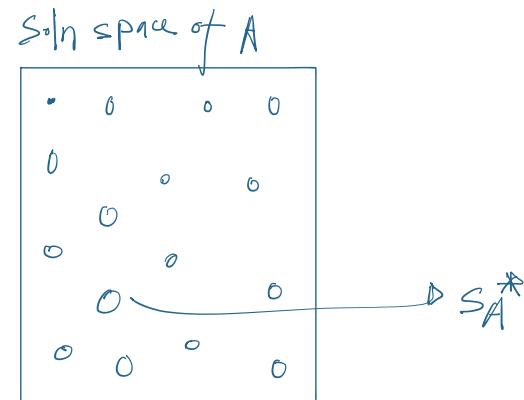


Chapter 3

Problem Solving By Searching

* যানে solution এরটা optimal solution

Problem A \rightarrow S_A^*
solution



প্রতিটা A এর soln
যোগীর quality আলা বালা থাকা থাকা...

State Space Model: Soln space থেকে S_A^* কে খুল দেয় বলো

শুধুর randomly একটি solution pick করবে।

initial solution লাগব যেটা দিয়ে বাটুক করবে।

→ randomly pick একটা লাগব

→ evaluate করবে → requirements fulfill করে রিসে

বর্ণনা করো

না করলে আবেক্ষণ্য solution নিতে হবে।

→ randomly নিয়ে না কিভাবে নিন্তে ?

→ একটা state এ কিভাবে যাবে ?

BFS → level by level process

পরিয়ে node এ যাব

DFS → একটা check করে ওটার depth ।

Algo টা যাবে দিবে এবং state থাকে আরে state এ কিভাবে যাবে

→ guide করবে

Problem-Solving Agents

- Intelligent agents can solve problems by searching a state-space
- State-space Model
 - the agent's model of the world
 - usually a set of discrete states
 - e.g., in driving, the states in the model could be towns/cities
- Goal State(s) *এটা সেই ঠিকে search করা হবে*
 - a goal is defined as a desirable state for an agent
 - there may be many states which satisfy the goal test
 - e.g., drive to a town with a ski-resort
 - or just one state which satisfies the goal
 - e.g., drive to Mammoth
- Operators (actions, successor function)
 - operators are legal actions which the agent can take to move from one state to another

এক state থেকে আরেক state' এ যে action execute হয়ে আসবে।
এক state' এ গিয়ে আরেক evaluate হবে এই গুলি সেটা কী

Initial Simplifying Assumptions

real scenario টি এন্টরেন্স

- Environment is static
 - no changes in environment while problem is being solved
- Environment is observable
- Environment and actions are discrete
 - (typically assumed, but we will see some exceptions)
- Environment is deterministic

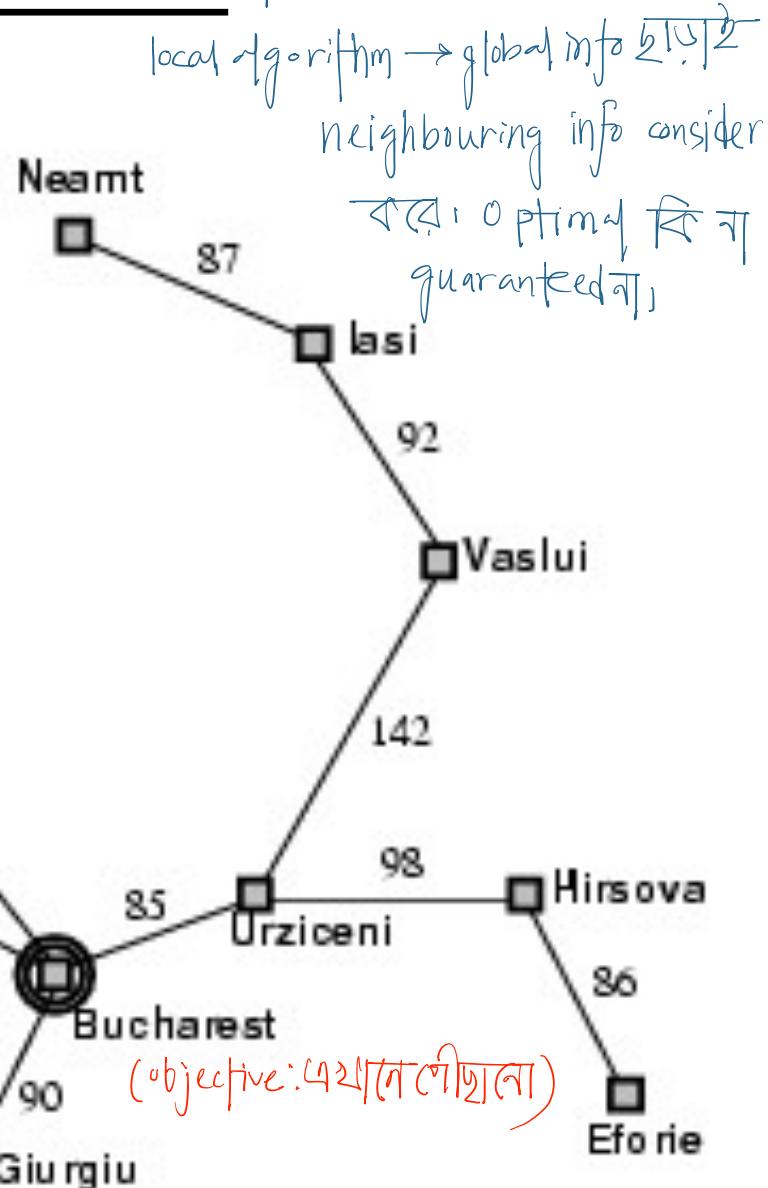
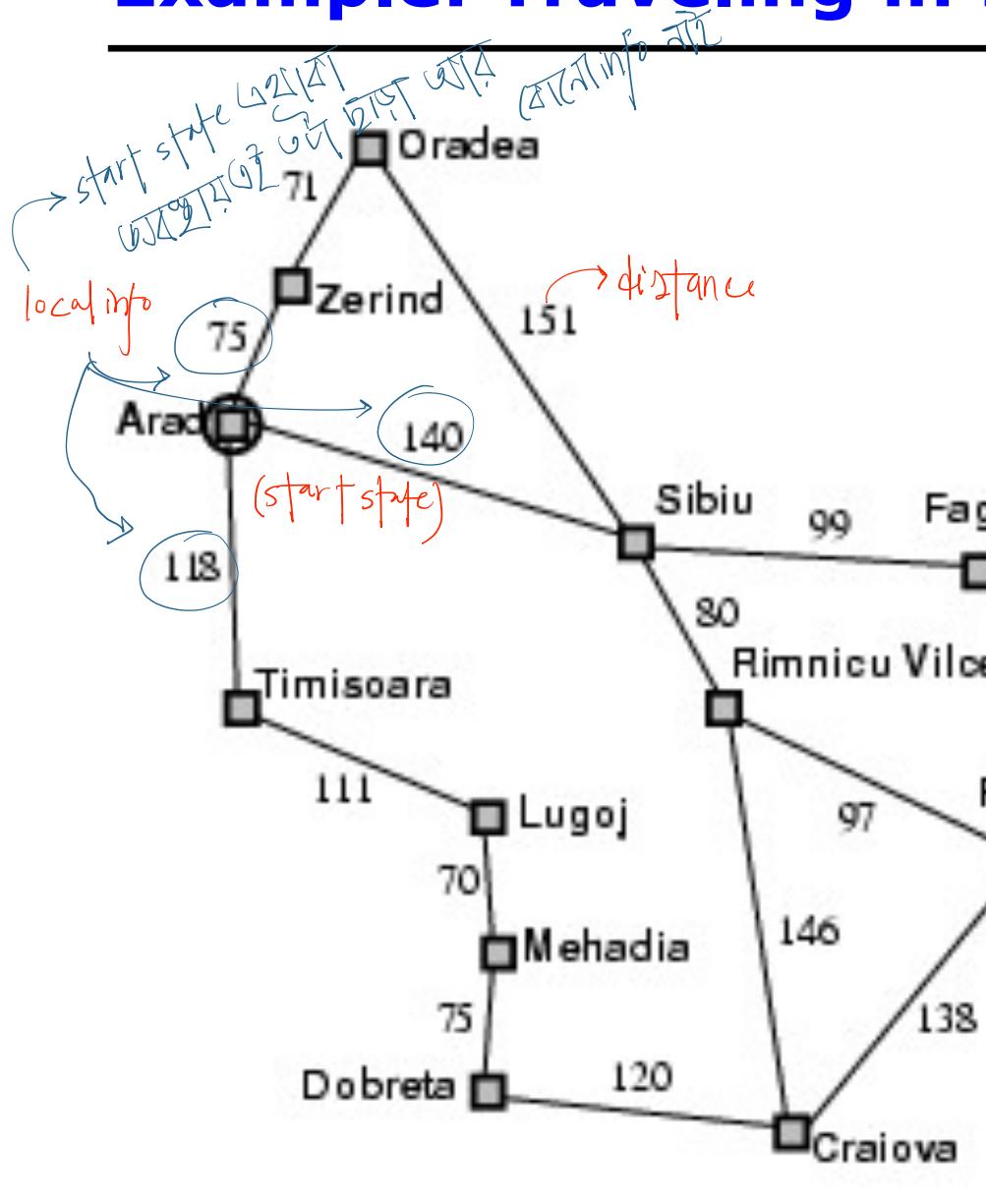
Example: Traveling in Romania

- On holiday in Romania; currently in Arad
- Flight leaves tomorrow from Bucharest
- Formulate goal:
 - be in Bucharest
- Formulate problem:
 - **states**: various cities
 - **actions/operators**: drive between cities
- Find solution
 - By searching through states to find a goal
 - sequence of cities, e.g., Arad, Sibiu, Fagaras, Bucharest
- Execute states that lead to a solution

state-space model

initially randomly রাখা নে কোথায় আবার start?
Bucharest এ সীমান্তে multiple path
আছে।

Example: Traveling in Romania



problem হলো graph এর representation এবং try করবে,

graph algorithm \rightarrow distinct

representation precise

and আরেক problem হলো graph এর map করা যায়,

মুশক algorithm দ্বাৰা AI incorporate \rightarrow approximate solution পাবা

State-Space Problem Formulation

A **problem** is defined by four items:

1. **initial state** e.g., "at Arad" (randomly)

2. **actions** or **successor function**

$S(x)$ = set of action-state pairs

e.g., $S(Arad) = \{<Arad \rightarrow Zerind, Zerind>, \dots\}$

3. **goal test** (or set of goal states)

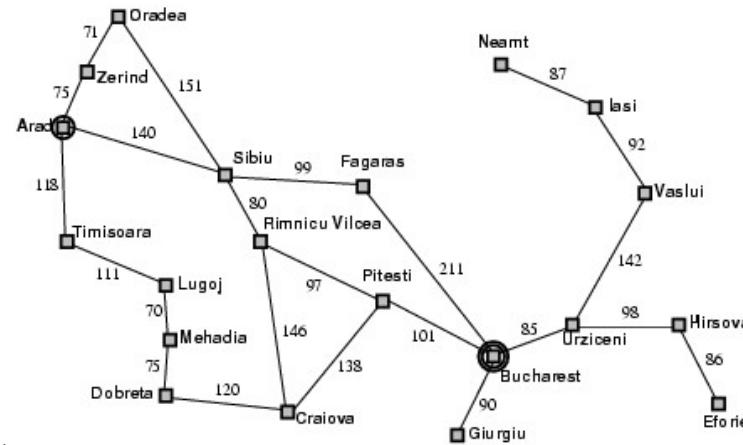
e.g., $x = \text{"at Bucharest", Checkmate}(x)$

4. **path cost** (additive)

e.g., sum of distances, number of actions executed, etc.

$c(x, a, y)$ is the step cost, assumed to be ≥ 0

A **solution** is a sequence of actions leading from the initial state to a goal state



series of action \rightarrow
search algorithm \rightarrow outcome
এবং start state থেকে
যোগ কীভাবে কীভাবে
goal state থেকে পোর্ট

Example: Formulating the Navigation Problem

- Set of States
 - individual cities
 - e.g., Irvine, SF, Las Vegas, Reno, Boise, Phoenix, Denver
- Operators
 - freeway routes from one city to another
 - e.g., Irvine to SF via 5, SF to Seattle, etc
- Start State
 - current city where we are, Irvine
- Goal States
 - set of cities we would like to be in
 - e.g., cities which are closer than Irvine
- Solution
 - a specific goal city, e.g., Boise
 - a sequence of operators which get us there,
 - e.g., Irvine to SF via 5, SF to Reno via 80, etc

Abstraction

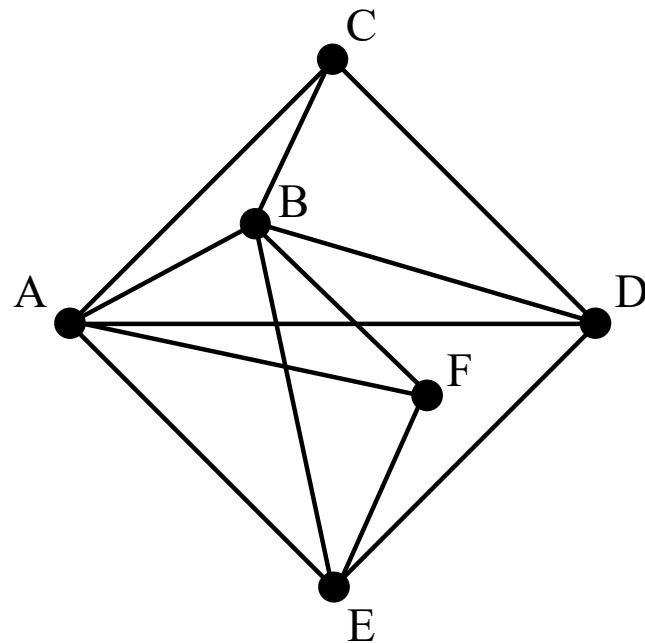
- Definition of Abstraction:
Process of removing irrelevant detail to create an abstract representation: ``high-level'', ignores irrelevant details
- Navigation Example: how do we define states and operators?
 - First step is to abstract “the big picture”
 - i.e., solve a map problem
 - nodes = cities, links = freeways/roads (a high-level description)
 - this description is an abstraction of the real problem
 - Can later worry about details like freeway onramps, refueling, etc
- Abstraction is critical for automated problem solving
 - must create an approximate, simplified, model of the world for the computer to deal with: real-world is too detailed to model exactly
 - good abstractions retain all important details

The State-Space Graph

- Graphs:
 - nodes, arcs, directed arcs, paths
- Search graphs:
 - States are nodes
 - operators are directed arcs
 - solution is a path from start S to goal G
- Problem formulation:
 - Give an abstract description of states, operators, initial state and goal state.
- Problem solving:
 - Generate a part of the search space that contains a solution

The Traveling Salesperson Problem

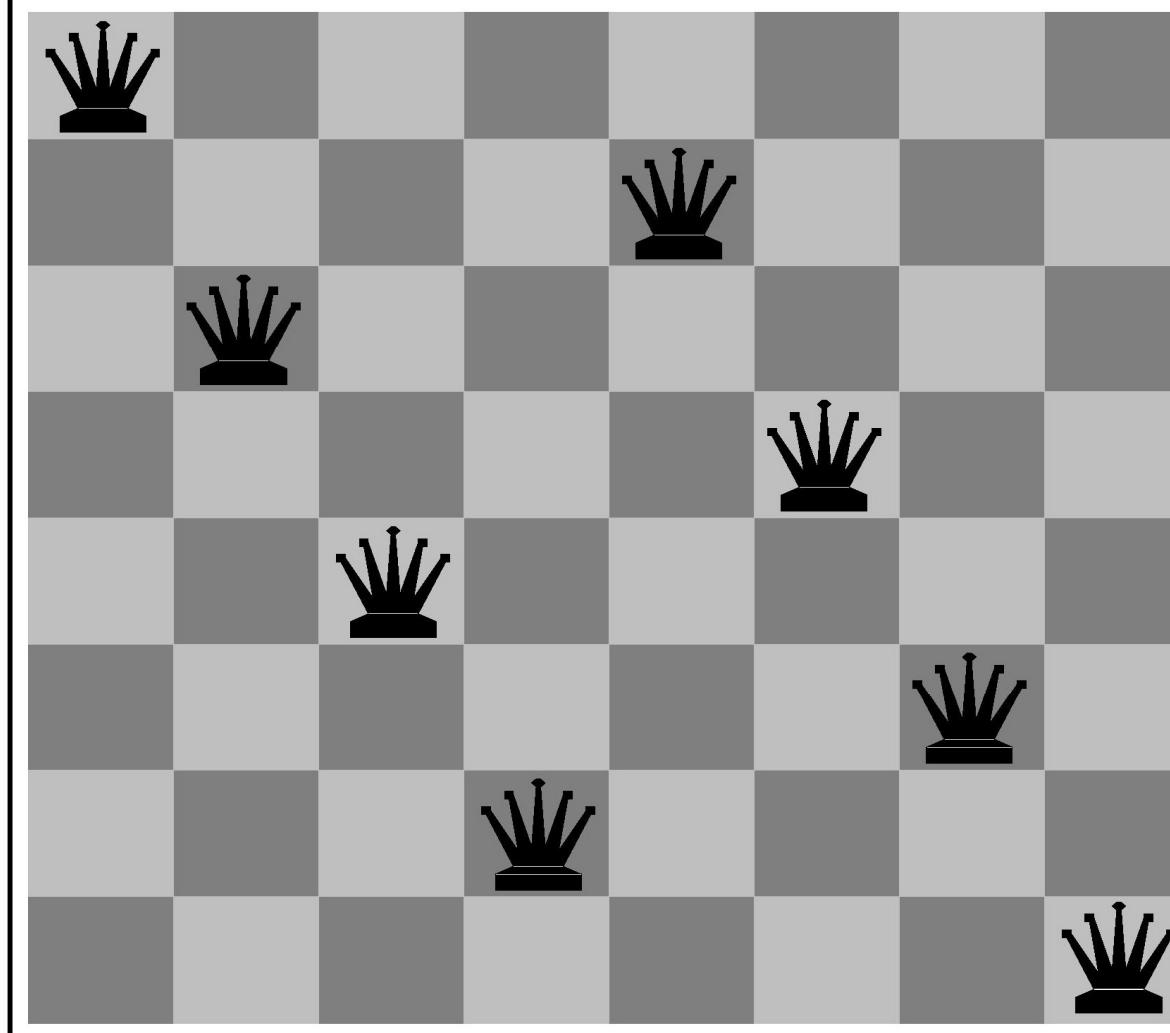
- ✓ Find the shortest tour that visits all cities without visiting any city twice and return to starting point.
- State: sequence of cities visited
- $S_0 = A$



- $G = \text{a complete tour}$

Example: 8-queens problem

$n=8$



8-queens problem 8x8
board ടെ ഏ മനഞ്ചലാ
വരുമ്പോൾ യോ താരാ വോ
യും പാക്ക് വരുതു
നാ പാഡ്രോ യാ യാ
ഒരു റോw / column /
diagonal ടെ വരുമ്പോൾ

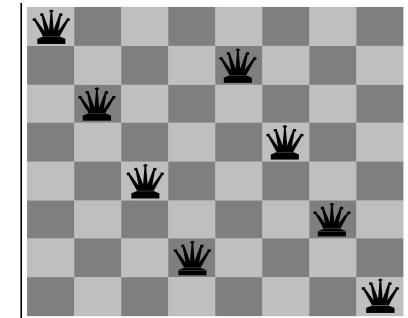
$$\begin{aligned}1 \text{ queen} &\rightarrow 8 \times 8 \\&= 64 \text{ cells} \\&\text{valid}\end{aligned}$$

$$2 \text{ queen} \rightarrow 63, 64 \text{ cells}$$

- ① incrementally one after another queen বসাই or
- ② শুধুমাত্র 8টি ক্ষেত্রে রেখা বরাবর বসাবাবা then attacking situation evaluate. Then 4টি ক্ষেত্রে ক্ষেত্রে queen কে
displace করাবে with the hope that this will
reduce the number of attacking position.
এটো ক্ষেত্রে incrementally করাবে, finally
ব্যাপারটাকে number of attacking
situation.

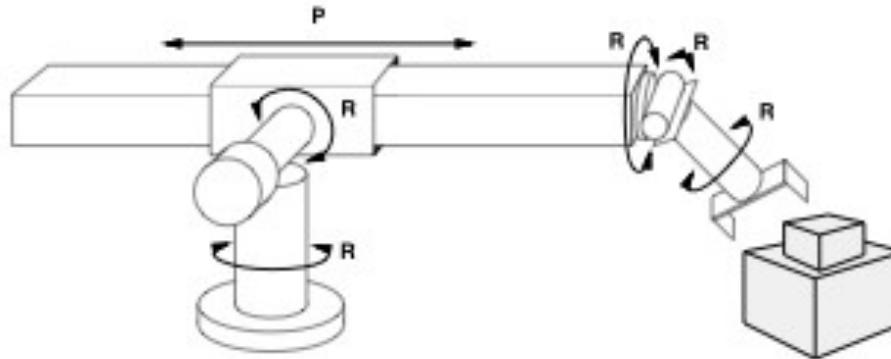
State-Space problem formulation

- states? -any arrangement of $n \leq 8$ queens
 - or arrangements of $n \leq 8$ queens in leftmost n columns, 1 per column, such that no queen attacks any other.
- initial state? no queens on the board
- actions? -add queen to any empty square
 - or add queen to leftmost empty square such that it is not attacked by other queens.
- goal test? 8 queens on the board, none attacked.
- path cost? 1 per move



Example: Robot Assembly

ex: mobile phone এর
component কূলোকে robotic arm দিয়ে আয়ান্ত
আয়ান্ত place

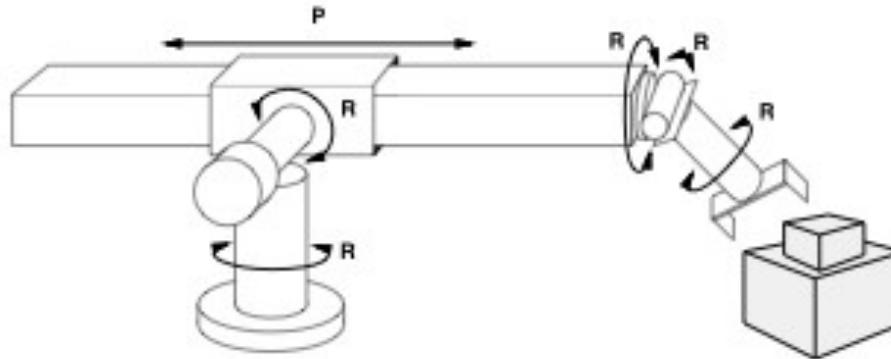


একটি component রাখ্যে যান →
 1. space বজালাবে
 2. short circuit possibility বজে
 যাব।
 so in incrementally এটা এবে
 রাখ্যে or সবুজ
 অস্থায়ে করে evaluate

- States
- Initial state
- Actions
- Goal test
- Path Cost

যাবের বিভিন্ন পার্শ্ব furniture (different size এবং)।
 গোল negative space এন্টেরি না হয়।
 ↳ ক্ষণ space use বরা ধারেন।
 → so প্রথমে calculate করে furniture কূলো রাখ্য সাক্ষাৎ।
 power transisitor → site এবড়।
 ↳ প্রথমে যশি power কে handle
 অপ্রয়োগিক রাখ হয়। heat কে release এবং
 সাক্ষাৎ। so in: body টু বড় রয়।
 Capacitor → বেশ এড়।

Example: Robot Assembly



- States: configuration of robot (angles, positions) and object parts
- Initial state: any configuration of robot and object parts
- Actions: continuous motion of robot joints
- Goal test: object assembled?
- Path Cost: time-taken or number of actions

Learning a **spam email classifier**

- States
- Initial state
- Actions
- Goal test
- Path Cost

যদি আপনি interested না।

subject, title দেখা করার ঘোষণার জন্য \rightarrow time consuming

machining learning টো। \rightarrow behavior থেকে learn করো।

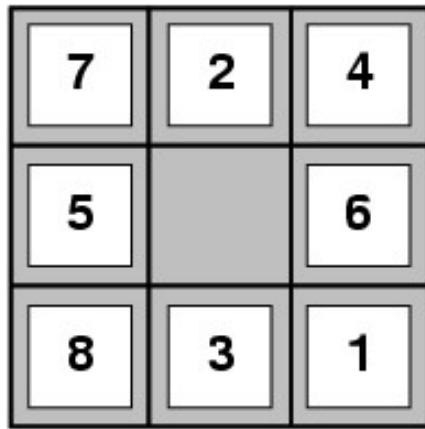
\rightarrow behaviour observe করো। কোনুলো আবিষ্ট কোনুলো
করে নিচ্ছি। But important email বা 5 টির পাইলেই
টো। আবারে এটি spam. Next এ প্রের কোনুলো email
ইত্তে spam ক পার্য।

Learning a spam email classifier

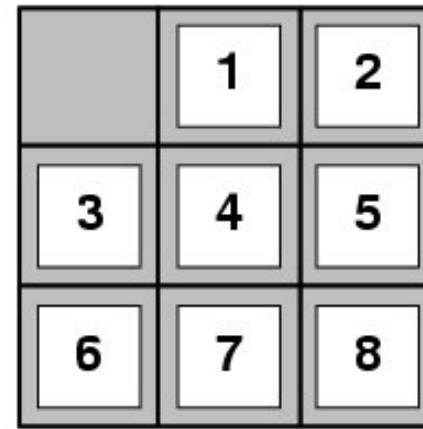
- States: settings of the parameters in our model
- Initial state: random parameter settings
- Actions: moving in parameter space
- Goal test: optimal accuracy on the training data
- Path Cost: time taken to find optimal parameters

(Note: this is an optimization problem – many machine learning problems can be cast as optimization)

Example: 8-puzzle



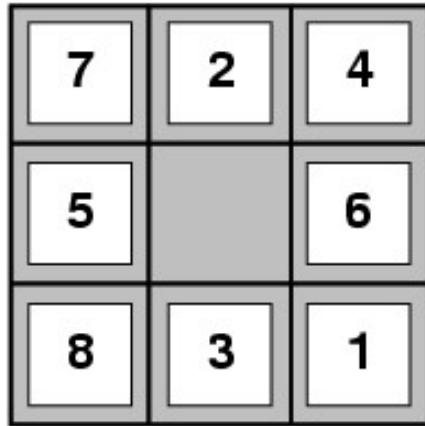
Start State



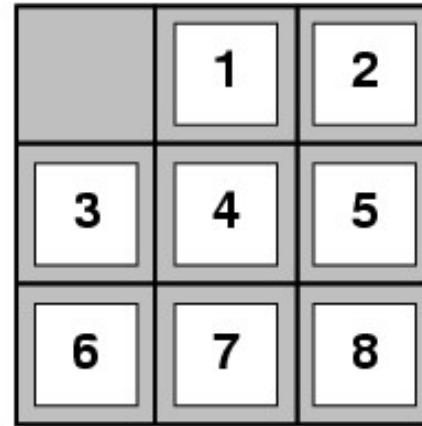
Goal State

- states?
- initial state?
- actions?
- goal test?
- path cost?

Example: 8-puzzle



Start State

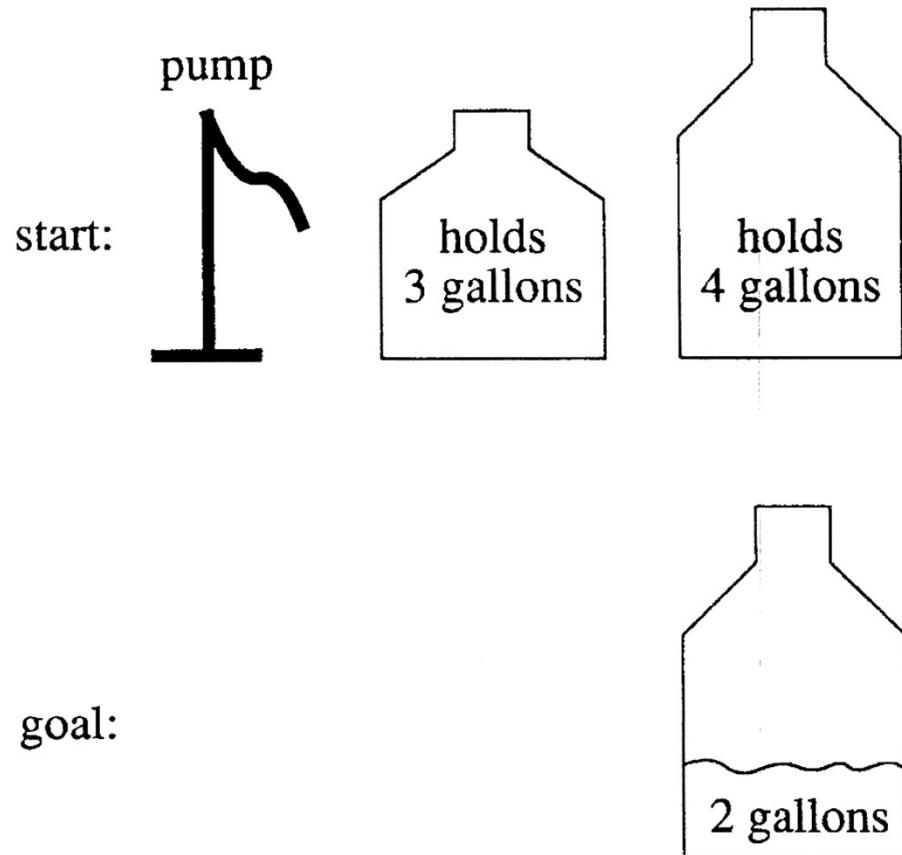


Goal State

- states? locations of tiles
- initial state? given
- actions? move blank left, right, up, down
- goal test? goal state (given)
- path cost? 1 per move

A Water Jug Problem

- You have a 4-gallon and a 3-gallon water jug
- You have a faucet with an unlimited amount of water
- You need to get exactly 2 gallons in 4-gallon jug



Puzzle-solving as Search

- State representation: **(x, y)**
 - x: Contents of four gallon
 - y: Contents of three gallon
- Start state: **(0, 0)**
- Goal state **(2, n)**
- Operators
 - Fill 3-gallon from faucet, fill 4-gallon from faucet
 - Fill 3-gallon from 4-gallon , fill 4-gallon from 3-gallon
 - Empty 3-gallon into 4-gallon, empty 4-gallon into 3-gallon
 - Dump 3-gallon down drain, dump 4-gallon down drain

Production Rules for the Water Jug Problem

1 $(x,y) \rightarrow (4,y)$
if $x < 4$

Fill the 4-gallon jug

2 $(x,y) \rightarrow (x,3)$
if $y < 3$

Fill the 3-gallon jug

3 $(x,y) \rightarrow (x - d,y)$
if $x > 0$

Pour some water out of the 4-gallon jug

4 $(x,y) \rightarrow (x,y - d)$
if $x > 0$

Pour some water out of the 3-gallon jug

5 $(x,y) \rightarrow (0,y)$
if $x > 0$

Empty the 4-gallon jug on the ground

6 $(x,y) \rightarrow (x,0)$
if $y > 0$

Empty the 3-gallon jug on the ground

7 $(x,y) \rightarrow (4,y - (4 - x))$
if $x + y \geq 4$ and $y > 0$

Pour water from the 3-gallon jug into the 4-gallon jug until the 4-gallon jug is full

The Water Jug Problem (cont'd)

$$8 (x,y) \rightarrow (x - (3 - y), 3) \\ \text{if } x + y \geq 3 \text{ and } x > 0$$

Pour water from the 4-gallon jug into the 3-gallon jug until the 3-gallon jug is full

$$9 (x,y) \rightarrow (x + y, 0) \\ \text{if } x + y \leq 4 \text{ and } y > 0$$

Pour all the water from the 3-gallon jug into the 4-gallon jug

$$10 (x,y) \rightarrow (0, x + y) \\ \text{if } x + y \leq 3 \text{ and } x > 0$$

Pour all the water from the 4-gallon jug into the 3-gallon jug

One Solution to the Water Jug Problem

Gallons in the 4-Gallon Jug	Gallons in the 3-Gallon Jug	Rule Applied
0	0	2
0	3	9
3	0	2
3	3	7
4	2	5
0	2	9
2	0	

VLSI Layout Problem

- Require positioning millions of components and connections on a chip to **minimize area**, **minimize circuit delays**, **minimize stray capacitances**, and **maximize manufacturing yield**
- The layout problem comes after the logical design phase and is usually split into two parts: **cell layout** and **channel routing**
- In cell layout, the primitive components of the circuit are grouped into cells, each of which performs some recognized function

VLSI Layout Problem (cont'd)

- Each cell has a fixed footprint (size and shape) and requires a certain number of connections to each of the other cells
- The aim is to place the cells on the chip so that they do not overlap and so that there is room for the connecting wires to be placed between the cells
- Channel routing finds a specific route for each wire through the gaps between the cells. These search problems are extremely complex, but definitely worth solving.

Next Topics

- Uninformed search
 - Breadth-first, depth-first
 - Uniform cost
 - Iterative deepening
- Informed (heuristic) search
 - Greedy best-first
 - A*
 - Memory-bounded heuristic search
 - And more....
- Local search and optimization
 - Hill-climbing
 - Simulated annealing
 - Genetic algorithms

Summary

- Problem-solving agents where search consists of
 - state space
 - operators
 - start state
 - goal states
- Abstraction and problem formulation