# CS 331 Assignment #1: Uninformed and Informed Search

Out: April 7, 2017
Due: April 21, 2017 at 17:00:00
Type: Team Assignment (Groups of 2 max)

---

## Description

In the cannibals and missionaries puzzle, M missionaries and C cannibals must cross from the right bank of a river to the left bank using a boat. The boat holds a maximum of two people. In addition, the boat cannot cross the river by itself and must have at least one person on board to drive it. This problem seems simple except for the following key constraint. If there are missionaries present on a bank, there cannot be more cannibals than missionaries, otherwise the cannibals will eat the missionaries.

You can try this puzzle out with 3 missionaries and 3 cannibals by clicking here.

If you can't figure out how to solve the 3 missionaries and 3 cannibals case, you can watch the solution here.

In this assignment, you will write code to solve the missionaries and cannibals puzzle using uninformed and informed search algorithms. The algorithms you will implement are breadth-first search, depth-first search, iterative deepening depth-first search and A-star search. Your code will print out the states along the solution path from the start state to the goal state. If no such path exists, your code must print out a *no solution found* message. In addition, your code must also print out the number of search nodes expanded. You will need to know this number to fill out a table comparing the different algorithms.

---

## File Format

The state of a game is represented by two comma-separated lines in a file. The first line stores the number of missionaries, the number of cannibals, and the number of boats on the left bank. The second line stores the number of missionaries, the number of cannibals, and the number of boats on the right bank. Since there is only one boat in the puzzle, the number of boats could be interpreted as a Boolean flag indicating the presence of the boat.

For example, the initial state

0,0,0
3,3,1

represents 3 missionaries, 3 cannibals and the boat being on the right bank of the river and nothing on the left bank.

Your program will take as input a starting state file and a goal state file. Your code must be able to read the start and goal state files.

---

# Requirements

You may the following programming languages: Java, C, C++, Python. If you use any other programming language, it must be available on the ENGR systems and you must contact the Teaching Assistant (Laurel Hopkins, for this assignment) for approval. Your code should take the following command line arguments: < initial state file > < goal state file > < mode > < output file >

The mode argument is either:

- bfs (for breadth-first search)
- dfs (for depth-first search)
- iddfs (for iterative deepening depth-first search)
- astar (for A-Star search below)

## I. Uninformed Search

You will implement Breadth-First Search, Depth-First Search and Iterative-Deepening Depth First Search. **Use the pseudocode for the GraphSearch function and the Expand function in either the slides or the textbook to help you design the algorithm**. If you don't follow the GraphSearch and Expand pseudocode, your algorithms may not work properly.

There are specific requirements regarding the uniformed search algorithms that you must follow in this assignment:

- In order to generate successors for a state, you need to do an action, generate the next state, then add that state to a list of successors. To make things consistent for grading, we require you to generate successor states using the following order on the actions:
    1. Put one missionary in the boat
    2. Put two missionaries in the boat
    3. Put one cannibal in the boat
    4. Put one cannibal and one missionary in the boat
    5. Put two cannibals in the boat

    If any of these actions are not valid for the current state, ignore that action and move on to the next one. To illustrate the successor generation, suppose we start with nothing on the left bank and 3 missionaries, 3 cannibals and 1 boat on the right bank. The successor states will be:

    1. Left Bank: 0 missionaries, 1 cannibal, 1 boat Right Bank: 3 missionaries, 2 cannibals, 0 boat
    2. Left Bank: 1 missionary, 1 cannibal, 1 boat Right Bank: 2 missionaries, 2 cannibals, 0 boat
    3. Left Bank: 0 missionaries, 2 cannibals, 1 boat Right Bank: 3 missionaries, 1 cannibal, 0 boat

    Notice that the actions for putting one missionary in the boat and putting two missionaries in the boat are invalid because it would result in more cannibals on the right bank than missionaries.

- You will need a counter for the number of nodes expanded. Update this counter after you remove the node from the fringe and just before you call the expand function (note that if it is a goal node, you don't need to expand it and hence don't count the goal node as an expanded node). At the end of the graph search, print out the number of nodes that were expanded.
- You need to print the solution path which shows how to move the missionaries and cannibals from the start state to the goal state. Please print this to the screen and to an output file. Make sure it is in the right order!

## II. Informed Search

In this part, you will implement A-star search. As with uninformed search, there are specific requirements that you must follow:

- When generating successors for a state, do it in the same order as for the Uninformed Search algorithms above.
- Keep track of the number of nodes expanded in the same way as specified for the Uninformed Search above.
- You will need a priority queue. You may use built-in code from your programming language (eg. Java) for the priority queue.
- You will need a heuristic for A-star search. Pick one but make sure it is admissible.
- You need to print the solution path which shows how to move from the start state to the goal state. Please print this to the screen and to an output file. Make sure it is in the right order!

---

# Test Files

There are three test cases that you should run your algorithm on. These tests and the corresponding files are:

1. Test1: testStart1.txt, testGoal1.txt
2. Test2: testStart2.txt, testGoal2.txt
3. Test3: testStart3.txt, testGoal3.txt

Record how many nodes were on the solution path and how many nodes were expanded for each of BFS, DFS, ID-DFS and A-Star Search. You will need to set a max depth for ID-DFS. Try to make the max depth as large as possible.

---

# The Assignment Report

Your assignment report should be about 1-2 pages and should not exceed 2 pages. We accept pdf or doc files (pdf is preferred). For your report, you will do the following sections:

- **Methodology:** Describe the experiments you ran on the three test cases. Make sure you specify all parameters that you used for the search algorithms (eg. the depth limit in DFS, heuristic for A-star search). You can assume the reader is familiar with search algorithms and you don't need to describe how the search algorithms operate. However, the description should be complete enough that someone else can reimplement these search algorithms on their own and re-run the exact same set of experiments using the parameters you used. Make sure you explain why you chose the heuristic for A-star search.
- **Results:** Summarize your results for the three test cases in terms of the # of nodes on the solution path and # of nodes expanded on each test graph for each of the search techniques. Use either a table or a graph to illustrate the results.
- **Discussion:** Discuss your results. Are they as expected? Was there any interesting behavior that you found?
- **Conclusion:** Add a conclusion which answers the following questions: What can you conclude from these results? Which search algorithm performs the best? Was this expected?

---

# Hints

1. You will need a closed list to keep track of the states that you have already visited. Use a hash table for this.

2. If you are using Java and running out of memory, add the -mx1024M flag to the Java Virtual Machine (this is done through Eclipse).

---

# Hand in

Please hand in all of your source code and your report. Also include a text file called team.txt which lists the names of the members of your team. Zip everything up with a zip program. To hand in your assignment, go to the TEACH electronic handin site: https://secure.engr.oregonstate.edu:8000/

1. Login to the TEACH system
2. Click on the "Submit Assignment" link on the left hand side
3. Select ProgrammingAssn1 from the dropdown menu, hit submit query
4. For redundancy, leave a web comment stating who your team members are. If I don't know who is on your assignment team, they won't get credit.
5. Enter the path of your zip file. Hit Submit Query to hand everything in.

---

# Grading Criteria

The assignment is out of 50. Note that the bulk of the marks are allocated to your report. The code is intended to help you produce results for the experiments. We will run the code to make sure it works.

- Report (30 points):
  - Methodology (5 points)
  - Results (10 points)
  - Discussion (10 points)
  - Conclusion (5 points)
- Code (20 points)