TP S (Search)
Techniques of AI [INFO-H-410]

v1.0.0

Source files, code templates and corrections related to practical sessions can be found on the UV or on github (`https://github.com/iridia-ulb/INFOH410`).

Fundamentals for search

Question 1. Suppose you have a data structure D with the push and pop operation. Suppose you have the following sequence of push operations: push(1), push(3), push(2).

a) Which elements are returned by two consecutive pop operations if D is a stack, if D is a queue?

b) Now suppose that D is a priority queue and that the priority of each element is the value of the element itself. Which elements are returned by two consecutive pop operations now?

Use python to validate your answers.

General search algorithms

Question 2. The outline of a general tree search algorithm is sketched below. How can this outline be turned into:

a) depth first search

b) breadth first search

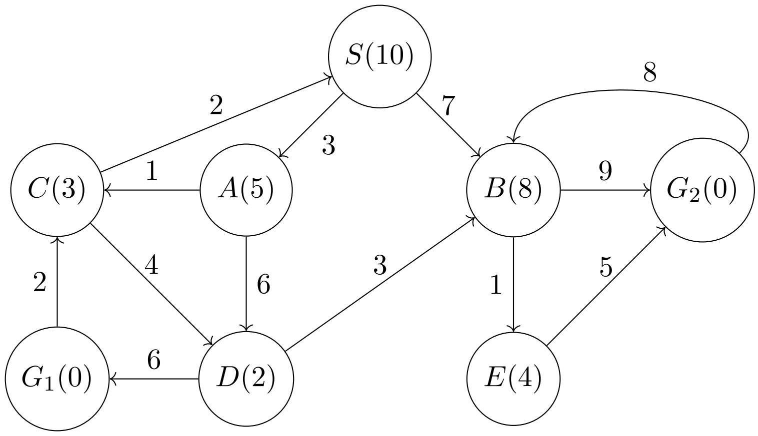
c) best first search

```
q = [start]
while len(q) != 0:
    current = q.pop()
    if current == goal:
        print("Found goal")
        break
    for n in neighbors(current):
        q.append(n)
```

For each of those search algorithms, start from the general algorithm above and add more details to it, such that it becomes the desired search algorithm.

Search spaces

Question 3. Consider the following state space:



- S is the initial state
- G1 and G2 are goal states
- Arcs show actions between states (e.g., the successor function for state S returns A, B).
- Arcs are labelled with actual cost of the action (e.g. the action from S to A has a cost of 3).
- The numeric value in parentheses in each state is the states h-value (e.g. $h(A) = 5$).

You should assume the following on the operational details of the algorithms:

- The algorithm does not check if a state is revisited, so there may be several nodes with the same state in the search tree.
- The algorithm terminates only when it selects a goal node for expansion, not when it is generated by the successor function.
- The successor function always orders states alphabetically.

a) Give the sequence of states that will be visited, together with the total cost of reaching the goal state, for (1) depth first search, (2) breadth first search and (3) A*.

b) In python what data structure could be used to store this graph ? Implement one of the 3 algorithms in python and verify your previous answers (you can use the code template (see page 1)).

The Missionaries and Cannibals Puzzle

Question 4. There are three missionaries and three cannibals on the west bank of a river. There is a boat on the west bank that can hold no more than two people. The missionaries wish to cross to the east bank. But they have a problem: If on either bank the cannibals ever outnumber the missionaries, the outnumbered missionaries will be eaten. Is there a way for all the missionaries and cannibals to get to the east bank without losing anyone?

a) Represent this puzzle as a search problem by defining the states and actions.

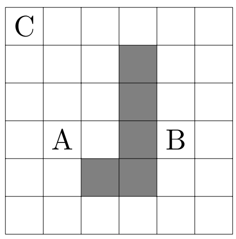
b) Investigate your initial representation of the puzzle as you did in part (a): is there any redundant information in your representation? Try to remove any redundant information from it.

c) Try to solve the puzzle by enumerating possible states.

d) Implement and solve the puzzle using python.

Path planning

Question 5. Imagine a maze with some obstacles, you can only move 1 cell at a time in any of the four directions (no diagonals). We have to find the shortest path from A to B.



a) Discuss whether depth first search and breadth first search will always find a path from A to B. Will they find the shortest path?

b) Suppose that we want to use informed search in order to guide our search. What heuristic could you use ? Is this heuristic admissible ?

c) Implement the A* algorithm for this maze using python (you can use the code template (see page 1)).

d) Now suppose that there is a direct underground connection from C to B. Hence, the shortest path from A to B now is over C (it is only five steps). Is your heuristic still admissible ?

Question 6. The optimality condition of A* is expressed as follows: "If the heuristic h(n) is admissible, then the solution returned by A* is optimal". In order to prove this, we assume that the optimal solution n* has an optimal cost f(n*) = C*.

a) Suppose that there is a solution G2 which is not optimal in the agenda. What does this teach us about f(G2)?

b) Suppose that there is a node n in the agenda which is on the path to the optimal solution G. What does this learn about f(n)?

c) Proove that if h(n) is admissible, then the solution returned by A* is optimal.