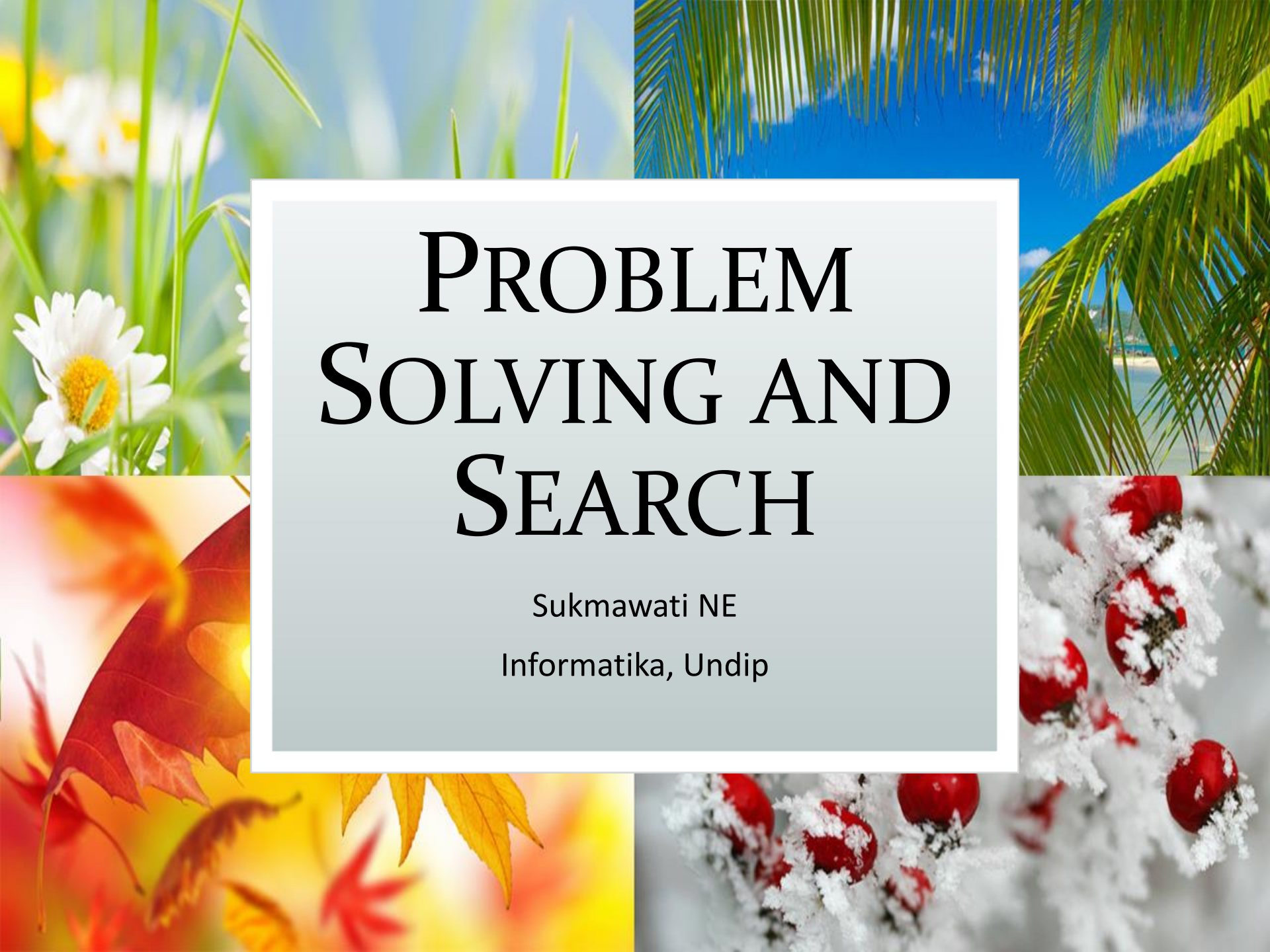


Problem Solving and Search

Sukmawati NE

Informatika, Undip

CONTENTS

- Problem Solving Agent
- Representasi Masalah : State Space
- Pencarian Solusi : Search
- Search Strategies

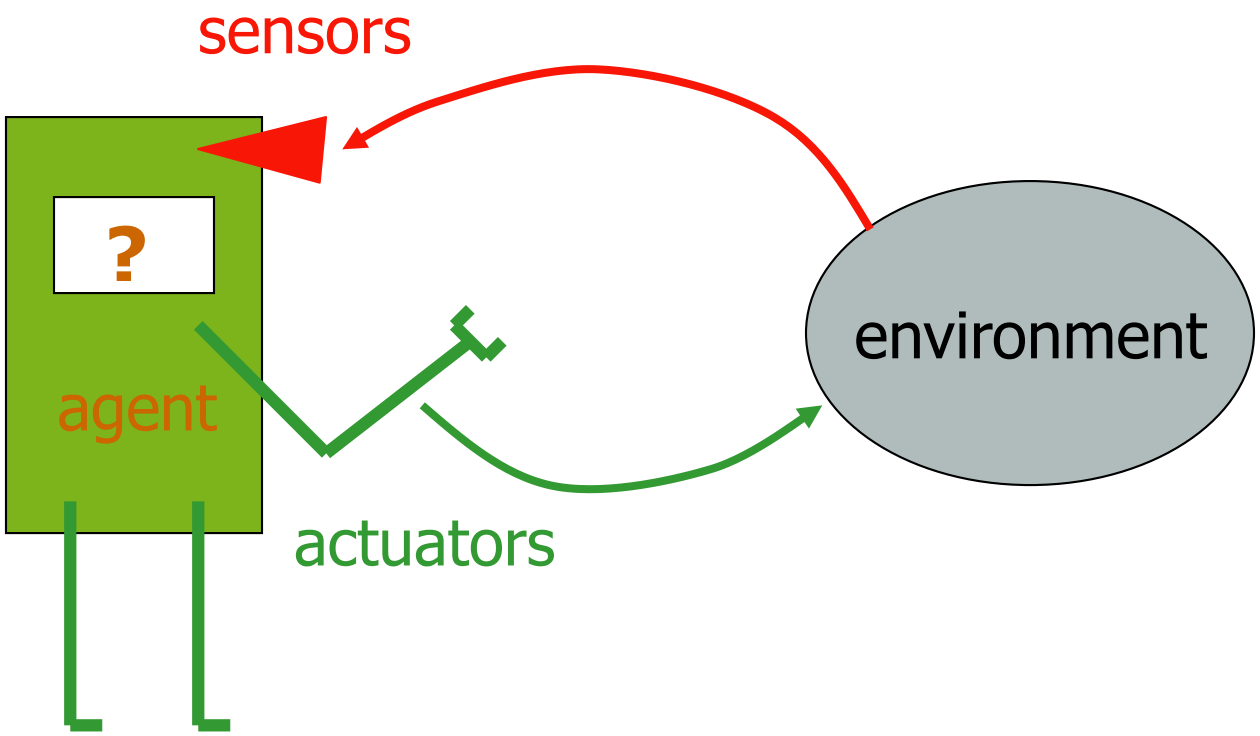
Ingat Kembali : Jenis-jenis Agent

- Simple reflex agents: hanya berdasarkan percept terakhir.
- Model-based reflex agents: memiliki representasi internal mengenai keadaan lingkungan.
- Goal-based agents: memiliki informasi mengenai tujuan, memilih tindakan yang mencapai tujuan.
- Utility-based agents: melakukan penilaian kuantitatif terhadap suatu keadaan lingkungan $\rightarrow$ utility function. Berkaitan dengan performance measure.
- Learning agents: belajar dari pengalaman, meningkatkan kinerja.

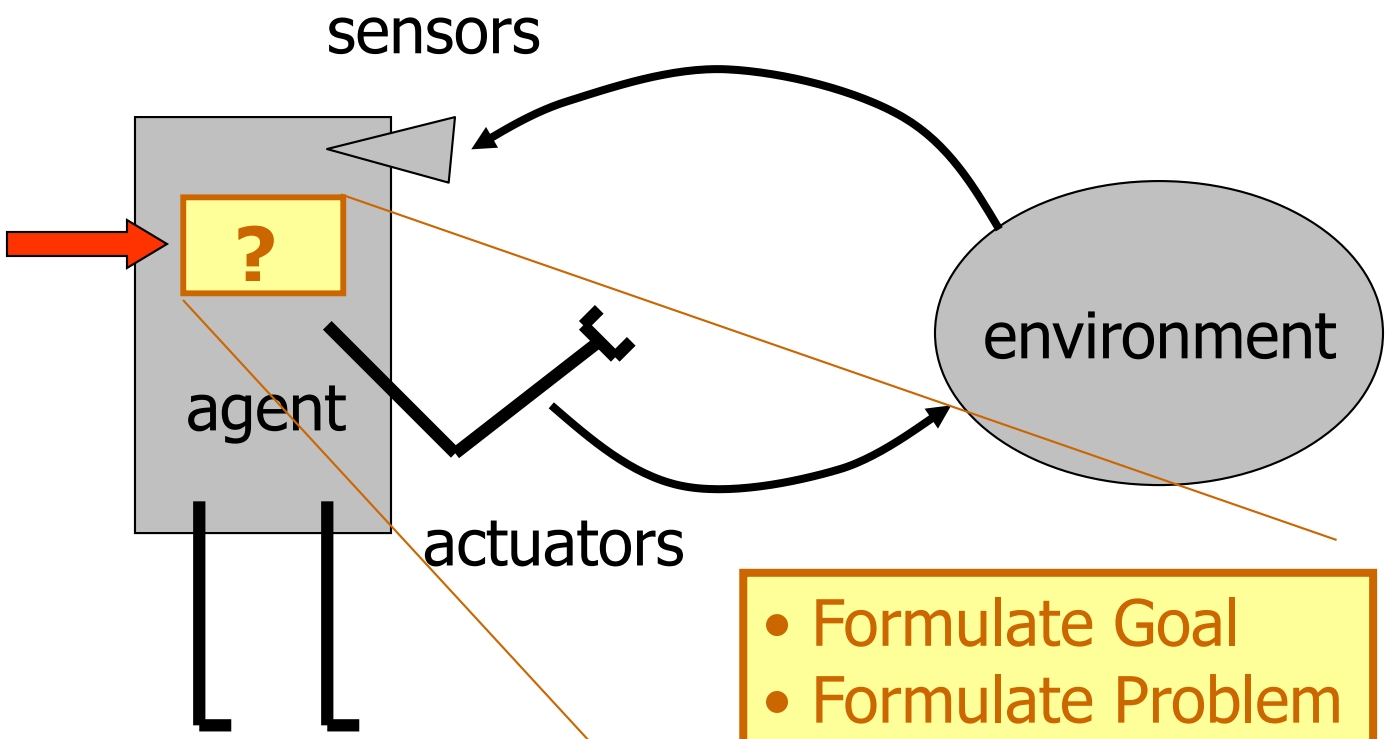
Problem-Solving Agent

- Problem Solving Agent merupakan salah satu jenis dari Goal-based Agent
- Goal-based agent: memiliki tujuan, memungkinkannya evaluasi tindakan dan memilih yang terbaik.
- Problem-solving agent menghasilkan solusi dalam bentuk serangkaian tindakan yang diambil untuk mencapai tujuan.
- Apa problem-nya? Apa solution-nya?

Problem-Solving Agent



Problem-Solving Agent



- Formulate Goal
- Formulate Problem
  - States
  - Actions
- Find Solution

Mekanisme kerja Problem-Solving Agent

- Perumusan tujuan (*goal formulation*):
  - Tentukan tujuan yang ingin dicapai
- Perumusan masalah (*problem formulation*):
  - Tentukan keadaan (*state*) dan tindakan (*action*) yang dipertimbangkan dalam mencapai tujuan
- Pencarian solusi masalah (*searching*):
  - Tentukan rangkaian tindakan yang perlu diambil untuk mencapai tujuan
- Pelaksanaan solusi (*execution*):
  - Laksanakan rangkaian tindakan yang sudah ditentukan di tahap sebelumnya

PROBLEM-SOLVING AGENT PROGRAM

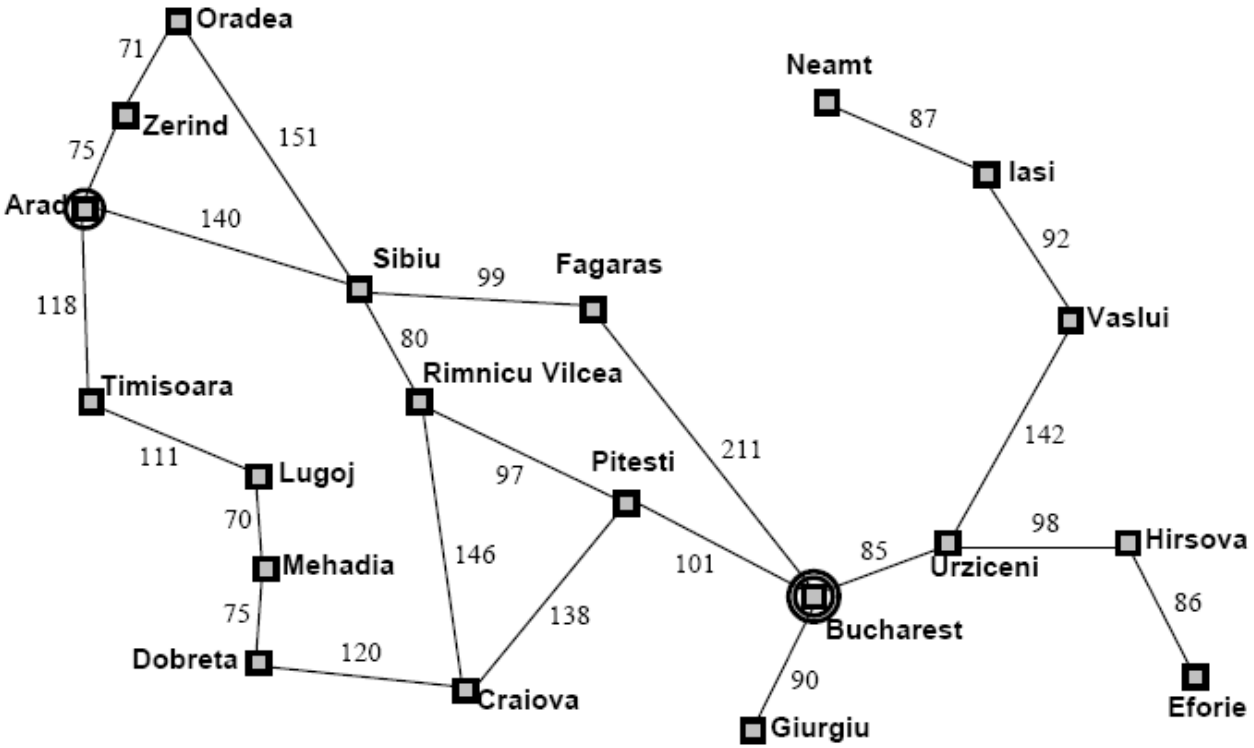
```
Function Simple-Problem-Solving-Agent( percept ) returns action

Inputs: percept          a percept
Static: seq              an action sequence initially empty
        state            some description of the current world
        goal             a goal, initially null
        problem          a problem formulation

state <- UPDATE-STATE( state, percept )
if seq is empty then do
  goal <- FORMULATE-GOAL( state )
  problem <- FORMULATE-PROBLEM( state, goal )
  seq <- SEARCH( problem )                          # SEARCH
action <- RECOMMENDATION ( seq )                    # SOLUTION
seq <- REMAINDER( seq )
return action                                       # EXECUTION
```

SIFAT PROBLEM-SOLVING AGENT

- Biasanya problem-solving agent mengasumsikan bahwa environment-nya:
  - Fully observable :
    - Agent mengetahui initial state
  - Deterministic :
    - Agent dapat merencanakan sejumlah action sedemikian hingga masing-masing action tersebut diawali dengan sebuah state penghubung
  - Sequential :
    - Terurut
  - Static :
    - Rencana yang disusun relatif sama
  - Discrete :
    - Agent memiliki pilihan yang berhingga

CONTOH : TURIS DI RUMANIA

- Suatu “tourist agent” yang sedang berlibur di Rumania, kini berada di Arad. Besok, dia harus terbang dari bandara Bucharest.



CONTOH : TURIS DI RUMANIA

- Perumusan tujuan:
  - berada di Bucharest
- Perumusan masalah:
  - Tindakan (action): menyetir dari kota ke kota
  - Keadaan (state): kota-kota di Rumania
- Pencarian solusi:
  - rangkaian kota yang dituju,
  - mis: Arad, Sibiu, Fagaras, Bucharest

Representasi Masalah : State Space

- State Space (Ruang Keadaaan):
  - Suatu Ruang yang berisi semua keadaan yang mungkin
- Untuk mendeskripsikan masalah :
  - Initial State
  - Actions and Successor Function
  - Goal test
  - Path Cost

Representasi Masalah : State Space

- Initial state:
  - keadaan awal di mana si agent mulai,
  - mis: BeradaDi(Arad)
- Possible actions:
  - tindakan yang dapat dilakukan si agent,
  - mis: Nyetir (Arad, Zerind).
- Sebuah successor function S menentukan untuk suatu state X, himpunan tindakan yang mungkin diambil beserta state yang dihasilkan. Contoh:
  - X = BeradaDi(Arad)
  - S(X) = {< Nyetir (Arad, Zerind), BeradaDi(Zerind) >, ...}
- Initial state dan successor function mendefinisikan state space:
  - himpunan semua state yang dapat dicapai dari initial state.
  - Dapat direpresentasikan sebagai graph.
  - Path dalam state space adalah serangkaian state (dihubungkan serangkaian action)

MENELUSURI SEBUAH STATE SPACE

- Goal test: penentuan apakah suatu state adalah tujuan yang ingin dicapai.
  - Eksplisit: himpunan goal state, mis: {BeradaDi(Bucharest)}.
  - Implisit: deskripsi tujuan, mis: dalam catur ➔ skak mat.
- Path cost function: sebuah fungsi yang memberikan nilai numerik terhadap setiap path. Fungsi ini merefleksikan performance measure si agent.
- Asumsi path cost function = Σ step cost: cost action a dari state x ke y: c(x, a, y).
- Sebuah solusi adalah path dari initial state ke goal state.
- Sebuah solusi optimal adalah solusi dengan path cost function minimal.

Contoh : Penuangan Air di Teko

- Diberikan dua buah teko berkapasitas 4 galon (teko A)dan 3 galon (teko B) (tanpa skala ukuran).
  - Bagaimana mendapatkan tepat 2 galon pada teko A tersebut?

CONTOH : PENUANGAN AIR DI TEKO

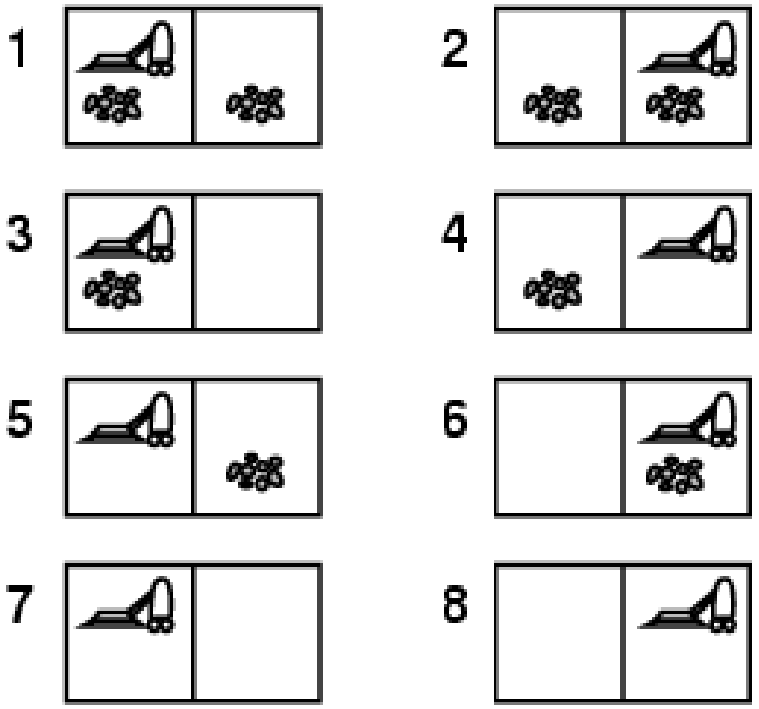
- Initial state:
  - Teko dalam kondisi kosong
  - Direpresentasikan dalam pasangan (0 , 0)
- Goal state:
  - Teko A berisi 2 galon
  - Direpresentasikan (2 , y)
- Path cost:
  - Asumsikan step cost sama untuk setiap actionnya
  - Misalkan bernilai 1 setiap langkahnya
  - Path cost adalah jumlah step costs

CONTOH : PENUANGAN AIR DI TEKO

- Actions and Successor Function
  - Mengisi Teko
    - (x , y) -> (4 , y) # Mengisi teko A
    - (x , y) -> (x , 3) # Mengisi teko B
  - Mengosongkan teko
    - (x , y) -> (0 , y) # Mengosongkan teko A
    - (x , y) -> (x , 0) # Mengosongkan teko B
  - Menuangkan isi satu teko ke teko lainnya
    - (x , y) -> (0 , x+y) or (x+y-3, 3) # Menuangkan seluruh air dari teko A ke teko B atau Menuangkan air dari teko A ke teko B sampai teko B penuh
    - (x , y) -> (x+y , 0) or (4, x+y-4) # Menuangkan seluruh air dari teko B ke teko A atau Menuangkan air dari teko B ke teko A sampai teko A penuh

CONTOH : VACUUMCLEANER WORLD

- Vacuum Cleaner Agent yang bertugas membersihkan debu dilantai. Misalkan di berikan world states sebagai berikut:
- Ada 2 jenis :
  - Single-state Problem
    - Start in #5
    - Solution : [Kanan, Bersih]
  - Multiple-state Problem
    - Start in {1,2,3,4,5,6,7,8}
    - Solution : [Kanan, Bersih, Kiri, Bersih]



CONTOH : VACUUMCLEANER WORLD

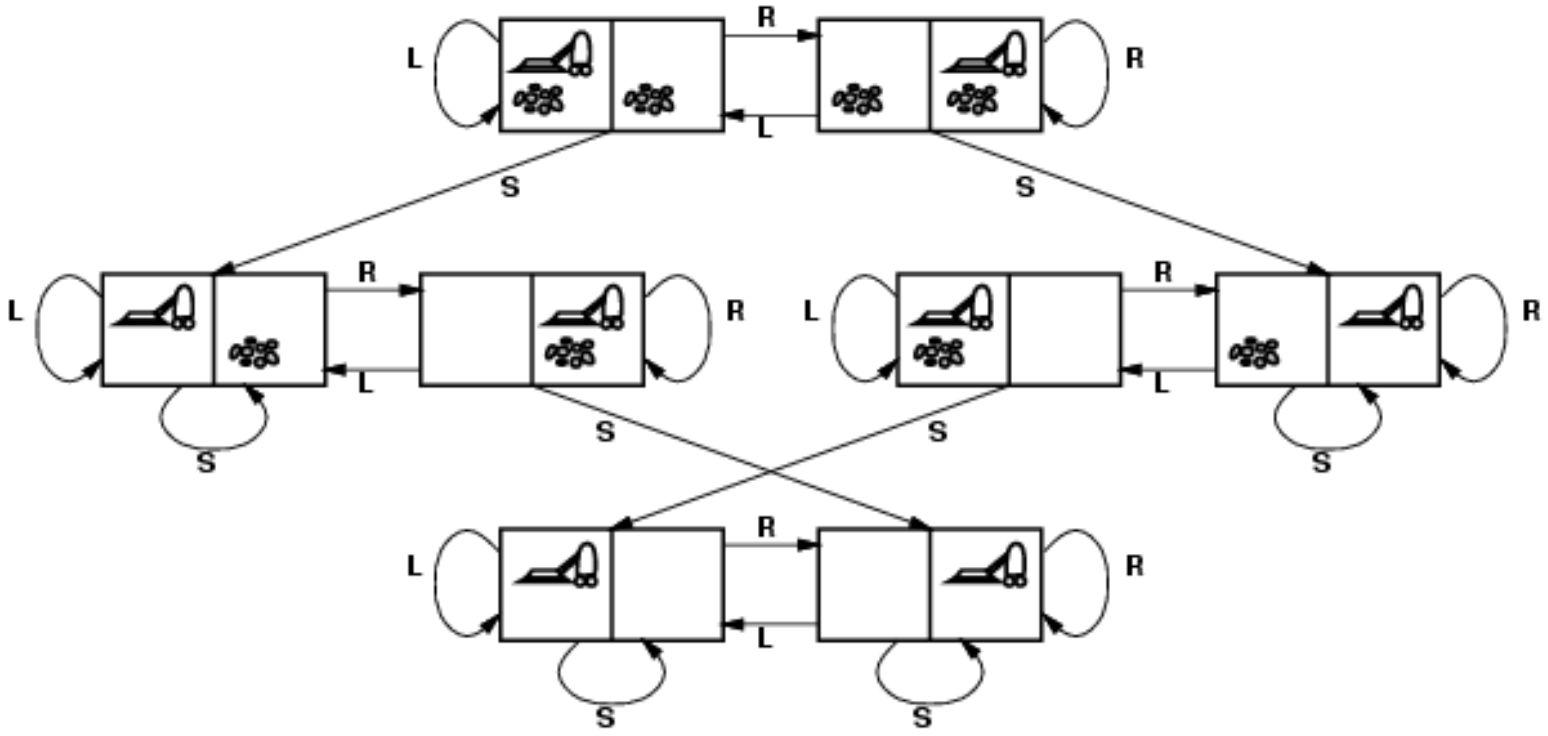
- States?
- Actions?
- Goal test?
- Path cost?

CONTOH : VACUUMCLEANER WORLD

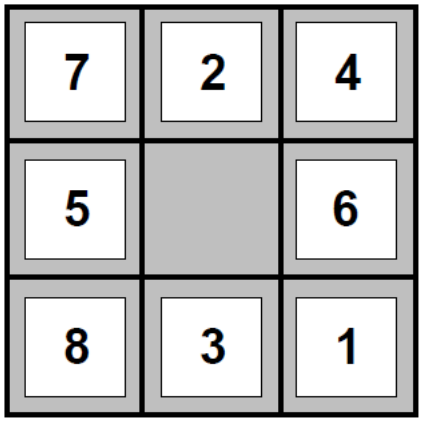
- States : Lokasi Agent dan Status Debu
- Actions: Ke Kanan (R), Ke Kiri (L), Bersihkan/Sedot (S)
- Goal test: Tidak ada debu di semua lokasi
- Path cost: 1 per action

VACUUM WORLD STATE SPACE GRAPH

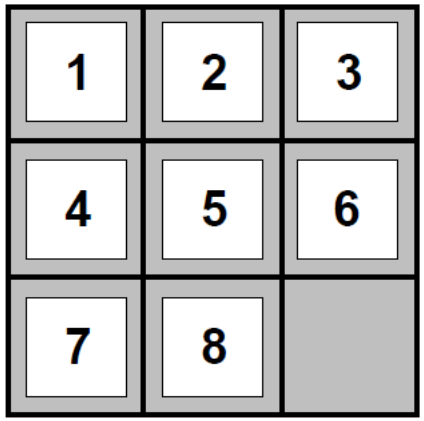
- Successor function mendefinisikan state space sbb:



CONTOH : 8 - PUZZLE



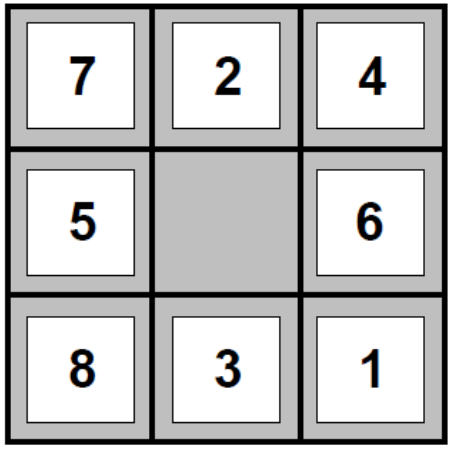
Start State



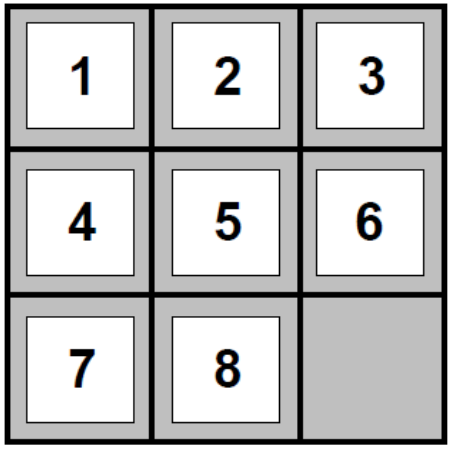
Goal State

- States?
- Actions?
- Goal test?
- Path cost?

CONTOH : 8 - PUZZLE



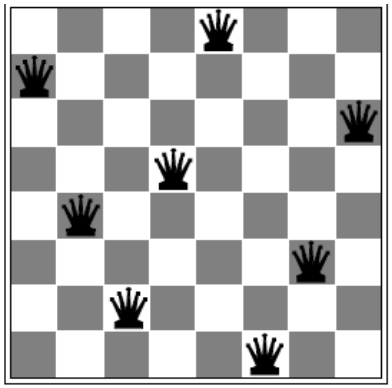
Start State



Goal State

- State: lokasi 8 buah angka dalam matriks 3x3
- Possible action: *Kiri*, *Kanan*, *Atas*, *Bawah*
- Goal test: apakah konfigurasi angka seperti goal state di atas
- Path cost: asumsi step cost = 1. Path cost = jumlah langkah dalam path.

CONTOH : 8-QUEENS PROBLEM



Letakkan 8 bidak menteri (*queen*!) sedemikian sehingga tidak ada yang saling "makan" (menteri bisa makan dalam satu baris, kolom, diagonal).

- State: Papan catur dengan n bidak menteri, $0 \leq n \leq 8$.
- Initial state: Papan catur yang kosong.
- Possible action: Letakkan sebuah bidak menteri di posisi kosong.
- Goal test: 8 bidak menteri di papan, tidak ada yang saling makan.

Masalah state space... combinatorial explosion!

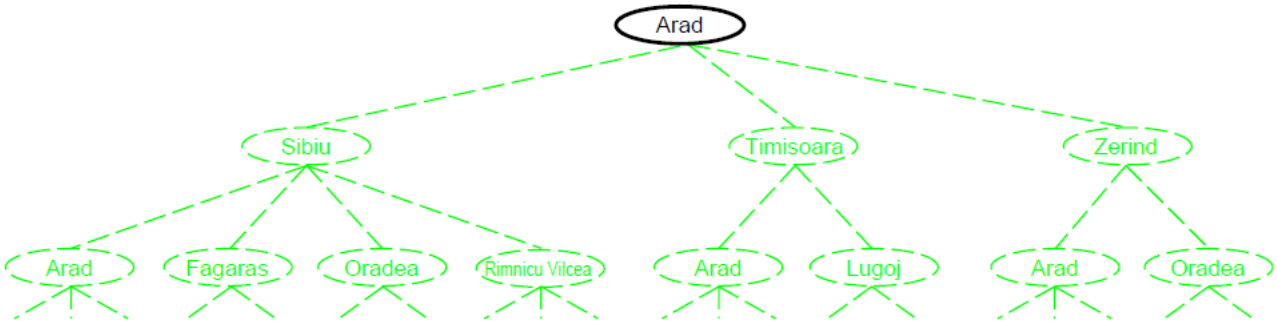
- Dengan definisi masalah demikian, ada $64 \times 63 \times \ldots \times 57 \approx 1.8 \times 10^{14}$ path!
- Mustahil kita selesaikan dengan komputer tercanggih apapun. Definisi masalah bisa diperjelas:
  - State: Papan catur dengan n bidak menteri, $0 \leq n \leq 8$, satu per kolom di n kolom paling kiri.
  - Possible action: Letakkan sebuah bidak menteri di posisi kosong di kolom paling kiri yang belum ada bidaknya sehingga tidak ada yang saling makan.
- State space sekarang ukurannya tinggal 2057, dan mudah dipecahkan.
- Perumusan masalah yang tepat bisa berakibat drastis!

PENCARIAN SOLUSI : SEARCH

- Mencari Solusi dapat melalui Search Tree
  - Setelah merumuskan masalah → cari solusinya menggunakan sebuah search algorithm
  - Search tree merepresentasikan *state space*.
  - Search tree terdiri dari kumpulan node: struktur data yang merepresentasikan suatu *state* pada suatu *path*, dan memiliki parent, children, depth, dan path cost.
  - Root node merepresentasikan initial state.
  - Penerapan *successor function* terhadap (*state* yang diwakili) *node* menghasilkan *children* baru → ini disebut node expansion.
  - Kumpulan semua node yang belum di-*expand* disebut fringe (pinggir) sebuah *search tree*.

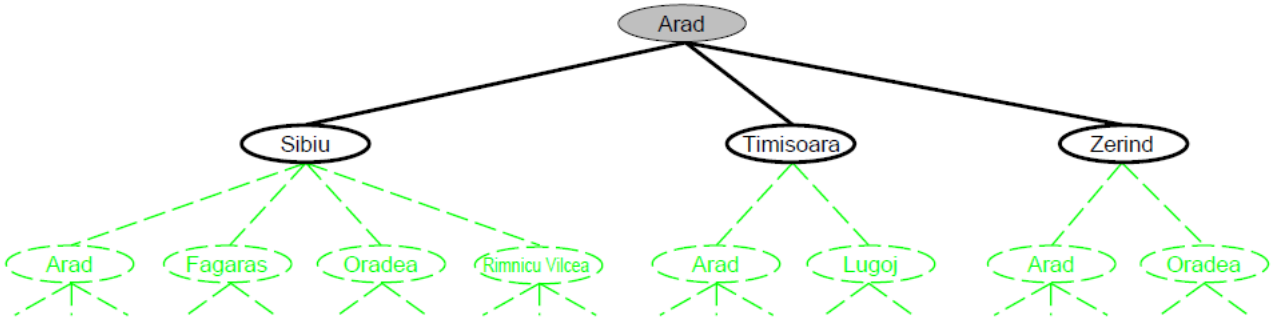
Contoh Penelusuran Search Tree

- Masalah : Turis Rumania
  - Mulai dari root node (Arad) sebagai current node.
  - Lakukan *node expansion* terhadapnya.
  - Pilih salah satu node yang di-expand sebagai current node yang baru. Ulangi langkah sebelumnya.



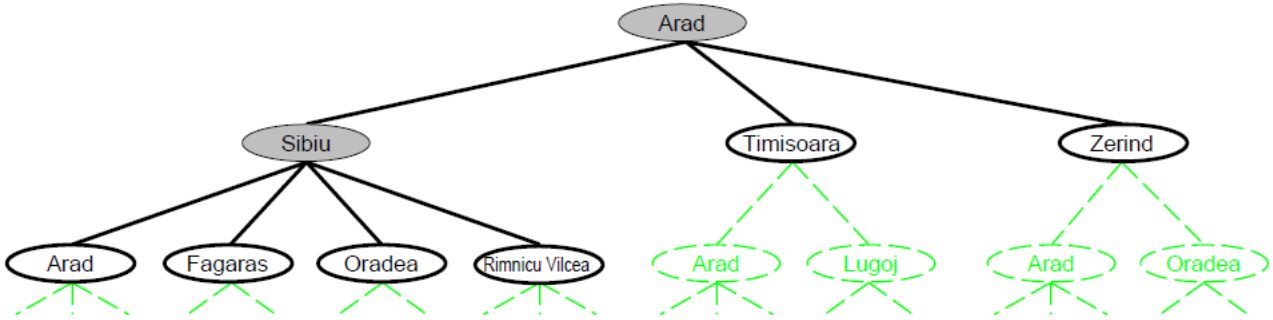
CONTOH PENELUSURAN SEARCH TREE

- Masalah : Turis Rumania
  - Mulai dari root node (Arad) sebagai current node.
  - Lakukan *node expansion* terhadapnya.
  - Pilih salah satu node yang di-expand sebagai current node yang baru. Ulangi langkah sebelumnya.



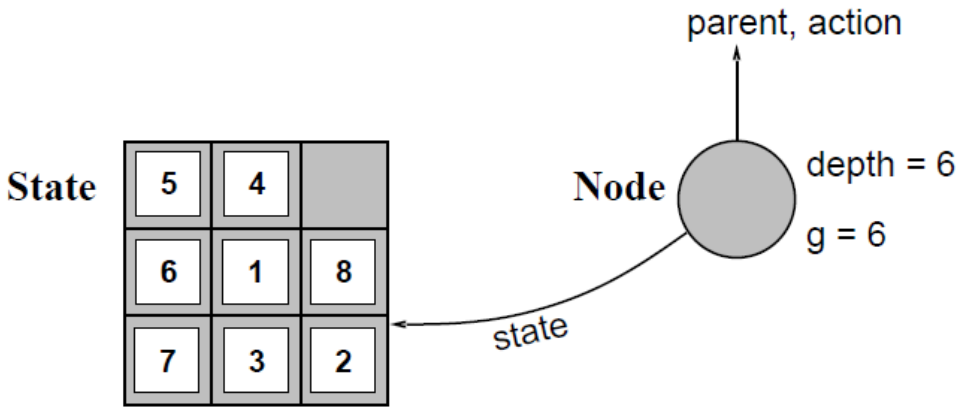
CONTOH PENELUSURAN SEARCH TREE

- Masalah : Turis Rumania
  - Mulai dari root node (Arad) sebagai current node.
  - Lakukan *node expansion* terhadapnya.
  - Pilih salah satu node yang di-expand sebagai current node yang baru. Ulangi langkah sebelumnya.



STATE VS NODE

- Sebuah state merepresentasikan abstraksi keadaan nyata dari masalah.
- Sebuah node adalah struktur data yang menjadi bagian dari search tree.
- State tidak memiliki parent, children, depth, path cost!
- Node = state pada path tertentu. Dua node berbeda bisa mewakili state yang sama!



LATIHAN

- Bagaimana solusi dari masalah penuangan air pada teko?

Strategi Pencarian

- Terdapat berbagai jenis strategi untuk melakukan search.
- Semua strategi ini berbeda dalam satu hal: urutan dari node expansion.
- Search strategy di-evaluasi berdasarkan:
  - completeness: apakah solusi (jika ada) pasti ditemukan?
  - time complexity: jumlah node yang di-expand.
  - space complexity: jumlah maksimum node di dalam memory.
  - optimality: apakah solusi dengan minimum cost pasti ditemukan?
- Time & space complexity diukur berdasarkan
  - *b* - branching factor dari search tree
  - *d* - depth (kedalaman) dari solusi optimal
  - *m* - kedalaman maksimum dari search tree (bisa infinite!)

UNINFORMED SEARCH STRATEGIES

- Uninformed strategy hanya menggunakan informasi dari definisi masalah.
- Bisa diterapkan secara generik terhadap semua jenis masalah yang bisa direpresentasikan dalam sebuah state space.
- Ada beberapa jenis:
  - Breadth-first search
  - Uniform-cost search
  - Depth-first search
  - Depth-limited search
  - Iterative-deepening search

DIPELAJARI NEXT MINGGU DEPAN!!!

SOAL-SOAL LATIHAN

- Definisikanlah Initial State, successor function, goal test, dan path cost function untuk masalah berikut:

  Seekor monyet yang tingginya 3 kaki berada di dalam sebuah ruangan yang tingginya 8 kaki. Di dalam ruangan tersebut terdapat beberapa buah pisang yang dilekatkan ke langit-langit. Tentunya si monyet tadi ingin mendapatkan pisang-pisang tersebut. Untungnya, di dalam ruangan ini juga terdapat dua kardus besar yang masing-masing tingginya 3 kaki. Kardus ini dapat dipindahkan, ditumpuk, dan didaki oleh si monyet tadi.

SOAL-SOAL LATIHAN

- Alkisah di sebuah penjara di daerah Jakarta Utara terdapat 3 orang polisi dan 3 orang narapidana. Ketika dilanda musibah banjir, keenam orang ini terpaksa dipindahkan ke suatu tempat lain menggunakan sebuah perahu karet. Perahu karet ini memiliki daya tampung maksimal 2 orang. Masalahnya, jika terjadi keadaan di mana jumlah narapidana melebihi jumlah polisi di suatu lokasi (baik itu di penjara lama, di dalam perahu karet, maupun di tempat yang baru), maka para napi akan mengeroyok para polisi!
- (Asumsikan bahwa kalau tidak ada seorangpun polisi di suatu lokasi, meskipun ada banyak napi, mereka tidak akan berani kabur!).
- Bagaimana caranya agar keenam orang ini bisa dipindahkan ke tempat yang baru ini tanpa ada polisi yang dikeroyok?
- (a) Definisikan masalahnya dengan jelas, seperti yang anda lakukan untuk masalah di atas.
- (b) Cobalah gambarkan state space yang lengkap.
- (c) Pilih sebuah strategi search dan gunakanlah untuk mencari sebuah solusi.