Uninformed Search– Solution

1) (Thanks to Ariel Felner.) Three missionaries and three cannibals must cross a river from its left bank to its right bank using a boat which can carry at most two people, under the constraint that, for both banks, if there are missionaries present on the bank, they cannot be outnumbered by cannibals (if they were, the cannibals would eat the missionaries). The boat cannot cross the river by itself with no people on board. In addition, when a boat arrives at a destination bank then anyone inside the boat is considered to be on the destination bank (even if the boat returns to the other bank right away). How can the 6 people move without violating the constraint?

a) Express this problem as a state space as follows: Define the different states, define the different operators and their applicability (do not allow states that violate the constraints) and define the start and goal states.

Answer:
One way of defining the state space is as follows.

States: We can represent a state as the 5-tuple (C_L, M_L, C_R, M_R, B) where C_L denotes the number of cannibals on the left bank, M_L denotes the number of missionaries on the left bank, C_R denotes the number of cannibals on the right bank, M_R denotes the number of missionaries on the right bank, and B denotes the position of the boat – either L (left bank) or R (right bank). Note that the following should hold for any valid state:

$C_L + C_R = M_L + M_R = 3$
if $M_L > 0$, then $M_L \geq C_L$
if $M_R > 0$, then $M_R \geq C_R$

Note: One could add additional constraints (for example, if $C_R + M_R = 0$, then $B = L$).

Start state: $(3, 3, 0, 0, L)$.

Goal state: $(0, 0, 3, 3, R)$.

Operators: We have five operators:

M: Move one missionary. Applicable if $B = L$ and ($M_L = 1$ or $M_L > C_L$), or if $B = R$ and ($M_R = 1$ or $M_R > C_R$).

C: Move one cannibal. Applicable if $B = L$, $C_L > 0$, and ($M_R = 0$ or $M_R > C_R$), or if (symmetric formula for $B = R$).

MM: Move two missionaries. Applicable if $B = L$ and ($M_L = 2$ or $M_L > C_L + 1$), or if (symmetric formula for $B = R$).

CC: Move two cannibals. Applicable if $B = L$, $C_L > 1$ and ($M_R = 0$ or $M_R < C_R - 1$), or if (symmetric formula for $B = R$).

MC: Move one cannibal and one missionary. Applicable if $B = L$, $M_L > 0$, and $C_L > 0$, or if (symmetric formula for $B = R$).

b) How many different states are possible?

Answer:
There are 4 different ways how we can distribute the missionaries (or cannibals) among the two banks, and two different positions for the boat. Therefore, there are a total of $4 \times 4 \times 2 = 32$ states. However, some of these states violate the constraint of the problem. Removing these states, we are left with 20 states that do not violate the constraint, which can be calculated as follows:

If $M_R = 0$ (and $M_L = 3$), then $C_R \in \{0, 1, 2, 3\}$ – 4 possibilities.
If $M_R = 1$ (and $M_L = 2$), then $C_R = 1$ (and $C_L = 2$) – 1 possibility.
If $M_R = 2$ (and $M_L = 1$), then $C_R = 2$ (and $C_L = 1$) – 1 possibility.
If $M_R = 3$ (and $M_L = 0$), then $C_R \in \{0, 1, 2, 3\}$ – 4 possibilities.

There are only $4+1+1+4 = 10$ valid distributions of missionaries and cannibals among the two banks, and 2 positions for the boat, for a total of $10 \times 2 = 20$ valid states.

c) Write a shortest sequence of states (path in the state space) that will solve the problem.

Answer:
$(3, 3, 0, 0, L)$ (apply CM)
$(2, 2, 1, 1, R)$ (apply M)
$(3, 2, 0, 1, L)$ (apply CC)
$(3, 0, 0, 3, R)$ (apply C)
$(3, 1, 0, 2, L)$ (apply MM)
$(1, 1, 2, 2, R)$ (apply CM)
$(2, 2, 1, 1, L)$ (apply MM)
$(0, 2, 3, 1, R)$ (apply C)
$(0, 3, 3, 0, L)$ (apply CC)
$(0, 1, 3, 2, R)$ (apply C)
$(0, 2, 3, 1, L)$ (apply CC)
$(0, 0, 3, 3, R)$

2) A 4-neighbor gridworld is given below. In which order do depth-first search and breadth-first search (both with a sensible node pruning strategy) expand the cells when searching from s to g? Ties are broken in lexicographic order. That is, A1 is preferred over A2 and B1, and A2 is preferred over B1.

	A	B	C	D	E
1					s
2	■	■		■	
3	g			■	
4				■	
5					

Answer:
The node pruning strategy of breadth-first search is to prune nodes whenever some node in the search tree is already labeled with the same state. The node pruning strategy of depth-first search is to prune nodes whenever some node on the same branch of the search tree is already labeled with the same state (cycle detection).

Breadth-first search: (E1), (D1, E2), (C1, E3), (B1, C2, E4), (A1, C3, E5), (B3, C4, D5), (A3). The parentheses group states based on their depth in the search tree. Depending on the tie-breaking strategy, the order of expansions can change, but only within a group.

Depth-first search: E1, D1, C1, B1, A1, (backtrack to B1 and then C1), C2, C3, B3, A3.

3) Compare the advantages and disadvantages of breadth-first and depth-first search and discuss to which degree pruning of tree nodes is important for them.

Answer:
In a finite state space, breadth-first search (BFS) and depth-first search (DFS) compare as follows:

- Both breadth-first search and depth-first search can use node pruning to decrease the number of expanded nodes and thus increase their efficiency. Breadth-first search needs to keep more information in memory than depth-first search and can then use more powerful node pruning strategies.
- BFS is complete (even without node pruning). DFS is not complete without cycle detection and thus not complete without any node pruning.
- BFS is optimal (assuming unit-cost edges), DFS offers no such guarantee.

In a finite tree with branching factor b, goal depth d, tree depth m, BFS and DFS compare as follows:

- In the worst case, BFS expands $O(b^d)$ states to find a solution, whereas DFS expands $O(b^m)$ states. Therefore, since the goal depth cannot be larger than the tree depth (that is, $d \leq m$), BFS has a better worst case performance.
- In the worst case, BFS requires memory to store $O(b^d)$ states, whereas DFS requires memory to store only $O(bm)$ states with an appropriate memory-deallocation strategy.

4) Does depth-first search always terminate if there is a path of finite length from the start to the goal? Why?

Answer:
No. If it does not use proper cycle detection, it might get stuck in a loop. Even with cycle detection, if the state space is infinite (but there is a finite length path from the start to the goal), DFS might get stuck exploring an infinite subspace of the state space.

5) In Manhattan, you want to reach a given destination from your current location with as few left turns as possible. Can this be formulated as finding a minimum cost path in a graph? If so, how? If not, why not?

Answer:
Yes. Each state is a pair of the agent's current location and the compass direction it is facing (North, East, South or West). There are four goal states, namely pairs of the given destination and the four compass directions. The actions are to 'move forward' (cost 0, changes location depending on which way the agent is facing, the direction the agent is facing does not change), 'turn right and move forward' (cost 0) and 'turn left and move forward' (cost 1). If we also want to minimize the number of edges traversed by the agent as a secondary objective, we can instead use a cost of ϵ for the 'move forward' and 'turn right and move forward' actions, where ϵ is smaller than 1/the number of states in our state space. Assuming Manhattan is finite, our state space representation is also finite.