Chapter 3

SEARCH ALGORITHMS

These slides were created by Dr. Rashid Al-zubaidy for Fundamentals of Artificial Intelligence at Philadelphia University

Introduction to search Algorithms

- Real world is very complex. So, the state space is an abstraction of the reality
- Search algorithms is considered as one of the most important aspects in problem solving within AI applications
- State space search are used to model real problems that face us in every day life

Introduction to search Algorithms

- Search algorithms are considered as a way that lead us to solve problem within a specific state space. This will occur through the following:
  - State space
  - finding all alternatives in a systematic way
  - finds the sequence of states that leads to the solution

Definition of a Search Problem

- State space: it is a graph that shows all possible states
- The initial state represents the start state
- The final state (goal) represents the end
- An action is transition from one state to another

Definition of a Search Problem

- A path in the state space is a sequence of actions that leads from one state to another
- Goal test: is a procedure applicable to check whether a state is the required final state
- Path cost: it helps to determine what is the cost of the path leading from the initial state to the goal

Problem Solving

- Problem Solving requires the following:
  - Understand the state of the world in consideration (get initial state)
  - Formulate the problem and goal
  - Search through the state space for a solution
  - Implement the solution

Problem Solving

- To solve a problem, we need normally some sort of object that helps to find the goal
- This requires the following formal approach
- The concept that shows the problem solving. It normally takes an input and gives an output
- Input represents the initial state
- Output represents the final state

Problem Solving Algorithm

function ProblemSolving (input) returns a path

inputs: an input
s, state sequence, initially empty,
State ← initial state
Goal ← final state

if s is empty then
        Search (state space)
action ← move from one state to the next

return path (output)

Example Problems

- Introduction to the examples

  – A.I. problem solving domain deals with both a toy problem and real world problems

  – Such problems will be presented in this lecture to explore the areas of A.I. techniques that deal with toys problems

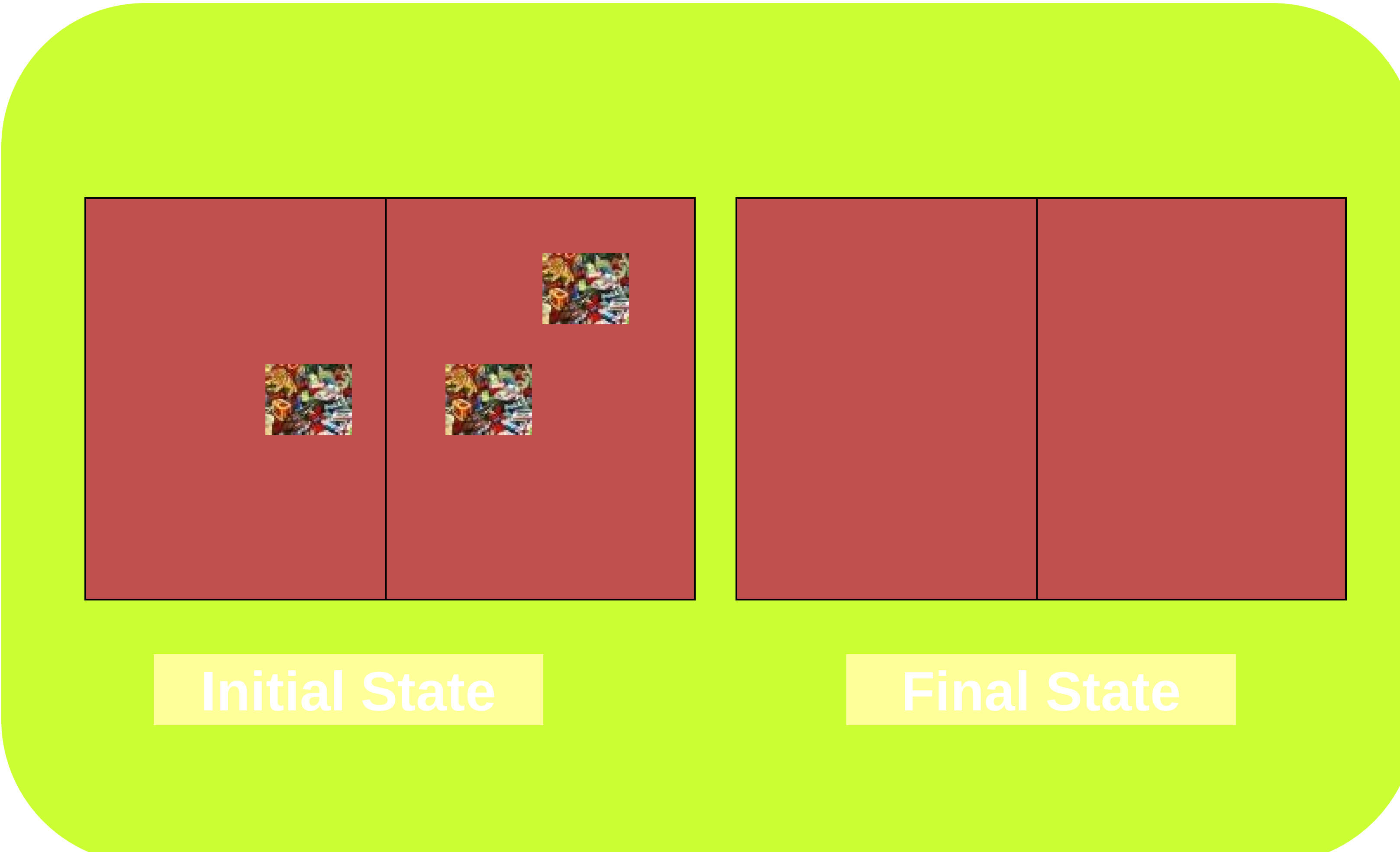
Toys Problems

- Toys problems deals with simple problems within specific domain like:
  - Cleaner robot
  - 8-puzzle
  - Blocks world

Cleaner Robot

- Initial state:
  - cleaner robot in one part of the room
- Actions:
  - move left, move right, suck
- Final state:
  - cleaner robot in the other part of the room
  - dirt is sucked
- Cost-path:
  - number of moves

Cleaner Robot



8-Puzzle

- Initial state:
  - positions of the 8 tablets in one of the 9 slots
- Actions:
  - move the empty slot on the left (L), on the right (R), in top (U), in bottom (D)
- Final state:
  - current state = final state
- Cost-path:
  - number of moves

8-Puzzle

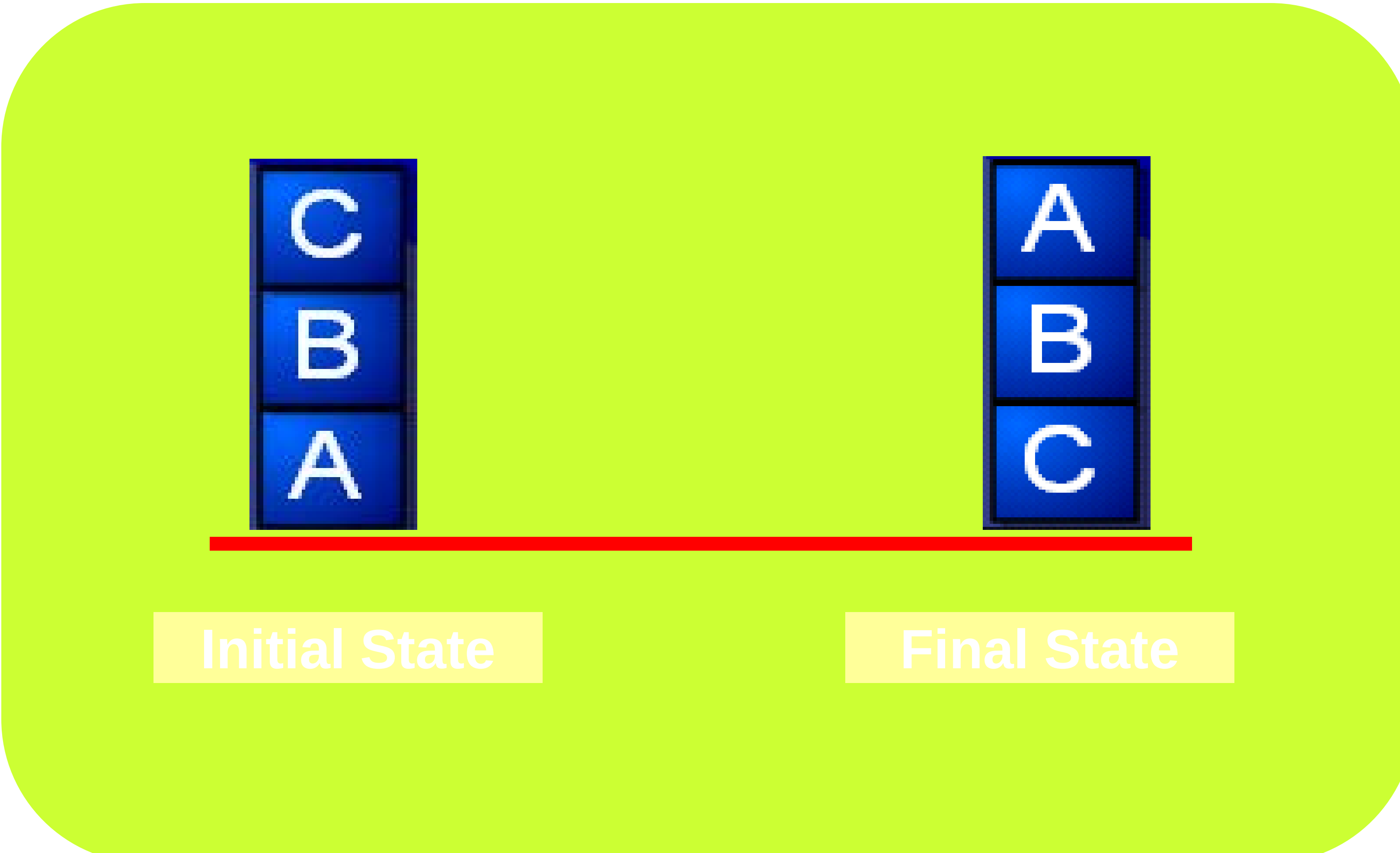
Initial State

Final State

Blocks-World

- Initial state:
  - positions of the blocks on the table
- Actions:
  - move the block from another block or put it at the top of the table
- Final state:
  - current state = final state
- Cost - path:
  - number of moves

Blocks-World



Real World Problems

- A.I experiments start to solve problems in a toy domain in order to see whether such problem solving can be achieved through using real world problems that face human beings

- The following examples try to explode such possibilities

Real World Problems

- Real world problems deals with complex problems within specific domain like:
  - Route finding
  - Robot (components assembler)
  - Routing in computer networks

Route Finding

- Initial state:
  - tourist in Marseille
- Actions:
  - tourist move from a city to another one
- Final state:
  - tourist in Paris
- Cost-path:
  - Sum of distances taken by the tourist

Route Finding

Robot Components Assembler

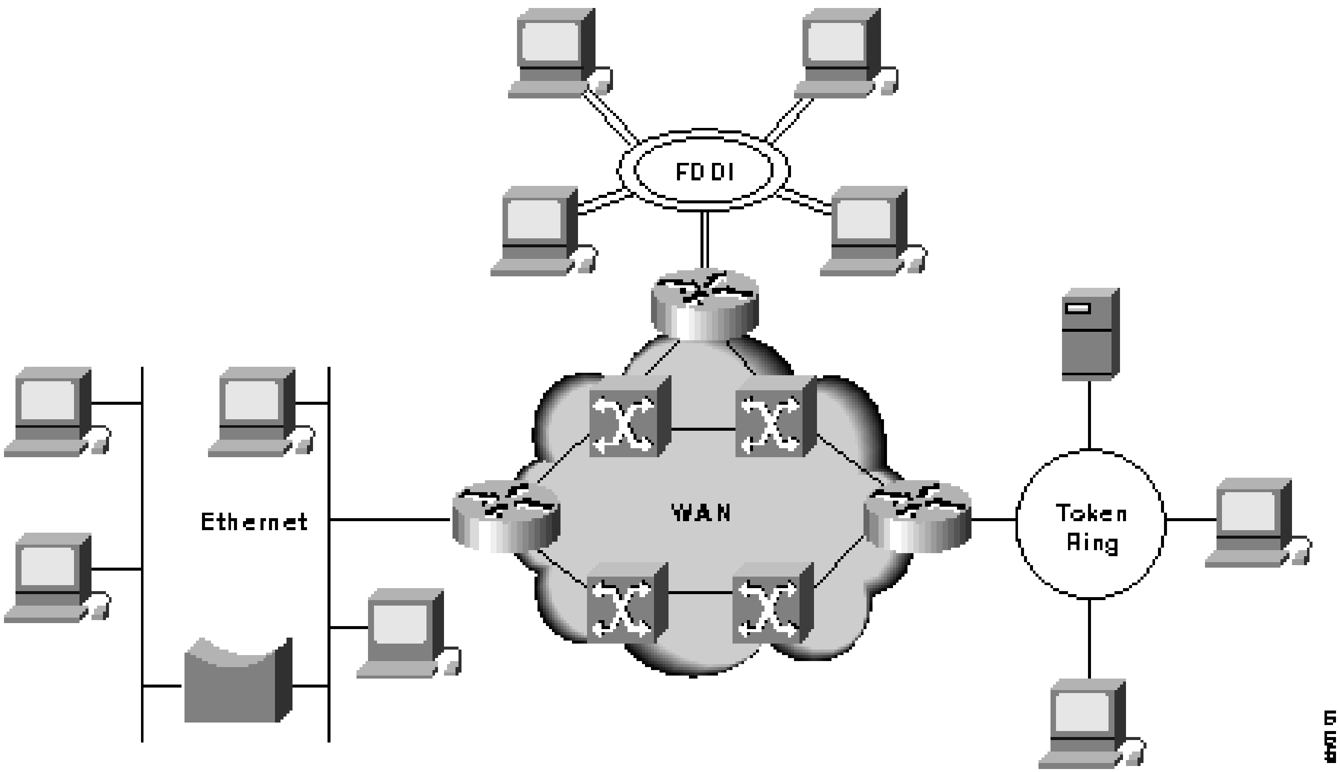
- Initial state:
  - actual position of the robot arms
- Actions:
  - motions of the robot arms
- Final state:
  - complete components assembly
- Cost-path:
  - time of execution

Robot Components Assembler

Routing in Computer Networks

- Initial state:
  - message at the sender location
- Operators:
  - message traveling across the network through nodes
- Final state:
  - message at the receiver location
- Cost-path:
  - time of transmission

Routing in Computer Networks



Basic Terminologies

- A state space is a graph composed of a set of nodes and a set of arcs, where each node is connected to another node by an arc
- In the state space, a node represents a state
- An arc represents applicable action

Basic Terminologies

- The node generation produces a node by applying an action to another node which has been previously generated

- The node expansion generates all children of a node by applying all applicable actions to that node

Basic Terminologies

- One or more nodes are named as start nodes (initial state)
- One or more nodes are named as goal nodes (final state)
- A solution is a sequence of actions that leads from a start node to a goal node

Basic Terminologies

- The path-cost is the sum of the arc costs on the solution path

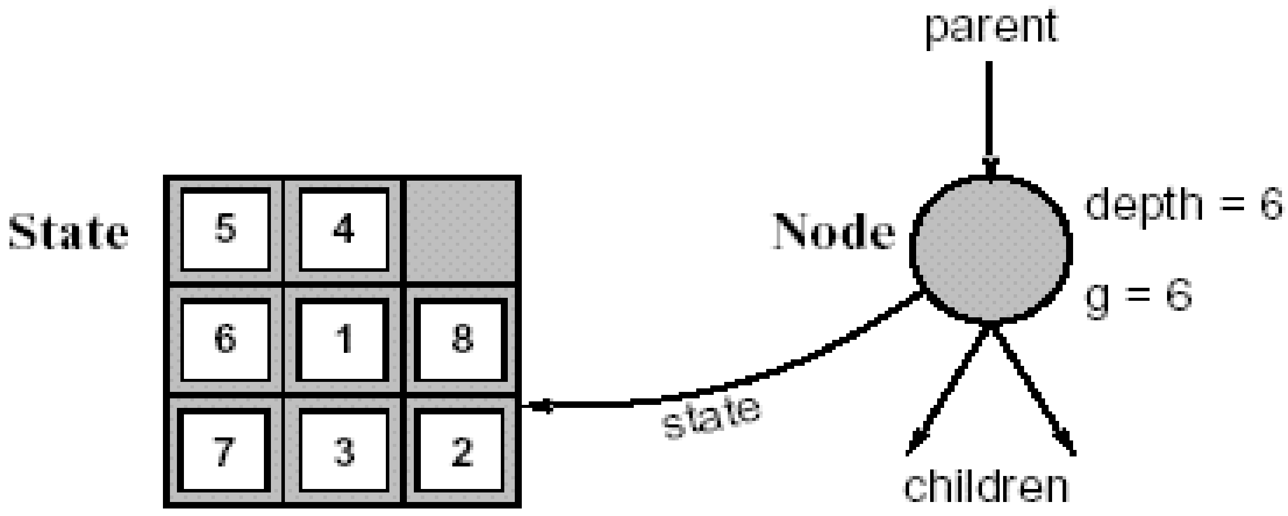
- State-space search is the process of searching for a solution by moving from the initial state to a final state through a state space

States

- A state is a representation of the physical world and it consists of the following components:
  - Initial state
  - Final state
  - A state transition is an action that changes the current state to another state

State vs. Node

- A node is a data structure constituting part of a search tree includes parent, children, depth, path cost *g(x)*
- States do not have parents, children, depth or path cost



States

- The size of a problem is usually described in terms of the number of states (or the size of the state space) that are possible
  - Tic-Tac-Toe has about 3^9 states
  - 8 - Puzzle has about 9! states
  - Chess has about 10^{120} states in a typical game

Assessment Criteria

- Four criteria for assessing search strategies:
  - Completeness: Does the strategy always find the solution if one exists?
  - Time complexity: number of nodes visited
  - Space complexity: maximum number of nodes in memory
  - Optimality: Does the strategy always find the least-cost solution (best one)?

Assessment Criteria

- Time and space complexity are measured in terms of:
  - b: maximum branching factor of the search tree
  - m: depth of the least-cost solution in search tree
  - d: maximum depth of search tree (may be ∞)