Solutions to FIT5047 Tutorial on Problem Solving as Search: Backtrack, Tree Search and Graphsearch

Exercise 1: Backtracking

Specify a state representation, operators and goal test to solve the missionaries and cannibals problem:

Three missionaries and three cannibals come to a river. There is a boat on their side of the river that can be used either on one or two persons. How should they use this boat to cross the river in such a way that cannibals never outnumber missionaries on either side of the river?

Illustrate the workings of the backtracking algorithm (Backtrack1) to solve this problem. To expedite your solution, you may want to specify a heuristic function to order the actions for different states.

SOLUTION:
Before we solve this problem, I would like to point out the following.

The main idea of a backtracking algorithm is that it selects **one successor** for a parent node, and only considers that particular successor, then, in turn, it selects a successor for it, etc. When a successor can not be selected, then (and only then), another alternative is attempted. At any point in time, the algorithm is aware of only one path from a particular node (which is the node under consideration) to the start node.

Now that this is out of the way, let us consider the solution:

STATE REPRESENTATION

- (# missionaries at start,# cannibals at start) and total number of missionaries and cannibals.
- B – has value 0 if boat is at the start, and value 1 if boat is at the finish.
- i – number of times the boat has crossed the river.

Note that a database containing only the number of cannibals and missionaries on one side of the river is not enough, since the algorithm would not know what are the conditions and the other side, e.g., whether a constraint has been violated, whether the termination condition has been fulfilled, etc. That's why we also need the total.

OPERATORS
R_{ij} – put in the boat i missionaries and j cannibals, where $0 < i + j \leq 2$.

Rules of life which govern this process:

- The boat cannot travel when it is unmanned.
- The boat will travel when there is at least one person in it (it will, however, wait for an additional passenger, if the intent is to send two people).
- Person(s) in the boat automatically disembark at the opposite side of the river.

Constraints:
The cannibals can never outnumber the missionaries on either side of the river. For c = # of cannibals on the left bank, and m = # of missionaries on the left bank, the constraints are: $[(c \leq m) \wedge (3 - c \leq 3 - m)] \vee (m = 3) \vee m = 0$. We can collapse these inequalities to: $(c = m) \vee (m = 0) \vee (m = 3)$.

GOAL TEST:
#-of-cannibals(start) + #-of-missionaries(start) = 0

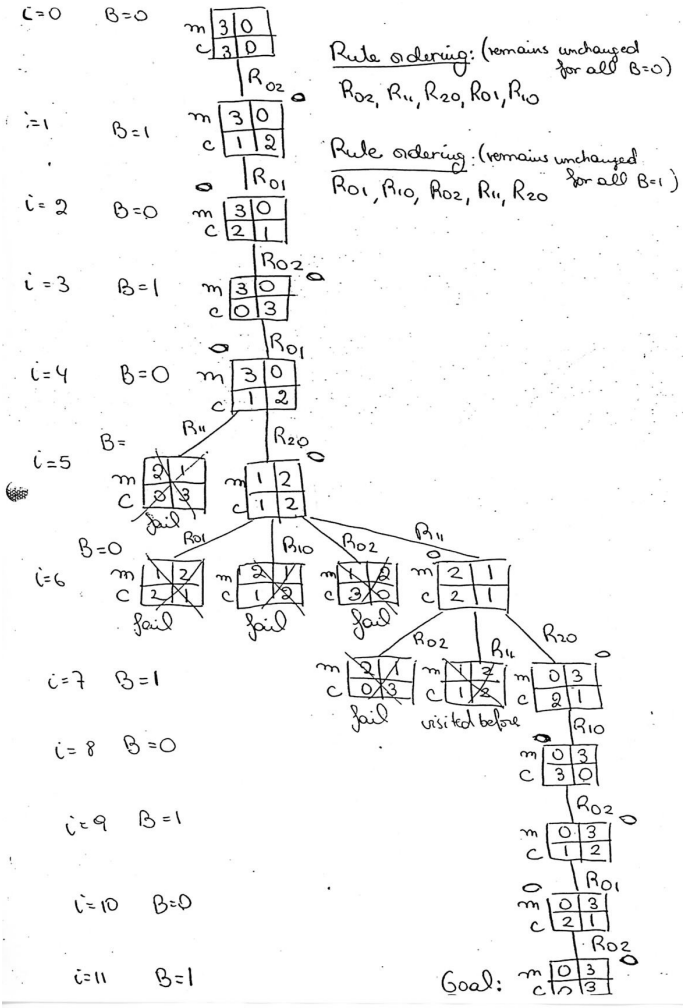
Note that the termination condition can **not** take into consideration just the cannibals or just the missionaries, because then the algorithm would stop with 3 cannibals or 3 missionaries on the finishing river bank, and some people could be left on the starting bank.

HEURISTIC FUNCTION:
#-of-cannibals(finish) + #-of-missionaries(finish)

This number ought to be maximized.

Tie-breaking rule (arbitrary): Take the cannibals first.

BACKTRACKING SOLUTION:
When performing backtracking, keep in mind that the function APPRULES orders the applicable rules (either arbitrarily or according to heuristic merit). This is done **once** for each recursive call to Backtrack1. After the ordering has been established, the first rule is attempted, if it fails, then the next one is tried, and so on.



Exercise 2: Backtracking, Constraint Satisfaction

Consider a problem where one has to insert the numbers 1, 2, 3, 4, 5 and 6 in the perimeter of the following rectangle, so that each side adds up to 18.

	8		
9	X	X	10
		7	

(a) Express the state space of this problem as a set of constraints on the variables assigned to the different positions in the rectangle, where the following variable assignment is to be used:

x_1	8	x_2	x_3
9	X	X	10
x_4	x_5	7	x_6

The constraints in question may be sums of variables, inequalities and actual variable assignments. The Goal is an assignment of a number to each variable.

(b) Apply algorithm Backtrack1 to find an assignment of numbers to variables which satisfies the problem statement. You can make only one variable assignment in each step (what will happen if you make more assignments?), but any variable assignment which follows directly from this variable assignment should be performed in the same step.

Number each state created. Indicate clearly which path is active at each point in time, and also indicate deadend states and the number of the state returned to, if a deadend state is encountered.

(c) Draw the rectangle with the variable assignment you have found.

SOLUTION:

(a) The set of equations is as follows:

From the rectangle:
(1) $x_1 + x_2 + x_3 = 10$
(2) $x_1 + x_4 = 9$
(3) $x_3 + x_6 = 8$
(4) $x_4 + x_5 + x_6 = 11$
From the problem statement:
(5) $x_i \neq x_j$ for $i \neq j$ where $i, j = 1, ..., 6$
(6) $1 \leq x_1, \ldots, x_6 \leq 6$
Derived:
(7) $3 \leq x_1, x_4 \leq 6$ from (2), (6) and problem statement
(8) $2 \leq x_3, x_6 \leq 6$ from (3), (6) and problem statement

(b) Backtrack1 is applied by performing a variable assignment, and for this assignment performing an additional assignment, etc, until either a consistent assignment for all variables is obtained or we have to backtrack and undo an assignment. Due to the nature of the problem, i.e., the existence of derived variables, we don't have to go down to depth 6, but we can derive some dependent variables in each step. Note that additional relationships between variables may be derived, thus

reducing the computations, e.g., $x_3, x_6 \neq 4$. However, facts such as this have not been taken advantage of, in order to demonstrate the backtracking process more clearly.

Assignment 1: $x_1 = 3 \Rightarrow x_4 = 6$ (from (2))
Recompute equations:
(1') $x_2 + x_3 = 7$
(4') $x_5 + x_6 = 5$
Assignment 1.1: $x_3 = 2 \Rightarrow x_6 = 6$ (from (3)) – DEADEND (from (5))
Assignment 1.2: $x_3 = 3$ – DEADEND (from (5))
Assignment 1.3: $x_3 = 4 \Rightarrow x_6 = 4$ (from (3)) – DEADEND (from (5))
Assignment 1.4: $x_3 = 5 \Rightarrow x_6 = 3$ (from (3)) – DEADEND (from (5))
Assignment 1.5: $x_3 = 6$ – DEADEND (from (5))

Backtrack – the first assignment has to be undone.
Assignment 2: $x_1 = 4 \Rightarrow x_4 = 5$ (from (2))
Recompute equations:
(1') $x_2 + x_3 = 6$
(4') $x_5 + x_6 = 6$
Assignment 2.1: $x_3 = 2 \Rightarrow x_6 = 6$ (from (3)) $\Rightarrow x_5 = 0$ (from (4')) – DEADEND (from (6))
Assignment 2.2: $x_3 = 3 \Rightarrow x_6 = 5$ (from (3)) – DEADEND (from (5))
Assignment 2.3: $x_3 = 4$ – DEADEND (from (5))
Assignment 2.4: $x_3 = 5$ – DEADEND (from (5))
Assignment 2.5: $x_3 = 6 \Rightarrow x_6 = 2$ (from (3)) $\Rightarrow x_5 = 4$ – DEADEND (from (5))

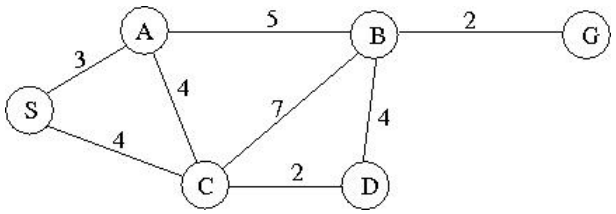
Backtrack – the second assignment has to be undone.
Assignment 3: $x_1 = 5 \Rightarrow x_4 = 4$ (from (2))
Recompute equations:
(1') $x_2 + x_3 = 5$
(4') $x_5 + x_6 = 7$
Assignment 3.1: $x_3 = 2 \Rightarrow x_6 = 6$ (from (3)) $\Rightarrow x_5 = 1$ (from (4')) $\Rightarrow x_2 = 3$ (from (1')) – SOLVED!!

(c) The rectangle with the final variable assignment is:

5	8	3	2
9	X	X	10
4	1	7	6

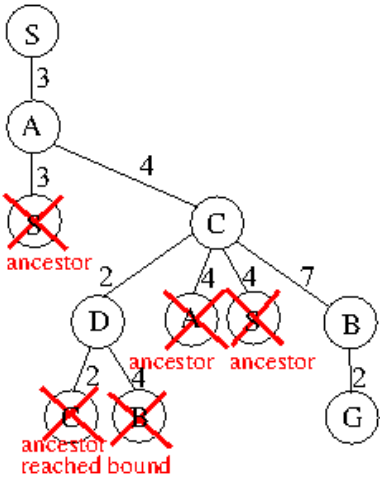
Exercise 3: Search Algorithms

The diagram below depicts the cost of travelling between cities.



(a) Draw the search trajectory generated by the Backtrack1 procedure to reach the goal G starting from S, assuming that the BOUND for discontinuing the search is 4 steps. Use the heuristic *cheapest road* to order the rules. When backtracking, state clearly the reason for backtracking. What is the generated path? What is its cost?

SOLUTION:

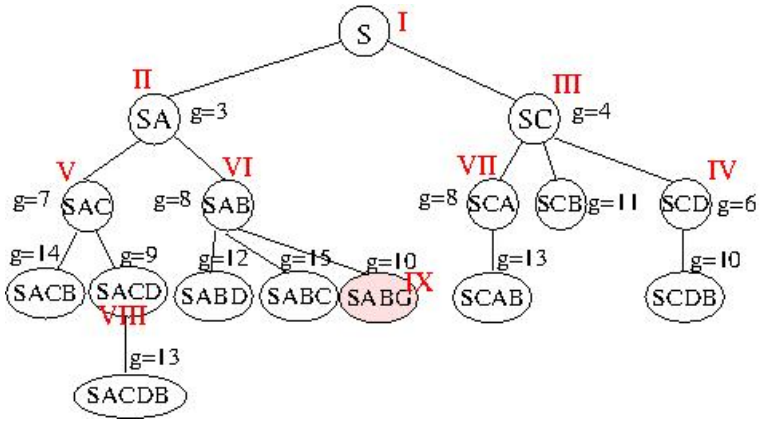


Total path length is 3+4+7+2=16.
Path: S-A-C-B-G.

(b) Draw the search *tree* generated by the basic Tree Search algorithm (page 29 in the lecture notes) to reach the goal G starting from S. Use the cost incurred so far to choose a node from the frontier. For instance, after expanding S, your frontier has nodes A (cost 3) and C (cost 4), so you should choose node A. **Indicate clearly the frontier after each execution of Step 4 of the algorithm.** What is the generated path? What is its cost?

SOLUTION:

To use Tree Search, we must use a unique representation for each node. In this example, we use the path to the node to make it unique.



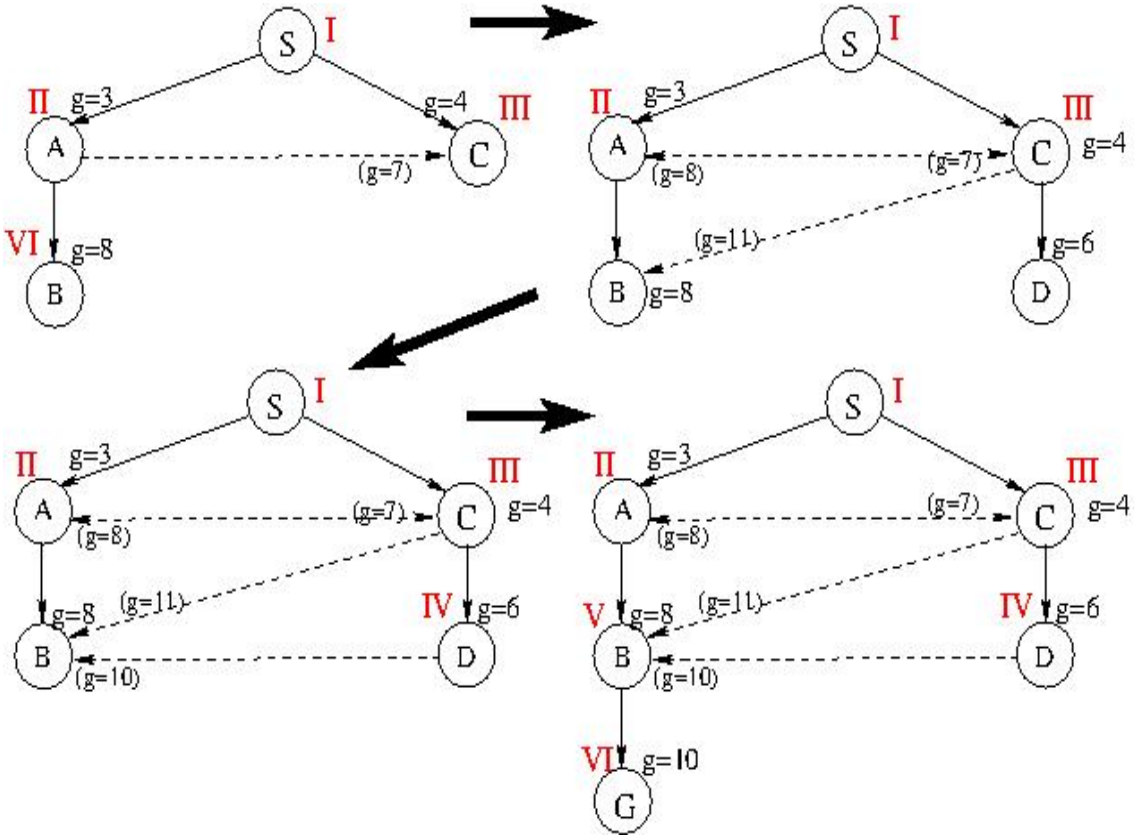
Iteration	Choose	Frontier
		{S}
I	S	{SA (3),SC (4)}
II	SA	{SC (4), SAC (7), SAB (8)}
III	SC	{SCD (6), SAC (7), SAB (8), SCA (8), SCB (11)}
IV	SCD	{SAC (7), SAB (8), SCA (8), SCDB (10), SCB (11)}
V	SAC	{SAB (8), SCA (8), SACD (9), SCDB (10), SCB (11), SACB (14)}
VI	SAB	{SCA (8), SACD (9), SABG (10), SCDB (10), SCB (11), SABD (12), SACB (14), SABC (15)}
VII	SCA	{SACD (9), SABG (10), SCDB (10), SCB (11), SABD (12), SCAB (13), SACB (14), SABC (15)}
VIII	SACD	{SABG (10), SCDB (10), SCB (11), SABD (12), SACBD (13), SCAB (13), SACB (14), SABC (15)}
IX	SABG	{SCDB (10), SCB (11), SABD (12), SACBD (13), SCAB (13), SACB (14), SABC (15)}

Total path length is 10.
Path: S-A-B-G.

IMPORTANT: Unlike BFS, we need to expand all the nodes whose cost is less than the cost of the goal node.

(c) Draw the search *graph* generated by the Graph Search algorithm (page 35 in the lecture notes) to reach the goal G starting from S. As above, use the cost incurred so far to choose a node from the frontier. **Indicate clearly the frontier and explored set after each execution of Step 5 of the algorithm.** What is the generated path? What is its cost?

SOLUTION:
Note that after merging the children of a newly expanded node with the nodes in the explored set, the cost of reaching a node in the explored set may be updated.



Iteration	Choose	Frontier	Explored
		{S}	
I	S	{A (3), C (4) }	{S (0) }
II	A	{C (min{**4**,7}), B (8) }	{S (0), A (3) }
III	C	{D (6), B (min{**8**,11}) }	{S (0), A (min{**3**,8}), C (min{**4**,7}) }
IV	D	{B (min{**8**,11,10}) }	{S (0), A (min{**3**,8}), C (min{**4**,7}), D (6) }
V	B	{G (10) }	{S (0), A (min{**3**,8}), C (min{**4**,7}), D (6), B (min{**8**,11,10}) }
VI	G		{S (0), A (min{**3**,8}), C (min{**4**,7}), D (6), B (min{**8**,11,10}) }