CS 5100 – PS1

Yijing Xiao

Problem 1. In what order does breadth-first search (BFS), depth-first search (DFS) and uniform-cost search (UCS) expand the nodes for each graph?

	BFS		DFS		UCS	
Iter	Queue	CurrNode	Stack	CurrNode	Queue	CurrNode
0.	s		s		s	
1.	a, b	s	a, b	s	a(1), b(1)	s
2.	b, c	a	a, c, d, g	b	b(1), c(2)	a
3.	c, d, g	b	a, c, d	g	c(2), g(5), d(3)	b
4.	d, g	c			g(5), d(3)	c
5.	g	d			g(4)	d
6.		g				g(4)

Extra explanation: Since this is graph search, we have explored set to make sure that we don't add same node repeatedly. For DFS, since we reach out g quickly, we don't have to go through all iterations even though the frontier/ stack is not empty. For UCS, our final path cost is 4.

Problem 2. Three mice and three cats are on one side of a river, along with a boat that can hold one or two animals.

(a) Formulate the problem precisely, making only those distinctions necessary to ensure a valid solution.

State space:

Define mice as M, cat as C, and boat as B. When B = 1, it means that the boat in the river left side and B = 0 means in another side. We have limited that is the number of M must be bigger than or equal to the number of C and the number of animals in the boat must be smaller than 2. The initial state is 3 M, 3 C and B in the left side and the goal test is all Ms and Cs are on the other side. The path cost would be each one per action.

(M, C, B) left side	(M, C, B) right side
(0, 0, 1)	(0, 0, 0)
(0, 1, 1)	(0, 1, 0)
(0, 2, 1)	(0, 2, 0)
(0, 3, 1)	(0, 3, 0)
~~(1, 0, 1)~~ invalid	~~(1, 0, 0)~~ invalid

(1, 1, 1)	(1, 1, 0)
~~(1, 2, 1)~~ invalid	~~(1, 2, 0)~~ invalid
~~(1, 3, 1)~~ invalid	~~(1, 3, 0)~~ invalid
~~(2, 0, 1)~~ invalid	~~(2, 0, 0)~~ invalid
~~(2, 1, 1)~~ invalid	~~(2, 1, 0)~~ invalid
(2, 2, 1)	(2, 2, 0)
~~(2, 3, 1)~~ invalid	~~(2, 3, 0)~~ invalid
(3, 0, 1)	(3, 0, 0)
(3, 1, 1)	(3, 1, 0)
(3, 2, 1)	(3, 2, 0)
(3, 3, 1)	(3, 3, 0)

(b) Write out the four components of the search problem for your formulation.

According to our problem precisely in part a, we have:

1) Initial State: (3, 3, 1)

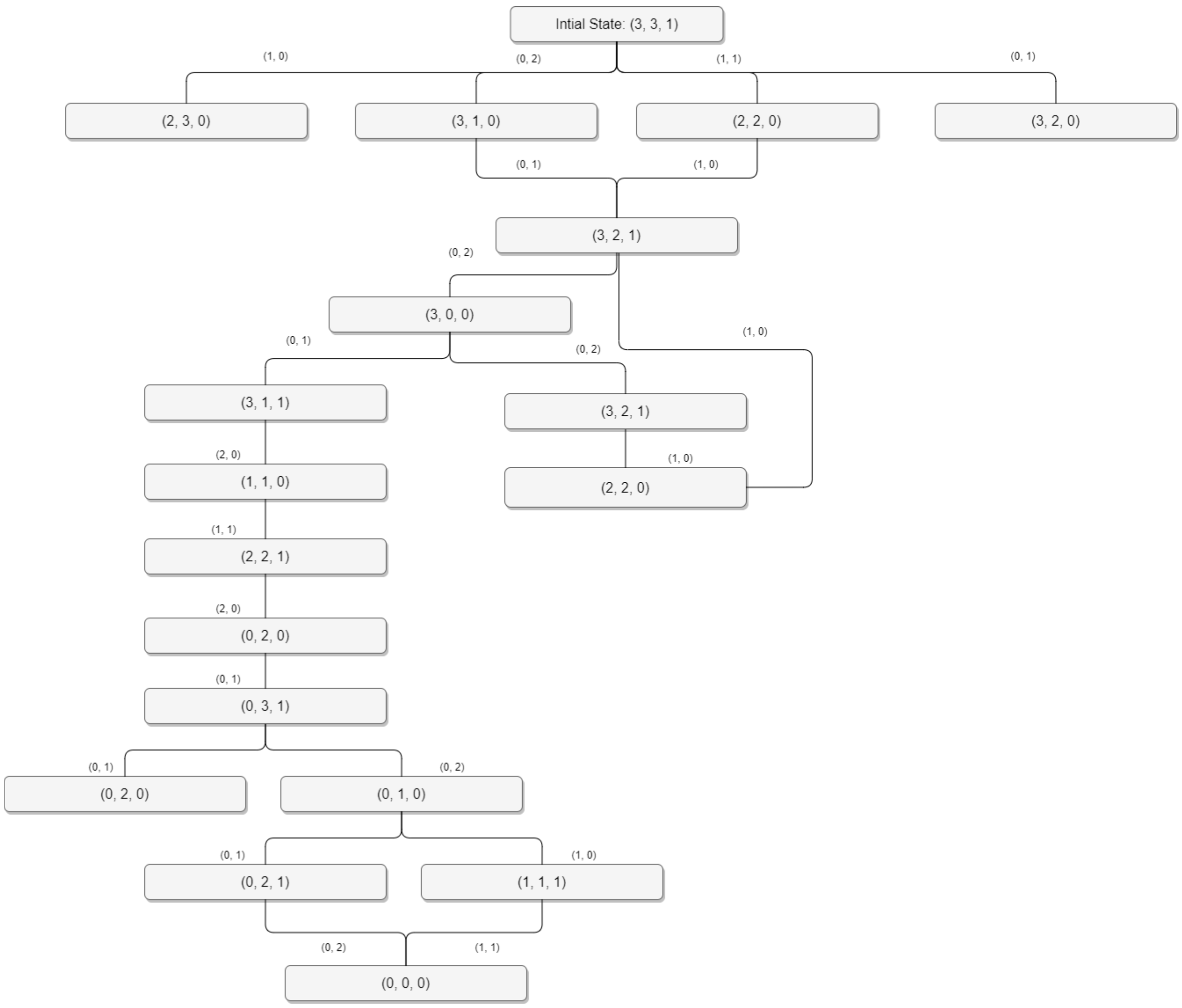
2) Successor Function: Assume that ML means the number of mice in the left side, CL means the number of cats in the left side.

Actions (M in B, C in B)	Transition function (left to right/ right to left)	
(2, 0)	(ML-2, CL, 0)	(ML+2, CL, 1)
(0, 2)	(ML, CL-2, 0)	(ML, CL+2, 1)
(1, 1)	(ML-1, CL-1, 0)	(ML+1, CL+1, 1)
(1, 0)	(ML-1, CL, 0)	(ML+1, CL, 1)
(0, 1)	(ML, CL-1, 0)	(ML, CL+1, 1)

3) Goal State: (0, 0, 0)

4) Path Cost Function: Each time the boat crosses the river will cost 1 and the total path cost would be the summation of the step.

5) Draw a diagram of the valid state space (shown in the next page):



(c) Can you find a solution in this space? If so, what solution have you found?

Yes, I could find at least four solutions in this space:

1) Initial State (3, 3, 1), (3, 1, 0), (3, 2, 1), (3, 0, 0), (3, 1, 1), (1, 1, 0), (2, ,2, 1), (0, 2, 0), (0, 3, 1), (0, 1, 0), (0, 2, 1), Goal State (0, 0, 0)

2) Initial State (3, 3, 1), (3, 1, 0), (3, 2, 1), (3, 0, 0), (3, 1, 1), (1, 1, 0), (2, ,2, 1), (0, 2, 0), (0, 3, 1), (0, 1, 0), (1, 1, 1), Goal State (0, 0, 0)

3) Initial State (3, 3, 1), (2, 2, 0), (3, 2, 1), (3, 0, 0), (3, 1, 1), (1, 1, 0), (2, ,2, 1), (0, 2, 0), (0, 3, 1), (0, 1, 0), (0, 2, 1), Goal State (0, 0, 0)

4) Initial State (3, 3, 1), (2, 2, 0), (3, 2, 1), (3, 0, 0), (3, 1, 1), (1, 1, 0), (2, ,2, 1), (0, 2, 0), (0, 3, 1), (0, 1, 0), (1, 1, 1), Goal State (0, 0, 0)

Optional Problem 3: AIMA 3.10. So far, we have assumed non-negative edge costs. In this exercise, we explore this decision in more depth.

(a) **Suppose that actions can have arbitrarily large negative costs; explain why this possibility would force any optimal algorithm to explore the entire state space.**

Since the costs are negative, it’s hard to say that an unexplored node will have a path which make total cost less than current best cost or not unless we explore the entire state space.

(b) **Does it help if we insist that step costs must be greater than or equal to some negative constant −e < 0 (i.e., step costs are not arbitrarily negative)?**

If the state spaces have loop, it could go over the loop many times and always reduce the current best path cost. Hence, it still not helpful.

(c) **What if, in addition to the assumption in part (b), we assumed that the search graph is a (directed) tree with maximum depth m?**

Since we know the maximum depth m, we would know that the possible minimum path cost is -e * m for remaining nodes. We don’t need to go through any path if current path cost is -e * m because its’ve already minimum.