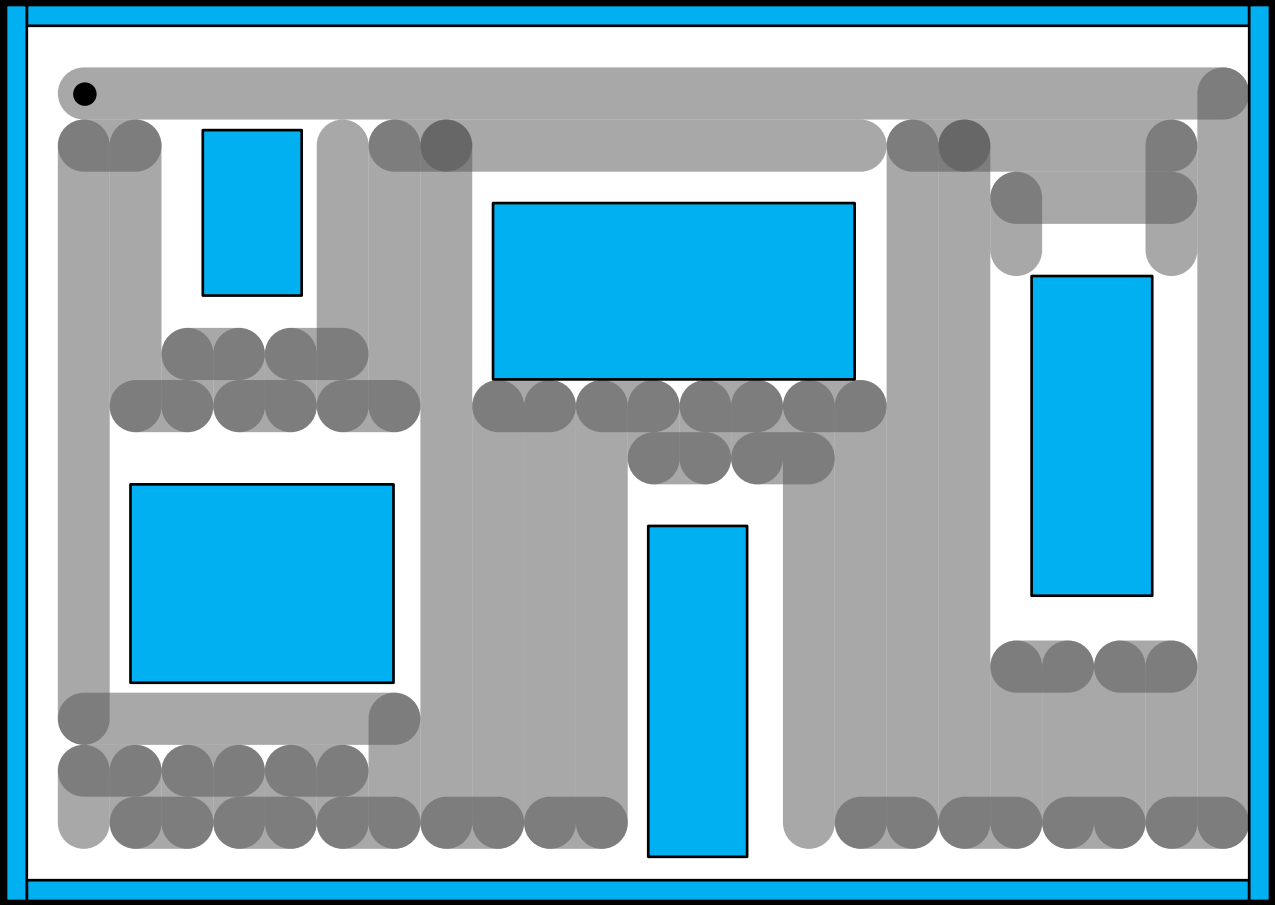
Improving Coverage

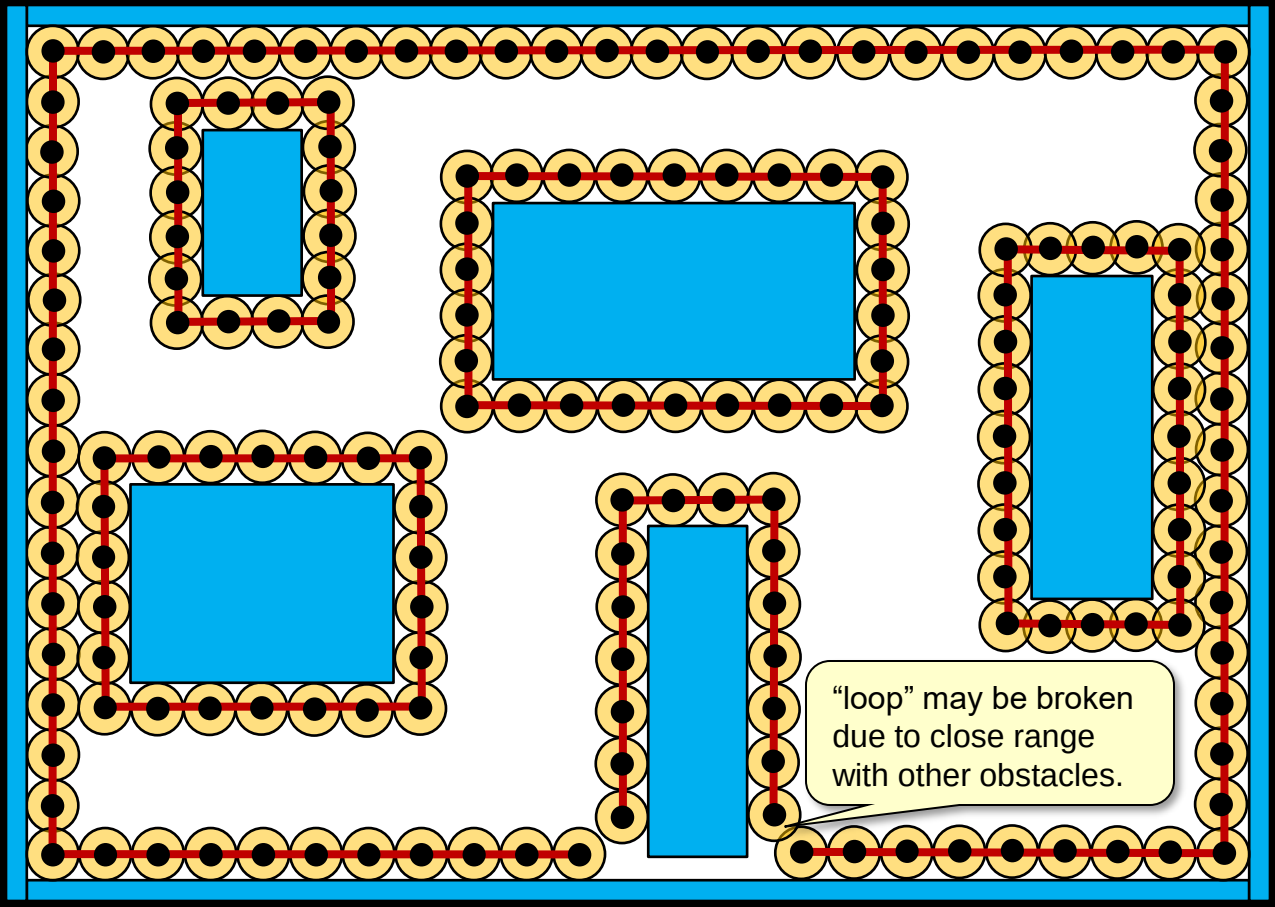
Spanning Tree Coverage

- Recall that spanning tree coverage left areas untouched around the obstacles and border:



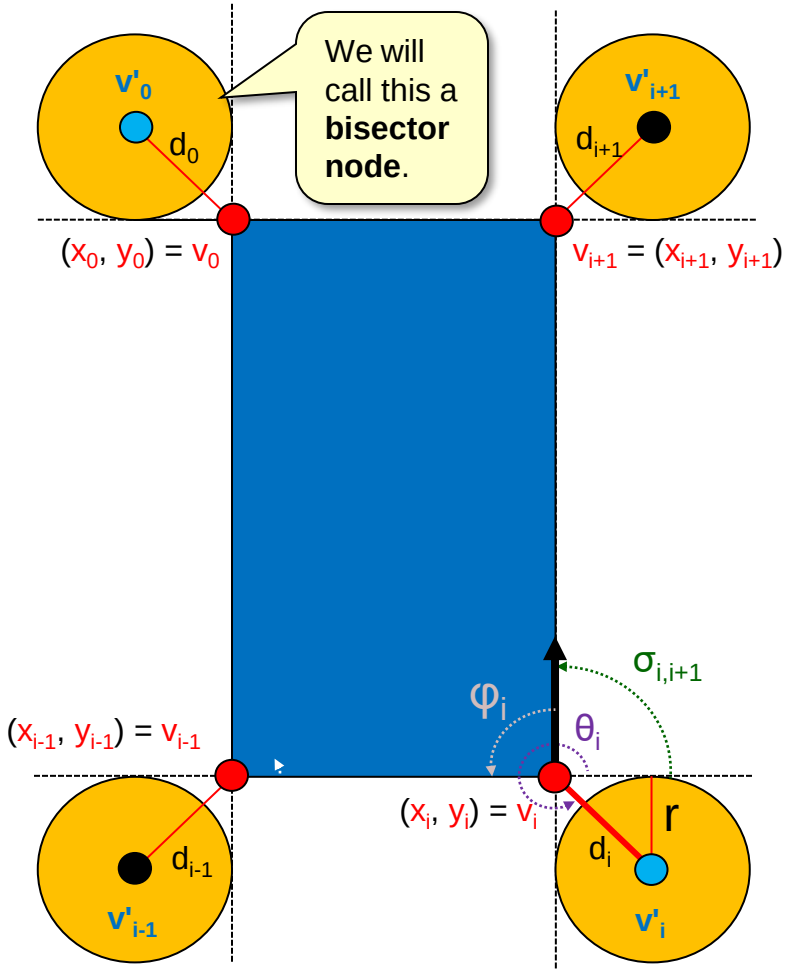
Border/Obstacle Coverage

- How do we get better coverage around the obstacles?
- Place additional graph “loops” around the obstacles:



Bisector Vertices

- Add graph nodes around obstacles by considering the robot radius and the bisectors of pairs of adjacent obstacle edges:



$$dot = (x_{i+1}-x_i)\cdot(x_{i-1}-x_i) + (y_{i+1}-y_i)\cdot(y_{i-1}-y_i)$$

$$magA = \sqrt{((x_{i+1}-x_i)^2 + (y_{i+1}-y_i)^2)}$$

$$magB = \sqrt{((x_{i-1}-x_i)^2 + (y_{i-1}-y_i)^2)}$$

$$\varphi_i = acos(dot / magA / magB)$$

$$d_i = (r+\varepsilon) / \sin(\varphi_i/2)$$

$$\sigma_{i,i+1} = atan2((y_{i+1}-y_i) / (x_{i+1}-x_i))$$

$$\theta_i = \sigma_{i,i+1} + \varphi_i/2 + \pi$$

$$x'_i = x_i + d_i\cdot\cos(\theta_i)$$

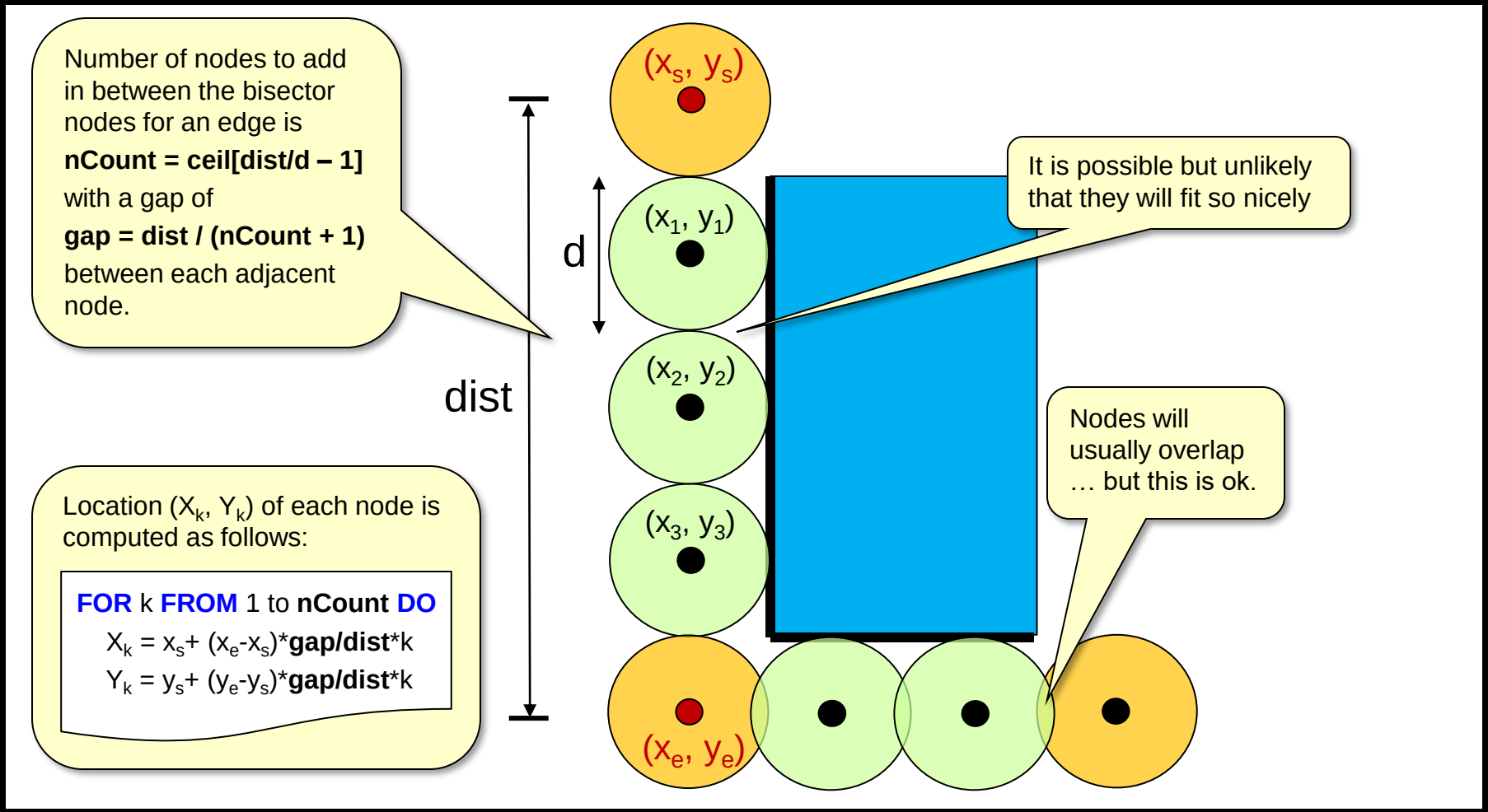
$$y'_i = y_i + d_i\cdot\sin(\theta_i)$$

$$v'_i = (x'_i , y'_i)$$

r is robot radius. Add small **ε** value of **0.01** or so to account for round-off errors so that vertices are not too close to obstacle edge.

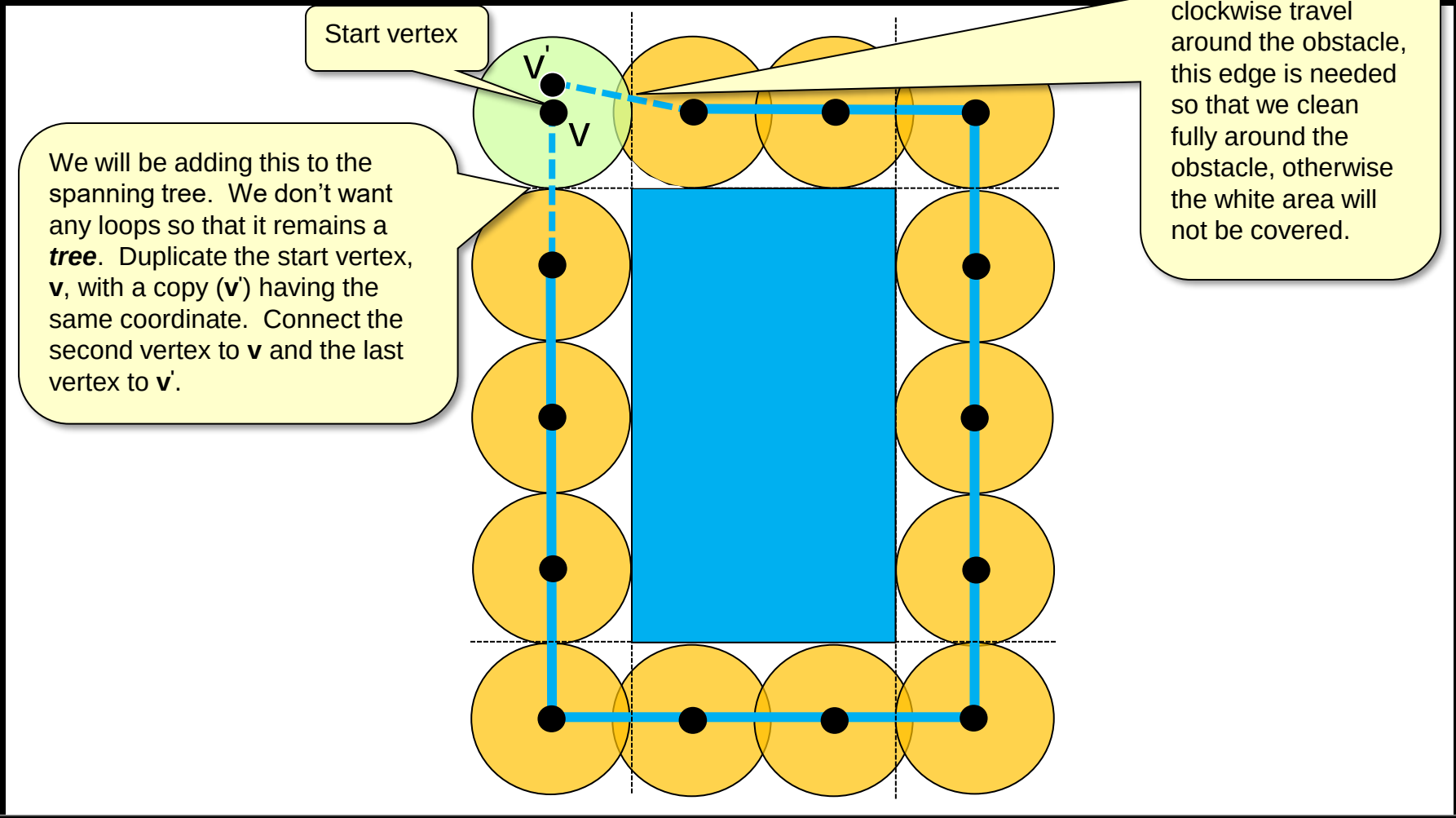
Creating the Graph Nodes

- For each edge of the obstacle, add nodes along each edge from its two endpoints.



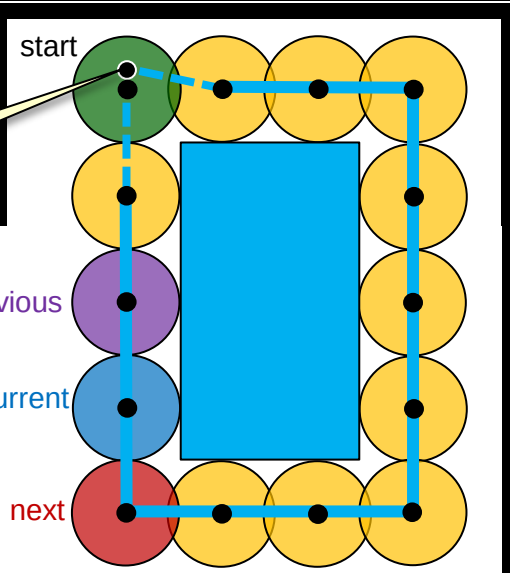
Connecting the Nodes

- Connect adjacent nodes along each edge:



Iterating Through Node Loop

- When looking for invalid nodes, we need to iterate through the Node "loop":



This happens when **current** is here

```
start = the starting node of the loop

bad  = an empty list
previous = NULL
current = start
next = node at other end of start's first and only edge

WHILE (previous is NULL) or (current does not have the same coordinates as start) THEN {

    IF current is invalid THEN
        add current to bad
    previous = current
    current = next
    next = node at other end of next's first edge
    IF (next == previous) AND (there is at least one more edge connected to current) THEN
        next = node at other end of current's 2nd edge
}
IF current is invalid THEN
    add current to bad
```

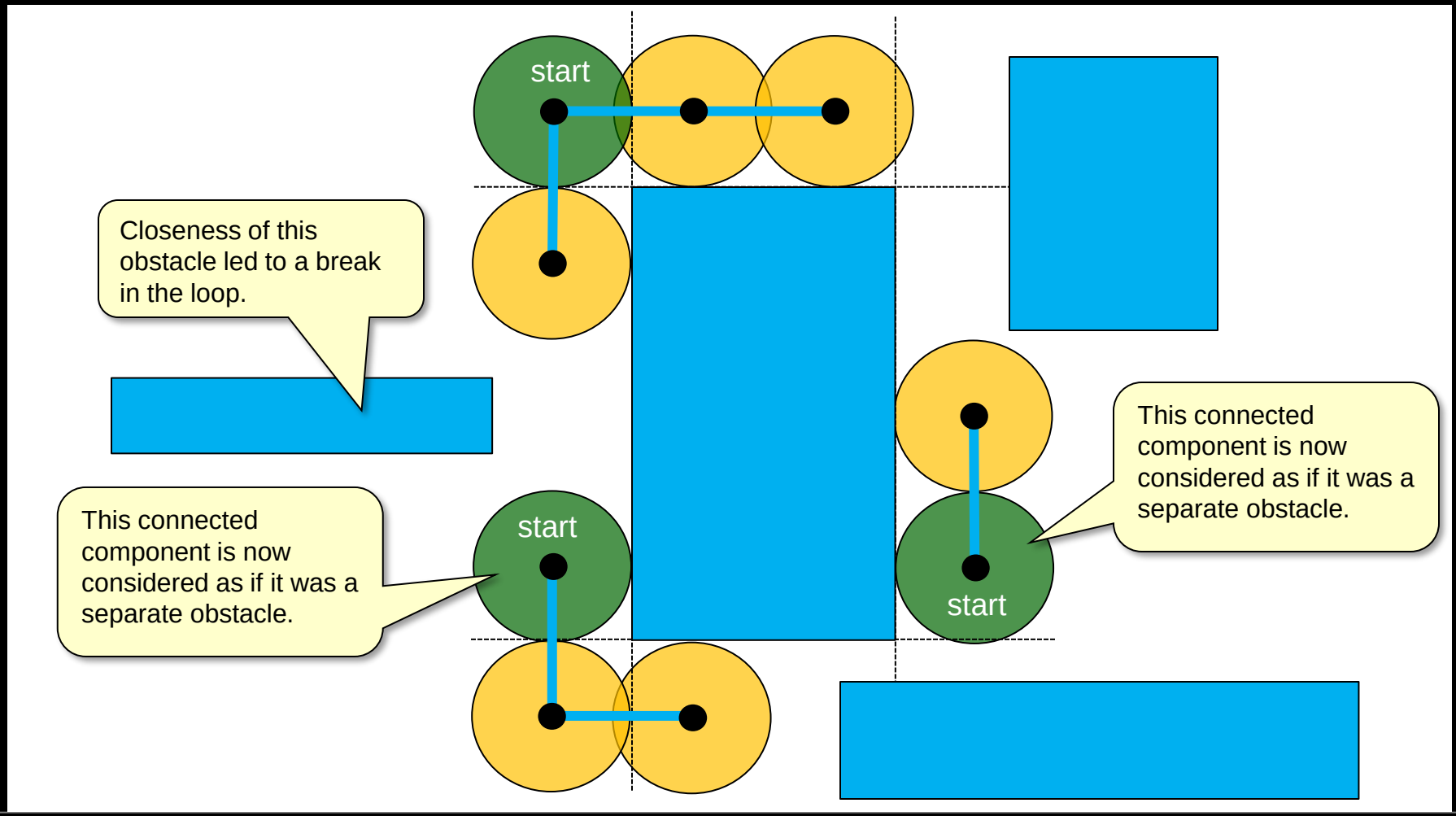
This happens only first time in loop.

This handles the very last **current** node in case it is invalid.

This case is needed in case the next edge is not the first one in **next** node's list. This will happen later when we are disconnecting the duplicate start vertex (slide 9).

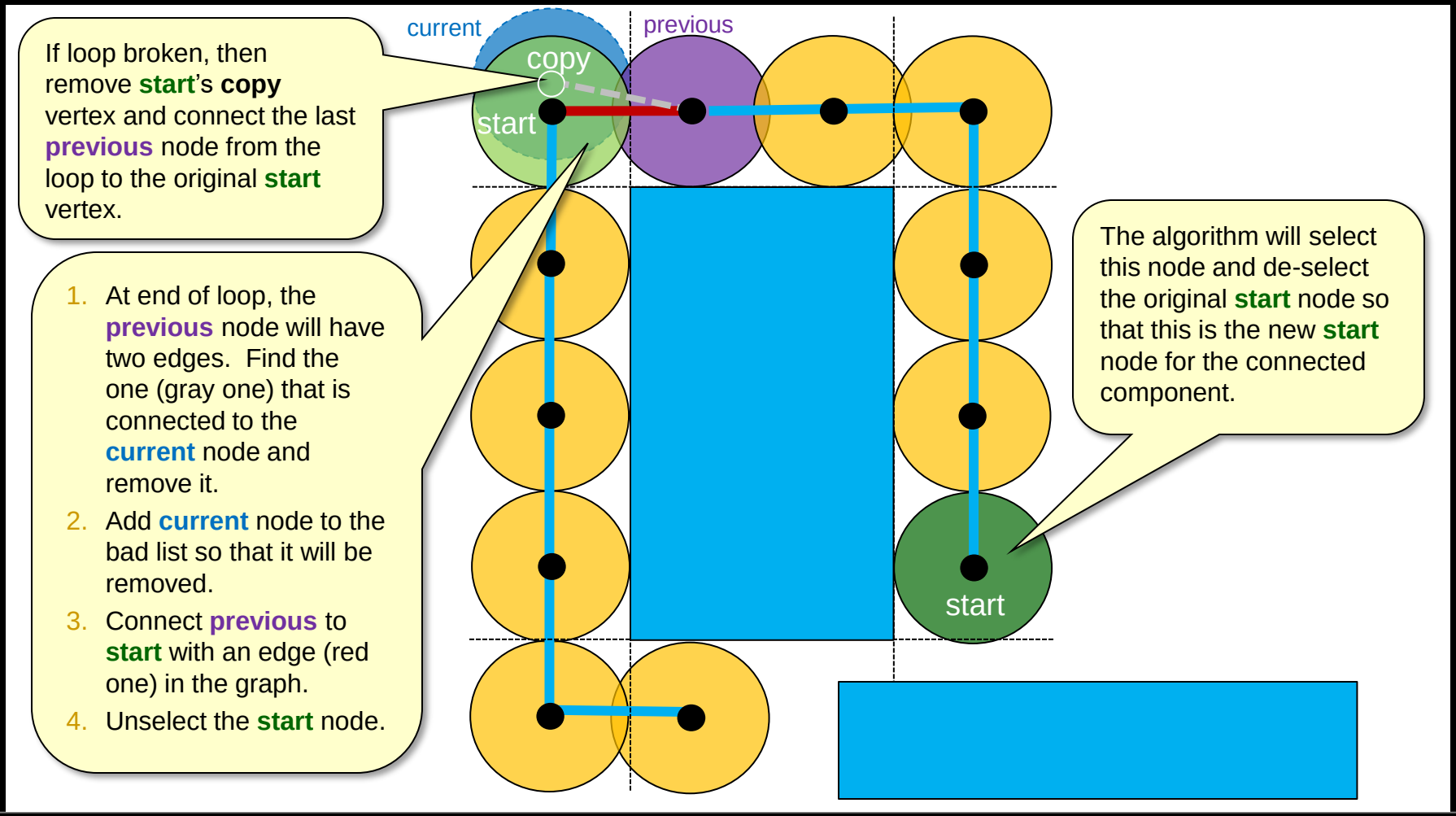
Handling “Really” Broken Loops

- If loop gets broken into multiple components, we must determine the **start** of each connected component:



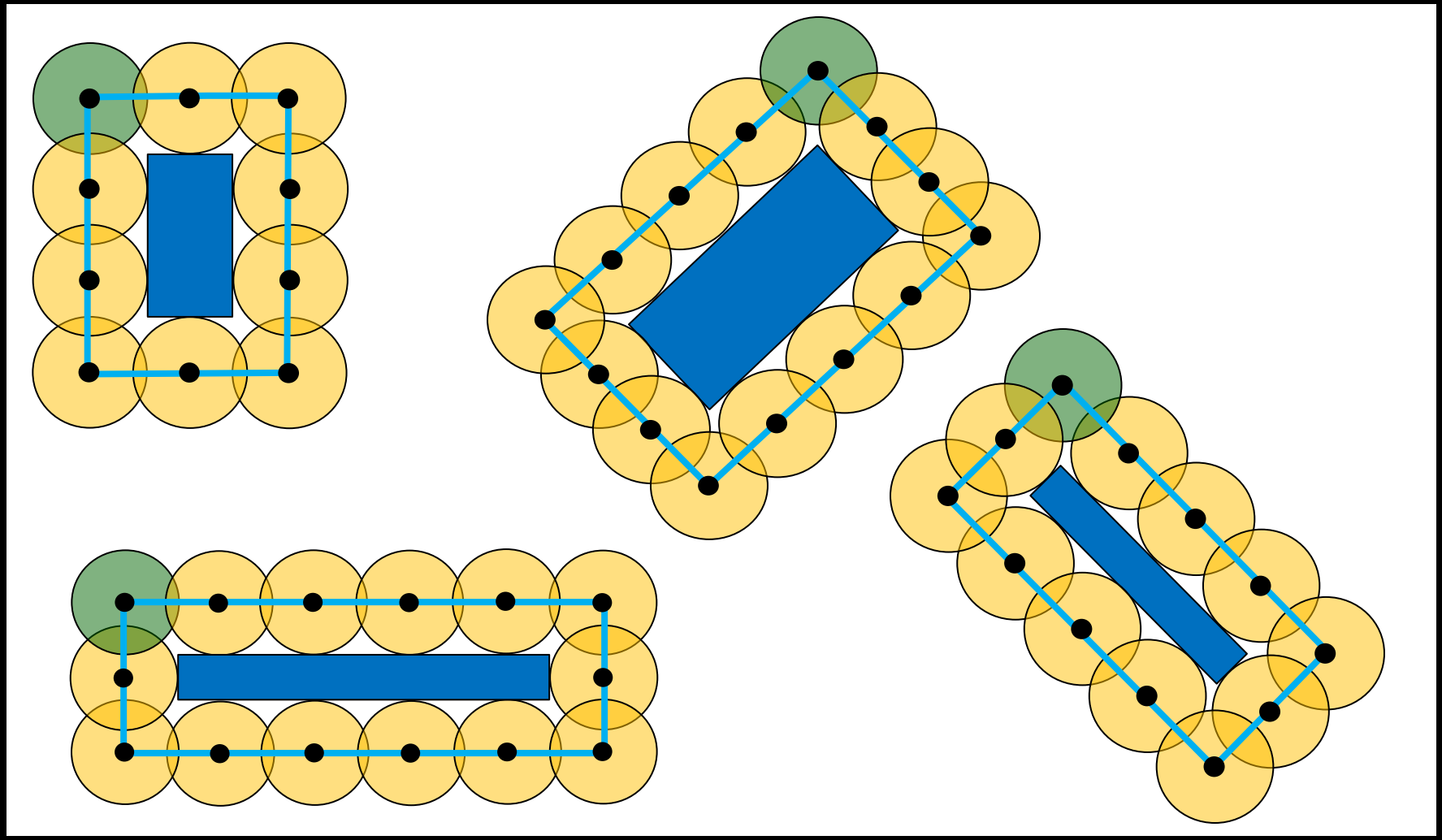
Handling Broken Loops

- If loop gets broken at all, it is still a single connected component and there is no special case on the first vertex:



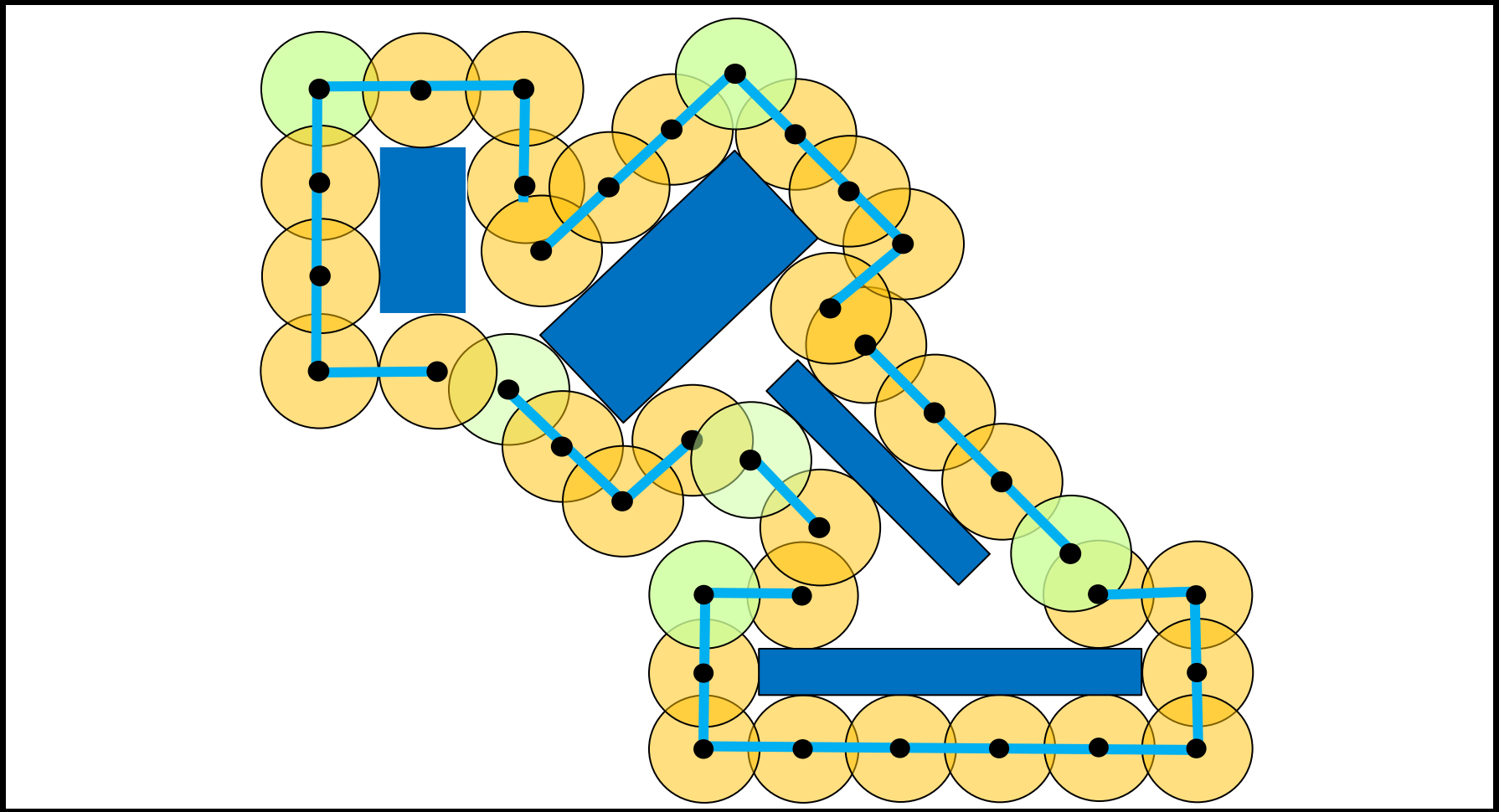
Obstacle-Coverage Vertices

- Do each connected-component separately



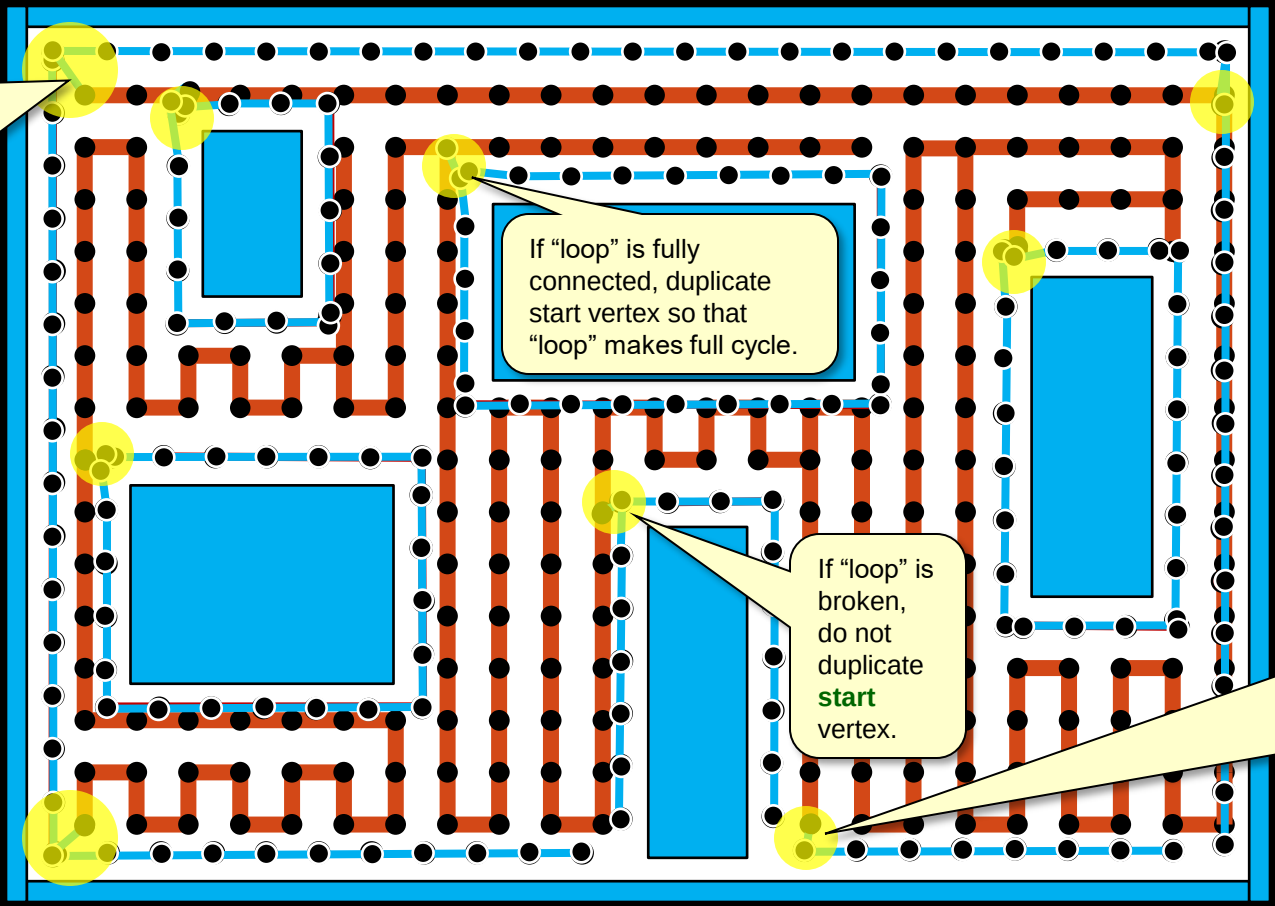
Connected Components

- Eliminate invalid vertices that intersect other obstacles … resulting in connected components:



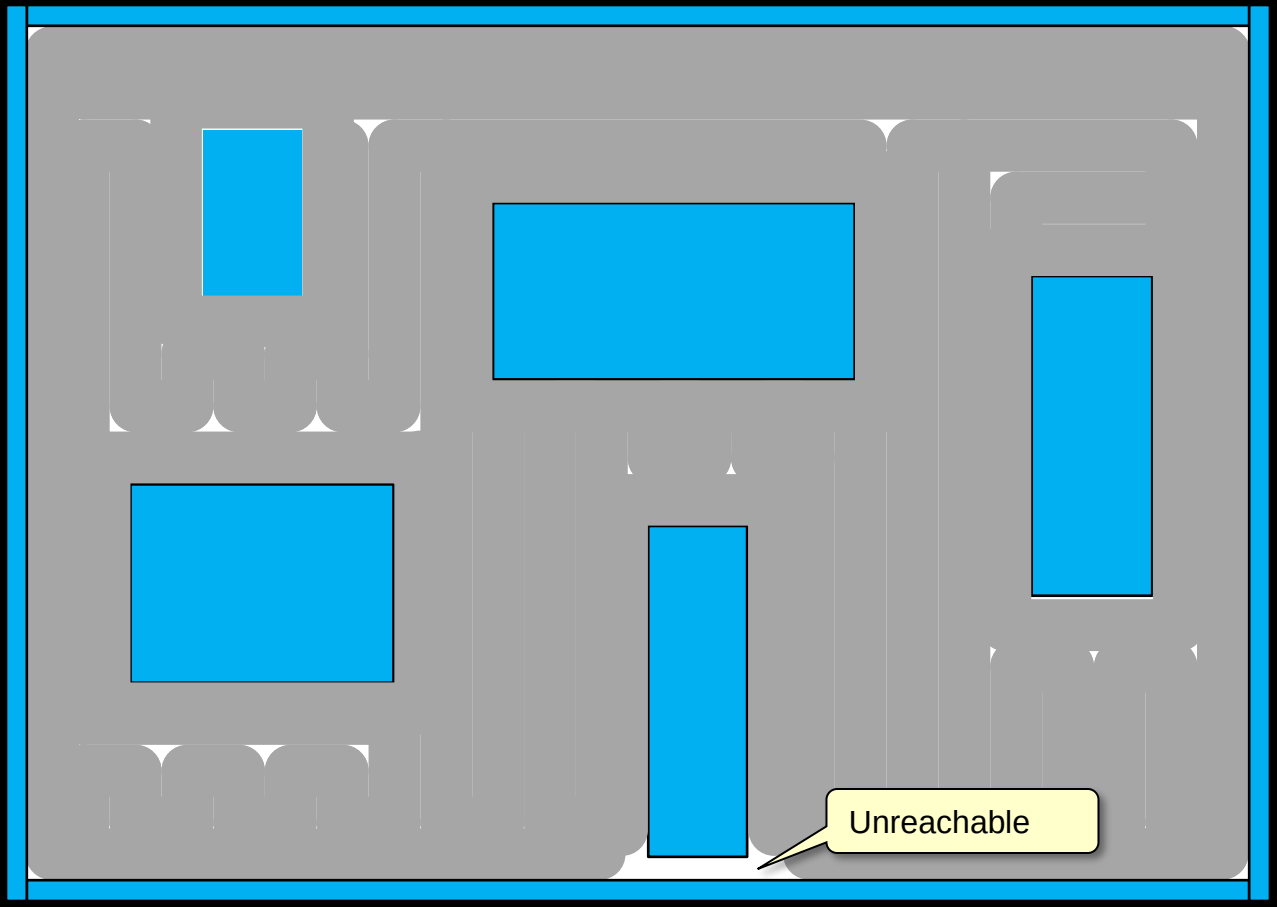
Merged Spanning Tree

- Attach obstacle and border “loops” to spanning tree
  - Result is still a tree since each obstacle added only branches:



Final Area Coverage

- Environment is reasonably well-covered in the end:



Start the Lab …