Academic Integrity Statement

If your written submission on Learn does not include this academic integrity statement with your signature (typed name), we will deduct 5 marks from your A1 final mark.

I declare the following statements to be true:

- The work I submit here is entirely my own.
- I have not shared and will not share any of my code with anyone at any point.
- I have not posted and will not post my code on any public or private forum or website.
- I have not discussed and will not discuss the contents of this assessment with anyone at any point.
- I have not posted and will not post the contents of this assessment and its solutions on any public or private forum or website.
- I will not search for assessment solutions online.
- I am aware that misconduct related to assessments can result in significant penalties, possibly including failure in the course and suspension. This is covered in Policy 71: https://uwaterloo.ca/secretariat/policies-procedures-guidelines/policy-71.

Failure to accept the integrity policy will result in your assignment not being graded.

By typing or writing my full legal name below, I confirm that I have read and understood the academic integrity statement above.

Thien An Nguyen

1 a/ State : Given in the form $S_L W_L B S_R W_R$ where S and W are # of sheep and wolves respectively and L and R denote which bank they are on. A state is valid if the following

$$S_L + S_R = 3 \quad \text{and} \quad W_L + W_R = 3$$

B denotes boat position, 0 for left, 1 for right.

$$0 \leq S_L, S_R, W_L, W_R \leq 3$$

Successor: Given state A and B, assume they are valid.
If A has $S_L > 0$, A must have $S_L \geq W_L$ and if A has $S_R > 0$, A must have $S_R \geq W_R$.
If A does not, A is an end node and has no successors.
Otherwise B is a successor of A if we can move from A to B by one movement of the boat from A to B.

b/ The problem with Taylor's formulation is that he treats each sheep and wolf as a unique entity. He essentially names each sheep and wolf and treats them uniquely instead of in a group. This significantly affects the performance of the search algorithm since it creates states that could be treated the same as different entities to be treated differently. For instance any state with 3 sheeps on the left and one wolf likewise and boat on left can be treated the same and does not need to be split up like how Taylor does. In the example it should be a single state that we can see if it has been explored wheras Taylor will have to push atleast 3 states onto the explored set. This means he revisits what are essentially the same node multiple times.

1c) I believe that Taylor's formulation is better.
His is better as Taylor only pushes onto the frontier states that have successors that can be expanded since he checks whether or not they fulfill the constraints before putting them on the frontier. This results in calling the successor function less often and has less nodes in the search graph. Avery does not check the constraints before adding nodes to the frontier and only declares a node to be a dead end when calling the successor function on it. This results in a lot of nodes that do not fulfill the constraints being put on the frontier. This results in a larger search tree with more dead end nodes.

1d/ State: Each state is given by $S_L W_L B S_R W_R$. S and W denote the number of sheep and wolves respectively. The subscript denotes the side it is on, L for left and R for right. The values can be one of {0,1,2,3}. B denotes the position of the boat; B can be either 0 for left or 1 for right.

The state is valid if and only if it satisfies the following constraints

$W_L + W_R = 3$; $S_L + S_R = 3$

If there is at least one sheep on the left bank, then the number of wolves on the left bank must be less than or equal to the number of sheep on the left bank.

If there is at least one sheep on the right bank, then the number of wolves on the right bank must be less than or equal to the number of sheep on the right bank.

initial state: 33000

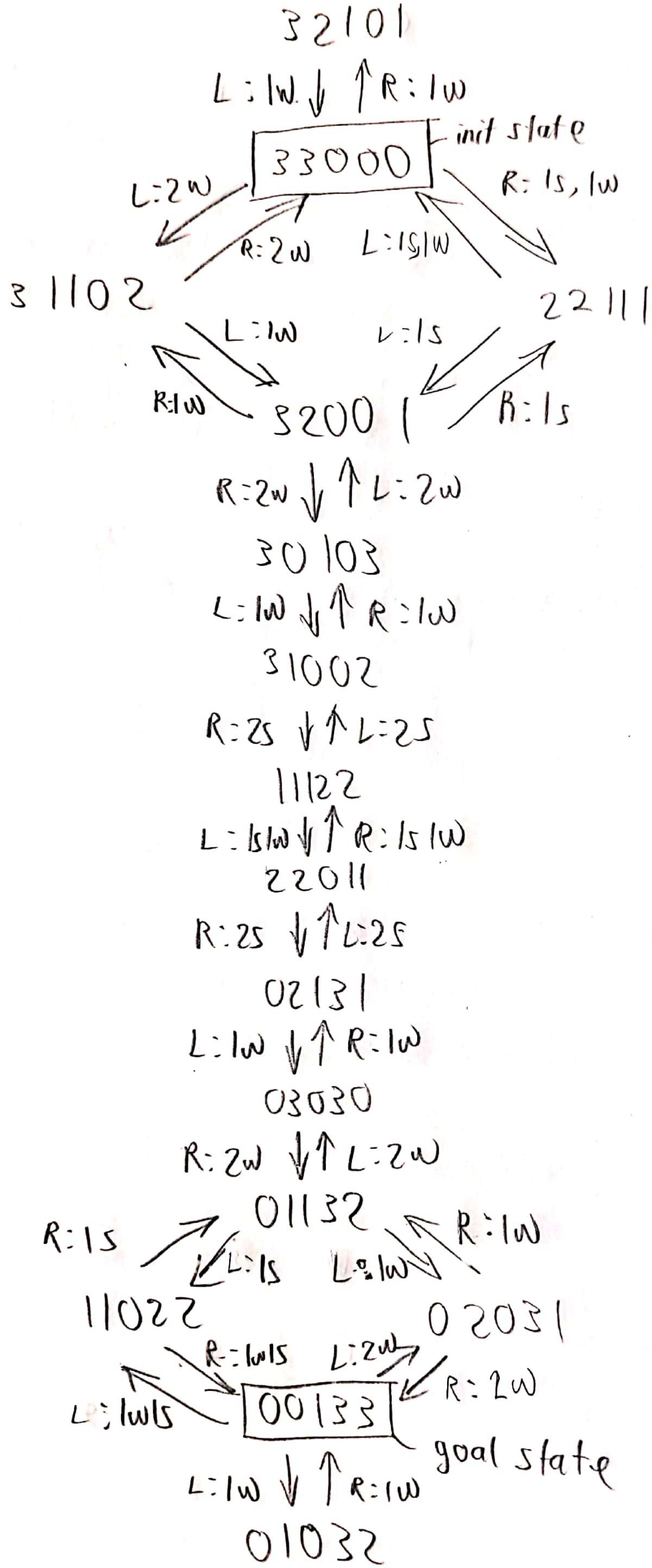
Goal state: 00133

Successor function: Assume that A and B are two states satisfying the state definition. B is a successor of A if and only if we can start at A, move the boat with one or two animals from one side of the river to the other side of the river, and arrive at state B.

Cost function: the cost of each boat trip is one.

7e/

format: $S_L W_L B S_R W_R$
direction: animals moved
Legend



f/ The search algorithm that I would choose is depth first search with multipath pruning. We use multipath pruning as a successor of the current node could be it's parent and we do not want to search the parent again, thus multipah is appropriate. We also want to use DFS since given the search tree we can go down the optimal and quickes path on the first try. Although it doesn't guarentee we will, it will be better than BFS or IDS or LCFS as none of those could allow us to go down the optimal route quickly.

g/ To generate an admissable heuristic we want to solve a relaxed version of the problem. We can relax the constraint of wolves eating sheep. This allows us to count how many trips it will take. Since the boat only allows max 2 and has min 1 animal we have the heuristic

$h(x) = $ # of animals on starting side -1

since a max net difference of 1 is possible.
This is also the reason why it is admissible as all boat movements except the one creating the goal state will result in a net difference of 1 animal to the target side.

h/ An A* search algorithm would not be a better choice than DFS as it will expand unnecesary nodes even if it had a good heuristic. Since A* partially decides based on cost, it will open up more nodes than DFS which can choose the optimal path on the first try and at worst perform as well as A* based on the given search tree if not better as it will only expand one of the sides of the loop at the end of the tree.

An Nguyen 20126030

2b/ The blocking heuristic returns 0 if the board is a goal state and 1 + number of cars blocking the goal car otherwise.
We want to prove $h(m) - h(n) \leq cost(m, n)$: consistency
Let m and n be neighbors. Thus the $cost(m, n) = 1$; show $h(m) - h(n) \leq 1$
We can break down the cases caused by moving from m to n with respect to changes in the blocking heuristic.
Consider n to be a goal state.
(i) If m is not a goal state, $h(m) = 1$ since m and n are neighbors, m must not have any cars blocking the goal car to get to n. $\Rightarrow h(m) - h(n) = 1 - 0 = 1$
(ii) If m is goal state $\Rightarrow h(m) - h(n) = 0 - 0 = 0$
(iii) Consider n to be same number of cars blocking as m and n is not goal state. In this case the $h(m) = h(n)$ since both states have the same number of blocking cars. $\Rightarrow h(m) - h(n) = 0$
(iv) Consider n to have more blocking cars. Since n can only move 1 car, $h(n) = h(m) + 1$ as they are neighbors. $\Rightarrow h(m) - h(n) = -1$
(v) Consider n to have less blocking cars. Similarly, $h(n) = h(m) - 1$ (note: $h(n) \neq h(m) - 2$ since for n to have less blocking cars, movement of m to n must move a non-goal car, thus n cannot be a goal state.) $\Rightarrow h(m) - h(n) = 1$.
Since we have shown it is consistent for neighbors, we can show it is consistent for all m, n as we can break down path m to n into its constituting neighbors. Each step is incremental by 1 and the heuristic is incremental by at most 1. By applying the above logic for all neighbors between m and n, we can have an upper bound of cost(m, n) since the heuristic increments slower than the cost.

2c/ The heuristic I implemented for the advanced heuristic is an extension of the blocking heuristic where it follows all the rules of the blocking heuristic except if there is a car blocking the goal car, we see if that car can move, if it cannot due to being blocked, we add 1 more as another car must move to make way for the blocking car to move out of the goal car's way.

consistency
This is consistent similar to the blocking heuristic by nature as it is an extension of it. It is the same in all cases of m, n except if m has a blocking car that is also blocked. In this case we can consider the following cases.
Consider n has less blocking cars, to get to this state all the blocking cars and cars blocking them must move. Thus these cars must move and the heuristic will always underestimate the moves required to do so, if not exactly so.
Consider n has the same or more, then the heuristic will estimate a negative difference causing the cost to be higher.

dominating
The advanced heuristic dominates the blocking as it will always be greater or equal to since it is additive to the blocking as it can generally be considered an extention of and will add to the blocking in the case of the blocking cars being blocked or will guess the blocking heuristic otherwise by def^n.

The output files
shows that in almost all cases the advanced takes less time then blocking.
outblocking.txt → blocking h(x) ; outadvanced.txt → advanced h(x)
line # = test case. value is loop# nodes expanded.

outblocking.txt	outadvanced.txt
File Edit Format	File Edit Format
625	566
541	734
455	419
115	46
659	178
583	518
1958	2418
795	271
347	473
1490	1472
668	579
590	489
3668	3880
3715	4634
520	518
1848	1735
1820	1499
1324	1021
462	441
393	529
240	213
2567	2452
1397	1251
4022	3772
6563	6170
3896	3693
2281	2037
1249	833
4230	4115
1048	1049
3636	3542
506	499
2556	2044
4124	4061
3803	3725
2046	1977
1855	1619
3033	3003
3511	3486
2620	2306