# Assignment2

Boya Zhou

Chapter 3, Ex. 3.8

a. For any optimal algorithm(we learn in class and textbook), it will first “judging" the nodes(by distance and other methods), try to pick the optimal one. However, if actions can have arbitrary large negative cost,  this cost has no relationship with the way optimal algorithm judging the nodes, so this “judging” method have no use at all. So if a optimal algorithm want to find the optimal one, it has to search the whole state space.
b. If the state has loop path, this condition will not guarantee any help. If there is no loop path, assume **m** is maximum depth of the state space, as least we can prune the path which (f(n) + mc) is greater than zero.
c. The agent will be trapped in this loop forever.
d. The freshness of scenic road will decrease, like marginal effect. With the number of times people drive on one road, the utility will decrease, too. So the absolutely negative cost, means abs(c), will decrease, too. To avoid looping, one can design a state space with memory: a state is now represented not just by the current location, but also a bag of already visited  locations, the rewards of visiting a scenic road will decrease with the number of time visiting it.
e. Going to class.

Chapter 3, Ex. 3.9

a. The initial state are two bags, first bag contain 3 missionaries, 3 cannibals and one boat, second bag is empty. The goal state should be first bag is empty and second bag contain 3 missionaries, 3 cannibals plus the boat. The action can be move 1 or 2 element(except boat) in bag to another through boat, meaning any action must taken from the bag with boat. Each action cost 1. The successors of all states are the states take actions but fit the restrictions, means the number of cannibals can not be larger than the number of missionaries in any bags except the number of missionaries is 0.
b. Any optimal algorithm will work. For simplicity, I use the picture to show how it works. The picture I get from: http://www.aiai.ed.ac.uk/~gwickler/missionaries.html. In this picture, the whole search space is here. And red dot

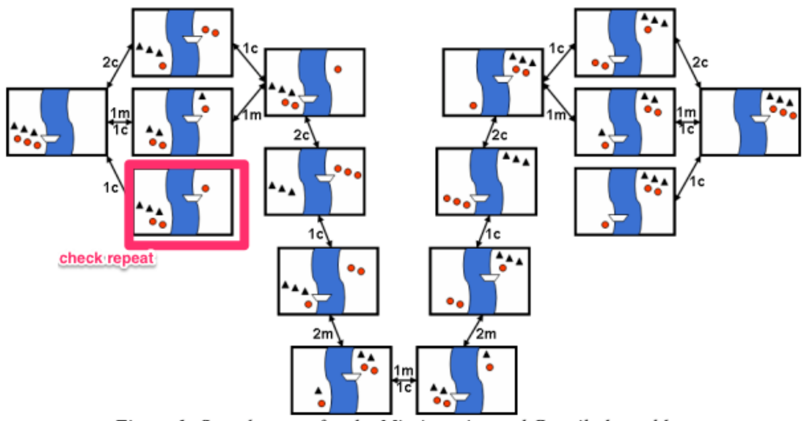

*Figure 1: Search-space for the Missionaries and Cannibals problem*

means cannibals, black triangle means missionaries. As we can see, we can easily solve the problem if follow the BFS method. As for check for repeat, after we explore the pink rectangle place, we meet a dead end, so we need to check repeat, at other place, no need to do that, since except first and last state, there is only one other choice.

c. One reason maybe: there is only one right successor in middle part, if make wrong choice, rather revert to previous state or the game fail. This may make people confused.

Chapter 3, Ex. 3.21

a. Each branch in uniformed-tree have its own weight, the weight between different branches can be different or equal. Branches of BFS have equal weight. Other aspect are all the same. So BFS search is a special case of uniformed-cost search.
b. The best-first expand the node with least h(n).In depth first search, this evaluation function, h(n) = -depth(n). Other aspects are same. So depth-first search is a special case of best-first search tree.
c. In A* search, the evaluation function, f(n) = g(n) + h(n). In uniformed-cost search, f(n) = g(n), h(n) is always zero. So uniformed-cost search is a special case of A* search.

Chapter 3, Ex. 3.22

a. Three compare:
   1. A* with RBFS in 8 puzzles:
      i. A* store more nodes than RBFS. Cost more memory. RBFS do not need to store a queue.
      ii. RBFS re-expand more nodes(not that much because the best path seldom change) than A* since there are many tied values in 8 puzzle and RBFS do not detect repeated values.
   2. A* with RBFS in TSP:
      i. A* store more nodes than RBFS. Cost more memory. RBFS do not need to store a queue.
      ii. Since the state space of TSP is a tree, there are seldom tied values, so RBFS will get more penalty from re-generate nodes.
b. If add random number to 8 puzzle domain: It will disturb the work of RBFS and make RBFS performance much worse. Because the slightly random number will decide the search of RBFS.

Chapter 4, Ex. 4.3

Compare the optimal solution between A* and hill-climbing.

A* with MST as its heuristic function can always have equal or better solution than hill-climbing. Since the solution of hill-climbing depends on its initial states.