# 3.8

**a.** Because some path may not seem like a good choice, but it may have an arbitrarily negative cost, on the other words, a reward to choose it. So it need to explore the entire state space to get the optimal path.

**b.** Graphs: No, it doesn’t help. Because we may get a loop and the loop includes the negative costs C. Then it could get C every time.

Tree: Yes, if we could know the depth of the tree. If we could know the depth of the tree, we could know the steps. If the steps are x, the most negative we could get is xc. So we could ignore the costs more than xc.

**c.** It should just go around the loop forever.

**d**. When people see the beautiful view the first time, they could get a negative cost, but if they see the view again and again, they may feel tedious and do not get any negative cost. So human do not drive a loop indefinitely. So we could develop a memory function of the clients. If it has been a state before, the negative costs decrease or disappear, then we could avoid the clients drive a loop again and again.

**e.** Go to class and back home.

# 3.9

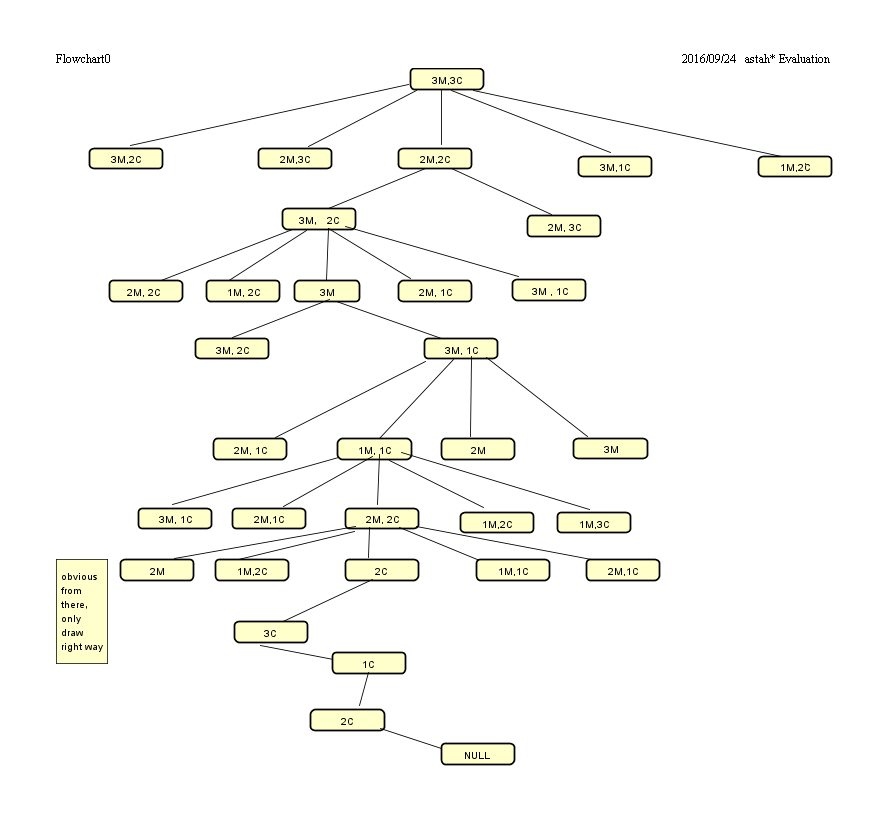
**a.** Initial Condition: 2 lists: each of them have 3 tuples to conserve the missionaries and cannibals, a boat, 2 sides. All people and the boat are from the side 1.

Goals: move all people to the side 2.

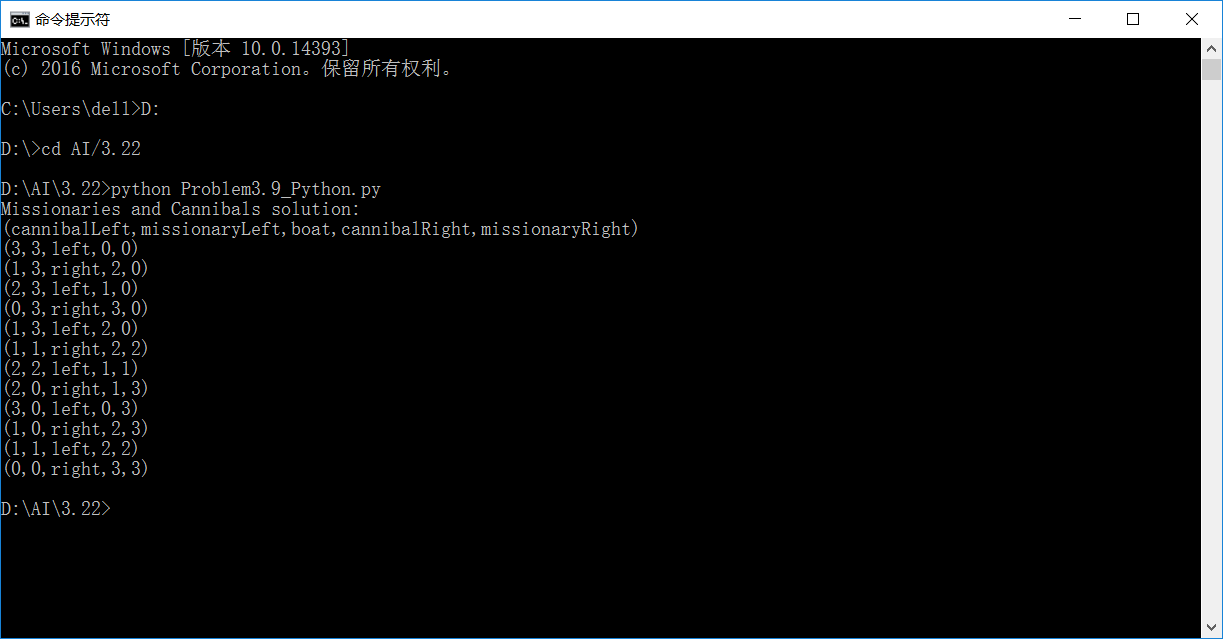
Cost: Each move of the boat from one side to another.

Condition: Each move could move 1 or 2 people from missionaries and 3 cannibals; the number of people from list missionaries cannot less than list cannibals of each sides.

**b.** Suppose the list of missionaries is M and cannibals is C, then I use the search algorithm. The search is as follows:



The result of the implementation is as follows:



According to the search algorithm, search the repeated state is meaningless.

**c.** I think because it is not easy to decide if the move is legal and it is easy to move back the previous place.

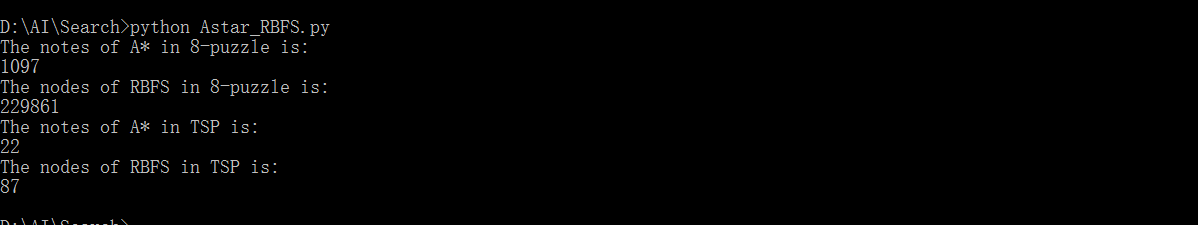
# 3.21

**a**. When the steps cost is the equal, the uniform-cost search is the same with the breadth-first search.

**b**. If , the depth-first search is best-first search.

**c.** If , the uniform-cost search is A∗ search.

# 3.22

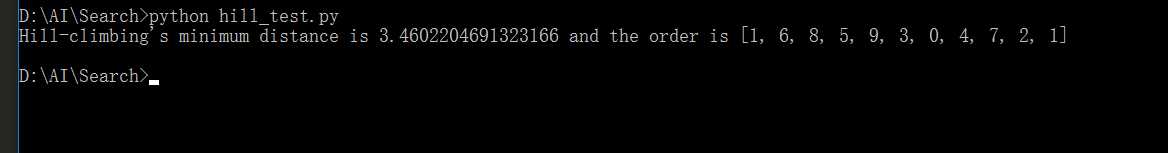


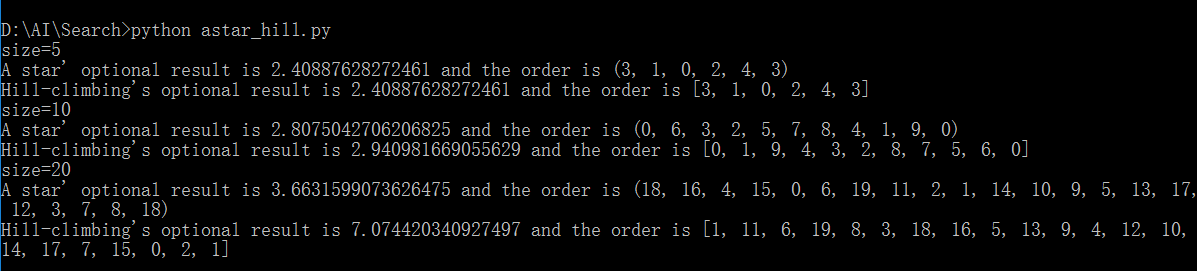
According to the consequence in 8-puzzle, RBFS need expands more nodes than A\* and spend more time than A\*. But according to the algorithm, the cost of nodes in RBFS is lower because it does not need to maintain all of them. When a small random number is added to the heuristic values in the 8-puzzle domain, RBFS become worse than A\* because the tired value does not exist, so it has to expansion much more than A\*

In the TSP problem, the space is a tree, so the nodes of RBFS is almost the same with A\*, because the repeated state do not need to be expend.

# 4.3

1. The output of Hill-Climbing algorithm is as follow:

 Compare of A\* and Hill-climbing:



1. Hill-climbing has a simple structure than A\*
2. A\* and Hill-climbing both could get a good result when the size is small.
3. When the size become bigger, the expression of A\* is obvious better than Hill-climbing.
4. The output of genetic algorithm is as follow:

