# MATERI PERKULIAHAN MINGGU ke-3

Masalah, Ruang Keadaan, dan Pencarian

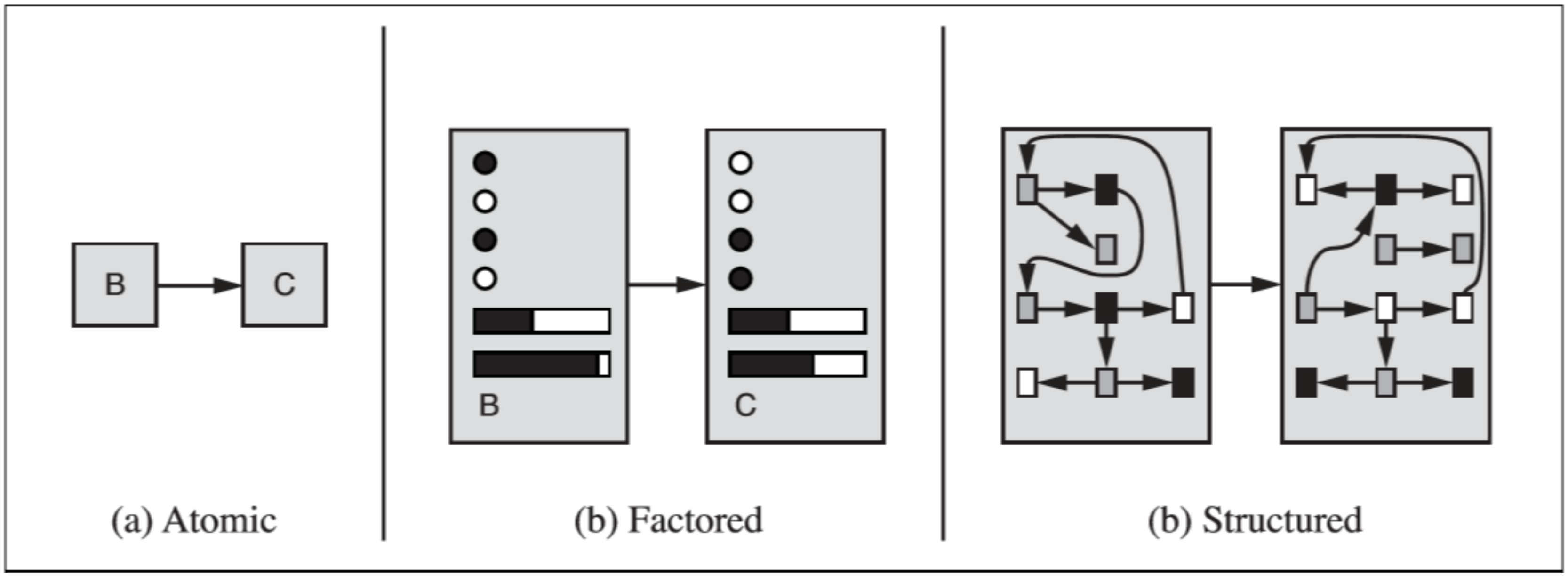
*Sri Supatmi, S.Kom., M.T., D.Sc.*

# Tujuan

- Memahami konsep algoritma ***Searching***

# AGENT PROGRAM

# Kategori STATE



# PROBLEM SOLVING

*A problem **well defined** is a problem half solved*

~ John Dewey

# Well-defined Problem

- *Initial State*
- *Path Cost*
- *Actions*
- *Goal Test*

# PROBLEM

# PROBLEM

| ***Toy** Problem* | ***Real World** Problem* |
| --- | --- |
| *Vacuum World*<br>*8 Puzzle* | *TSP*<br>*Route Finding* |

# Vacuum World

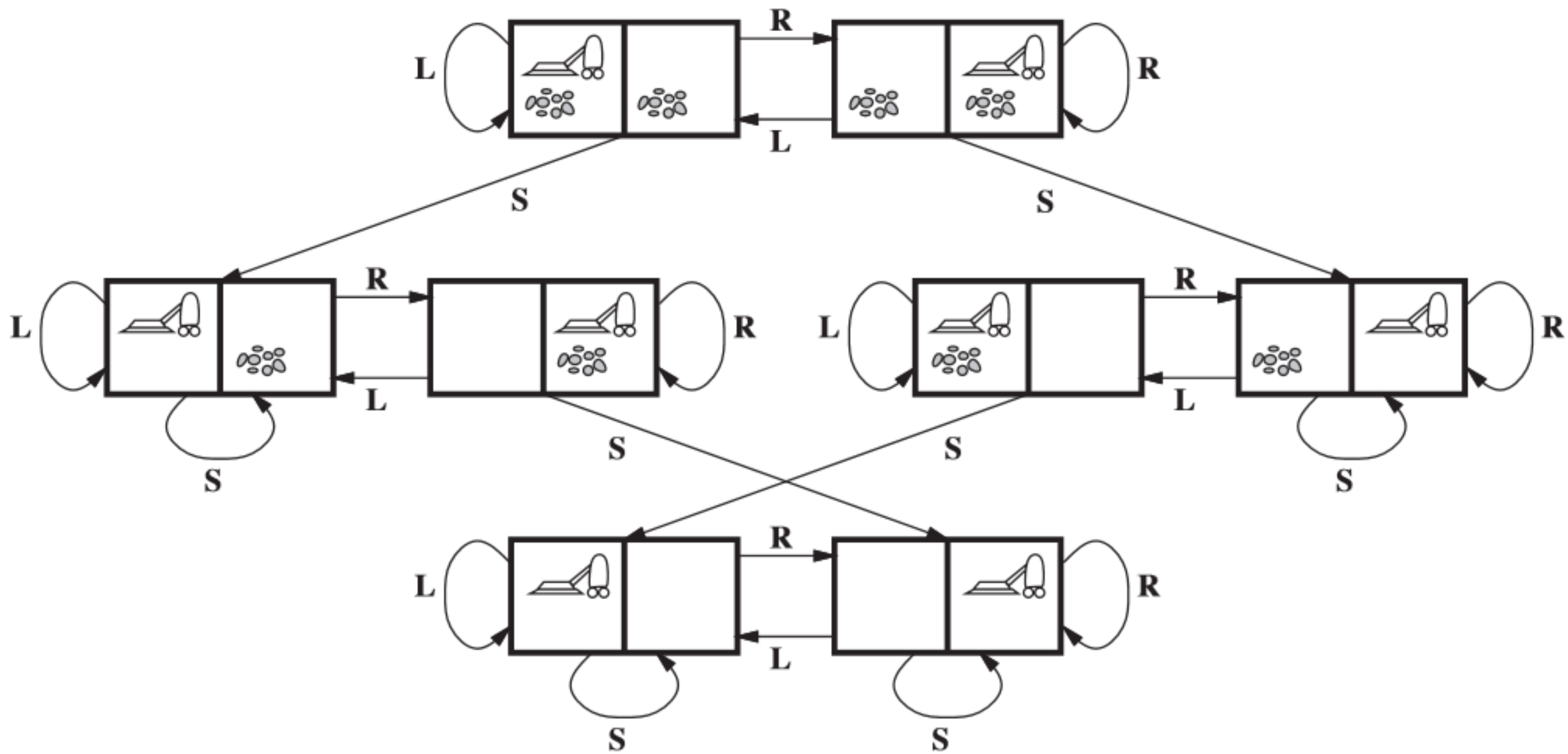



**Figure 3.3** The state space for the vacuum world. Links denote actions: L = *Left*, R = *Right*, S = *Suck*.

# Vacuum World

| KRITERIA | KETERANGAN |
|---|---|
| **Initial State** | Salah satu state |
| **Path Cost** | 1 untuk setiap langkah |
| **Action** | Left, Right, Suck |
| **Goal Test** | Cek apakah semua lokasi bersih |

# 8 Puzzle

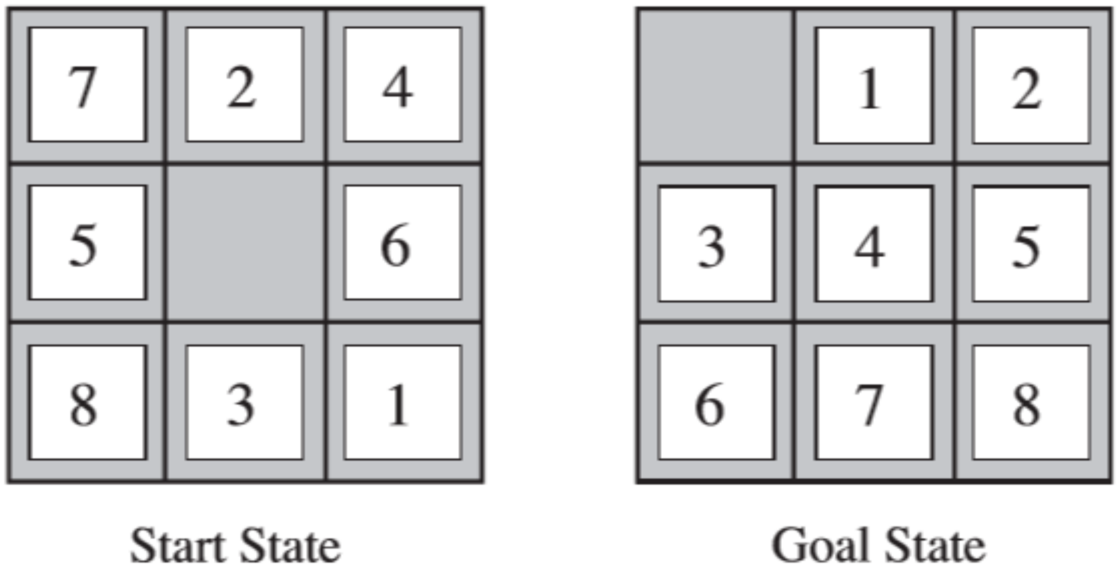



**Figure 3.4** A typical instance of the 8-puzzle.

# 8 Puzzle

| KRITERIA | KETERANGAN |
|---|---|
| **Initial State** | Salah satu state |
| **Path Cost** | 1 untuk setiap langkah |
| **Action** | Left, Right, Up, Down |
| **Goal Test** | Cek apakah setiap state ada di tempat yang seharusnya (*Goal State*) |

# 8 Queens

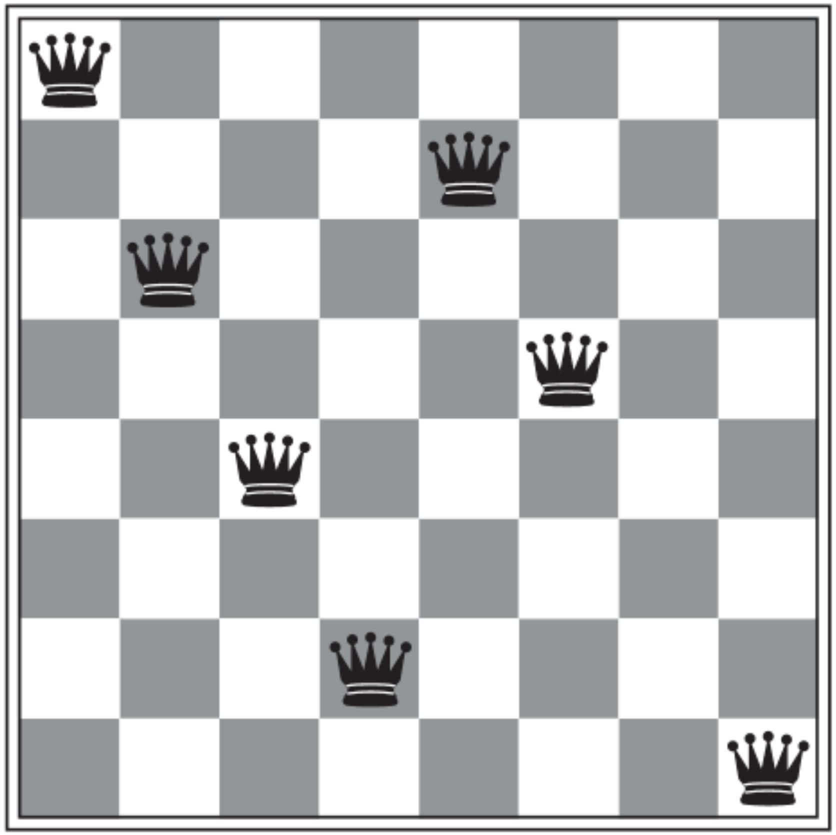


**Figure 3.5** Almost a solution to the 8-queens problem.

# 8 Queens

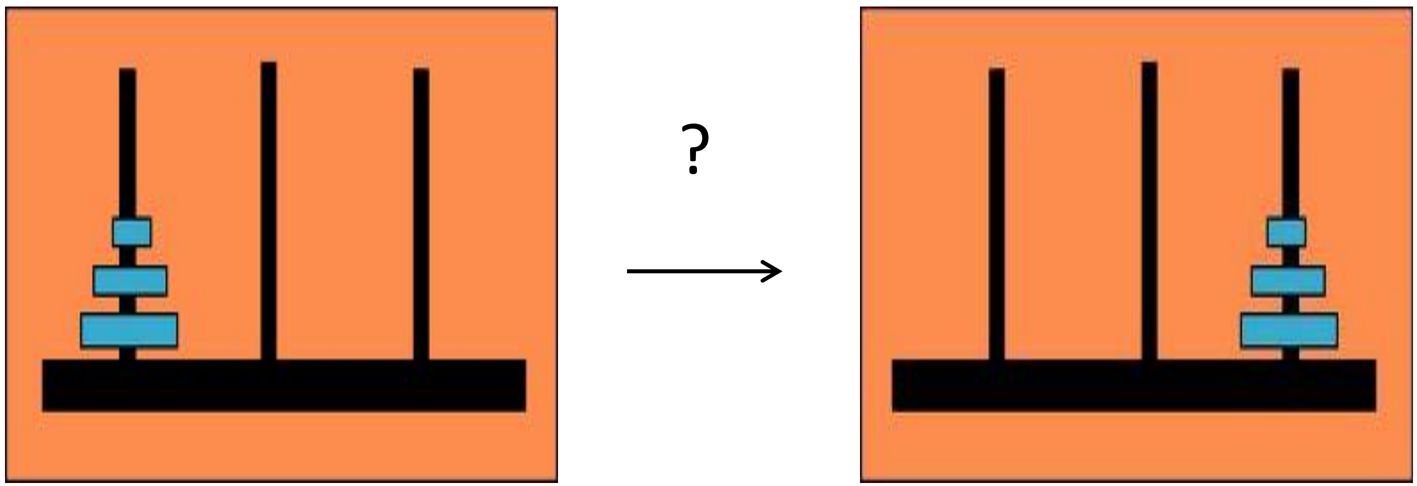
| KRITERIA | KETERANGAN |
|---|---|
| **Initial State** | Tidak ada Queen di Board |
| **Path Cost** | - |
| **Action** | Tambah Queen di salah satu lokasi di papan |
| **Goal Test** | 8 Queen ada di papan |

# Cannibals & Missionaries

# Cannibals & Missionaries

| KRITERIA | KETERANGAN |
|---|---|
| Initial State | |
| Path Cost | |
| Action | |
| Goal Test | |

# Towers of Hanoi

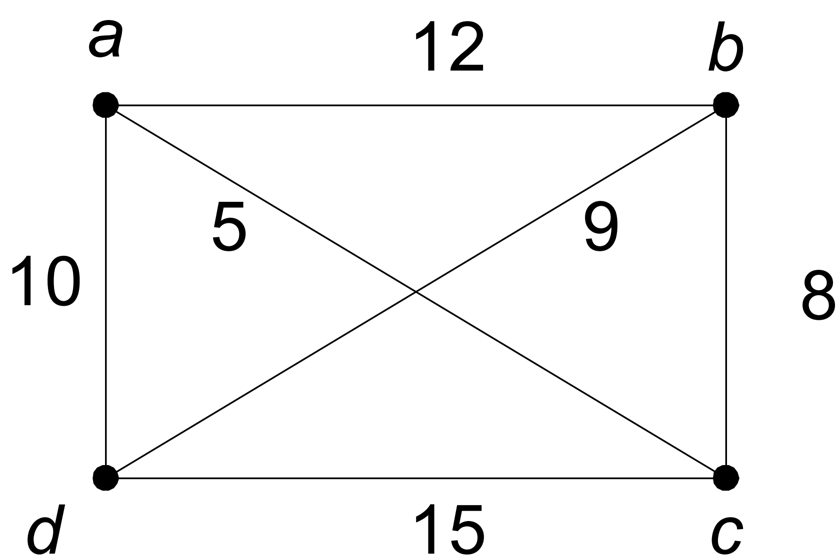



START GOAL

# Towers of Hanoi

| KRITERIA | KETERANGAN |
|---|---|
| **Initial State** | |
| **Path Cost** | |
| **Action** | |
| **Goal Test** | |

# TSP (Traveling Salesman Problem)




Temukan perjalanan (*tour*) **terpendek** yang melalui setiap kota lainnya hanya sekali dan **kembali *ke kota asal*** keberangkatan !

# TSP

| KRITERIA | KETERANGAN |
|---|---|
| **Initial State** | Isian Pengguna (kota Asal) |
| **Path Cost** | Jarak antar kota, … |
| **Action** | Pilih kota yang dituju |
| **Goal Test** | Semua kota sudah dikunjungi?<br>Sampai ke kota asal? |

# Route Finding

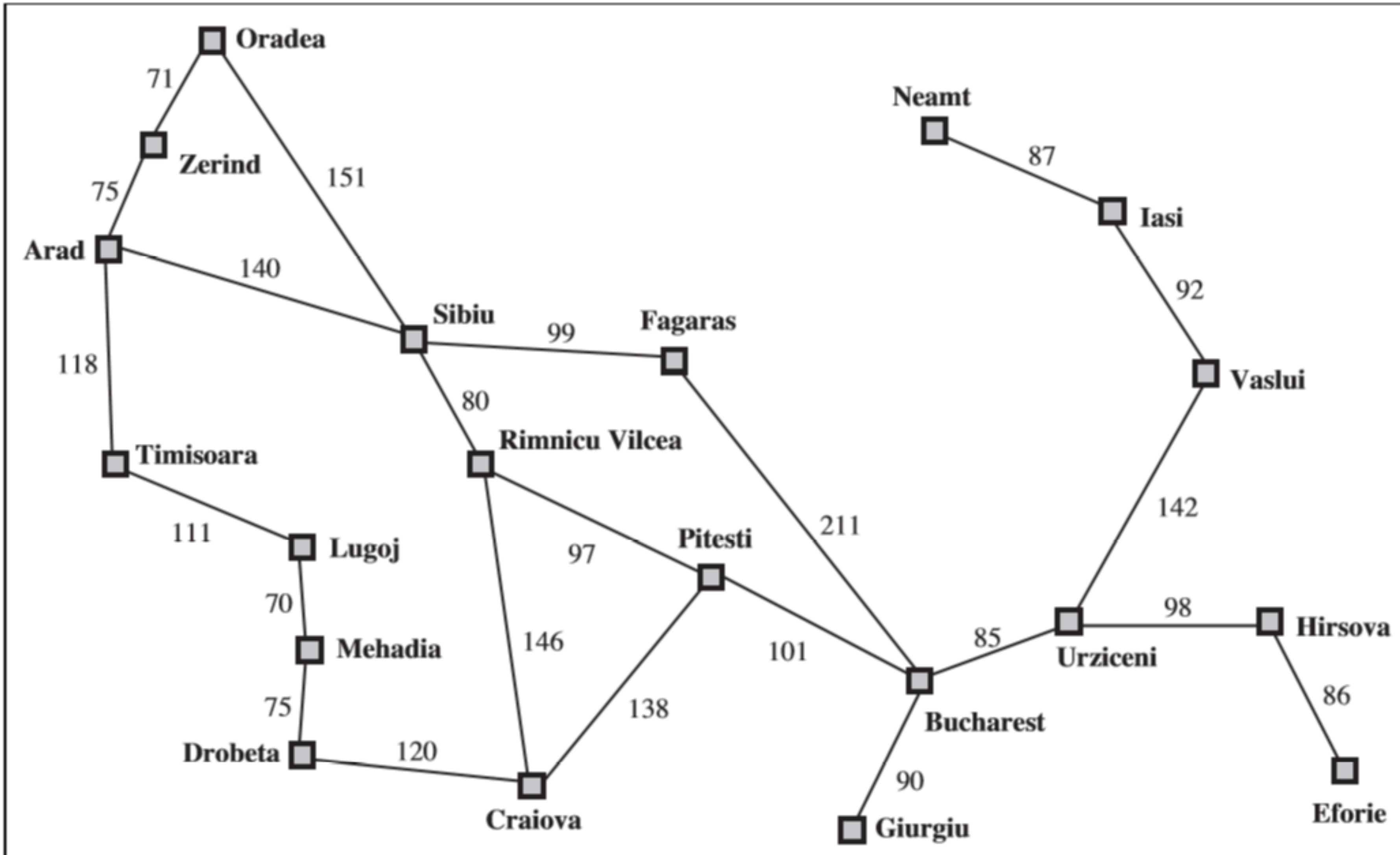



**Figure 3.2** A simplified road map of part of Romania.

# Route Finding

| KRITERIA | KETERANGAN |
|---|---|
| **Initial State** | Isian Pengguna (Kota Asal) |
| **Path Cost** | Biaya, waktu tunggu, waktu penerbangan, prosedur imigrasi, kualitas kelas penerbangan, jenis pesawat, … |
| **Action** | Ambil waktu penerbangan dari suatu kota, di salah satu kelas penerbangan, di suatu waktu |
| **Goal Test** | Sampai di tujuan? |

# SOLVING (SEARCHING)

# SEARCHING

*Proses **memilih** aksi untuk mencapai tujuan.*

**Goal-based Agent**

**Deterministic, Observable, Static, Known**

**Atomic**

**Formulate – Search – Execute**

# SYARAT SEARCHING

**GOAL Formulation** → **PROBLEM Formulation** → **SEARCH**

*Tentukan Tujuan*

*Tentukan Detail Masalah*

*Lakukan Pencarian*

# Kriteria Pencarian

**Completeness**

apakah metode tersebut menjamin penemuan solusi jika solusinya memang ada?

**Space Complexity**

berapa banyak memori yang diperlukan?

**Time Complexity**

berapa lama waktu yang diperlukan?

**Optimality**

apakah metode tersebut menjamin penemuan solusi yang terbaik jika terdapat beberapa solusi berbeda?

# Jenis Pencarian

**BLIND SEARCH**

*tidak ada informasi awal* yang digunakan dalam proses pencarian

**HEURISTIC SEARCH**

*ada informasi awal* yang digunakan dalam proses pencarian

# BLIND SEARCH

# BLIND SEARCH

Breadth-first Search (BFS)

Depth-first Search (DFS)

Uniform-cost Search

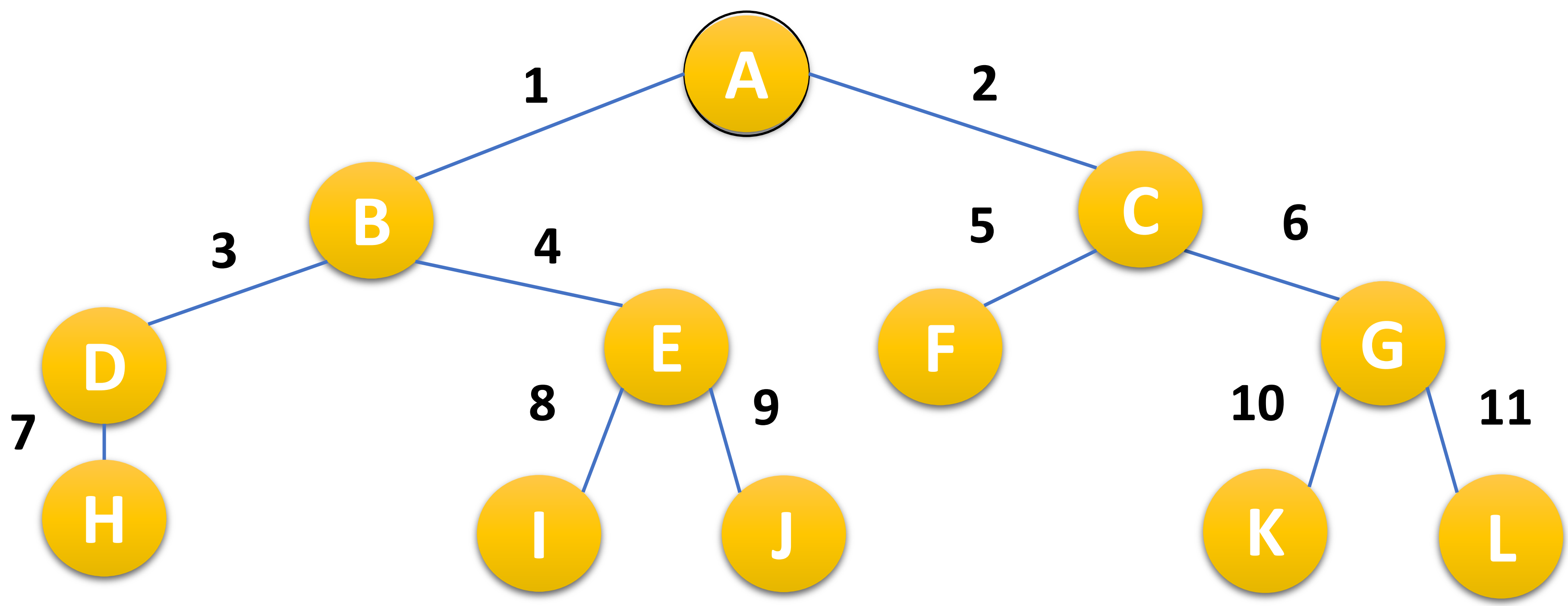
Iterative Deepening Search

Bidirectional Search

# Breadth-first Search

– Breadth-first search expands the shallowest nodes first; it is complete, optimal for unit step costs, but has exponential space complexity.

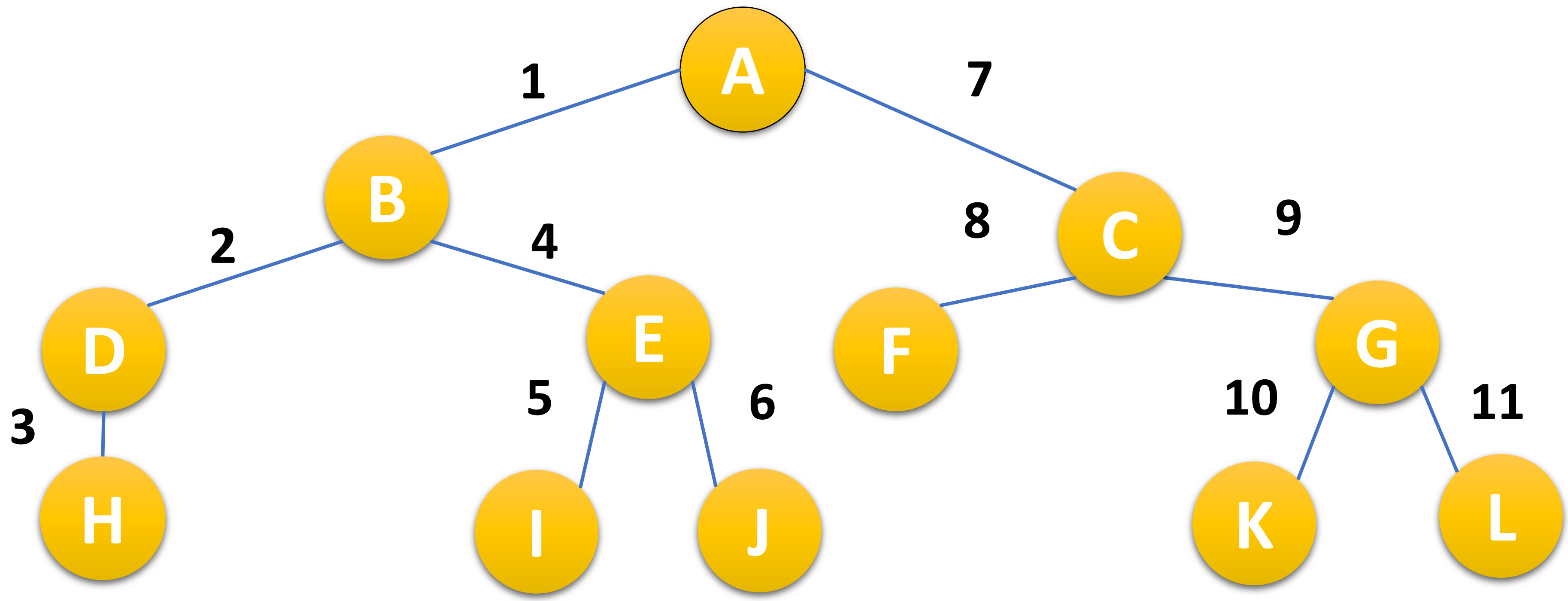
# Breadth-first Search



# Depth-first Search

– Depth-first search expands the deepest unexpanded node first. It is neither complete nor optimal, but has linear space complexity. Depth-limited search adds a depth bound

# Depth-first Search



# Iterative Deepening Search

. – Iterative deepening search calls depth-first search with increasing depth limits until a goal is found. It is complete, optimal for unit step costs, has time complexity comparable to breadth-first search, and has linear space complexity.

# Uniform-cost Search

– Uniform-cost search expands the node with lowest path cost, $g(n)$, and is optimal for general step costs.

# Bidirectional Search

– Bidirectional search can enormously reduce time complexity, but it is not always applicable and may require too much space.

# Perbandingan Kriteria

| Criterion | Breadth-First | Uniform-Cost | Depth-First | Depth-Limited | Iterative Deepening | Bidirectional (if applicable) |
|---|---|---|---|---|---|---|
| Complete? | Yes$^a$ | Yes$^{a,b}$ | No | No | Yes$^a$ | Yes$^{a,d}$ |
| Time | $O(b^d)$ | $O(b^{1+\lfloor C^*/\epsilon \rfloor})$ | $O(b^m)$ | $O(b^\ell)$ | $O(b^d)$ | $O(b^{d/2})$ |
| Space | $O(b^d)$ | $O(b^{1+\lfloor C^*/\epsilon \rfloor})$ | $O(bm)$ | $O(b\ell)$ | $O(bd)$ | $O(b^{d/2})$ |
| Optimal? | Yes$^c$ | Yes | No | No | Yes$^c$ | Yes$^{c,d}$ |

**Figure 3.21** Evaluation of tree-search strategies. $b$ is the branching factor; $d$ is the depth of the shallowest solution; $m$ is the maximum depth of the search tree; $l$ is the depth limit. Superscript caveats are as follows: $^a$ complete if $b$ is finite; $^b$ complete if step costs $\geq \epsilon$ for positive $\epsilon$; $^c$ optimal if step costs are all identical; $^d$ if both directions use breadth-first search.

# HEURISTIC SEARCH

# HEURISTIC SEARCH

Generic Best-first Search

Greedy Best-first Search

A*

RBFS & SMA*

# Generic Best-first Search

– The generic best-first search algorithm selects a node for expansion according to an evaluation function.

# Greedy Best-first Search

– Greedy best-first search expands nodes with minimal $h(n)$. It is not optimal but is often efficient.

# A*

– A ∗ search expands nodes with minimal f(n)=g(n)+h(n).

A ∗ is complete and optimal, provided that h(n) is admissible (for TREE-SEARCH) or consistent (for GRAPH-SEARCH). The space complexity of A∗ is still prohibitive.

# RBFS

– RBFS(recursive best-first search) and SMA∗ (simplified memory-bounded A∗) are robust, optimal search algorithms that use limited amounts of memory; given enough time, they can solve problems that A∗ cannot solve because it runs out of memory.

# SMA*

– RBFS(recursive best-first search) and SMA∗ (simplified memory-bounded A∗) are robust, optimal search algorithms that use limited amounts of memory; given enough time, they can solve problems that A∗ cannot solve because it runs out of memory.

# Any Question???

# REFERENSI . . .

Russell, S., Norvig, P. ***Artificial Intelligence A Modern Approach*** *(Third Edition)*. 2010. Pearson Education, USA.

# TUGAS KELOMPOK

Buatlah **Makalah** tentang salah satu dari algoritma di samping, dengan rincian sebagai berikut.

- Pengertian
- Contoh Kasus
- Hasil Pengukuran Kriteria (*Completeness, Time Complexity, Space Complexity, Optimality*)

***Uniform-cost Search***

***Iterative Deepening Search***

***Bidirectional Search***

***Generic Best-first Search***

***Greedy Best-first Search***

***A****

***RBFS & SMA****

***Deadline : H-1 pertemuan selanjutnya***