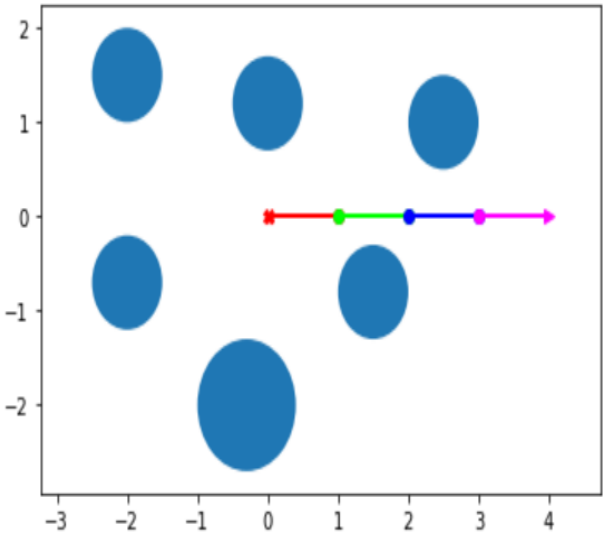
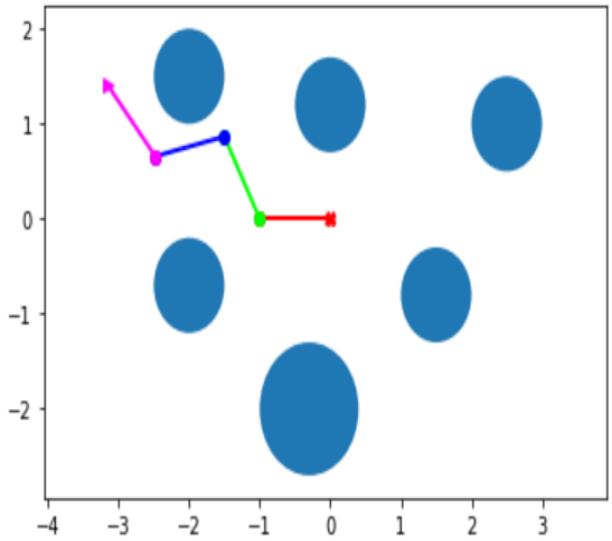
Task 1

A. Visualize the manipulator in the start state and target state. Comment on your thought comparison the discretized orientation space from PS1 vs continuous orientation space in current problem set.

Visualize the manipulator in the start state and target state



Pic1. Start state



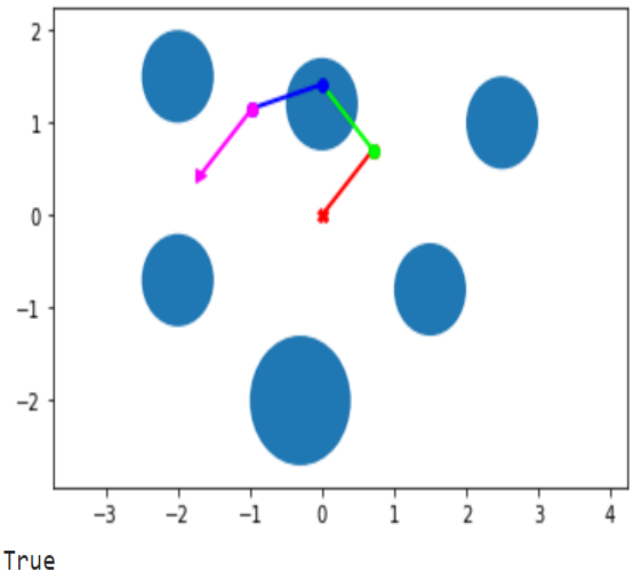
Pic2. Goal state

Comment on your thought comparison the discretized orientation space from PS1 vs continuous orientation space in current problem set.

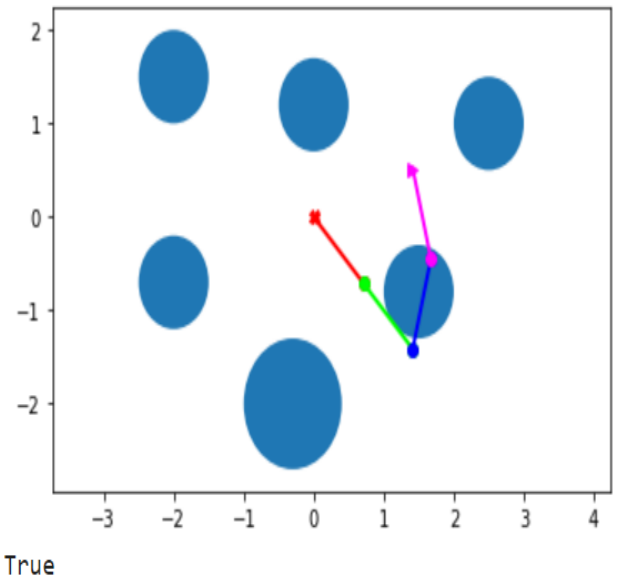
The discretized orientation space from PS1 is small, so it leads to visiting an unnecessary number of states and takes a larger memory than the continuous orientation space in current problem set.

B. Visualize the manipulator in 4 random orientations that include both colliding and non-colliding configurations. Check what does the ManipulatorEnv.check_ collision function returns for those configurations. Comment on your observations.

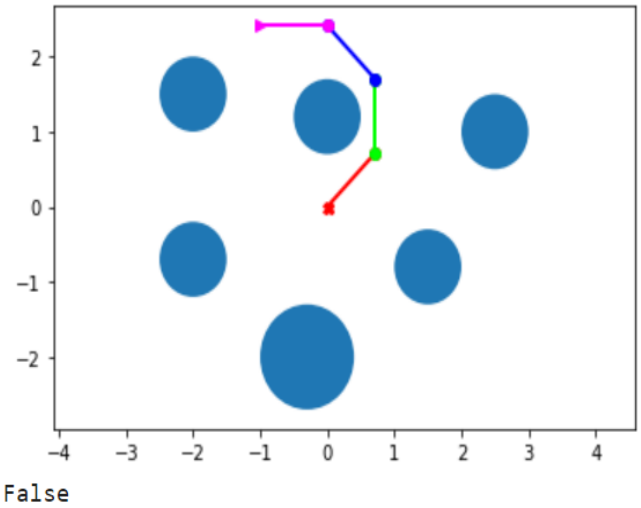
Visualize the manipulator in 4 random orientations that include both colliding and non-colliding configurations.



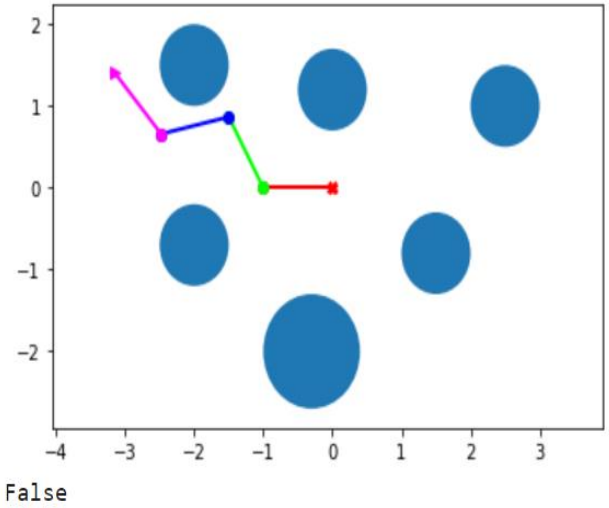
Pic3. Colliding configurations
(angles = `[45,90,60,30]`)



Pic4. Colliding configurations
(angles = `[-45,0,120,30]`)



Pic5. Non-colliding configurations
(angles = `[45,45,45,45]`)



Pic6. Non-colliding configurations
(angles = `[-180,-60,72,-60]`)

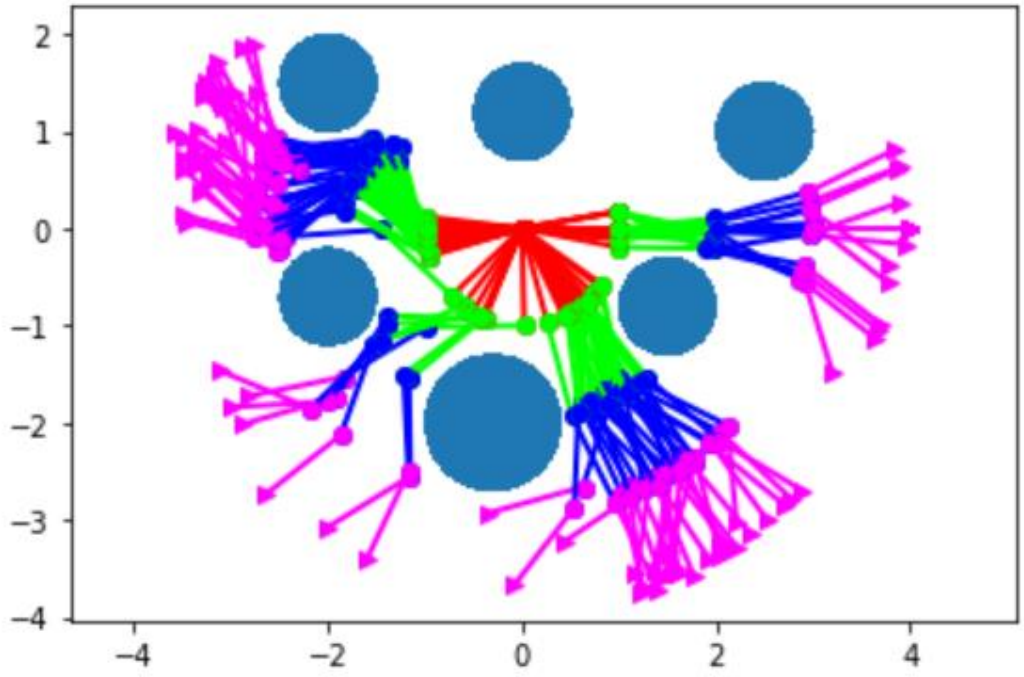
Comment on your observations

The manipulator is fixed in (0,0) because in 4 random orientations that include both colliding and non-colliding configurations still have the same coordinate of root (0,0), while 3 links are in different coordinate.

Task 2

A. You need to implement the RRT algorithm for agent in continuous domain. The starting configuration of the agent is (0, 0, 0, 0) and the goal configuration is (-180.0, -60.0, 72.0, -60.0). For searching nearest pose use L1 distance between two configuration vectors.

```
The number of states: 210
The final state: [180. -60.  72. -60.]
RRT planner has finished successfully
```



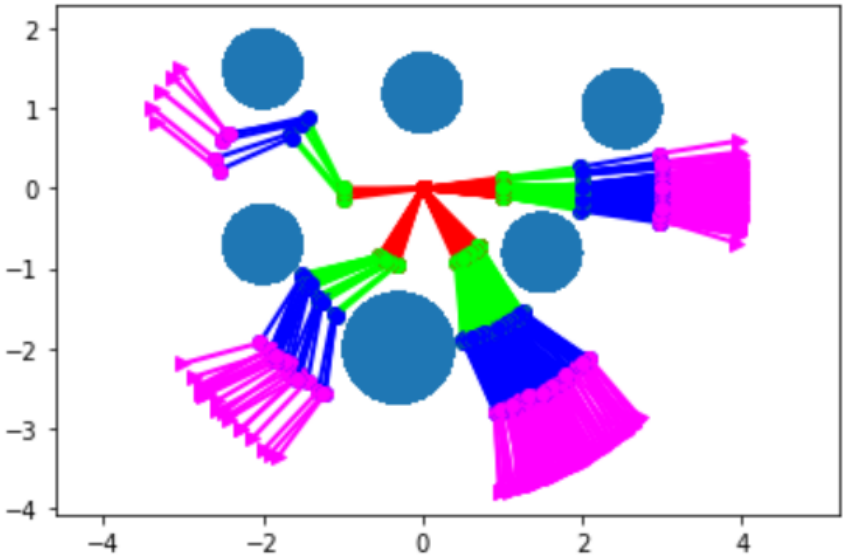

```
<Figure size 432x288 with 0 Axes>
```

B. Comment on how many states have been visited? What is the final trajectory size? Can you comment on the optimality of the plan? You can also collect some observations and statistics across multiple runs.

210 states have been visited. The final trajectory size is 210×4. I optimize by using goal bias sampling method. It adds a higher probability for goal configuration point on top of the uniform sampling.

C. Try to change weight of rotation in calculation of distance between two agent positions. We suggest you to build a distance function based on weighted sum of the angle distances. Comment on the results.

```
The number of states: 555
The final state: [-179.95143869  -67.33375515   80.68644136  -56.06875782]
[array([-180.,  -60.,   72.,  -60.]),
 <__main__.Tree at 0x7ff932859b90>,
 <__main__.Tree at 0x7ff939eb2810>]
```

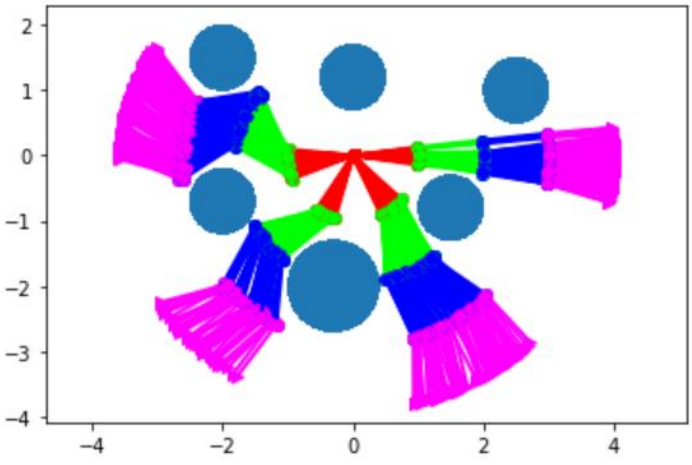


My first distance function compares angles of near_tree with 10 degrees and chooses the smaller to become new_state.

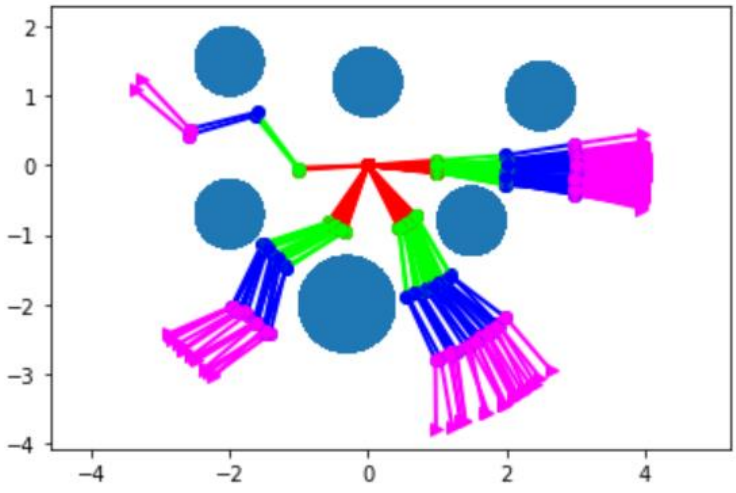
My second distance function creates new_state by adding $10\times\frac{dist}{||dist||_1} \leq 10$ (where dist - angle different between near_tree and random_tree) to angles of near_tree.

The first method returns the correct value of final state [-180.0, -60.0, 72.0, -60.0], while the second's final state approximates [-179.95143869, -67.33375515, 80.68644136 -56.06875782]

D. Try to change step size used for RRT branches. Comment on the results.



Pic7. Step size = 5



Pic8. Step size = 15

When step size is decreased from 10 to 5, the final trajectory size is bigger. Besides that, the program runs slower and does not reach to the goal state.

When step size is increased from 10 to 15, the final trajectory size is smaller. Program run faster but the distances between angles are bigger.