## CS 1571 Introduction to AI
## Lecture 2

# Introduction (cont.)
# Problem solving by searching

**Milos Hauskrecht**
milos@cs.pitt.edu
5329 Sennott Square



## Course administrivia

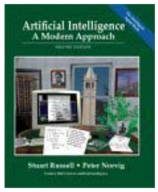
**Instructor: Milos Hauskrecht**
5329 Sennott Square
milos@cs.pitt.edu

**TA: Swapna Somasundaran**
5422 Sennott Square
swapna@cs.pitt.edu

**Course web page:**
http://www.cs.pitt.edu/~milos/courses/cs1571/

# Textbook

**Course textbook:**




Stuart Russell, Peter Norvig.
*Artificial Intelligence: A modern approach*.
2nd edition, Prentice Hall, 2002

**Other widely used AI textbooks:**

Dean, Allen, Aloimonos: Artificial Intelligence.
P. Winston: Artificial Intelligence, 3rd ed.
N. Nillson: Principles of AI.



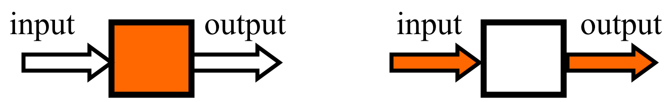
# Artificial Intelligence

- The field of **Artificial intelligence**:
  - The design and study of computer systems that behave intelligently
- **AI programs:**
  - Go beyond numerical computations and manipulations
  - Focus on problems that require reasoning (intelligence)
- **Two aspects of AI research:**
  - Engineering
    - solving of hard problems
  - Cognitive
    - Understanding the nature of human intelligence

# What is Artificial Intelligence ?

**Four different views on what makes an AI system!!** Depends on what matters more in the evaluation.

- **Reasoning vs. Behavior**




  - the **computational process** or the **end-product** matters

- **Human performance vs. Rationality**
  - Compare against human model (with its weaknesses) or a normative "ideal" model (rational system)



# AI today

AI is more rigorous and depends strongly on: applied math, statistics, probability, control and decision theories

**Recent theoretical advances and solutions:**
- Methods for dealing with uncertainty
- Planning
- Learning
- Optimizations

**Applications:**
- Focus on **partial intelligence** (not all human capabilities)
- Systems with components of intelligence in a specific application area; not general multi-purpose intelligent systems

## AI applications: Software systems.

**Diagnosis of software**, technical components

**Adaptive systems**
- **Adapt to the user**

- **Examples:**
  - **Intelligent interfaces** (http://www.research.microsoft.com/research/dtg/))
  - **Intelligent helper applications**, intelligent tutoring systems
  - **Web agents:**
    - crawlers
    - softbots, shopbots (see e.g. http://www.botspot.com/ )



## AI applications: Speech recognition.

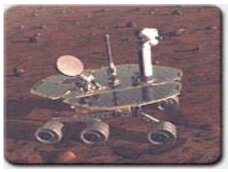
- **Speech recognition systems:**
  - Hidden Markov models

- **Adaptive speech systems**
  - Adapt to the user (training)
  - continuous speech
  - commercially available software (e.g. IBM http://www-3.ibm.com/software/speech/ )

- **Multi-user speech recognition systems**
  - Restricted (no training)
  - Customer support:
    - Airline schedules, baggage tracking;
    - Credit card companies.

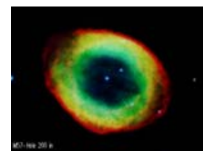
# Applications: Space exploration

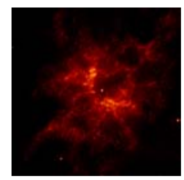
Autonomous rovers, intelligent probes



Telescope scheduling

Analysis of data



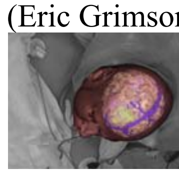




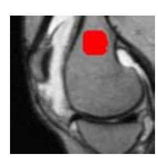
# AI applications: Medicine.

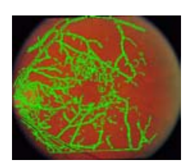
- **Medical diagnosis:**
  - Pathfinder. Lymph-node pathology.
  - QMR system. Internal medicine.
- **Medical imaging**

  http://www.ai.mit.edu/projects/medical-vision/
  - Image guided surgery (Eric Grimson, MIT)



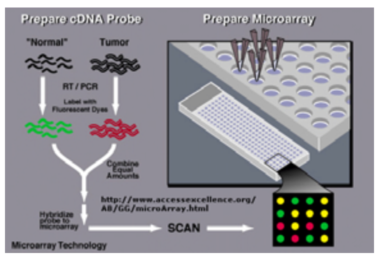
  - Image analysis and enhancement

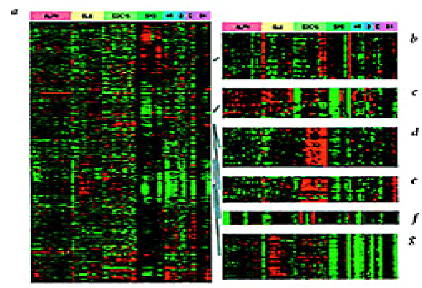




# AI applications: Bioinformatics.

- **Genomics and Proteomics**
  - Sequence analysis
  - Prediction of gene regions on DNA
  - Analysis of micro-array and proteomic MS profiles: find genes, proteins (peptides) that characterize a specific disease
  - Regulatory networks



Example of a microarray used in gene sequencing



# AI applications: Transportation.

**Autonomous vehicle control:**

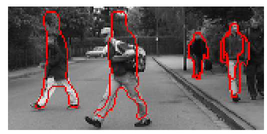
- ALVINN (CMU, Pomerleau 1993) .
  - Autonomous vehicle
  - Driving across US

- DARPA challenge (http://www.darpa.mil/grandchallenge/)
  - Drive across a Mojave desert course
  - 2004 – no vehicle finished the course
  - 2005 – 5 vehicles finished
    - The winner: Stanford team

# AI applications: Transportation.

- **Vision systems:**
  – Automatic plate recognition

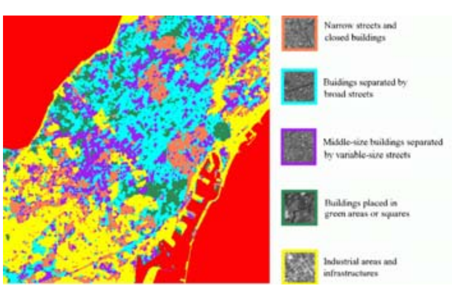

  – Pedestrian detection (Daimler-Benz)



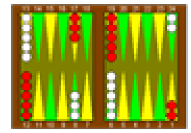
  – Traffic monitoring
- **Route optimizations**



# Classification of images or its parts




# AI applications: Game playing.



- **Backgammon**
  - TD-backgammon
    - a program that learned to play at the championship level (from scratch).
    - reinforcement learning

- **Chess**
  - Deep blue (IBM) program beats Kasparov.
- **Bridge**

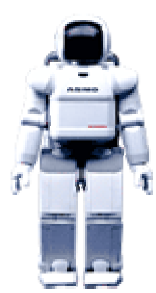
- **Etc.**



# AI applications.

- **Robotic toys**
  - Sony's Aibo (http://www.us.aibo.com/ )

- **Humanoid robot**
  - Honda's ASIMO (http://world.honda.com/robot/ )



# Other application areas

- **Text classification, document sorting**:
  - Web pages, e-mails
  - Articles in the news
- **Video, image classification**
- **Music composition, picture drawing**
- **Entertainment** ☺




# Topics

- **Problem solving and search.**
  - Formulating a search problem, Search methods, Combinatorial and Parametric Optimization.
- **Logic and knowledge representations.**
  - Logic, Inference
- **Planning.**
  - Situation calculus, STRIPS, Partial-order planners,
- **Uncertainty.**
  - Modeling uncertainty, Bayesian belief networks, Inference in BBNs, Decision making in the presence of uncertainty.
- **Machine Learning**
  - May be …

# Problem solving by searching



## Example

- Assume a problem of computing the roots of the quadratic equation

$$ax^2 + bx + c = 0$$

Do you consider it a challenging problem?

## Example

- Assume a problem of computing the roots of the quadratic equation

$$ax^2 + bx + c = 0$$

Do you consider it a challenging problem?

Hardly we just apply the standard formula:

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$



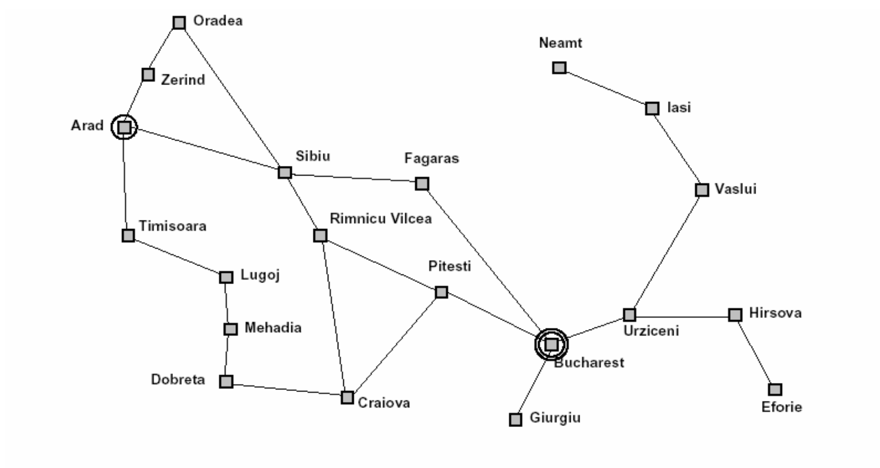
## Solving problems by searching

- Some problems have a straightforward solution
  - Just apply the formula, or follow a standardized procedure

    **Example:** solution of the quadratic equation
  - Hardly a sign of intelligence
- More interesting problems require **search**:
  - more than one possible alternative needs to be explored before the problem is solved
  - the number of alternatives to search among can be very large, even infinite.

# Search example: Traveler problem

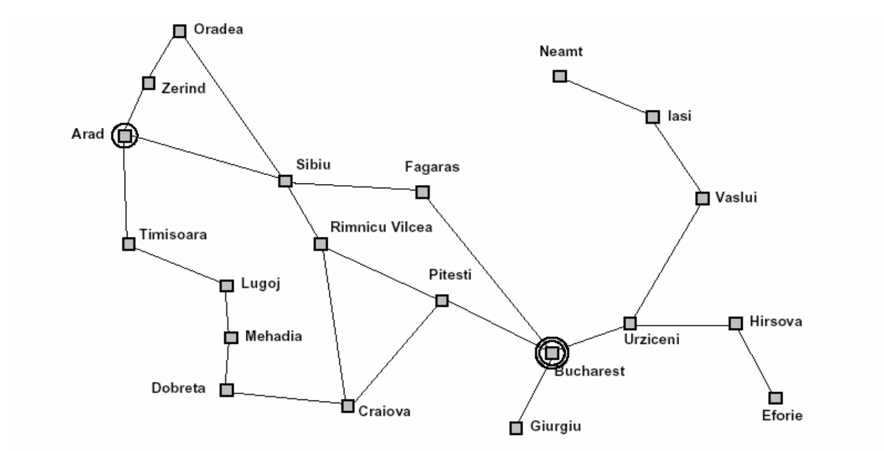
- Find a route from one city (**Arad**) to the other (**Bucharest**)






# Example. Traveler problem

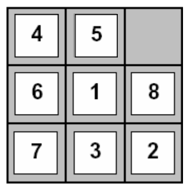
- Another flavor of the traveler problem:
  - find the route with the minimum length between S and T

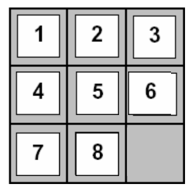


# Example. Puzzle 8.

- Find the sequence of the empty tile moves from the initial game position to the designated target position

**Initial position** **Goal position**

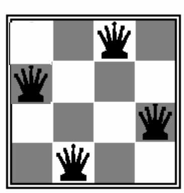


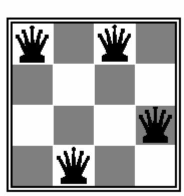



# Example. N-queens problem.

Find a configuration of n queens not attacking each other



**A goal configuration**



**A bad configuration**

# A search problem

**is defined by:**

- **A search space:**
  - The set of objects among which we search for the solution Example: objects = routes between cities, or N-queen configurations
- **A goal condition**
  - What are the characteristics of the object we want to find in the search space?
  - Examples:
    - Path between cities A and B
    - Path between A and B with the smallest number of links
    - Path between A and B with the shortest distance
    - Non-attacking n-queen configuration



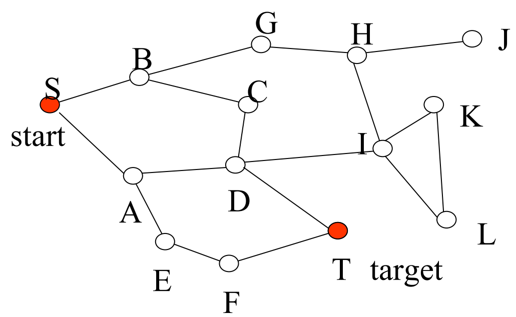
# Search

- **Search (process)**
  - The process of exploration of the search space
- **The efficiency of the search depends on:**
  - The search space and its size
  - Method used to explore (traverse) the search space
  - Condition to test the satisfaction of the search objective (what it takes to determine I found the desired goal object)

- **Important to remember !!!**
  - Conveniently chosen **search space** and the **exploration policy** can have a profound effect on the efficiency of the solution

# Graph search

- **Many search problems can be naturally represented as graph search problems**
- **Typical example: Route finding**
  - Map corresponds to the graph, nodes to cities, links to available connections between cities
  - **Goal:** find a route (path) in the graph from S to T

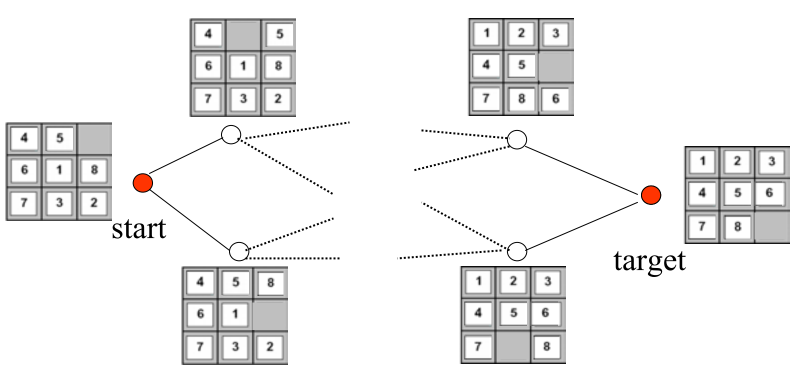





# Graph search

- **Less obvious conversion:**

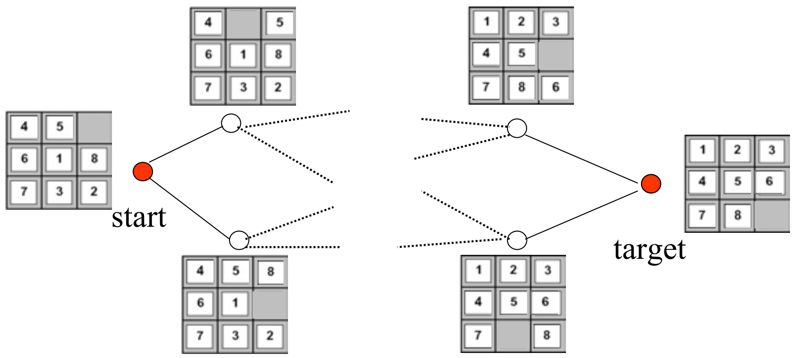
  **Puzzle 8.** Find a sequence of moves from the initial configuration to the goal configuration.
  - nodes corresponds to states of the game,
  - links to valid moves made by the player



# Graph search

- **Less obvious conversion:**

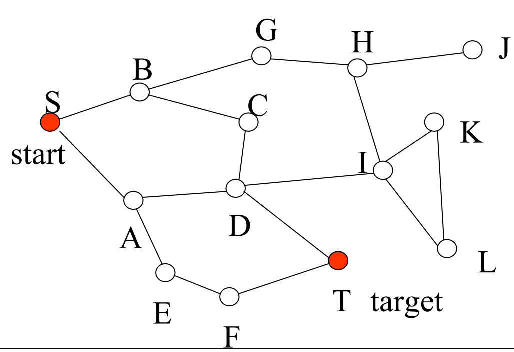
  **Puzzle 8.** Find a sequence of moves from the initial configuration to the goal configuration.
  - nodes corresponds to states of the game,
  - links to valid moves made by the player






# Graph search problem

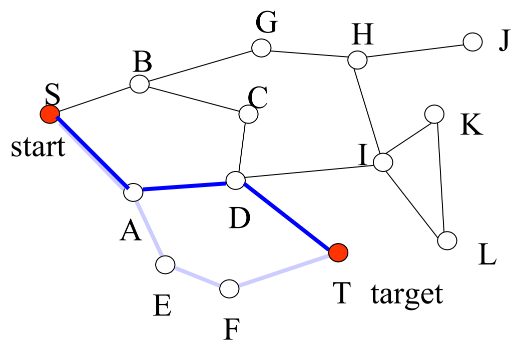
- **States** - game positions, or locations in the map that are represented by nodes in the graph
- **Operators -** connections between cities, valid moves
- **Initial state** – start position, start city
- **Goal state** – target position (positions), target city (cities)



# Graph search

- **More complex versions of the graph search problems:**
  - Find a minimal length path

    (= route with the smallest number of connections, the shortest sequence of moves that solves Puzzle 8)

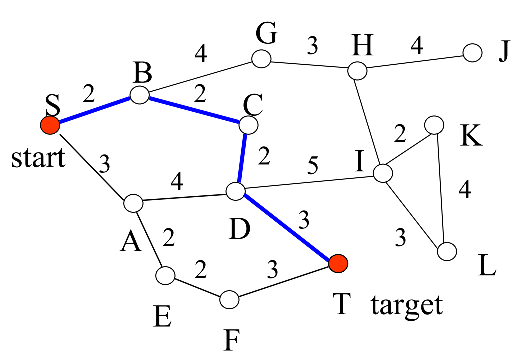





# Graph search

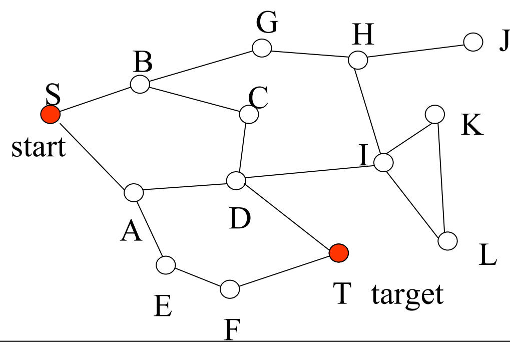
- **More complex versions of the graph search problems:**
  - Find a minimum cost path

    (= a route with the shortest distance)



# Graph search

- How to find the path in between S and T ?
- **A strawman solution:**
  - Generate systematically all sequences of 1, 2, 3, … edges
  - Check if the sequence yields a path between S and T.
- Can we do better?

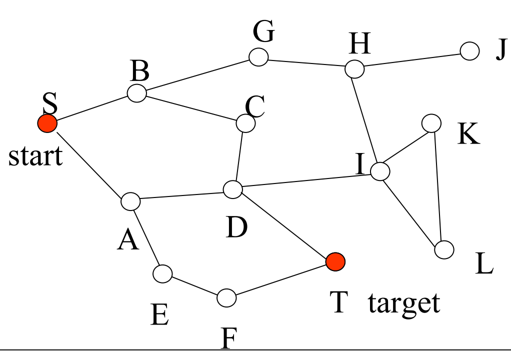





# Graph search

**Can we do better?**

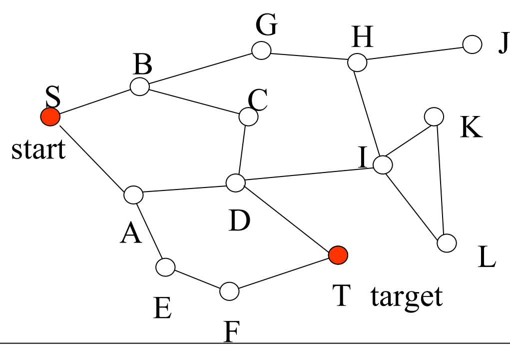
- We are not interested in sequences that do not start in S and that are not valid paths
- **Solution:**
  - ?



## Graph search

**Can we do better?**

- We are not interested in sequences that do not start in S and that are not valid paths
- **Solution:**
  - Look only on valid paths starting from S






## Graph search

- Being smarter about the space we search for the solution pays off in terms of the search process efficiency**.**