# COMP3411/9414 Artificial Intelligence
# Session 1, 2019

# Week 4 Tutorial Solutions

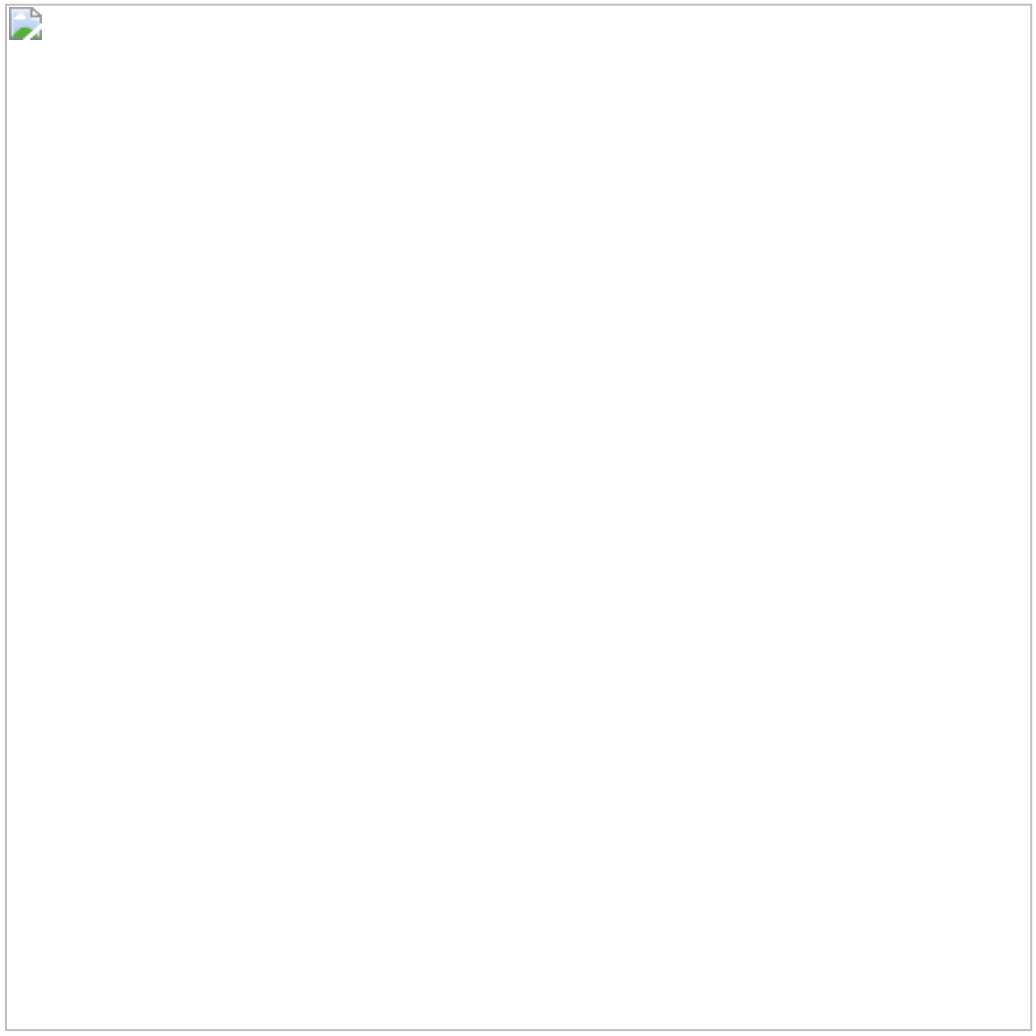
**This page was last updated: 04/30/2019 06:05:01**

---

**Activitity 4.1** Path Search Algorithms on a Graph

---

Consider the task of finding a path from start state S to goal state G, given the distances and heuristic values in this diagram:



For each of the following strategies, list the order in which the states are expanded. Whenever there is a choice of states, you should select the one that comes first in alphabetical order. In each case, you should skip any states that have previously been expanded, and you should continue the search until the goal node is expanded.

a. Breadth First Search

S, A, C, B, D, T, U, X, V, W, Y, Z, E, F, G

b. Depth First Search

S, A, B, T, D, C, U, W, E, F, G

c. Uniform Cost Search [Hint: first compute g() for each state in the graph]

S(0), A(2), C(3), B(4), D(5), U(6), X(7), T(8), V(9), W(10), Y(11), F(12), Z(13), E(14), G(15)

d. Greedy Search, using the heuristic shown

S, C, X, Z, G

e. A*Search, using the heuristic shown

S(9), C(10), A(11), B(11), D(12), X(12), W(12), U(13), Z(14), F(15), G(15)

Note that g(X) becomes 8 after C is expanded, but drops to 7 after D is expanded. Similarly, g(G) becomes 16 when Z is expanded, but drops to 15 after F is expanded.

---

**Activitity 4.2** A*Search for the 8-Puzzle

---

Consider the following arrangement of tiles in the 8-puzzle:

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 8 | 5 | |
| 4 | 7 | 6 |

Keeping in mind that the goal state is:

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | |

Trace the A* Search algorithm using the Total Manhattan Distance heuristic, to find the shortest path from the initial state shown above, to the goal state.

Note: The algorithm begins by exploring the left branch which, according to the heuristic, seems the most promising. After a few steps the heuristic for the left branch starts to exceed that of middle branch, so exploration shifts to the middle branch, where the optimal solution is ultimately found.

---

**Activitity 4.3** Relationships Between Search Strategies

---

Prove each of the following statements, or give a counterexample:

a. Breadth First Search is a special case of Uniform Cost Search

Uniform Cost Search reduces to Breadth First Search when all edges have the same cost.

b. Breadth First Search, Depth First Search and Uniform Cost Search are special cases of best-first search.

best-first search reduces to Breadth-First Search when $f(n)$=number of edges from start node to $n$, to UCS when $f(n)=g(n)$;
it can be reduced to DFS by, for example, setting $f(n)$=-(number of nodes from start state to $n$) (thus forcing deep nodes on the current branch to be searched before shallow nodes on other branches). Another way to produce DFS is to set $f(n)=-g(n)$ (but this might produce a particularly bad choice of nodes within the DFS framework - for example, try tracing the order in which nodes are expanded when traveling from Urziceni to Craiova).

c. Uniform Cost Search is a special case of A$^*$Search

A$^*$Search reduces to UCS when the heuristic function is zero everywhere, i.e. $h(n)=0$ for all $n$;
this heuristic is clearly admissible since it always (grossly!) underestimates the distance remaining to reach the goal.

---

**Activitity 4.4** Heuristic Path Algorithm

---

The **heuristic path algorithm** is a best-first search in which the objective function is

$$f(n) = (2-w)g(n) + wh(n)$$

What kind of search does this perform when $w$=0? when $w$=1? when $w$=2?

This algorithm reduces to Uniform Cost Search when $w$=0, to A$^*$Search when $w$=1 and to Greedy Search when $w$=2.

For what values of $w$ is this algorithm complete? For what values of $w$ is it optimal, assuming $h()$ is admissible?

It is guaranteed to be optimal when $0 \leq w \leq 1$, because it is equivalent to A$^*$Search using the heuristic

$h'(n) = [w/(2-w)]h(n) \leq h(n)$

When $w > 1$ it is not guaranteed to be optimal (however, it might work very well in practice, for some problems).

---

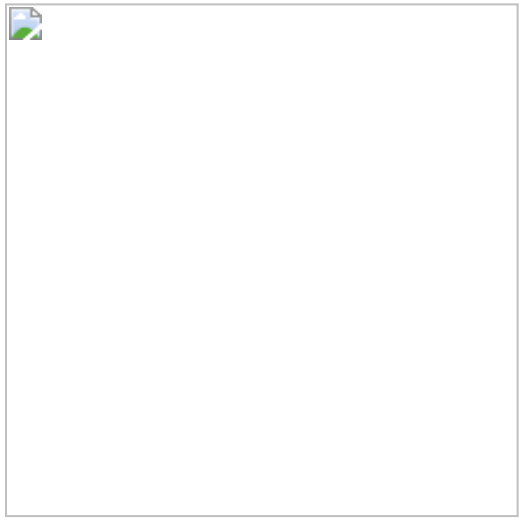
**Activitity 4.5** Understanding Informed Search Algorithms with Mazes

---

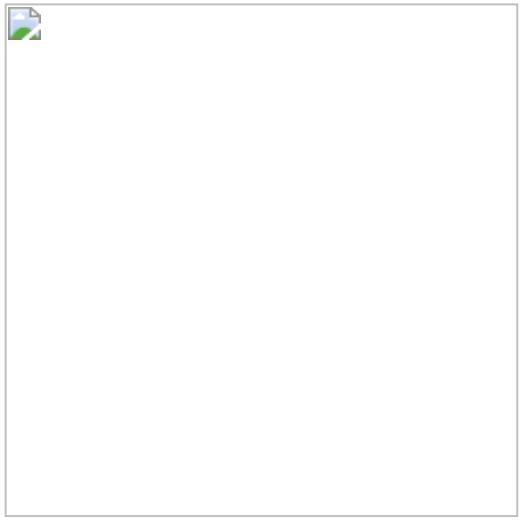
Discuss your findings and insights from the 'Fun with Mazes' activity. Compare your findings and discuss any discrepancies.

a. Use the widget to create a maze for which Bidirectional Search would find a solution faster than Breadth First Search.

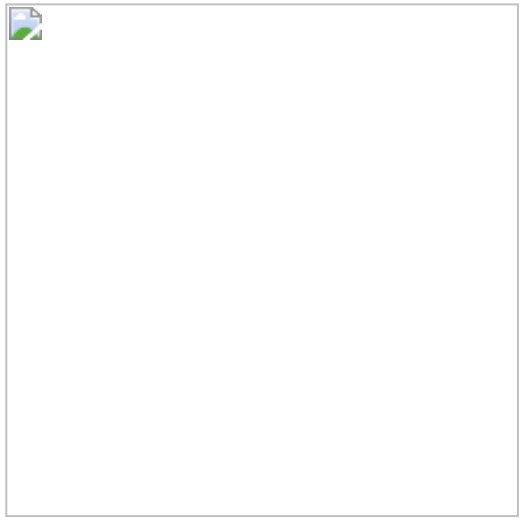
To see the real benefits of bidirectional search you would need a larger branching factor than is possible in a 2-dimensional maze. However, in a fairly open maze with the initial and goal state equally spaced at one-third and two-thirds of the width, bidirectional search should theoretically take about half the time of BFS.



maze

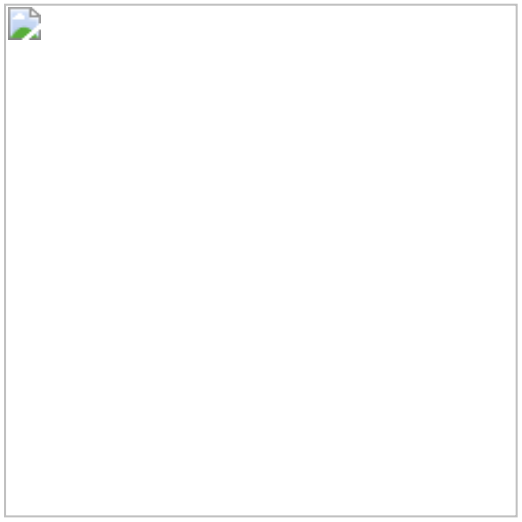


Breadth First Search

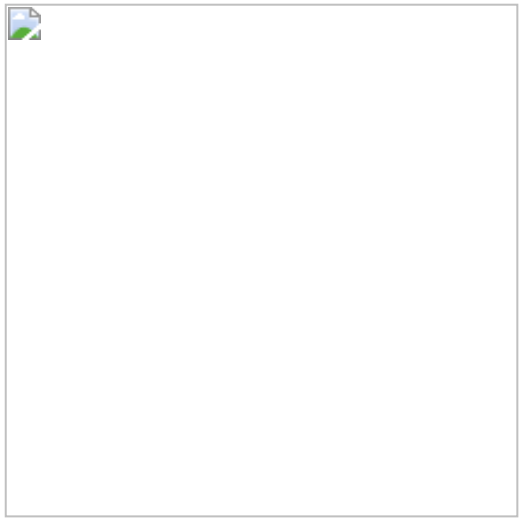


Bidirectional Search

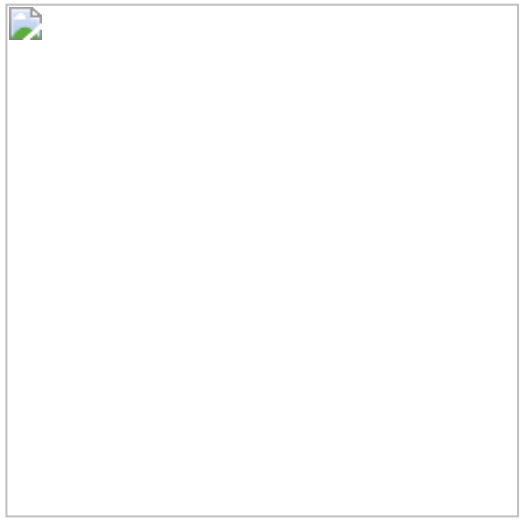
b. an environment for which Greedy Search takes much longer than A*Search.



Maze

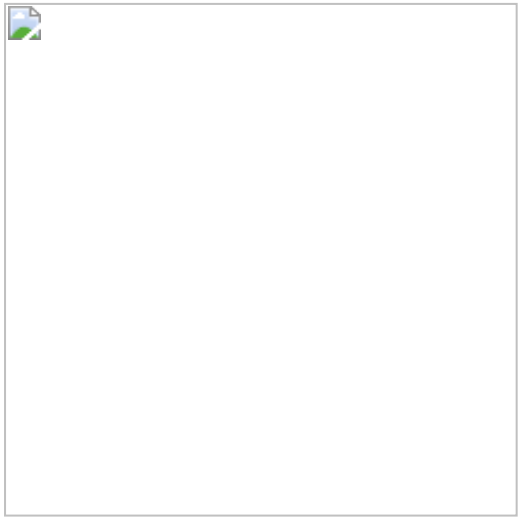


Greedy Search

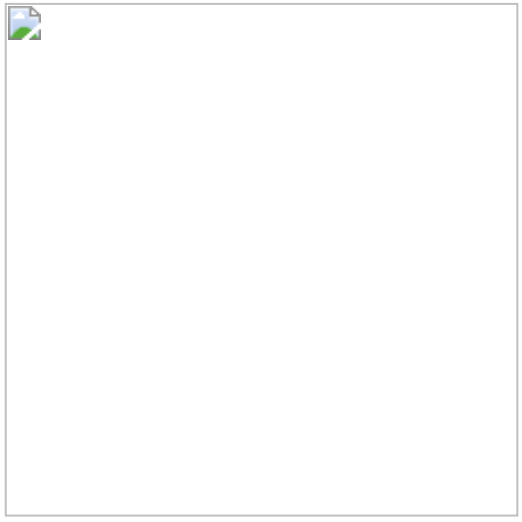


A*Search

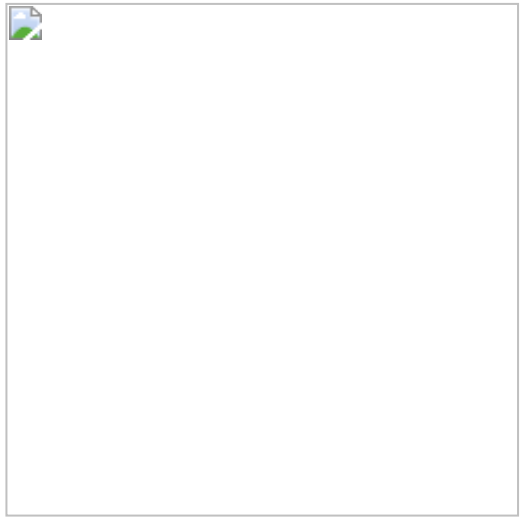
c. an environment for which Greedy Search produces a path that is much longer than the optimal path.



Maze

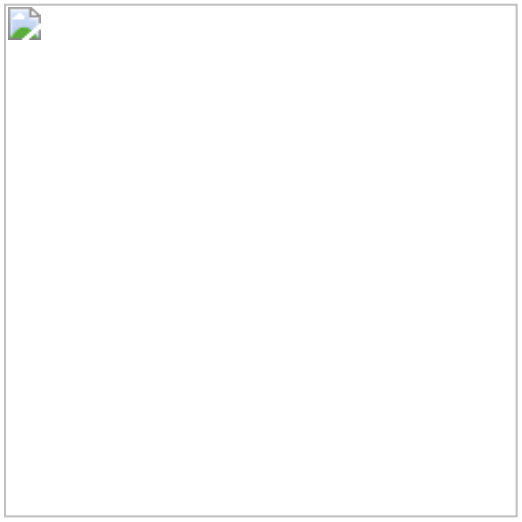


Greedy Search (1127)

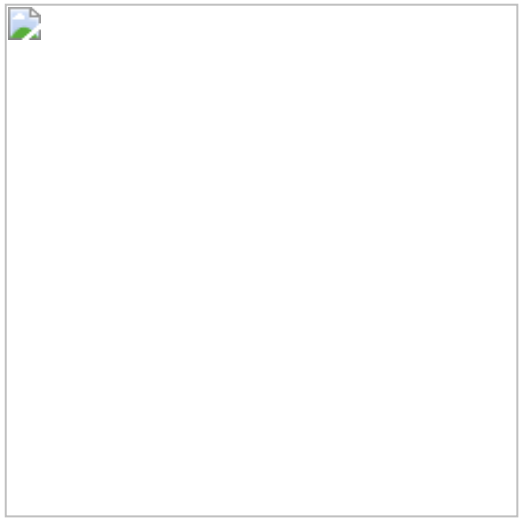


A*Search (77)

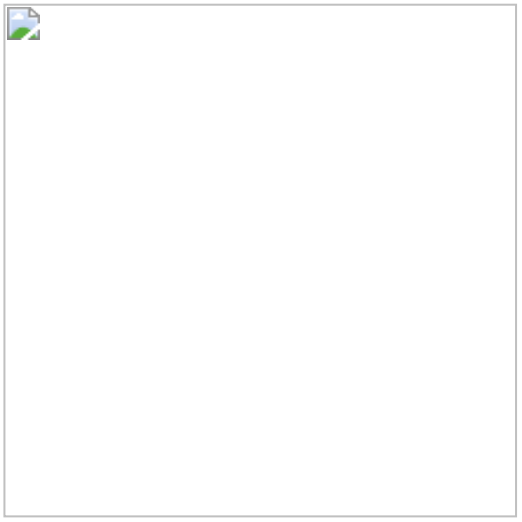
d. an environment for which A*Search with the Euclidean Distance heuristic takes much longer than with the Manhattan Distance heuristic.



Maze



A*Search (Euclidean)



A*Search (Manhattan)

e. an environment that is interesting for some other reason.

---