# Comp3620/Comp6320 Artificial Intelligence
# Tutorial 1: Search Formulations, Strategies, and Algorithms

March 13-16, 2018

## Exercise 1 (problem formulation)

For each of the following problem, explain how states and actions can be represented, and give the initial state, goal test, successor function, and a plausible step cost function. Remember from the lectures that these elements constitutes a search problem formulation.

1. Color a planar map using a minimum of colors in such a way that no two adjacent regions have the same color.
2. Using a water tap and three jugs of possibly different capacities (e.g. 12, 5, and 8 litres), collect a given amount of water (e.g. 9 litres) in one of the jugs as fast as possible. The available operations are: filling up a jug completely with the water tap, pouring the contents of one jug into another until one of them is full or empty, or completely emptying the contents of one jug onto the ground. Assume that moving 1 liter takes 1 unit of time.
3. Three superheroes and three supervillains are on the side of a river, along with a boat which can hold one or two people. Find a way of getting them all to the other side, without ever leaving superheroes outnumbered by supervillains at any place. Naturally, the boat cannot move across the river by itself, and none of our super-humans can fly!

## Exercise 2 (properties of search strategies)

True or False? shallowest. (with least step)

True False: Depth-first graph search is guaranteed to return a shortest solution.
True False: Breadth-first graph search is guaranteed to return a shortest solution.
True False: Uniform-cost graph search is guaranteed to return an optimal solution.
True False: Greedy graph search is guaranteed to return an optimal solution.
True False: Breadth-first graph search is a special case of uniform-cost search.
True False: A* graph search with an admissible heuristic is guaranteed to return an optimal solution.
True False: A* graph search is guaranteed to expand no more nodes than depth-first graph search.
True False: A* graph search with a consistent heuristic is guaranteed to expand no more nodes than uniform-cost graph search.

## Exercise 3 (search strategies at work)

Consider the search space below, where $S$ is the initial state and $G1$ and $G2$ both satisfy the goal test. Arcs are labelled with the cost of traversing them and the estimated cost to a goal is reported inside nodes.

For each of the following search strategies: breadth-first, depth-first, iterative deepening, uniform cost, greedy search, and A*, indicate the path found (if any) by graph-search and list, in order, all the states

Adjacent matrix
location

| | |
|---|---|
| States | list of colours |
| Actions | colour(i,c) |
| Goal tests | colour[i] not None for all i |
| initial state | [None, ... ] |

successors (S)

$\{S', \text{colour}(i,c) \} \mid S[i] = \text{None},\ S'[i] = c$

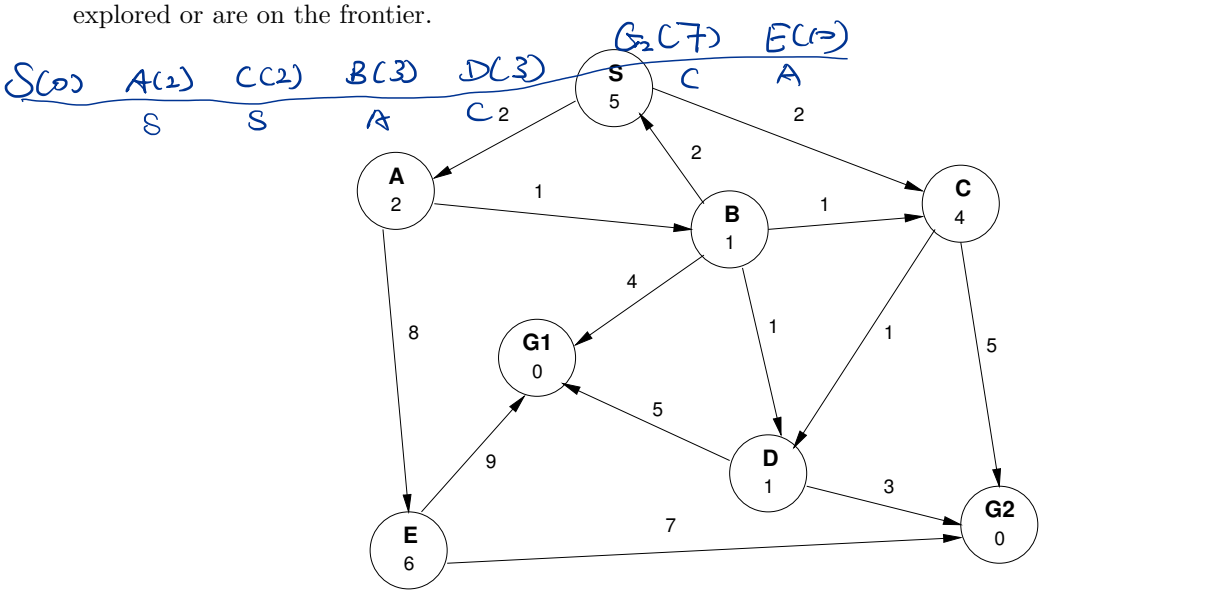
$c \neq \text{None},\ \forall j\ j \neq i\ \ A[i,j] \Rightarrow S[j] \neq c$

$\forall j, i \neq j,\ S'[j] = S[j]$

Step cost (S, colour(i,c) S')

= 1 iff $S[j] = c\ \forall j$

expanded — recall that a state is expanded when it is removed from the frontier. Everything else being equal, nodes should be removed from the frontier in alphabetical order. Assume that regardless of the strategy, the goal-test is performed when a node is dequeued from the fontier. For breadth-first and depth-first search, assume that newly generated nodes are not added to the frontier if they have already been explored or are on the frontier.




## Exercise 4 (Graph Search algorithm)

List all the issues in the following implementation of Graph Search, and explain their impact on completeness and optimality of the various strategies. Assume the EXPAND function is correct and as given in the lectures.

```
function GRAPH-SEARCH( problem, frontier) returns a solution, or failure
    explored ← an empty set of nodes
    frontier ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), frontier)
    loop do
        if frontier is empty then return failure
        node ← REMOVE-FRONT(frontier)
        if GOAL-TEST(problem, STATE[node]) then return node
        frontier ← INSERTNODES(EXPAND(node, problem), frontier)

function INSERTNODES( nodes, frontier) returns updated frontier
    for each n in nodes do
        add n to frontier
      if ∃ m in explored ∪ frontier s.t. STATE[m] = STATE[n] then
        PATH-COST[m] ← PATH-COST[n]
        PARENT[m] ← PARENT[n]
        ACTION[m] ← ACTION[n]
        DEPTH[m] ← DEPTH[n]
    return frontier
```

## Recommended exercises from the book:

3.3, 3.6/b-c, 3.7, 3.9, 3.10, 3.11, 3.13, 3.14, 3.23