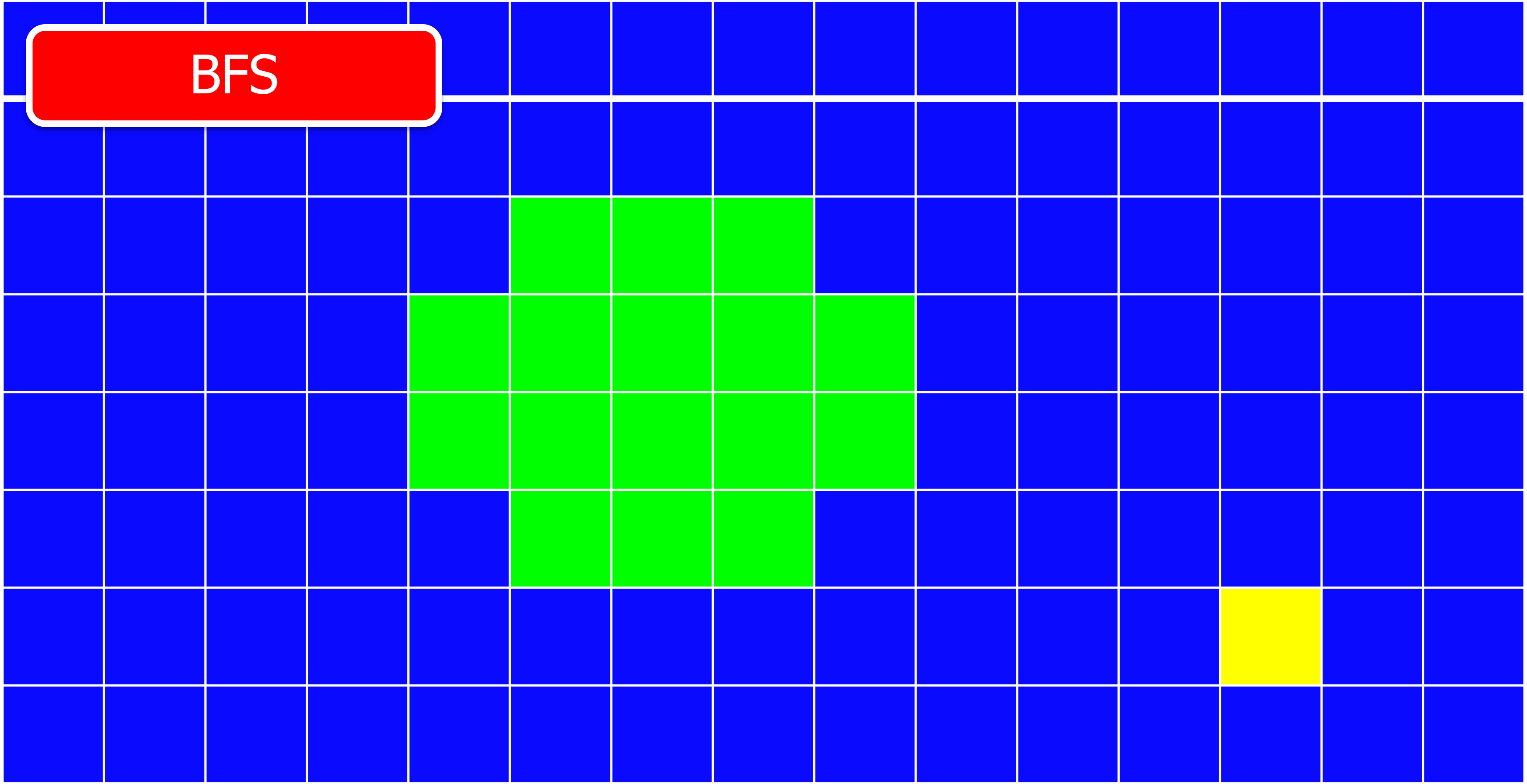
COMP 3200
Artificial Intelligence

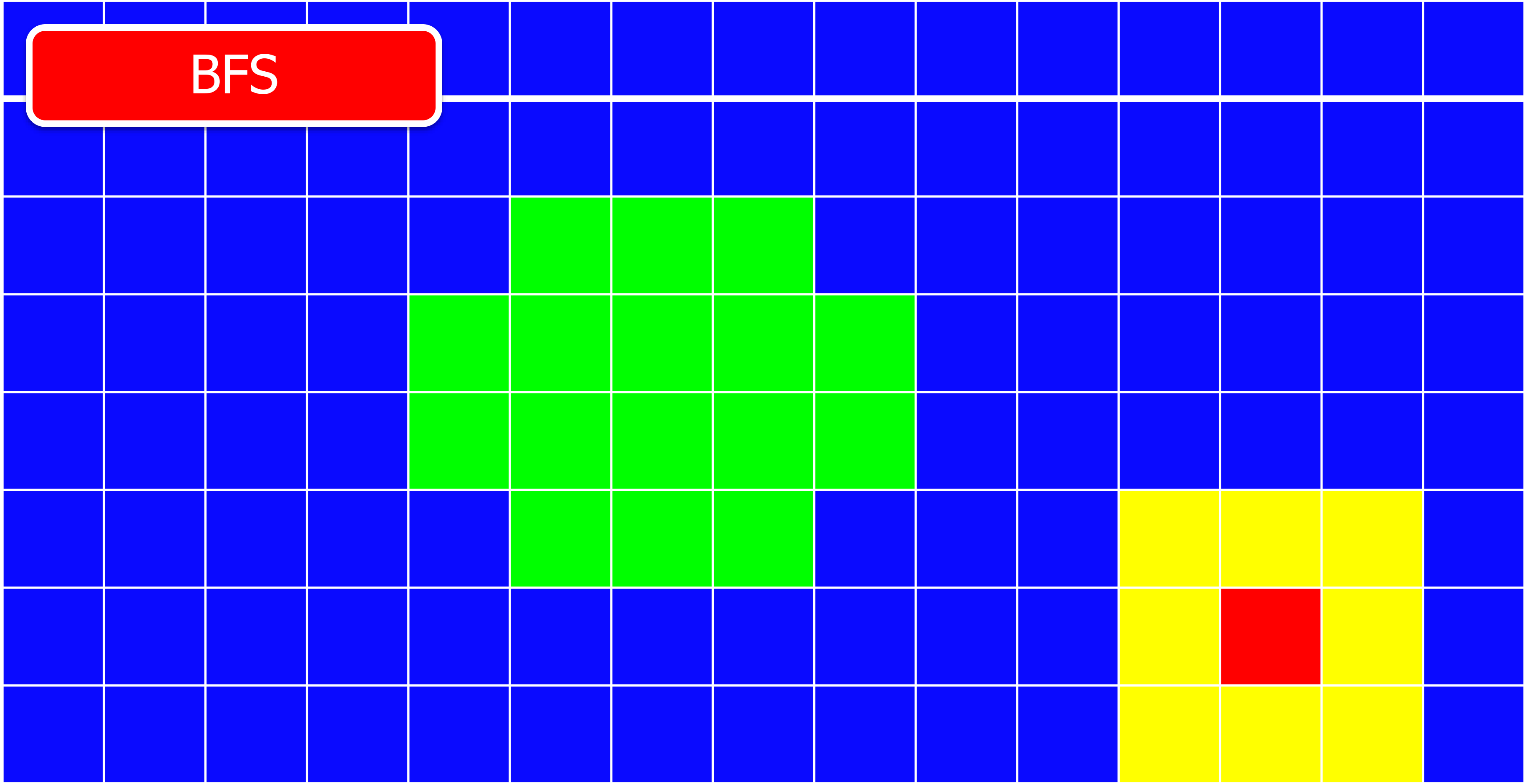
Lecture 5
Heuristic Search
A* Search Algorithm

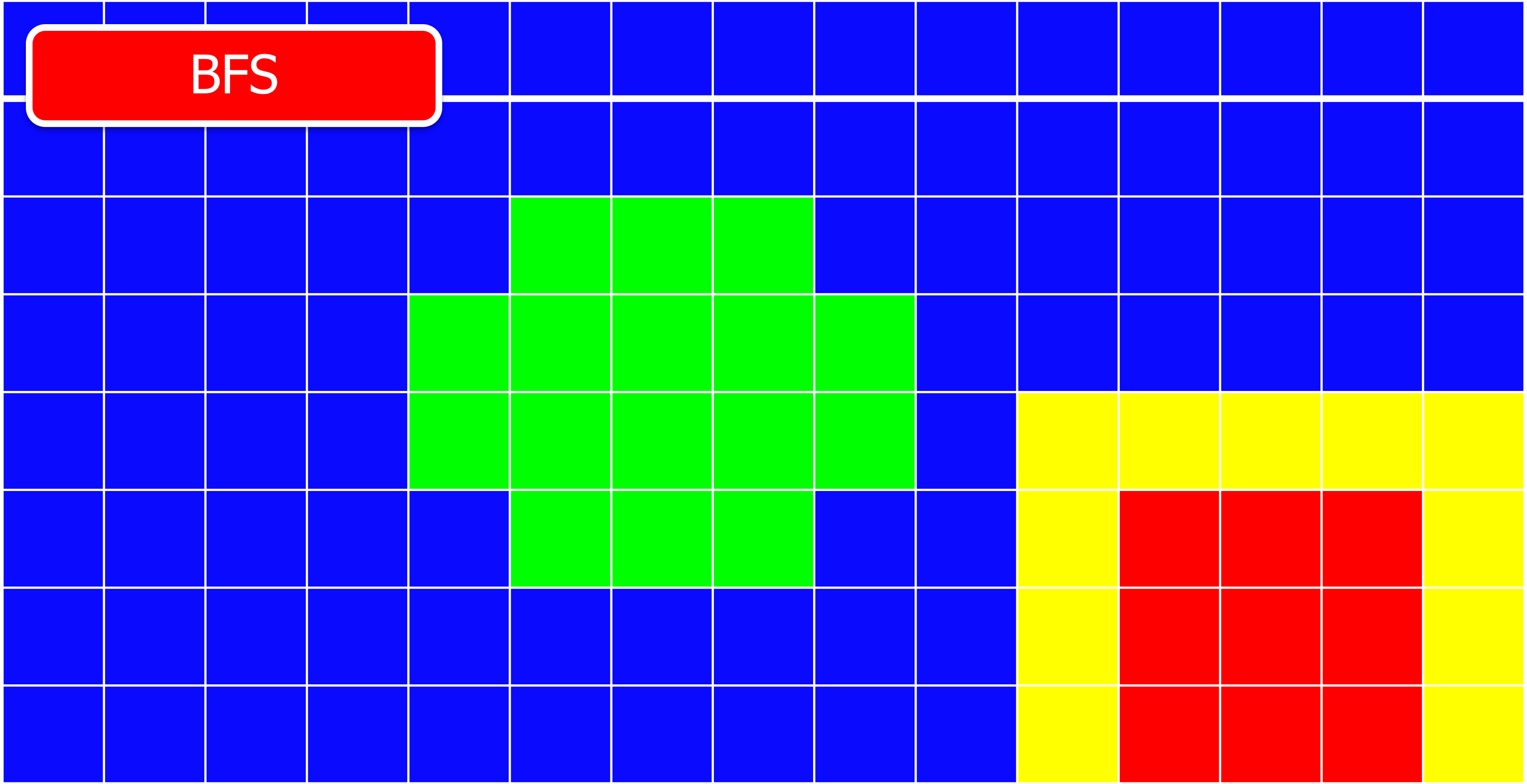
Search Demo

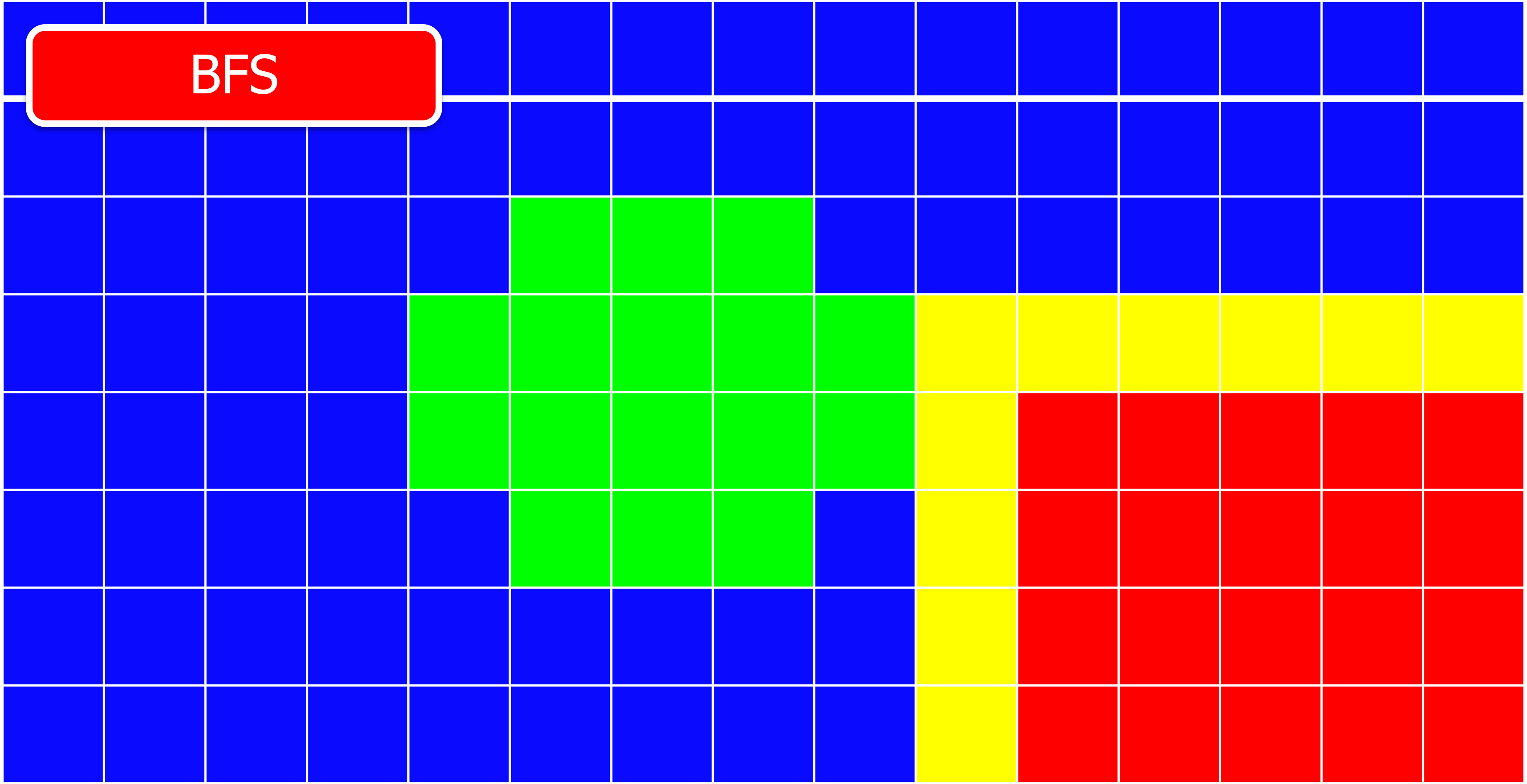
Informed (Heuristic) Search

- An informed search strategy uses problem-specific knowledge beyond just the problem description itself
- Uses guesses (heuristics) to guide search toward the direction of the goal
- Reduce total nodes searched (ideally)
- Speeds up search times to goal (ideally)

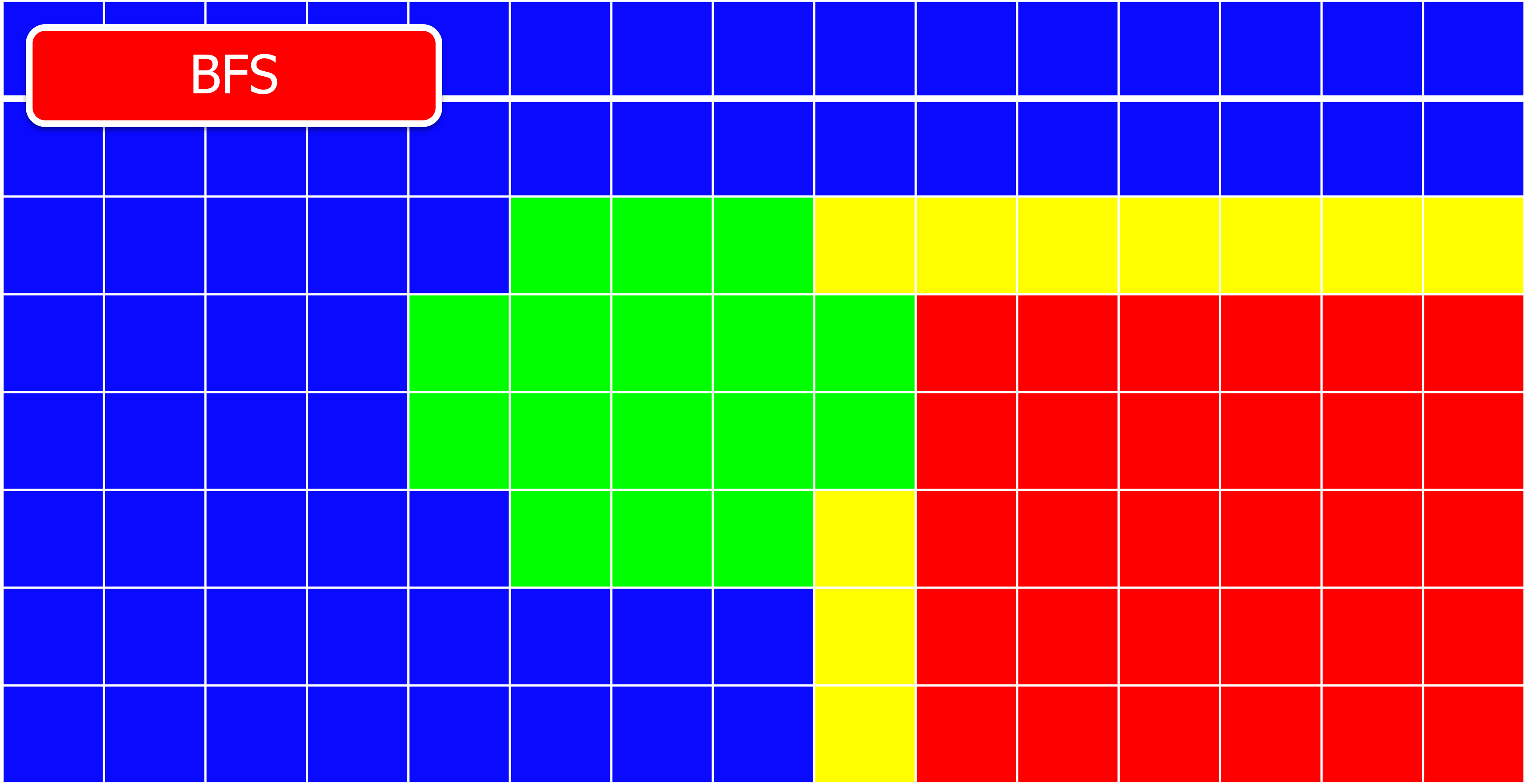

BFS


BFS

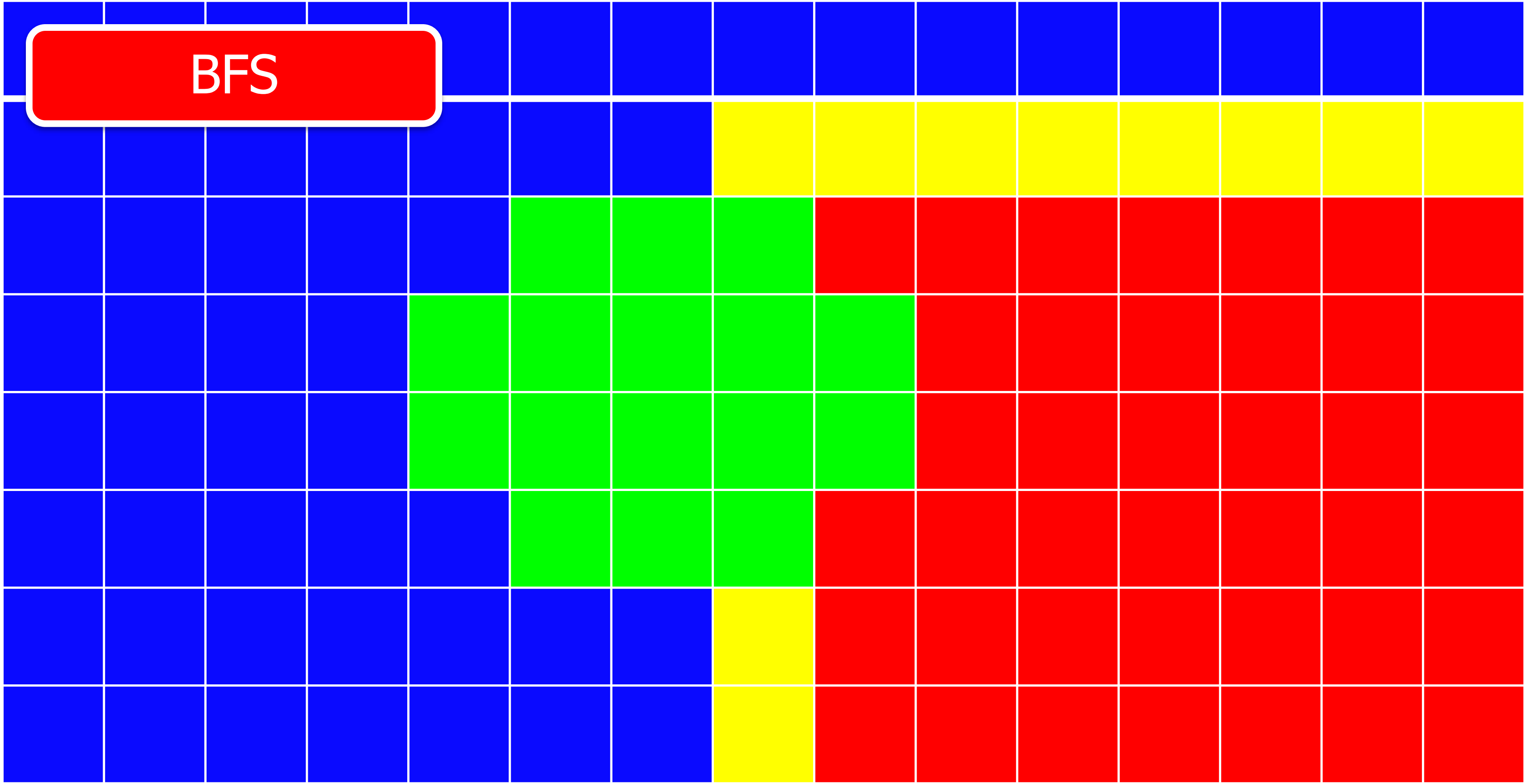

BFS


BFS

BFS



BFS



Best-First Search (BeFS)

- Instance of general tree search
- Select a node for expansion based on an evaluation function = f(n)
- Select the node with minimum value of f(n), measures distance to goal
- "Best First" is slightly misleading
  - Knowing the true best node = go straight to goal
  - More like "Best Guess First"

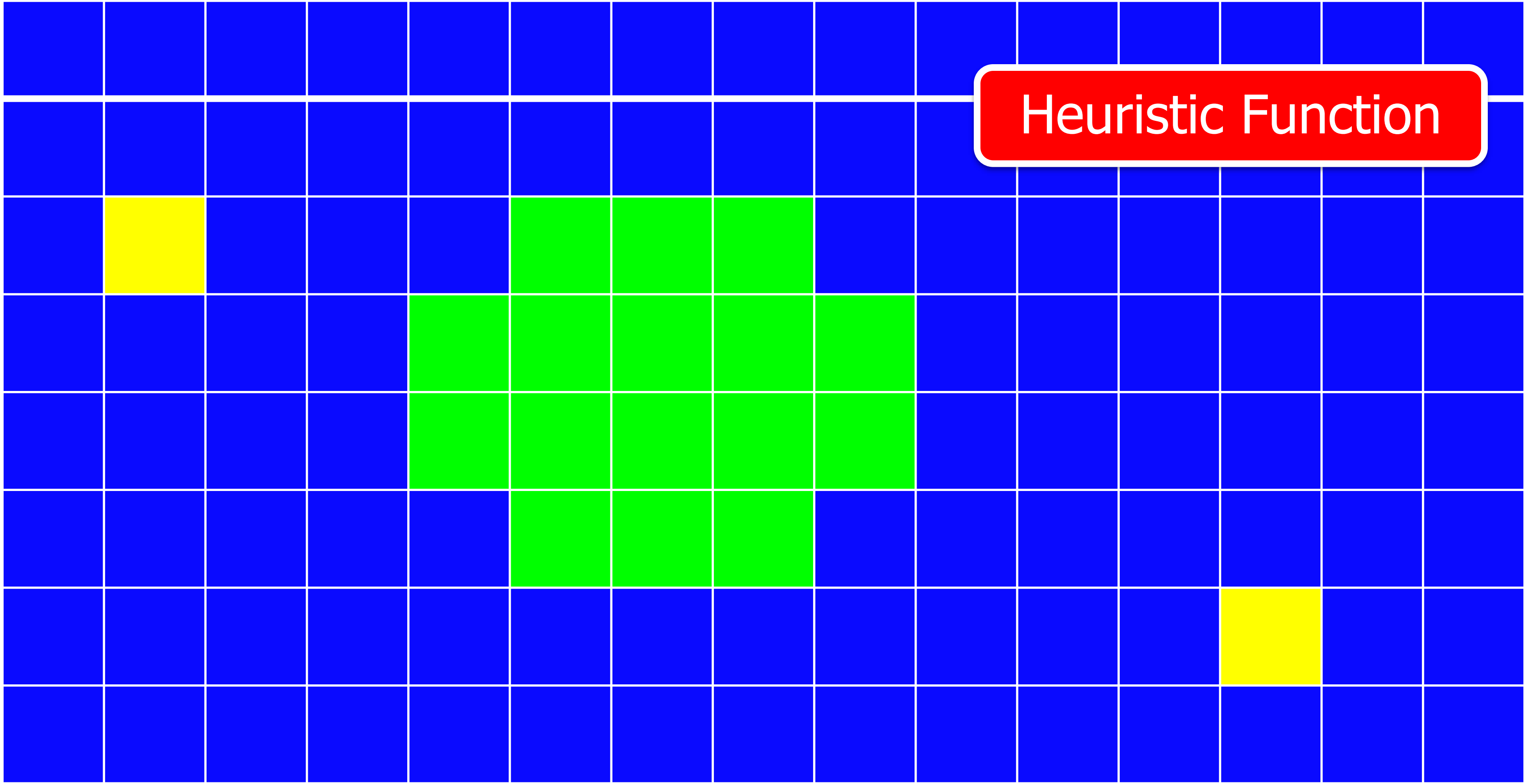
Best-First Search (BeFS)

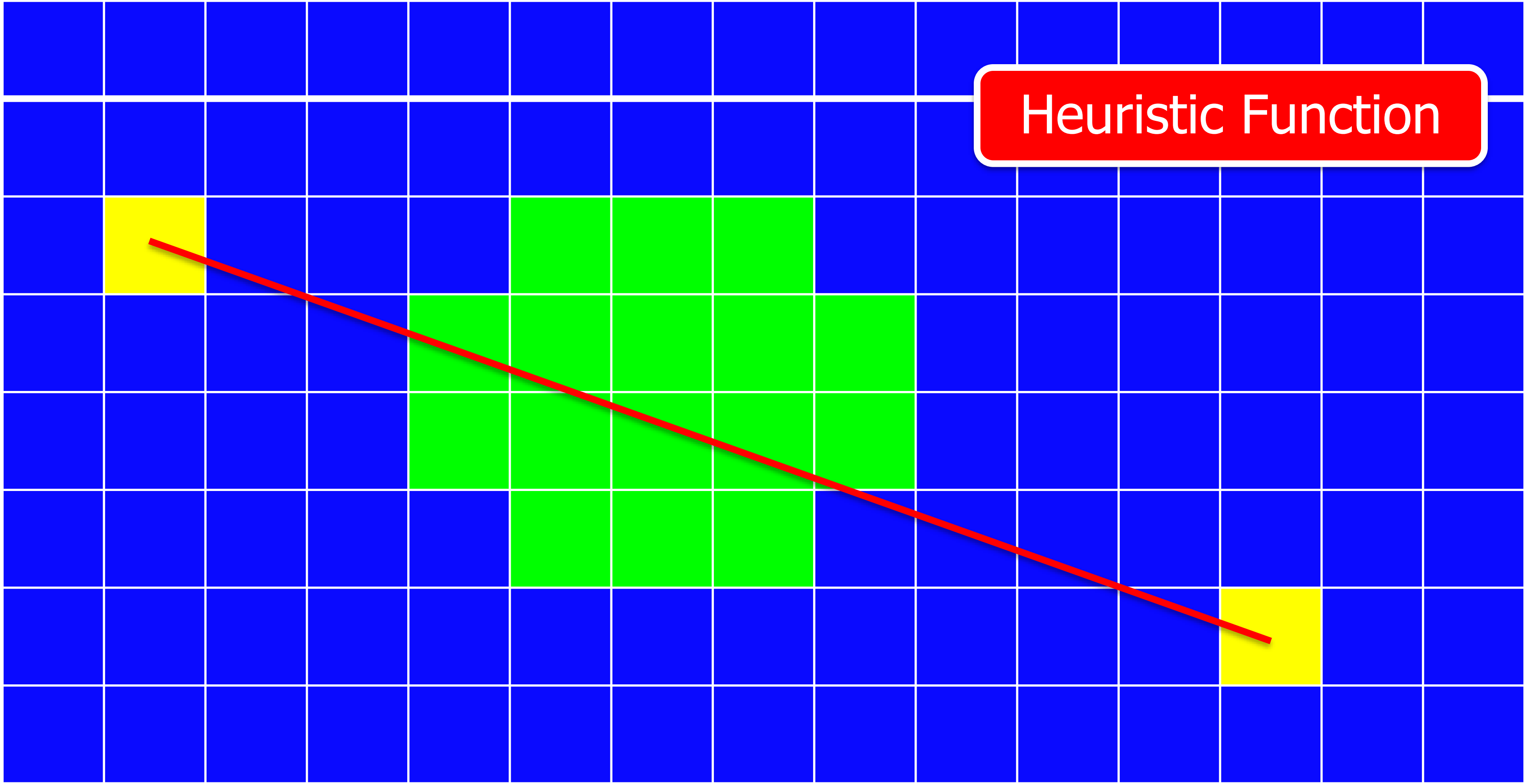
- BeFS can be implemented with general Tree-Search by using a priority queue for the open list, sorted on f(n)
- Whole family of BeFS algorithms
  - Have different evaluation functions
  - Most have a heuristic function h(n)

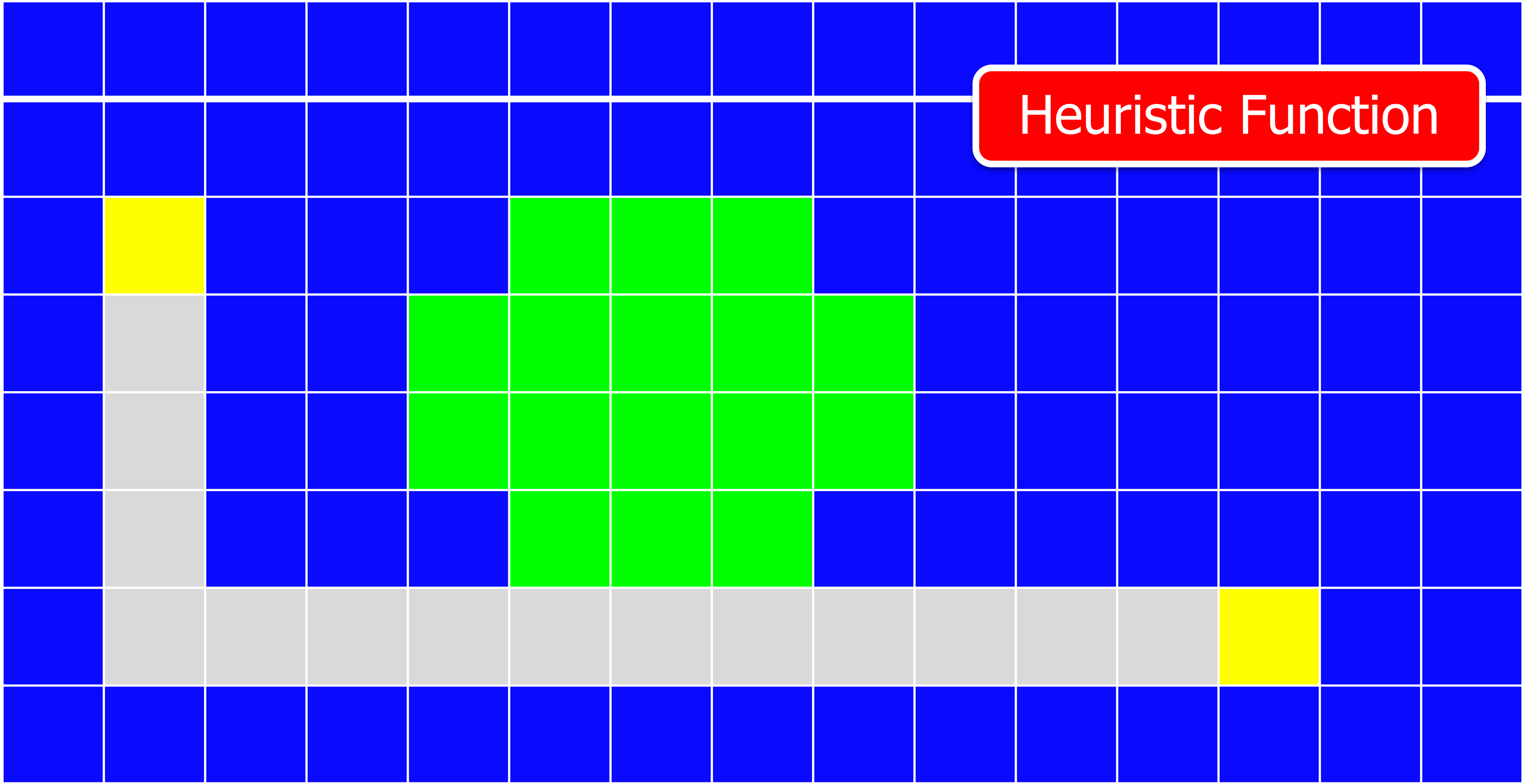
Heuristic Function h(n)

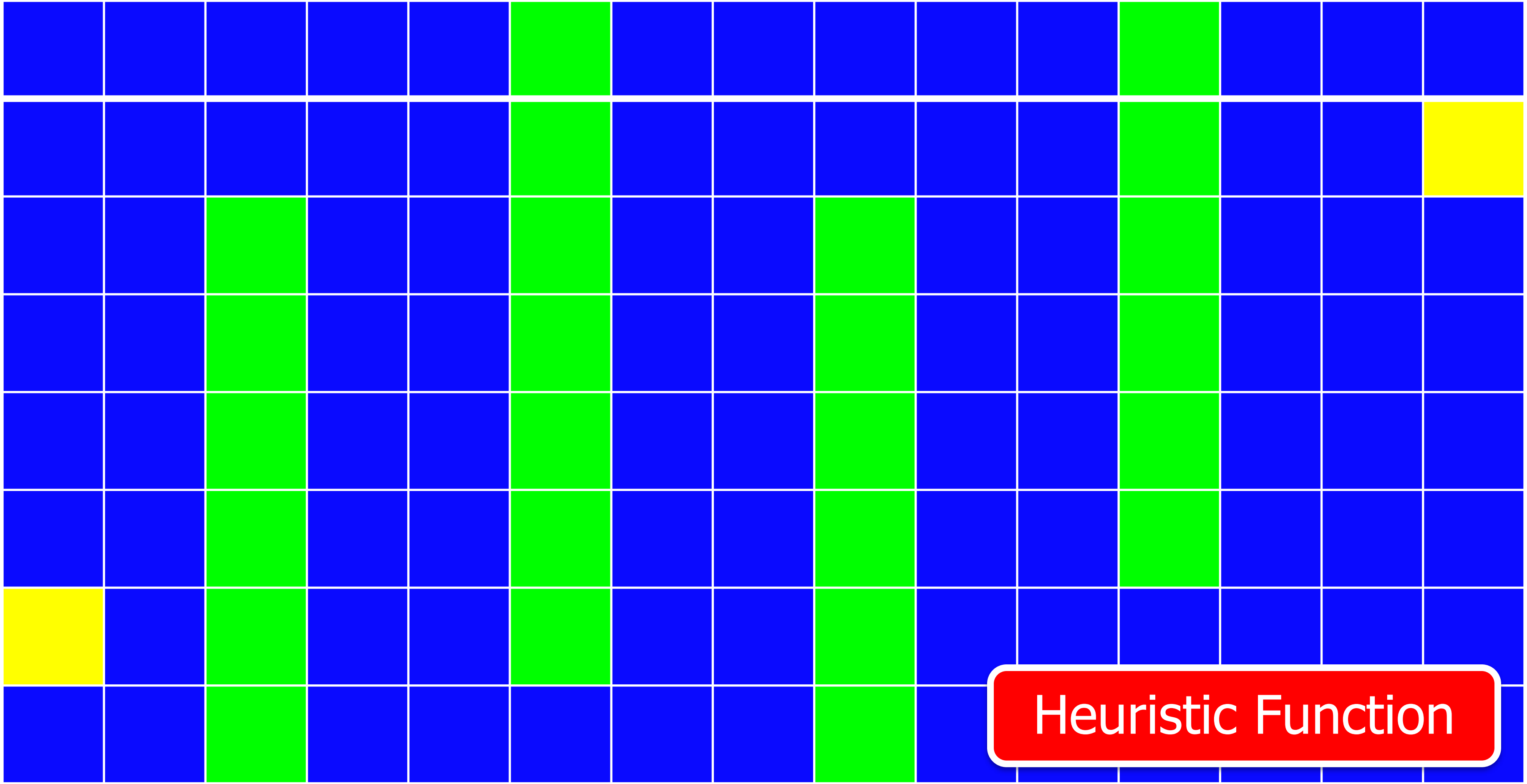
- Estimated cost of the optimal path from node n to a goal node
- Heuristic functions are the most common way that additional knowledge of a problem is given to the search algorithm
- If n is a goal node, $h(n) = 0$
- Perfect heuristic = $h^*(n)$

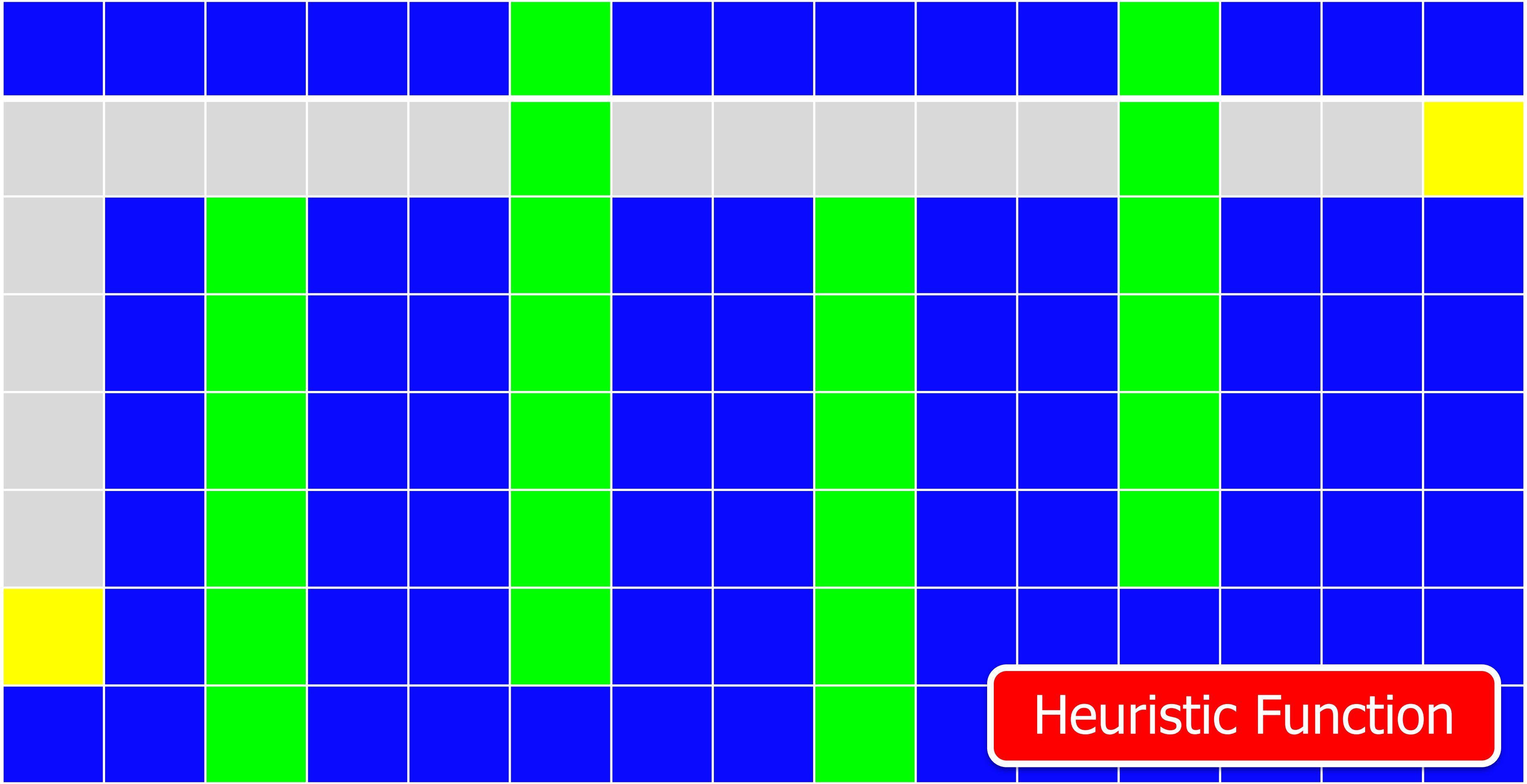
Heuristic Function




Heuristic Function


Heuristic Function


Heuristic Function

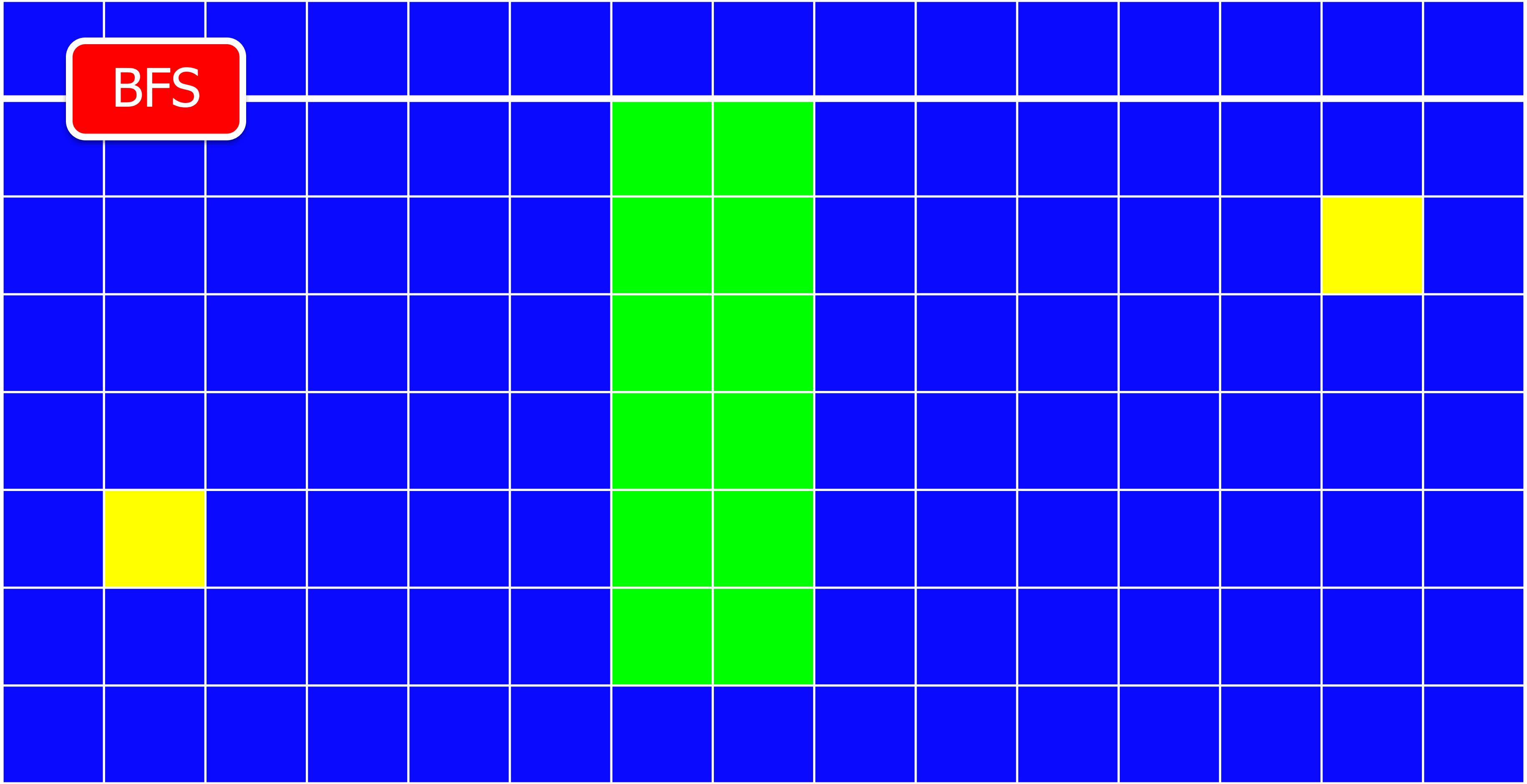

Heuristic Function

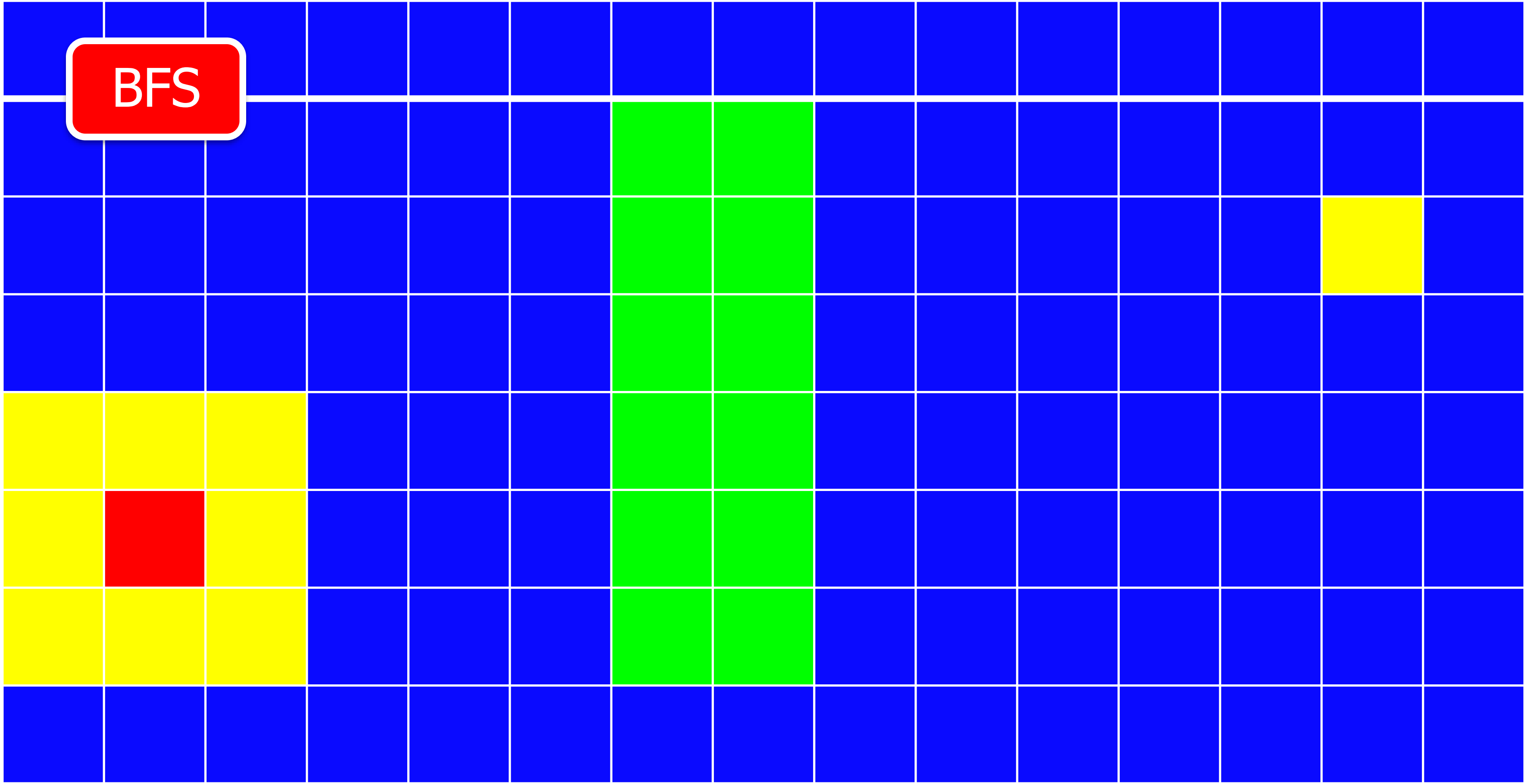
Greedy Best-First Search

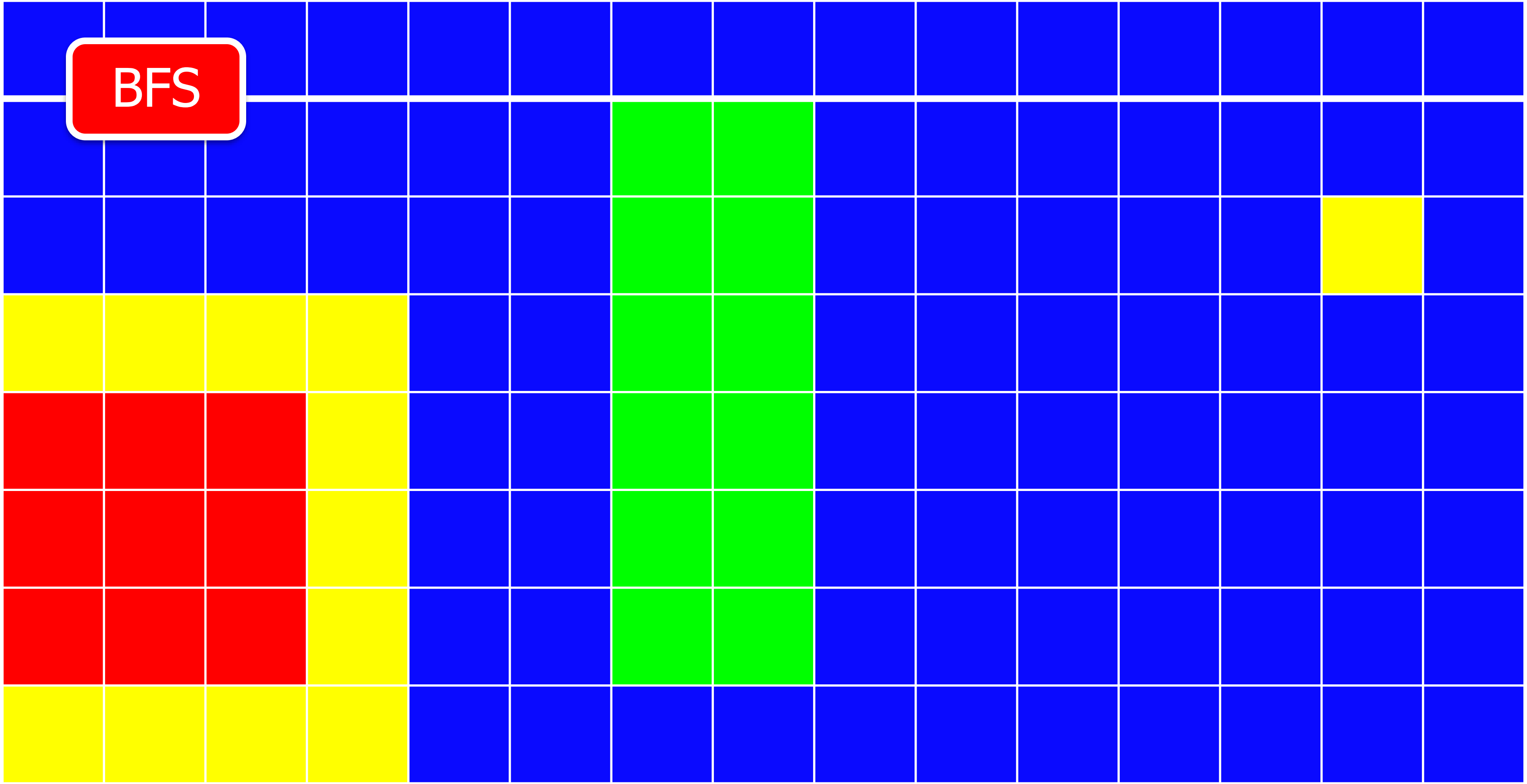
- Expands the node on the open list that it thinks is closest to the goal node
- This will hopefully lead us to the goal
- Thus, for GBeFS: $f(n) = h(n)$
- Resembles DFS
  - Tries a single path all the way to a goal
  - Backs up when it hits a ‘dead-end’

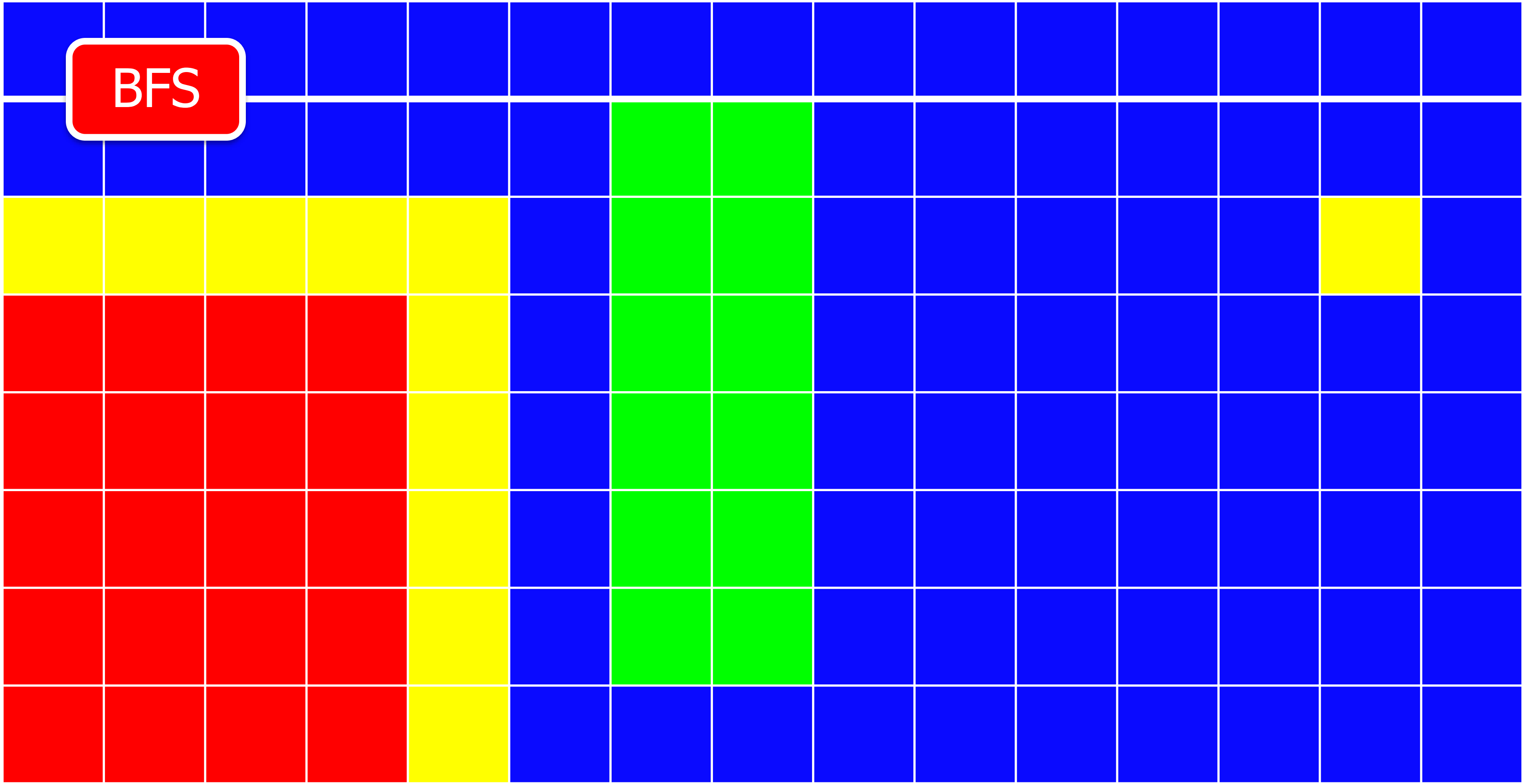
GeBFS Tree-Search

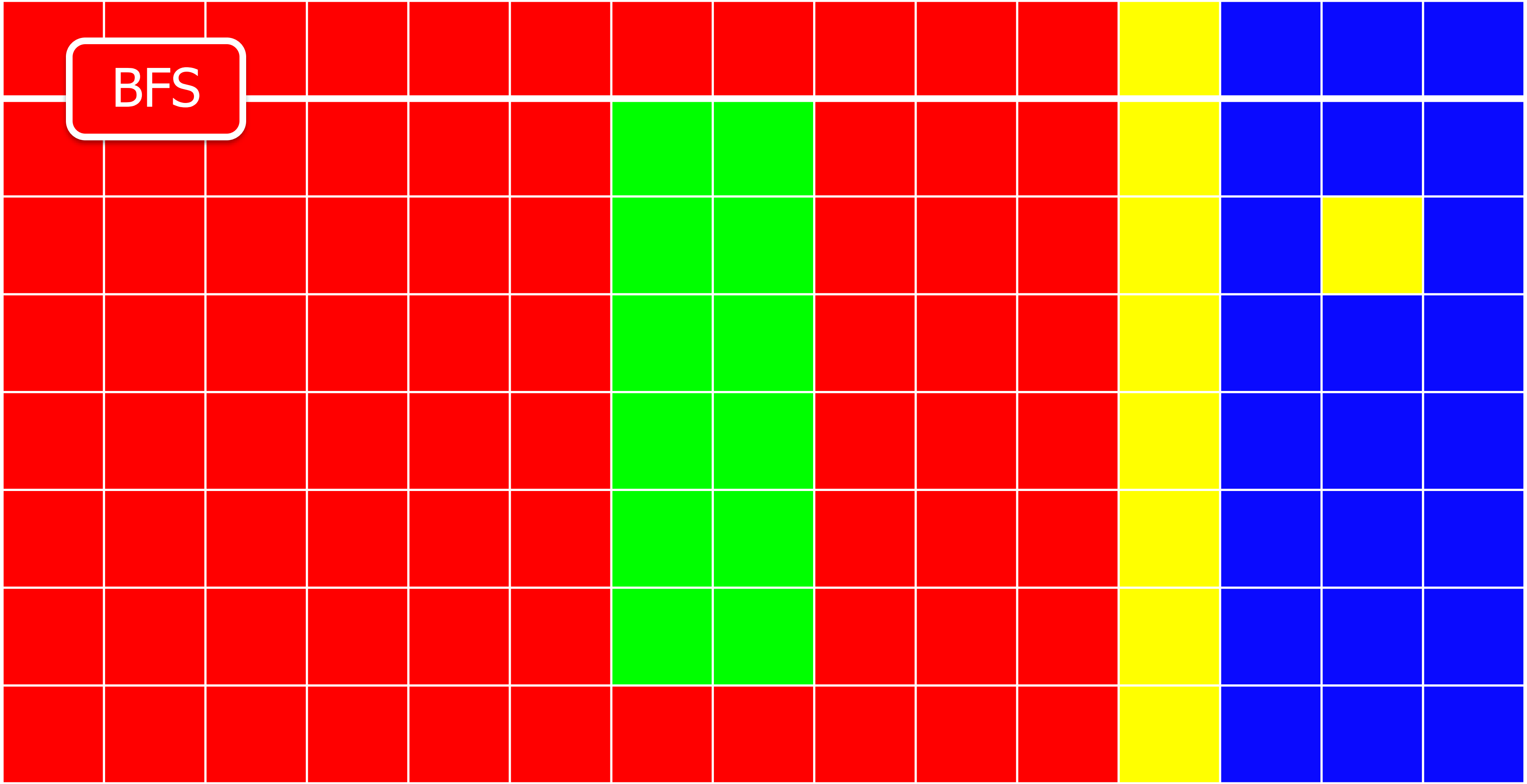
```
Function GeBFS(problem, h)
  open = PQ{f=h,  Node(problem.init_state)}
  while (true)
    if (open.empty) return fail
    node = pop_min_f(open)
    if (node.state is goal) return solution
    for c in Expand(node, problem)
      open.add(c)
```

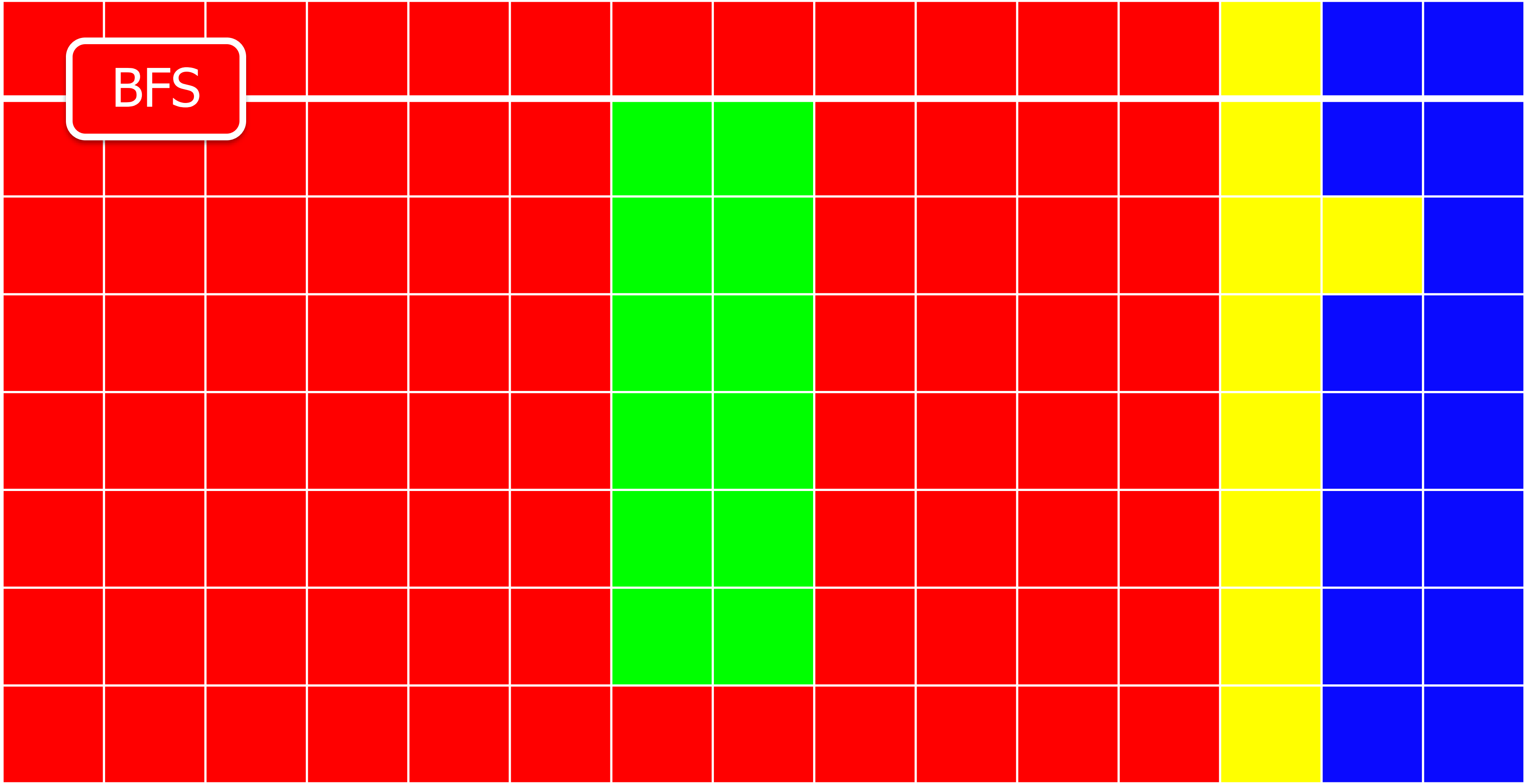

BFS

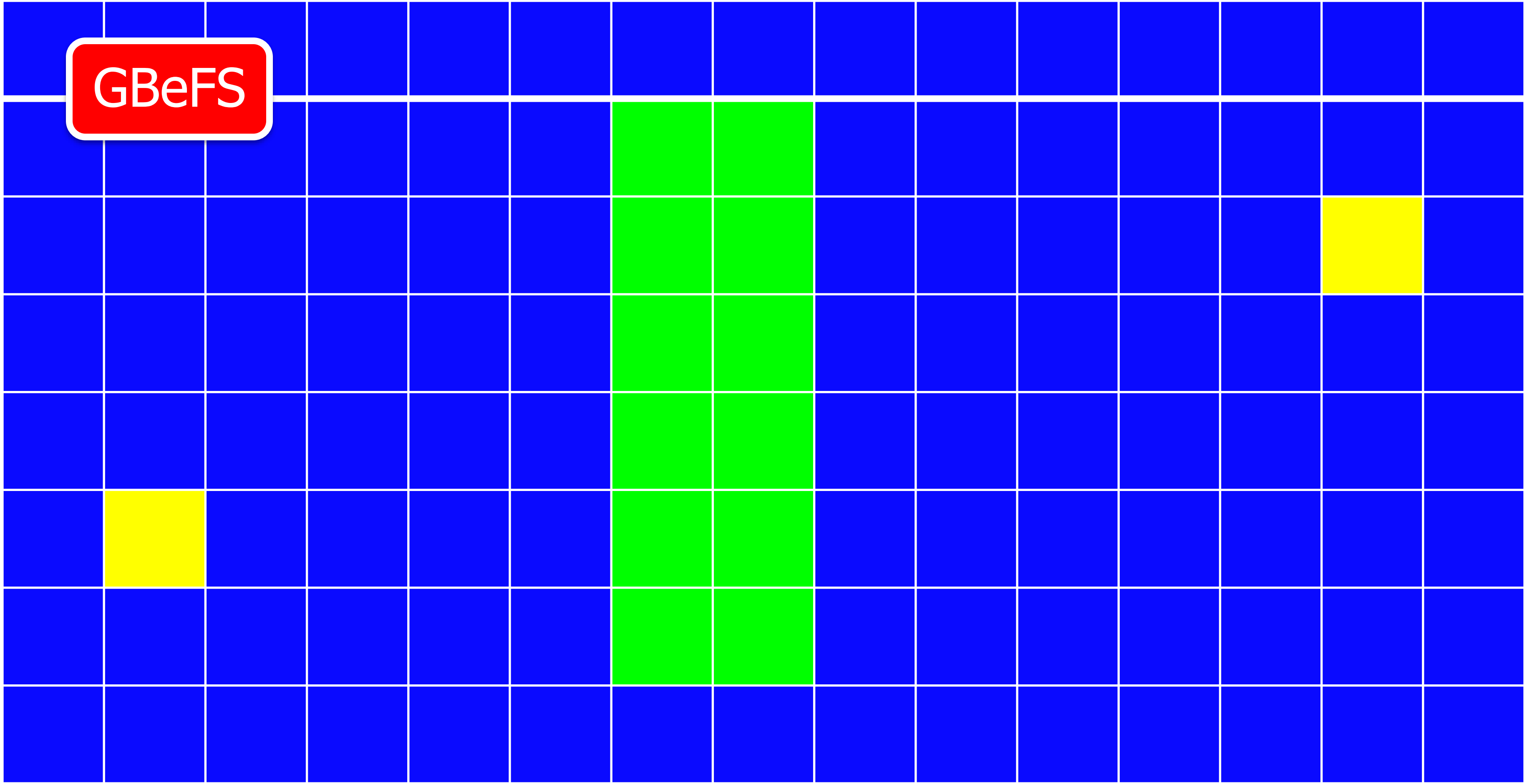

BFS

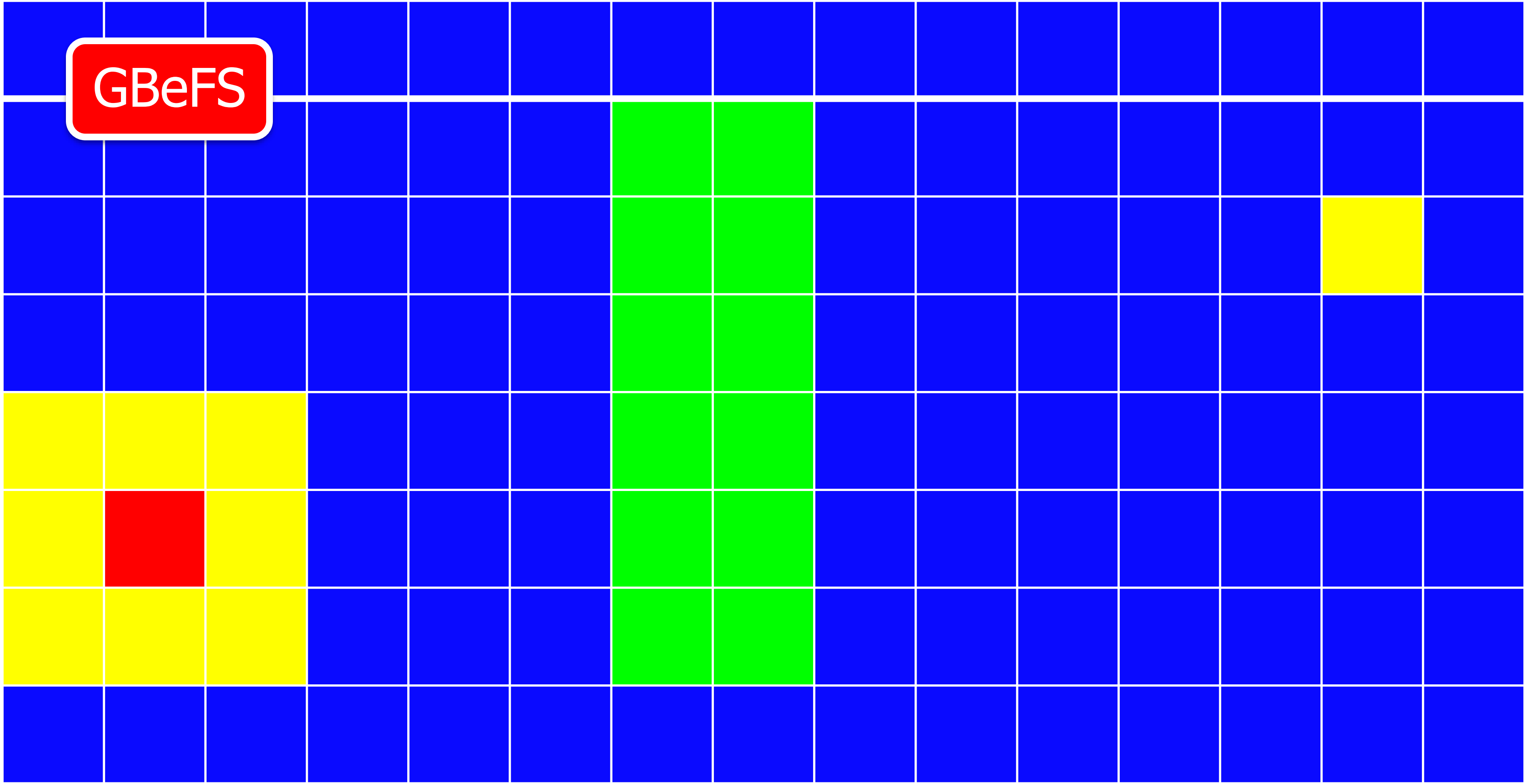

BFS

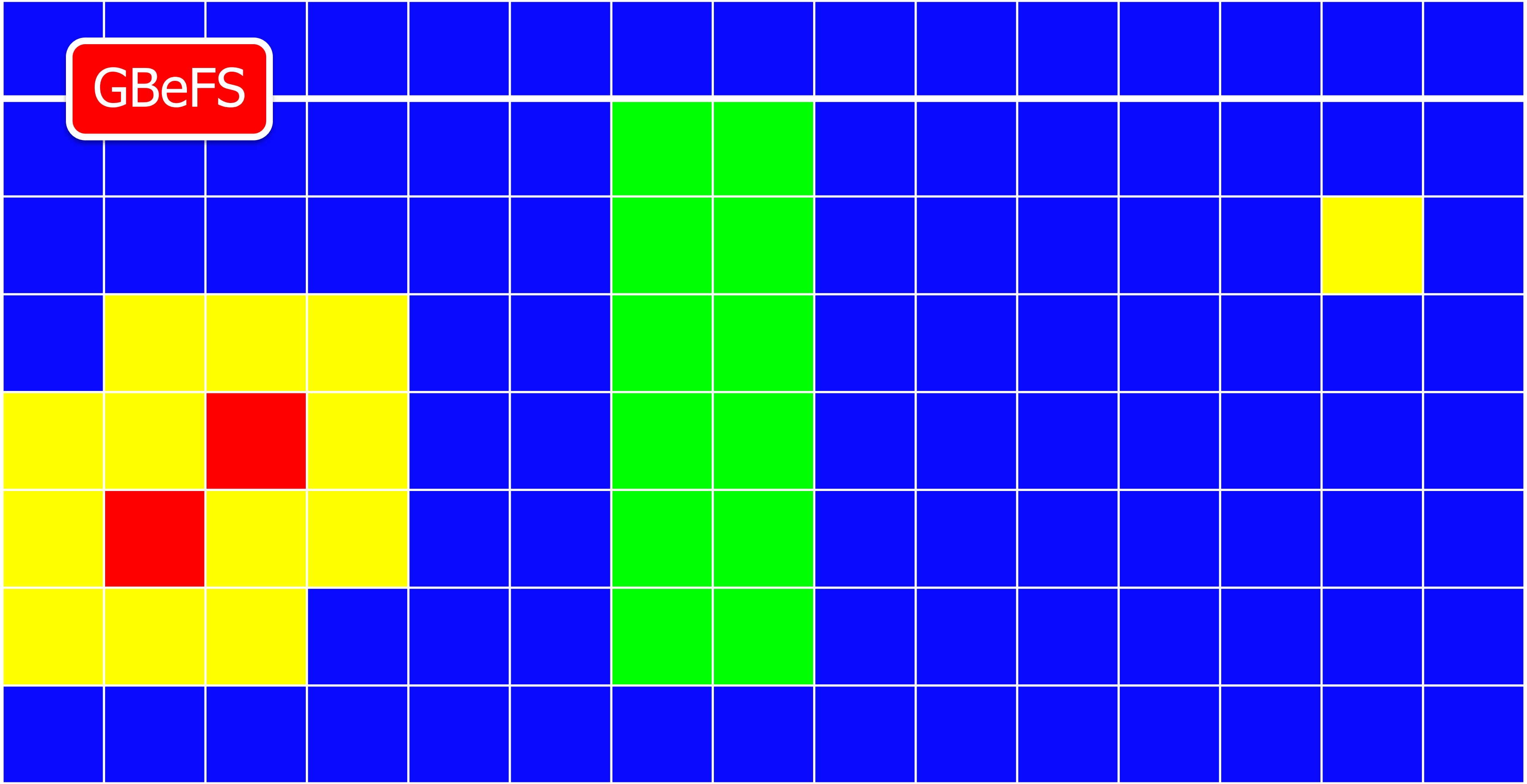

BFS

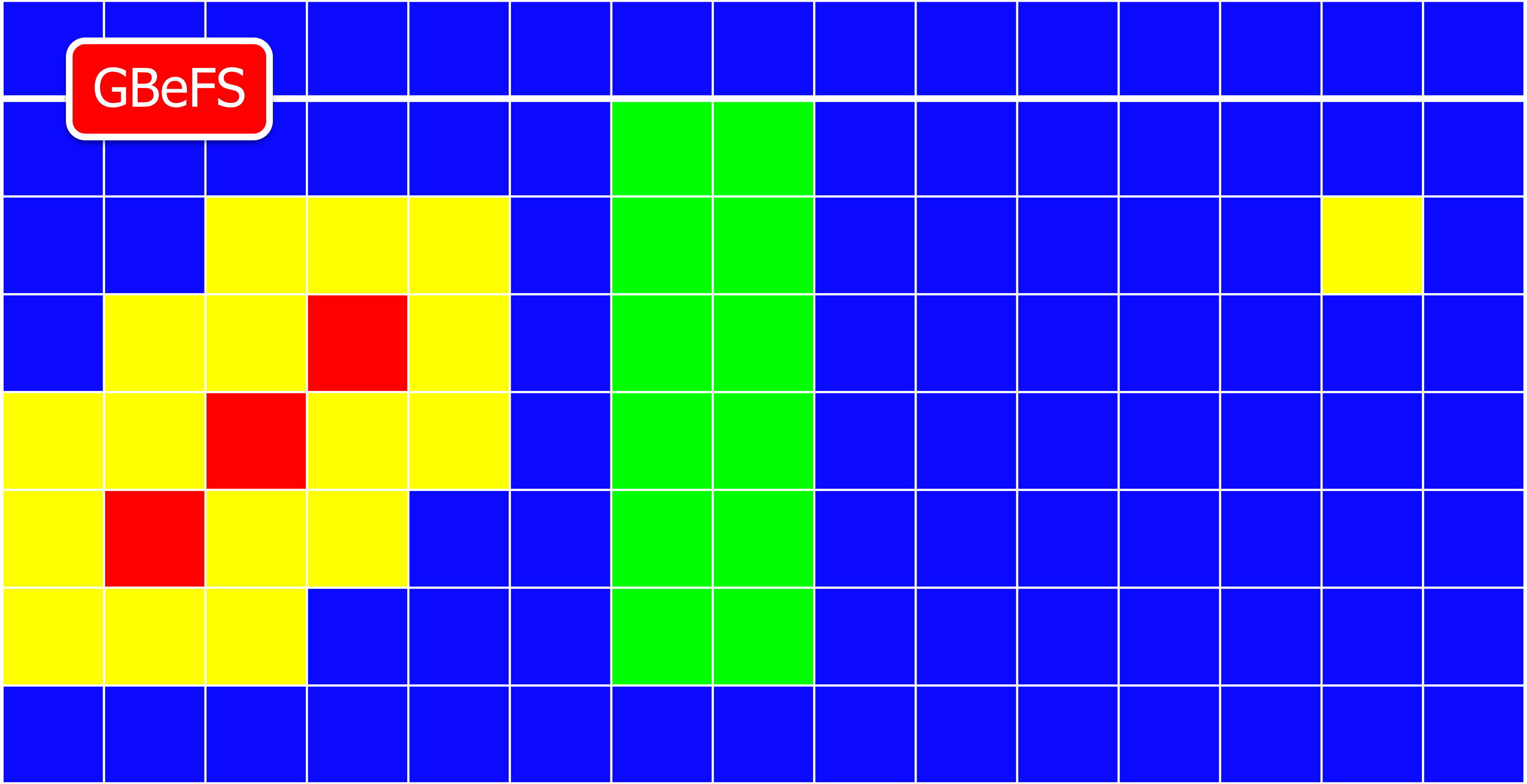

BFS

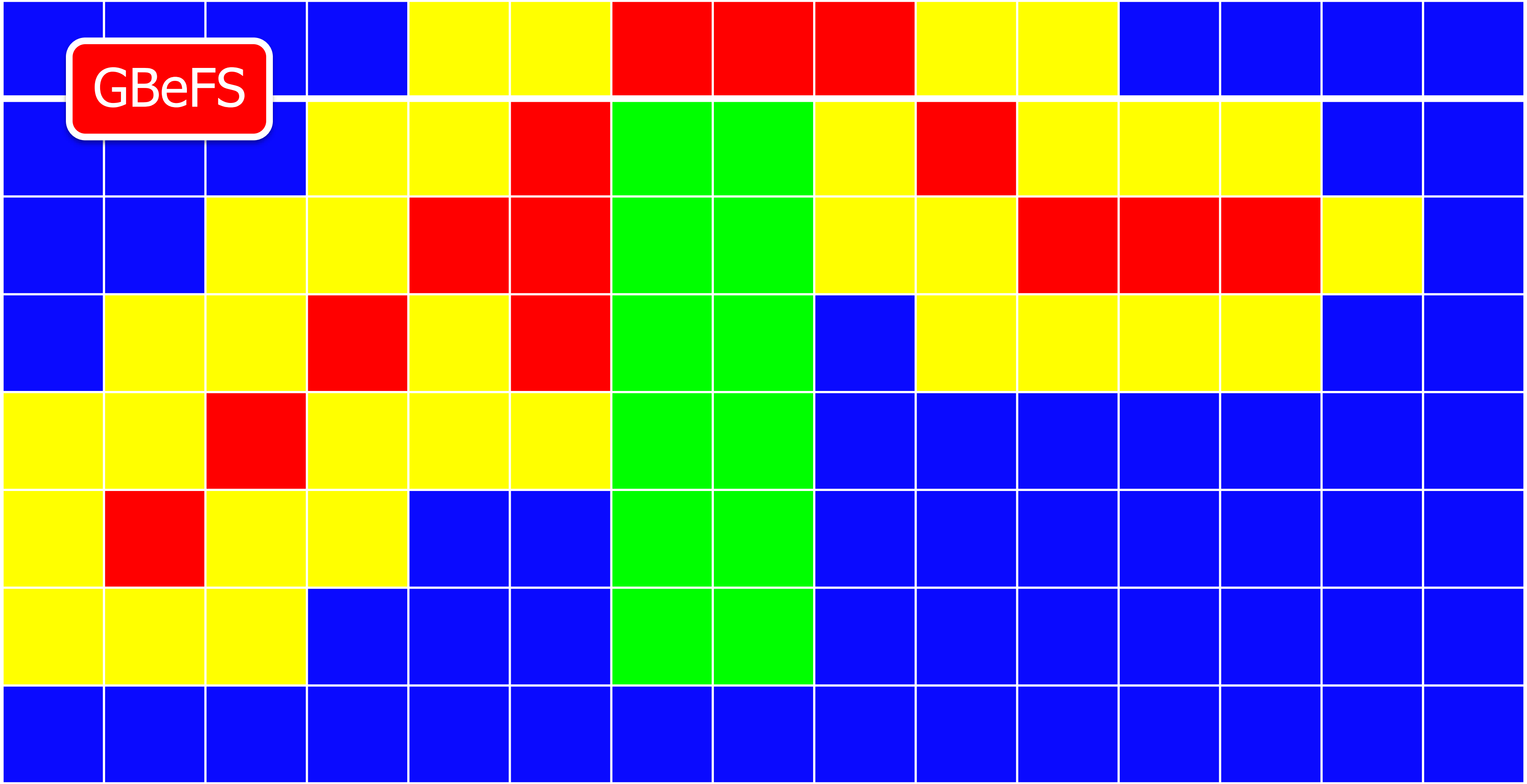

BFS

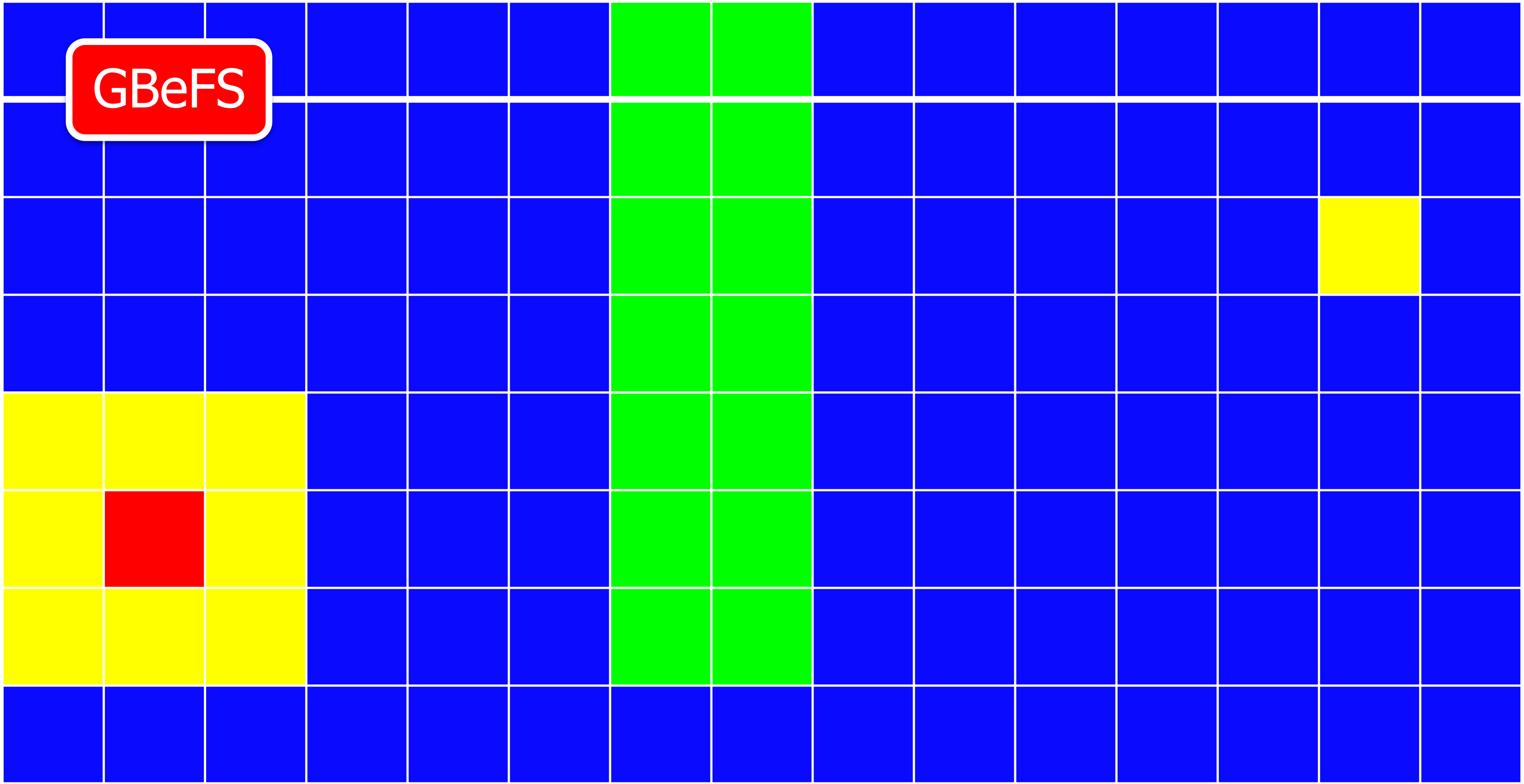

GBeFS

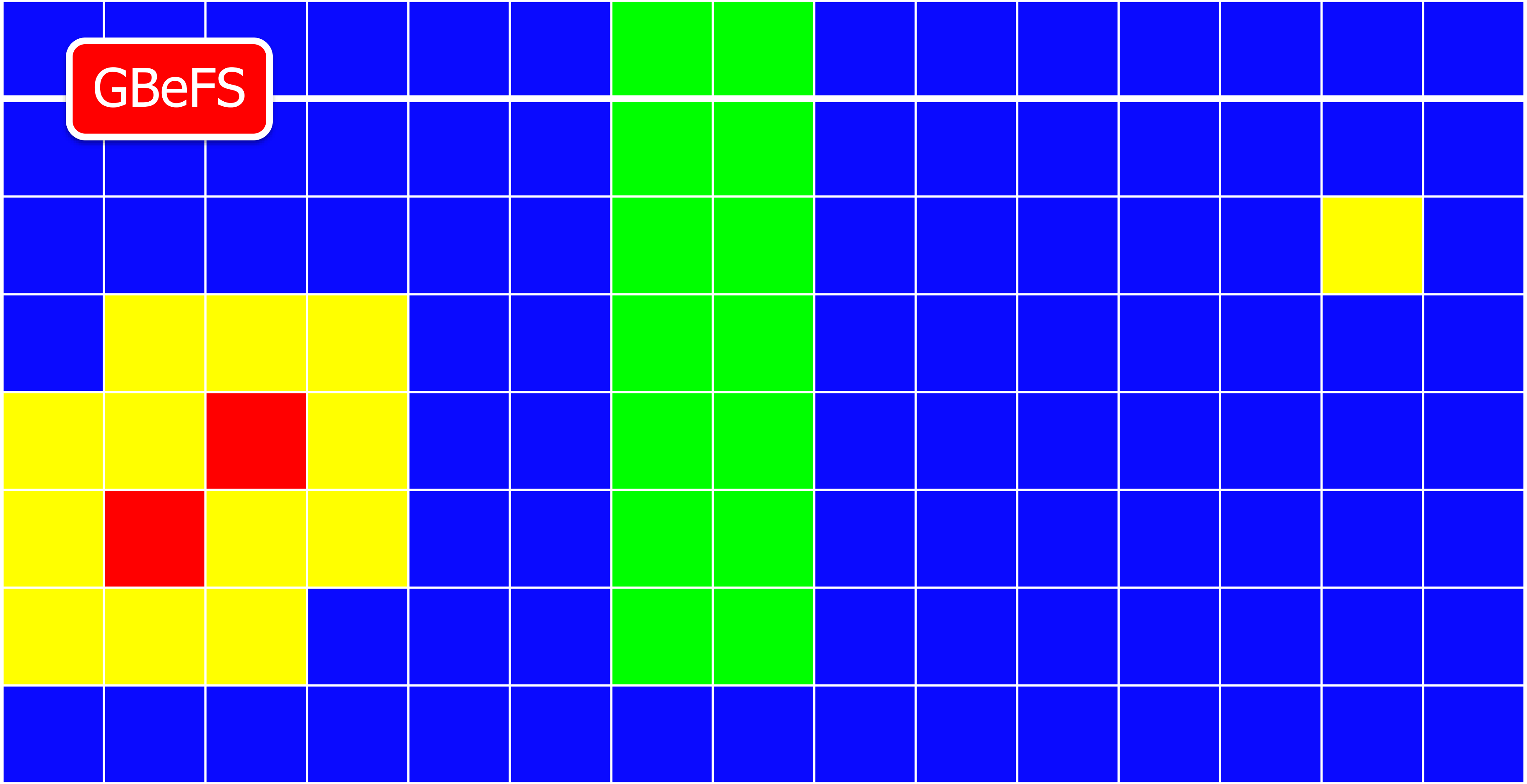

GBeFS

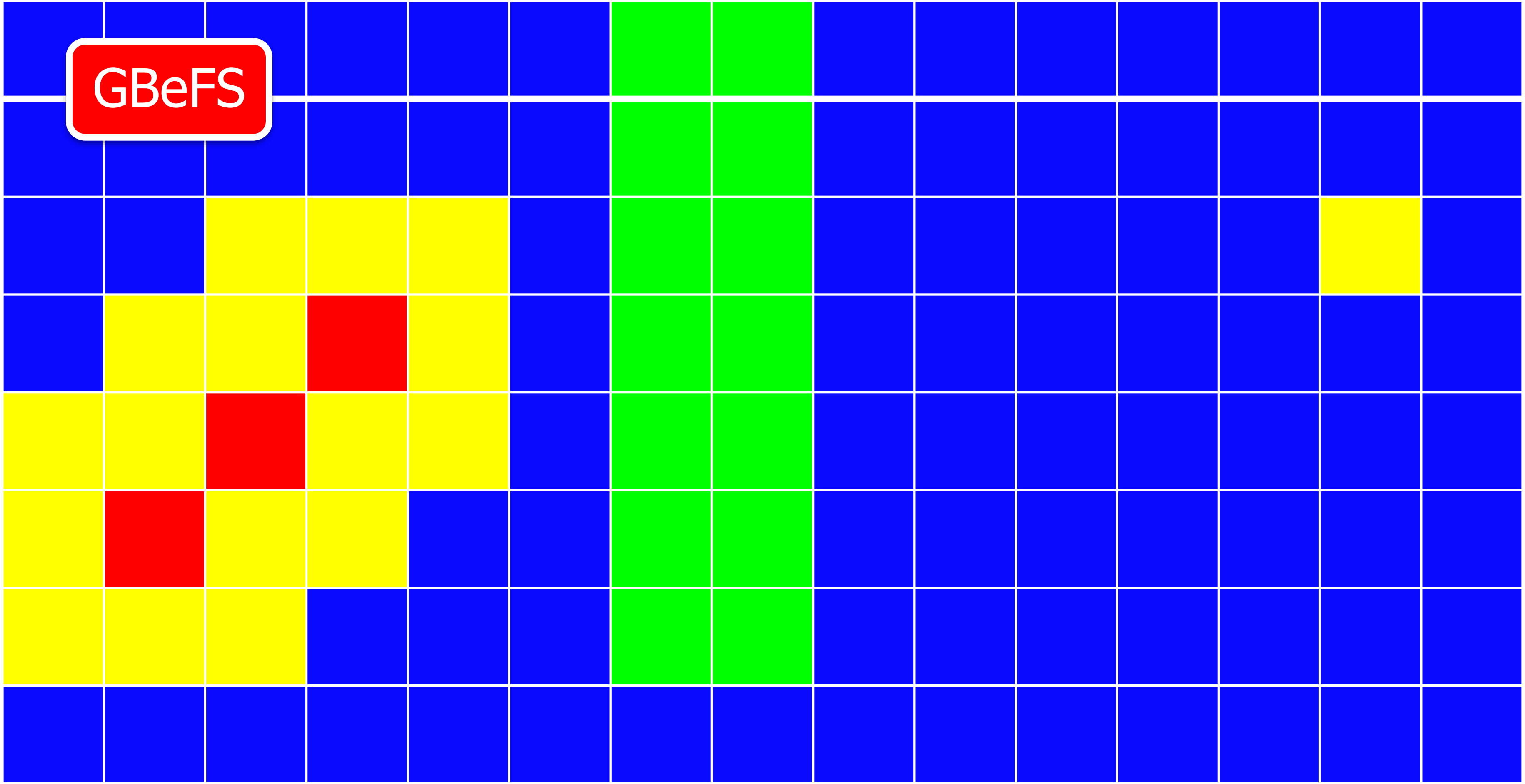

GBeFS

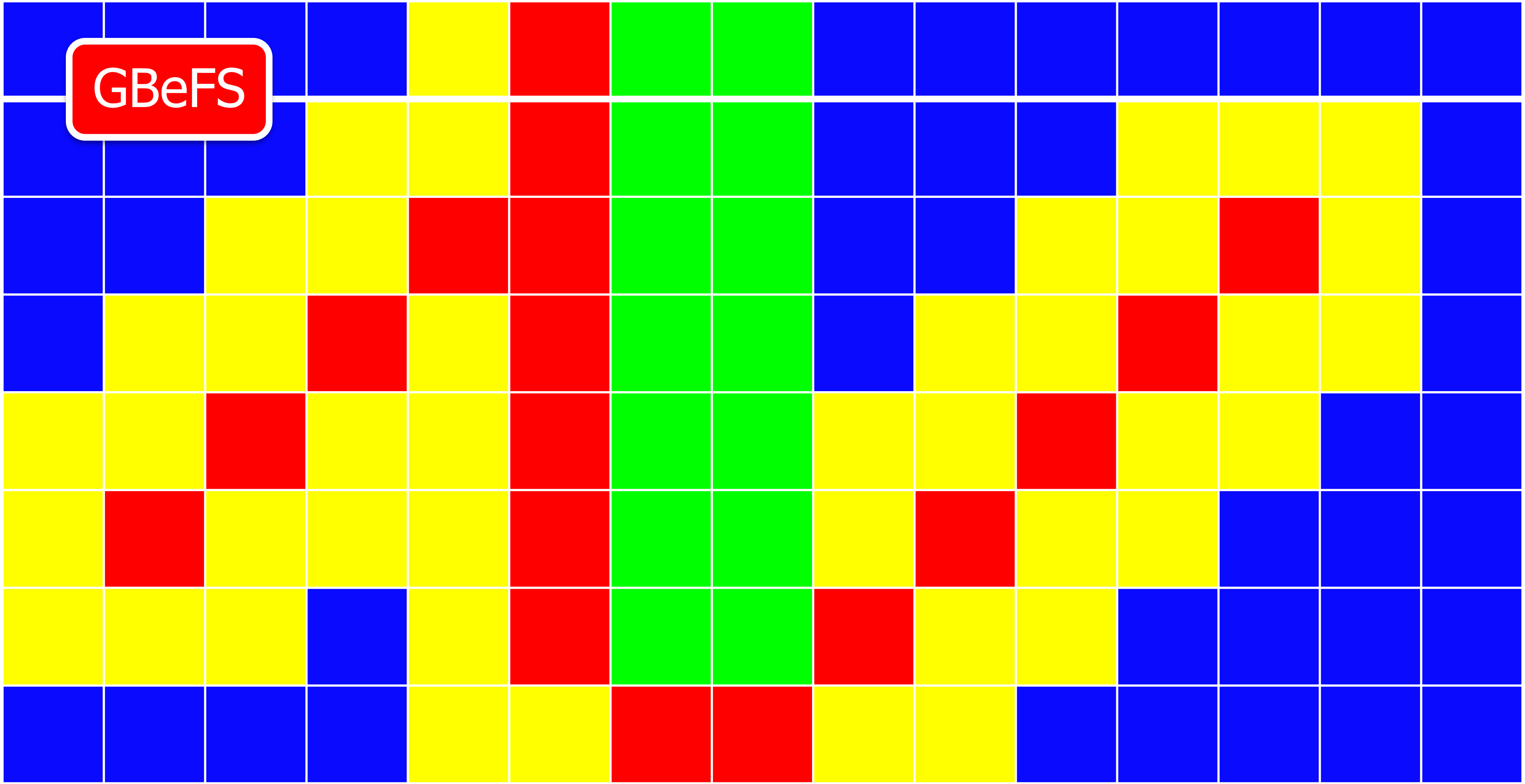

GBeFS


GBeFS


GBeFS


GBeFS


GBeFS


GBeFS

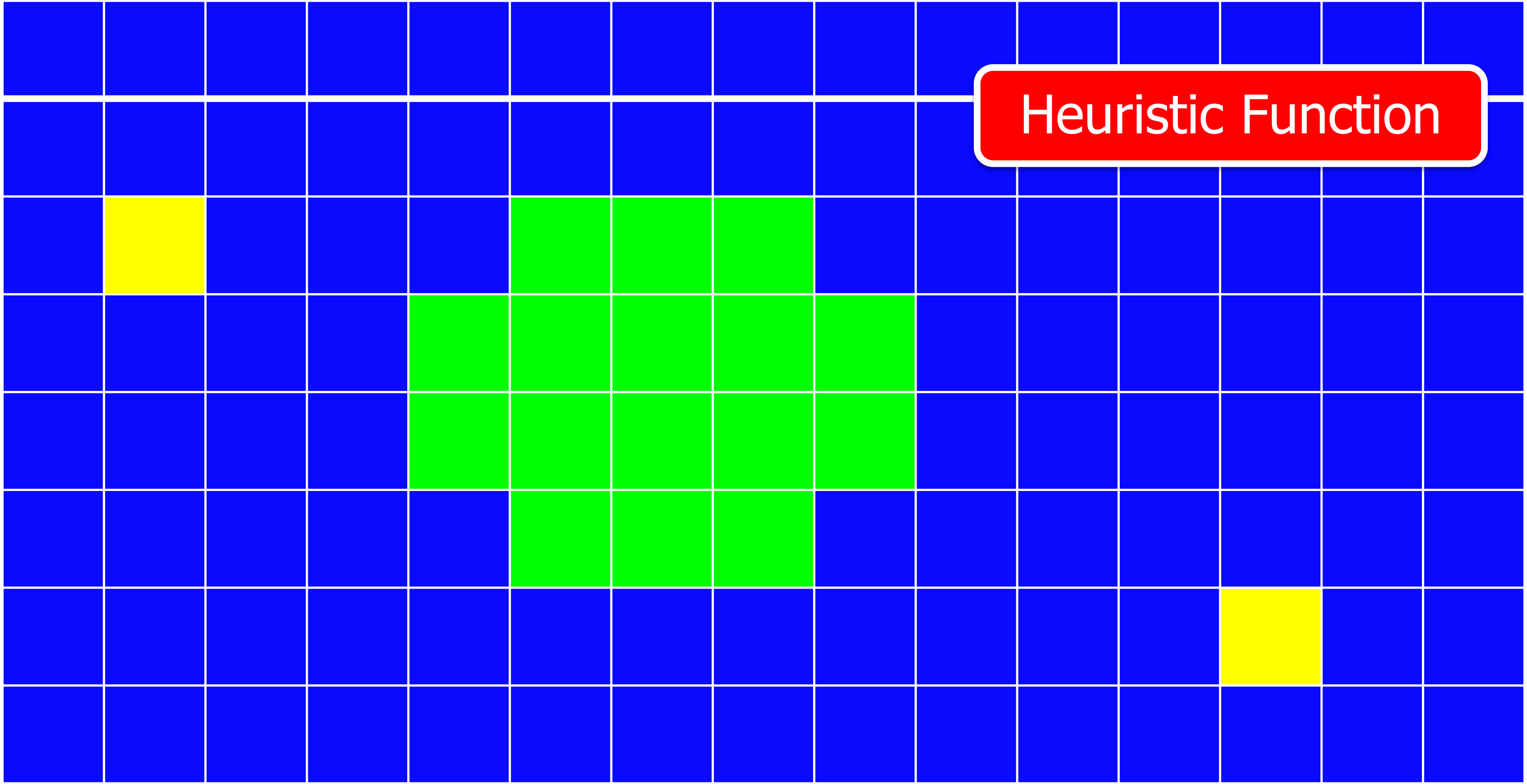
GBeFS Performance

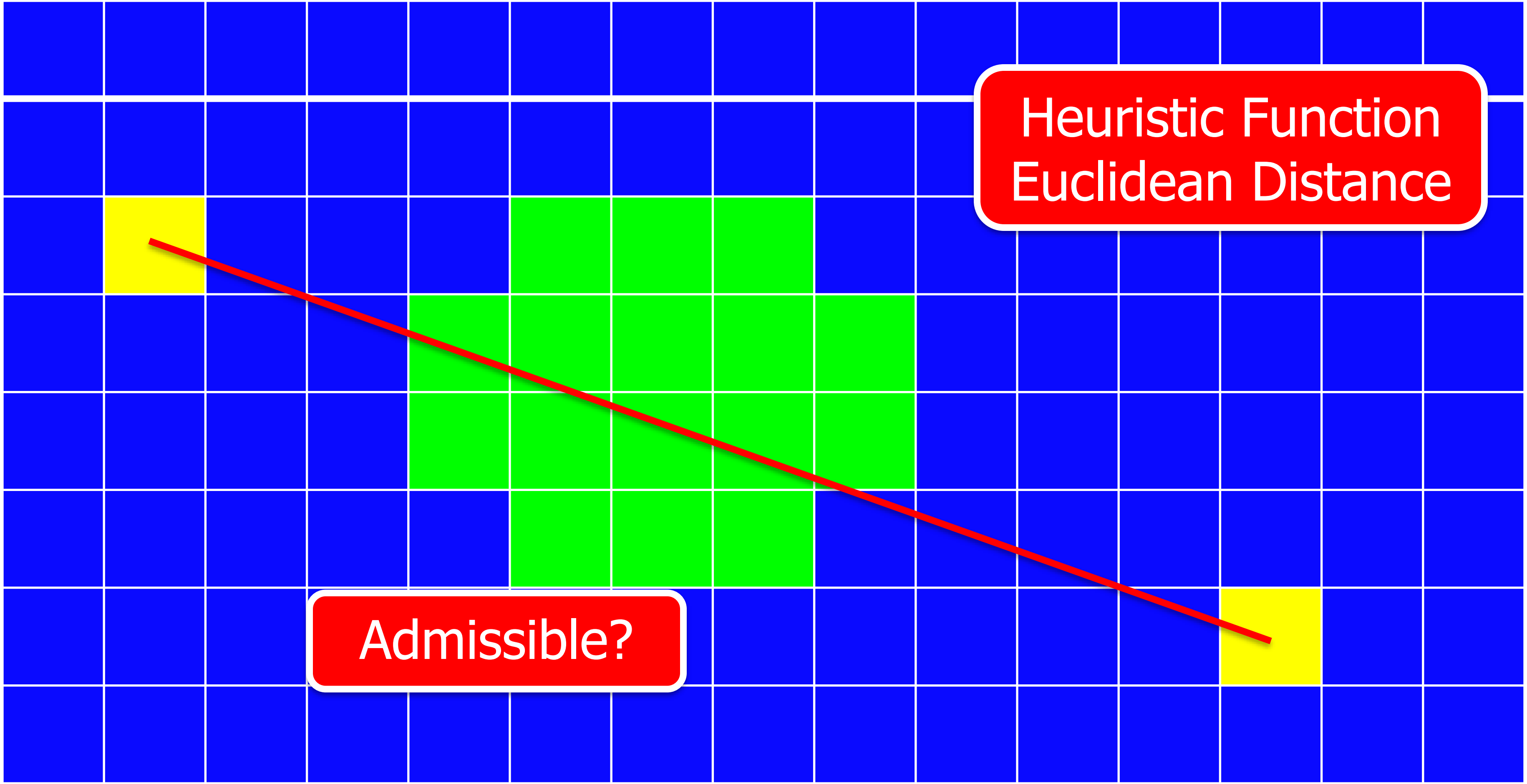
- Suffers similarly to DFS
- Incomplete in General
  - May not find a goal
  - May get lost in paths if heuristic is bad
- Not Optimal
  - May find a higher cost path than optimal
- Time complexity $O(b^m)$, m = max depth

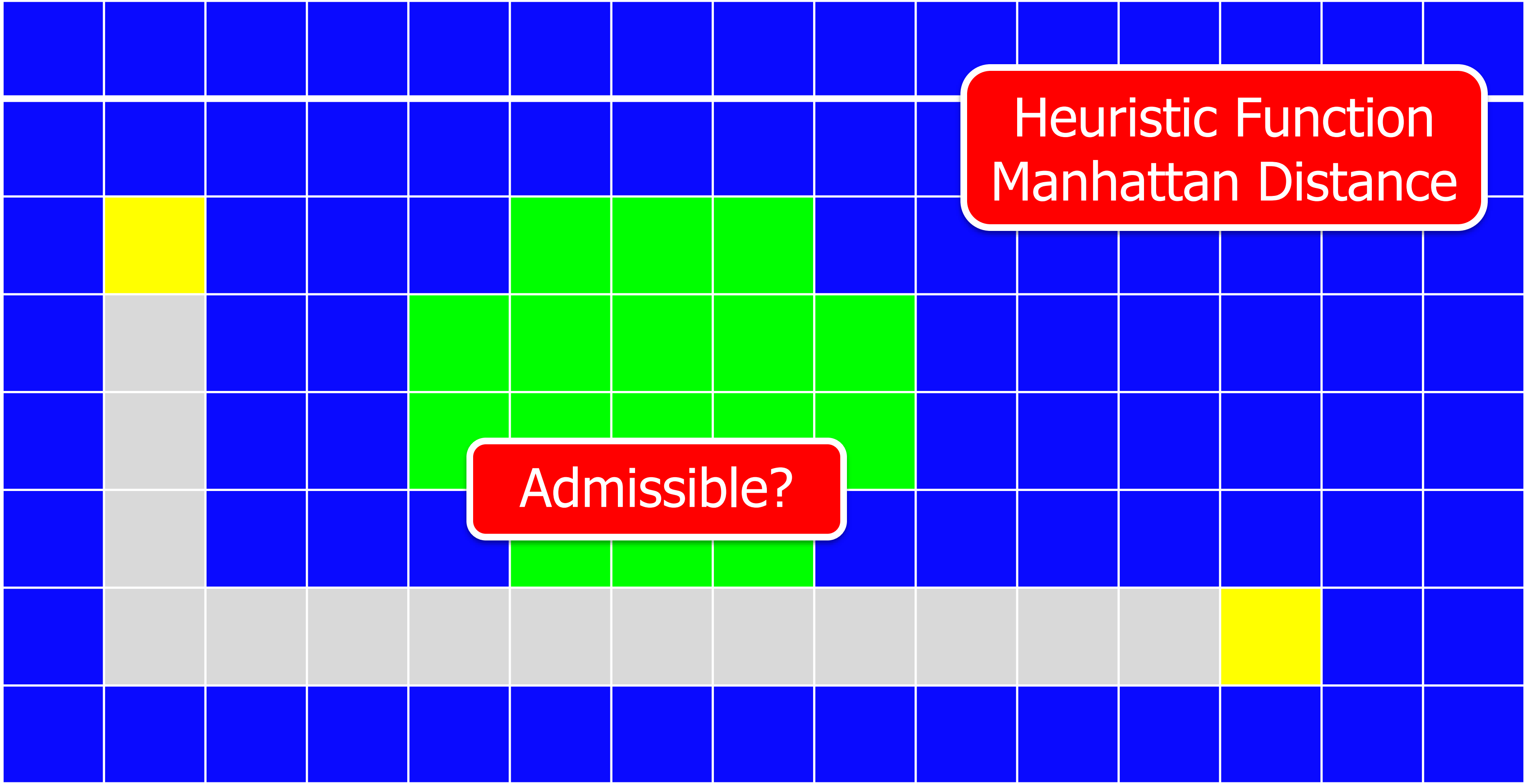
Admissible Heuristic

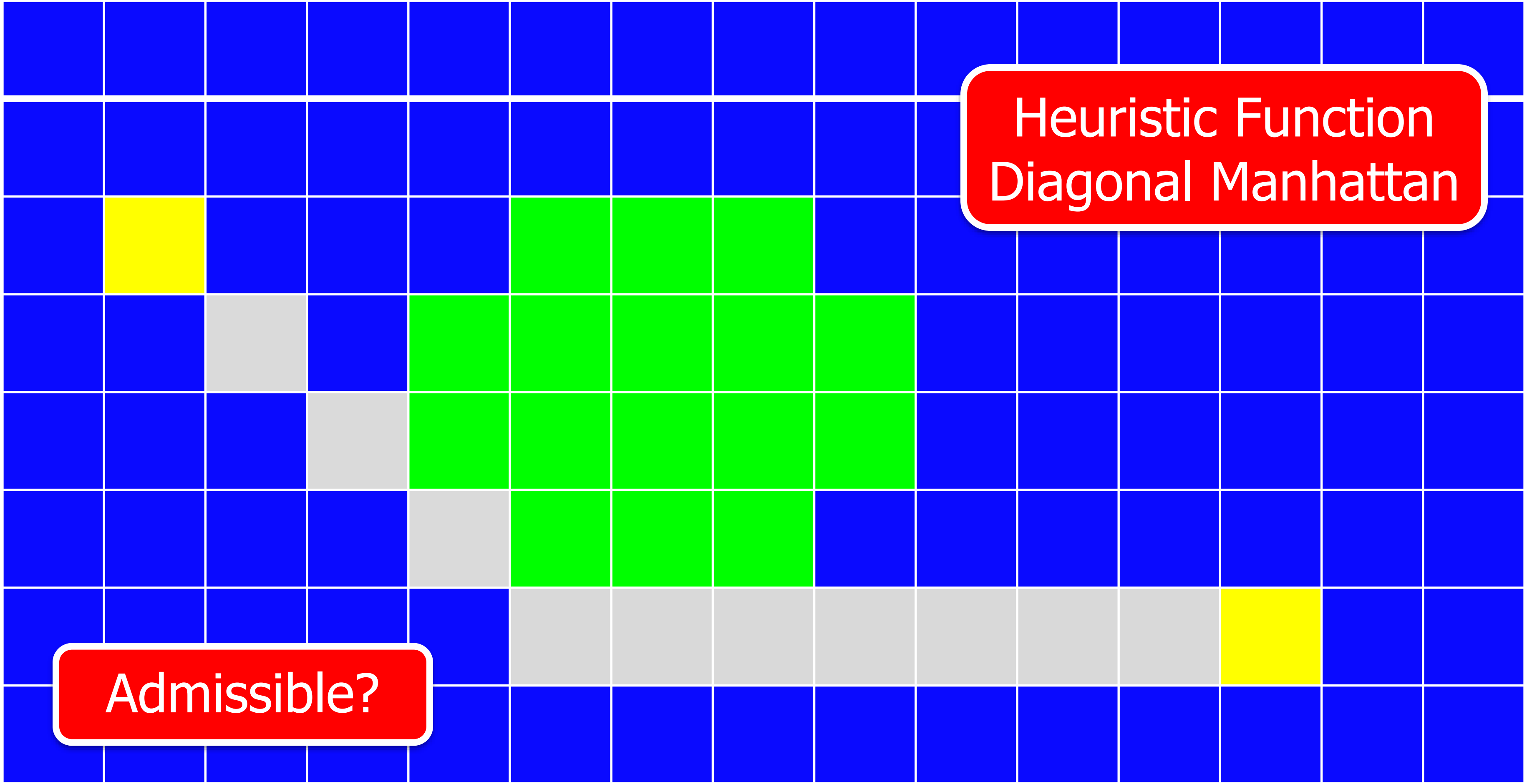
- Never overestimates distance to goal
- Can be seen as ‘optimistic’ guesses
- $h(n) <= h^*(n)$
- $f(n) = g(n) + h(n)$ never overestimates true cost of a path through n when $h(n)$ is admissible

Heuristic Function




Heuristic Function
Euclidean Distance
Admissible?


Heuristic Function
Manhattan Distance
Admissible?


Heuristic Function
Diagonal Manhattan
Admissible?

A* Search Algorithm

- Most well-known BeFS algorithm
- Evaluates nodes by using g(n) and h(n)
  - Heuristic function f(n) = g(n) + h(n)
  - f(n) = estimate of cheapest solution via n
- Select node from open list with min f(n)
- Performance depends on properties of the heuristic function

A* Tree-Search

```
Function GeBFS(problem, h)
  open = PQ{f=g+h,Node(problem.init_state)}
  while (true)
    if (open.empty) return fail
    node = pop_min_f(open)
    if (node.state is goal) return solution
    for c in Expand(node, problem)
      open.add(c)
```

GeBFS Tree-Search

```
Function GeBFS(problem, h)
  open = PQ{f=h,  Node(problem.init_state)}
  while (true)
    if (open.empty) return fail
    node = pop_min_f(open)
    if (node.state is goal) return solution
    for c in Expand(node, problem)
      open.add(c)
```

A* Performance

- A* using Tree-Search is complete
  - Will eventually search the entire tree, it will just search parts where the heuristic is lower first
- A* using the Tree-Search algorithm is optimal if heuristic h(n) is admissible
  - Using a heuristic that is not admissible can (and usually does) produce non-optimal paths

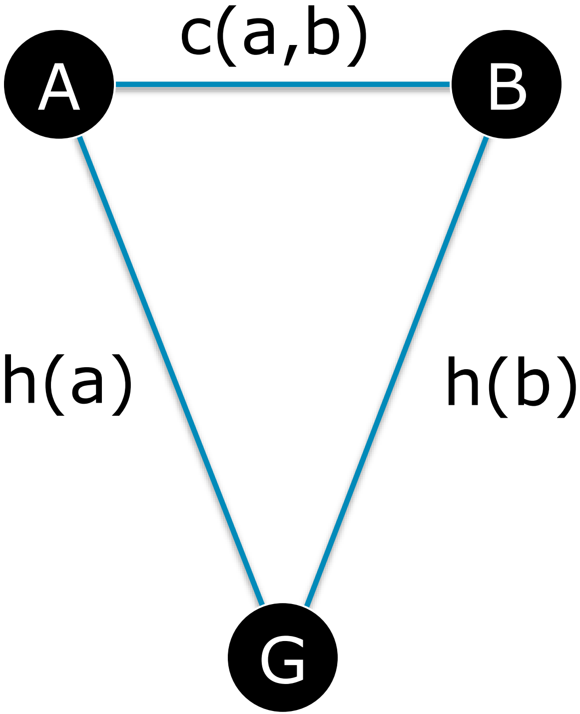
A* Using Graph-Search

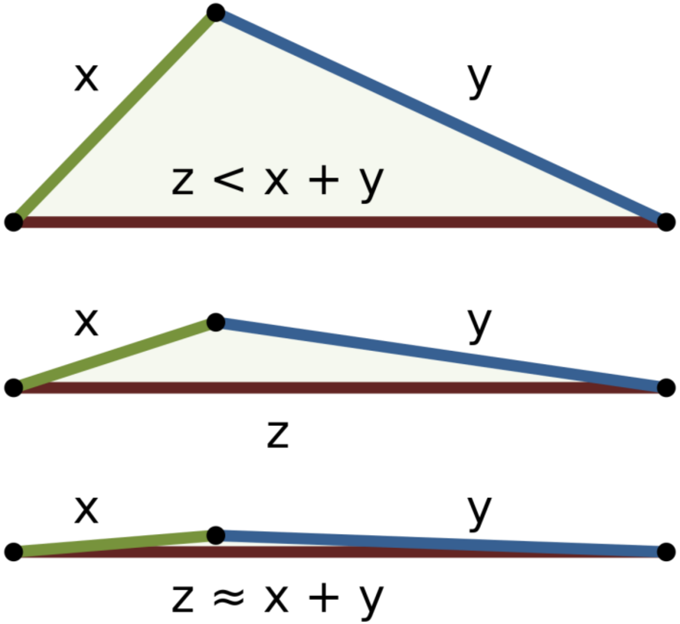
- Can return suboptimal solutions when we use an admissible heuristic
- A* Graph-Search using an admissible heuristic can discard an optimal path to a repeated state if it is not the first one that is generated during the search (later one might be better)
- We can fix this by imposing a consistent heuristic

Consistent Heuristic

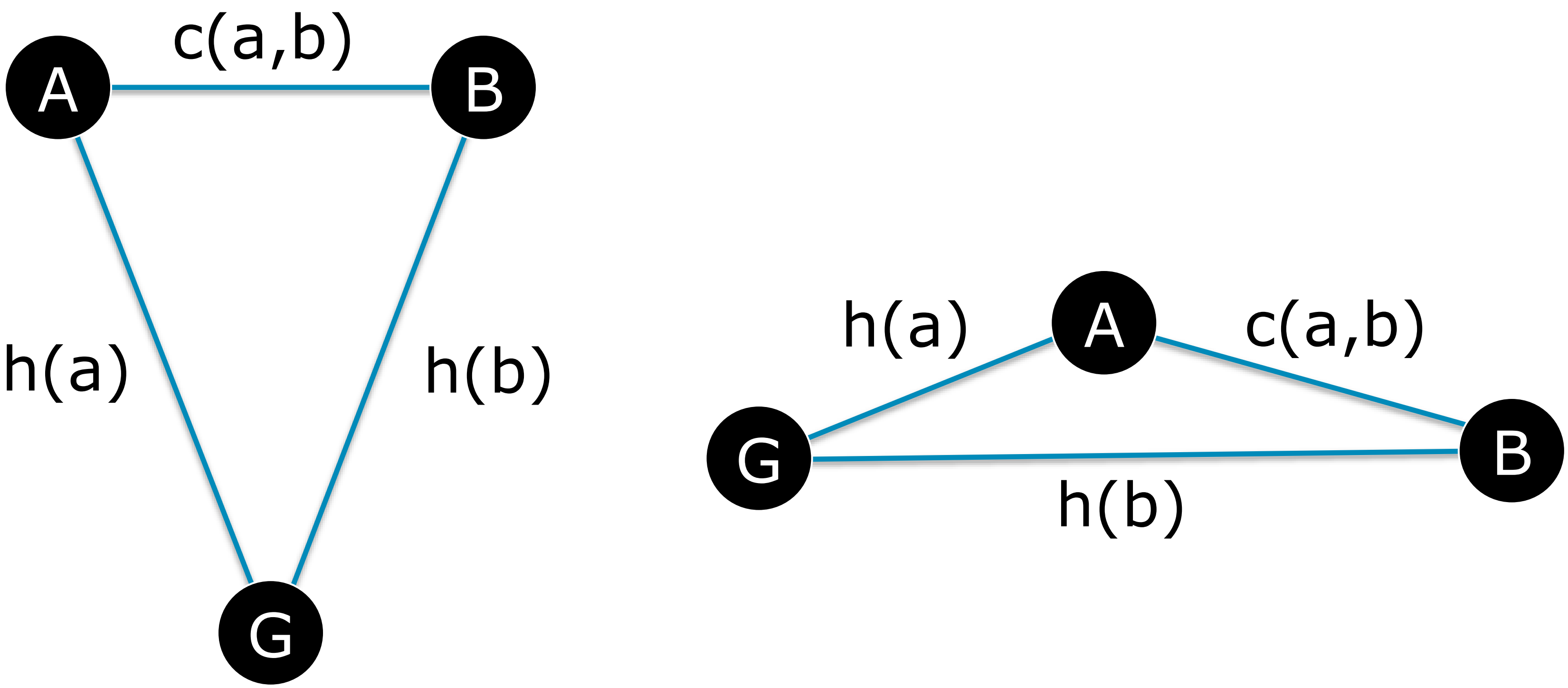
- Also called monotone heuristic
- Consistent if $| h(a) - h(b) | \leq c(a,b)$
  - $h(n)$ = estimate path cost from n to goal
  - $c(a,b)$ = cost of going from a to b
- The change in the heuristic value between two states must be less than or equal to the actual cost to go between them

$| h(a) - h(b) | \leq c(a,b)$

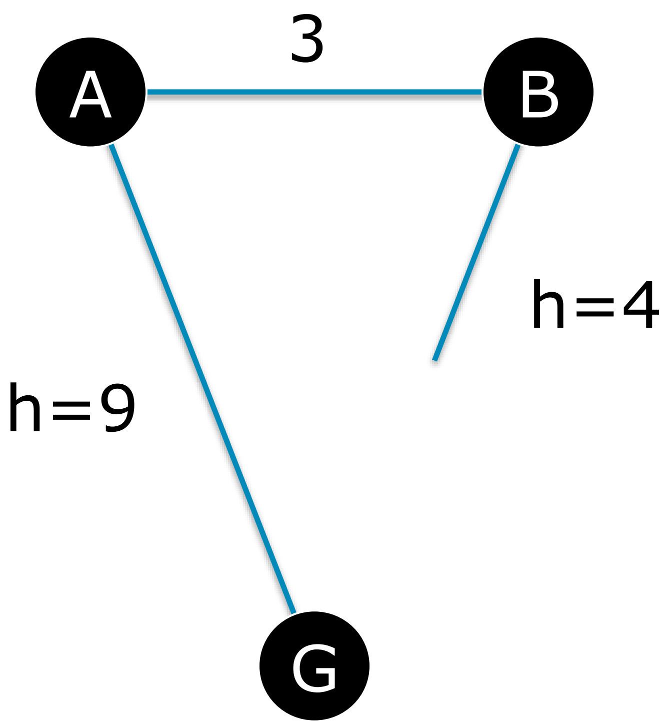




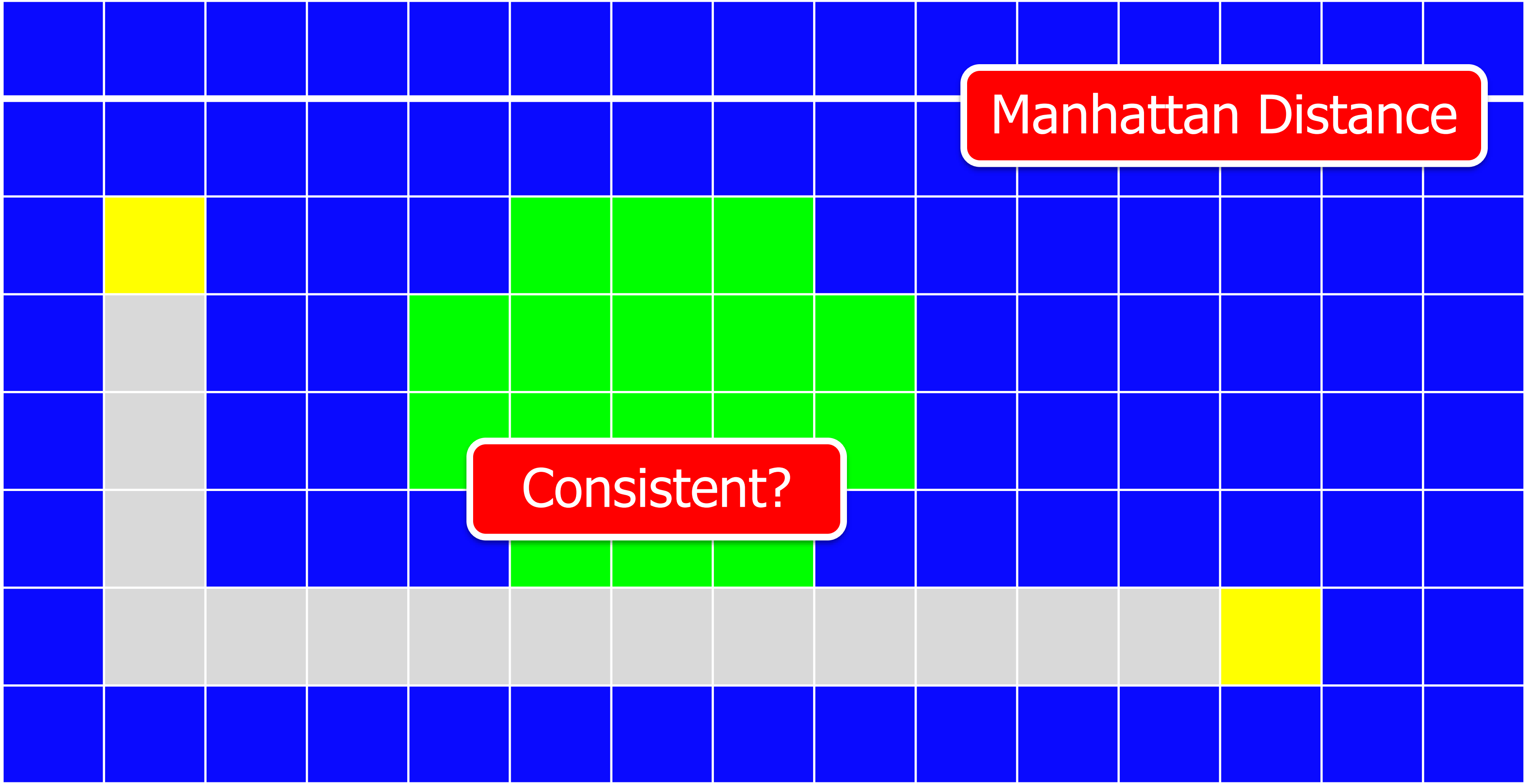
$$| h(a) - h(b) | \leq c(a,b)$$

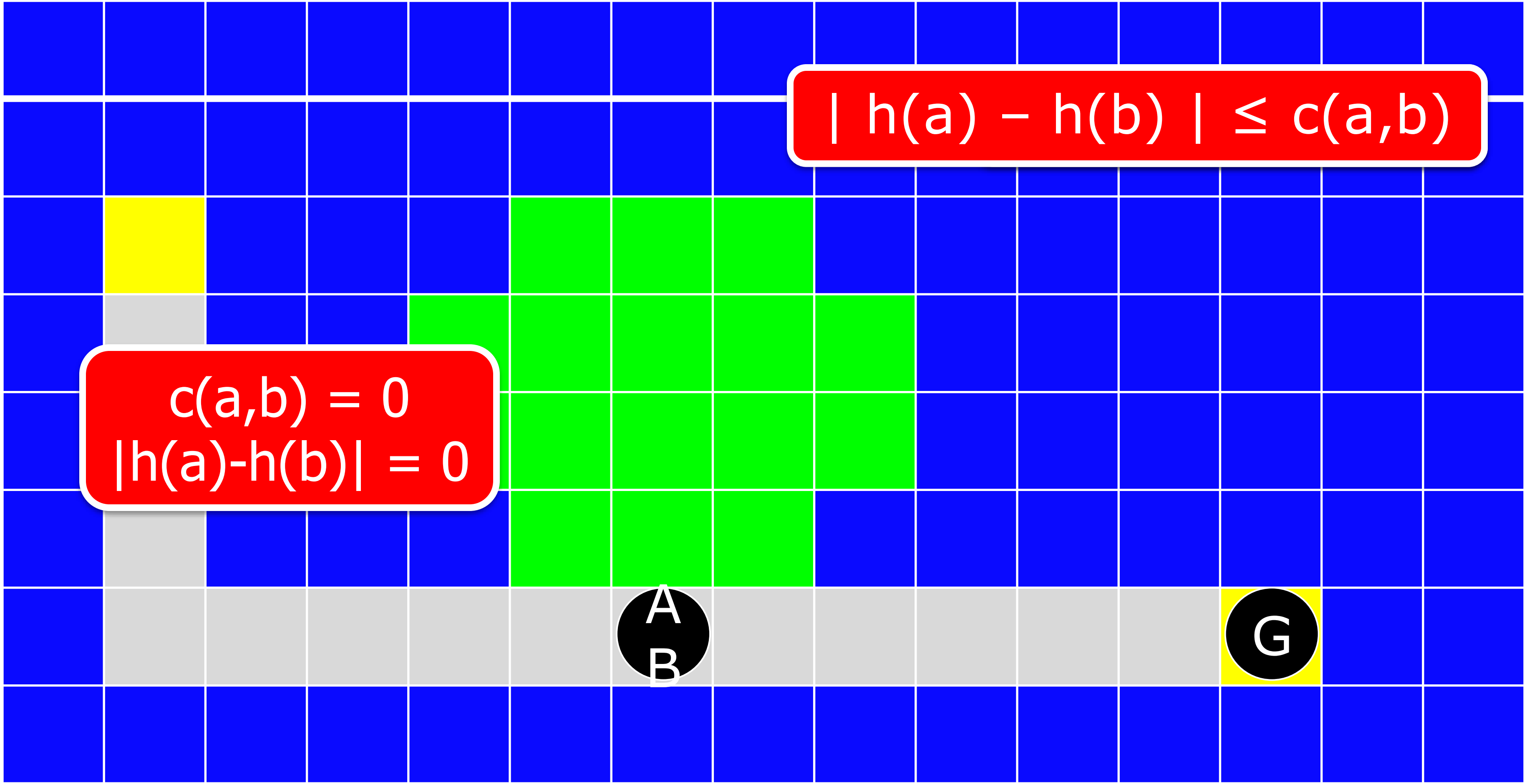


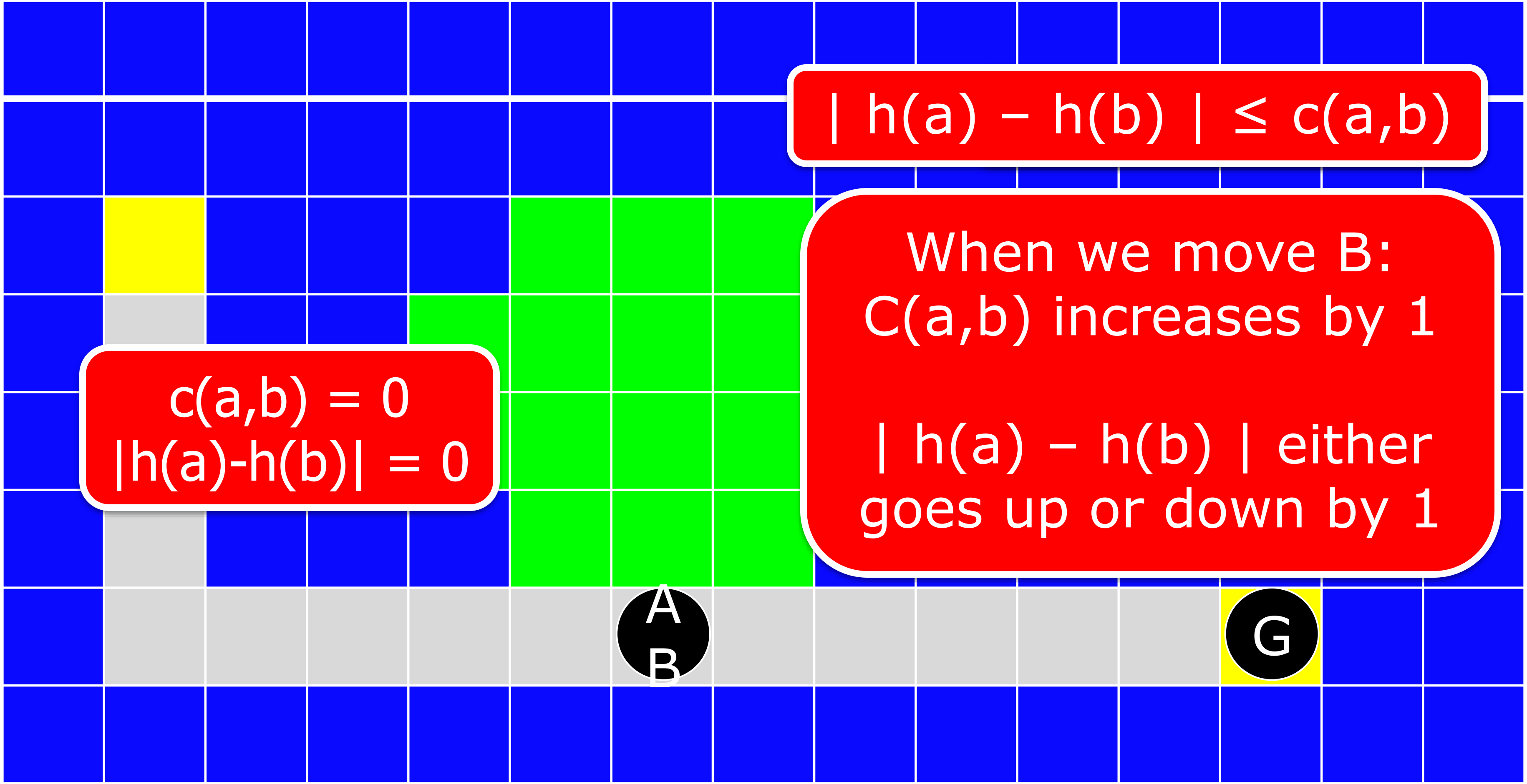
$$| h(a) - h(b) | \leq c(a,b)$$

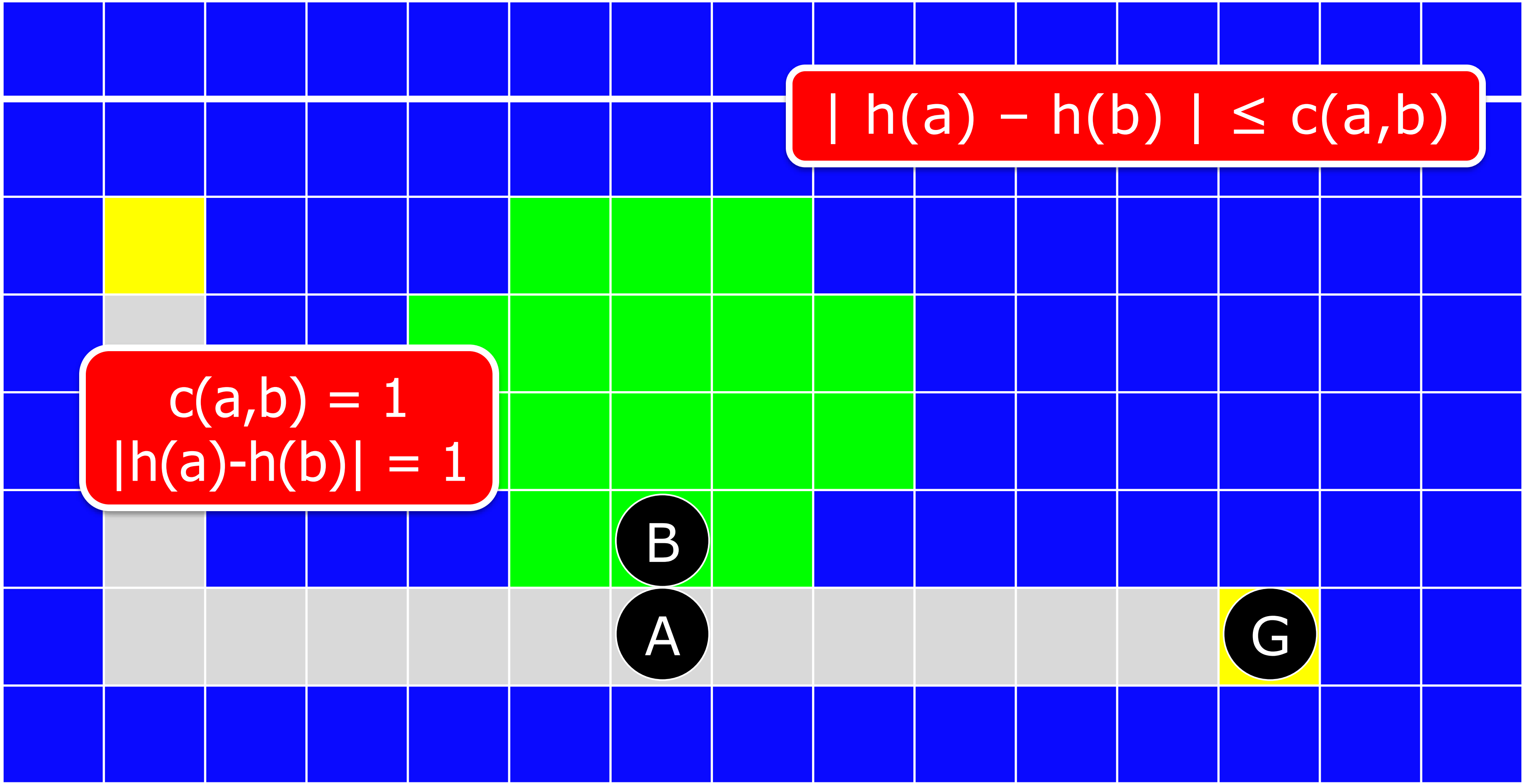


Inconsistent


Manhattan Distance
Consistent?


| h(a) – h(b) | ≤ c(a,b)
c(a,b) = 0
|h(a)-h(b)| = 0
A
B
G


| h(a) – h(b) | ≤ c(a,b)
When we move B:
C(a,b) increases by 1
| h(a) – h(b) | either goes up or down by 1
c(a,b) = 0
|h(a)-h(b)| = 0
A
B
G


| h(a) − h(b) | ≤ c(a,b)
c(a,b) = 1
|h(a)-h(b)| = 1
B
A
G

Consistent Heuristic

- Every consistent heuristic is also admissible
- In practice, it is hard to construct an admissible heuristic that isn't consistent
- If h(n) is consistent, the values of f(n) along any path are non-decreasing
- The sequence of nodes expanded by A* using Graph-Search is in non-decreasing order of f(n)
- Therefore, first goal node selected for expansion is optimal, since all later nodes are at least as expensive
- A* using Graph-Search is optimal if h(n) is consistent

A* Tree-Search

```
Function AStar(problem, h)
  open = PQ(f=g+h, Node(problem.init_state))
  while (true)
    if (open.empty) return fail
    node = pop_min_f(open)
    if (node.state is goal) return solution
    for c in Expand(node, problem)
      open.add(c)
```

Optimal if h(n) is Admissible

A* Graph-Search (Closed List)

```
Function AStar(problem, h)
  closed = {}
  open = PQ(f=g+h, Node(problem.init_state))
  while (true)
    if (open.empty) return fail
    node = pop_min_f(open)
    if (node.state is goal) return solution
    if (node.state in closed) continue
    closed.add(node.state)
    for c in Expand(node, problem)
      open.add(c)
```

Optimal if
h(n) is Consistent

A* Search

```
Function AStar(problem, h)
  closed = {}
  open = PQ(f=g+h, Node(problem.init_state))
  while (true)
    if (open.empty) return fail
    node = pop_min_f(open)
    if (node.state is goal) return solution
    if (node.state in closed) continue
    closed.add(node.state)
    for c in Expand(node, problem)
      open.add(c)
```

Greedy Best-First Search

```
Function GBeFS(problem)
  closed = {}
  open = PQ(f=h,   Node(problem.init_state))
  while (true)
    if (open.empty) return fail
    node = pop_min_f(open)
    if (node.state is goal) return solution
    if (node.state in closed) continue
    closed.add(node.state)
    for c in Expand(node, problem)
      open.add(c)
```

Uniform Cost Search

```
Function UCS(problem)
  closed = {}
  open = PQ(f=g,   Node(problem.init_state))
  while (true)
    if (open.empty) return fail
    node = pop_min_f(open)
    if (node.state is goal) return solution
    if (node.state in closed) continue
    closed.add(node.state)
    for c in Expand(node, problem)
      open.add(c)
```

Breadth-First Search

```
Function BFS(problem)
  closed = {}
  open = Queue(Node(problem.initial_state))
  while (true)
    if (open.empty) return fail
    node = pop(open)
    if (node.state is goal) return solution
    if (node.state in closed) continue
    closed.add(node.state)
    for c in Expand(node, problem)
      open.add(c)
```

Depth-First Search

```
Function DFS(problem)
  closed = {}
  open = Stack(Node(problem.initial_state))
  while (true)
    if (open.empty) return fail
    node = pop(open)
    if (node.state is goal) return solution
    if (node.state in closed) continue
    closed.add(node.state)
    for c in Expand(node, problem)
      open.add(c)
```

A* Search

```
Function AStar(problem, h)
  closed = {}
  open = PQ(f=g+h, Node(problem.init_state))
  while (true)
    if (open.empty) return fail
    node = pop_min_f(open)
    if (node.state is goal) return solution
    if (node.state in closed) continue
    closed.add(node.state)
    for c in Expand(node, problem)


      open.add(c)
```

A* Search (Memory Optimization 1)

```
Function AStar(problem, h)
  closed = {}
  open = PQ(f=g+h, Node(problem.init_state))
  while (true)
    if (open.empty) return fail
    node = pop_min_f(open)
    if (node.state is goal) return solution
    if (node.state in closed) continue
    closed.add(node.state)
    for c in Expand(node, problem)
      if (c.state in closed) continue

      open.add(c)
```

If the state of the child is already in the closed list, we will eventually skip it

Saves Memory in Open List

A* Search (Memory Optimization 2)

```
Function AStar(problem, h)
  closed = {}
  open = PQ(f=g+h, Node(problem.init_state))
  while (true)
    if (open.empty) return fail
    node = pop_min_f(open)
    if (node.state is goal) return solution
    if (node.state in closed) continue
    closed.add(node.state)
    for c in Expand(node, problem)
      if (c.state in closed) continue
      if (node n in open with c.state and n.g <= c.g) continue
      open.add(c)
```

If a better path to c.state already exists in the open list, skip this child node

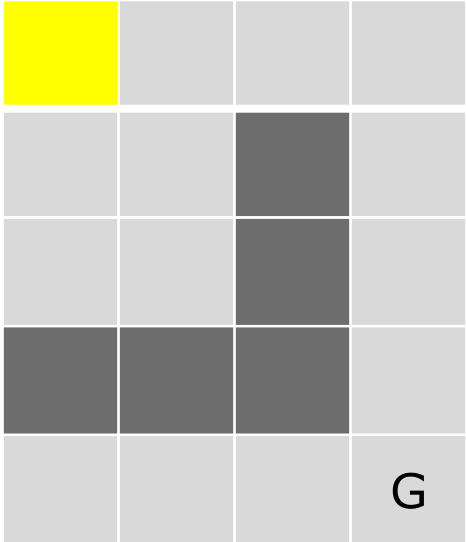
May be very slow in practice

A* Graph-Search Performance

- Assuming we are using a consistent heuristic
- Let C* be optimal solution path cost
- A* expands all nodes with f(n) < C*
- A* expands no nodes with f(n) > C*
- A* is optimally efficient for a given h(n)
  - No other optimal algorithm is guaranteed to expand fewer nodes than A*
  - Any algorithm not expanding all nodes with f(n) < C* risks missing the optimal path

A* Graph-Search Performance

- Pros
  - A* is Complete
  - A* is Optimal
  - A* is Optimally Efficient for given h(n)
- Cons
  - Number of nodes searched can still be exponential in the length of the solution
  - For many large problems, A* is much better than uninformed search, but still infeasible in time / memory




```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

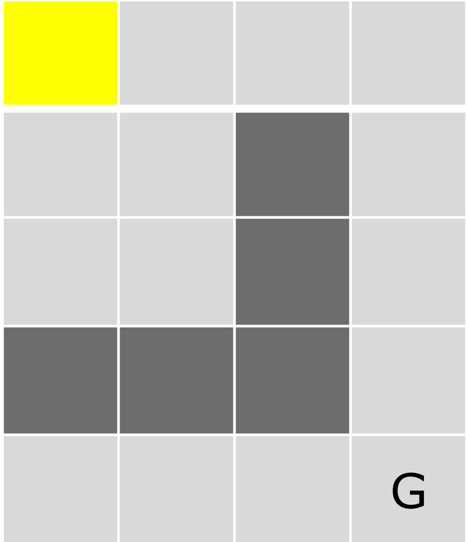
Open List (Nodes)

ID	S	G	H	F	A	P

Closed

State

Action Costs:
Cardinal: 100
Diagonal: 141



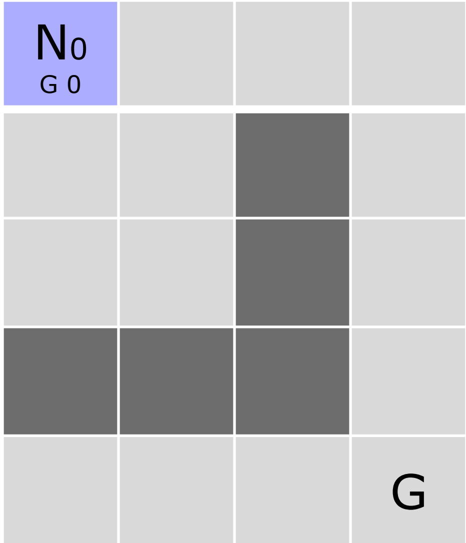

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
0	0,0	0	523	523		

Closed

State



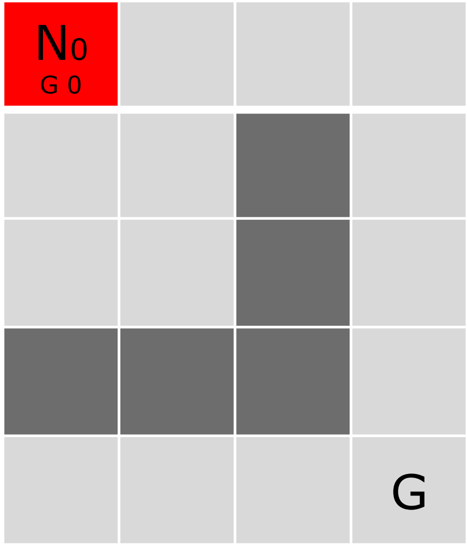

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P
0	0,0	0	523	523		

Closed

State



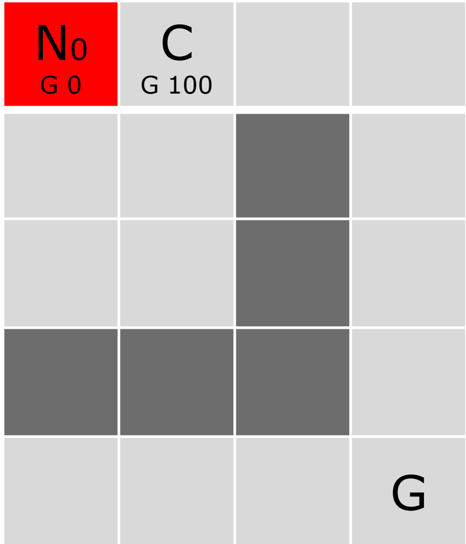

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P

Closed

State
0, 0



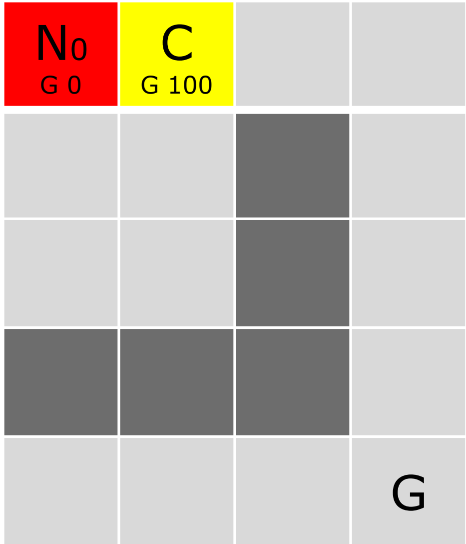

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

Open List (Nodes)

ID	S	G	H	F	A	P

Closed

State
0, 0



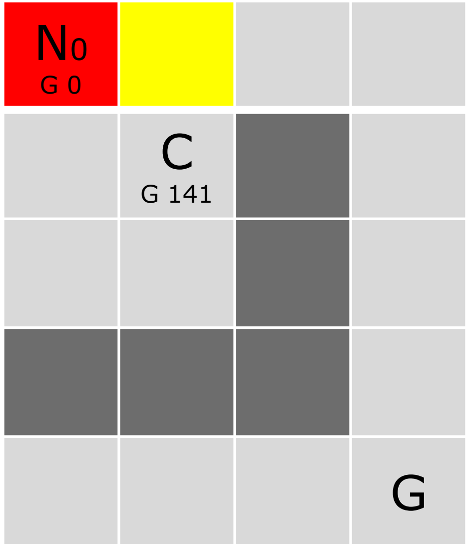

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

Open List (Nodes)

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

Closed

State
0, 0



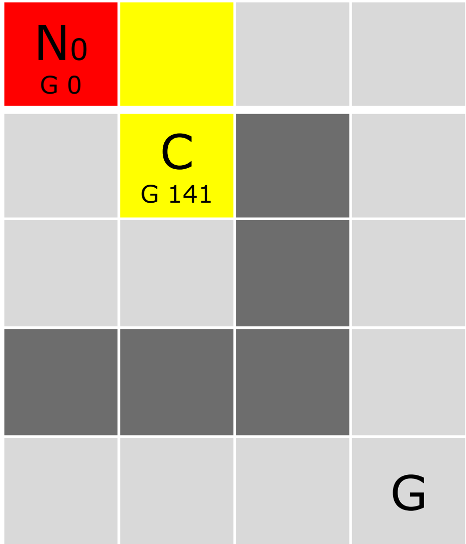

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

Open List (Nodes)

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

Closed

State
0, 0



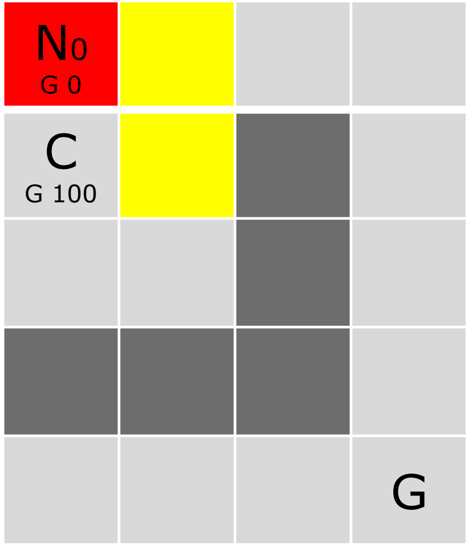

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



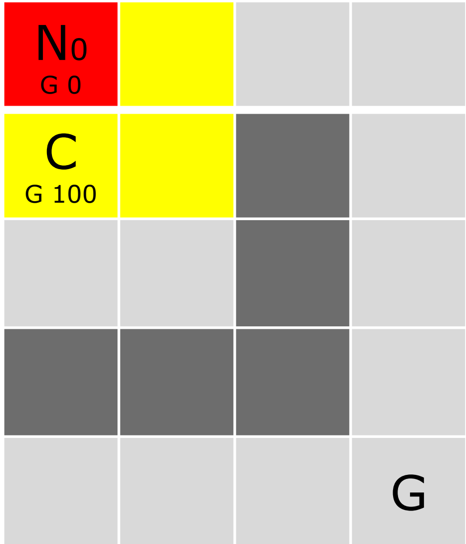

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



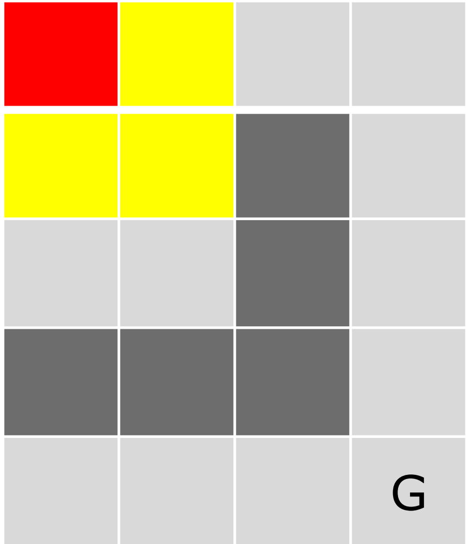

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



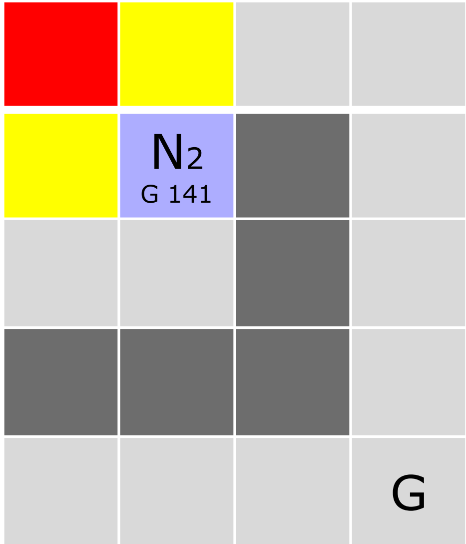

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



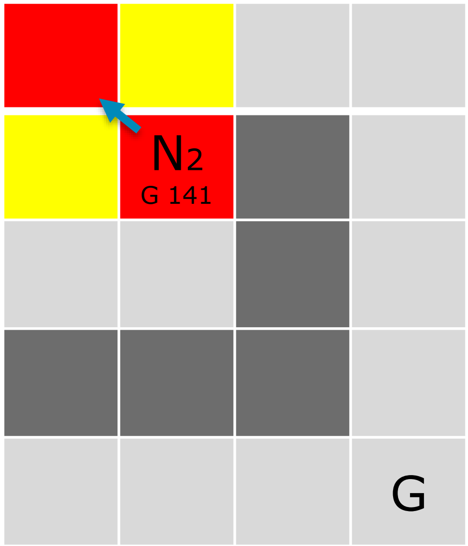

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



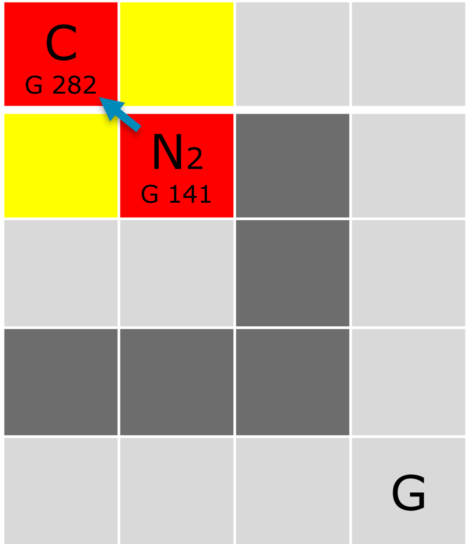

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



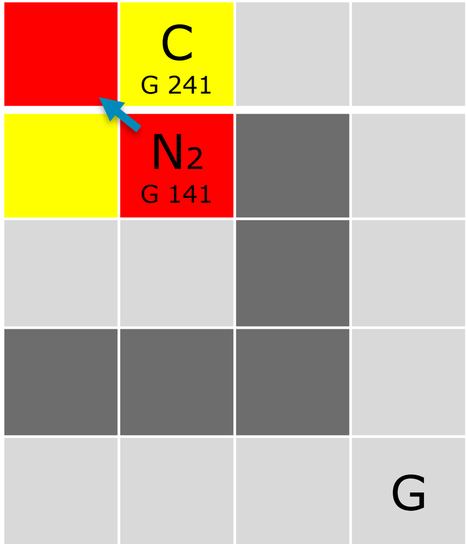

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



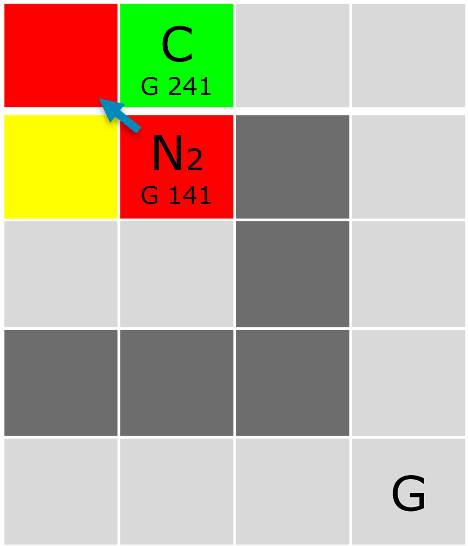

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



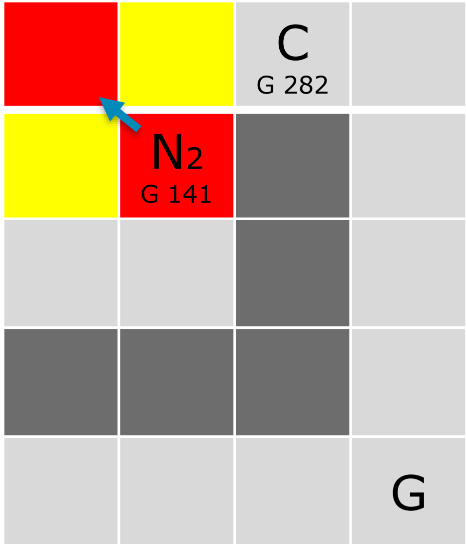

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



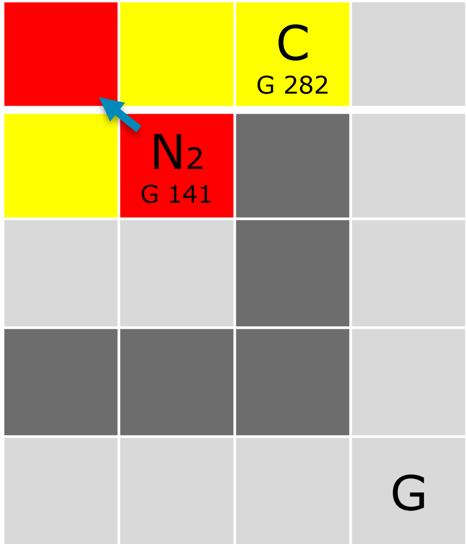

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



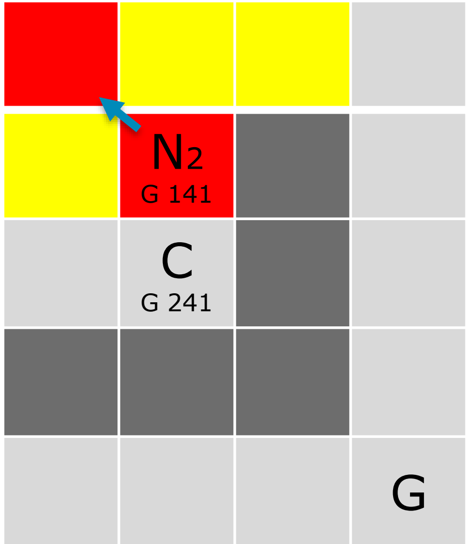

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



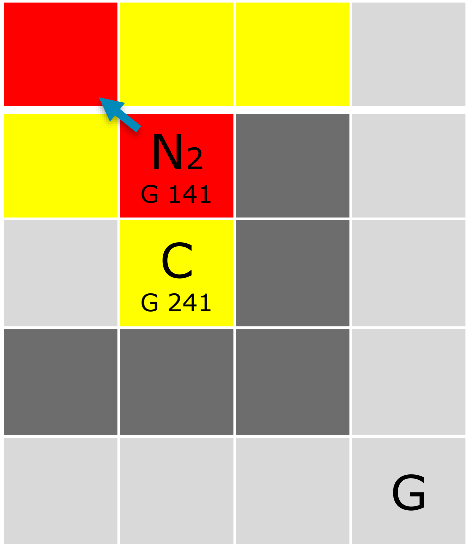

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



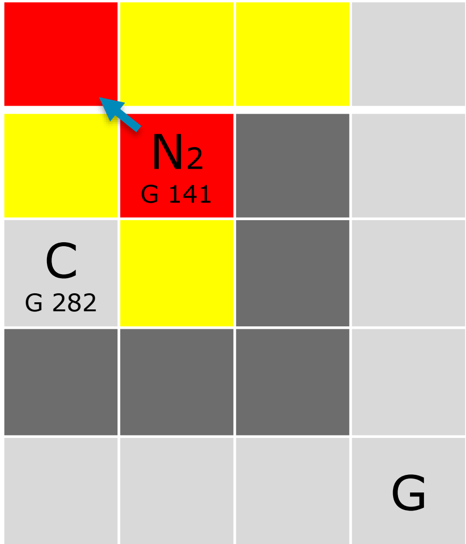

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



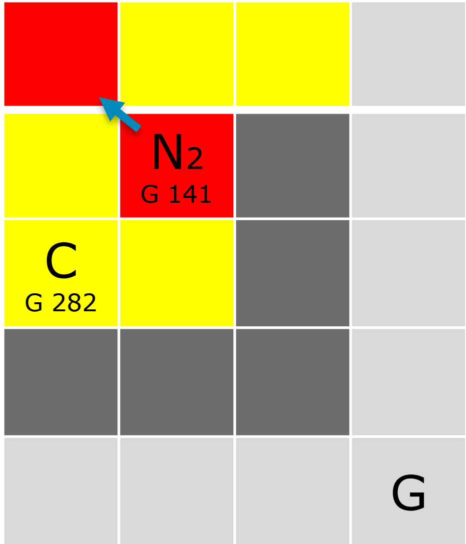

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



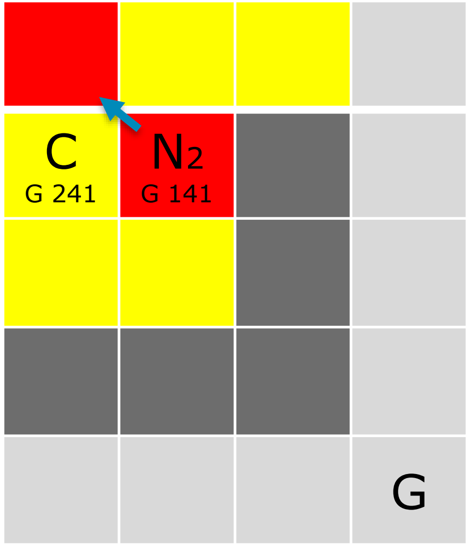

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



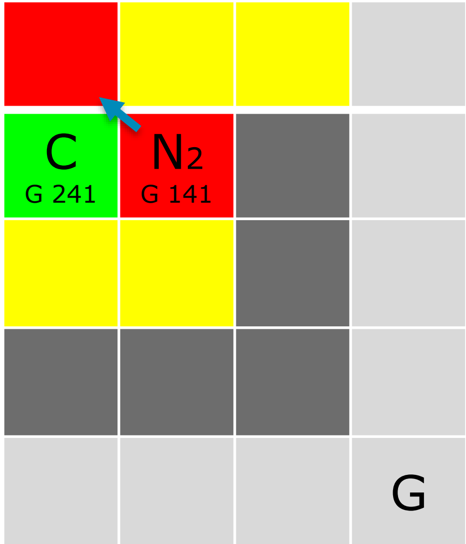

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



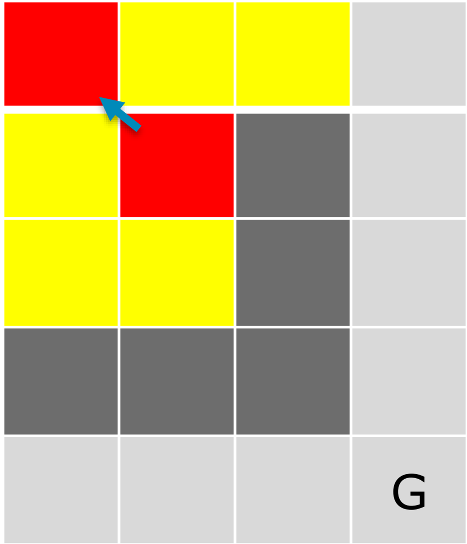

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



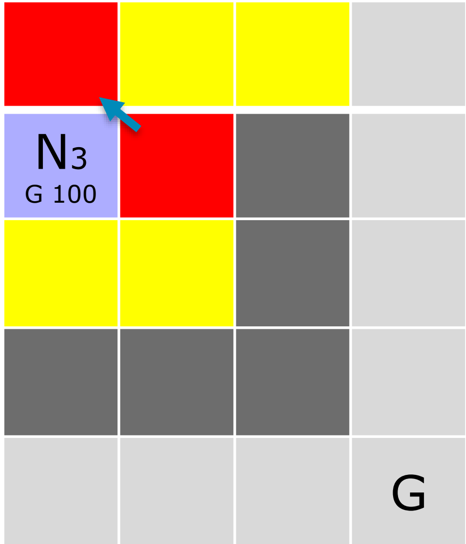

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



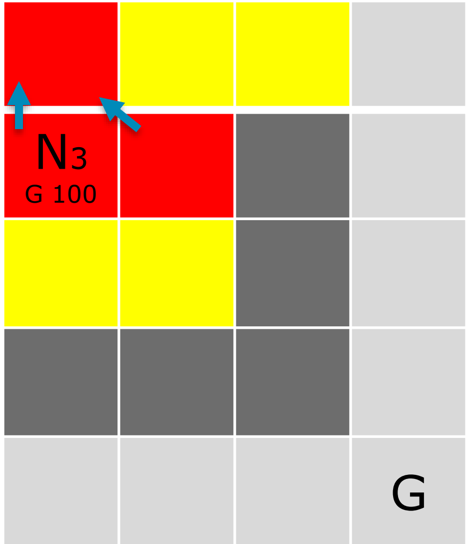

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



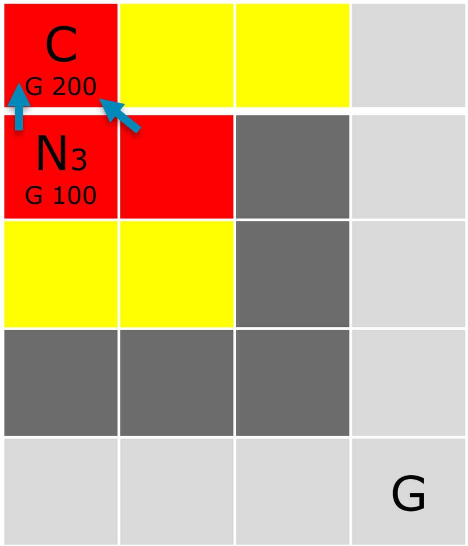

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



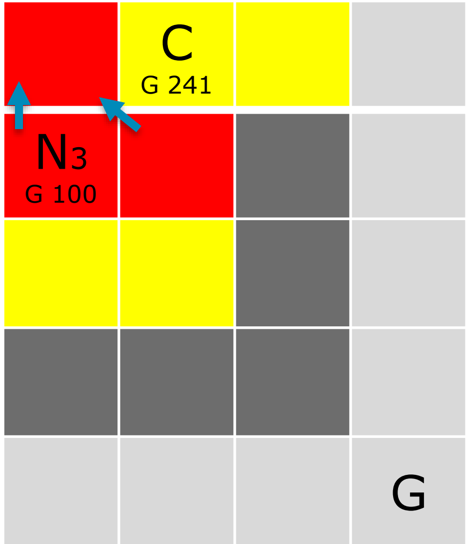

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



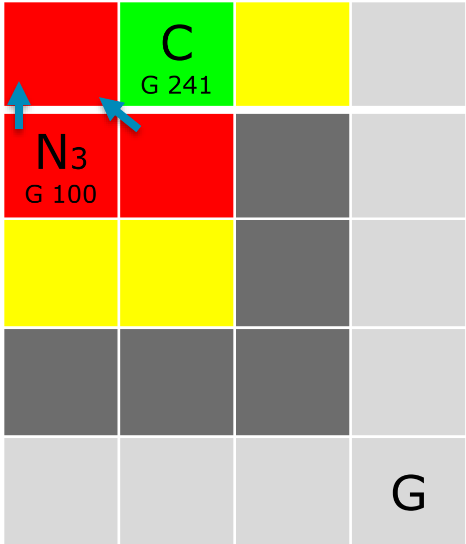

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



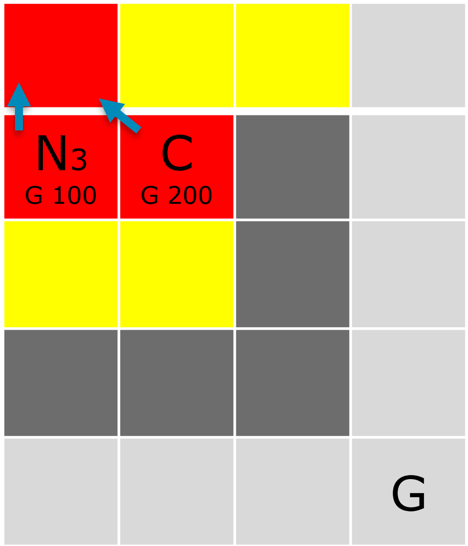

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



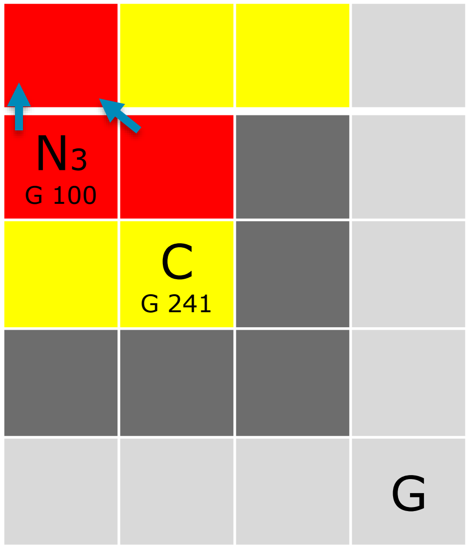

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



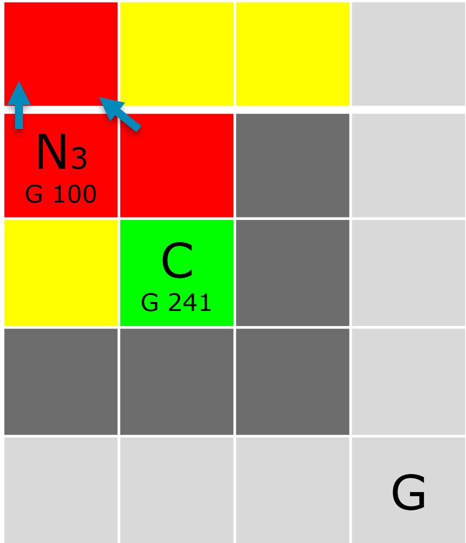

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



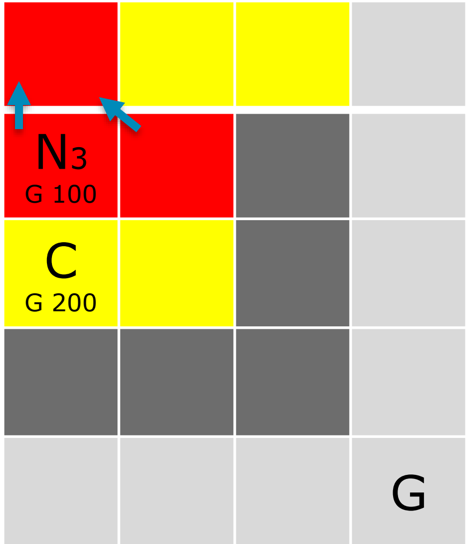

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



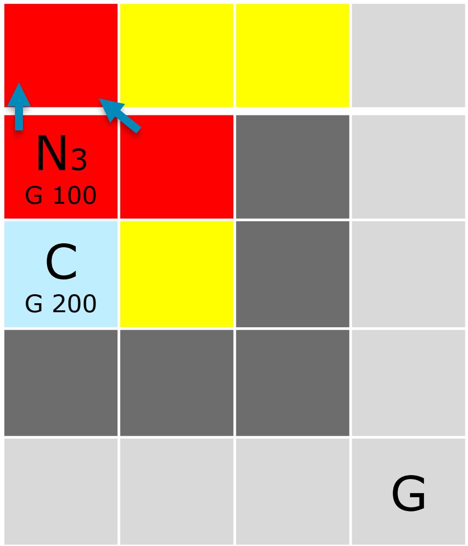

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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




```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

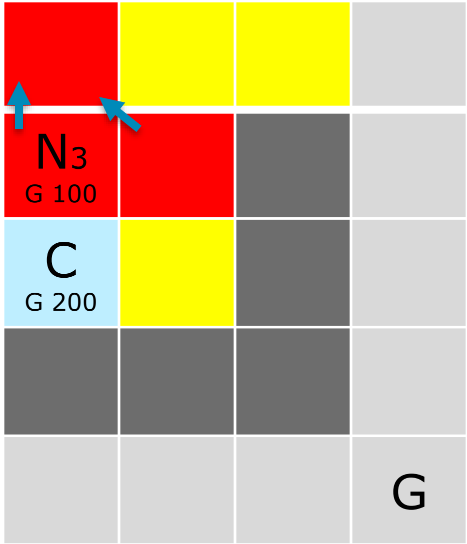
Open List (Nodes)

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

Important:
Node 6 Will Never Expand
Remove (if efficient)



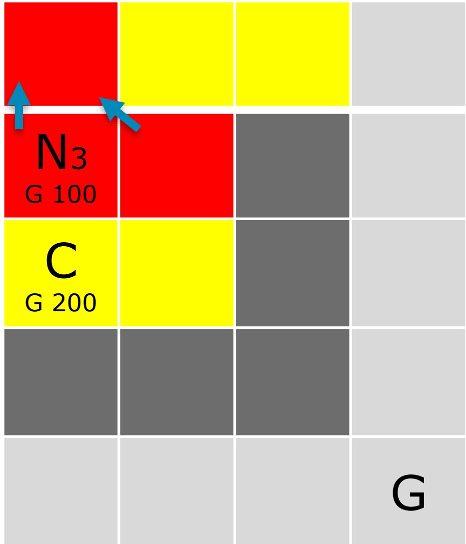
1. while (!open.empty)
2. node = remove min f from open
3. if (node.state is goal) return path
4. if (node.state in closed) continue
5. add node to closed
6. for c in expand(node)
7. if c.state in closed: continue
8. if c in open with <= g: continue
9. else open.add(c)

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



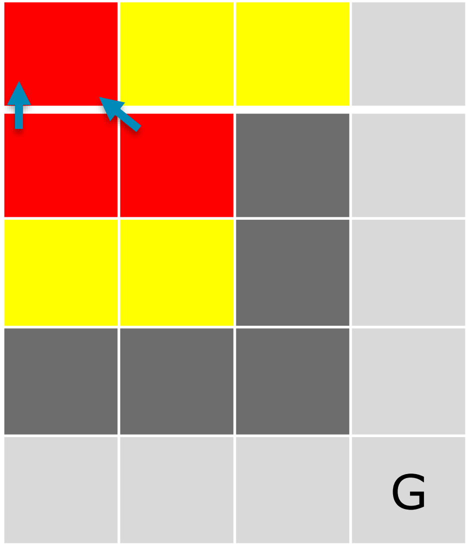

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



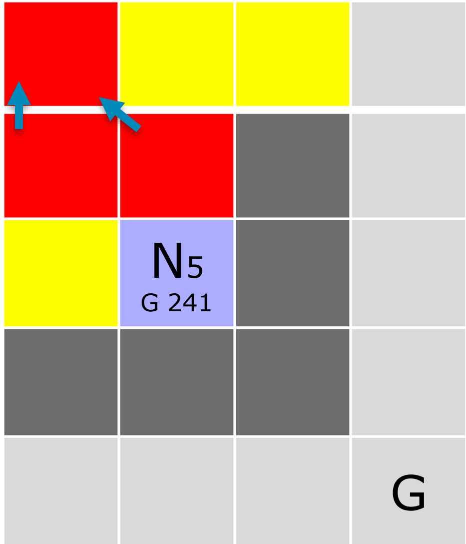

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



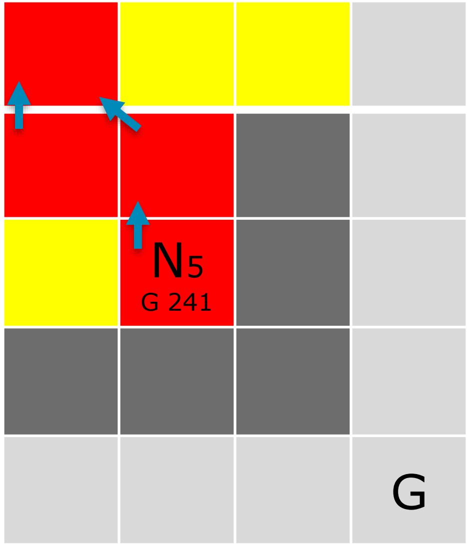

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



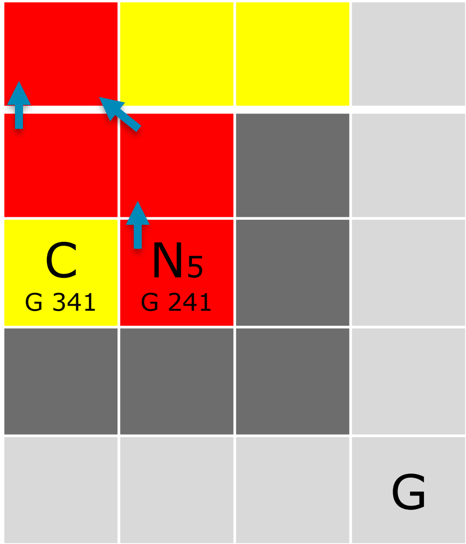

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



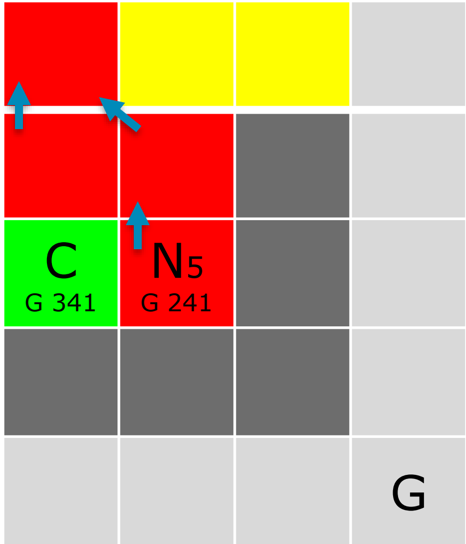

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



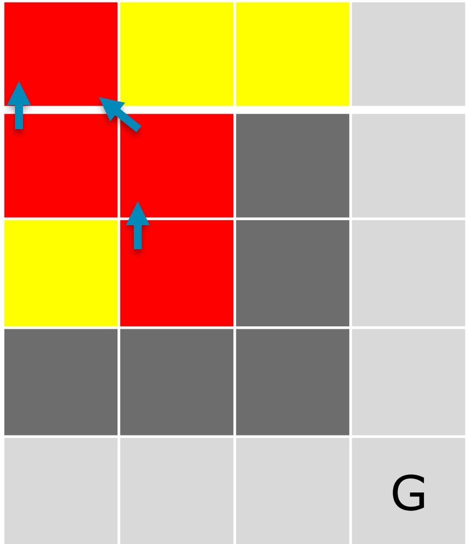

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



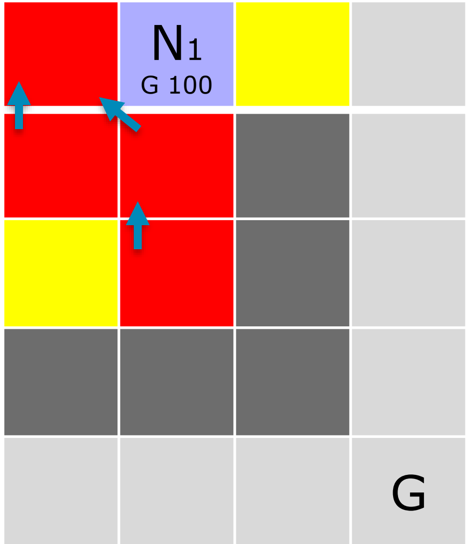

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



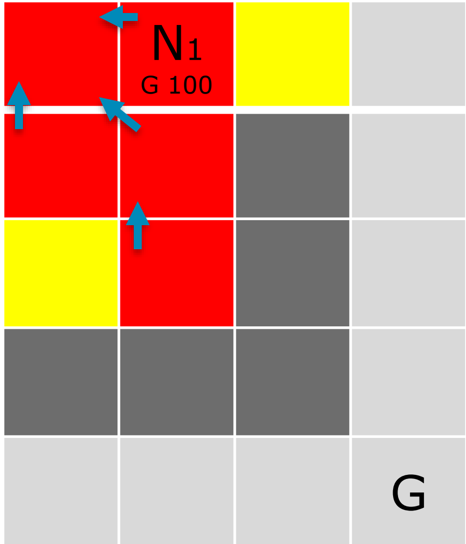

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



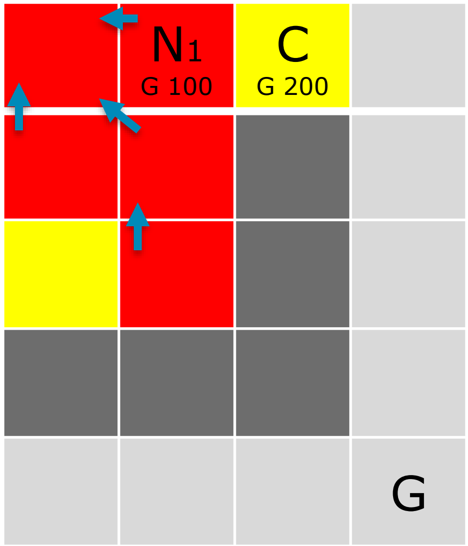

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



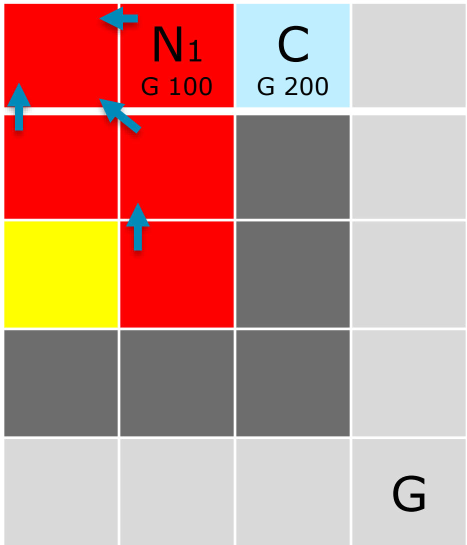

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



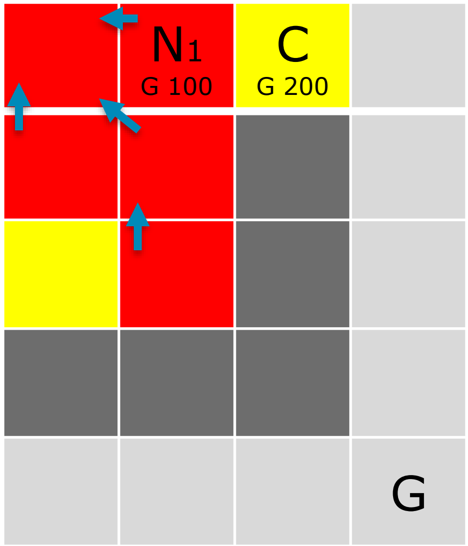

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



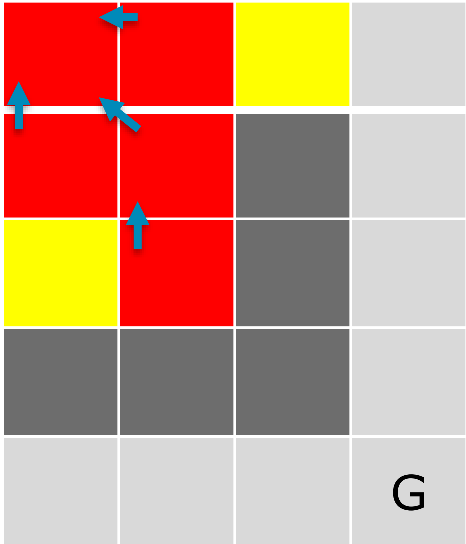

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



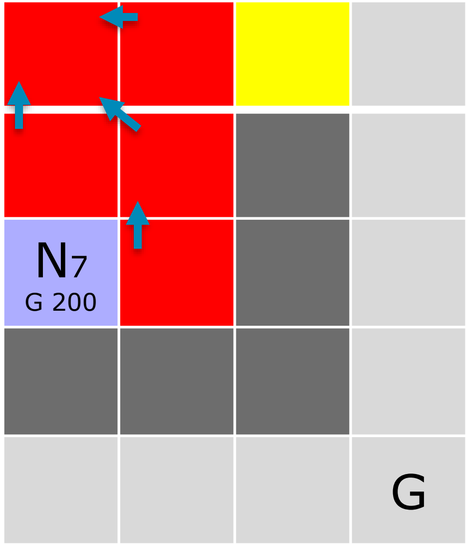

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



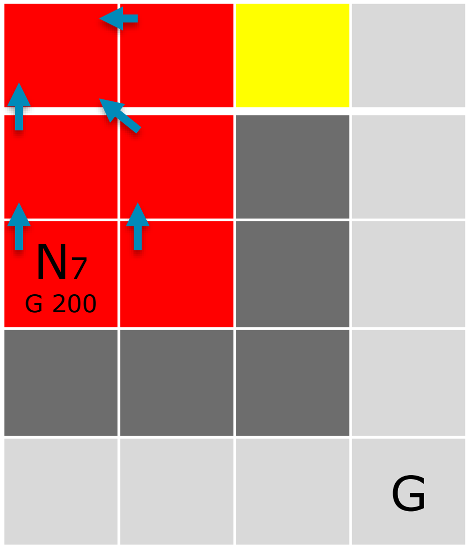

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



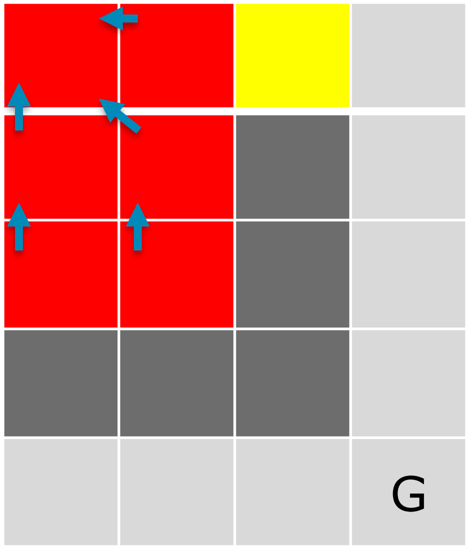

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



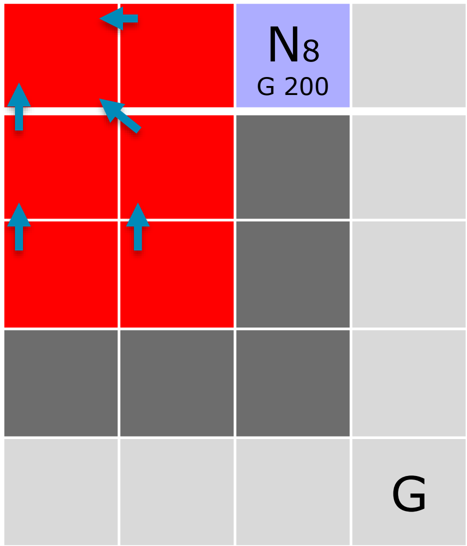

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



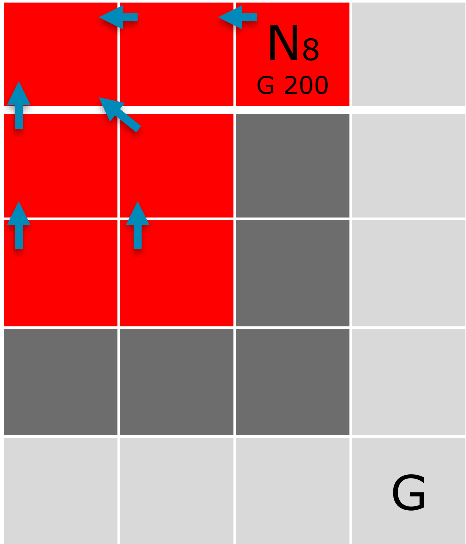

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



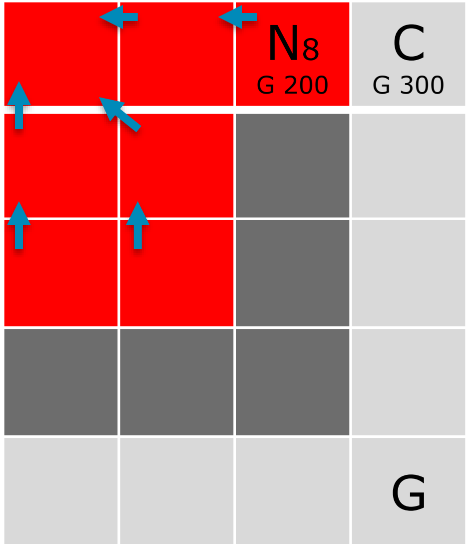

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



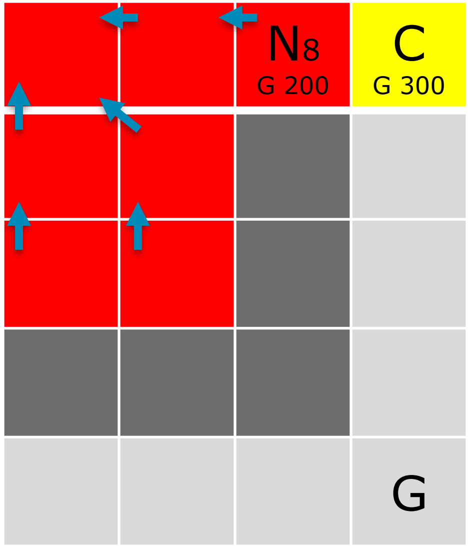

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



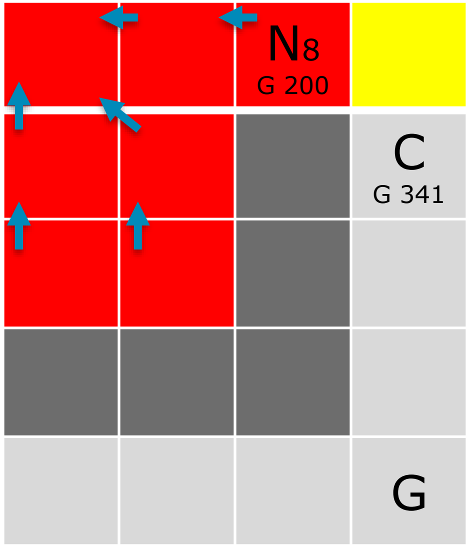

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



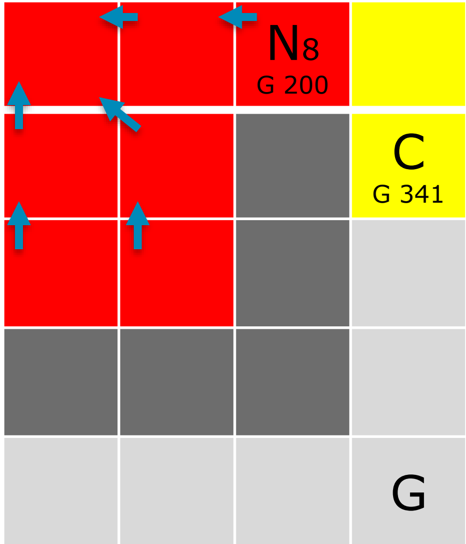

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



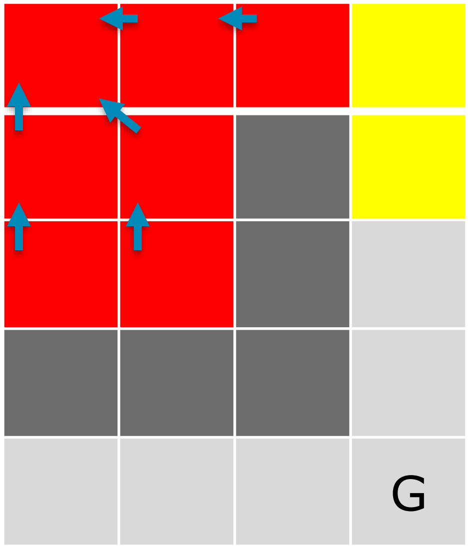

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



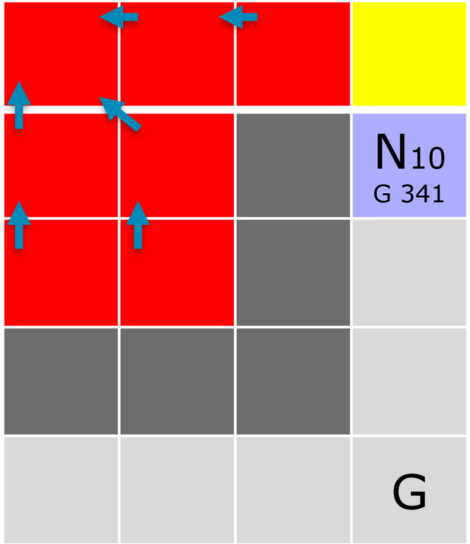

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



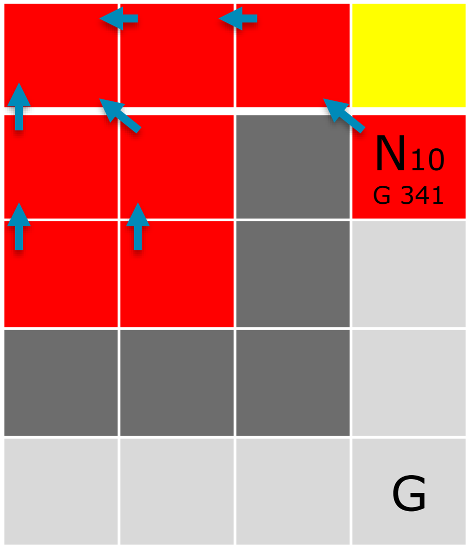

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



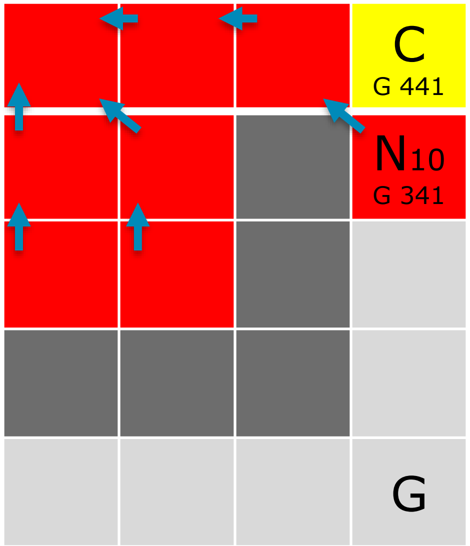

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



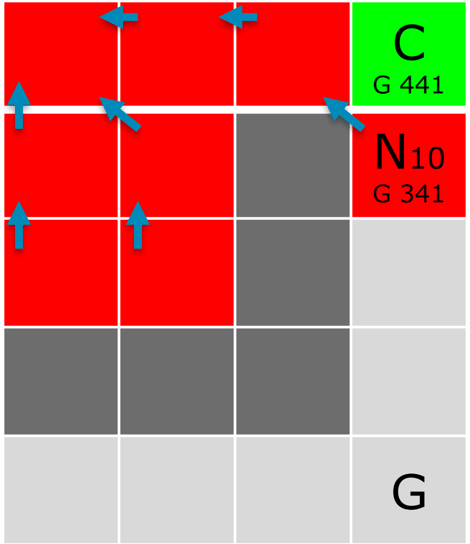

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



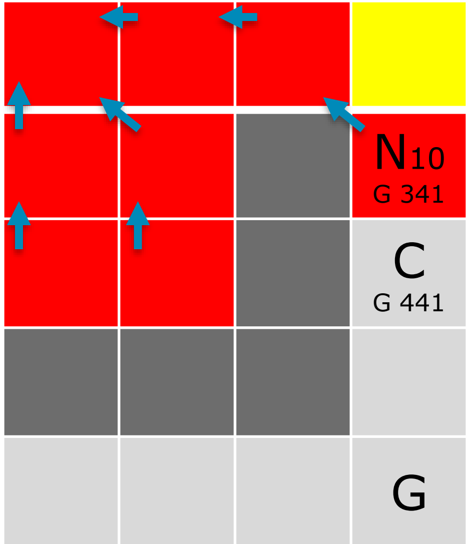

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



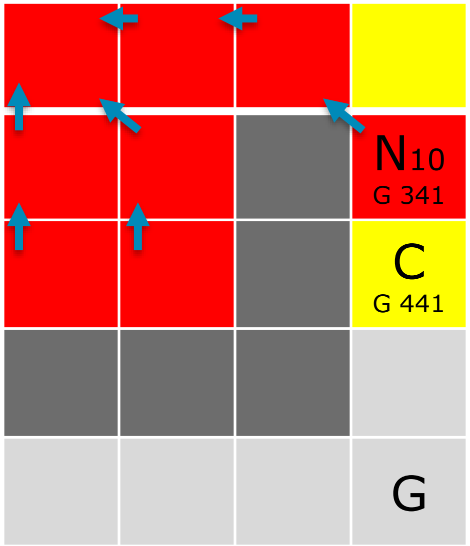

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



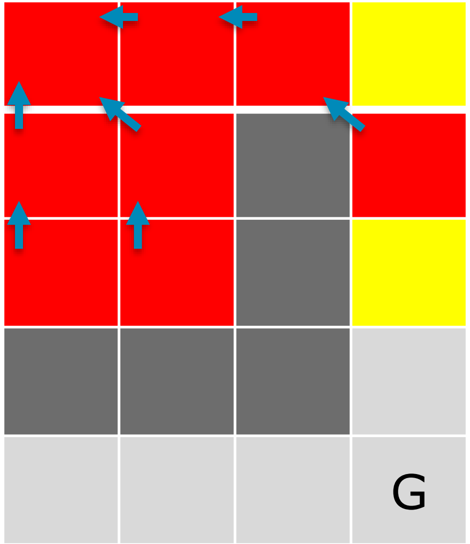
Open List (Nodes)

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

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```



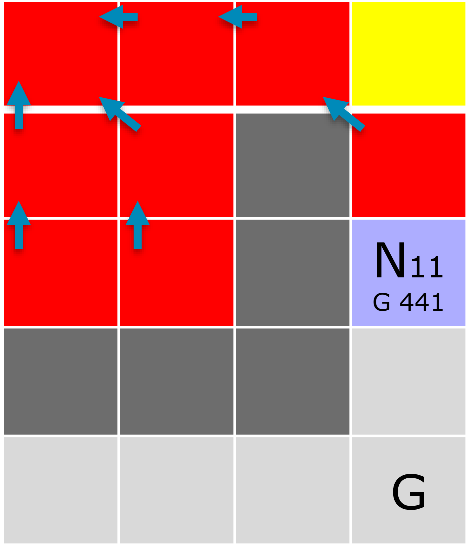

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



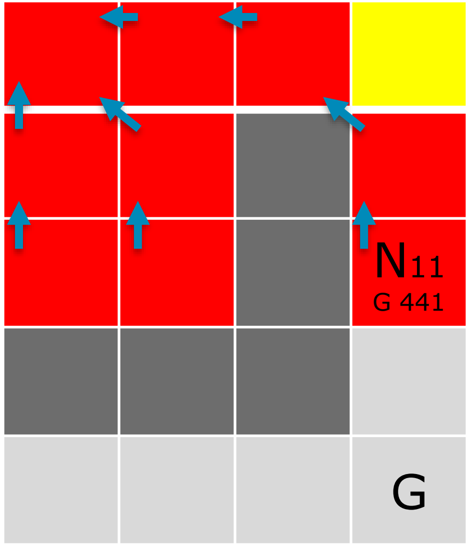

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



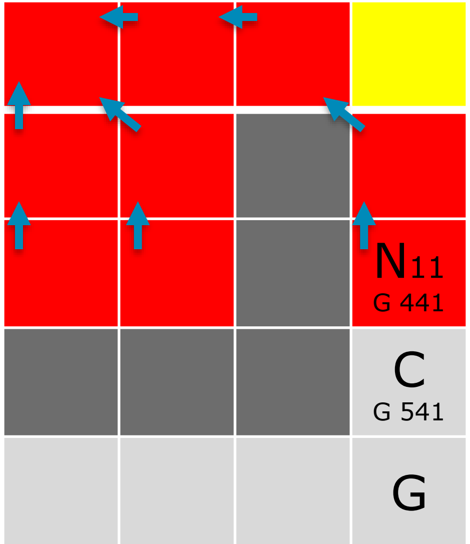

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



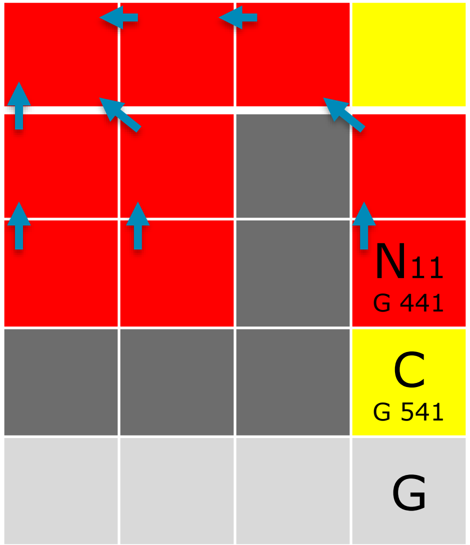

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



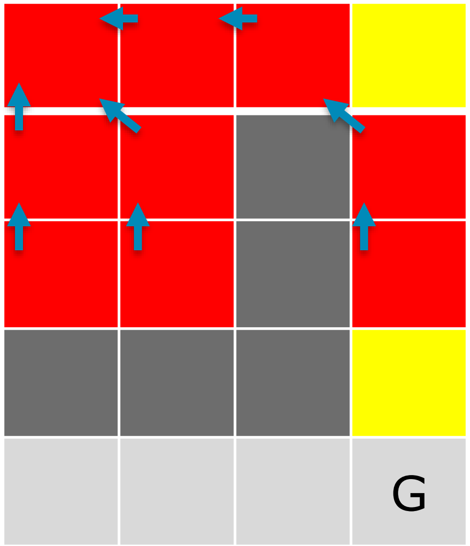

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



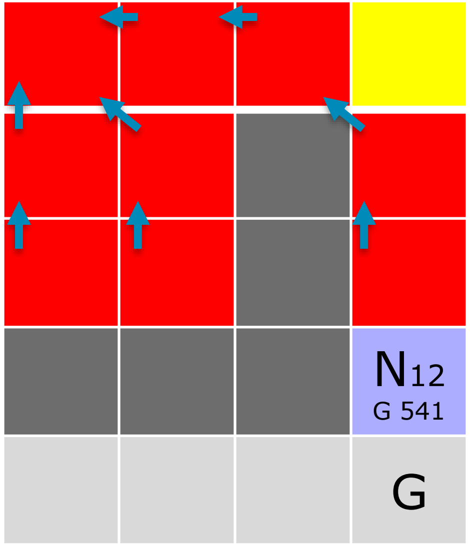

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



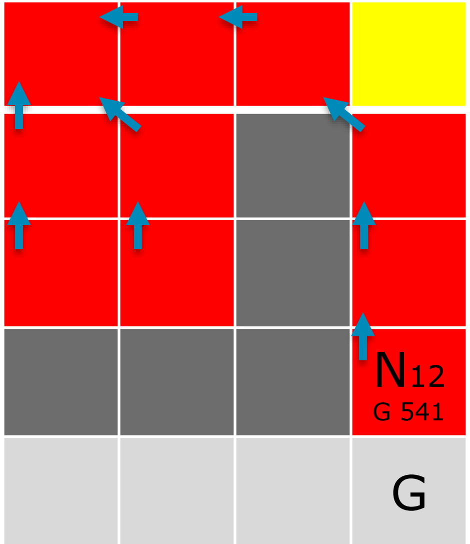

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



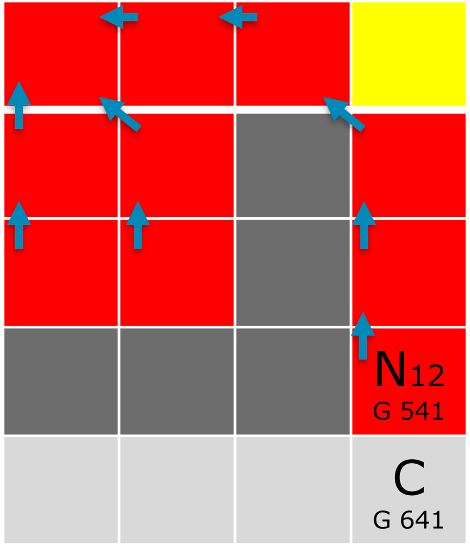

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



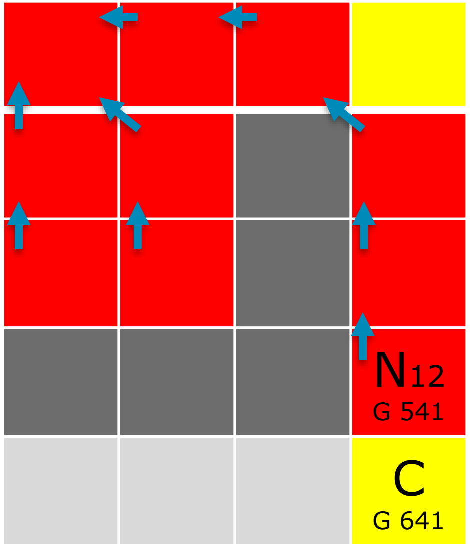

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



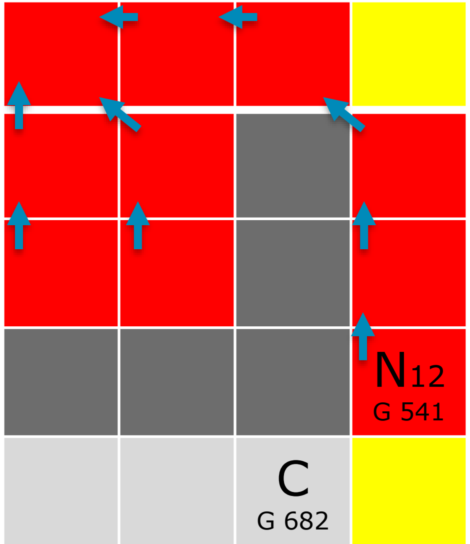

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



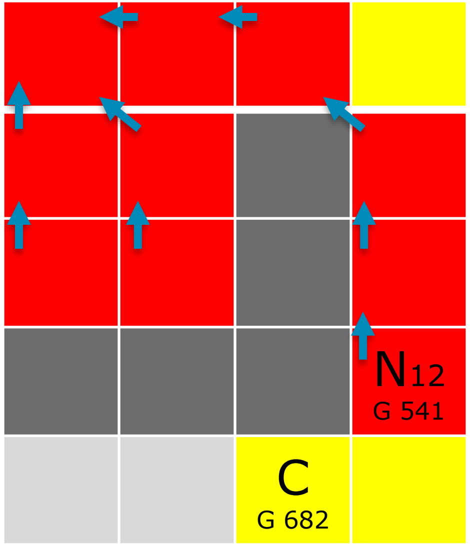

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



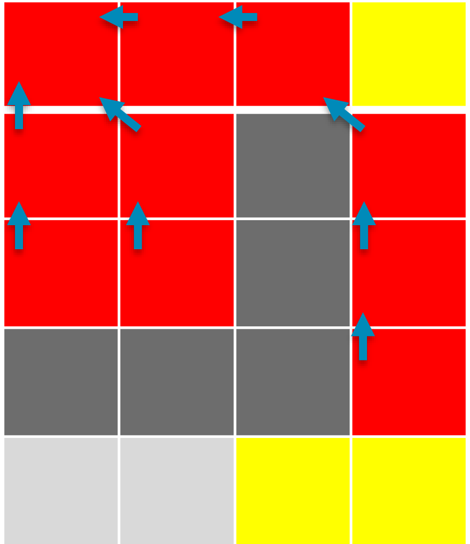

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



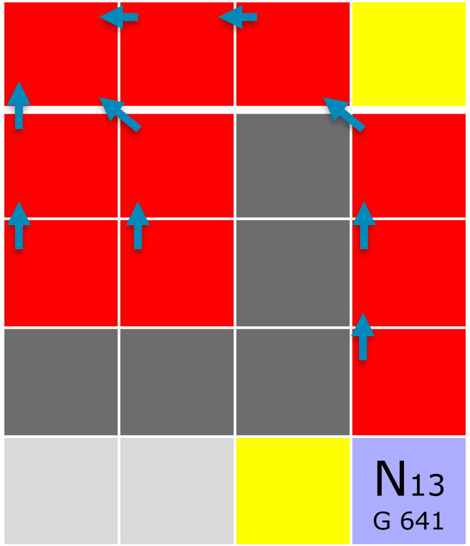
```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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



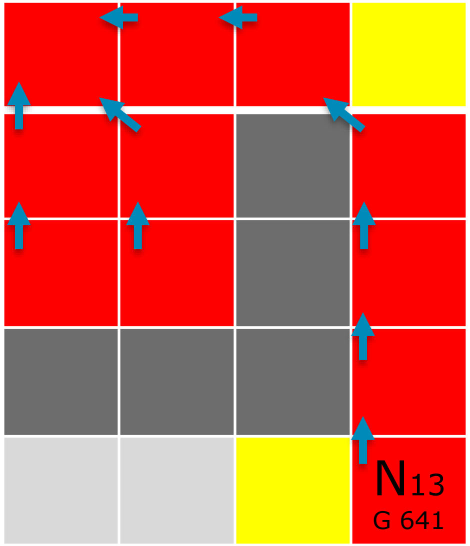

```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        else open.add(c)
```

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




```
while (!open.empty)
    node = remove min f from open
    if (node.state is goal) return path
    if (node.state in closed) continue
    add node to closed
    for c in expand(node)
        if c.state in closed: continue
        if c in open with <= g: continue
        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

Weighted A*

- We can tweak the performance of A* by adding a parameter weight, w
- We then multiply our heuristic by w
  - $f(n) = g(n) + w*h(n)$
- As we set the weight > 1, then A* favors going in the direction of the heuristic more and more
- If the weight is < 1, then A* trusts the heuristic less and less.
- What happens as w approaches 0 or infinity?

Weighted A*

A*	$f(n) =$	$g(n)$	$+$	$h(n)$
UCS	$f(n) =$	$g(n)$	$+$	0
GBeFS	$f(n) =$	0	$+$	$h(n)$
WA*	$f(n) =$	$g(n)$	$+$	$h(n)*w$

UCS ———————————————— GBeFS

$w=0$ WA* $w=\infty$

Search Demo