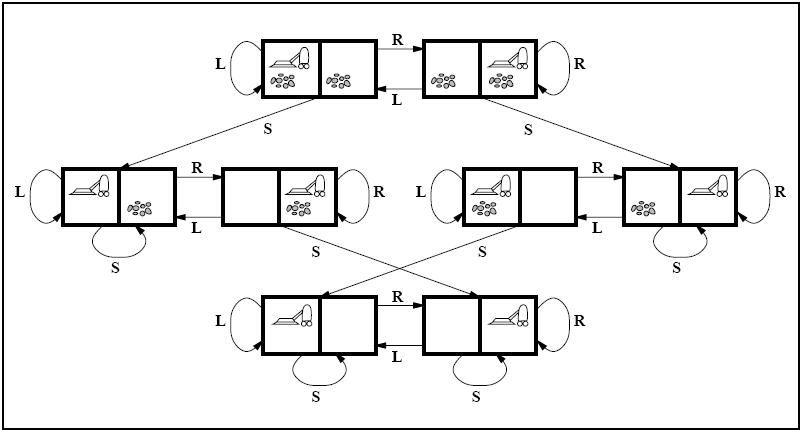
**Week 3 Topic Outline  
Chapter 3 – Searching… finding a sequence of actions that achieves its goals when no single action will do!**

1. Basic outline – remember: Intelligent agents aim to maximize their performance measure (fitness function)
   1. **Define goal** – what do we want to accomplish?
   2. **Search** – process of looking for a sequence of actions that reach the goal. When found, it returns the solution to the agent.
   3. **Solution** – a set of actions to take.
   4. **Begin execution phase** – carry out the step-by-step solution.
   5. Algorithm illustrated on p.67 figure 3.1
2. How to define a problem…called **Well-defined problems**
   1. **Initial State** – where the agent starts
   2. **Actions** – given a particular state, what actions are available
   3. **Transition model** – a description of what each action does (the state that would result from each action)
   4. **State space** – set of all states reachable from the initial state by any sequence of actions
   5. **Path** – sequence of states connected by a sequence of actions
   6. **Goal test** – determines whether a given state is a goal state
      1. Why is this needed?
   7. **Path cost** – a numeric cost for each path (cost of performing an action)
      1. Are path costs always the same? Examples of yes, no.
3. **Well-defined problem example – Vacuum world**

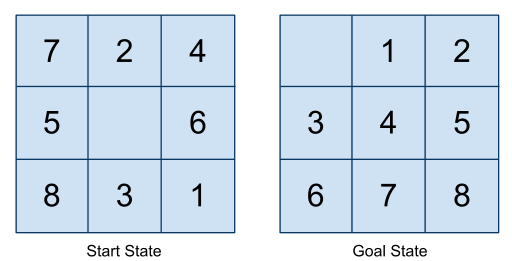


L – **left**, R – **right**, S - **suck**

* 1. **States:**The state is determined by both the agent location and the dirt locations. The agent is in one of two locations, each of which might or might not contain dirt. Thus there are 2 \* 22 = 8 possible world states. A larger environment with n locations has n \* 2n states.
     1. Where does 2 \* 22 come from?
     2. The image above represents the **state space** of the problem.
  2. **Initial State**: Any state can be designated as the initial state.
  3. **Actions**: In this simple environment, each state has just three actions: Left, Right, and Suck. Larger environments might also include Up and Down.
  4. **Transition Model:**The actions have their expected effects, except that moving Left in the leftmost square, moving Right in the rightmost square, and Sucking in a clean square have no effect.
  5. **Goal Test:** This checks whether all the squares are clean.
  6. **Path Cost:**Each steps costs 1, so the path cost is the number of steps in the path.

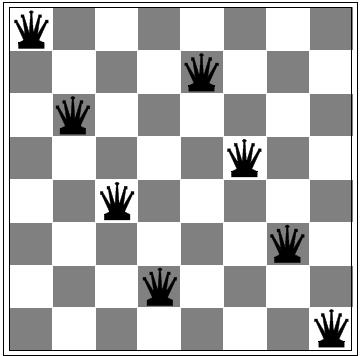
1. **Well-defined problem #1– 9 tile slider**

Define the state, initial state, actions, transition model, goal test, and path cost for the following problem



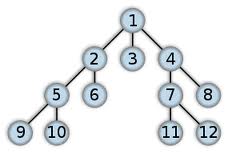
* 1. **States** – where the tiles are. A state description specifies the location of each of the eight tiles and the blank in one of the nine squares
  2. **Initial state** – The arrangement of the tile as shown in the picture. Any state can be designated as the initial state.
  3. **Actions** – Movements of the blank space (*Left, Right, Up, Down*). It is easier to state the actions in terms of the blank space (one) versus the potential movement of the other (eight) tiles
  4. **Transition model** – Given a state and action, this returns the resulting state. The tile in the direction that the action is taken is swapped with the position of the blank tile.
  5. **Goal test** – Checks whether the state matches the goal configuration shown in the figure.
  6. **Path cost** – Each steps costs 1, so the path cost is the number of steps in the path (solution).

1. **Well-defined problem #1– 7 queens**



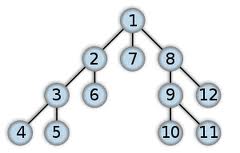
* 1. **States** – Any arrangement of 0 to 8 queens on the board is a state
  2. **Initial state** – No queens on the board
  3. **Actions** – Add a queen to any empty square
  4. **Transition model** – Returns the board with a queen added to the specified square
  5. **Goal test** – 8 queens on the board, none attacked
  6. **Path cost** – Placing a queen on the board – cost 1.

1. **Breadth first search** – Searches shallowest unexpanded node in the current frontier



* 1. Root -> all of root’s successors -> all of successor’s successors -> etc.
  2. FIFO queue
  3. Benefits? Downsides?
  4. Does it always find the goal if it exists?
  5. What happens to the size of the queue

1. **Depth first search** – searches the deepest node in the current frontier



* 1. Proceeds immediately to the deepest level of the search tree, where the nodes have no successors
  2. Children are added in a right-to-left manner in the picture
  3. LIFO queue (Stack)
  4. Good breakdown (p.86)
  5. What happens if there are infinite states?

1. **Depth-limited search** – depth first search with a limit on how deep the search will go
   1. Solves the infinite path problem
2. **Iterative deepening depth-first search** – like the depth-limited search, but gradually increases the depth limit
   1. Sometimes just called **Iterative deepening search**
   2. Continually performs a depth first search at an increasing depth limit
   3. Shown below

