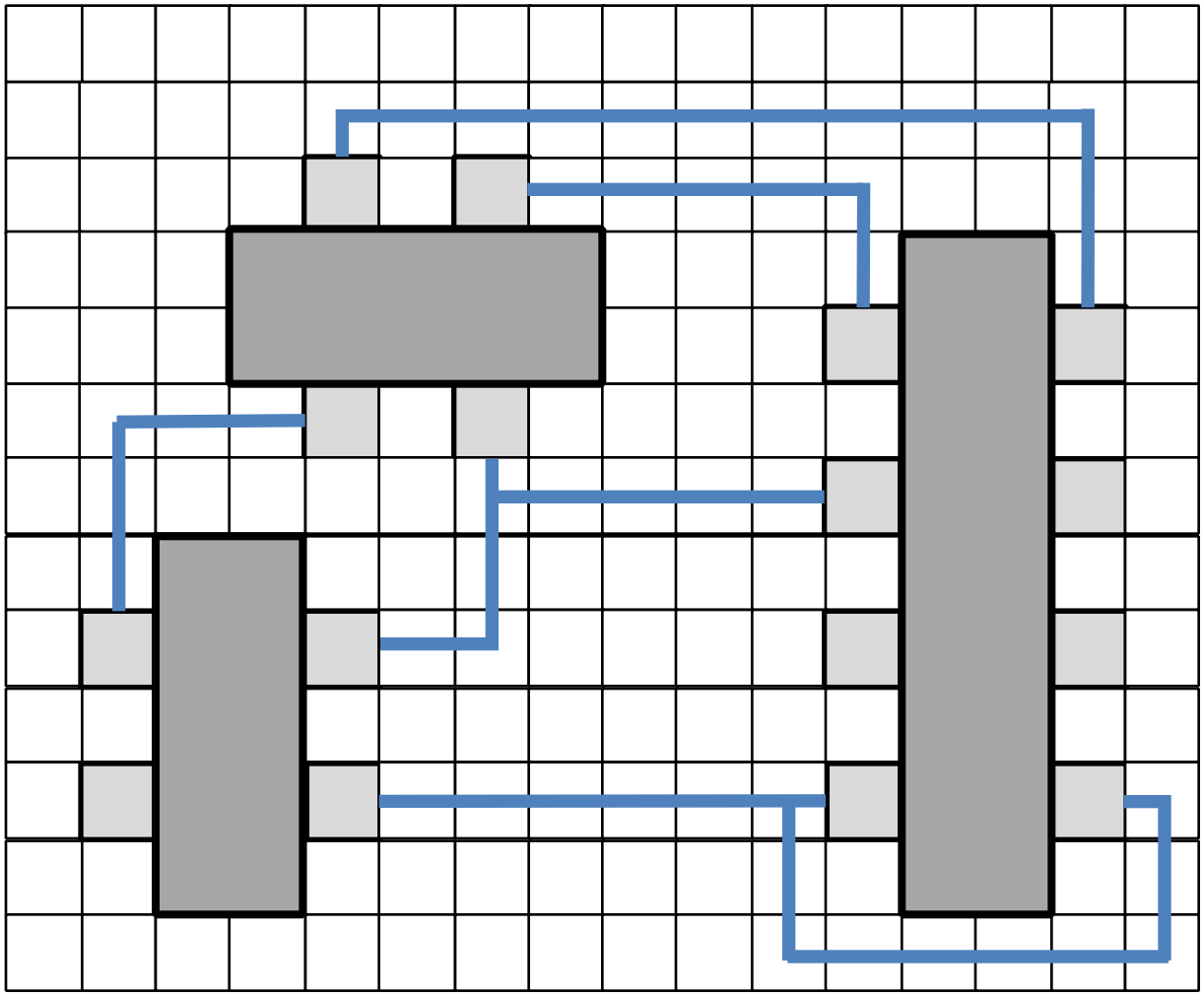
Grid Routing Algorithms

Dr. Philip Brisk
Department of Computer Science and Engineering
University of California, Riverside

CS 220
Synthesis of Digital Systems

Example

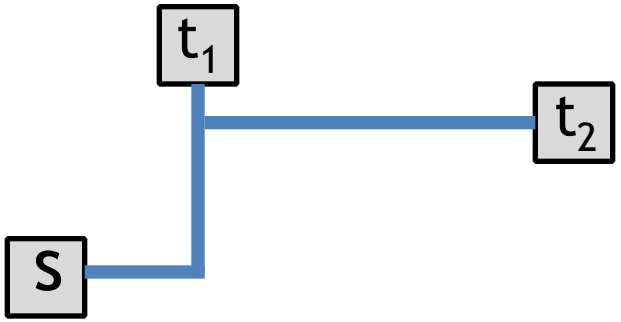


Problem Formulation

- MxN Grid
  - Some sub-regions may be blocked off
- Set of nets to route
  - Nets may have more than two terminals
- Constraints
  - Spacing rules between routed nets
  - Are intersections allowed?
    - No for electrical signals (short-circuit)
    - Yes/No/Sometimes for microfluidics
    - Paths may not lie on top of one another

Typical Nomenclature for a Net

- N = (s, T)
  - s: source (starting point of the search)
  - T: set of sinks
  - The search must compute a route from s to each sink t_i in T
  - Shared segments along routes are allowed

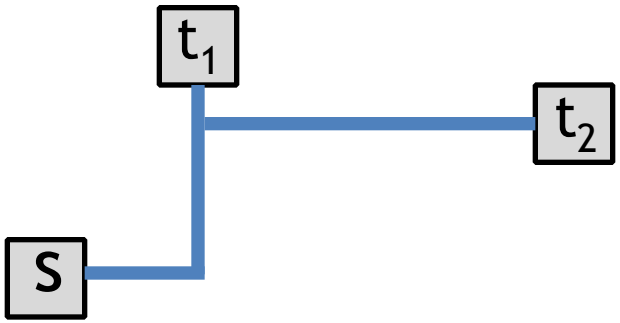


Overview of Techniques

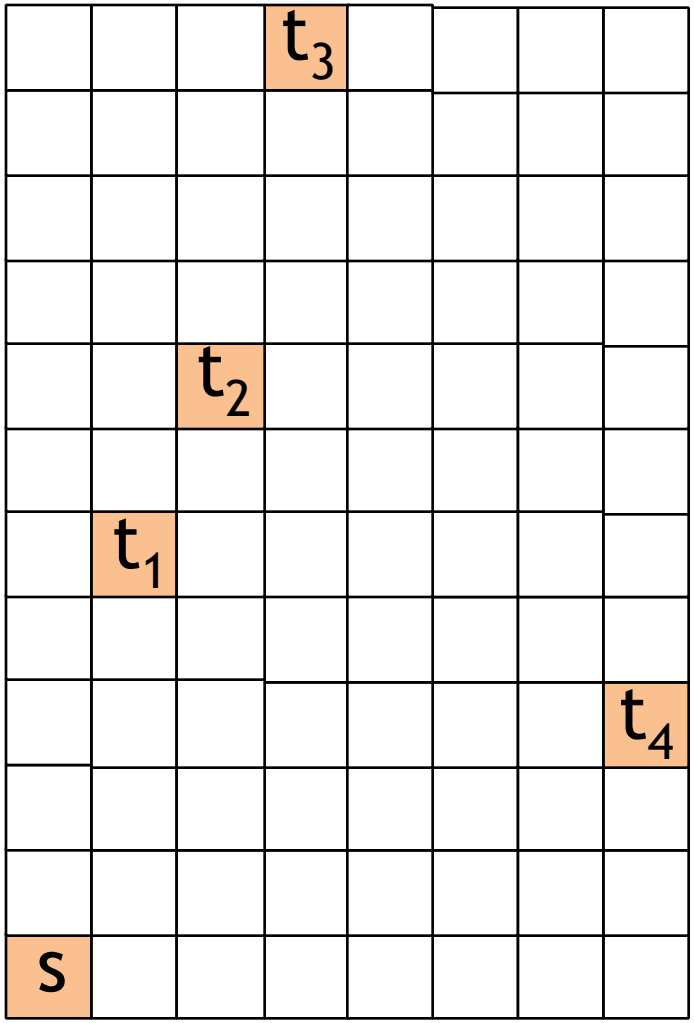
- Search techniques
  - Breadth First Search
  - Depth First Search + Detour (For Obstacles)
  - Directed Search
    - Try to combine the benefits for Breadth and Depth-First Search
  - Line Search

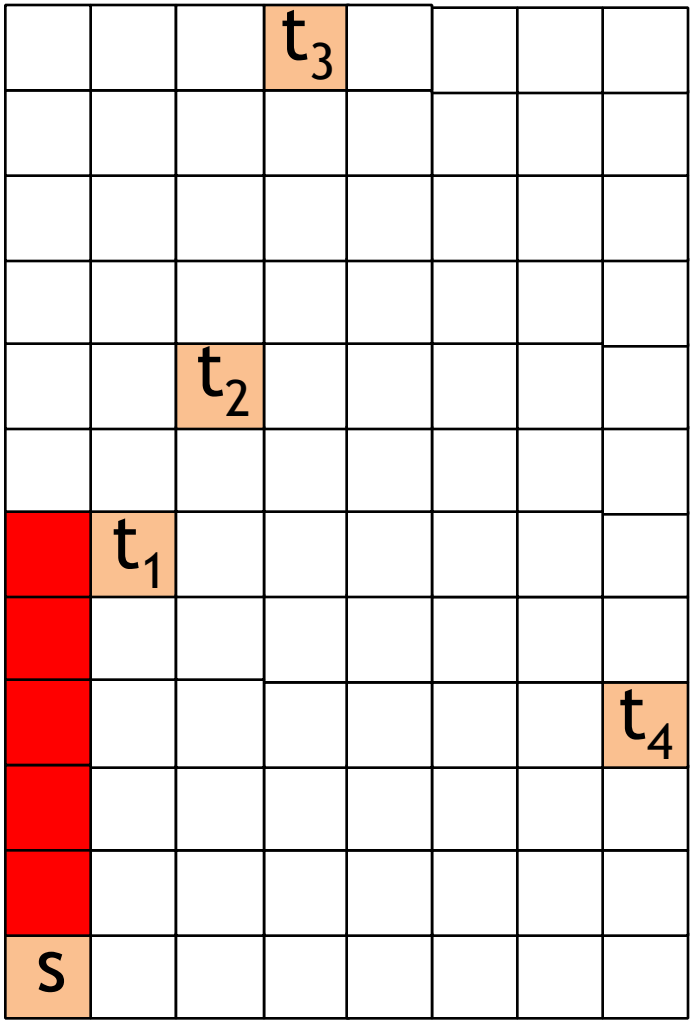
Routing Multi-Terminal Nets

1. Route from s to the closest sink t_i
2. All cells on the path from s to t_i become new sources
3. Continue the search starting from all sources until the next sink t_j is discovered
4. Stop when sinks are discovered
5. (Optional) Post-processing refinement to reduce the size/cost of the routing tree

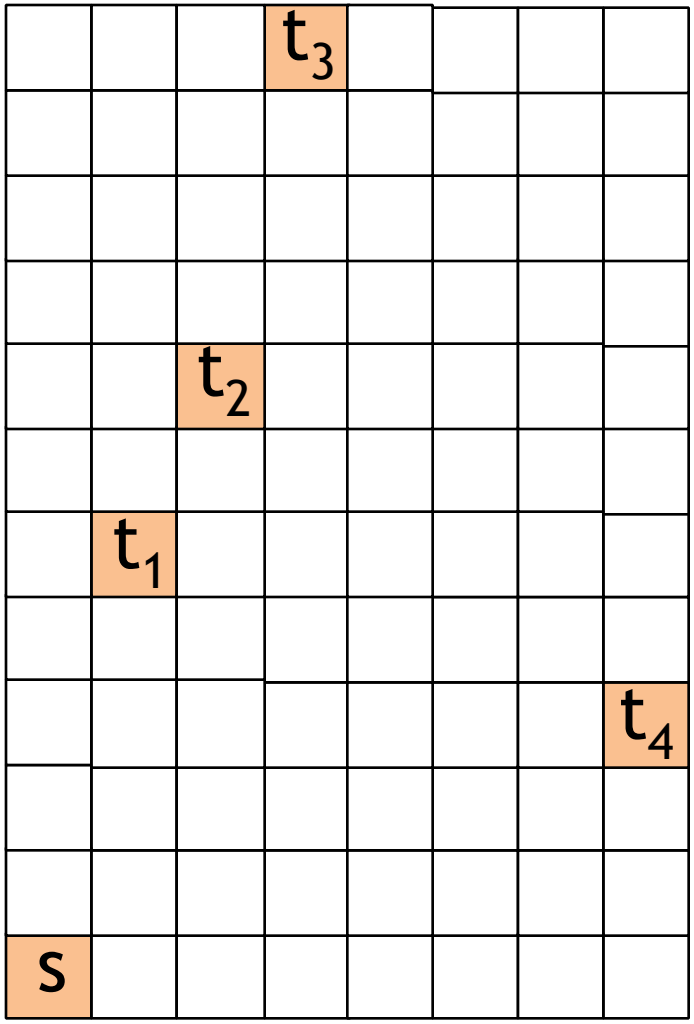


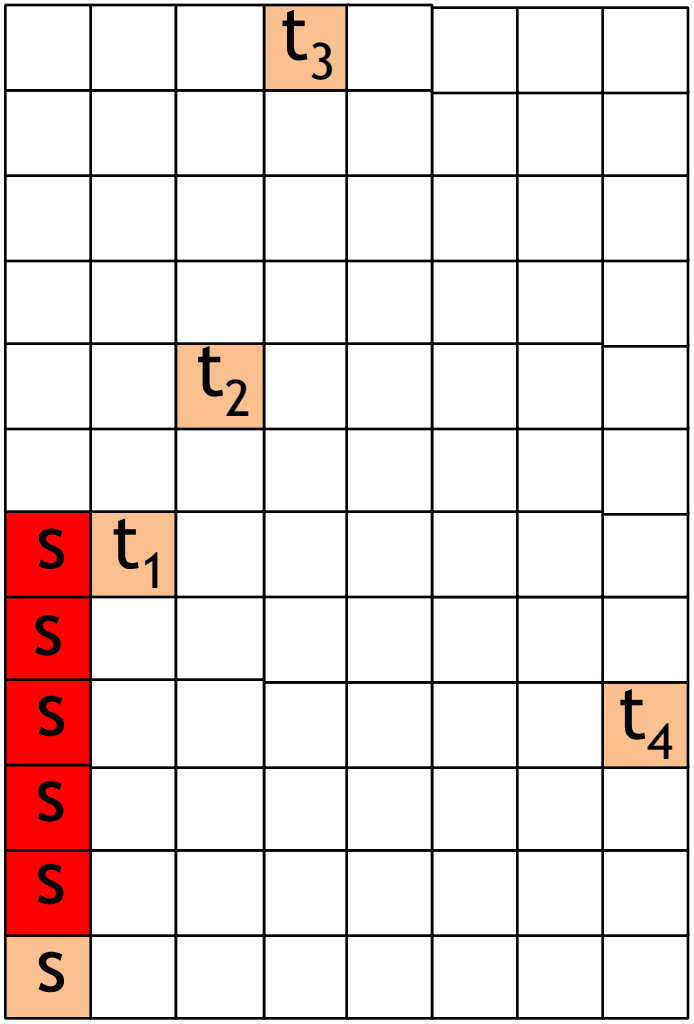
Find a Path to the First Sink



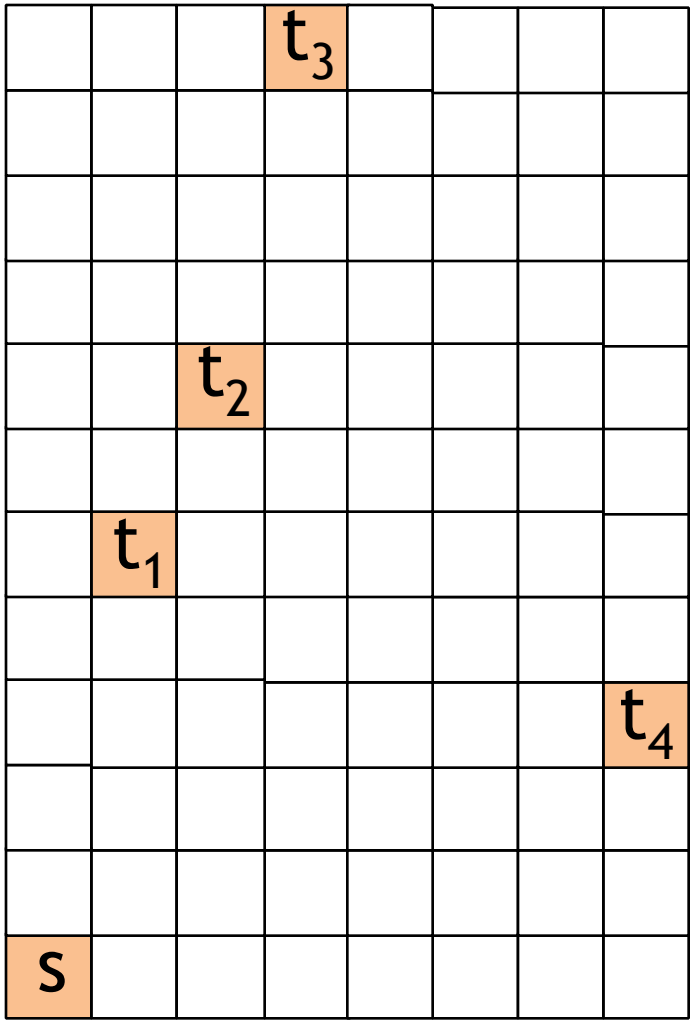


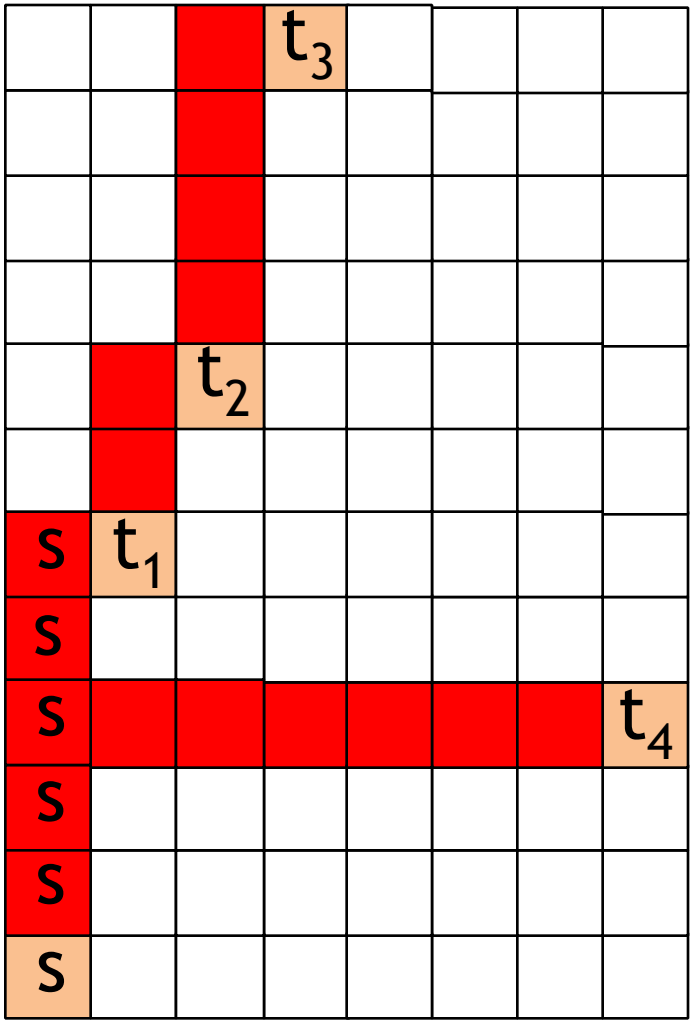
All Vertices on the Path Become Sources



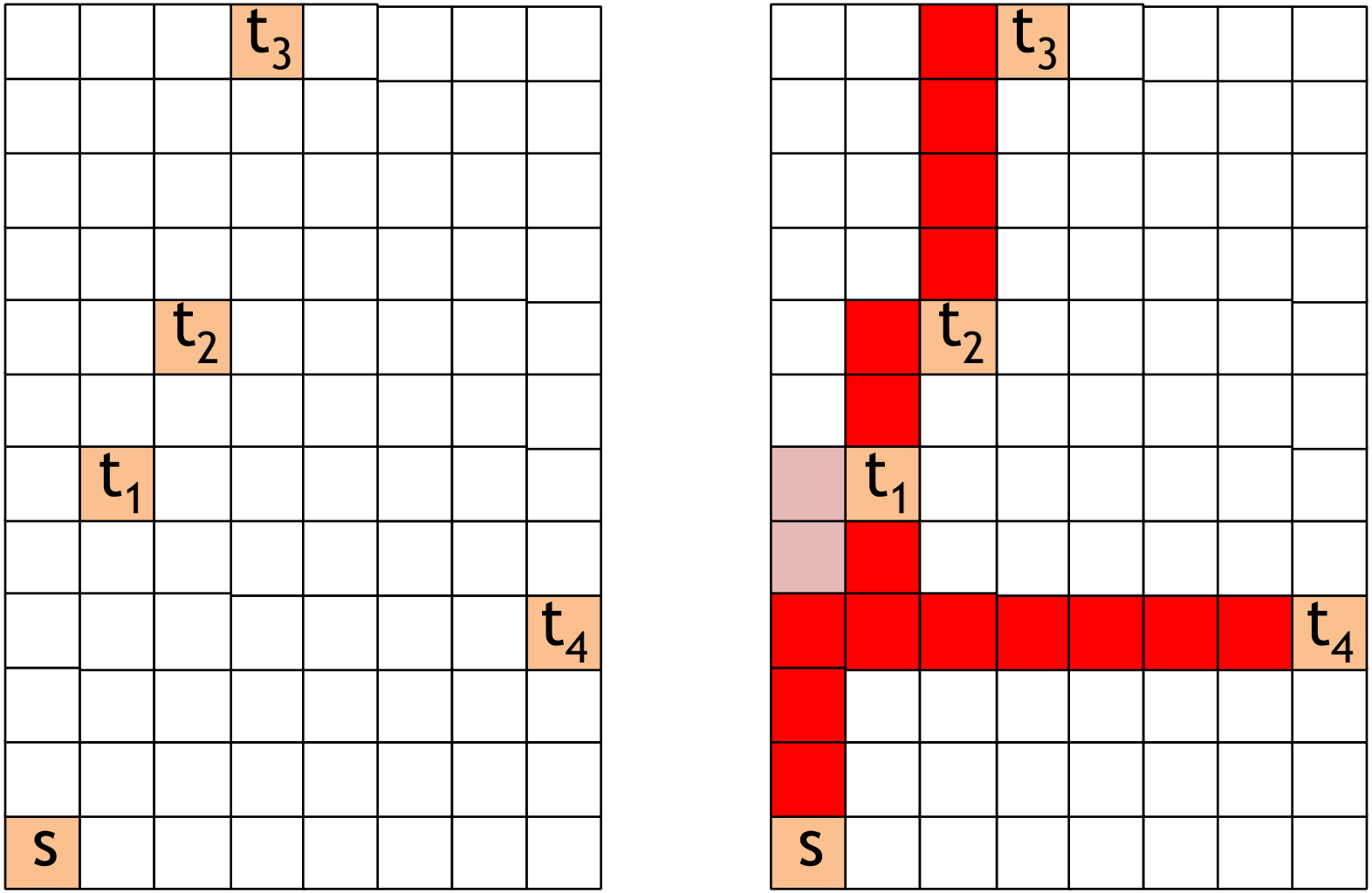


Find Paths to All Sinks





Refine the Routing Tree



Lee's Algorithm

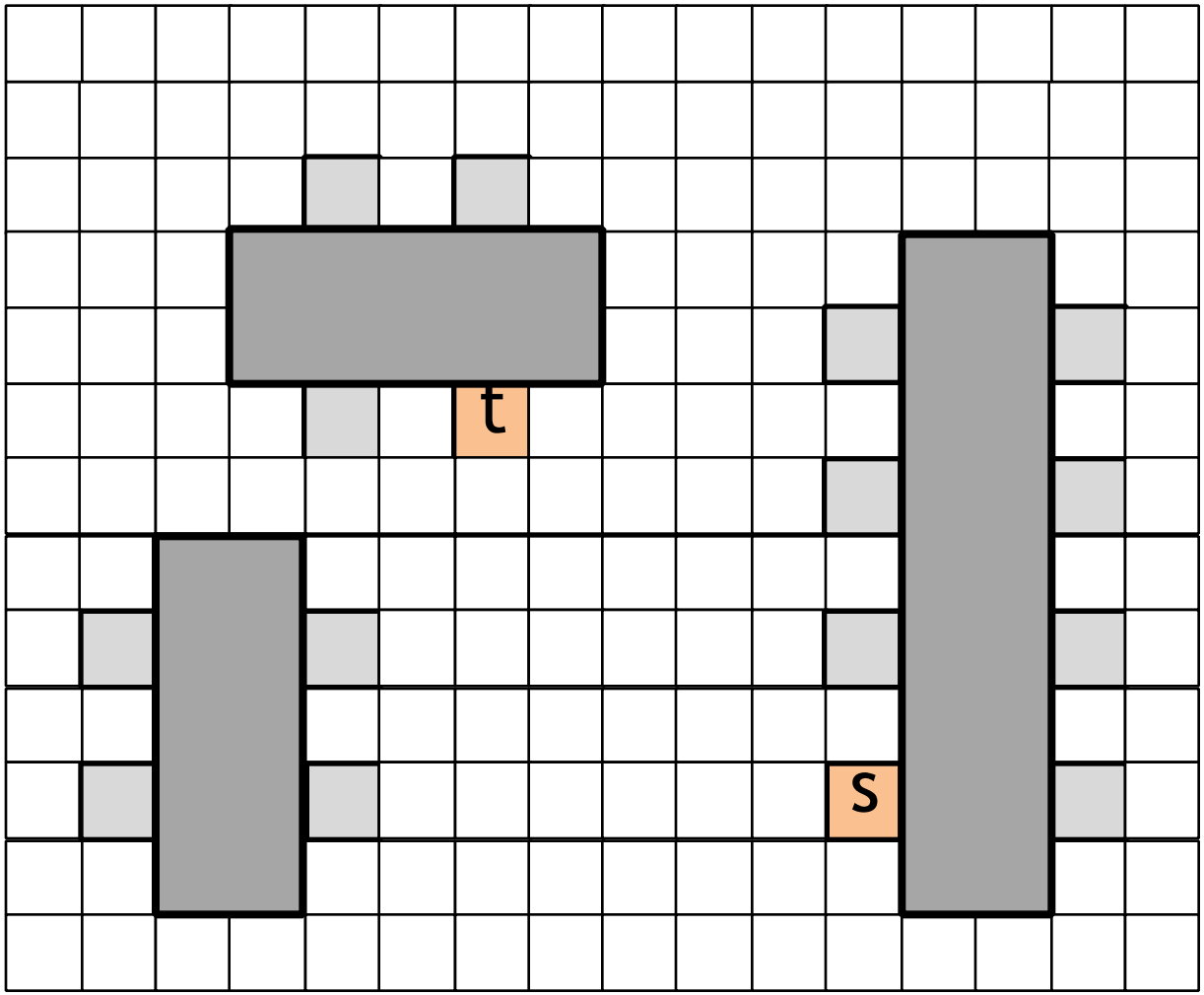
Lee, C. Y. (1961), *An Algorithm for Path Connections and Its Applications*, *IRE Transactions on Electronic Computers*, EC-10 (2): 346–365

Lee's Algorithm

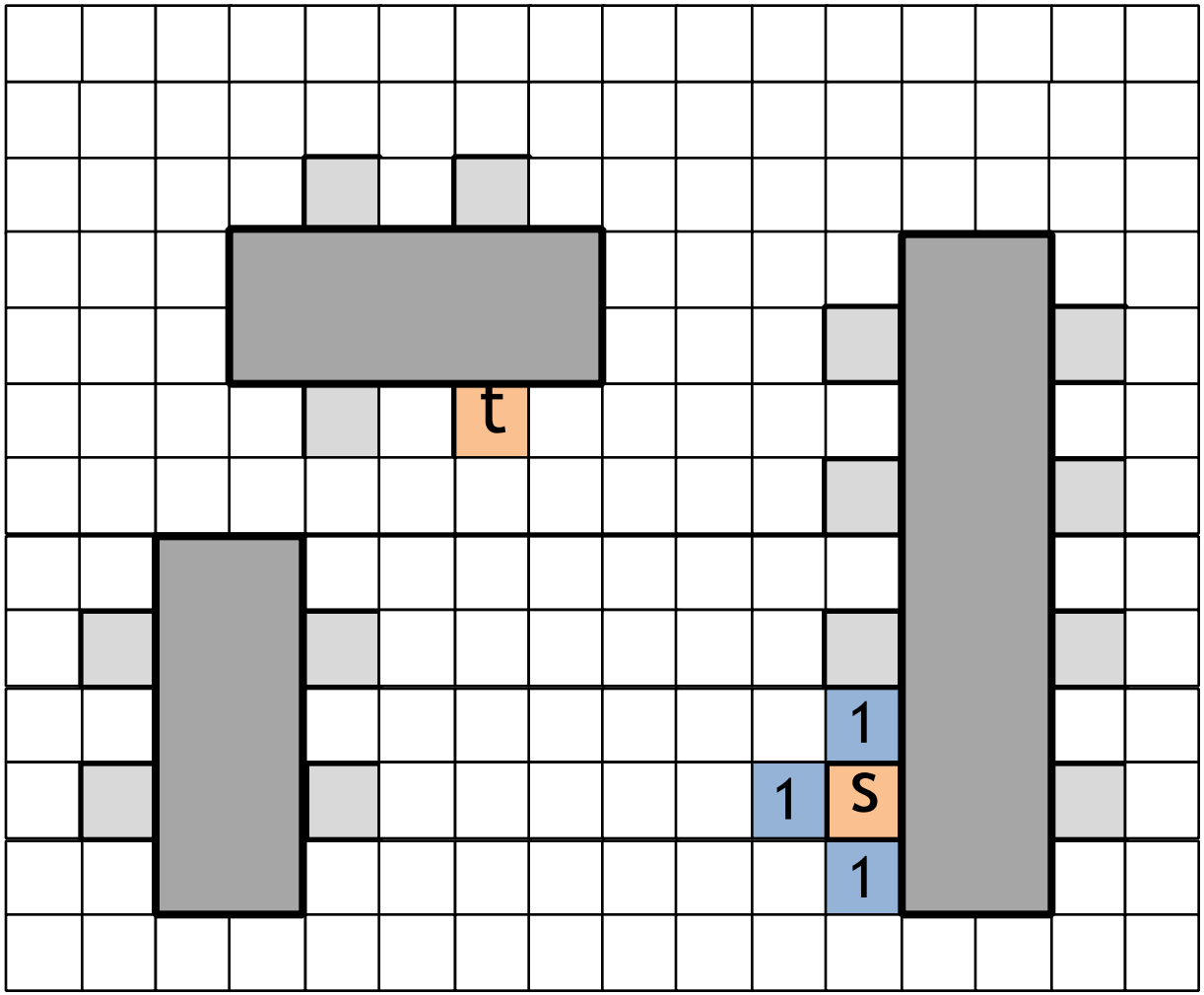
1. Breadth first search in a 2D grid
2. Backtrace to construct the path

- O(mn) time complexity
- Finds the shortest path for each net
  - (Given blockages due to previous nets)

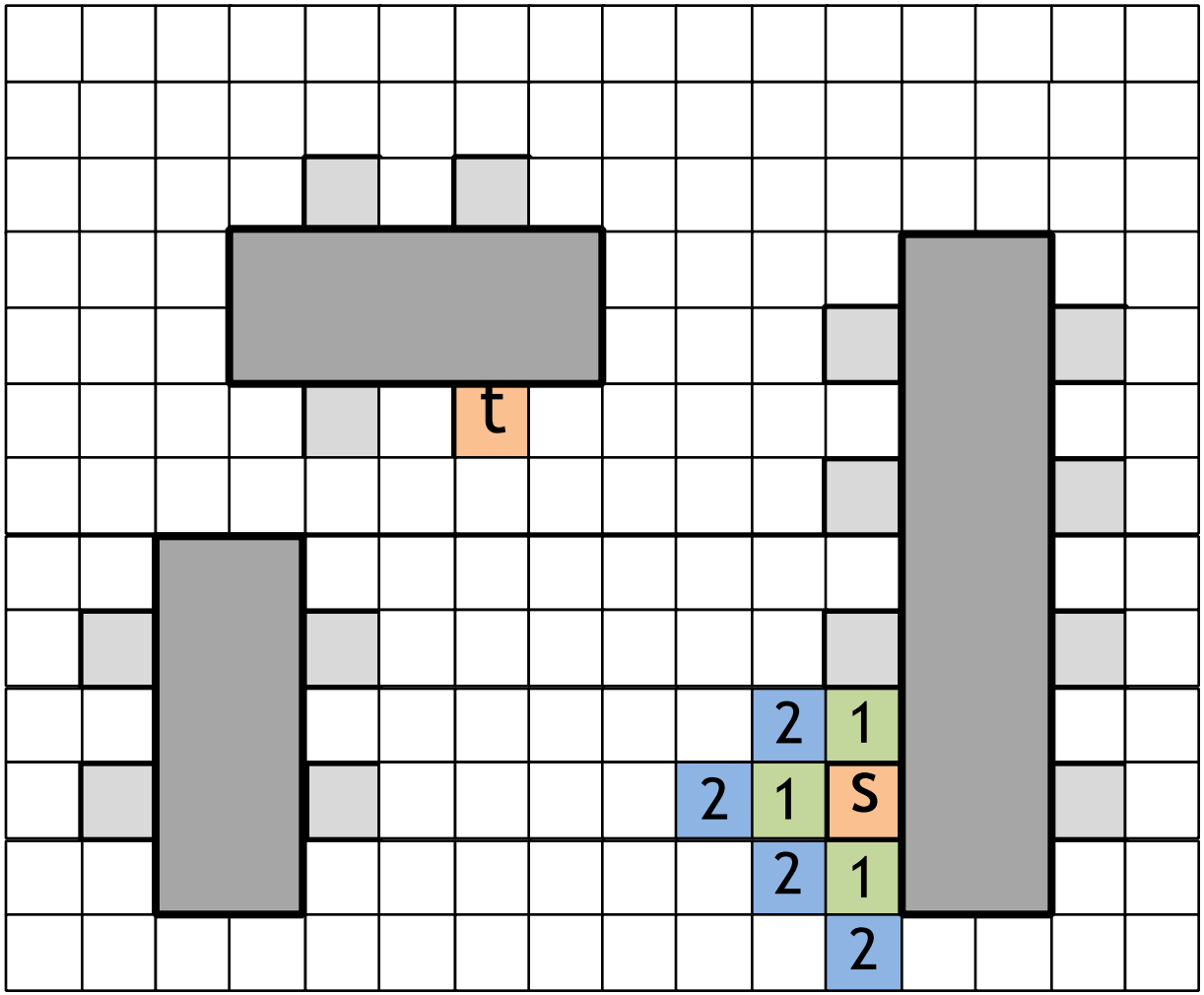
Example



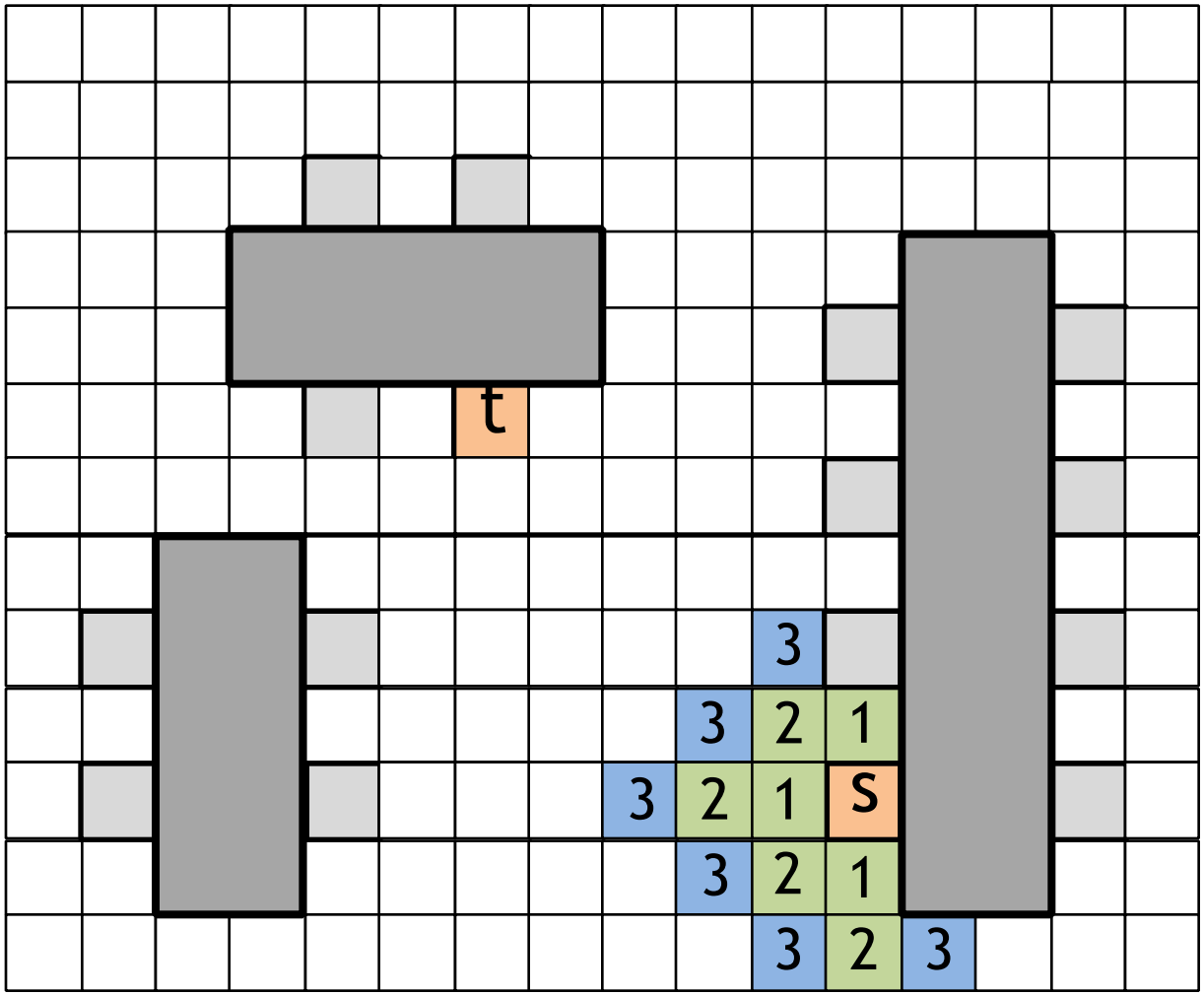
Wavefront Expansion



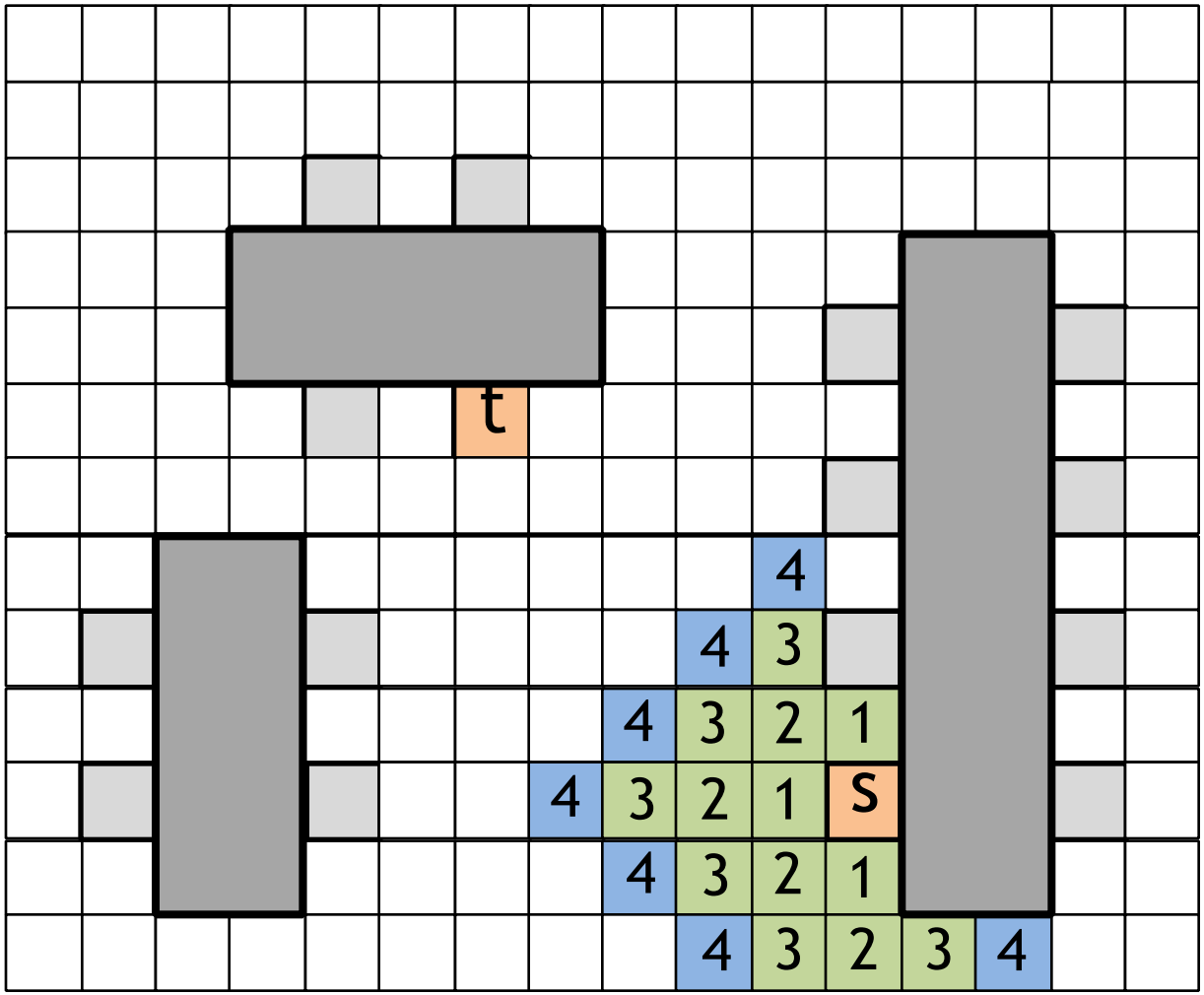
Wavefront Expansion



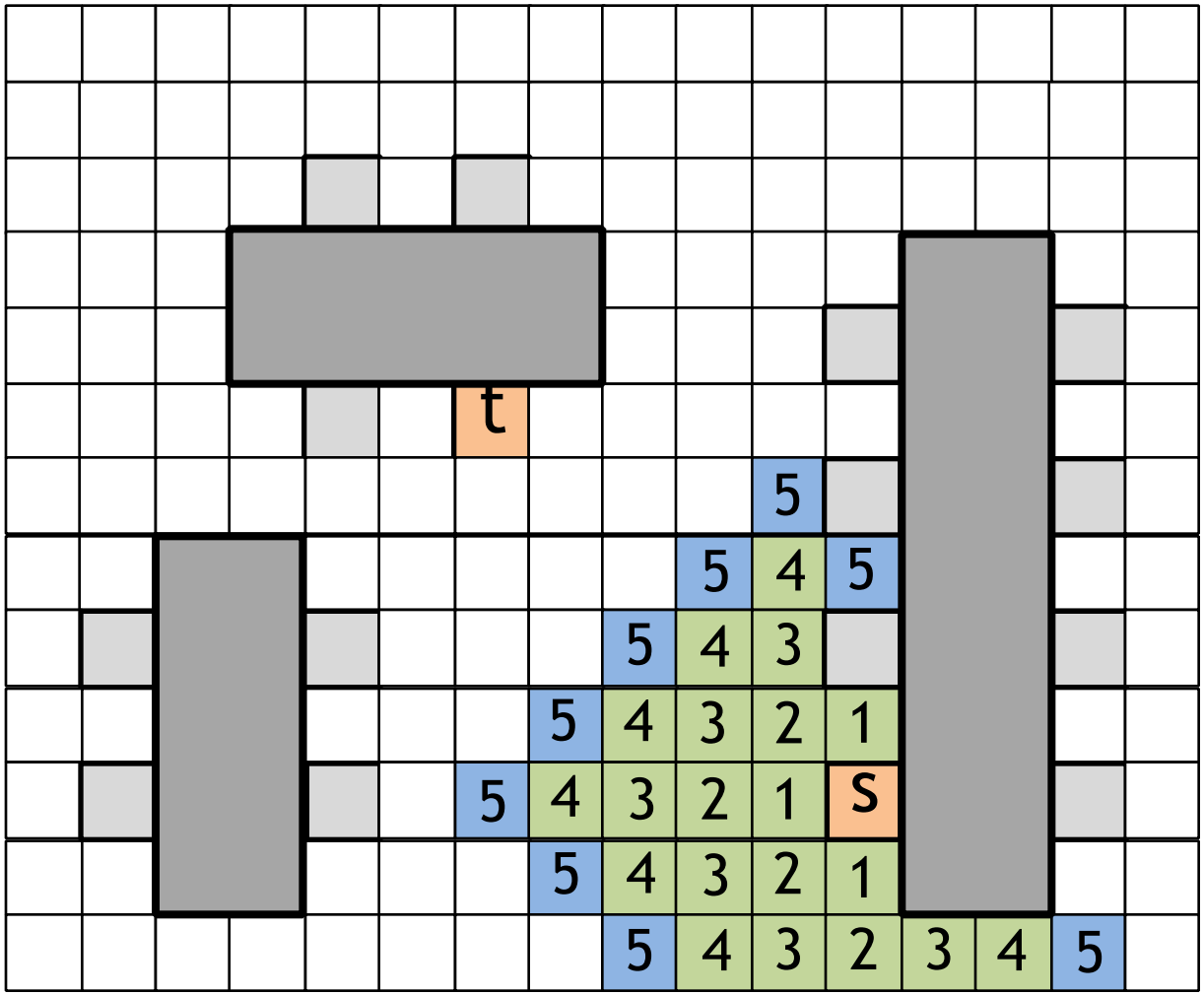
Wavefront Expansion



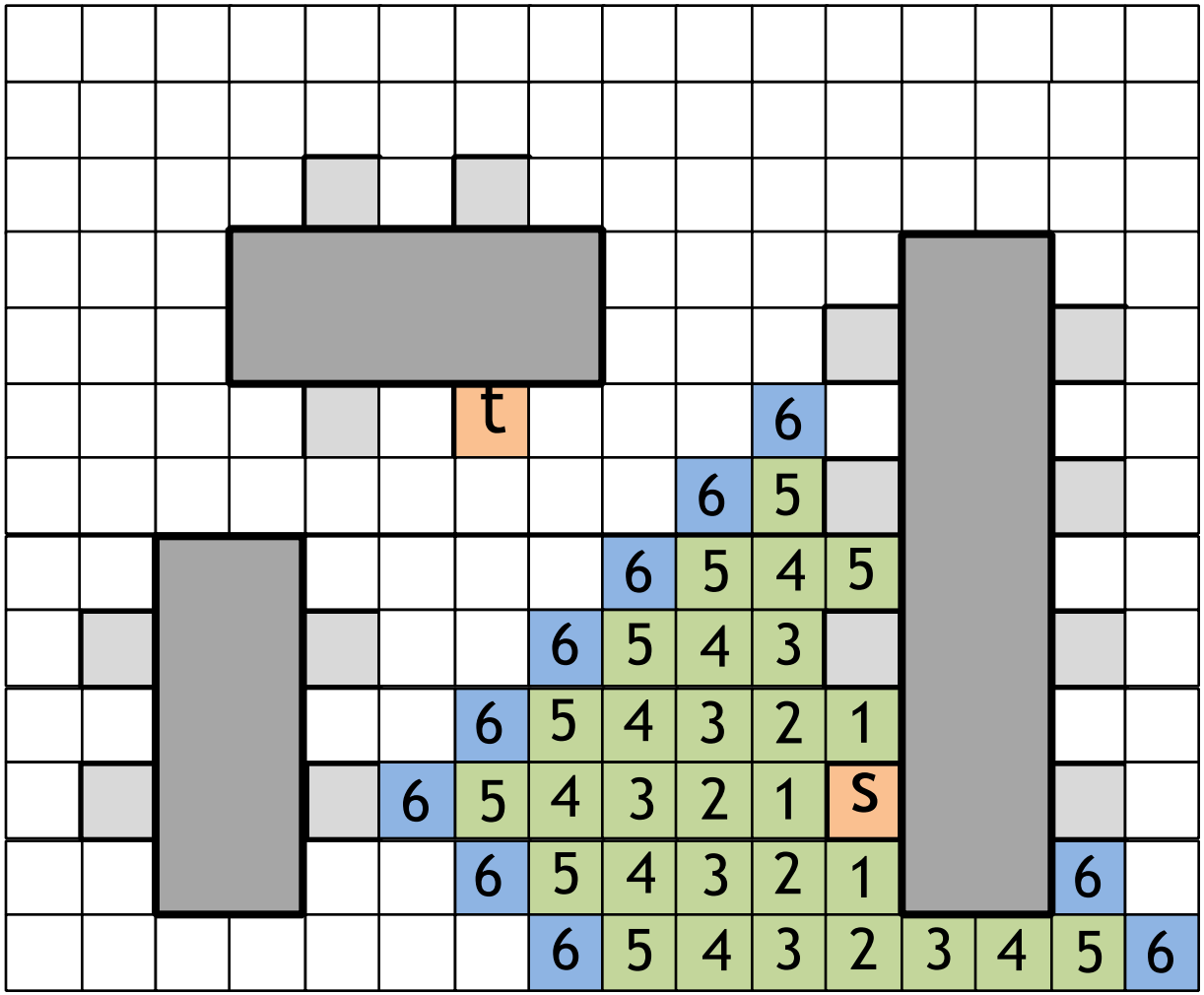
Wavefront Expansion



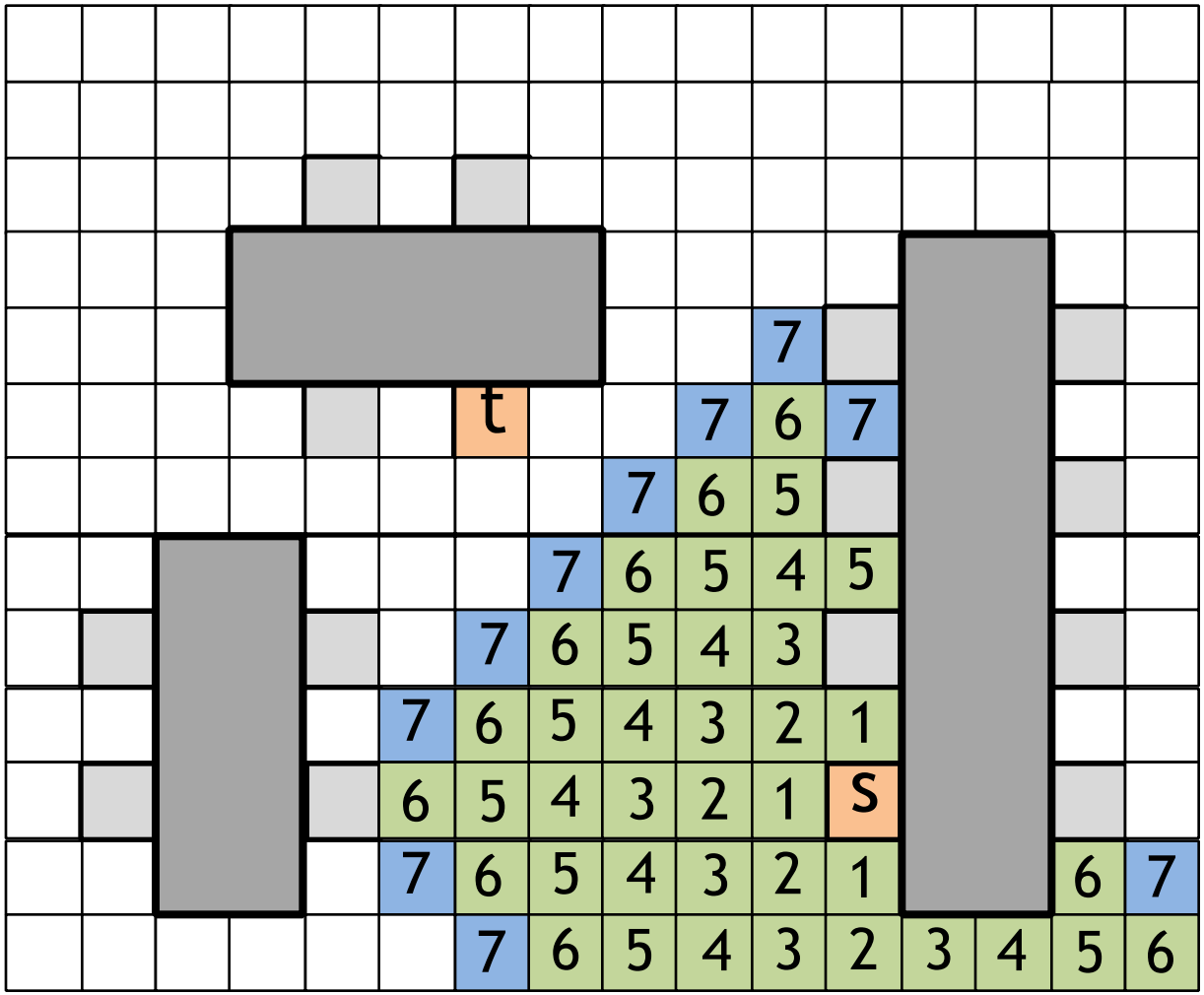
Wavefront Expansion



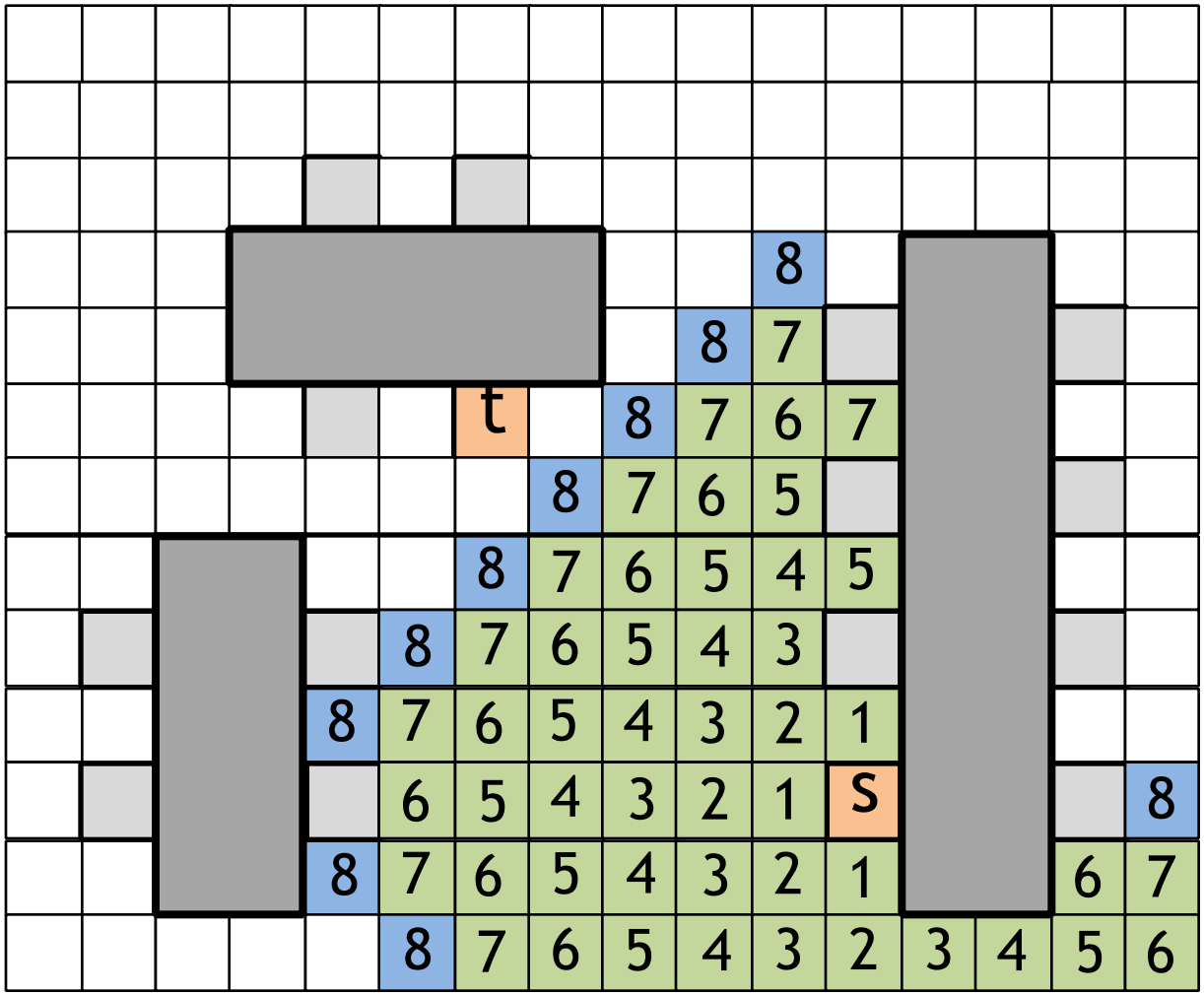
Wavefront Expansion



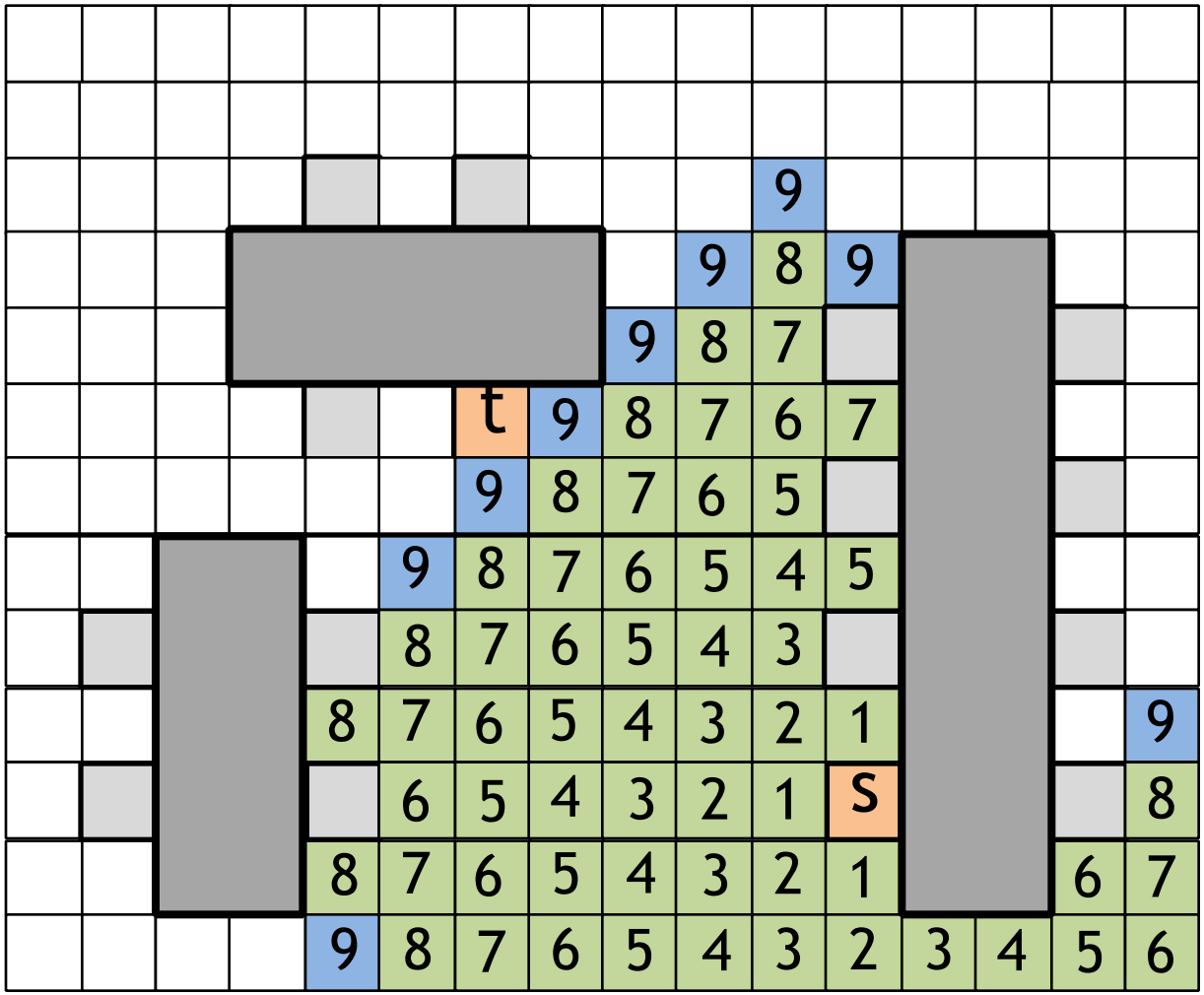
Wavefront Expansion



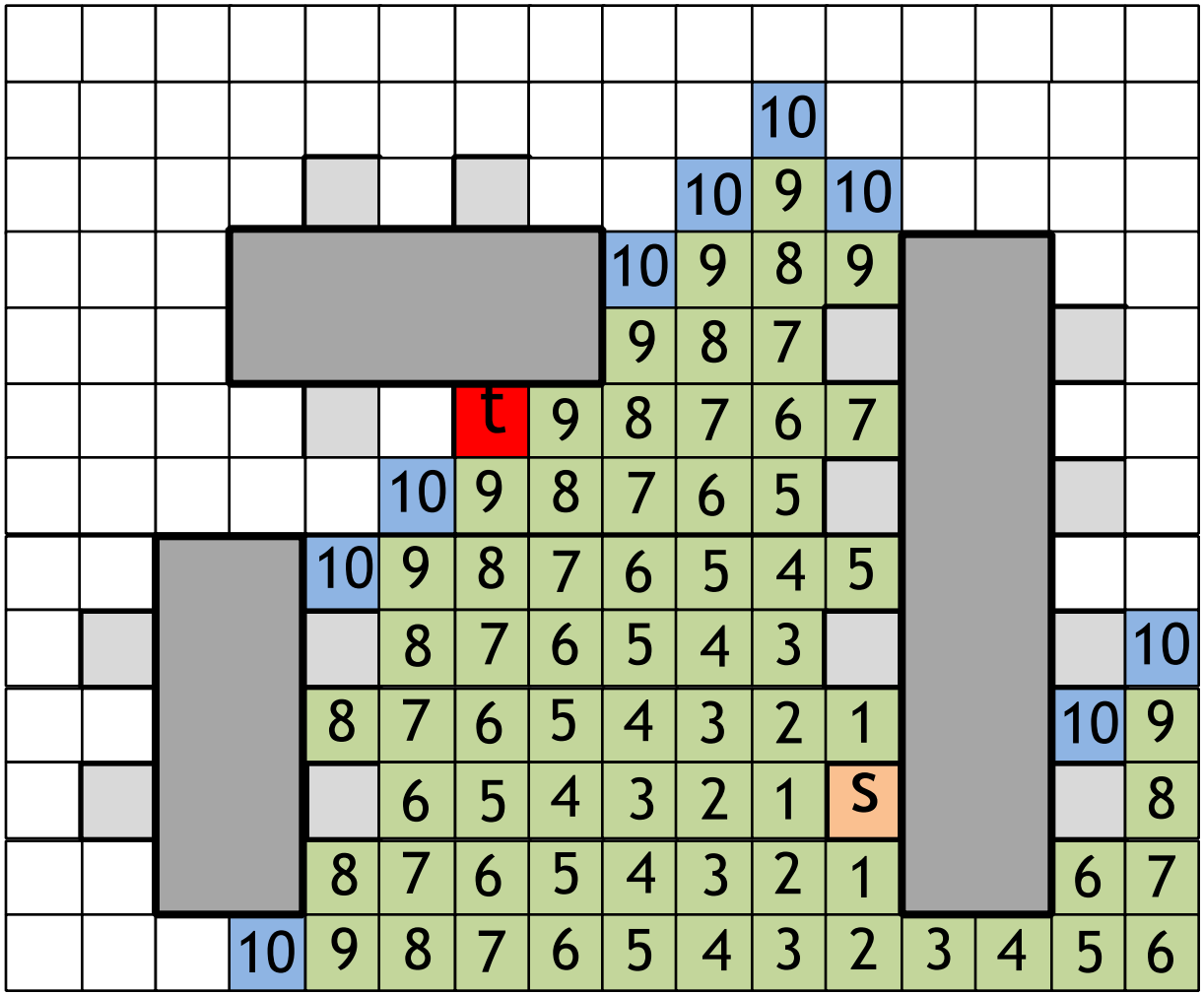
Wavefront Expansion



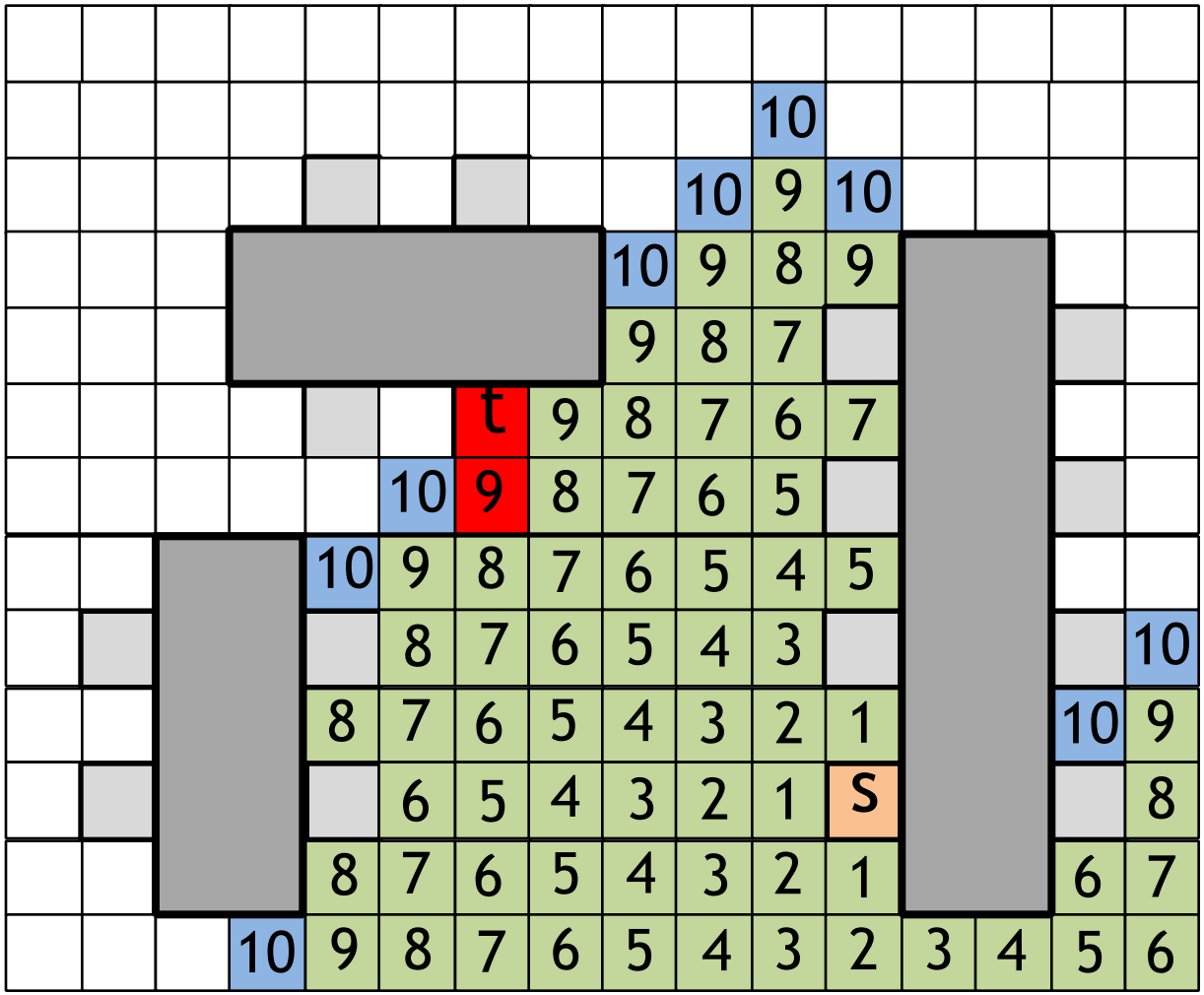
Wavefront Expansion



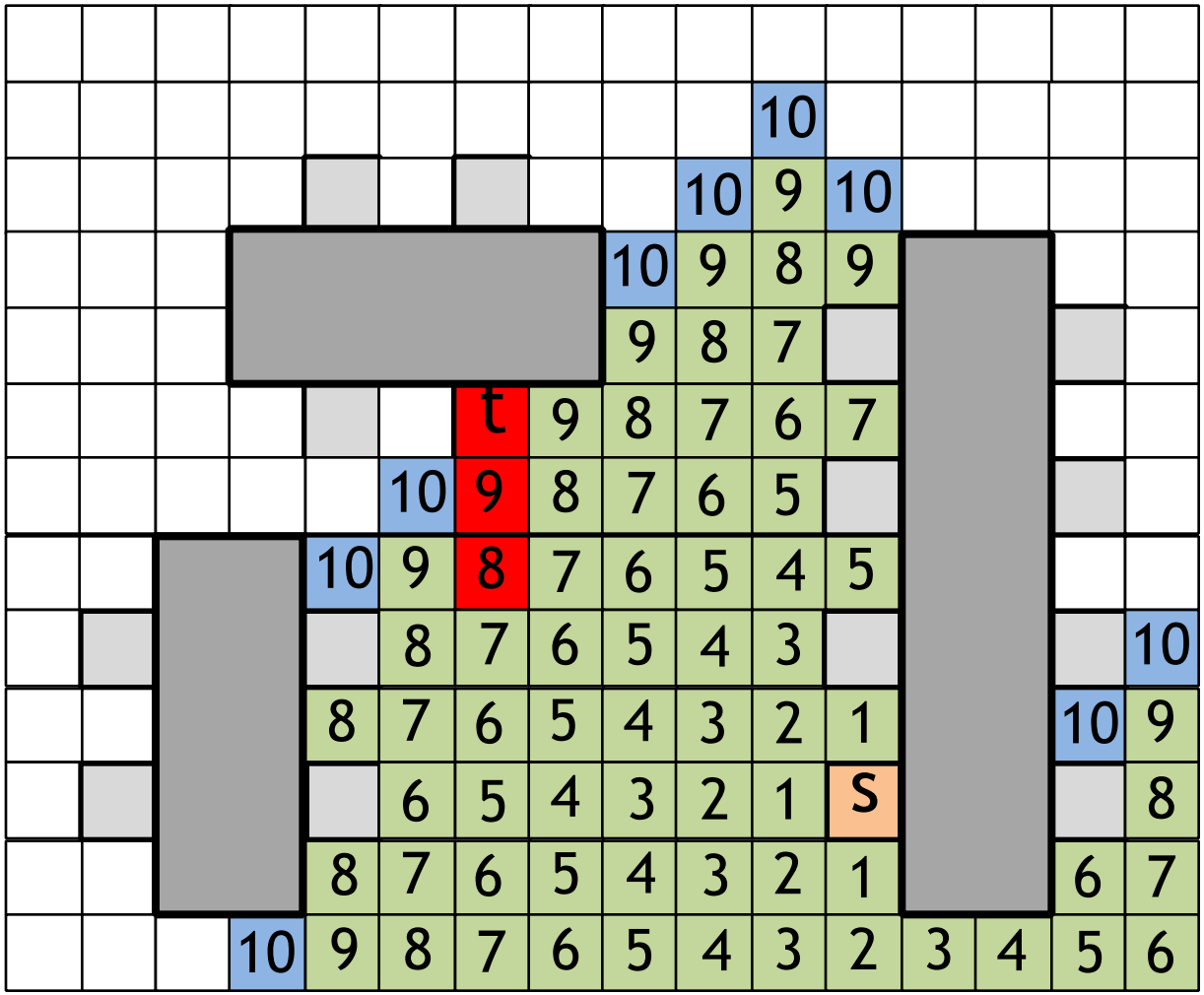
Wavefront Expansion



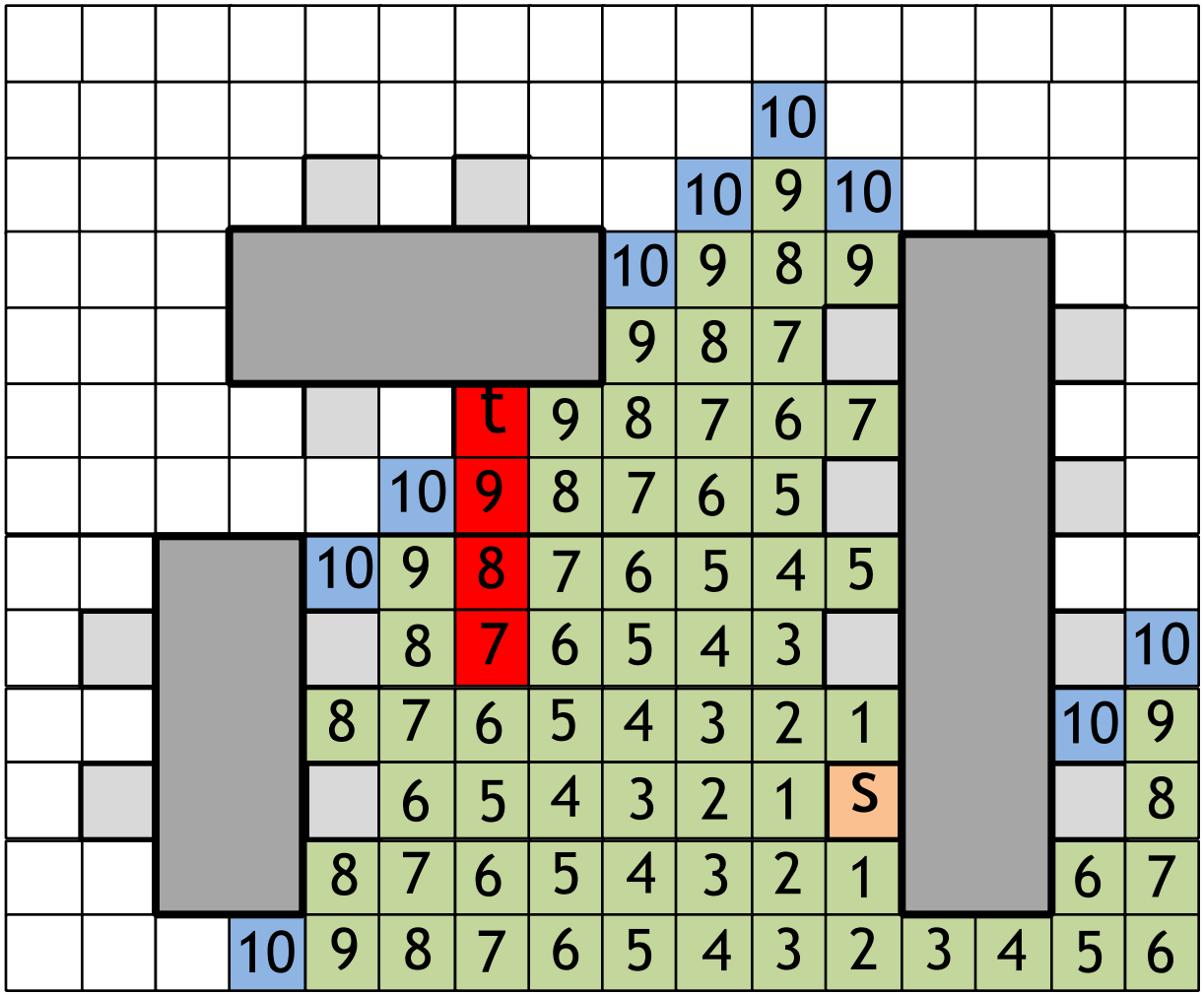
Backtrace


10
10 9 10
10 9 8 9
9 8 7
t 9 8 7 6 7
10 9 8 7 6 5
10 9 8 7 6 5 4 5
8 7 6 5 4 3 10
8 7 6 5 4 3 2 1 10 9
6 5 4 3 2 1 s 8
8 7 6 5 4 3 2 1 6 7
10 9 8 7 6 5 4 3 2 3 4 5 6

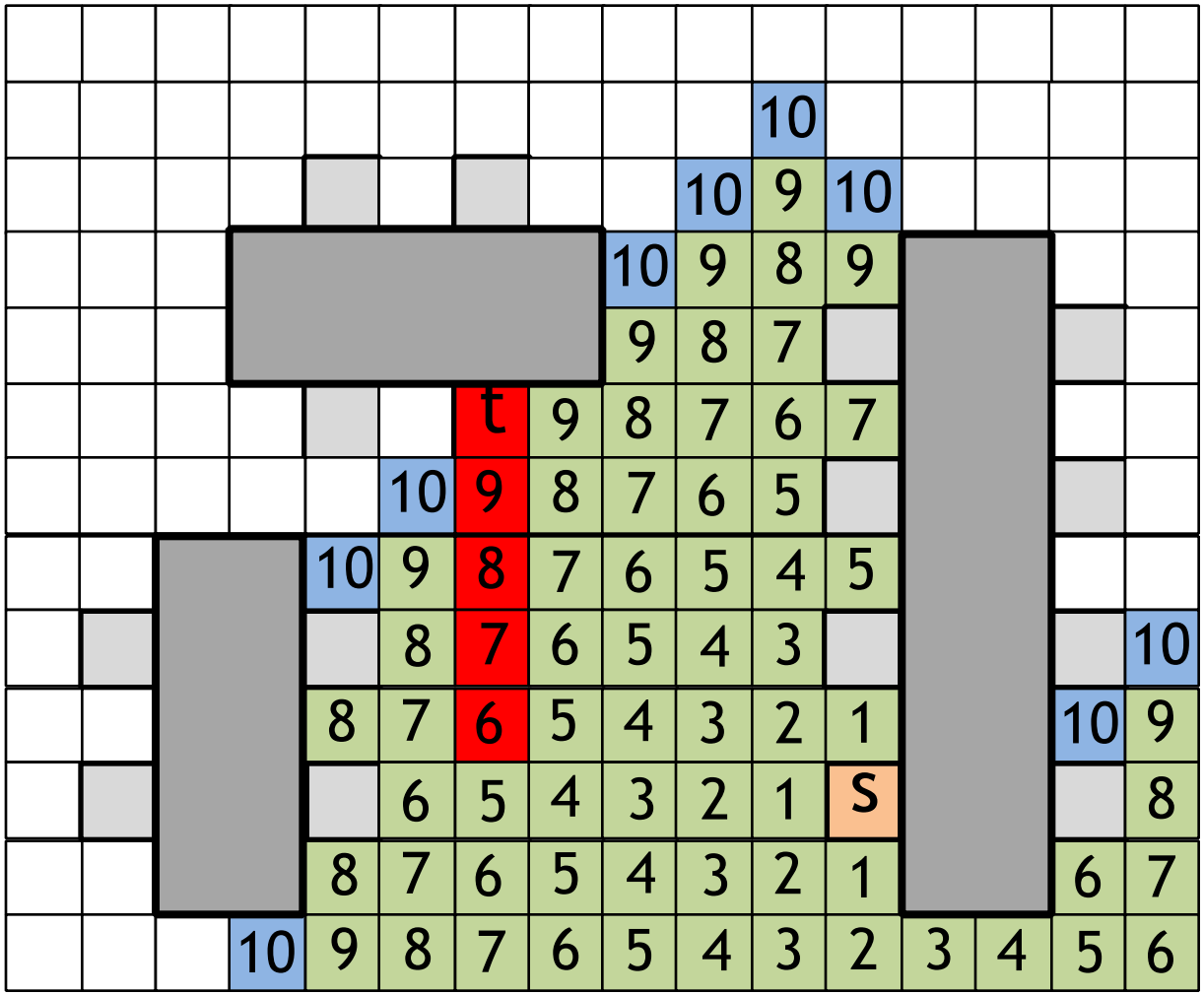
Backtrace


10
10 9 10
10 9 8 9
9 8 7
t 9 8 7 6 7
10 9 8 7 6 5
10 9 8 7 6 5 4 5
8 7 6 5 4 3 10
8 7 6 5 4 3 2 1 10 9
6 5 4 3 2 1 s 8
8 7 6 5 4 3 2 1 6 7
10 9 8 7 6 5 4 3 2 3 4 5 6

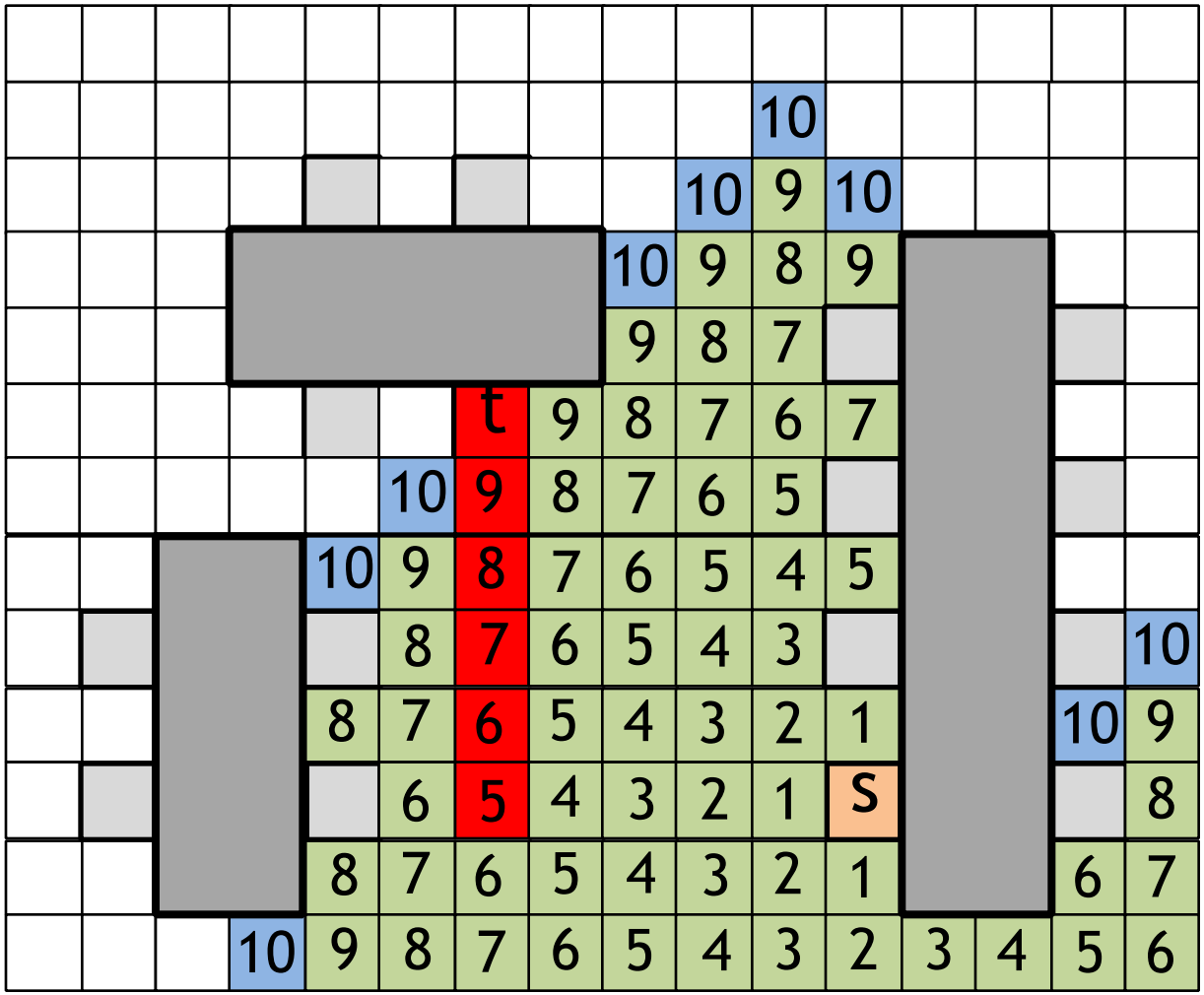
Backtrace


10
10 9 10
10 9 8 9
9 8 7
t 9 8 7 6 7
10 9 8 7 6 5
10 9 8 7 6 5 4 5
8 7 6 5 4 3 10
8 7 6 5 4 3 2 1 10 9
6 5 4 3 2 1 s 8
8 7 6 5 4 3 2 1 6 7
10 9 8 7 6 5 4 3 2 3 4 5 6

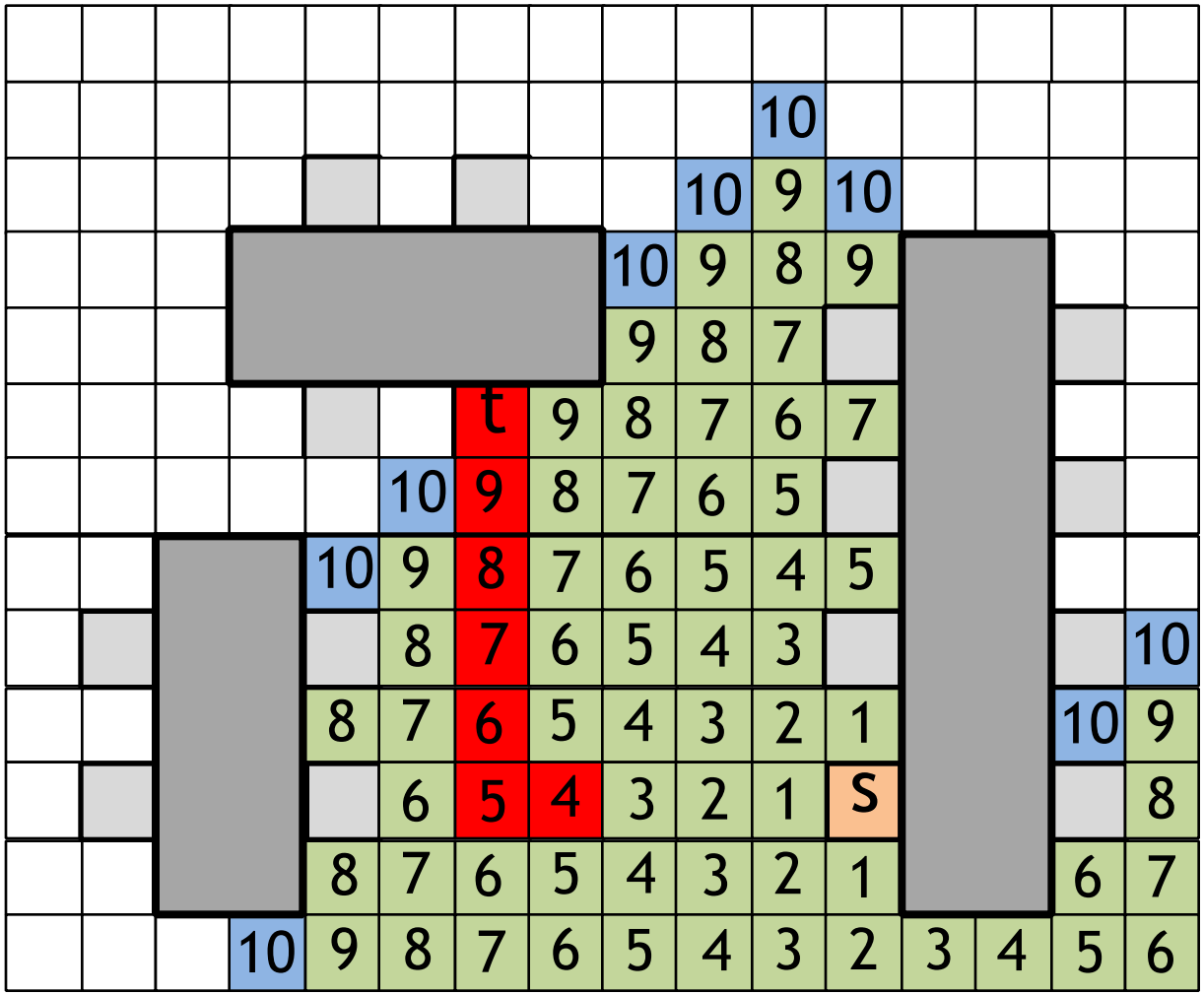
Backtrace


10
10 9 10
10 9 8 9
9 8 7
t 9 8 7 6 7
10 9 8 7 6 5
10 9 8 7 6 5 4 5
8 7 6 5 4 3 10
8 7 6 5 4 3 2 1 10 9
6 5 4 3 2 1 s 8
8 7 6 5 4 3 2 1 6 7
10 9 8 7 6 5 4 3 2 3 4 5 6

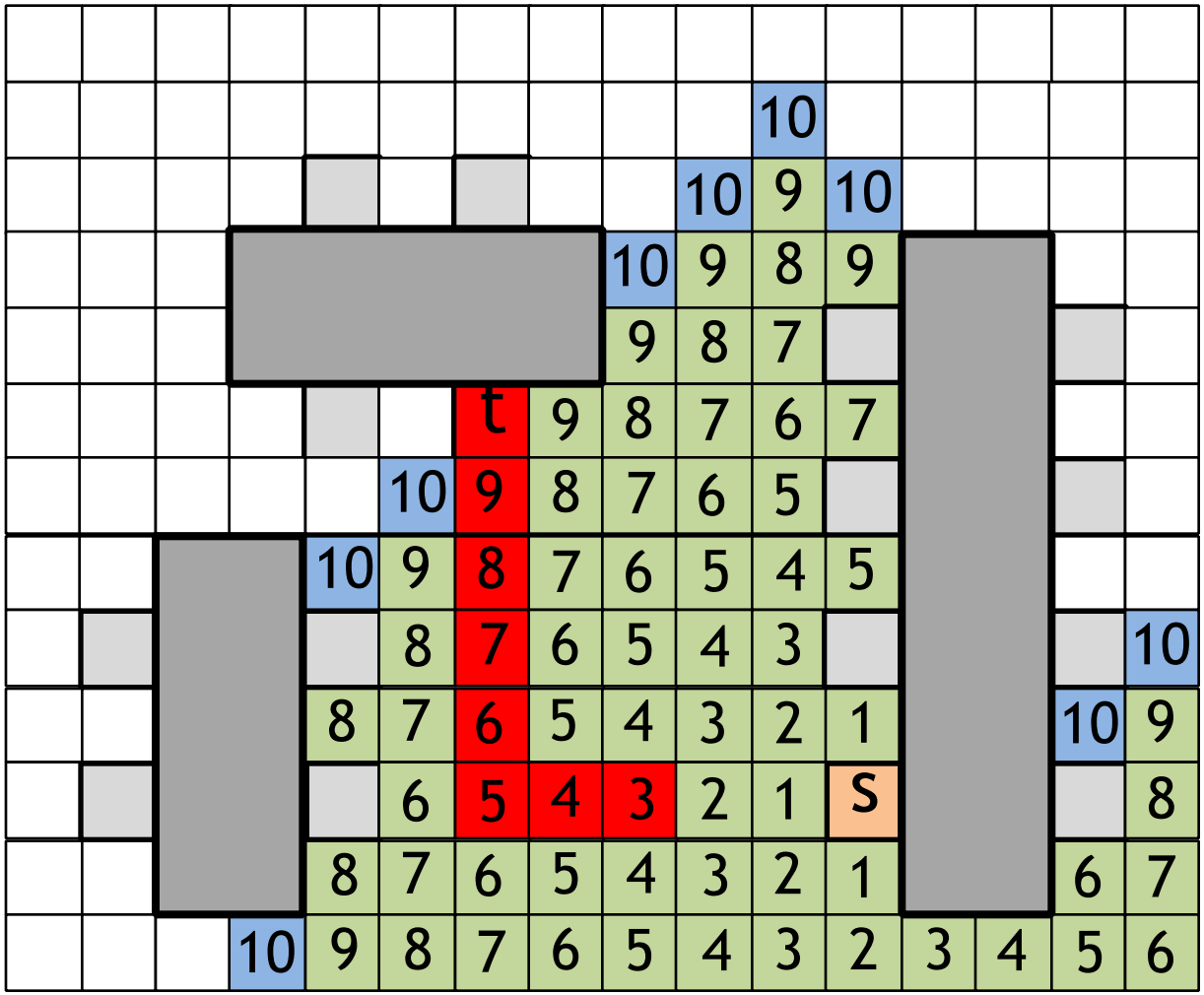
Backtrace


10
10 9 10
10 9 8 9
9 8 7
t 9 8 7 6 7
10 9 8 7 6 5
10 9 8 7 6 5 4 5
8 7 6 5 4 3 10
8 7 6 5 4 3 2 1 10 9
6 5 4 3 2 1 s 8
8 7 6 5 4 3 2 1 6 7
10 9 8 7 6 5 4 3 2 3 4 5 6

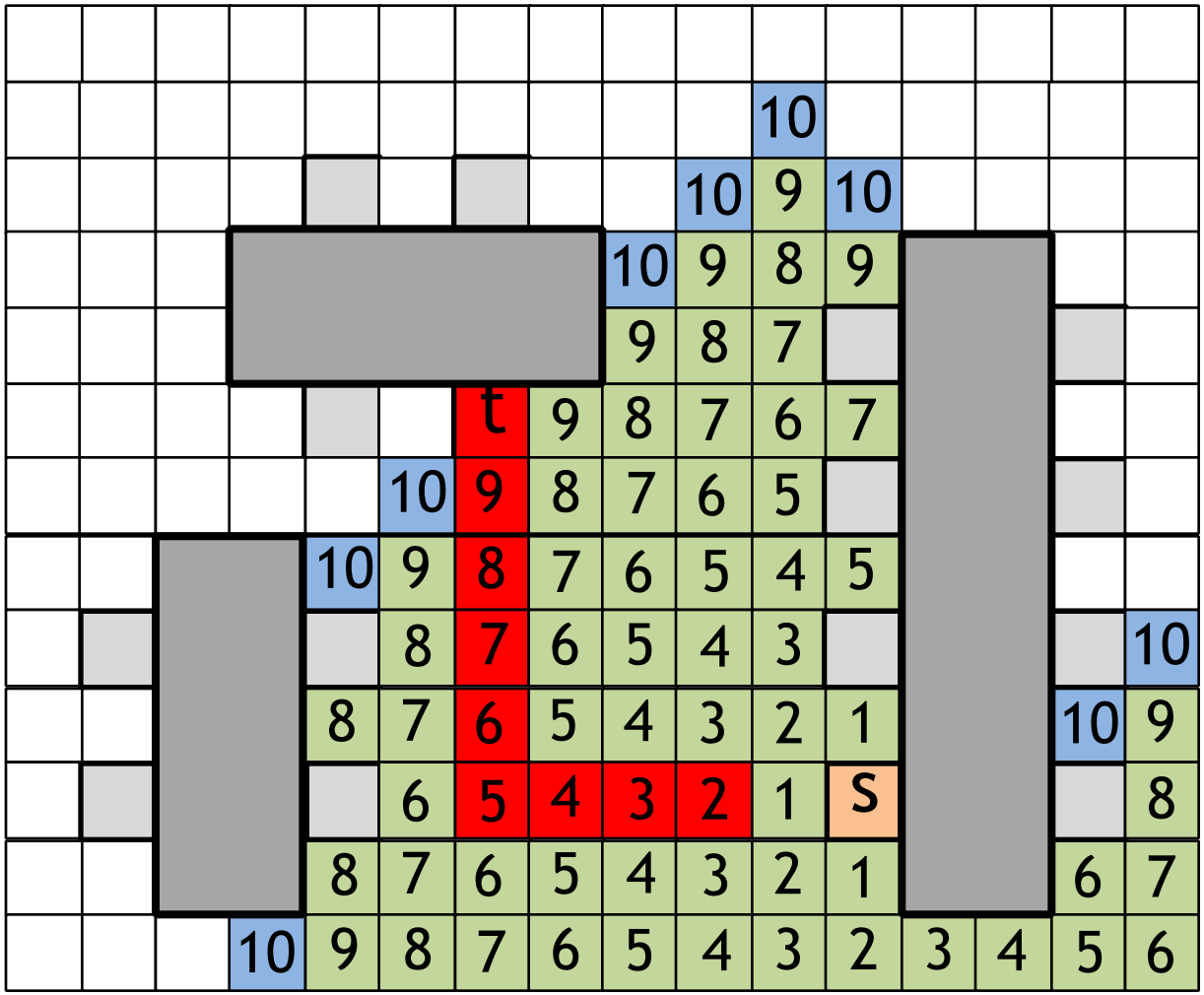
Backtrace


10
10 9 10
10 9 8 9
9 8 7
t 9 8 7 6 7
10 9 8 7 6 5
10 9 8 7 6 5 4 5
8 7 6 5 4 3 10
8 7 6 5 4 3 2 1 10 9
6 5 4 3 2 1 s 8
8 7 6 5 4 3 2 1 6 7
10 9 8 7 6 5 4 3 2 3 4 5 6

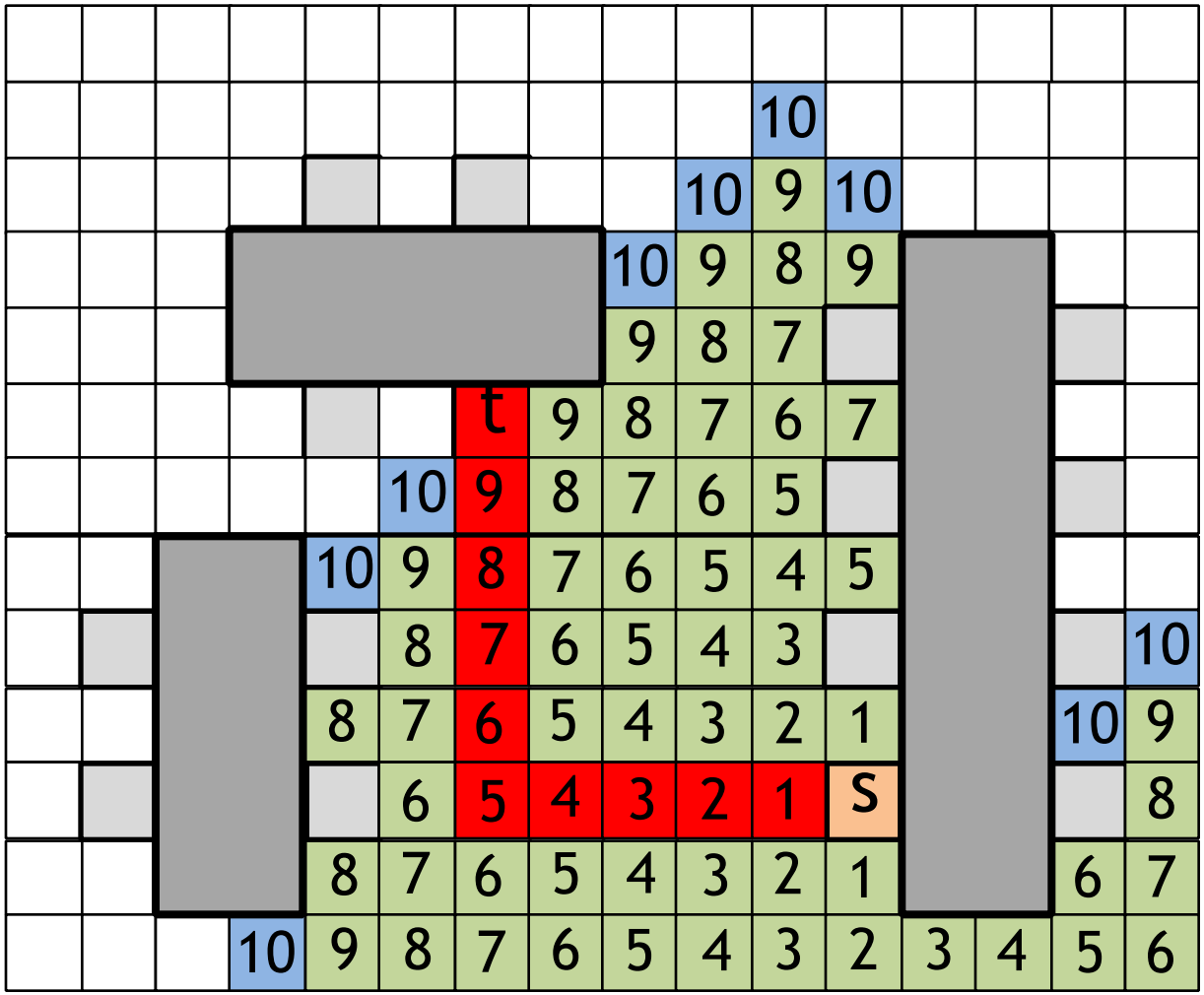
Backtrace


10
10 9 10
10 9 8 9
9 8 7
t 9 8 7 6 7
10 9 8 7 6 5
10 9 8 7 6 5 4 5
8 7 6 5 4 3 10
8 7 6 5 4 3 2 1 10 9
6 5 4 3 2 1 s 8
8 7 6 5 4 3 2 1 6 7
10 9 8 7 6 5 4 3 2 3 4 5 6

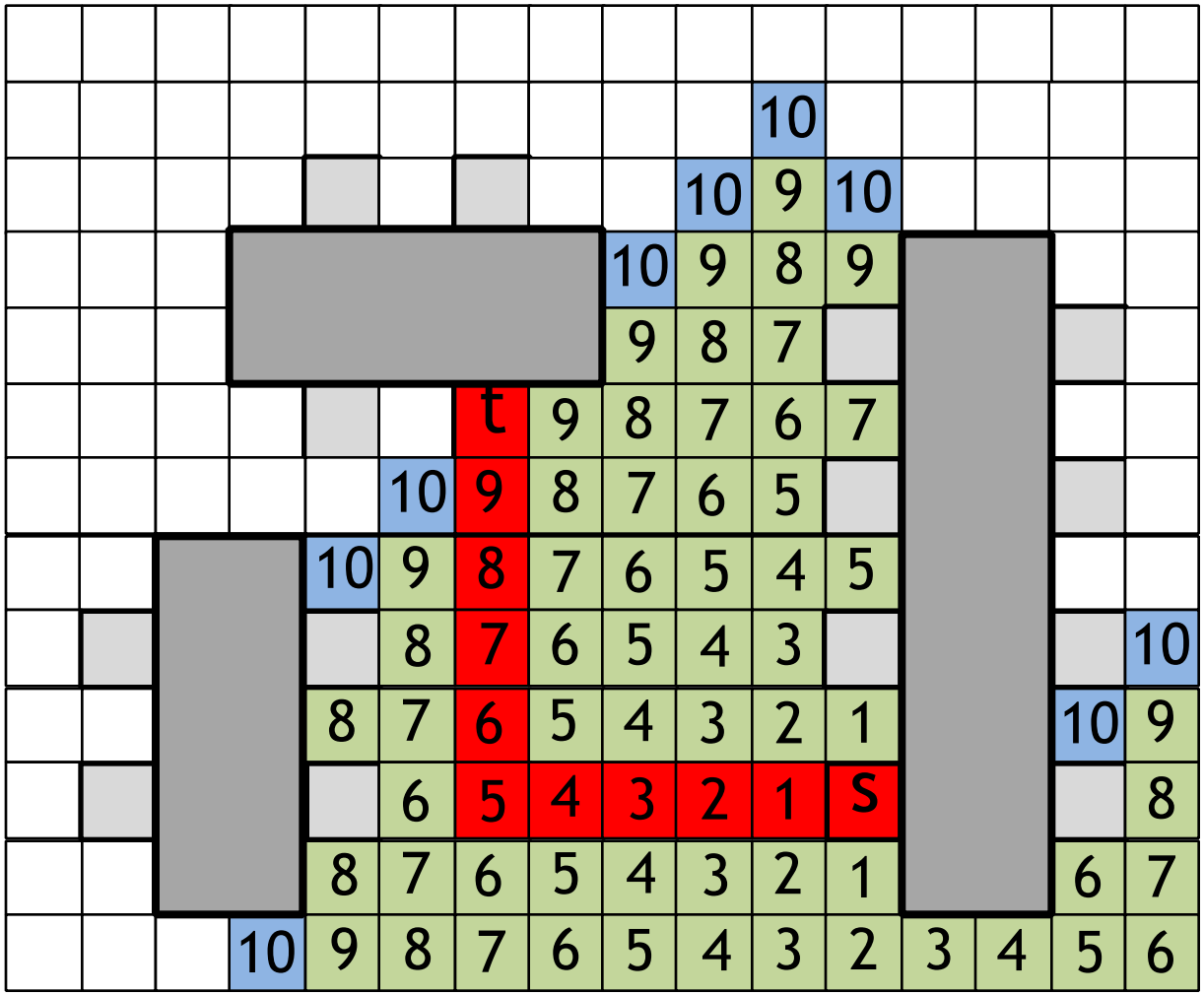
Backtrace


10
10 9 10
10 9 8 9
9 8 7
t 9 8 7 6 7
10 9 8 7 6 5
10 9 8 7 6 5 4 5
8 7 6 5 4 3 10
8 7 6 5 4 3 2 1 10 9
6 5 4 3 2 1 s 8
8 7 6 5 4 3 2 1 6 7
10 9 8 7 6 5 4 3 2 3 4 5 6

Backtrace

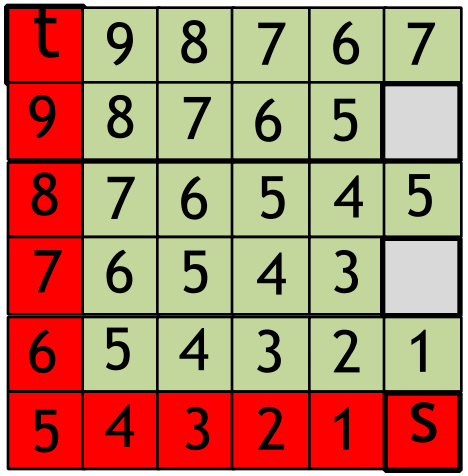

10
10 9 10
10 9 8 9
9 8 7
t 9 8 7 6 7
10 9 8 7 6 5
10 9 8 7 6 5 4 5
8 7 6 5 4 3 10
8 7 6 5 4 3 2 1 10 9
6 5 4 3 2 1 s 8
8 7 6 5 4 3 2 1 6 7
10 9 8 7 6 5 4 3 2 3 4 5 6

Backtrace


10
10 9 10
10 9 8 9
9 8 7
t 9 8 7 6 7
10 9 8 7 6 5
10 9 8 7 6 5 4 5
8 7 6 5 4 3 10
8 7 6 5 4 3 2 1 10 9
6 5 4 3 2 1 s 8
8 7 6 5 4 3 2 1 6 7
10 9 8 7 6 5 4 3 2 3 4 5 6

Key Observation

- Every path from t back to s goes from 9 down to 1 in decreasing order
- Just follow the labels and you will find a legal path

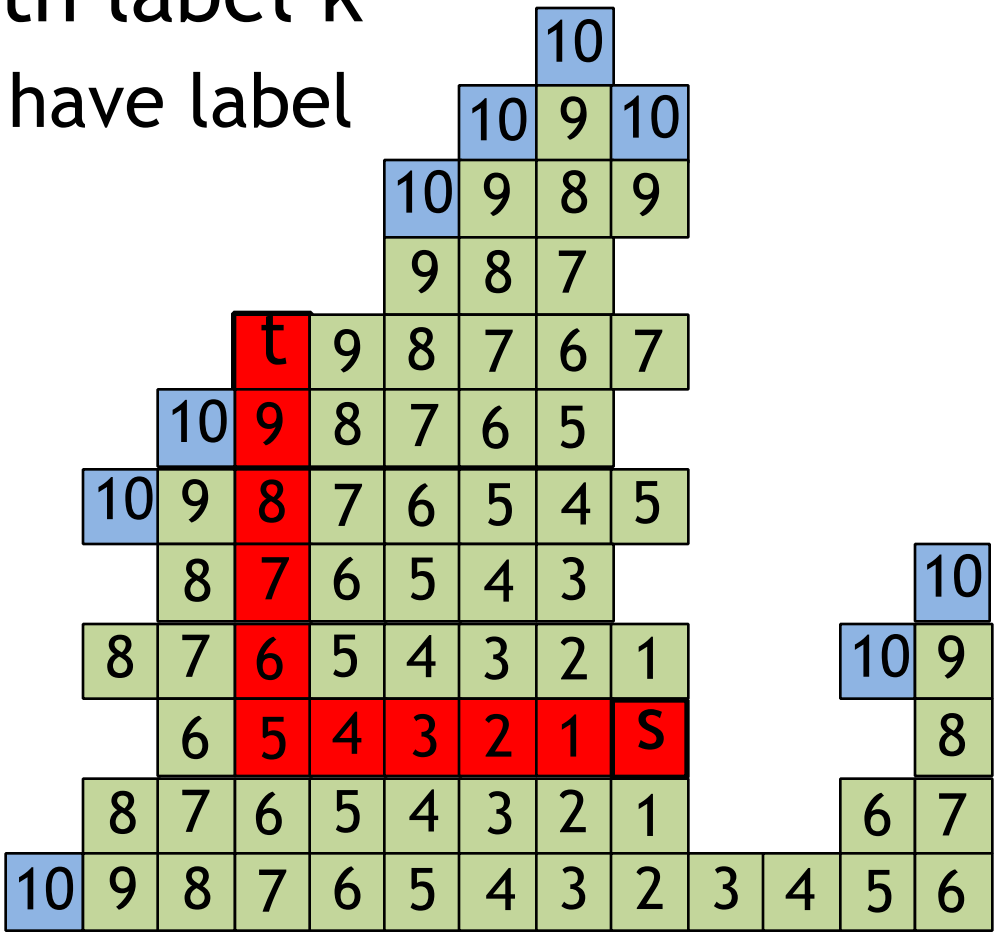


Label Encoding

Akers, S. B. (1967), *A Modification of Lee's Path Connection Algorithm*, IEEE Transactions on Electronic Computers, EC-16(1): 97–98

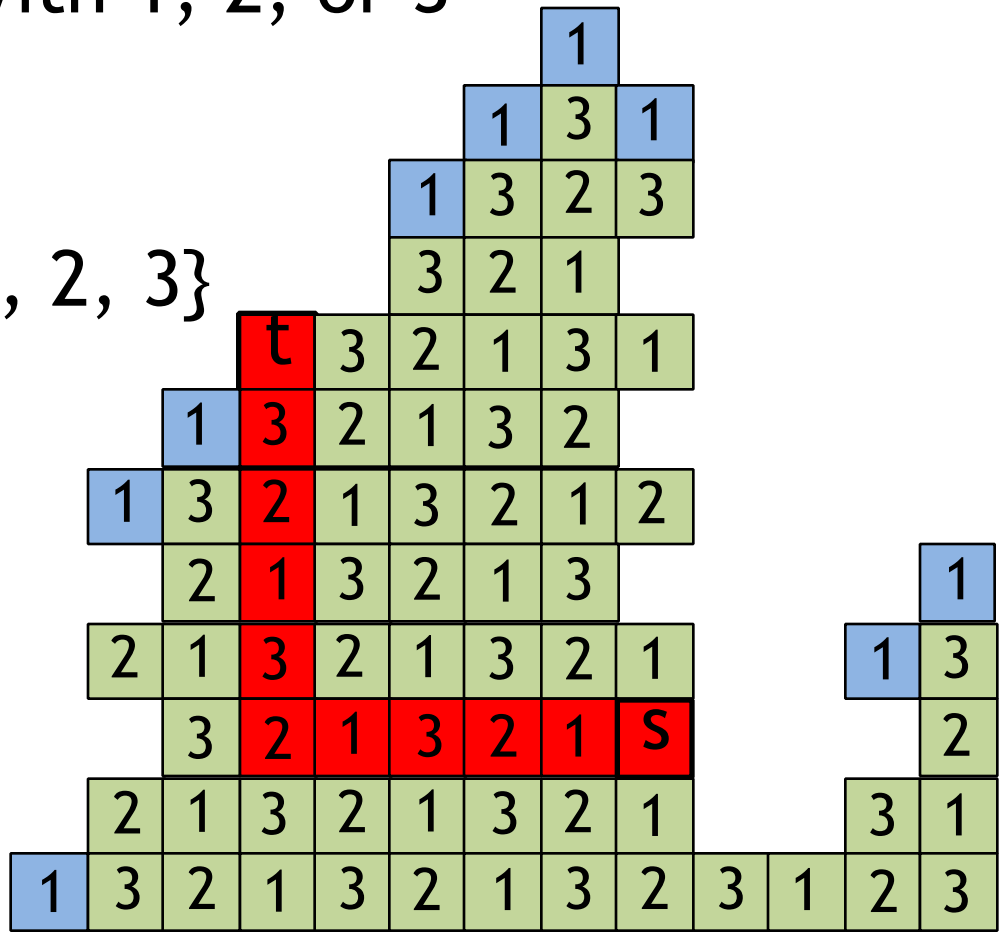
Observation

- Consider a cell with label k
  - All adjacent cells have label k-1 or k+1



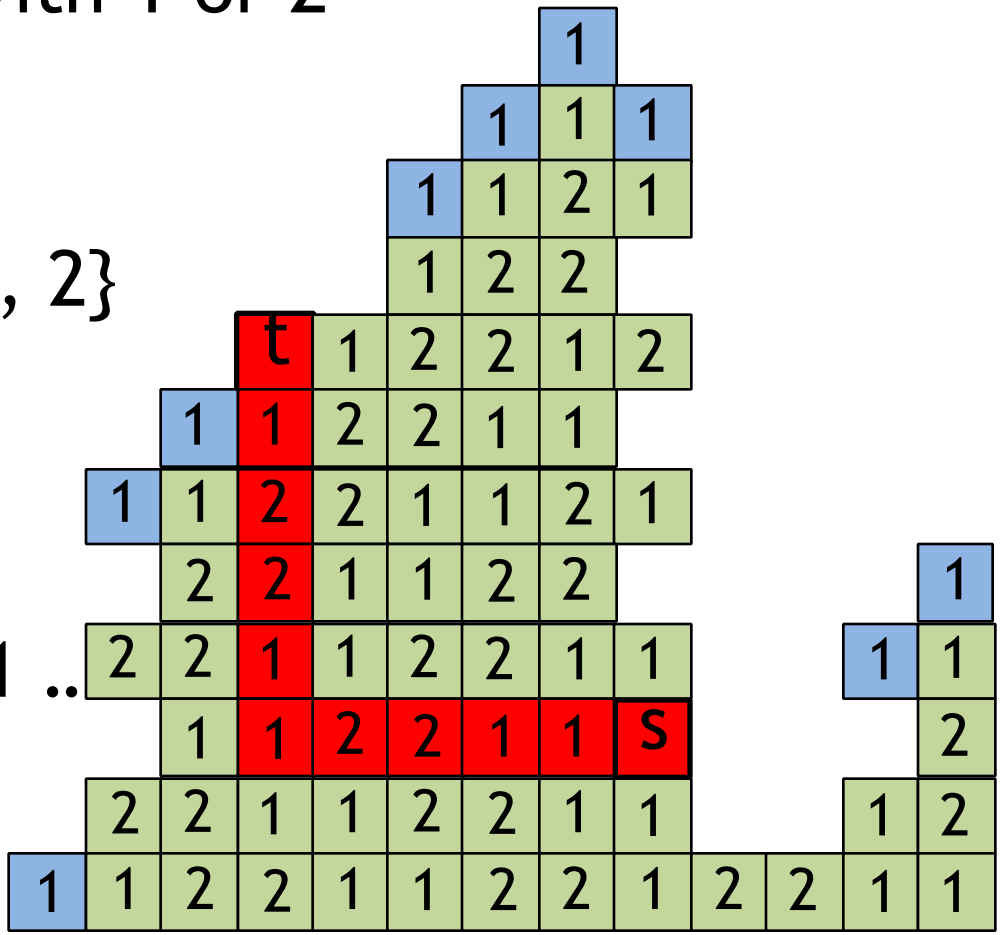
3-bit Encoding

- Replace all labels with 1, 2, or 3
- 3 bits per label
  {empty, blocked, 1, 2, 3}
- Traceback path will have the general form 3 2 1 3 2 1 …



2-bit Encoding

- Replace all labels with 1 or 2
- 2 bits per label
  {empty, blocked, 1, 2}
- Traceback path will have the general form 2 2 1 1 2 2 1 1 ..



Bit Encoding in C/C++

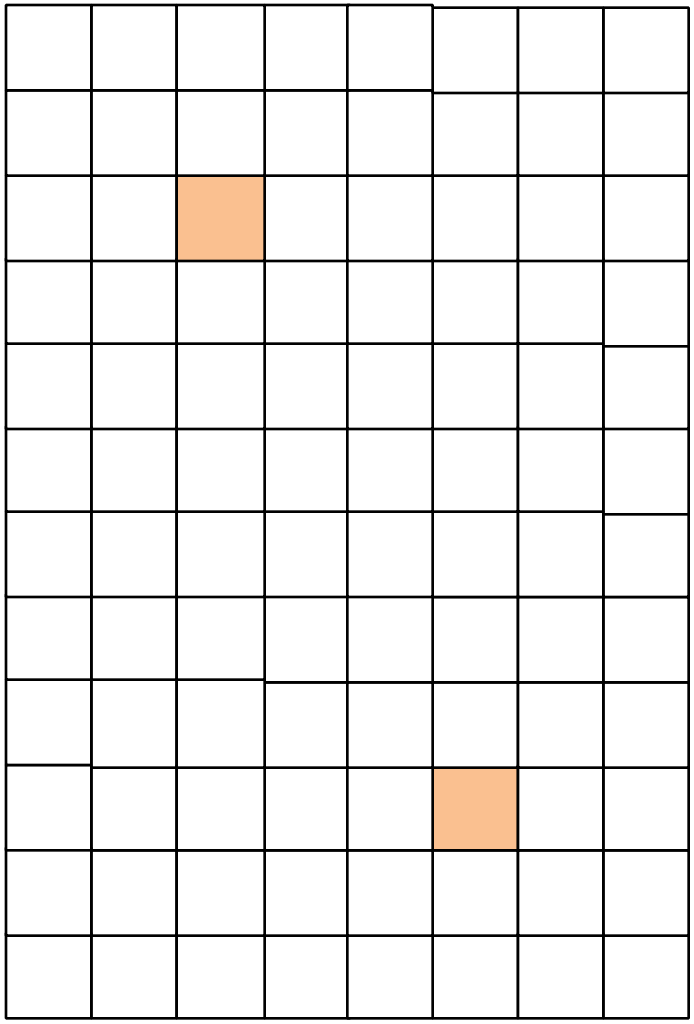
```
typedef enum Labels { Empty, Blocked, L1, L2 } LabelType;

int main(void) {
    LabelType grid_labels[M][N]; // MxN grid

    // initialize to empty
    for( int i = 0; i < M; i++)
        for ( int j = 0; j < N; j++ )
            grid_labels[i][j] = Empty;
}
```

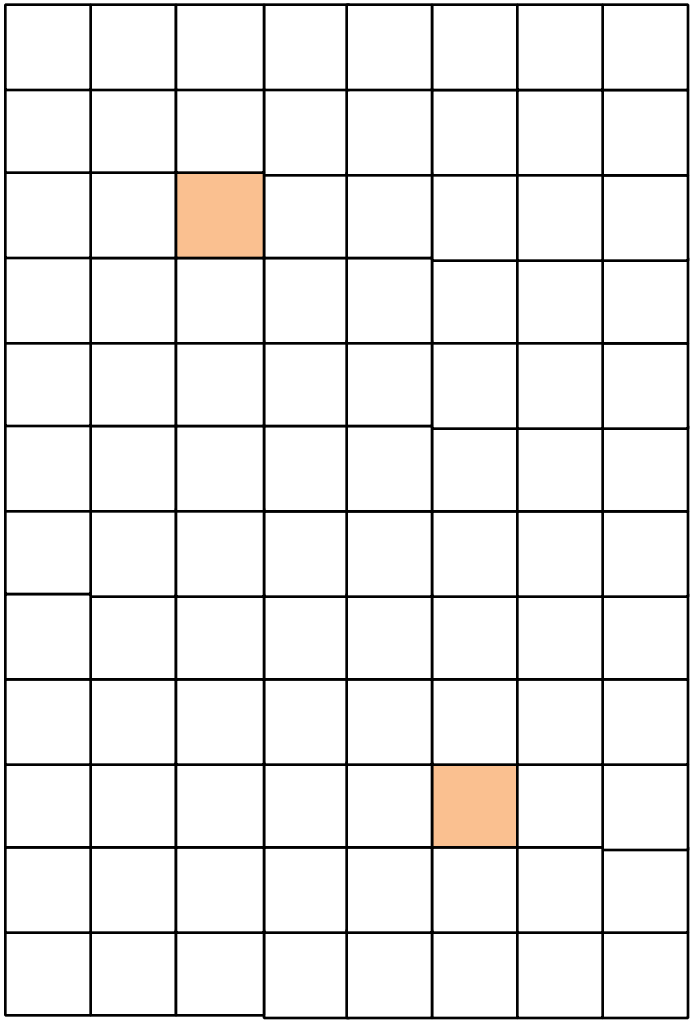
- If there are k values in the enumerated list, then log2k bits are required per instance.

Bi-directional Search

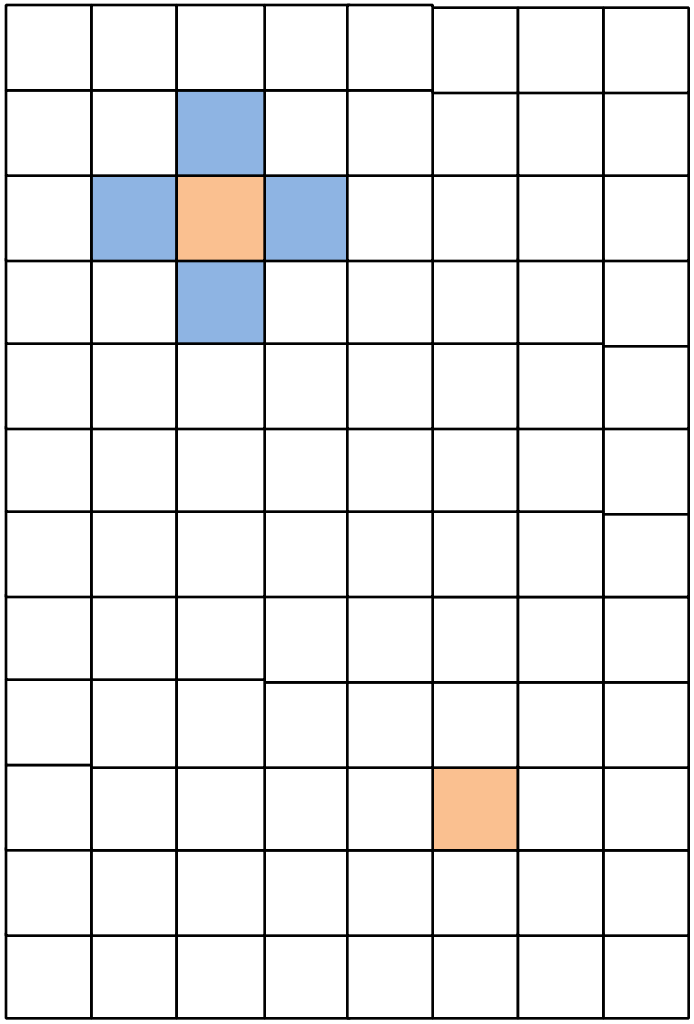
Pohl, Ira (1971), "Bi-directional Search", in Meltzer, Bernard; Michie, Donald, *Machine Intelligence* 6, Edinburgh University Press, pp. 127–140.

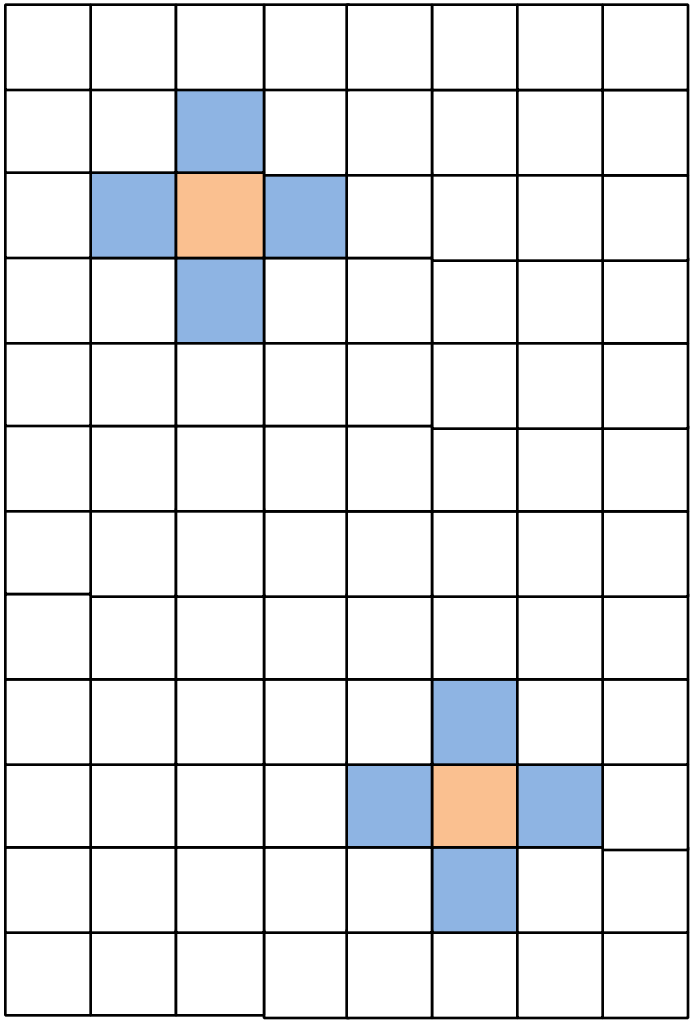
Bi-directional Search



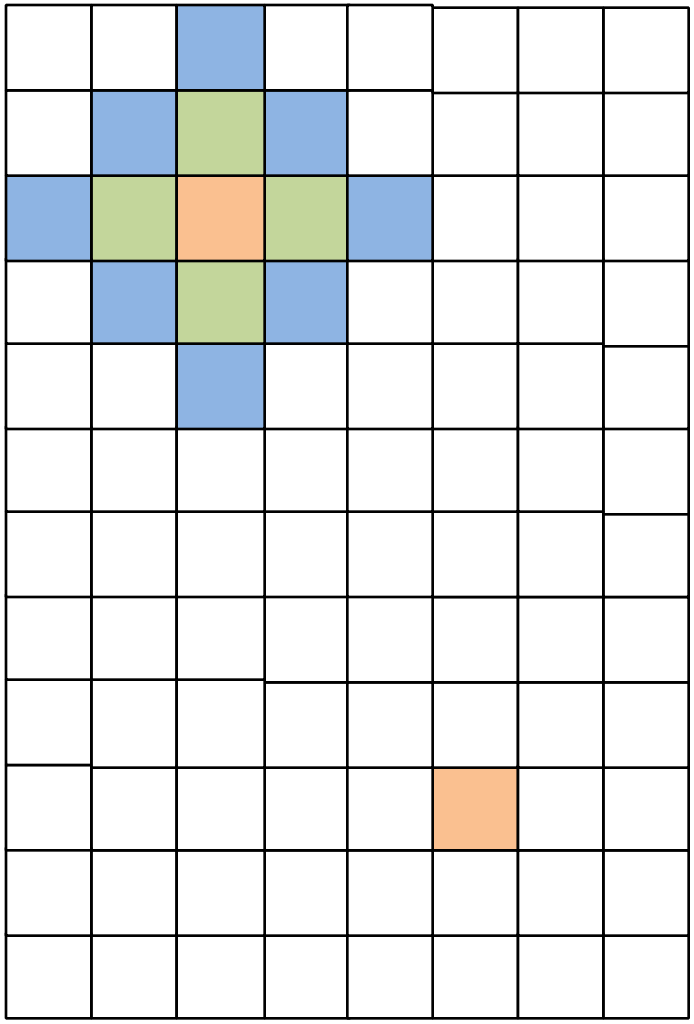


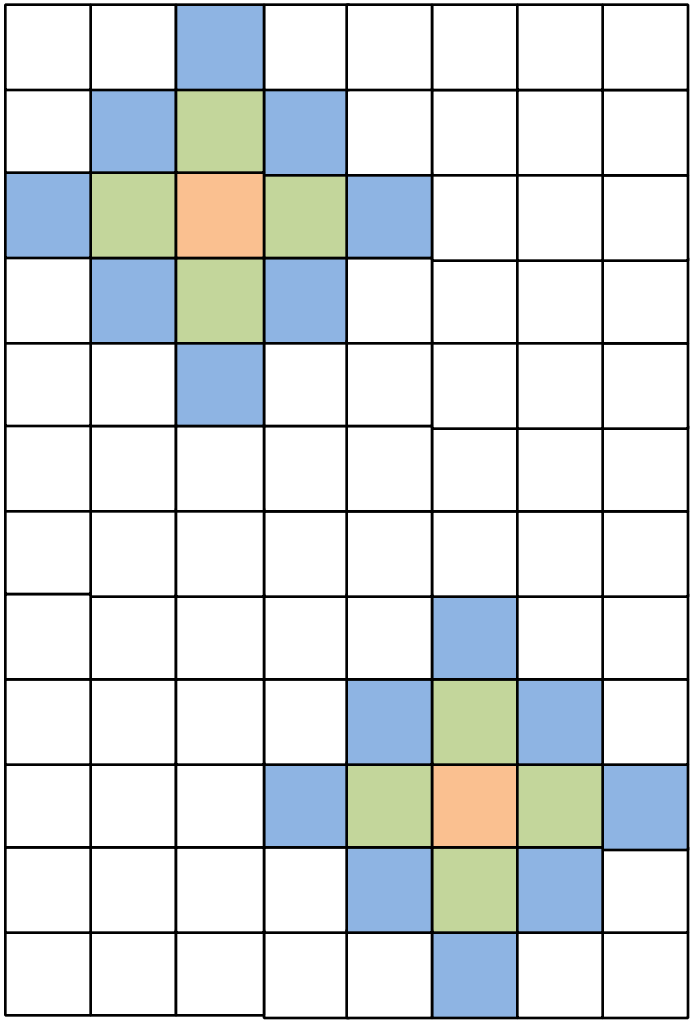
Bi-directional Search



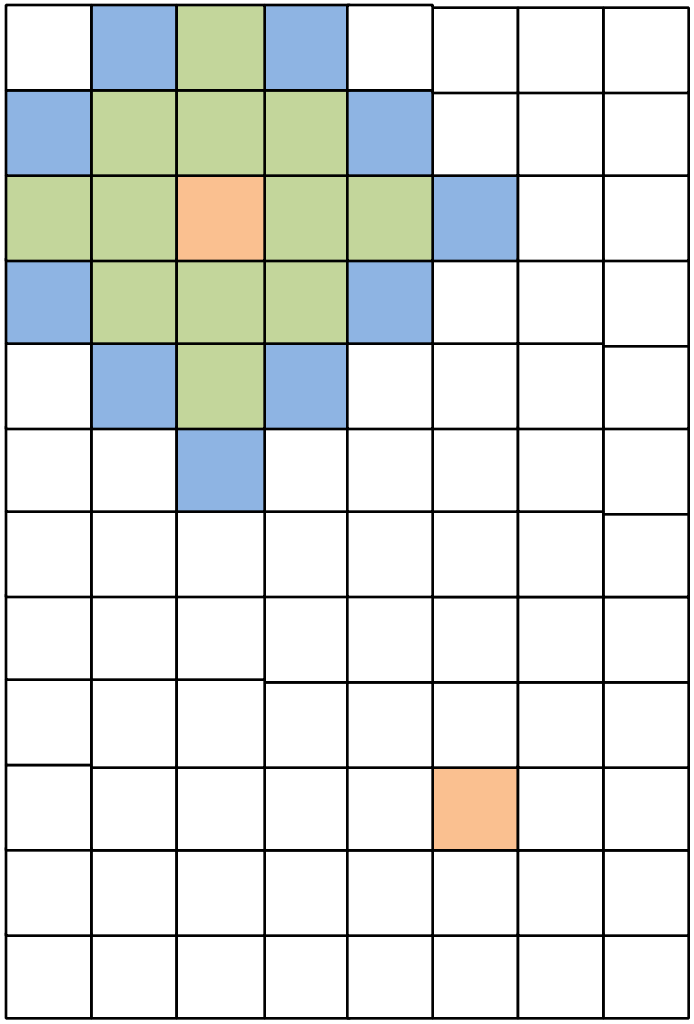


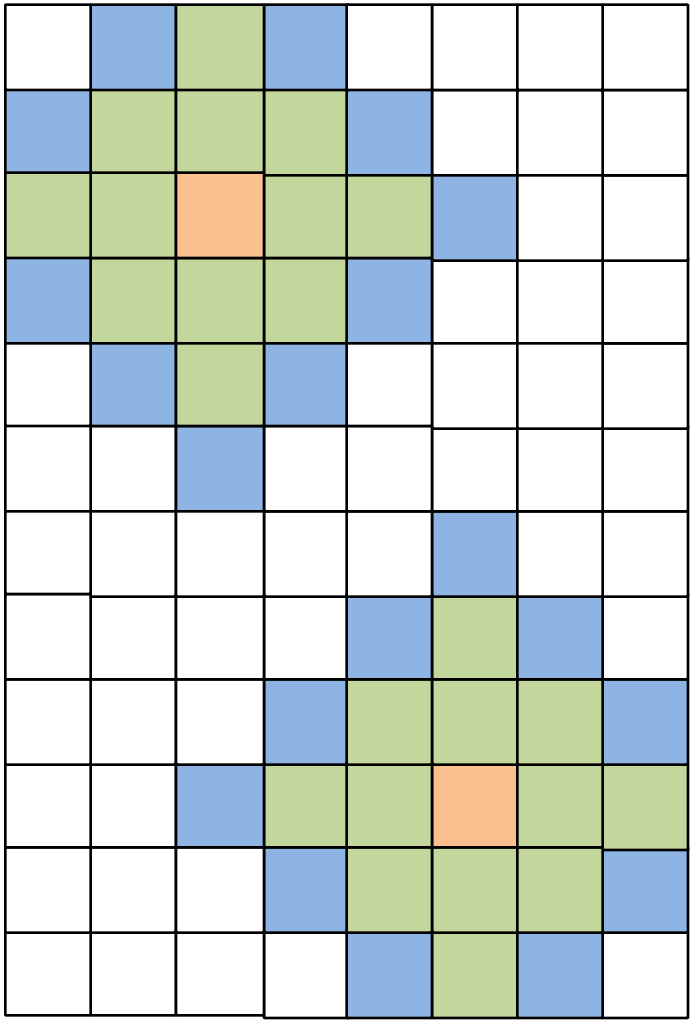
Bi-directional Search



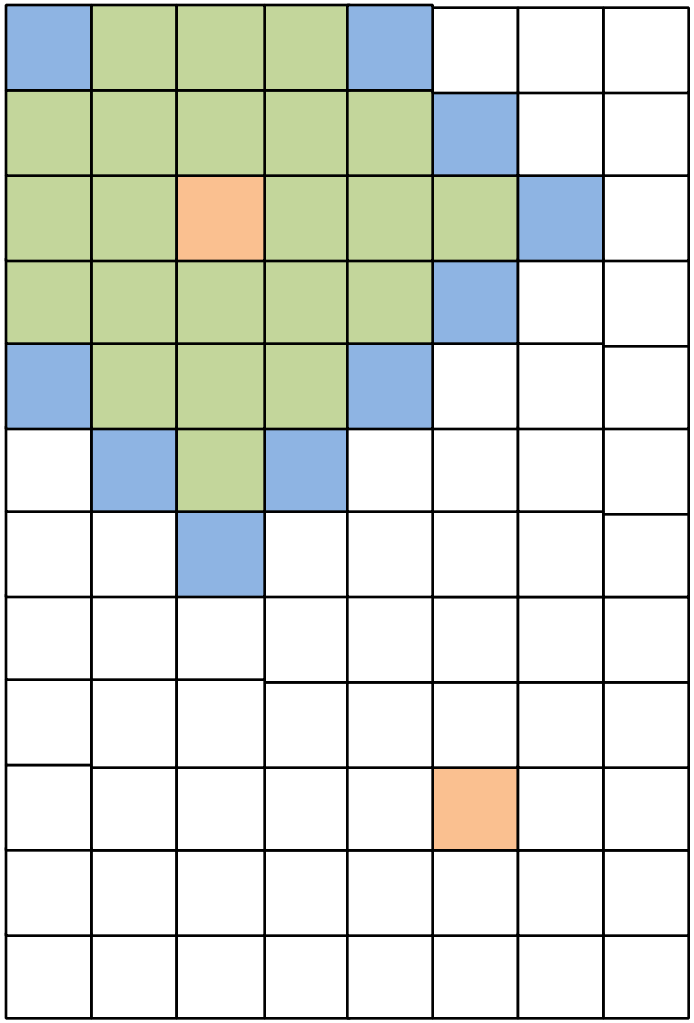


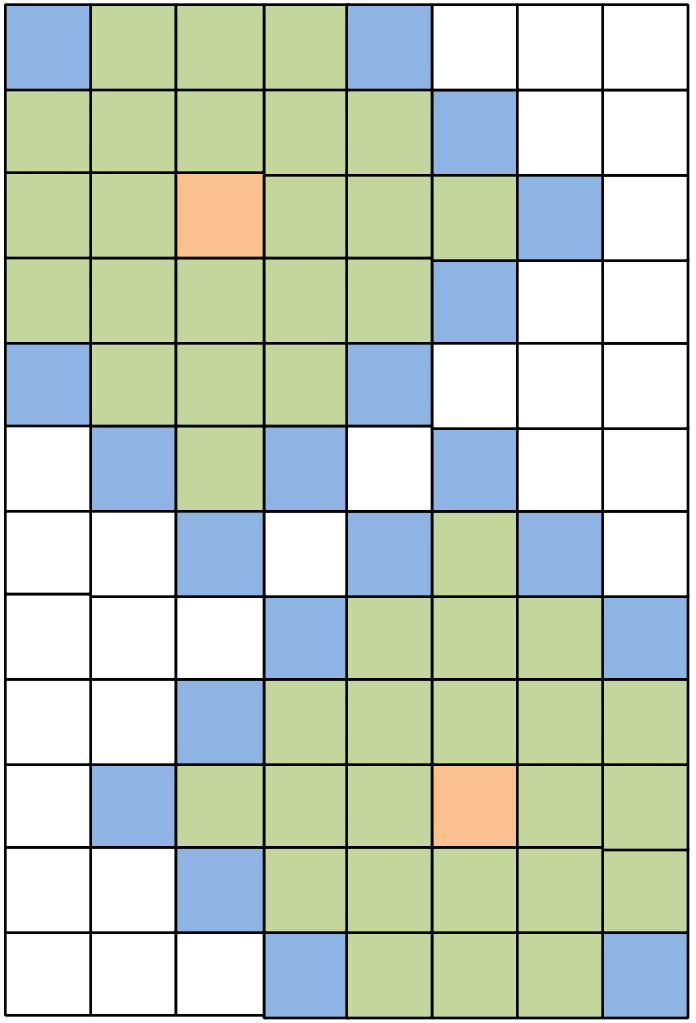
Bi-directional Search



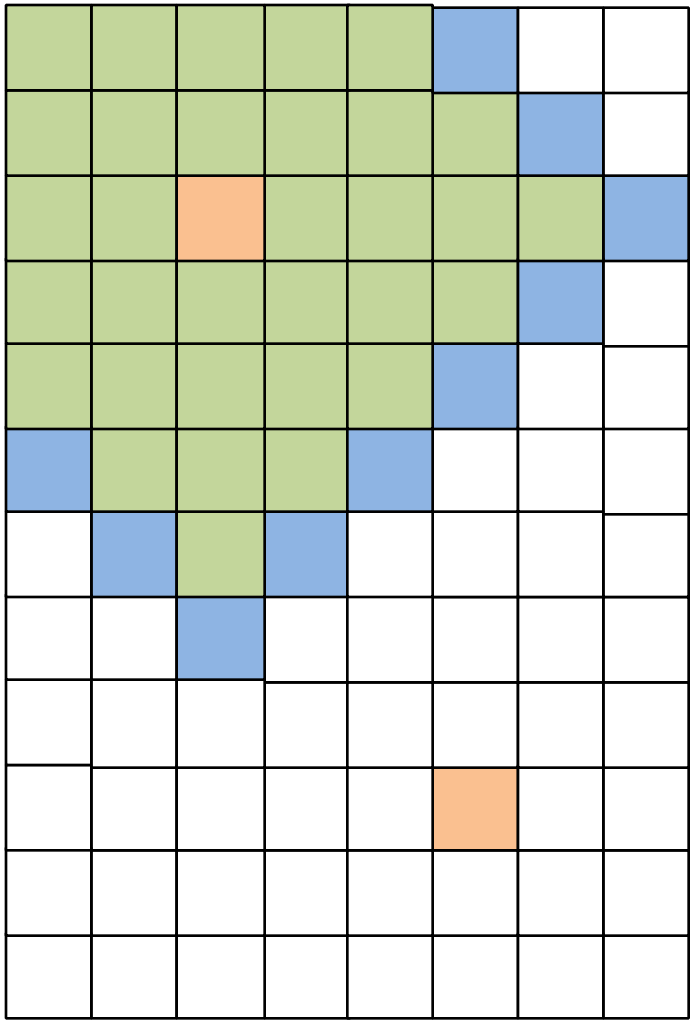


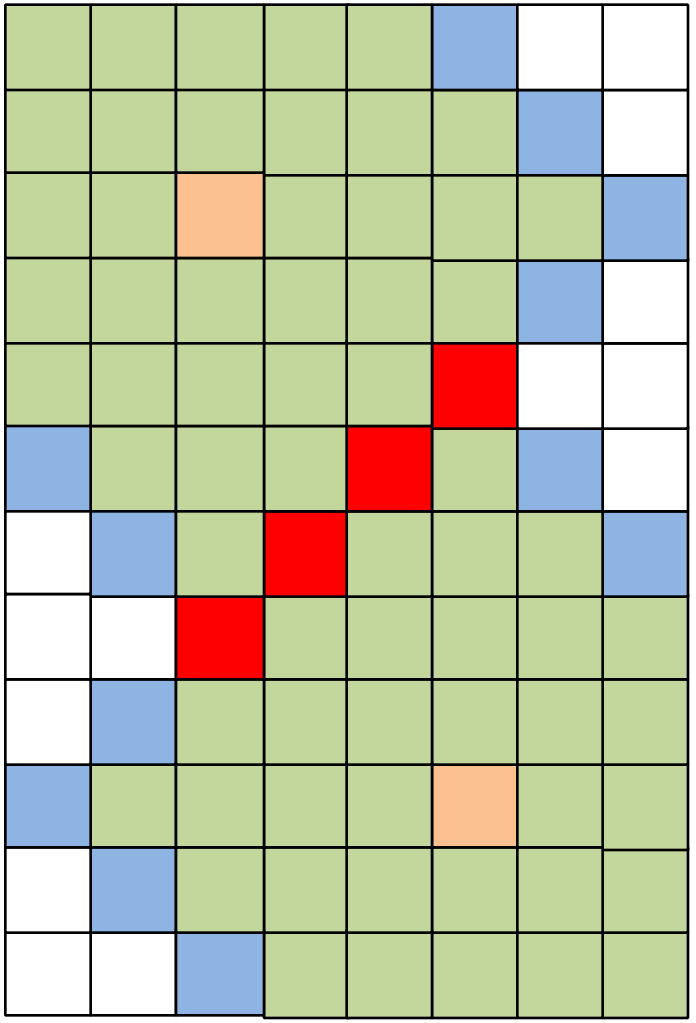
Bi-directional Search



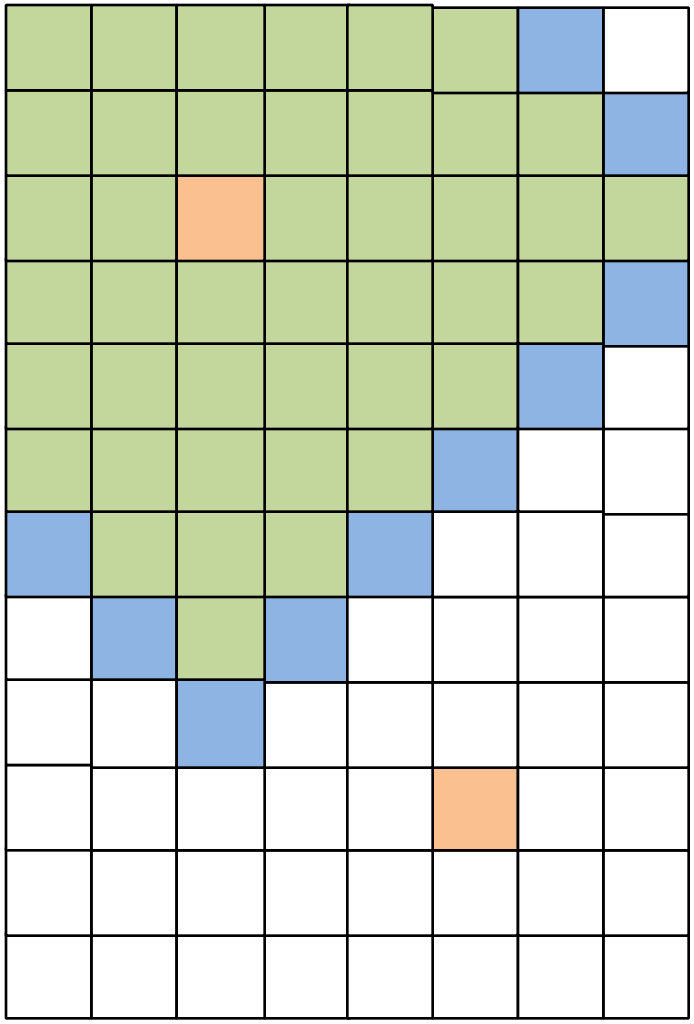


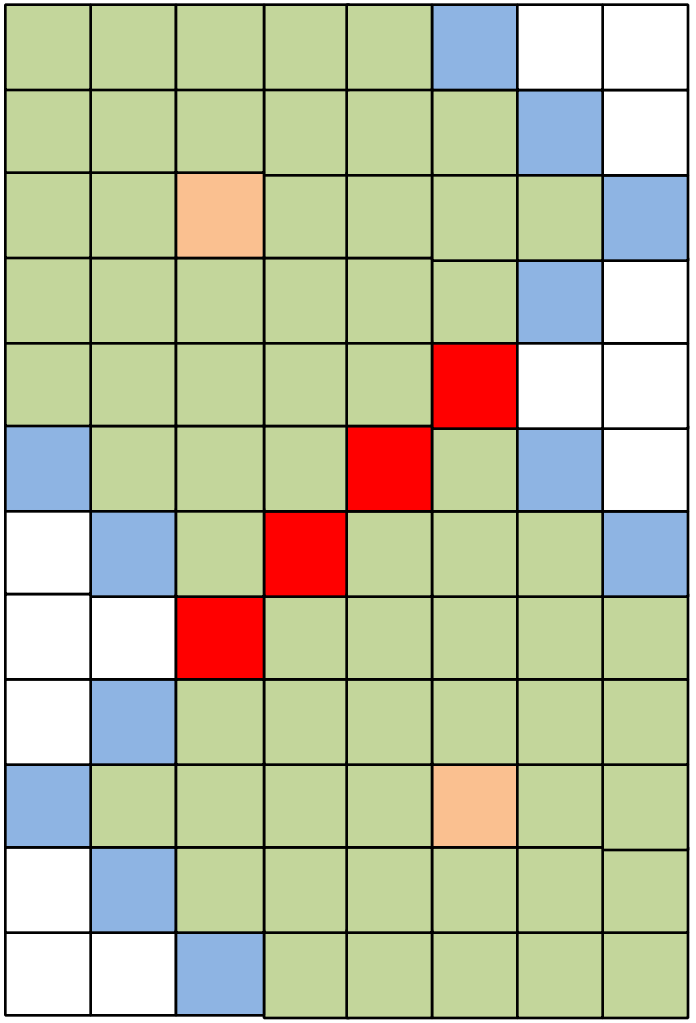
Bi-directional Search



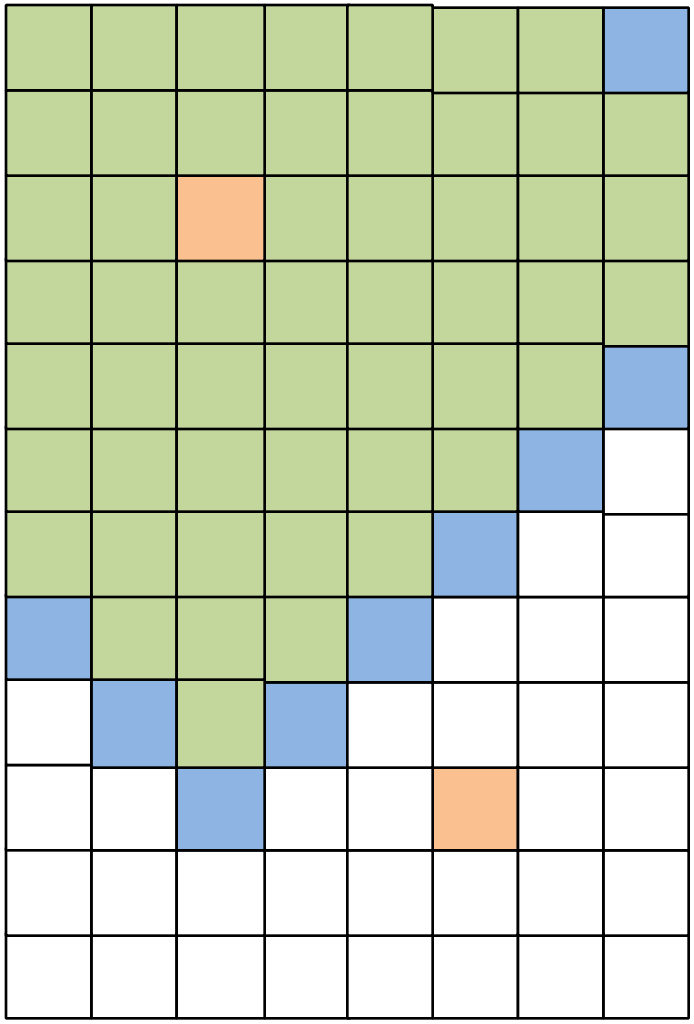


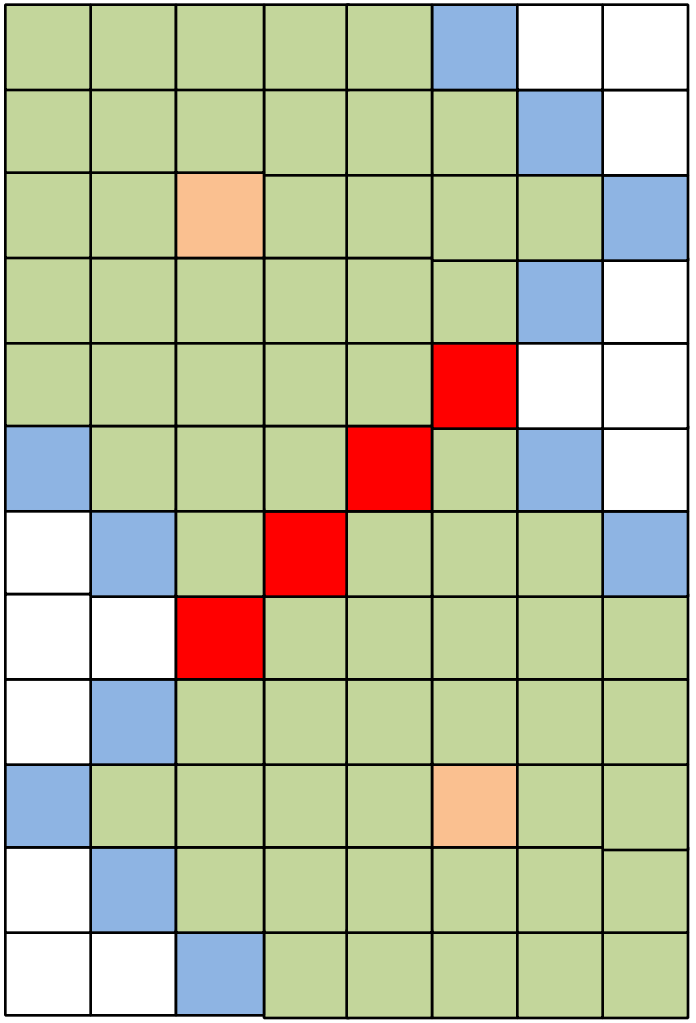
Bi-directional Search



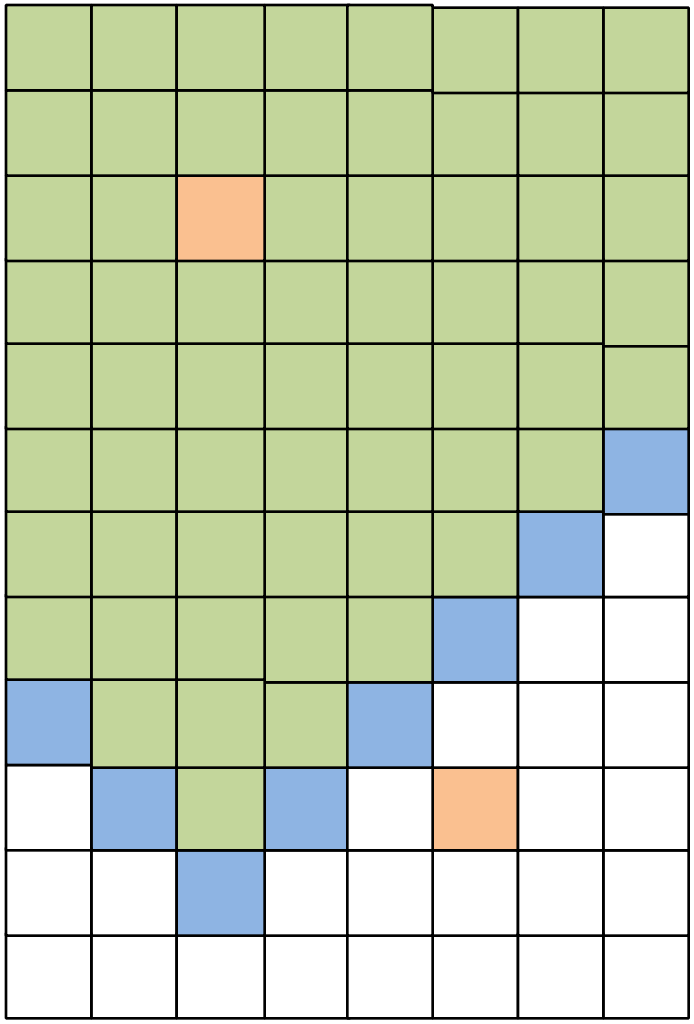


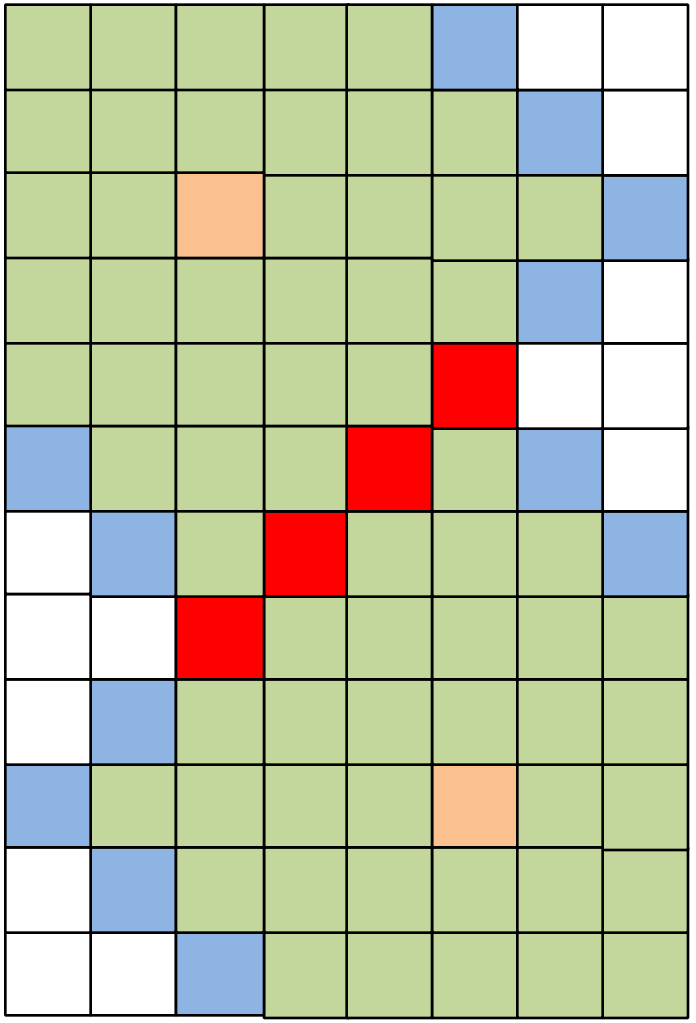
Bi-directional Search



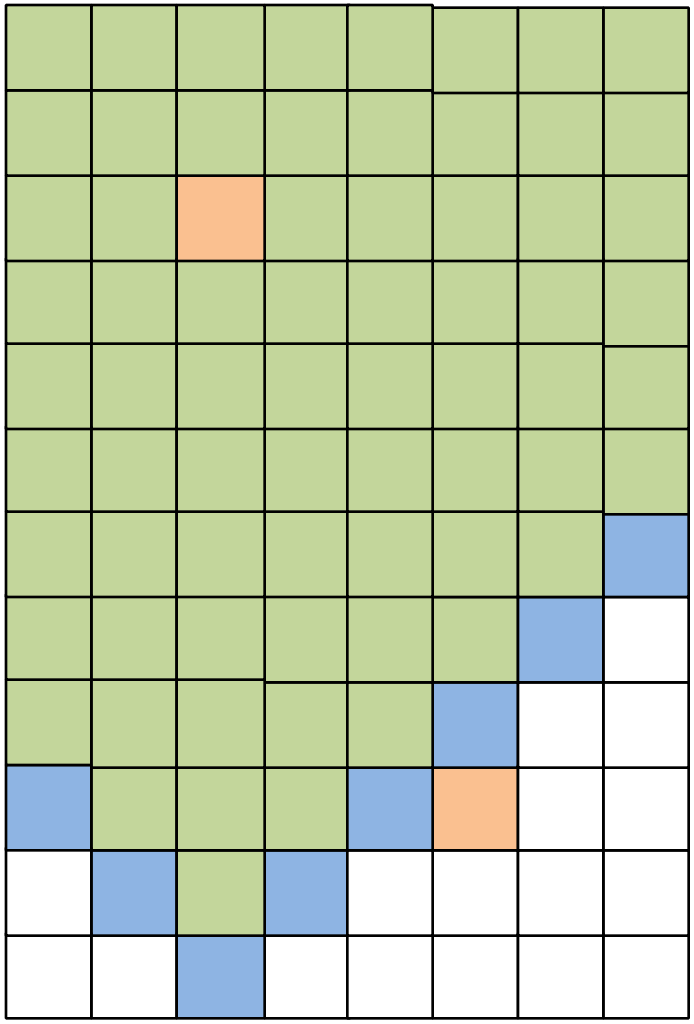


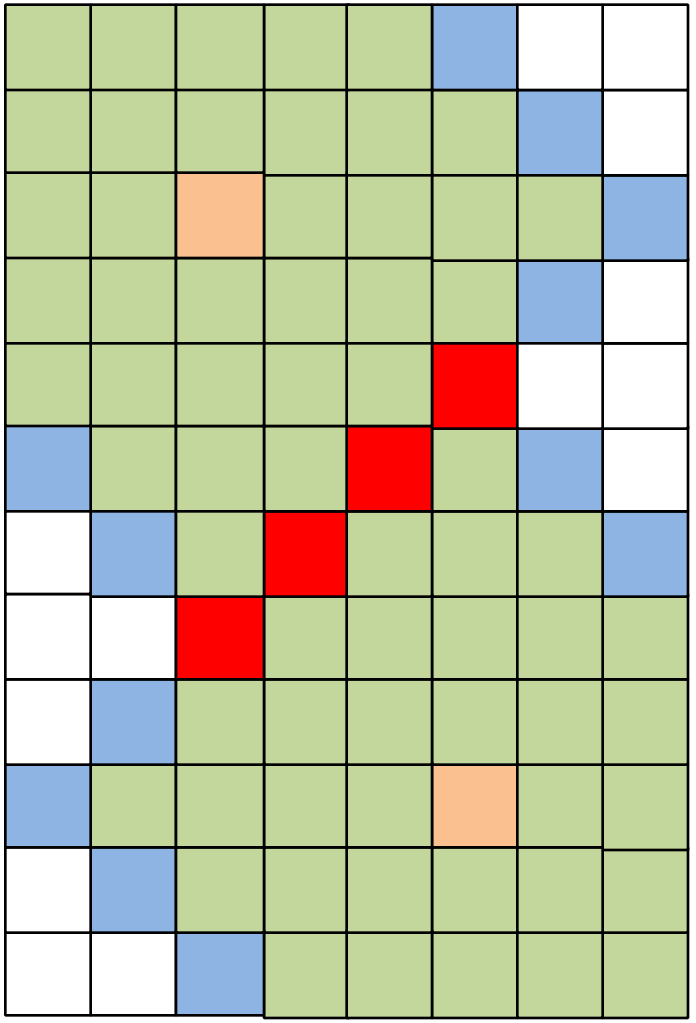
Bi-directional Search





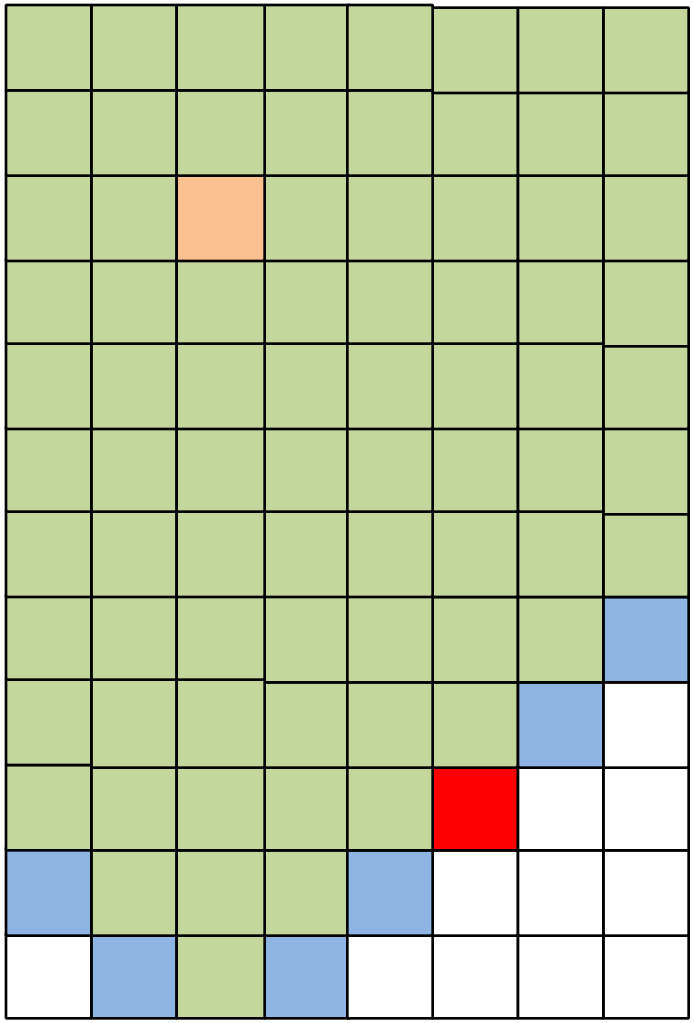
Bi-directional Search



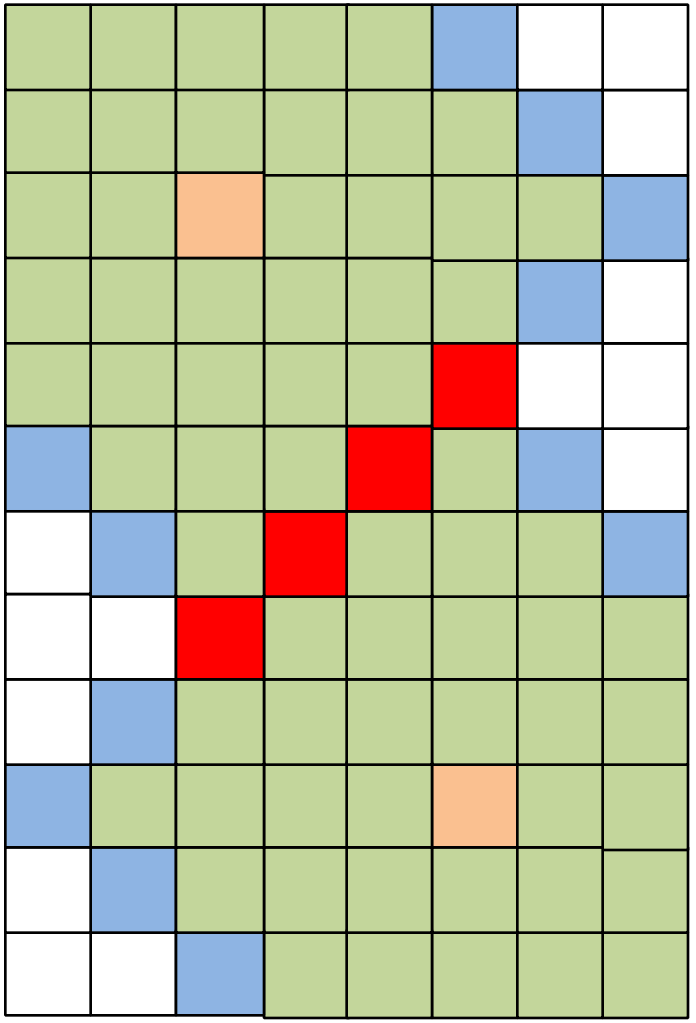


Bi-directional Search

11 Unexplored Cells



14 Unexplored Cells



Ruben's Enhancements

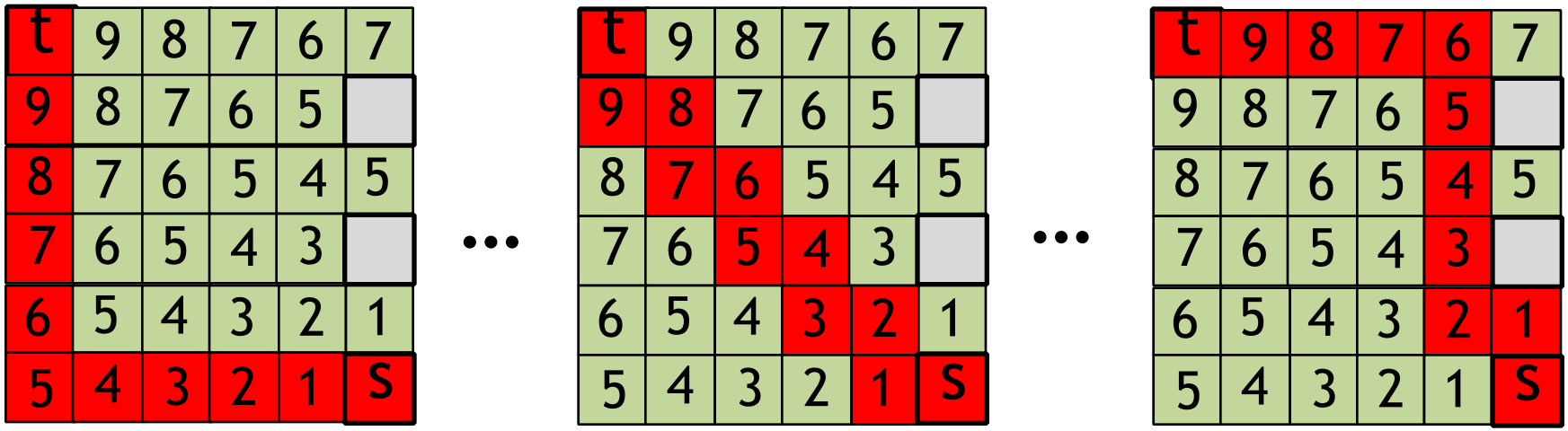
Ruben, F. (1974), *The Lee Path Connection Algorithm*, IEEE Transactions on Computers, C-23(9): 907–914

Ruben's Enhancments

- Minimize the number of turns in a path
  - Perhaps more important for fluid flows than electronic circuits?
- Direct the search to reduce the size of the search space

Minimizing the Number of Turns

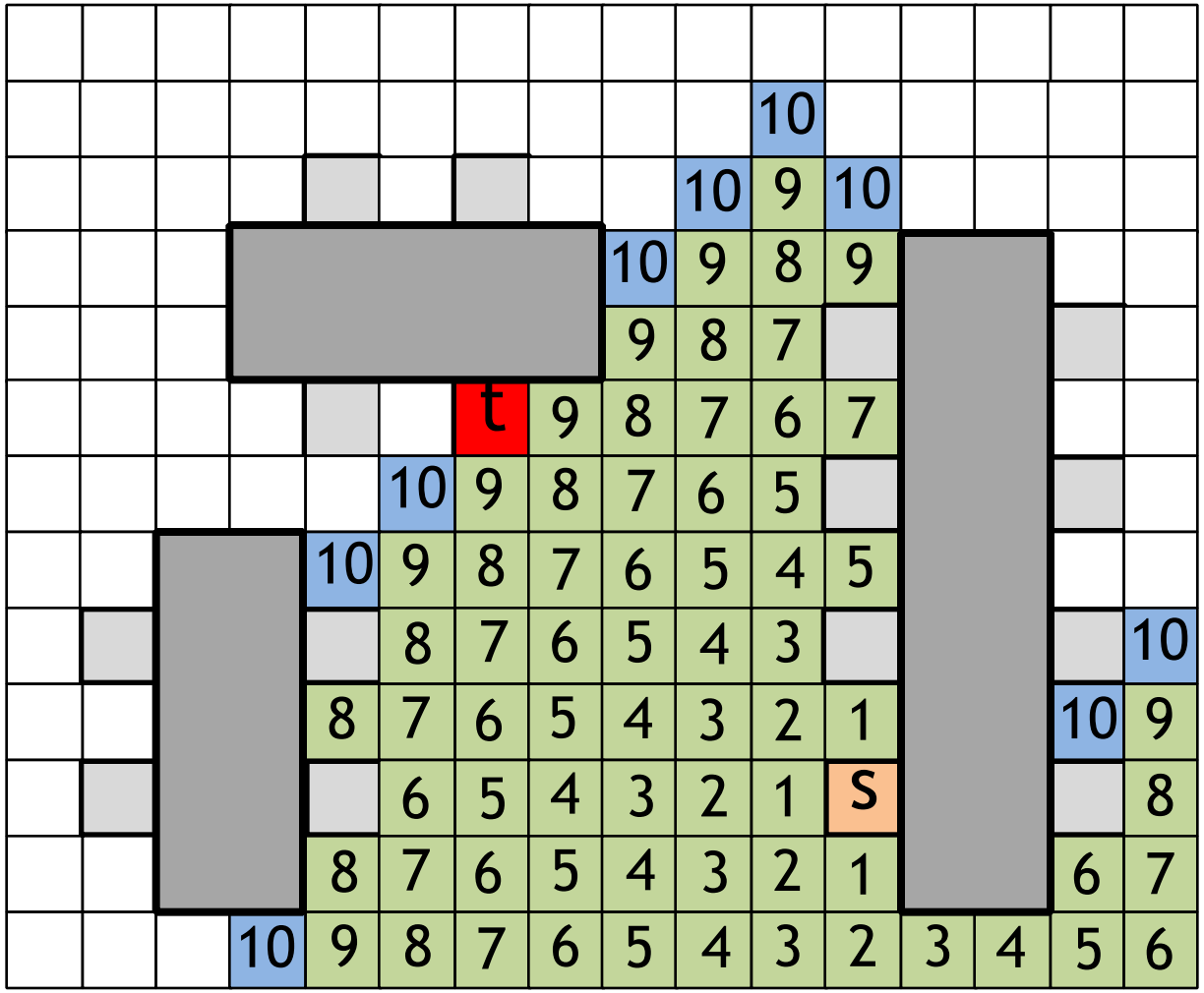
- Many paths of equal length…



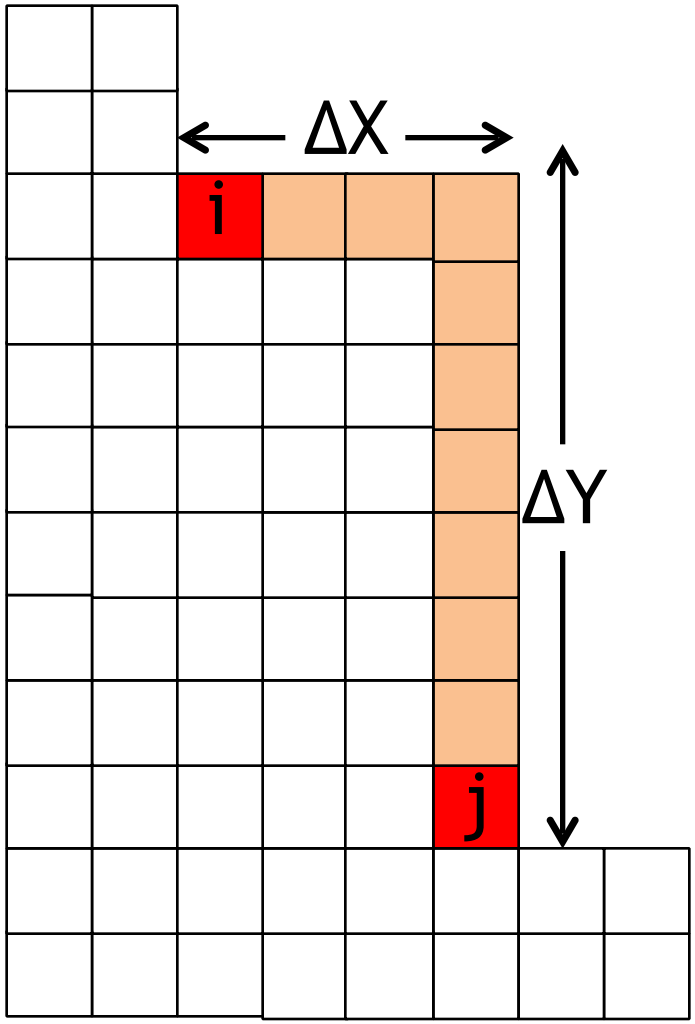
- Carefully consider the order of processing neighbors during wavefront expansion

Directing the Search

- Lee's Algorithm
  - Label (cost) is the distance from s to t
  - Searching from s in directions not toward t is only useful if there are blockages between s and t

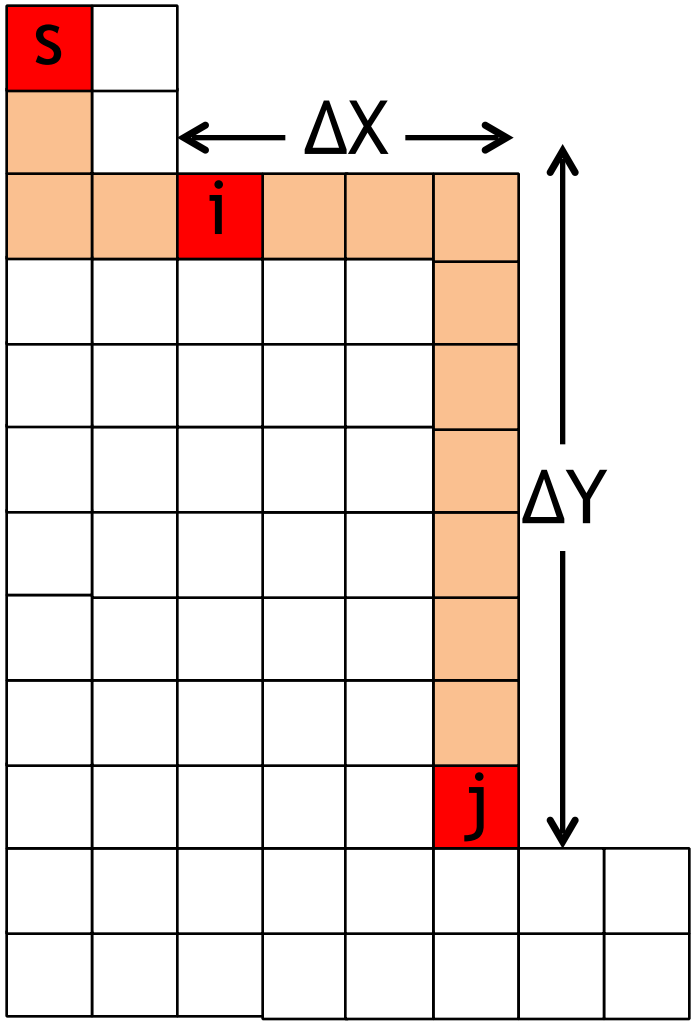


Manhattan Distance



$$MD(i, j) = \Delta X + \Delta Y$$

New Cost Function

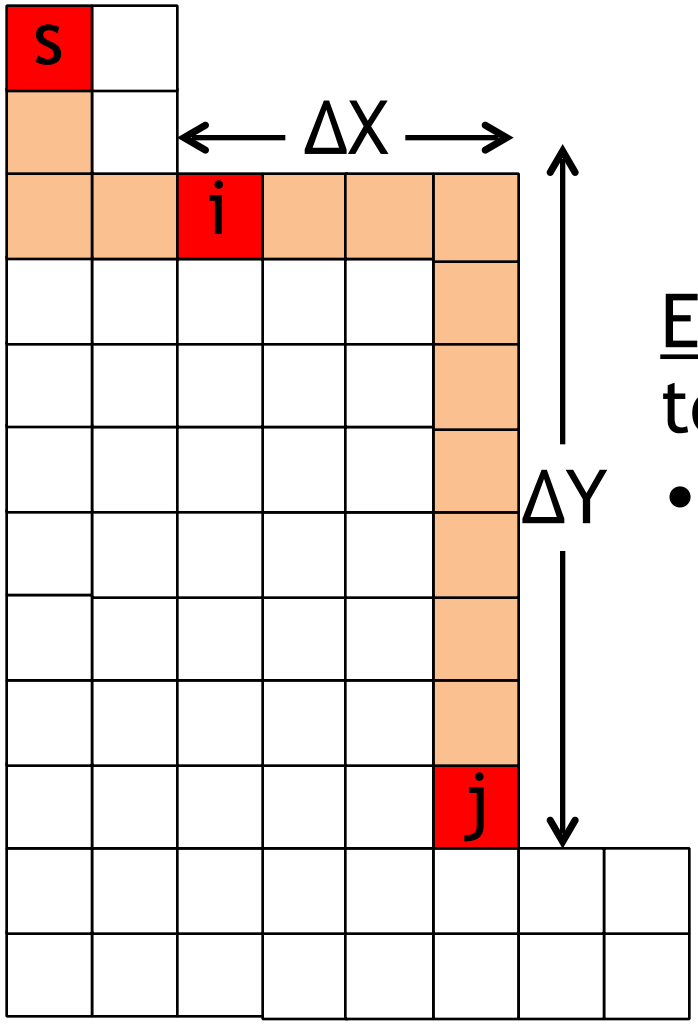


$C(i) = D(s, i) + MD(i, t)$

Distance from s to i

- Known from the search

New Cost Function

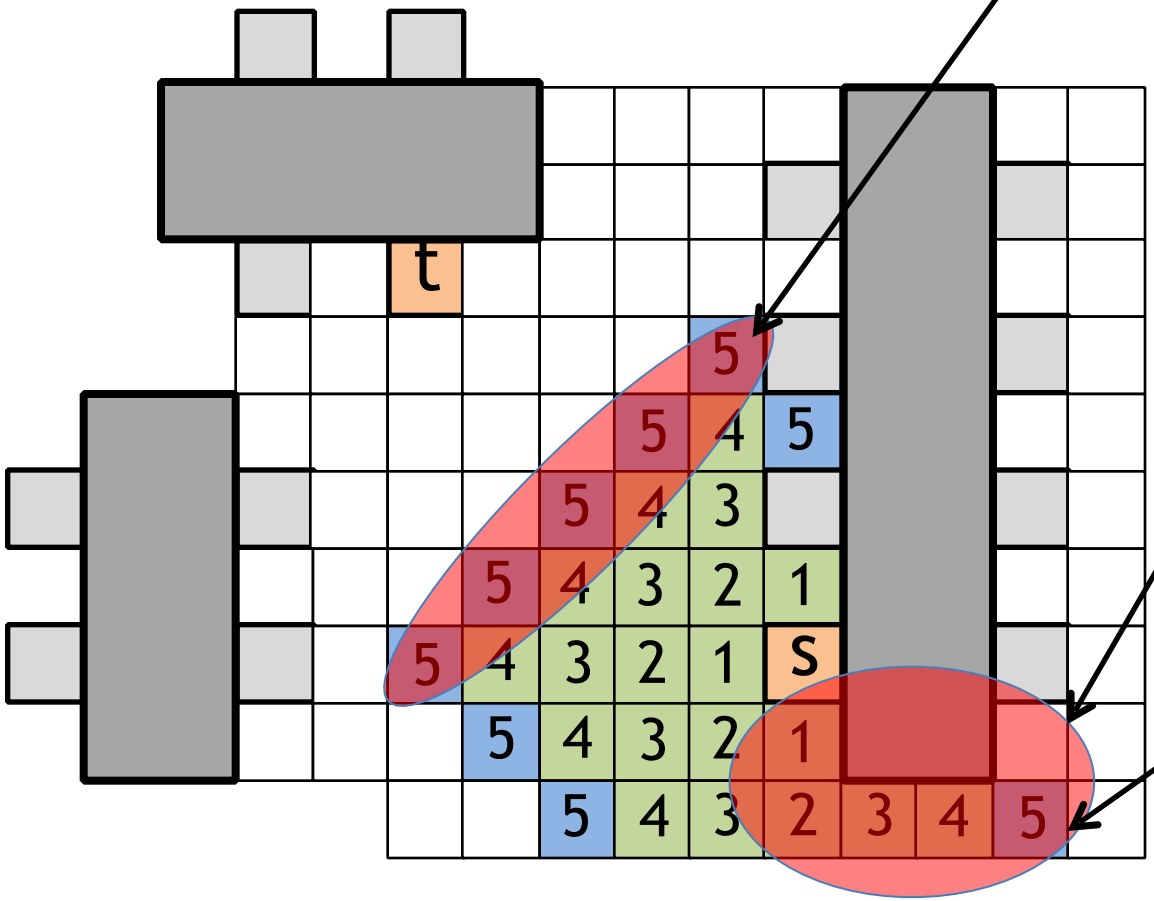


$$C(i) = D(s, i) + MD(i, t)$$

Estimate of the distance from i to t (Manhattan Distance)

- Blockages on paths from i to t are uncovered during the search

Back to Lee's



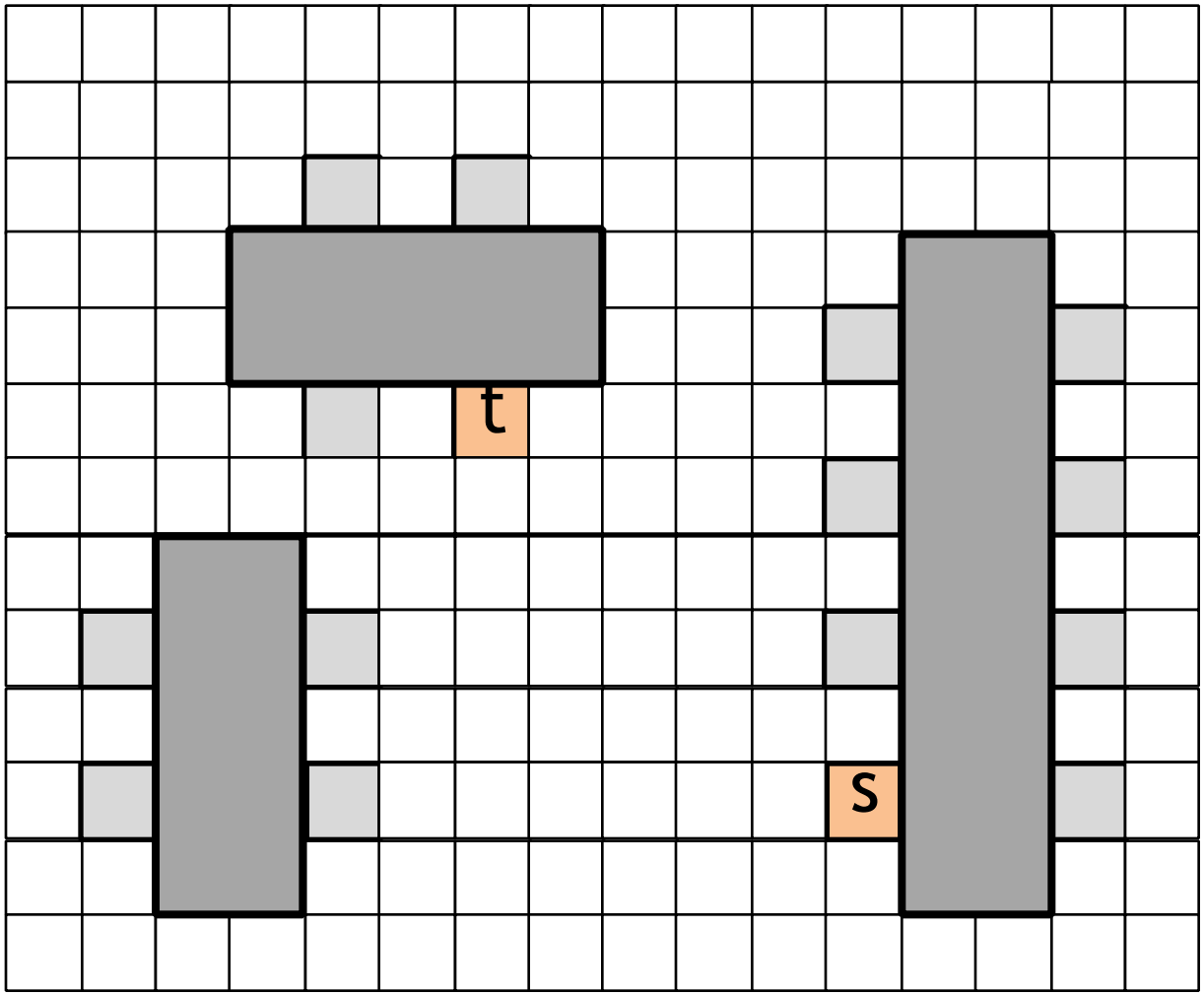
Expand along this region of the wavefront

D(s, i) = 5
MD(i, t) = 5
C(i) = 10

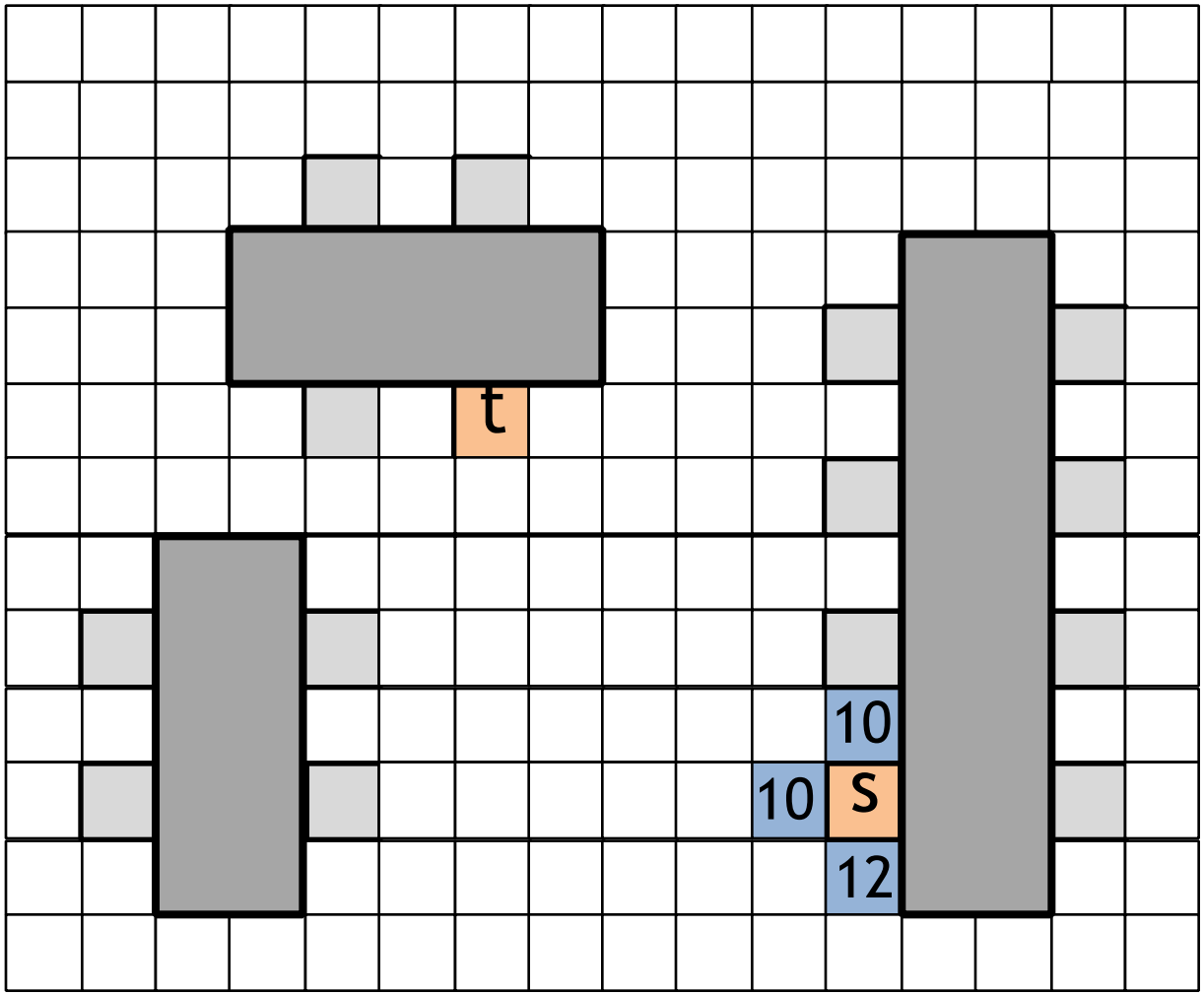
We never would have wasted time expanding in this direction

D(s, i) = 5
MD(i, t) = 15
C(i) = 20

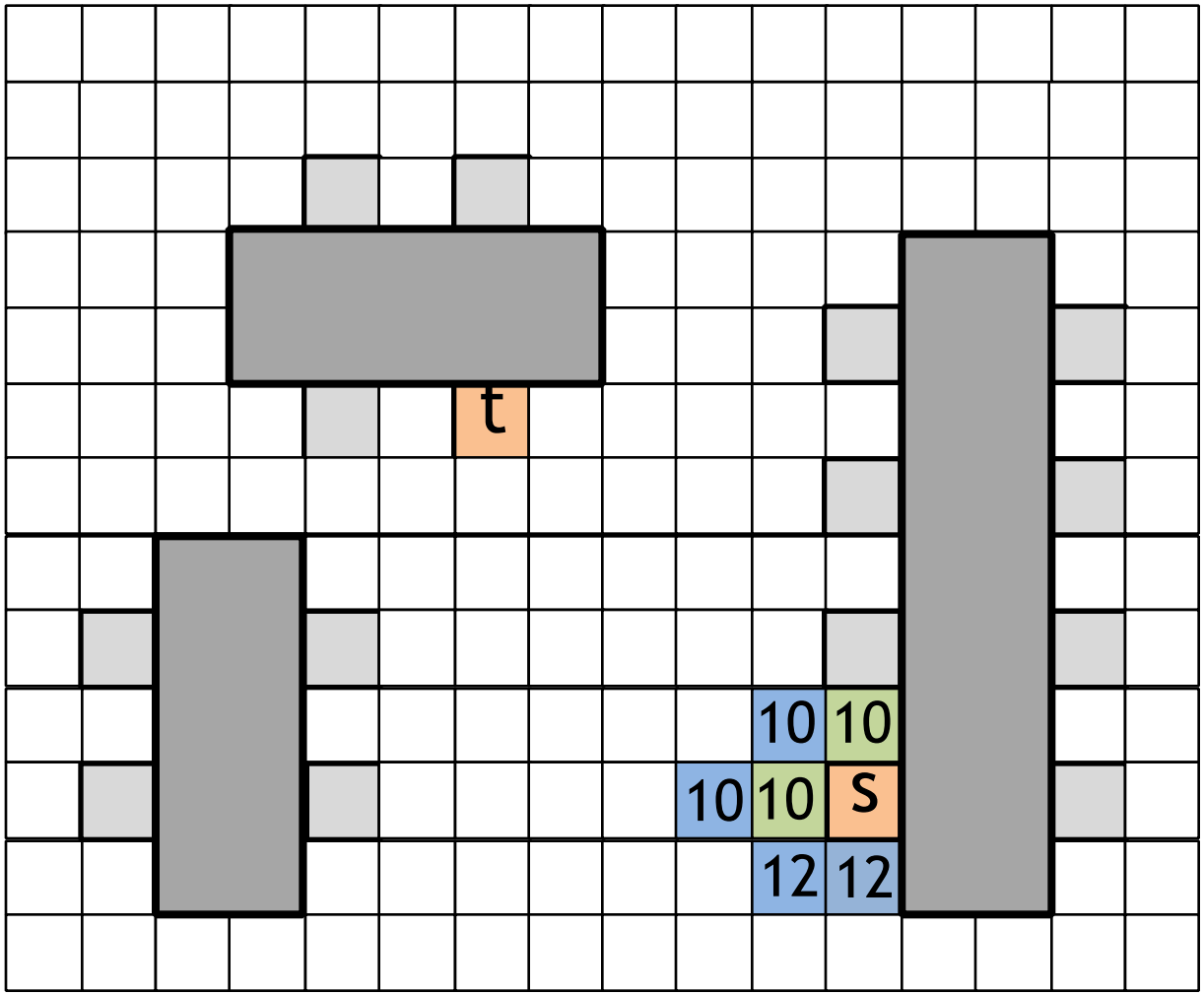
Lee's with Ruben's Costs



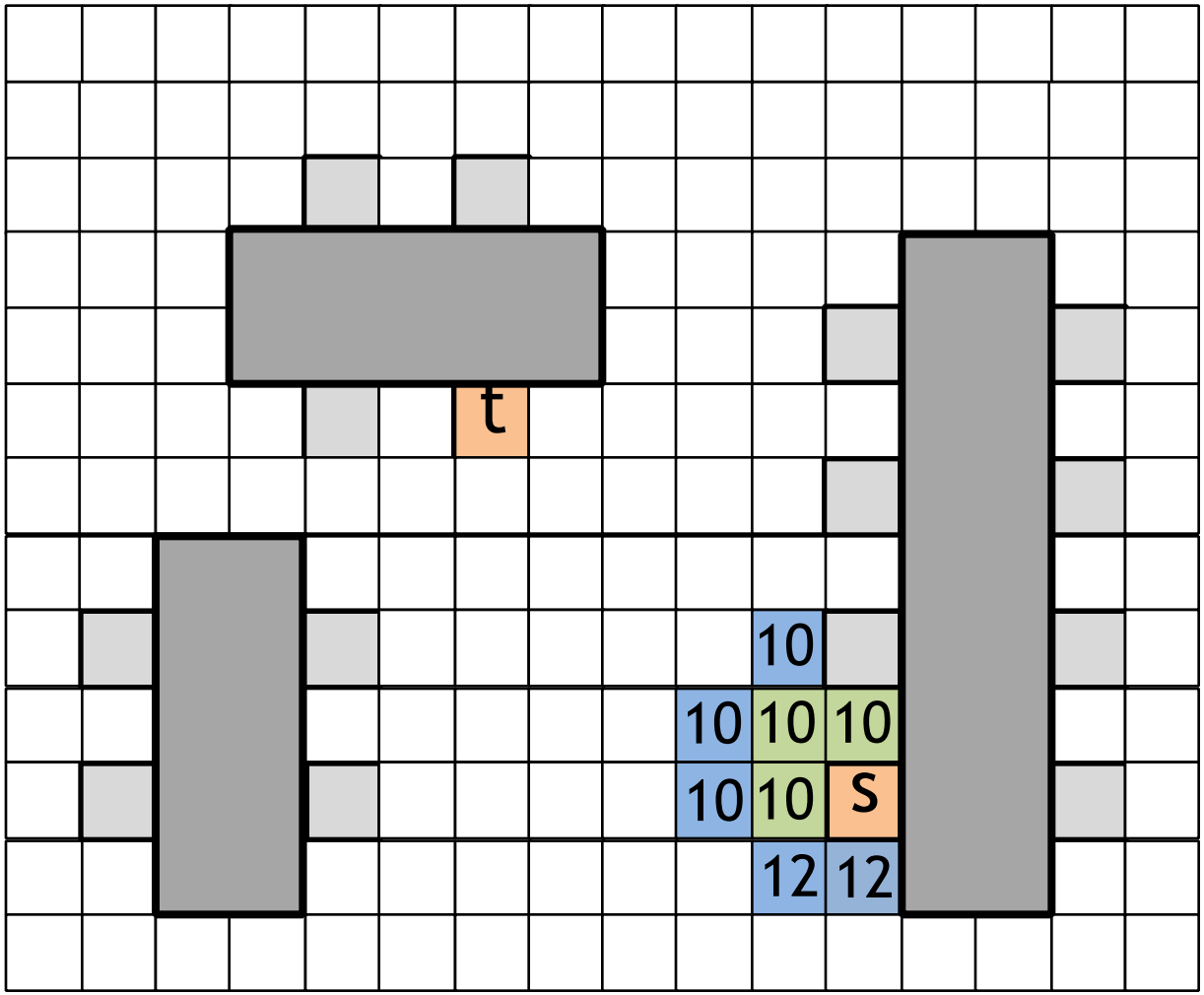
Lee's with Ruben's Costs



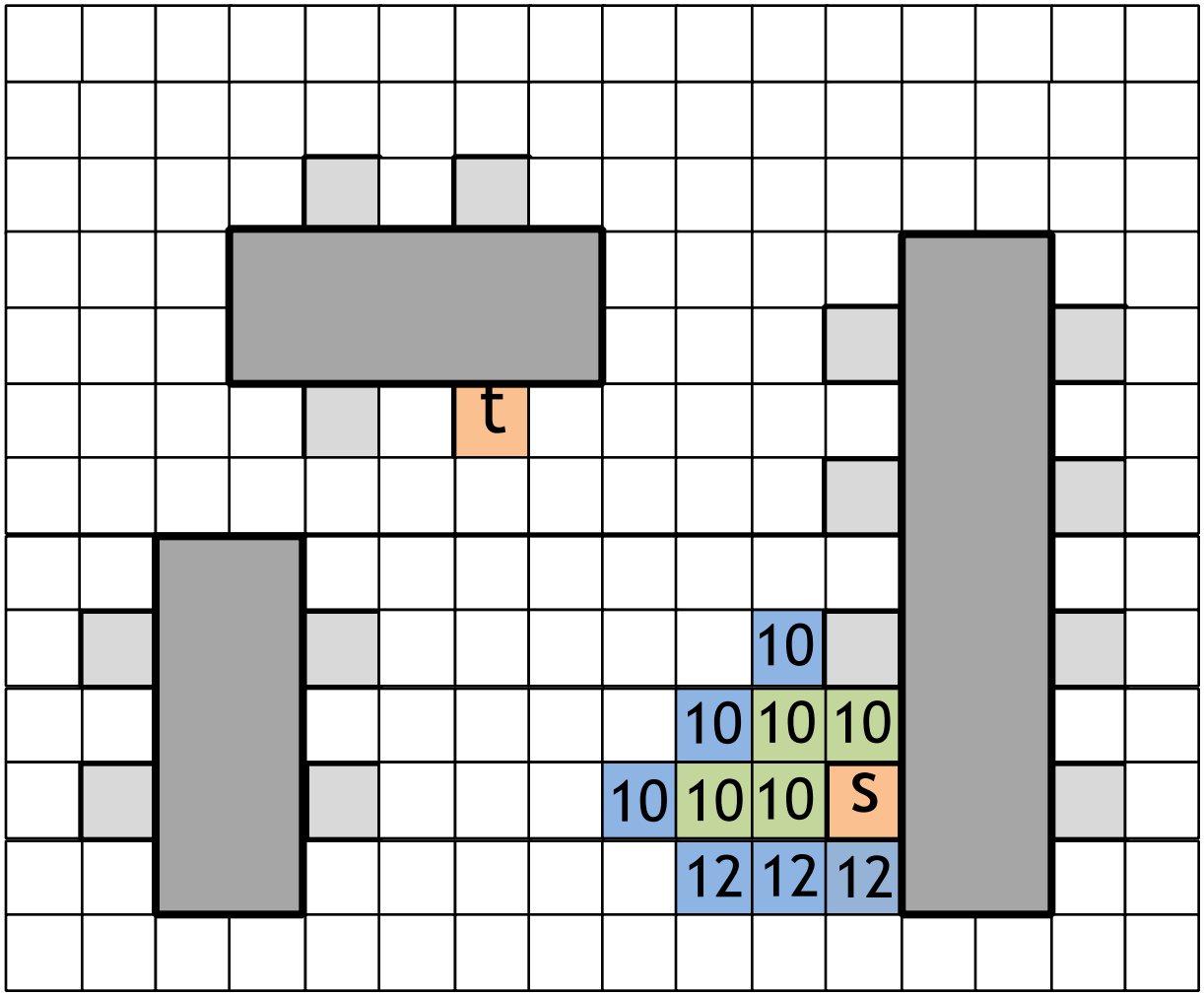
Lee's with Ruben's Costs



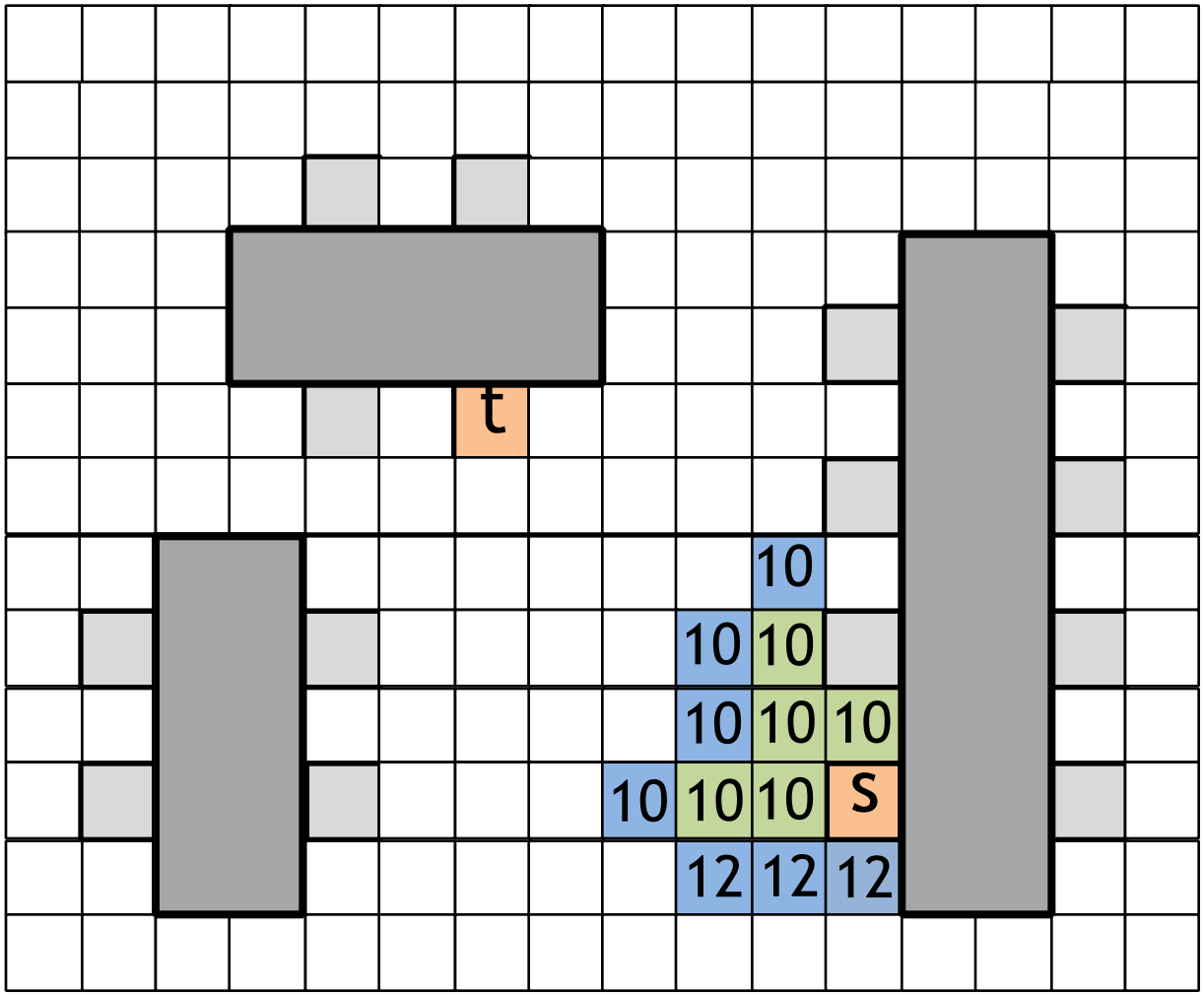
Lee's with Ruben's Costs



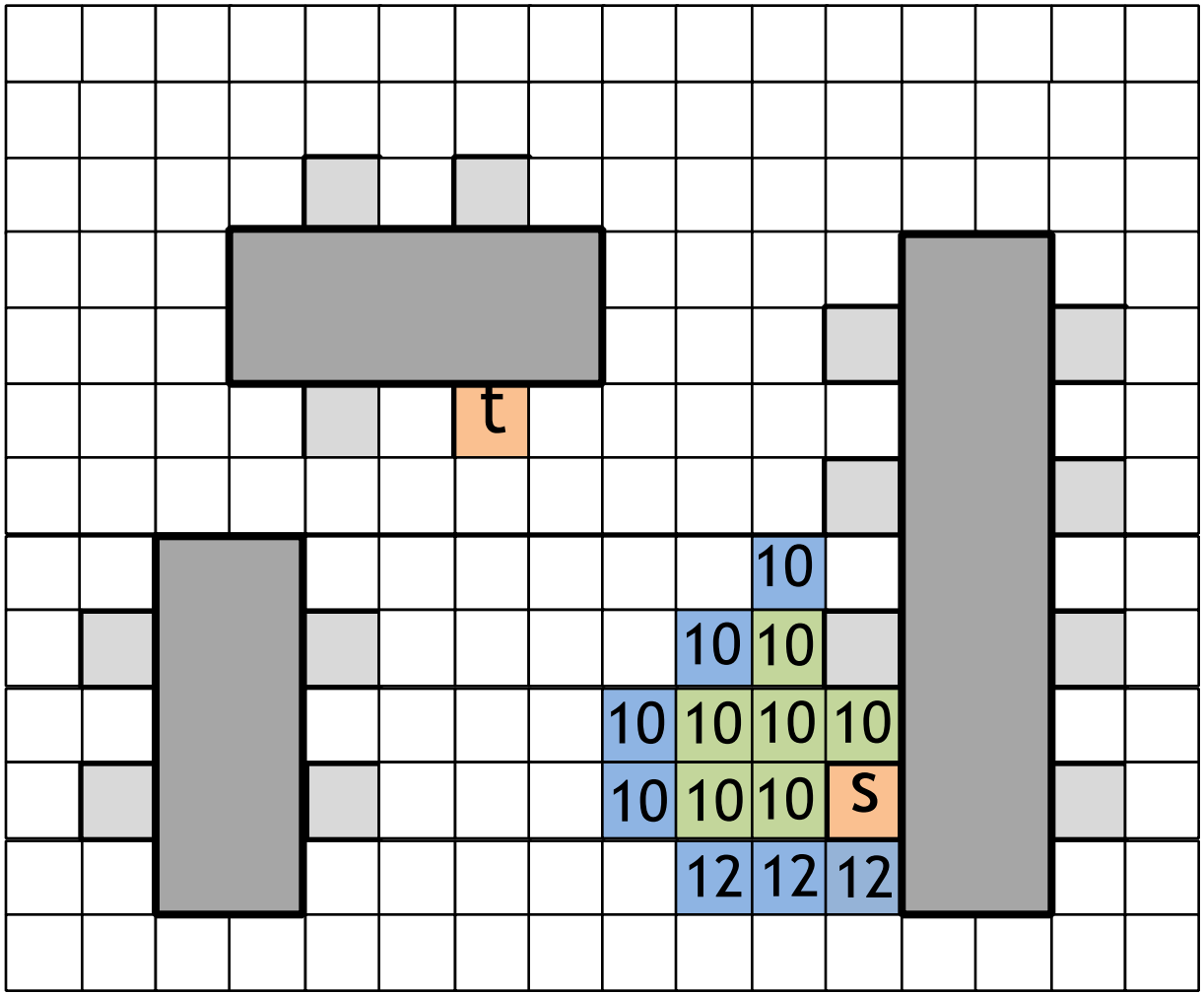
Lee's with Ruben's Costs



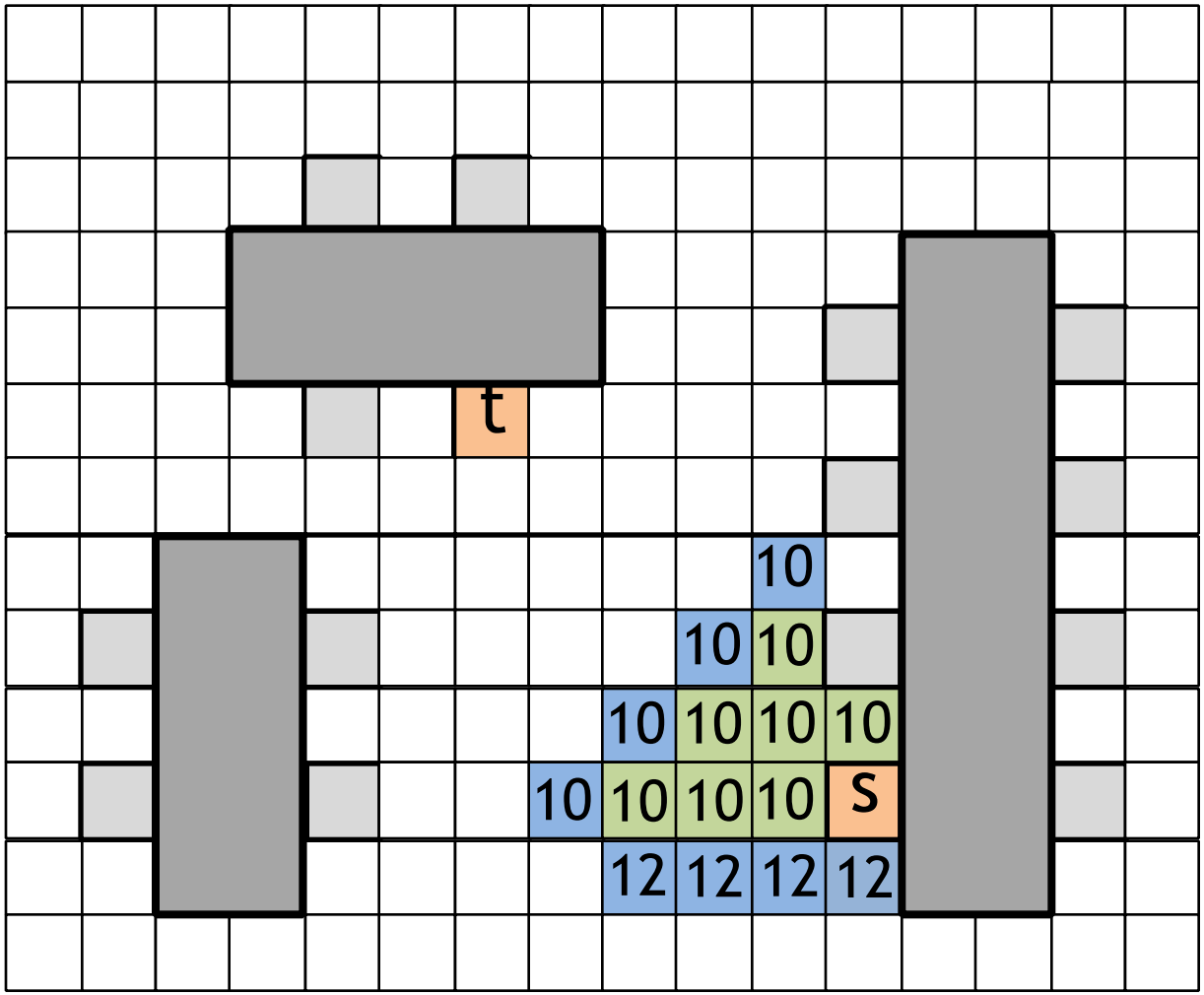
Lee's with Ruben's Costs



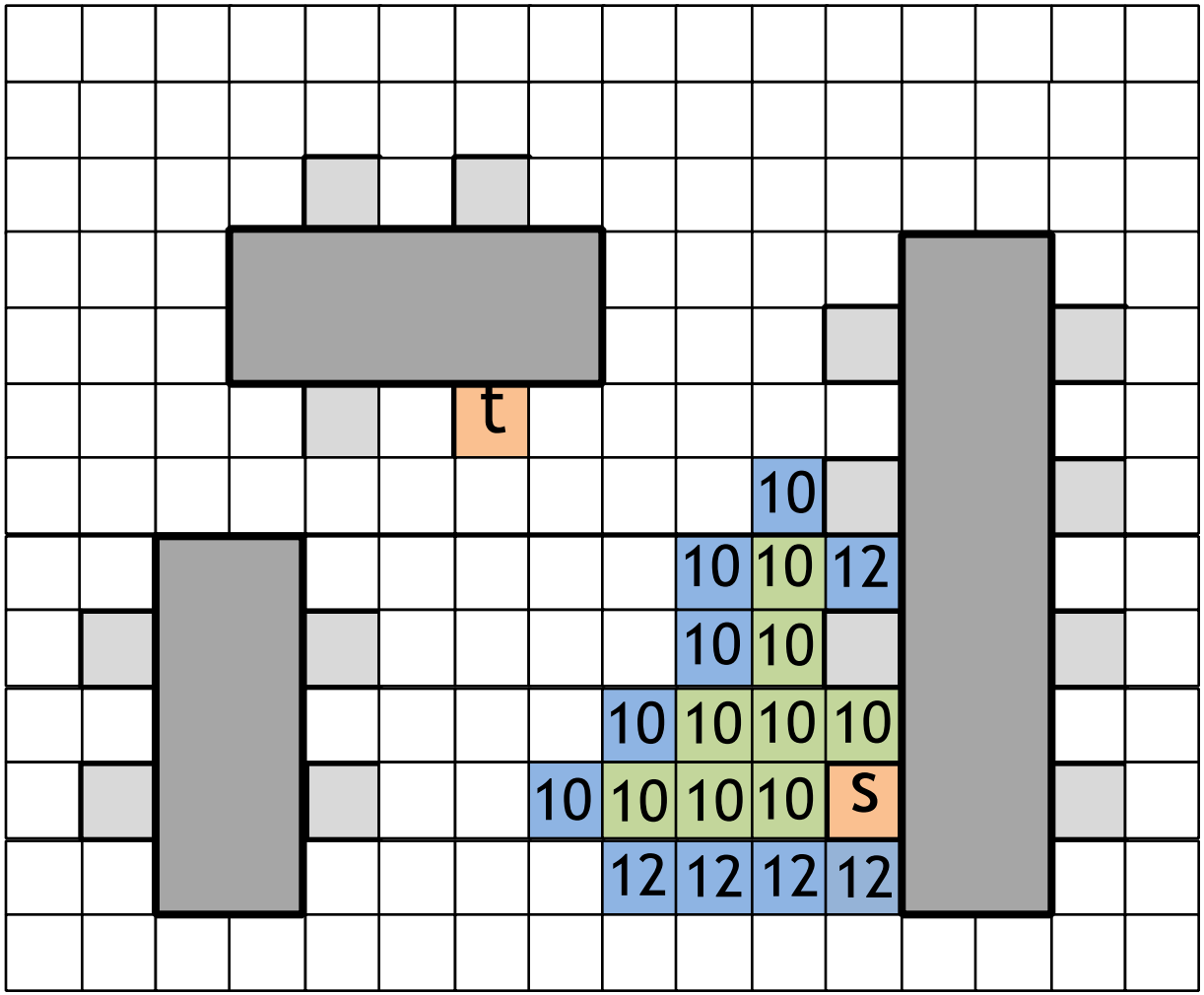
Lee's with Ruben's Costs



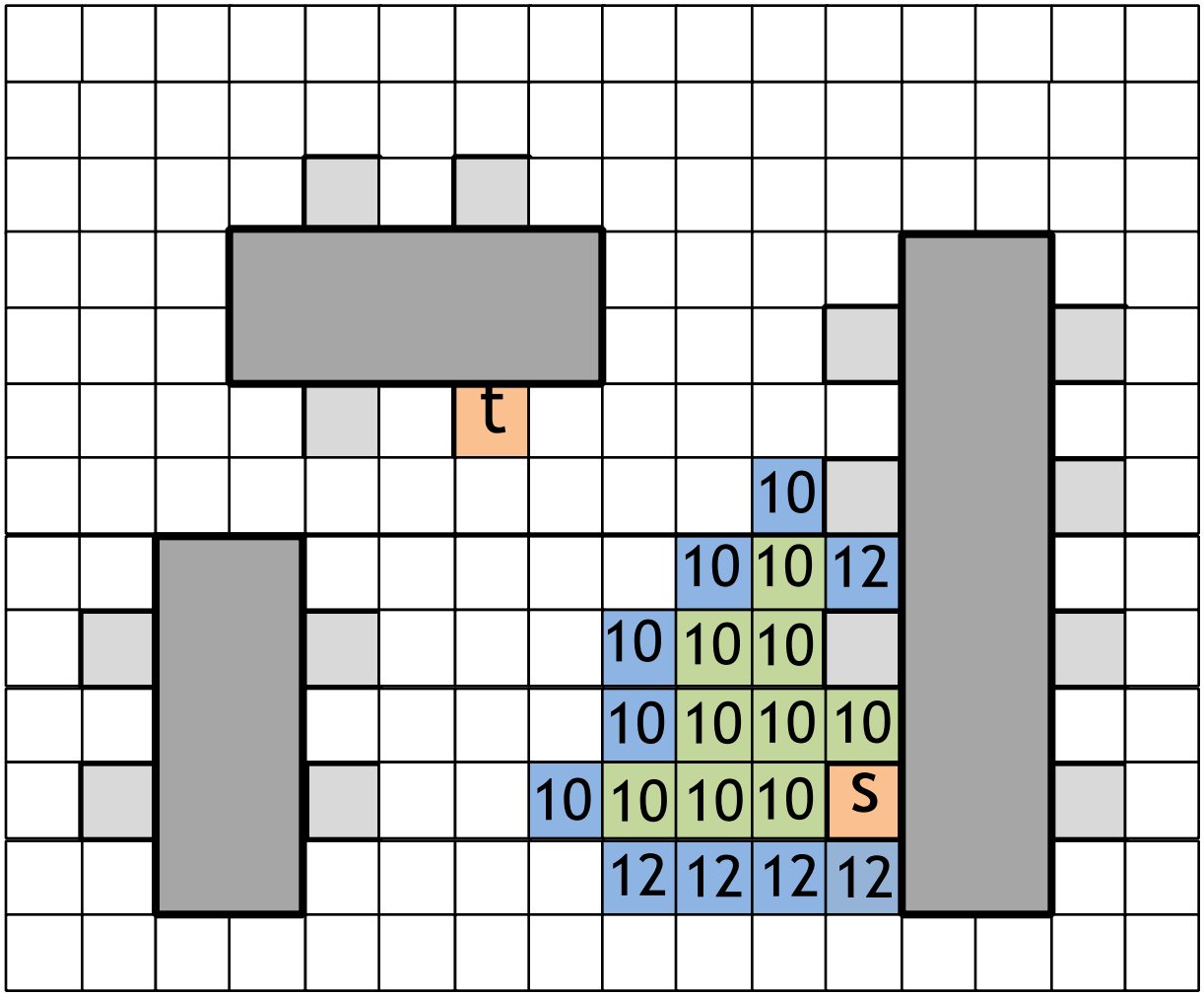
Lee's with Ruben's Costs



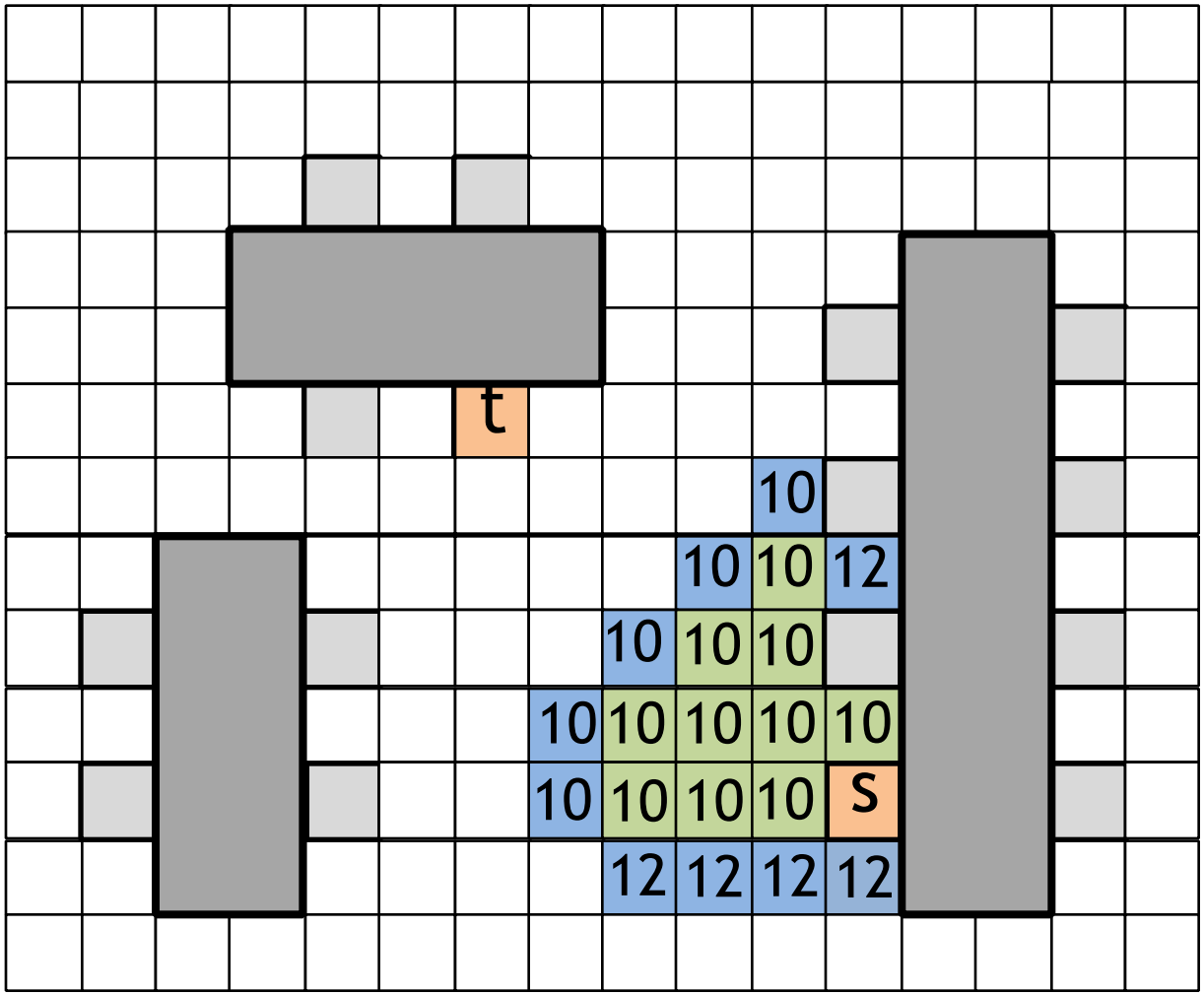
Lee's with Ruben's Costs



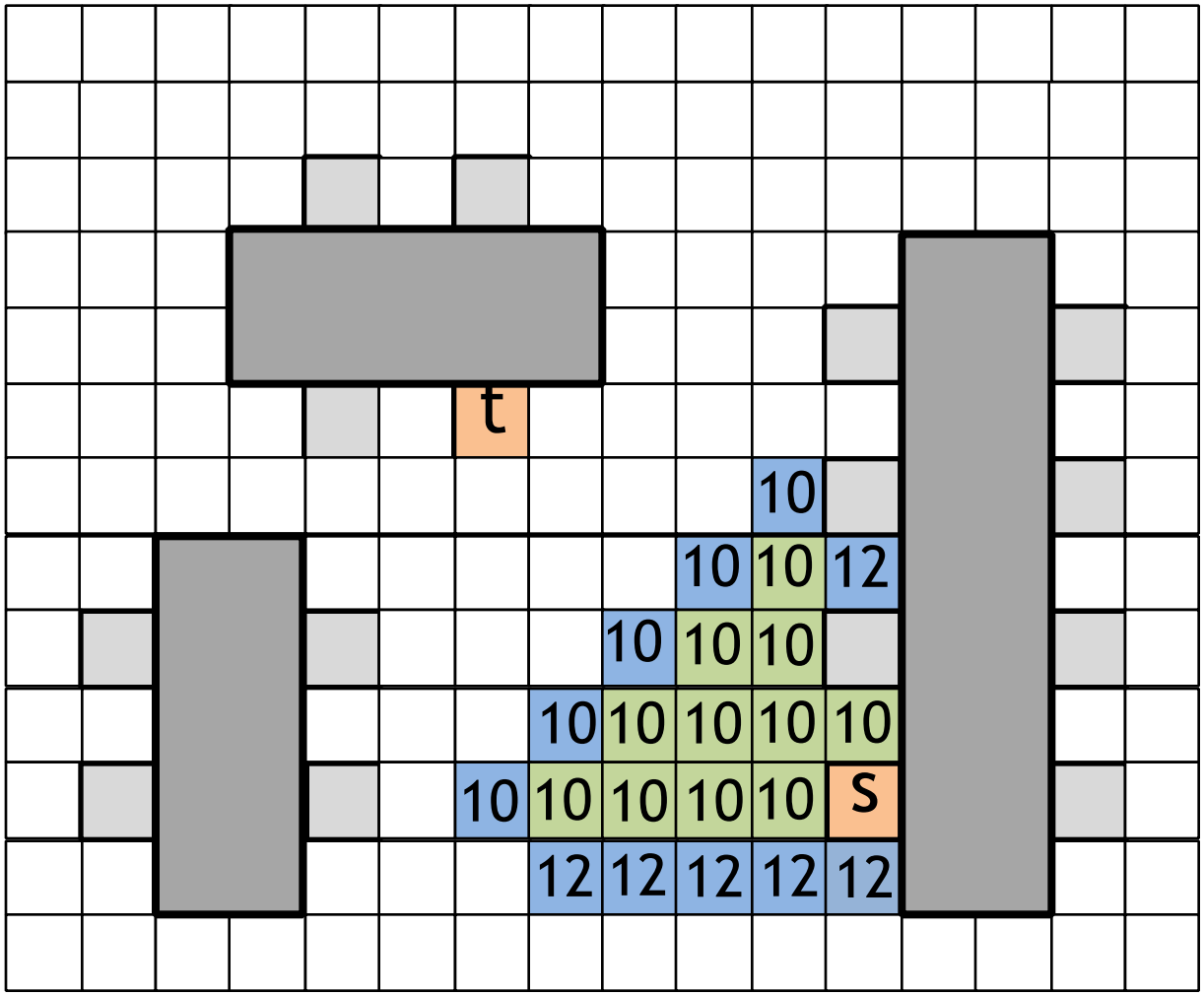
Lee's with Ruben's Costs



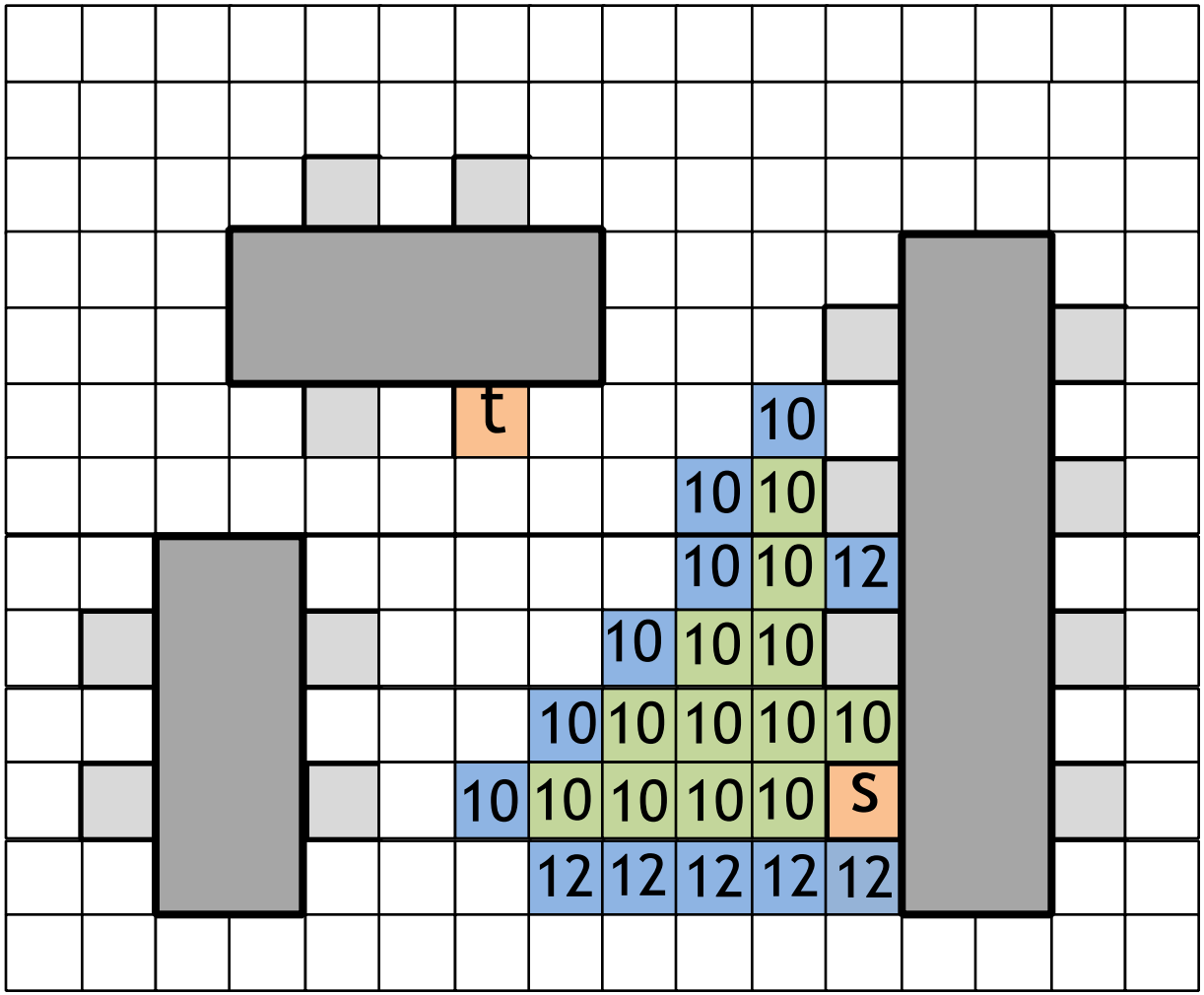
Lee's with Ruben's Costs



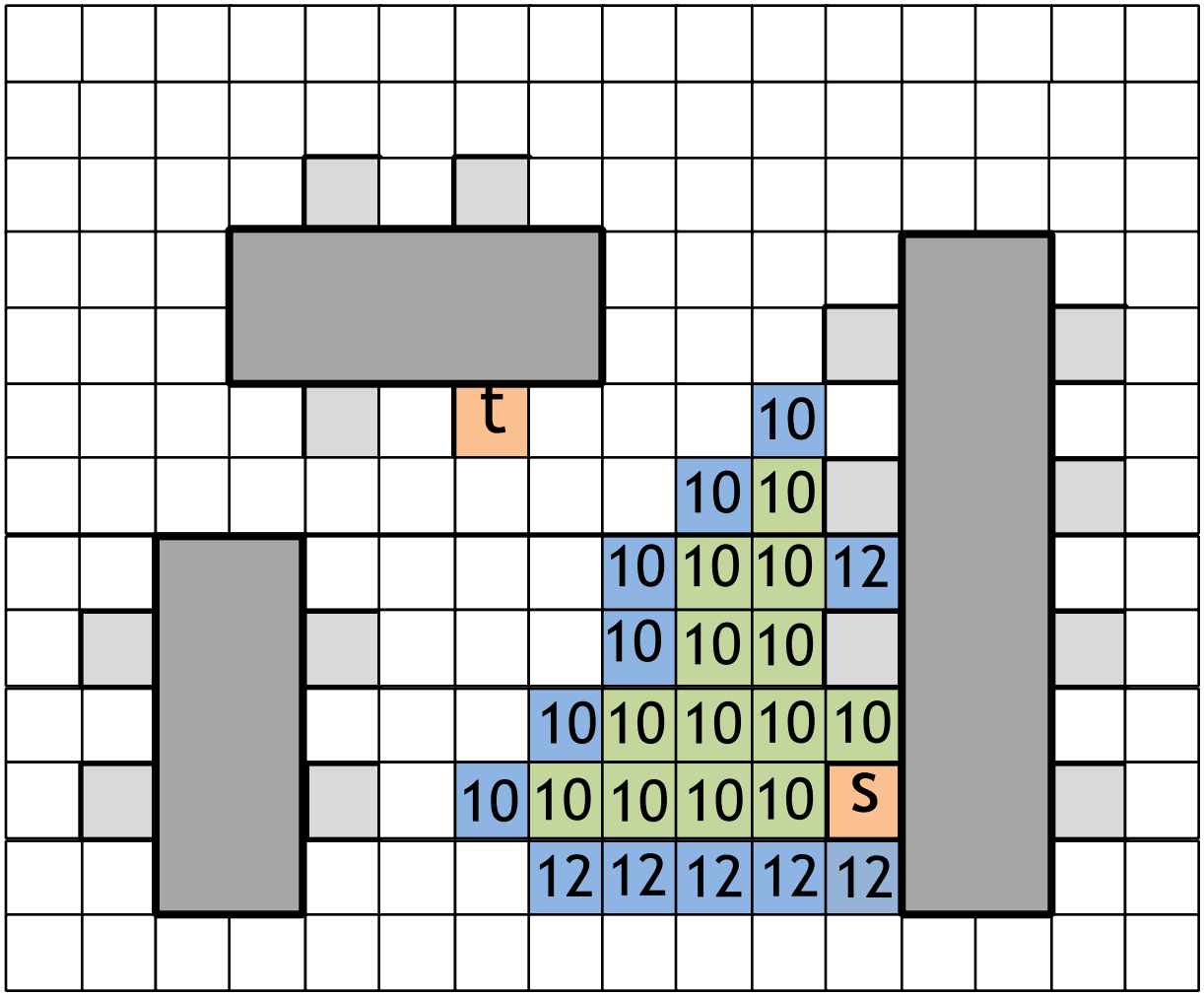
Lee's with Ruben's Costs



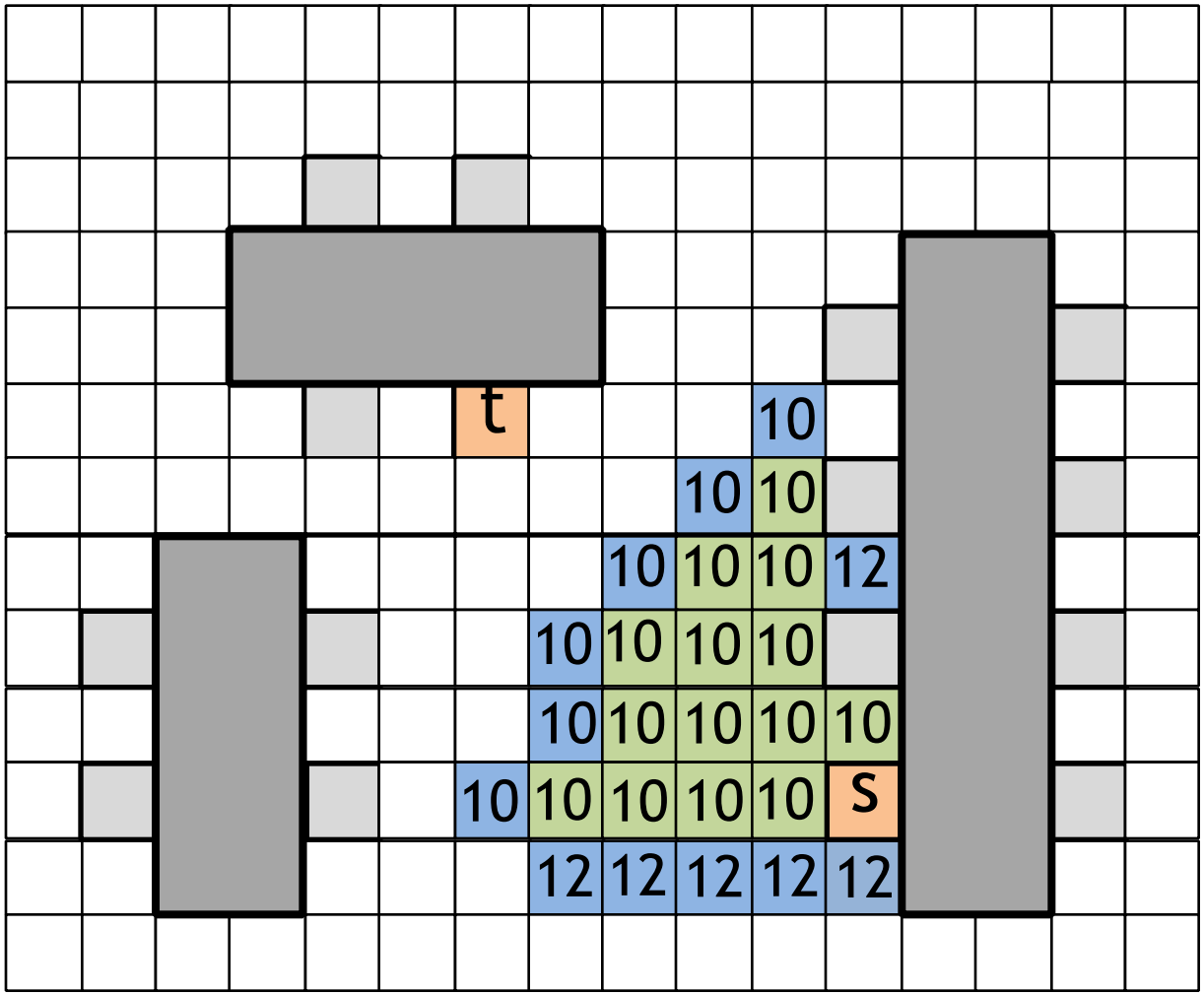
Lee's with Ruben's Costs



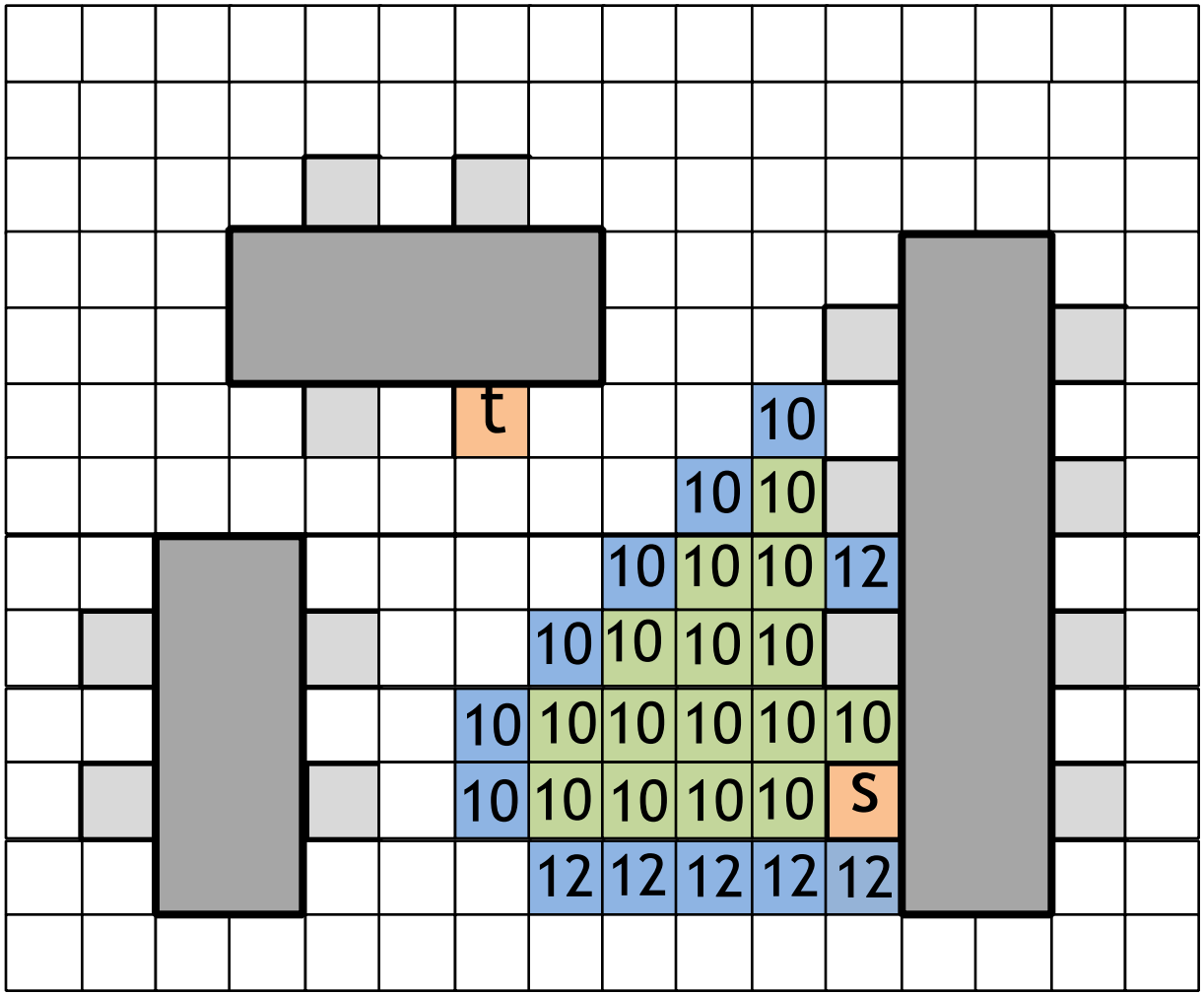
Lee's with Ruben's Costs



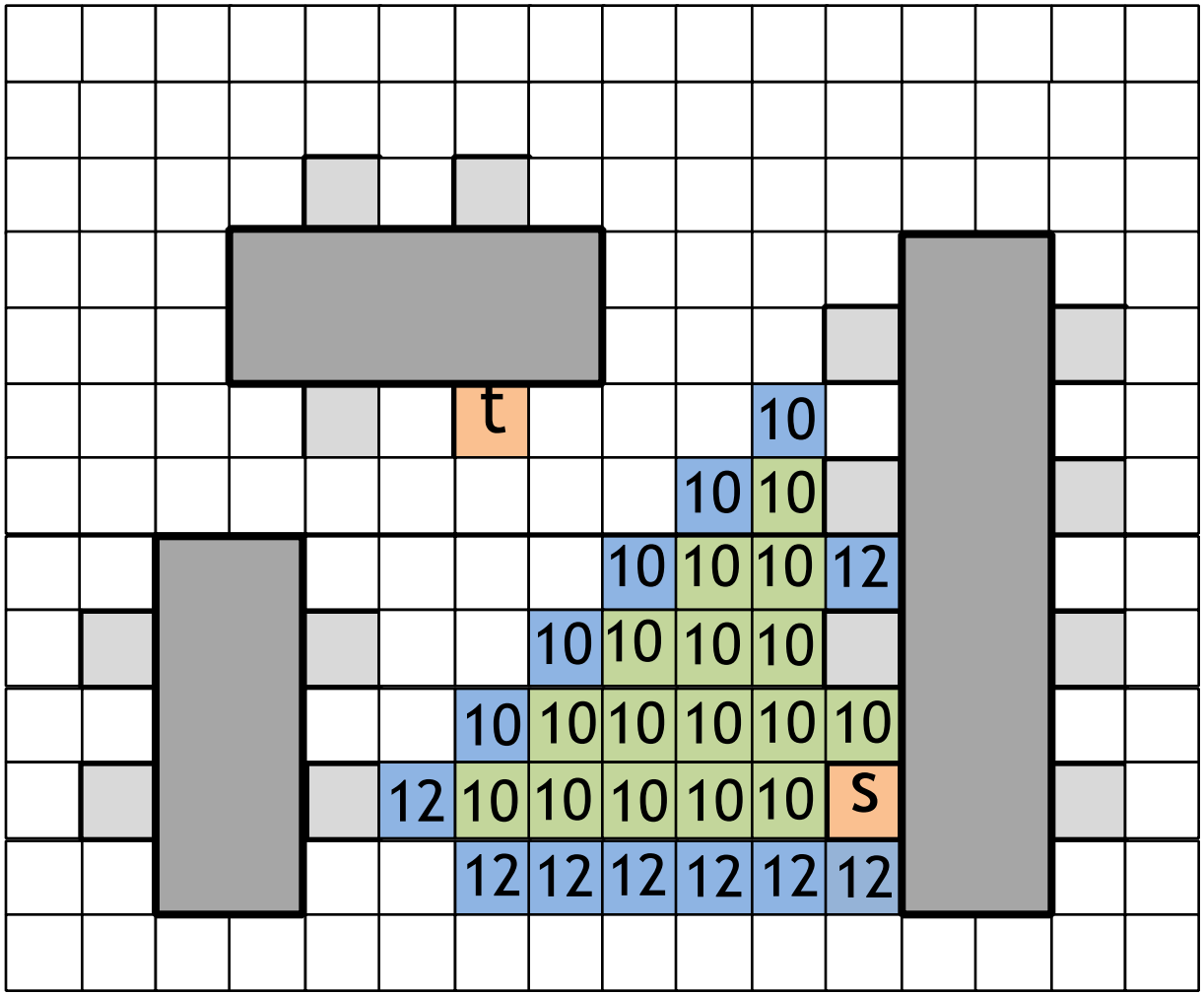
Lee's with Ruben's Costs



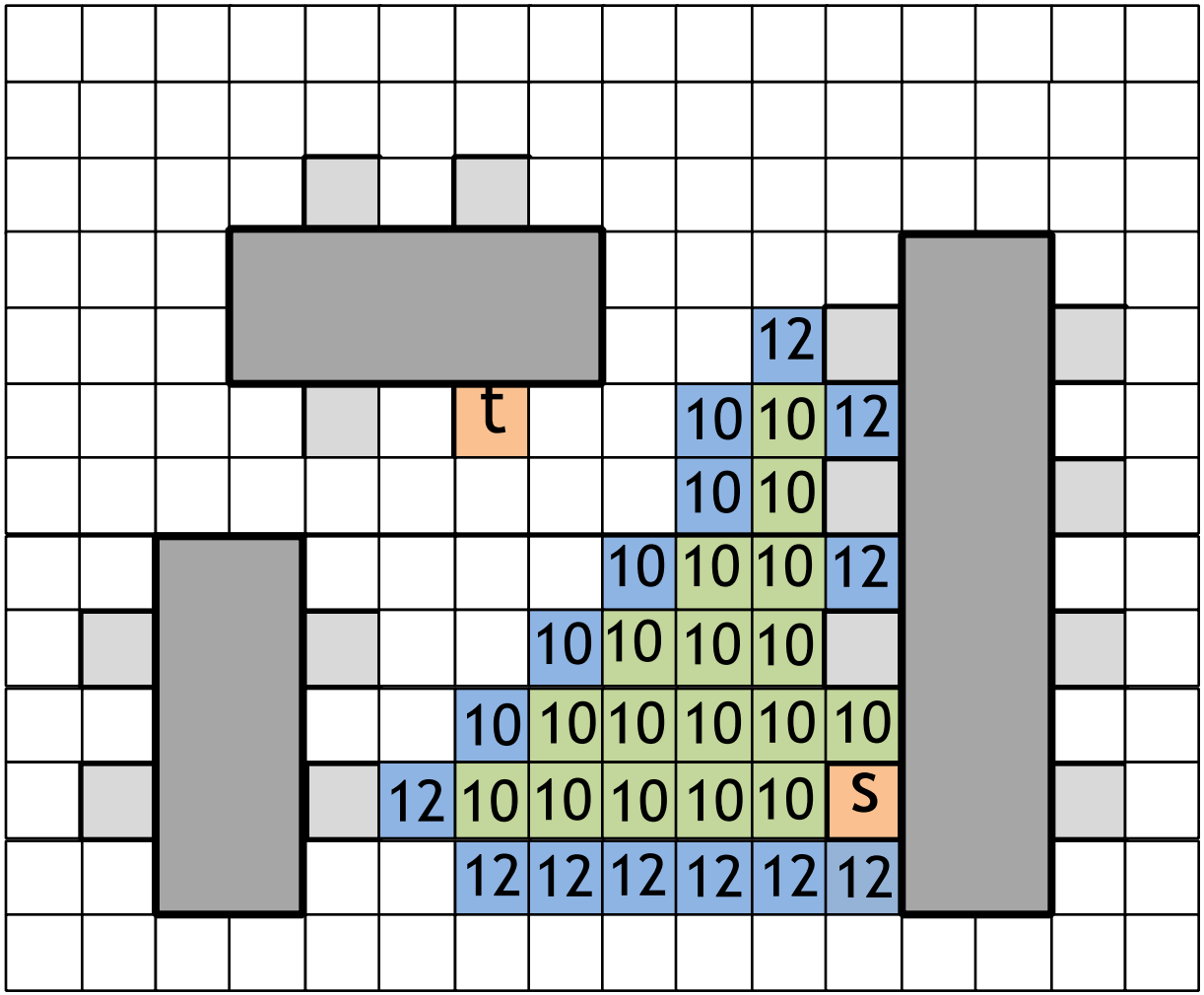
Lee's with Ruben's Costs



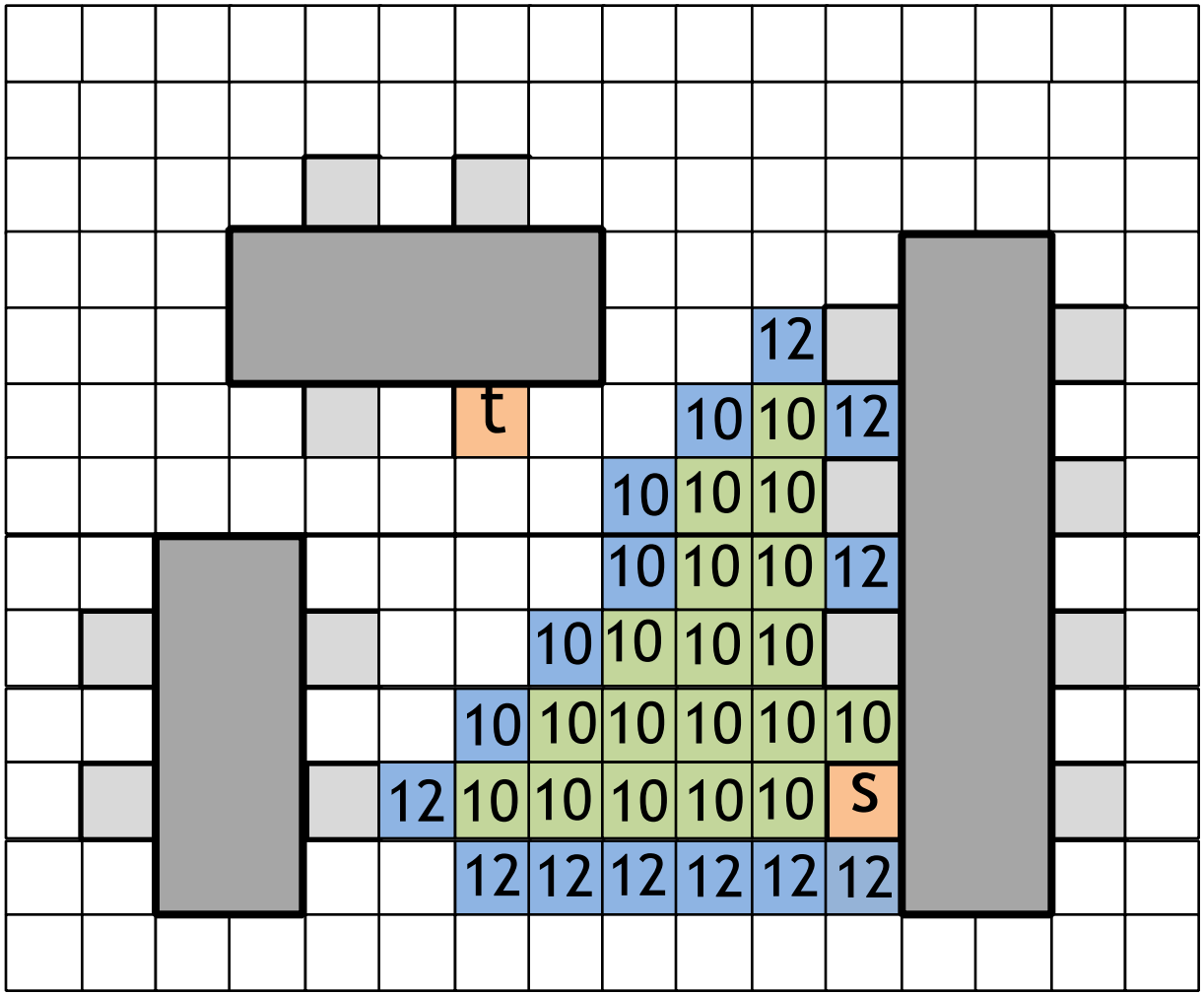
Lee's with Ruben's Costs



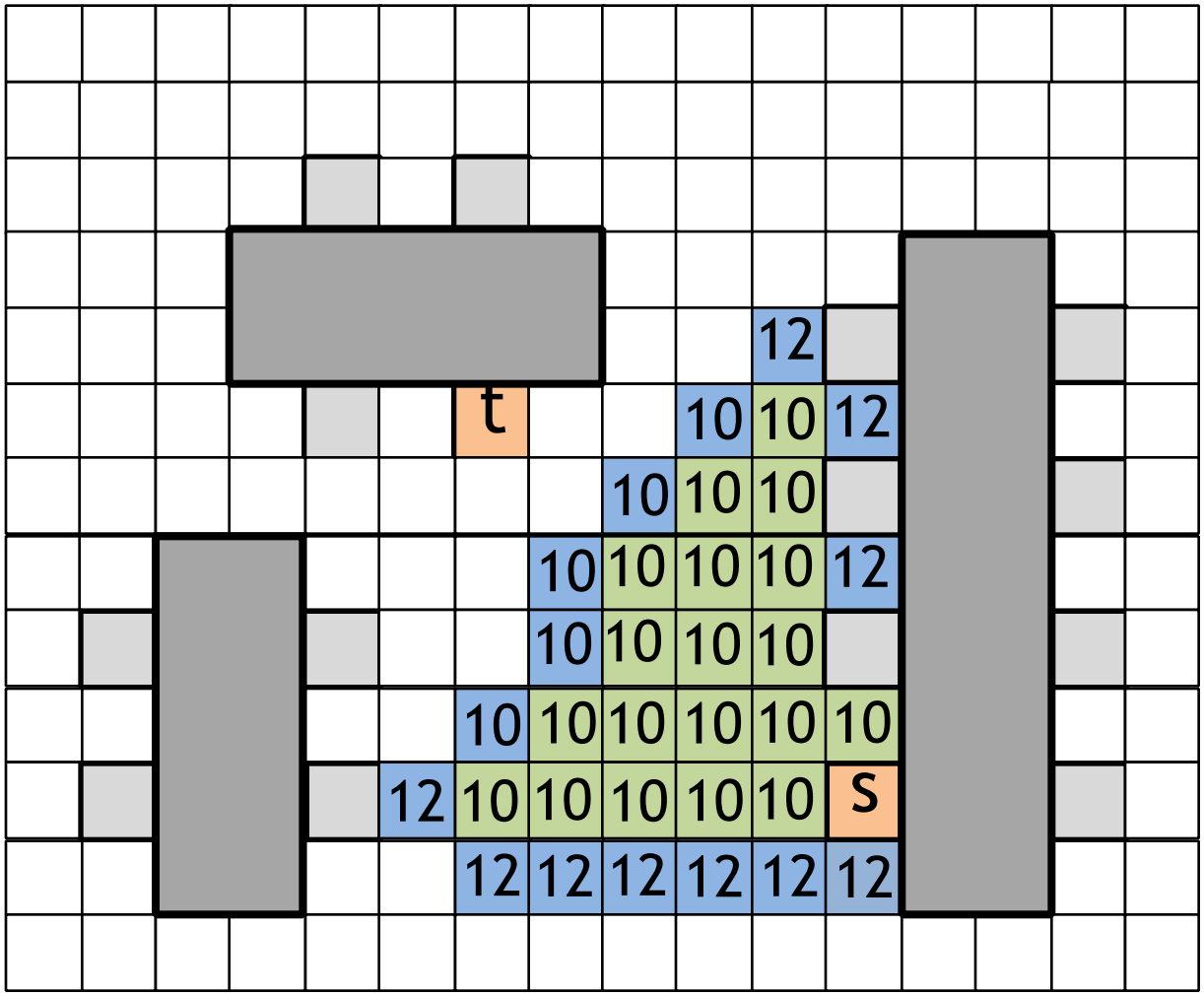
Lee's with Ruben's Costs



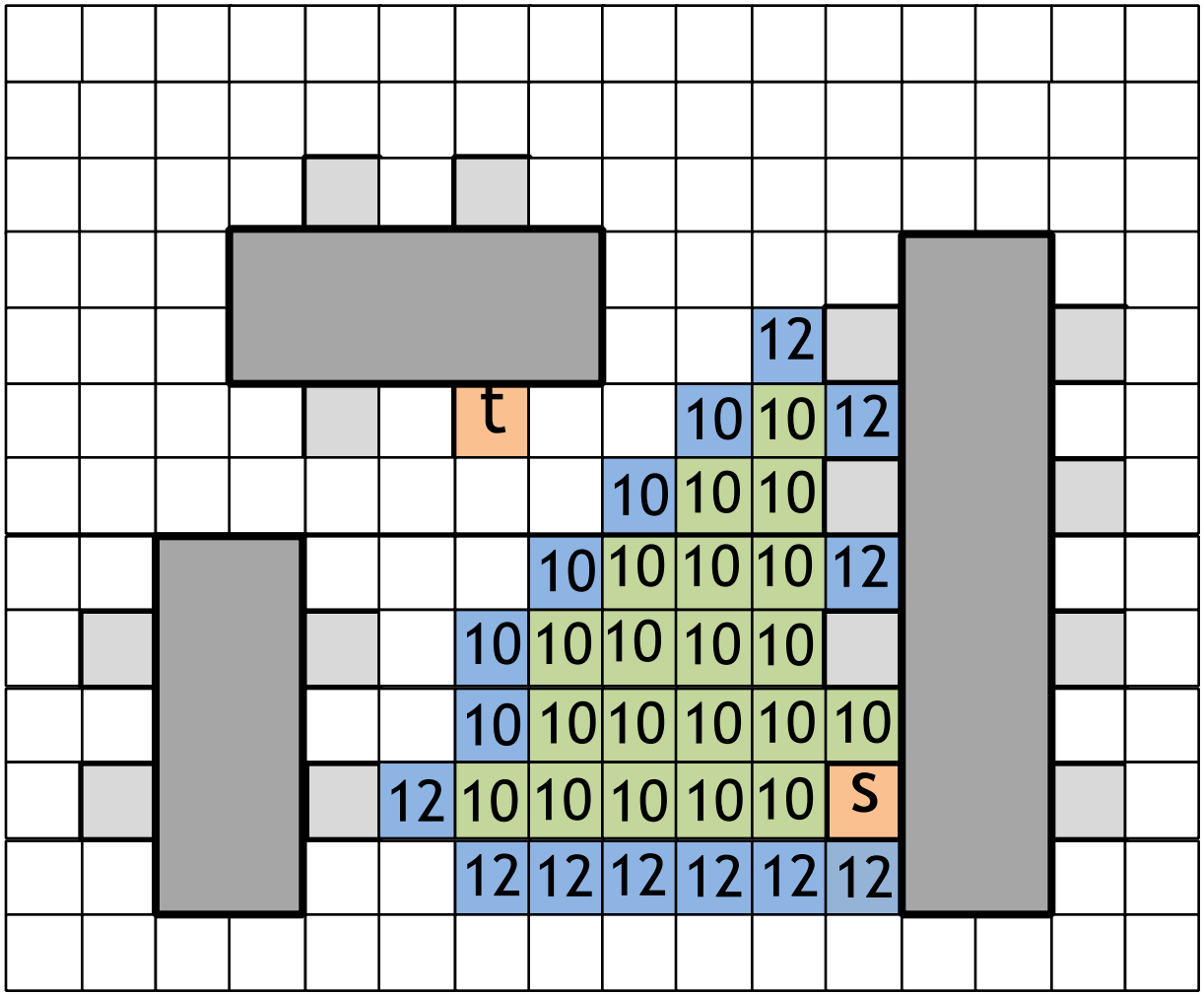
Lee's with Ruben's Costs



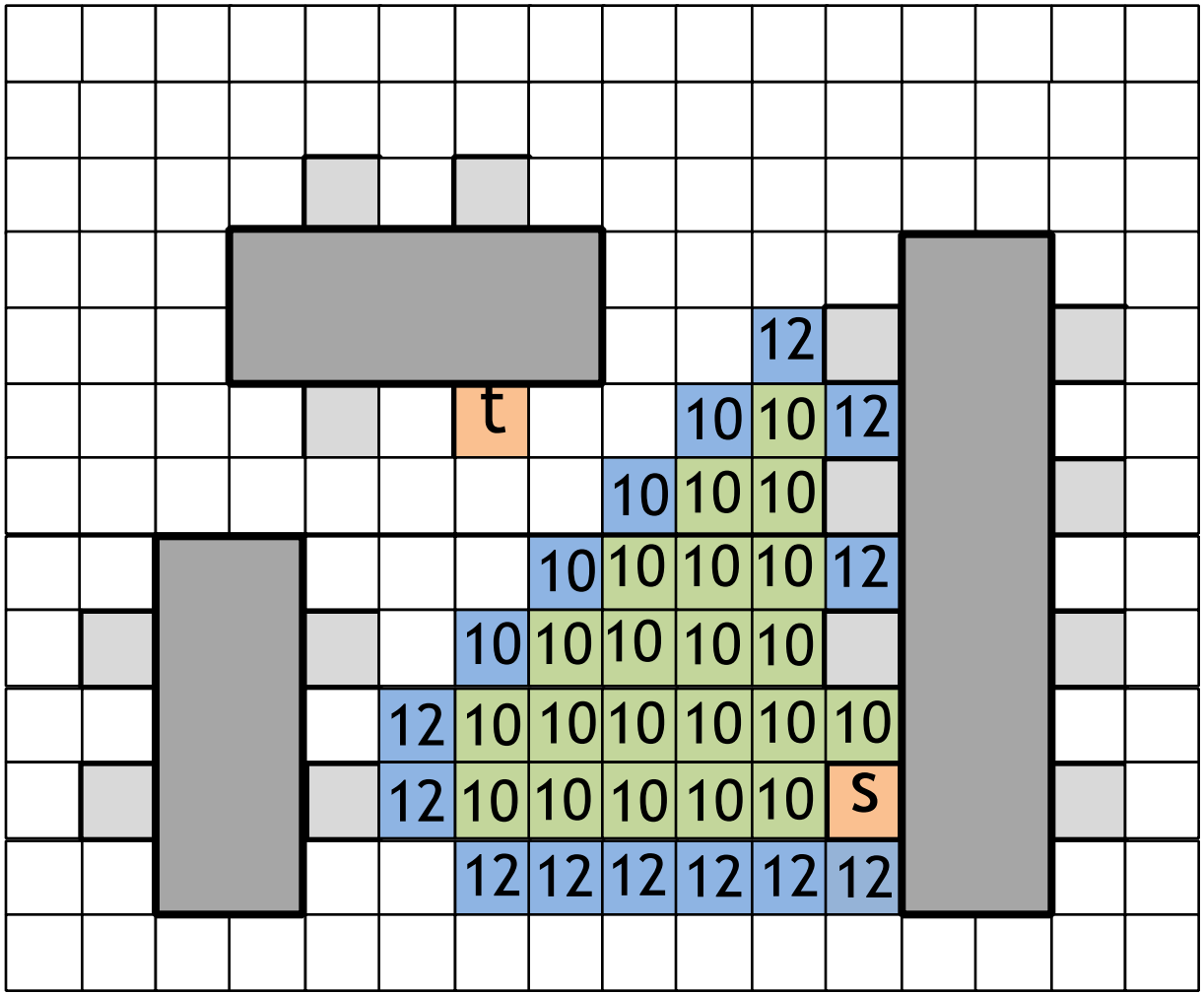
Lee's with Ruben's Costs



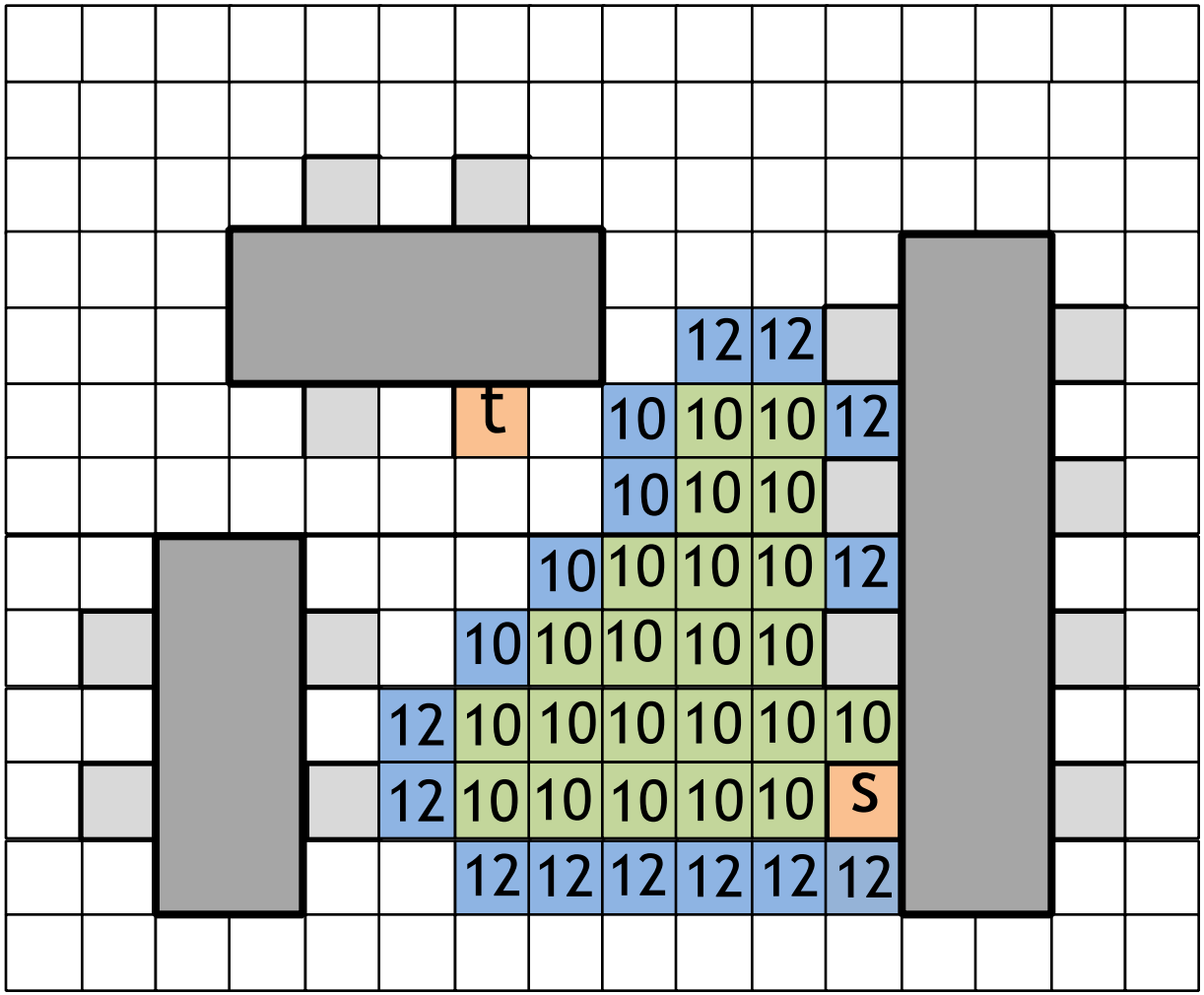
Lee's with Ruben's Costs



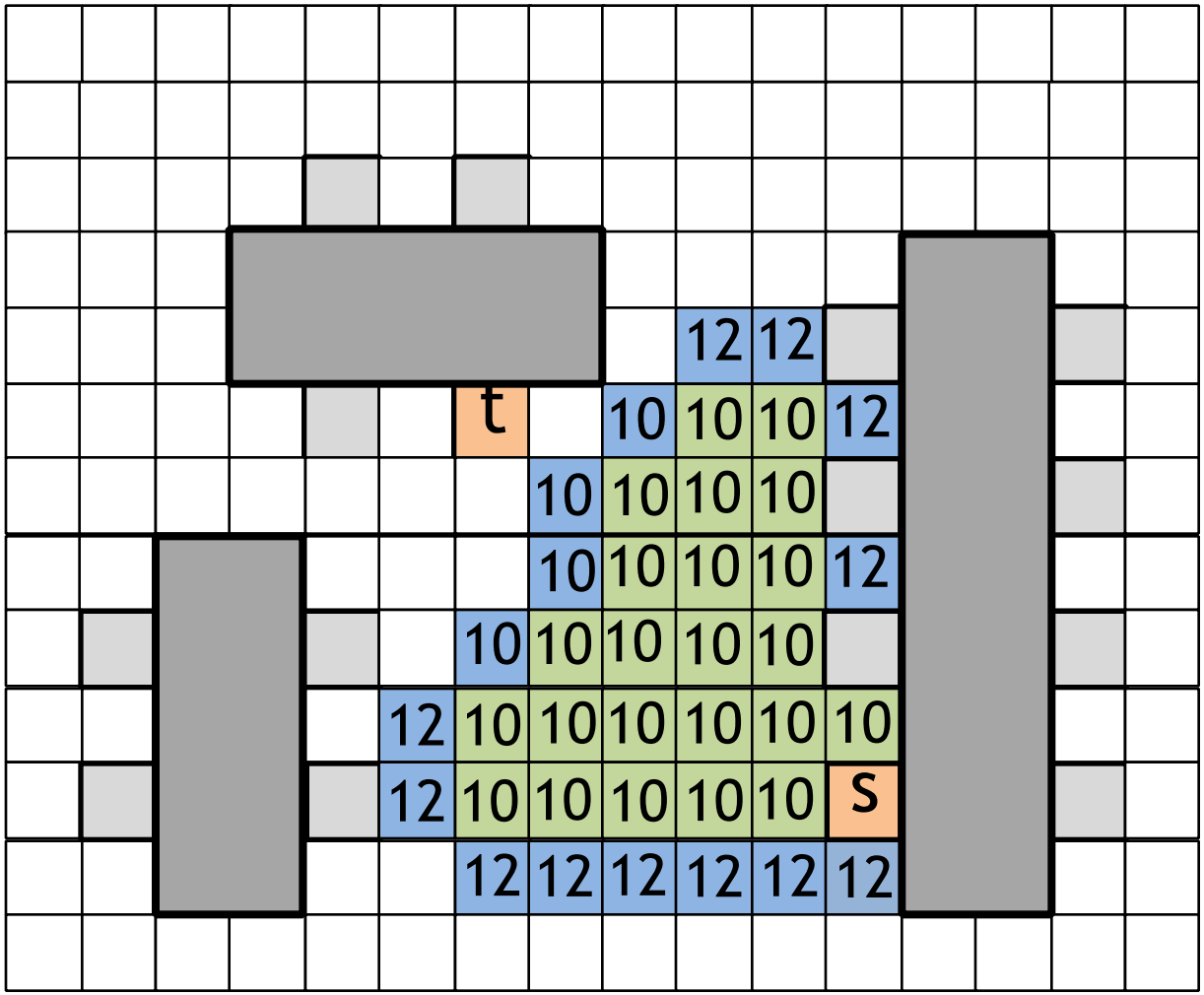
Lee's with Ruben's Costs



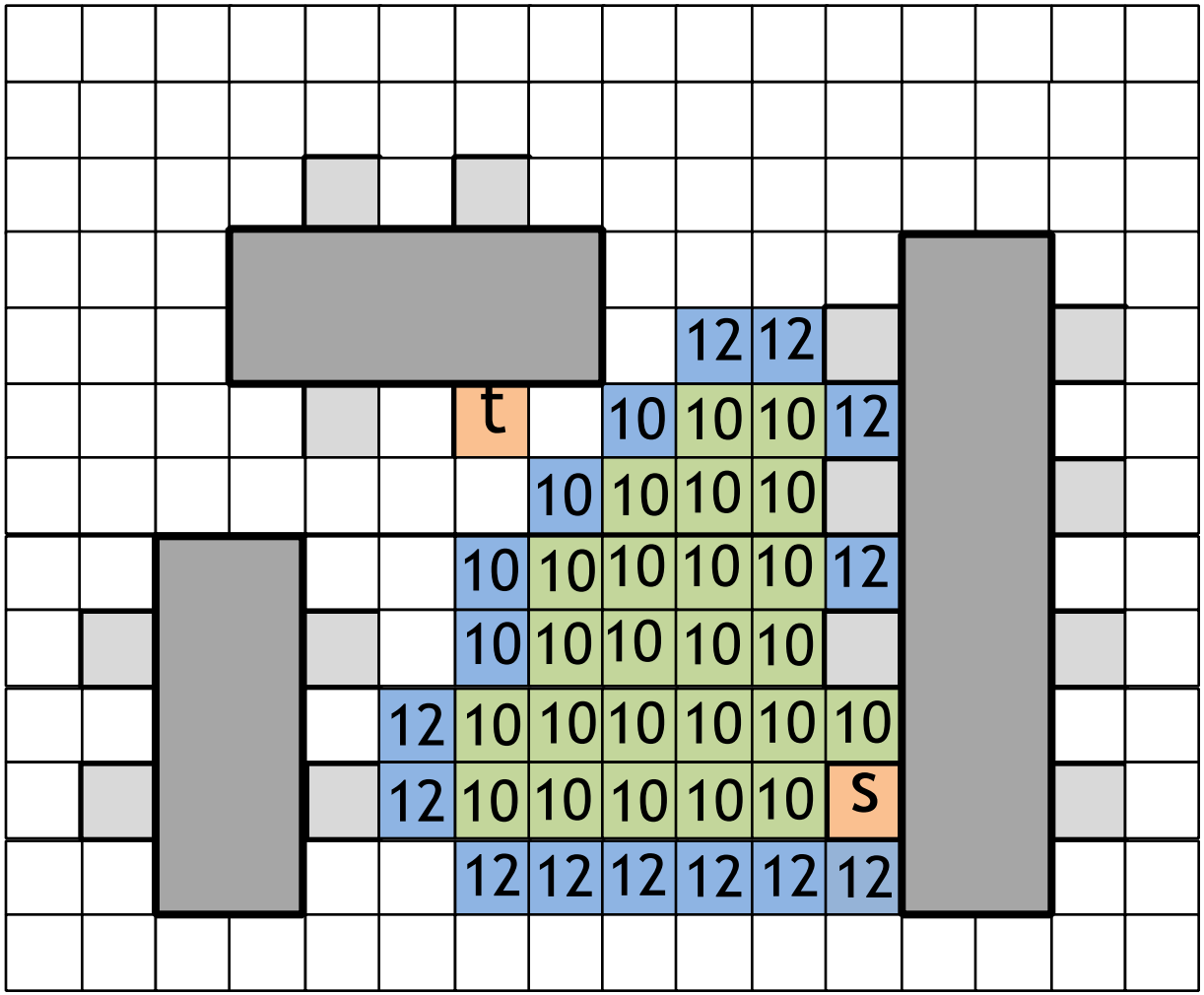
Lee's with Ruben's Costs



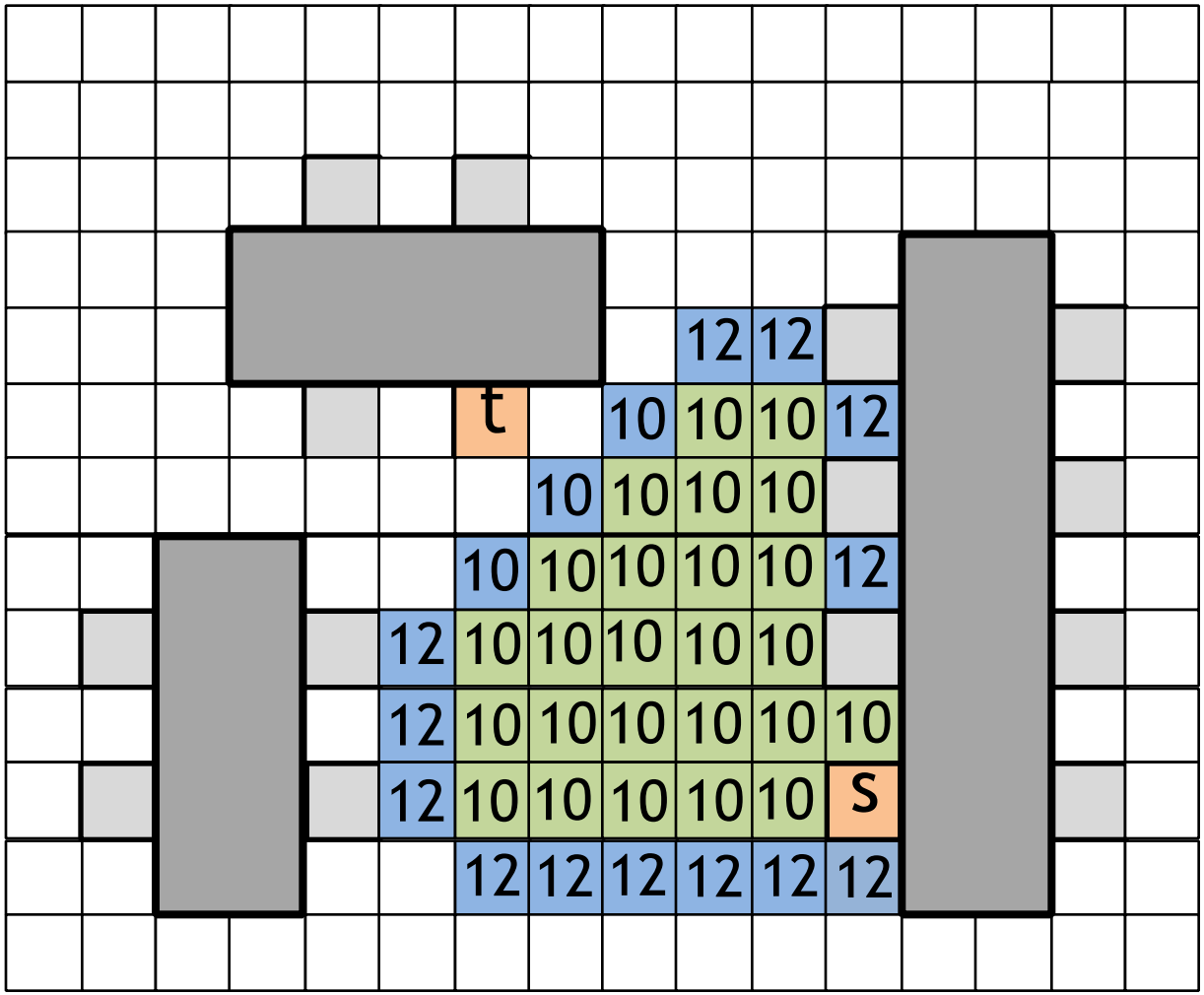
Lee's with Ruben's Costs



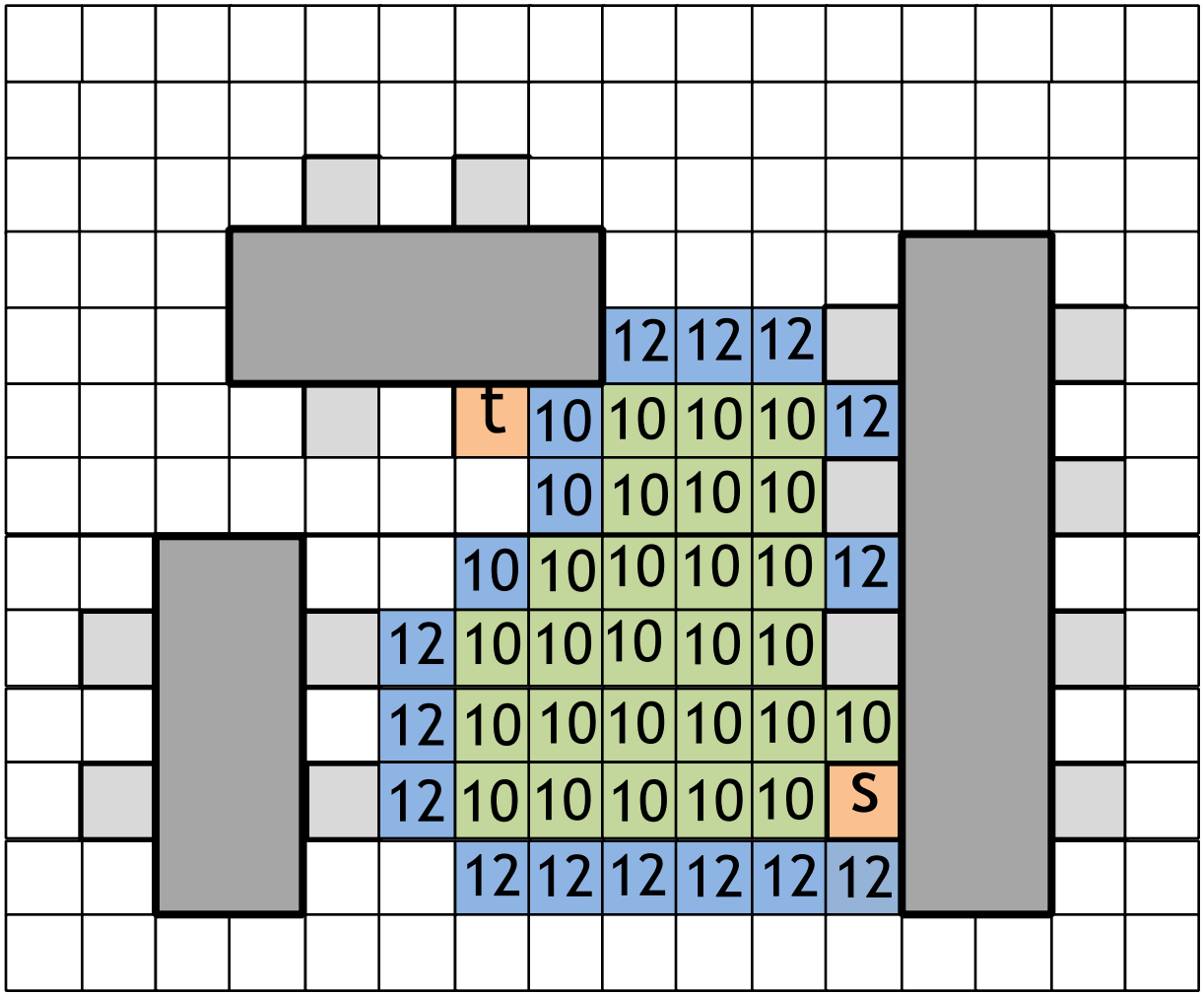
Lee's with Ruben's Costs



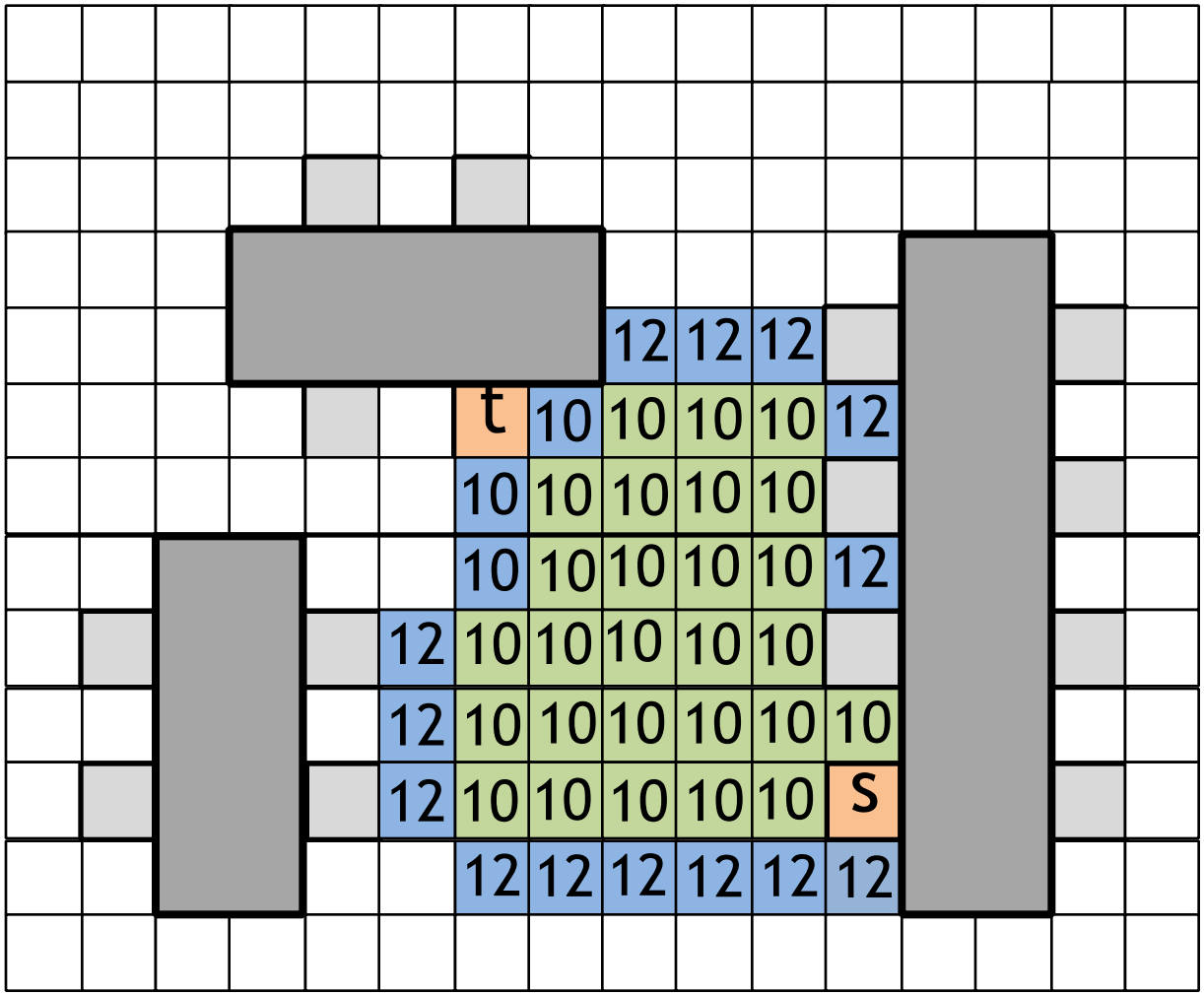
Lee's with Ruben's Costs



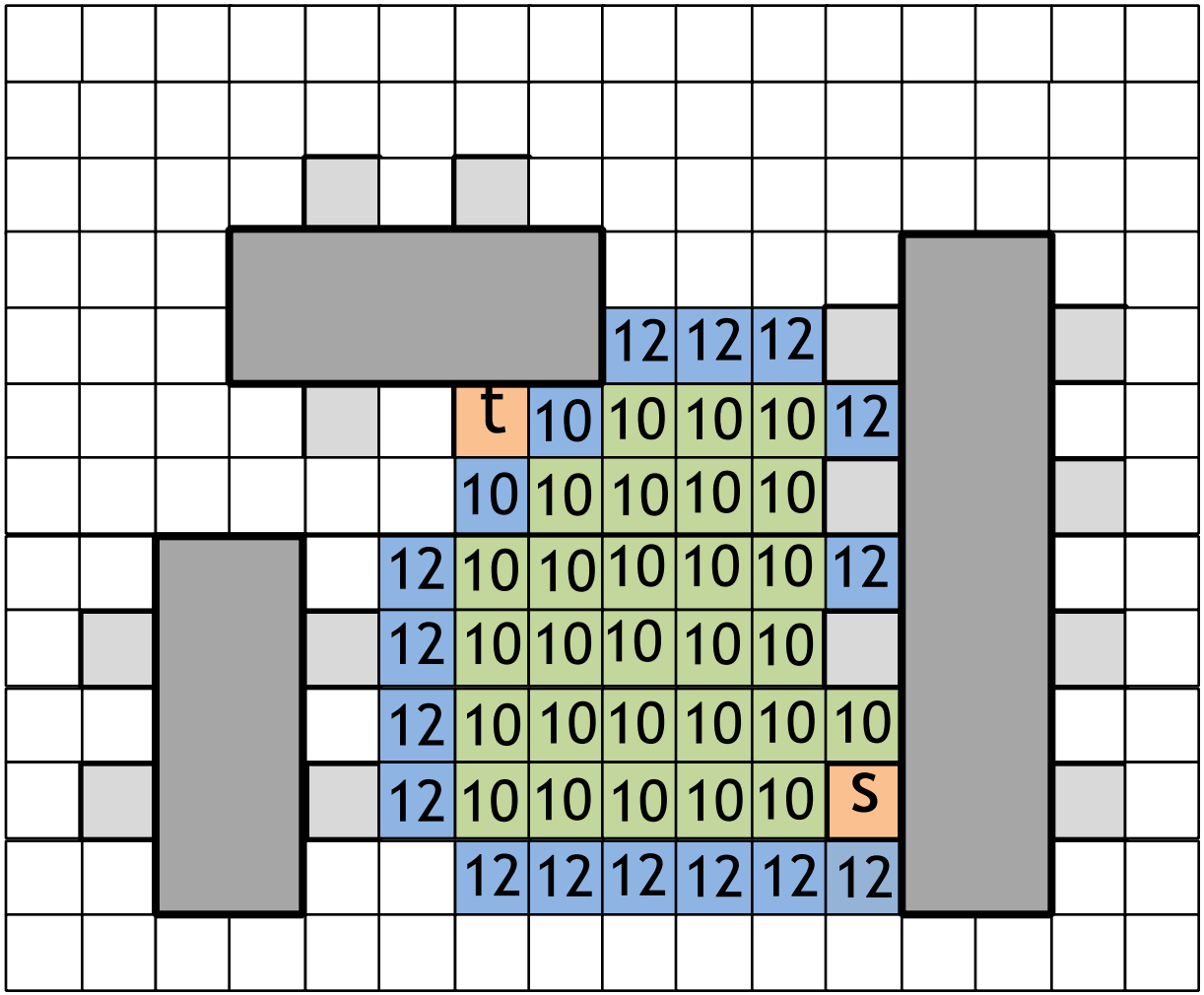
Lee's with Ruben's Costs



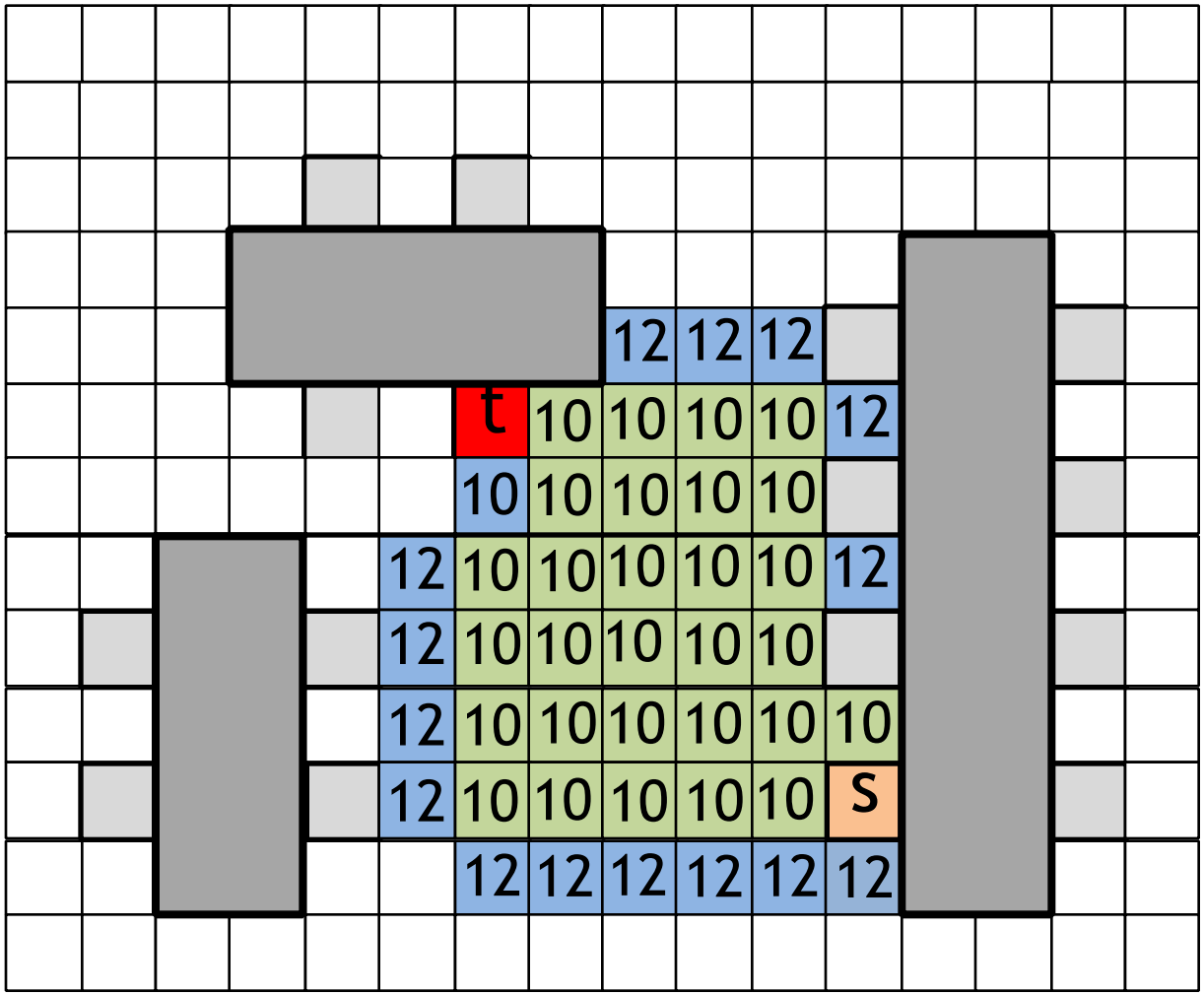
Lee's with Ruben's Costs



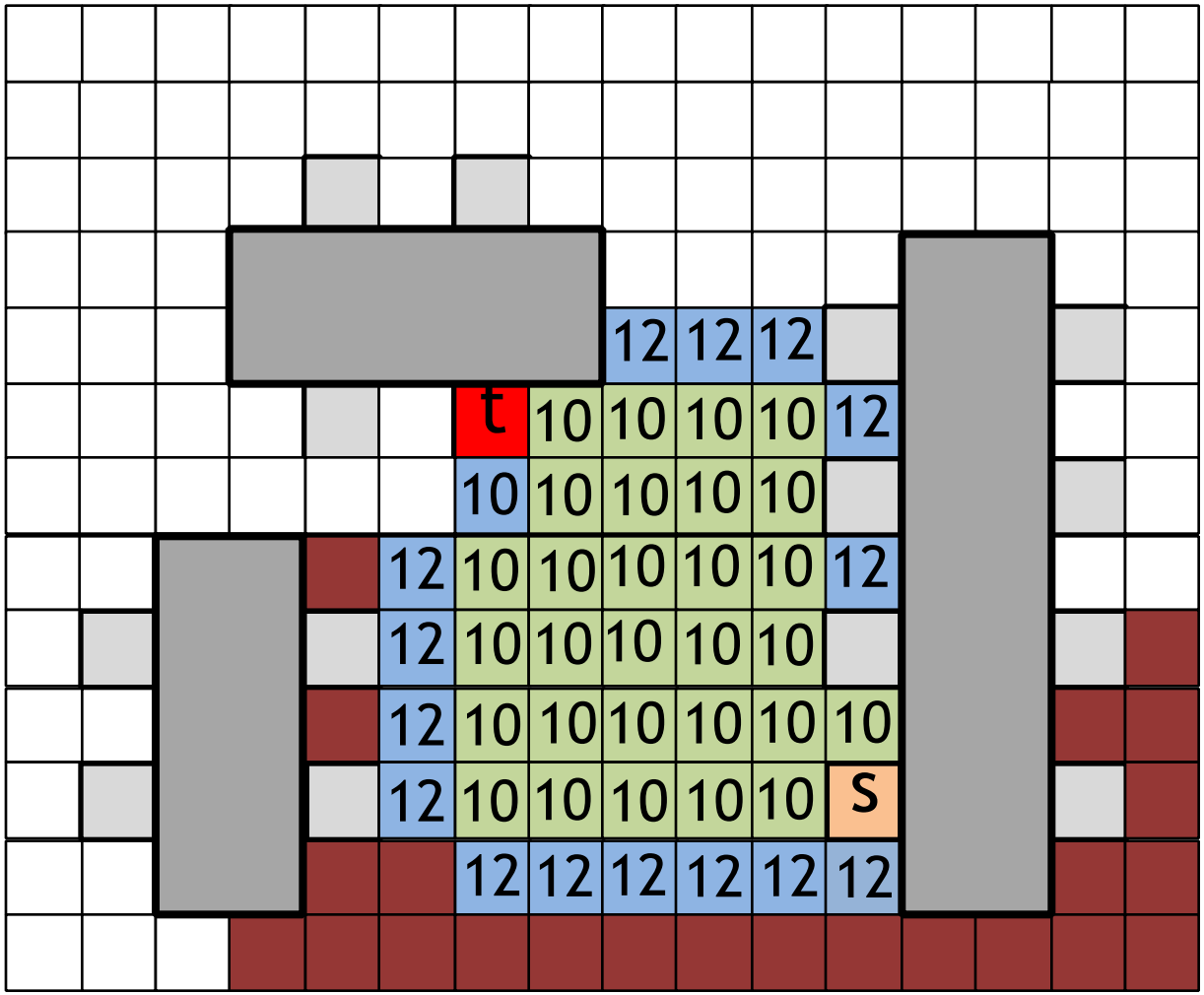
Lee's with Ruben's Costs



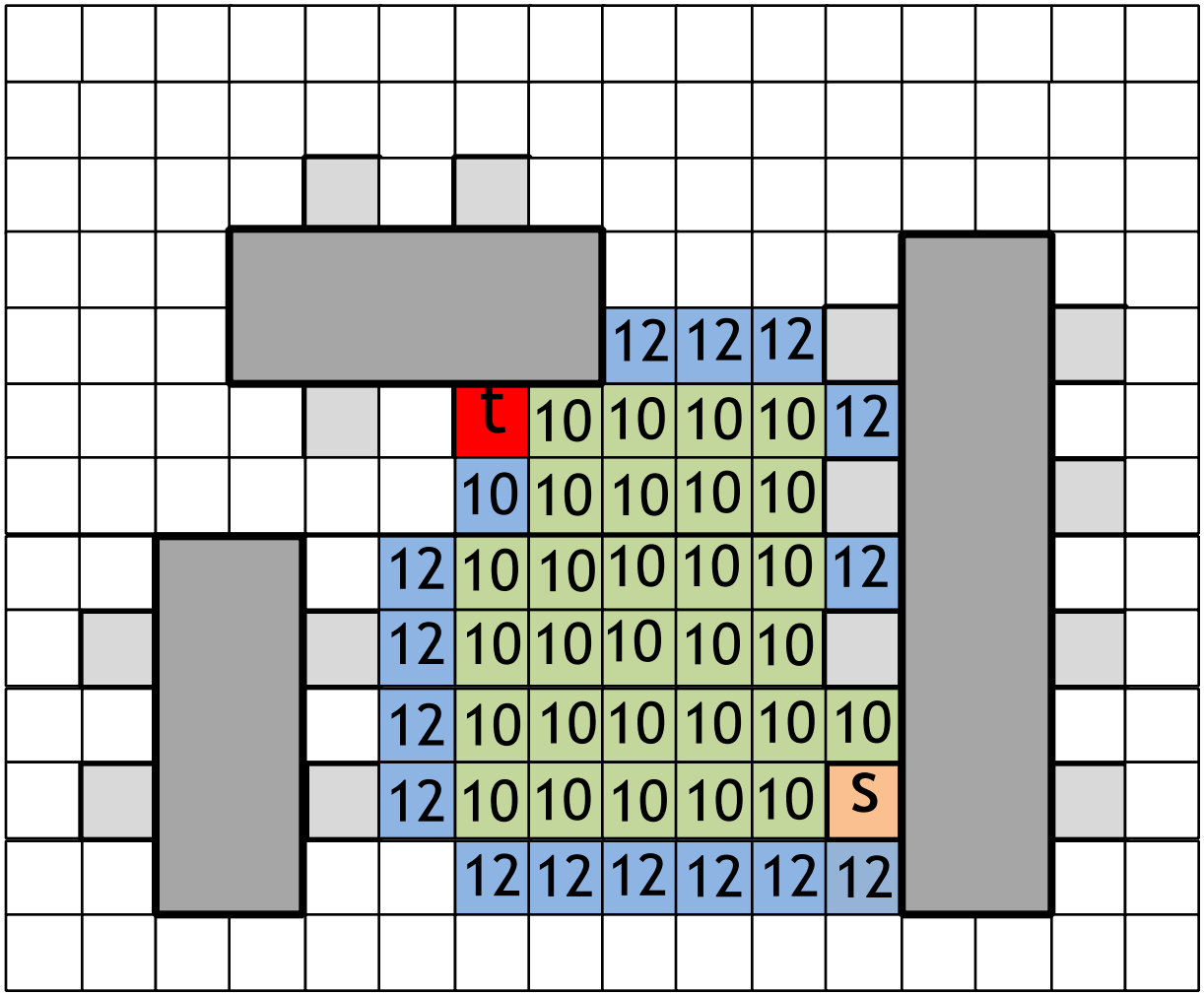
Lee's with Ruben's Costs



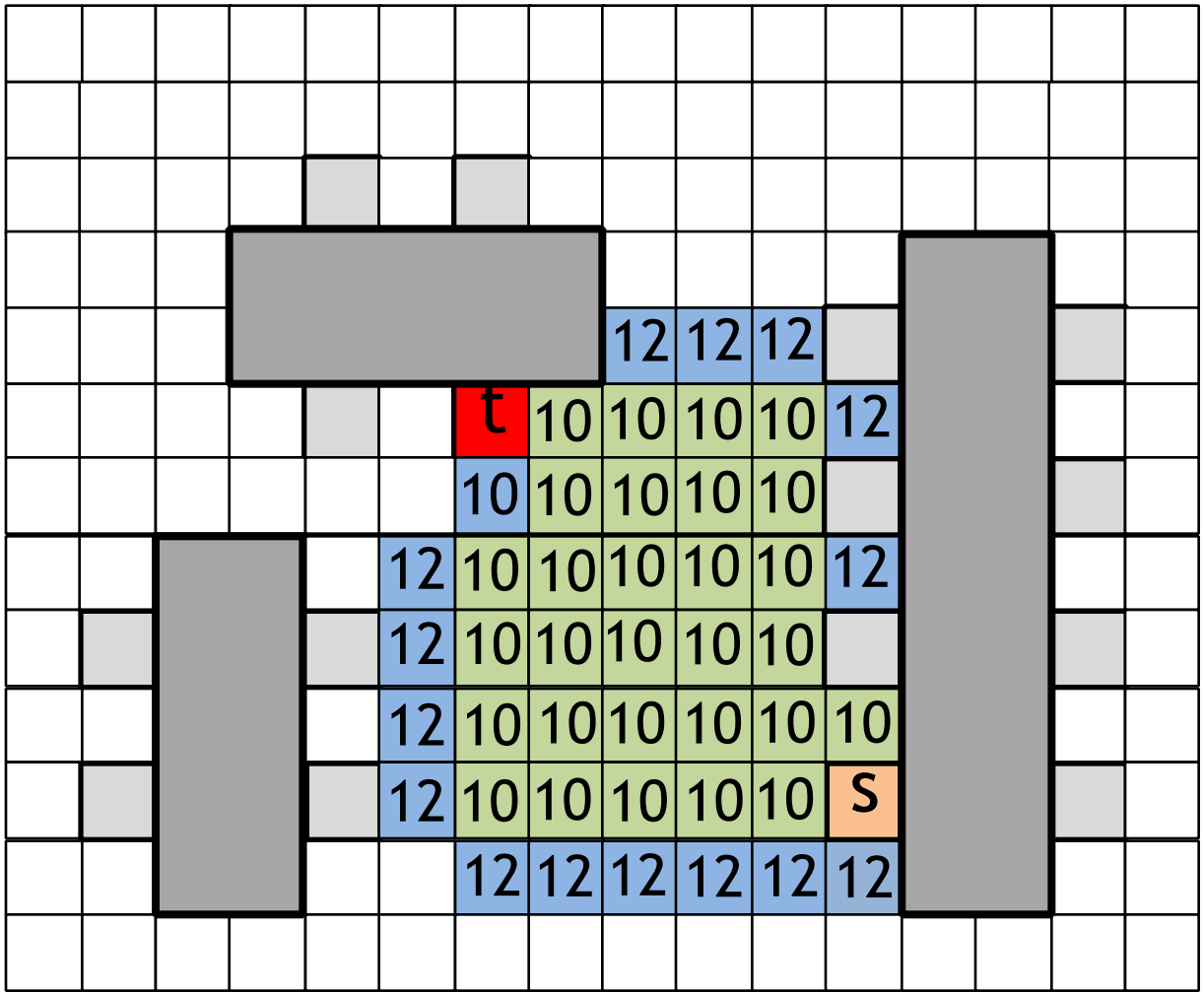
Extra Cells that Lee's Algorithm Expands



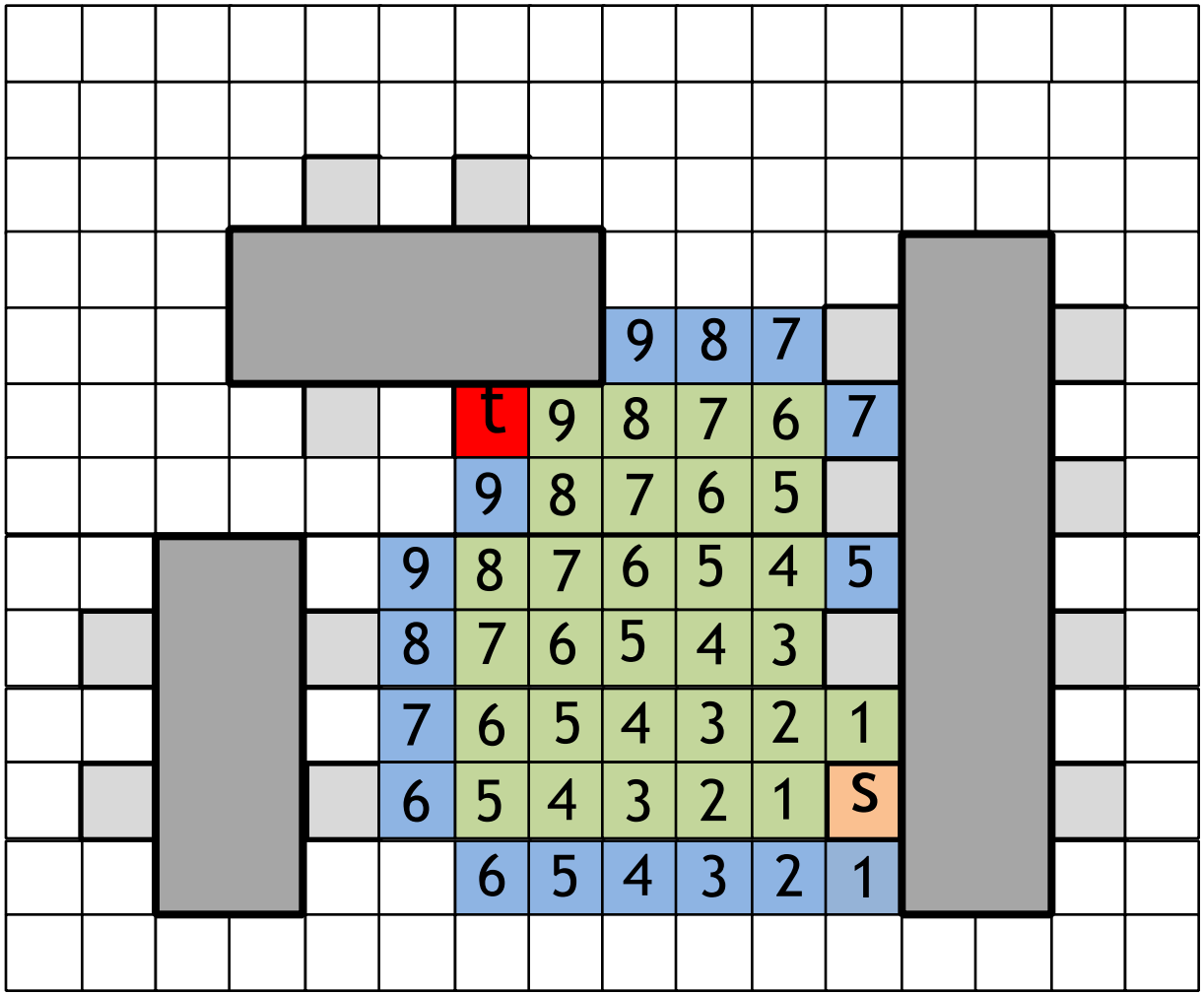
Backtrace?


12 12 12
t 10 10 10 10 12
10 10 10 10 10
12 10 10 10 10 10 12
12 10 10 10 10 10
12 10 10 10 10 10 10
12 10 10 10 10 10 s
12 12 12 12 12 12

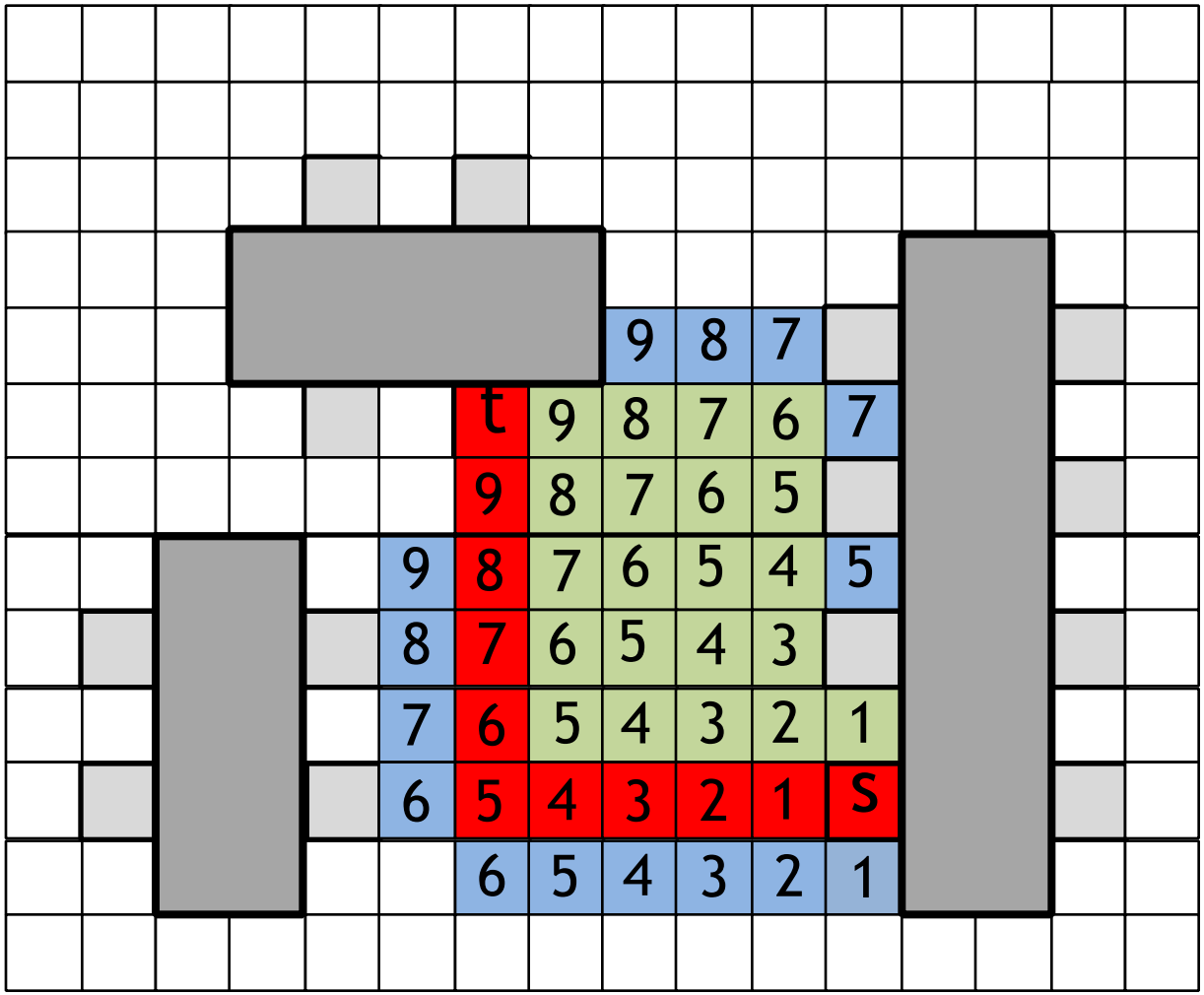
C(i) = D(s, i) + MD(i, t)


12 12 12
t 10 10 10 10 12
10 10 10 10 10
12 10 10 10 10 10 12
12 10 10 10 10 10
12 10 10 10 10 10 10
12 10 10 10 10 10 s
12 12 12 12 12 12

Save the D(s, i) values!

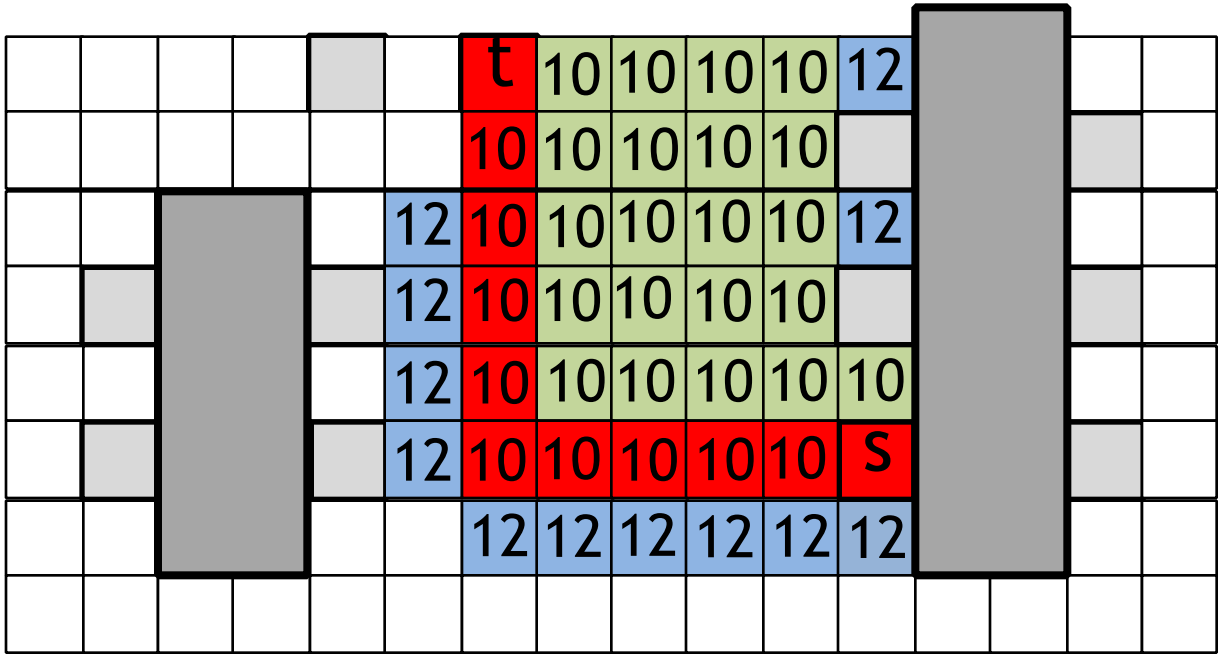


Backtrace is the Same

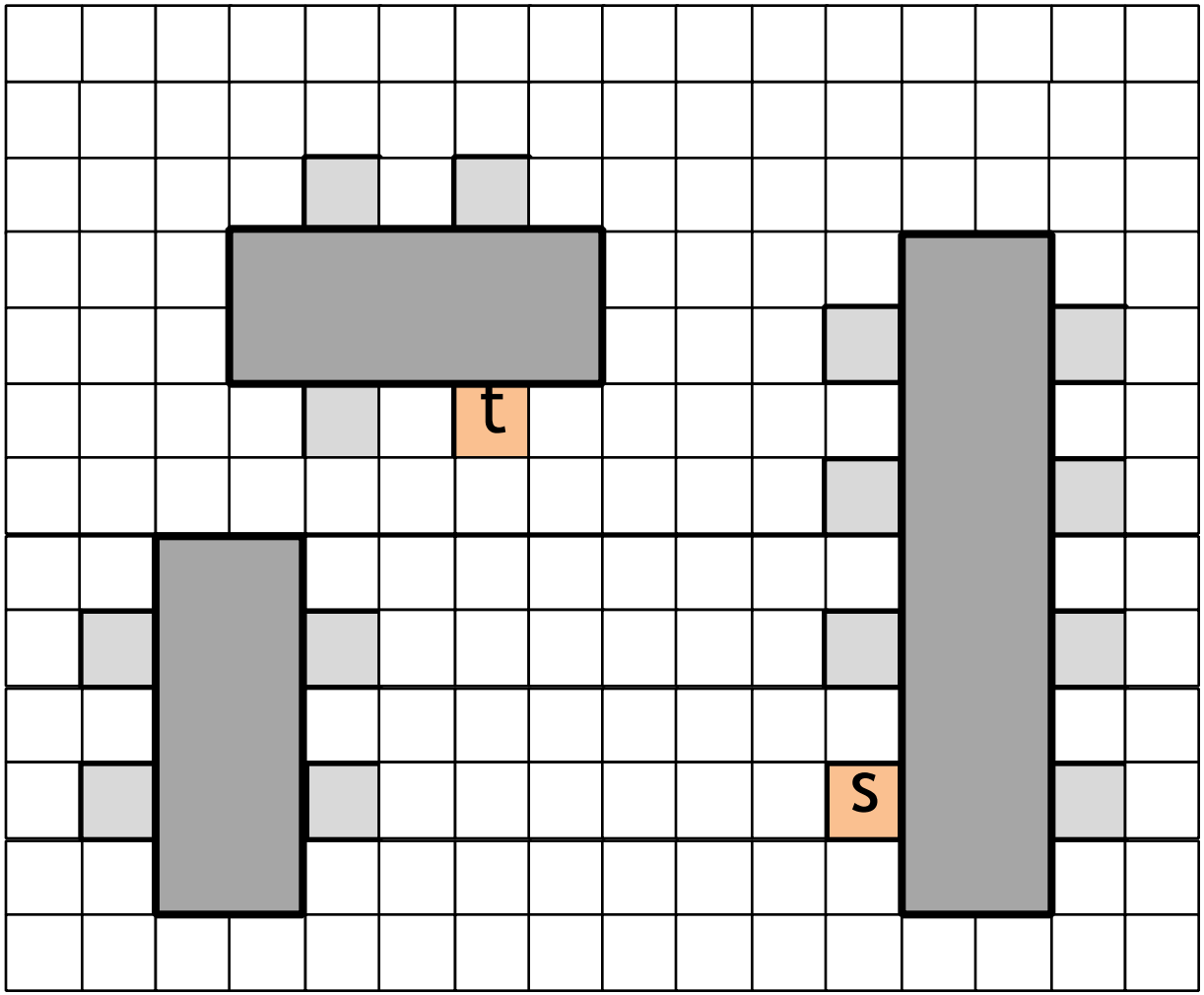


An Improvement

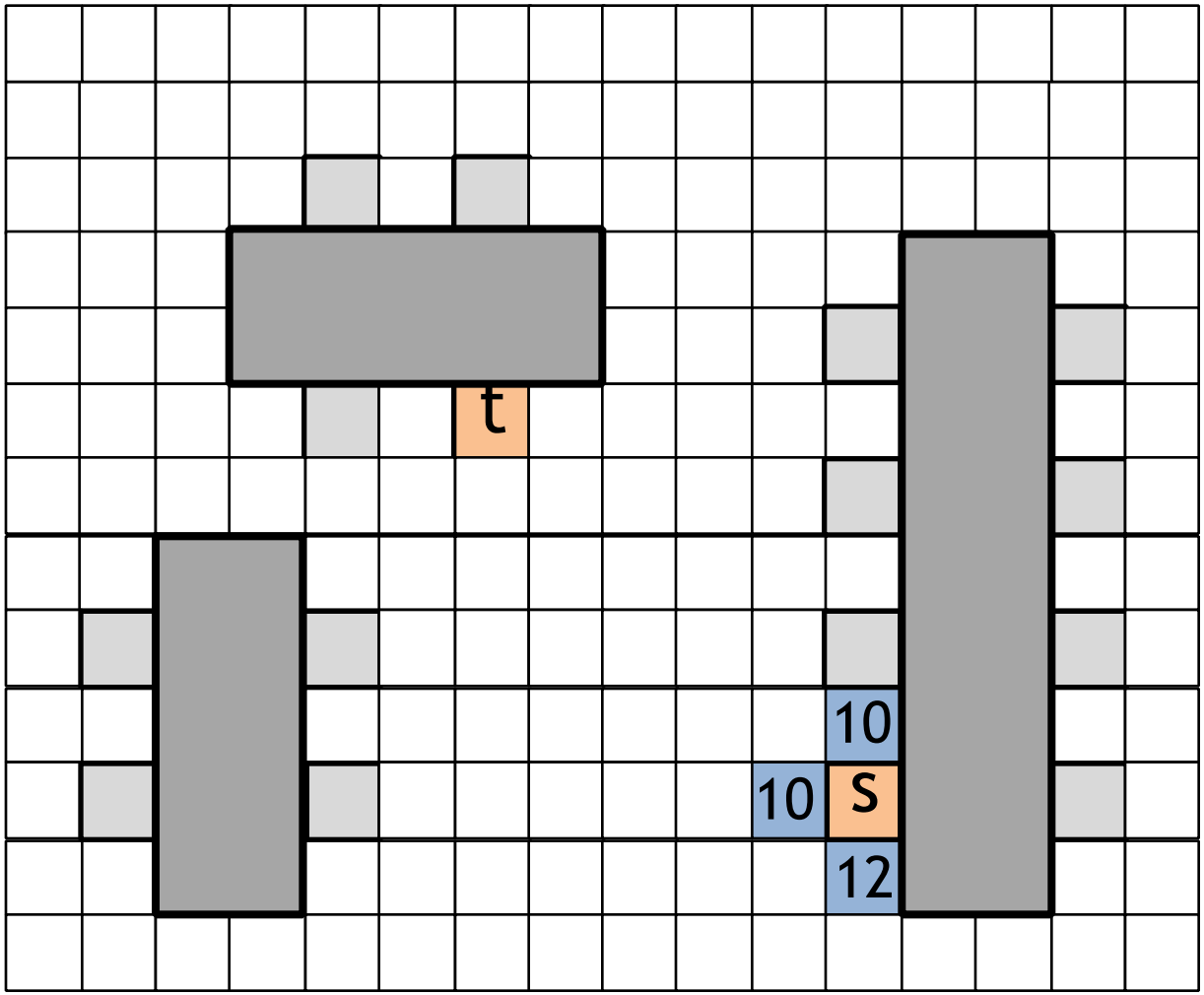
- In a region with no blockages, among cells with equal costs always expand the cell most recently added to the wavefront



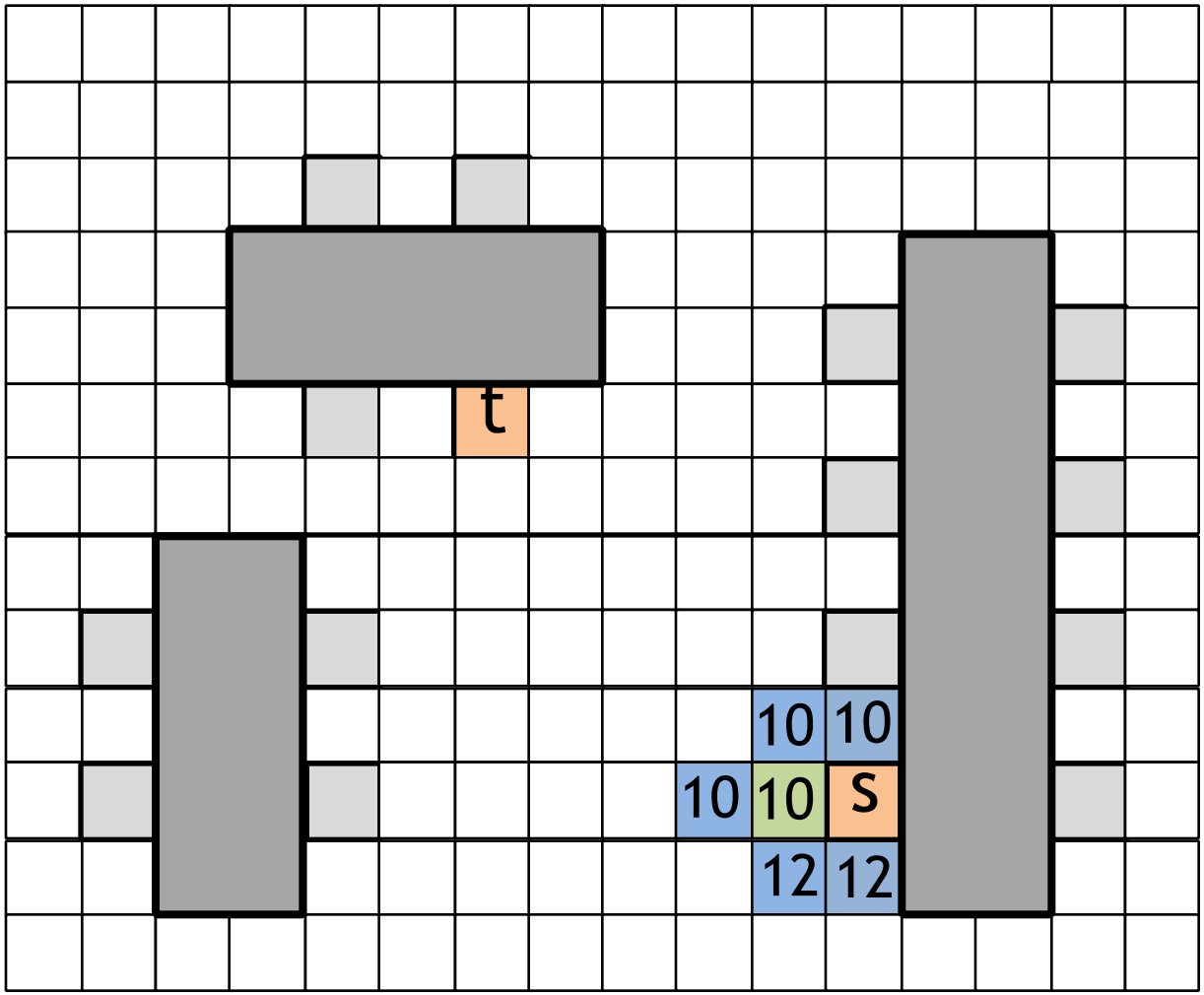
Lee's with Ruben's Costs



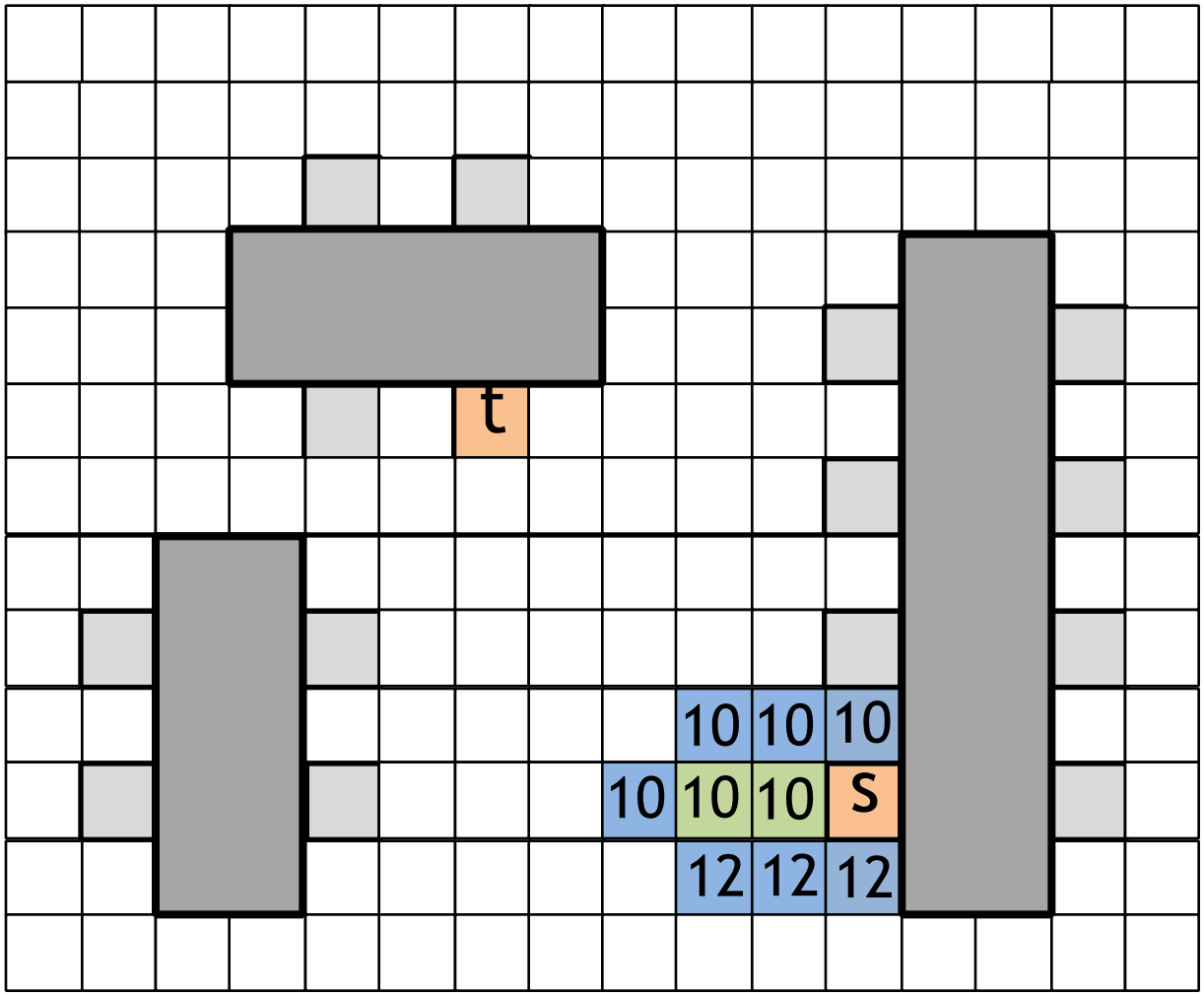
Lee's with Ruben's Costs



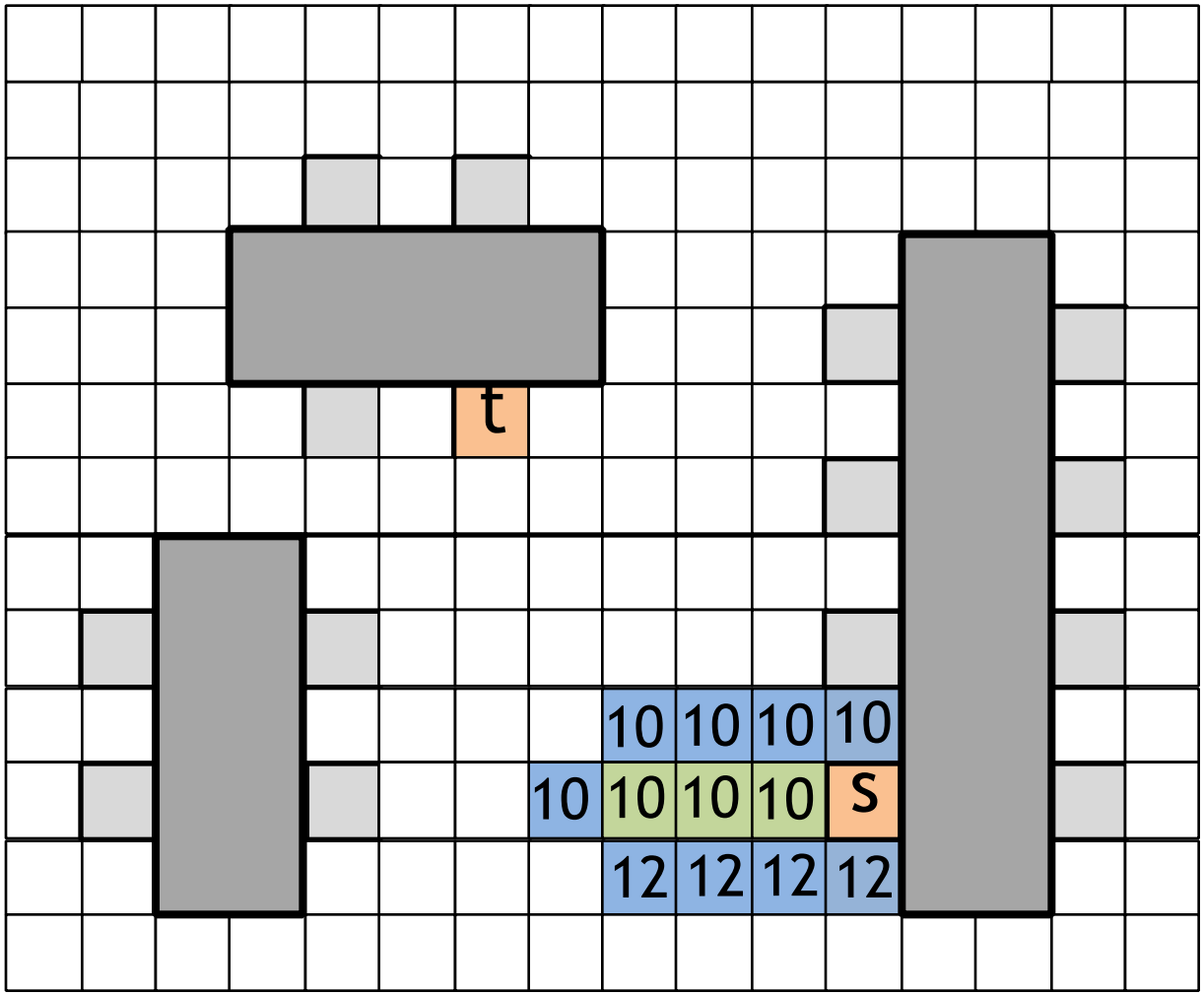
Lee's with Ruben's Costs



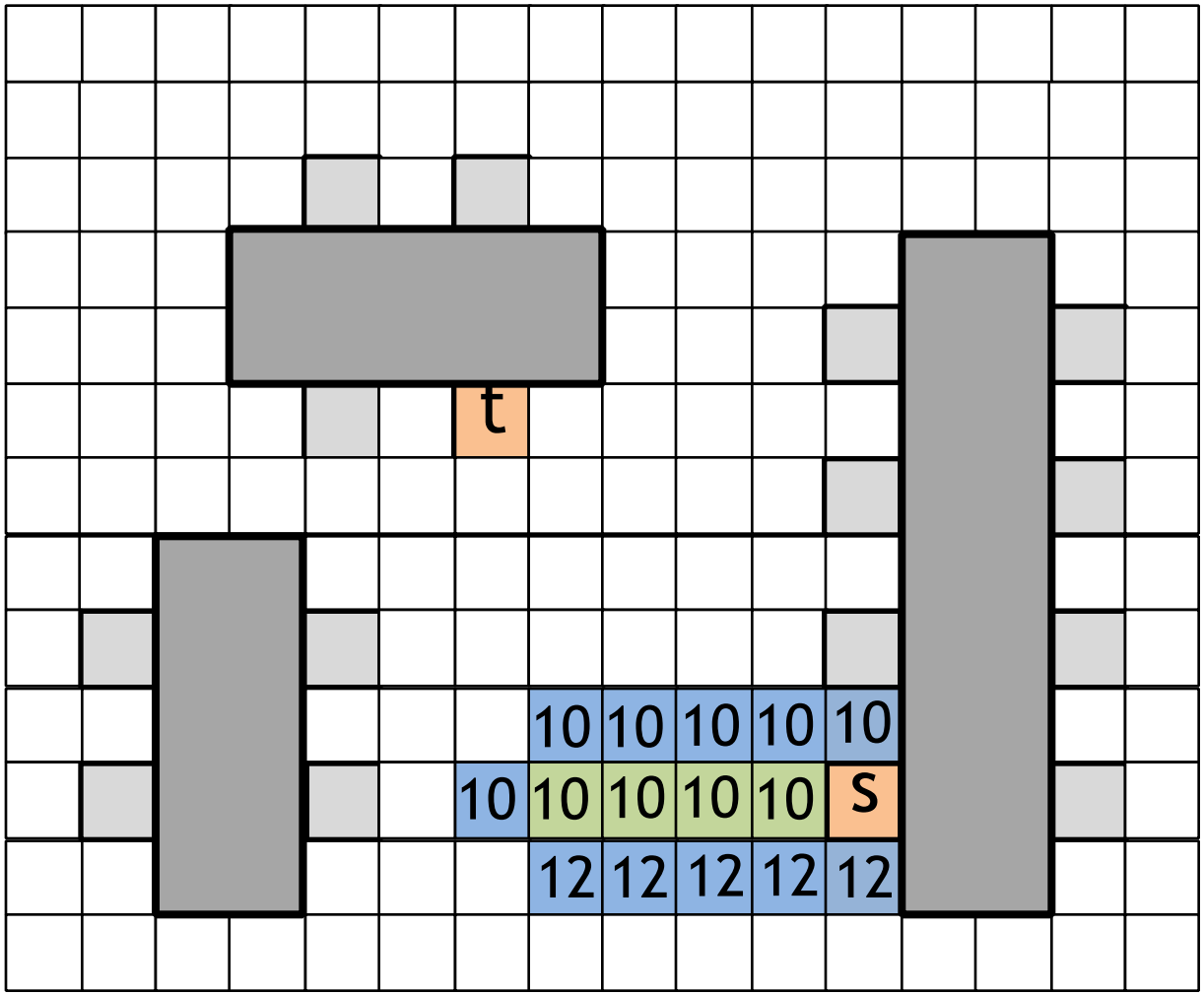
Lee's with Ruben's Costs



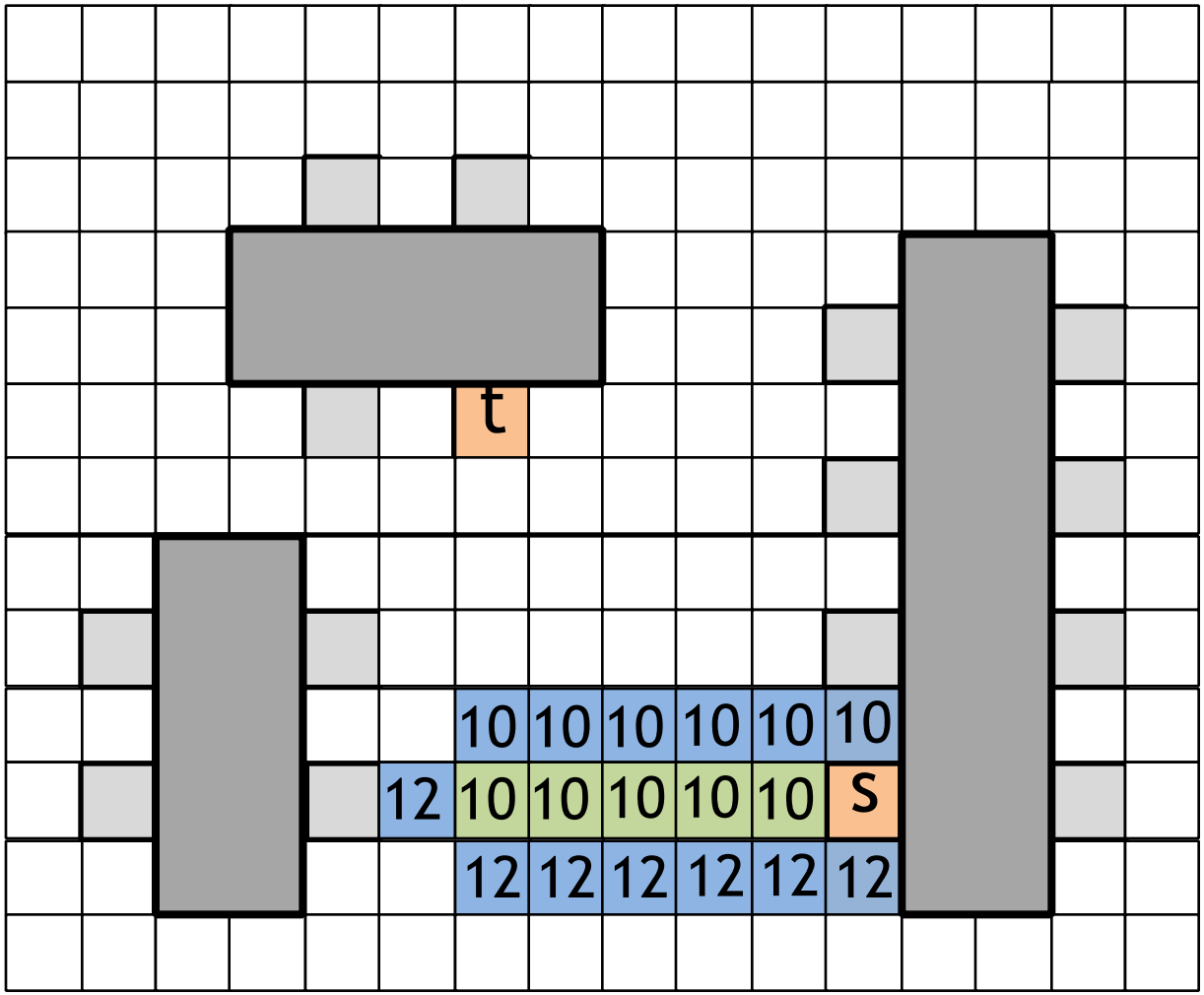
Lee's with Ruben's Costs



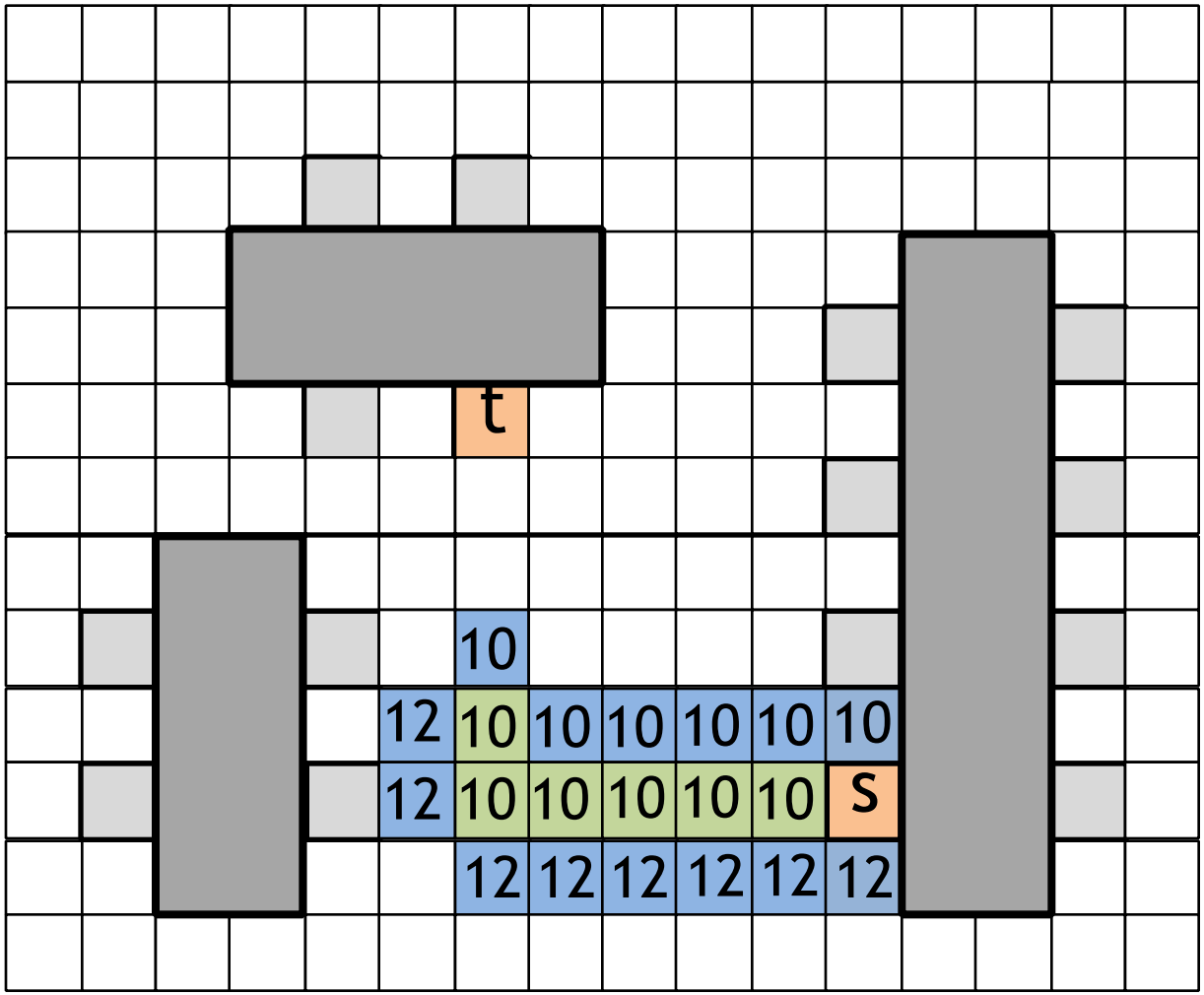
Lee's with Ruben's Costs



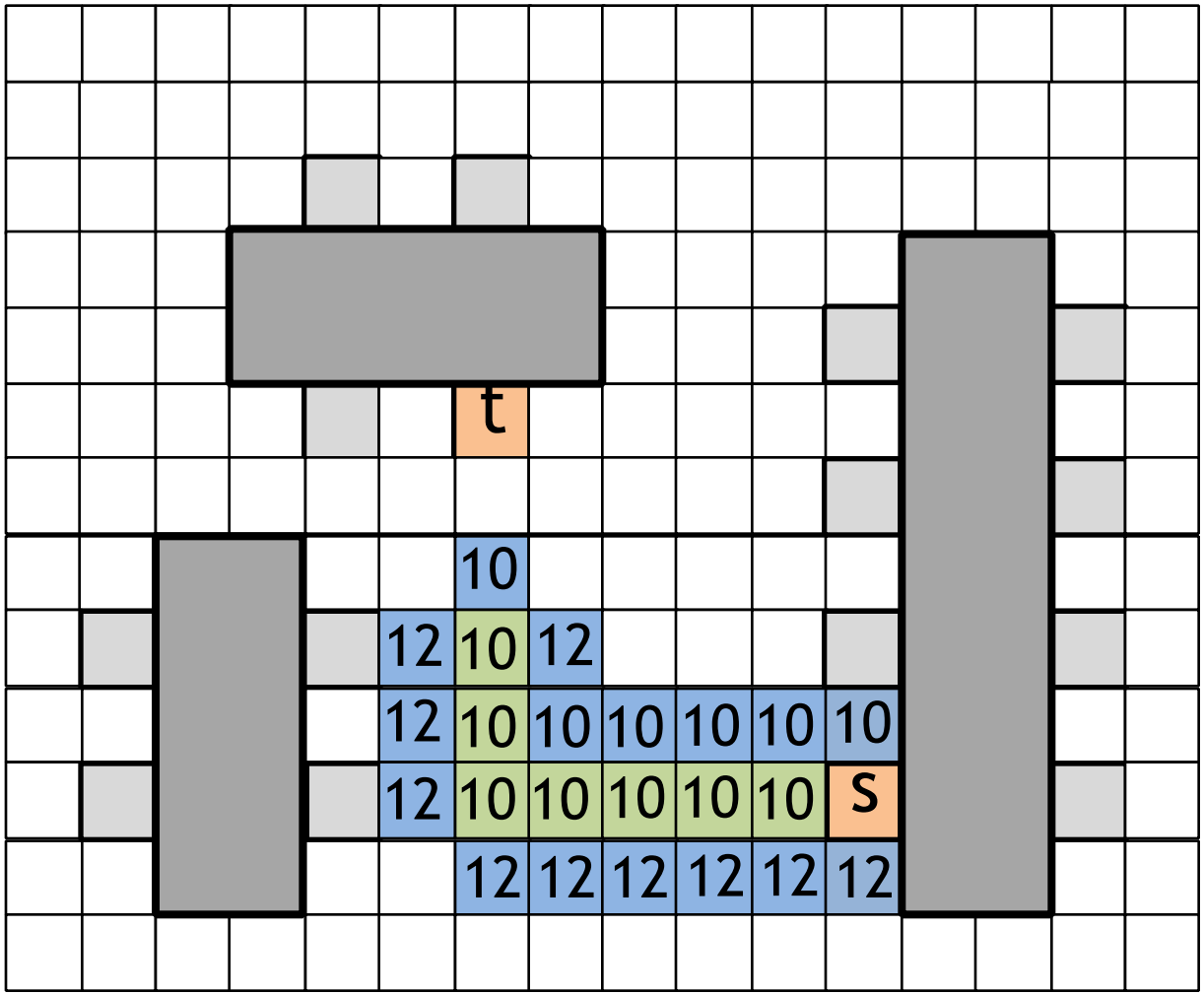
Lee's with Ruben's Costs



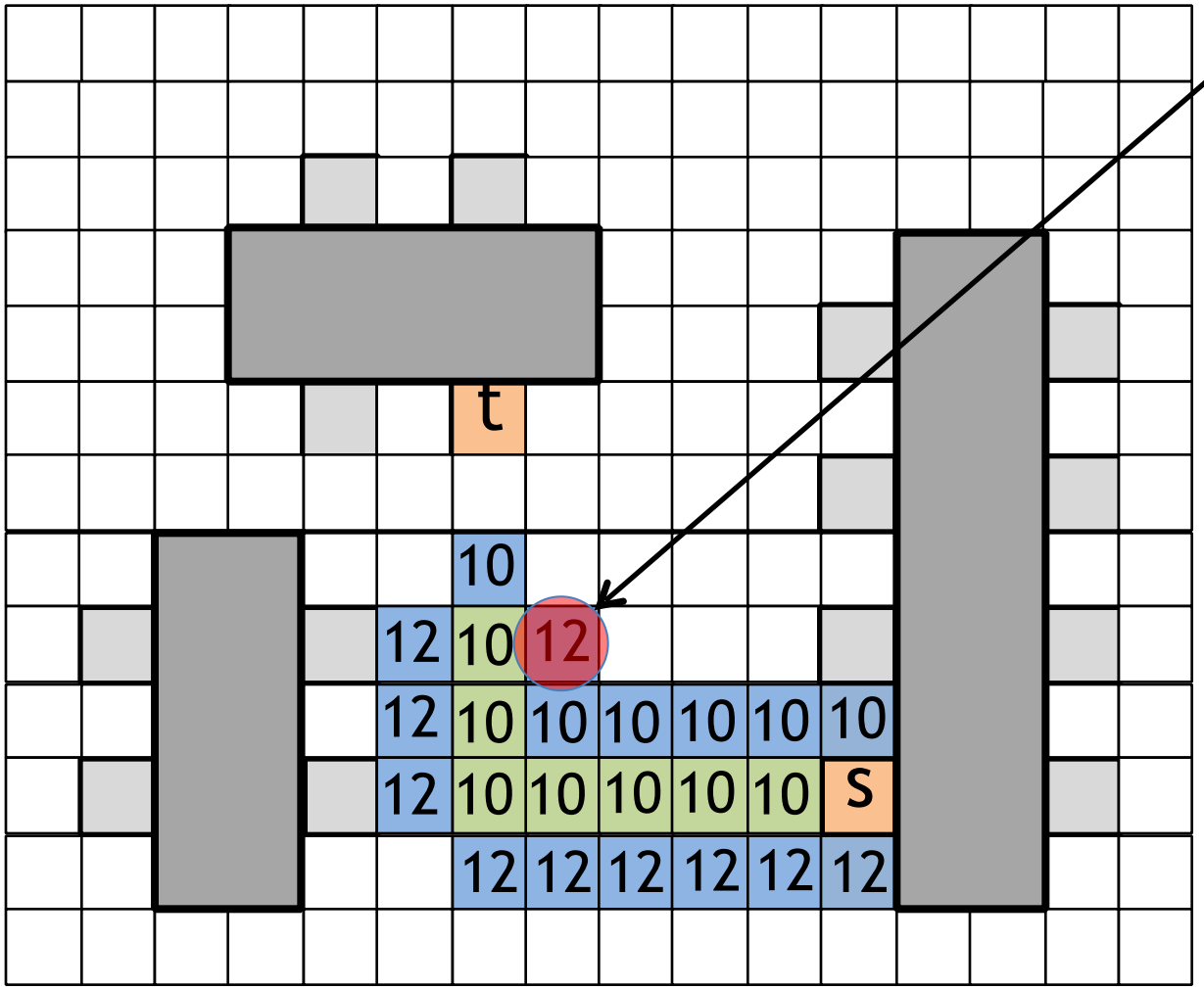
Lee's with Ruben's Costs



Lee's with Ruben's Costs



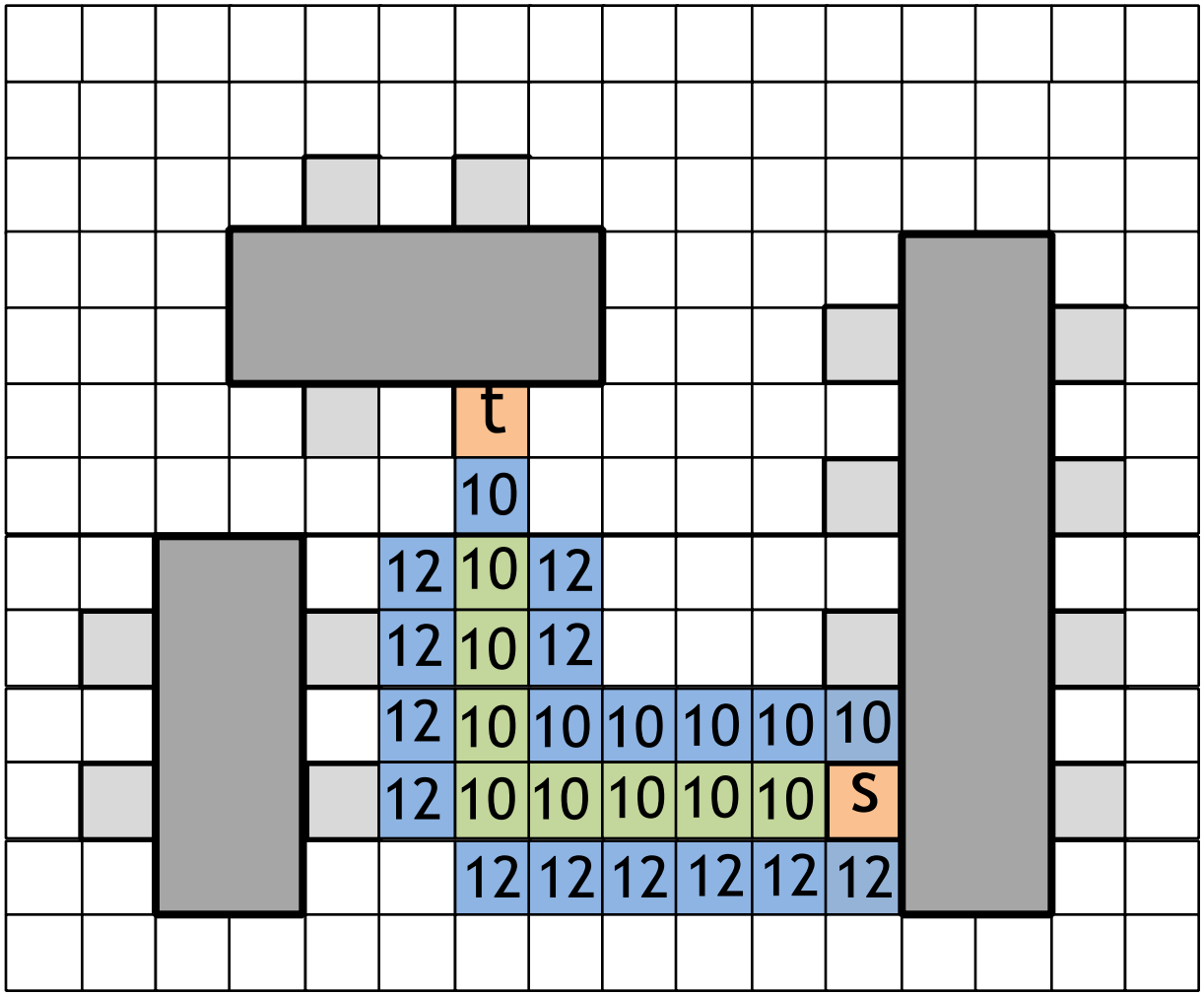
Lee's with Ruben's Costs



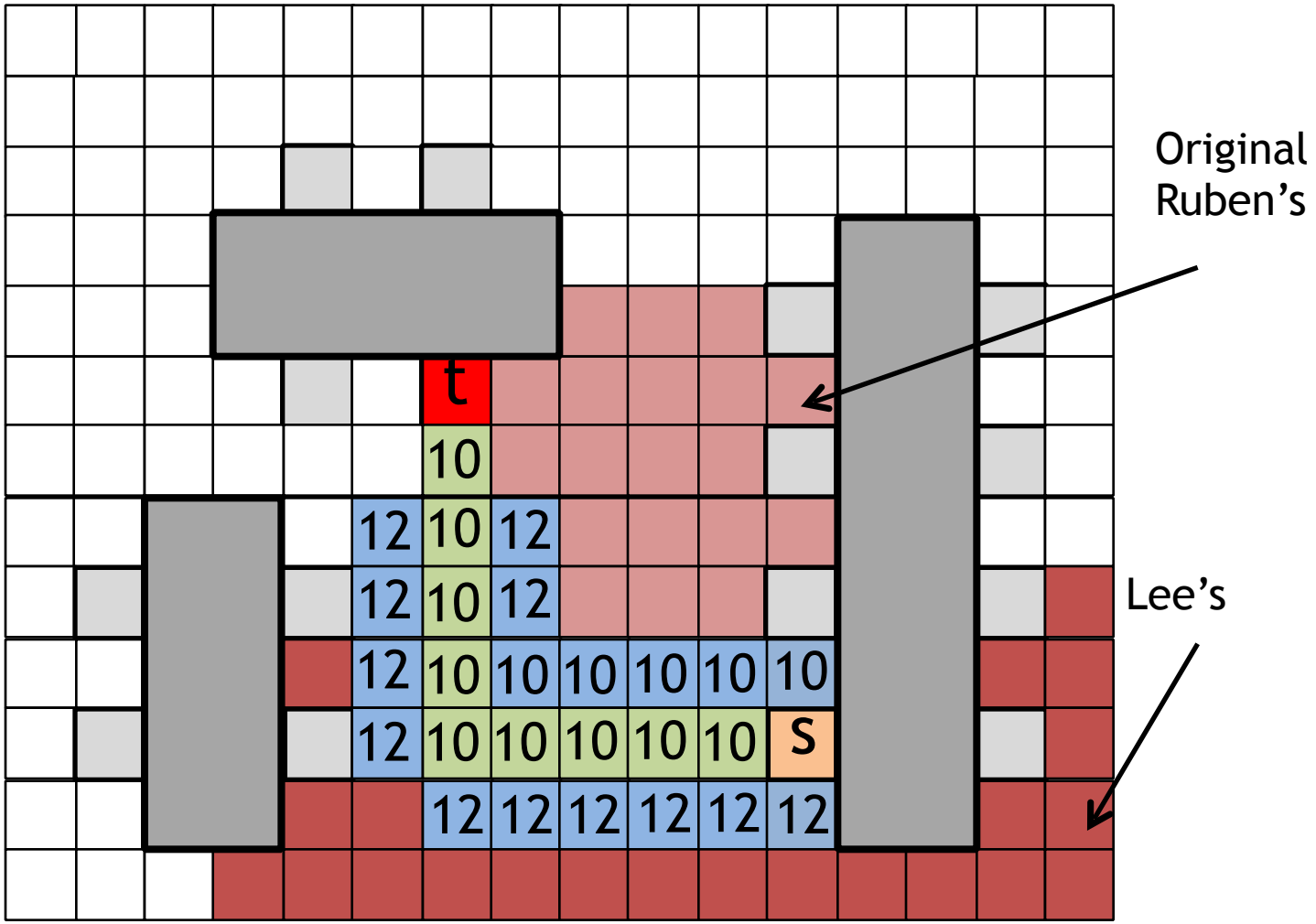
Notice the path to discovery of this cell

Can reduce the cost if it is "rediscovered" by wavefront expansion from another cell

Lee's with Ruben's Costs



Cells Avoided

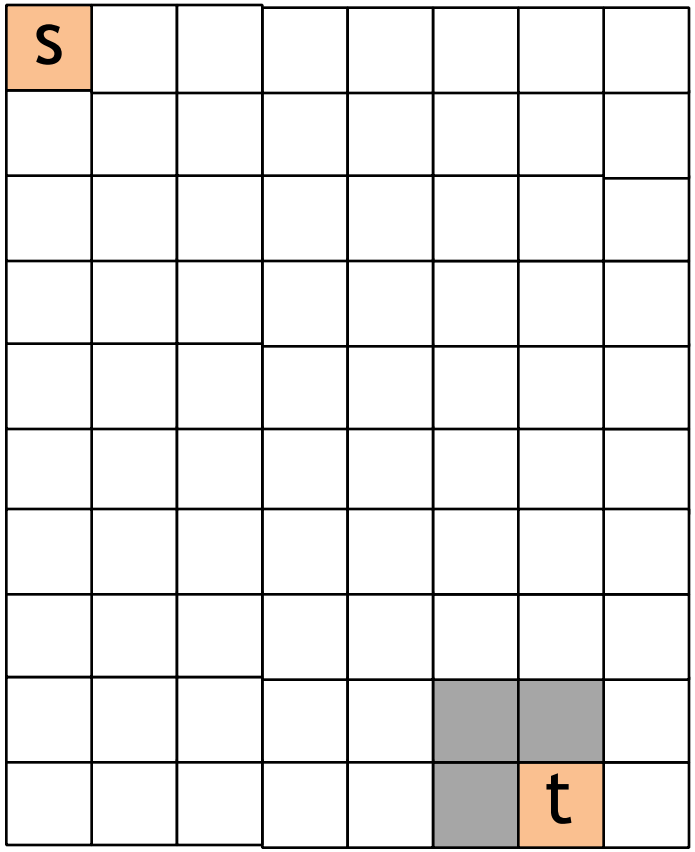


A Few Notes

- Ruben's Cost Function is not compatible with Akers' 2 and 3-bit encodings

- Ruben's cost function directs the breadth first search from s toward t
  - If there are no blockages between s and t
  - It doesn't do so well if there are blockages

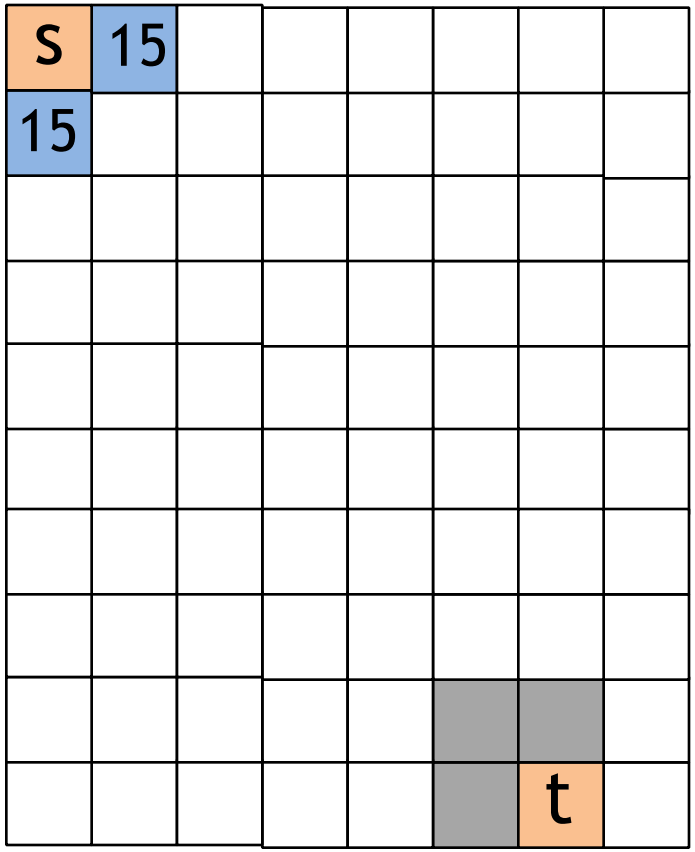
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

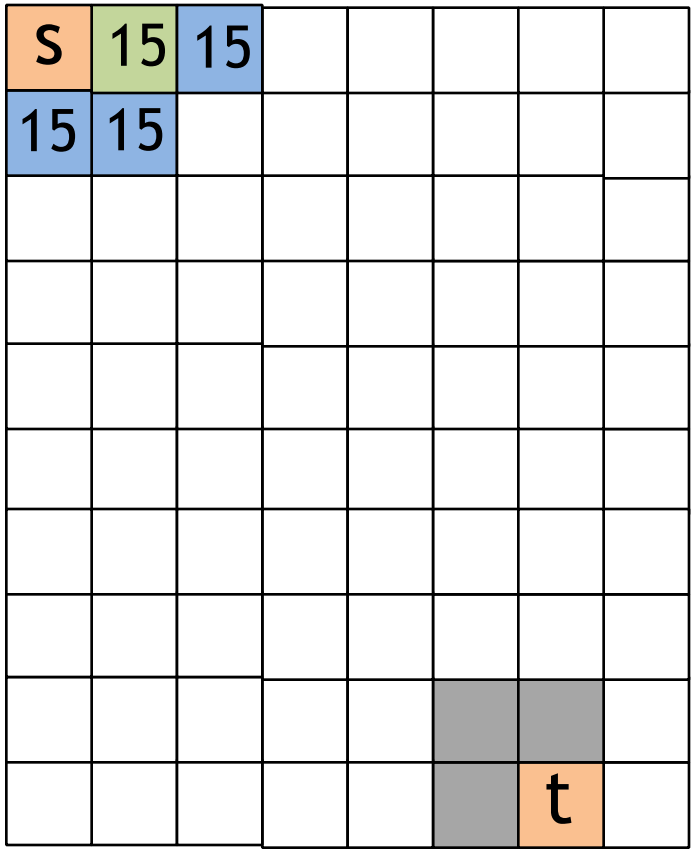
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

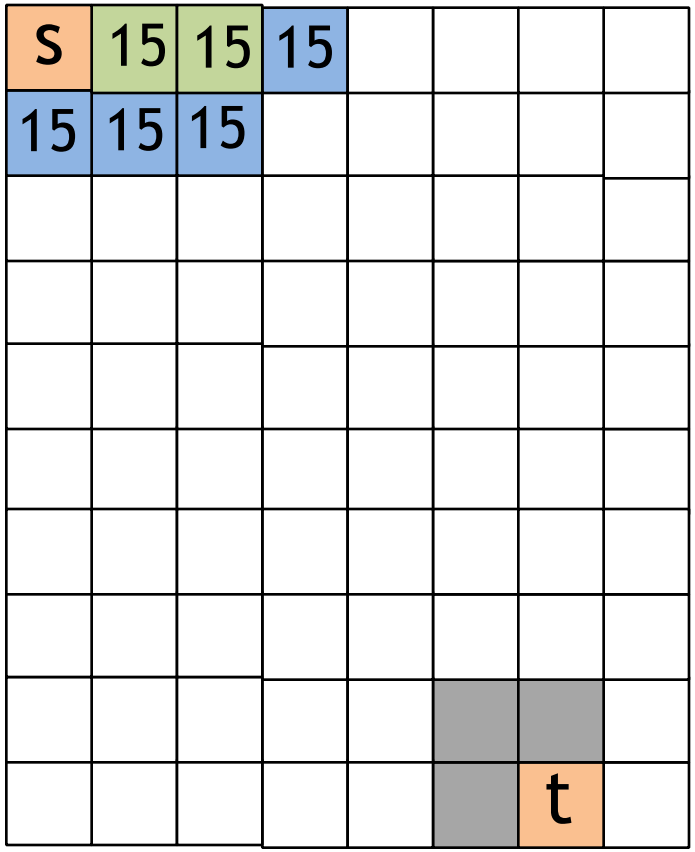
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

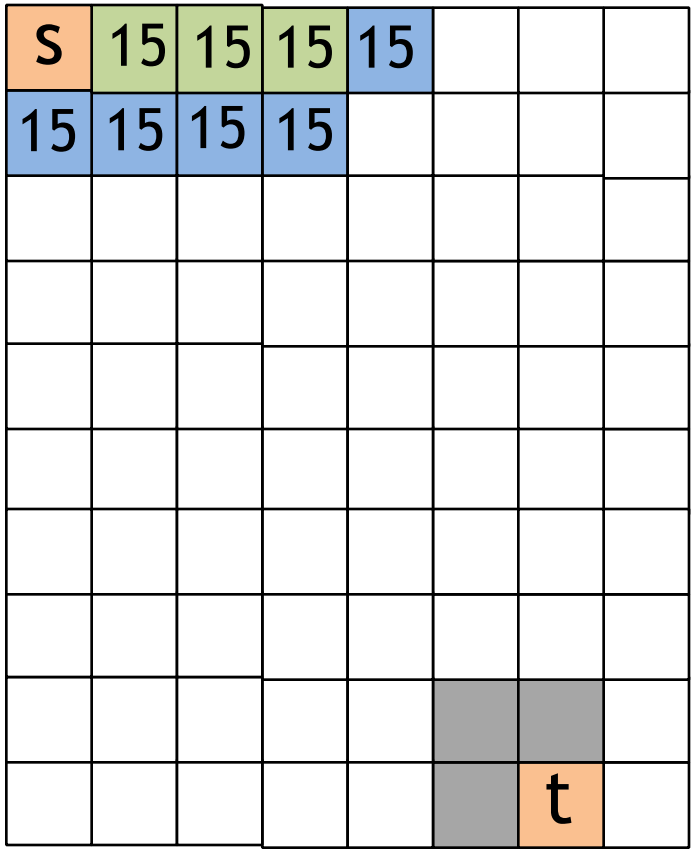
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

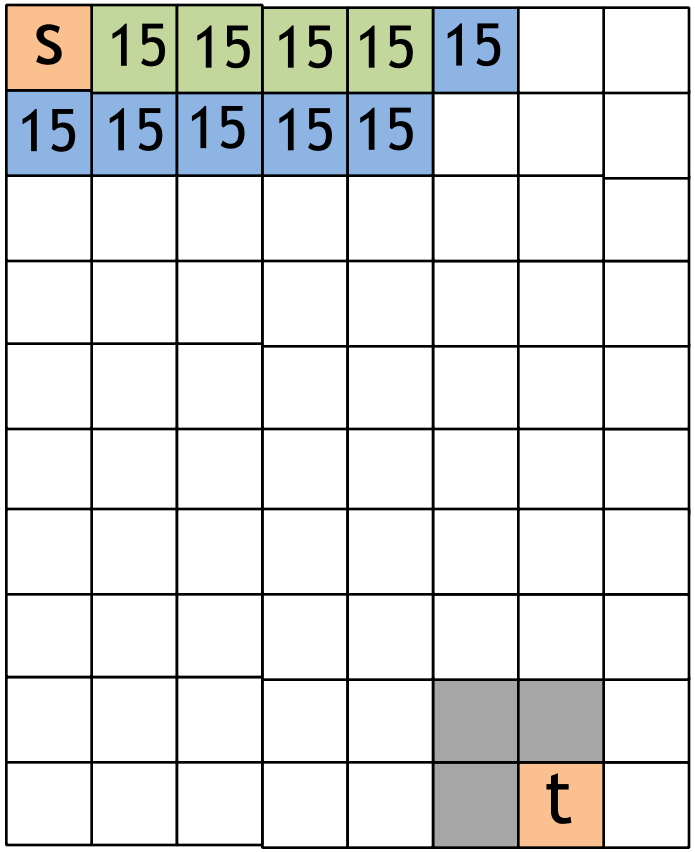
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

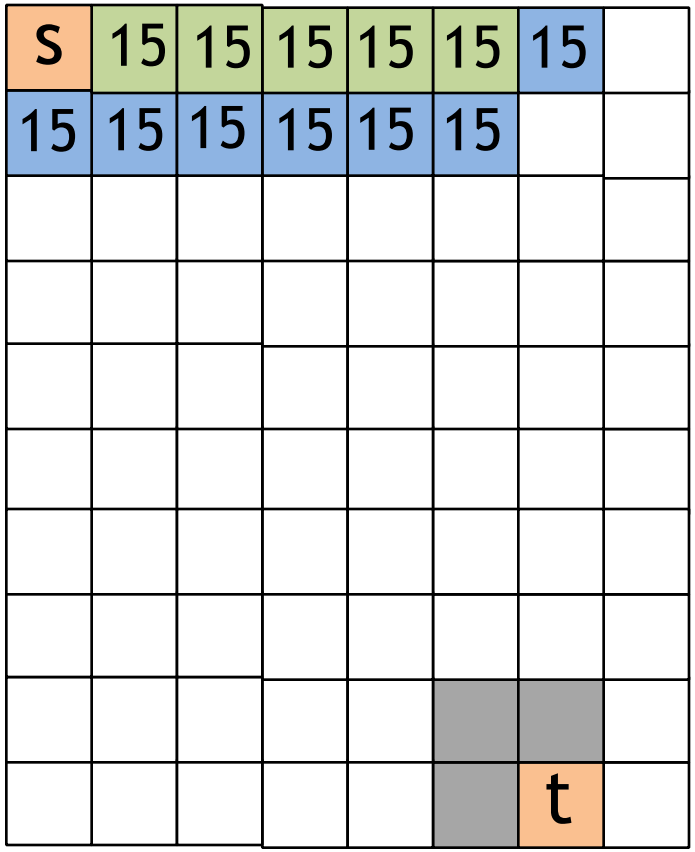
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

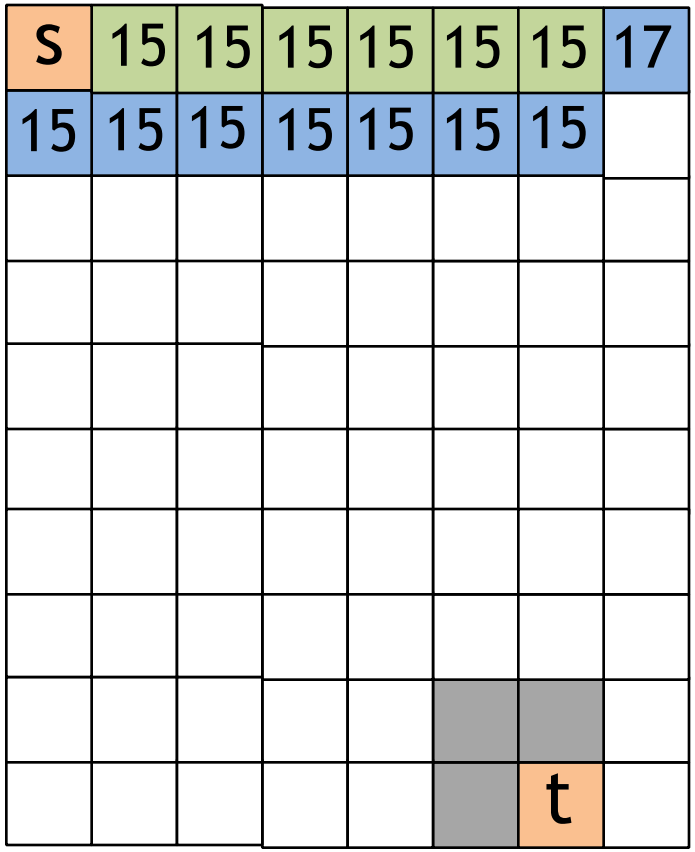
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

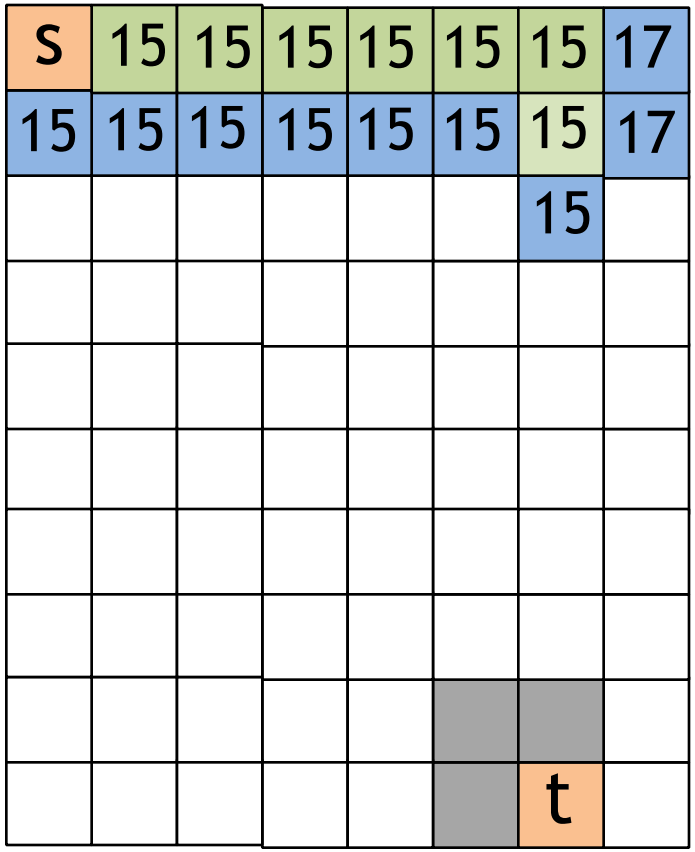
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

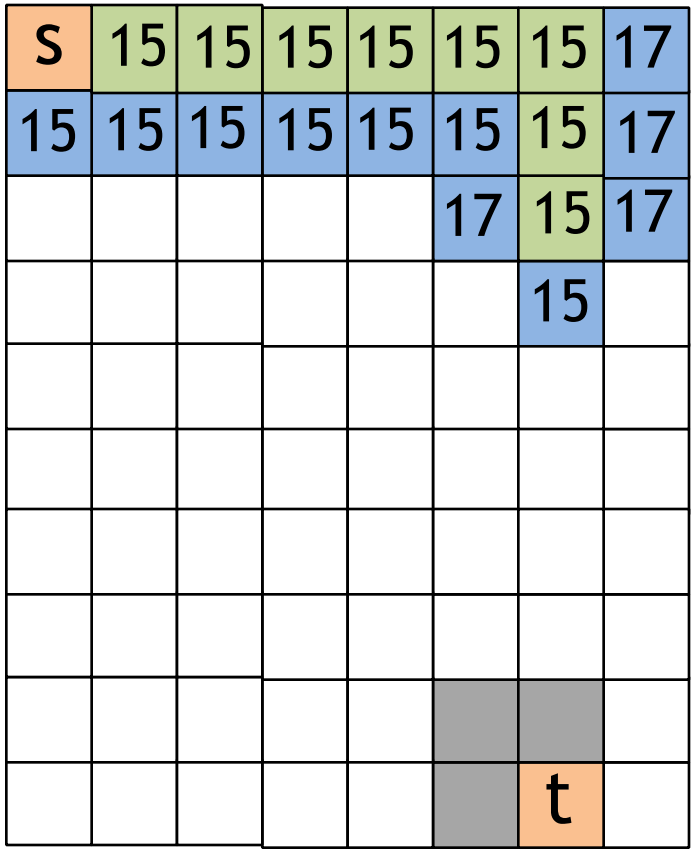
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

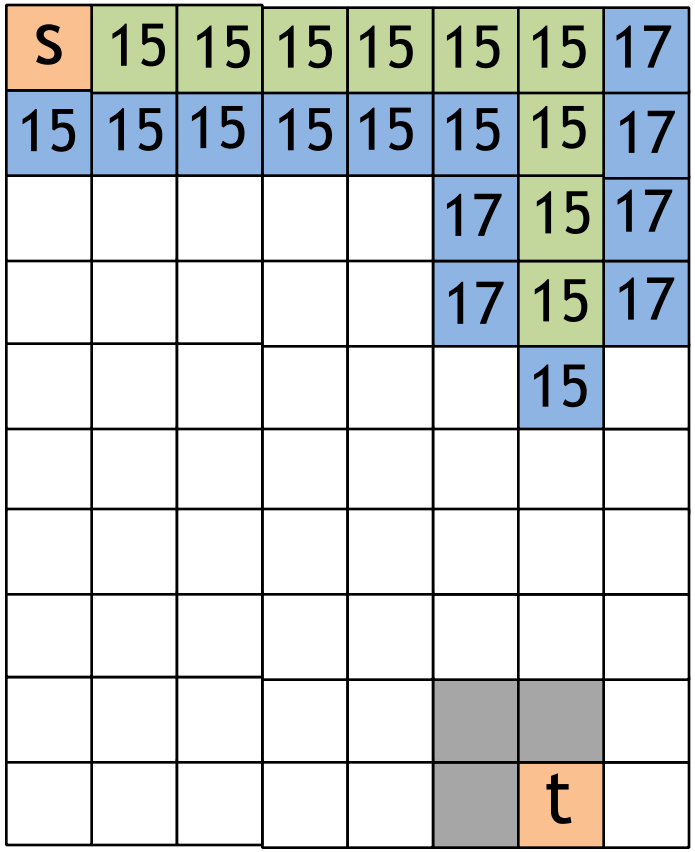
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

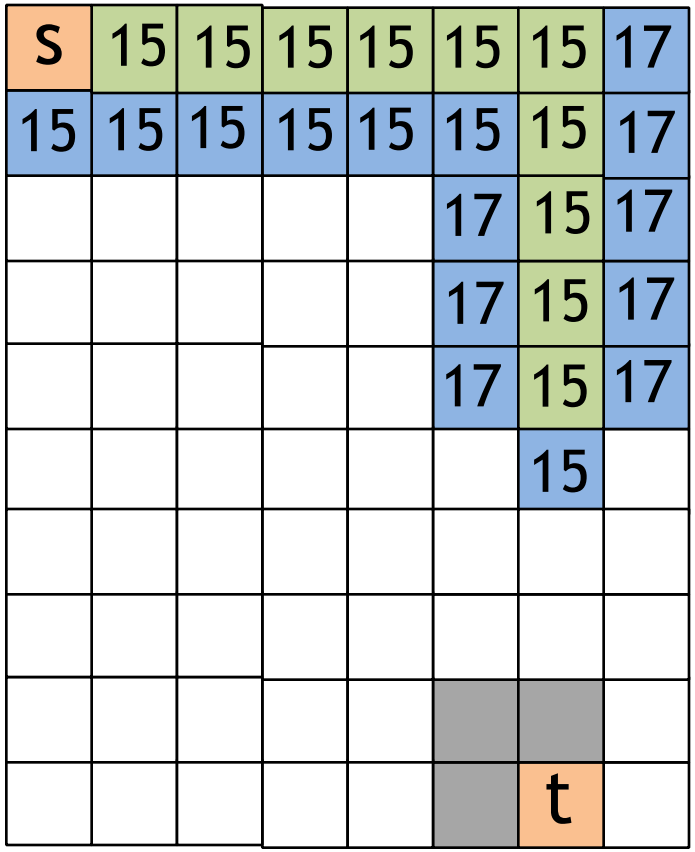
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

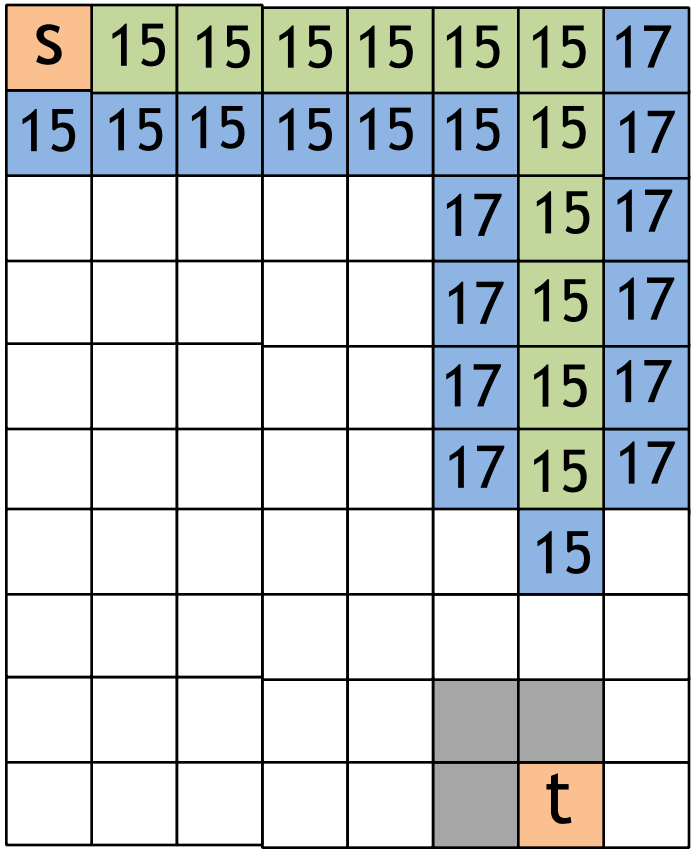
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

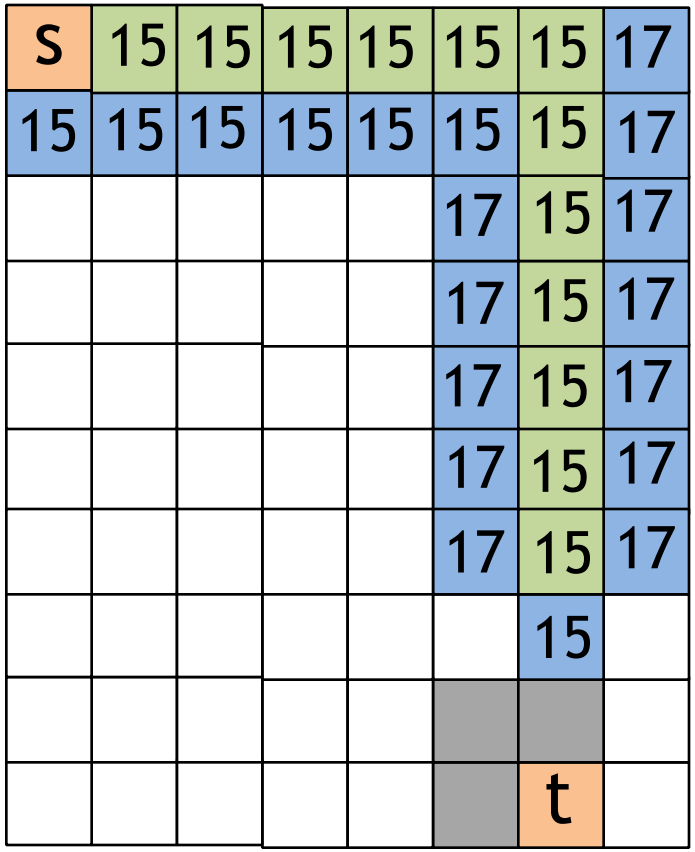
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

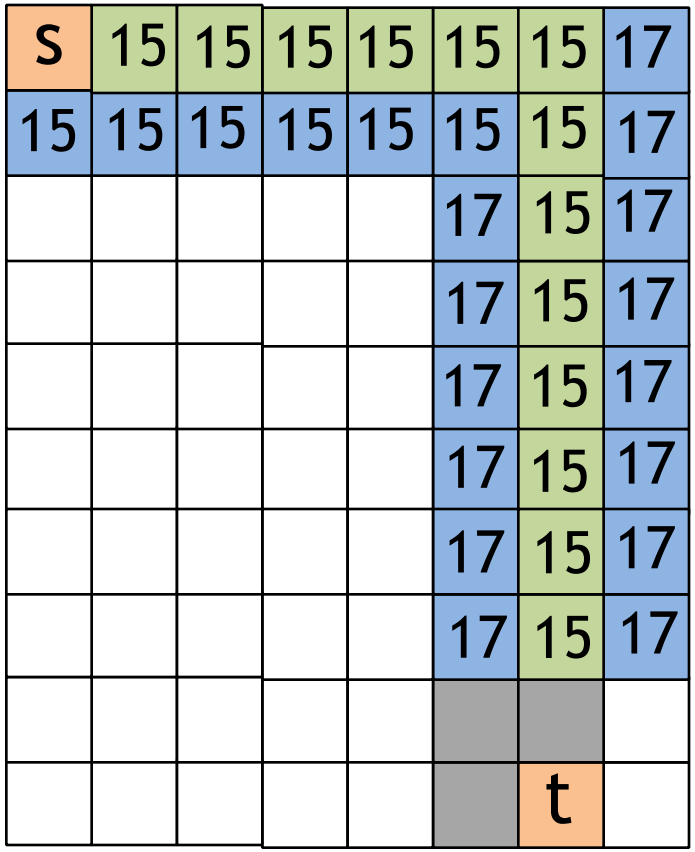
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

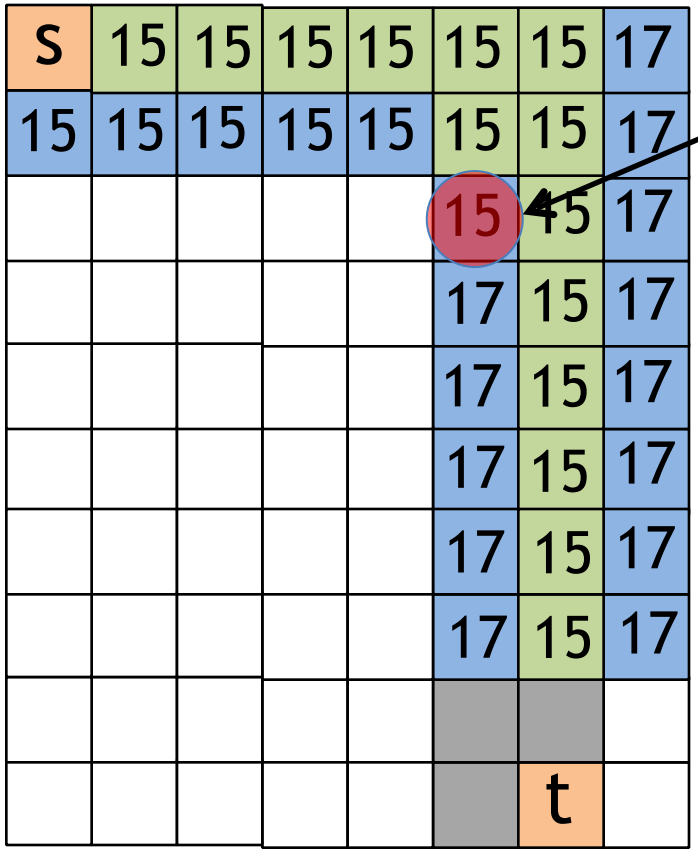
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

Another Ruben Example

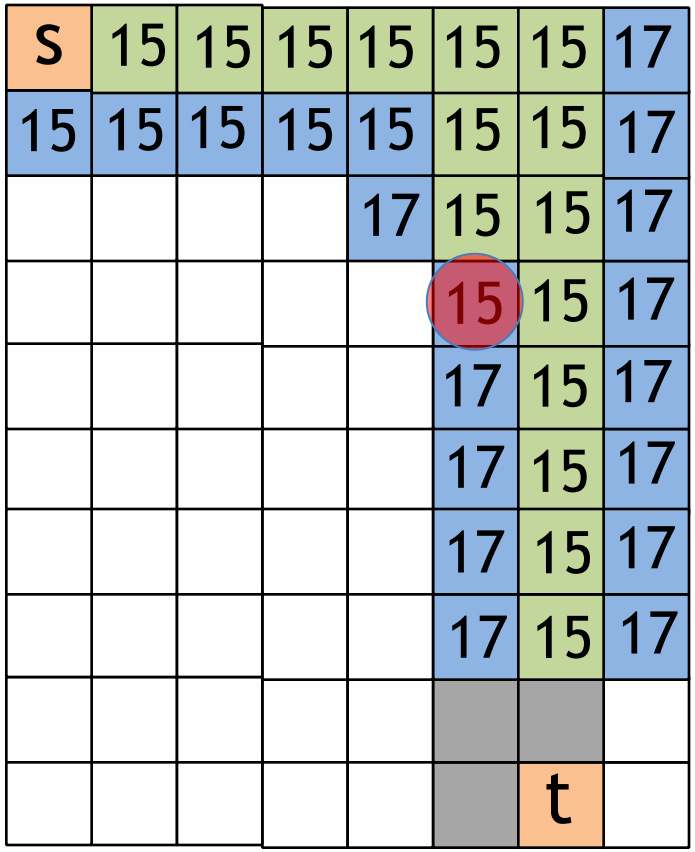


Adjust the cost because a shorter path to discover this cell has been uncovered

MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

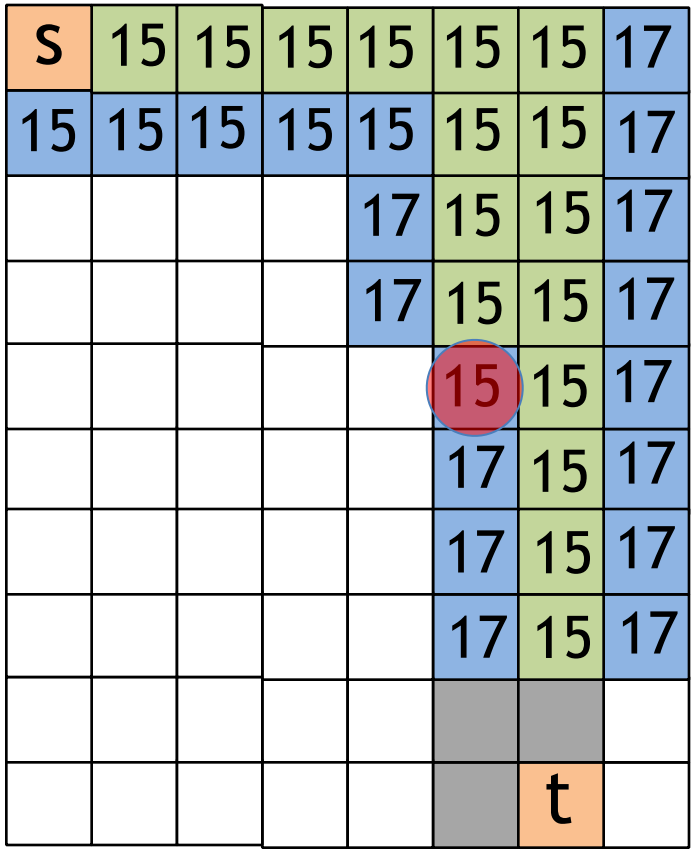
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

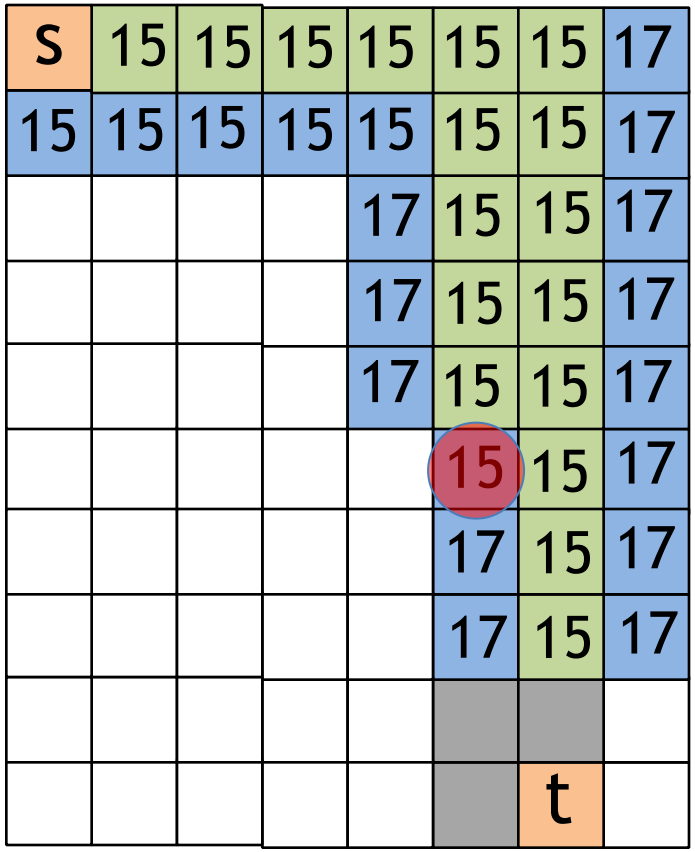
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

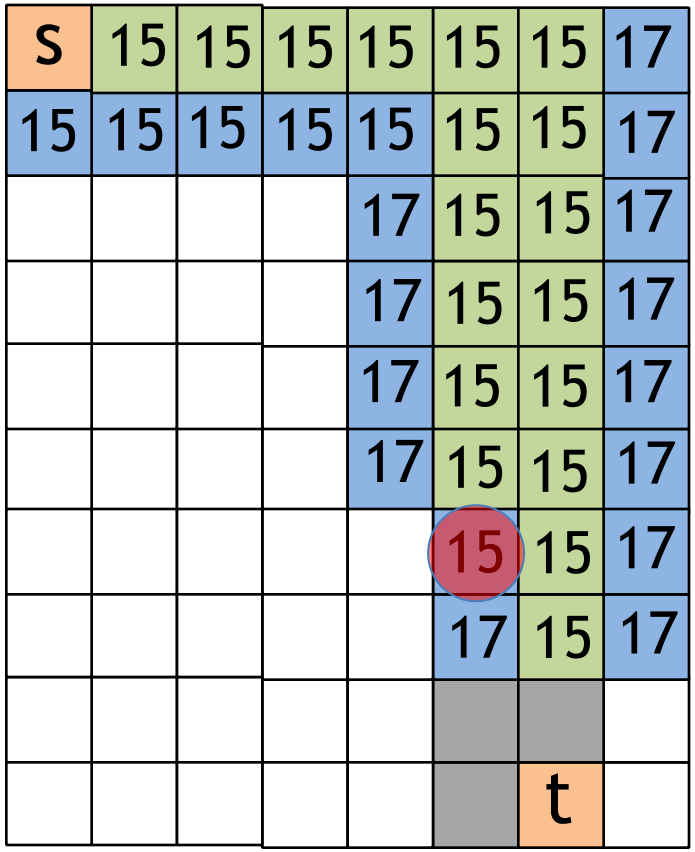
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

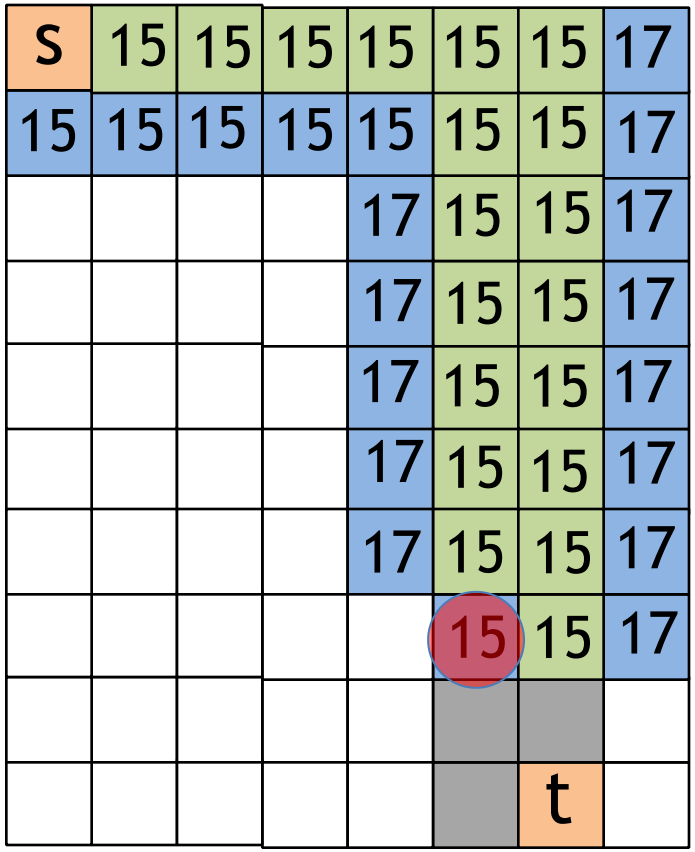
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

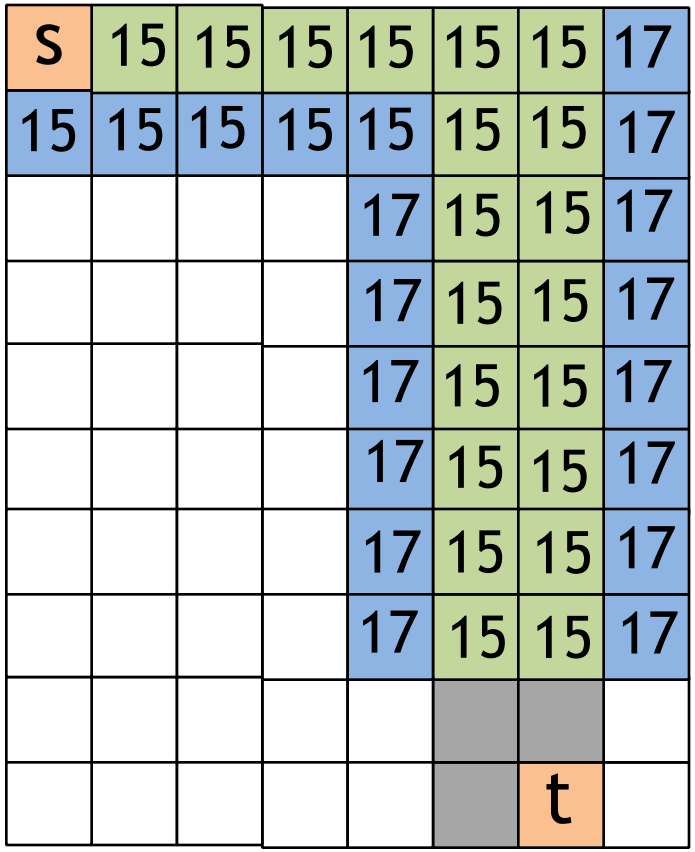
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

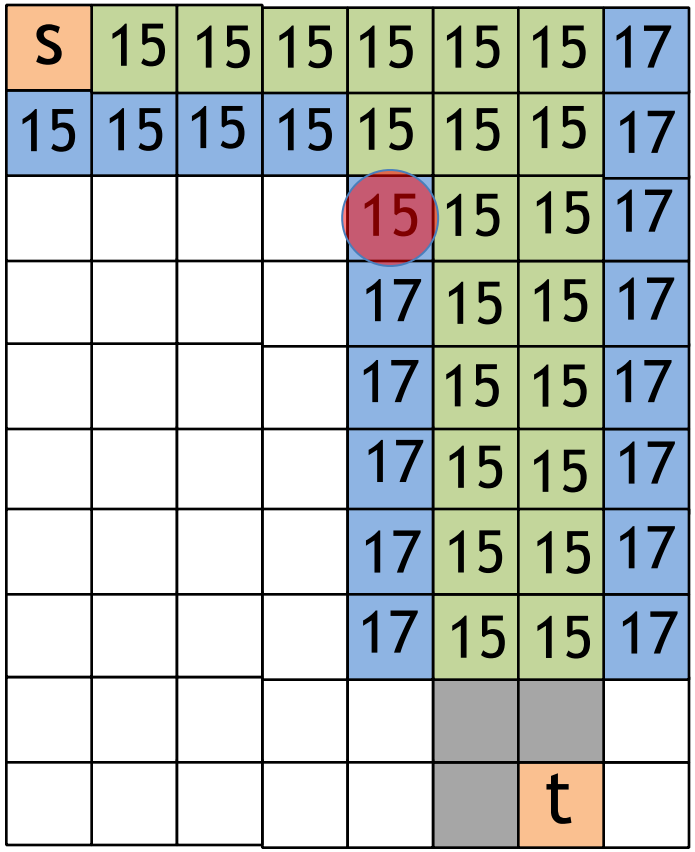
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

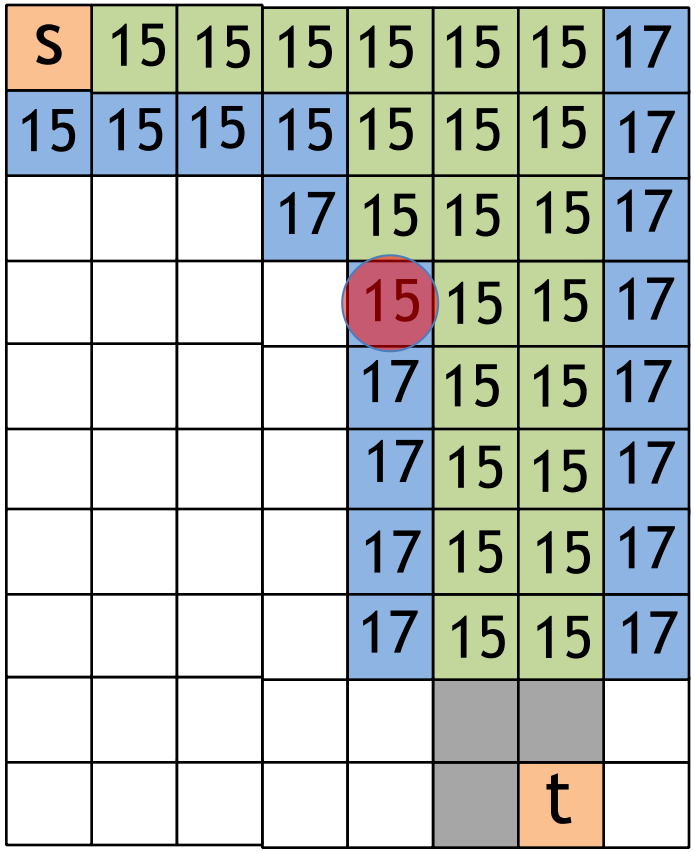
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

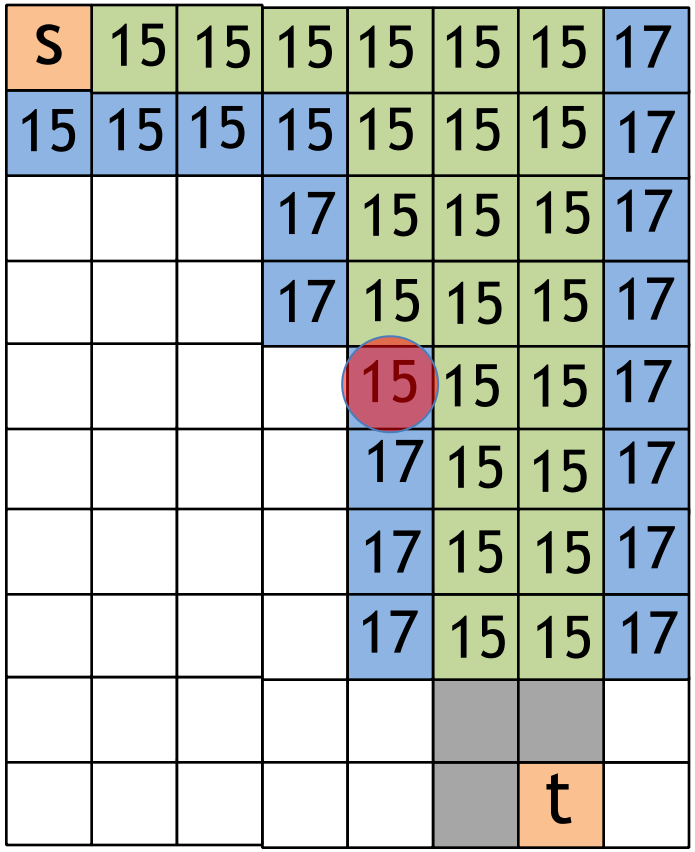
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

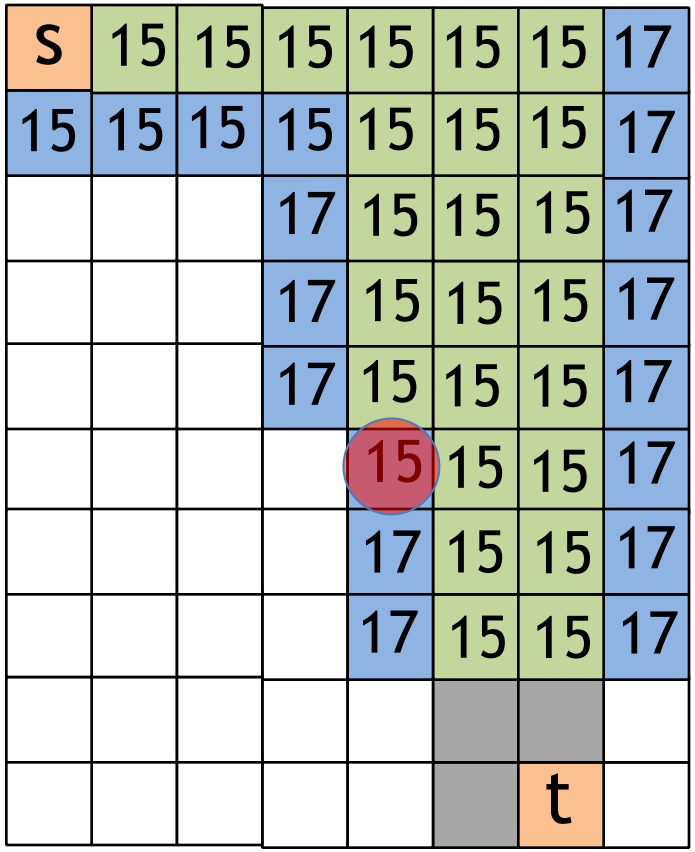
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

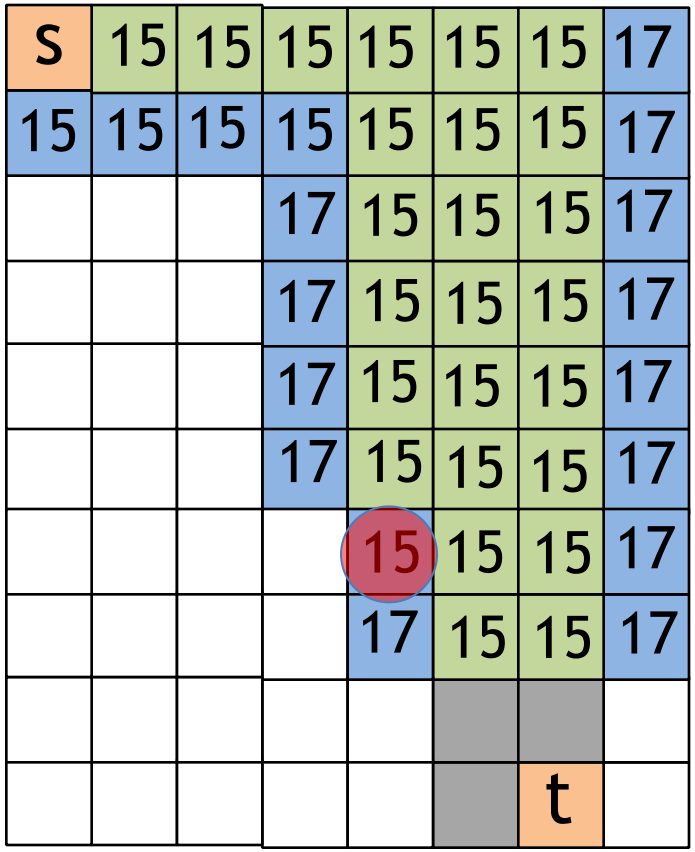
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

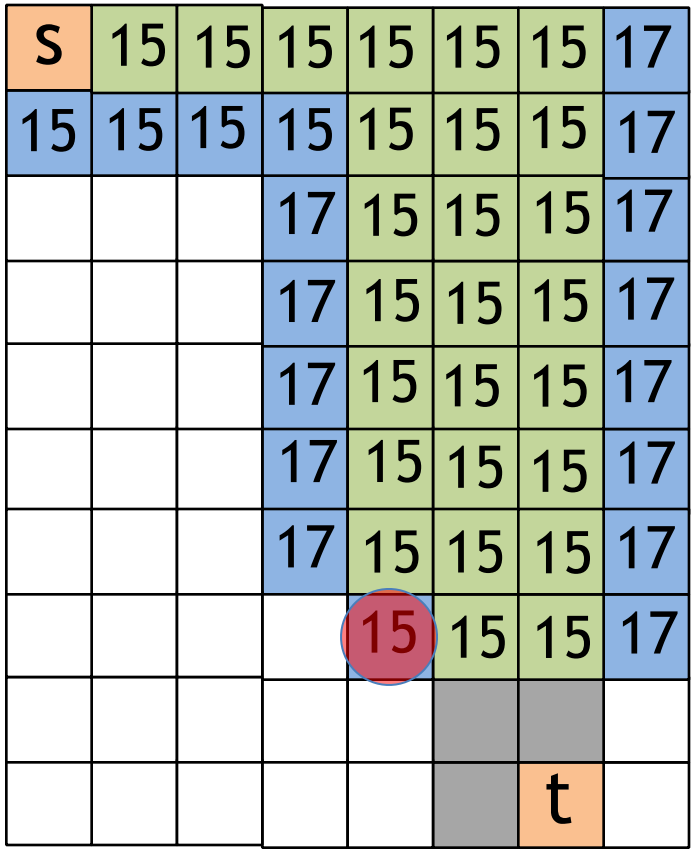
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

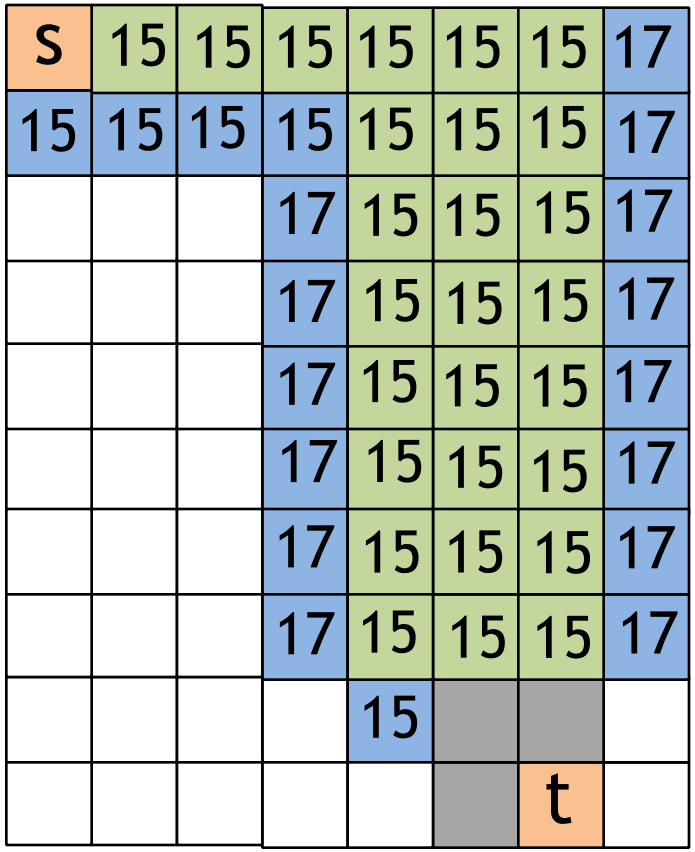
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

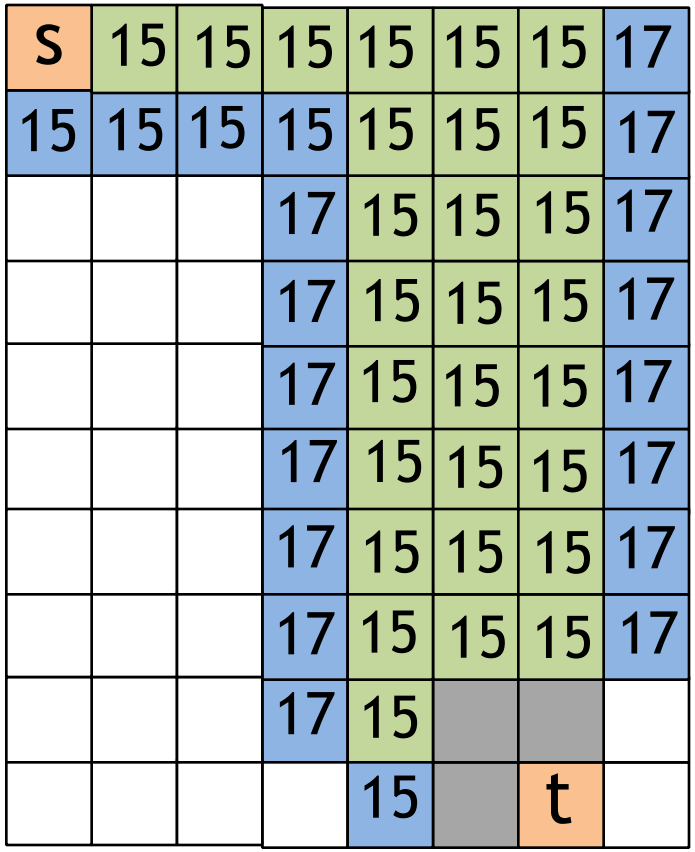
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

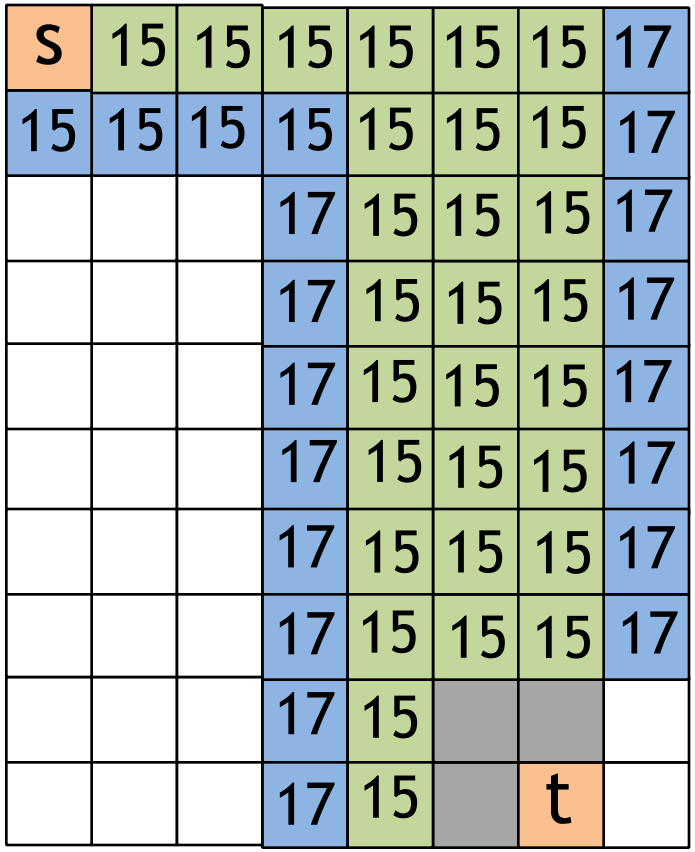
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

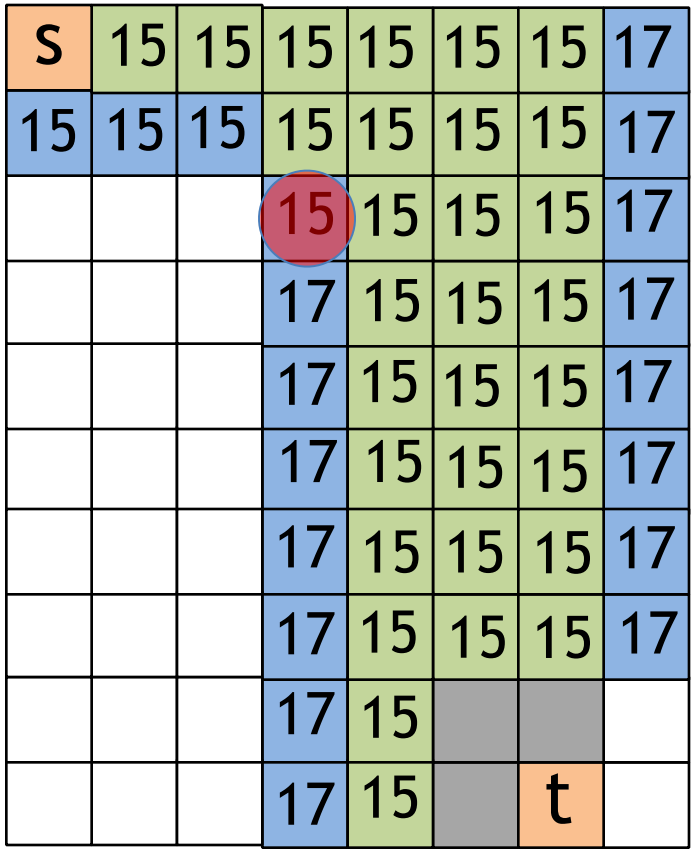
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

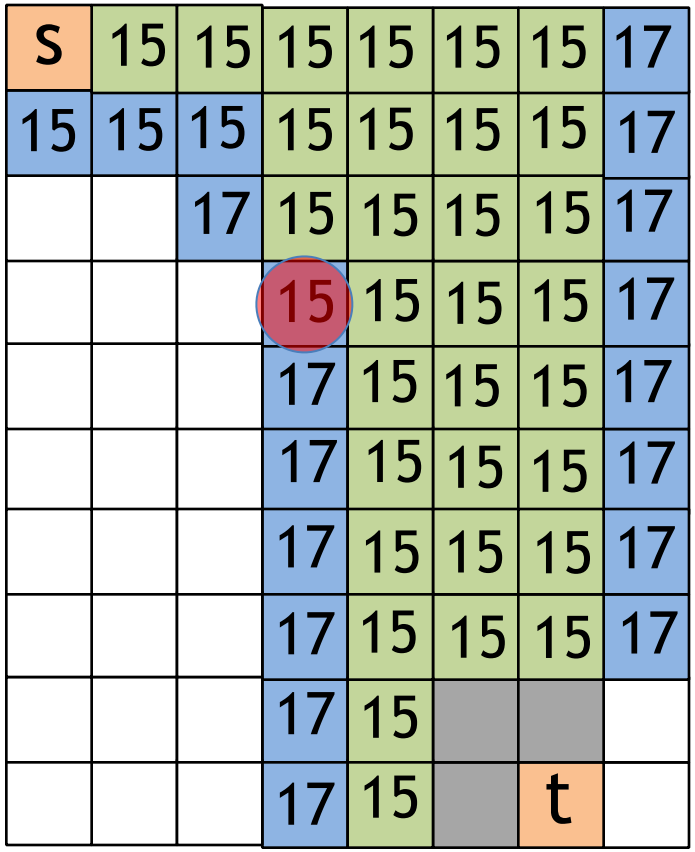
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

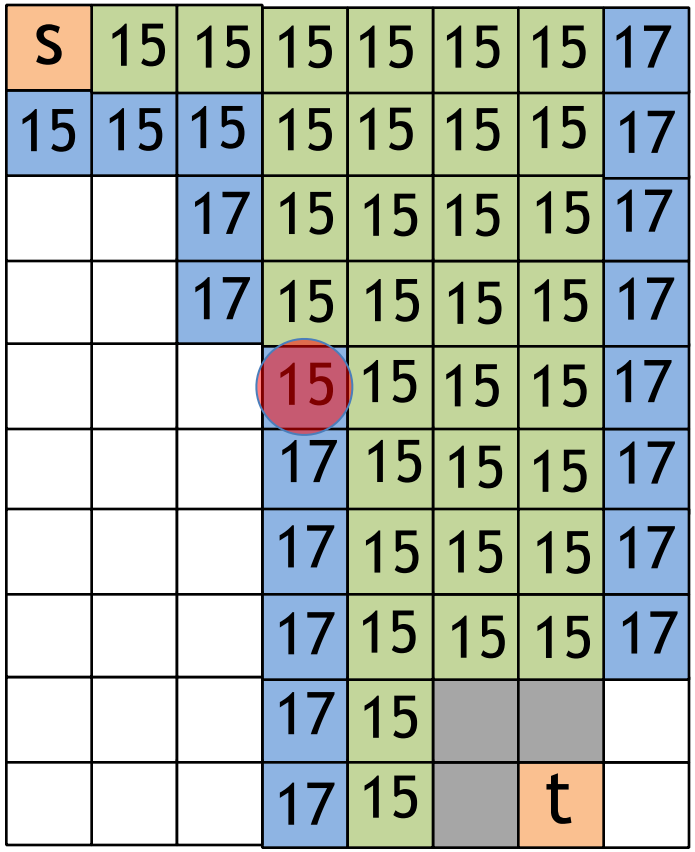
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

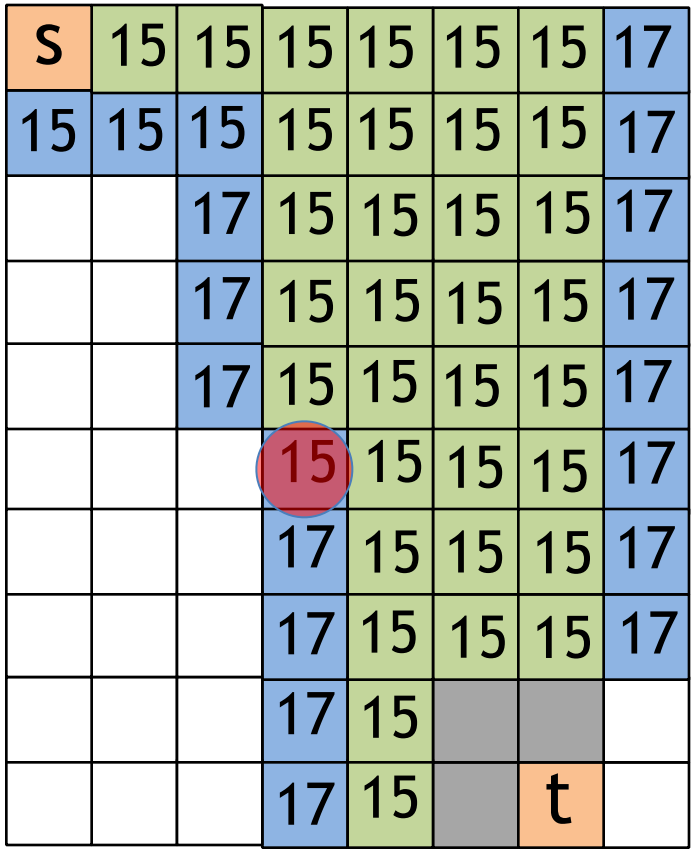
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

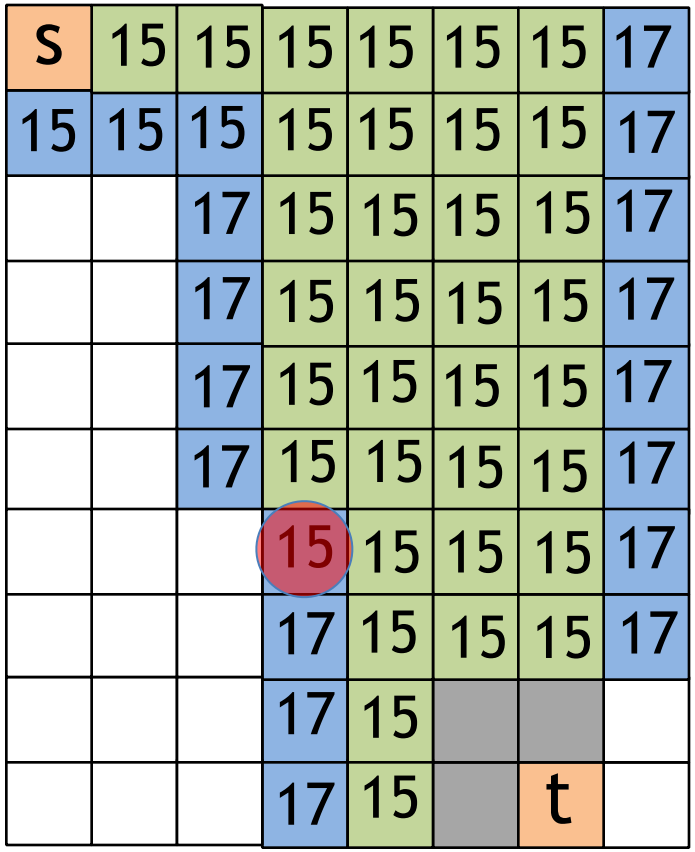
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

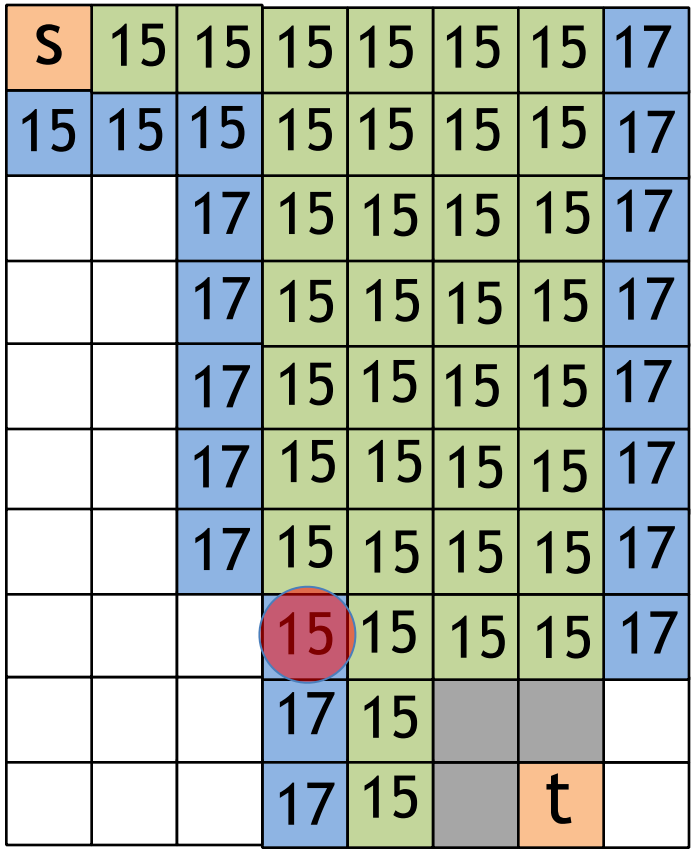
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

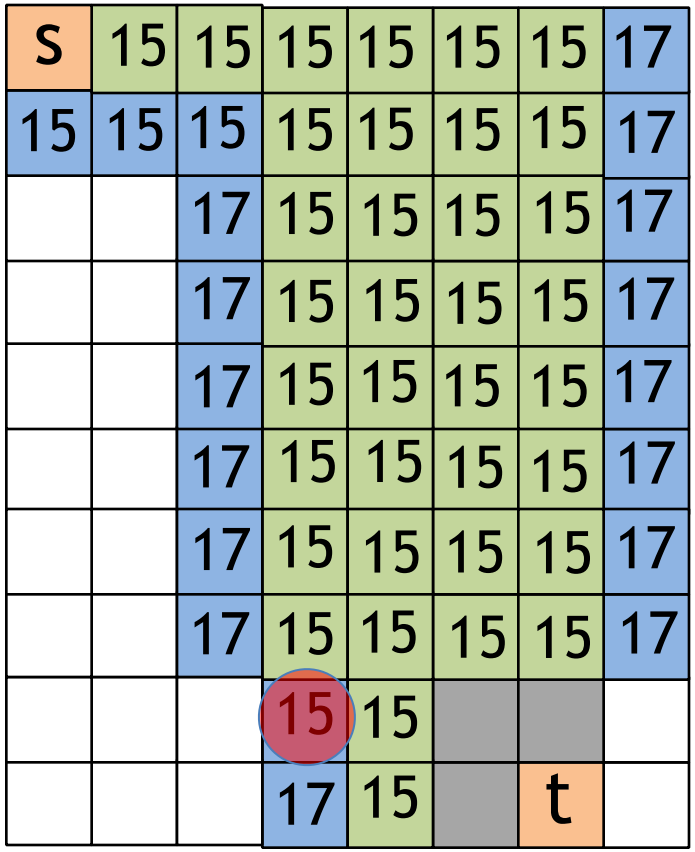
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

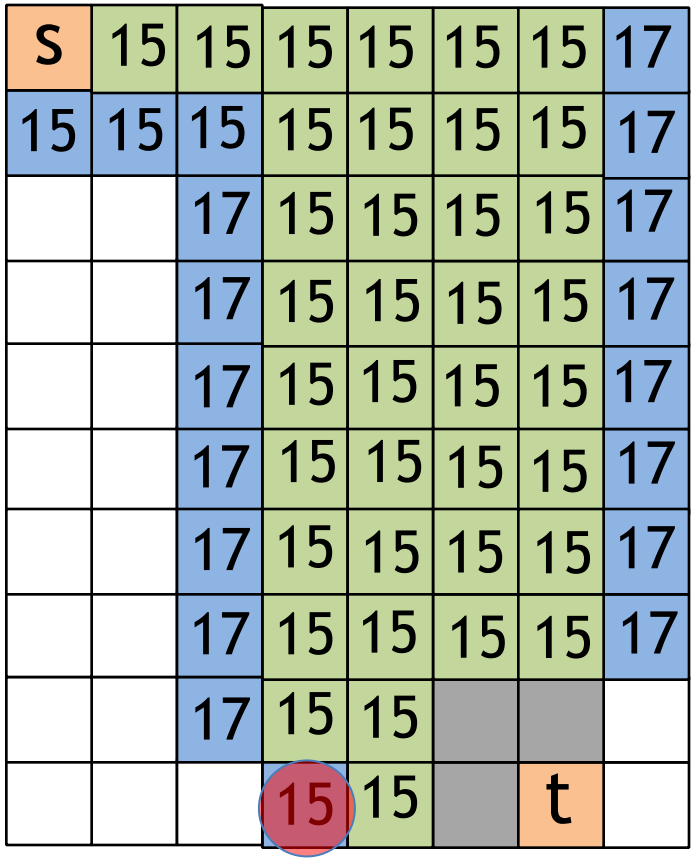
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

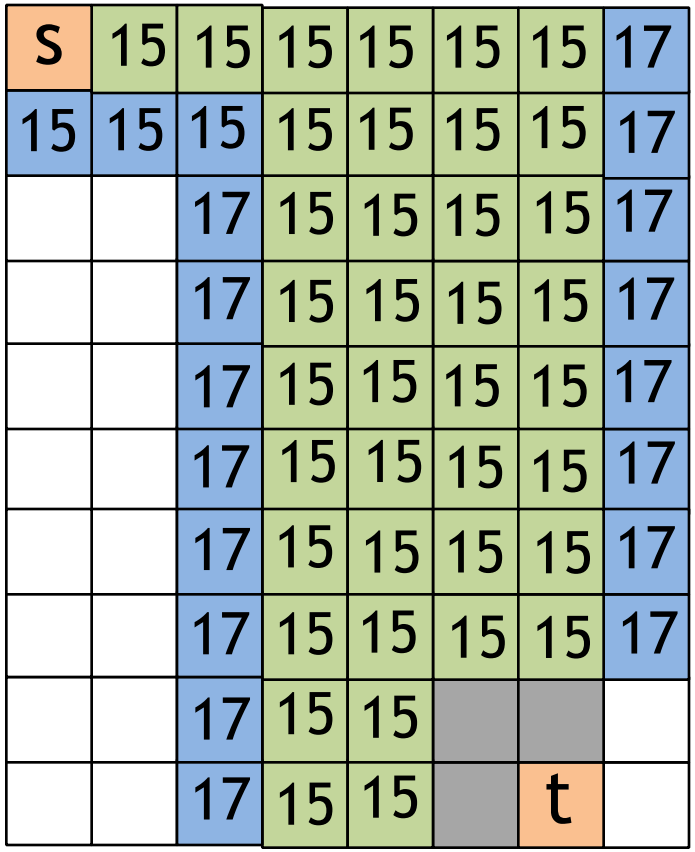
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

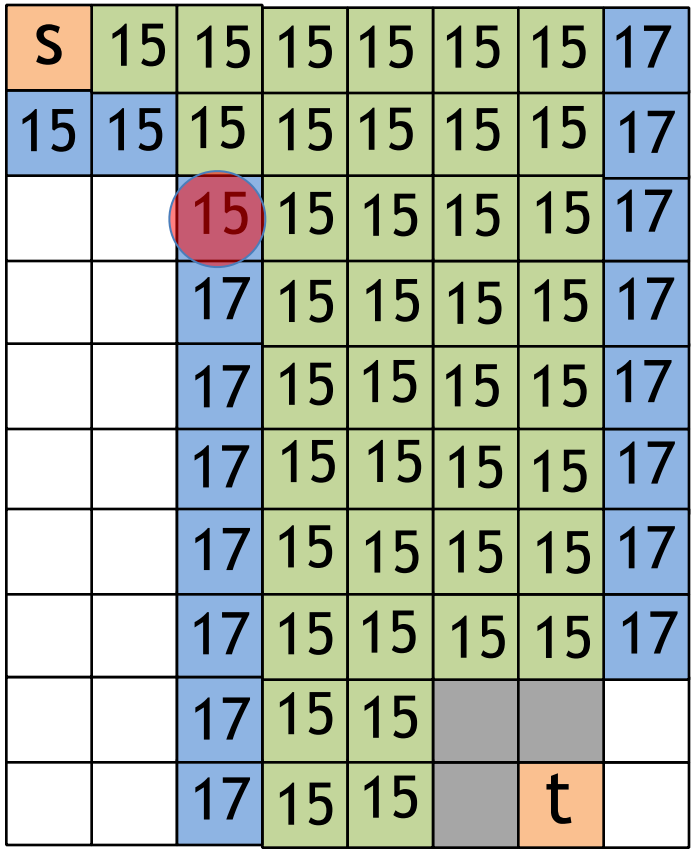
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

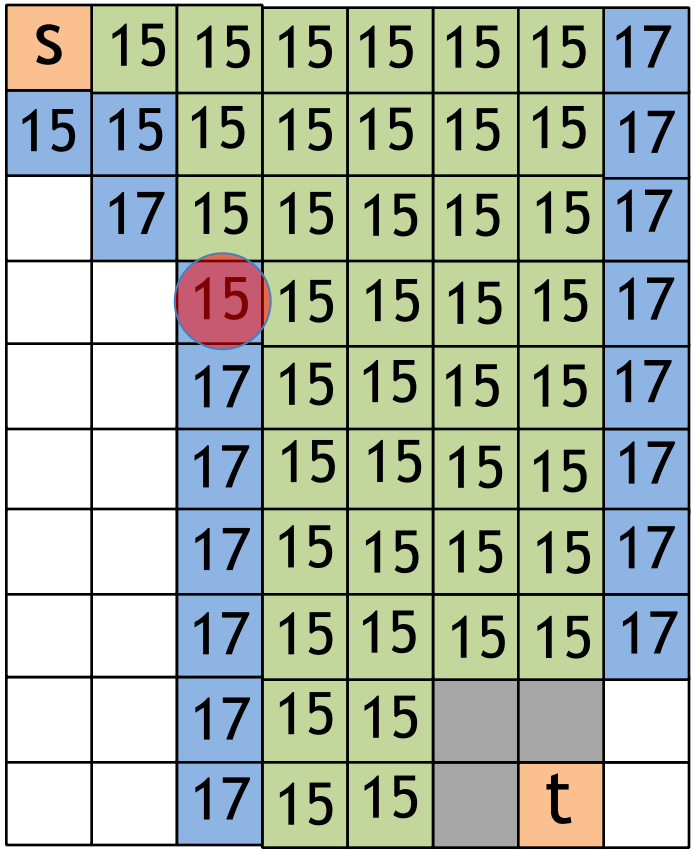
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

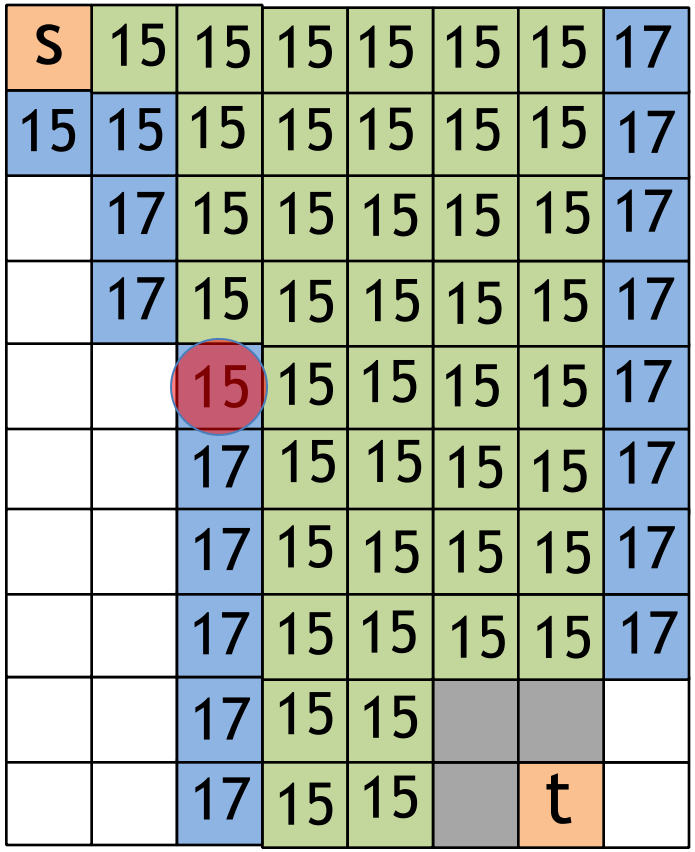
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

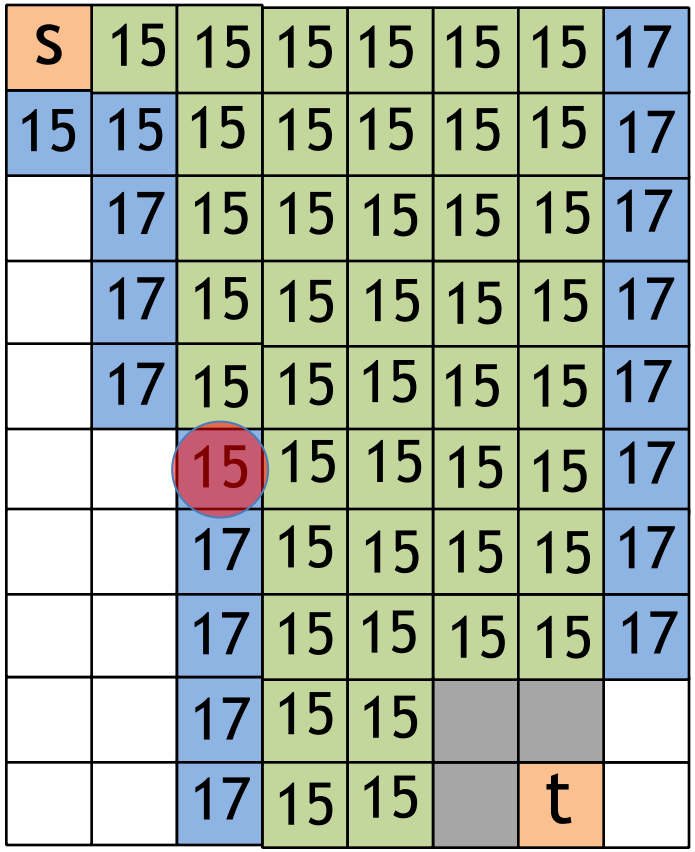
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

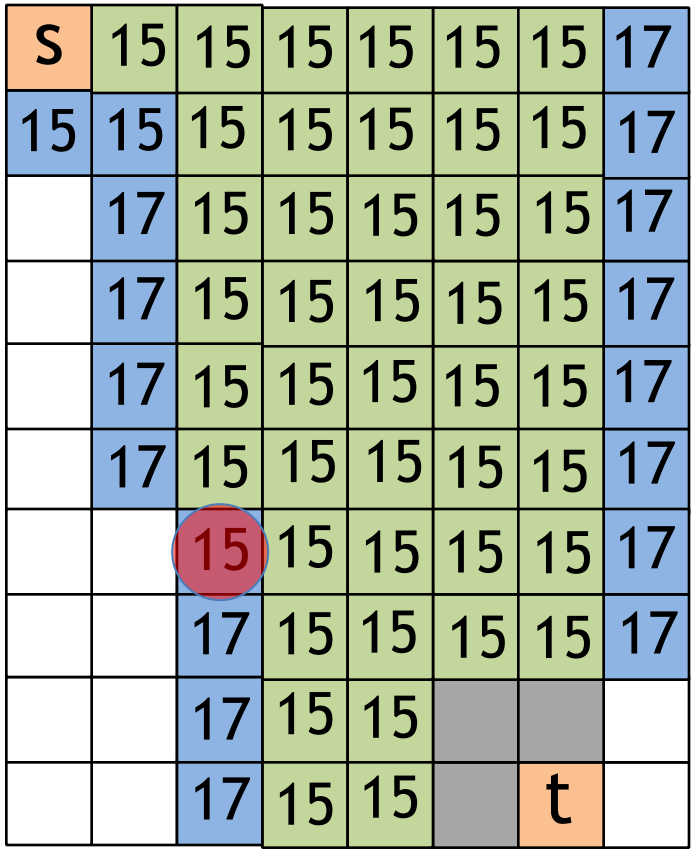
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

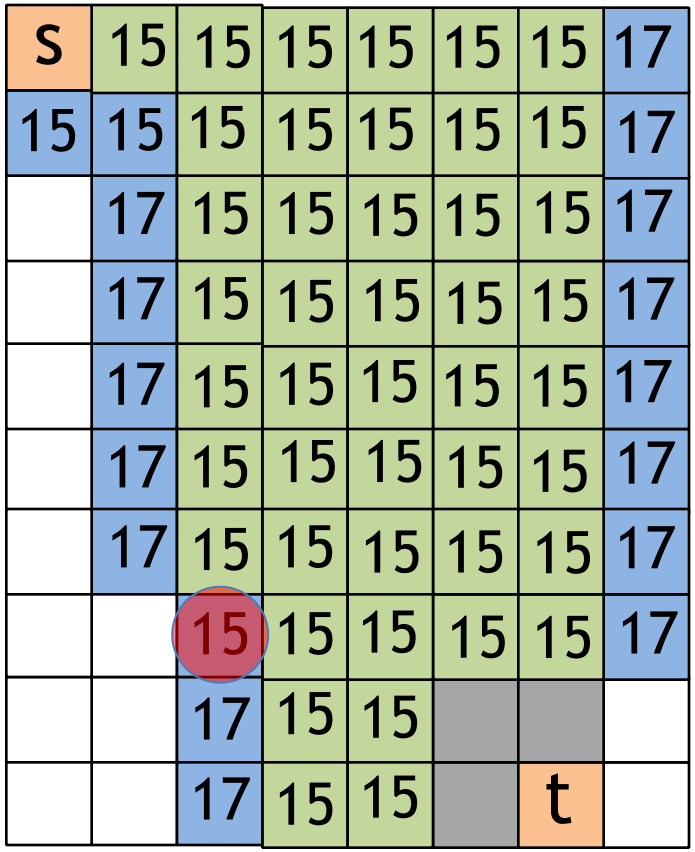
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

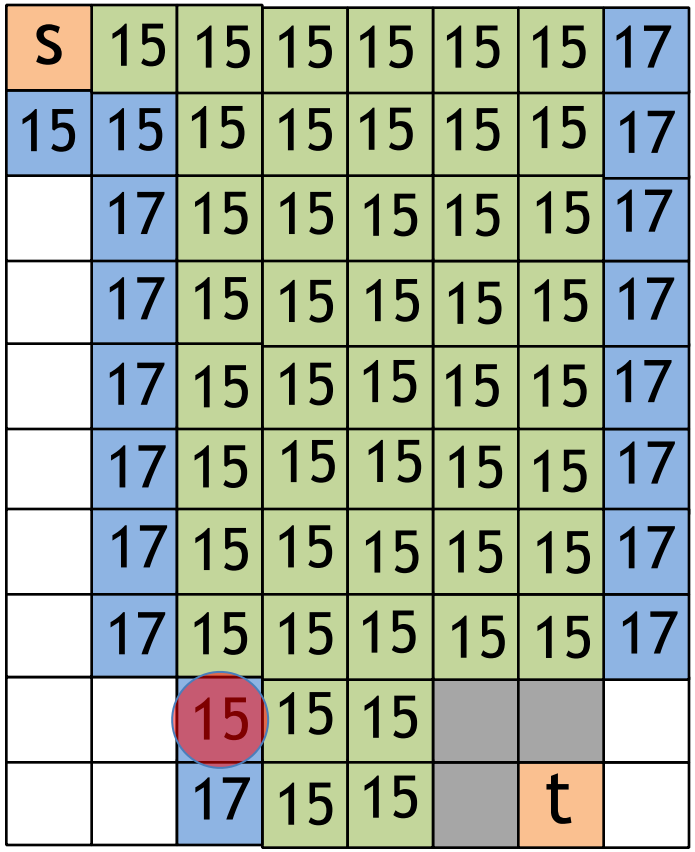
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

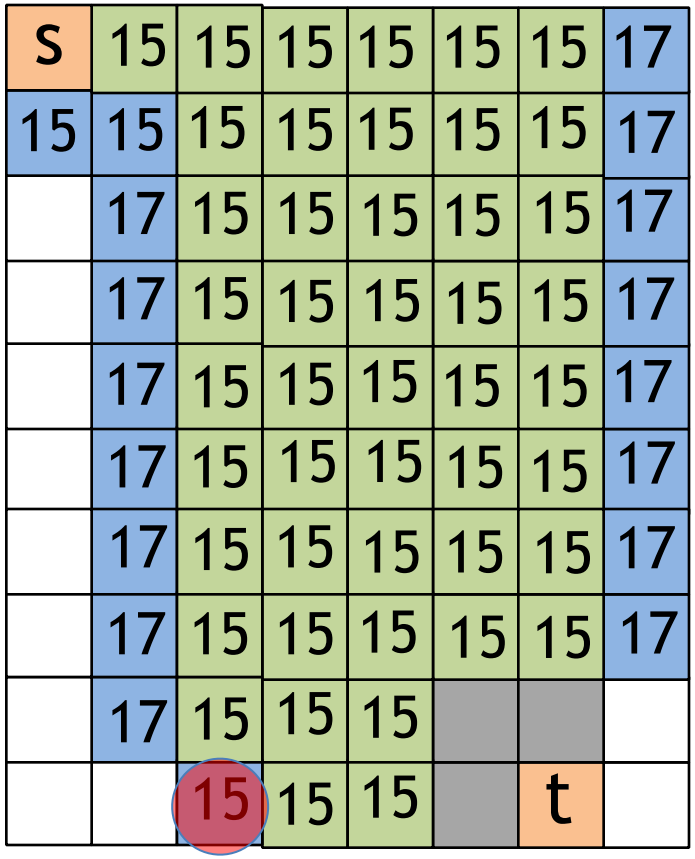
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

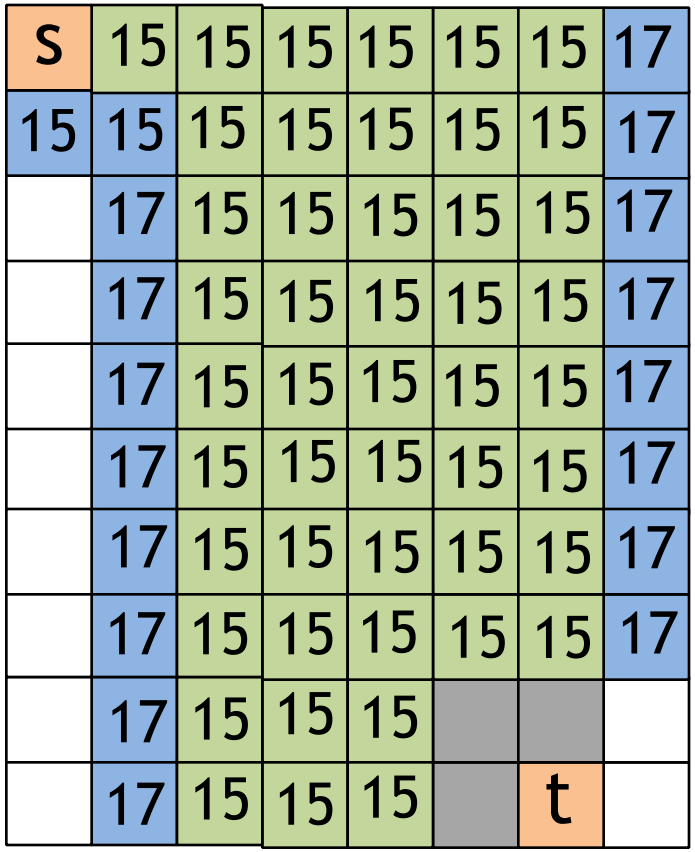
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

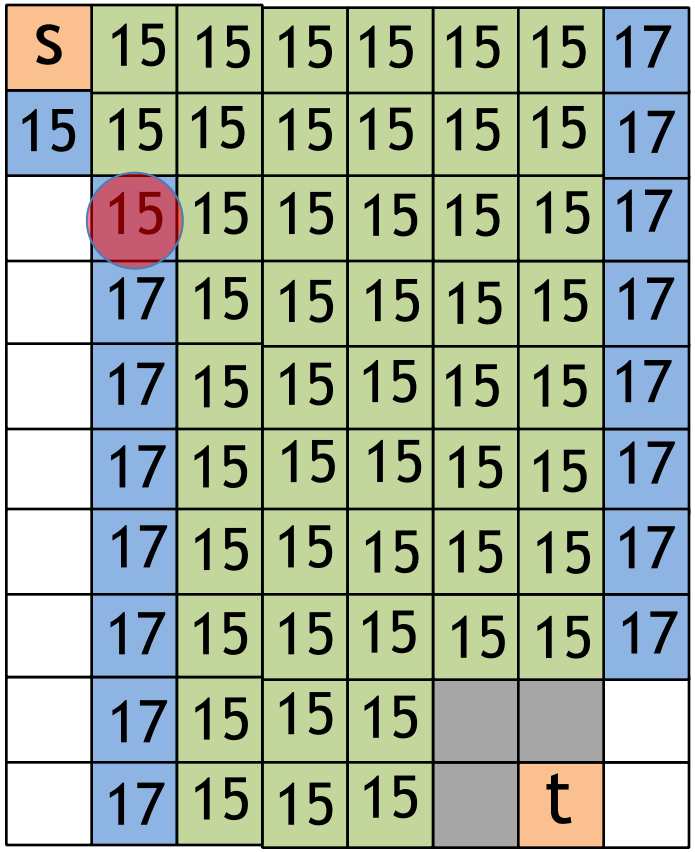
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

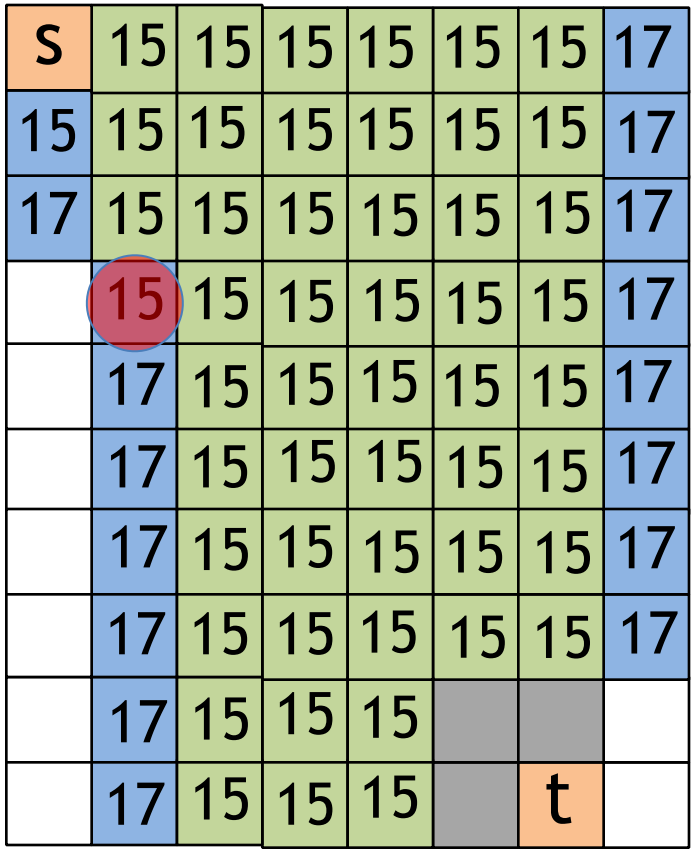
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

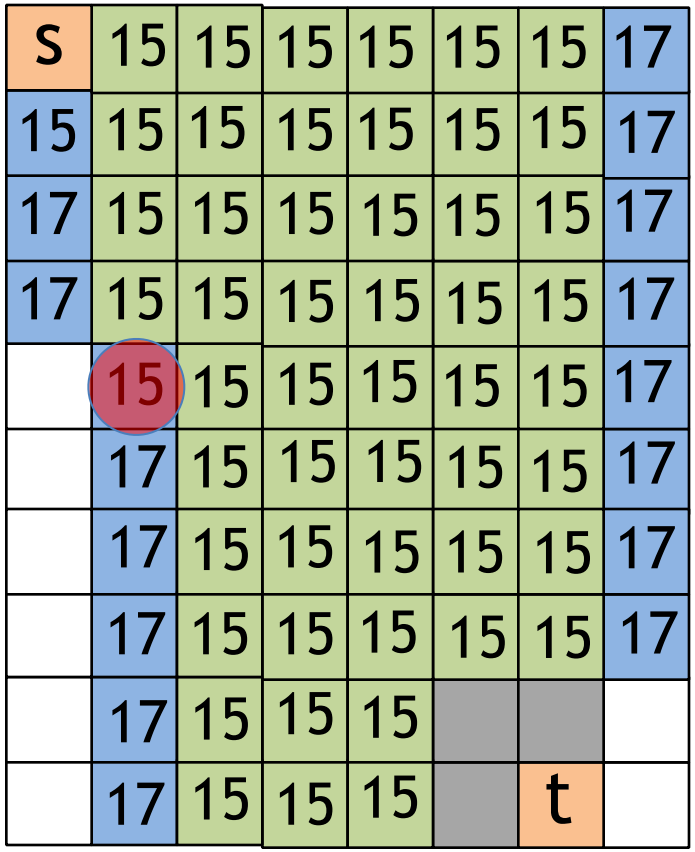
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

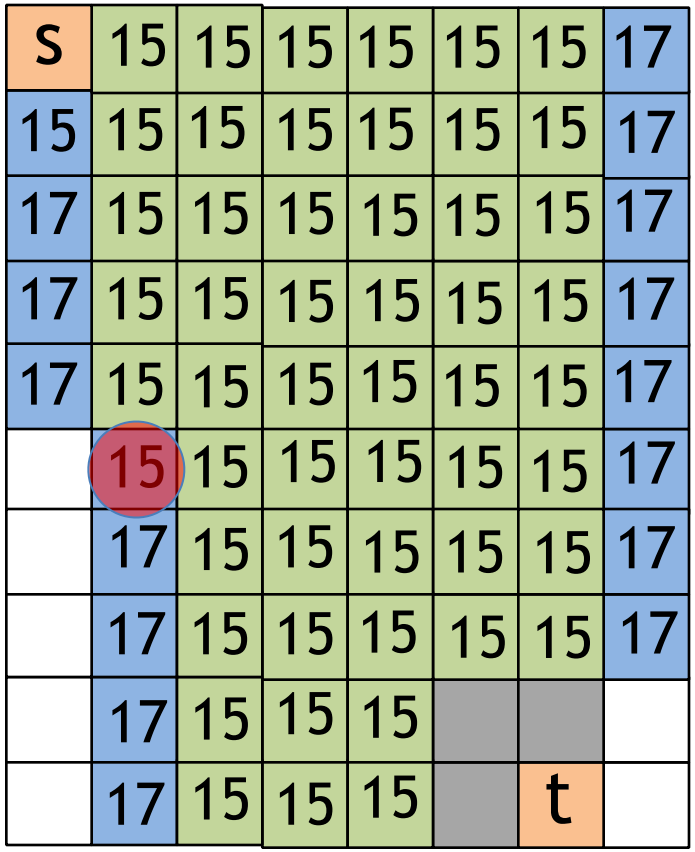
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

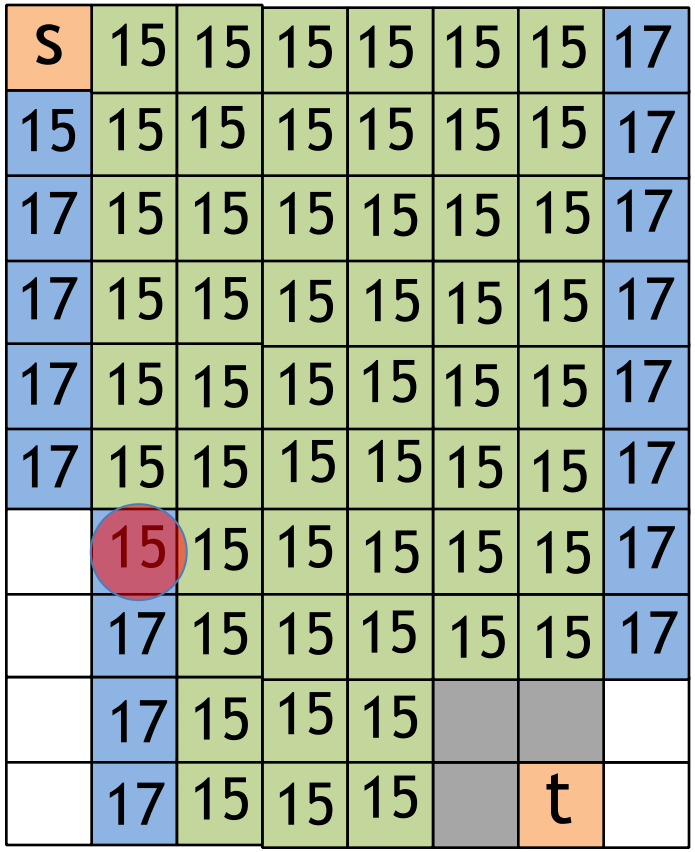
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

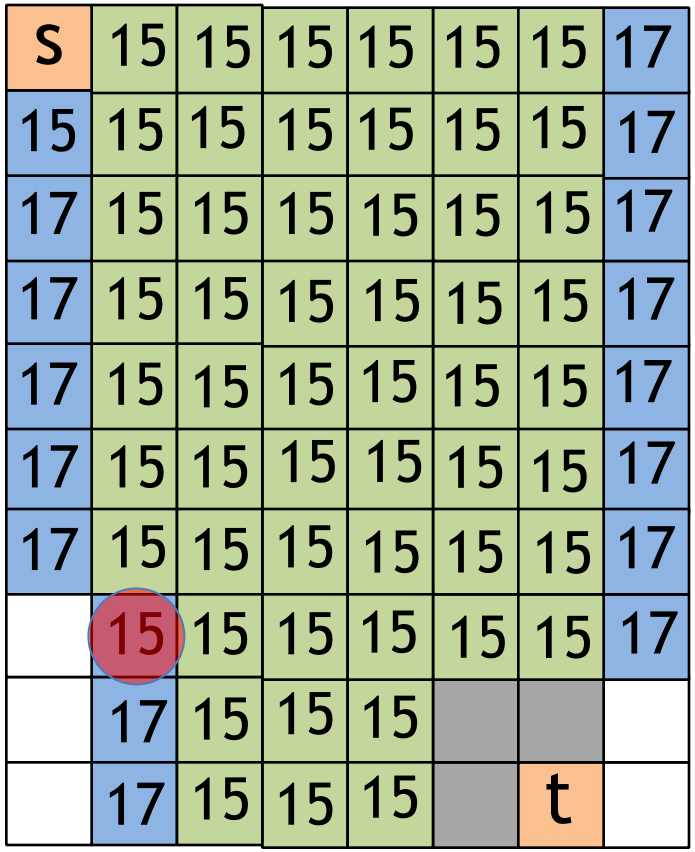
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

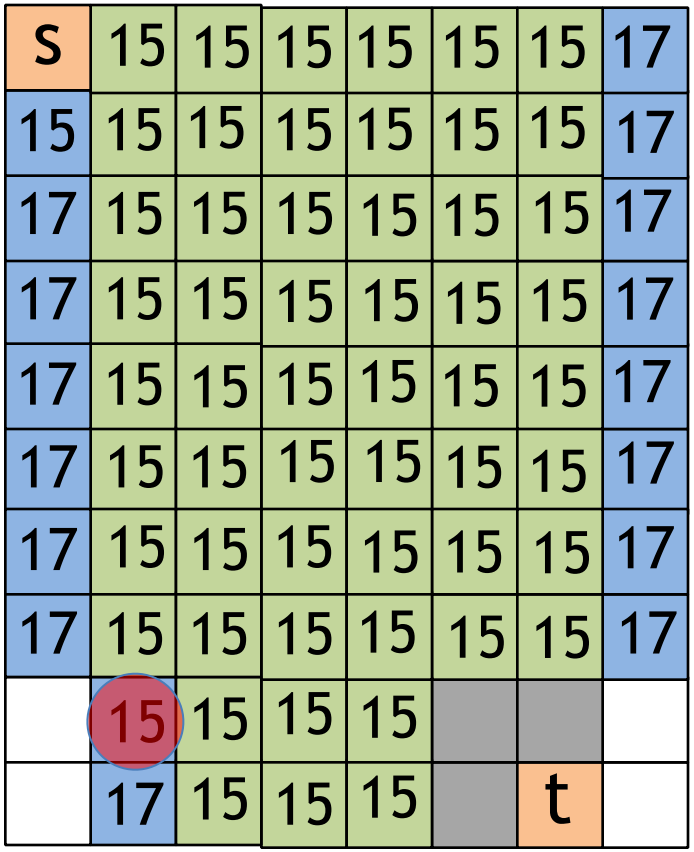
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

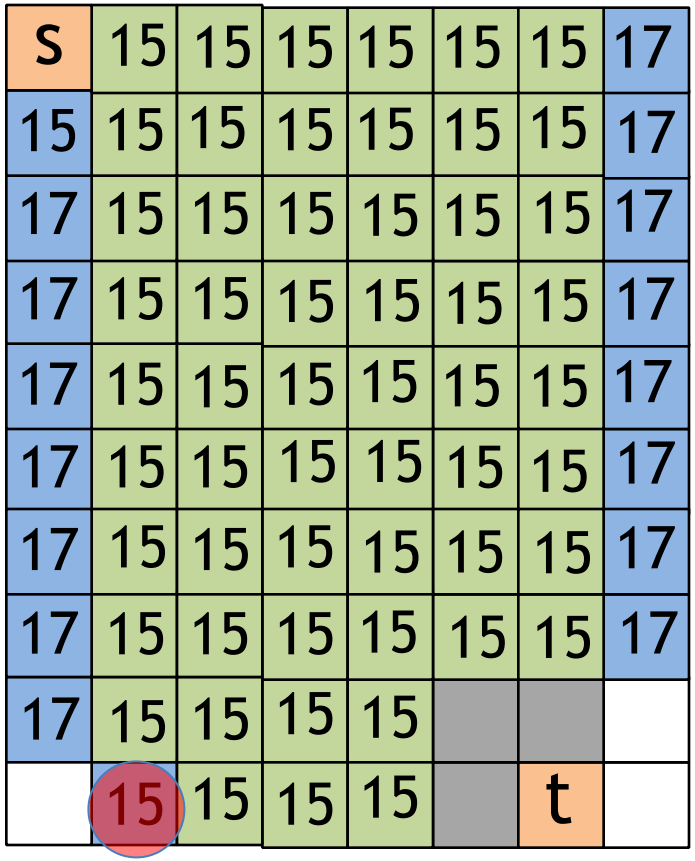
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

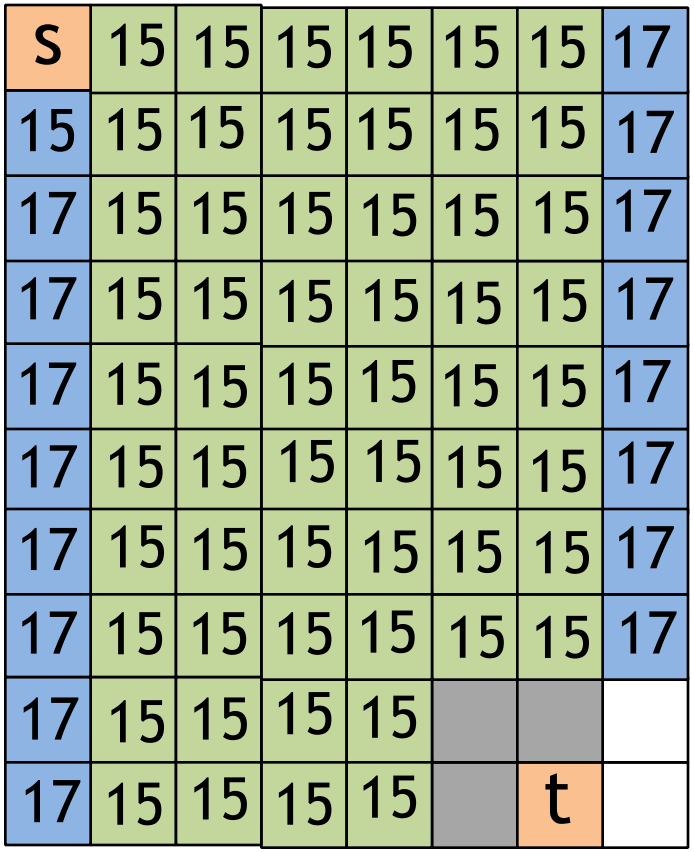
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

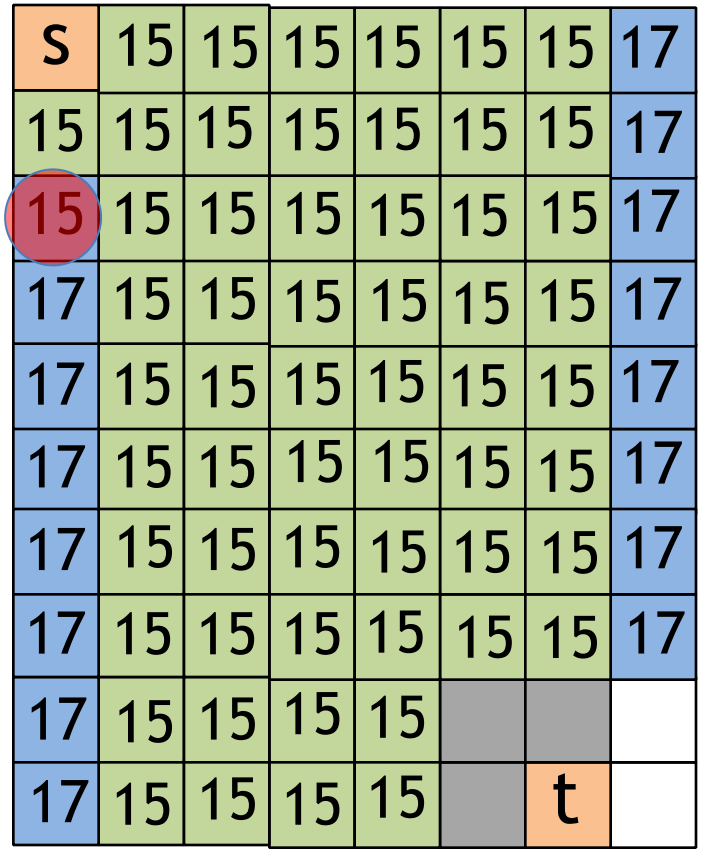
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

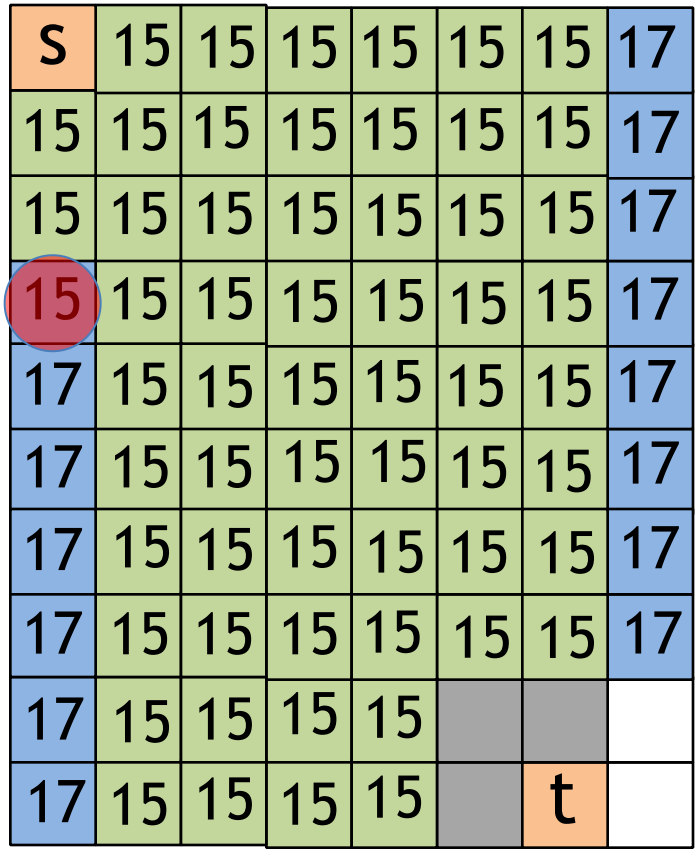
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

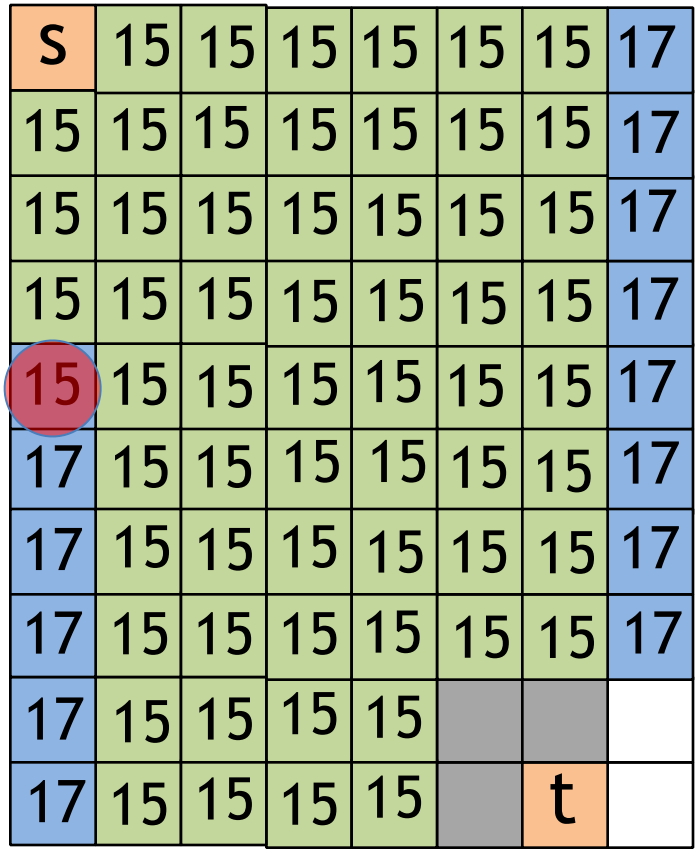
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

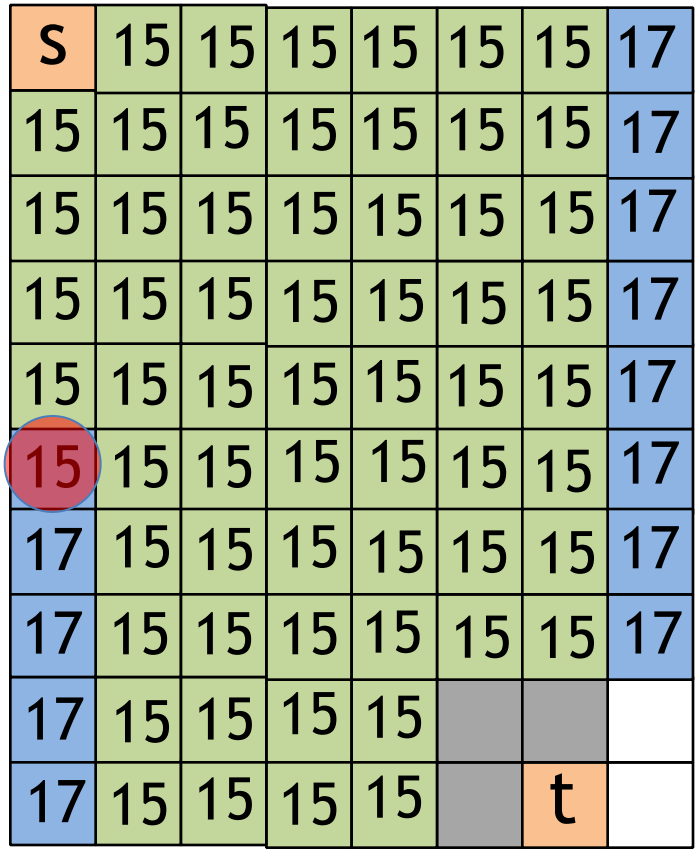
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

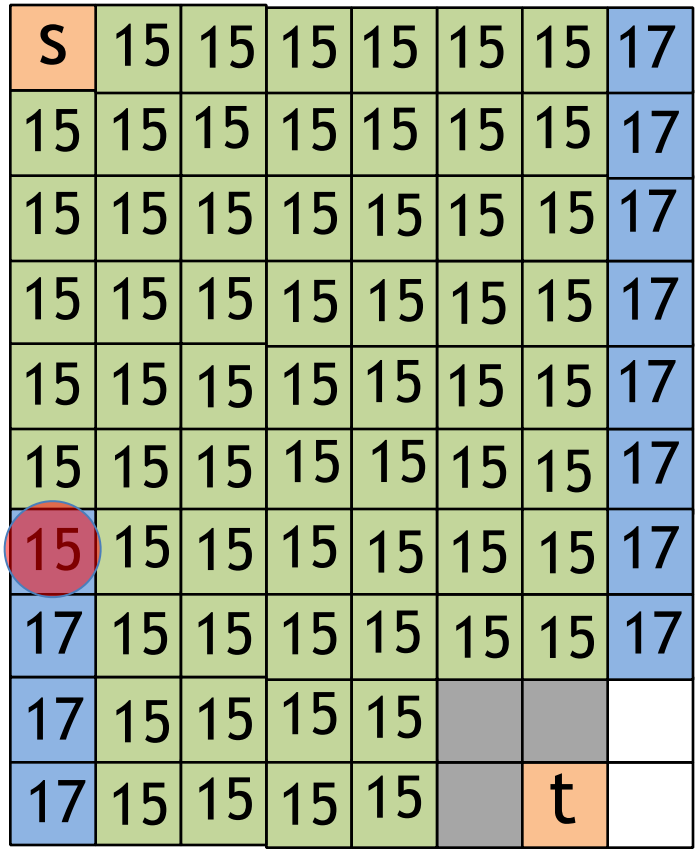
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

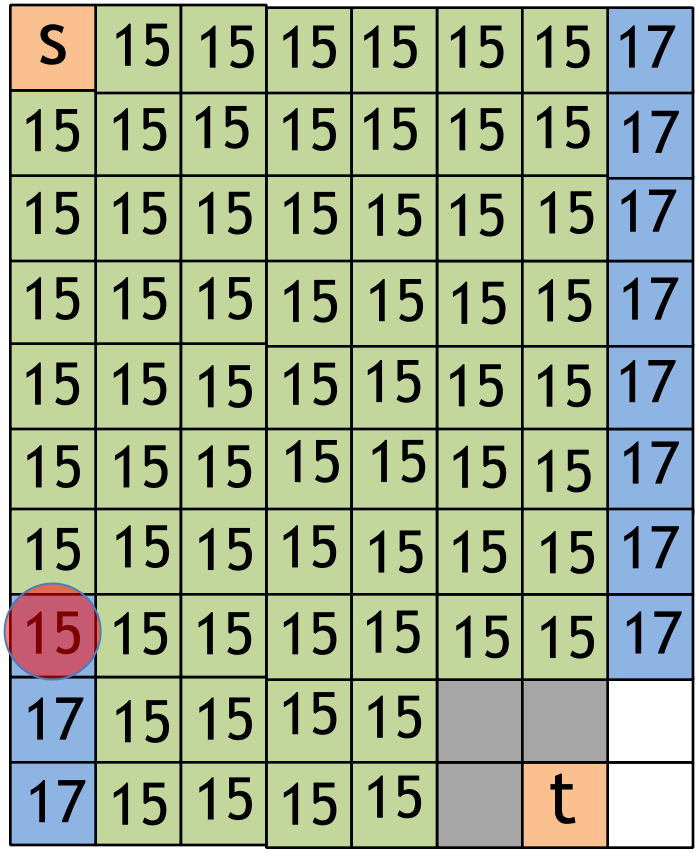
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

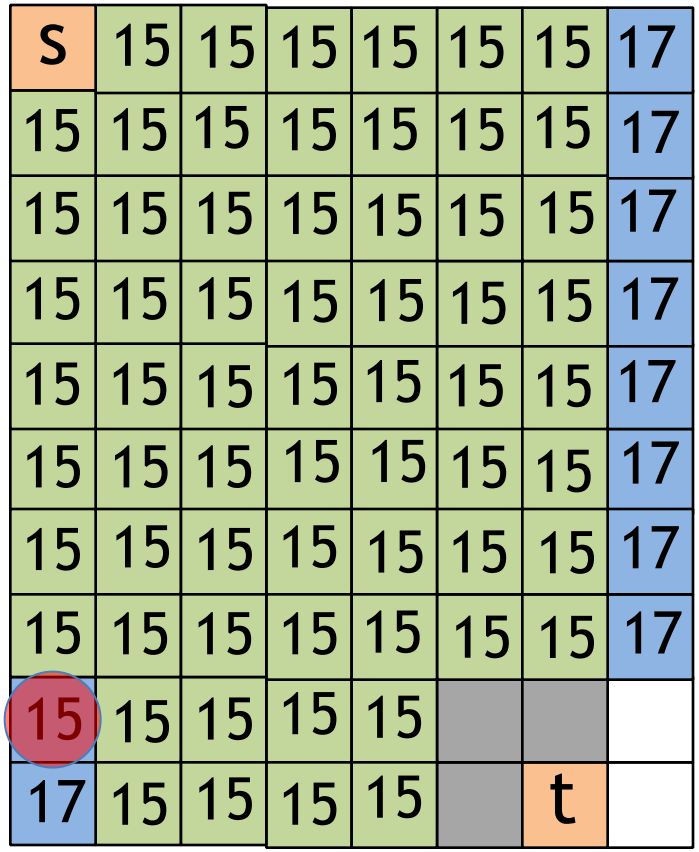
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

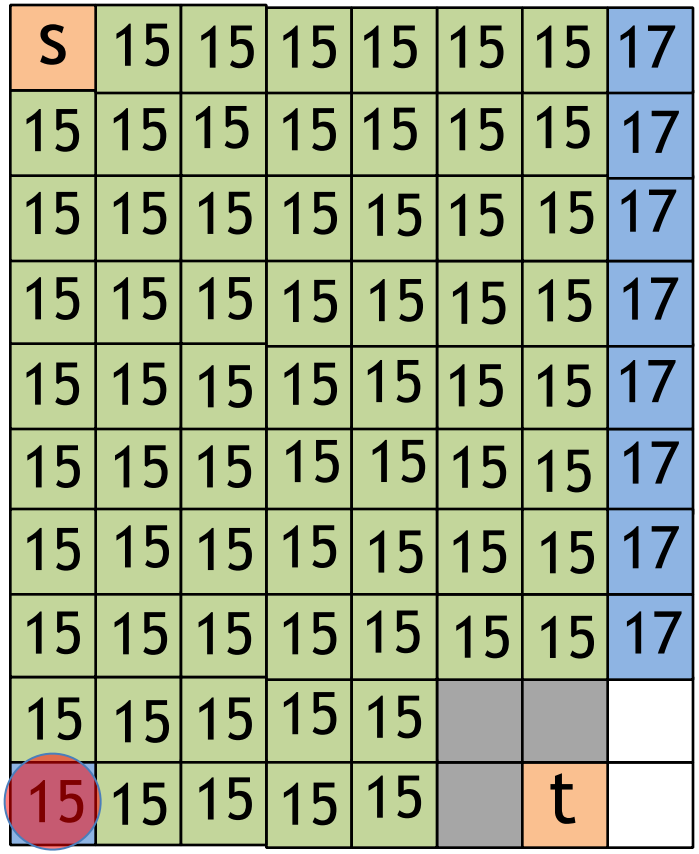
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

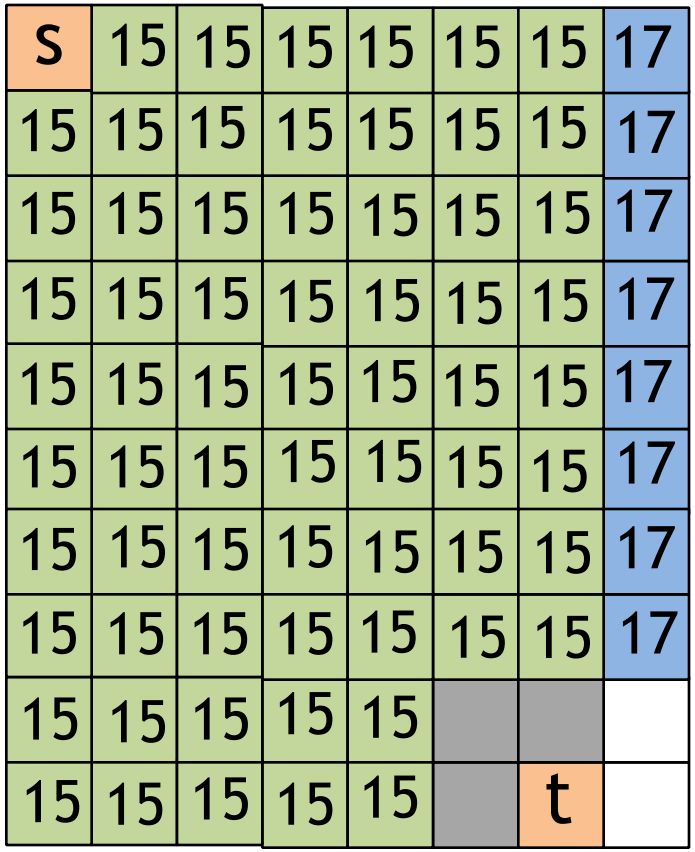
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

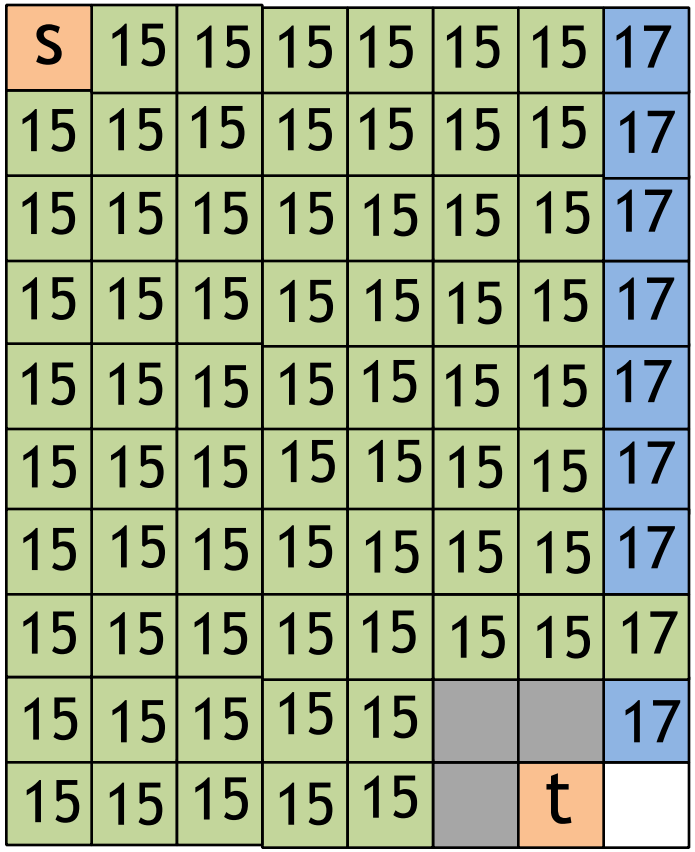
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

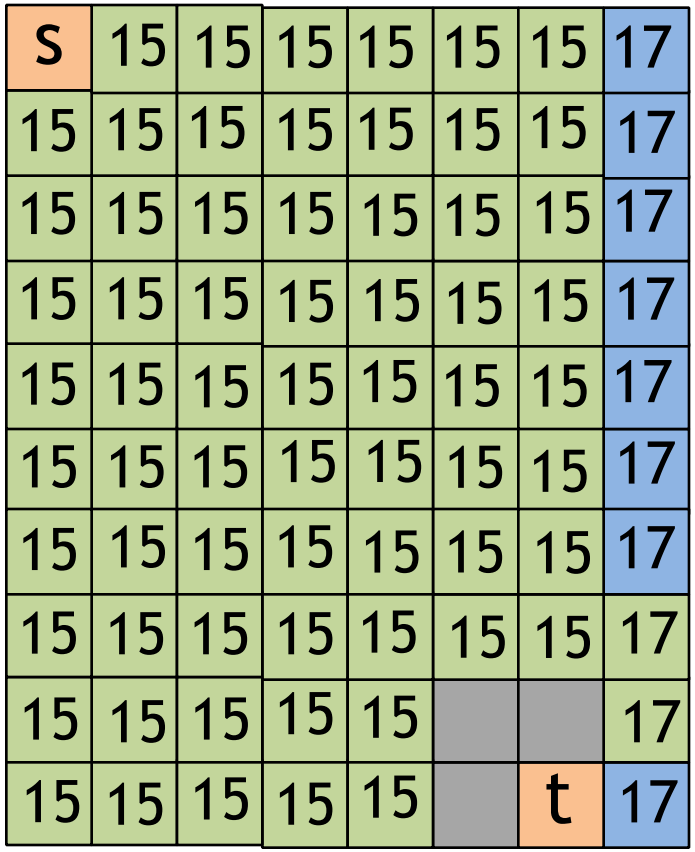
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

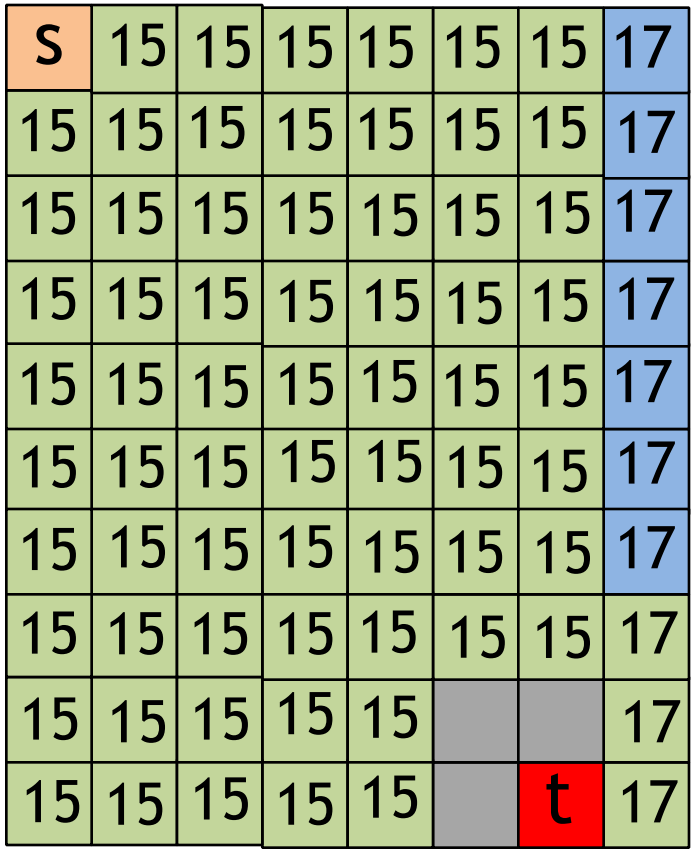
Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

Another Ruben Example



MD(s, t) = 15

Shortest s-t path is 17 (due to blockages)

Korn's Cost Function

Korn, R. K. (1982), *An Efficient Variable Cost Maze Router*, Proceedings of the Design Automation Conference (DAC), 425-431

Korn's Approach

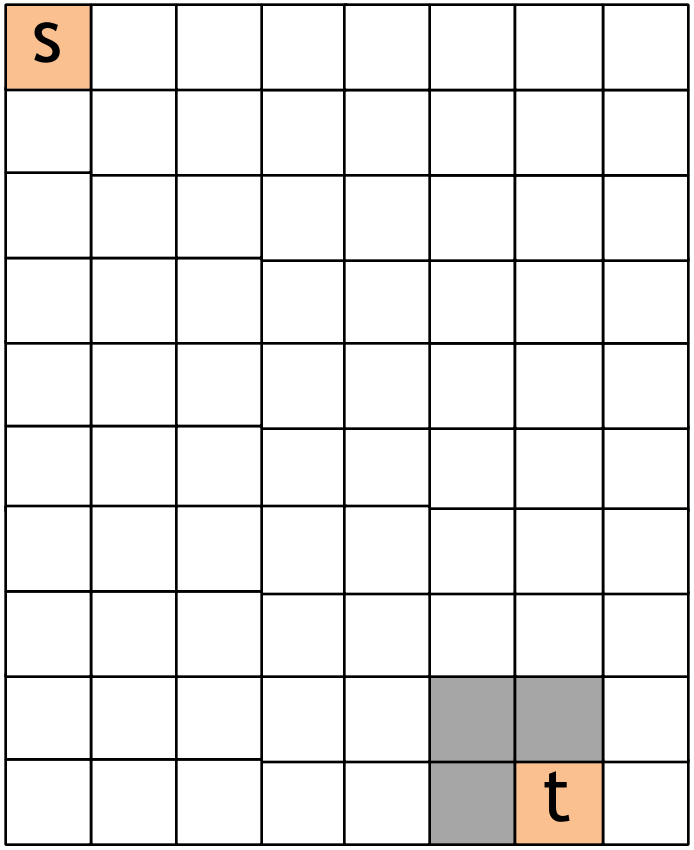
- Heavily weight MD(i, t) term over D(s, i) term
- Sacrifice guarantee of optimality

$$C(i) = D(s, i) + A \bullet MD(i, t)$$

Overpull ($A \geq 1$)

- General problem with parameterization:
  - What is a good/bad Overpull value?

Korn Example (50% Overpull)

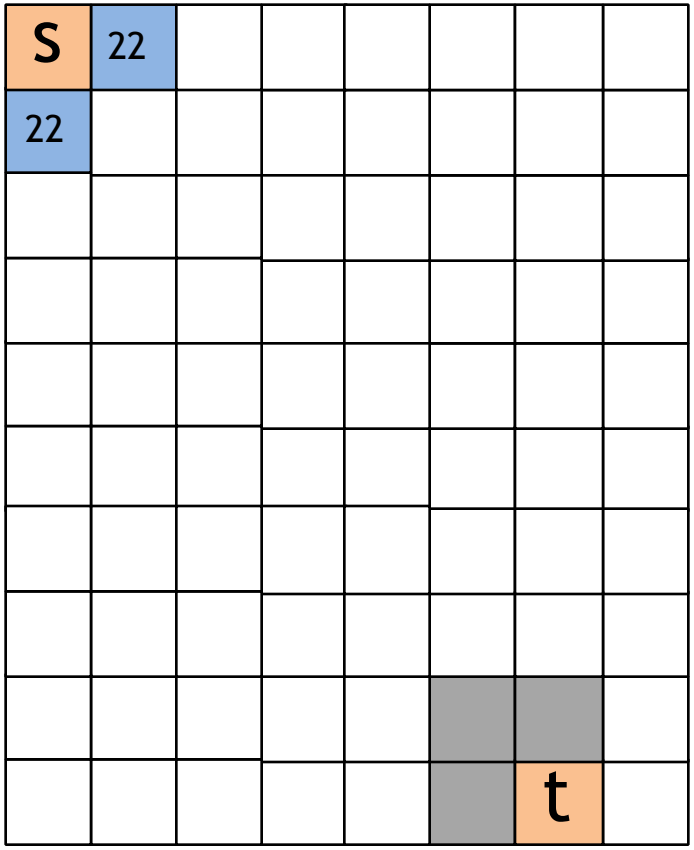


1.5•MD(…) values

S	21	19.5	18	16.5	15	13.5	15
21	19.5	18	16.5	15	13.5	12	13.5
19.5	18	16.5	15	13.5	12	10.5	12
18	16.5	15	13.5	12	10.5	9	10.5
16.5	15	13.5	12	10.5	9	7.5	9
15	13.5	12	10.5	9	7.5	6	7.5
13.5	12	10.5	9	7.5	6	4.5	6
12	10.5	9	7.5	6	4.5	3	4.5
10.5	9	7.5	6	4.5			3
9	7.5	6	4.5	3		t	1.5

Korn Example (50% Overpull)

1+21



1.5•MD(…) values

S	21	19.5	18	16.5	15	13.5	15
21	19.5	18	16.5	15	13.5	12	13.5
19.5	18	16.5	15	13.5	12	10.5	12
18	16.5	15	13.5	12	10.5	9	10.5
16.5	15	13.5	12	10.5	9	7.5	9
15	13.5	12	10.5	9	7.5	6	7.5
13.5	12	10.5	9	7.5	6	4.5	6
12	10.5	9	7.5	6	4.5	3	4.5
10.5	9	7.5	6	4.5			3
9	7.5	6	4.5	3		t	1.5

Korn Example (50% Overpull)

2+19.5

S	22	21.5					
22	21.5						
						t	

1.5•MD(…) values

S	21	19.5	18	16.5	15	13.5	15
21	19.5	18	16.5	15	13.5	12	13.5
19.5	18	16.5	15	13.5	12	10.5	12
18	16.5	15	13.5	12	10.5	9	10.5
16.5	15	13.5	12	10.5	9	7.5	9
15	13.5	12	10.5	9	7.5	6	7.5
13.5	12	10.5	9	7.5	6	4.5	6
12	10.5	9	7.5	6	4.5	3	4.5
10.5	9	7.5	6	4.5			3
9	7.5	6	4.5	3		t	1.5

Korn Example (50% Overpull)

3+18

S	22	21.5	21				
22	21.5	21					
						t	

1.5•MD(…) values

S	21	19.5	18	16.5	15	13.5	15
21	19.5	18	16.5	15	13.5	12	13.5
19.5	18	16.5	15	13.5	12	10.5	12
18	16.5	15	13.5	12	10.5	9	10.5
16.5	15	13.5	12	10.5	9	7.5	9
15	13.5	12	10.5	9	7.5	6	7.5
13.5	12	10.5	9	7.5	6	4.5	6
12	10.5	9	7.5	6	4.5	3	4.5
10.5	9	7.5	6	4.5			3
9	7.5	6	4.5	3		t	1.5

Korn Example (50% Overpull)

4+16.5

S	22	21.5	21	20.5			
22	21.5	21	20.5				
					t		

1.5•MD(…) values

S	21	19.5	18	16.5	15	13.5	15
21	19.5	18	16.5	15	13.5	12	13.5
19.5	18	16.5	15	13.5	12	10.5	12
18	16.5	15	13.5	12	10.5	9	10.5
16.5	15	13.5	12	10.5	9	7.5	9
15	13.5	12	10.5	9	7.5	6	7.5
13.5	12	10.5	9	7.5	6	4.5	6
12	10.5	9	7.5	6	4.5	3	4.5
10.5	9	7.5	6	4.5			3
9	7.5	6	4.5	3		t	1.5

Korn Example (50% Overpull)

5+15

S	22	21.5	21	20.5	20		
22	21.5	21	20.5	20			
						t	

1.5•MD(…) values

S	21	19.5	18	16.5	15	13.5	15
21	19.5	18	16.5	15	13.5	12	13.5
19.5	18	16.5	15	13.5	12	10.5	12
18	16.5	15	13.5	12	10.5	9	10.5
16.5	15	13.5	12	10.5	9	7.5	9
15	13.5	12	10.5	9	7.5	6	7.5
13.5	12	10.5	9	7.5	6	4.5	6
12	10.5	9	7.5	6	4.5	3	4.5
10.5	9	7.5	6	4.5			3
9	7.5	6	4.5	3		t	1.5

Korn Example (50% Overpull)

6+13.5

S	22	21.5	21	20.5	20	19.5	
22	21.5	21	20.5	20	19.5		
						t	

1.5•MD(…) values

S	21	19.5	18	16.5	15	13.5	15
21	19.5	18	16.5	15	13.5	12	13.5
19.5	18	16.5	15	13.5	12	10.5	12
18	16.5	15	13.5	12	10.5	9	10.5
16.5	15	13.5	12	10.5	9	7.5	9
15	13.5	12	10.5	9	7.5	6	7.5
13.5	12	10.5	9	7.5	6	4.5	6
12	10.5	9	7.5	6	4.5	3	4.5
10.5	9	7.5	6	4.5			3
9	7.5	6	4.5	3		t	1.5

Korn Example (50% Overpull)

7+15

S	22	21.5	21	20.5	20	19.5	22
22	21.5	21	20.5	20	19.5	19	
						7+12	
						t	

1.5•MD(…) values

S	21	19.5	18	16.5	15	13.5	15
21	19.5	18	16.5	15	13.5	12	13.5
19.5	18	16.5	15	13.5	12	10.5	12
18	16.5	15	13.5	12	10.5	9	10.5
16.5	15	13.5	12	10.5	9	7.5	9
15	13.5	12	10.5	9	7.5	6	7.5
13.5	12	10.5	9	7.5	6	4.5	6
12	10.5	9	7.5	6	4.5	3	4.5
10.5	9	7.5	6	4.5			3
9	7.5	6	4.5	3		t	1.5

Korn Example (50% Overpull)

S	22	21.5	21	20.5	20	19.5	22
22	21.5	21	20.5	20	19.5	19	21.5
						18.5	
						t	

1.5•MD(…) values

S	21	19.5	18	16.5	15	13.5	15
21	19.5	18	16.5	15	13.5	12	13.5
19.5	18	16.5	15	13.5	12	10.5	12
18	16.5	15	13.5	12	10.5	9	10.5
16.5	15	13.5	12	10.5	9	7.5	9
15	13.5	12	10.5	9	7.5	6	7.5
13.5	12	10.5	9	7.5	6	4.5	6
12	10.5	9	7.5	6	4.5	3	4.5
10.5	9	7.5	6	4.5			3
9	7.5	6	4.5	3		t	1.5

Korn Example (50% Overpull)

S	22	21.5	21	20.5	20	19.5	22
22	21.5	21	20.5	20	19.5	19	21.5
					21	18.5	21
						18	
						t	

1.5•MD(…) values

S	21	19.5	18	16.5	15	13.5	15
21	19.5	18	16.5	15	13.5	12	13.5
19.5	18	16.5	15	13.5	12	10.5	12
18	16.5	15	13.5	12	10.5	9	10.5
16.5	15	13.5	12	10.5	9	7.5	9
15	13.5	12	10.5	9	7.5	6	7.5
13.5	12	10.5	9	7.5	6	4.5	6
12	10.5	9	7.5	6	4.5	3	4.5
10.5	9	7.5	6	4.5			3
9	7.5	6	4.5	3		t	1.5

Korn Example (50% Overpull)

S	22	21.5	21	20.5	20	19.5	22
22	21.5	21	20.5	20	19.5	19	21.5
					21	18.5	21
					20.5	18	20.5
						17.5	
						t	

1.5•MD(…) values

S	21	19.5	18	16.5	15	13.5	15
21	19.5	18	16.5	15	13.5	12	13.5
19.5	18	16.5	15	13.5	12	10.5	12
18	16.5	15	13.5	12	10.5	9	10.5
16.5	15	13.5	12	10.5	9	7.5	9
15	13.5	12	10.5	9	7.5	6	7.5
13.5	12	10.5	9	7.5	6	4.5	6
12	10.5	9	7.5	6	4.5	3	4.5
10.5	9	7.5	6	4.5			3
9	7.5	6	4.5	3		t	1.5

Korn Example (50% Overpull)

S	22	21.5	21	20.5	20	19.5	22
22	21.5	21	20.5	20	19.5	19	21.5
					21	18.5	21
					20.5	18	20.5
					20	17.5	20
						17	
						t	

1.5•MD(…) values

S	21	19.5	18	16.5	15	13.5	15
21	19.5	18	16.5	15	13.5	12	13.5
19.5	18	16.5	15	13.5	12	10.5	12
18	16.5	15	13.5	12	10.5	9	10.5
16.5	15	13.5	12	10.5	9	7.5	9
15	13.5	12	10.5	9	7.5	6	7.5
13.5	12	10.5	9	7.5	6	4.5	6
12	10.5	9	7.5	6	4.5	3	4.5
10.5	9	7.5	6	4.5			3
9	7.5	6	4.5	3		t	1.5

Korn Example (50% Overpull)

S	22	21.5	21	20.5	20	19.5	22
22	21.5	21	20.5	20	19.5	19	21.5
					21	18.5	21
					20.5	18	20.5
					20	17.5	20
					19.5	17	19.5
						16.5	
						t	

1.5•MD(…) values

S	21	19.5	18	16.5	15	13.5	15
21	19.5	18	16.5	15	13.5	12	13.5
19.5	18	16.5	15	13.5	12	10.5	12
18	16.5	15	13.5	12	10.5	9	10.5
16.5	15	13.5	12	10.5	9	7.5	9
15	13.5	12	10.5	9	7.5	6	7.5
13.5	12	10.5	9	7.5	6	4.5	6
12	10.5	9	7.5	6	4.5	3	4.5
10.5	9	7.5	6	4.5			3
9	7.5	6	4.5	3		t	1.5

Korn Example (50% Overpull)

S	22	21.5	21	20.5	20	19.5	22
22	21.5	21	20.5	20	19.5	19	21.5
					21	18.5	21
					20.5	18	20.5
					20	17.5	20
					19.5	17	19.5
					19	16.5	19
						16	
						t	

1.5•MD(…) values

S	21	19.5	18	16.5	15	13.5	15
21	19.5	18	16.5	15	13.5	12	13.5
19.5	18	16.5	15	13.5	12	10.5	12
18	16.5	15	13.5	12	10.5	9	10.5
16.5	15	13.5	12	10.5	9	7.5	9
15	13.5	12	10.5	9	7.5	6	7.5
13.5	12	10.5	9	7.5	6	4.5	6
12	10.5	9	7.5	6	4.5	3	4.5
10.5	9	7.5	6	4.5			3
9	7.5	6	4.5	3		t	1.5

Korn Example (50% Overpull)

S	22	21.5	21	20.5	20	19.5	22
22	21.5	21	20.5	20	19.5	19	21.5
					21	18.5	21
					20.5	18	20.5
					20	17.5	20
					19.5	17	19.5
					19	16.5	19
					18.5	16	18.5
						t	

1.5•MD(…) values

S	21	19.5	18	16.5	15	13.5	15
21	19.5	18	16.5	15	13.5	12	13.5
19.5	18	16.5	15	13.5	12	10.5	12
18	16.5	15	13.5	12	10.5	9	10.5
16.5	15	13.5	12	10.5	9	7.5	9
15	13.5	12	10.5	9	7.5	6	7.5
13.5	12	10.5	9	7.5	6	4.5	6
12	10.5	9	7.5	6	4.5	3	4.5
10.5	9	7.5	6	4.5			3
9	7.5	6	4.5	3		t	1.5

Korn Example (50% Overpull)

S	22	21.5	21	20.5	20	19.5	22
22	21.5	21	20.5	20	19.5	19	21.5
					21	18.5	21
					20.5	18	20.5
					20	17.5	20
					19.5	17	19.5
					19	16.5	19
				21	18.5	16	18.5
						t	

1.5•MD(…) values

S	21	19.5	18	16.5	15	13.5	15
21	19.5	18	16.5	15	13.5	12	13.5
19.5	18	16.5	15	13.5	12	10.5	12
18	16.5	15	13.5	12	10.5	9	10.5
16.5	15	13.5	12	10.5	9	7.5	9
15	13.5	12	10.5	9	7.5	6	7.5
13.5	12	10.5	9	7.5	6	4.5	6
12	10.5	9	7.5	6	4.5	3	4.5
10.5	9	7.5	6	4.5			3
9	7.5	6	4.5	3		t	1.5

Korn Example (50% Overpull)

S	22	21.5	21	20.5	20	19.5	22
22	21.5	21	20.5	20	19.5	19	21.5
					21	18.5	21
					20.5	18	20.5
					20	17.5	20
					19.5	17	19.5
					19	16.5	19
				21	18.5	16	18.5
							18
						t	

1.5•MD(...) values

S	21	19.5	18	16.5	15	13.5	15
21	19.5	18	16.5	15	13.5	12	13.5
19.5	18	16.5	15	13.5	12	10.5	12
18	16.5	15	13.5	12	10.5	9	10.5
16.5	15	13.5	12	10.5	9	7.5	9
15	13.5	12	10.5	9	7.5	6	7.5
13.5	12	10.5	9	7.5	6	4.5	6
12	10.5	9	7.5	6	4.5	3	4.5
10.5	9	7.5	6	4.5			3
9	7.5	6	4.5	3		t	1.5

Korn Example (50% Overpull)

S	22	21.5	21	20.5	20	19.5	22
22	21.5	21	20.5	20	19.5	19	21.5
					21	18.5	21
					20.5	18	20.5
					20	17.5	20
					19.5	17	19.5
					19	16.5	19
				21	18.5	16	18.5
							18
						t	17.5

1.5•MD(…) values

S	21	19.5	18	16.5	15	13.5	15
21	19.5	18	16.5	15	13.5	12	13.5
19.5	18	16.5	15	13.5	12	10.5	12
18	16.5	15	13.5	12	10.5	9	10.5
16.5	15	13.5	12	10.5	9	7.5	9
15	13.5	12	10.5	9	7.5	6	7.5
13.5	12	10.5	9	7.5	6	4.5	6
12	10.5	9	7.5	6	4.5	3	4.5
10.5	9	7.5	6	4.5			3
9	7.5	6	4.5	3		t	1.5

Korn Example (50% Overpull)

S	22	21.5	21	20.5	20	19.5	22
22	21.5	21	20.5	20	19.5	19	21.5
					21	18.5	21
					20.5	18	20.5
					20	17.5	20
					19.5	17	19.5
					19	16.5	19
				21	18.5	16	18.5
							18
						t	17.5

1.5•MD(…) values

S	21	19.5	18	16.5	15	13.5	15
21	19.5	18	16.5	15	13.5	12	13.5
19.5	18	16.5	15	13.5	12	10.5	12
18	16.5	15	13.5	12	10.5	9	10.5
16.5	15	13.5	12	10.5	9	7.5	9
15	13.5	12	10.5	9	7.5	6	7.5
13.5	12	10.5	9	7.5	6	4.5	6
12	10.5	9	7.5	6	4.5	3	4.5
10.5	9	7.5	6	4.5			3
9	7.5	6	4.5	3		t	1.5

Better than Ruben's Cost Function

S	22	21.5	21	20.5	20	19.5	22
22	21.5	21	20.5	20	19.5	19	21.5
					21	18.5	21
					20.5	18	20.5
					20	17.5	20
					19.5	17	19.5
					19	16.5	19
				21	18.5	16	18.5
							18
						t	17.5

1.5•MD(…) values

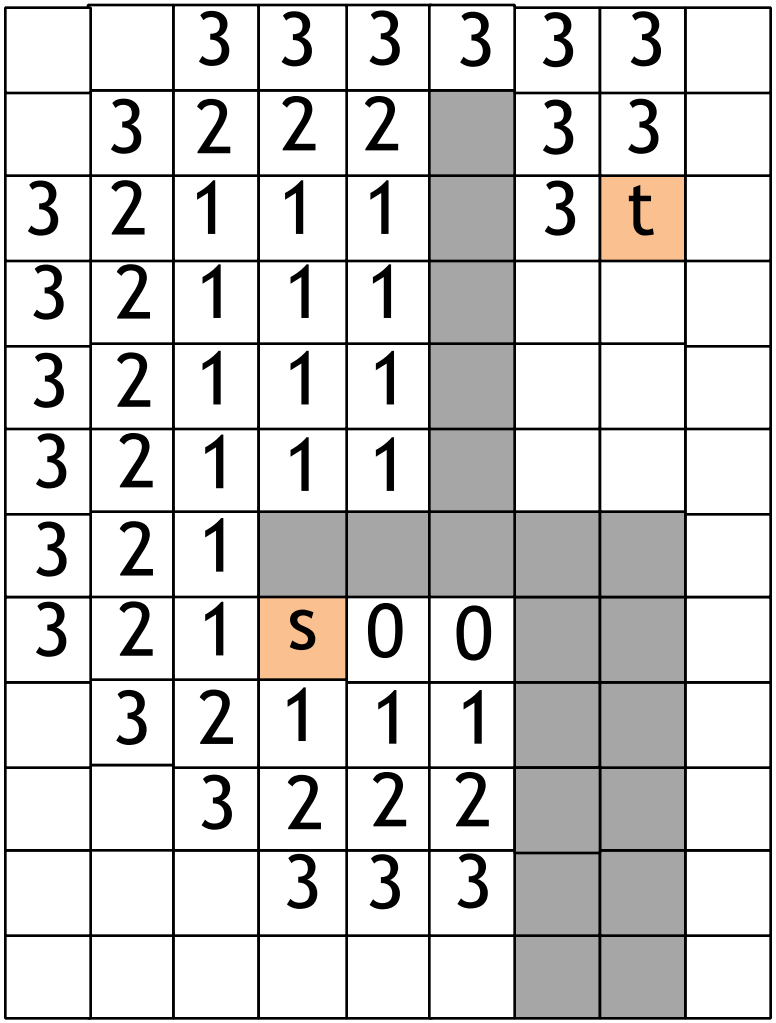
S	21	19.5	18	16.5	15	13.5	15
21	19.5	18	16.5	15	13.5	12	13.5
19.5	18	16.5	15	13.5	12	10.5	12
18	16.5	15	13.5	12	10.5	9	10.5
16.5	15	13.5	12	10.5	9	7.5	9
15	13.5	12	10.5	9	7.5	6	7.5
13.5	12	10.5	9	7.5	6	4.5	6
12	10.5	9	7.5	6	4.5	3	4.5
10.5	9	7.5	6	4.5			3
9	7.5	6	4.5	3		t	1.5

Hadlock's Algorithm

Hadlock, F. O. (1977), *A Shortest Path Algorithm for Grid Graphs*, Networks, 7(4): 323-334.

See: http://workbench.lafayette.edu/~nestorj/cadapplets/HadlockRouter/HadlockRouter.html

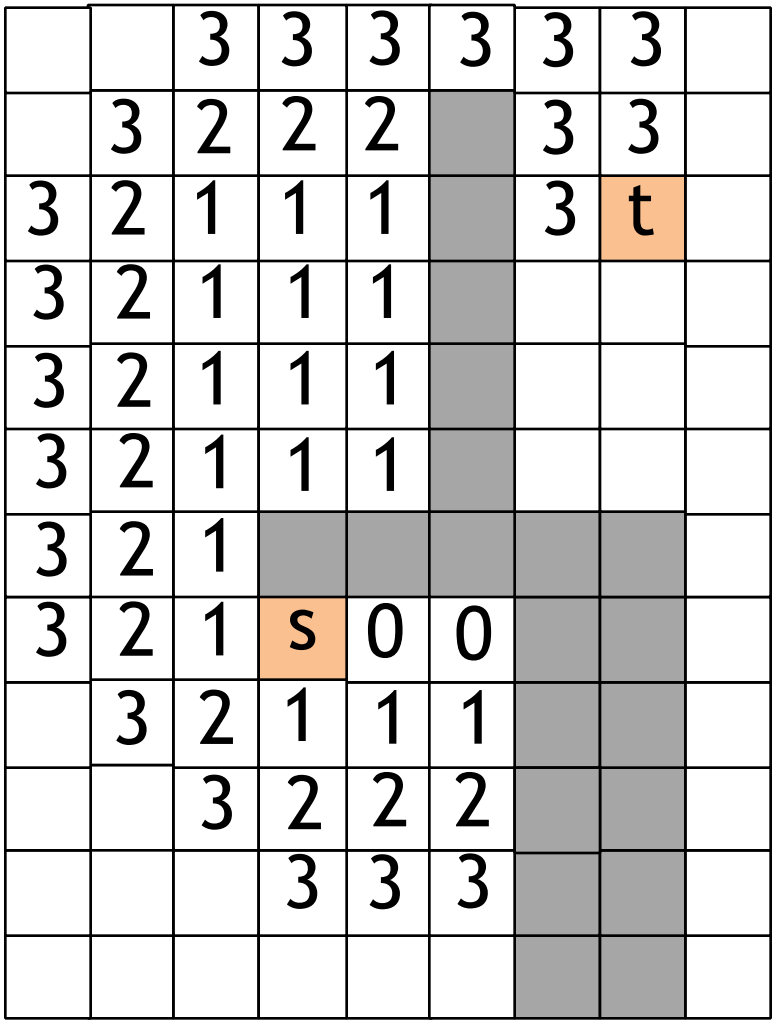
Detour Number d(P)



- Number of hops taken in the direction away from t on the most direct path from s to the current location

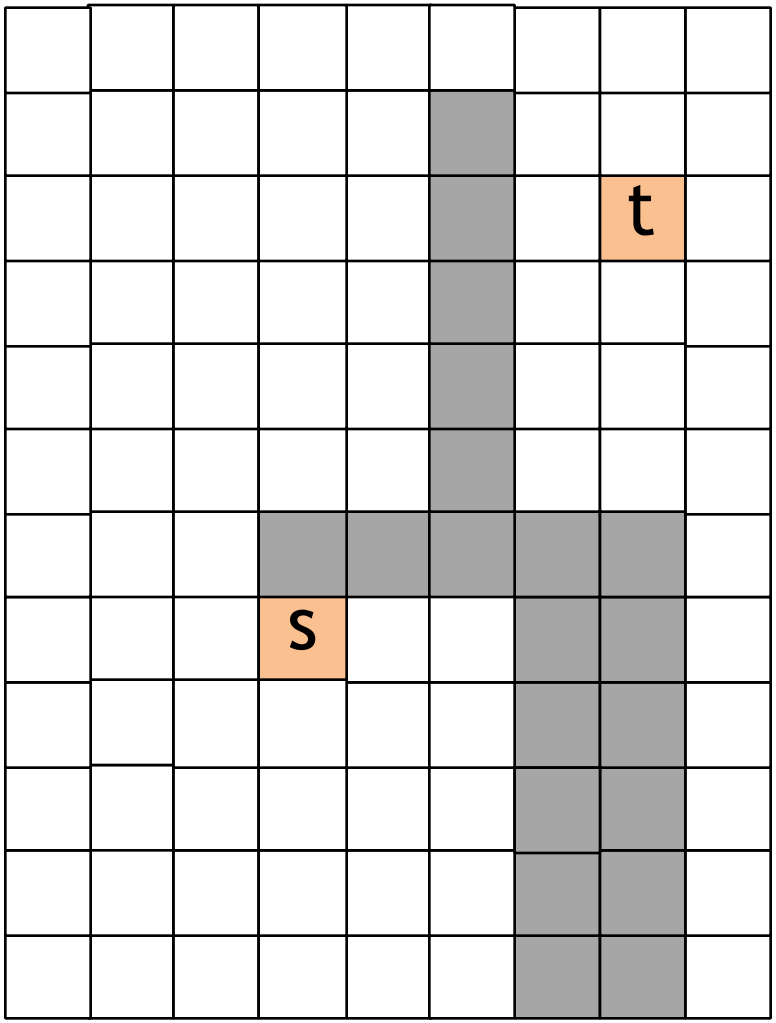
- Path length is MD(s, t) + 2d(P)
  - MD(s, t) is fixed
  - Find a path that minimizes d(P)

Search Process



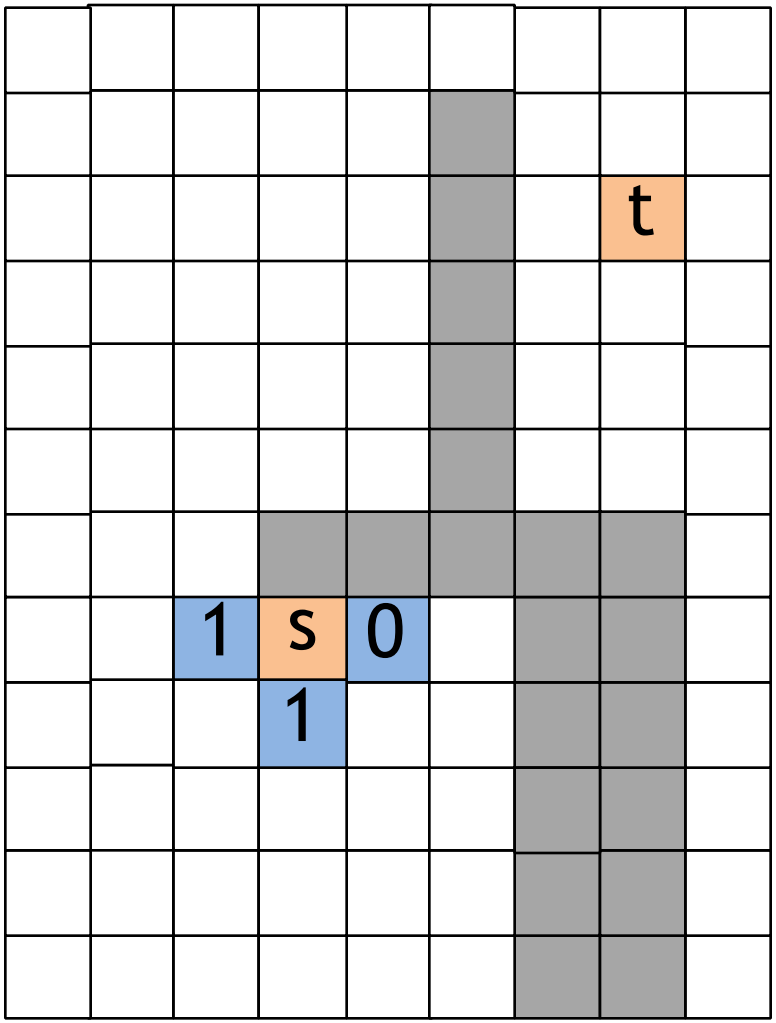
- Consider a cell with label J
  - Label each neighbor with either J or J+1, depending on the move
- Favor expansion of nodes with smaller labels

Example



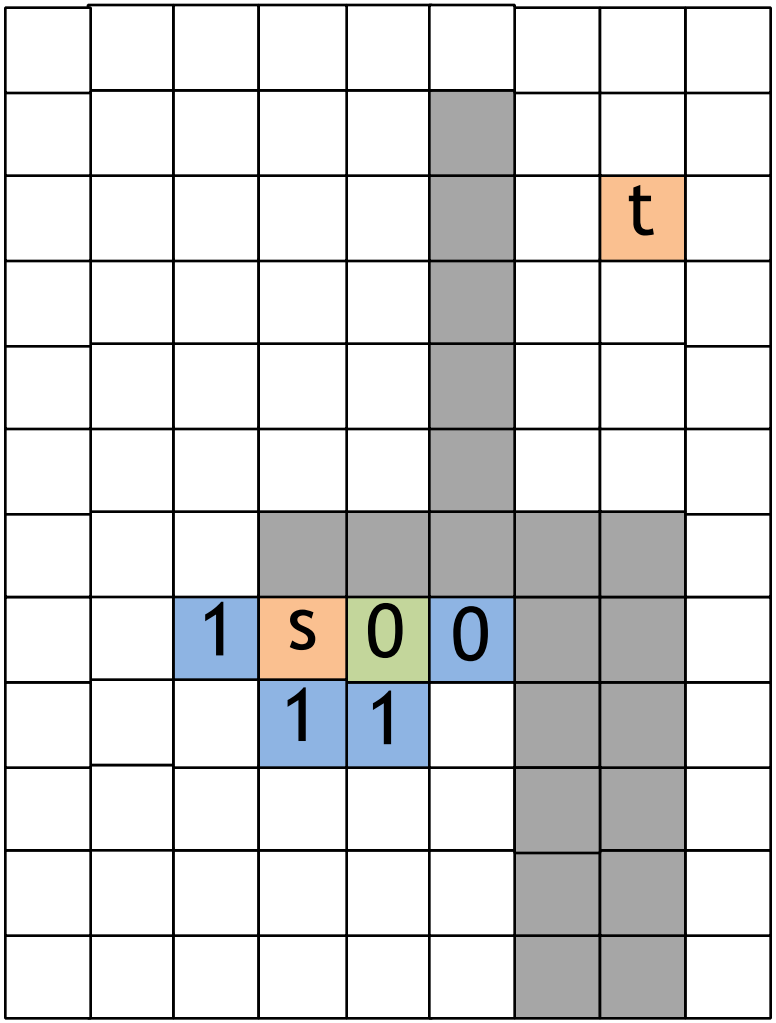
- Consider a cell with label J
  - Label each neighbor with either J or J+1, depending on the move
- Favor expansion of nodes with smaller labels

Example



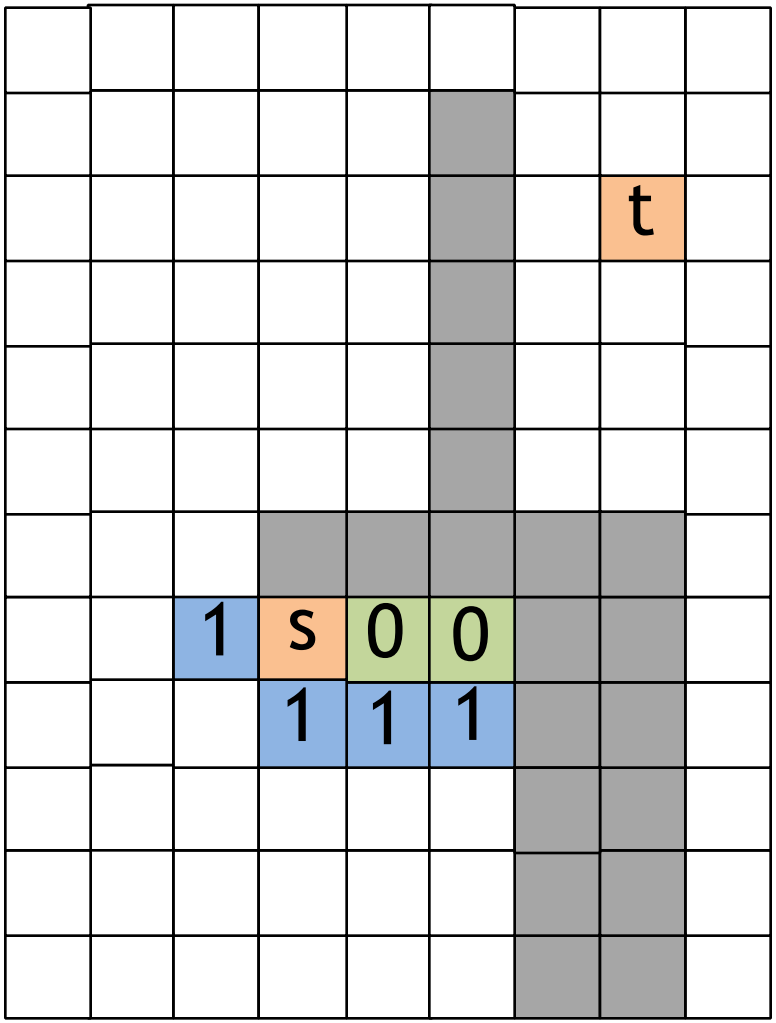
- Consider a cell with label J
  - Label each neighbor with either J or J+1, depending on the move
- Favor expansion of nodes with smaller labels

Example



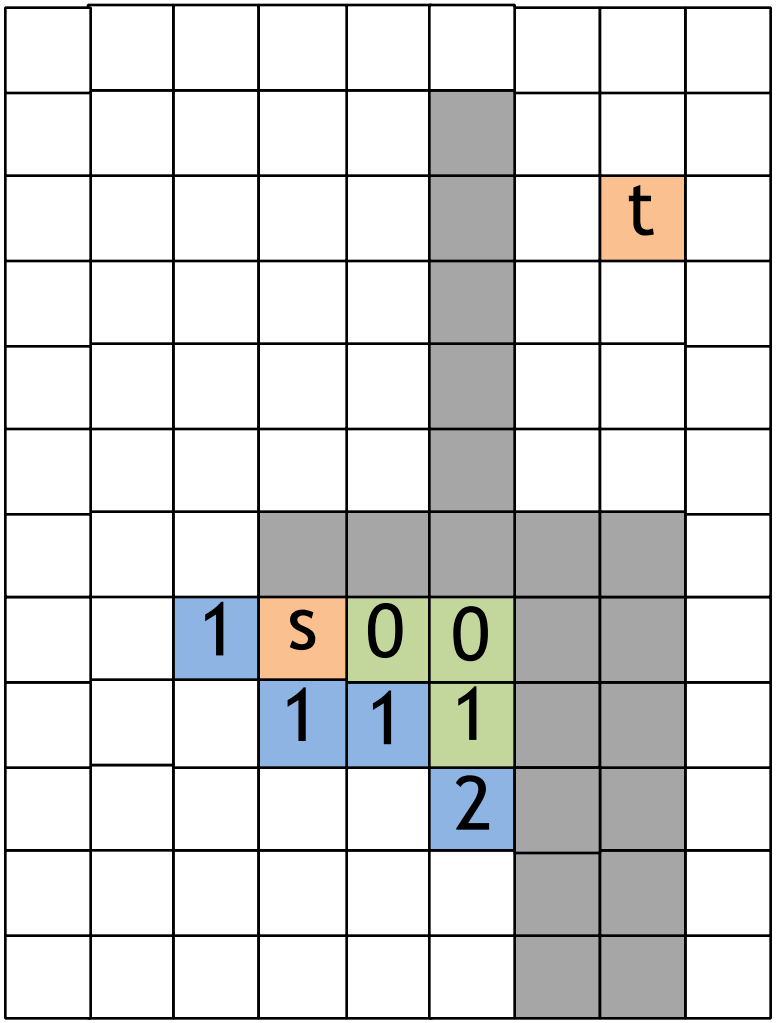
- Consider a cell with label J
  - Label each neighbor with either J or J+1, depending on the move
- Favor expansion of nodes with smaller labels

Example



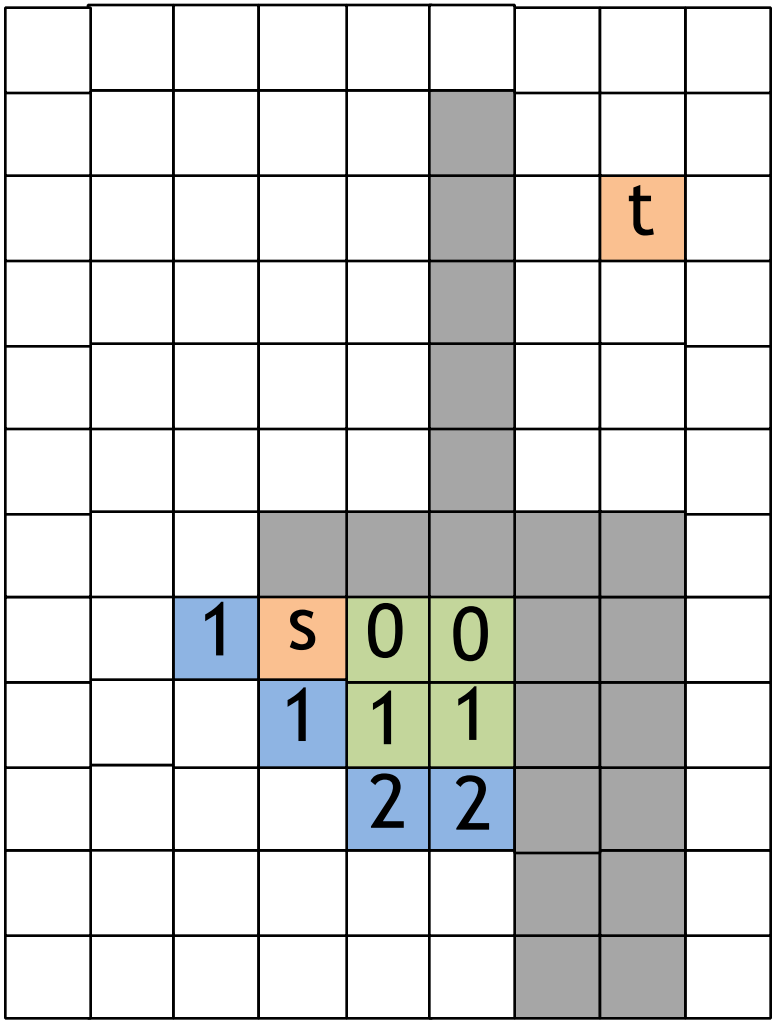
- Consider a cell with label J
  - Label each neighbor with either J or J+1, depending on the move
- Favor expansion of nodes with smaller labels

Example



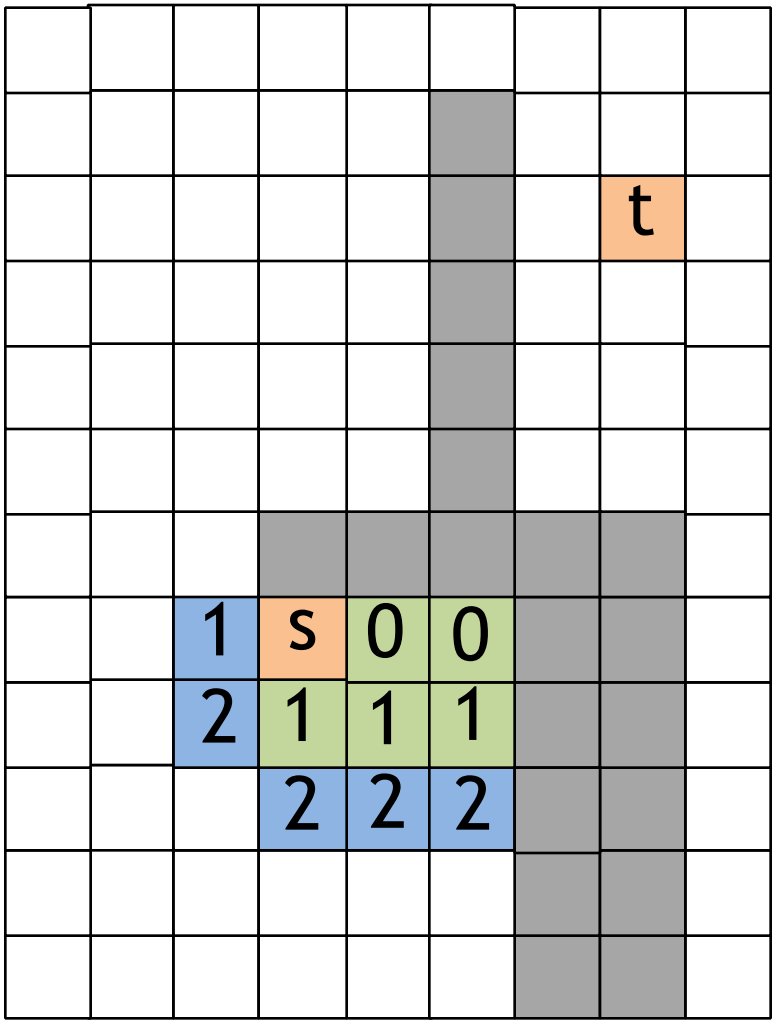
- Consider a cell with label J
  - Label each neighbor with either J or J+1, depending on the move
- Favor expansion of nodes with smaller labels

Example



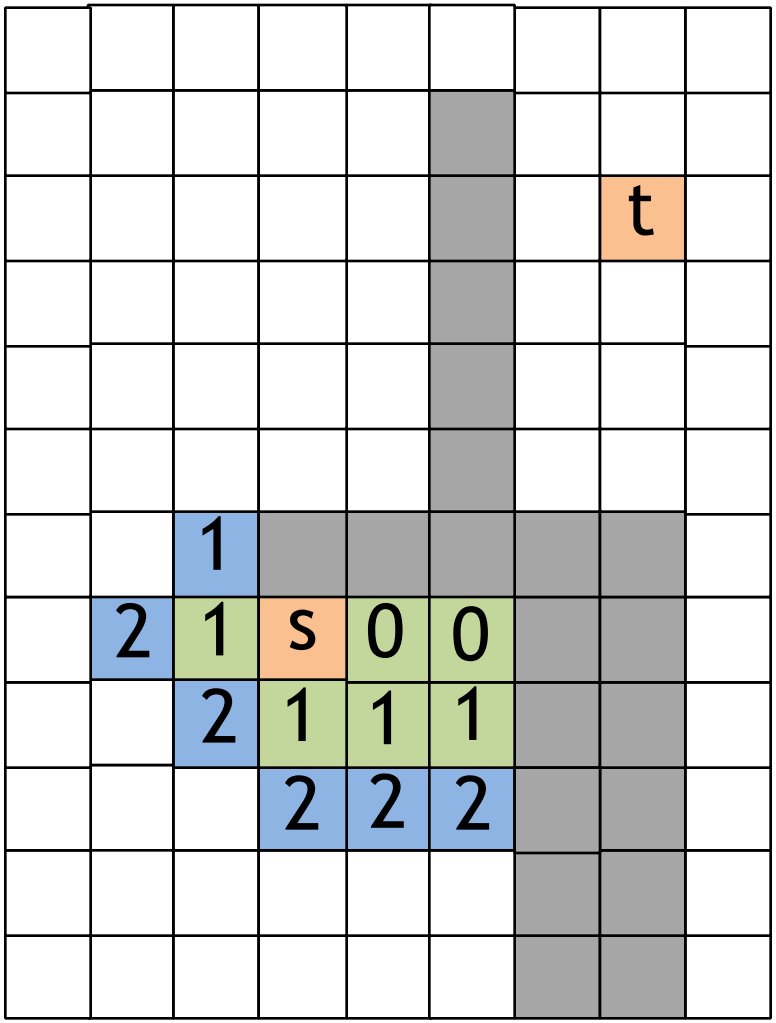
- Consider a cell with label J
  - Label each neighbor with either J or J+1, depending on the move
- Favor expansion of nodes with smaller labels

Example



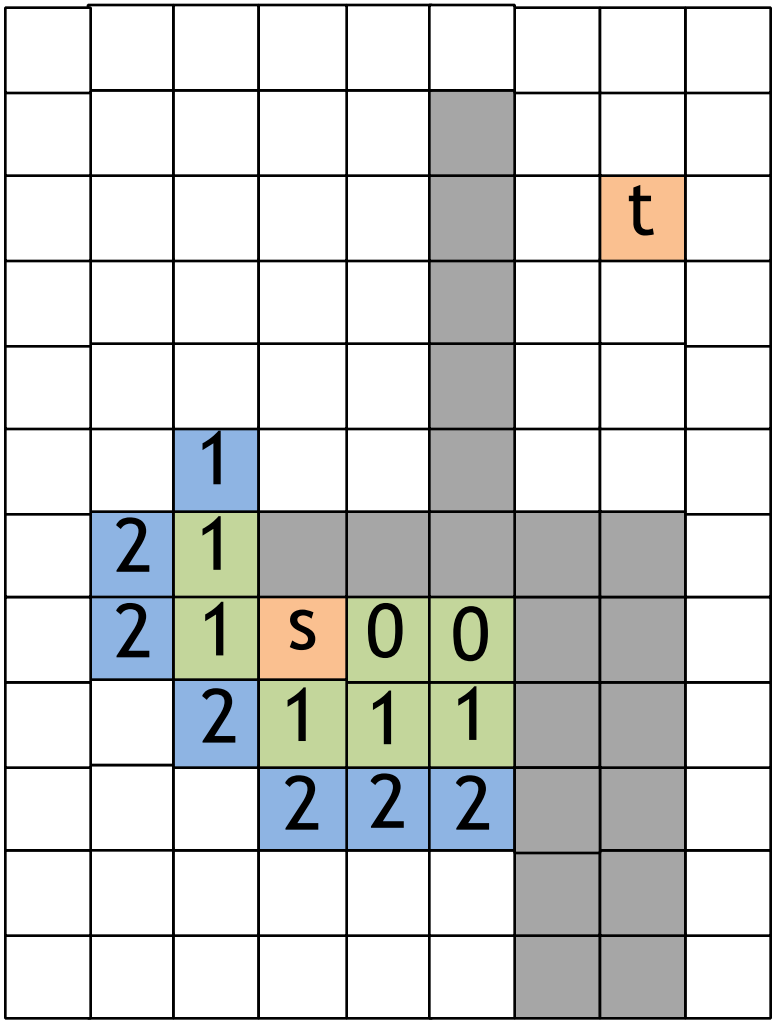
- Consider a cell with label J
  - Label each neighbor with either J or J+1, depending on the move
- Favor expansion of nodes with smaller labels

Example



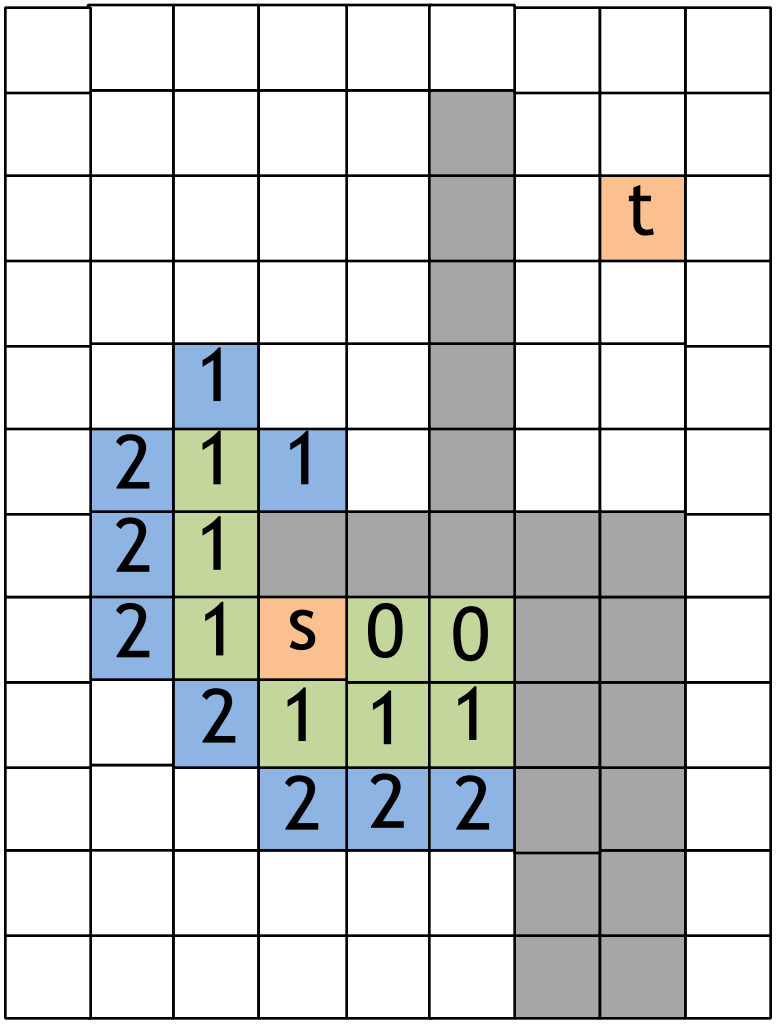
- Consider a cell with label J
  - Label each neighbor with either J or J+1, depending on the move
- Favor expansion of nodes with smaller labels

Example



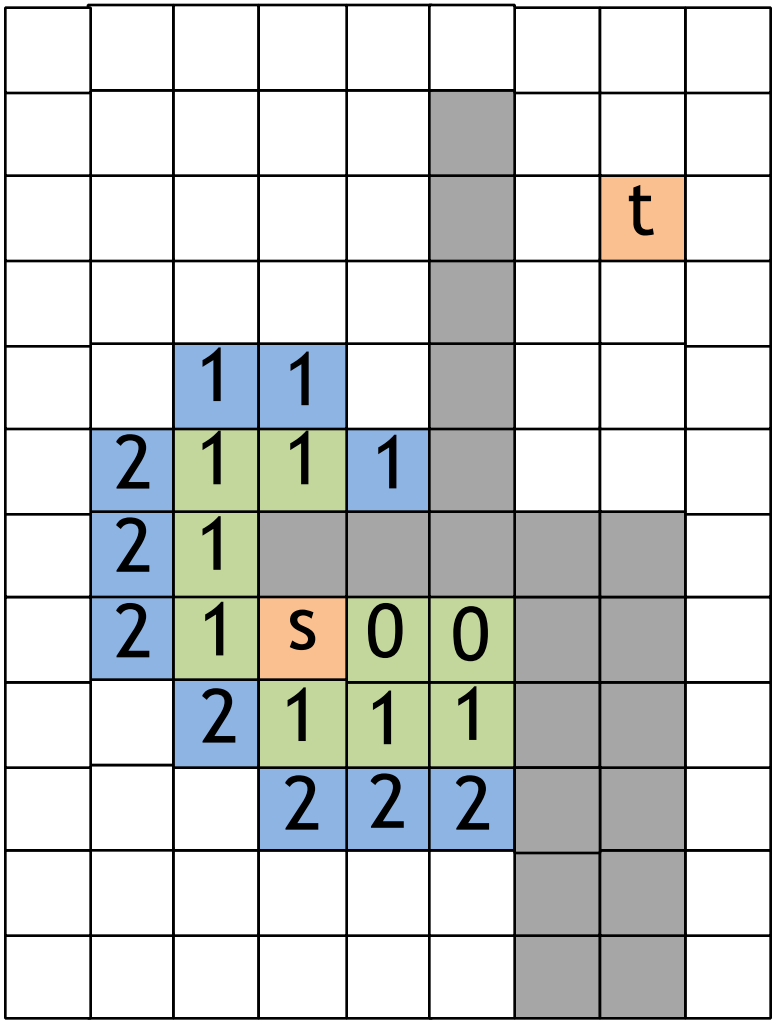
- Consider a cell with label J
  - Label each neighbor with either J or J+1, depending on the move
- Favor expansion of nodes with smaller labels

Example



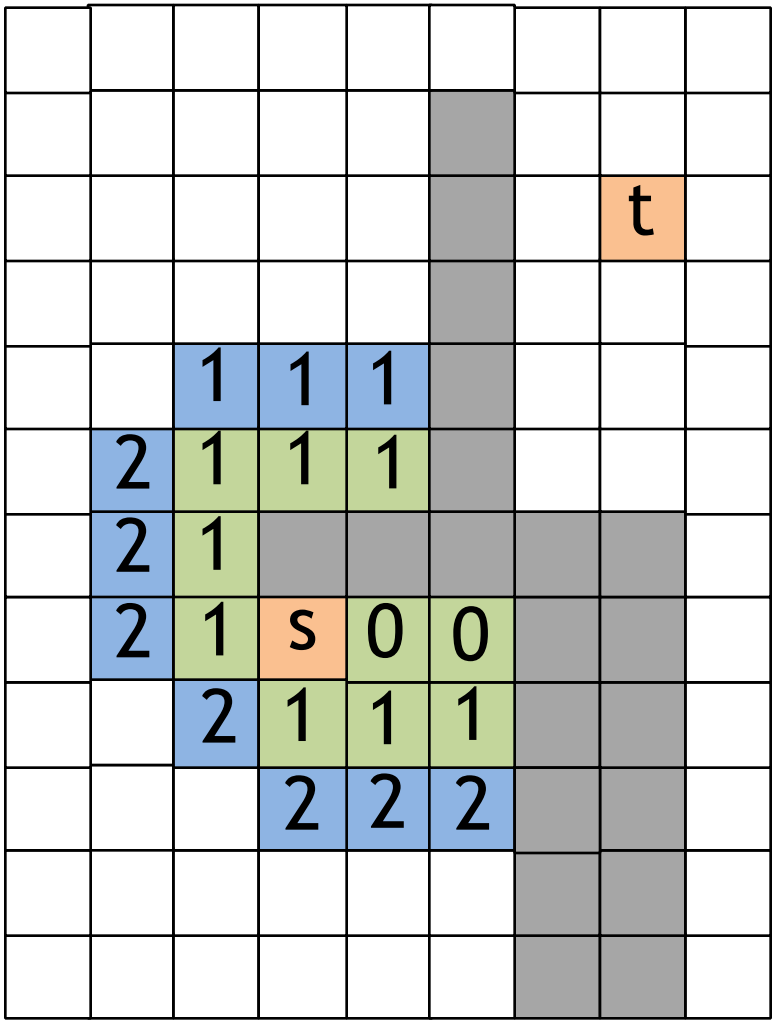
- Consider a cell with label J
  - Label each neighbor with either J or J+1, depending on the move
- Favor expansion of nodes with smaller labels

Example



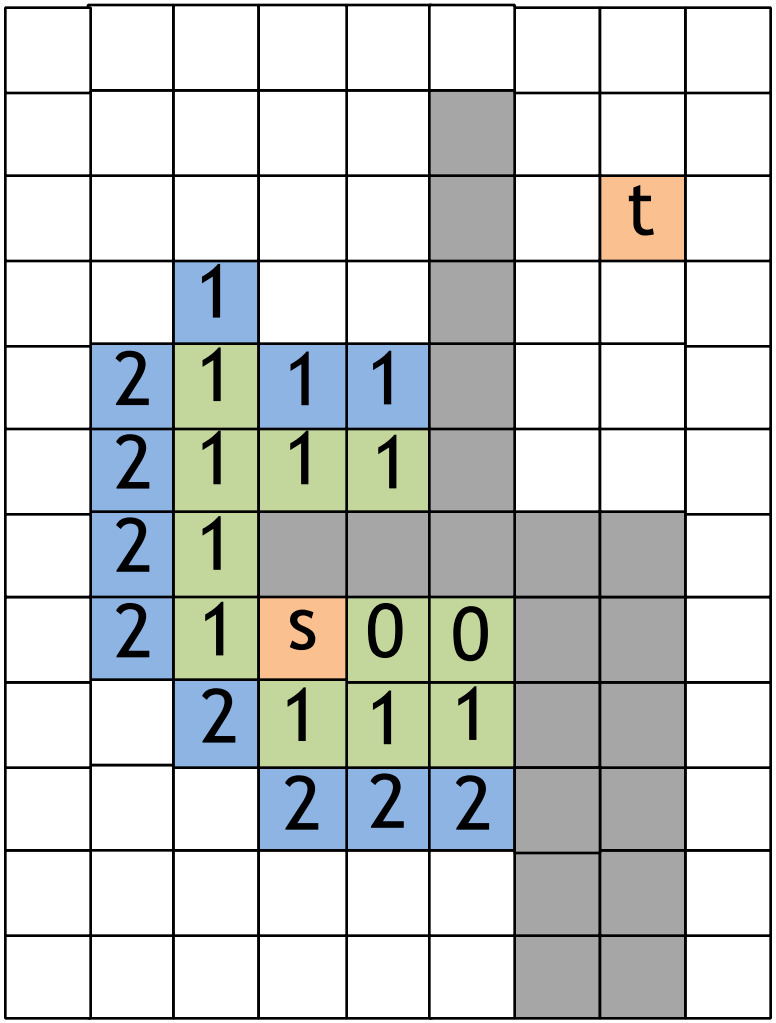
- Consider a cell with label J
  - Label each neighbor with either J or J+1, depending on the move
- Favor expansion of nodes with smaller labels

Example



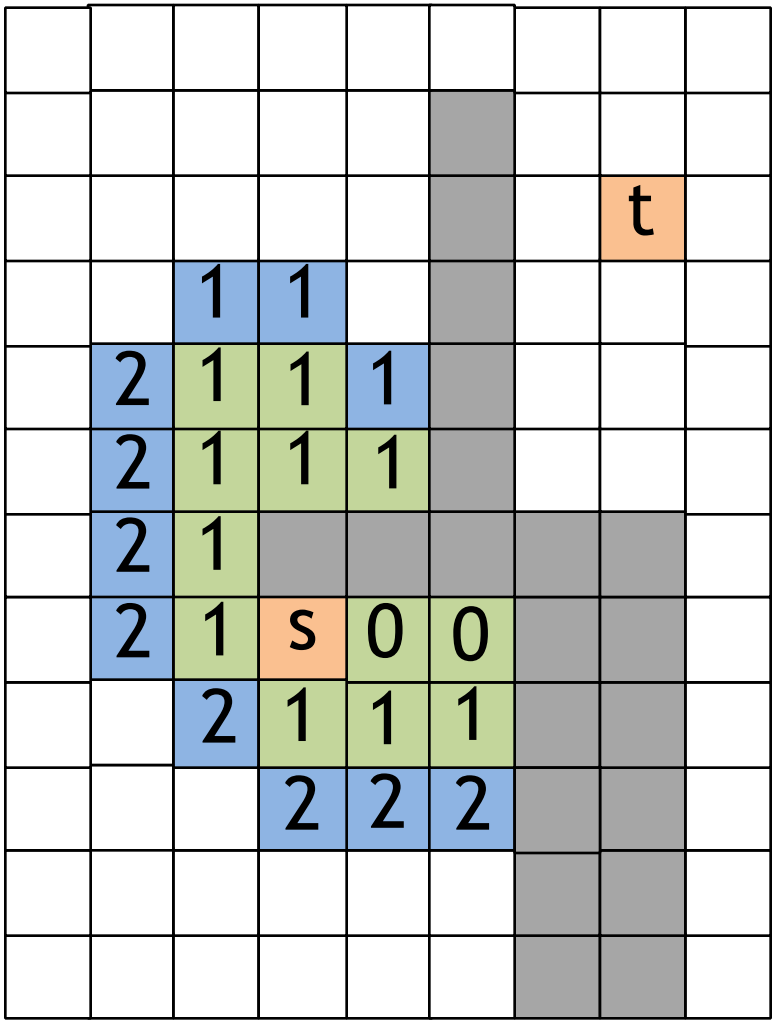
- Consider a cell with label J
  - Label each neighbor with either J or J+1, depending on the move
- Favor expansion of nodes with smaller labels

Example



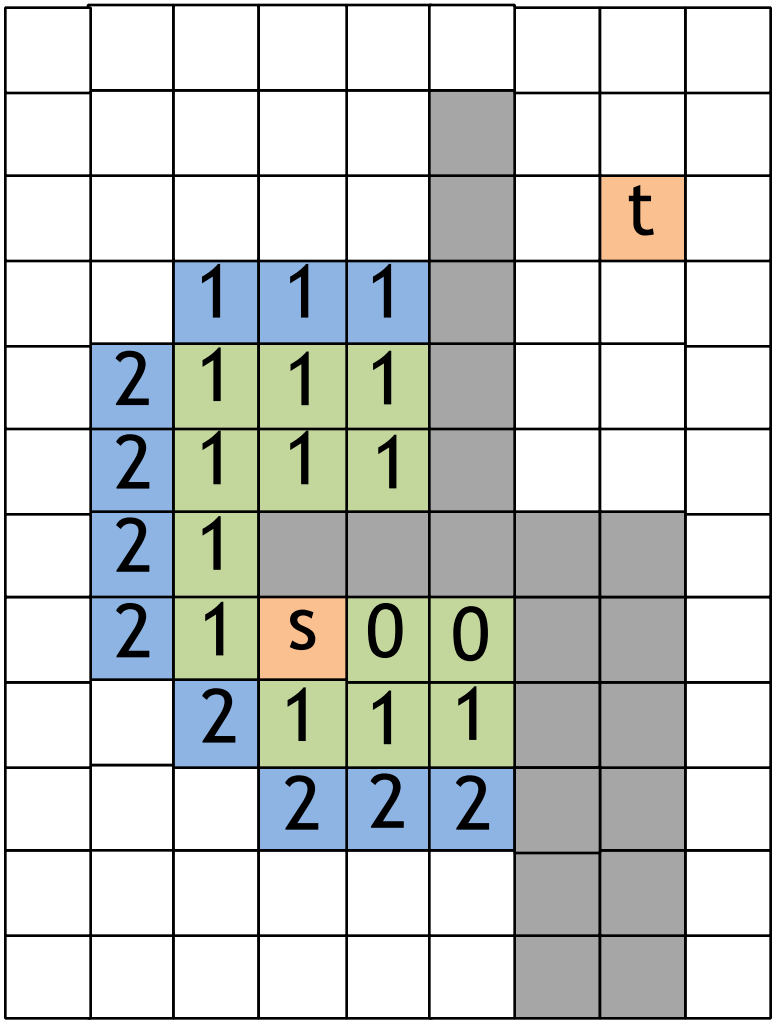
- Consider a cell with label J
  - Label each neighbor with either J or J+1, depending on the move
- Favor expansion of nodes with smaller labels

Example



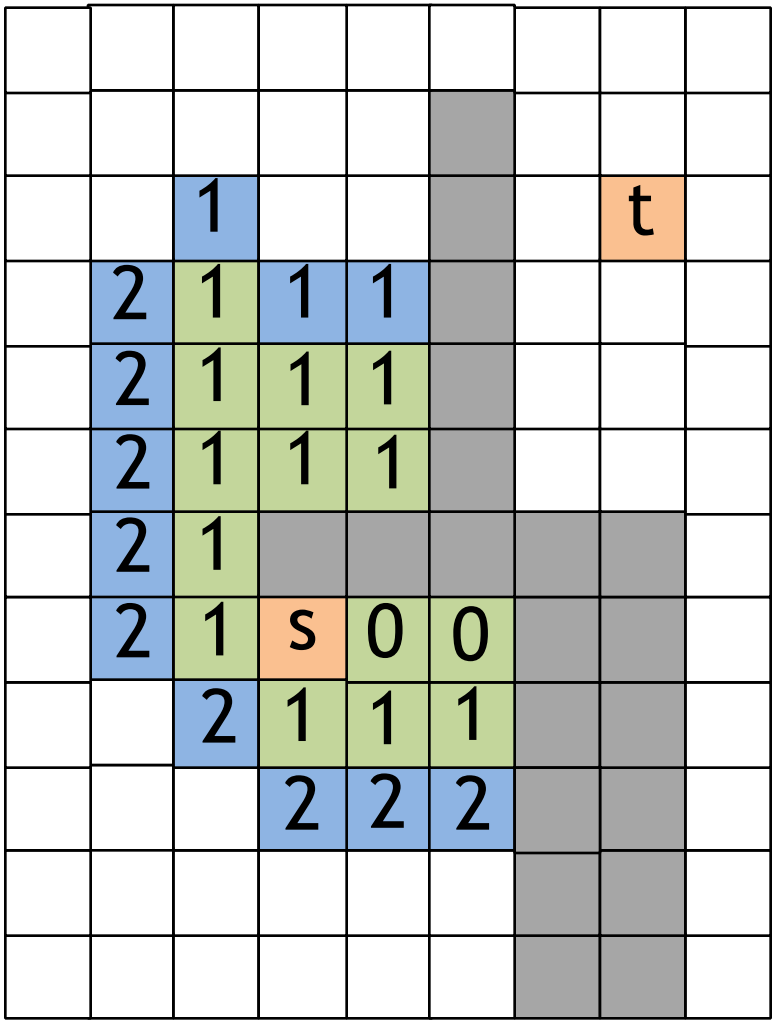
- Consider a cell with label J
  - Label each neighbor with either J or J+1, depending on the move
- Favor expansion of nodes with smaller labels

Example



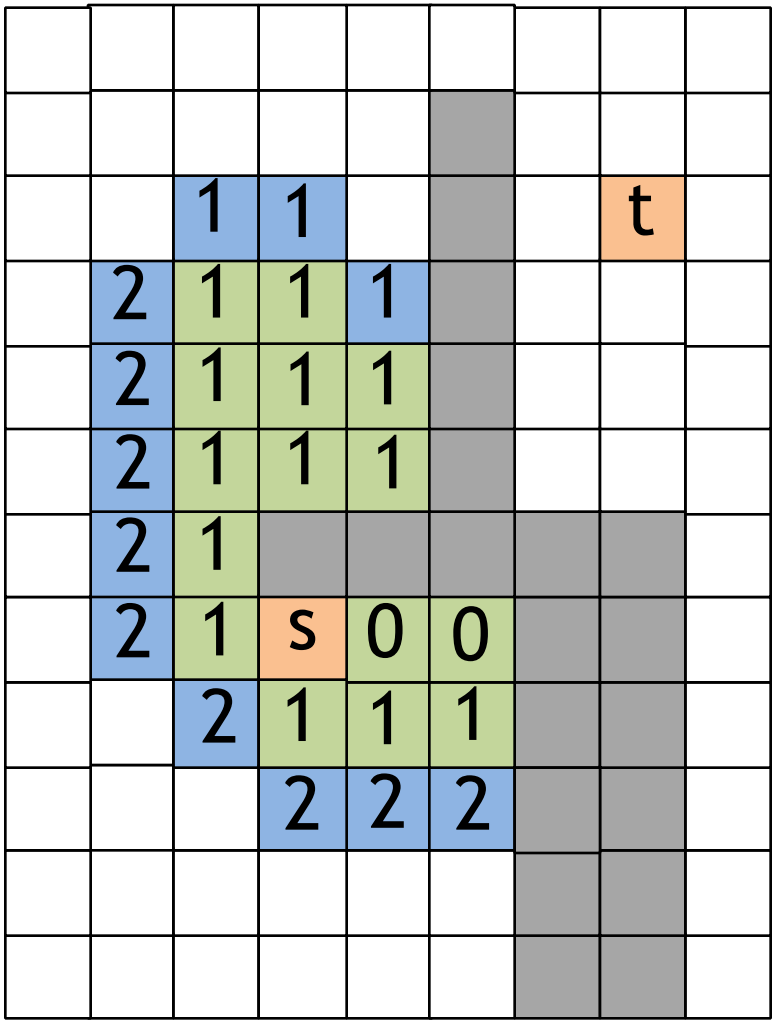
- Consider a cell with label J
  - Label each neighbor with either J or J+1, depending on the move
- Favor expansion of nodes with smaller labels

Example



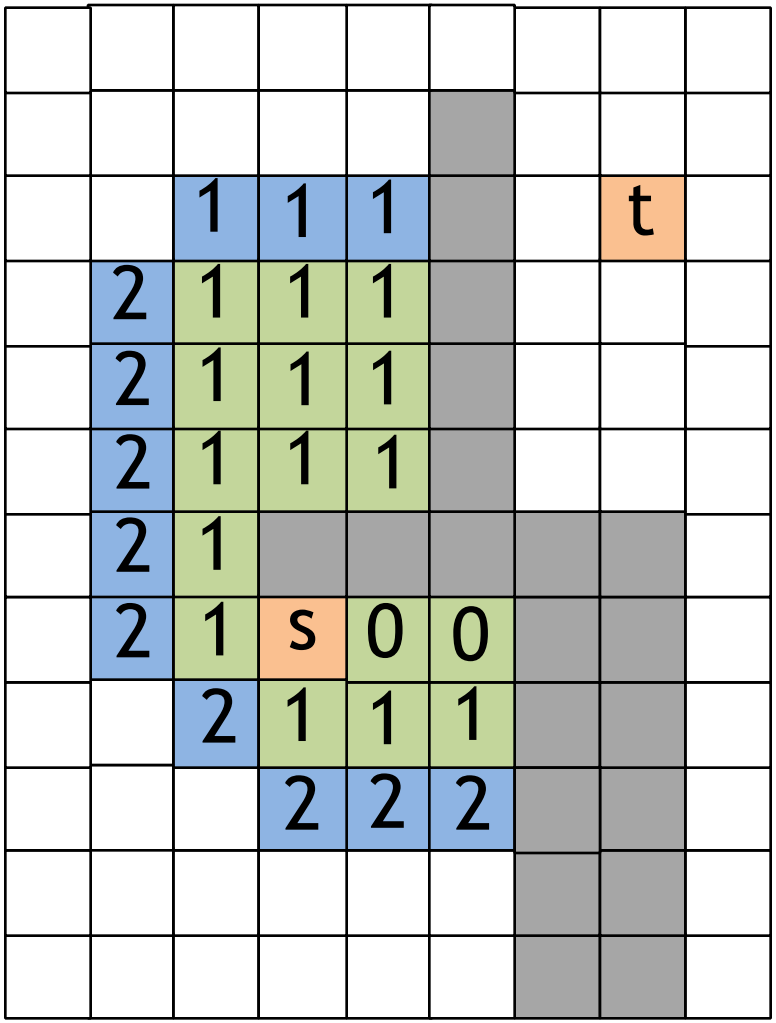
- Consider a cell with label J
  - Label each neighbor with either J or J+1, depending on the move
- Favor expansion of nodes with smaller labels

Example



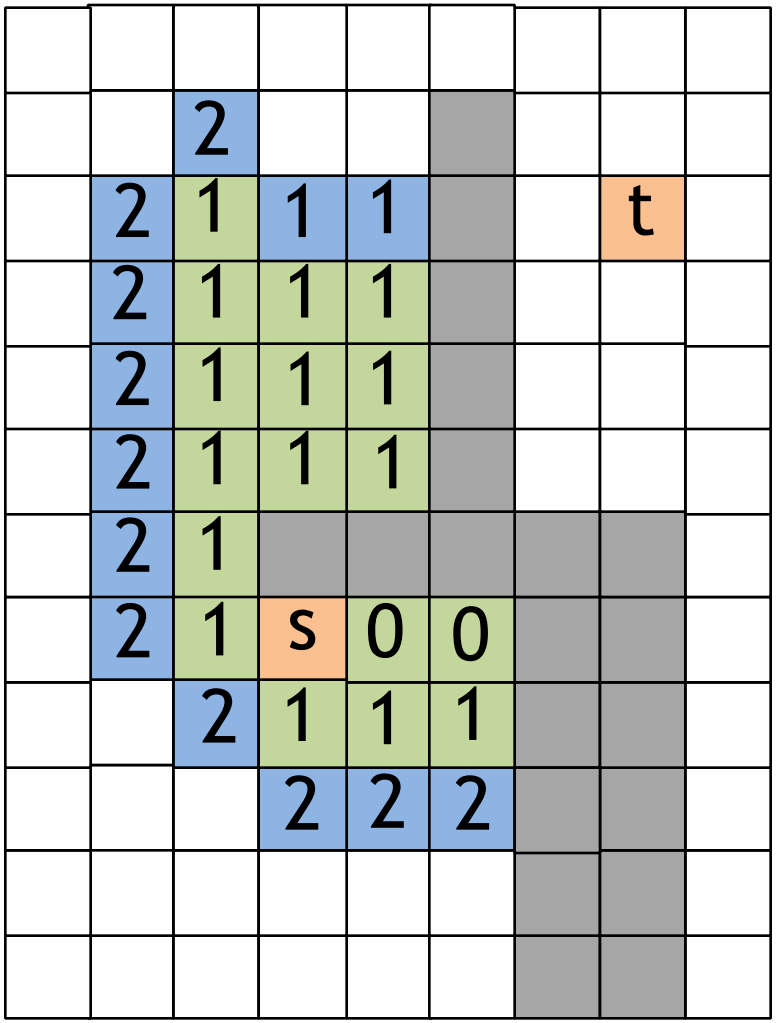
- Consider a cell with label J
  - Label each neighbor with either J or J+1, depending on the move
- Favor expansion of nodes with smaller labels

Example



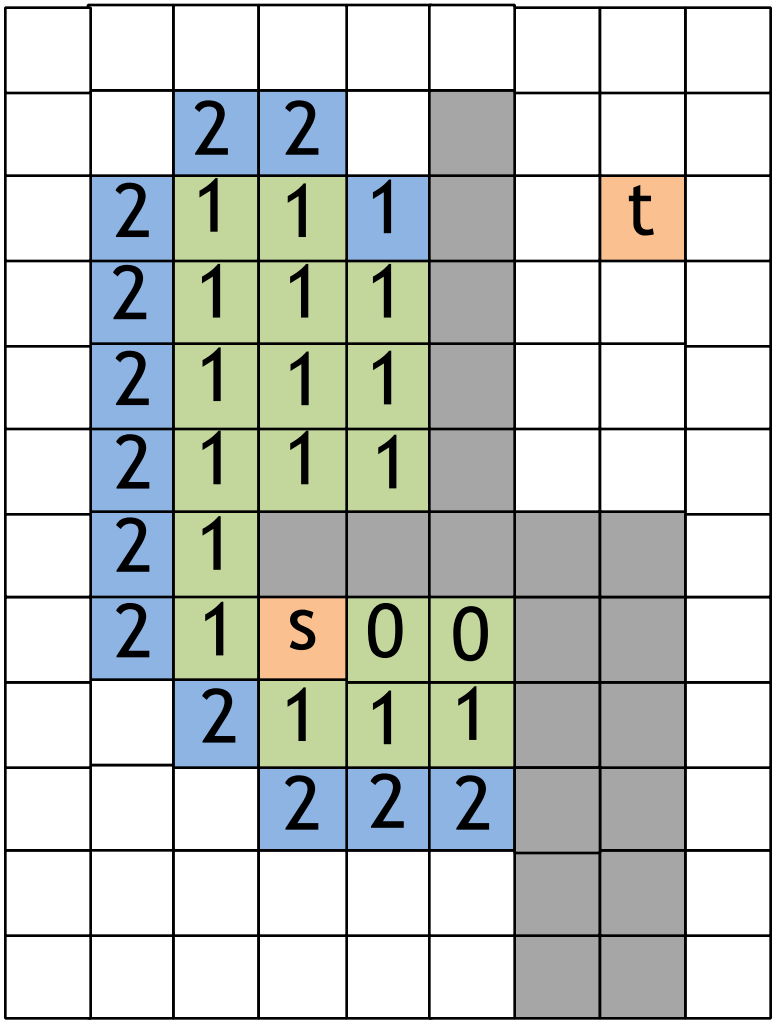
- Consider a cell with label J
  - Label each neighbor with either J or J+1, depending on the move
- Favor expansion of nodes with smaller labels

Example



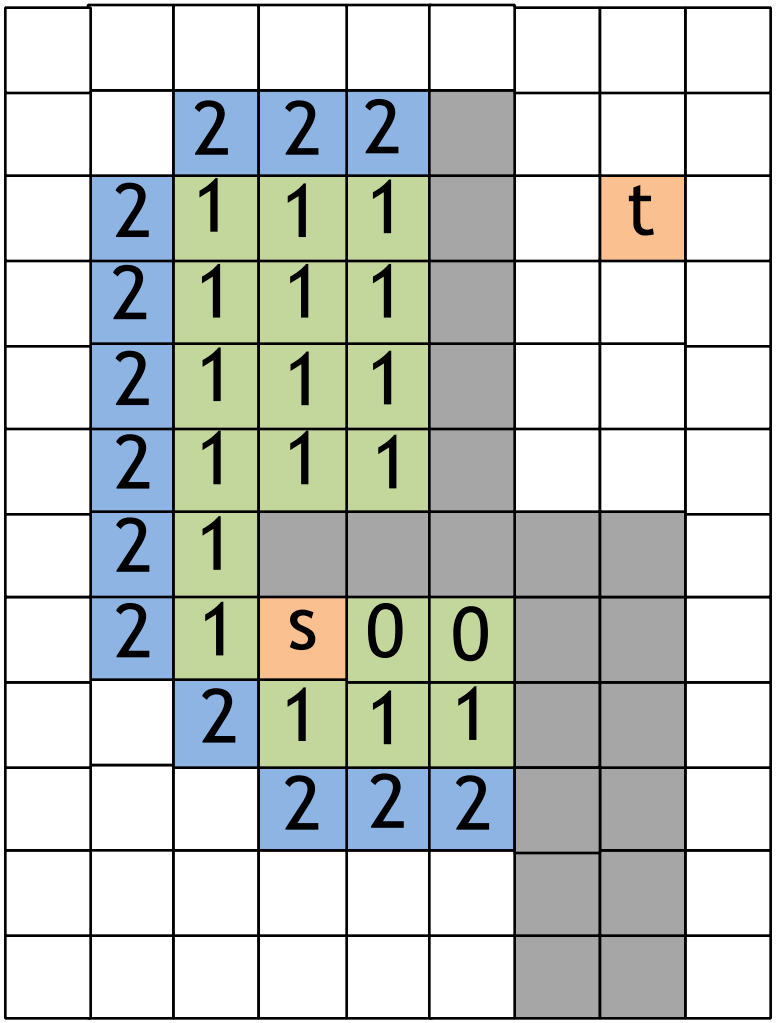
- Consider a cell with label J
  - Label each neighbor with either J or J+1, depending on the move
- Favor expansion of nodes with smaller labels

Example



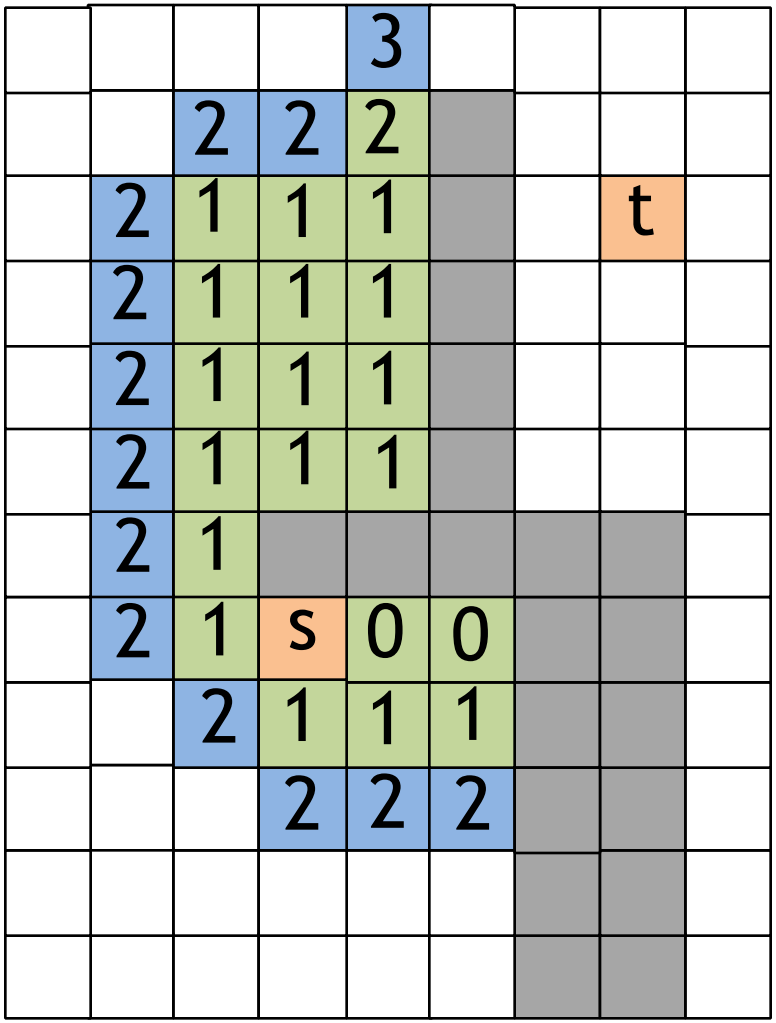
- Consider a cell with label J
  - Label each neighbor with either J or J+1, depending on the move
- Favor expansion of nodes with smaller labels

Example



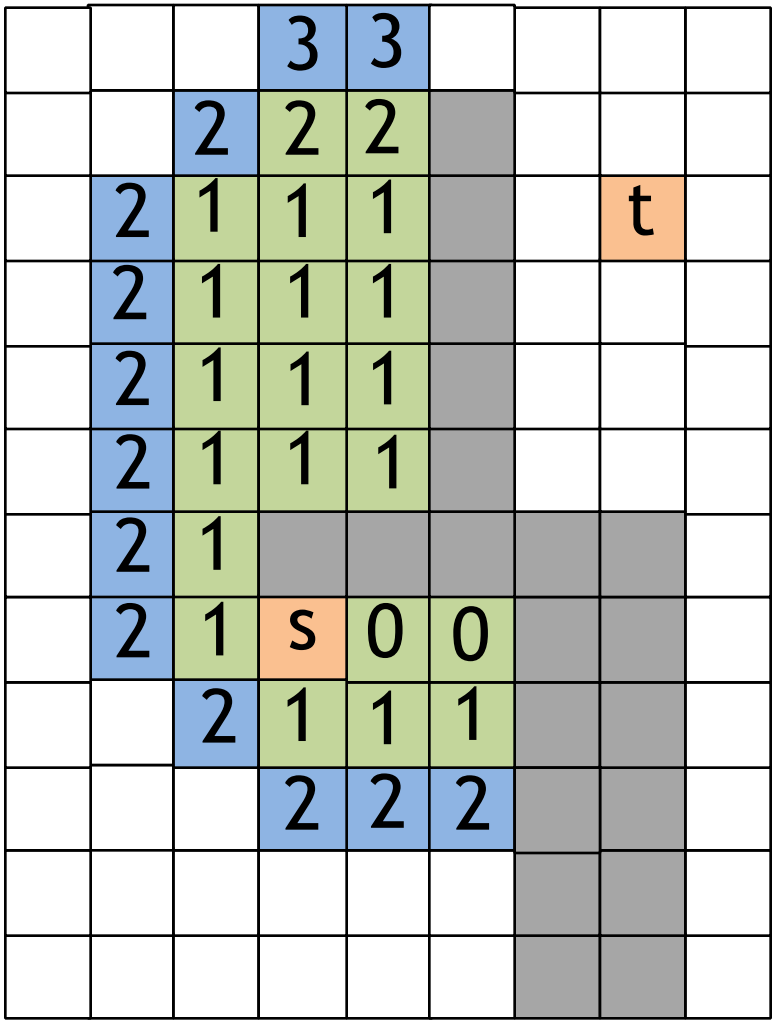
- Consider a cell with label J
  - Label each neighbor with either J or J+1, depending on the move
- Favor expansion of nodes with smaller labels

Example



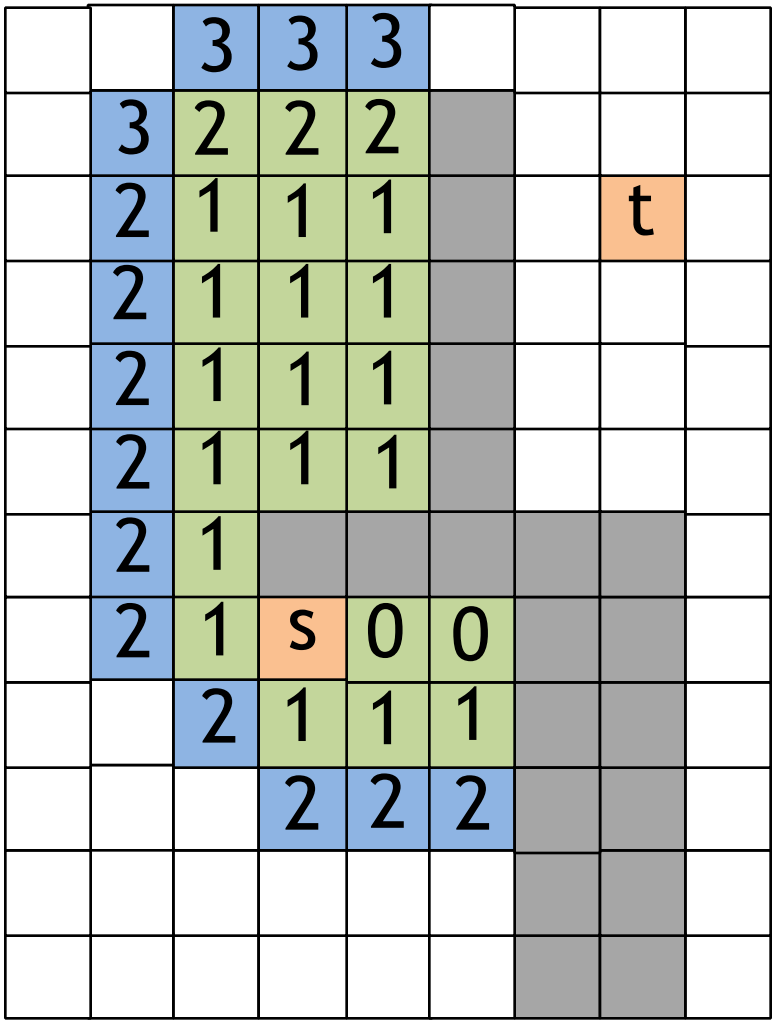
- Consider a cell with label J
  - Label each neighbor with either J or J+1, depending on the move
- Favor expansion of nodes with smaller labels

Example



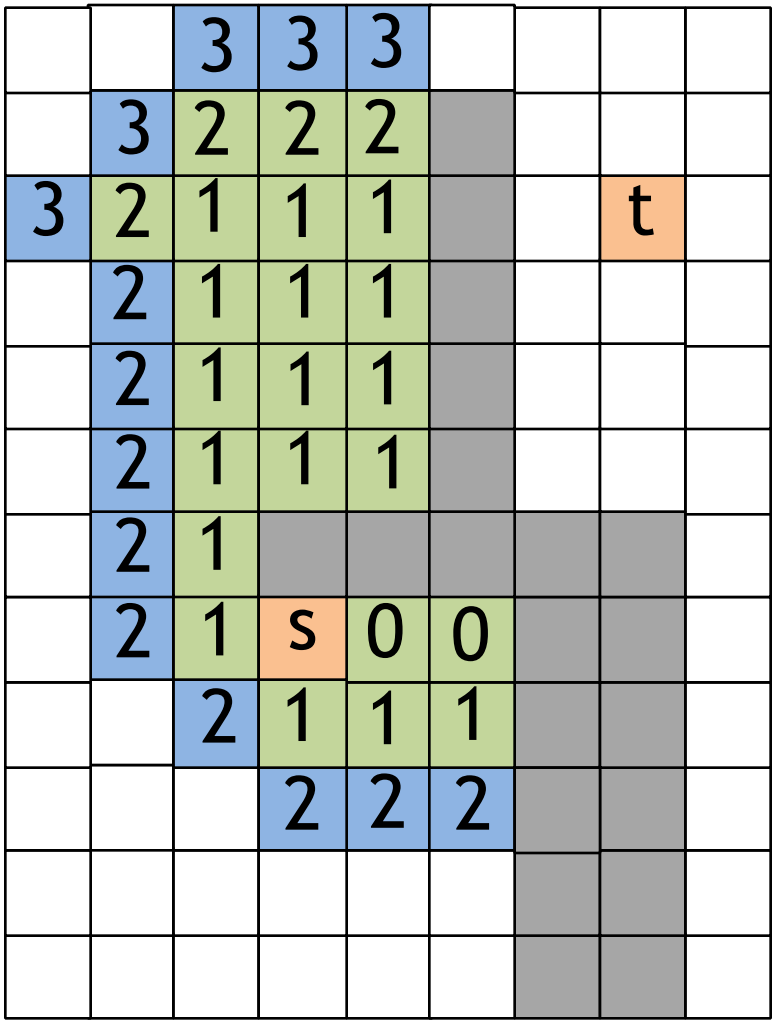
- Consider a cell with label J
  - Label each neighbor with either J or J+1, depending on the move
- Favor expansion of nodes with smaller labels

Example



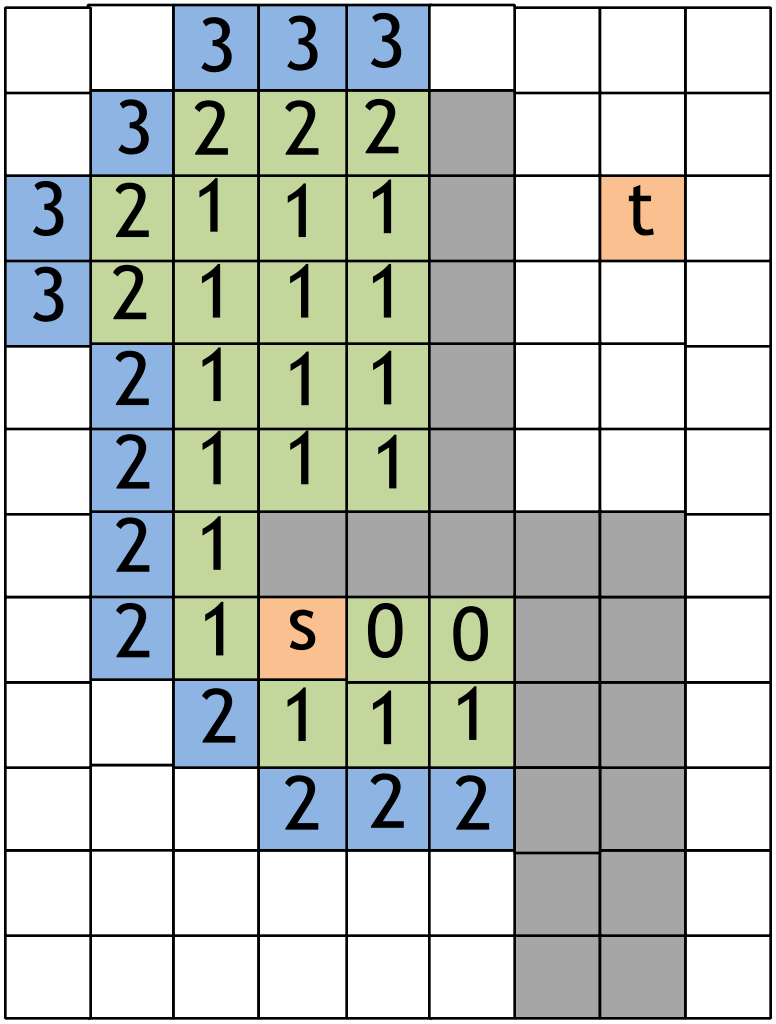
- Consider a cell with label J
  - Label each neighbor with either J or J+1, depending on the move
- Favor expansion of nodes with smaller labels

Example



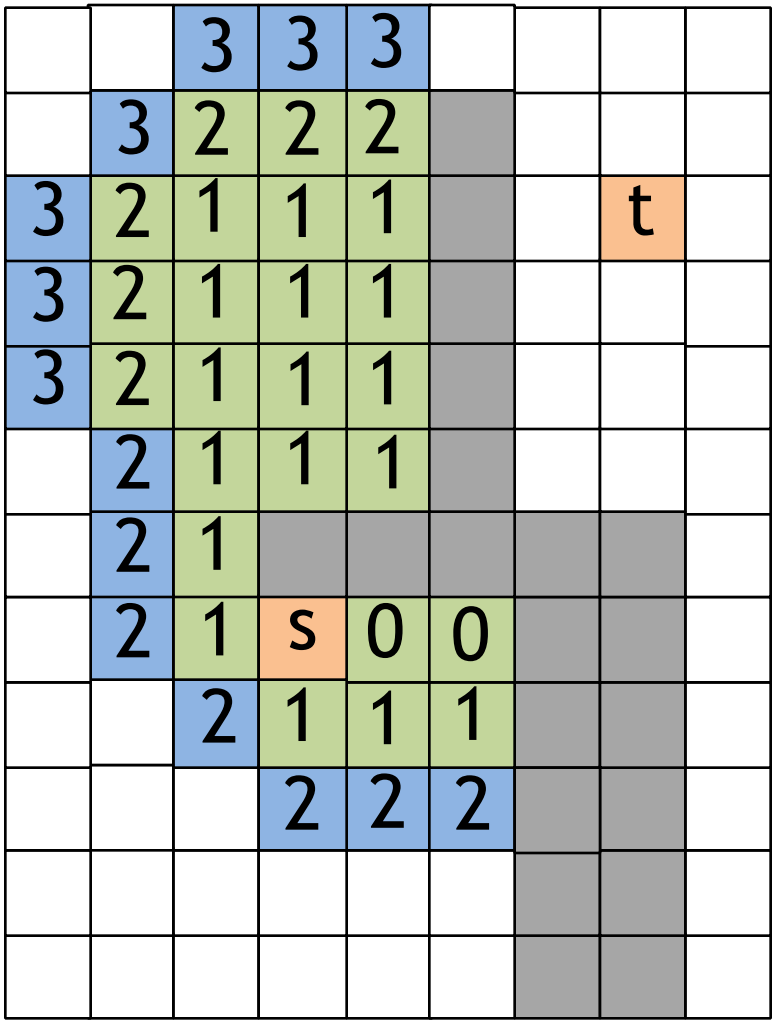
- Consider a cell with label J
  - Label each neighbor with either J or J+1, depending on the move
- Favor expansion of nodes with smaller labels

Example



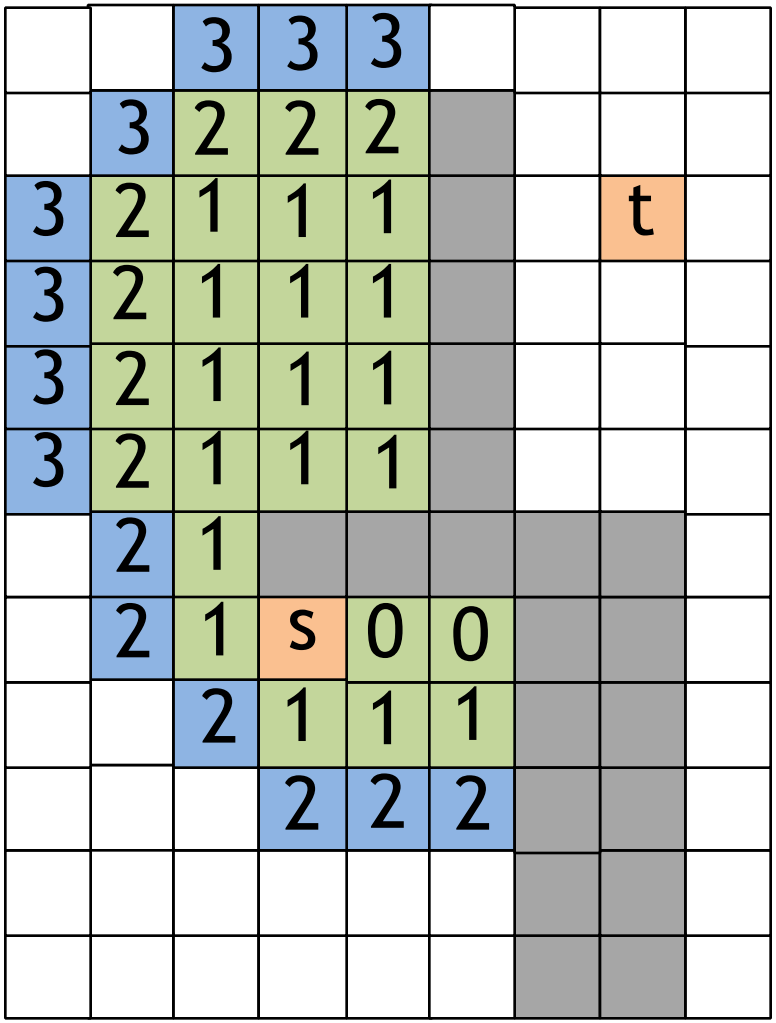
- Consider a cell with label J
  - Label each neighbor with either J or J+1, depending on the move
- Favor expansion of nodes with smaller labels

Example



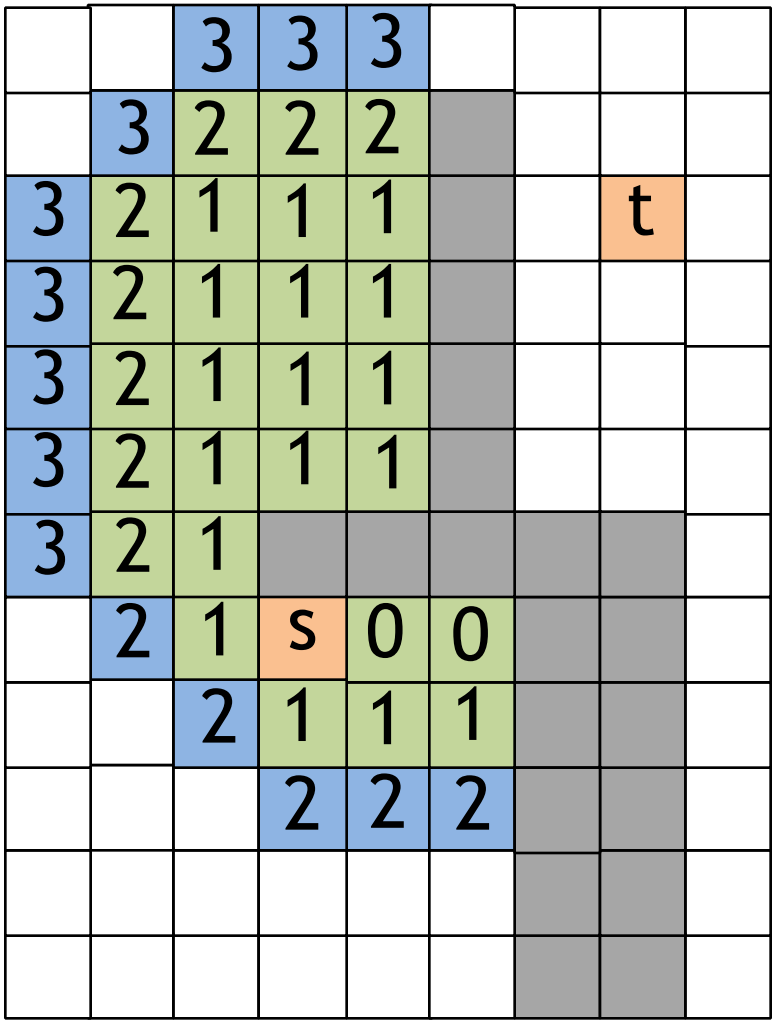
- Consider a cell with label J
  - Label each neighbor with either J or J+1, depending on the move
- Favor expansion of nodes with smaller labels

Example



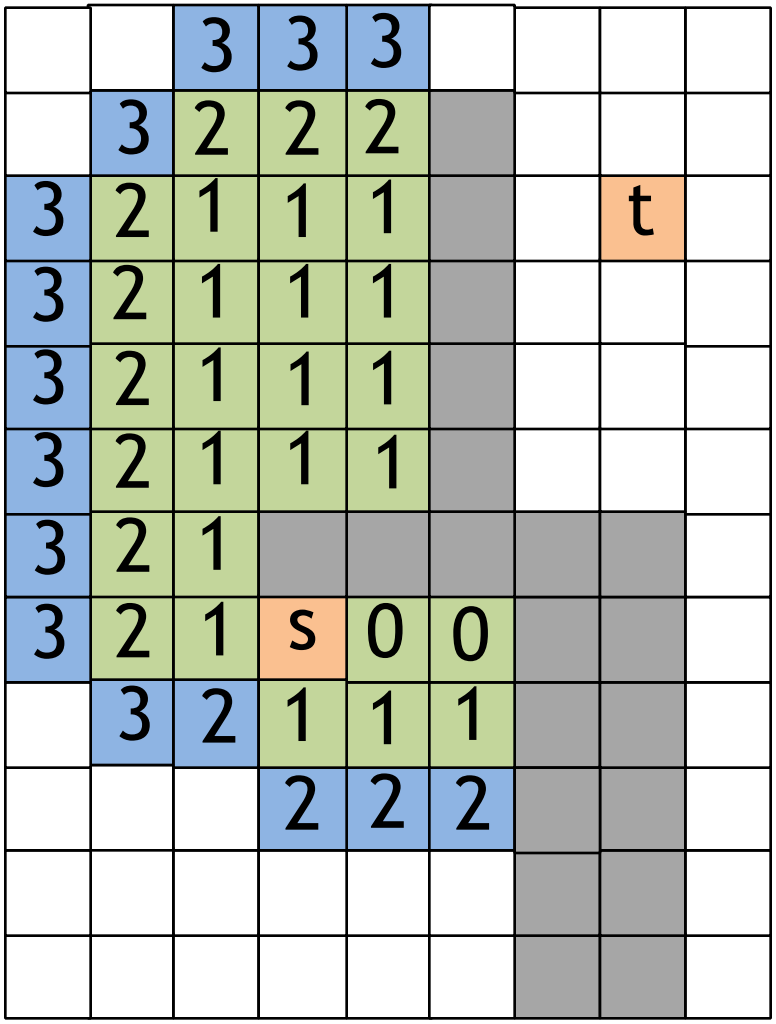
- Consider a cell with label J
  - Label each neighbor with either J or J+1, depending on the move
- Favor expansion of nodes with smaller labels

Example



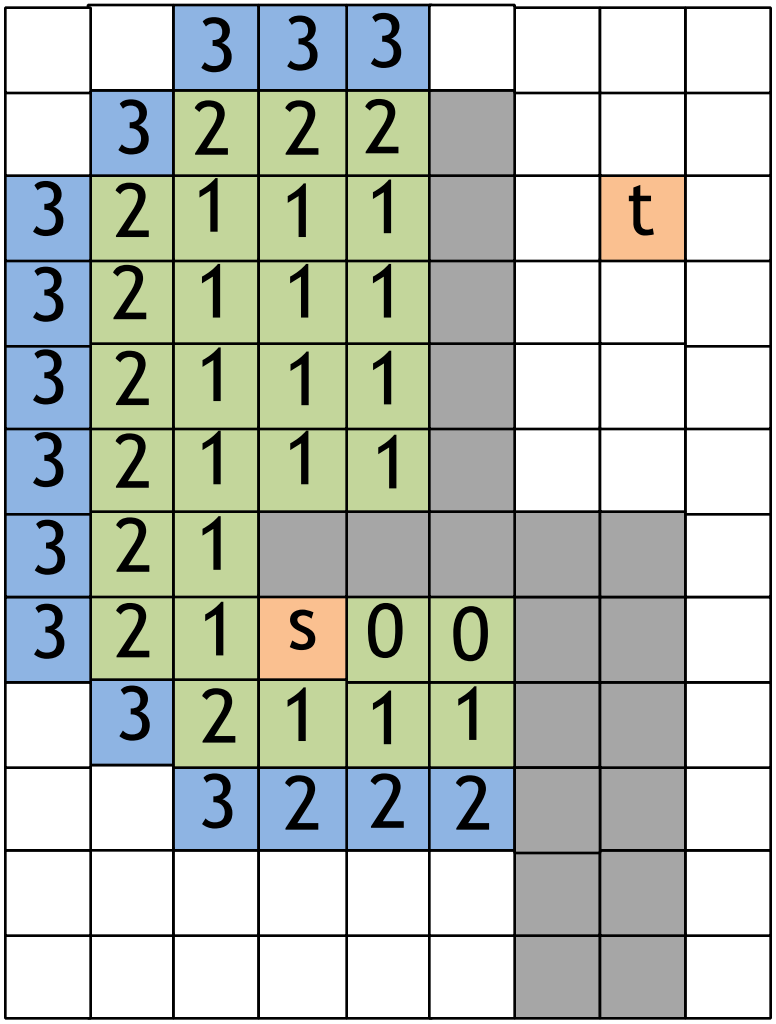
- Consider a cell with label J
  - Label each neighbor with either J or J+1, depending on the move
- Favor expansion of nodes with smaller labels

Example



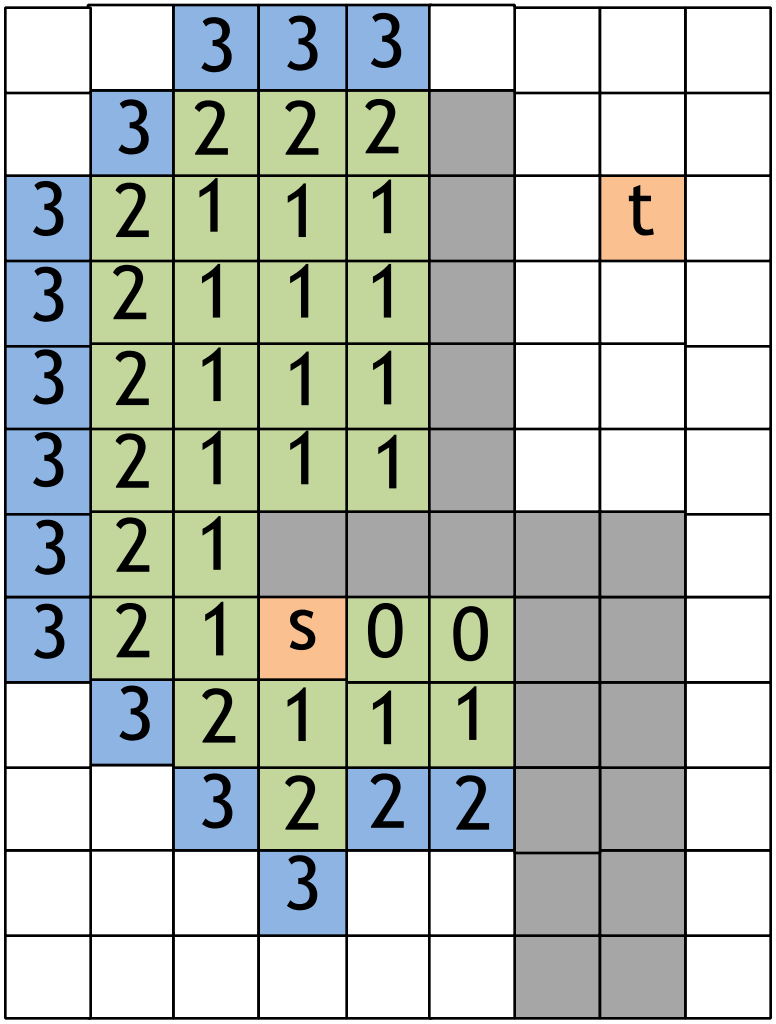
- Consider a cell with label J
  - Label each neighbor with either J or J+1, depending on the move
- Favor expansion of nodes with smaller labels

Example



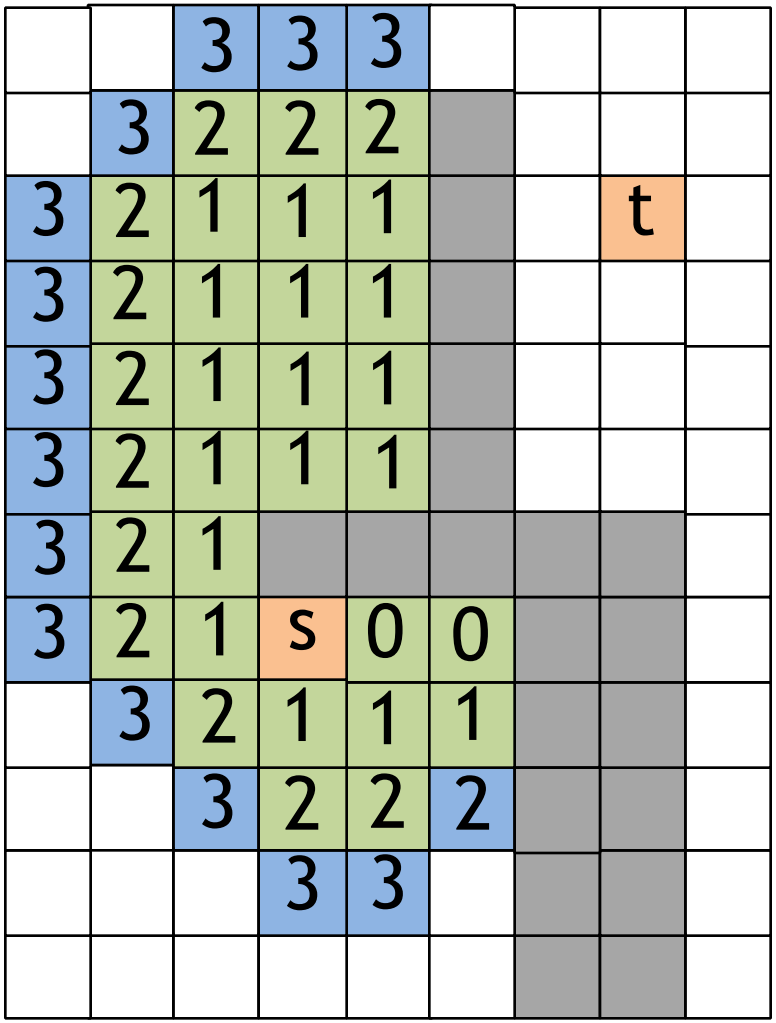
- Consider a cell with label J
  - Label each neighbor with either J or J+1, depending on the move
- Favor expansion of nodes with smaller labels

Example



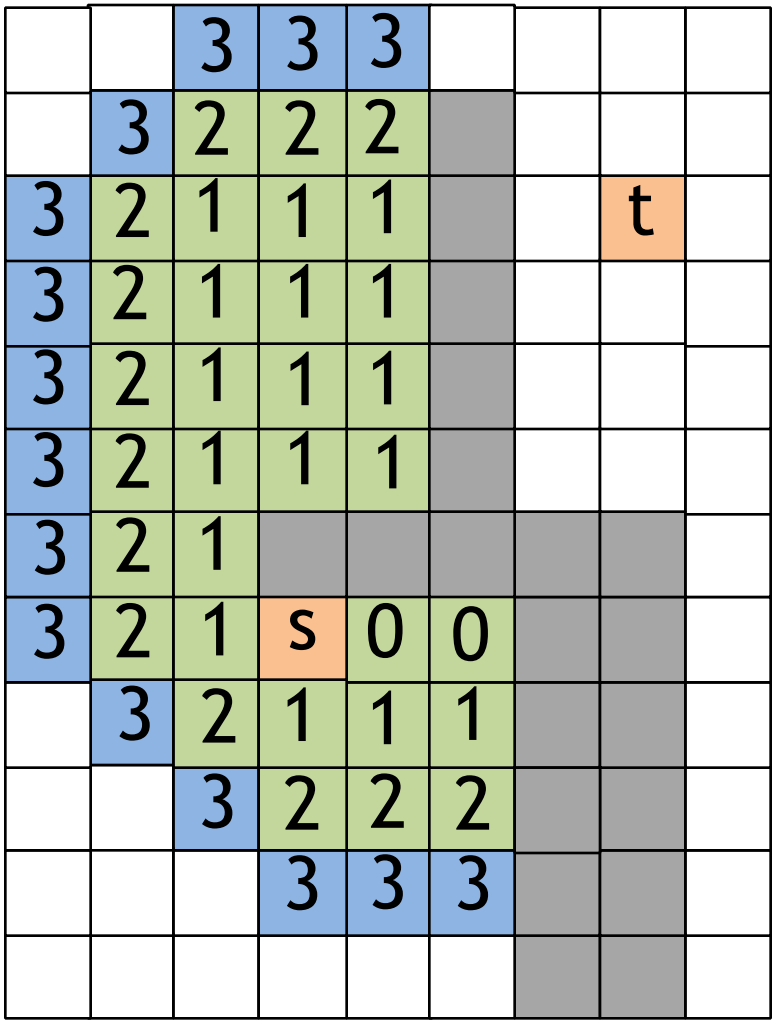
- Consider a cell with label J
  - Label each neighbor with either J or J+1, depending on the move
- Favor expansion of nodes with smaller labels

Example



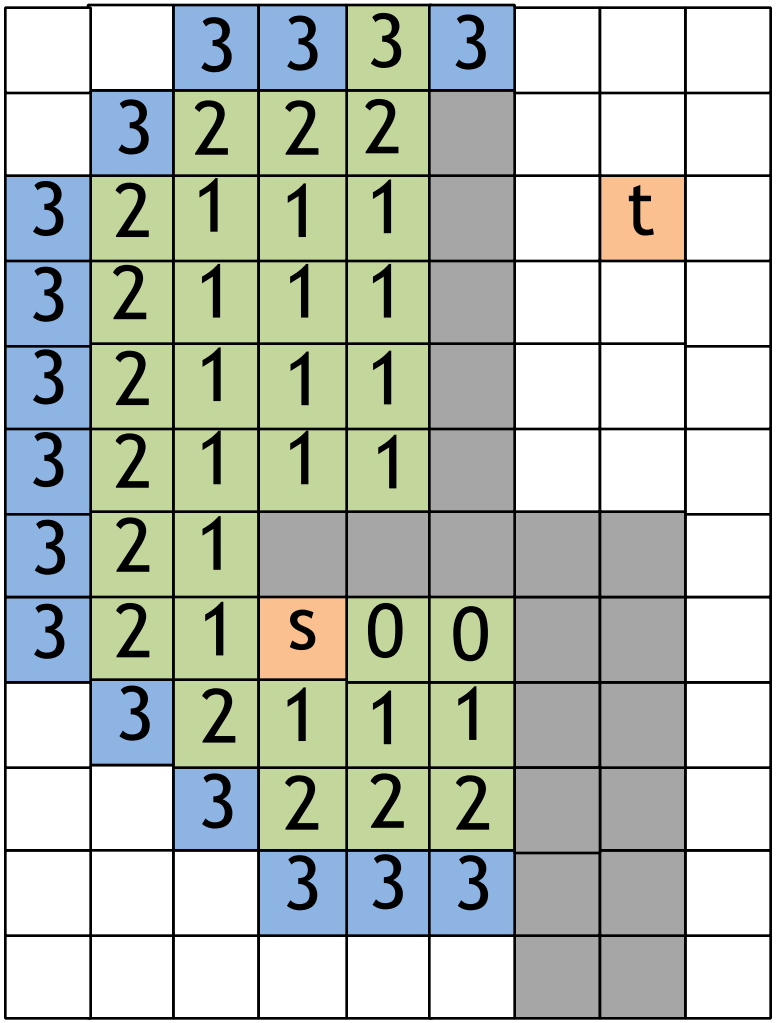
- Consider a cell with label J
  - Label each neighbor with either J or J+1, depending on the move
- Favor expansion of nodes with smaller labels

Example



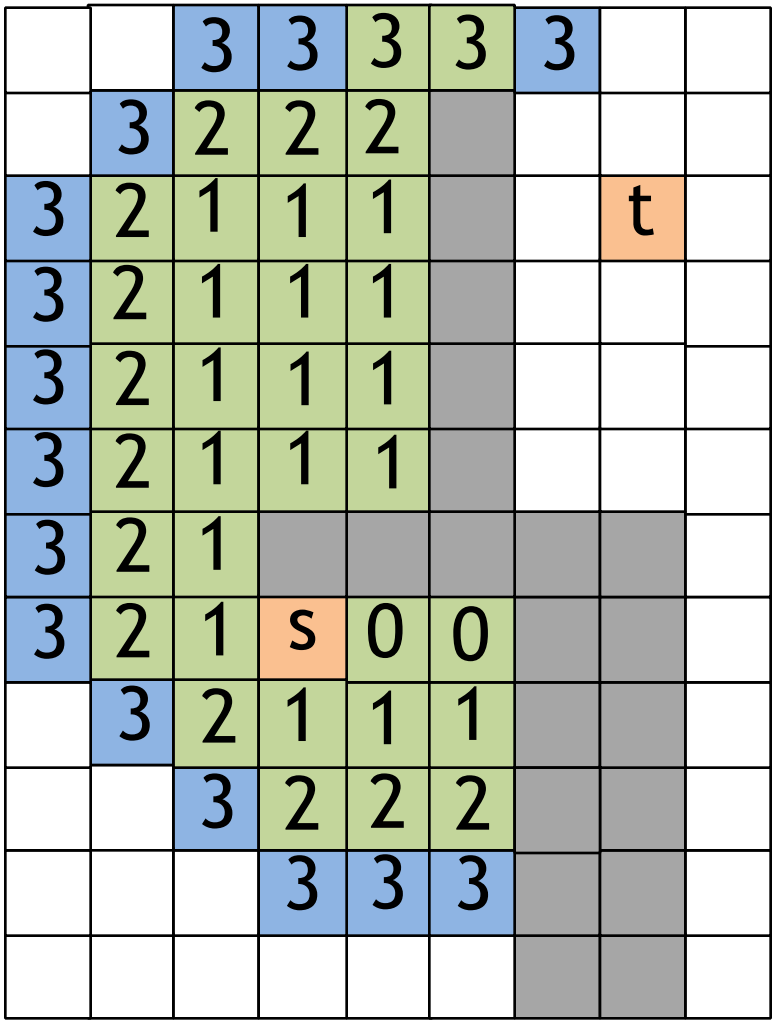
- Consider a cell with label J
  - Label each neighbor with either J or J+1, depending on the move
- Favor expansion of nodes with smaller labels

Example



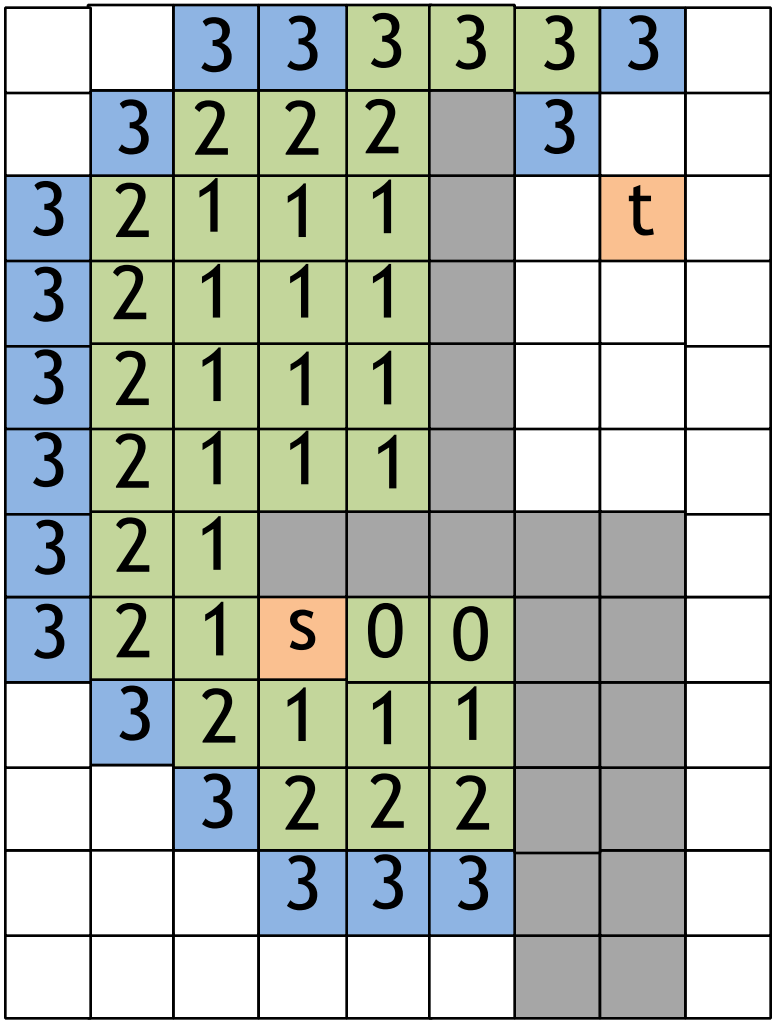
- Consider a cell with label J
  - Label each neighbor with either J or J+1, depending on the move
- Favor expansion of nodes with smaller labels

Example



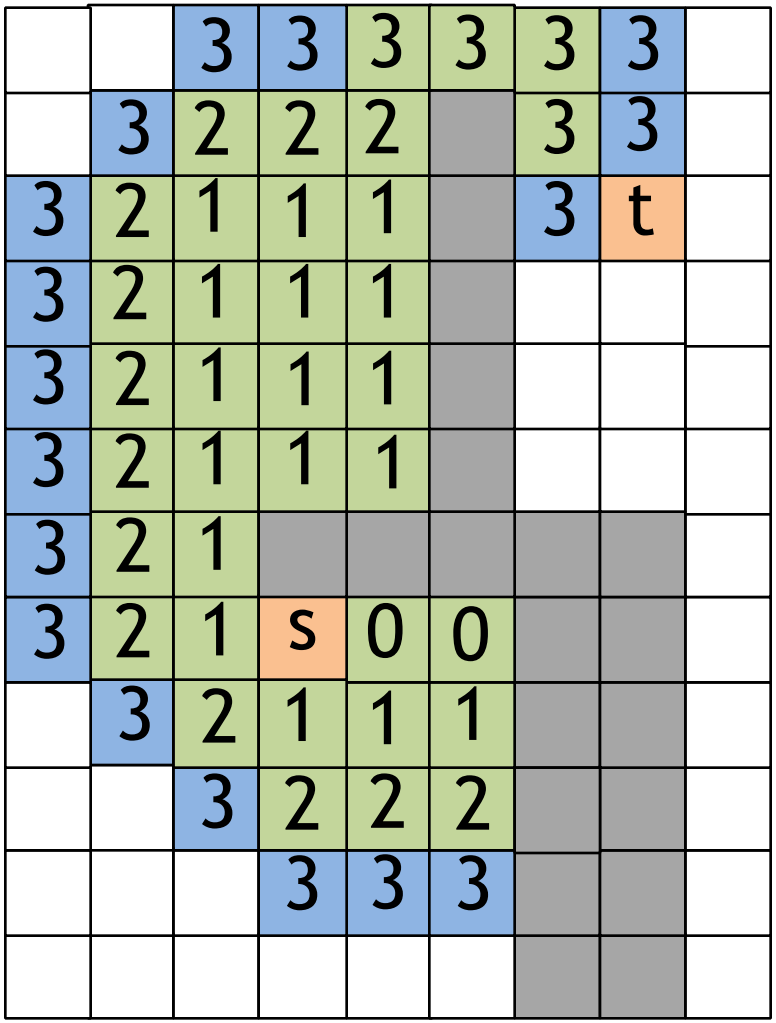
- Consider a cell with label J
  - Label each neighbor with either J or J+1, depending on the move
- Favor expansion of nodes with smaller labels

Example



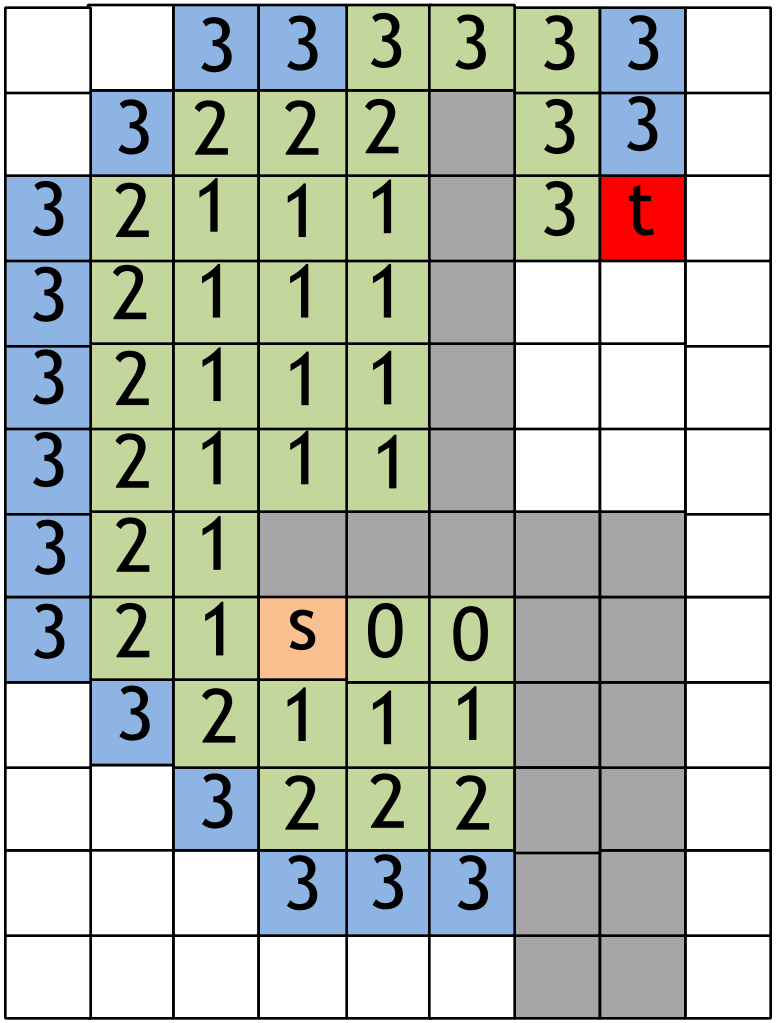
- Consider a cell with label J
  - Label each neighbor with either J or J+1, depending on the move
- Favor expansion of nodes with smaller labels

Example



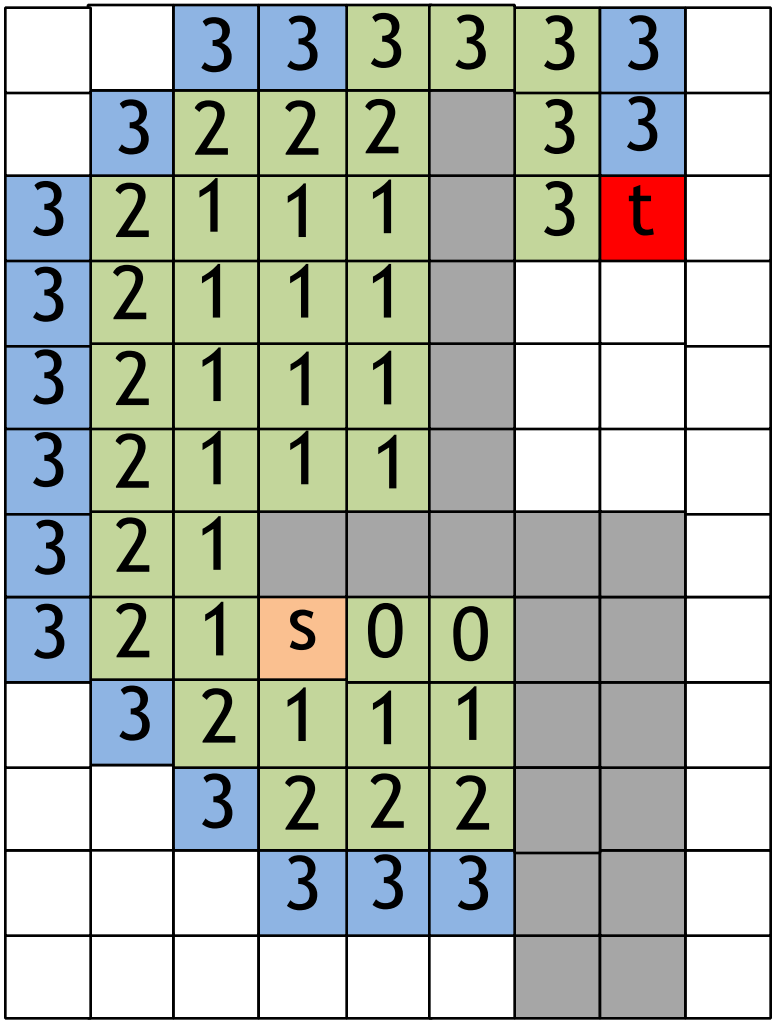
- Consider a cell with label J
  - Label each neighbor with either J or J+1, depending on the move
- Favor expansion of nodes with smaller labels

Example



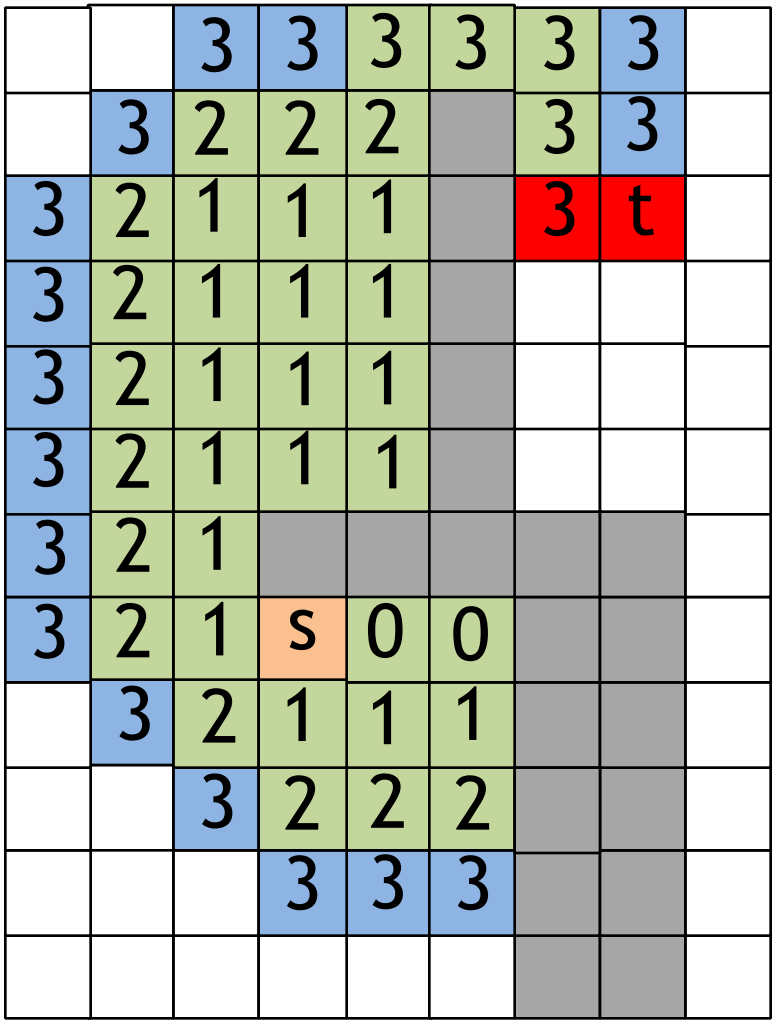
- Consider a cell with label J
  - Label each neighbor with either J or J+1, depending on the move
- Favor expansion of nodes with smaller labels

Path Retracing



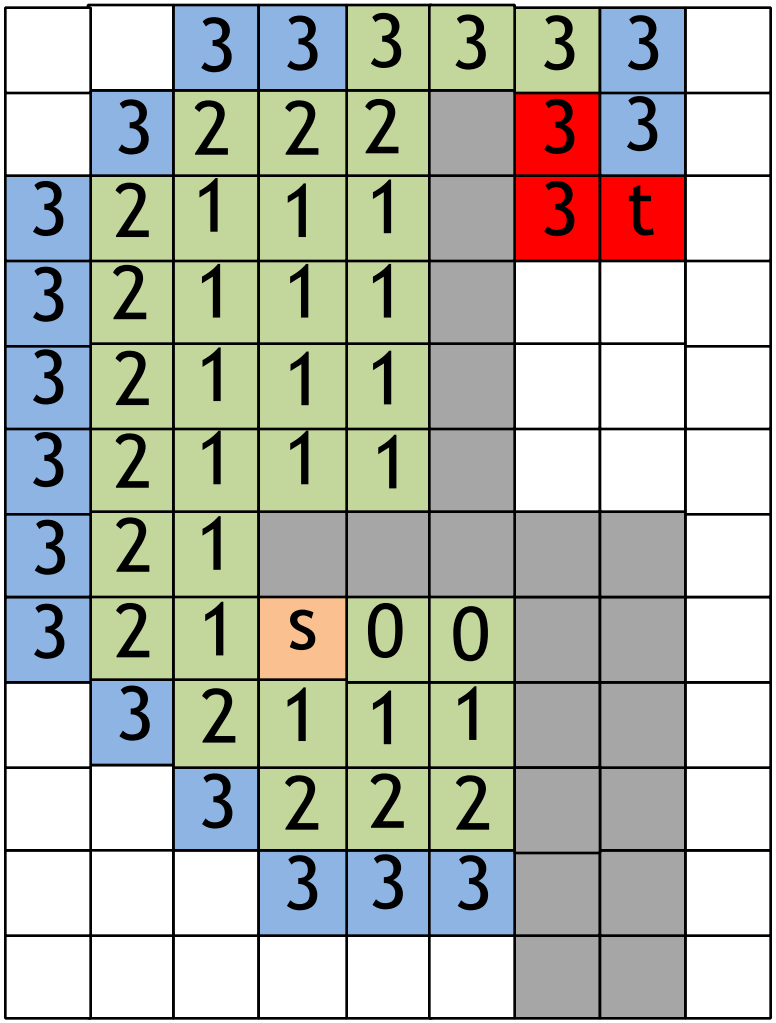
- Somewhat more complex
- Move to an expanded node with a lower label if possible
- Move toward s whenever possible

Path Retracing



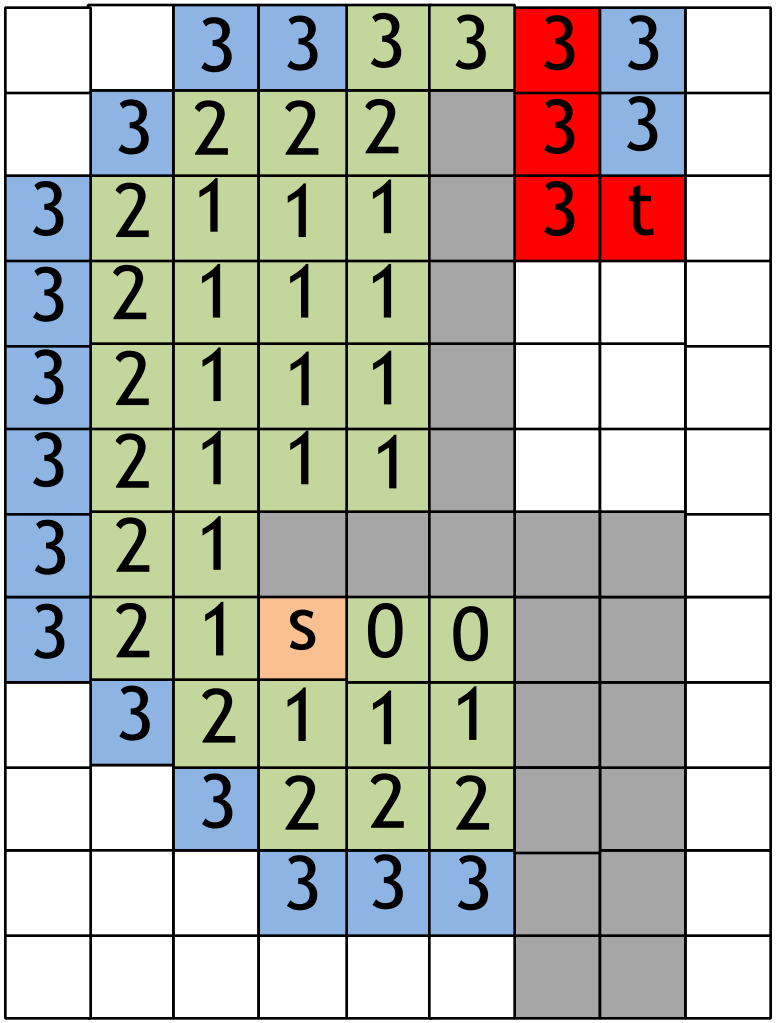
- Somewhat more complex
- Move to an expanded node with a lower label if possible
- Move toward s whenever possible

Path Retracing



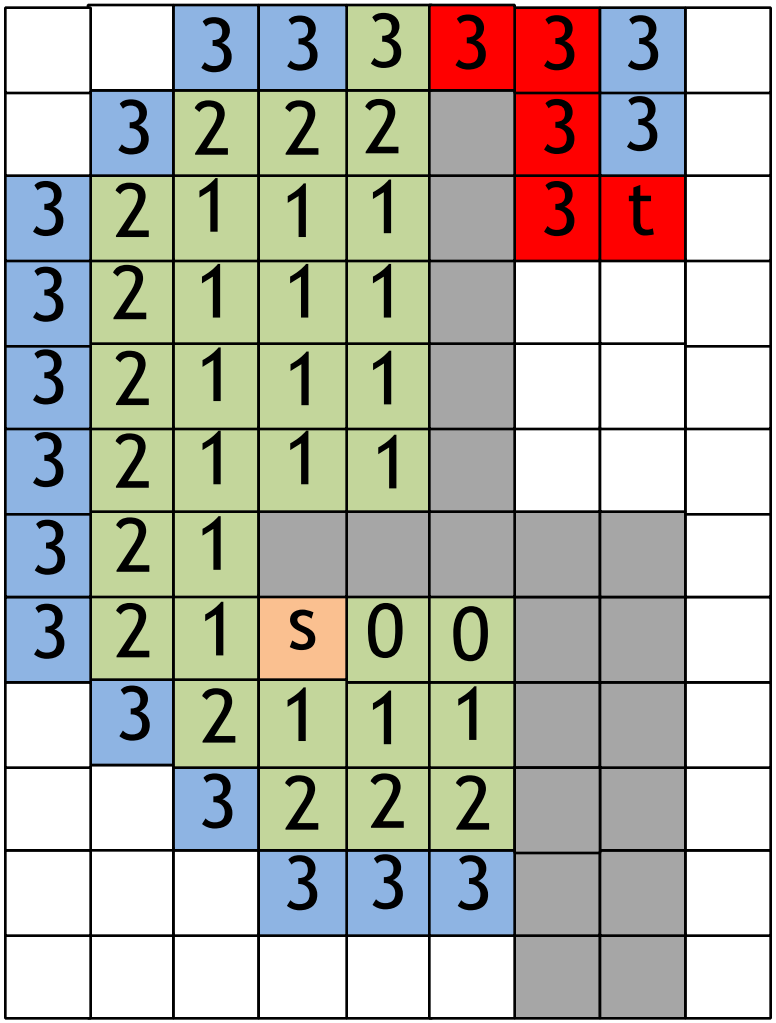
- Somewhat more complex
- Move to an expanded node with a lower label if possible
- Move toward s whenever possible

Path Retracing



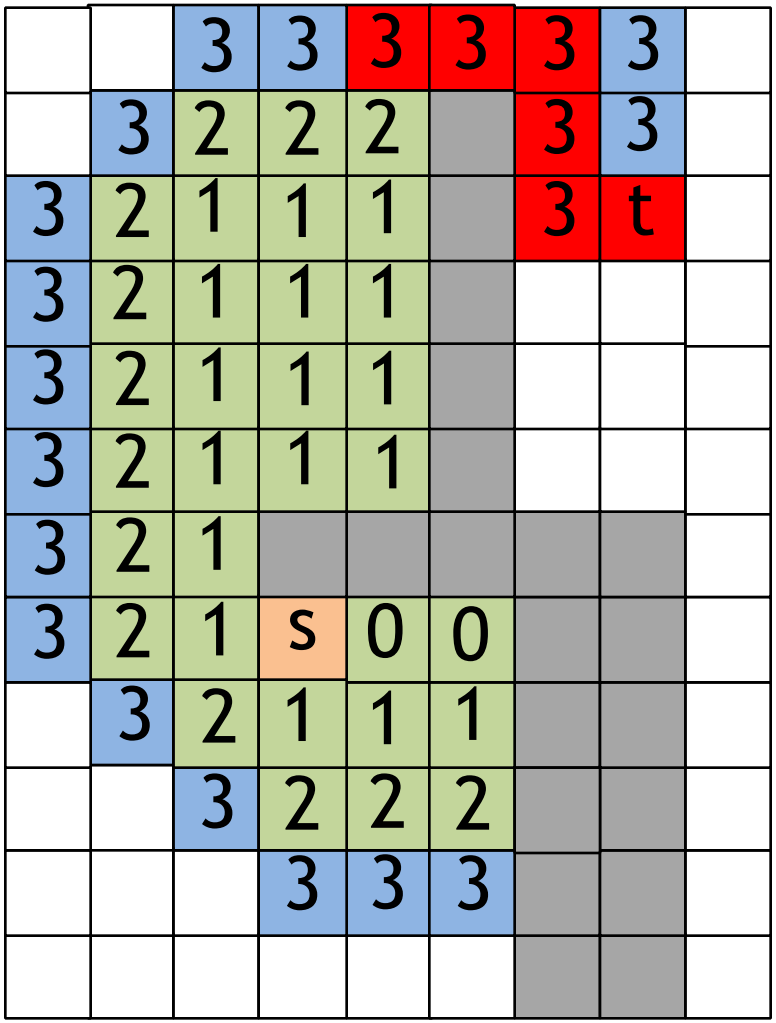
- Somewhat more complex
- Move to an expanded node with a lower label if possible
- Move toward s whenever possible

Path Retracing



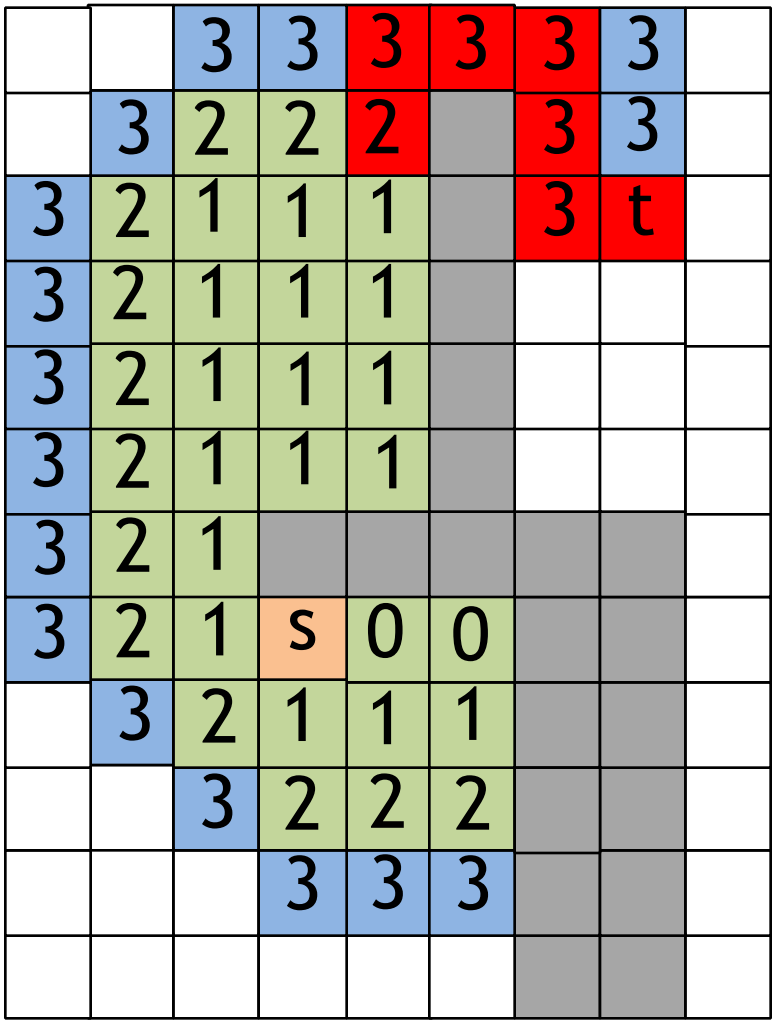
- Somewhat more complex
- Move to an expanded node with a lower label if possible
- Move toward s whenever possible

Path Retracing



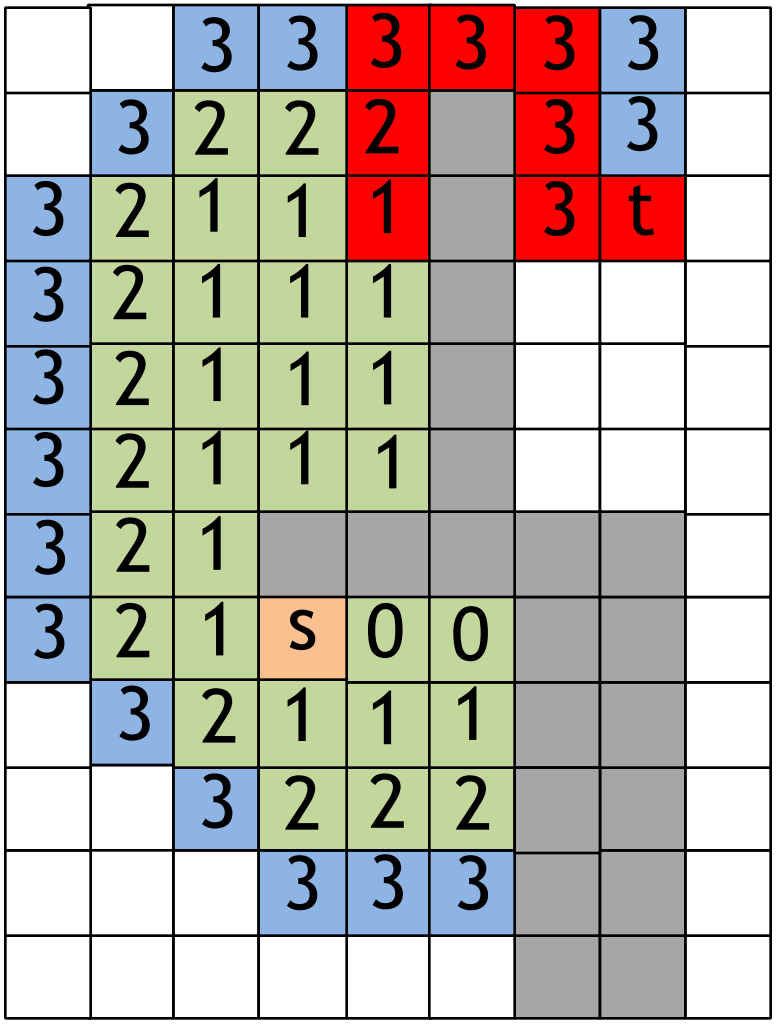
- Somewhat more complex
- Move to an expanded node with a lower label if possible
- Move toward s whenever possible

Path Retracing



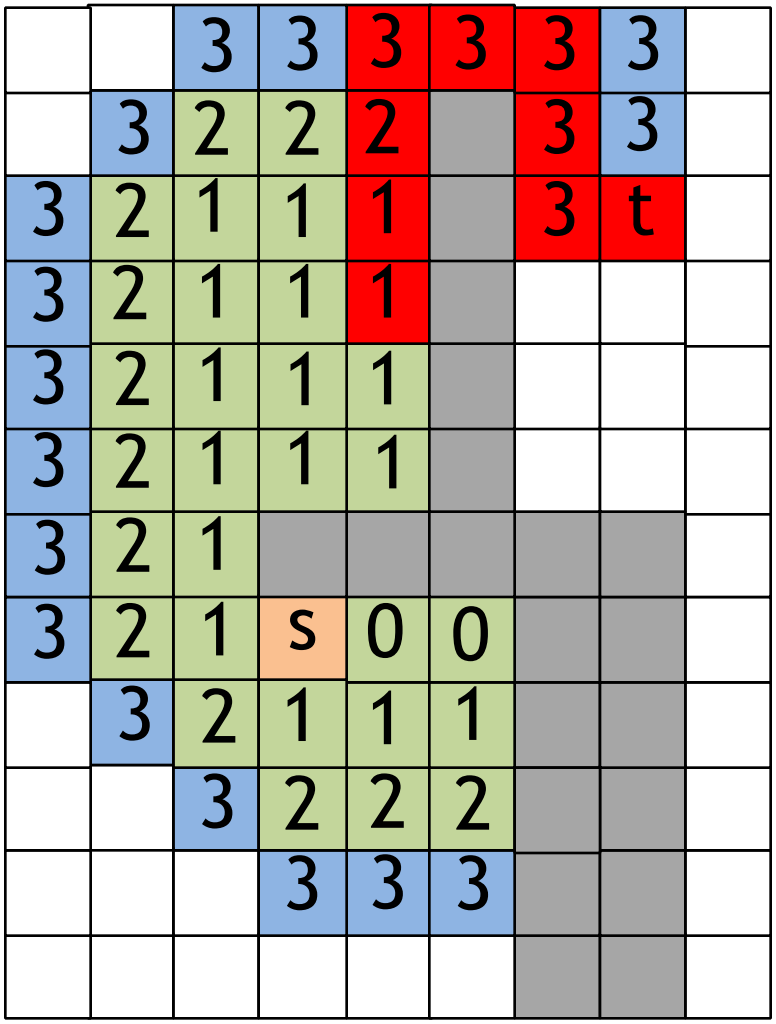
- Somewhat more complex
- Move to an expanded node with a lower label if possible
- Move toward s whenever possible

Path Retracing



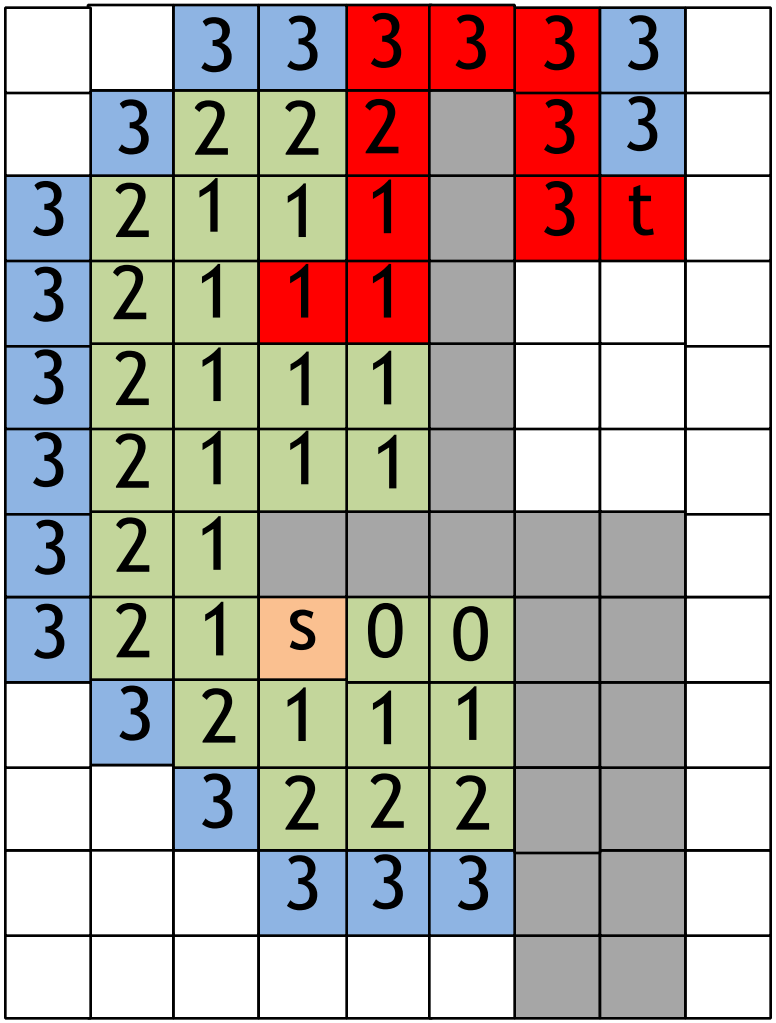
- Somewhat more complex
- Move to an expanded node with a lower label if possible
- Move toward s whenever possible

Path Retracing



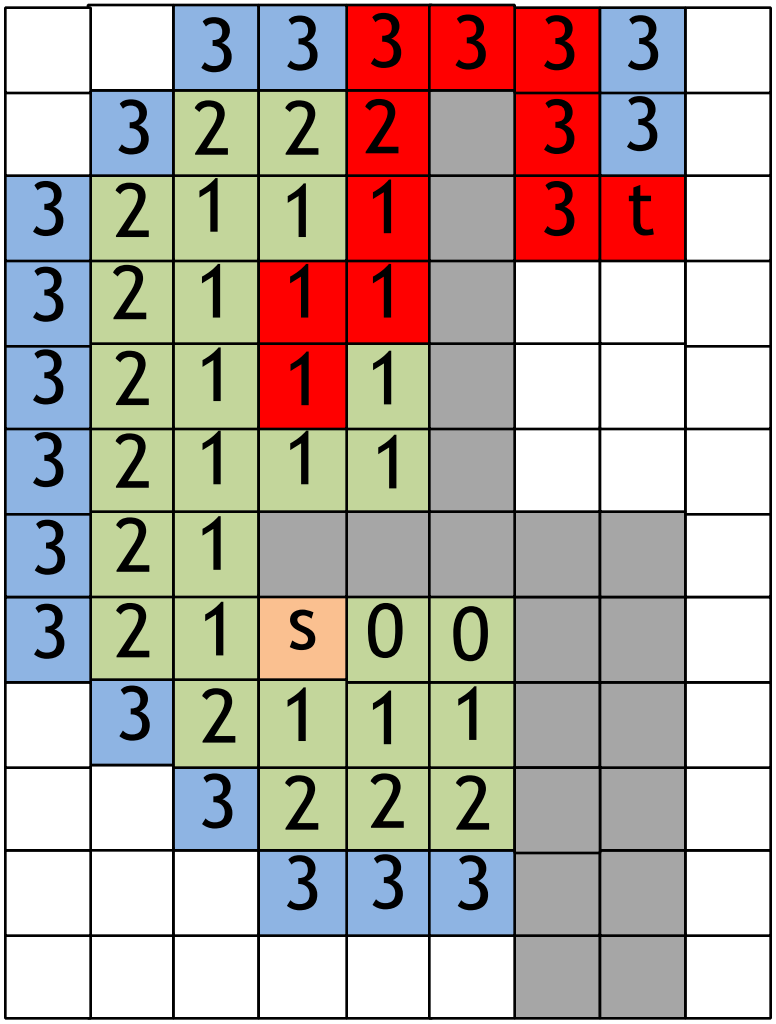
- Somewhat more complex
- Move to an expanded node with a lower label if possible
- Move toward s whenever possible

Path Retracing



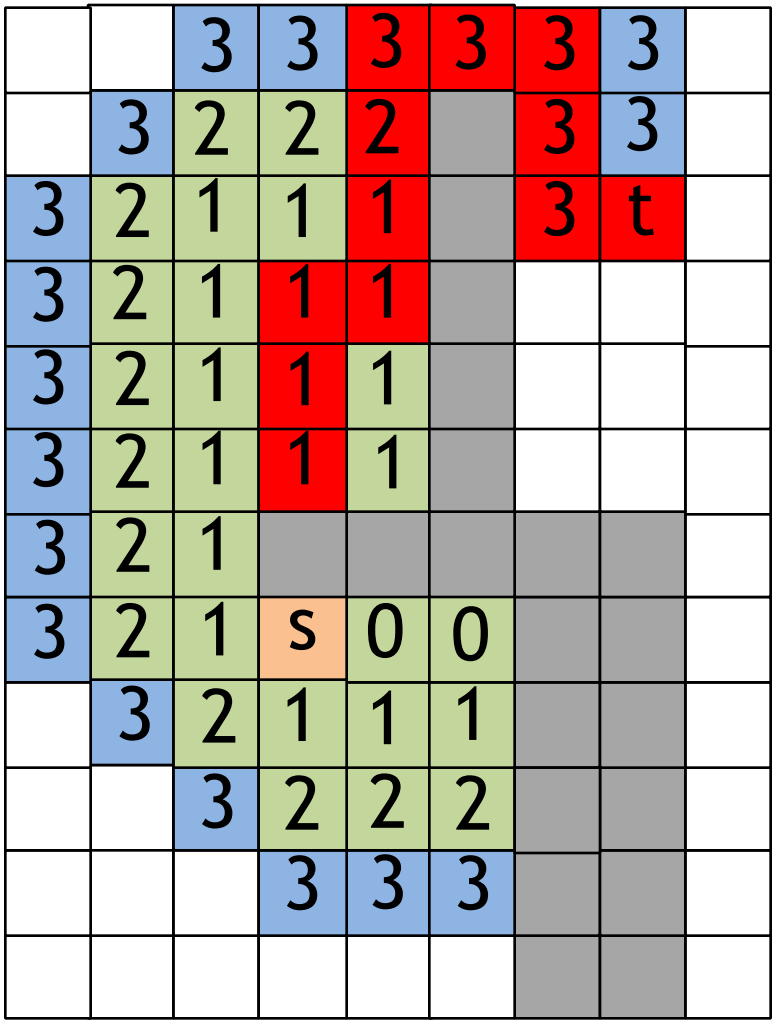
- Somewhat more complex
- Move to an expanded node with a lower label if possible
- Move toward s whenever possible

Path Retracing



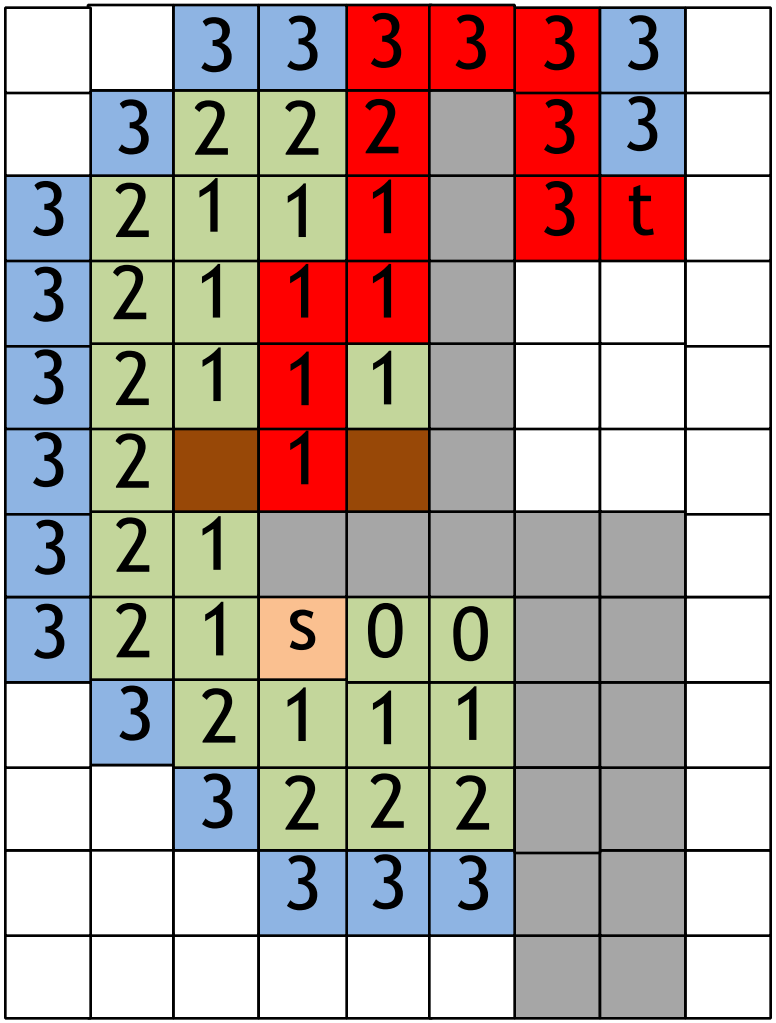
- Somewhat more complex
- Move to an expanded node with a lower label if possible
- Move toward s whenever possible

Path Retracing



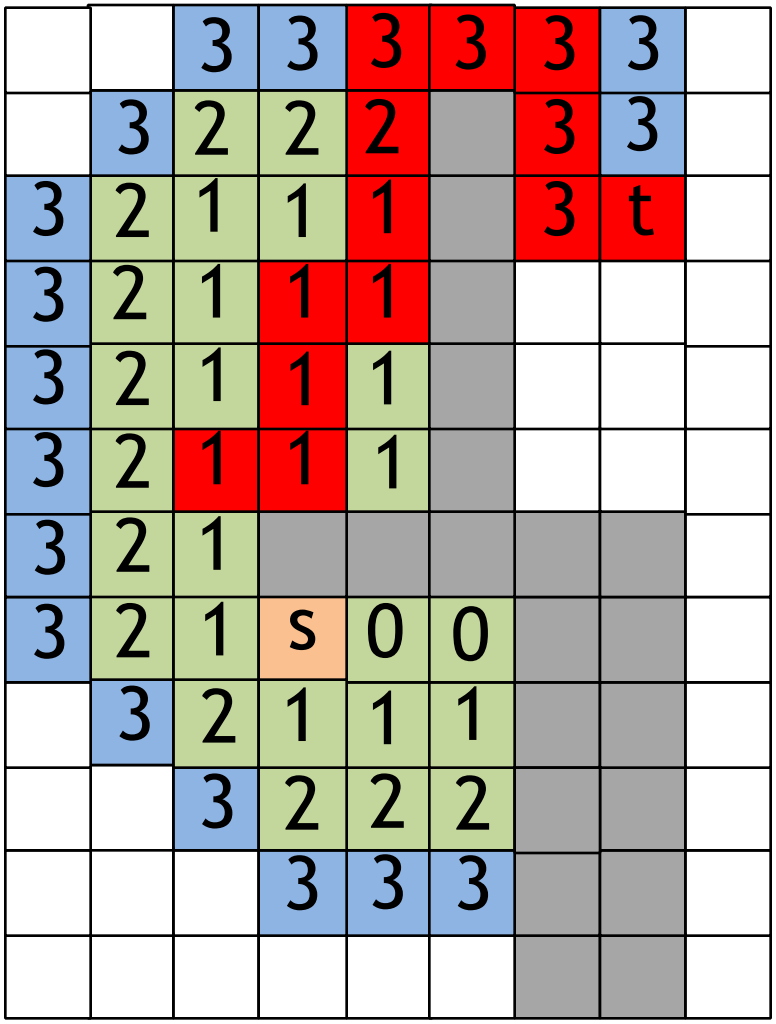
- Somewhat more complex
- Move to an expanded node with a lower label if possible
- Move toward s whenever possible

Path Retracing



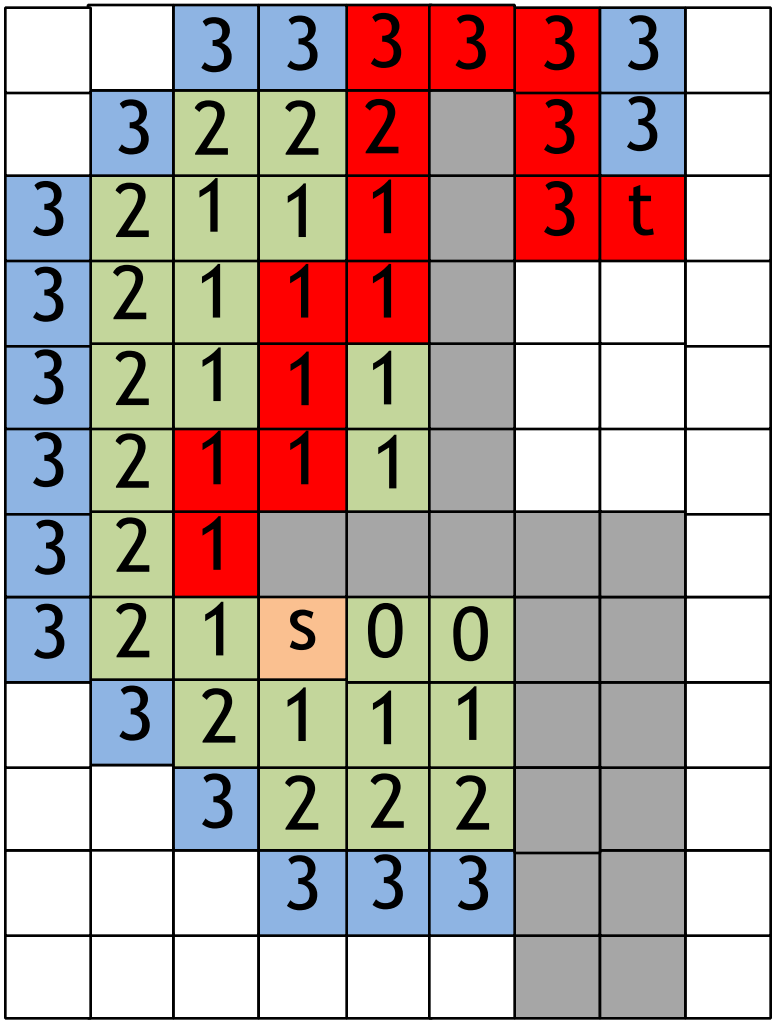
- Somewhat more complex
- Move to an expanded node with a lower label if possible
- Move toward s whenever possible
- Could go either way…

Path Retracing



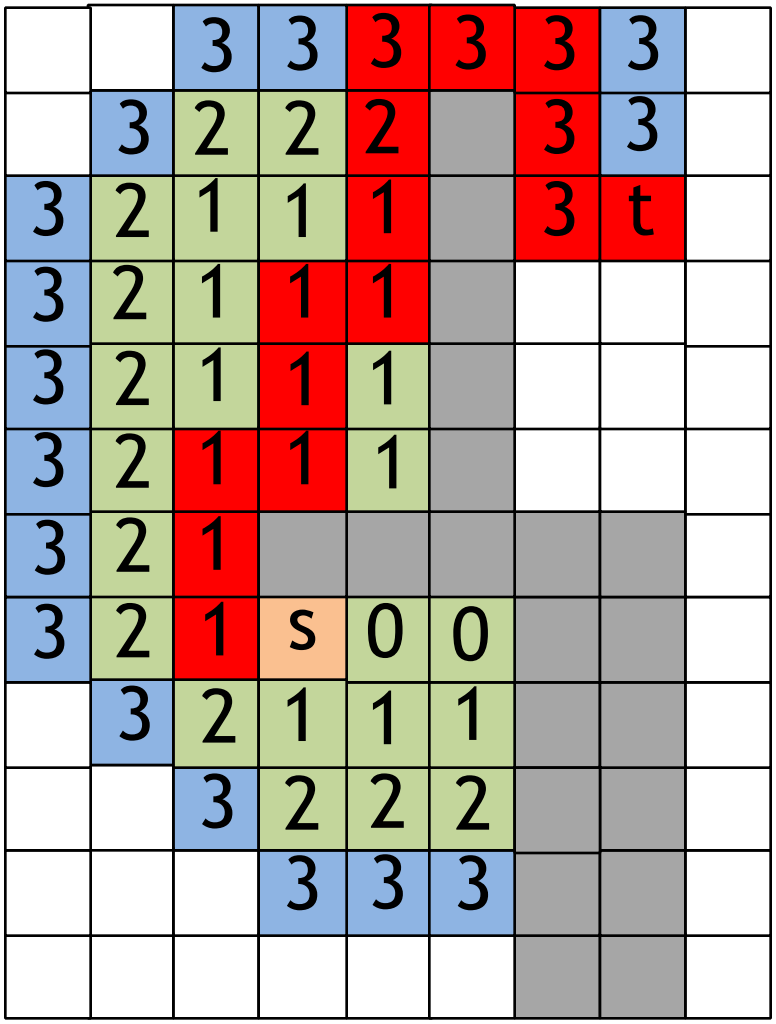
- Somewhat more complex
- Move to an expanded node with a lower label if possible
- Move toward s whenever possible

Path Retracing



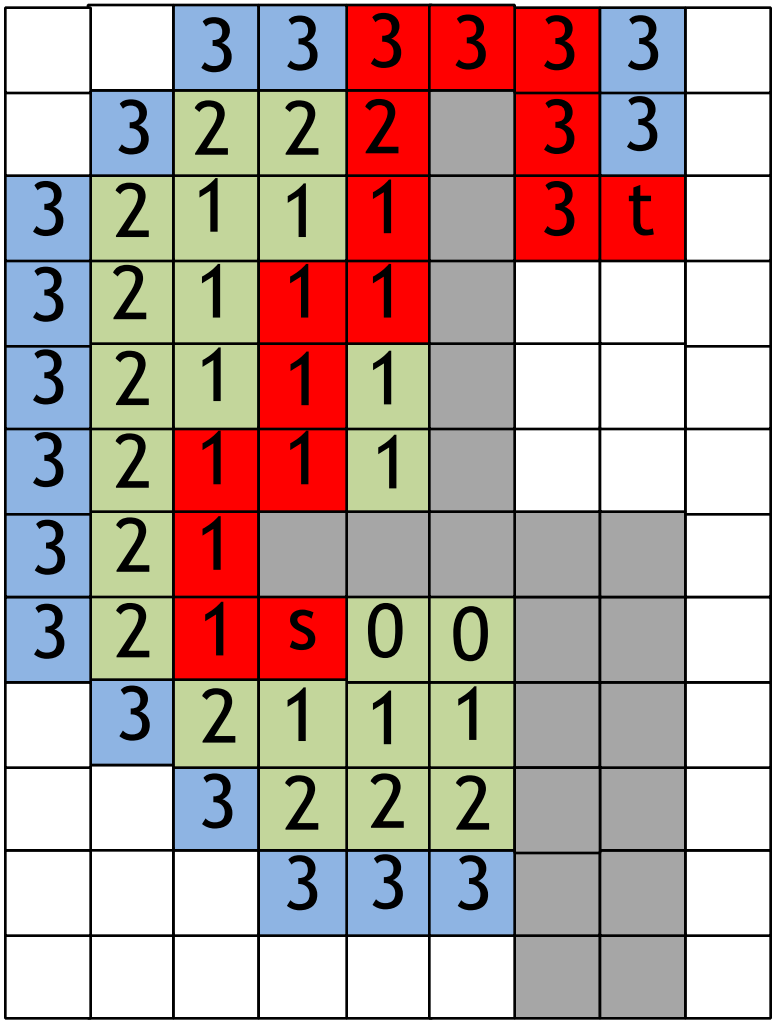
- Somewhat more complex
- Move to an expanded node with a lower label if possible
- Move toward s whenever possible

Path Retracing



- Somewhat more complex
- Move to an expanded node with a lower label if possible
- Move toward s whenever possible

Path Retracing



- Somewhat more complex
- Move to an expanded node with a lower label if possible
- Move toward s whenever possible

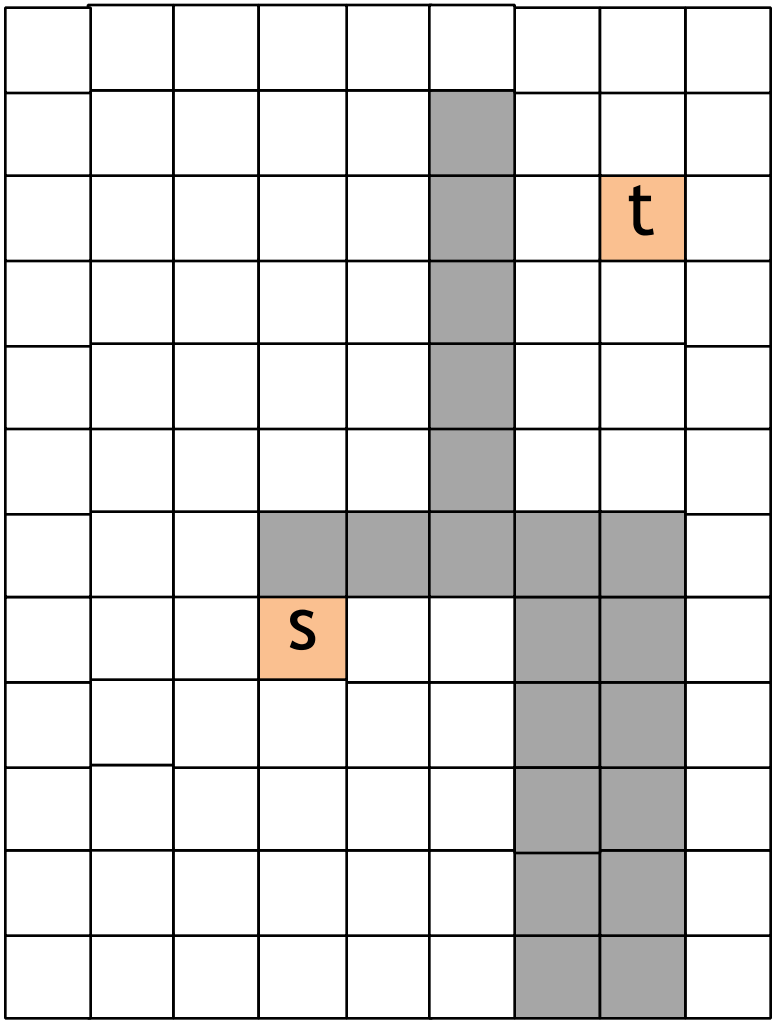
Soukup's Algorithm

Soukup, J. (1978), *Fast Maze Router*, Design Automation Conference, 100-102

Soukup's Algorithm

- Use depth first search to guide search from source to sink
- Revert to breadth-first search to find your way around obstacles
- Cannot guarantee shortest path

Example



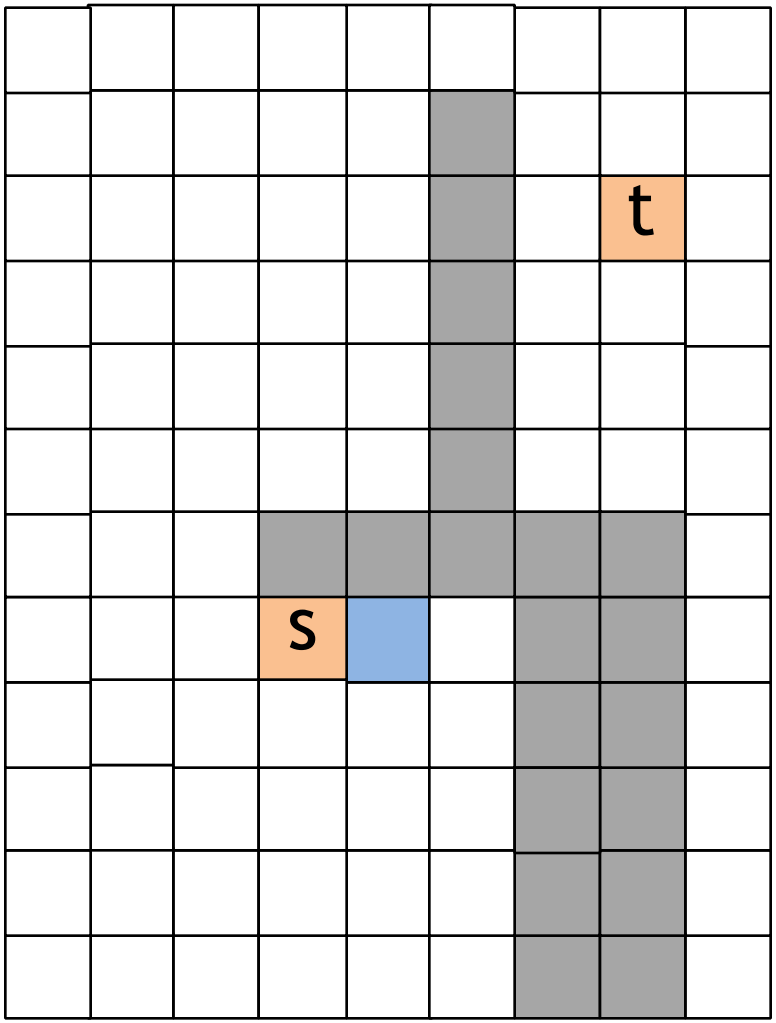
Legend

Wavefront

Cells discovered by depth-first search

Cells expanded by breadth-first search

Depth-First Search



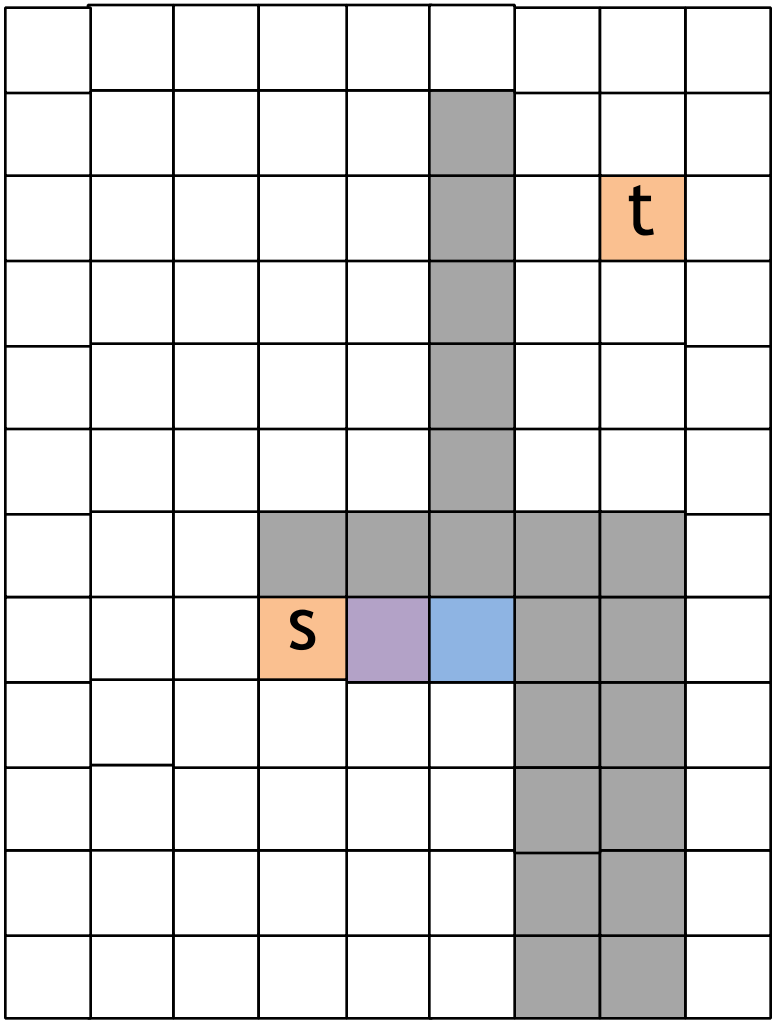
Legend

Wavefront

Cells discovered by depth-first search

Cells expanded by breadth-first search

Depth-First Search



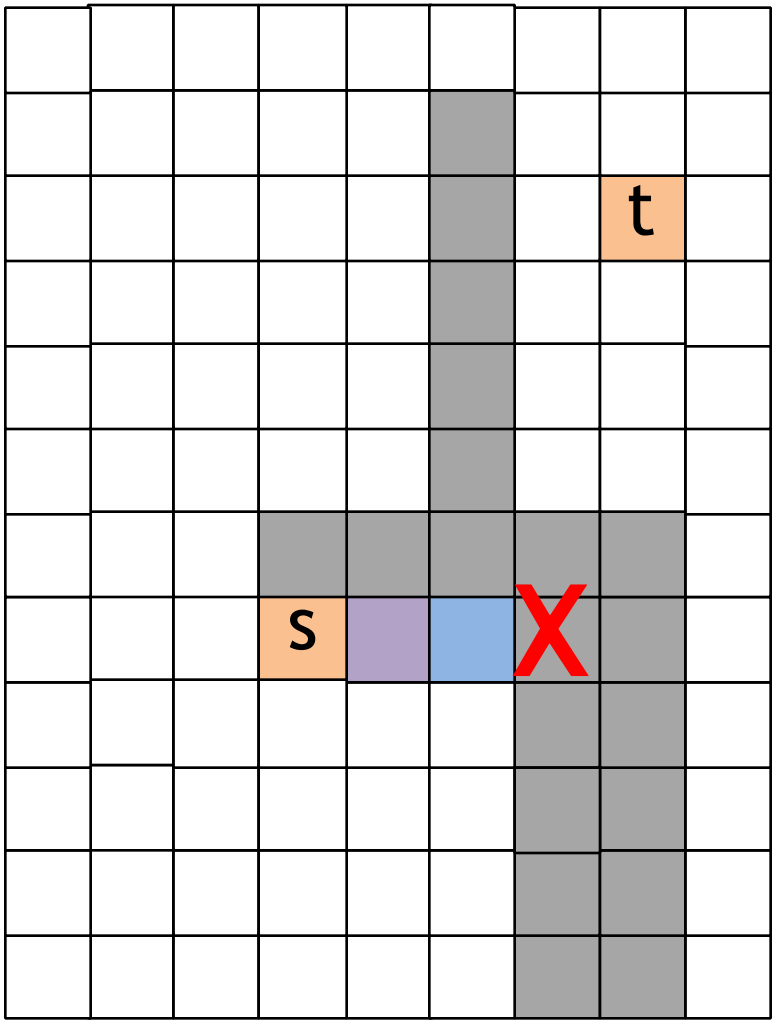
Legend

Wavefront

Cells discovered by depth-first search

Cells expanded by breadth-first search

Depth-First Search Blocked



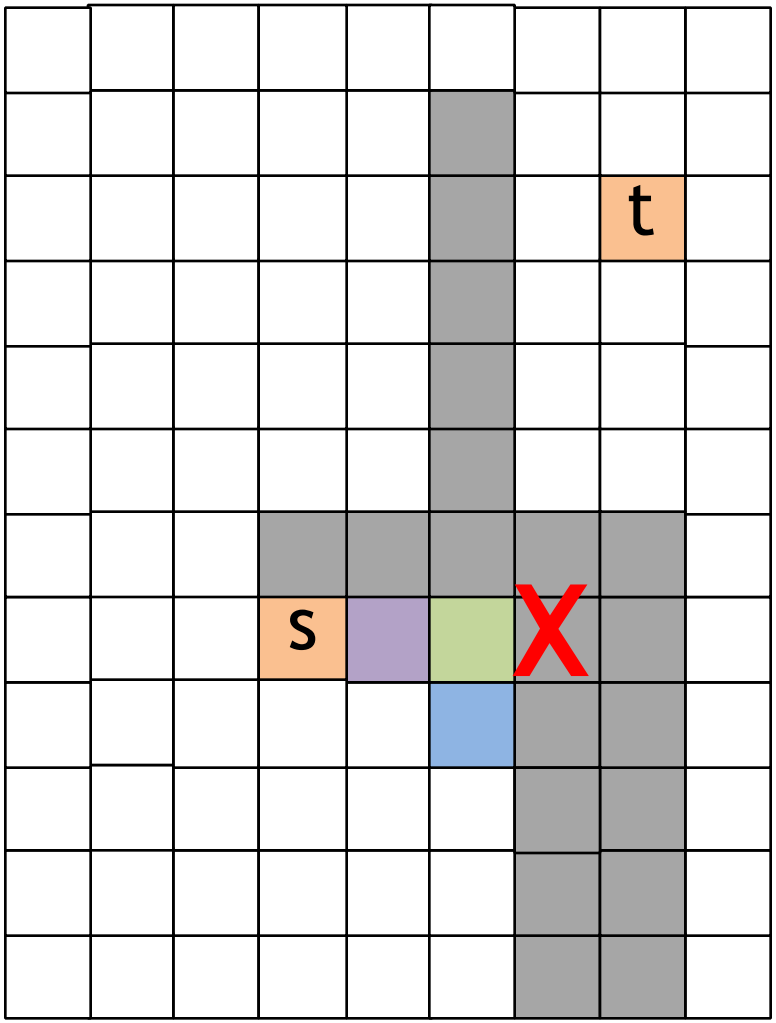
Legend

Wavefront

Cells discovered by depth-first search

Cells expanded by breadth-first search

Switch to Breadth-First Search



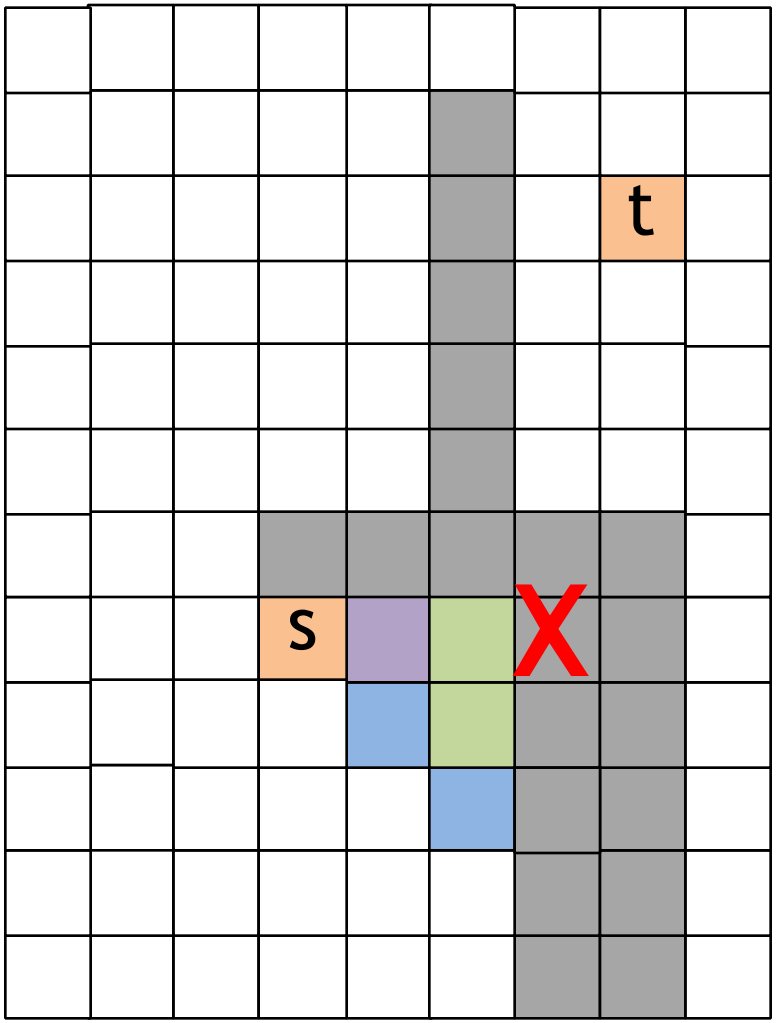
Legend

Wavefront

Cells discovered by depth-first search

Cells expanded by breadth-first search

Breadth-First Search



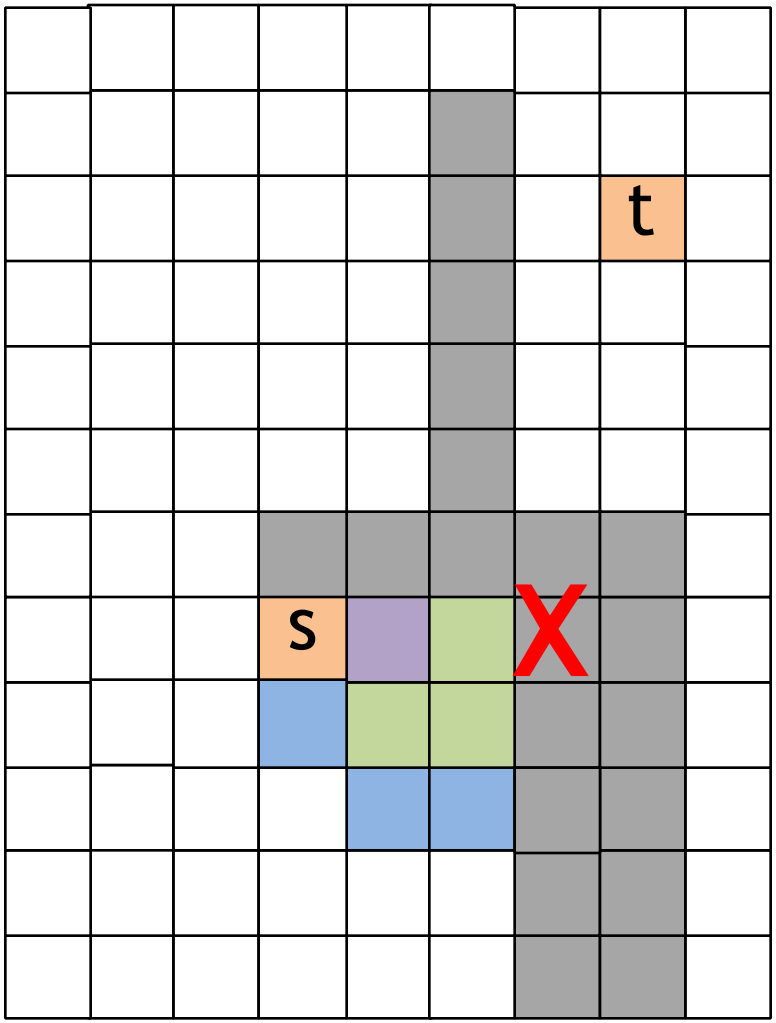
Legend

Wavefront

Cells discovered by depth-first search

Cells expanded by breadth-first search

Breadth-First Search



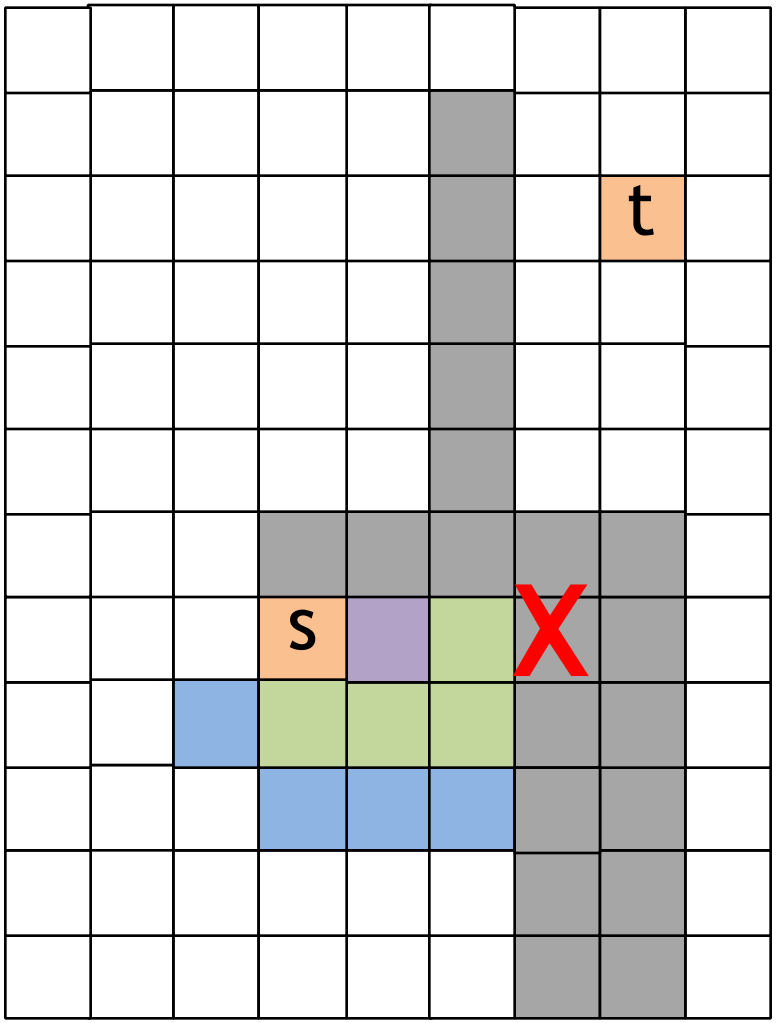
Legend

Wavefront

Cells discovered by depth-first search

Cells expanded by breadth-first search

Breadth-First Search



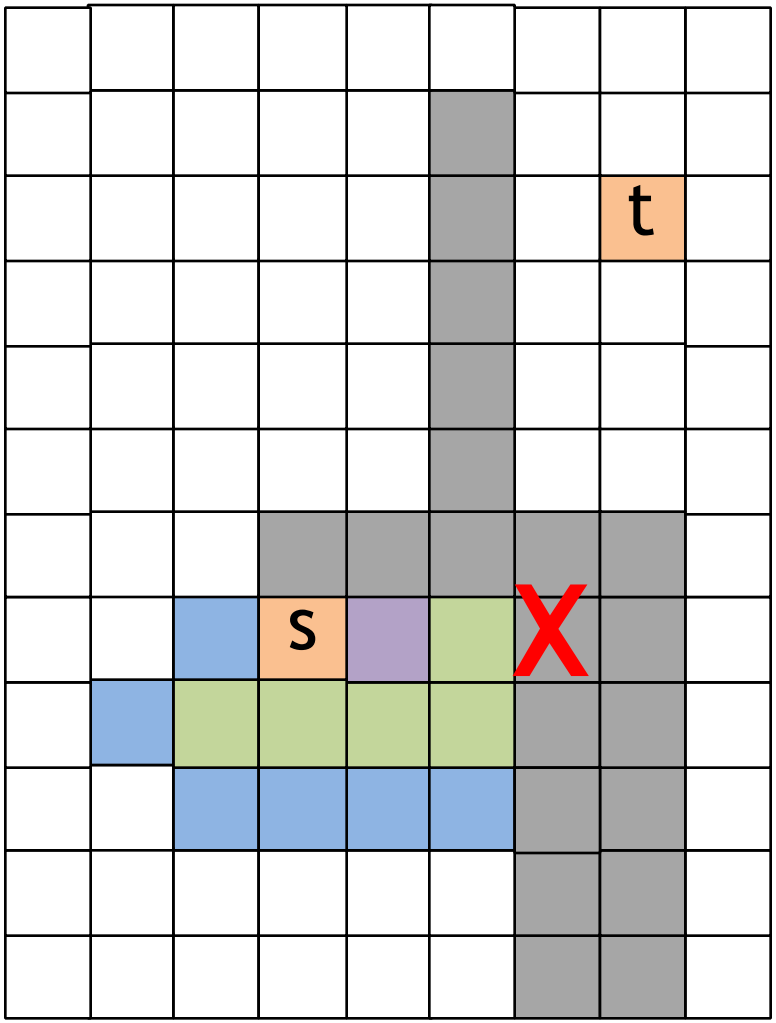
Legend

Wavefront

Cells discovered by depth-first search

Cells expanded by breadth-first search

Breadth-First Search



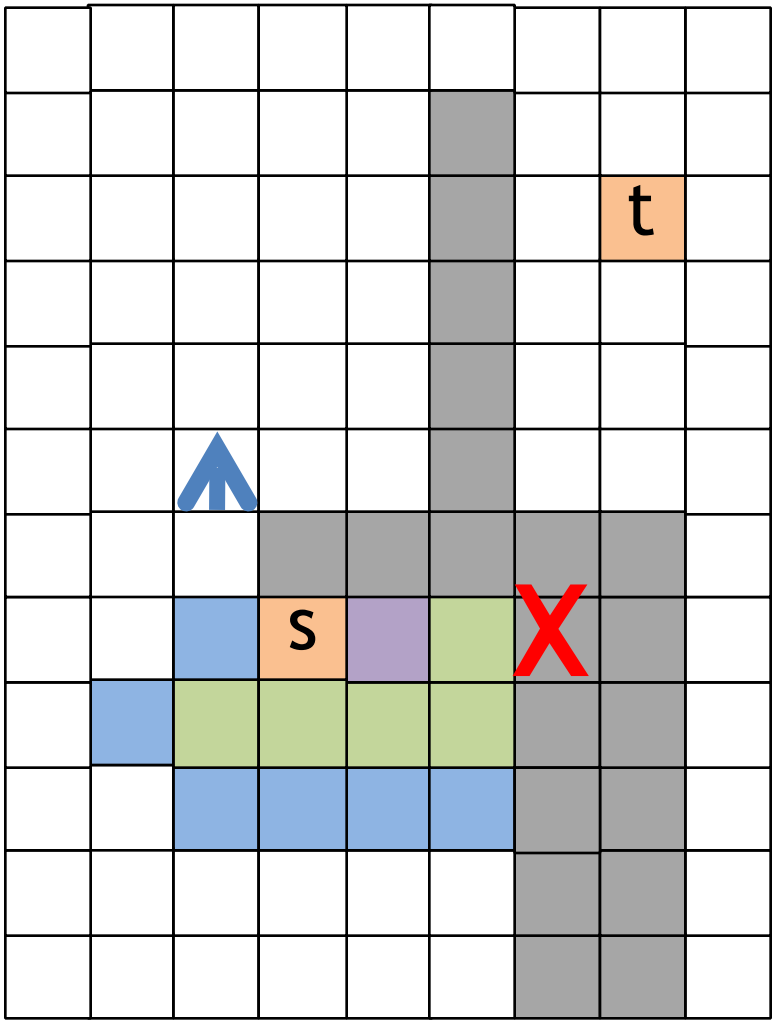
Legend

Wavefront

Cells discovered by depth-first search

Cells expanded by breadth-first search

No Longer Blocked



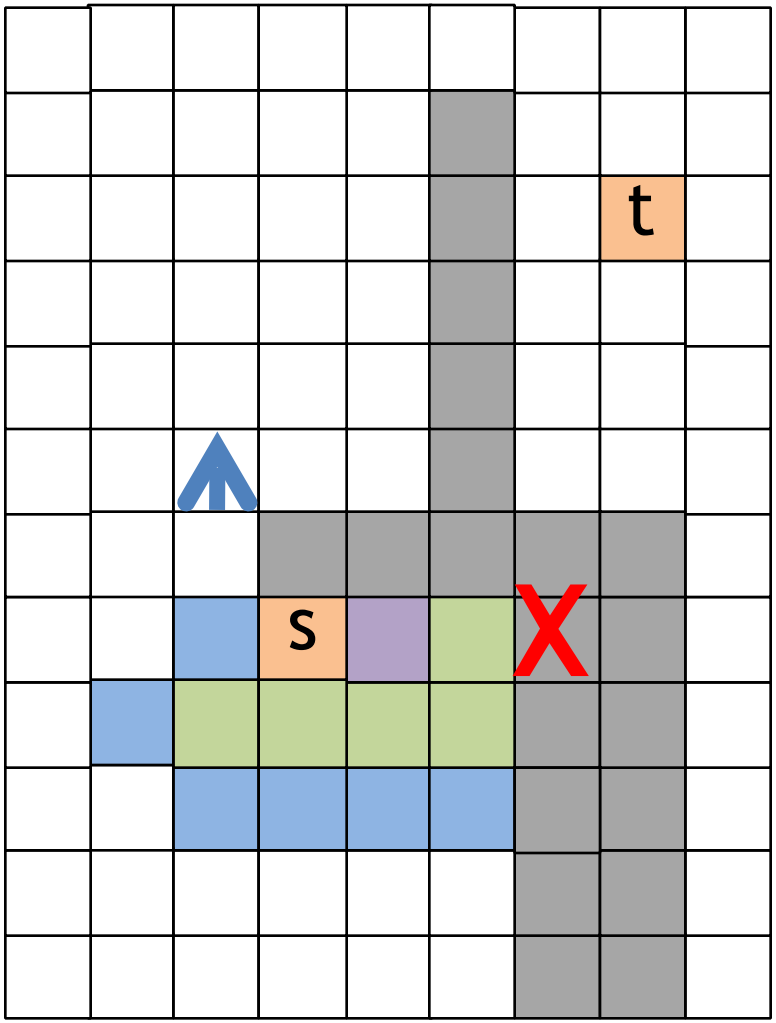
Legend

Wavefront

Cells discovered by depth-first search

Cells expanded by breadth-first search

Switch Back to Depth-First Search



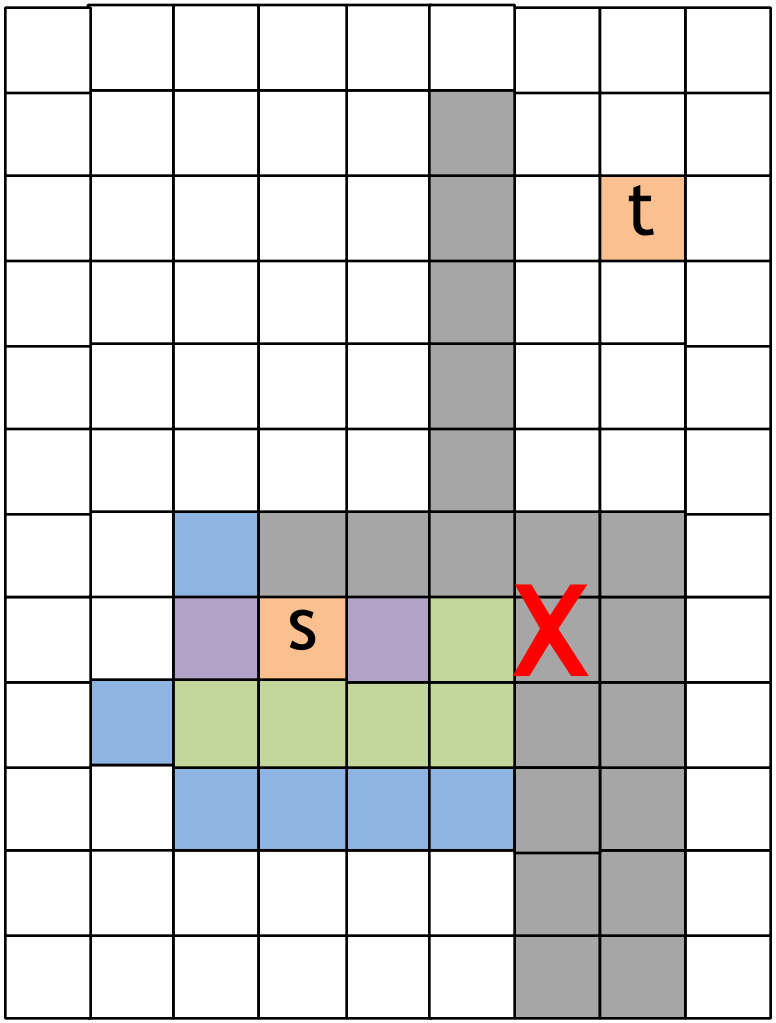
Legend

Wavefront

Cells discovered by depth-first search

Cells expanded by breadth-first search

Depth-First Search



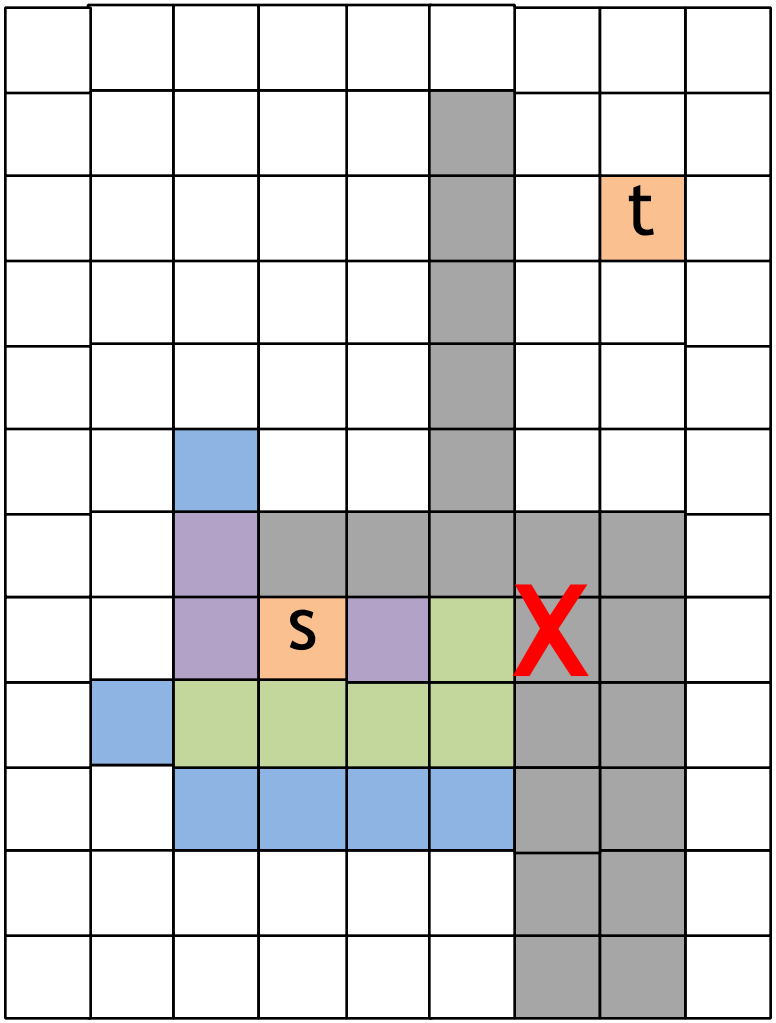
Legend

Wavefront

Cells discovered by depth-first search

Cells expanded by breadth-first search

Depth-First Search



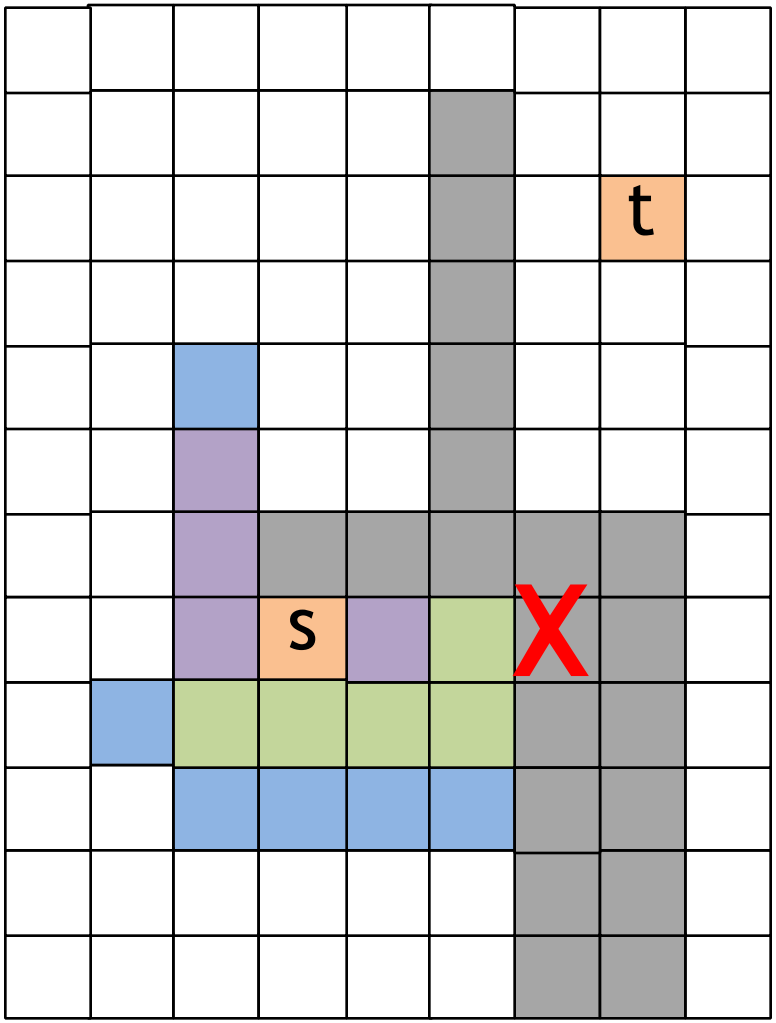
Legend

Wavefront

Cells discovered by depth-first search

Cells expanded by breadth-first search

Depth-First Search



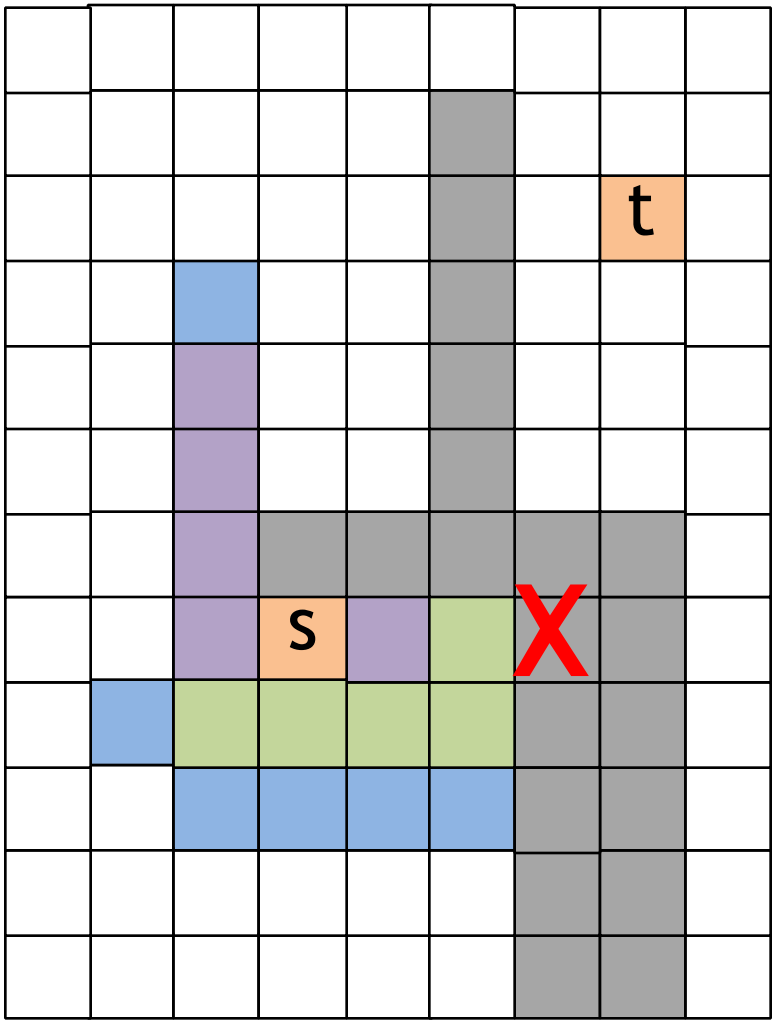
Legend

Wavefront

Cells discovered by depth-first search

Cells expanded by breadth-first search

Depth-First Search



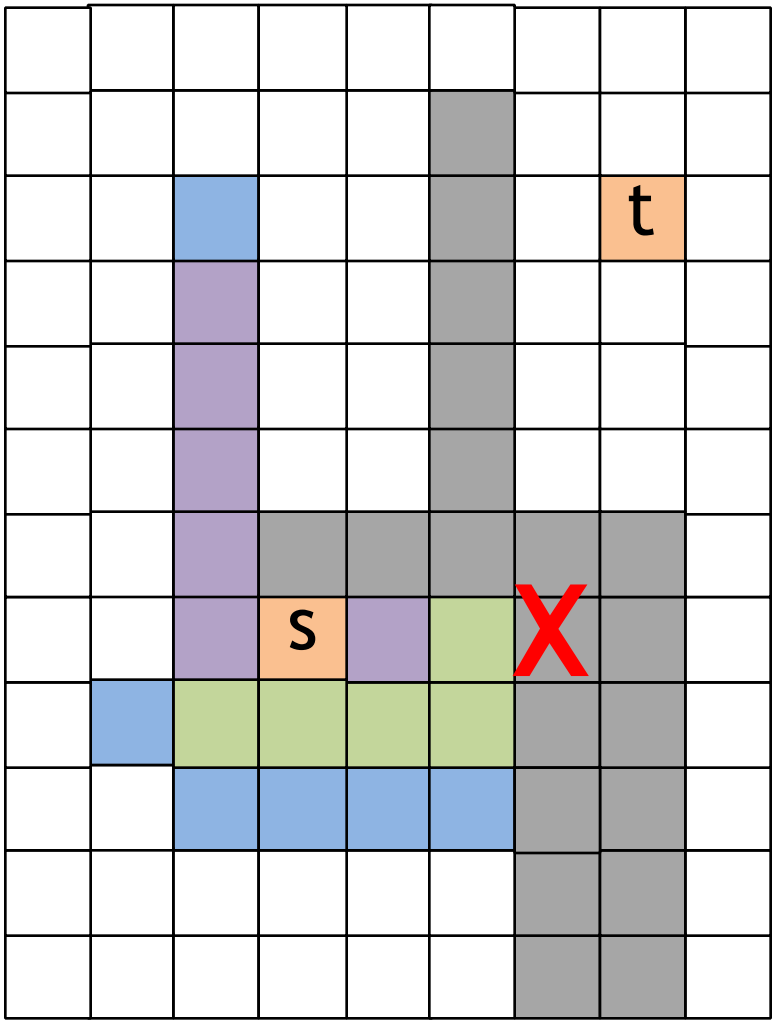
Legend

Wavefront

Cells discovered by depth-first search

Cells expanded by breadth-first search

Depth-First Search



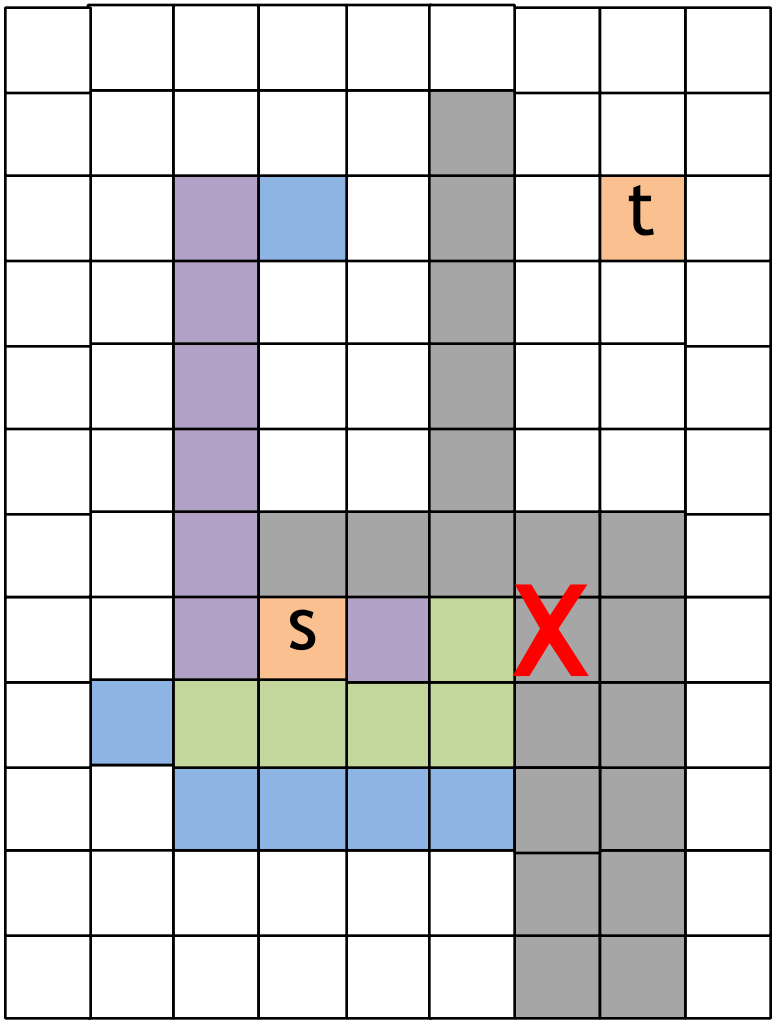
Legend

Wavefront

Cells discovered by depth-first search

Cells expanded by breadth-first search

Depth-First Search



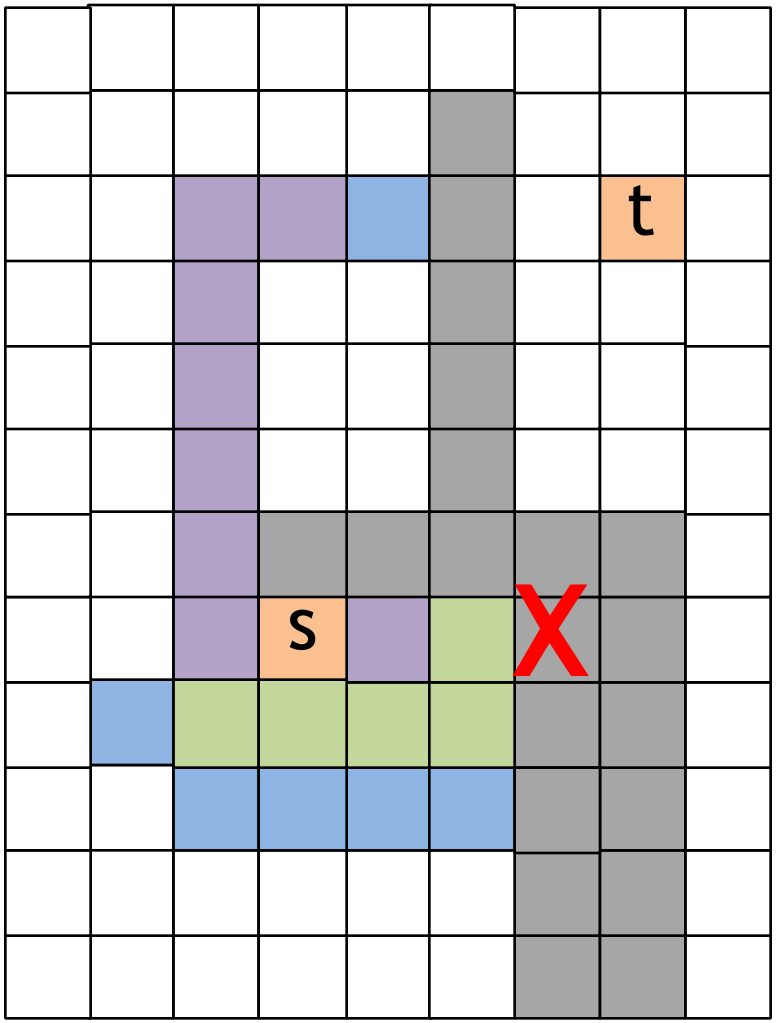
Legend

Wavefront

Cells discovered by depth-first search

Cells expanded by breadth-first search

Depth-First Search



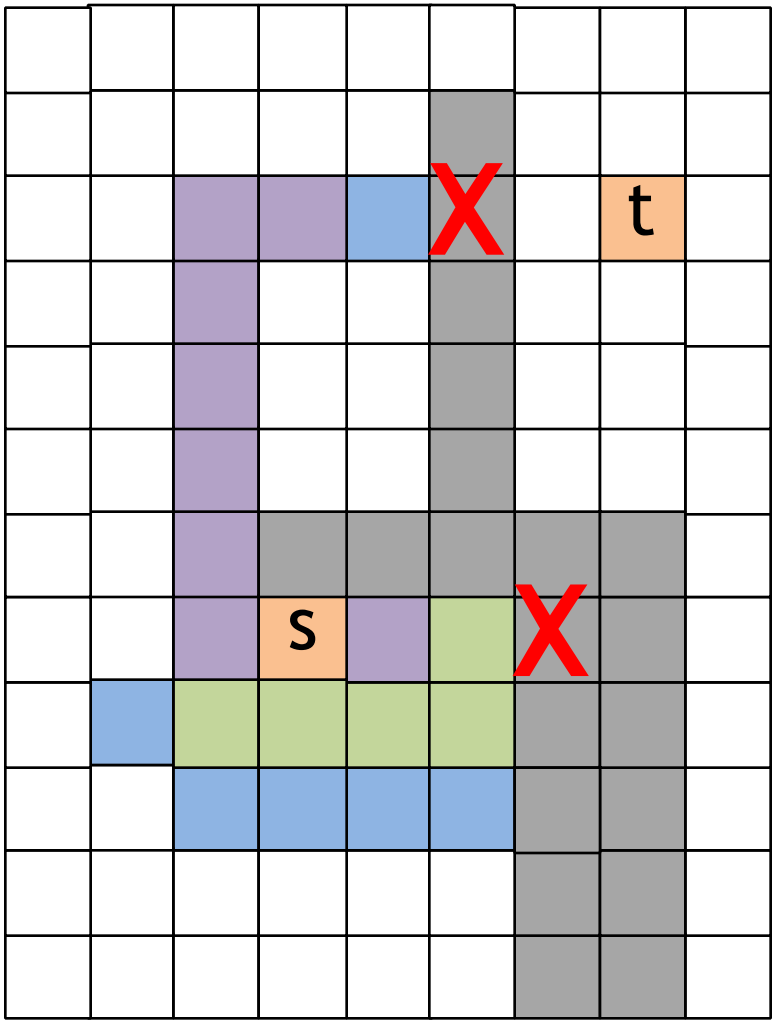
Legend

Wavefront

Cells discovered by depth-first search

Cells expanded by breadth-first search

Depth First Search Blocked



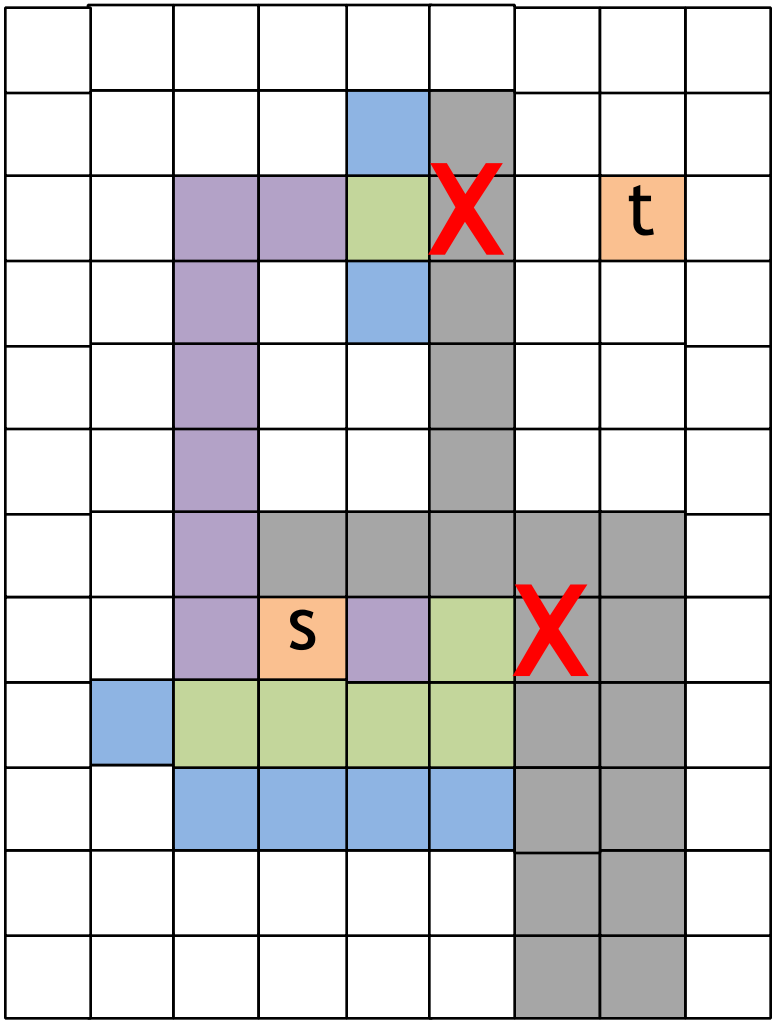
Legend

Wavefront

Cells discovered by depth-first search

Cells expanded by breadth-first search

Switch to Breadth-First Search



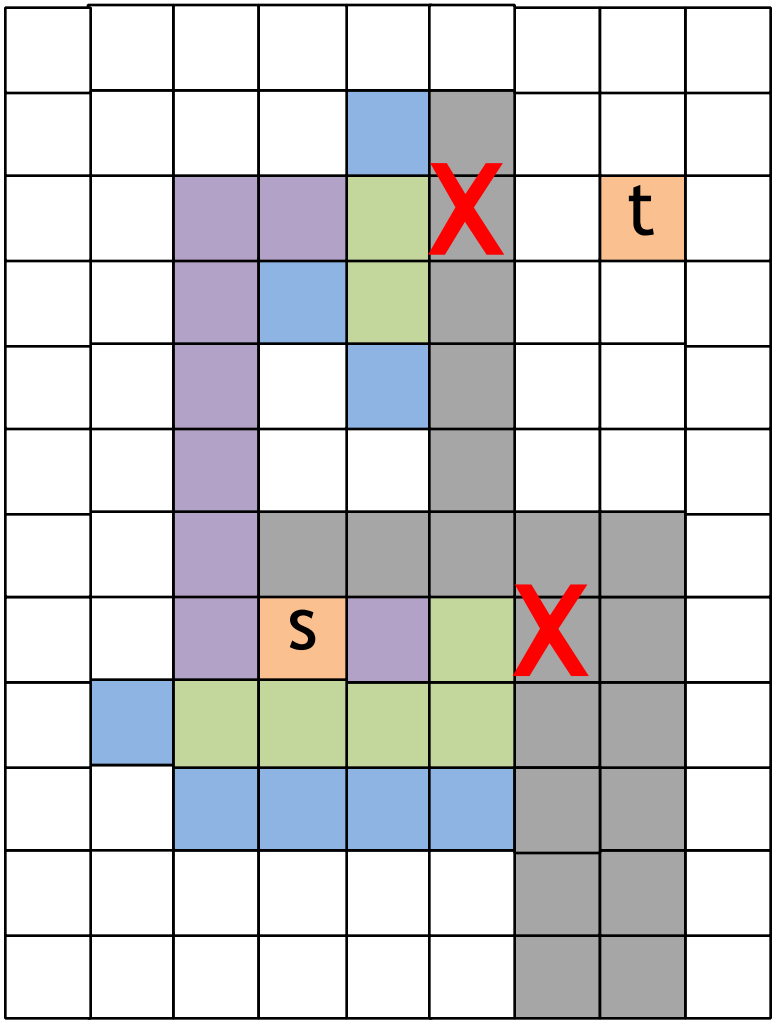
Legend

Wavefront

Cells discovered by depth-first search

Cells expanded by breadth-first search

Breadth-First Search



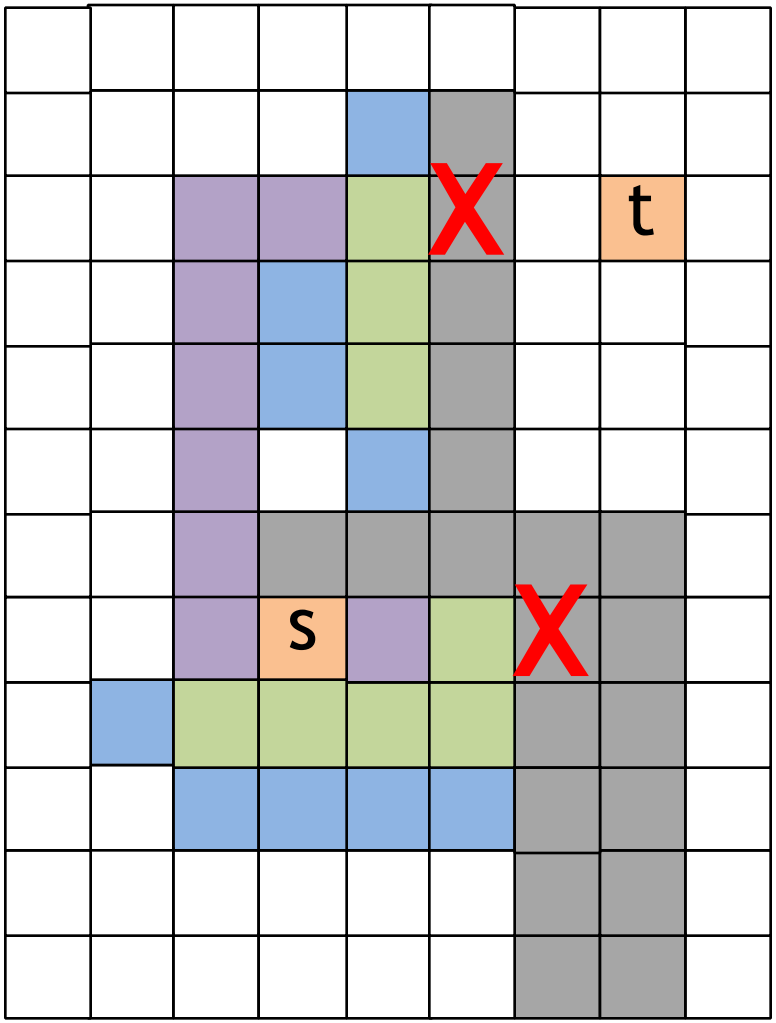
Legend

Wavefront

Cells discovered by depth-first search

Cells expanded by breadth-first search

Breadth-First Search



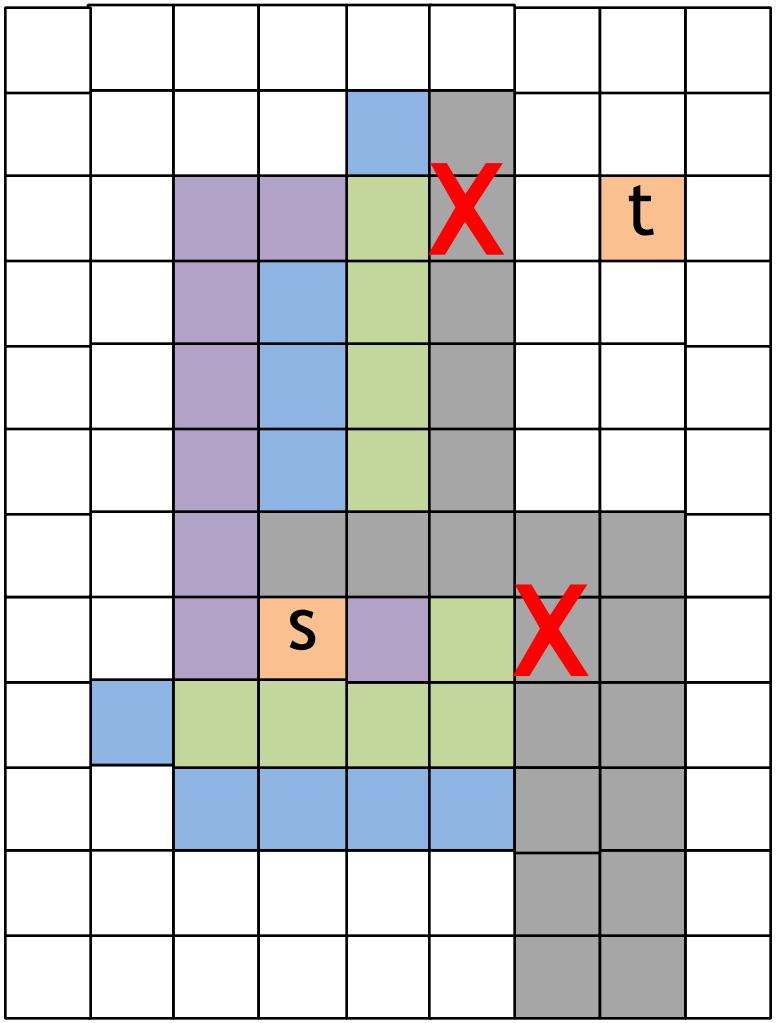
Legend

Wavefront

Cells discovered by depth-first search

Cells expanded by breadth-first search

Breadth-First Search



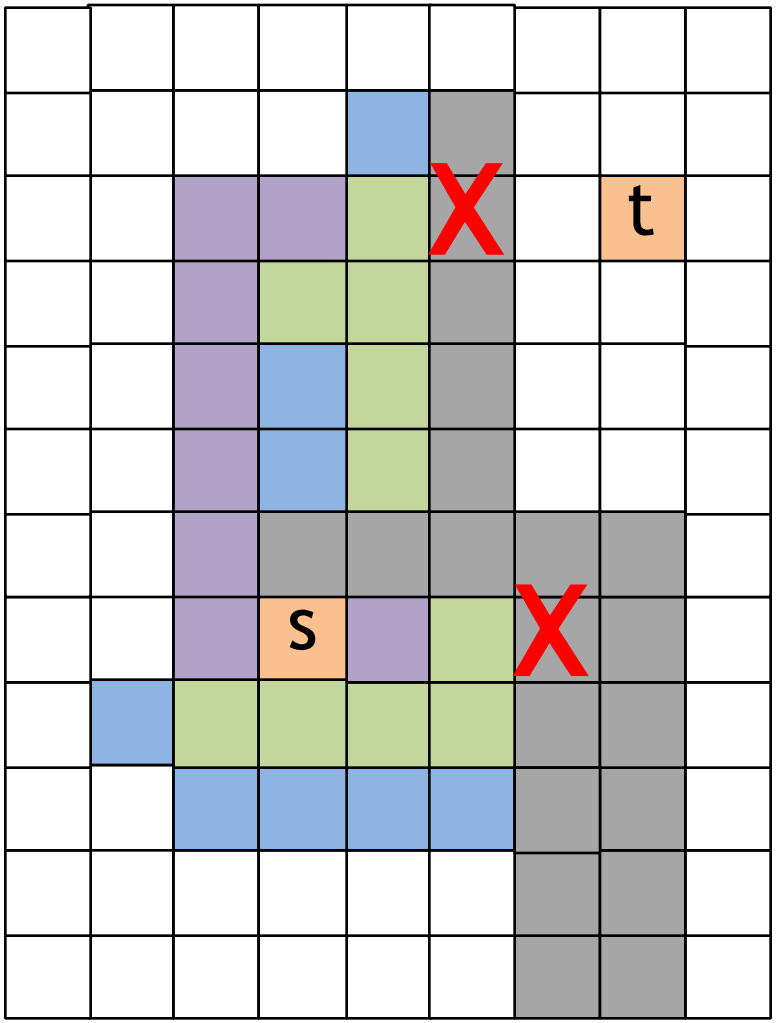
Legend

Wavefront

Cells discovered by depth-first search

Cells expanded by breadth-first search

Breadth-First Search



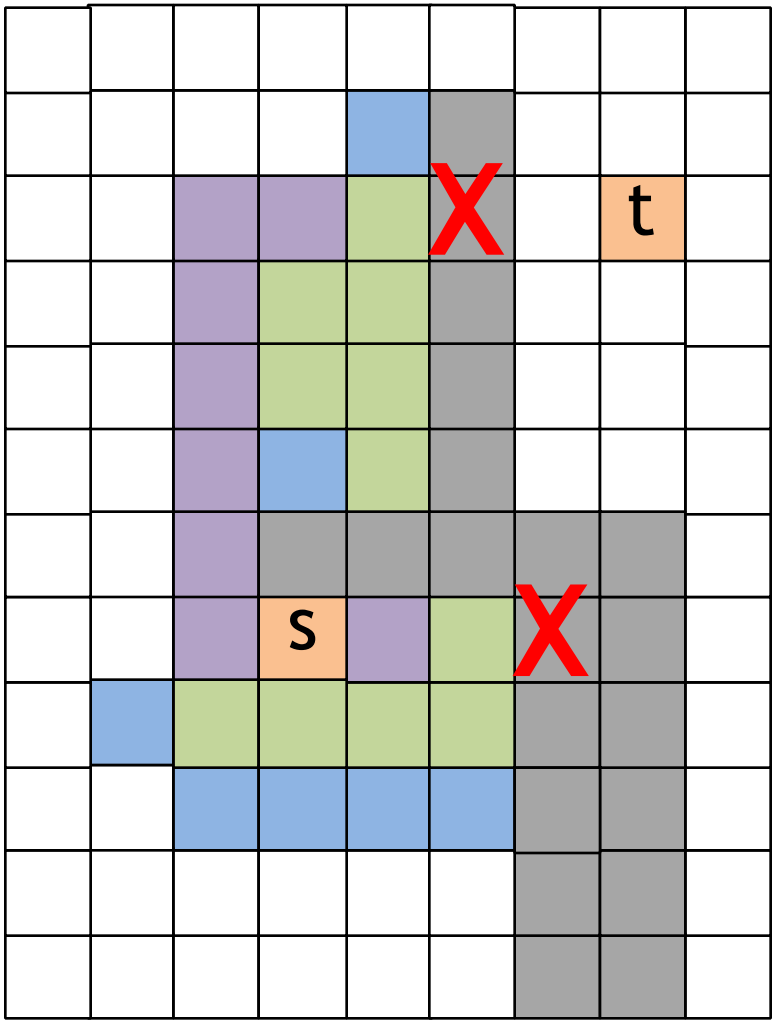
Legend

Wavefront

Cells discovered by depth-first search

Cells expanded by breadth-first search

Breadth-First Search



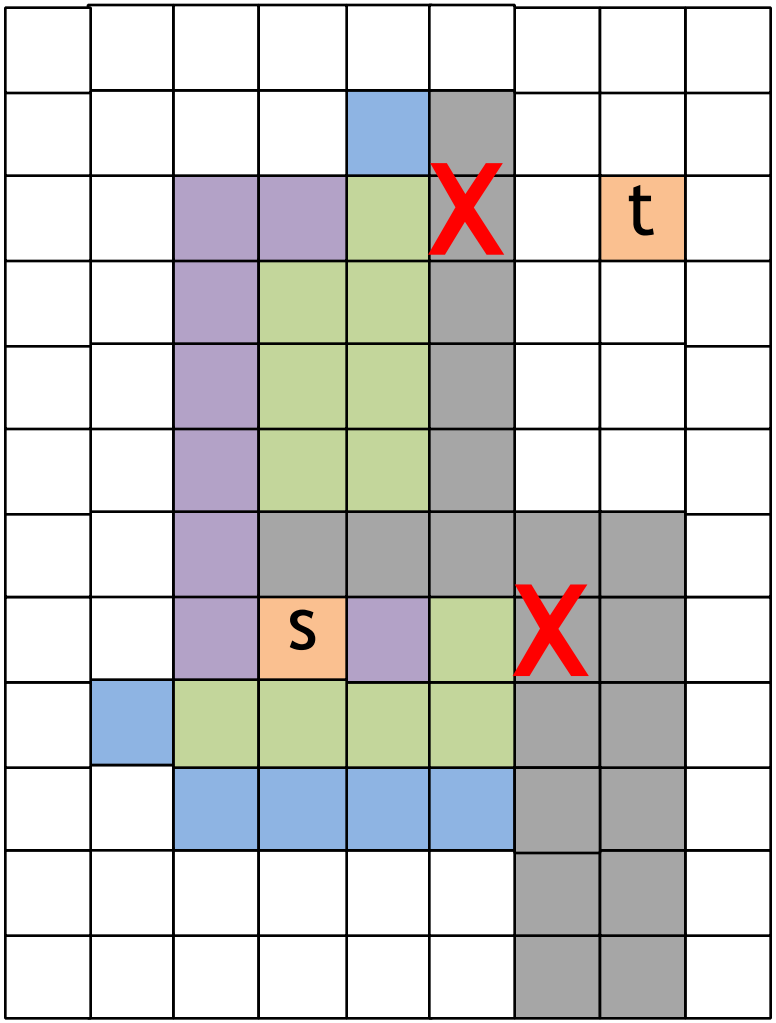
Legend

Wavefront

Cells discovered by depth-first search

Cells expanded by breadth-first search

Breadth-First Search



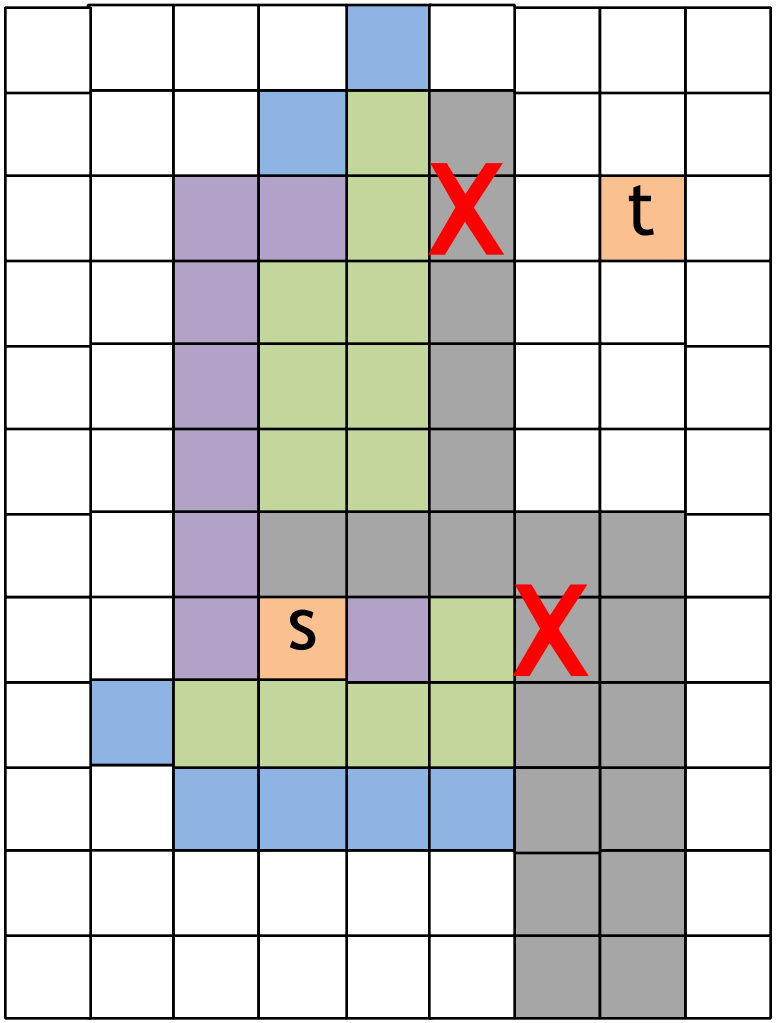
Legend

Wavefront

Cells discovered by depth-first search

Cells expanded by breadth-first search

Breadth-First Search



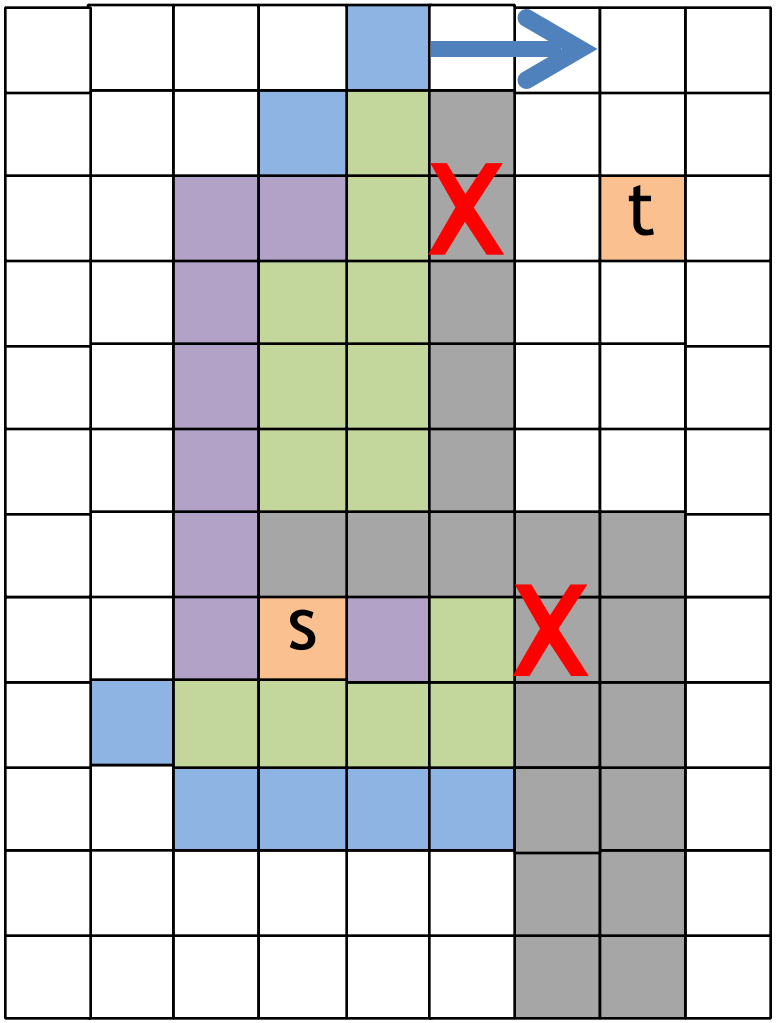
Legend

Wavefront

Cells discovered by depth-first search

Cells expanded by breadth-first search

No Longer Blocked



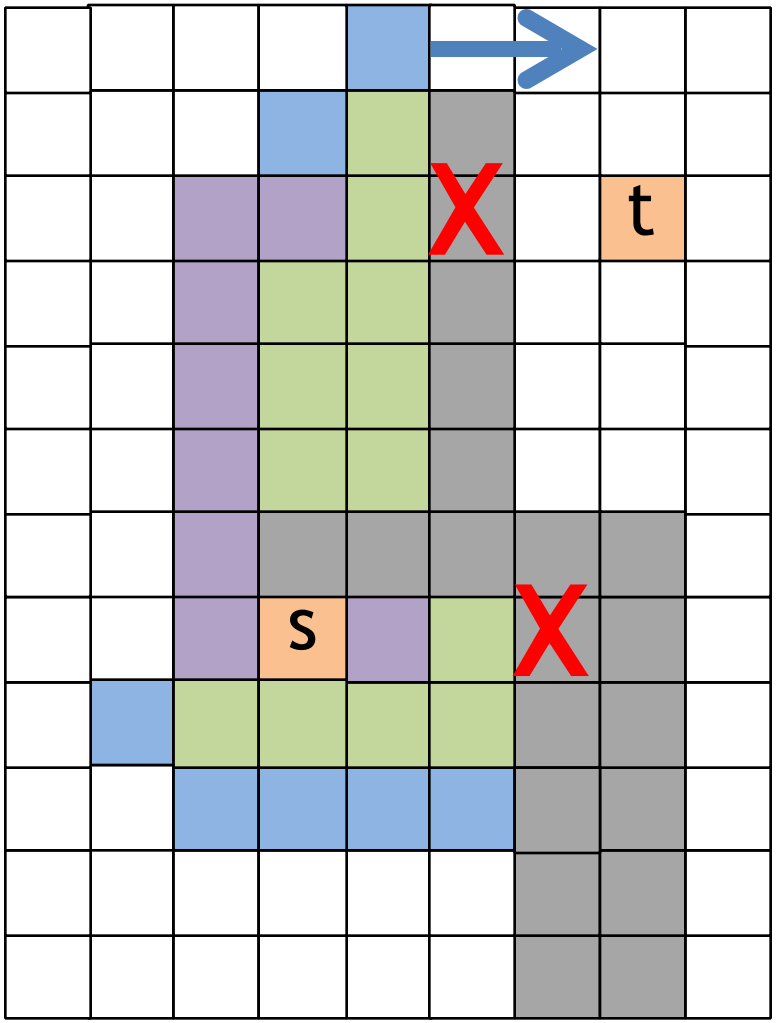
Legend

Wavefront

Cells discovered by depth-first search

Cells expanded by breadth-first search

Switch Back to Depth First Search



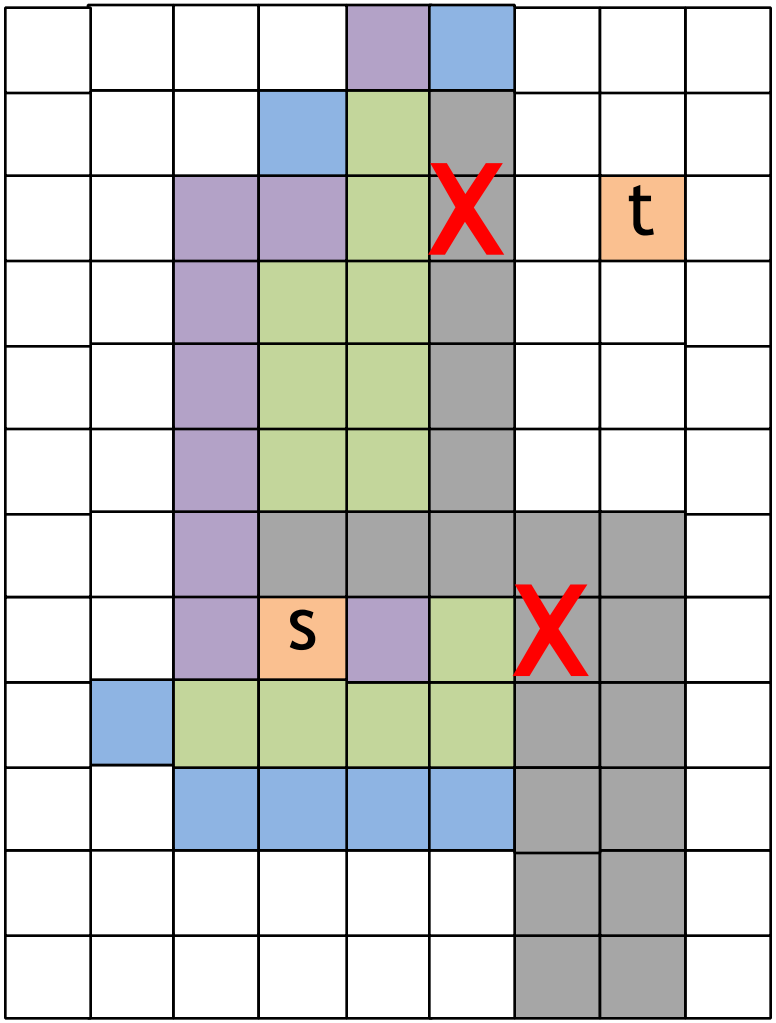
Legend

Wavefront

Cells discovered by depth-first search

Cells expanded by breadth-first search

Depth First Search



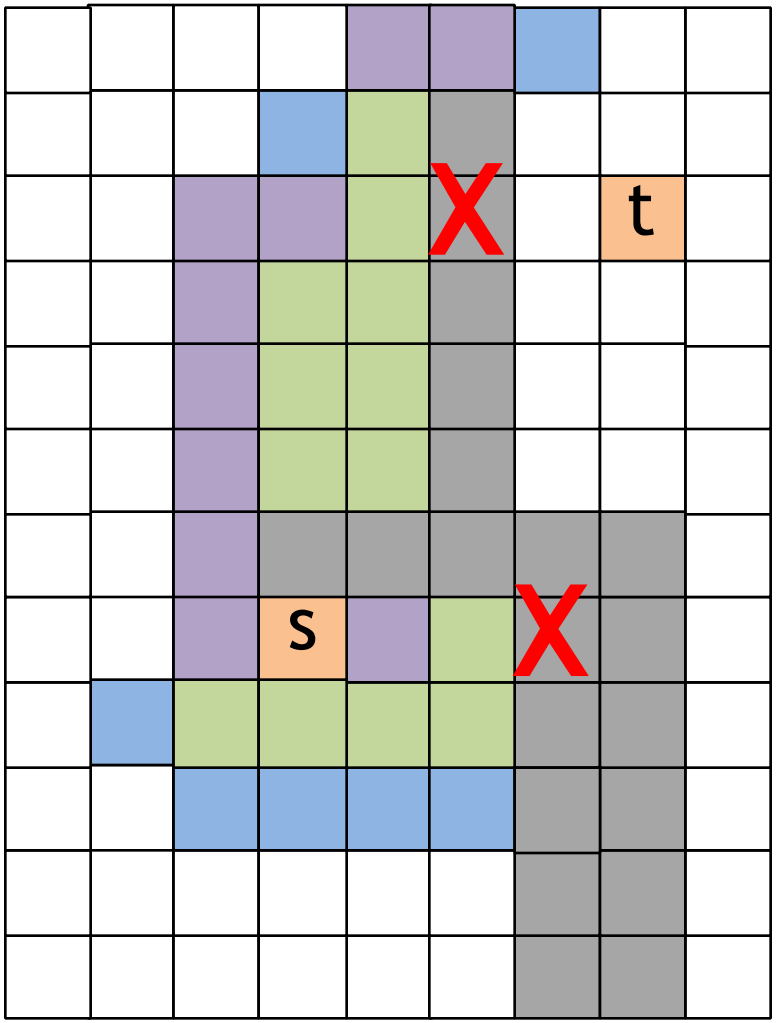
Legend

Wavefront

Cells discovered by depth-first search

Cells expanded by breadth-first search

Depth First Search



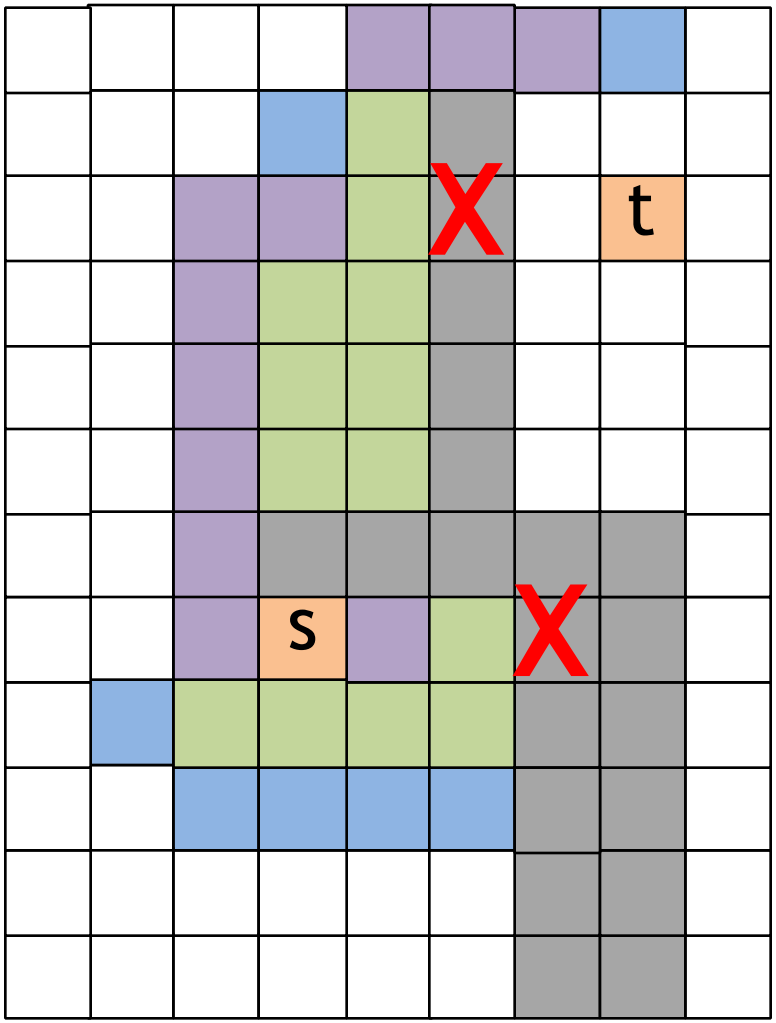
Legend

Wavefront

Cells discovered by depth-first search

Cells expanded by breadth-first search

Depth First Search



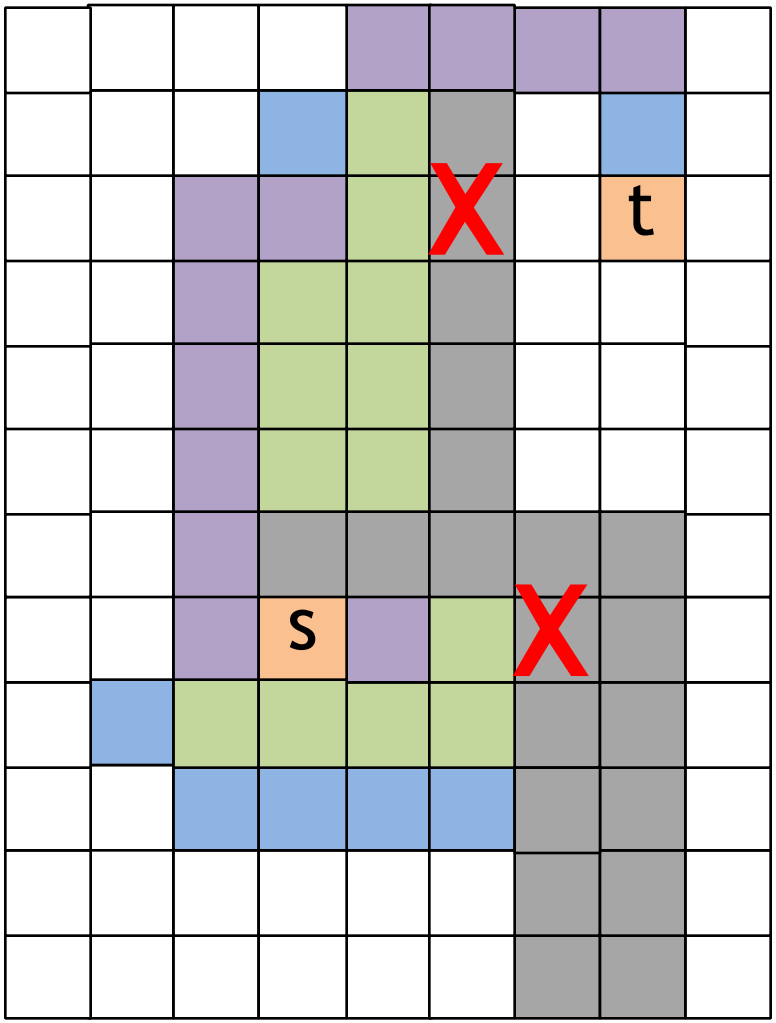
Legend

Wavefront

Cells discovered by depth-first search

Cells expanded by breadth-first search

Depth First Search



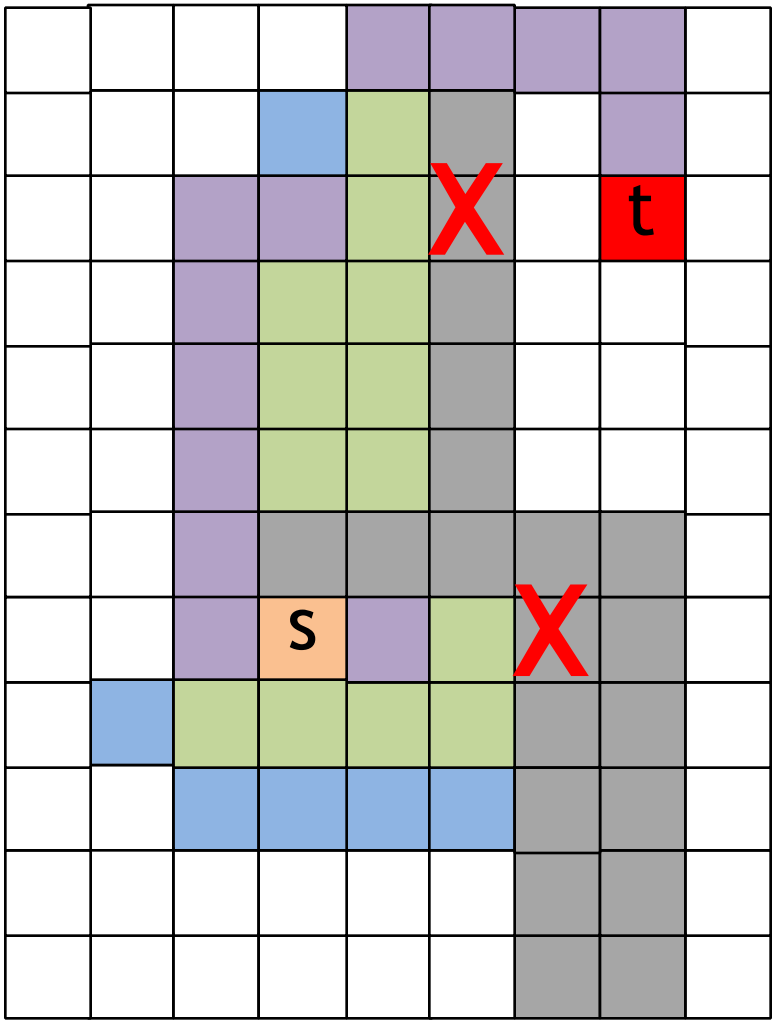
Legend

Wavefront

Cells discovered by depth-first search

Cells expanded by breadth-first search

Depth First Search



- Details for path retracing are omitted

Legend

Wavefront

Cells discovered by depth-first search

Cells expanded by breadth-first search