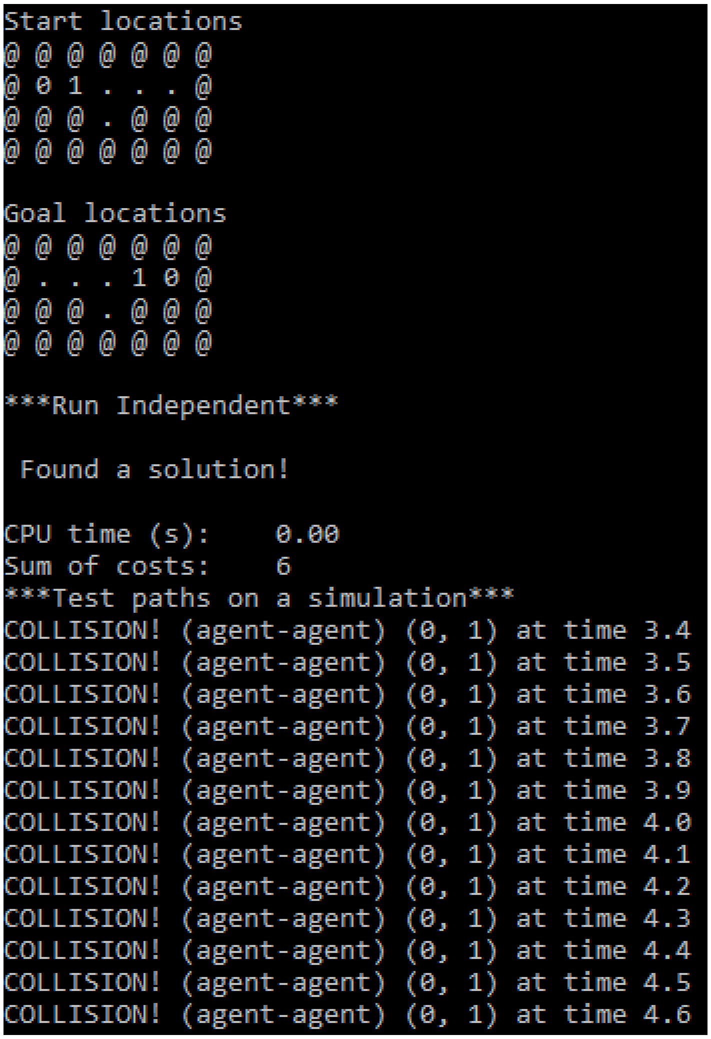
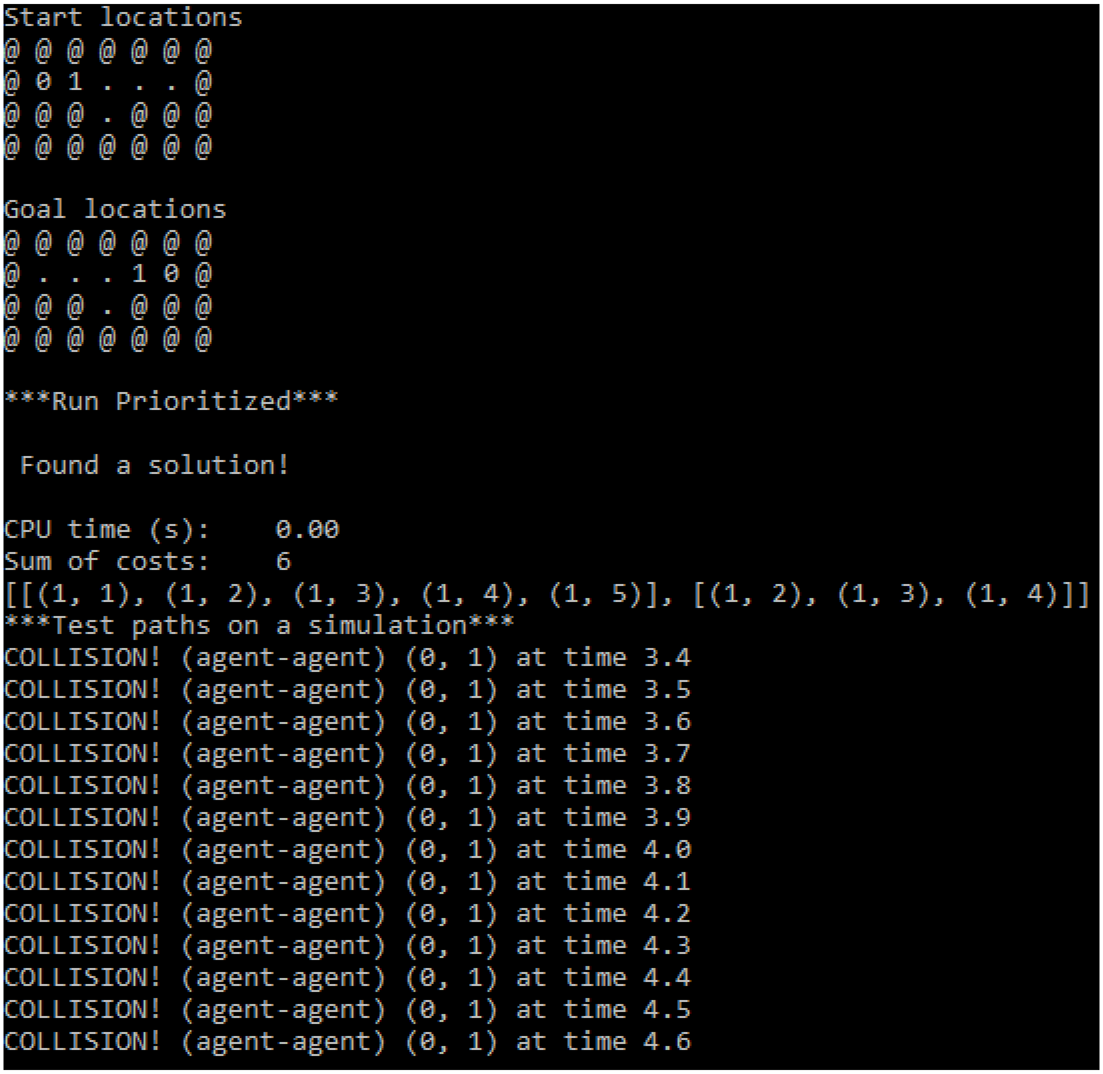
Implementing Space-Time A*

1.1.



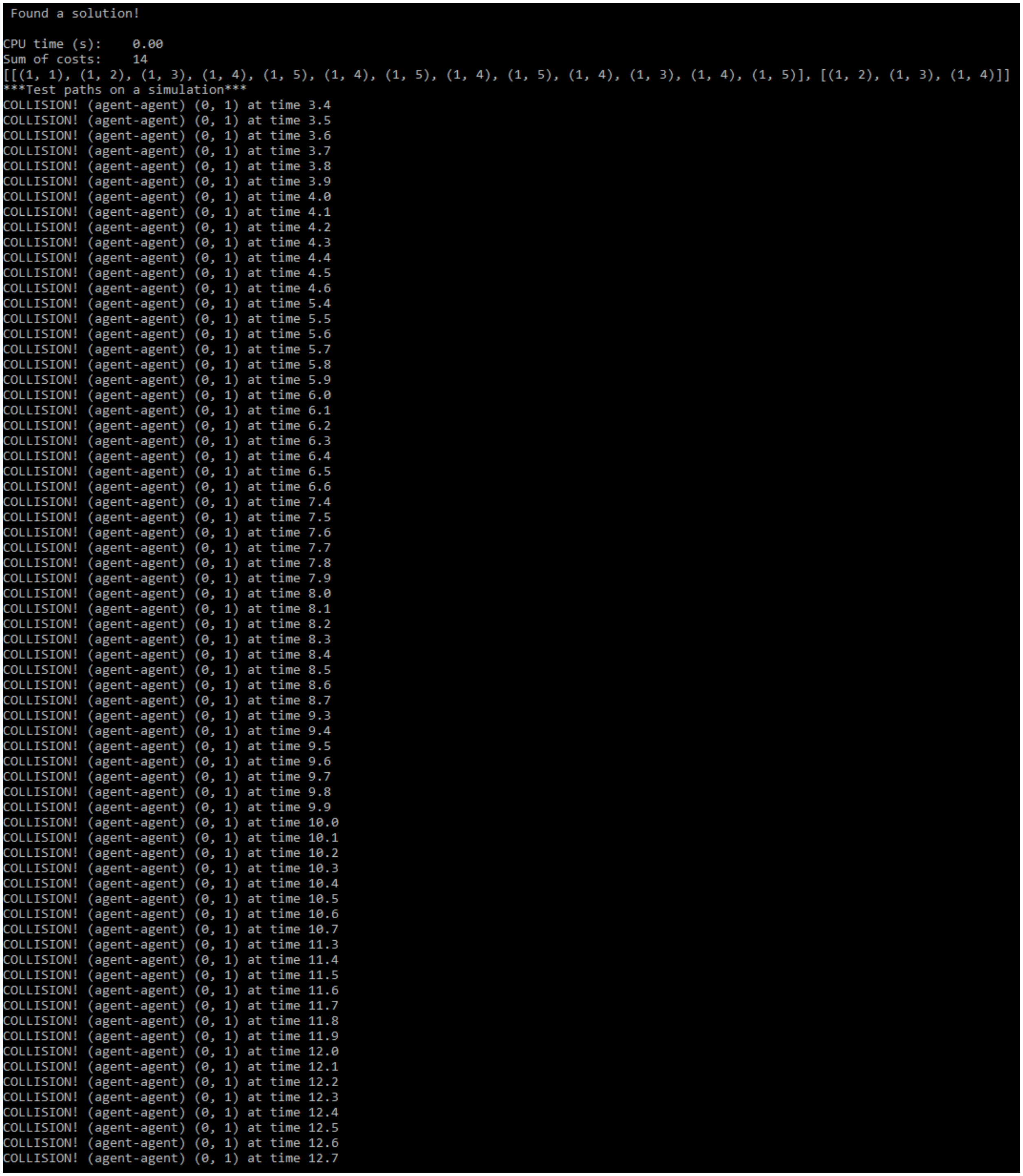
```
Start locations
@ @ @ @ @ @ @
@ 0 1 . . . @
@ @ @ . @ @ @
@ @ @ @ @ @ @

Goal locations
@ @ @ @ @ @ @
@ . . . 1 0 @
@ @ @ . @ @ @
@ @ @ @ @ @ @

***Run Independent***

 Found a solution!

CPU time (s):    0.00
Sum of costs:    6
***Test paths on a simulation***
COLLISION! (agent-agent) (0, 1) at time 3.4
COLLISION! (agent-agent) (0, 1) at time 3.5
COLLISION! (agent-agent) (0, 1) at time 3.6
COLLISION! (agent-agent) (0, 1) at time 3.7
COLLISION! (agent-agent) (0, 1) at time 3.8
COLLISION! (agent-agent) (0, 1) at time 3.9
COLLISION! (agent-agent) (0, 1) at time 4.0
COLLISION! (agent-agent) (0, 1) at time 4.1
COLLISION! (agent-agent) (0, 1) at time 4.2
COLLISION! (agent-agent) (0, 1) at time 4.3
COLLISION! (agent-agent) (0, 1) at time 4.4
COLLISION! (agent-agent) (0, 1) at time 4.5
COLLISION! (agent-agent) (0, 1) at time 4.6
```

1.2.



```
Start locations
@ @ @ @ @ @ @
@ 0 1 . . . @
@ @ @ . @ @ @
@ @ @ @ @ @ @

Goal locations
@ @ @ @ @ @ @
@ . . . 1 0 @
@ @ @ . @ @ @
@ @ @ @ @ @ @

***Run Prioritized***

 Found a solution!

CPU time (s):    0.00
Sum of costs:    6
[[(1, 1), (1, 2), (1, 3), (1, 4), (1, 5)], [(1, 2), (1, 3), (1, 4)]]
***Test paths on a simulation***
COLLISION! (agent-agent) (0, 1) at time 3.4
COLLISION! (agent-agent) (0, 1) at time 3.5
COLLISION! (agent-agent) (0, 1) at time 3.6
COLLISION! (agent-agent) (0, 1) at time 3.7
COLLISION! (agent-agent) (0, 1) at time 3.8
COLLISION! (agent-agent) (0, 1) at time 3.9
COLLISION! (agent-agent) (0, 1) at time 4.0
COLLISION! (agent-agent) (0, 1) at time 4.1
COLLISION! (agent-agent) (0, 1) at time 4.2
COLLISION! (agent-agent) (0, 1) at time 4.3
COLLISION! (agent-agent) (0, 1) at time 4.4
COLLISION! (agent-agent) (0, 1) at time 4.5
COLLISION! (agent-agent) (0, 1) at time 4.6
```

1.3.

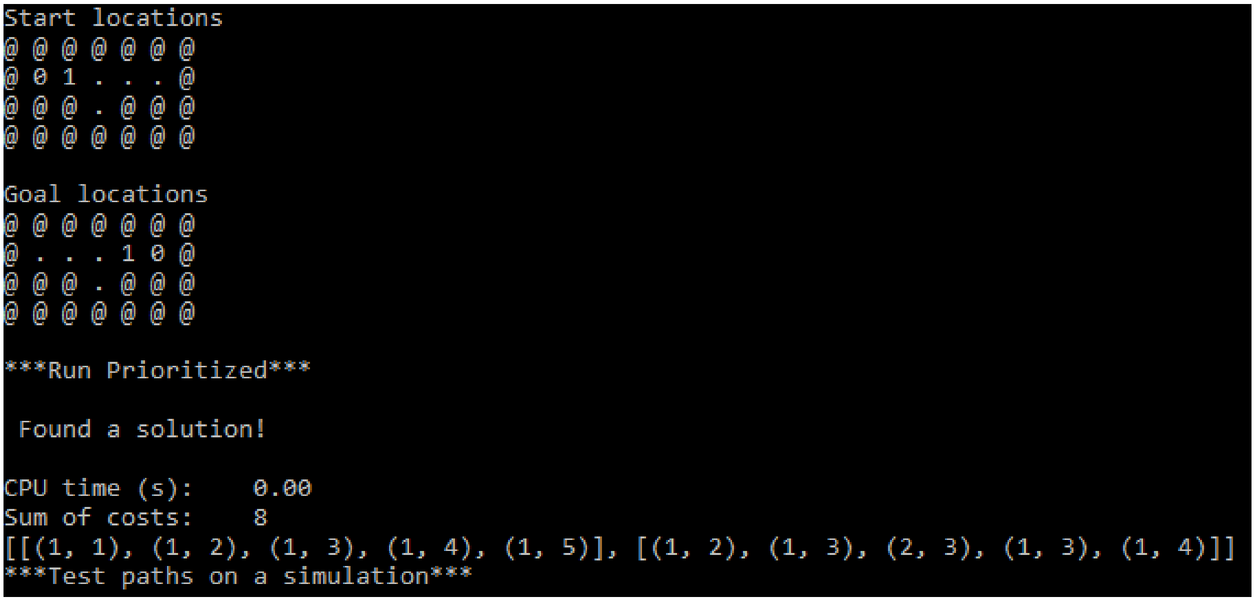


```
 Found a solution!

CPU time (s):    0.00
Sum of costs:    14
[[(1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (1, 4), (1, 5), (1, 4), (1, 5), (1, 4), (1, 3), (1, 4), (1, 5)], [(1, 2), (1, 3), (1, 4)]]
***Test paths on a simulation***
COLLISION! (agent-agent) (0, 1) at time 3.4
COLLISION! (agent-agent) (0, 1) at time 3.5
COLLISION! (agent-agent) (0, 1) at time 3.6
COLLISION! (agent-agent) (0, 1) at time 3.7
COLLISION! (agent-agent) (0, 1) at time 3.8
COLLISION! (agent-agent) (0, 1) at time 3.9
COLLISION! (agent-agent) (0, 1) at time 4.0
COLLISION! (agent-agent) (0, 1) at time 4.1
COLLISION! (agent-agent) (0, 1) at time 4.2
COLLISION! (agent-agent) (0, 1) at time 4.3
COLLISION! (agent-agent) (0, 1) at time 4.4
COLLISION! (agent-agent) (0, 1) at time 4.5
COLLISION! (agent-agent) (0, 1) at time 4.6
COLLISION! (agent-agent) (0, 1) at time 5.4
COLLISION! (agent-agent) (0, 1) at time 5.5
COLLISION! (agent-agent) (0, 1) at time 5.6
COLLISION! (agent-agent) (0, 1) at time 5.7
COLLISION! (agent-agent) (0, 1) at time 5.8
COLLISION! (agent-agent) (0, 1) at time 5.9
COLLISION! (agent-agent) (0, 1) at time 6.0
COLLISION! (agent-agent) (0, 1) at time 6.1
COLLISION! (agent-agent) (0, 1) at time 6.2
COLLISION! (agent-agent) (0, 1) at time 6.3
COLLISION! (agent-agent) (0, 1) at time 6.4
COLLISION! (agent-agent) (0, 1) at time 6.5
COLLISION! (agent-agent) (0, 1) at time 6.6
COLLISION! (agent-agent) (0, 1) at time 7.4
COLLISION! (agent-agent) (0, 1) at time 7.5
COLLISION! (agent-agent) (0, 1) at time 7.6
COLLISION! (agent-agent) (0, 1) at time 7.7
COLLISION! (agent-agent) (0, 1) at time 7.8
COLLISION! (agent-agent) (0, 1) at time 7.9
COLLISION! (agent-agent) (0, 1) at time 8.0
COLLISION! (agent-agent) (0, 1) at time 8.1
COLLISION! (agent-agent) (0, 1) at time 8.2
COLLISION! (agent-agent) (0, 1) at time 8.3
COLLISION! (agent-agent) (0, 1) at time 8.4
COLLISION! (agent-agent) (0, 1) at time 8.5
COLLISION! (agent-agent) (0, 1) at time 8.6
COLLISION! (agent-agent) (0, 1) at time 8.7
COLLISION! (agent-agent) (0, 1) at time 9.3
COLLISION! (agent-agent) (0, 1) at time 9.4
COLLISION! (agent-agent) (0, 1) at time 9.5
COLLISION! (agent-agent) (0, 1) at time 9.6
COLLISION! (agent-agent) (0, 1) at time 9.7
COLLISION! (agent-agent) (0, 1) at time 9.8
COLLISION! (agent-agent) (0, 1) at time 9.9
COLLISION! (agent-agent) (0, 1) at time 10.0
COLLISION! (agent-agent) (0, 1) at time 10.1
COLLISION! (agent-agent) (0, 1) at time 10.2
COLLISION! (agent-agent) (0, 1) at time 10.3
COLLISION! (agent-agent) (0, 1) at time 10.4
COLLISION! (agent-agent) (0, 1) at time 10.5
COLLISION! (agent-agent) (0, 1) at time 10.6
COLLISION! (agent-agent) (0, 1) at time 10.7
COLLISION! (agent-agent) (0, 1) at time 11.3
COLLISION! (agent-agent) (0, 1) at time 11.4
COLLISION! (agent-agent) (0, 1) at time 11.5
COLLISION! (agent-agent) (0, 1) at time 11.6
COLLISION! (agent-agent) (0, 1) at time 11.7
COLLISION! (agent-agent) (0, 1) at time 11.8
COLLISION! (agent-agent) (0, 1) at time 11.9
COLLISION! (agent-agent) (0, 1) at time 12.0
COLLISION! (agent-agent) (0, 1) at time 12.1
COLLISION! (agent-agent) (0, 1) at time 12.2
COLLISION! (agent-agent) (0, 1) at time 12.3
COLLISION! (agent-agent) (0, 1) at time 12.4
COLLISION! (agent-agent) (0, 1) at time 12.5
COLLISION! (agent-agent) (0, 1) at time 12.6
COLLISION! (agent-agent) (0, 1) at time 12.7
```

1.4. Agent 0 at time step 10 is on its goal. I made a lower bound using the timesteps given in the constraints. If there are constraints, I set my lower bound variable "earliest_goal_timestep" to the max timestep in the constraint table with the assumption that the solution path accounts for all obstacles. As long as the goal path time is ≥ earliest_goal_timestep we can assume the found path has avoided all obstacles. If there are no constraints, we set the lowerbound to 0.

1.5. The set of constraints set to achieve a proper solution is the following:

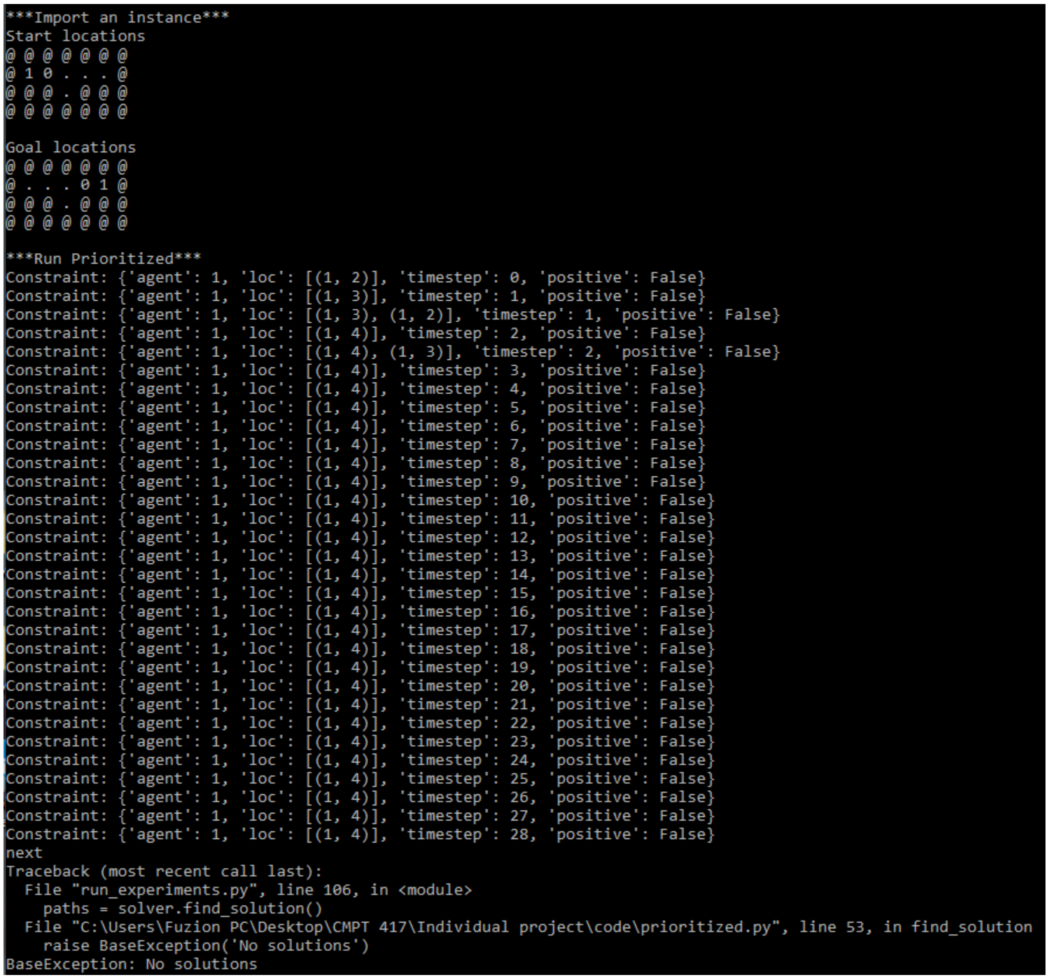
{'agent': 1,'loc': [(1,3),(1,4)],'timestep': 2}
{'agent': 1,'loc': [(1,3)],'timestep': 2}
{'agent': 1,'loc': [(1,3),(1,2)],'timestep': 2}



```
Start locations
@ @ @ @ @ @ @
@ 0 1 . . . @
@ @ @ . @ @ @
@ @ @ @ @ @ @

Goal locations
@ @ @ @ @ @ @
@ . . . 1 0 @
@ @ @ . @ @ @
@ @ @ @ @ @ @

***Run Prioritized***

 Found a solution!

CPU time (s):    0.00
Sum of costs:    8
[[(1, 1), (1, 2), (1, 3), (1, 4), (1, 5)], [(1, 2), (1, 3), (2, 3), (1, 3), (1, 4)]]
***Test paths on a simulation***
```

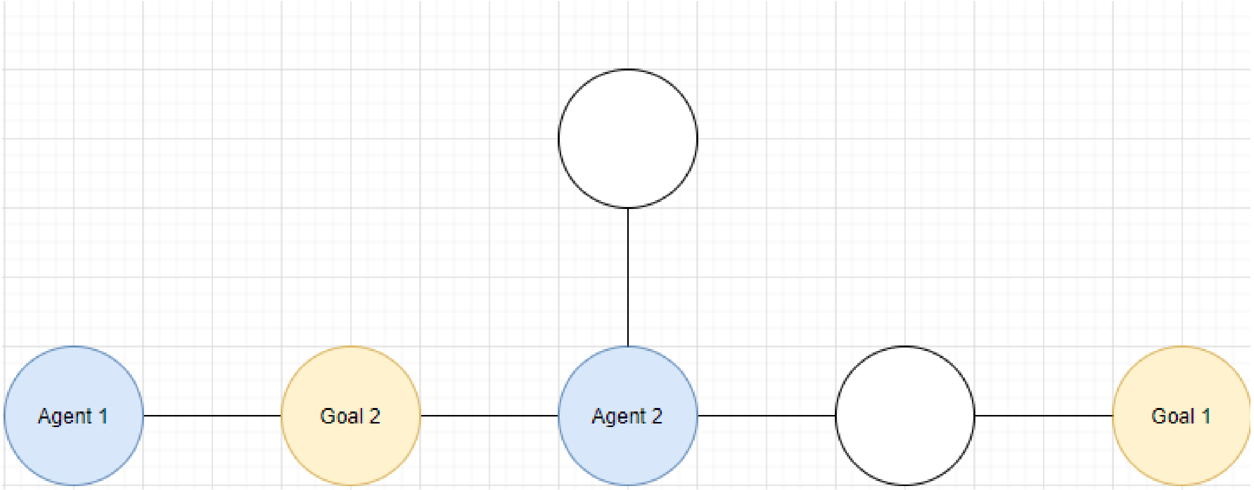
Implementing Prioritized Planning

2.3. Yes, it reported back with *No Solutions* as shown below:



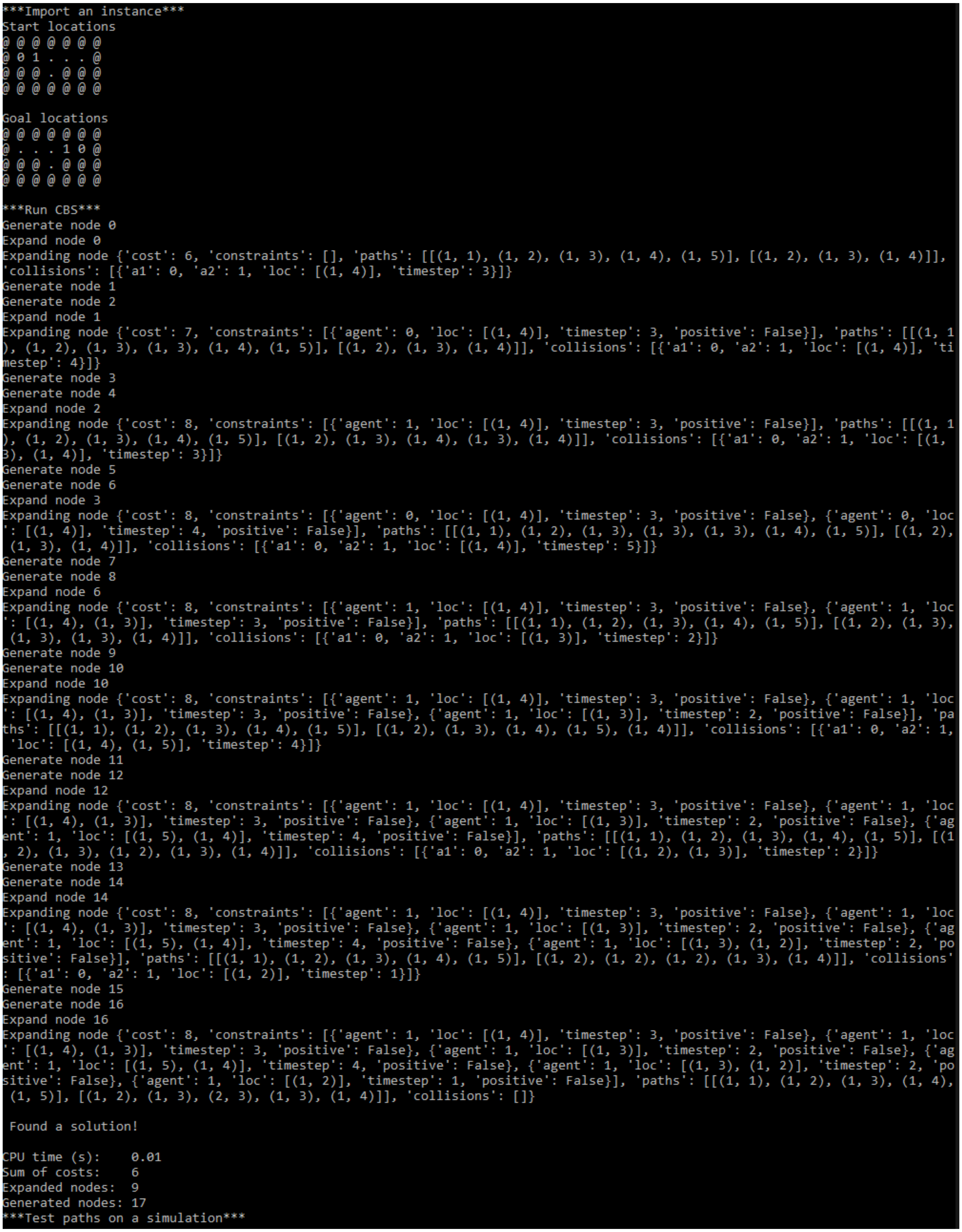
```
***Import an instance***
Start locations
@ @ @ @ @ @ @
@ 1 0 . . . @
@ @ @ . @ @ @
@ @ @ @ @ @ @

Goal locations
@ @ @ @ @ @ @
@ . . . 0 1 @
@ @ @ . @ @ @
@ @ @ @ @ @ @

***Run Prioritized***
Constraint: {'agent': 1, 'loc': [(1, 2)], 'timestep': 0, 'positive': False}
Constraint: {'agent': 1, 'loc': [(1, 3)], 'timestep': 1, 'positive': False}
Constraint: {'agent': 1, 'loc': [(1, 3), (1, 2)], 'timestep': 1, 'positive': False}
Constraint: {'agent': 1, 'loc': [(1, 4)], 'timestep': 2, 'positive': False}
Constraint: {'agent': 1, 'loc': [(1, 4), (1, 3)], 'timestep': 2, 'positive': False}
Constraint: {'agent': 1, 'loc': [(1, 4)], 'timestep': 3, 'positive': False}
Constraint: {'agent': 1, 'loc': [(1, 4)], 'timestep': 4, 'positive': False}
Constraint: {'agent': 1, 'loc': [(1, 4)], 'timestep': 5, 'positive': False}
Constraint: {'agent': 1, 'loc': [(1, 4)], 'timestep': 6, 'positive': False}
Constraint: {'agent': 1, 'loc': [(1, 4)], 'timestep': 7, 'positive': False}
Constraint: {'agent': 1, 'loc': [(1, 4)], 'timestep': 8, 'positive': False}
Constraint: {'agent': 1, 'loc': [(1, 4)], 'timestep': 9, 'positive': False}
Constraint: {'agent': 1, 'loc': [(1, 4)], 'timestep': 10, 'positive': False}
Constraint: {'agent': 1, 'loc': [(1, 4)], 'timestep': 11, 'positive': False}
Constraint: {'agent': 1, 'loc': [(1, 4)], 'timestep': 12, 'positive': False}
Constraint: {'agent': 1, 'loc': [(1, 4)], 'timestep': 13, 'positive': False}
Constraint: {'agent': 1, 'loc': [(1, 4)], 'timestep': 14, 'positive': False}
Constraint: {'agent': 1, 'loc': [(1, 4)], 'timestep': 15, 'positive': False}
Constraint: {'agent': 1, 'loc': [(1, 4)], 'timestep': 16, 'positive': False}
Constraint: {'agent': 1, 'loc': [(1, 4)], 'timestep': 17, 'positive': False}
Constraint: {'agent': 1, 'loc': [(1, 4)], 'timestep': 18, 'positive': False}
Constraint: {'agent': 1, 'loc': [(1, 4)], 'timestep': 19, 'positive': False}
Constraint: {'agent': 1, 'loc': [(1, 4)], 'timestep': 20, 'positive': False}
Constraint: {'agent': 1, 'loc': [(1, 4)], 'timestep': 21, 'positive': False}
Constraint: {'agent': 1, 'loc': [(1, 4)], 'timestep': 22, 'positive': False}
Constraint: {'agent': 1, 'loc': [(1, 4)], 'timestep': 23, 'positive': False}
Constraint: {'agent': 1, 'loc': [(1, 4)], 'timestep': 24, 'positive': False}
Constraint: {'agent': 1, 'loc': [(1, 4)], 'timestep': 25, 'positive': False}
Constraint: {'agent': 1, 'loc': [(1, 4)], 'timestep': 26, 'positive': False}
Constraint: {'agent': 1, 'loc': [(1, 4)], 'timestep': 27, 'positive': False}
Constraint: {'agent': 1, 'loc': [(1, 4)], 'timestep': 28, 'positive': False}
next
Traceback (most recent call last):
  File "run_experiments.py", line 106, in <module>
    paths = solver.find_solution()
  File "C:\Users\Fuzion PC\Desktop\CMPT 417\Individual project\code\prioritized.py", line 53, in find_solution
    raise BaseException('No solutions')
BaseException: No solutions
```

2.5. Here is an MAPF instance which prioritized planning does not find an (optimal or suboptimal) collision-free solution for a given ordering of the agents. This can only be solved with priority 1 > 2, but if we switch the priorities to 2 > 1, it cannot solve it.



Implementing Conflict-Based Search (CBS)

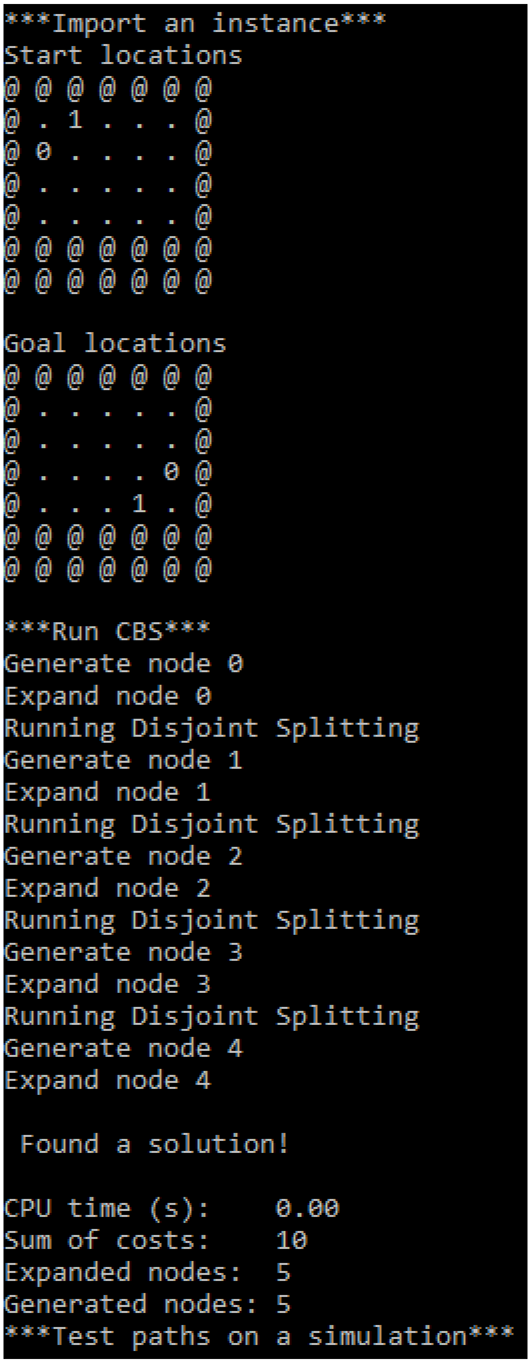
3.3. Here is the transcript:



```
***Import an instance***
Start locations
@ @ @ @ @ @ @
@ 0 1 . . . @
@ @ @ . @ @ @
@ @ @ @ @ @ @

Goal locations
@ @ @ @ @ @ @
@ . . . 1 0 @
@ @ @ . @ @ @
@ @ @ @ @ @ @

***Run CBS***
Generate node 0
Expand node 0
Expanding node {'cost': 6, 'constraints': [], 'paths': [[(1, 1), (1, 2), (1, 3), (1, 4), (1, 5)], [(1, 2), (1, 3), (1, 4)]],
'collisions': [{'a1': 0, 'a2': 1, 'loc': [(1, 4)], 'timestep': 3}]}
Generate node 1
Generate node 2
Expand node 1
Expanding node {'cost': 7, 'constraints': [{'agent': 0, 'loc': [(1, 4)], 'timestep': 3, 'positive': False}], 'paths': [[(1, 1
), (1, 2), (1, 3), (1, 3), (1, 4), (1, 5)], [(1, 2), (1, 3), (1, 4)]], 'collisions': [{'a1': 0, 'a2': 1, 'loc': [(1, 4)], 'ti
mestep': 4}]}
Generate node 3
Generate node 4
Expand node 2
Expanding node {'cost': 8, 'constraints': [{'agent': 1, 'loc': [(1, 4)], 'timestep': 3, 'positive': False}], 'paths': [[(1, 1
), (1, 2), (1, 3), (1, 4), (1, 5)], [(1, 2), (1, 3), (1, 4), (1, 3), (1, 4)]], 'collisions': [{'a1': 0, 'a2': 1, 'loc': [(1,
3), (1, 4)], 'timestep': 3}]}
Generate node 5
Generate node 6
Expand node 3
Expanding node {'cost': 8, 'constraints': [{'agent': 0, 'loc': [(1, 4)], 'timestep': 3, 'positive': False}, {'agent': 0, 'loc
': [(1, 4)], 'timestep': 4, 'positive': False}], 'paths': [[(1, 1), (1, 2), (1, 3), (1, 3), (1, 3), (1, 4), (1, 5)], [(1, 2),
 (1, 3), (1, 4)]], 'collisions': [{'a1': 0, 'a2': 1, 'loc': [(1, 4)], 'timestep': 5}]}
Generate node 7
Generate node 8
Expand node 6
Expanding node {'cost': 8, 'constraints': [{'agent': 1, 'loc': [(1, 4)], 'timestep': 3, 'positive': False}, {'agent': 1, 'loc
': [(1, 4), (1, 3)], 'timestep': 3, 'positive': False}], 'paths': [[(1, 1), (1, 2), (1, 3), (1, 4), (1, 5)], [(1, 2), (1, 3),
 (1, 3), (1, 3), (1, 4)]], 'collisions': [{'a1': 0, 'a2': 1, 'loc': [(1, 3)], 'timestep': 2}]}
Generate node 9
Generate node 10
Expand node 10
Expanding node {'cost': 8, 'constraints': [{'agent': 1, 'loc': [(1, 4)], 'timestep': 3, 'positive': False}, {'agent': 1, 'loc
': [(1, 4), (1, 3)], 'timestep': 3, 'positive': False}, {'agent': 1, 'loc': [(1, 3)], 'timestep': 2, 'positive': False}], 'pa
ths': [[(1, 1), (1, 2), (1, 3), (1, 4), (1, 5)], [(1, 2), (1, 3), (1, 4), (1, 5), (1, 4)]], 'collisions': [{'a1': 0, 'a2': 1,
 'loc': [(1, 4), (1, 5)], 'timestep': 4}]}
Generate node 11
Generate node 12
Expand node 12
Expanding node {'cost': 8, 'constraints': [{'agent': 1, 'loc': [(1, 4)], 'timestep': 3, 'positive': False}, {'agent': 1, 'loc
': [(1, 4), (1, 3)], 'timestep': 3, 'positive': False}, {'agent': 1, 'loc': [(1, 3)], 'timestep': 2, 'positive': False}, {'ag
ent': 1, 'loc': [(1, 5), (1, 4)], 'timestep': 4, 'positive': False}], 'paths': [[(1, 1), (1, 2), (1, 3), (1, 4), (1, 5)], [(1
, 2), (1, 3), (1, 2), (1, 3), (1, 4)]], 'collisions': [{'a1': 0, 'a2': 1, 'loc': [(1, 2), (1, 3)], 'timestep': 2}]}
Generate node 13
Generate node 14
Expand node 14
Expanding node {'cost': 8, 'constraints': [{'agent': 1, 'loc': [(1, 4)], 'timestep': 3, 'positive': False}, {'agent': 1, 'loc
': [(1, 4), (1, 3)], 'timestep': 3, 'positive': False}, {'agent': 1, 'loc': [(1, 3)], 'timestep': 2, 'positive': False}, {'ag
ent': 1, 'loc': [(1, 5), (1, 4)], 'timestep': 4, 'positive': False}, {'agent': 1, 'loc': [(1, 3), (1, 2)], 'timestep': 2, 'po
sitive': False}], 'paths': [[(1, 1), (1, 2), (1, 3), (1, 4), (1, 5)], [(1, 2), (1, 2), (1, 2), (1, 3), (1, 4)]], 'collisions'
: [{'a1': 0, 'a2': 1, 'loc': [(1, 2)], 'timestep': 1}]}
Generate node 15
Generate node 16
Expand node 16
Expanding node {'cost': 8, 'constraints': [{'agent': 1, 'loc': [(1, 4)], 'timestep': 3, 'positive': False}, {'agent': 1, 'loc
': [(1, 4), (1, 3)], 'timestep': 3, 'positive': False}, {'agent': 1, 'loc': [(1, 3)], 'timestep': 2, 'positive': False}, {'ag
ent': 1, 'loc': [(1, 5), (1, 4)], 'timestep': 4, 'positive': False}, {'agent': 1, 'loc': [(1, 3), (1, 2)], 'timestep': 2, 'po
sitive': False}, {'agent': 1, 'loc': [(1, 2)], 'timestep': 1, 'positive': False}], 'paths': [[(1, 1), (1, 2), (1, 3), (1, 4),
 (1, 5)], [(1, 2), (1, 3), (2, 3), (1, 3), (1, 4)]], 'collisions': []}

 Found a solution!

CPU time (s):    0.01
Sum of costs:    6
Expanded nodes:  9
Generated nodes: 17
***Test paths on a simulation***
```

Implementing CBS with Disjoint Splitting

4.3. With standard splitting my implementation generated 29 nodes, expanding 15, with a sum of costs at 10. Disjoint splitting generated 5 nodes, expanding 5 nodes, with a sum of costs at 10 as well. Below is a picture of the output for disjoint splitting:



```
***Import an instance***
Start locations
@ @ @ @ @ @ @
@ . 1 . . . @
@ 0 . . . . @
@ . . . . . @
@ . . . . . @
@ @ @ @ @ @ @
@ @ @ @ @ @ @

Goal locations
@ @ @ @ @ @ @
@ . . . . . @
@ . . . . . @
@ . . . . 0 @
@ . . . 1 . @
@ @ @ @ @ @ @
@ @ @ @ @ @ @

***Run CBS***
Generate node 0
Expand node 0
Running Disjoint Splitting
Generate node 1
Expand node 1
Running Disjoint Splitting
Generate node 2
Expand node 2
Running Disjoint Splitting
Generate node 3
Expand node 3
Running Disjoint Splitting
Generate node 4
Expand node 4

 Found a solution!

CPU time (s):    0.00
Sum of costs:    10
Expanded nodes:  5
Generated nodes: 5
***Test paths on a simulation***
```