Wiring Assistant

Selected Fun Problems of the ACM Programming Contest

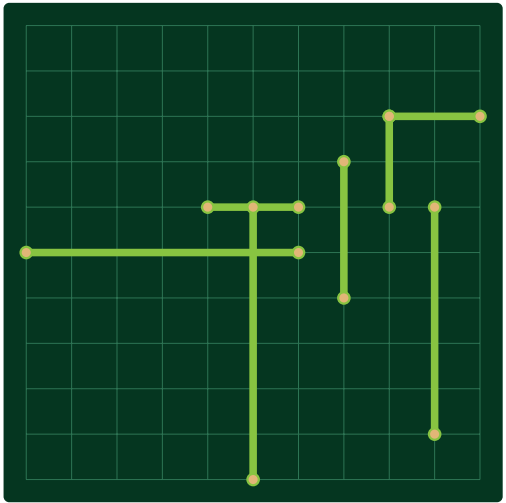
Samuel Füßinger

Database Systems
Department of Computer Science
University of Tübingen

21.06.2024

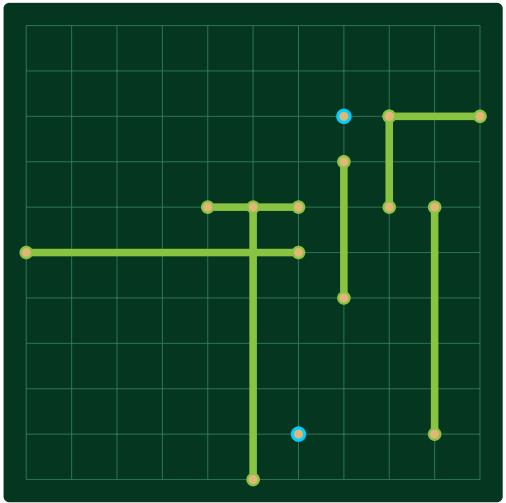
The Task

- Grid-based circuit board
- Some existing wires
- Two points to connect
- Wires cannot be stacked
- Intersections allowed but costly
- What is the minimum possible cost?



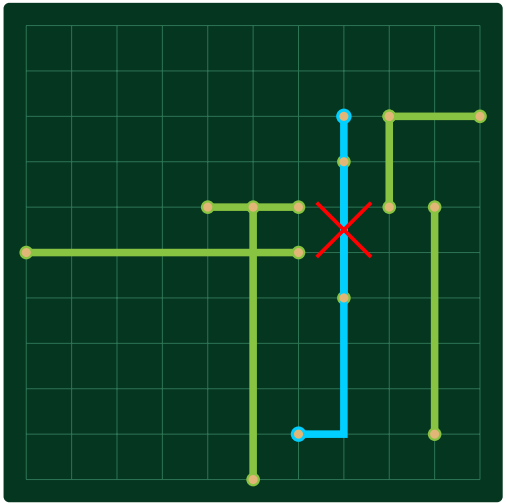
The Task

- Grid-based circuit board
- Some existing wires
- Two points to connect
- Wires cannot be stacked
- Intersections allowed but costly
- What is the minimum possible cost?



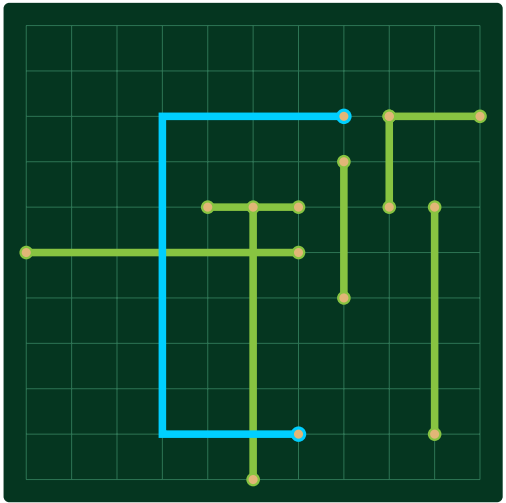
The Task

- Grid-based circuit board
- Some existing wires
- Two points to connect
- Wires cannot be stacked
- Intersections allowed but costly
- What is the minimum possible cost?



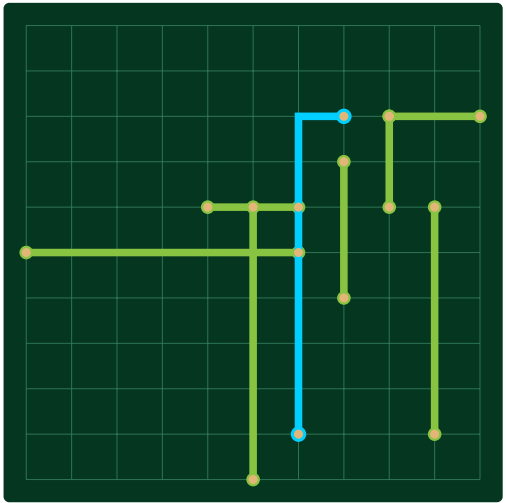
The Task

- Grid-based circuit board
- Some existing wires
- Two points to connect
- Wires cannot be stacked
- Intersections allowed but costly
- What is the minimum possible cost?



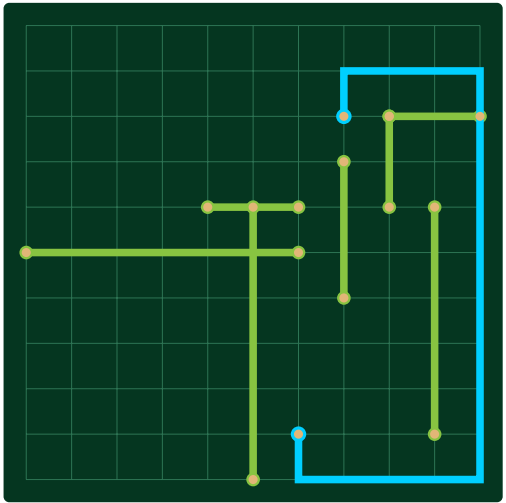
The Task

- Grid-based circuit board
- Some existing wires
- Two points to connect
- Wires cannot be stacked
- Intersections allowed but costly
- What is the minimum possible cost?



The Task

- Grid-based circuit board
- Some existing wires
- Two points to connect
- Wires cannot be stacked
- Intersections allowed but costly
- What is the minimum possible cost?



Input Format

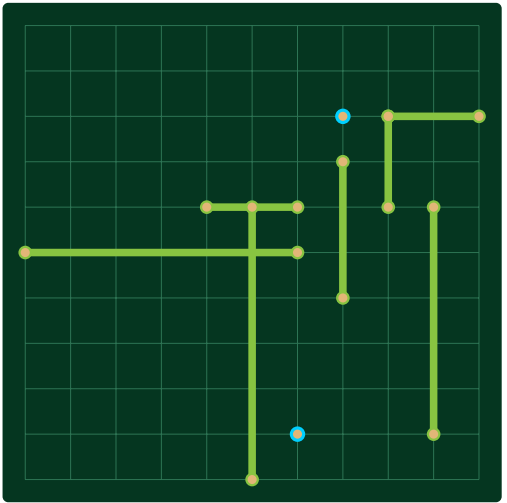
```
7  11
M   S

5  0  5  6  0  5  6  5  9  1  9  6  7  4  7  7  …
```
Coordinates of wire end points

```
7  8  6  1
```
Coordinates of points to connect

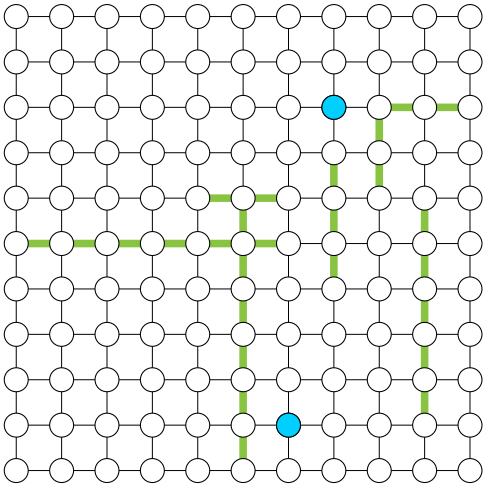
M := Number of wires

S := Grid size



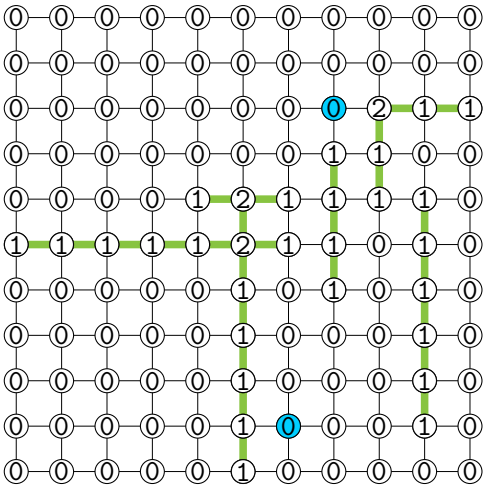
Naive Approach

- Graph problem
- Node cost = Number of wires
- Remove edges of existing wires
- Use any pathfinding algorithm



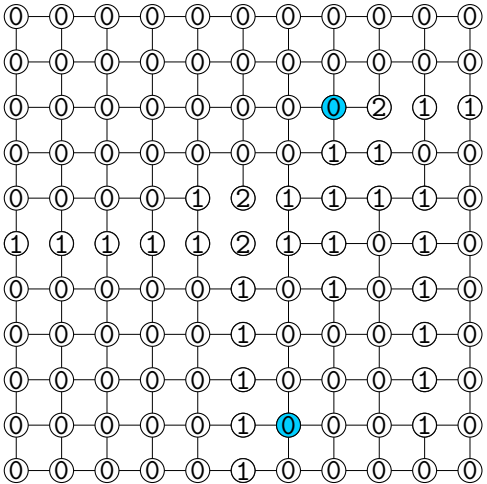
Naive Approach

- Graph problem
- Node cost = Number of wires
- Remove edges of existing wires
- Use any pathfinding algorithm



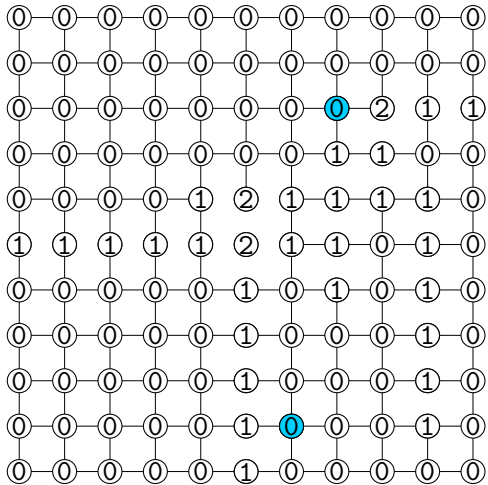
Naive Approach

- Graph problem
- Node cost = Number of wires
- Remove edges of existing wires
- Use any pathfinding algorithm



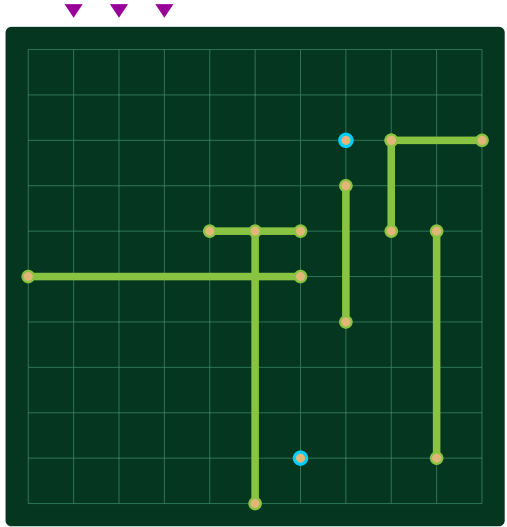
Problem

- Big problem: graph size
- Up to $10^9 \times 10^9$ nodes
- Searching through 10^{18} nodes could take literal decades!
- But ≤ 99 wires
- We need to reduce the graph



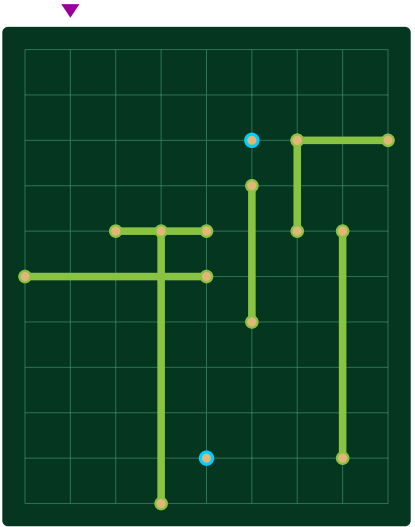
Reduction Idea

- Identical neighboring columns/rows have no effect
  $\Rightarrow$ Remove all but one
- How to implement?
- Is this good enough?



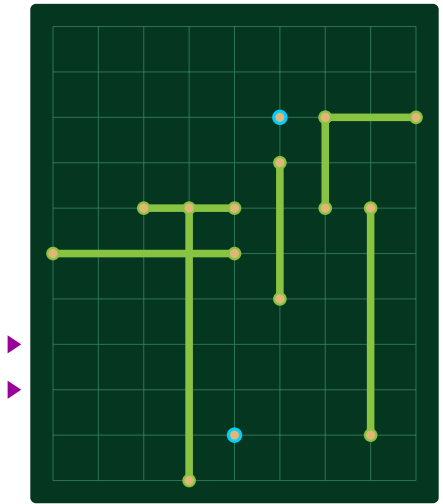
Reduction Idea

- Identical neighboring columns/rows have no effect
  $\Rightarrow$ Remove all but one
- How to implement?
- Is this good enough?



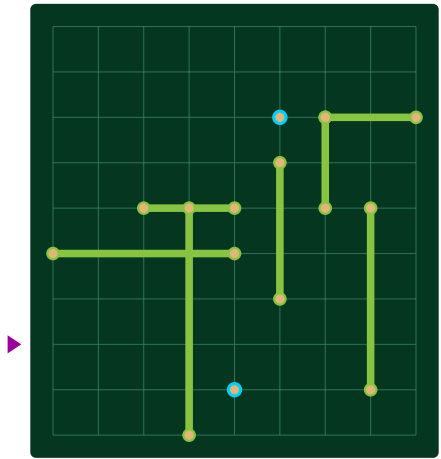
Reduction Idea

- Identical neighboring columns/rows have no effect
  $\Rightarrow$ Remove all but one
- How to implement?
- Is this good enough?



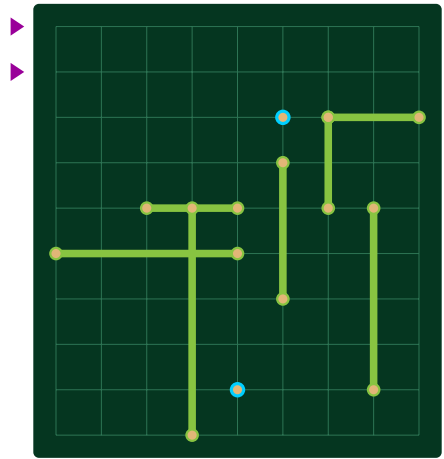
Reduction Idea

- Identical neighboring columns/rows have no effect
  $\Rightarrow$ Remove all but one
- How to implement?
- Is this good enough?



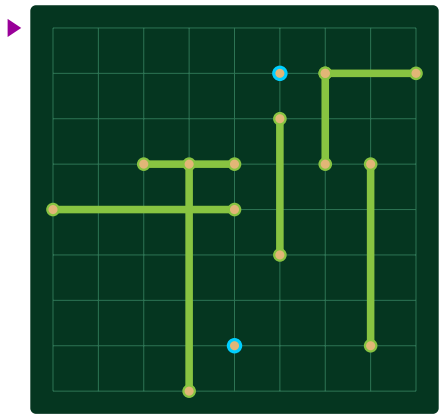
Reduction Idea

- Identical neighboring columns/rows have no effect
  $\Rightarrow$ Remove all but one
- How to implement?
- Is this good enough?



Reduction Idea

- Identical neighboring columns/rows have no effect
  $\Rightarrow$ Remove all but one
- How to implement?
- Is this good enough?



Reduction Implementation

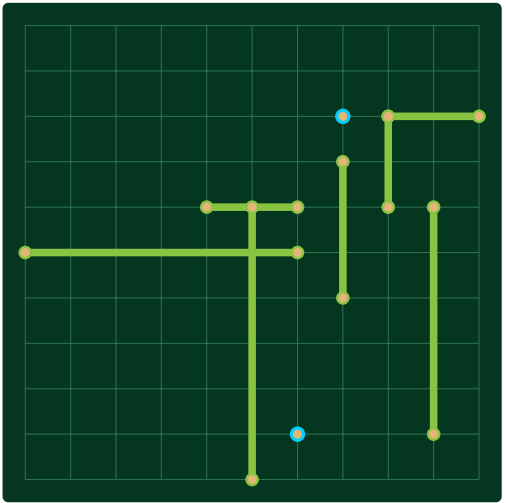
5 0 5 6 0 5 6 5 9 1 9 6 7 4 7 7 …
Coordinates of wire end points

7 8 6 1
Coordinates of points to connect

Split into arrays, include bounds and sort:

```
xs = [-1, 0, 4, 5, 5, 6, 6, 6, 7,
      7, 7, 8, 8, 9, 9, 10, 11]
ys = [-1, 0, 1, 1, 4, 5, 5, 6, 6,
      6, 6, 6, 7, 8, 8, 8, 11]
```

Search for gaps ≥ 3



Reduction Implementation

5 0 5 6 0 5 6 5 9 1 9 6 7 4 7 7 …
Coordinates of wire end points

7 8 6 1
Coordinates of points to connect

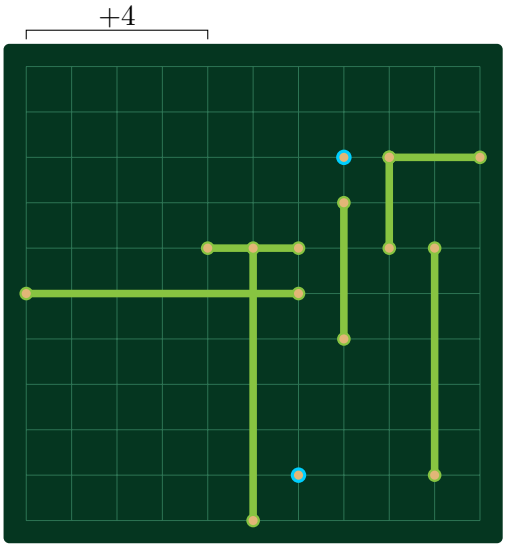
Split into arrays, include bounds and sort:

```
xs = [-1, 0, 4, 5, 5, 6, 6, 6, 7,
      7, 7, 8, 8, 9, 9, 10, 11]

ys = [-1, 0, 1, 1, 4, 5, 5, 6, 6,
      6, 6, 6, 7, 8, 8, 8, 11]
```

$+4$

Search for gaps ≥ 3



Reduction Implementation

5 0 5 6 0 5 6 5 9 1 9 6 7 4 7 7 …
Coordinates of wire end points

7 8 6 1
Coordinates of points to connect

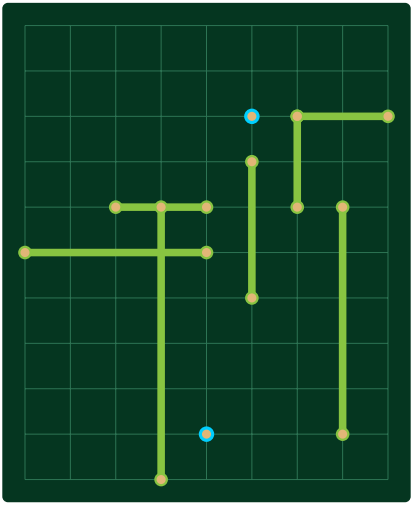
Split into arrays, include bounds and sort:

```
xs = [-1, 0, 4, 5, 5, 6, 6, 6, 7,
      7, 7, 8, 8, 9, 9, 10, 11]
```

+4

```
ys = [-1, 0, 1, 1, 4, 5, 5, 6, 6,
      6, 6, 6, 7, 8, 8, 8, 11]
```

Search for gaps ≥ 3



Reduction Implementation

5 0 5 6 0 5 6 5 9 1 9 6 7 4 7 7 …
Coordinates of wire end points

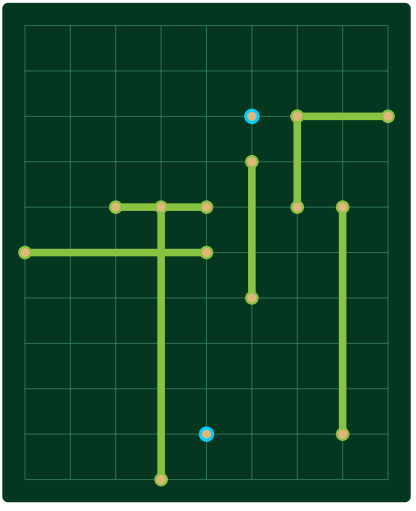
7 8 6 1
Coordinates of points to connect

Split into arrays, include bounds and sort:

```
xs = [-1, 0, 2, 3, 3, 4, 4, 4, 5,
      5, 5, 6, 6, 7, 7, 8, 9]

ys = [-1, 0, 1, 1, 4, 5, 5, 6, 6,
      6, 6, 6, 7, 8, 8, 8, 11]
```

Search for gaps ≥ 3



Reduction Implementation

5 0 5 6 0 5 6 5 9 1 9 6 7 4 7 7 …
Coordinates of wire end points

7 8 6 1
Coordinates of points to connect

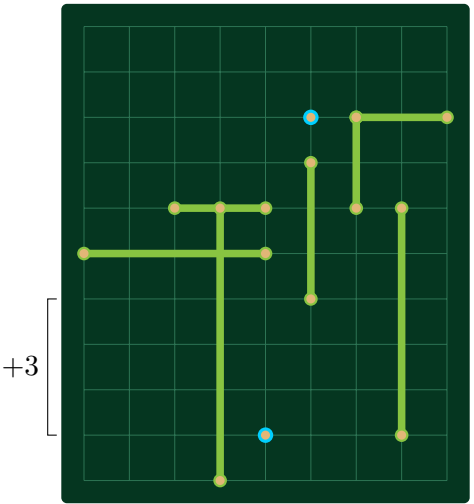
Split into arrays, include bounds and sort:

```
xs = [-1, 0, 2, 3, 3, 4, 4, 4, 5,
      5, 5, 6, 6, 7, 7, 8, 9]

ys = [-1, 0, 1, 1, 4, 5, 5, 6, 6,
      6, 6, 6, 7, 8, 8, 8, 11]
```

$+3$

Search for gaps ≥ 3



Reduction Implementation

5 0 5 6 0 5 6 5 9 1 9 6 7 4 7 7 …
Coordinates of wire end points

7 8 6 1
Coordinates of points to connect

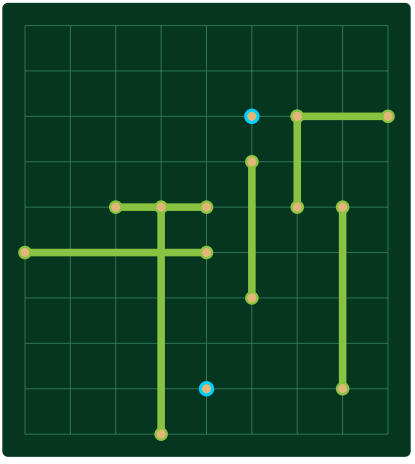
Split into arrays, include bounds and sort:

```
xs = [-1, 0, 2, 3, 3, 4, 4, 4, 5,
      5, 5, 6, 6, 7, 7, 8, 9]

ys = [-1, 0, 1, 1, 4, 5, 5, 6, 6,
      6, 6, 6, 7, 8, 8, 8, 11]
```

(+3)

Search for gaps ≥ 3



Reduction Implementation

5 0 5 6 0 5 6 5 9 1 9 6 7 4 7 7 …
Coordinates of wire end points

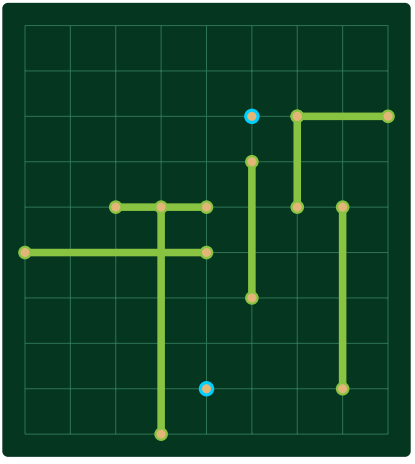
7 8 6 1
Coordinates of points to connect

Split into arrays, include bounds and sort:

```
xs = [-1, 0, 2, 3, 3, 4, 4, 4, 5,
      5, 5, 6, 6, 7, 7, 8, 9]

ys = [-1, 0, 1, 1, 3, 4, 4, 5, 5,
      5, 5, 5, 6, 7, 7, 7, 10]
```

Search for gaps ≥ 3



Reduction Implementation

5 0 5 6 0 5 6 5 9 1 9 6 7 4 7 7 …
Coordinates of wire end points

7 8 6 1
Coordinates of points to connect

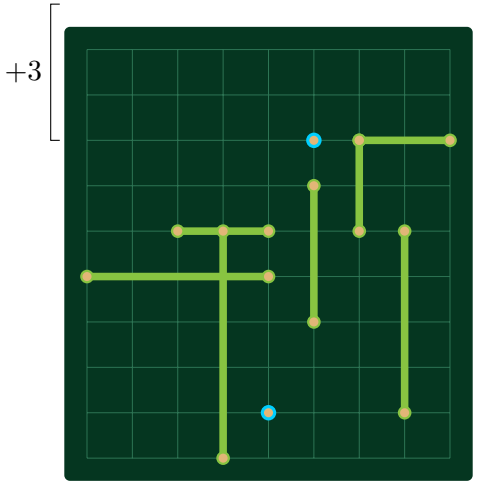
Split into arrays, include bounds and sort:

```
xs = [-1, 0, 2, 3, 3, 4, 4, 4, 5,
      5, 5, 6, 6, 7, 7, 8, 9]

ys = [-1, 0, 1, 1, 3, 4, 4, 5, 5,
      5, 5, 5, 6, 7, 7, 7, 10]
```

$+3$

Search for gaps ≥ 3



Reduction Implementation

5 0 5 6 0 5 6 5 9 1 9 6 7 4 7 7 …
Coordinates of wire end points

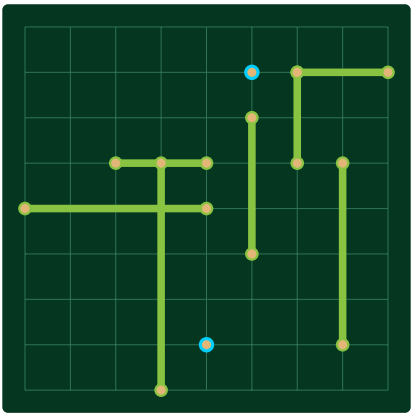
7 8 6 1
Coordinates of points to connect

Split into arrays, include bounds and sort:

```
xs = [-1, 0, 2, 3, 3, 4, 4, 4, 5,
      5, 5, 6, 6, 7, 7, 8, 9]

ys = [-1, 0, 1, 1, 3, 4, 4, 5, 5,
      5, 5, 5, 6, 7, 7, 7, 10]
                    +3
```

Search for gaps ≥ 3



Reduction Implementation

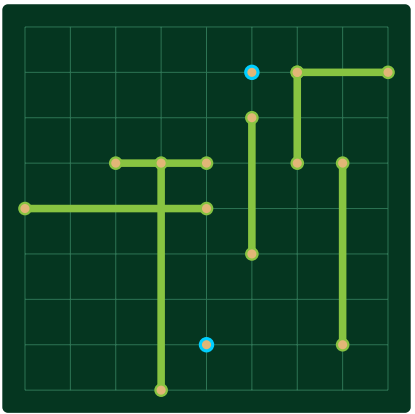
5 0 5 6 0 5 6 5 9 1 9 6 7 4 7 7 …
Coordinates of wire end points

7 8 6 1
Coordinates of points to connect

Split into arrays, include bounds and sort:

```
xs = [-1, 0, 2, 3, 3, 4, 4, 4, 5,
      5, 5, 6, 6, 7, 7, 8, 9]
ys = [-1, 0, 1, 1, 3, 4, 4, 5, 5,
      5, 5, 5, 6, 7, 7, 7, 9]
```

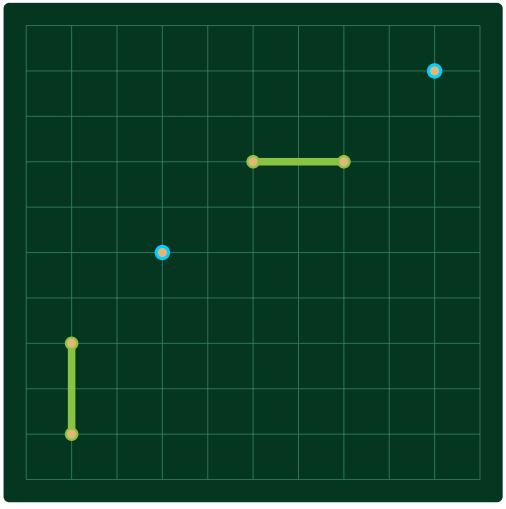
Search for gaps ≥ 3



Worst Case

Width:
$W = (2 \cdot |\textit{unique x-coordinates}| + 1)$

Height:
$H = (2 \cdot |\textit{unique y-coordinates}| + 1)$



Worst Case

Resulting size: $W \cdot H = (2 \cdot |\textit{unique x-coordinates}| + 1) \cdot (2 \cdot |\textit{unique y-coordinates}| + 1)$

- Up to 99 wires (up to three unique coordinate components each)
- Start and end points (two components each)

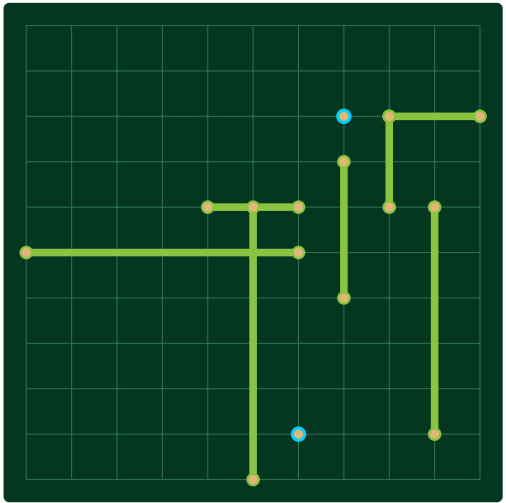
$\Rightarrow 99 \cdot 3 + 2 \cdot 2 = 301$ unique components

$W + H = (2 \cdot |\textit{unique x-coordinates}| + 1) + (2 \cdot |\textit{unique y-coordinates}| + 1)$
$= 2 \cdot (|\textit{unique x-coordinates}| + |\textit{unique y-coordinates}|) + 2$
$= 2 \cdot (|\textit{unique components}|) + 2$
$= 2 \cdot 301 + 2 = 604$

$\Rightarrow W \cdot H \leq \left(\frac{W+H}{2}\right)^2 = \left(\frac{604}{2}\right)^2 = 91204$
$\Rightarrow$ Worst case: ≤ 91204 nodes after reduction

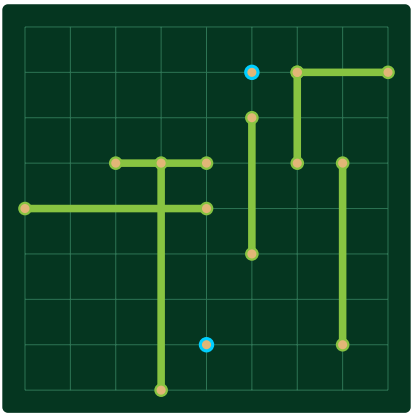
Solution In A Nutshell

1. Parse input
2. Reduce
3. Build graph
4. A* on graph



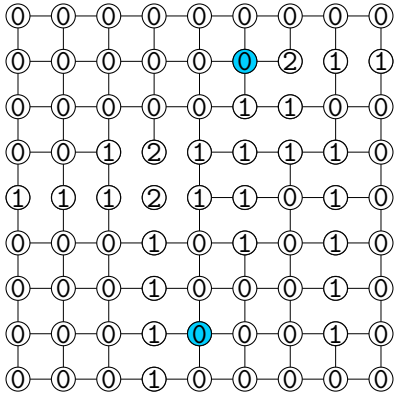
Solution In A Nutshell

1. Parse input
2. Reduce
3. Build graph
4. A* on graph



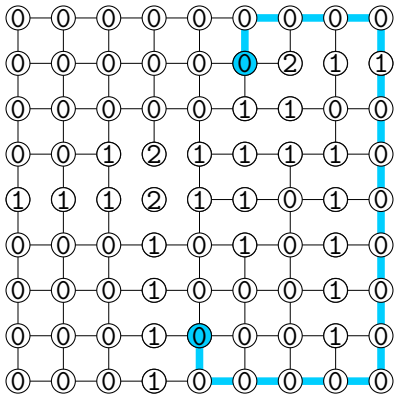
Solution In A Nutshell

1. Parse input
2. Reduce
3. Build graph
4. A* on graph



Solution In A Nutshell

1. Parse input
2. Reduce
3. Build graph
4. A* on graph



Why C?

- C is fast
- Pointers enable elegant solutions
- Personal preference

Any Questions?

Samuel Füßinger

Database Systems
Department of Computer Science
University of Tübingen

21.06.2024

Complexity (simplified[‡])

For grid size s and number of wires m:
(Ignoring the $\log(s)$-factor for growing number of digits)

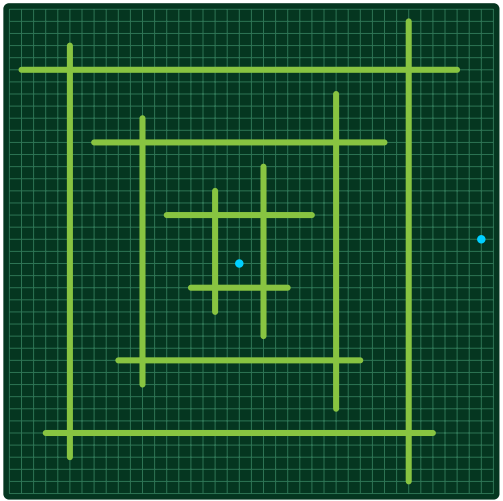
	Time	**Space**
Parsing	$\mathcal{O}(m)$	$\mathcal{O}(m)$
Reduction	$\mathcal{O}(m \log m)$	$\mathcal{O}(m)$
Building the Graph	$\mathcal{O}(\lvert V\rvert) = \mathcal{O}(m^2)$	$\mathcal{O}(\lvert V\rvert) = \mathcal{O}(m^2)$
A*	$\mathcal{O}(\lvert E\rvert \log \lvert V\rvert) = \mathcal{O}(m^2 \log m)$[†]	$\mathcal{O}(\lvert V\rvert) = \mathcal{O}(m^2)$
Total	$\mathcal{O}(m^2 \log m)$	$\mathcal{O}(m^2)$

Basically independent of input grid size s!

[†] Because $\lvert E\rvert \approx 2\lvert V\rvert$
[‡] see the paper for the exact complexities

Worst Case (Time/Space)

Explanatory simplification
with fewer cables



Worst Case (Cost)

Explanatory simplification
with fewer cables