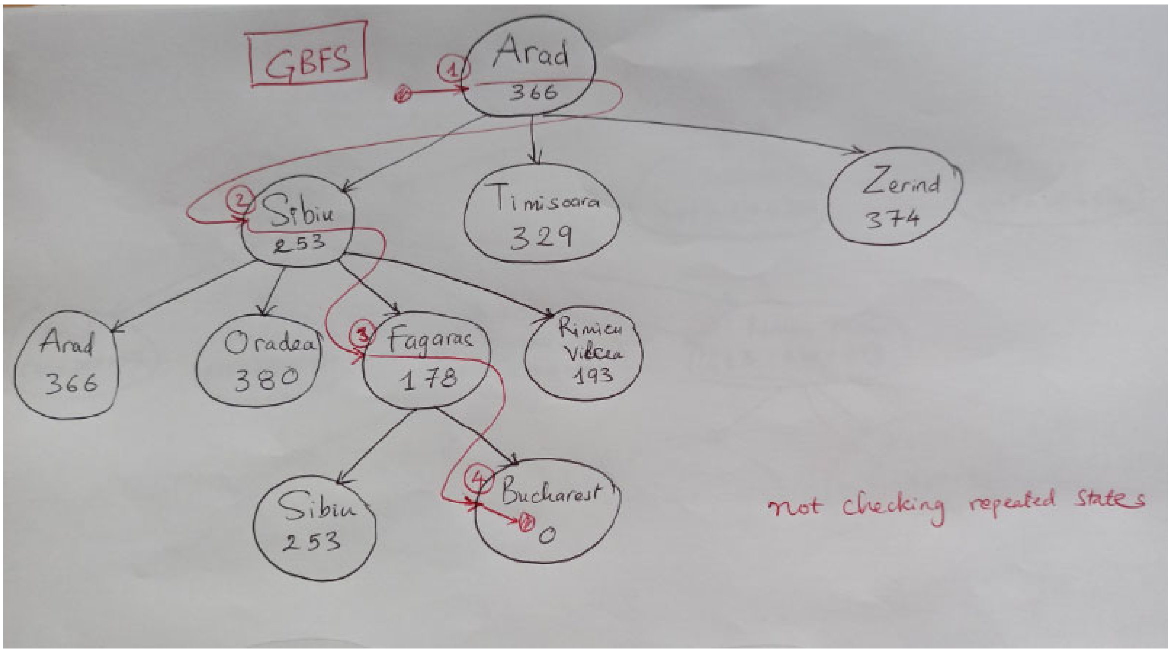
COS30019 - Introduction to Artificial Intelligence
Tutorial Problems Week 4

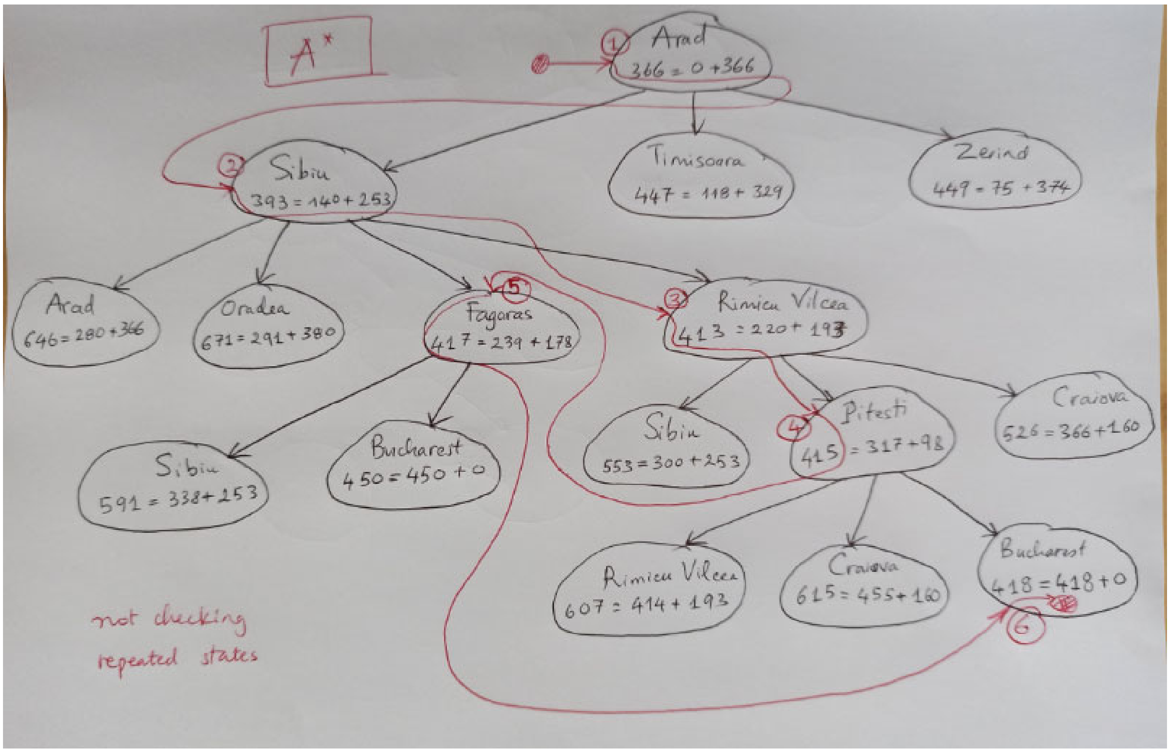
**Task 1:** Consider the problem of getting from Arad to Bucharest in Romania and assume the straight line distance (SLD) heuristic will be used.

1. **Give the part of the search space that is realized in memory and the order of node expansion for:**
   a. **Greedy search & A* assuming that a list of states already visited is maintained.**

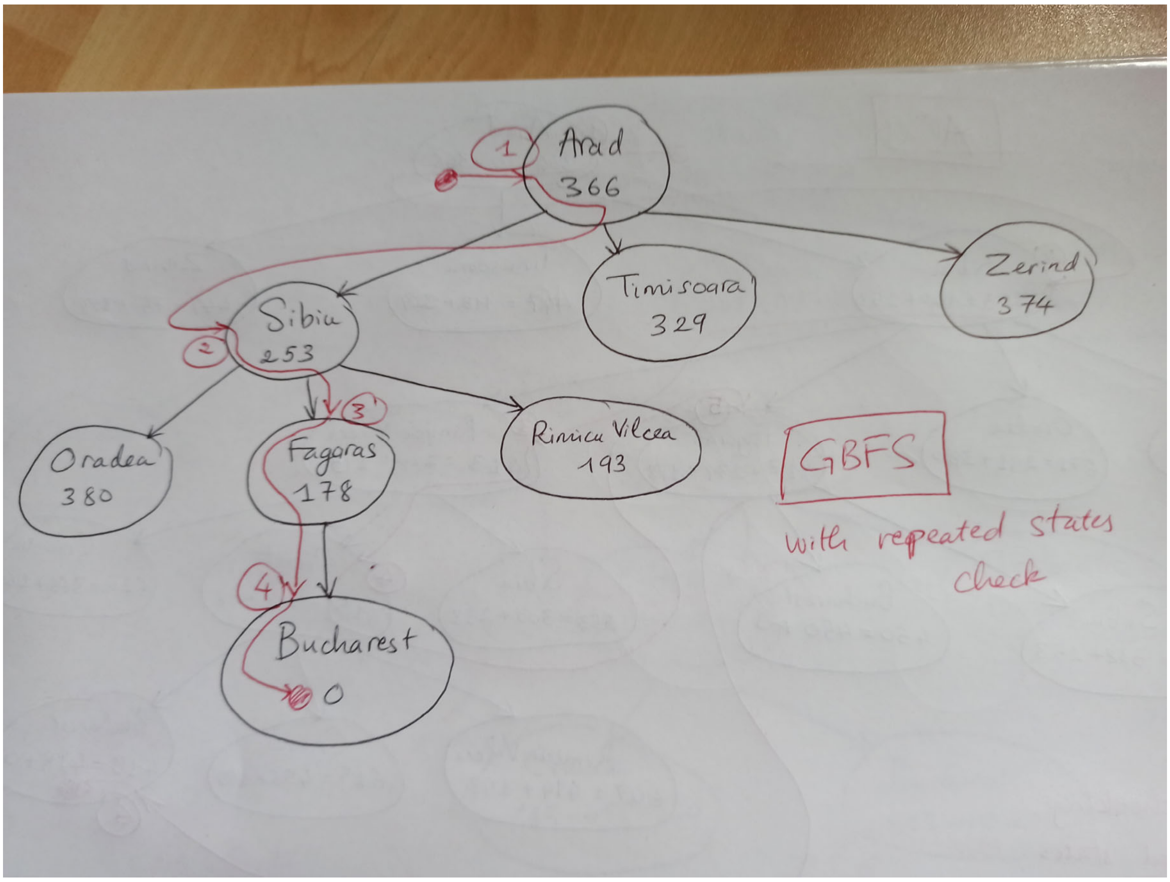
Greedy search evaluation function:
**f(n) = h(n)**, where **h(n)** is the estimated cost of the cheapest path from node n to goal state (Straight Line Distance to the goal, in this example)

A* search evaluation function:
**f(n) = g(n) + h(n)**, where
**g(n)** – the cost so far to reach the goal (how many km have we made so far, in this example)
**h(n)** – estimated cost from the node to the goal (Straight Line Distance to the goal)



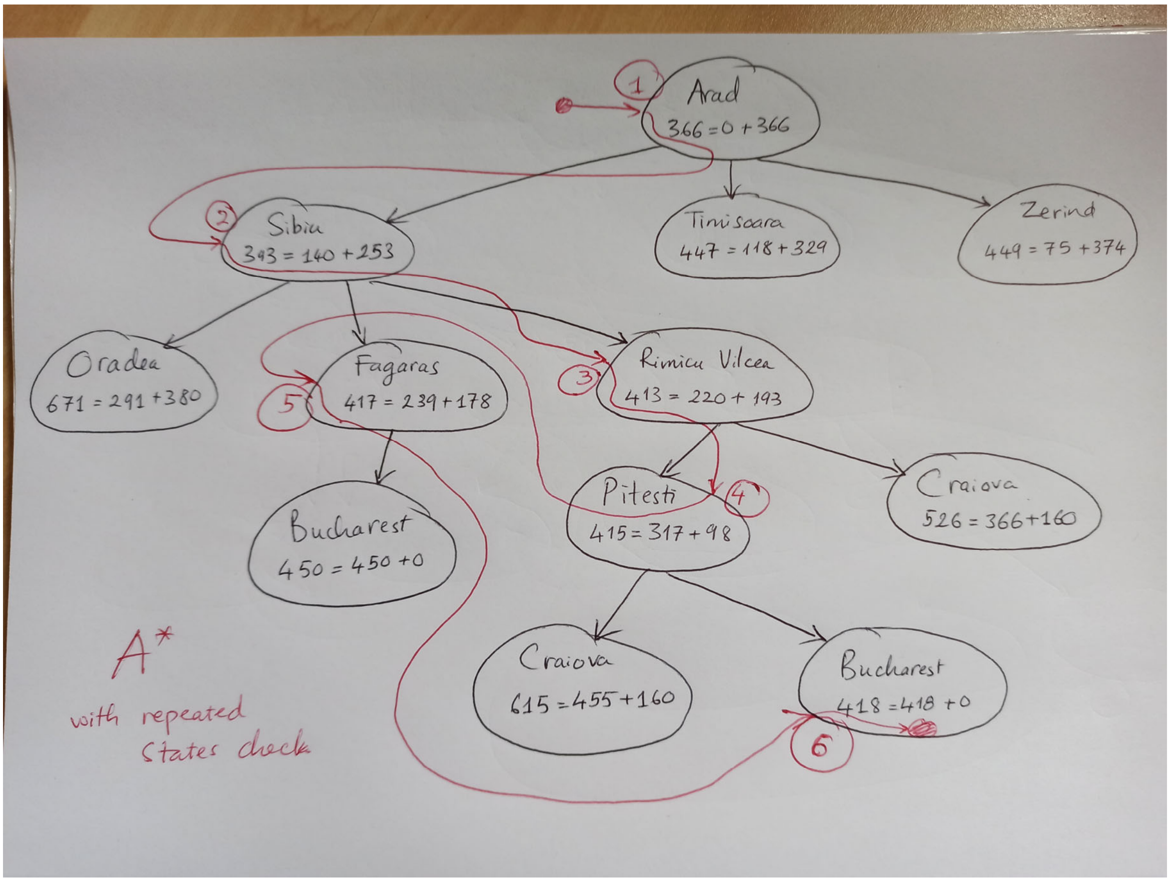


b. **GBFS & A* search assuming that a list of states already visited is maintained.**



GBFS

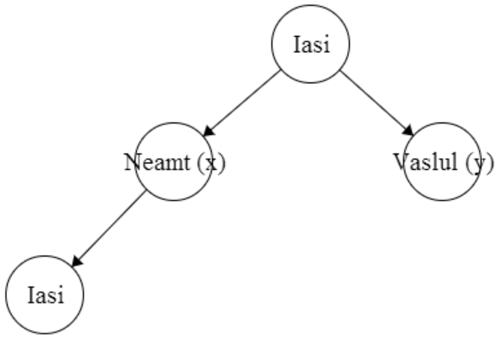
with repeated states check



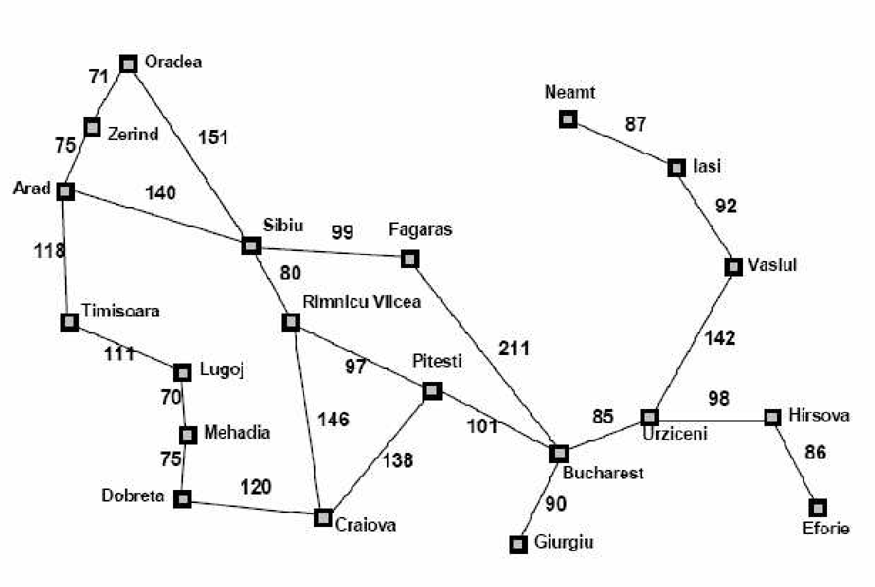
2. How would the above searches differ if the list of states already visited is NOT maintained?
   In the example, where we have to travel from Arad to Bucharest, if there is no check for repeated states, then we will get additional search nodes in memory.

Please pay special attention to the case with A* when repeated states are checked: The city of Craiova (the same state) was created in 2 different search nodes in memory. WHY? This is the key difference between TREE-SEARCH and GRAPH-SEARCH. We are only required to implement TREE-SEARCH but it would be important to understand the difference between these two. Of course, those of you who are confident of correctly implementing GRAPH-SEARC in your Assignment 1, please feel free to do so.

3. How do the above searches perform for planing a trip from Iasi to Fagaras?
   a) Greedy search would result in infinite loop between Neamt and Iasi, if there is no check for repeated states.



b) A* search would suggest to go to Vaslui even when repeated state checks are not performed because going to Neamt and back would cost more than not moving to Vaslui. This is an exercise for **you** (students) to create search trees similar to the ones above.



Straight−line distance
to Bucharest

| | |
|---|---|
| Arad | 366 |
| Bucharest | 0 |
| Craiova | 160 |
| Dobreta | 242 |
| Eforie | 161 |
| Fagaras | 178 |
| Giurgiu | 77 |
| Hirsova | 151 |
| Iasi | 226 |
| Lugoj | 244 |
| Mehadia | 241 |
| Neamt | 234 |
| Oradea | 380 |
| Pitesti | 98 |
| Rimnicu Vilcea | 193 |
| Sibiu | 253 |
| Timisoara | 329 |
| Urziceni | 80 |
| Vaslui | 199 |
| Zerind | 374 |

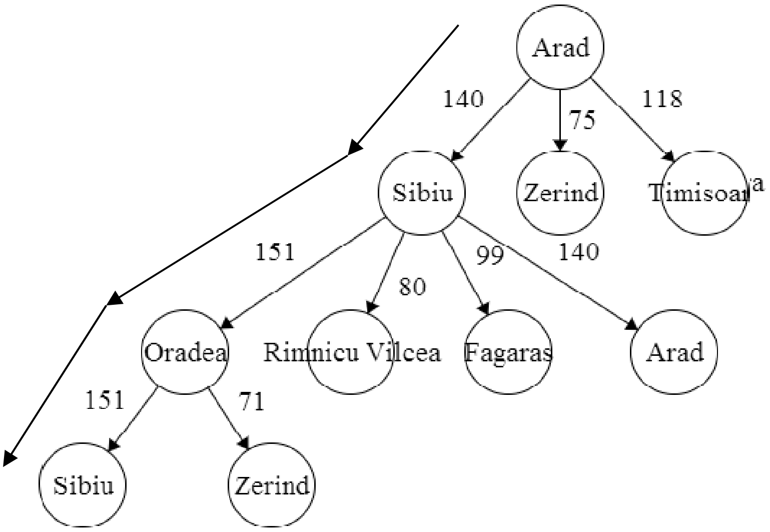
**Task 2:**

1. **Suppose we run a greedy search algorithm with *h(n)* = *-g(n)*. What sort of search will result?**

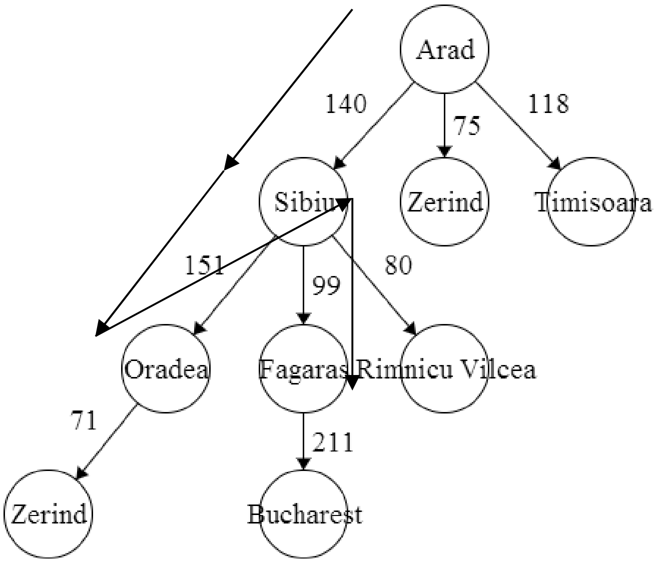
   We always go down one branch (which one? **Hint:** *As we try to minimize h(n), we equivalently maximize g(n)*) and then we subsequently expand the deepest unexpanded node along that branch. That is **DFS**.

   If there is no check for repeated states, such greedy algorithm can result in infinite loop, similar to DFS.

   a) Infinite loop, when no check for repeated states



   b) Search with backtacking similar to DFS (check for repeated states)

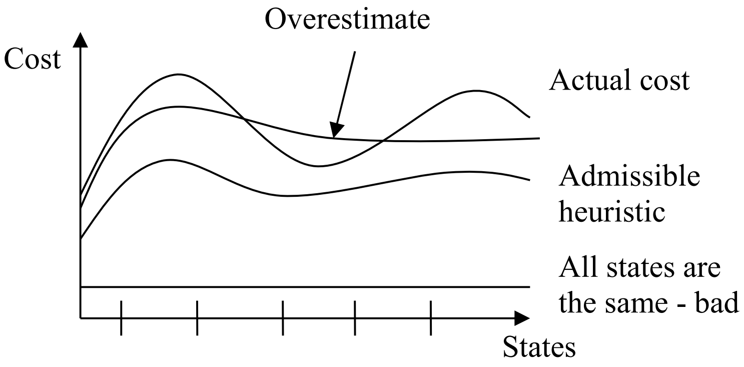


2. **Suppose we run an A* algorithm with *h(n)* = 0. What sort of search will result?**

It will result in **uniform cost** search, because all the nodes will have the same estimated cost = 0. **f(n)=g(n)+0**, means that we consider only the cost so far and the estimated cost is the same, which is a uniform cost search.

*h(n)* = 0 is not a good heuristic because it does not really help the algorithm to choose the nodes.

Trend is important for search, heuristic estimates cost (try to be as close as possible to the actual cost).



3. **Explain why the set of states examined by A* is <u>often</u> a subset of those examined by breadth-first search.**

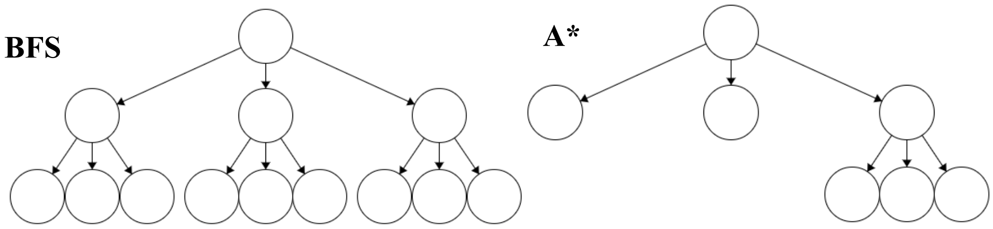
BFS is a blind search:

- No notion of the "right direction"
- Can only recognise goal once it is achieved

Heuristic search (including A*):

- We have rough idea of how good various states
- Use this knowledge to guide our search

When the step cost for all actions is the same, both BFS and A* will return optimal solutions. However, A* search performs guided search, and it does not expand all the nodes at each level (unlike BFS).



Note: It is often a subset but not always because, if step costs are NOT the same then BFS will find the solution that reaches a goal via the smallest NUMBER of STEPS even if it is not optimal while A* will still be able to find the optimal solution. For example, for the route-finding problem in Romania, BFS will return the solution Arad, Sibiu, Fagaras, Bucharest; while A* will return the optimal solution Arad, Sibiu, Rimicu Vilcea, Pitesti, Bucharest.

**Task 3: The missionaries and cannibals problem is usually stated as follows. Three missionaries and three cannibals are on one side of a 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.**

1. **Implement and solve the problem optimally using an appropriate search algorithm. Is it a good idea to check for repeated states?**

😃😃😃 - 3 missionaries (M)

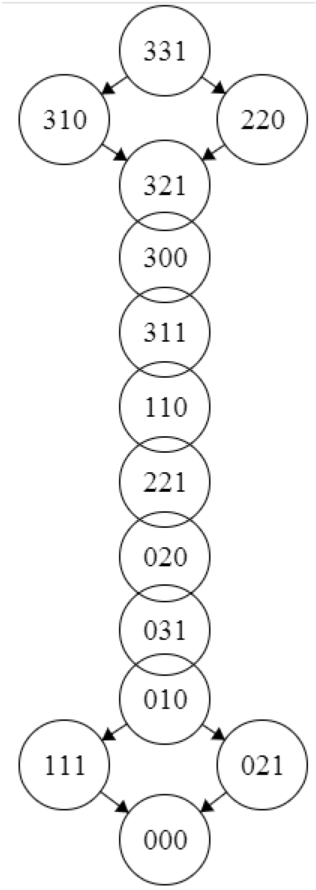
☹☹☹ - 3 cannibals (C)

⛵ - boat

We need to track:

- the number of missionaries and cannibals at each side of the river
- at which side of the river the boat is

If we know the number of M and C at one side, we can extract and get the number of M and C at the other side. Thus we can represent the state as vector S(x,y,z), where
S[1] – number of M at the right side (integer from 0 to 3)
S[2] – number of C at the right side (integers from 0 to 3)
S[3] – Boolean, 1 if the boat is at the right side (0 or 1)



Valid state:

- number of M must be greater than the number of C at each side of the river
- S[1] ≥ S[2] or S[1] = 0

Initial State: S = [3,3,1]
Goal State: S = [0,0,0]

Possible actions:

- When the boat is at the right side S[3] = 1, substract of the current state the following vectors:
  - [1,0,1]
  - [0,1,1]
  - [2,0,1]
  - [0,2,1]
- When the boat is at the left side S[3] = 0, add to the current state each of the vectors
  - [1,0,1]
  - [0,1,1]
  - [2,0,1]
  - [0,2,1]

The problem state space (any other state is invalid) (at the right)

Blind search:

- DFS: not optimal, complete (if check for repeated states)
- BFS: optimal, complete

Heuristic search:

- Greedy: not optimal, complete (if check for repeated states)
- A*: optimal (if heuristic is admissible), complete

As there is only one solution to the problem, any complete mechanism (even not optimal) would find this only solution, hence any of the mechanisms above can be applied.

2. **Is there a heuristic that would be useful for the missionaries and cannibals problem? The generalized missionaries and cannibals problem (*n* missionaries and *n* cannibals)?**

We need to define admissible heuristic. Our initial state is [3,3,1]. Our goal state is [0,0,0]. Thus, the less people are at the right side of the river, the better it is in terms of cost of transportation.
The heuristic:
**h(n)=( S[1] + S[2] ) – 1**

**The minus 1 (– 1) in the above expression is important to keep the heuristic function admissible:**
Example of extreme case with 2 people: 1 M and 1 C. It would require 1 move to reach the goal state, which is estimated by our heuristic function: h(n) = (1 + 1) – 1 = 1 (cost of 1 transportation). Note that **h(n)=( S[1] + S[2] )** heuristic function will not be admissible.