Summary of agents kinds

- A **rational** agent is one that does the right thing
- The **omniscience** of an agent knows the *actual outcome assuming infinite knowledge*
- The **autonomy** of an agent is the extent to which its behaviour is determined by its own experience

Environment types

Environments in which agents operate can be defined in different ways

- Fully observable (vs. partially observable):
  - In a fully observable environment all of the environment relevant to the action being considered is observable. In such environments, the agent does not need to keep track of the changes in the environment. Ex. Chess
  - In a partially observable environment, the relevant features of the environment are only partially observable.

Environment types

- Deterministic (vs. stochastic):
  - The next state of the environment is completely determined by the current state and the action executed by the agent.
  - If an element of interference or uncertainty occurs then the environment is stochastic.
  - (If the environment is deterministic; i.e. wholly determined by the preceding state except for the actions of other agents, then the environment is strategic)

Environment types

- Episodic (vs. sequential):
  - The agent's experience is divided into atomic "episodes" (each episode consists of the agent perceiving and then performing a single action), and the choice of action in each episode depends only on the episode itself.
  - In a sequential environment, the agent engages in a series of connected episodes.

Environment types

- Episodic (vs. sequential):
  - The agent's experience is divided into atomic "episodes" (each episode consists of the agent perceiving and then performing a single action), and the choice of action in each episode depends only on the episode itself.
  - In a sequential environment, the agent engages in a series of connected episodes.

Environment types

- Static (vs. dynamic):
  - A static environment does not change while the agent is thinking.
  - The time as an agent deliberates is irrelevant.
  - The agent doesn't need to observe the world during deliberation.
  - A Dynamic Environment changes over time independent of the actions of the agent (if an agent does not respond in a timely manner, this counts as a choice to do nothing)

Environment types

- Discrete (vs. continuous):
  - A limited number of distinct, clearly defined percepts and actions is discrete, otherwise it is continuous.
- Single agent (vs. multiagent):
  - An agent operating by itself in an environment.
  - If the environment contains other intelligent agents, the agent needs to be concerned about strategic

Knowledge of Environment

- Knowledge of Environment (World)
  - Different to sensory information from environment
- World knowledge can be (pre)-programmed in
  - Can also be updated/inferred by sensory information
- Choice of actions informed by knowledge of world
  - Current state of the world
  - Previous states of the world
  - How its actions change the world
- Example: Chess agent
  - World knowledge is the board state (all the pieces)
  - Sensory information is the opponents move
  - Its moves also change the board state

The real world is partially observable, stochastic, sequential, dynamic, continuous, multi-agent

Specifying the task environment PEAS

- In designing an agent, the first step must always be to specify the task environment, i.e. PEAS
- PEAS: Performance measure, Environment, Actuators, Sensors
- Consider a medical diagnosis agent

Agent Type	Performance Measure	Environment	Actuators	Sensors
Medical diagnosis system	Healthy patient, minimize costs, lawsuits	Patient, hospital, staff	Display questions, tests, diagnoses, treatments, referrals	Keyboard entry of symptoms, findings, patient's answers

PEAS: Taxi Deriver

- Consider the task of designing an automated taxi driver:
- A fully automated taxi is beyond the capabilities of existing technology
- The fully driving task is extremely open-ended

Agent Type	Performance Measure	Environment	Actuators	Sensors
Taxi driver	Safe: fast, legal, comfortable trip, maximize profits	Roads, other traffic, pedestrians, customers	Steering, accelerator, brake, signal, horn, display	Cameras, sonar, speedometer, GPS, odometer, accelerometer, engine sensors, keyboard

Figure 2.4 PEAS description of the task environment for an automated taxi.

PEAS: Tutor

- Agent: Interactive English tutor
- Performance measure: Maximize student's score on test
- Environment: Set of students
- Actuators: Screen display (exercises, suggestions, corrections)
- Sensors: Keyboard

Internet shopping agent

Performance measure?? price, quality, appropriateness, efficiency

Environment?? current and future WWW sites, vendors, shippers

Actuators?? display to user, follow URL, fill in form

Sensors?? HTML pages (text, graphics, scripts)

Conclusion

- AI deals with exciting but hard problems. A goal of AI is to build intelligent agents that act so as to optimize performance.
- An **agent perceives** and acts in an environment, has an **architecture,** and is implemented by an **agent program.**
- An **ideal agent** always chooses the action which **maximizes its expected performance**
- An **autonomous** agent uses its **own experience** rather than built-in knowledge of the environment by the designer.

The role of search in AI

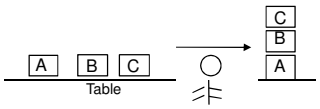
- There is a hypothesis: intelligent action can be reduced to search.
- While intelligence is certainly more than search, the hypothesis is attractive: search is well understood, easily mechanized, manageable, and so on.
- Formally represented knowledge can also be easily used in search.
- Search means systematic traversal of a space of possible solutions of a problem.

The role of search

- A search space is usually a graph (often simple as a tree).
  - A node represents a partial solution.
  - An edge represents a step in the construction of a solution.
- The purpose of search may be:
  - to find a path in the graph from a start node to a goal node (that is, from an initial to a final situation),
  - to find a goal node.

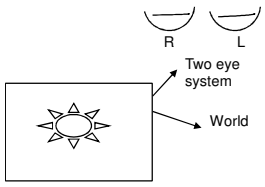
Search Ex1: Planning

- (a) which block to *pick*, (b) which to *stack*, (c) which to *unstack*, (d) whether to *stack* a block or (e) whether to *unstack* an already stacked block. These options have to be searched in order to arrive at the right sequence of actions.



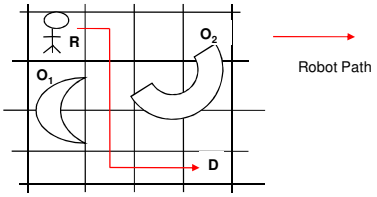
Search Ex2: Vision

- A search needs to be carried out to find which point in the image of *L* corresponds to which point in *R*. Naively carried out, this can become an $O(n^2)$ process where *n* is the number of points in the image.



Search Ex3: Robot Path Planning

- searching amongst the options of moving ***L***eft, ***R***ight, ***U***p or ***D***own. Additionally, each movement has an associated cost representing the relative difficulty of each movement. The search then will have to find the *optimal*, *i.e.*, the *least cost* path.



Search Ex4: Natural Language Processing

- search among many combinations of parts of speech on the way to deciphering the meaning. This applies to every level of processing- *syntax, semantics, pragmatics* and *discourse*.

The	man	would	like	to	play.
Noun		Verb	Preposition	Verb	

Search Ex5: Expert Systems

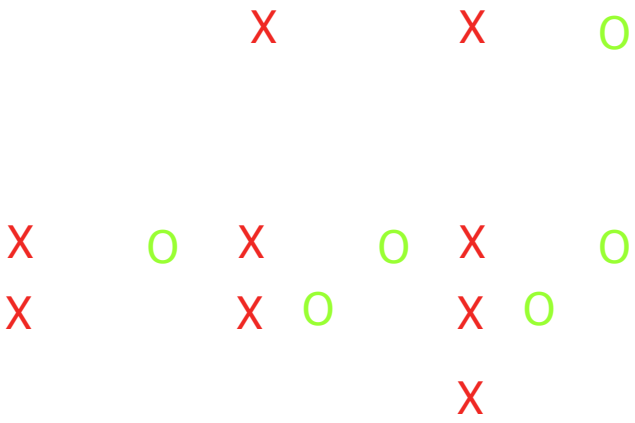
Search among rules, many of which can apply to a situation:

If-conditions
the infection is primary-bacteremia
AND the site of the culture is one of the sterile sites
AND the suspected portal of entry is the gastrointestinal tract
THEN
there is suggestive evidence (0.7) that infection is bacteroid

(from MYCIN)

Search Ex6: Game Playing

Path to goal isn't quite right.



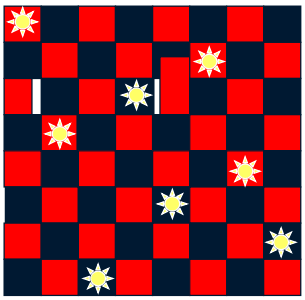
Search Space Problems

- State Space : Graph of states (Express constraints and parameters of the problem)
- Operators : Transformations applied to the states.
- Start state : S_0 (Search starts from here)
- Goal state(s) : $\{G\}$ - Search terminates here.
- Cost : Effort involved in using an operator.
- Optimal path : Least cost path

Search Space Problems

- State Space : Graph of states (Express constraints and parameters of the problem)
- Operators : Transformations applied to the states.
- Start state : S_0 (Search starts from here)
- Goal state(s) : $\{G\}$ - Search terminates here.
- Cost : Effort involved in using an operator.
- Optimal path : Least cost path

8-queen problem



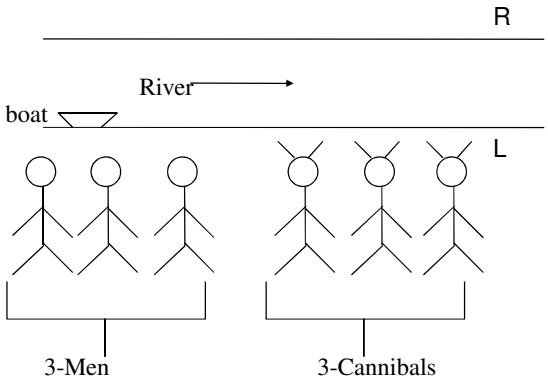
States: Any arrangements of 0 – 8 queens on board

Initial State: No queens on board

Successor function: Add a queen to any empty square

Goal State: 8 queens on the board, unattacked

Men and Cannibals



Constraints

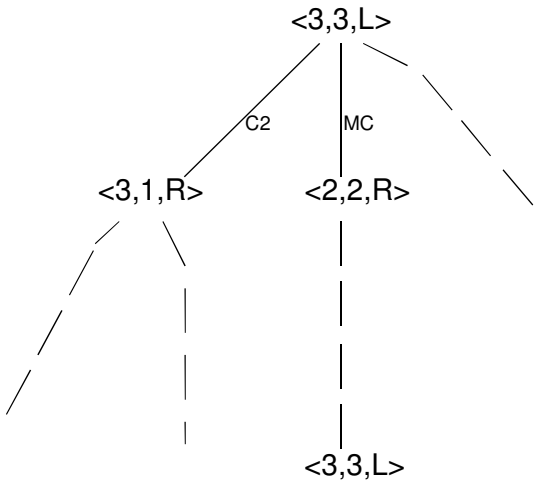
- The boat can carry at most 2 people
- On no bank should the cannibals outnumber the Men
- Move all people to the other side of the river

Men and Cannibals

State : *<#M, #C, P>*
#M = Number of men on bank *L*
#C = Number of cannibals on bank *L*
P = Position of the boat

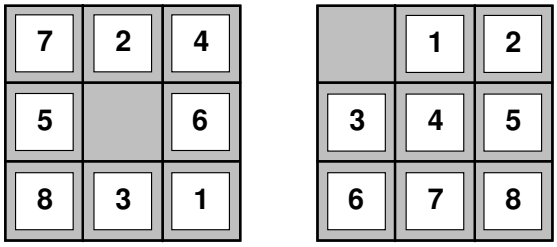
S0 = *<3, 3, L>*
G = *< 0, 0, R >*

Operations
M2 = Two men take boat
M1 = One man takes boat
C2 = Two cannibals take boat
C1 = One cannibal takes boat
MC = One man and one cannibal takes boat



Search tree

8 -puzzle



Standard for the problem... Tile movement represented as the movement of the blank space.

Operators:

L : Blank moves left

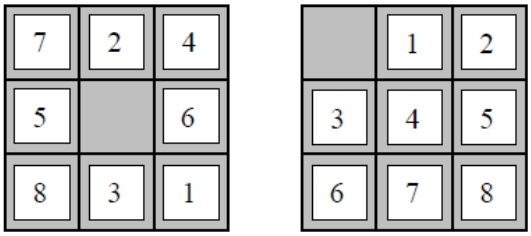
R : Blank moves right

U : Blank moves up

D : Blank moves down

$C(L) = C(R) = C(U) = C(D) = 1$

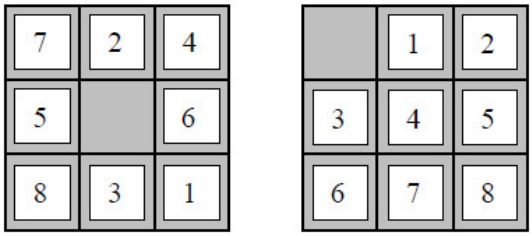
Example: The 8-puzzle



Start State Goal State

states??
actions??
goal test??
path cost??

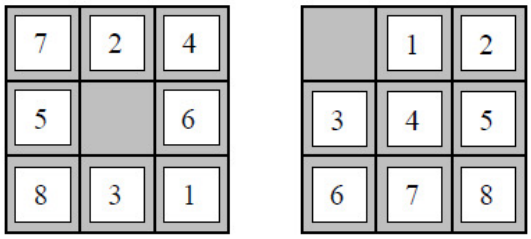
Example: The 8-puzzle



Start State Goal State

states??: integer locations of tiles (ignore intermediate positions)
actions??
goal test??
path cost??

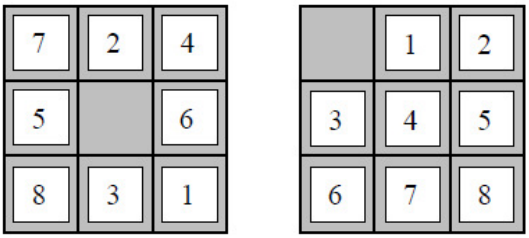
Example: The 8-puzzle



Start State Goal State

states??: integer locations of tiles (ignore intermediate positions)
actions??: move blank left, right, up, down (ignore unjamming etc.)
goal test??
path cost??

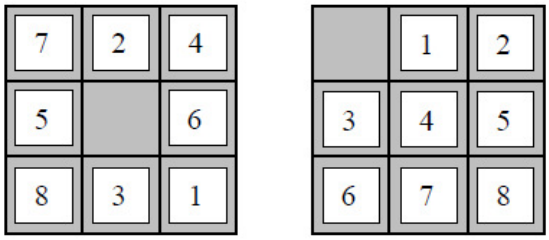
Example: The 8-puzzle



Start State Goal State

states??: integer locations of tiles (ignore intermediate positions)
actions??: move blank left, right, up, down (ignore unjamming etc.)
goal test??: = goal state (given)
path cost??

Example: The 8-puzzle

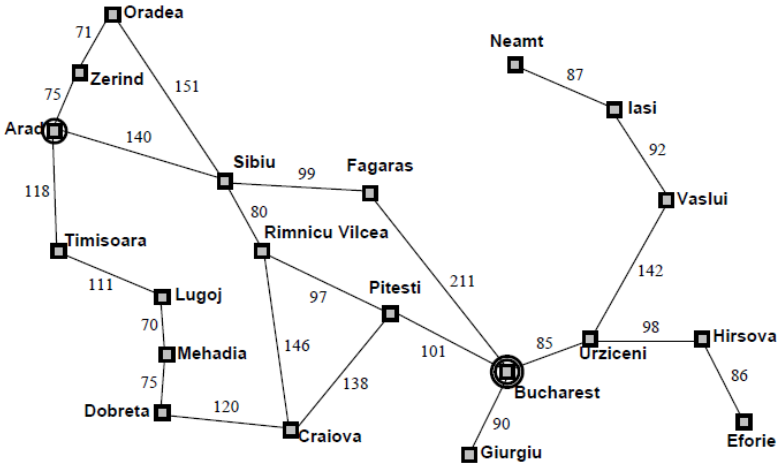


Start State Goal State

states??: integer locations of tiles (ignore intermediate positions)
actions??: move blank left, right, up, down (ignore unjamming etc.)
goal test??: = goal state (given)
path cost??: 1 per move

[Note: optimal solution of n-Puzzle family is NP-hard]

Example: Romania



Example: Romania

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

Formulate goal:
be in Bucharest

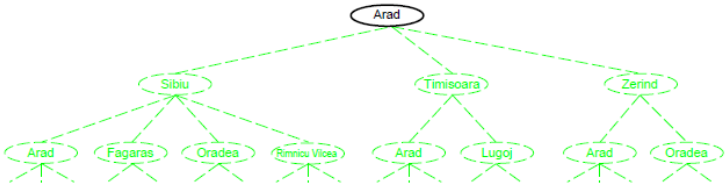
Formulate problem:
states: various cities
actions: drive between cities

Find solution:
sequence of cities, e.g., Arad, Sibiu, Fagaras, Bucharest

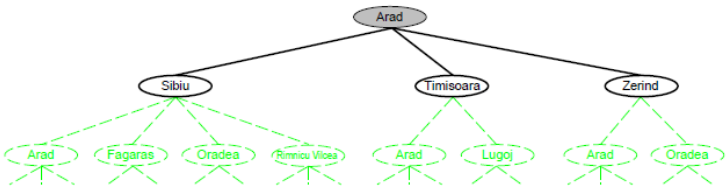
Example: Romania

- Initial State: at Arad
- States Space: being at any city
- Successor function: set of state-pairs
  S(Arad) = Zerind
- Goal State: Bucharest
- Path cost : sum of distances
- Solution : sequence of states

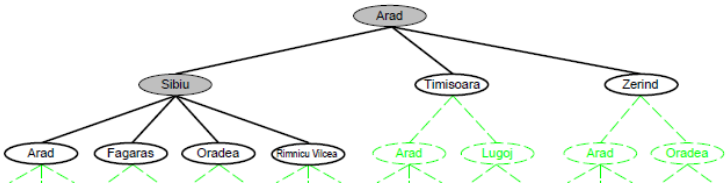
Example: Romania



Example: Romania



Example: Romania

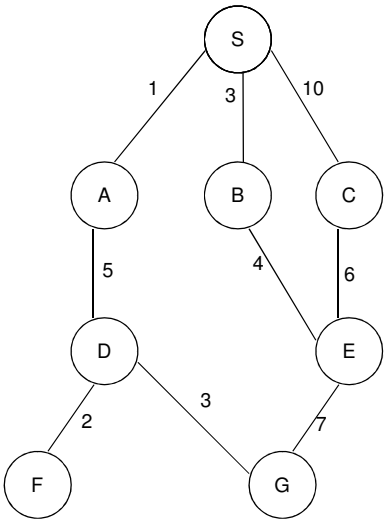


Tree Search Algorithm

- Basic Idea: Simulated exploration of state space by generating successors of already explored

```
function TREE-SEARCH( problem, strategy) returns a solution, or failure
   initialize the search tree using the initial state of problem
   loop do
      if there are no candidates for expansion then return failure
      choose a leaf node for expansion according to strategy
      if the node contains a goal state then return the corresponding solution
      else expand the node and add the resulting nodes to the search tree
   end
```

General Graph search Algorithm



Graph G = (V,E)

1) Open List : S $^{(Ø, 0)}$
   Closed list : Ø

2) OL : A$^{(S,1)}$, B$^{(S,3)}$, C$^{(S,10)}$
   CL : S

3) OL : B$^{(S,3)}$, C$^{(S,10)}$, D$^{(A,6)}$
   CL : S, A

4) OL : C$^{(S,10)}$, D$^{(A,6)}$, E$^{(B,7)}$
   CL: S, A, B

5) OL : D$^{(A,6)}$, E$^{(B,7)}$
   CL : S, A, B , C

6) OL : E$^{(B,7)}$, F$^{(D,8)}$, G$^{(D, 9)}$
   CL : S, A, B, C, D

7) OL : F$^{(D,8)}$, G$^{(D,9)}$
   CL : S, A, B, C, D, E

8) OL : G$^{(D,9)}$
   CL : S, A, B, C, D, E, F

9) OL : Ø
   CL : S, A, B, C, D, E,
   F, G

Steps of GGS

1. Create a search graph *G*, consisting only of the start node *S*; put *S* on a list called *OPEN.*
2. Create a list called *CLOSED* that is initially empty.
3. Loop: if *OPEN* is empty, exit with failure.
4. Select the first node on *OPEN*, remove from *OPEN* and put on *CLOSED*, call this node *n*.
5. if *n* is the goal node, exit with the solution obtained by tracing a path along the pointers from *n* to *s* in *G*.
6. Expand node *n*, generating the set *M* of its successors that are not ancestors of *n*.

GGS steps (contd.)

7. Establish a pointer to *n* from those members of *M* that were not already in *G* (*i.e.*, not already on either *OPEN* or *CLOSED*). Add these members of *M* to *OPEN*. For each member of *M* that was already on *OPEN* or *CLOSED*, decide whether or not to redirect its pointer to *n*. For each member of M already on *CLOSED*, decide for each of its descendents in *G* whether or not to redirect its pointer.
8. Reorder the list *OPEN* using some strategy.
9. Go *LOOP.*

Search Strategies

Uninformed/Blind Search

- Breadth First Search
- Depth First Search
- Depth Limited Search
- Bidirectional Search

Informed/Heuristic Search

- Hill Climbing Search (Improvements)
- A* Algorithm