**Timezone has changed.**

Your current timezone is set to: Australia/Sydney but you are now in Asia/Shanghai.

You can change timezone settings in your **account settings (https://www.openlearning.com/accounts/account-settings/)**

11
(/progress/?course=unswcourses%2Fcourses%2Fcomp3411-9814-21t0&student=tongzhang-qmji6o)

29%

Tutorial: Path Search

## 1: The Missionaries & Cannibals Problem

The "Missionaries and Cannibals" problem is usually stated as follows: Three missionaries and three cannibals are on one side of the river, along with a boat that can hold one or two people. Find a way to get everyone to the other side, without ever leaving a group of missionaries in one place outnumbered by the cannibals in that place. This problem is famous in Artificial Intelligence because it was the subject of the first paper that approached problem formulation from an analytical viewpoint (Amarel, 1968).

What would be the best way to represent each "state" in the Missionaries and Cannibals problem? How many possible states are there? Make only those distinctions necessary to ensure a valid solution. Draw a diagram of the complete state space.

1 Formulate the problem precisely, making only those distinctions necessary to ensure a valid solution, and draw a diagram of the complete state space.

2 Solve the problem optimally using an appropriate search algorithm&semi&semi&semi&semi; is it a good idea to check for repeated states?

3 Why do you think people have a hard time solving this puzzle, given that the state space is so simple?

**Skip to main content**

# Activity 2: Refining Your Understanding of Searches using Mazes

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

(a) Generate a random Tree Maze and compare the running time of Breadth First Search and Depth First Search on this maze. Repeat this for a couple of other random Tree Mazes. Time the algorithms with a stopwatch if you can. Which algorithm is faster?

(b) Repeat the steps from part (a), this time with Concentric Graph Mazes. Which algorithm is faster?

(c) Use the widget to edit a maze by clicking on the coloured squares, and then clicking on the Maze. Try to design a Maze for which BFS finds a solution considerably faster than DFS. You can specify a starting and ending point anywhere inside the maze. Try to (briefly) explain in words what makes your environment easy for BFS but hard for DFS.

(d) Use the widget to create a maze for which DFS would find a solution considerably faster than BFS. You should assume that there can be multiple ending points (red squares) and that the algorithm only needs to reach one of them. Try to explain in words what makes your environment easy for DFS but hard for BFS.

(e) Try running Iterative Deepening Search on a random maze. Why is it so slow? For which type of problem (probably not a maze) would IDS be superior to both BFS and DFS?

Skip to main content

# Activity 3: Path Search Algorithms on a Graph

**Timezone has changed.**

Your current timezone is set to: Australia/Sydney but you are now in Asia/Shanghai.

You can change timezone settings in your account settings (https://www.openlearning.com/accounts/account-settings/)

For the route from Arad to Bucharest, what order are nodes in the state space expanded for each of the following algorithms when searching for the shortest path between Arad and Bucharest? Where there is a choice of nodes, take the first one by alphabetical ordering. Make sure you understand the key properties of the different algorithms, as listed below.

Anna Trofimova (https://www.openlearning.com/u/atrofimova/) - a month ago

1 person likes this Like Subscribe History Subpages

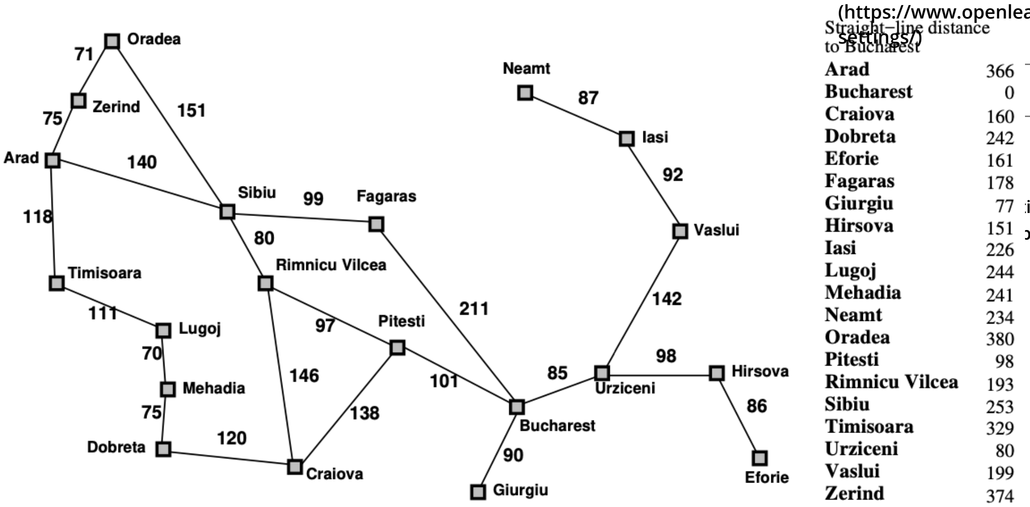
Discussion Space

Sort by: **New Threads**

Use this space to note

**Back:** Uniform Cost Se
(https://www.openlearnin
cost-search/)

ive Deepening Search
o3411-9814-21t0/iterative-
deepening-search/)



| Straight–line distance to Bucharest | |
|---|---|
| **Arad** | 366 |
| **Bucharest** | 0 |
| **Craiova** | 160 |
| **Dobreta** | 242 |
| **Eforie** | 161 |
| **Fagaras** | 178 |
| **Giurgiu** | 77 |
| **Hirsova** | 151 |
| **Iasi** | 226 |
| **Lugoj** | 244 |
| **Mehadia** | 241 |
| **Neamt** | 234 |
| **Oradea** | 380 |
| **Pitesti** | 98 |
| **Rimnicu Vilcea** | 193 |
| **Sibiu** | 253 |
| **Timisoara** | 329 |
| **Urziceni** | 80 |
| **Vaslui** | 199 |
| **Zerind** | 374 |

1. Depth-first search (efficient use of space but may not terminate

2. Breadth-first search (space inefficient, guaranteed to find a solution)

3. Uniform-cost search (similar to breadth-first, but order nodes by cost)

4. Iterative deepening depth-first search (space efficient, but repeated work)