# Bidirectional Sampling-Based Motion Planning

In [73]:
```
# The autoreload extension will automatically load in new code as you e
dit files,
# so you don't need to restart the kernel every time
%load_ext autoreload
%autoreload 2

import numpy as np
import matplotlib.pyplot as plt
from P2_rrt import *
from P4_bidirectional_rrt import *

plt.rcParams['figure.figsize'] = [7, 7] # Change default figure size
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

### Set up workspace

In [74]:
```
MAZE = np.array([
    (( 5, 5), (-5, 5)),
    ((-5, 5), (-5,-5)),
    ((-5,-5), ( 5,-5)),
    (( 5,-5), ( 5, 5)),
    ((-5, 2), (-1, 2)),
    ((-1, 2), (-1,-1)),
    (( 0, 2), ( 0,-1)),
    (( 0, 2), ( 5, 2))
])
```

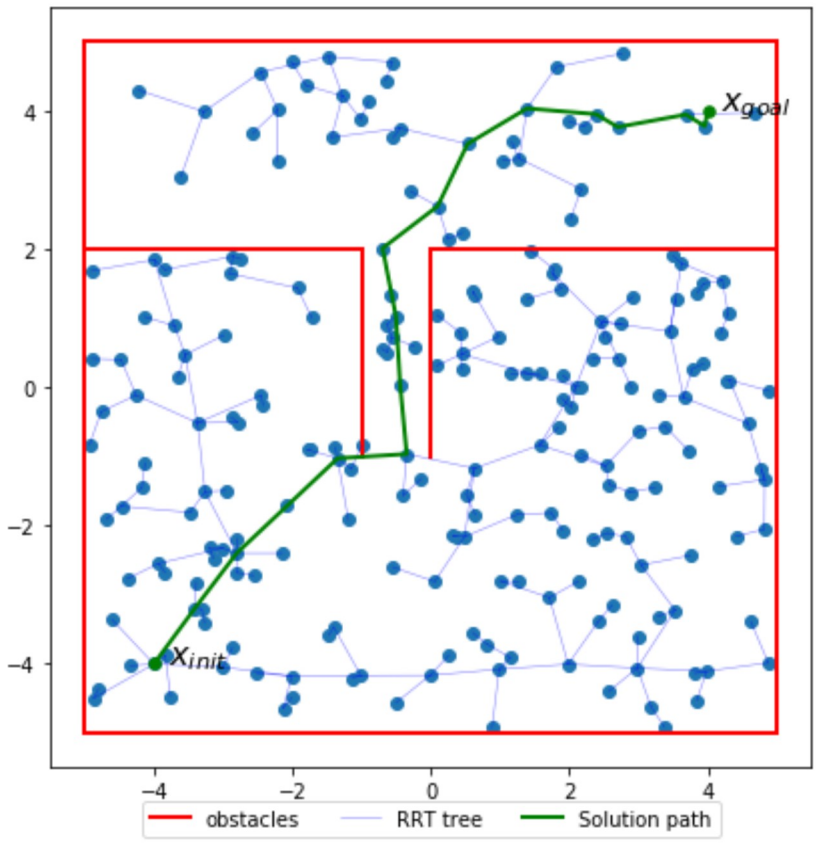
# Normal RRT

On this "bugtrap" problem, normal RRT often will fail to find a find a path.

### Geometric planning

In [75]:
```
grrt = GeometricRRT([-5,-5], [5,5], [-4,-4], [4,4], MAZE)
grrt.solve(1.0, 2000)
```

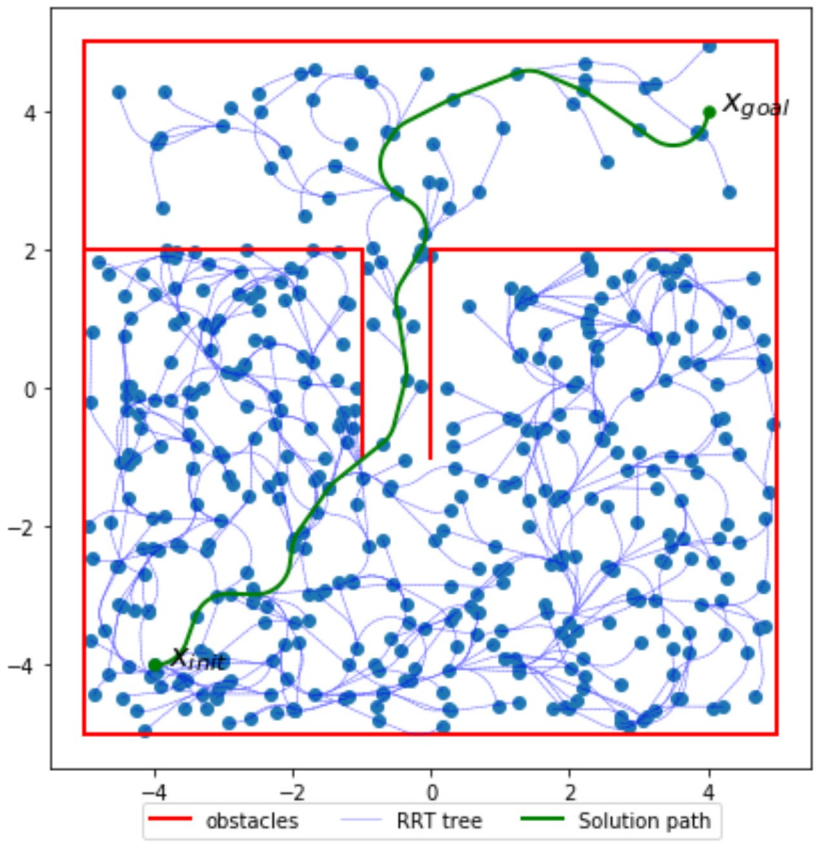
Out[75]: True




## Dubins car planning

In [80]:
```
drrt = DubinsRRT([-5,-5,0], [5,5,2*np.pi], [-4,-4,0], [4,4,np.pi/2], MA
ZE, .5)
drrt.solve(1.0, 1000)
```

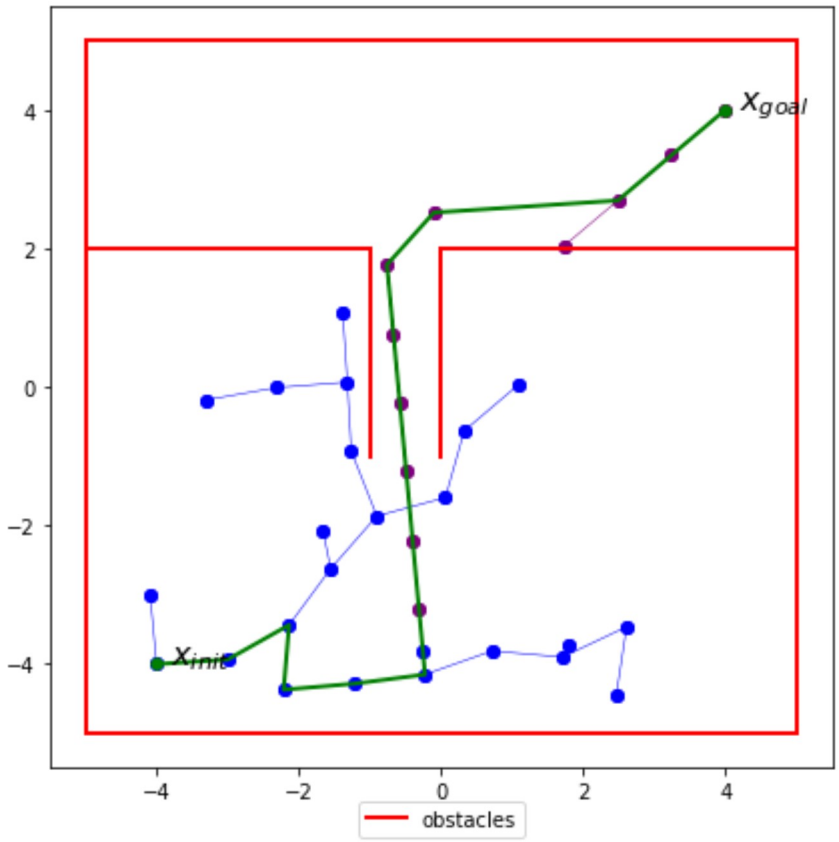
Out[80]: True




# RRTConnect

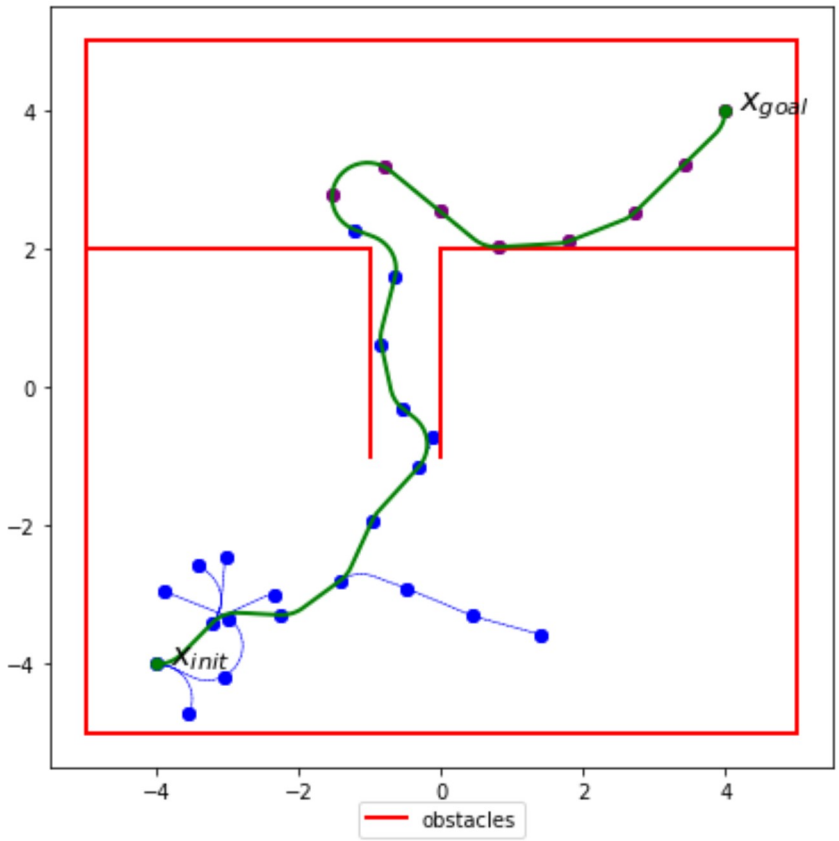
## Geometric planning

In [77]:
```
grrt = GeometricRRTConnect([-5,-5], [5,5], [-4,-4], [4,4], MAZE)
grrt.solve(1.0, 2000)
```




## Dubins car planning

In [78]:
```
drrt = DubinsRRTConnect([-5,-5,0], [5,5,2*np.pi], [-4,-4,0], [4,4,np.pi
/2], MAZE, .5)
drrt.solve(1.0, 1000)
```




In [ ]:
```

```