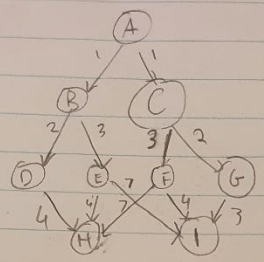
1.

1a



where F is our goal node.

b.

Step	Frontier	Explored Nodes	Path
1	A	/	/
2	B, C	A	A
3	D, E, C	A, B	A, B
4	H, E, C	A, B, D	A, B, D
5	E, C	A, B, D, H	A, B, D, H
6	C, I	A, B, D, E, H	A, B, E
7	C	A, B, D, E, H, I	A, B, E, I
8	F, G	A, B, C, D, H, I	A, C
9	G	A, B, C, D, E, F, H, I	A, C, F

We reach our Goal Node F with a cost of 21

c.

Step	Path	Frontier	Explored Nodes
1	'	A	'
2	A	B, C	A
3	A, B	C, D, E	A, B
4	A, C	D, E, F, G	A, B, C
5	A, B, D	H, E, F, G	A, B, C, D
6	A, B, E	I, F, G, H, I	A, B, C, D, E
7	A, C, F	G, H, I	A, B, C, D, E, F

We reach our Goal F with cost 12

d.

Step	Path	Frontier	Explored Nodes
1	'	A	'
2	A	B, C	A
3	A, C	B, F, G	A, C
4	A, C, G	B, F, I	A, C, G
5	A, C, G, I	B, F	A, C, G, I
6	A, C, F	B, H	A, C, G, F, I

We reach our Goal Node F with cost 10

2.

a. Recalling that the 3 primary dimensions are Uncertainty, Interaction and the number of agents, I will go by through them one by one. For Uncertainty, our aim for this problem is t have no randomness due to the nature of the fox, hen and grain and for this problem, our observation will determine what state we will transition to and from. Given those two, we have a fully observable, deterministic dynamic for this problem. For interaction with the environment, we have an offline interaction as the farmer will need to decide what item to take with him before doing it so that that the chicken doesn't eat the grain or the fox to eat the hen. Finally, we have a single agent within the environment. This is because, the hen and fox both a fixed behaviour (chicken wants to eat grain and fox wants to eat the hen) and they do not have any other behaviours defined. This means the farmer only needs to worry about those behaviours so while there are 3 agents within this environment, the other 2 only have a fixed nature.
b. Our start state is when the hen, fox, grain and farmer are all on the left side.

   Our goal function is when all of them are at the right side.

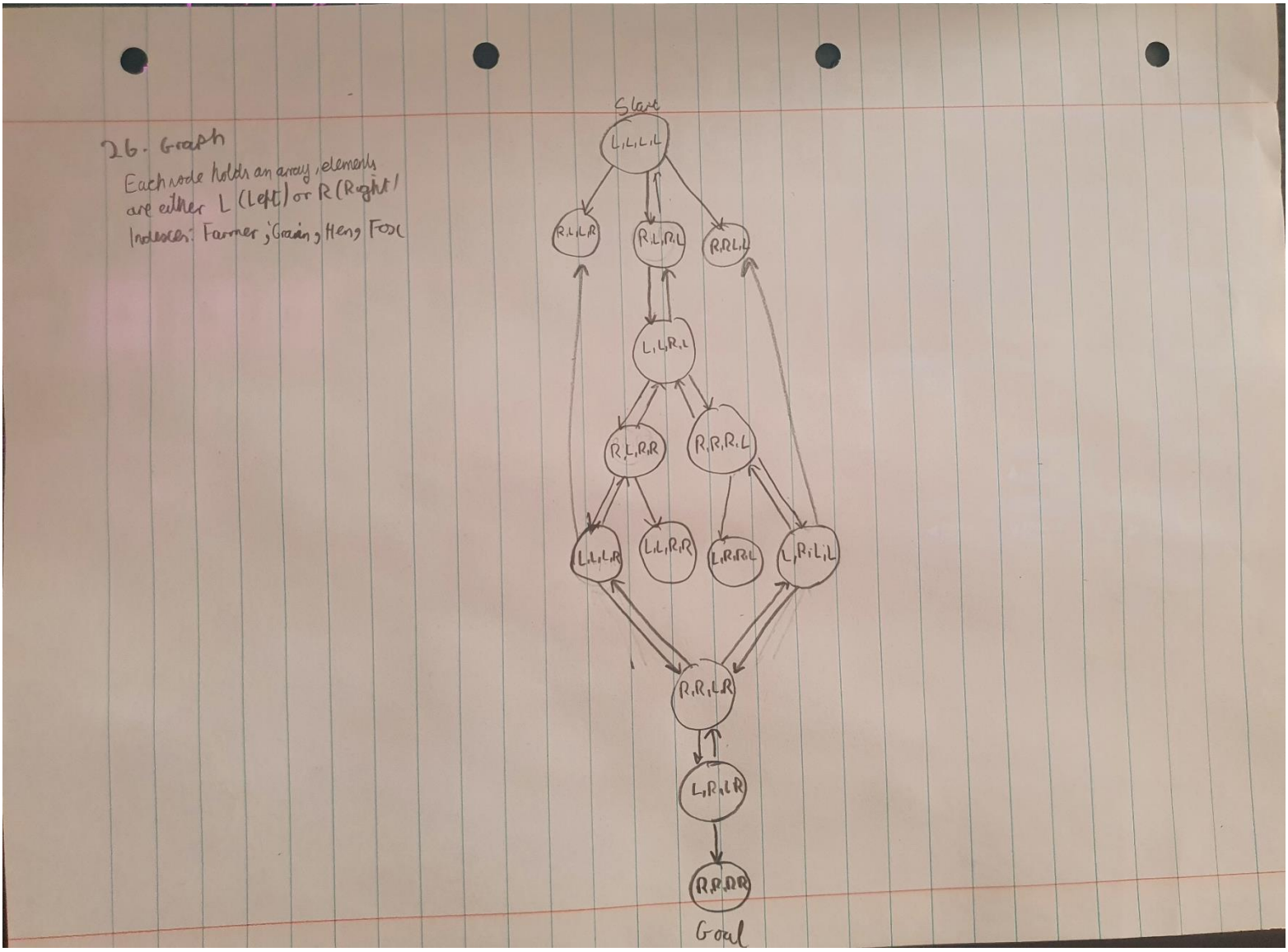
   Our possible actions are to move one of the items from the left side to the right side or from the right side to the left, load an item or unload an item.

   The costs associated with this search problem is when we move the objects from one side to the other side with our goal is to reach the goal state with the lowest possible cost (least trips).

   The successor function in this case is to load an item(when we have an empty raft) or unload an item(We have a full raft).

   Our possible states are when the grain/hen or fox are on one side of the river while the remaining two are on the other side.

   Picture of Graph: The indexes in the arrays store the position of the Farmer, Grain, Hen ,Fox in that order while the elements holds L (Left) or R (Right)



c. The forward branching factor of this graph on average is about 3. This is because this is the largest forward branching factor out of all the nodes

d. A non-constant admissible heuristic for this search problem is we add a value of one for all the items on the wrong side(they are on the left side) and for every invalid combination (hen + fox left) we add 1. That way the heuristic will give us a number that sees what animals are left + invalid combinations.

e. The above Heuristic is admissible because it will firstly underestimate the cost to the cheapest path which means its it will be less than the actual cost of the cheapest path t the goal node, hence it is admissible. We also relax the hen + grain constraint which also allows the heuristic to be admissible

f. Done in attached Python

3.

a.

i. A graph search is better for this as we need to take and remember the specific actions taken to solve the Rubik's cube and a local search does need to

remember the steps we took previously whereas in a graph search it does remember all of the steps we took, hence being more appropriate.

ii. Graph Search would be better(in this case A* is the most appropriate) as we aim to find the most optimal path to reach the end (which A* star fulfils). We also need to remember the steps we have taken previously so we can't disregard those hence a graph search is better.

iii. A local search would be more appropriate here as we do not need to remember all of the steps we have taken to the goal states, as long as we fulfill the constraints, the steps do not need to be written down.

b. It is not a complete algorithm as while we are able to get a local maxima/score, it does not guarantee it is a global maxima/score nor does it give us a guaranteed satisfying assignment.

c. This not optimal. This is because the increments for the neighbourhoods is really small so it is more likely to get stuck at a local maxima and not find the global maxima on this graph as it will think the local maxima is the global maxima. It will think this because of how small the jumps are, it will reach a point where it finds there will be a decrease on the y-axis value and thinks it will be a global maxima when it is not.

d. This is optimal as the interval for jumps on the x axis is decently sized so that it will not get stuck at a local maxima as it will continuously find large values until it reaches the global maxima on the given graph