

*Inteligentny agent i jego otoczenie*  
*Szukanie jako metoda rozwiązywania problemów*  
*wykład 1*

dr inż. Joanna Kołodziejczyk

`jkolodziejczyk@wi.ps.pl`

Zakład Sztucznej Inteligencji ISZiMM

## Literatura

---

- Slajdy z wykładów: <http://ksir.wi.ps.pl/ESI>
- Linki podane na ww stronie
- Podręczniki podawane na slajdach z wykładu
- Artificial Intelligence: A Modern Approach by Stuart Russell and Peter Norvig, Prince Hall, 1995  
<http://www.cs.berkeley.edu/~russell/aima/>

# Plan wykładu

---

1. Wprowadzenie do sztucznej inteligencji
2. Inteligentny agent
  - (a) Agent i jego otoczenie
  - (b) Racjonalność
  - (c) Ocena działania, otoczenie, czujniki
  - (d) Typy środowisk w jakim może pracować agent
  - (e) Typy agentów
3. Rozwiązywanie problemów
  - (a) Agent z zadaniem rozwiązywania problemu
  - (b) Typy problemów
  - (c) Sformułowanie problemu
  - (d) Przykłady problemów

## Dlaczego chcemy studiować SI?

---

- SI próbuje odkryć jak działają jednostki inteligentne
- SI próbuje budować inteligentne jednostki
- SI na pewno ma i będzie miała wpływ na naszą przyszłość
- SI jest dziedziną silnie interdyscyplinarną
- SI w chwili obecnej składa się z wielu poddziedzin (niektóre zostaną omówione na wykładzie)
- Czy można zrozumieć jak to jest możliwe, że za pomocą małego i wolnego mózgu można postrzegać, rozumieć, przewidywać i manipulować elementami świata dużo bardziej złożonego?

## Czym jest inteligencja?

---

- **Psychologia**: cecha umysłu odpowiadająca za sprawność w zakresie myślenia, rozwiązywania problemów i innych czynności poznawczych.
- **Filozofia**: czynność intelektu, polegająca na aktualnym rozumieniu poznawanej rzeczy; *INTELEKT - umysł, rozum; całość wiedzy, doświadczenia i zdolności umysłowych człowieka; utożsamiany niekiedy z inteligencją.*
- **Biologia**: obserwowana w warunkach naturalnych lub eksperymentalnych umiejętność niektórych zwierząt szybkiego znalezienia najtrafniejszego postępowania w nowej, nieznanej sytuacji.

# Czym jest sztuczna inteligencja?

Różne definicje według różnych kryteriów.

Dwa wymiary:

1. myślenie/rozumowanie kontra działanie/zachowanie
2. sukces w naśladowaniu ludzkich standardów kontra sukces w osiągnięciu idealnej inteligencji (niezawodnej) nazwijmy ją **racjonalnej**

System, który <b>myśli</b> jak człowiek.	System, który <b>myśli</b> racjonalnie.
System, który <b>zachowuje się</b> jak człowiek.	System, który <b>zachowuje się</b> racjonalnie.

# Ludzkie myślenie (zachowanie)

Alan Turing (1950) „Computing machinery and intelligence”:

- „Czy maszyna myśli?” → „Czy maszyna może zachowywać się inteligentnie?”
- Test operacyjny na inteligentne zachowanie: Gra w naśladownictwo (Imitation Game)



## Dziedzictwo pomysłu Turinga

- Turing twierdził, że do 2000 roku będzie można mówić o myślących maszynach i taka maszyna będzie miała 30% szans by w 5 minutowej konwersacji zwieść odpytującego.
- Maszyna wymaga współdziałania wszystkich głównych komponentów SI: wiedzy, wnioskowania, rozumienia języka naturalnego, uczenia się. Dodatkowo w pełnym teście potrzebne będą takie elementy jak widzenie i robotyka.
- Myślenie w sensie Turinga
  - Myślenie jest domeną człowieka i nie dotyczy maszyn?  
„Asking whether computers can think is like asking whether submarines can swim.”
  - Jak sprawdzić czy ktoś myśli?
- Problemy z testem: nie da się go powtórzyć i nie da się go matematycznie przeanalizować.



## Nagroda Loebnera

---

### What is the Loebner Prize?

The Loebner Prize for artificial intelligence ( AI ) is the first formal instantiation of a Turing Test. The test is named after Alan Turing the brilliant British mathematician...

In 1990 Hugh Loebner agreed with The Cambridge Center for Behavioral Studies to underwrite a contest designed to implement the Turing Test. Dr. Loebner pledged a Grand Prize of **\$100,000** and a Gold Medal (pictured above) for the first computer whose responses were indistinguishable from a human's. Such a computer can be said "to think. Each year an annual prize of **\$2000** and a bronze medal is awarded to the most human-like computer. The winner of the annual contest is the best entry relative to other entries that year, irrespective of how good it is in an absolute sense.

<http://www.loebner.net/Prizef/loebner-prize.html>

# Myślenie ludzkie: kognitywistka

---

## Jak ludzie myślą?

- Wymagana jest teoria naukowa wewnętrznej aktywności mózgu.
- Dwie drogi badań
  - introspekcja — próba przechwycenia myśli podczas ich pojawiania się — testowanie zachowań na ludziach
  - w oparciu o neurologię (a dokładnie **cognitive neuroscience**) (odkrycie zasad działania mózgu da szansę na jego implementację)

Dostępne teorie nie wyjaśniają niczego, co byłoby podobne do ludzkiej inteligencji w ujęciu ogólnym.

# Podstawy sztucznej inteligencji

---

<b>Filozofia</b>	logika, metody dowodzenia rozum jako fizyczny model podstawy uczenia się, język naturalny, rozumowanie
<b>Matematyka</b>	formalna reprezentacja i dowodzenie algorytmy, obliczenia, teoria złożoności prawdopodobieństwo
<b>Psychologia</b>	adaptacja zjawisko postrzegania i sterowania motorycznego techniki eksperymentalne
<b>Ekonomia</b>	formalna teoria racjonalnych decyzji
<b>Lingwistyka</b>	reprezentacja wiedzy gramatyka

# Myślenie racjonalne

Normatywne (lub niepodważalne), nie domyślne.

Arystoteles: Na czym polega „poprawny” proces myślenia? —> sylogizm.

W Grecji opracowano różne logiki:

- notacje,
- reguły wyprowadzania wniosków („myśli”).

Tradycja logistyczna dała nadzieję na zmechanizowanie myślenia i stworzenie systemów z użyciem programowania logicznego.

Główne problemy:

- Nie cała inteligencja jest oparta na „żelaznej” logice.
- Jest różnica w rozwiązaniach praktycznych i teoretycznych.

## Działanie racjonalne

**Racjonalne** zachowanie: „działanie poprawne”.

**Działanie poprawne**: to takie, dzięki któremu maksymalizujemy szansę osiągnięcia celu. Działanie odbywa się na podstawie dostępnych informacji.

Z pewnych działań można nawet wyłączyć myślenie np. refleks.

Zalety:

- Bardziej uogólniona idea niż myślenie.
- Daje łatwiejszy grunt dla badań naukowych.

Racjonalność jest możliwa do osiągnięcia w warunkach **idealnych**.

*Racjonalność nie jest dobrym modelem rzeczywistości.*

# Racjonalny agent

**Agent** jest jednostką, która postrzega i działa.

Niniejszy kurs jest o tym, jak projektować **racjonalnych agentów**.

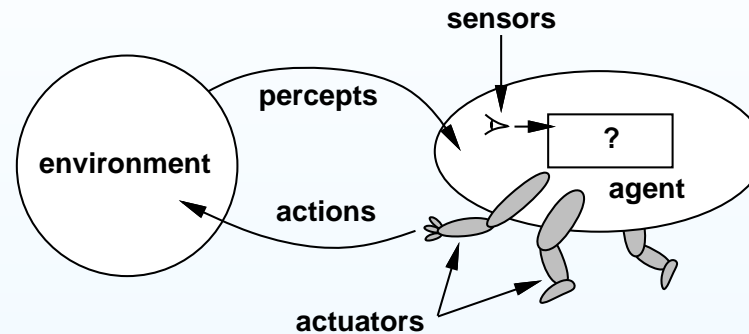
Formalizując, agent jest **funkcją** odwzorowującą zanotowane spostrzeżenia w akcje.

$$f : \mathcal{P}^* \rightarrow \mathcal{A}$$

Dla każdej klasy otoczenia, w którym agent będzie działał będziemy poszukiwać agenta lub grupy agentów o najlepszych osiągnięciach.

*Ograniczenia obliczeniowe są główną przyczyną porażki w poszukiwaniu idealnego racjonalizmu* → stąd celem jest opracowanie najlepszego **programu** na obecne zasoby obliczeniowe.

# Agent i jego otoczenie



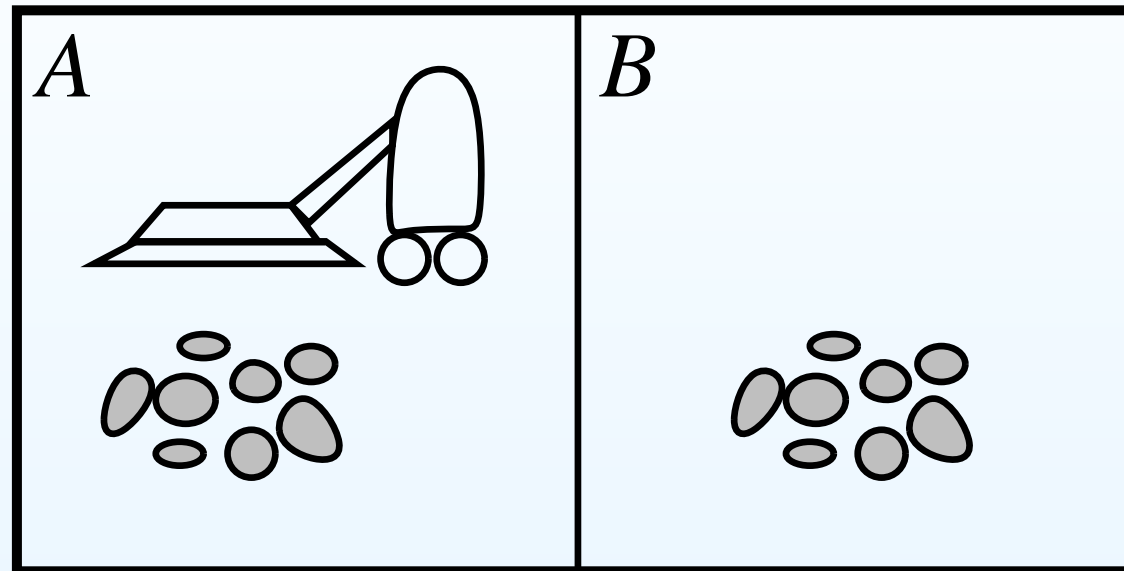
Za **inteligentnego agenta** uznamy człowieka, robota, softbota, termostat, itp.

**Funkcja agent** mapuje historię percepcji (odczyty z czujników, klawiatury) na akcje:

$$f : \mathcal{P}^* \rightarrow \mathcal{A}$$

**Program agent** uruchamiany jest w urządzeniu (**hardware:**) by realizować  $f$ .

## Odkurzacz i jego świat



Percepcja: lokalizacja i stan podłogi, np., [*A*, *Brud*]

Akcja: *Lewo*, *Prawo*, *Odkurz*, *Nic*



# Agent - odkurzacz

Sekwencje percepcji	Działanie
[A, Czysto]	<i>Prawo</i>
[A, Brud]	<i>Odkurz</i>
[B, Czysto]	<i>Lewo</i>
[B, Brud]	<i>Odkurz</i>
[A, Czysto], [A, Czysto]	<i>Prawo</i>
[A, Czysto], [A, Brud]	<i>Odkurz</i>
[A, Czysto], [B, Czysto]	<i>Lewo</i>
⋮	⋮

```
1 function Reflex-Vacuum-Agent(location, status) return action
2   if status = Brud then return Odkurz
3   else if location = A then return Prawo
4   else if location = B then return Lewo
```

## Oczekiwane zachowanie agenta

---

Funkcja, którą agent ma wypełnić ma dawać takie efekty, iż zachowanie agenta będzie można określić jako „**dobre**”.

W przybliżeniu dobre akcje, to takie, które przynoszą jak największą korzyść. Zatem należy znaleźć sposób na mierzenie odnoszonych przez agenta korzyści (miarę jego sukcesu).

Pełna specyfikacja agenta obejmuje:

- jak mierzyć korzyści z działania agenta,
- w jakim środowisku (z czym ma kontakt) agent i jak to wpływa na jego zachowanie o ile w ogóle,
- jak i jakie informacje przekazywane są do agenta,
- jak i jakie informacje stanowią informację zwrotną od agenta.

## Mierzenie korzyści z działania agenta

---

**Miara osiągnięć** to kryterium sukcesu. Nie ma uniwersalnego miernika dla agentów. Dla każdego zadania konstruktor określa taką miarę.

Przykład miary osiągnięć dla agenta-odkurzacza:

- Ile razy odkurzono brud w ciągu ośmiogodzinnej zmiany.
- Jedno pole sprzątnięte w jednostce czasu  $T$ .
- Jeden punkt za posprzątnięcie pola w jednostce czasu minus jeden punkt za ruch, który powoduje zużycie energii.

Lepiej konstruować miarę osiągnięć według tego jak agent ma zmieniać środowisko, a nie jak ma się zachowywać w oderwaniu od niego.

## Agent racjonalny

**Agent racjonalny** wybiera taką akcję dla pojawiającej się sekwencji wejściowej, która maksymalizuje **oczekiwaną** wartość miary osiągnięć, określoną na podstawie wejść i wiedzy jaka jest w agencie zapamiętana.

Racjonalny  $\neq$  wszechwiedzący

– percepcja nie musi dostarczać całkowitej wymaganej informacji

Racjonalny  $\neq$  jasnowidzący

– wynik akcji nie musi być zgodny z oczekiwaniami

Racjonalny  $\neq$  odnoszący zawsze sukces

Racjonalny  $\rightarrow$  **zbiera informacje** (zwiększenie szans na nagrodę), **uczy się** (rozszerzanie wiedzy), **jest autonomiczny** (więcej niż tylko wbudowana wiedza - zmiana zachowań na podstawie doświadczenia).

# Automatyczna taksówka

---

Aby stworzyć racjonalnego agenta należy określić **PEAS**, czyli **P**erformance, **E**nvironment, **A**ctuators, **S**ensors. Ważne jest, że agent musi działać zgodnie z oczekiwaniami czyli miarą osiągnięć.

Rozważmy agenta - automatyczna taksówka:

1. Ocena osiągnięć

# Automatyczna taksówka

Aby stworzyć racjonalnego agenta należy określić **PEAS**, czyli **P**erformance, **E**nvironment, **A**ctuators, **S**ensors. Ważne jest, że agent musi działać zgodnie z oczekiwaniami czyli miarą osiągnięć.

Rozważmy agenta - automatyczna taksówka:

1. **Ocena osiągnięć** — bezpieczeństwo, cel podróży, profit, legalność, komfort
2. **Otoczenie** — ulice, ruch uliczny, piesi, pogoda
3. **Aktualizatory** — kierownica, pedał gazu, hamulec, klakson, mikrofon
4. **Czujniki** — wideo, prędkościomierz, wskaźnik poziomu paliwa, czujniki silnika, klawiatura, GPS

# Agent zakupów internetowych

---

## 1. Ocena osiągnięć

# Agent zakupów internetowych

---

1. **Ocena osiągnięć** — cena, jakość, zgodność z upodobaniami, efektywność
2. **Otoczenie** — obecne i przyszłe strony WWW, sprzedawca, dostawca
3. **Aktualizatory** — wyświetlacz dla użytkownika, śledzenie linków, wypełnianie formularzy
4. **Czujniki** — tekst, grafika, pismo



# Typy otoczenia

	Krzyżówka	Warcaby	Analiza obrazów	taksówka
Obserwowalne				

## Typy otoczenia

Otoczenie w **pełni** obserwowalne (czujniki wykrywają wszystkie istotne aspekty, które są niezbędne dla wykonania dobrej akcji) lub **częściowo obserwowalne**.

	Krzyżówka	Warcaby	Analiza obrazów	taksówka
Obserwowalne Deterministyczne	w pełni	w pełni	w pełni	częściowo

## Typy otoczenia

**Deterministyczne** lub **stochastyczne**: jeżeli kolejny stan otoczenia jest całkowicie określony na podstawie stanu bieżącego i wykonanej akcji, to otoczenie jest deterministyczne. Niepewność skutkuje otoczeniem stochastycznym.

	Krzyżówka	Warcaby	Analiza obrazów	taksówka
Obserwowalne	w pełni	w pełni	w pełni	częściowo
Deterministyczne	tak	stochast.	tak	stochast.
Epizodyczne				

## Typy otoczenia

**Epizodyczne** lub **sekwencyjne**: W otoczeniu epizodycznym doświadczenie agenta może być podzielone na atomowe kroki, w których agent postrzega i wykonuje pojedynczą akcję. Wybór działania zależy jedynie od listy epizodów, a nie od działania wykonanego poprzednio.

	Krzyżówka	Warcaby	Analiza obrazów	taksówka
Obserwowalne	w pełni	w pełni	w pełni	częściowo
Deterministyczne	tak	stochast.	tak	stochast.
Epizodyczne	sekwen.	sekwen.	tak	sekwen.
Statyczne				

## Typy otoczenia

**Statyczne** lub **dynamiczne**: Jeżeli otoczenie może zmieniać się w czasie kiedy agent wybiera akcję, to środowisko jest dynamiczne. Semi-dynamiczne to takie, gdy miara osiągnięć agenta zmienia się nawet, gdy otoczenie jest niezmiennie w czasie.

	Krzyżówka	Warcaby	Analiza obrazów	taksówka
Obserwowalne	w pełni	w pełni	w pełni	częściowo
Deterministyczne	tak	stochast.	tak	stochast.
Epizodyczne	sekwen.	sekwen.	tak	sekwen.
Statyczne	tak	tak	semi	dynamic.
Dyskretne				

## Typy otoczenia

**Dyskretne** lub **ciągłe**: Zależy od tego jak widziany jest czas zmian stanu otoczenia oraz percepcji i działania agenta.

	Krzyżówka	Warcaby	Analiza obrazów	taksówka
Obserwowalne	w pełni	w pełni	w pełni	częściowo
Deterministyczne	tak	stochast.	tak	stochast.
Epizodyczne	sekwen.	sekwen.	tak	sekwen.
Statyczne	tak	tak	semi	dynamic.
Dyskretne	tak	tak	ciągłe	ciągłe
Pojedynczy agent				

## Typy otoczenia

**Pojedynczy** lub **multiagent**: Czy otoczenie zawiera innych agentów, którzy również maksymalizują swoją miarę osiągnięć? Ich działanie zależy od działania rozpatrywanego agenta.

	Krzyżówka	Warcaby	Analiza obrazów	taksówka
Obserwowalne	w pełni	w pełni	w pełni	częściowo
Deterministyczne	tak	stochast.	tak	stochast.
Epizodyczne	sekwen.	sekwen.	tak	sekwen.
Statyczne	tak	tak	semi	dynamic.
Dyskretne	tak	tak	ciągłe	ciągłe
Pojedynczy agent	tak	multi	tak	multi

# Typy otoczenia

	Krzyżówka	Warcaby	Analiza obrazów	taksówka
Obserwowalne	w pełni	w pełni	w pełni	częściowo
Deterministyczne	tak	stochast.	tak	stochast.
Epizodyczne	sekwen.	sekwen.	tak	sekwen.
Statyczne	tak	tak	semi	dynamic.
Dyskretne	tak	tak	ciągłe	ciągłe
Pojedynczy agent	tak	multi	tak	multi

*Typ otoczenia silnie wpływa na projekt agenta.*

Rzeczywistość jest częściowo obserwowalna, stochastyczna, sekwencyjna, dynamiczna, ciągła, wielo-agentowy.



# Struktura agenta

---

$$\textit{agent} = \textit{hardware} + \textit{program}$$

Hardware, to rodzaj urządzenia zapewniającego sensory i aktualizatory.

Program musi być dostosowany do sprzętu.

Kurs poświęcony jest projektowaniu programów agentów (część sprzętowa jest pomijana).

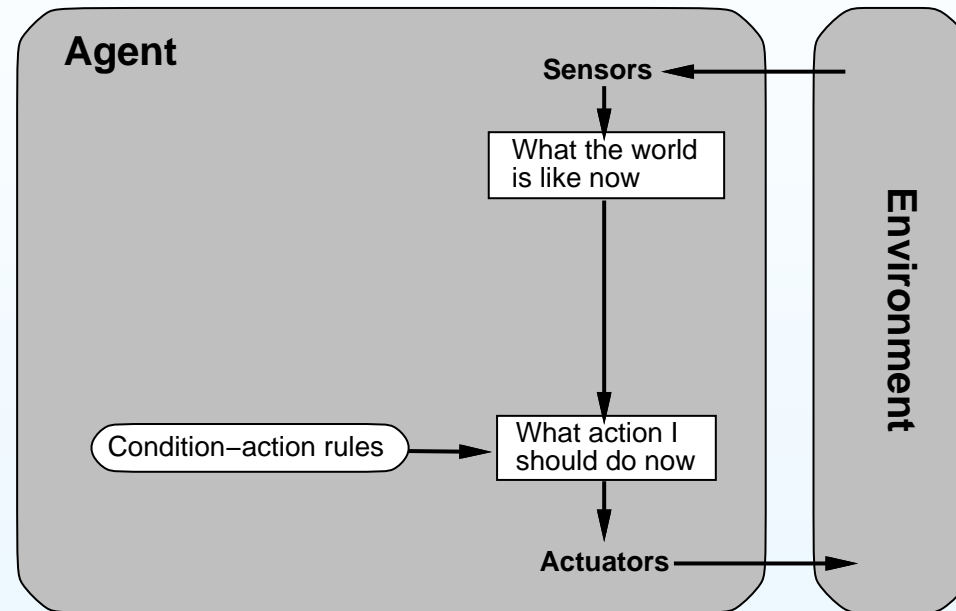
## Typy agentów

Cztery główne typy w kolejności zwiększającej się ogólności ich zastosowań

- Prosty agent działający na zasadzie odruchów — **reflex agent**.
- Prosty agent działający na zasadzie odruchów ze stanem — **reflex agent with state**.
- Agent ukierunkowany — **goal-based agent**.
- Agent użyteczny — **utility-based agent**.

Wszystkie mogą dodatkowo zawierać moduł odpowiedzialny za uczenie się.

# Reflex agent

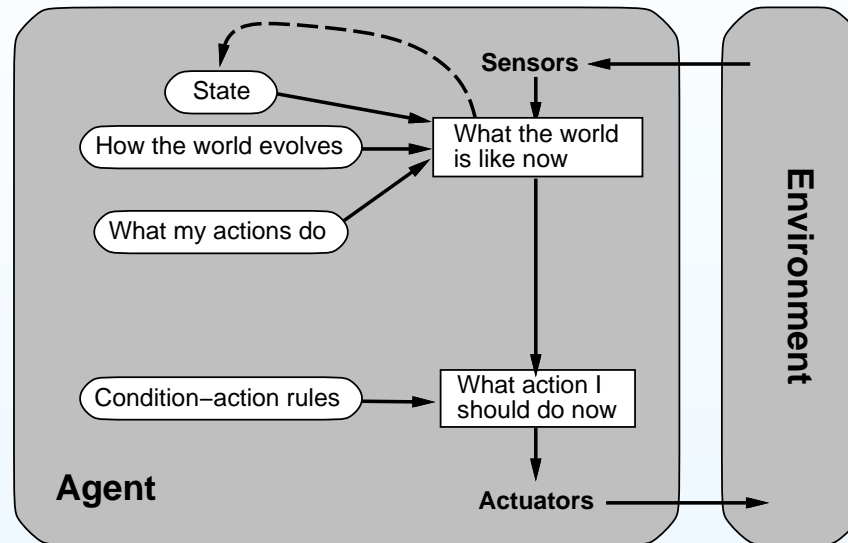


```
1 function Simple-Reflex-Agent(percept) return action
2   static: rules: a set of condition-action rules
3
4   state = InterpretInput(percept)
5   rule = RuleMatch(state, rules)
6   action = RuleAction[rule]
```

## Reflex agent

- nie wykorzystuje historii percepcji
- condition-action rule to np.: if samochód przed tobą then zainicjalizuj hamowanie
- może pracować tylko w środowisku w pełni obserwowalnym
- sposobem na uniknięcie nieskończonych pętli zachowań (powstających przy niepełnej obserwowalności) jest stosowanie losowych zmian w akcjach

# Model-based reflex agents



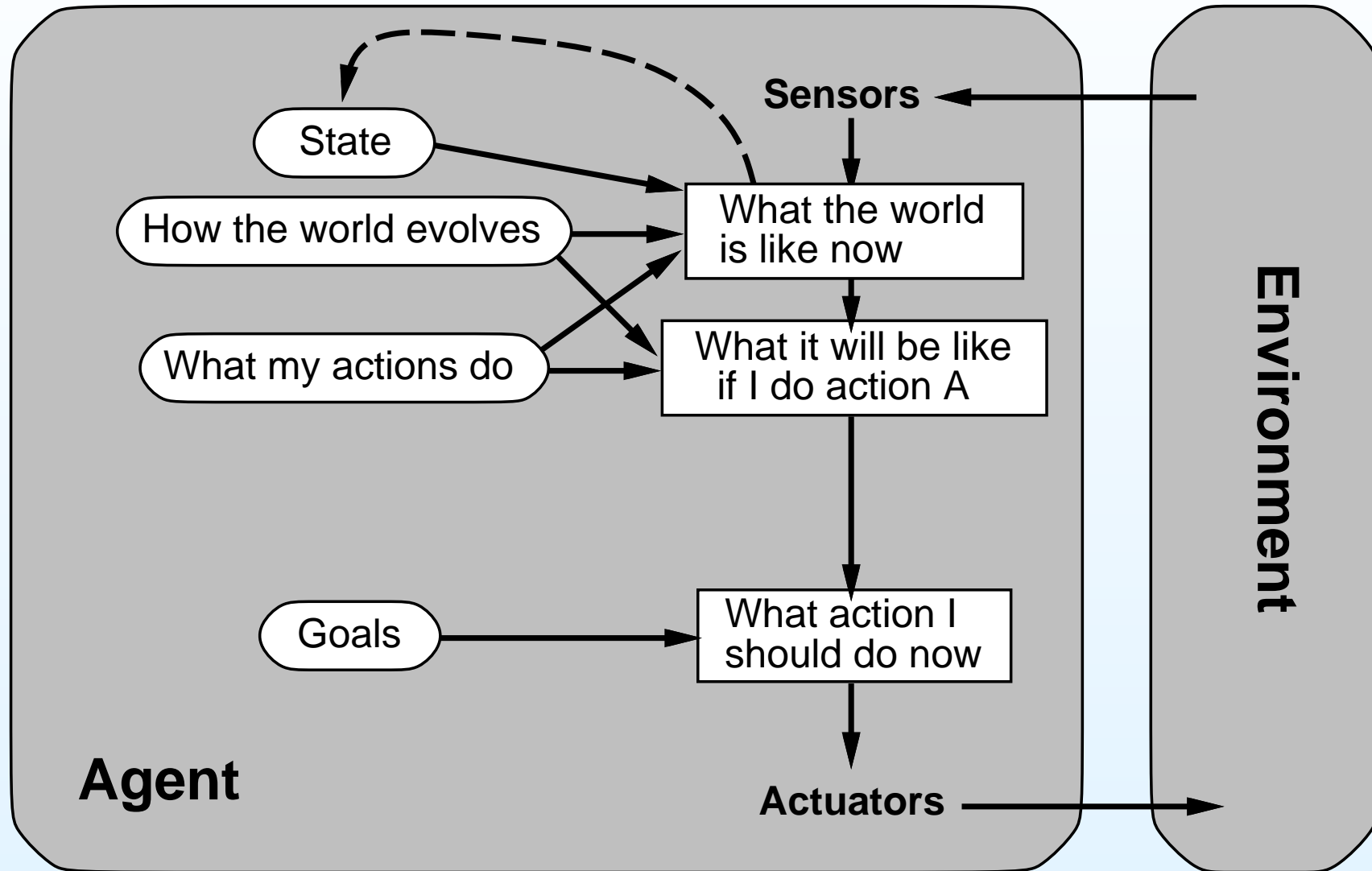
```
1 function Model-based-Reflex-Agent(percept) return action
2   static: rules: a set of condition-action rules
3           state: a description of the current world state
4           action: the most recent action, initially none
5
6   state = UpdateState(state,action,percept)
7   rule = RuleMatch(state, rules)
8   action = RuleAction[rule]
```

## Model-based reflex agents

---

- przechowuje pewną wiedzę o stanach np. przeszłych, które teraz nie są obserwowalne
- podjęcie akcji zależy od tego jak wyewoluowało środowisko
- przechowuje informacje jak jego własne akcje wpłyną na otoczenie, czyli przechowuje wiedzę „how the world works”, co nazywa się modelem świata
- funkcja uaktualniania stanu jest odpowiedzialna za stworzenie wewnętrznego opisu stanu jak i interpretację w świetle wiedzy o świecie i tym jak się zmienił

# Goal-based agents



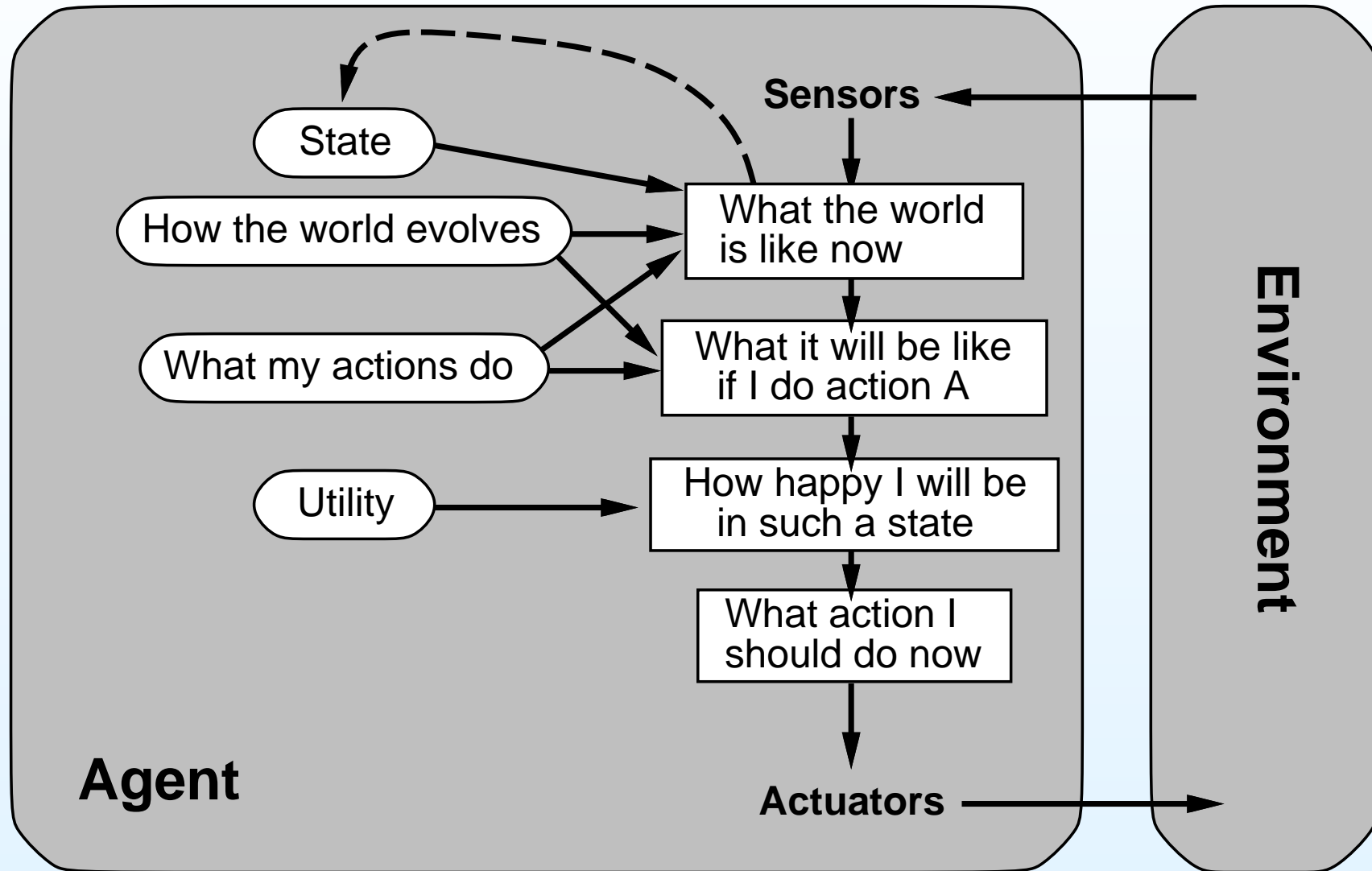
## Goal-based agents

---

- podejmowana decyzja może być zależna nie tylko od bieżącego stanu, ale od celu jaki agent ma postawiony
- jeżeli cel osiągnie się przez wykonanie pojedynczej akcji, to zadanie jest proste
- w rzeczywistych zadaniach należy znaleźć długi i nie zawsze łatwy do znalezienia ciąg akcji prowadzący do celu
- przeszukiwanie i planowanie, to poddziedziny, w których wykorzystuje się goal-based agentów
- wiedza, na podstawie której podejmuje decyzje jest zewnętrzna i dlatego łatwiej ją zmieniać, a agent jest przenośny



# Utility-based agents

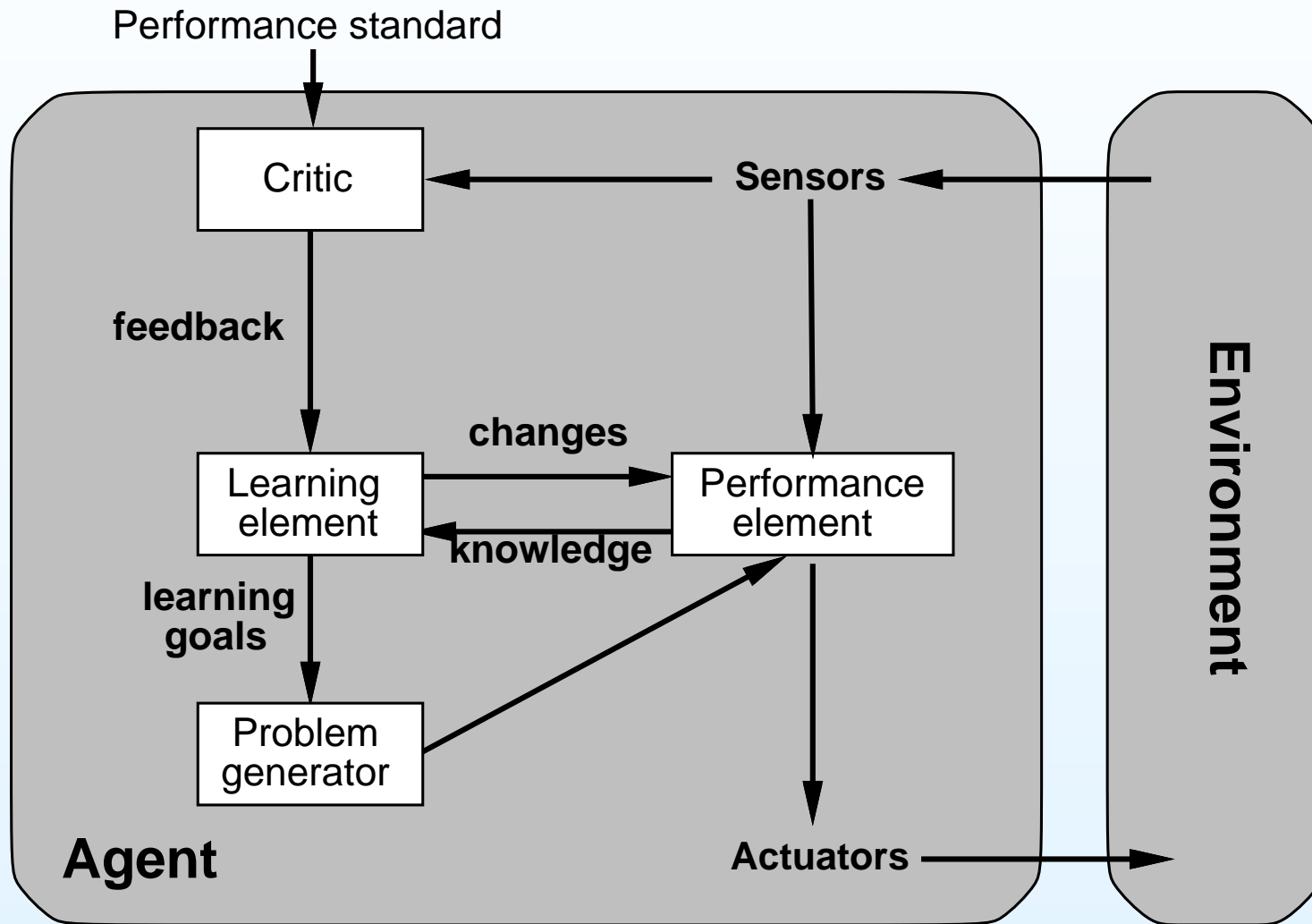


## Utility-based agents

---

- sam cel nie zawsze wystarczy, można również oceniać jak osiągnąć go wydajniej
- tworzy się funkcję użyteczności, która oszacowuje jakość bieżącego stanu
- w wypadku kilku akceptowalnych sekwencji prowadzących do celu pozwala wybrać taką, o najlepszych parametrach użyteczności

# Learning agents



## Learning agents

- learning element — odpowiedzialny za postępy
- performance element — wybór akcji (wcześniej prezentowane typy agentów dokładnie tym się zajmowały)
- critic — ocenia zachowanie bieżące agenta, by ewentualnie skorygować jego zachowanie w przyszłości i przekazuje te informacje do modułu uczącego się
- problem generator — sugeruje akcje prowadzące do nowych doświadczeń

# Rozwiązywanie problemów przez przeszukiwanie

czyli jak zbudować agenta, który znajdzie sekwencję akcji prowadzącą do oczekiwanego celu (lub celów).

Problem-solving agent to agent typu goal-based. **Problem**, to elementy, wśród których szuka się elementu **solution**.

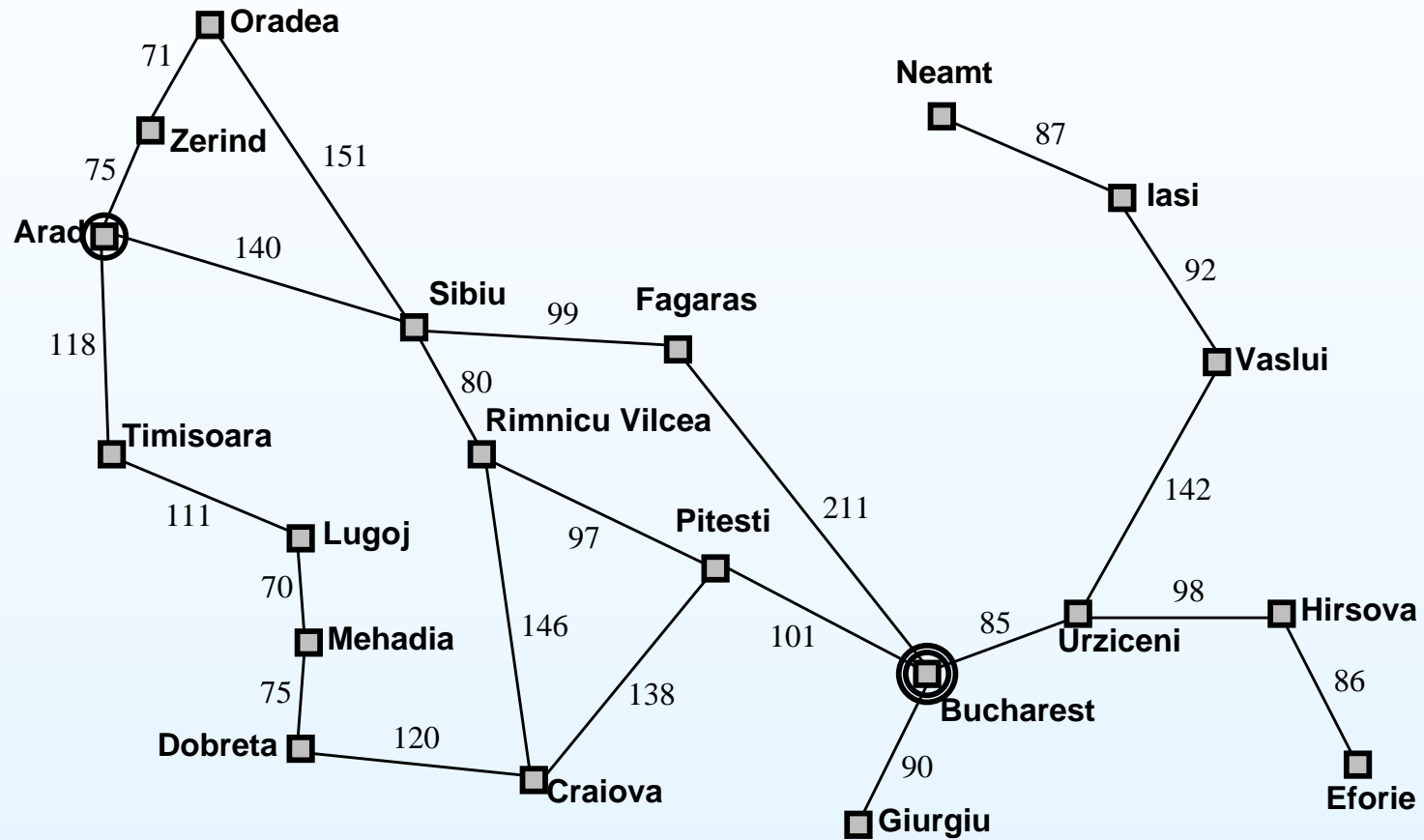
# Problem-solving agent

---

Cztery główne kroki w zadaniach (problem-solving)

1. **Sformułowanie celu:** Jaki stan dla rozpatrywanego mikroświata jest stanem oczekiwanym (zwycięskim)?
2. **Określenie problemu:** Jakie stany i akcje należy rozważyć by osiągnąć cel?
3. **Poszukiwanie rozwiązania:** Określenie możliwej sekwencji akcji, która prowadzi do stanów o znanych wartościach i wybór najlepszej sekwencji.
4. **Wykonanie:** Podaj rozwiązanie jako ciąg wykonanych akcji.

# Przykład: Rumunia



## Przykład: Rumunia

Jesteś na wakacjach w Rumunii. Obecnie w Aradzie. Jutro masz samolot z Bukaresztu.

**Określenie celu:** dotrzeć do Bukaresztu

**Określenie problemu:**

- **stany:** różne miasta
- **akcje:** podróż pomiędzy miastami

**Określenie rozwiązania:** porządek odwiedzanych miast np. Arad, Sibiu, Fagaras, Bukareszt



## W jakim środowisku pracuje problem-solving agent?

---

- **Styczne** — problem-solving (stanów i rozwiązania) jest określone bez udziału wpływu środowiska i zmian jakie mogą w nim nastąpić.
- **W pełni obserwowalne** — stan inicjalizujący jest dany.
- **Dyskretne** — alternatywne akcje mają charakter wyliczeniowy.
- **Deterministyczne** — rozwiązanie jest sekwencją akcji i nie ma możliwości wystąpienia nieprzewidzianych wydarzeń.

# Poprawne określanie problem-solving

**Problem** definiuje się przez :

- Stan początkowy: np. Arad
- Akcja - funkcja generująca potomka:  $S(X)$  zbiór par  $\langle akcja, stannastpny \rangle$  np.  
 $S(Arad) = \{ \langle Arad \rightarrow Zerind, Zerind \rangle, \dots \}$
- Stan końcowy - cel:
  - Wyrażny: np.  $X = Bukareszt$
  - Ukryty: np.  $szach - mat(X)$
- Koszt ścieżki: funkcja przypisująca numeryczną wartości do sekwencji stanów prowadzących do rozwiązania lub innego stanu pośredniego (informacja dodatkowa) np. suma odległości, numer wykonanej akcji, itp.  
 $c(x, a, y)$  jest kosztem wykonania kroku i zakłada się, że  $\geq 0$

**Rozwiązanie:** To ciąg, w którym zapisane są kolejne akcje, jakie wykonano przechodząc od stany początkowego do stanu końcowego. **Optymalne** rozwiązanie ma najmniejszy koszt ścieżki.

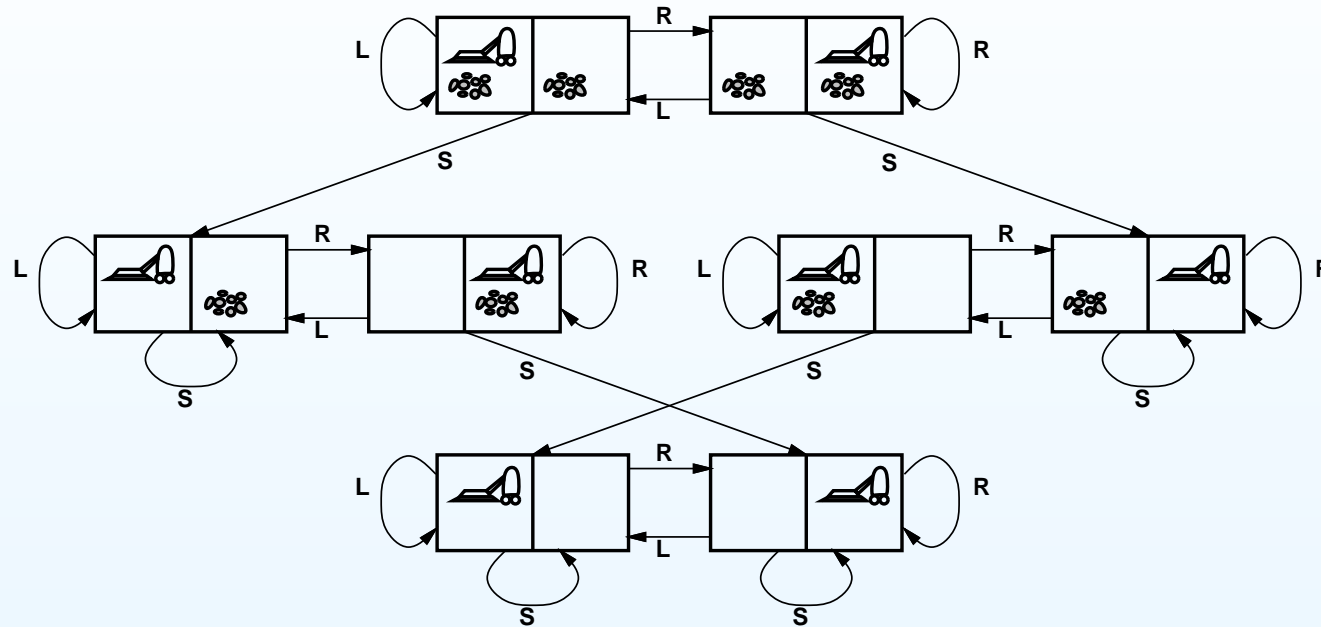
## Szczegóły o określaniu problem-solving

Rzeczywistość jest szalenie skomplikowana i wymaga uproszczenia (**abstraction**) (wycięcia szczegółowych informacji) reprezentacji by możliwe było rozpatrywanie przestrzeni stanów.

- **(Uproszczony) stan:** zbiór stanów rzeczywistych
- **(Uproszczona) akcja:** kombinacja rzeczywistych działań  
— np: *Arad*  $\rightarrow$  *Zerind* reprezentuje złożony zbiór możliwych dróg, miejsc na odpoczynek, itp.  
— Uproszczenie jest dopuszczalne jeżeli ścieżka odzwierciedli rzeczywisty świat.
- **(Uproszczone) rozwiązanie:** zbiór ścieżek, które występują w świecie rzeczywistym.

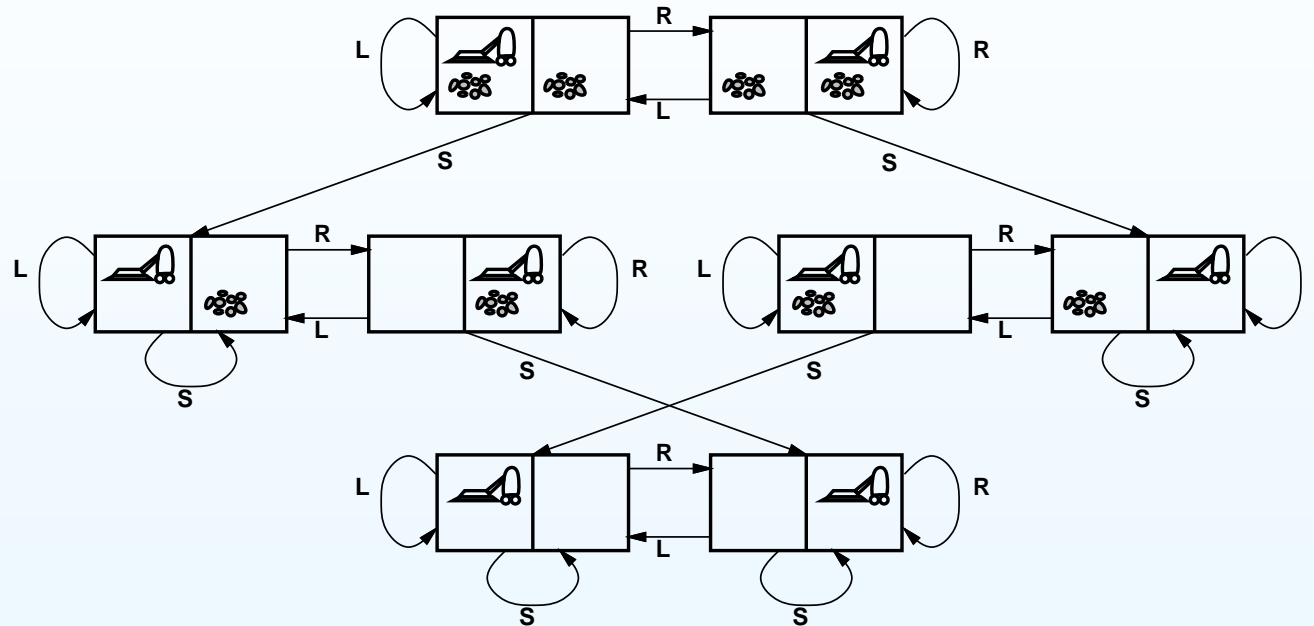
Każda uproszczona akcja jest łatwiejsza niż problem rzeczywisty.

# Przykład: odkurzacz



- Stany:
- Stan początkowy:
- Akcja:
- Jak sprawdzić cel:
- Koszt ścieżki:

## Przykład: odkurzacz



- **Stany:** dwa pola z brudem lub bez:  $2 \cdot 2^2 = 8$  stanów.
- **Stan początkowy:** Każdy stan może nim być.
- **Akcja:**  $\{Lewo, Prawo, Odkurz\}$
- **Jak sprawdzić cel:** Czy oba pola czyste?
- **Koszt ścieżki:** Liczba wykonanych akcji do osiągnięcia celu.

## Przykład: układanka ośmioelementowa

7	2	4
5		6
8	3	1

Start State

1	2	3
4	5	6
7	8	

Goal State

- Stany:
- Stan początkowy:
- Akcja:
- Jak sprawdzić cel:
- Koszt ścieżki:

## Przykład: układanka ośmioelementowa

7	2	4
5		6
8	3	1

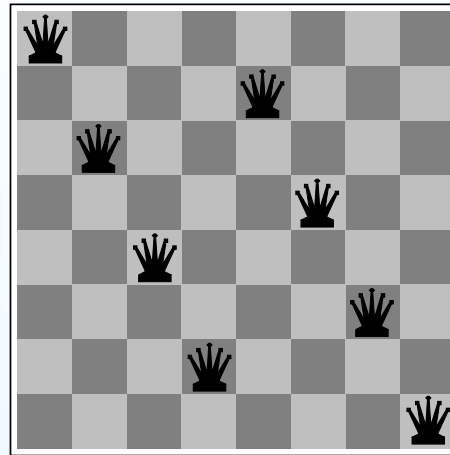
Start State

1	2	3
4	5	6
7	8	

Goal State

- **Stany:** integer: położenie każdego klocka.
- **Stan początkowy:** Każdy stan może nim być.
- **Akcja:**  $\{Lewo, Prawo, Gra, D\}$
- **Jak sprawdzić cel:** Sprawdź czy położenie końcowe zostało uzyskane.
- **Koszt ścieżki:** Liczba wykonanych akcji do osiągnięcia celu.

## Przykład: 8 hetmanów

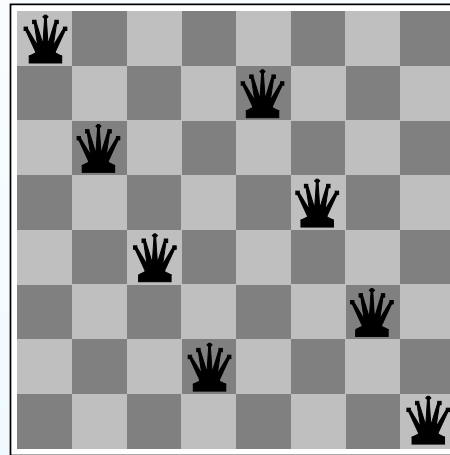


### Podejście inkrementacyjne

- Stany:
- Stan początkowy:
- Akcja:
- Jak sprawdzić cel:
- Koszt ścieżki:



## Przykład: 8 hetmanów

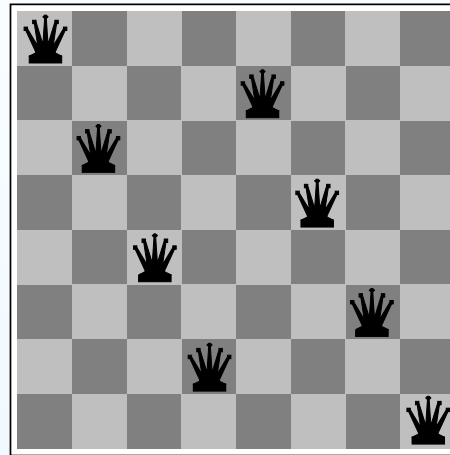


### Podójście inkrementacyjne

- **Stany:** dowolne połozenie od 0 do 8 hetmanów na szachownicy.
- **Stan początkowy:** Pusta szachownica.
- **Akcja:** Dodaj hetmana na pustym polu.
- **Jak sprawdzić cel:** Czy ustawione na szachownicy hetmany nie atakują się?
- **Koszt ścieżki:** zero - nie interesuje nas.

Istnieje  $64^8$  możliwych sekwencji do przebadania.

## Przykład: 8 hetmanów

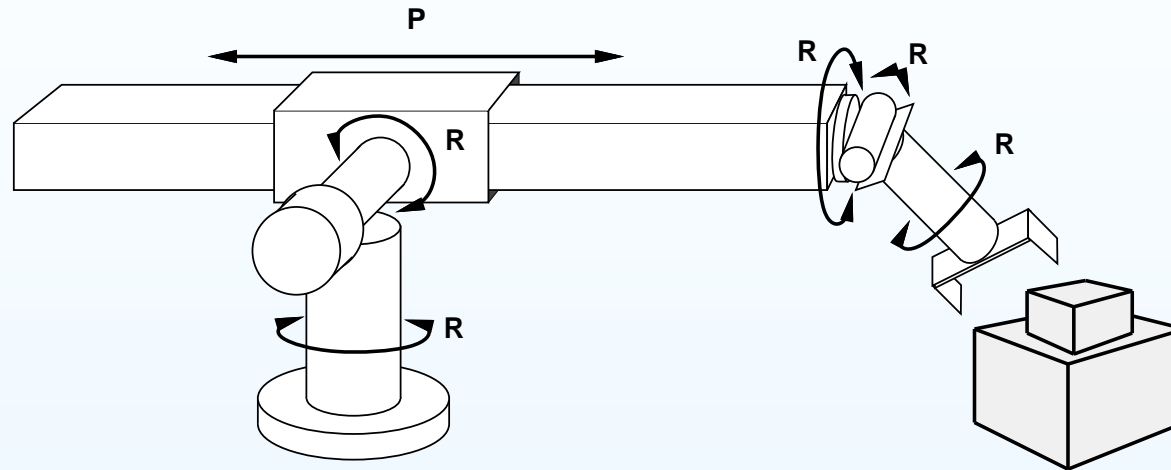


### Podejście inkrementacyjne

- **Stany:** Ustawienie od 0 do 8 hetmanów nie atakujących się.
- **Stan początkowy:** Pusta szachownica.
- **Akcja:** Dodawaj hetmana w kolejnych kolumnach zaczynając od lewej dbając, by ustawiaby hetman nie atakował innych.
- **Jak sprawdzić cel:** Czy ustawione na szachownicy hetmany nie atakują się?
- **Koszt ścieżki:** zero - nie interesuje nas.

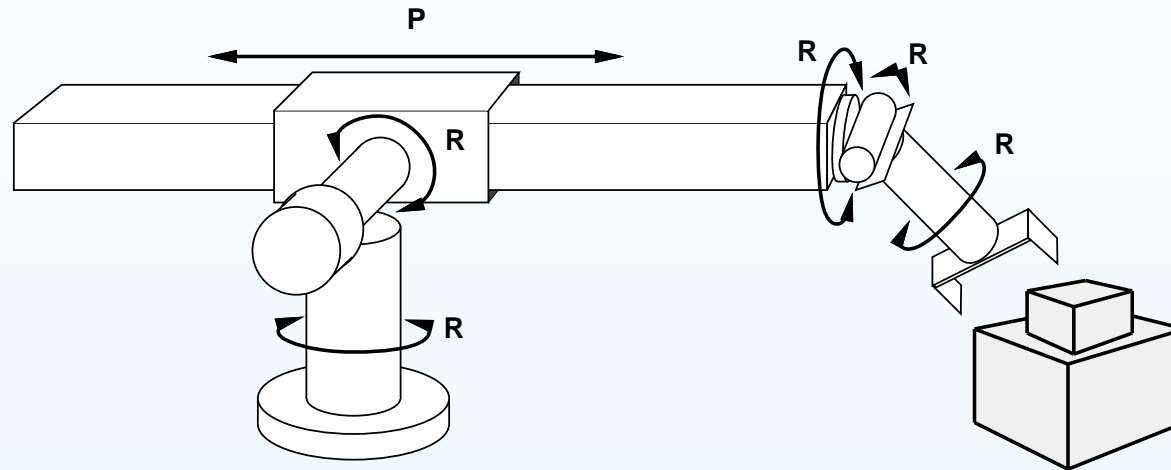
Istnieje 2057 możliwych sekwencji do przebadania. Zatem właściwe sformułowanie problemu może mieć duże znaczenie dla wyników.

## Przykład: robot montażowy



- Stany:
- Stan początkowy:
- Akcja:
- Jak sprawdzić cel:
- Koszt ścieżki:

## Przykład: robot montażowy



- **Stany:** koordynaty położenia ramienia robota; części elementów do montażu.
- **Stan początkowy:** Dowolne położenie ramienia i dowolna konfiguracja elementów.
- **Akcja:** Ciągłe obroty ramienia w przegubach.
- **Jak sprawdzić cel:** Pełen montaż.
- **Koszt ścieżki:** Czas wykonania.

# Podstawowy algorytm przeszukiwania

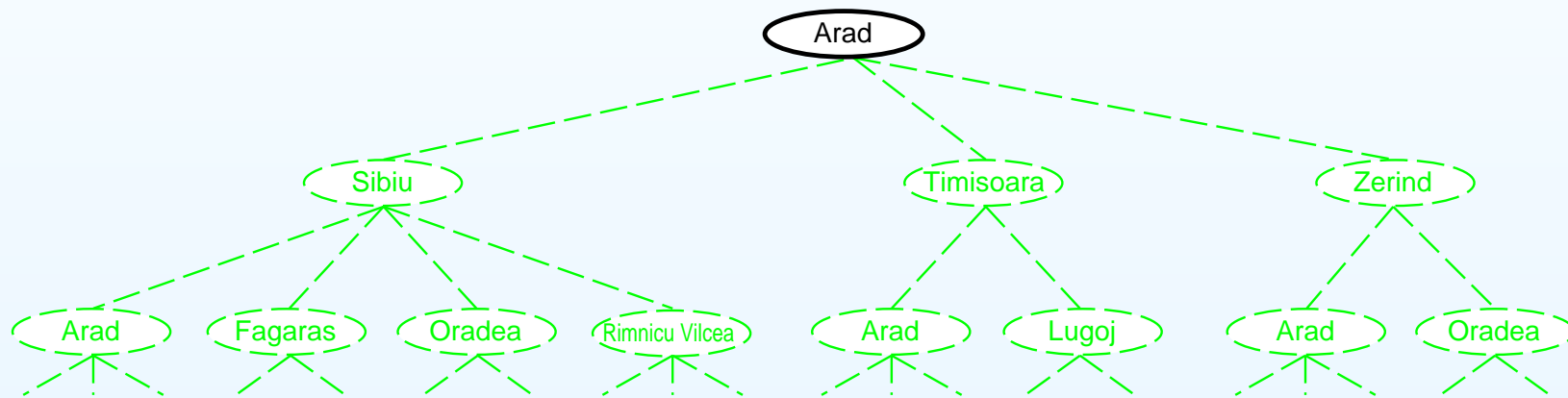
---

Jak znaleźć rozwiązanie na zdefiniowane problemy?

- Przeszukać przestrzeń stanów - należy pamiętać, że przestrzeń stanów zależy silnie od reprezentacji stanu.
- Przeszukiwanie przez jawne rozwijanie drzewa: korzeń drzewa = stan początkowy; węzły i liście generowane poprzez zastosowanie funkcji tworzącej potomków.
- Uogólniając, przeszukiwanie generuje graf, w którym ten sam węzeł może być wygenerowany na różnych ścieżkach.

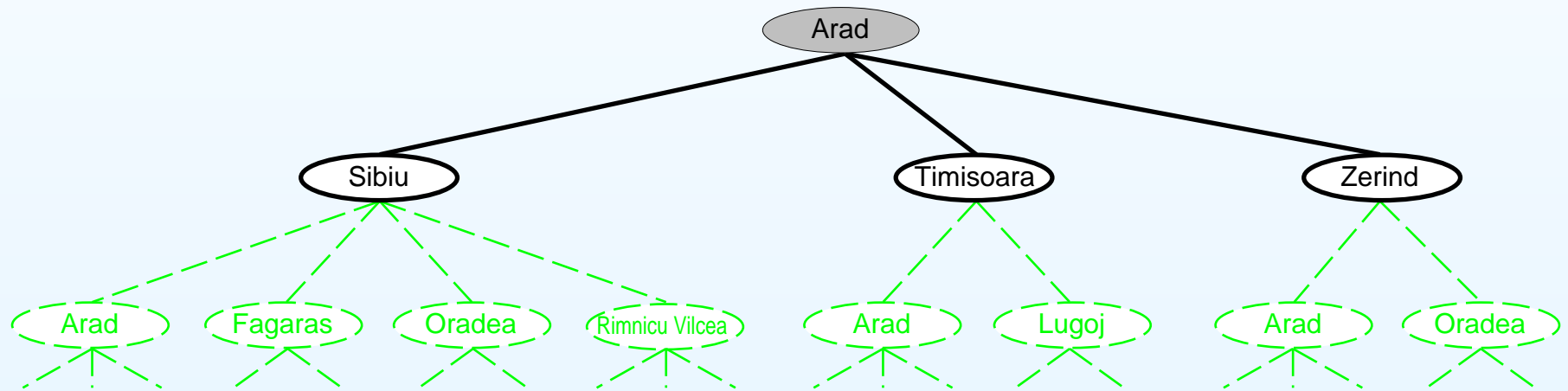
# Przykład przeszukiwania drzewa

Stan początkowy



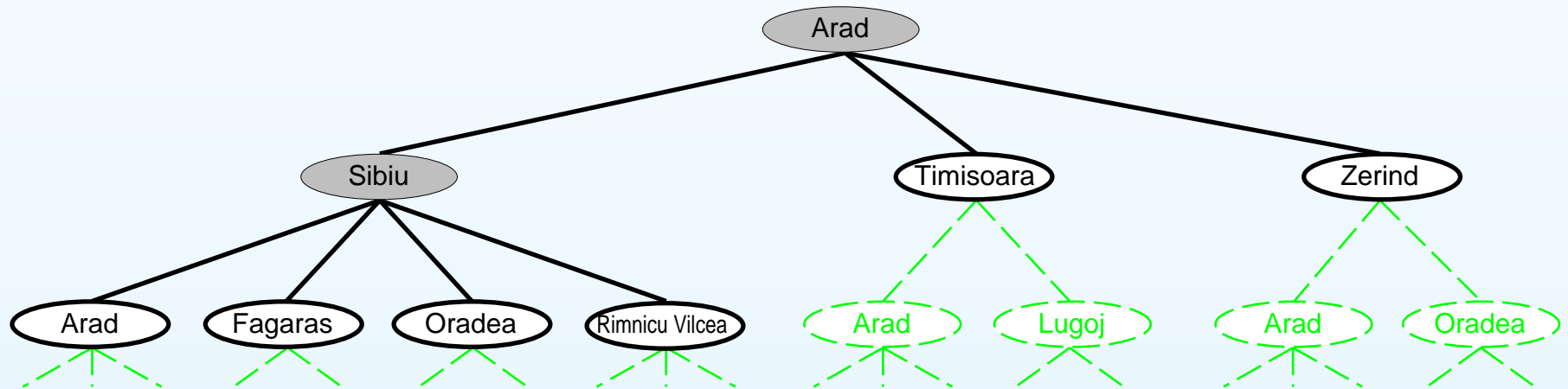
# Przykład przeszukiwania drzewa

Po rozwinięciu węzła Arad



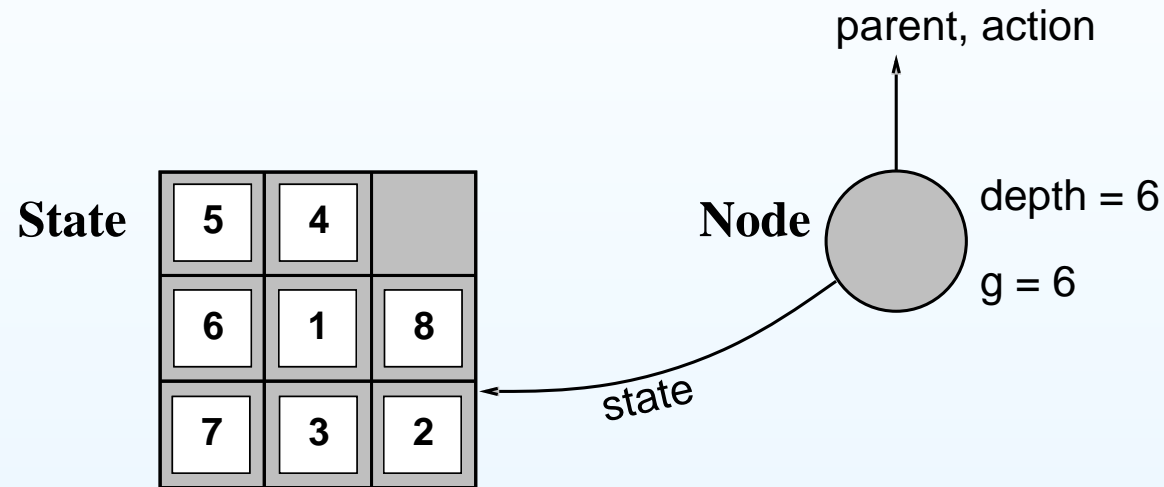
# Przykład przeszukiwania drzewa

Po rozwinięciu węzła Sibiu





# Implementacja stanów - węzły



**Stan** jest reprezentacją fizycznej konfiguracji.

**Węzeł** jest strukturą danych stanowiącą część drzewa przeszukiwań.

Węzeł ma rodzica, dzieci, głębokość, koszt ścieżki  $g(X)$ .

$node = \langle state, parent - node, action, path - cost, depth \rangle$

Węzły skrajne czekają na **rozwińnięcie** z użyciem funkcji generującej potomków węzła rozwijanego.

# Implementacja: przeszukiwanie drzewa

```
1 function TREE-SEARCH(problem,fringe) return a solution or failure
2   fringe = INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
3
4   do
5   if fringe is EMPTY then return failure
6   node = REMOVE-FIRST(fringe)
7   if GOAL-TEST(problem, STATE[node]) then return node
8   fringe = INSERT-ALL(EXPAND(node, problem), fringe)
```

```
1 function Expand( node, problem) returns a set of nodes
2   successors = the empty set
3   do for each action, result in Successor-Fn(problem, State[node])
4     s = a new Node
5     Parent-Node[s] = node; Action[s] = action; State[s] = result
6     Path-Cost[s] = Path-Cost[node] + Step-Cost(node, action, s)
7     Depth[s] = Depth[node] + 1
8     add s to successors
9   return successors
```

# Strategie przeszukiwania

Strategie (**node expanding**) są definiowane przez wybór **kolejności rozwijania węzłów**.

Kryteria oceny strategii przeszukiwania:

- **Zupełność (completeness)**: czy zawsze znajdzie rozwiązanie jeżeli ono istnieje?
- **Złożoność czasowa (time complexity)**: liczba węzłów generowana/rozwijana.
- **Obszar zajmowanej pamięci (space complexity)**: maksymalna liczba węzłów w pamięci.
- **Optymalność (optimality)**: czy zawsze znajdzie najmniej kosztowne rozwiązanie?

Złożoność obliczeniowa mierzona jest z uwzględnieniem:

- $b$  — maksymalnego współczynnika rozgałęzienia w drzewie,
- $d$  — głębokości najmniej kosztownego rozwiązania,
- $m$  — maksymalnej głębokości przestrzeni stanów (może być  $\infty$ ).

## Ślepe (uninformed) strategie przeszukiwania

---

Strategie ślepe używają tylko informacji podanej w definicji problemu.

- Przeszukiwanie wszerz — BFS (Breadth-first search).
- Przeszukiwanie o stałym koszcie — UCS (Uniform-cost search).
- Przeszukiwanie w głąb — DFS (Depth-first search).
- Przeszukiwanie w głąb z ograniczeniem (Depth-limited search).
- Iteracyjne zagłębianie (Iterative deepening search).