CS 1571 Introduction to AI
Lecture 2

AI applications

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

Artificial Intelligence

- The field of **Artificial intelligence:**
  - The design and study of computer systems that behave intelligently
- **AI:**
  - Focus on nontrivial problems that require reasoning and are often solved by humans
  - Goes beyond numerical computations and manipulations
- **Benefits of AI research**
  - Engineering aspect
    - solving of hard problems
  - Cognitive aspect
    - Understanding the nature of human intelligence

Search and information retrieval

- **Web search engines**
  - Improve the quality of search
  - Rely on methods/algorithms developed in AI
    - Analysis of the link topology, web page content
  - Add inferences and knowledge to search queries
- **Semantic web (or web 2):**
  - From information to knowledge sharing
  - Ontology languages

AI applications: Computer systems

- **Diagnosis:**
  - software, technical components
- **Adaptive systems**
  - Adapt systems/software to user needs / preferences
  - Adapt /configure systems to specific tasks
- **Examples:**
  - Intelligent interfaces
  - Intelligent helper applications
  - Collaborative filtering
  - Target advertising

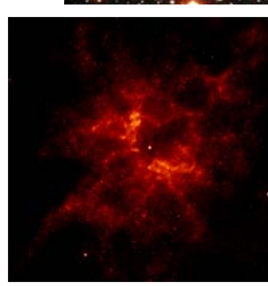
Speech recognition

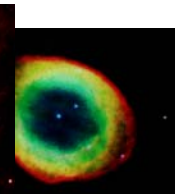
- **Speech recognition systems:**
  - Systems based on statistical and probabilistic models, (Hidden Markov models)
- **Adaptive speech systems**
  - Adapt to the user (training)
  - continuous speech
  - commercially available software – (Nuance, IBM)
    - http://www.nuance.com/
- **Multi-user speech recognition systems**
  - Voice command/voice activated devices
  - Customer support systems:
    - Airline schedules, baggage tracking
    - Credit card companies

Space exploration

Autonomous rovers, intelligent probes

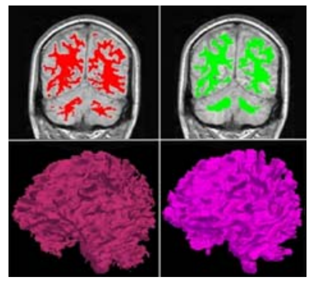
Analysis of sky survey data





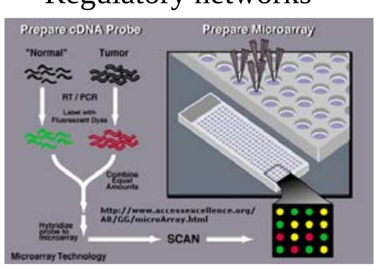
AI applications: Medicine.

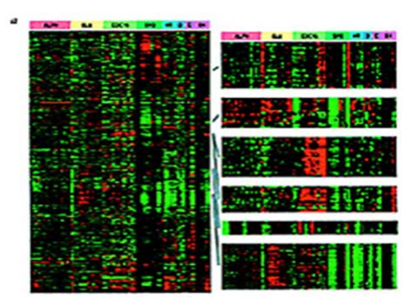
- **Medical diagnosis:**
  - QMR system. Internal medicine.
- **Patient Monitoring and Alerting:**
  - Cerner
- **Medical imaging**
  - Classification of body structures and visualization
- **Robotic surgeries**



AI applications: Bioinformatics

- **Genomics and Proteomics**
  - Sequence analysis
  - Prediction of gene regions on DNA
  - Analysis of DNA 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:

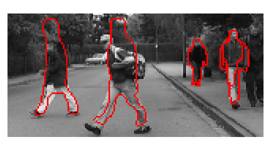
- ALVINN (CMU, Pomerleau 1993) .
  - Autonomous vehicle driving across US
- Series of DARPA challenges (http://www.darpa.mil/grandchallenge/)
  - 2004, 2005 Drive across Mojave desert
  - 2007 - DARPA Urban Challenge

- Google autonomous vehicles

AI applications: Transportation.

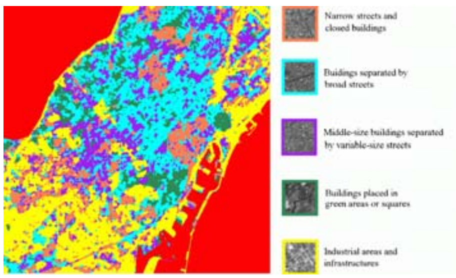
- **Vision systems:**
  - Automatic plate recognition

  - Pedestrian detection



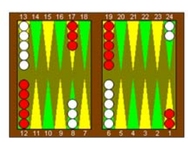
  - Traffic monitoring
- **Navigation/route optimizations**

Classification of images or its parts



Game playing

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



- **Chess**
  - Deep blue (IBM) program (defeated Kasparov)

- **Bridge, Poker**

Natural language processing

- **Understanding/annotation of free text**
- **IBM's Watson project**
  - www.ibm.com/**watson**
  - Successfully competed against the top human players in Jeopardy

CS 1571 Intro to AI M. Hauskrecht

Robots

- **Robotic toys**
  - Sony's Aibo (http://www.us.aibo.com/)
- **Vacuum cleaners**
- **Humanoid robot**
  - Honda's ASIMO (http://world.honda.com/robot/)

CS 1571 Intro to AI M. Hauskrecht

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**
  - Basic learning models, Supervised and unsupervised learning

CS 1571 Introduction to AI
Lecture 2

Problem solving by searching

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

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

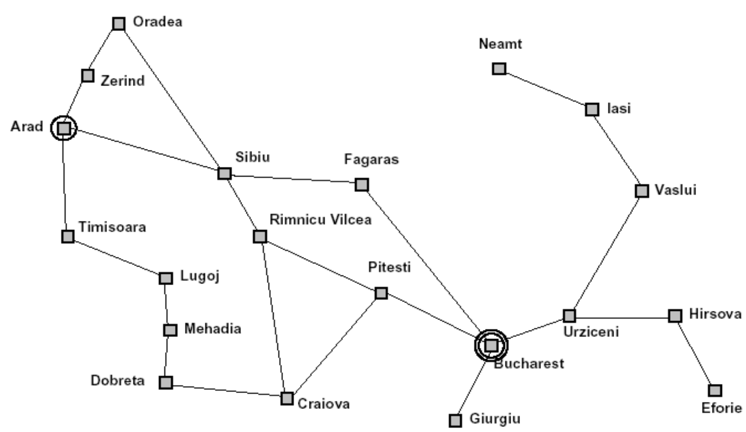
- Some problems have a straightforward solution
  - Just apply a known formula, or follow a standardized procedure

    Example: solution of the quadratic equation
  - Hardly a sign of intelligence

- Solving more interesting problems often requires **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

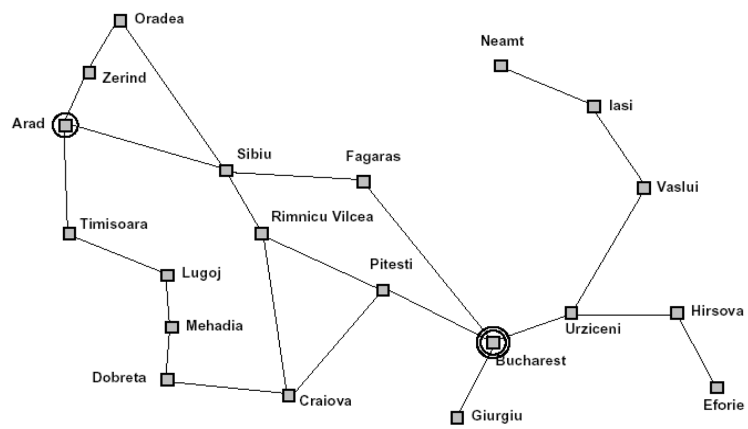
Search example: Route finding

- Find a route (path) from one city to another city



Example. Traveler problem

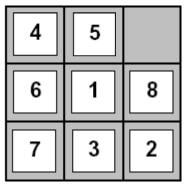
- Another flavor of the traveler problem:
  - find the route with **the minimum length** between S and T

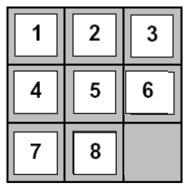


Example. Puzzle 8.

- Find the sequence of move of tiles from the initial game position to the designated target position

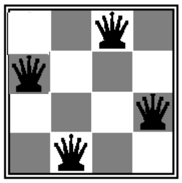
Initial position **Goal position**



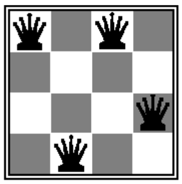


Example. N-queens problem.

Find a configuration of n queens on an n x n board such that queens do not attack each other



A goal configuration



A bad configuration

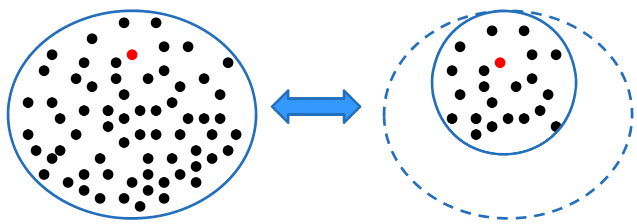
A search problem

is defined by:

- **A search space:**
  - The set of objects among which we search for the solution
  - **Example:** routes connecting two 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

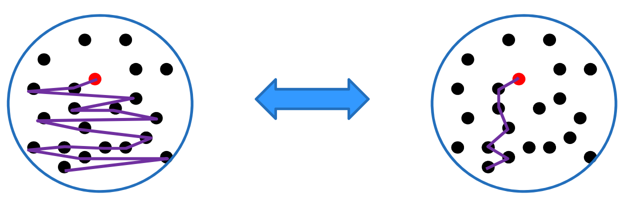
- **Search (process)**
  - The process of exploration of the search space
- **The efficiency of the search depends on:**
  - **The search space and its size**
  - Method used to explore (traverse) the search space
  - Condition to test the satisfaction of the search objective (what it takes to determine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



CS 1571 Intro to AI M. Hauskrecht

Search

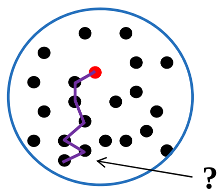
- **Search (process)**
  - The process of exploration of the search space
- **The efficiency of the search depends on:**
  - The search space and its size
  - **Method used to explore (traverse) the search space**
  - Condition to test the satisfaction of the search objective (what it takes to determine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



CS 1571 Intro to AI M. Hauskrecht

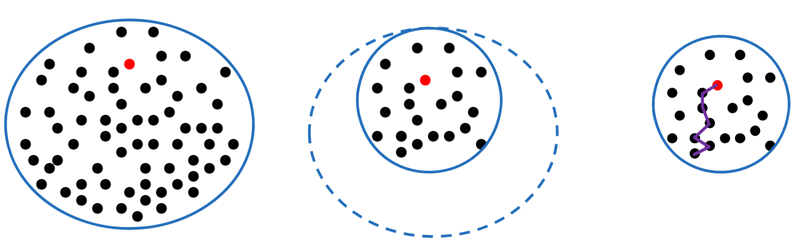
Search

- **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



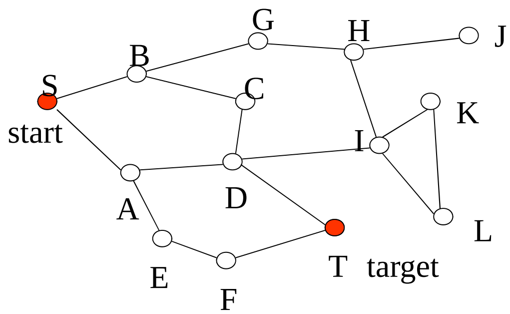
Search

- **Search (process)**
  - The process of exploration of the search space
- **Important**
  - We can often influence the efficiency of the search !!!!
  - We can be smart about choosing the **search space**, the **exploration policy, and the design of the goal test**



Graph search

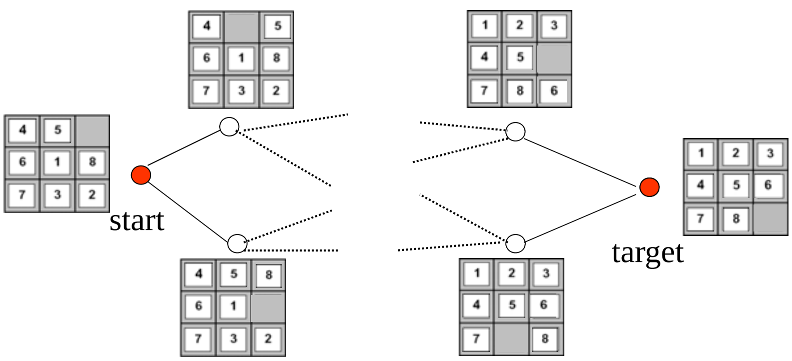
- **Search problems can be often represented using graphs**
- **Typical example: Route finding**
  - Map corresponds to the graph, nodes to cities, links valid moves via available connections
  - **Goal:** find a route (sequence of moves) in the graph from S to T



Graph search

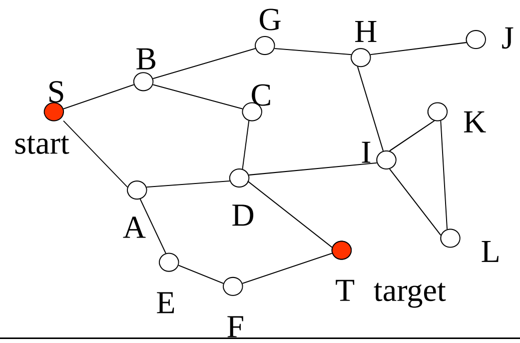
- **Less obvious conversion:**

  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



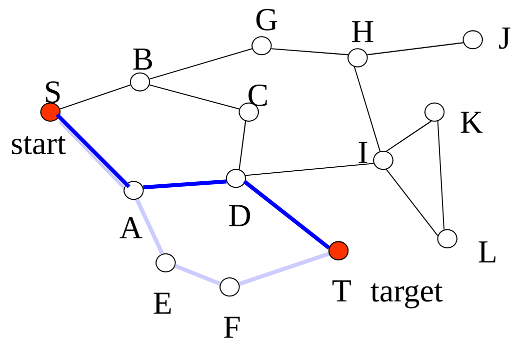
Graph search problem

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



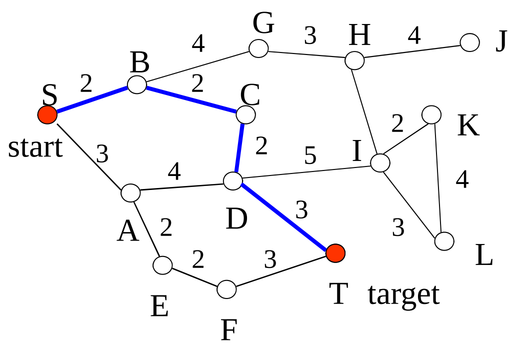
Graph search

- **More complex versions of the graph search problems:**
  - Find the minimal length path
    (= a route with the smallest number of connections, the shortest sequence of moves that solves Puzzle 8)



Graph search

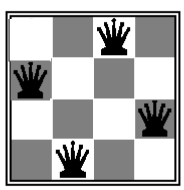
- **More complex versions of the graph search problems:**
  – Find the minimum cost path
  (= a route with the shortest distance)



N-queens

Some problems can be converted to the graph search problems

- **But some problems are harder and less intuitive**
  – Take e.g. N-queens problem.



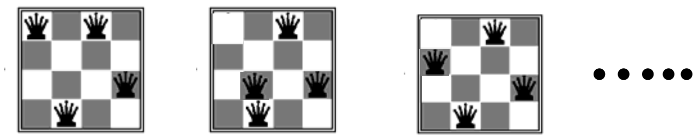
Goal configuration

- **Problem:**
  – **We look for a configuration, not a sequence of moves**
  – **No distinguished initial state, no operators (moves)**

N-queens

How to choose the search space for N-queens?

- Ideas? **Search space:**
  - all configurations of N queens on the board



- **Can we convert it to a graph search problem?**
- We need states, operators, initial state and goal condition.

N-queens

Search space:
- all configurations of N queens on the board

- **Can we convert it to a graph search problem?**
- We need states, operators, initial state and goal state.

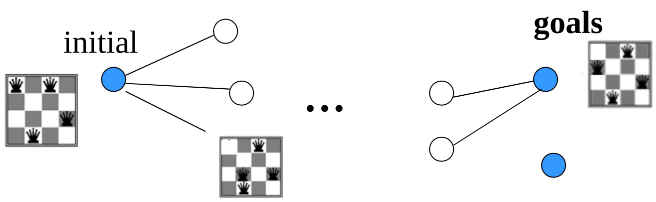
States are: N-queen configurations
Initial state: ?
Operators (moves)?

N-queens

Search space:

– all configurations of N queens on the board

- **Can we convert it to a graph search problem?**
- We need states, operators, initial state and goal condition.



Initial state: an arbitrary N-queen configuration
Operators (moves): change a position of one queen

N-queens

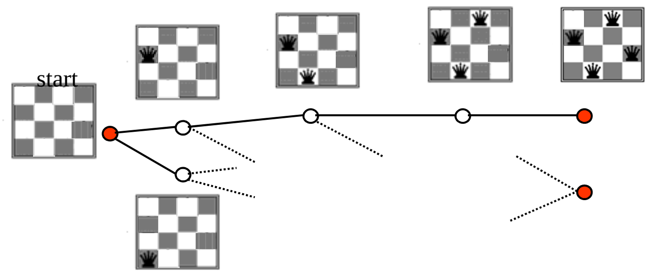
Is there an alternative way to formulate the N-queens problem as a search problem?

- Ideas?

N-queens

Is there an alternative way to formulate the N-queens problem as a search problem?

- **Search space:** configurations of 0,1,2, … N queens
- Graph search:
  - States configurations of 0,1,2,…N queens
  - Operators: additions of a queen to the board
  - Initial state: no queens on the board

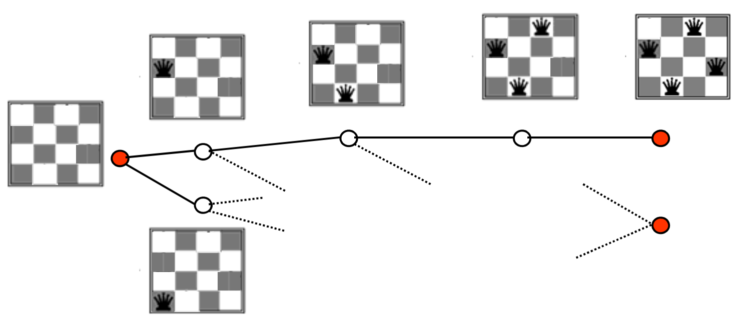


Graph search

N-queens problems

- This is a different graph search problem when compared to Puzzle 8 or Route planning:

  We want to find only the target configuration, not a path



Two types of graph search problems

- **Path search**
  - Find a path between states S and T
  - **Example:** traveler problem, Puzzle 8
  - **Additional goal criterion:** minimum length (cost) path

- **Configuration search (constraint satisfaction search)**
  - Find a state (configuration) satisfying the goal condition
  - **Example:** n-queens problem
  - **Additional goal criterion:** "soft" preferences for configurations, e.g. minimum cost design

Search problem

Search problems that can be represented or converted into a graph search problems can be defined in terms of:

- **Initial state**
  - State (configuration) we start to search from (e.g. start city, initial game position)
- **Operators**:
  - Transform one state to another (e.g. valid connections between cities, valid moves in Puzzle 8)
- **Goal condition:**
  - Defines the target state (destination, winning position)
- **Search space (**the set of objects we search for the solution) **:**
  - is now defined indirectly through**:**

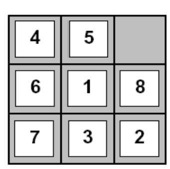
    the initial state + operators

Traveler problem.

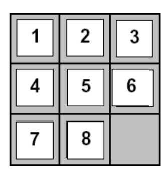
Traveler problem formulation:

- **States:** different cities
- **Initial state:** city Arad
- **Operators:** moves to cities in the neighborhood
- **Goal condition: city** Bucharest
- **Type of the problem:** path search
- **Possible solution cost:** path length

Puzzle 8 example



Initial state

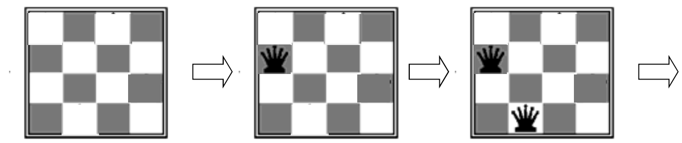


Goal state

Search problem formulation:

- **States:** tile configurations
- **Initial state:** initial configuration
- **Operators:** moves of the empty tile
- **Goal:** reach the winning configuration
- **Type of the problem: path search**
- **Possible solution cost:** a number of moves

N-queens problem



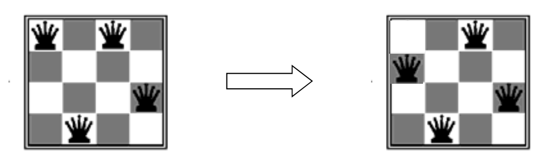
Initial configuration

Problem formulation:

- **States:** configurations of 0 to 4 queens on the board
- **Initial state:** no-queen configuration
- **Operators:** add a queen to the leftmost unoccupied column
- **Goal:** a configuration with 4 non-attacking queens
- **Type of the problem**: configuration search

N-queens problem

Alternative formulation of N-queens problem



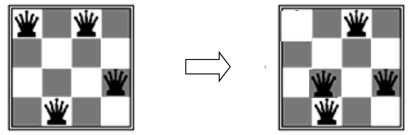
Bad goal configuration   Valid goal configuration

Problem formulation:

- **States:** different configurations of 4 queens on the board
- **Initial state:** an arbitrary configuration of 4 queens
- **Operators:** move one queen to a different unoccupied position
- **Goal:** a configuration with non-attacking queens
- **Type of the problem:** configuration search

Comparison of two problem formulations

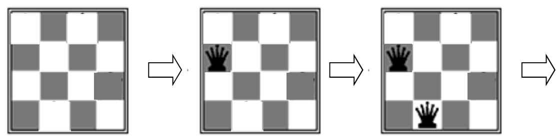
Solution 1:



Operators: switch one of the queens

$\binom{16}{4}$ - all configurations

Solution 2:

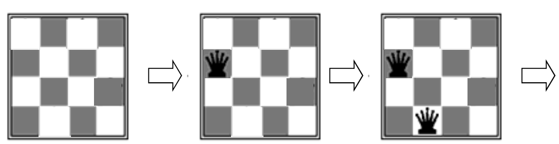


Operators: add a queen to the leftmost unoccupied column

$1 + 4 + 4^2 + 4^3 + 4^4 < 4^5$ - configurations altogether

Even better solution to the N-queens

Solution 2:



Operators: add a queen to the leftmost unoccupied column

$< 4^5$ - configurations altogether

Improved solution with a smaller search space

Operators: add a queen to the leftmost unoccupied column such that it does not attack already placed queens

$$\leq 1 + 4 + 4*3 + 4*3*2 + 4*3*2*1 = 65$$

- configurations altogether

Formulating a search problem

- **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)

- **Think twice before solving the problem by search:**
  – Choose the **search space** and the **exploration policy**