



مجموعة جامعة
الجميع الذكية
للتعليم القابضة



Master of Science in Artificial intelligent

Lecture 3

Solving problems by Searching

AI511

Foundations of Artificial Intelligence

Prepared by
Prof. Lamiaa Fattouh Ibrahim

www.midocean.ae

- **Solving problems by Searching**
 - **Types of agents**
 - **State Space Search**
 - **Example: Tic-Tac-Toe Game**
 - **Example: Mechanical Fault Diagnosing**
 - **How human beings think?**
 - **Heuristic Search**

- **Search**
 - **Search problem components**
 - **Example: Romania**
 - **State space & State Space Graph**
 - **Example: Vacuum world**
 - **Vacuum world state space graph**
 - **Example: The 8-puzzle**
 - **Example: Robot motion planning**
 - **Example: Tic- Tac-Toe**
 - **Example: Traveling Salesperson**
- **State Space Search Strategies**
- **Selecting Search Strategy**

Resources for this lecture

- This lecture covers the following chapters:
 - **Chapter 3 (Solving Problems by Searching; only sections 3.1, 3.2, & 3.3)** *from* Stuart J. Russell and Peter Norvig, "Artificial Intelligence: A Modern Approach," Third Edition (2010), by Pearson Education Inc.

... AND ...

- **Part II (pages 41 to 45) and Chapter 3 (only sections 3.0, 3.1, and 3.2)** *from* George F. Luger, "Artificial Intelligence: Structures and strategies for complex problem solving," Sixth edition (2009), Pearson Education Limited.

Solving Problems by SEARCHING



Reflex agent



Consider how the world IS

- Choose action based on current percept.
- Do not consider the future consequences of actions.

Planning agent



Consider how the world WOULD BE

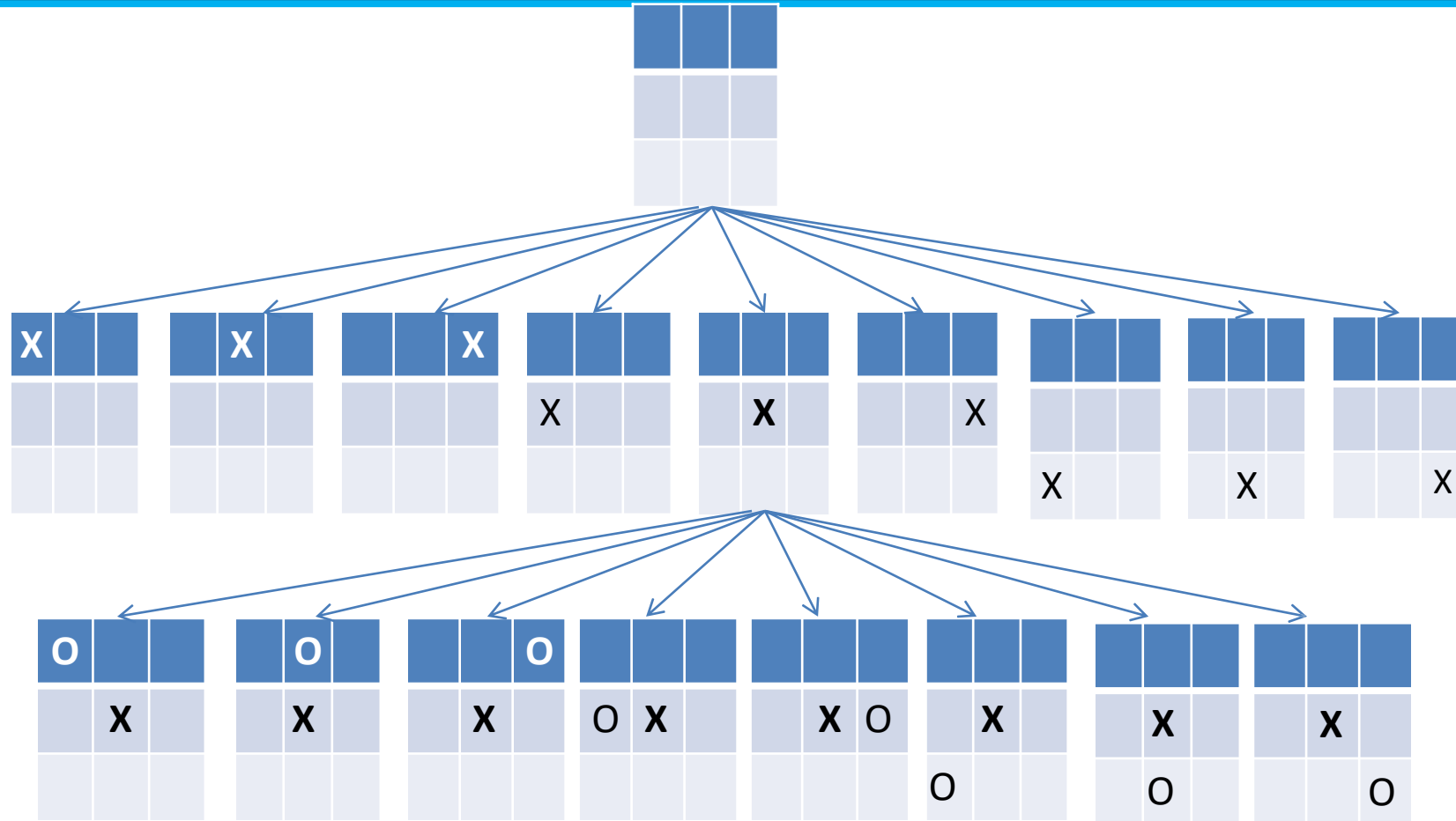
- Decisions based on (hypothesized) consequences of actions.
- Must have a model of how the world evolves in response to actions.
- Must formulate a goal.

State Space Search

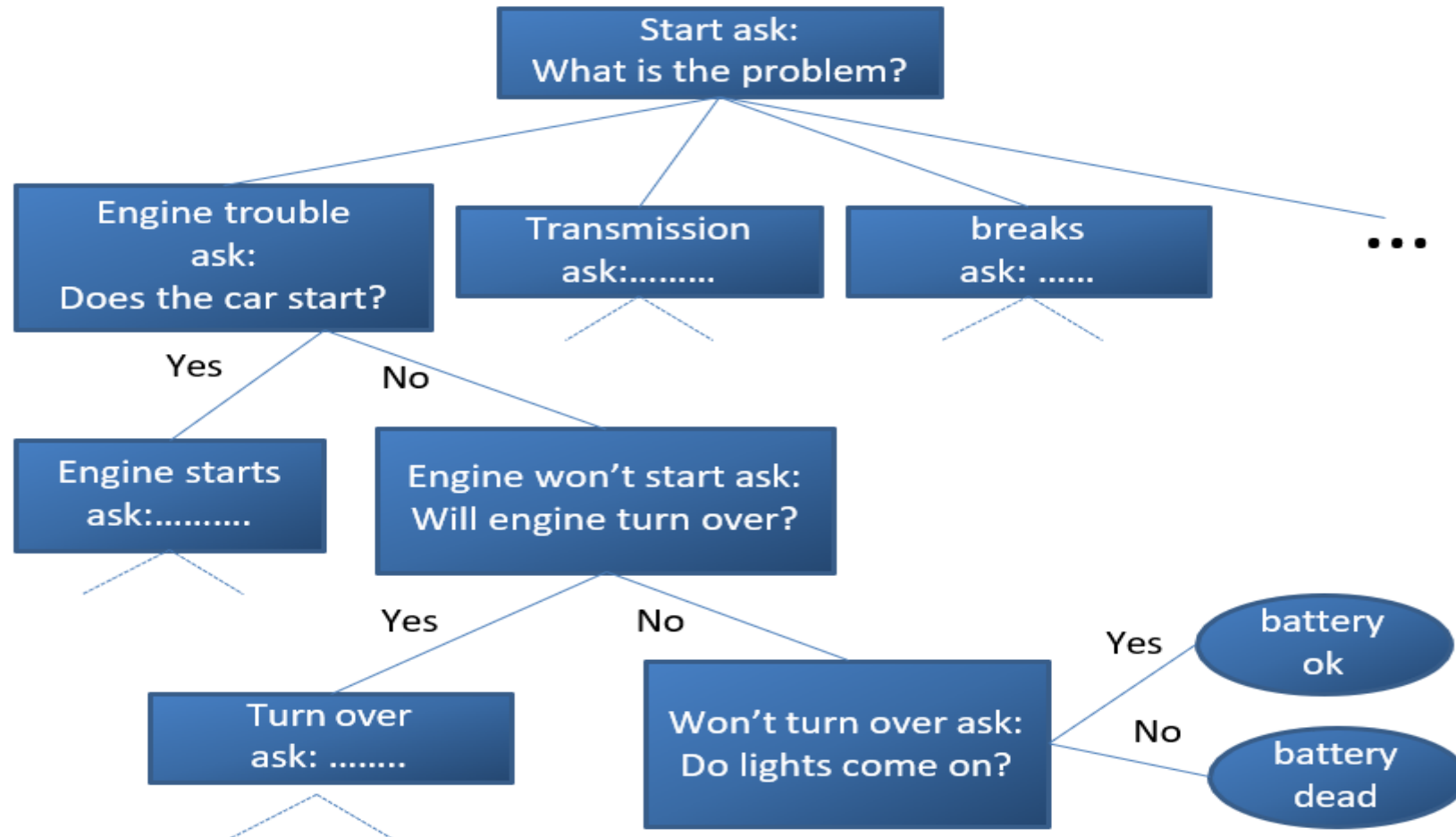
Problems are solved by searching among alternative choices.

- Humans consider a number of alternative strategies on their way to solving a problem.
 - A *Chess* player considers a number of alternative moves.
 - A *mathematician* chooses from a different strategies to find a proof for a theorem.
 - A *physician* evaluates a number of possible diagnoses.

Example: Tic-Tac-Toe Game

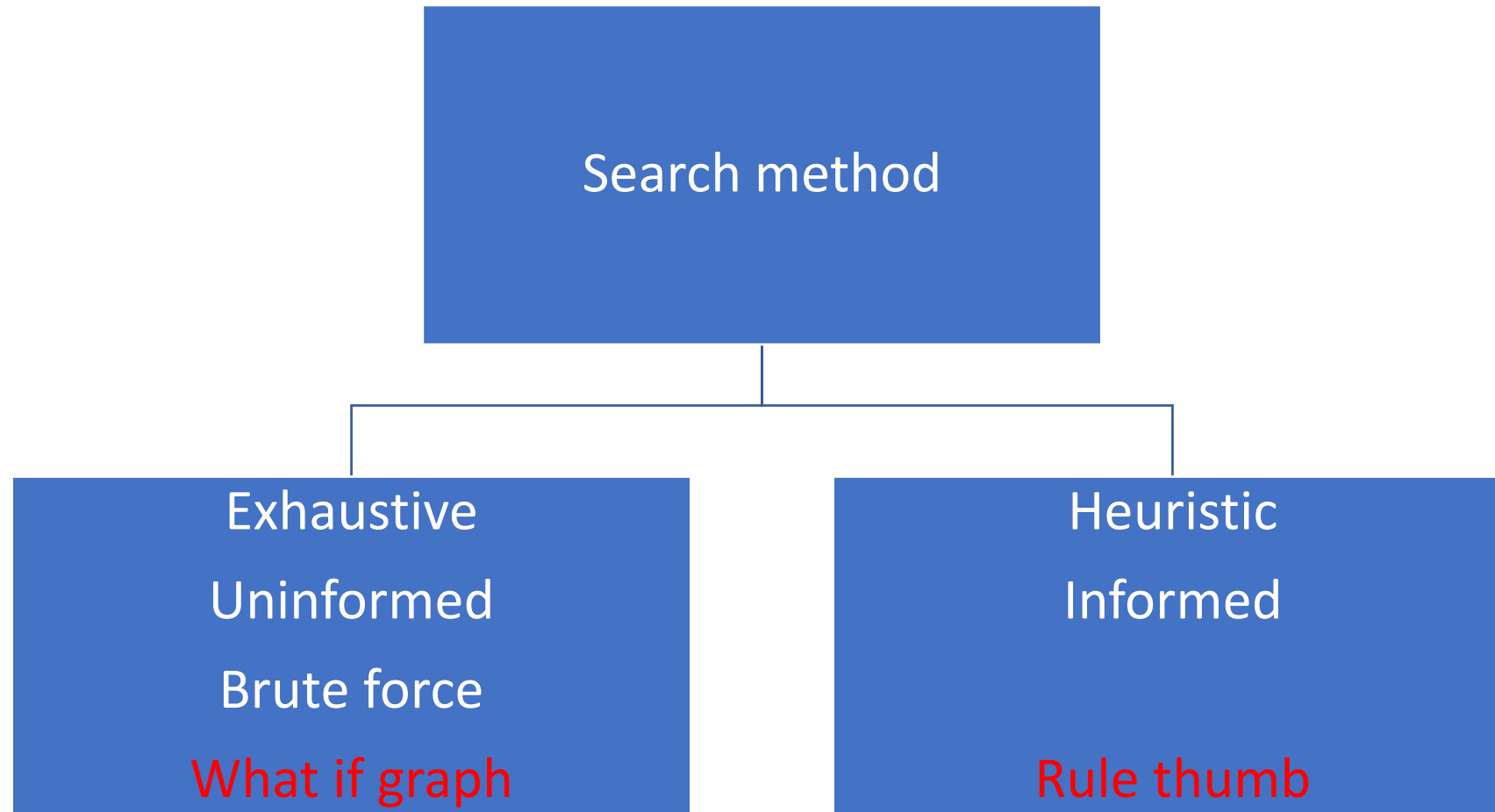


Example: Mechanical Fault Diagnosing



How human beings think?

- Human beings do not search the entire state space
- (*exhaustive search*).
- Only alternatives that experience has shown to be effective are explored.
- Human problem solving is based on judgmental rules that limit the exploration of search space to those portions of state
- space that seem somehow promising.
- These judgmental rules (rule of thumb) are known as “*heuristics*”.



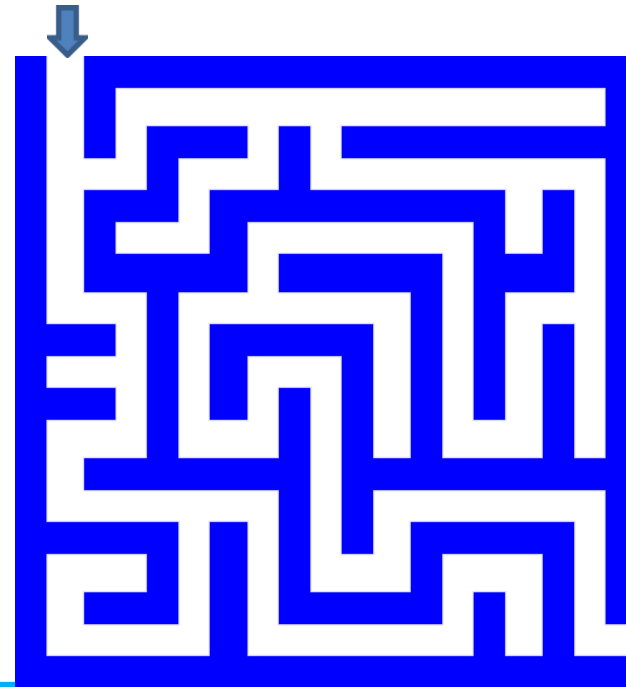
- A heuristic is a strategy for selectively exploring the search space.
- It guides the search along lines that have a high probability of success.
- It employs knowledge about the nature of a problem to find a solution.
- It does not guarantee an optimal solution to the problem but can come close most of the time.
- Human beings use a large number of heuristics in problem solving.

Properties of task environments	Descriptions
Fully observable (vs Partially Observable)	If an agent's sensors give it access to the complete state of the environment at each point in time. A task environment is effectively fully observable if the sensors detect all aspects that are <i>relevant</i> to the choice of action.
Deterministic (vs Stochastic)	If the next state of the environment is completely determined by the current state and the action executed by the agent, then we say the environment is deterministic; otherwise, it is stochastic.
Discrete (vs Continuous)	Chess also has a discrete set of percepts and actions. Taxi driving is a continuous-state and continuous-time problem: the speed and location of the taxi and of the other vehicles sweep through a range of continuous values and do so smoothly over time

Properties of task environments	Descriptions
Episodic (vs Sequential)	In an episodic task environment, the agent's experience is divided into atomic episodes. In each episode the agent receives a percept and then performs a single action. Crucially, the next episode does not depend on the actions taken in previous episodes.
Known vs Unknown	In a known environment, the outcomes (or outcome probabilities if the environment is stochastic) for all actions are given. Obviously, if the environment is unknown, the agent will have to learn how it works in order to make good decisions.

We will consider the problem of designing **goal-based agents** in **fully observable, deterministic, discrete, known** environments.

Start State



Goal State

We will consider the problem of designing **goal-based agents** in **fully observable, deterministic, discrete, known** environments.

The agent must find a *sequence of actions* that reaches the goal. The **performance measure** is defined by:

- (a) reaching the goal, and ...
- (b) how “expensive” the path to the goal is.

Initial state

Actions

Transition model

What state results from performing a given action in a given state?

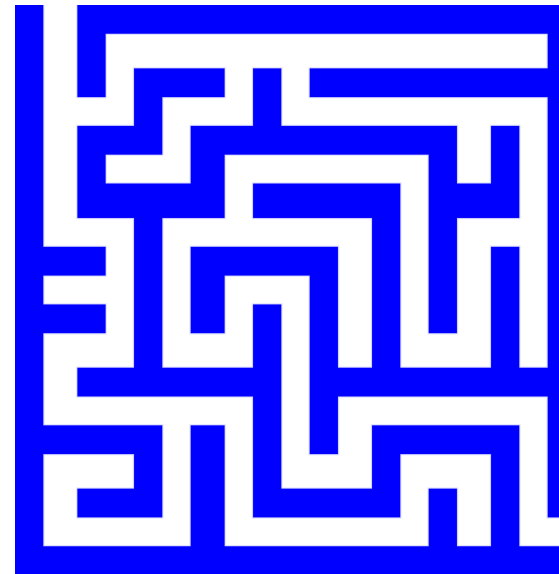
Goal state

Solution Path

Path cost

Assume that it is a sum of nonnegative *step costs*

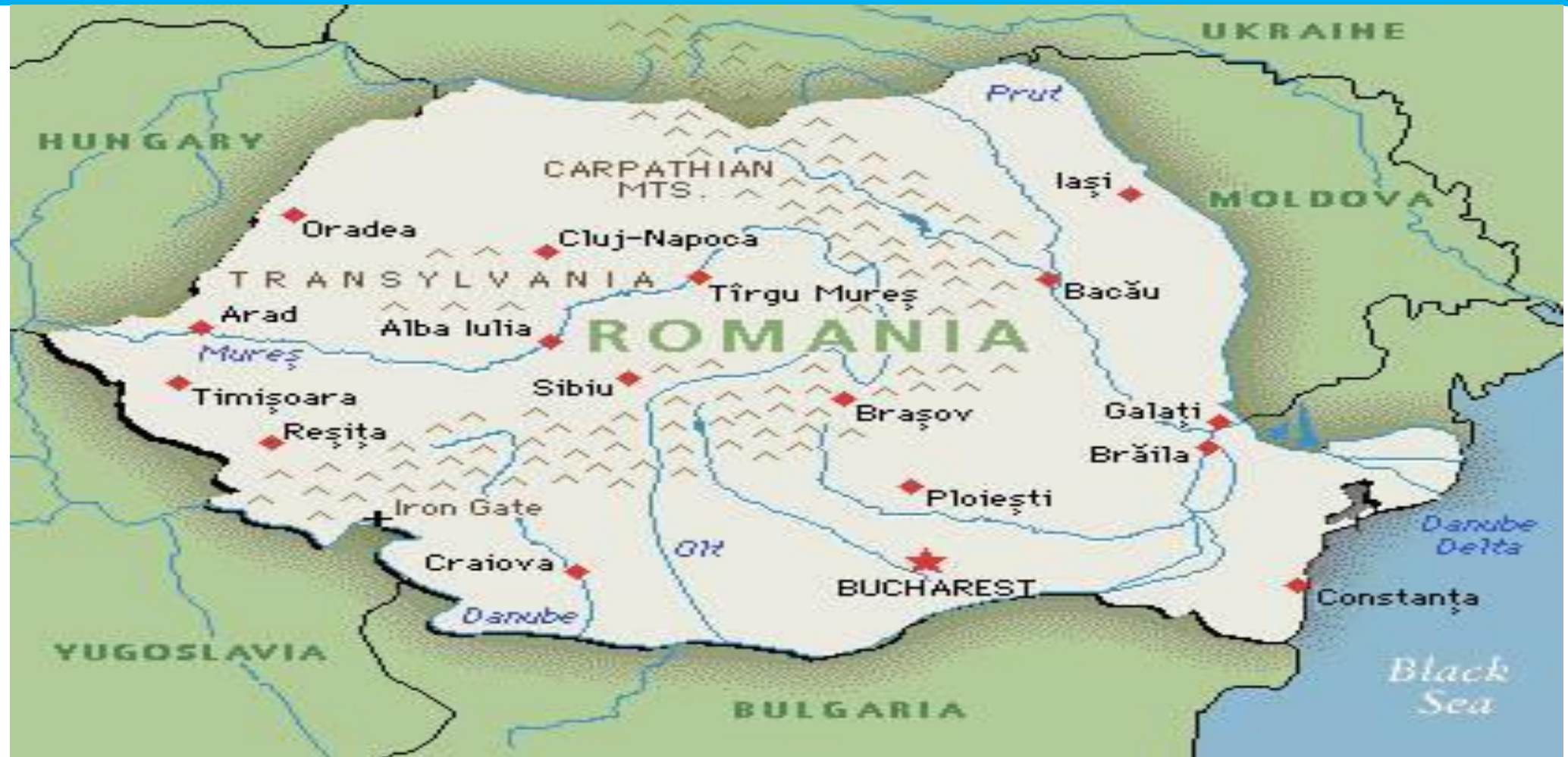
Initial
State



Goal
State

The **optimal solution** is the sequence of actions that gives the *lowest* path cost for reaching the goal.

Lecture 3



Example: Romania



- On vacation in Romania; currently in Arad.
- Flight leaves tomorrow from Bucharest.

Initial state

- Arad

Actions

- Go from one city to another

Transition model

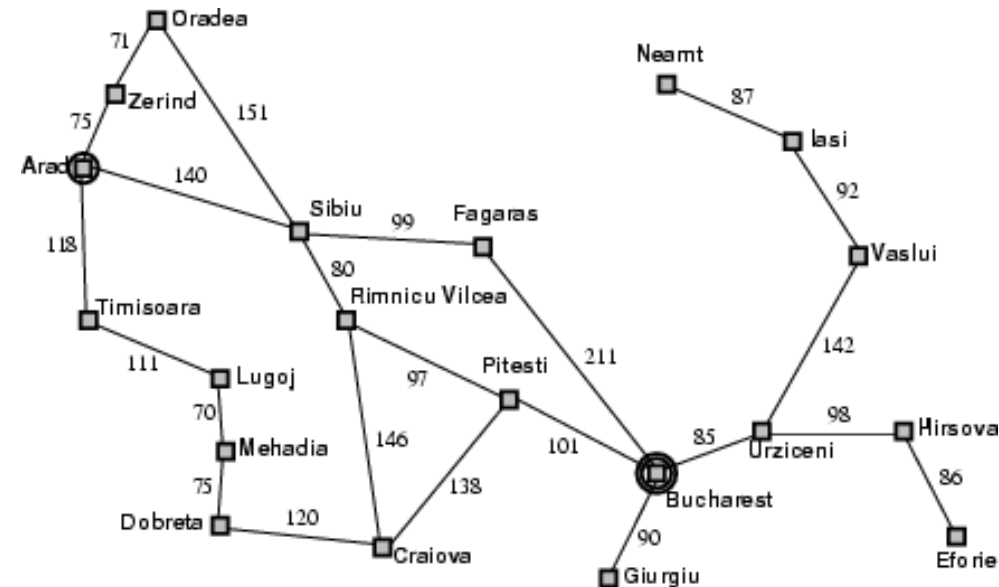
- If you go from city A to city B, you end up in city B

Goal state

- Bucharest

Path cost

- Sum of edge costs (*total distance traveled*)

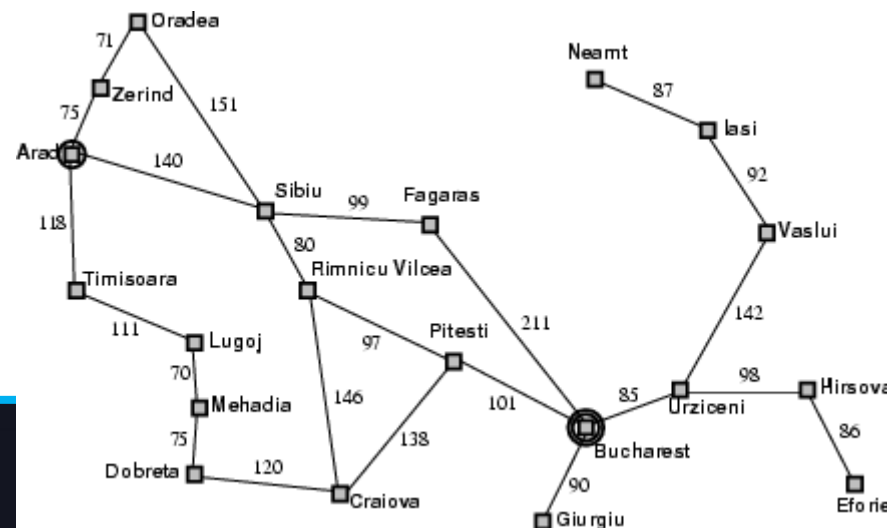


State Space Representation of Problems

The initial state, actions, and transition model define the **state space** of the problem;

- The set of all states reachable from initial state by any sequence of actions.
- Can be represented as a **directed graph** where the nodes are states and links between nodes are actions.

What is the state space for the Romania problem?



An AI problem can be represented as a *state space graph*.

A **graph** is a set of *nodes* and *links* that connect them.

Graph theory:

- Labeled graph.
- Directed graph.
- Path.
- Rooted graph.
- Tree.
- Parent.
- Child.
- Sibling.
- Ancestor.
- Descendant.

Graph Theory, Königsberg Bridges Problem, & Euler Tour ..

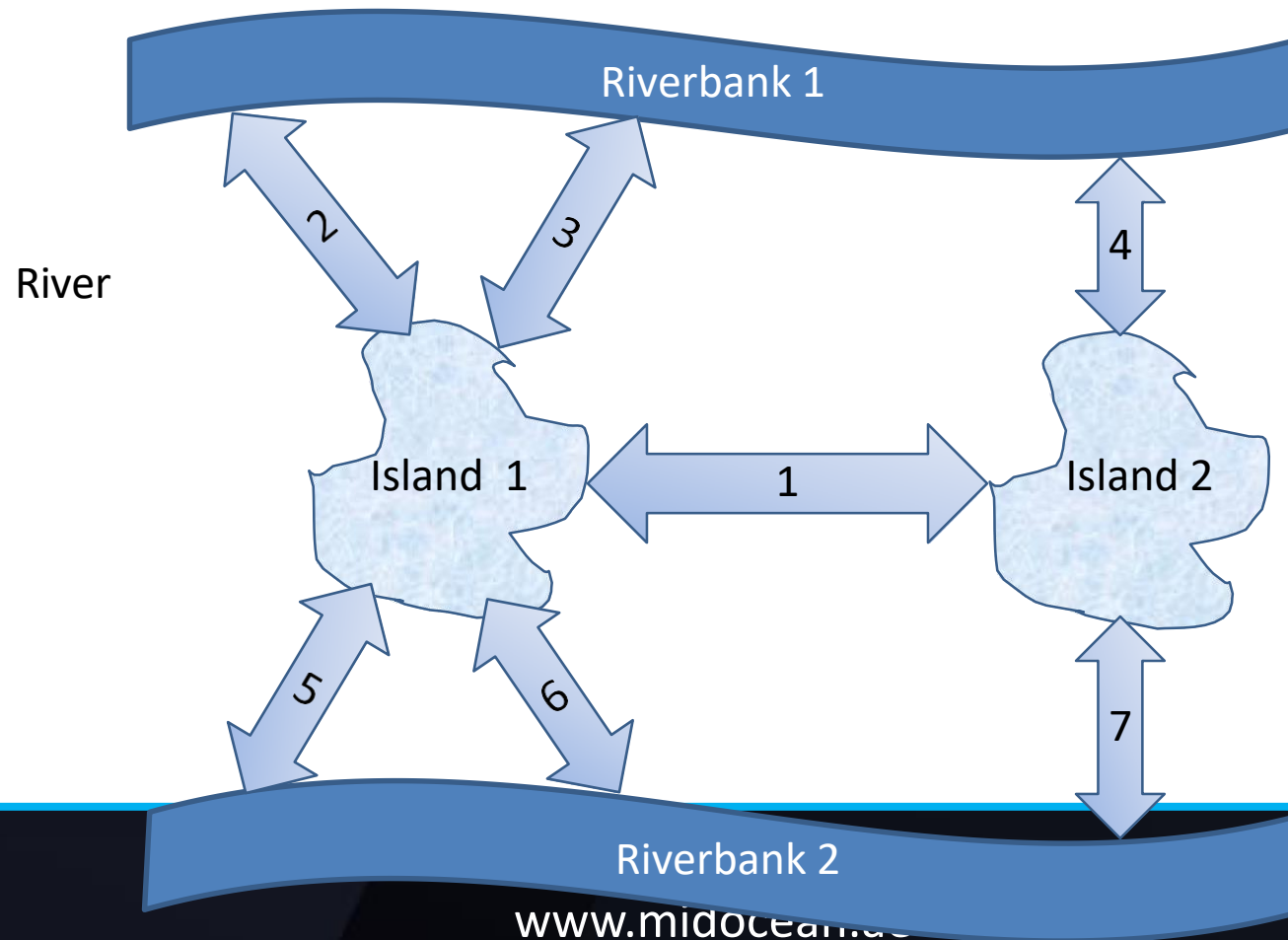
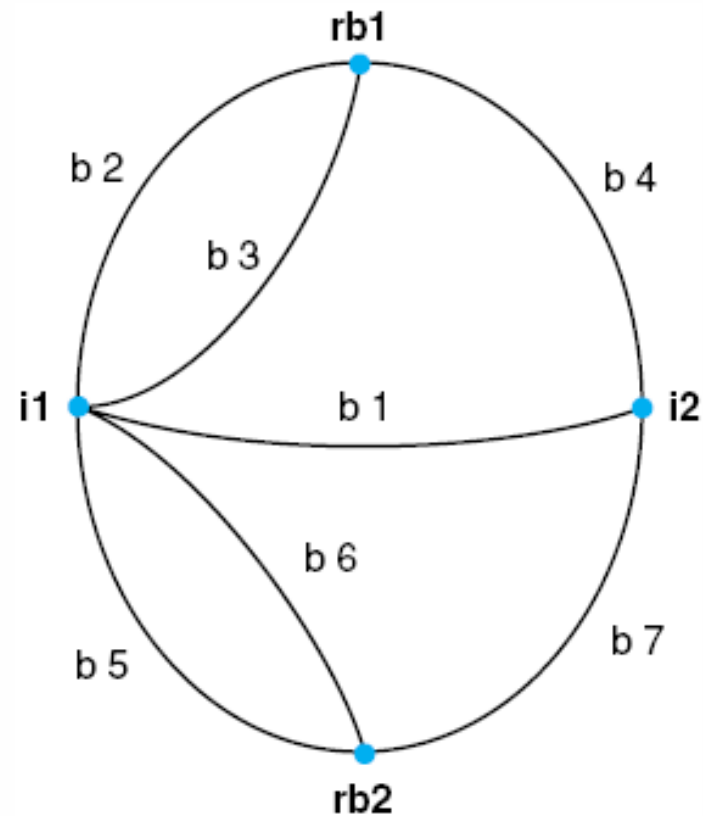


Figure 3.2: Graph of the Königsberg bridge system.



Graph Theory

Structures for State Space search

A graph is a set of nodes and arcs that connect them.

A **Labeled graph** has one or more descriptors (labels) attached to each node that distinguish that node from any other node in the graph.

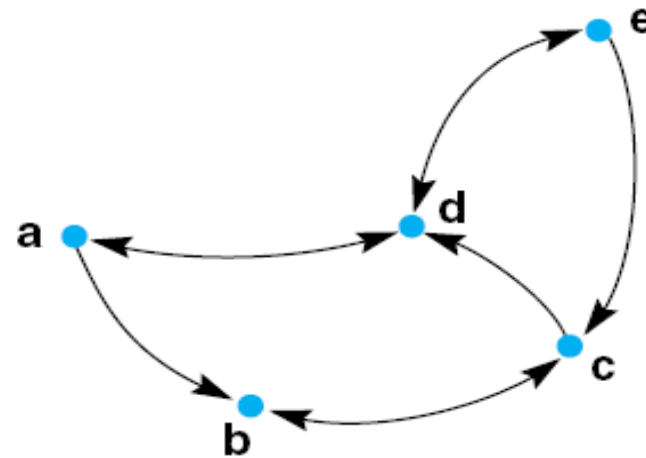
In **State Space graph**, these descriptors identify states in a problem-solving process.

The arcs of a graph may also be labeled. Arc labels are used to indicate that an arc represents a named relationship (as in a semantic network) or to attach weights to arcs.

If there are different arcs between the same two nodes, these can also be distinguished through labeling.

- A graph is **directed** if arcs have an associated directionality.
- Arcs that can be crossed in either direction may have two arrows attached but more often have no direction indicators at all.
- A **path** through a graph connects a sequence of nodes through successive arcs.
- A **rooted** graph has a unique node, called the **root**, such that there is a path from the root to all nodes within the graph.
- The state space graphs for games are usually rooted graphs with the start of the game as the root.
- A **tree** is a graph in which two nodes have at most one path between them, therefore it is impossible for a path to loop or cycle continuously through a sequence of nodes.
- Trees often have roots.
- For rooted trees or graphs, relationships between nodes include **parent**, **child**, and **sibling**.
- Ancestor comes before a descendant in some path of a directed graph.

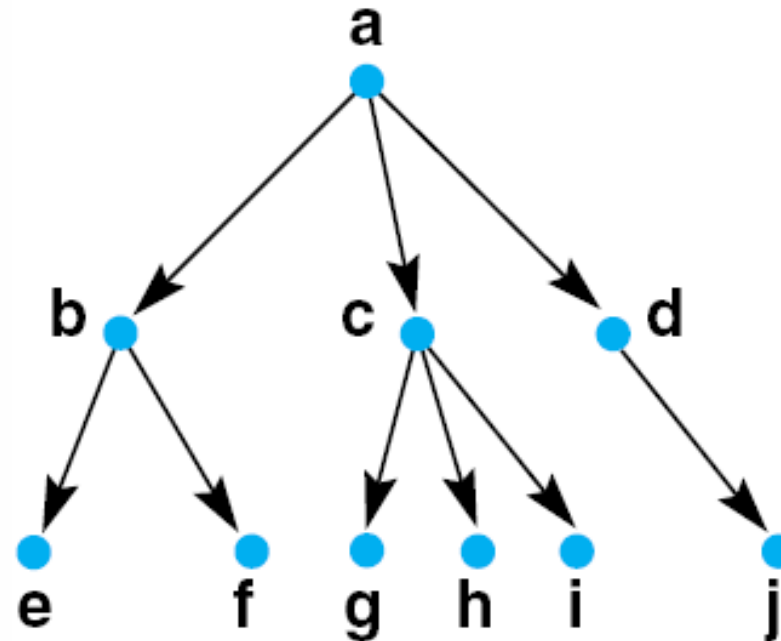
Figure 3.3: A labeled directed graph.



Nodes = {a,b,c,d,e}

Arcs = {(a,b),(a,d),(b,c),(c,b),(c,d),(d,a),(d,e),(e,c),(e,d)}

Figure 3.4: A rooted tree, example family relationships.

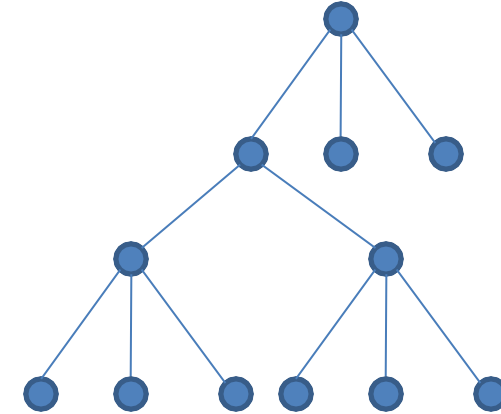


b is a parent of nodes **e** and **f** (which are, therefore, children of **b** and sibling of each other).

Nodes **a** and **c** are ancestors of states **g**, **h**, and **i**, and **g**, **h**, and **i** are descendants of **a** and **c**.

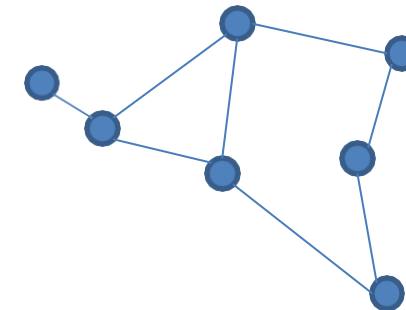
• Trees:

- Consists of nodes connected by links (or edges).
- every node has a single parent.
- The root has no parents.
- No cycles exist.



• Graphs:

- Consists of nodes connected by links (or edges).
- Nodes have any number of parents.
- Cycles may exist



- The of a search algorithm is to find a solution path through such a problem space.
- The actual generation of new states along the path is done by applying operators such as legal moves in a game.
- Search algorithms must keep track of the **paths from a start to a goal node**, because these paths contain the series of operations that lead to the problem solution.
- A goal may describe:
 - 1- A state, such as a winning board in a simple game.
 - 2- Some property of the solution path itself.

State Space

Lecture 3

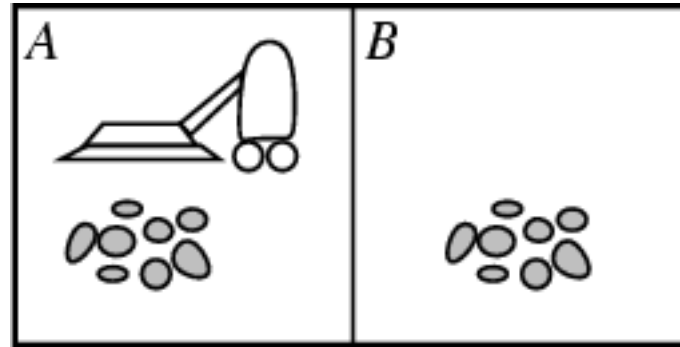
State space search is the process of finding a solution path from the start state to a goal

A state space is represented by four-tuple $[N, A, S, GD]$.

- **N**, is the set of nodes or states of the graph. These correspond to the states in the problem-solving process.
- **A**, is the set of arcs between nodes. These correspond to the steps in a problem-solving process.
- **S**, a nonempty subset of N , contains the **start-state (s)** of the problem.
- **GD**, a nonempty subset of N , contains the **goal-state(s)** of the problem. The states in GD are described using either:
 - A measurable property of the states encountered in the search.
 - A property of the solution path developed in the search, for example, the transition costs for the arcs of the path.

*A **solution path** is a path through this graph from a node in S to a node in GD .*

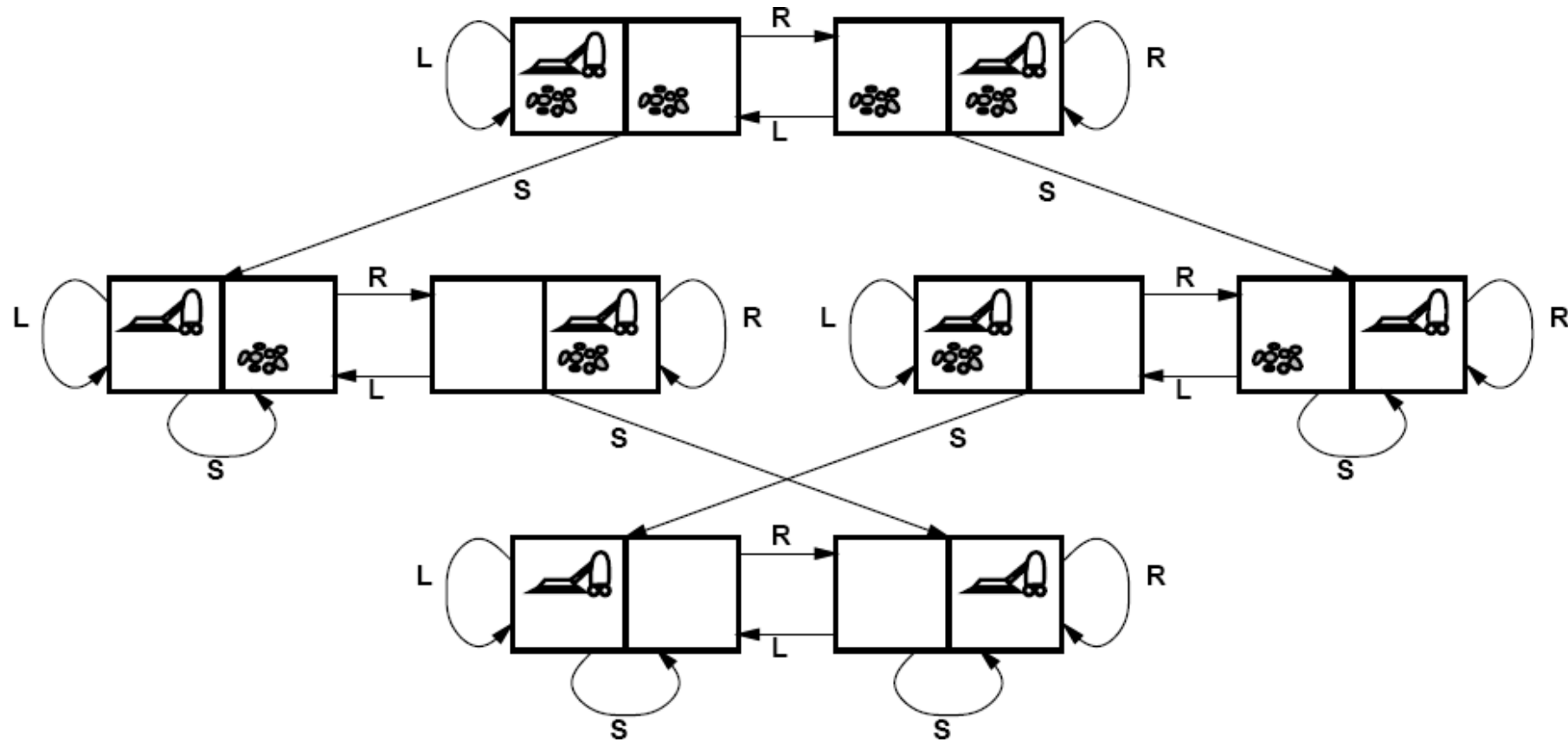
Example: Vacuum World



- **States:**
 - Agent location and dirt location
 - How many possible states?
 - What if there are n possible locations?
 - The size of the state space grows exponentially with the “size” of the world!
- **Actions:**
 - Left, right, suck.
- **Transition model ..**

Example: Vacuum World State Space Graph

Lecture 3



Example: The 8-puzzle

- **States**
 - Locations of tiles
 - 8-puzzle: 181,440 states ($9!/2$)
 - 15-puzzle: ~10 trillion states
 - 24-puzzle: $\sim 10^{25}$ states
- **Actions**
 - Move blank left, right, up, down
- **Path cost**
 - 1 per move

7	2	4
5		6
8	3	1

Start State

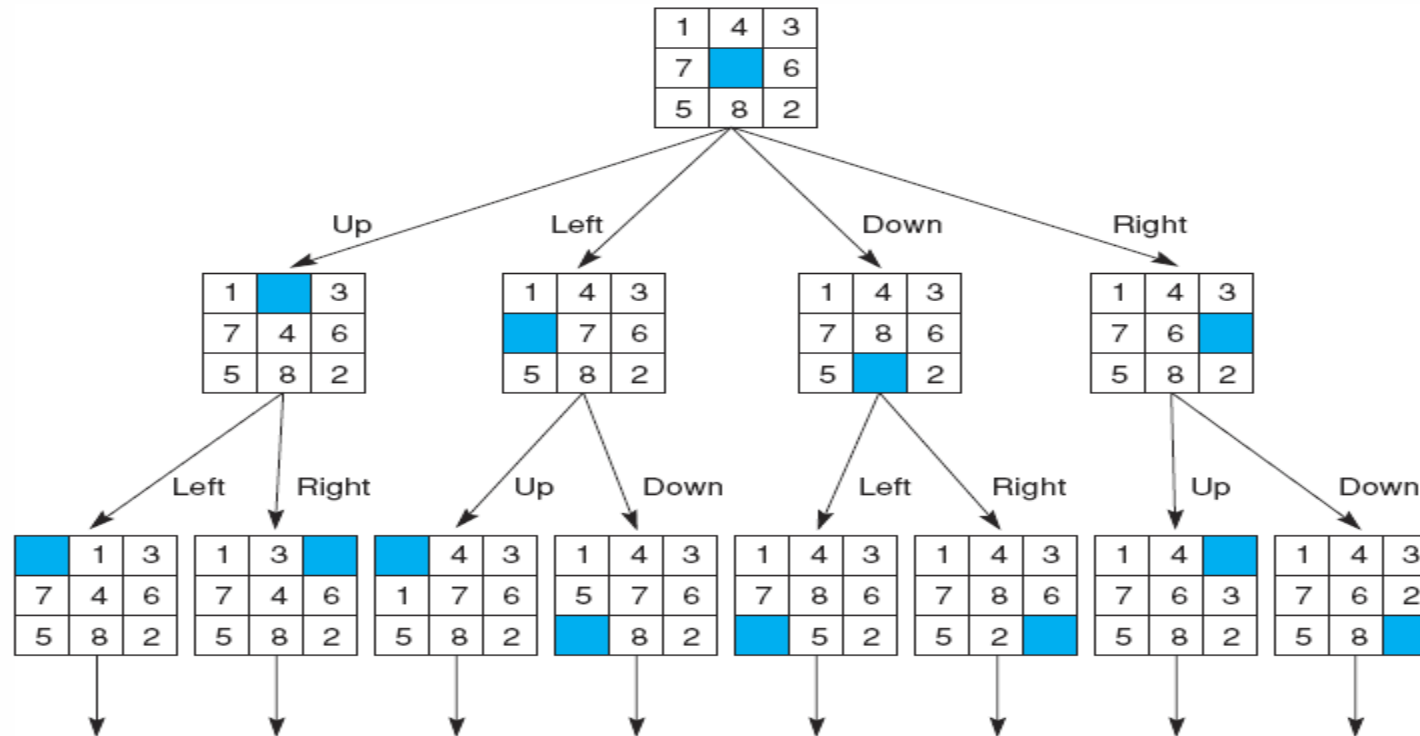
	1	2
3	4	5
6	7	8

Goal State

Example: Tic-Tac-Toe

- Nodes(N): all the different configuration of Xs and Os that the game can have.
- Arcs (A): generated by legal moves by placing an X or an O in unused location.
- Start state (S): an empty board.
- Goal states (GD): a board state having three Xs in a row, column, or diagonal.
- The arcs are directed, then no cycles in the state space, directed acyclic graph (DAG).
- Complexity: 9! Different paths can be generated.

Fig 3.8 State space of the 8-puzzle generated by “move blank” operations



Example: Traveling Sales person

A sales person has five cities to visit and then must return home.

- Nodes(N): represent 5 cities.
- Arcs(A): labeled with weight indicating the cost of traveling between connected cities.
- Start state(S): a home city.
- Goal states(GD): an entire path contains a complete circuit with minimum cost.

Fig 3.9 An instance of the travelling salesperson problem. Nodes of the graph represent cities. Each arc is labelled with the distance between two cities. The salesperson lives in city A and will return there.

In the traveling salesperson problem, search terminates when the 'shortest' path is found through all nodes of the graph.

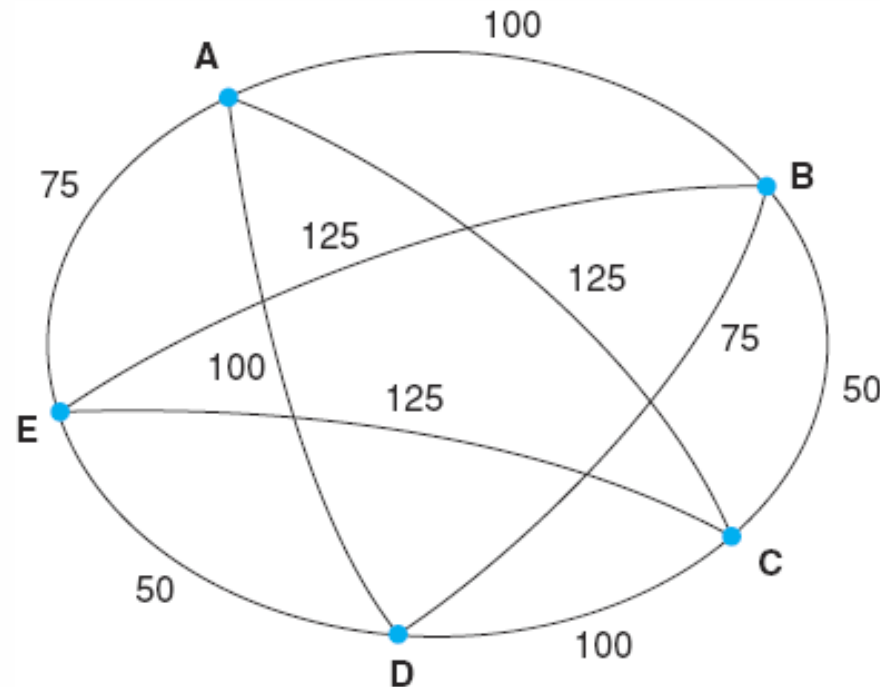
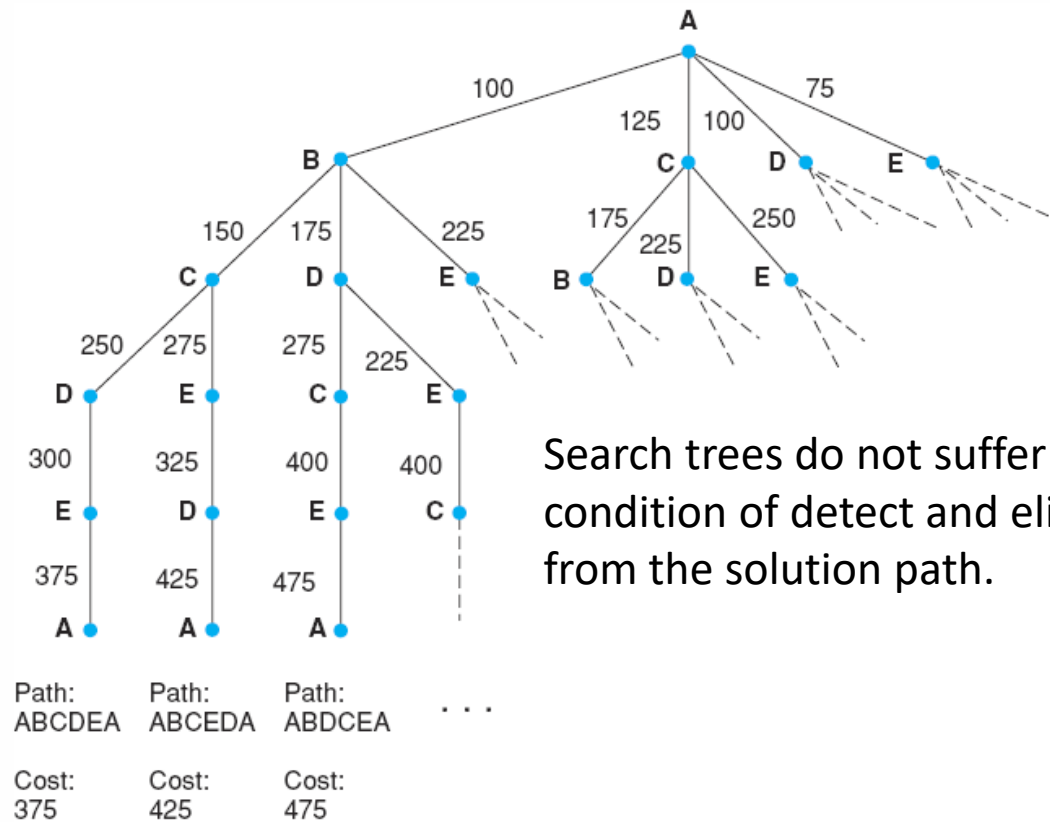
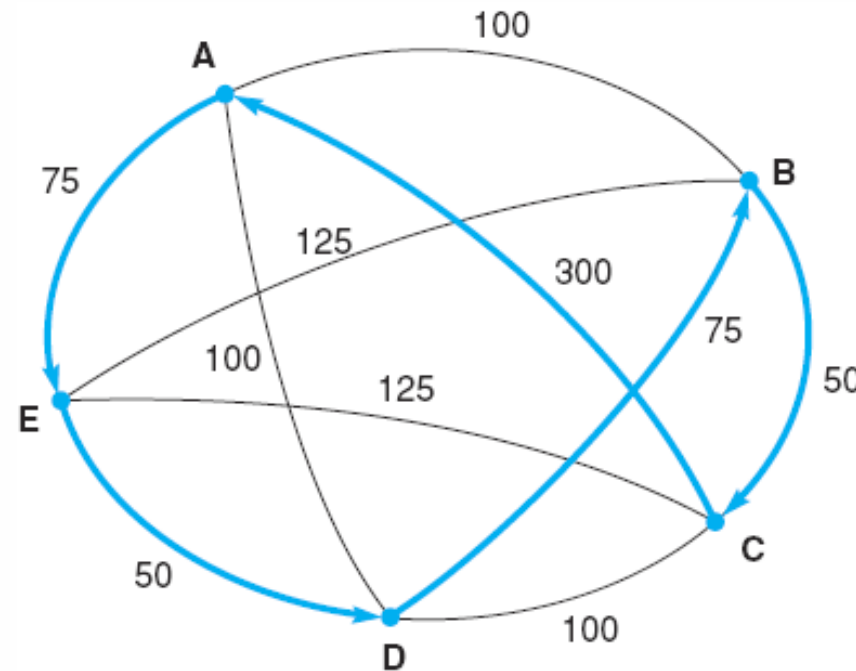


Fig 3.10 Search for the travelling salesperson problem. Each arc is marked with the total weight of all paths from the start node (A) to its endpoint.



Search trees do not suffer from the condition of detect and eliminate loops from the solution path.

Fig 3.11 An instance of the travelling salesperson problem with the nearest neighbour path in bold. Note this path (A, E, D, B, C, A), at a cost of 550, is not the shortest path. The comparatively high cost of arc (C, A) defeated the heuristic.



There are two distinct ways for searching a state space graph:

❖ **Data-Driven Search:** (Forward chaining)

Start searching from the given data of a problem instance toward a goal.

❖ **Goal-Driven Search:** (Backward chaining)

Start searching from a goal state to facts or data of the given problem.

State Space Search Strategies ..

Selecting Search Strategy

Data-Driven Search is suggested if:

- The data are given in the initial problem statement.
- There are few ways to use the given facts.
- There are large number of potential goals.
- It is difficult to form a goal or hypothesis.

Goal- Driven Search is appropriate if:

- A goal is given in the problem statement or can easily formulated.
- There are large number of rules to produce a new facts.
- Problem data are not given but acquired by the problem solver.

Fig 3.12 State space in which goal-directed search effectively prunes extraneous search paths.

1. Find what states could have produced the goal
2. Consider those states as *subgoals*
3. Repeat for each subgoal until an initial state is found

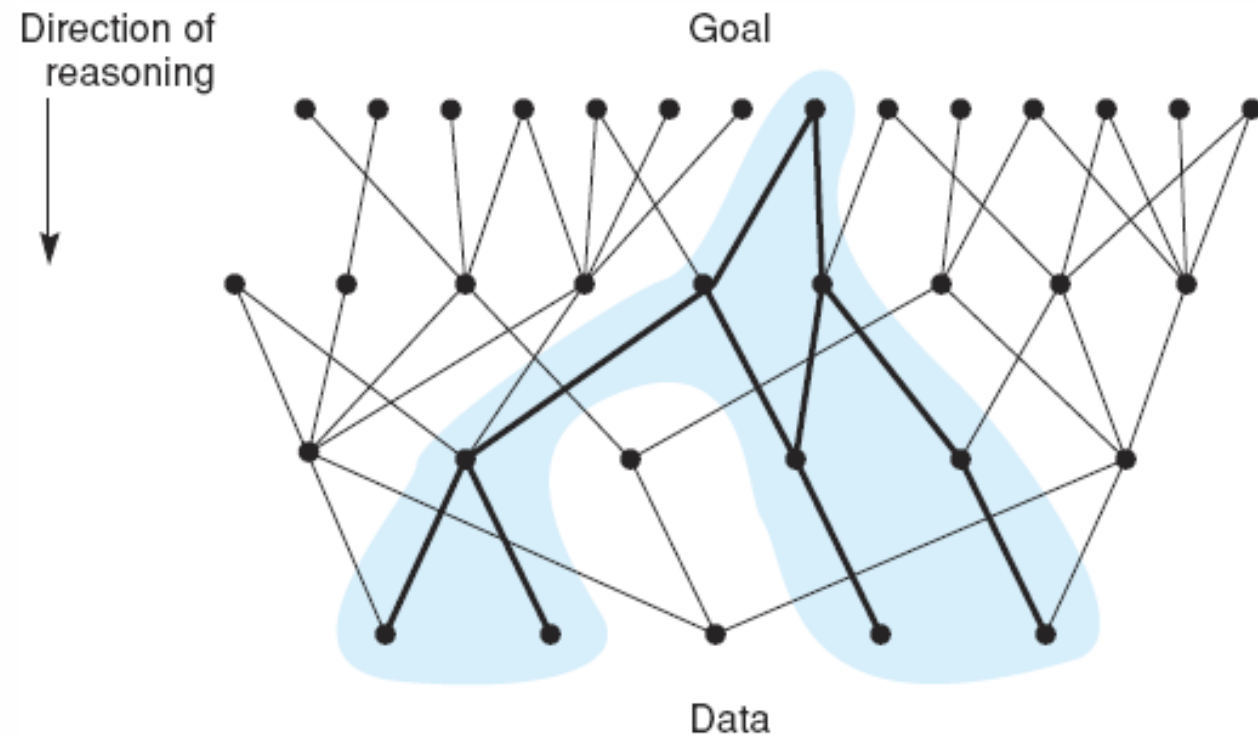
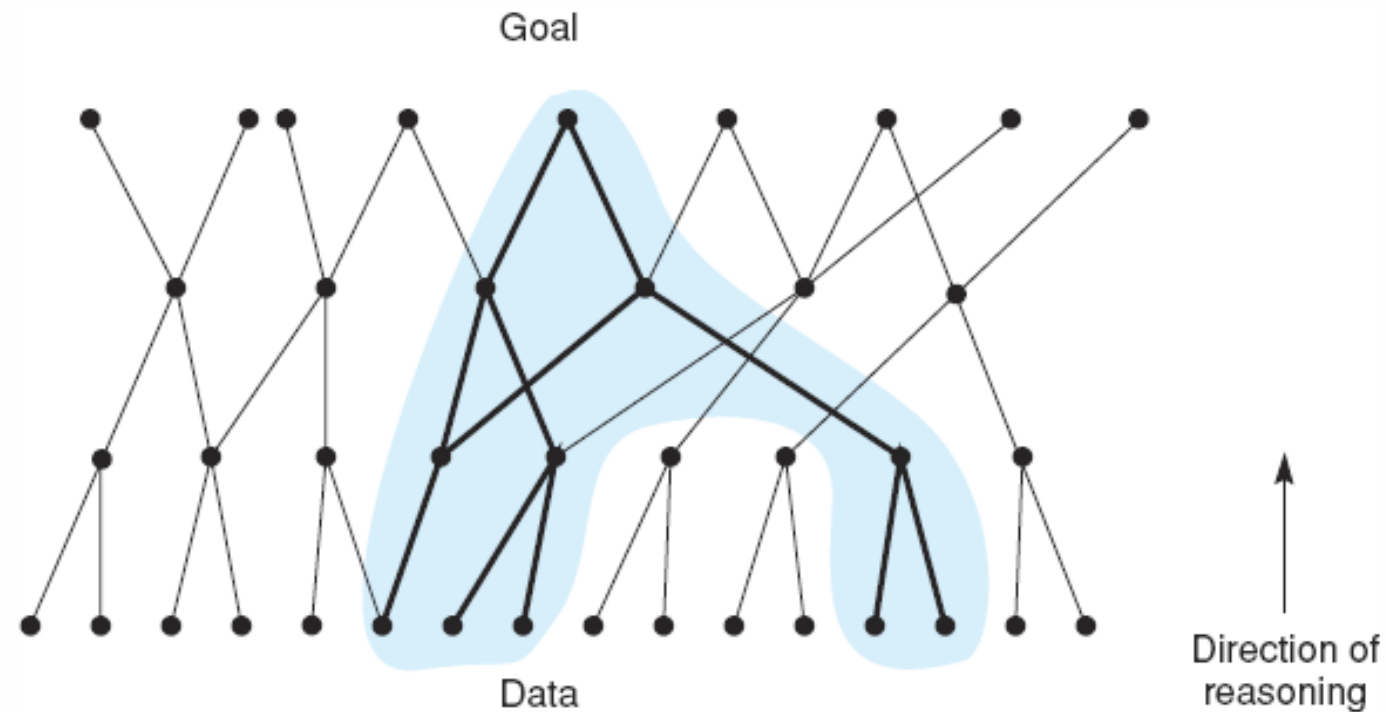


Fig 3.13 State space in which data-directed search prunes irrelevant data and their consequents and determines one of a number of possible goals.



Solving problems by Searching

- **Types of agents.**
- **State Space Search.**
- **Heuristic Search.**
- **Graph Theory.**
- **Strategies for Space State Search.**
- **Using Space State to Represent Reasoning.**
- **Examples.**



مجموعة جامعة
الجميع الذكية
للتعليم القابضة

MIDOCEAN
UNIVERSITY
BY EVERYONE'S SMART UNIVERSITY



Thank You

