Search in Python

Chapter 3

Today's topics

- AMAI Python code
- What it does
- How to use it
- Worked example: water jug program

Install AIMA Python code with pip

- For some of the HW assignments, you'll need access the aima python software
- Install aima module on your own Linux or Mac

  sudo pip install aima

- Install without sudo privileges

  pip install aima --user

- This won't work on UMBC's gl servers because pip is not installed

Working on gl

- On gl, you tell Python to look in the directory we've set up for 471 python code
- Or you can set up your own directory (e.g., ~/mypython) in which you install new packages
- For either, you must first add the appropriate directories to your PYTHONPATH environment variable
  - Do this by modifying your shell initialization file (e.g., ~/.cshrc or ~/.bashrc)

Python and PYTHONPATH

- Python's import command looks for modules to load in a list of places
- sys.path is the list, with ‘ ‘ as the current directory

```
>>> import sys
>>> sys.path
[‘ ‘, '/usr/lib64/python26.zip', …]
```

- On Unix, when python starts, it prepends directories on your PYTHONPATH environment variable
- Add new directories for python to search by setting PYTHONPATH in the init file used by your shell
- The Unix command *echo $SHELL* shows what shell you are using

AIMA Python code

- Install aima module on your own Linux or Mac

  sudo pip install aima

- Install without sudo privileges

  pip install aima --user

- Install on gl (no pip ☹)
  - Add to .bashrc to set directory for packages

    export PYTHONPATH= ~/mypy:
  - **easy_install -d ~/mypy aima**
- Use our installation, add to .bashrc
  - export PYTHONPATH= ~finin/pub/471python:

Using the 471 installation on gl

- *echo $SHELL* shows what shell you are using
- If using tcsh shell, add to your .cshrc file

  setenv PYTHONPATH ~finin/pub/471python
- If using bash shell, add

  PYTHONPATH= ~finin/pub/471python:

Installing your own packages on gl

- You can also install aima (or other packages) in your own library directory, e.g., *~/mypy*
- Step #1: add ~/mypy to PYTHONPATH in your shell initialization file
  - tcsh: setenv PYTHONPATH ~/mypy
  - bash: PYTHONPATH= ~finin/pub/471python:
- Step #2: use easy_install and specify the directory to put the files, e.g.
  - easy_install -d ~/mypy aima

Overview

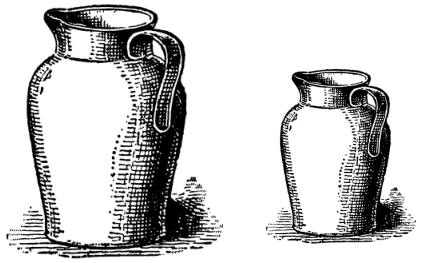
To use the AIMA python code for solving the two water jug problem (WJP) using search we need one problem-specific file:

- **wj.py**: defines the problem, states, goal, actions, costs, etc.

And one general file:

- **search.py**: AIMA's generic search framework, imported by wj.py

Two Water Jugs Problem

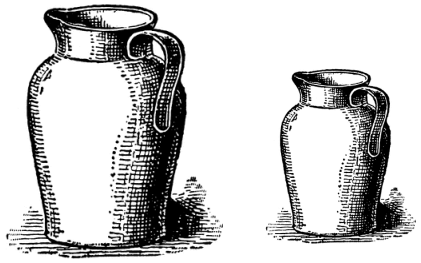


- Given two water jugs, J1 and J2, with capacities C1 and C2 and initial amounts W1 and W2, find actions to end up with amounts W1' and W2' in the jugs
- Example problem:
  - We have a 5 gallon and a 2 gallon jug
  - Initially both are full
  - We want to end up with exactly one gallon in J2 and don't care how much is in J1

search.py

- Defines a *Problem* class for a search problem
- Provides functions to perform various kinds of search given an instance of a Problem, e.g., breadth first, depth first, hill climbing, A*, …
- *InstrumentedProblem* subclasses *Problem* and is used with *compare_searchers* for evaluation
- To use for WJP: (1) decide how to represent the WJP, (2) define *WJP* as a subclass of *Problem* and (3) provide methods to (a) create a WJP instance, (b) compute successors and (c) test for a goal

Two Water Jugs Problem



Given J1 and J2 with capacities C1 and C2 and initial amounts W1 and W2, find actions to end up with W1' and W2' in jugs

State Representation

State = (x,y), where x & y are water in J1 & J2

- Initial state = (5,0)
- Goal state = (*,1), where * is any amount

Operator table

Actions	Cond.	Transition	Effect
Empty J1	–	$(x,y)\rightarrow(0,y)$	Empty J1
Empty J2	–	$(x,y)\rightarrow(x,0)$	Empty J2
2to1	$x \leq 3$	$(x,2)\rightarrow(x+2,0)$	Pour J2 into J1
1to2	$x \geq 2$	$(x,0)\rightarrow(x-2,2)$	Pour J1 into J2
1to2part	$y < 2$	$(1,y)\rightarrow(0,y+1)$	Pour J1 into J2 until full

Our WJ problem class

```
class WJ(Problem):
    def __init__(self, capacities=(5,2), initial=(5,0), goal=(0,1)):
        self.capacities = capacities
        self.initial = initial
        self.goal = goal

    def goal_test(self, state):  # returns True iff state is a goal state
        g = self.goal
        return (state[0] == g[0] or g[0] == '*' ) and \
                (state[1] == g[1] or g[1] == '*')

 def __repr__(self):     # returns string representing the object
        return "WJ({},{},{})".format(self.capacities, self.initial, self.goal)
```

Our WJ problem class

```
def actions(self, (J0, J1)):
    """ generates legal actions for state """
    (C0, C1) = self.capacities
    if J0 > 0: yield 'dump0'
    if J1>0: yield 'dump1'
    if J1<C1 and J0>0: yield 'pour_0_1'
    if J0<C0 and J1>0: yield 'pour_1_0'
```

Our WJ problem class

```
def result(self, state, action):
    (J0, J1) = state
    (C0, C1) = self.capacities
    if action == 'dump0': return (0, J1)
    elif action == 'dump1': return (J0, 0)
    elif action == 'pour_0_1':
      delta = min(J0, C1-J1); return (J0-delta, J1+delta)
    elif action == 'pour_1_0':
       delta = min(J1, C0-J0); return (J0+delta, J1-delta)
    raise ValueError('Unrecognized action: ' + action)
```

Our WJ problem class

```
def h(self, node):
    # heuristic function that estimates distance
    # to a goal node
    return 0 if self.goal_test(node.state) else 1
```

Solving a WJP

```
code> python
>>> from wj import *                                  # Import wj.py and search.py
>>> from aima.search import *
>>> p1 = WJ((5,2), (5,2), ('*', 1))                   # Create a problem instance
>>> p1
WJ((5, 2),(5, 2),('*', 1))
>>> answer = breadth_first_search(p1)    # Used the breadth 1st search function
>>> answer                                            # Will be None if the search failed or a
<Node (0, 1)>                                         #    a goal node in the search graph if successful
>>> answer.path_cost                                  # The cost to get to every node in the search graph
6                                                     #  is maintained by the search procedure
>>> path = answer.path()                              # A node's path is the best way to get to it from
>>> path                                              #   the start node, i.e., a solution
[<Node (0, 1)>, <Node (1, 0)>, <Node (1, 2)>, <Node (3, 0)>, <Node (3, 2)>, <Node (5, 0)>, <Node (5, 2)>]
>>> path.reverse()
>>> path
[<Node (5, 2)>, <Node (5, 0)>, <Node (3, 2)>, <Node (3, 0)>, <Node (1, 2)>, <Node (1, 0)>, <Node (0, 1)>]
```

Comparing Search Algorithms Results

Uninformed searches: breadth_first_tree_search, breadth_first_search, depth_first_graph_ search, iterative_deepening_search, depth_limited_ search

- All but depth_limited_search are **sound** (i.e., solutions found are correct)
- Not all are **complete** (i.e., can find all solutions)
- Not all are **optimal** (find best possible solution)
- Not all are **efficient**
- AIMA code has a comparison function

Comparing Search Algorithms Results

```
HW2> python
Python 2.7.6 |Anaconda 1.8.0 (x86_64)| ...
>>> from wj import *
>>> searchers=[breadth_first_search, depth_first_graph_search, iterative_deepening_search]
>>> compare_searchers([WJ((5,2), (5,0), (0,1))], ['SEARCH ALGORITHM', 'successors/goal tests/states generated/solution'], searchers)
SEARCH ALGORITHM            successors/goal tests/states generated/solution
breadth_first_search        <   8/   9/  16/(0, >
depth_first_graph_search    <   5/   6/  12/(0, >
iterative_deepening_search  <  35/  61/  57/(0, >
>>>
```

The Output

```
hhw2> python wjtest.py -s 5 0 -g 0 1
Solving WJ((5, 2),(5, 0),(0, 1)
   breadth_first_tree_search cost 5: (5, 0) (3, 2) (3, 0) (1, 2) (1, 0) (0, 1)
   breadth_first_search cost 5: (5, 0) (3, 2) (3, 0) (1, 2) (1, 0) (0, 1)
   depth_first_graph_search cost 5: (5, 0) (3, 2) (3, 0) (1, 2) (1, 0) (0, 1)
   iterative_deepening_search cost 5: (5, 0) (3, 2) (3, 0) (1, 2) (1, 0) (0, 1)
   astar_search cost 5: (5, 0) (3, 2) (3, 0) (1, 2) (1, 0) (0, 1)
SUMMARY: successors/goal tests/states generated/solution
breadth_first_tree_search    <  25/  26/  37/(0, >
breadth_first_search         <   8/   9/  16/(0, >
depth_first_graph_search     <   5/   6/  12/(0, >
iterative_deepening_search   <  35/  61/  57/(0, >
astar_search                 <   8/  10/  16/(0, >
```