AI Lab 2 - Search

The Jugs Puzzle

We are given 2 jugs, a 4 liter(x L) one and a 3 liter(y L) one. Neither has any measuring markers on it. A pump can be used to fill the jugs with water. How can we get exactly 2(z L) liters of water into any of the jugs?

Problem Solving Agent

- We are going to develop an agent.
- The agent will perform a sequence of actions.
- Will try to reach the goal. (2L in one of the bottles)

Characteristics of the Problem Solving Agent

- Problem Formulation:
  - Define State –limited
  - Define the steps – limited
- Goal Formulation
- Environment Characteristics:
  - Observable – the agent knows
  - Discreet – Finite no of states
  - Deterministic – one action, one state

Search Problem Characteristics

- **Initial State** – (0,0)
- **Actions Set** –
  - Empty left
  - Empty Right
  - Fill Left
  - Fill Right
  - Transfer from left to right
  - Transfers from right to left

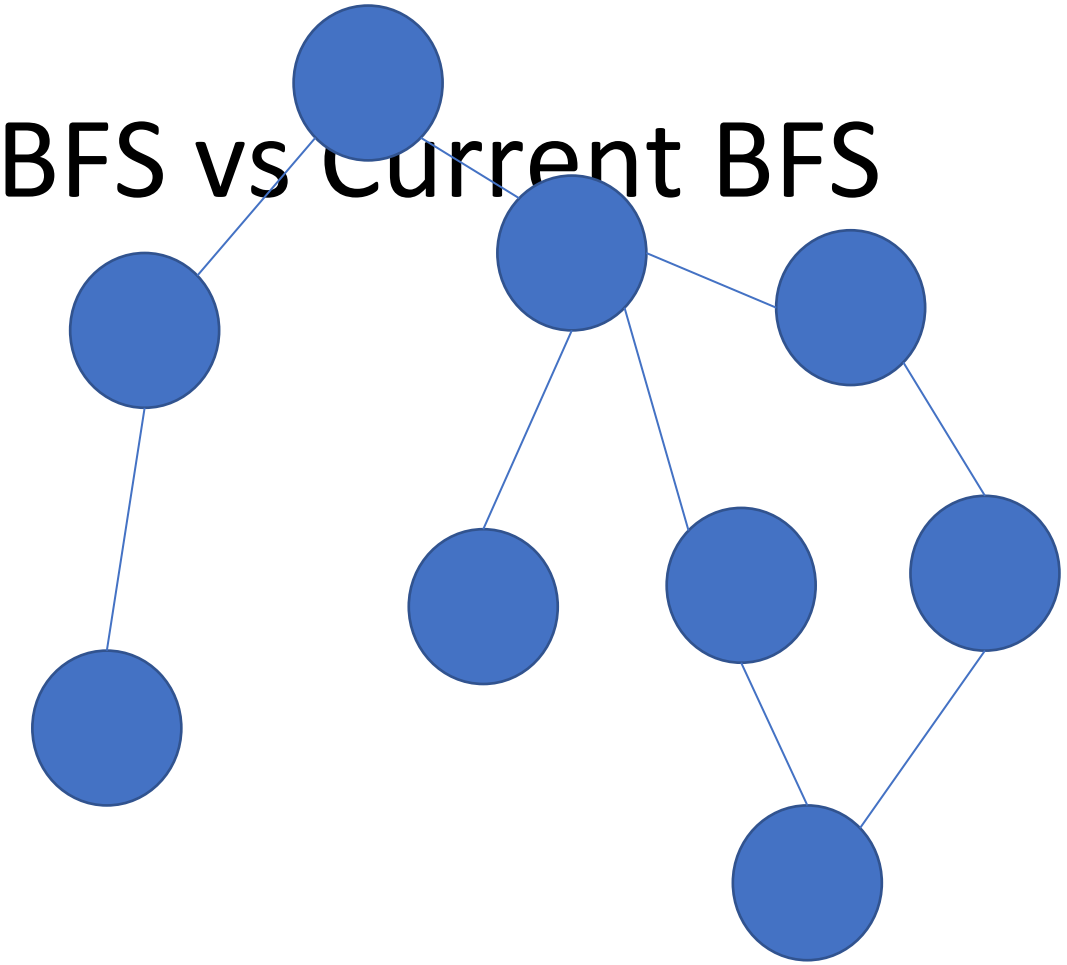
Search Problem Characersitics

- **Transition Model** -
  - (0,0)-----fill left--□ (3,0)
- **Goal Test** –
  - isGoal(2,4)-> True
  - isGoal(1,1) ->False
- **Path Cost** – The cost of each action taken

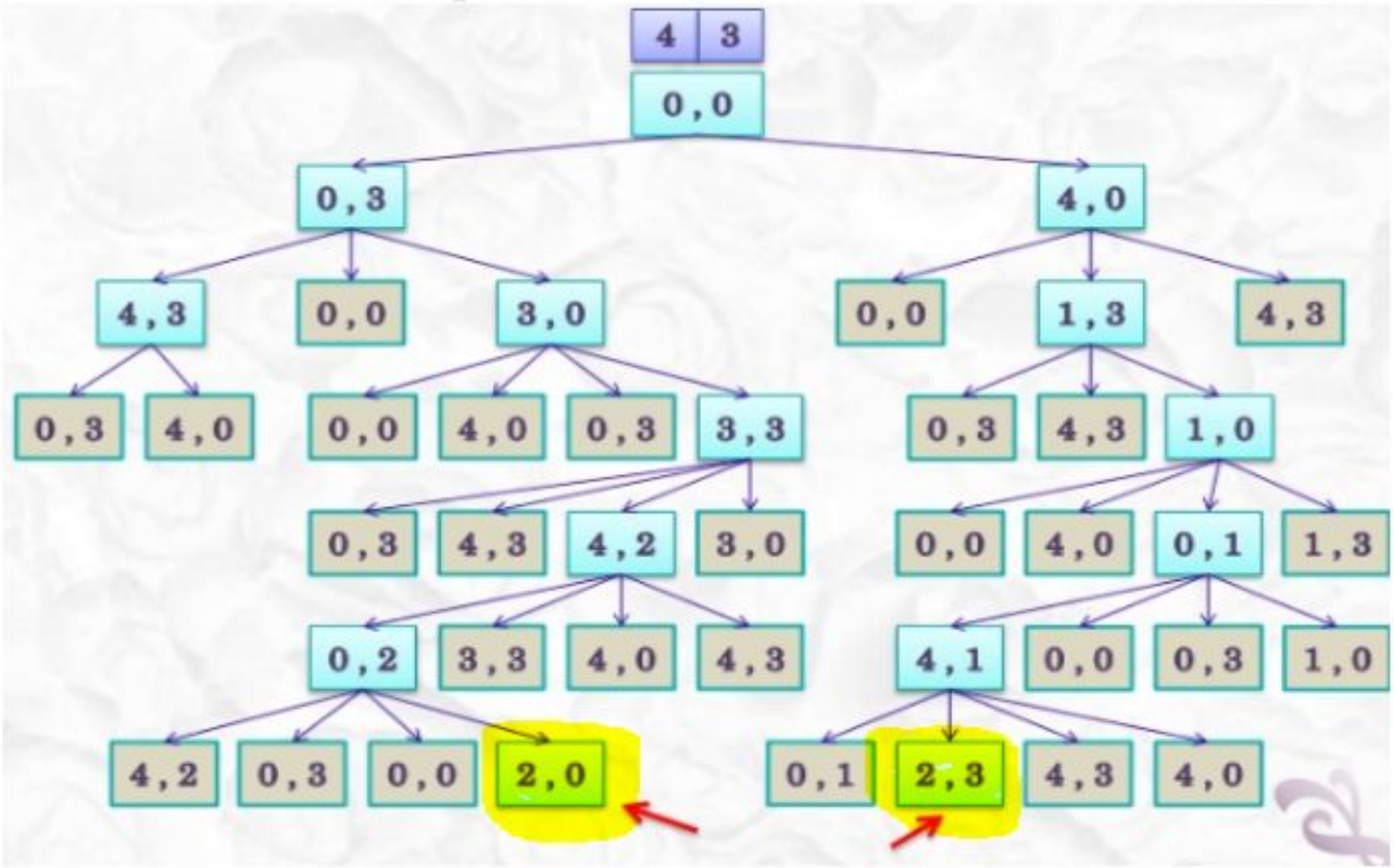
```
function BREADTH-FIRST-SEARCH(problem) returns a solution, or failure
    node ← a node with STATE = problem.INITIAL-STATE, PATH-COST = 0
    if problem.GOAL-TEST(node.STATE) then return SOLUTION(node)
    frontier ← a FIFO queue with node as the only element
    explored ← an empty set
    loop do
        if EMPTY?(frontier) then return failure
        node ← POP(frontier)  /* chooses the shallowest node in frontier */
        add node.STATE to explored
        for each action in problem.ACTIONS(node.STATE) do
            child ← CHILD-NODE(problem, node, action)
            if child.STATE is not in explored or frontier then
                if problem.GOAL-TEST(child.STATE) then return SOLUTION(child)
                frontier ← INSERT(child, frontier)
```

Figure 3.11 Breadth-first search on a graph.

Previous BFS vs Current BFS



State Map


4 3
0,0
0,3
4,0
4,3
0,0
3,0
0,0
1,3
4,3
0,3
4,0
0,0
4,0
0,3
3,3
0,3
4,3
1,0
0,3
4,3
4,2
3,0
0,0
4,0
0,1
1,3
0,2
3,3
4,0
4,3
4,1
0,0
0,3
1,0
4,2
0,3
0,0
2,0
0,1
2,3
4,3
4,0

Lets Code!