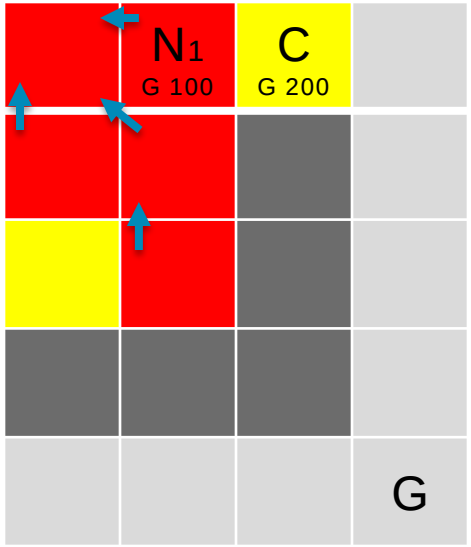

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
4	2,0	282	441	723	1,-1	2
7	0,2	200	382	582	0,1	3

Closed

State
0, 0
1, 1
0, 1
1, 2
1, 0



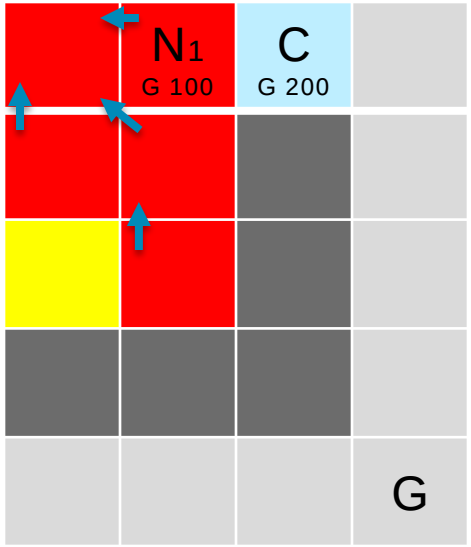

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
4	2,0	282	441	723	1,-1	2
7	0,2	200	382	582	0,1	3

Closed

State
0, 0
1, 1
0, 1
1, 2
1, 0



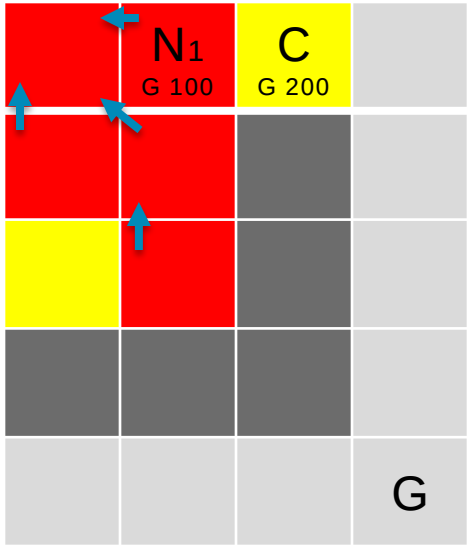

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
4	2,0	282	441	723	1,-1	2
7	0,2	200	382	582	0,1	3
8	2,0	200	441	641	1,0	1

Closed

State
0, 0
1, 1
0, 1
1, 2
1, 0



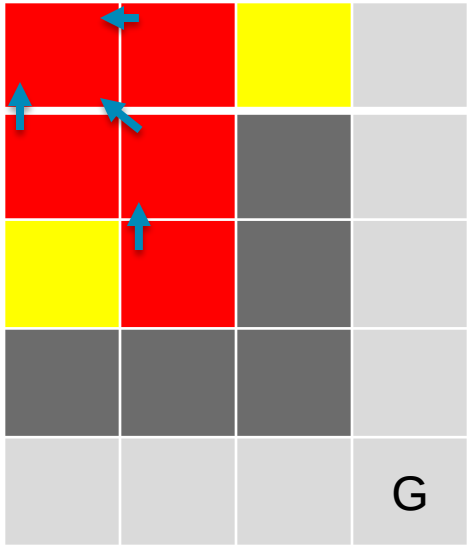

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
8	2,0	200	441	641	1,0	1
7	0,2	200	382	582	0,1	3

Closed

State
0, 0
1, 1
0, 1
1, 2
1, 0



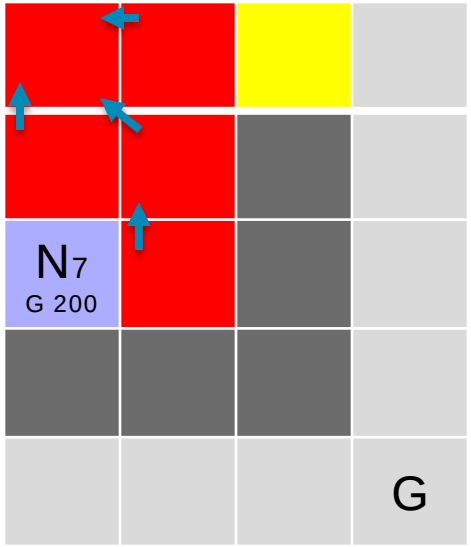

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
8	2,0	200	441	641	1,0	1
7	0,2	200	382	582	0,1	3

Closed

State
0, 0
1, 1
0, 1
1, 2
1, 0



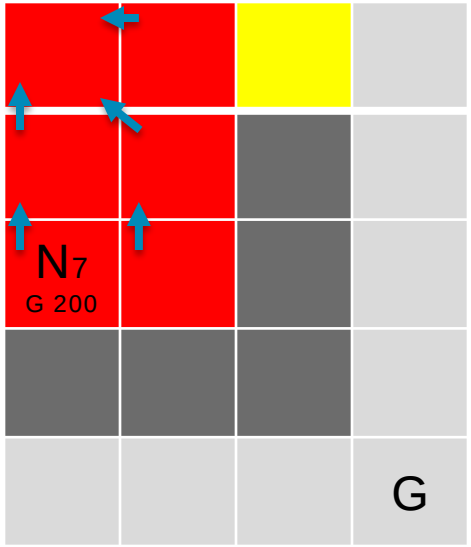

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
8	2,0	200	441	641	1,0	1
7	0,2	200	382	582	0,1	3

Closed

State
0, 0
1, 1
0, 1
1, 2
1, 0



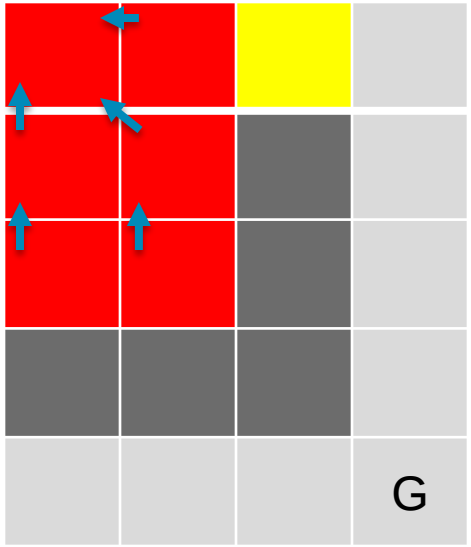

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
8	2,0	200	441	641	1,0	1

Closed

State
0, 0
1, 1
0, 1
1, 2
1, 0
0, 2



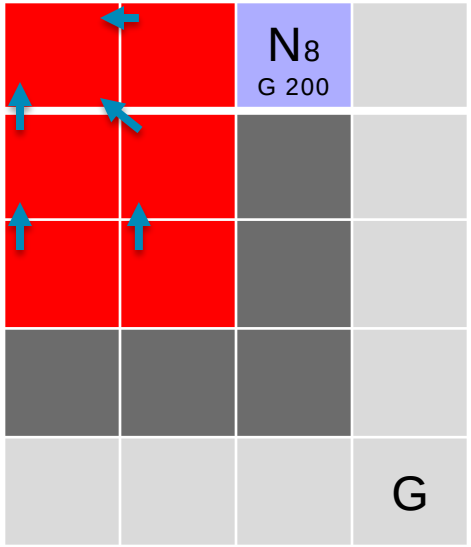

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
8	2,0	200	441	641	1,0	1

Closed

State
0, 0
1, 1
0, 1
1, 2
1, 0
0, 2



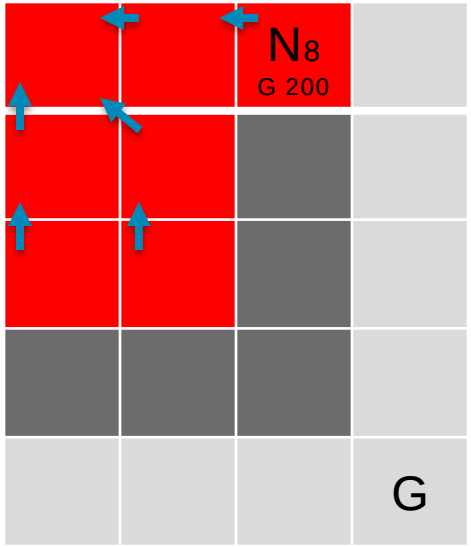

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
8	2,0	200	441	641	1,0	1

Closed

State
0, 0
1, 1
0, 1
1, 2
1, 0
0, 2



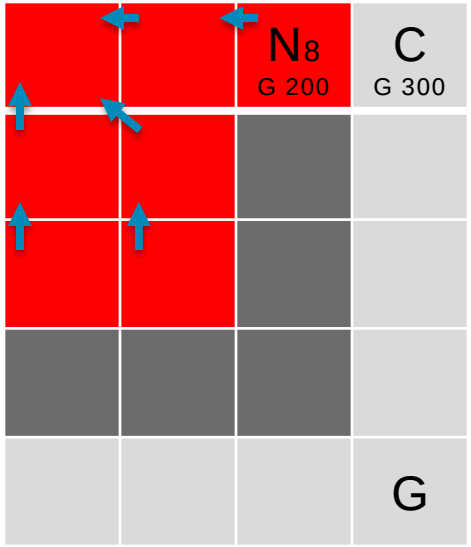

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P

Closed

State
0, 0
1, 1
0, 1
1, 2
1, 0
0, 2
2, 0



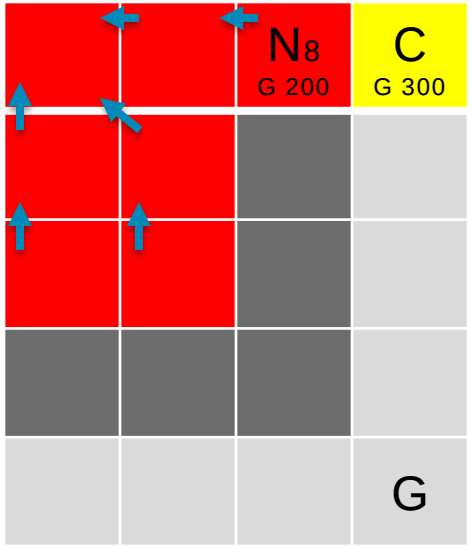

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P

Closed

State
0, 0
1, 1
0, 1
1, 2
1, 0
0, 2
2, 0



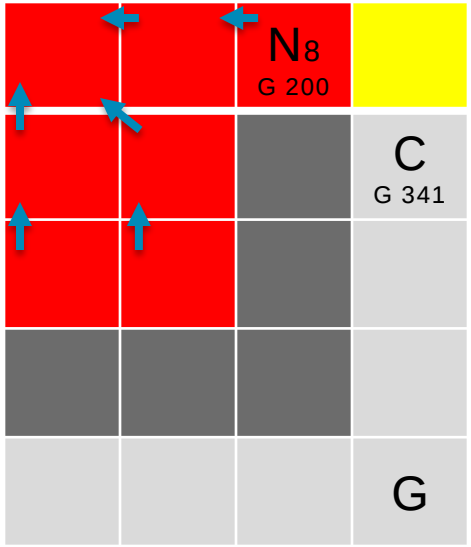

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
9	3,0	300	400	700	1,0	8

Closed

State
0, 0
1, 1
0, 1
1, 2
1, 0
0, 2
2, 0



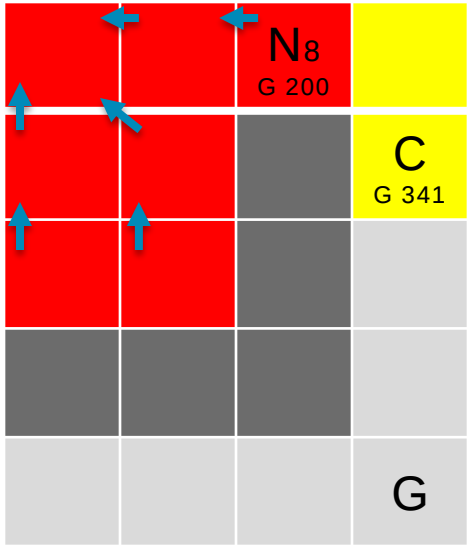

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
9	3,0	300	400	700	1,0	8

Closed

State
0, 0
1, 1
0, 1
1, 2
1, 0
0, 2
2, 0



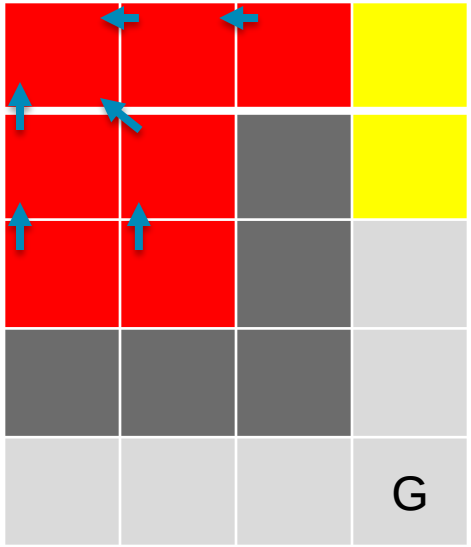

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
9	3,0	300	400	700	1,0	8
10	3,1	341	300	641	1,1	8

Closed

State
0, 0
1, 1
0, 1
1, 2
1, 0
0, 2
2, 0



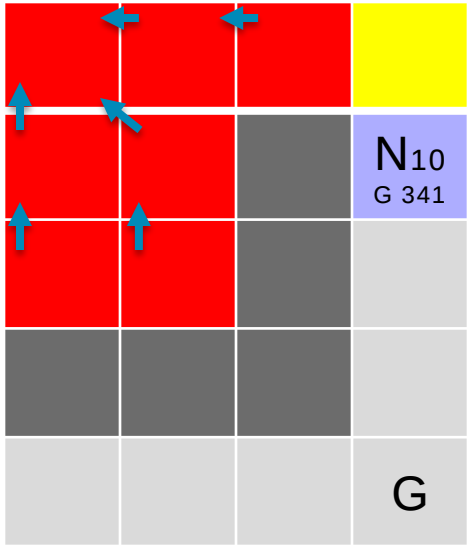

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
9	3,0	300	400	700	1,0	8
10	3,1	341	300	641	1,1	8

Closed

State
0, 0
1, 1
0, 1
1, 2
1, 0
0, 2
2, 0



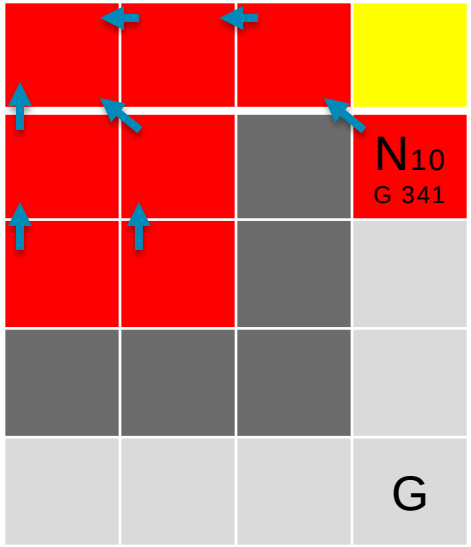

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
9	3,0	300	400	700	1,0	8
10	3,1	341	300	641	1,1	8

Closed

State
0, 0
1, 1
0, 1
1, 2
1, 0
0, 2
2, 0



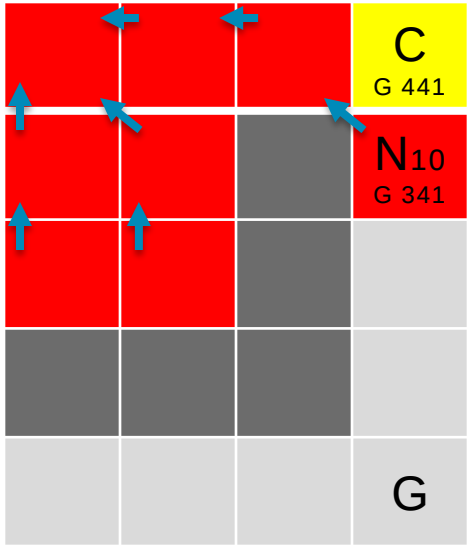

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
9	3,0	300	400	700	1,0	8

Closed

State
0, 0
1, 1
0, 1
1, 2
1, 0
0, 2
2, 0
3, 1



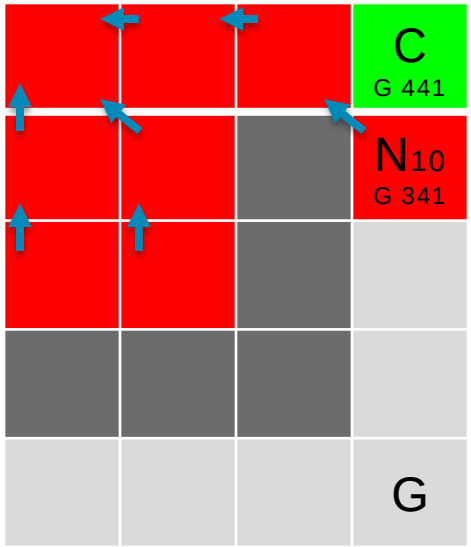

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
9	3,0	300	400	700	1,0	8

Closed

State
0, 0
1, 1
0, 1
1, 2
1, 0
0, 2
2, 0
3, 1



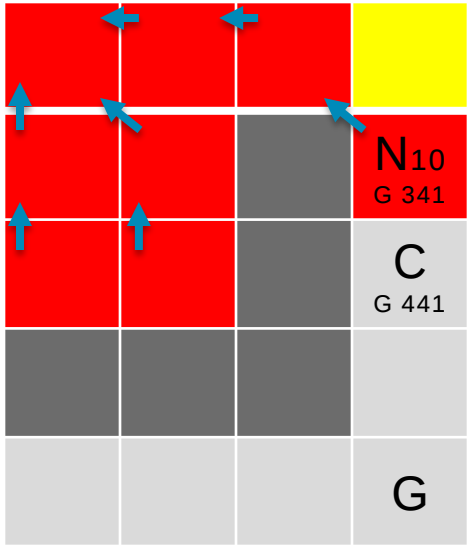

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
9	3,0	300	400	700	1,0	8

Closed

State
0, 0
1, 1
0, 1
1, 2
1, 0
0, 2
2, 0
3, 1



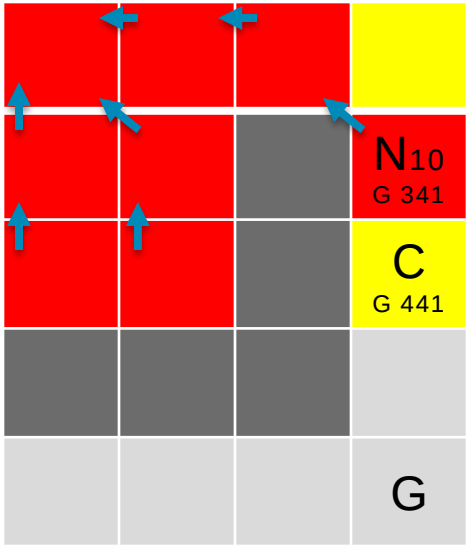

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
9	3,0	300	400	700	1,0	8

Closed

State
0, 0
1, 1
0, 1
1, 2
1, 0
0, 2
2, 0
3, 1



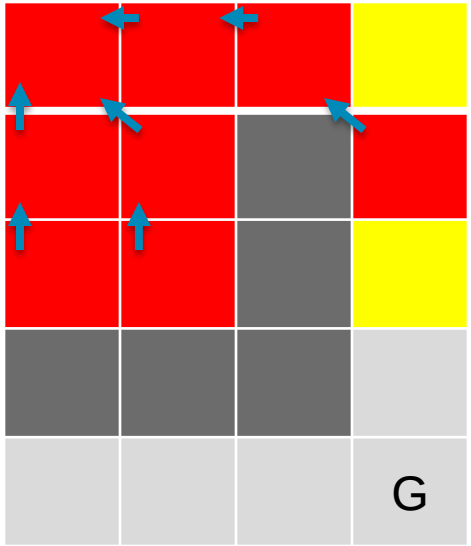

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
9	3,0	300	400	700	1,0	8
11	3,2	441	200	641	0,1	10

Closed

State
0, 0
1, 1
0, 1
1, 2
1, 0
0, 2
2, 0
3, 1



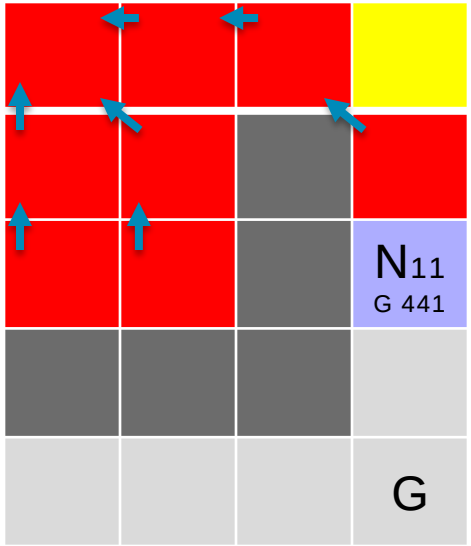

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
9	3,0	300	400	700	1,0	8
11	3,2	441	200	641	0,1	10

Closed

State
0, 0
1, 1
0, 1
1, 2
1, 0
0, 2
2, 0
3, 1



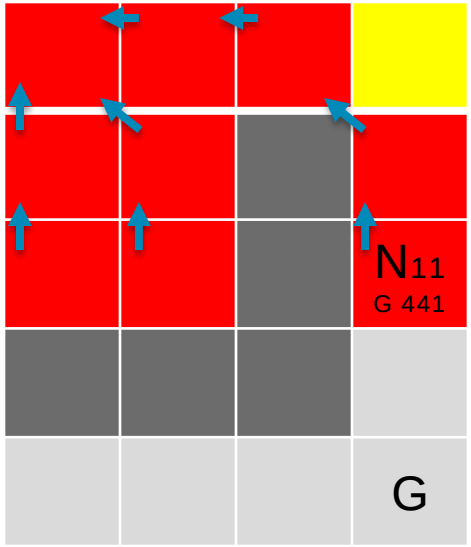

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
9	3,0	300	400	700	1,0	8
11	3,2	441	200	641	0,1	10

Closed

State
0, 0
1, 1
0, 1
1, 2
1, 0
0, 2
2, 0
3, 1



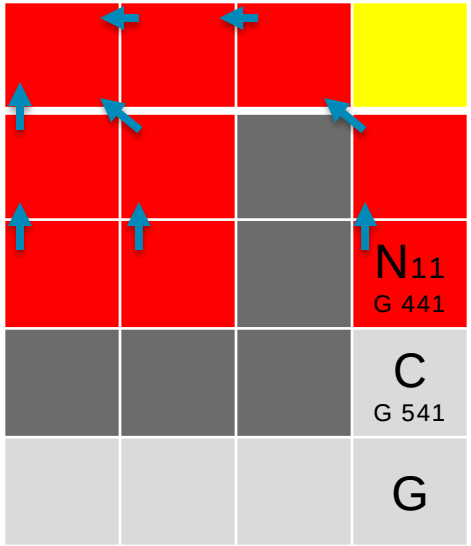

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
9	3,0	300	400	700	1,0	8

Closed

State
0, 0
1, 1
0, 1
1, 2
1, 0
0, 2
2, 0
3, 1
3, 2



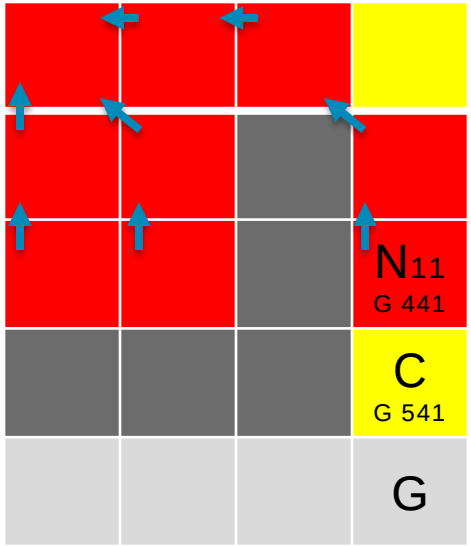

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
9	3,0	300	400	700	1,0	8

Closed

State
0, 0
1, 1
0, 1
1, 2
1, 0
0, 2
2, 0
3, 1
3, 2



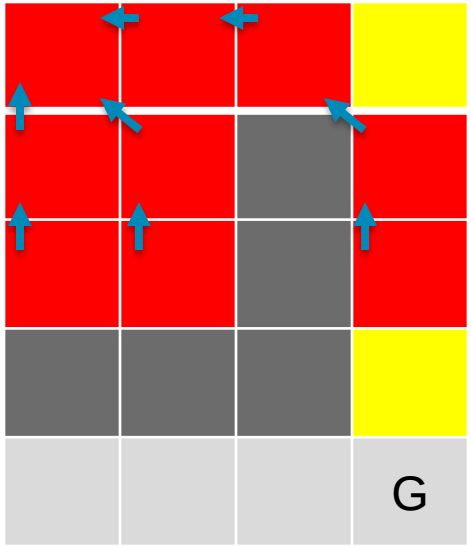

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
9	3,0	300	400	700	1,0	8
12	3,3	541	100	641	0,1	11

Closed

State
0, 0
1, 1
0, 1
1, 2
1, 0
0, 2
2, 0
3, 1
3, 2



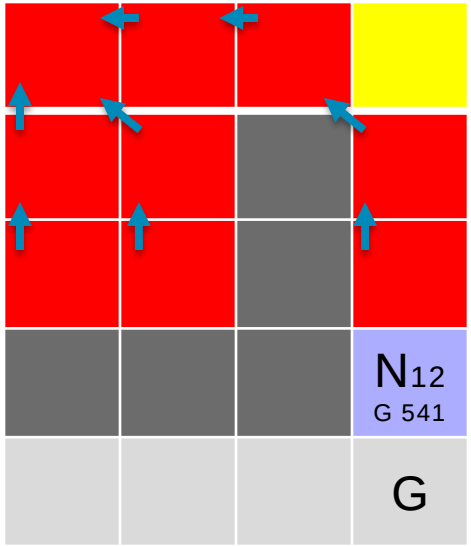

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
9	3,0	300	400	700	1,0	8
12	3,3	541	100	641	0,1	11

Closed

State
0, 0
1, 1
0, 1
1, 2
1, 0
0, 2
2, 0
3, 1
3, 2



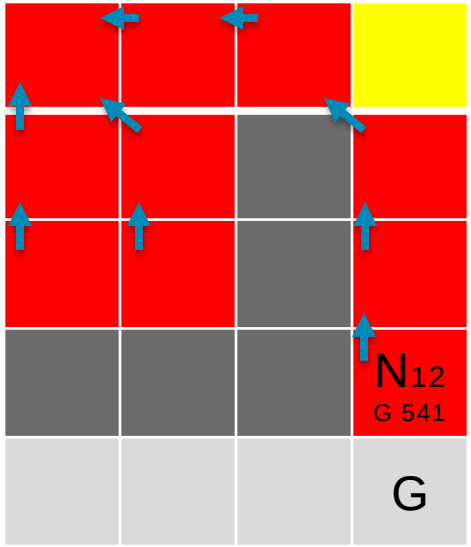

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
9	3,0	300	400	700	1,0	8
12	3,3	541	100	641	0,1	11

Closed

State
0, 0
1, 1
0, 1
1, 2
1, 0
0, 2
2, 0
3, 1
3, 2



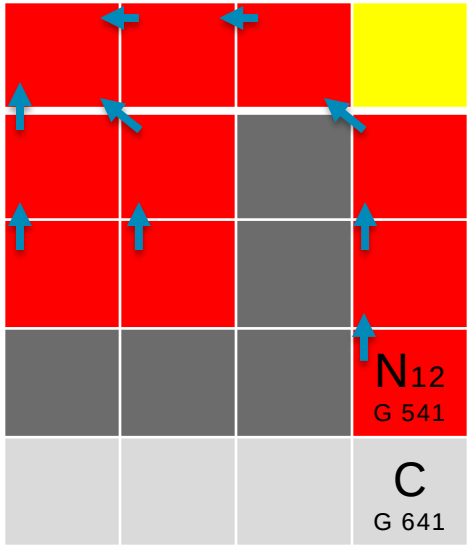

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
9	3,0	300	400	700	1,0	8

Closed

State
0, 0
1, 1
0, 1
1, 2
1, 0
0, 2
2, 0
3, 1
3, 2
3,3



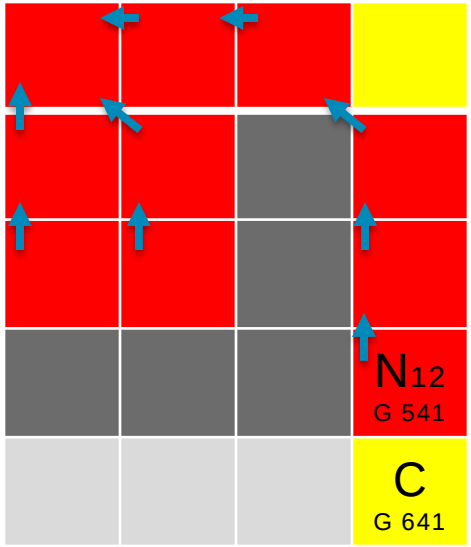

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
9	3,0	300	400	700	1,0	8

Closed

State
0, 0
1, 1
0, 1
1, 2
1, 0
0, 2
2, 0
3, 1
3, 2
3,3



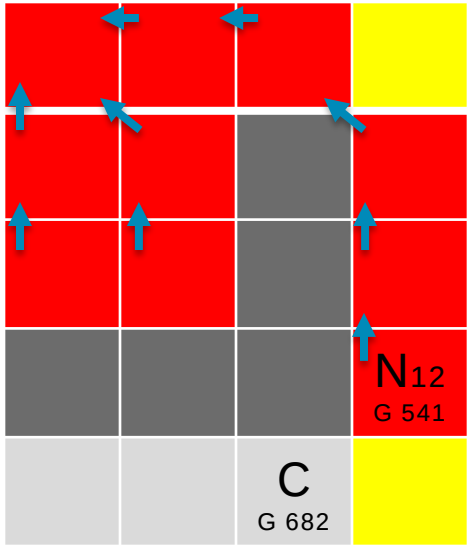

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
9	3,0	300	400	700	1,0	8
13	3,4	641	0	641	0,1	12

Closed

State
0, 0
1, 1
0, 1
1, 2
1, 0
0, 2
2, 0
3, 1
3, 2
3,3



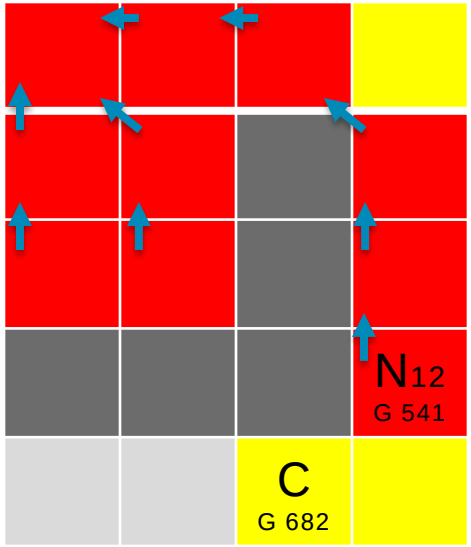

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
9	3,0	300	400	700	1,0	8
13	3,4	641	0	641	0,1	12

Closed

State
0, 0
1, 1
0, 1
1, 2
1, 0
0, 2
2, 0
3, 1
3, 2
3,3



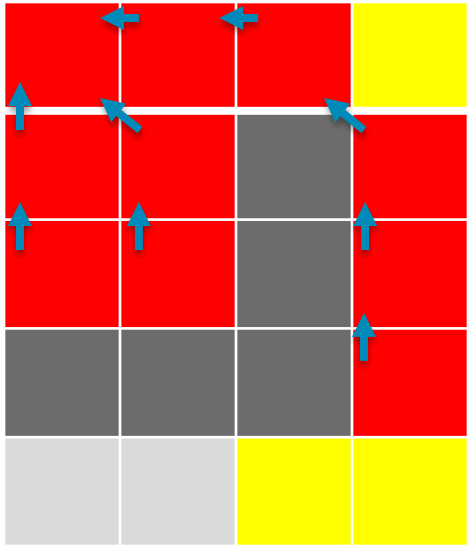

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
9	3,0	300	400	700	1,0	8
13	3,4	641	0	641	0,1	12
14	2,4	682	100	682	-1,1	12

Closed

State
0, 0
1, 1
0, 1
1, 2
1, 0
0, 2
2, 0
3, 1
3, 2
3,3



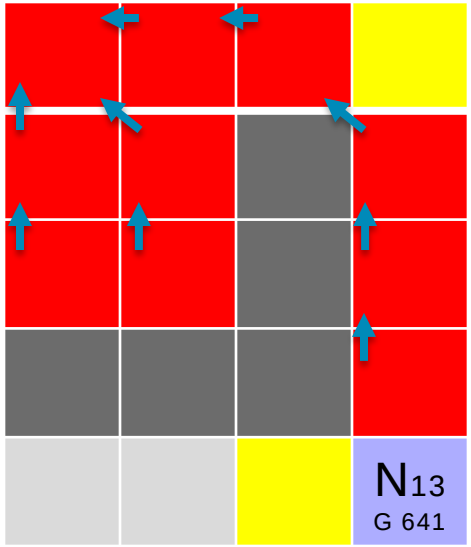
```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
9	3,0	300	400	700	1,0	8
13	3,4	641	0	641	0,1	12
14	2,4	682	100	682	-1,1	12

Closed

State
0, 0
1, 1
0, 1
1, 2
1, 0
0, 2
2, 0
3, 1
3, 2
3,3



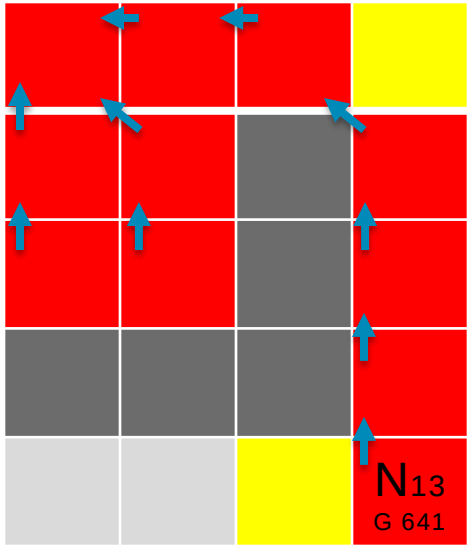

```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
9	3,0	300	400	700	1,0	8
13	3,4	641	0	641	0,1	12
14	2,4	682	100	682	-1,1	12

Closed

State
0, 0
1, 1
0, 1
1, 2
1, 0
0, 2
2, 0
3, 1
3, 2
3,3




```
1.  while (!open.empty)
2.      node = remove min f from open
3.      if (node is goal) return path
4.      add node to closed
5.      for c in expand(node)
6.          if c in closed: continue
7.          if c in open with <= g: continue
8.          else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
9	3,0	300	400	700	1,0	8
14	2,4	682	100	682	-1,1	12

Goal Expanded!
Reconstruct Optimal Path via Parents

Closed

State
0, 0
1, 1
0, 1
1, 2
1, 0
0, 2
2, 0
3, 1
3, 2
3,3