Tutorial - Week 3

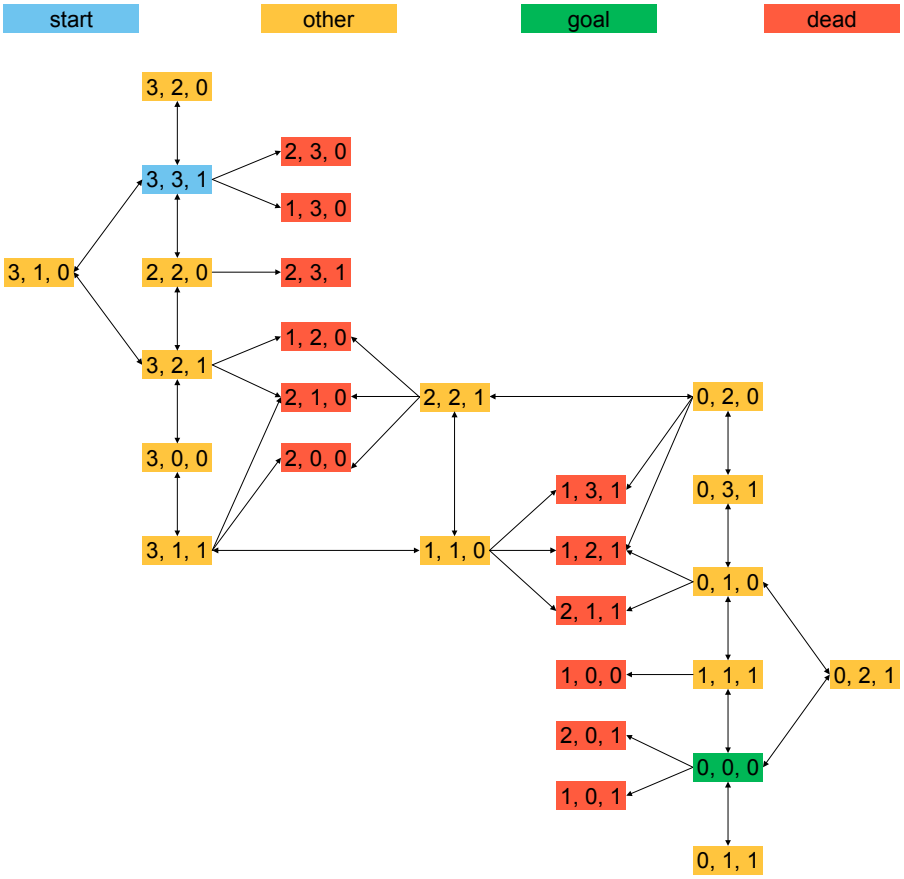
Uninformed Search

Activity 1

Activity 1. The "Missionaries and Cannibals" problem is usually stated as follows: Three missionaries and three cannibals are on one side of the river, along with a boat that can hold one or two people. Find a way to get everyone to the other side, without ever leaving a group of missionaries in one place outnumbered by the cannibals in that place. This problem is famous in AI because it was the subject of the first paper that approached problem formulation from an analytical viewpoint (Amarel, 1968).

1. Formulate the problem precisely, making only those distinctions necessary to ensure a valid solution, and draw a diagram of the complete state space.

Each state can be characterised by listing the number of missionaries (0-3) and cannibals (0-3) on the original side of the river, and 1 or 0 to indicate whether or not the boat is on that side; it is assumed that whatever is not listed must be on the far side of the river. There are 4 x 4 x 2 = 32 states in total, but only 28 of them are accessible from the initial state (3,3,1). Twelve of these represent "dead" states in which one or more missionaries are eaten. Here is a diagram of the complete state space:



2. Solve the problem optimally using an appropriate search algorithm; is it a good idea to check for repeated states?

There are four ways to get from the initial state (3,3,1) to the final state (0,0,0) in 11 steps. It is definitely a good idea to check for repeated states. Using Breadth-First Search, for example, there are 6 nodes at depth 2 and 25 at depth 3; but, if we avoid expanding previously encountered states, there will only be 2 nodes at depth 2 and 3 at depth 3. Depth-First Search is only able to avoid states which are repeated along the same branch; but the number of nodes is still reduced to 3 at depth 2 and 8 at depth 3.

3. Why do you think people have a hard time solving this puzzle, given that the state space is so simple?

The step at which people seem to have most difficulty is the one in the centre of the diagram, from (1,1,0) to (2,2,1). Since the objective is to get all 6 people to the far side of the river, it seems counterintuitive to bring two people back to the original side; it violates the "heuristic" of maximising the total number of people on the far side of the river.

Activity 2

For the route from Arad to Bucharest, what order are nodes in the state space expanded for each of the following algorithms when searching for the shortest path between Arad and Bucharest? Where there is a choice of nodes, take the first one by alphabetical ordering. Make sure you understand the key properties of the different algorithms, as listed below.

State space: set of cities on Romania map
Initial state: Arad
Successor function: s(x) is the set of cities adjacent to x on the map
Goal state: Bucharest
Path cost: Sum of the costs of distances between the cities on the path

(i) Depth-First Search
Arad, Sibiu, Fagaras, Bucharest (note that the solution is found on the first branch only because of the rule for ordering the successors alphabetically; this is not usually the case!)

(ii) Breadth-First Search
Arad, Sibiu, Timisoara, Zerind, Fagaras, Bucharest (assuming that the search stops once the goal state is generated and that when expanding a node, previously expanded nodes are checked to ensure that nodes with states already explored are not added to the frontier, e.g. Arad which is generated via the paths Arad → Sibiu → Arad and Arad → Sibiu → Oradea → Zerind → Arad)

(iii) Uniform-Cost Search
Arad (0), Zerind (75), Timisoara (118), Sibiu (140), Oradea (146), Rimnicu Vilcea (220), Lugoj (229), Fagaras (239), Mehadia (299), Pitesti (317), Craiova (366), Drobeta (374), Bucharest (418) (assuming a check that nodes with states previously generated are not added to the frontier, except when they have lower path cost than a node with that state already on the frontier, in which case the node with higher path cost is removed, so ignore Oradea (291) reached via Sibiu, and Sibiu (297) reached via Zerind and Oradea)

(iv) Iterative Deepening Search
Arad, Arad, Sibiu, Timisoara, Zerind, Arad, Sibiu, Fagaras, Oradea, Rimnicu Vilcea, Timisoara, Lugoj, Zerind, Oradea, Arad, Sibiu, Fagaras, Bucharest (assuming a cycle check on each path, so omit Arad reached via Arad → Sibiu → Arad)