COMP3308/3608 Artificial Intelligence

Week 2 Tutorial exercises
Search Problem Formulation. Uninformed Search. Informed Search: Greedy.

Welcome to your first AI tutorial! Please note the following:

- **The tutorial notes typically include more exercises than what we can do in the 1-hour tutorial. We will do only some of them in class, but you need to do the remaining at your own time. The exercises are very useful to help you understand the material, that's why we have prepared more. We will provide the solutions to all of them (at 6pm on Wednesday, after the last tutorial).**

- **The homework for this week includes Exercise 1. The homework exercises are marked with "(Homework)". Usually only 1 exercise is given for homework.**

- **The homework exercises for the regular and advanced stream are typically the same, unless otherwise specified.**

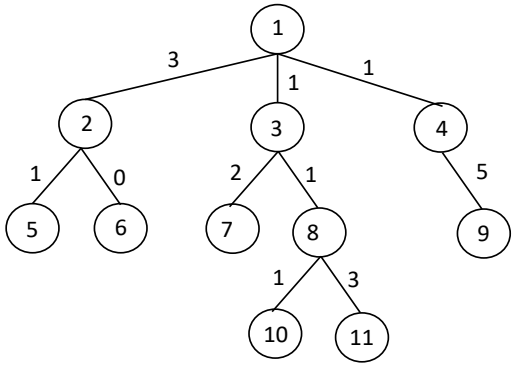
- **A reminder about how to submit the homework: via the submission box in Canvas, due on Tuesday 4pm every week.**

Exercise 1. **Uninformed search (Homework)**

Given is the tree below, where the tree branches are labeled with their cost. List the order in which the nodes are expanded using the following search strategies:

a) Breadth-First Search (BFS);
b) Uniform Cost Search (UCS) - when there is a tie, i.e. more than 1 candidate with the same cost, expand first the left-most node as shown on the figure.
c) Depth-First Search (DFS)
d) Iterative-Deepening Search (IDS).

Assume that none of the nodes are goal nodes, i.e. show the traversal of the whole tree using the search strategies.



a) BFS
Expanded:

b) UCS
Expanded:

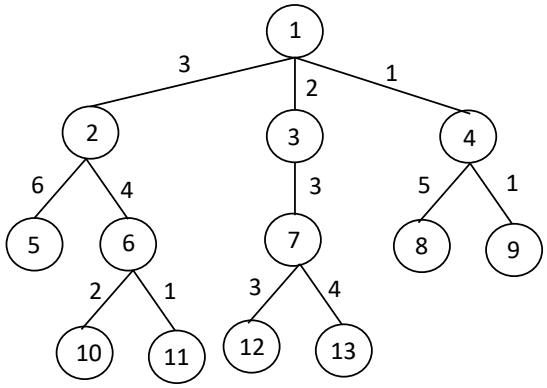
c) DFS
Expanded:

d) IDS
Expanded:

Solution:

a) BFS: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11
b) UCS: 1, 3, 4, 8, 2, 6, 7, 10, 5, 11, 9
c) DFS: 1, 2, 5, 6, 3, 7, 8, 10, 11, 4, 9
d) IDS: 1, 1, 2, 3, 4, 1, 2, 5, 6, 3, 7, 8, 4, 9, 1, 2, 5, 6, 3, 7, 8, 10, 11, 4, 9

Exercise 2. **Uninformed search**
The same questions as in the previous exercise but for the following tree:



a) BFS
Expanded:

b) UCS
Expanded:

c) DFS
Expanded:

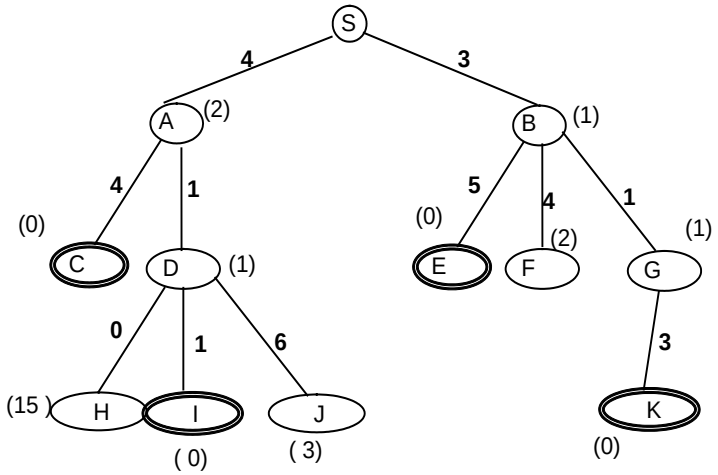
d) IDS
Expanded:

Solution:

a) BFS: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13
b) UCS: 1, 4 ,3, 9, 2, 7, 8, 6, 11, 12, 5, 10, 13 (the same rule for dealing with ties as in the previous exercise – the left-most node was expanded first)
c) DFS: 1, 2, 5, 6, 10, 11, 3, 7, 12, 13, 4, 8, 9
d) IDS: 1 1,2,3,4, 1,2,5,6,3,7,4,8,9, 1,2,5,6,10,11,3,7,12,13,4,8,9

Exercise 3. **Greedy search**
Consider the tree below. The step costs are indicated along the edges and the heuristic values *h* are shown in brackets. The goal nodes are circled twice, i.e. they are C, I, E and K. Run the greedy search

algorithms. Show the list of expanded nodes and the solution path found with its cost. If there are nodes with the same priority for expansion, expand the last added first.



Expanded nodes:
Path found:
Path cost:

Solution:

Expanded nodes: SBE
Path found: SBE
Path cost: 8

Explanation:
We will us the following notation: {expanded} | {fringe}
Greedy search selects nodes for expansion based on the h-value. Note that:

- Selection for expansion comes before checking for goal node, and this applies for all search strategies
- We keep the nodes in the fringe ordered in increasing order, based on their desirability which is determined by the search strategy. For greedy search this means that the node with the smallest *h* value should be at the front of the fringe. In other words, we maintain a priority queue. So, we always select the front of the fringe, and then check if it is a goal node.
- For the nodes in the fringe we will also put in brackets the value used by the algorithm when selecting a node for expansion, in this case the *h* value

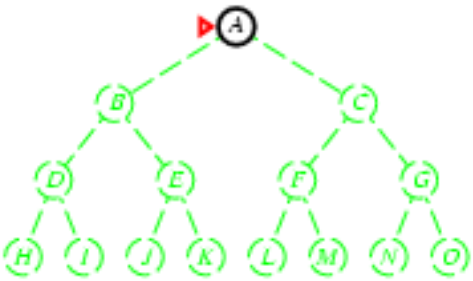
Empty | S (expanded: empty, fringe: S)
Select S for expansion (the only element in the fringe). Is S a goal node? No. Expand it, i.e. move it to expanded and add A and B to the fringe:
S | A[2], **B[1]**
Select a node for expansion from the fringe: B (as its *h* value is the smallest). Is B a goal node? No. Expand it. In **bold** we will show the selected nodes for expansion, i.e. the one that is at the front of the queue.

S, B | A[2], **E[0],** F[2], G[1]
S, B, E | A[2], F[2], G[1]

Exercise 4. IDS

Consider the following tree where O is the goal. (Ignore the different type of node outlines). The branching factor is b=2, the depth of the solution is d=3. Run IDS and check that the number of expanded nodes to find the solution is:

$1(d+1)+b^1d+b^2(d-1)+\ldots+b^d1 = 1(d+1) + b^1d + b^2(d-1) + b^31$



Answer:

IDS will expand the following nodes:
A
A, B, C
A, B, D, E, C, F, G
A, B, D, H, I, E, J, K, C, F, L, M, G, N, O

=> 1 node (A) will be expanded 4 times =1* 4 = 1(d+1)
=> 2 nodes (B and C) will be expanded 3 times = $2^13=b^1d$
=> 4 nodes (D, E, F and G) will be expanded 2 times = $2^22=b^2(d-1)$
=> 8 nodes (H, I, J, K, L, M, N and O) will be expanded 1 time = $2^31=b^31$

=>the number of expansions is $1(d+1) + b^1d + b^2(d-1) + b^31$

Exercise 5. The n-rooks problem

Given is a n x n board and the following task: starting with an empty board, put one rook at the left-most column on a square that is not attacked by any other rook. A rook attacks another rook horizontally or vertically.

Show that the state space has $n!$ states.

Solution:

The first rook can be placed in any square of column 1 (n choices), the second in any square in column 2 except in the row of the first rook (n-1 choices) and so on. In total this gives $S=n(n-1)\ldots1=n!$ states.

Exercise 6. The n-queens problem (Advanced students only)

Consider the n-queens problem using the efficient incremental formulation from the lectures. Recall that it starts with empty n x n board and at each step puts one queen at the left-most column on a square that is not attacked by any other queen. Show that the state space has at least $\sqrt[3]{n!}$ states.

Hint: Consider the maximum number of squares that a queen can attack in the following column and then derive a lower bound for the number of states.

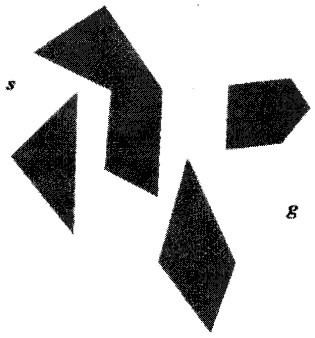
Solution:

There are *n* choices for the first queen. Notice that a queen attacks **at most** 3 squares from the next column, so for the second queen there are at least *(n-3)* choices, for the third queen there are at least *(n-6)* choices and so on:
$S = n(n-3)(n-6)\ldots$

Now let's consider S^3:
$S^3 = n\ n\ n\ (n-3)\ (n-3)(n-3)\ (n-6)\ (n-6)\ (n-6)\ \ldots$

This is even smaller than (i.e. we are deriving a lower bound):
$>= n\ (n-1)\ (n-2)\ (n-3)\ (n-4)\ (n-5)\ (n-6)\ (n-7)\ (n-8)\ ... = n!$
$=> S >= \sqrt[3]{n!}$

Exercise 7. (Advanced students only)
A robot must find the shortest path between a starting point *s* and a goal point *g* in the two-dimensional space populated by polygonal obstacles shown below. The path can be adjacent to (touching) the obstacles, but cannot intersect any of them. Assume the robot is of infinitesimal size.



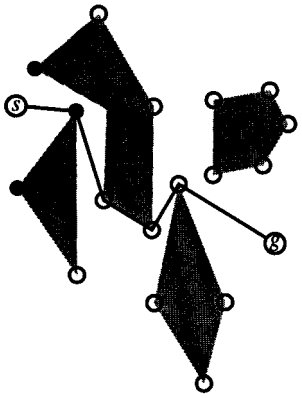
a) Draw the shortest path from s to g.
b) Although there are an infinite number of points in this two dimensional space, what is the minimal set of points that must be considered in searching for a shortest path between an arbitrary s point and an arbitrary *g* point?
c) Given one of the points in the minimal set you just found, describe a method for generating the successors of this point in the corresponding search graph. What are the successor points for point *s* in the figure?

Answer:
a) The shortest path is shown on the figure below. The shortest path between two points is a straight line. In this case it is not possible to travel in a straight line as some of the obstacles are in the way, so the next shortest distance is a sequence of line segments that deviate from the straight line as little as possible. So the first segment of this sequence must go from the start point to a tangent point on an obstacle – any path that gave the obstacle a wider girth would be longer. Because the obstacles are polygonal, the tangent points must be vertices of the obstacles, and hence the entire path must go from vertex to vertex.

An intuitive feeling for shortest paths among polygonal objects can be obtained by imagining that we tie one end of a string to the starting point, weave the string through the objects along the best path to the goal and then tighten the string. The best path will hit only vertices of the convex hulls of the objects.

b) The minimal set of points include *s*, *g* and all the vertices of the *convex hulls* of all polygonal objects (shown with circles).

c) The successors of a point *p* are all the vertices that are "visible" from *p* of the convex hulls of the polygons and also *g* if it is visible from *p*. The successors of *s* are shown by the three darkened circles.



Additional exercises (to be done at your own time)

Exercise 8. **Missionary and cannibals**

Three missionaries and three cannibals come to a river. There is a boat on their side of the river that can be used by either 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?

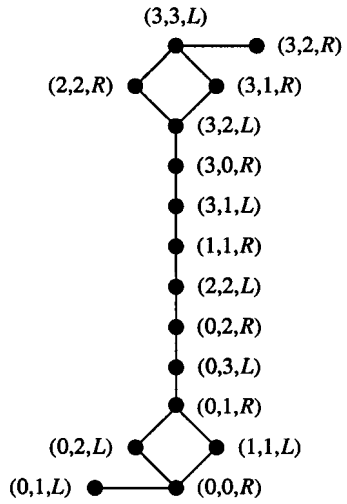
a) Design and describe an appropriate representation for a state in this search problem.
b) Under your chosen representation, what are the initial and goal states?
c) Draw the graph of the states and label its nodes. Include only those states that are "legal", i.e. states in which the cannibals do not outperform missionaries on either side of the river.
d) Why do you think people have a hard time solving this puzzle, give than the state space is so simple?

Answer:

a) We can use a state description of the following form:
(num_missionaries_start_side, num_cannibals_start_side, boat_location)
The location of the boat has two values: left (L) and right (R).

b) Then, the start state is (3,3,L) and the goal state is (0,0,R).

c) The state space is shown below:



d) Most of the moves are either illegal or go back to the previous state. There is a feeling of a large branching factor and no clear way to proceed.

***Exercise 9*. Farmer, wolf, goat and cabbage**

A farmer is trying to cross a river with a wolf, a goat and a cabbage. He has a row boat that he can use to carry at most one item at a time (plus himself) across the river. Only the farmer can row the boat so he has to be with the boat in each of its trips across the river. The farmer's problem is that he can't leave the wolf alone with the goat, or the goat alone with the cabbage, at any time. How should they use the boat to cross the river? For this question, you are to design a state-space search approach to this problem.

a) Design and describe an appropriate representation for a state in this search problem.
b) Under your chosen representation, what is a goal state for the search problem?
c) Under your chosen representation, show the initial state and its legal successors.
d) In general, what conditions need to be in order to generate legal successors of a state?
e) Conduct the search using your representation.

Answer:

a) A five component binary vector (F, W, G, C, B) where a value of 1 corresponds to the left hand side (the side they arrived first) and a value of 0 to right hand-side. Note that F and B will have the same values (the farmer is always on the same side as the boat as he drives the boat), so you may delete the last component and use a 4-component vector instead (remembering that F represent the position of the boat too).

b) Start state = (1, 1, 1, 1, 1), Goal state = (0,0,0,0, 0)
c) (1,1,1,1, 1) ->(0,1,0,1,0) (there is only one legal successor state)
d) Illegal states: if F=1 and W=G=0 or G=C=0; if F=0 and W=G=1 or G=C=1
e)