# Water Jugs (WJ) — Combined Report

fractal13

2025-09-01

Github Repository: https://github.com/fractal13/ut-cs4300-202540-simple-search

This document consolidates the PEAS analysis, problem formulation, summarized results table, search analysis, and raw results for the Water Jugs puzzle as used in this repository.

## PEAS Assessment

- Performance measure
  - Success: reach target volume in any jug.
  - Efficiency: minimize number of actions.
  - Robustness: handle multiple jugs and invalid actions gracefully.
- Environment
  - Deterministic, fully observable, discrete, finite state space (tuple of volumes for each jug).
  - Single-agent, episodic, turn-based actions.
- Actuators
  - Actions: fill(jug), empty(jug), pour(from, to).
- Sensors
  - Observe current volumes in all jugs.

## Problem model

- State: tuple of current volumes for each jug.
- Initial state: all jugs empty.
- Actions: fill(i), empty(i), pour(i,j) with deterministic transitions.
- Goal: some jug contains the target volume.
- Cost: unit cost per action.

## Results table (summary)

| Domain | Algorithm | Capacities | Target | Solution Cost | Depth | Nodes Generated | Nodes Expanded | Max Frontier |
|---|---|---|---|---|---|---|---|---|
| WJ | BFS | (3,5) | 4 | 9.0 | 6 | 46 | 13 | 3 |
| WJ | IDS | (3,5) | 4 | 9.0 | 6 | 847 | 843 | 13 |
| WJ | BFS | (8,5,3) | 4 | 9.0 | 6 | 522 | 72 | 25 |
| WJ | IDS | (8,5,3) | 4 | 9.0 | 6 | 13817 | 13808 | 23 |
| WJ | BFS | (2,4) | 2 | 1.0 | 1 | 2 | 1 | 2 |
| WJ | IDS | (2,4) | 2 | 1.0 | 1 | 4 | 3 | 2 |

Note: statistics copied from the raw results. See Discussion below about measurement methodology.

## Analysis: BFS vs IDS

- Expected behavior
  - BFS finds shortest-depth solutions for unit-cost steps; requires more frontier memory.
  - IDS finds shortest-depth solutions using less memory but repeats work across depth iterations.
- Observed behavior
  - Both algorithms found equal-cost solutions in these instances.
  - IDS generated far more nodes than BFS (repeated depth-limited searches), e.g., 847 vs 46 for (3,5) and 13817 vs 522 for (8,5,3).
  - Max frontier sizes remained modest; measurement definitions may vary between algorithms.
- Recommendation
  - For these small Water Jugs instances, BFS is more CPU-efficient; IDS trades CPU for lower memory usage and may be useful when memory is constrained.

## Measurement caveat

The reported "Max Frontier" values are taken from raw outputs. When comparing algorithms, ensure consistent definitions for counted structures (frontier only vs frontier+explored vs recursion stack).

## Addendum: Raw results

The following are the original raw run outputs used to build the summary table.

### BFS Reports

```
Running WaterJugsProblem capacities=(3, 5) target=4
Domain: WaterJugsProblem | Algorithm: BFS
```

```
Solution cost: 9.0 | Depth: 6
Nodes generated: 46 | Nodes expanded: 13 | Max frontier: 3
Path:
  1) ('fill', 1)     capacities=(3,5) volumes=(0,0) -> capacities=(3,5) volumes=(0,5)
  2) ('pour', 1, 0)  capacities=(3,5) volumes=(0,5) -> capacities=(3,5) volumes=(3,2)
  3) ('empty', 0)    capacities=(3,5) volumes=(3,2) -> capacities=(3,5) volumes=(0,2)
  4) ('pour', 1, 0)  capacities=(3,5) volumes=(0,2) -> capacities=(3,5) volumes=(2,0)
  5) ('fill', 1)     capacities=(3,5) volumes=(2,0) -> capacities=(3,5) volumes=(2,5)
  6) ('pour', 1, 0)  capacities=(3,5) volumes=(2,5) -> capacities=(3,5) volumes=(3,4)

Running WaterJugsProblem capacities=(8, 5, 3) target=4
Domain: WaterJugsProblem | Algorithm: BFS
Solution cost: 9.0 | Depth: 6
Nodes generated: 522 | Nodes expanded: 72 | Max frontier: 25
Path:
  1) ('fill', 1)     capacities=(8,5,3) volumes=(0,0,0) -> capacities=(8,5,3) volumes=(0,5,0
  2) ('pour', 1, 2)  capacities=(8,5,3) volumes=(0,5,0) -> capacities=(8,5,3) volumes=(0,2,3
  3) ('empty', 2)    capacities=(8,5,3) volumes=(0,2,3) -> capacities=(8,5,3) volumes=(0,2,0
  4) ('pour', 1, 2)  capacities=(8,5,3) volumes=(0,2,0) -> capacities=(8,5,3) volumes=(0,0,2
  5) ('fill', 1)     capacities=(8,5,3) volumes=(0,0,2) -> capacities=(8,5,3) volumes=(0,5,2
  6) ('pour', 1, 2)  capacities=(8,5,3) volumes=(0,5,2) -> capacities=(8,5,3) volumes=(0,4,3

Running WaterJugsProblem capacities=(2, 4) target=2
Domain: WaterJugsProblem | Algorithm: BFS
Solution cost: 1.0 | Depth: 1
Nodes generated: 2 | Nodes expanded: 1 | Max frontier: 2
Path:
  1) ('fill', 0)     capacities=(2,4) volumes=(0,0) -> capacities=(2,4) volumes=(2,0)
```

### IDS Reports

```
Running WaterJugsProblem capacities=(3, 5) target=4
Domain: WaterJugsProblem | Algorithm: IDS
Solution cost: 9.0 | Depth: 6
Nodes generated: 847 | Nodes expanded: 843 | Max frontier: 13
Path:
  1) ('fill', 1)     capacities=(3,5) volumes=(0,0) -> capacities=(3,5) volumes=(0,5)
  2) ('pour', 1, 0)  capacities=(3,5) volumes=(0,5) -> capacities=(3,5) volumes=(3,2)
  3) ('empty', 0)    capacities=(3,5) volumes=(3,2) -> capacities=(3,5) volumes=(0,2)
  4) ('pour', 1, 0)  capacities=(3,5) volumes=(0,2) -> capacities=(3,5) volumes=(2,0)
  5) ('fill', 1)     capacities=(3,5) volumes=(2,0) -> capacities=(3,5) volumes=(2,5)
  6) ('pour', 1, 0)  capacities=(3,5) volumes=(2,5) -> capacities=(3,5) volumes=(3,4)

Running WaterJugsProblem capacities=(8, 5, 3) target=4
Domain: WaterJugsProblem | Algorithm: IDS
Solution cost: 9.0 | Depth: 6
```

```
Nodes generated: 13817 | Nodes expanded: 13808 | Max frontier: 23
Path:
  1) ('fill', 1)     capacities=(8,5,3) volumes=(0,0,0) -> capacities=(8,5,3) volumes=(0,5,0
  2) ('pour', 1, 2)  capacities=(8,5,3) volumes=(0,5,0) -> capacities=(8,5,3) volumes=(0,2,3
  3) ('empty', 2)    capacities=(8,5,3) volumes=(0,2,3) -> capacities=(8,5,3) volumes=(0,2,0
  4) ('pour', 1, 2)  capacities=(8,5,3) volumes=(0,2,0) -> capacities=(8,5,3) volumes=(0,0,2
  5) ('fill', 1)     capacities=(8,5,3) volumes=(0,0,2) -> capacities=(8,5,3) volumes=(0,5,2
  6) ('pour', 1, 2)  capacities=(8,5,3) volumes=(0,5,2) -> capacities=(8,5,3) volumes=(0,4,3

Running WaterJugsProblem capacities=(2, 4) target=2
Domain: WaterJugsProblem | Algorithm: IDS
Solution cost: 1.0 | Depth: 1
Nodes generated: 4 | Nodes expanded: 3 | Max frontier: 2
Path:
  1) ('fill', 0)     capacities=(2,4) volumes=(0,0) -> capacities=(2,4) volumes=(2,0)
```