AI Searching Techniques

Uninformed Search Techniques

Dr.Sherif Saad

Learning Objectives

- Introduce the students to different search techniques and algorithms
- Present a general approach to model and represent problems as search problems
- Demonstrate how to implement AI search techniques and algorithms

Last-Time

- Introduction to AI Search Techniques
- Uninformed Search (blind or brute-force)

Outlines

- Search Problem Formulation
- Depth First Search (Uninformed Search Techniques)
- Solving N-Queens with DFS
- Constraint Satisfaction Problem
- Appendix-Graph Data Structure

Search Techniques Evaluation Criteria

- Completeness:
  - always find a solution if one exists
- Optimality:
  - always find the least-expensive solution
- Time Complexity:
  - Number of state or nodes generated or expanded
- Space Complexity:
  - Maximum number of nodes stored in memory to find the goal

Search Problem Formulation

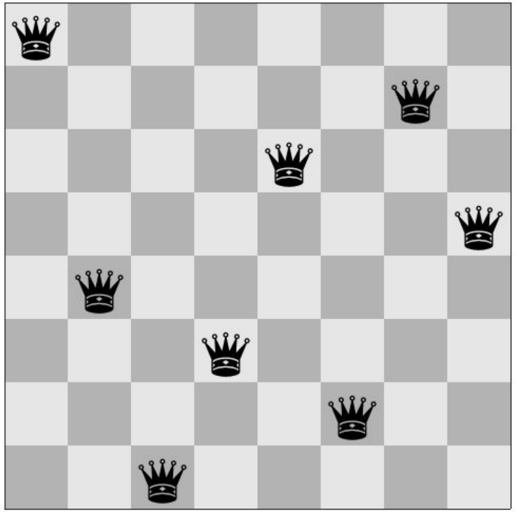
Given a problem description To formulate a search problem, we need to define the following

- States: all possible solutions or partial solutions
- Initial State: the initial state of the problem the starting point of the search
- Actions: any step towards the solution
- Goal Test: validate that we reached a solution (goal)
- Path Cost: the cost of the solution

Search Problem Formulation: Examples

N-Queens: *Given an **nxn** chessboard, arrange n queens so that none is attacking another.*

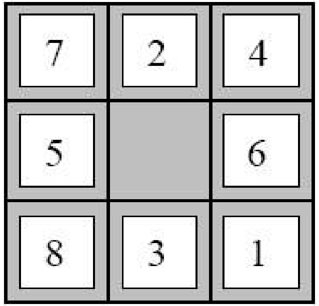
- *States: All arrangements of 0 to n queens on the board.*
- *Initial State: the board is empty (no queen on the board)*
- *Actions: Add or move a queen to any empty square.*
- *Goal Test: N queens the board with none attacked*



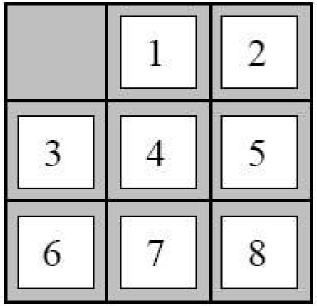
Search Problem Formulation: Examples

8-Puzzle: *Given a **3**x**3** grid.* One of the squares is empty. The objective is to move to squares around into different positions and having the numbers displayed in the "goal state".

- *States: location of each of the 8 blocks in the 3x3 grid*
- *Initial State: Any state with 8 blocks and one empty square*
- *Actions: Move a block Up, Down, Right, or Left*
- *Goal Test: the state match the goal state.*
- *Path Cost: total moves, each move costs X (e.g. 1)*



Start State

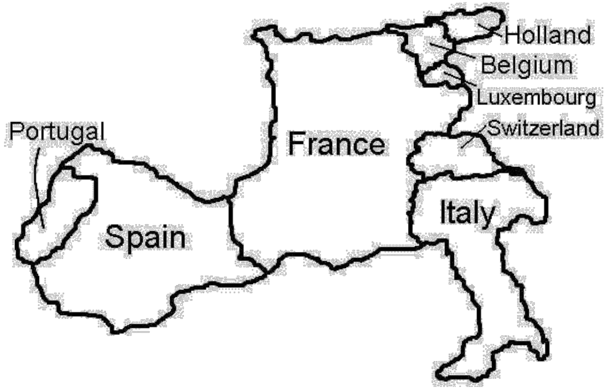


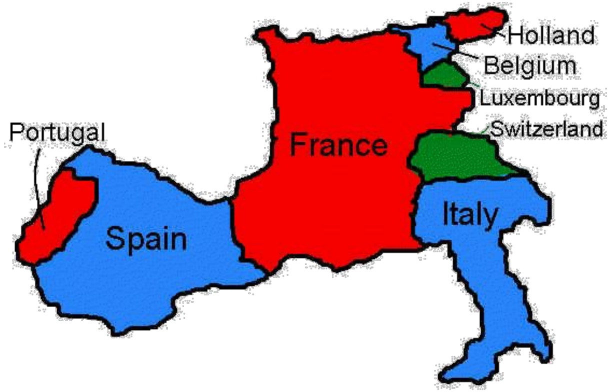
Goal State

Search Problem Formulation: Examples

Map Coloring: Given a 2D map of n countries and a set of K colors, color every country differently from its neighbors (countries with shared borders

- *States:* ??
- *Initial State:* ??
- *Actions:* ??
- *Goal Test:* ??
- *Path Cost:* ??





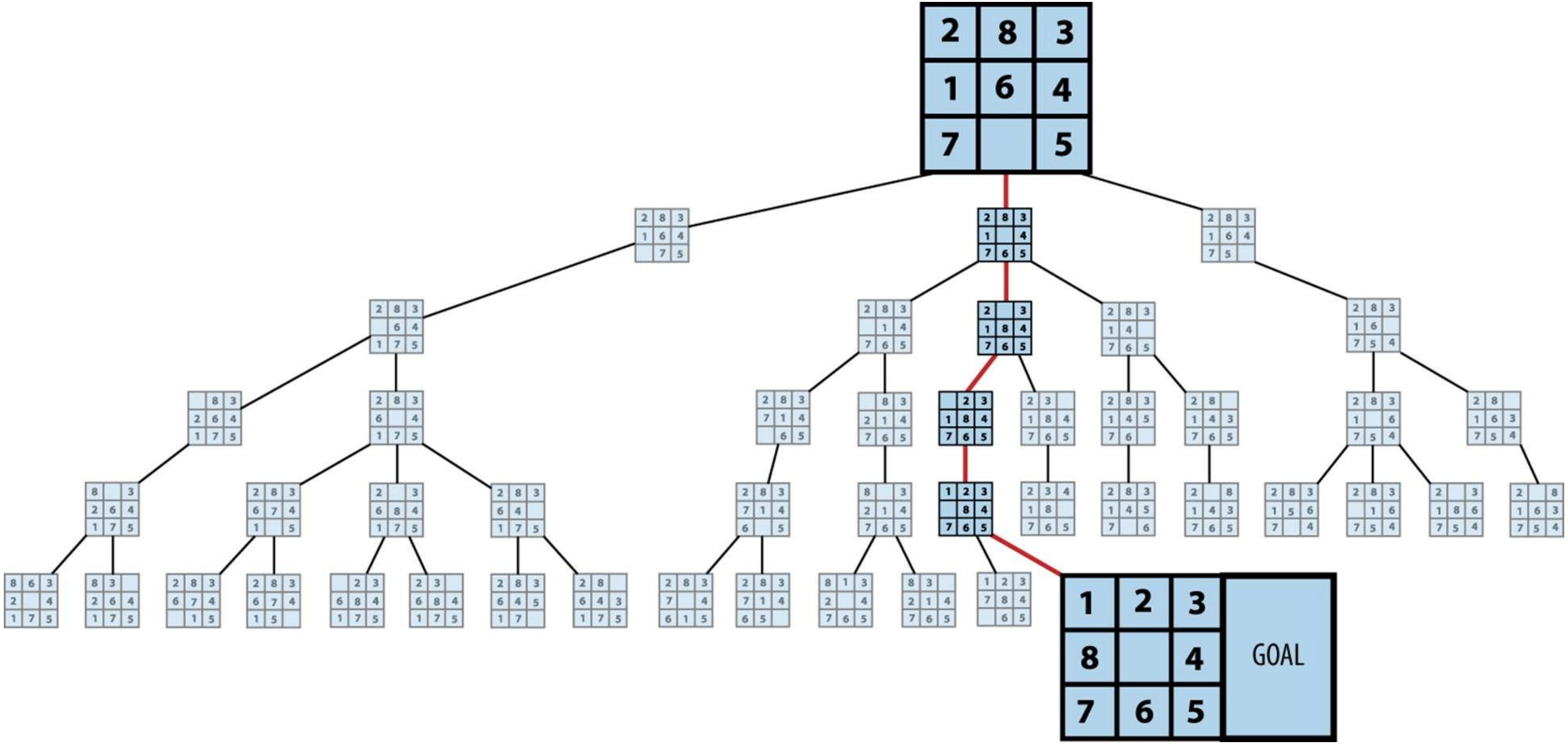
Modeling A Search Problem Using a Search Tree

- In most cases we can model the search space as a search tree or graph.
- When we model the search space as a tree, the tree has the following properties:
  - Root: initial state
  - Branches: each branch (edge) present one possible action
  - Nodes: results from applying actions and represent state in the state space
  - Path: results from applying a sequence of actions
  - Each tree has a depth, height, width (diameter)
- Expand Operation: given any node (state) creates all children nodes

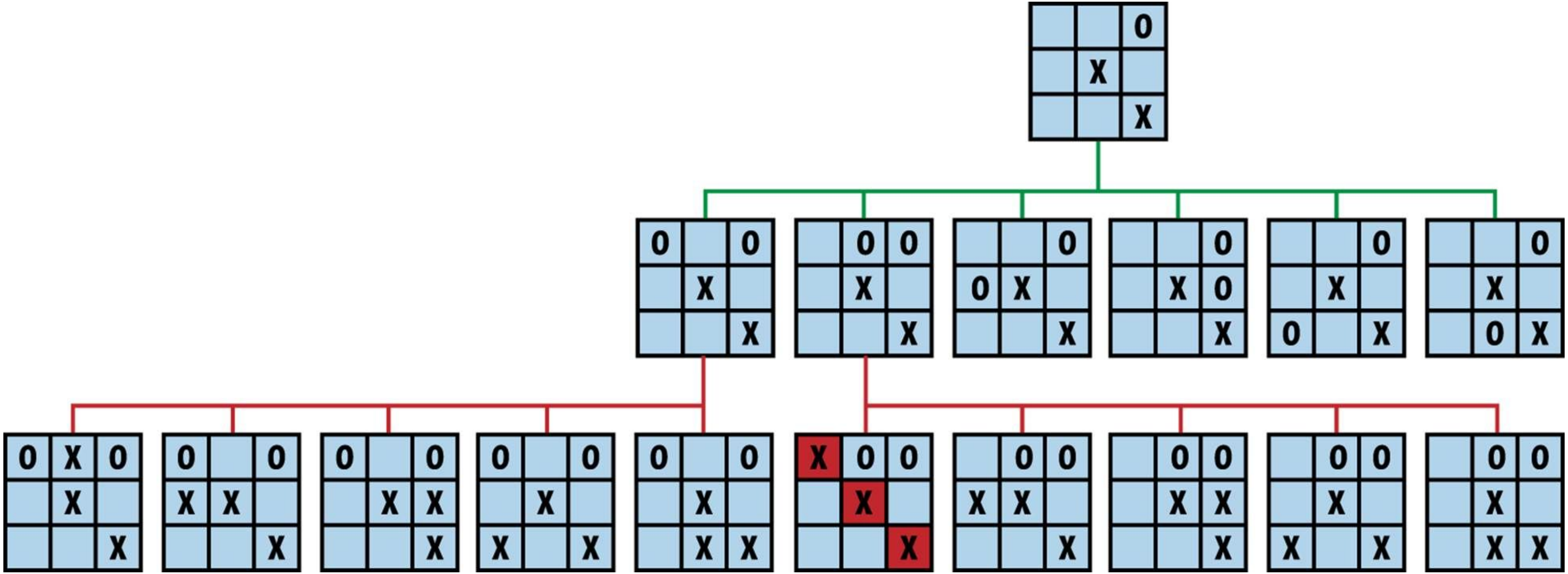
Working With Search Tree or Graph

- The nodes in the search (space) tree are divided into three sets:
  - Unexplored Set
  - Waiting Set
  - Visited Set
- The search algorithm move the nodes from unexplored set to the waiting set and finally to the visited set.

Search Tree Examples: 8 Puzzle Problem

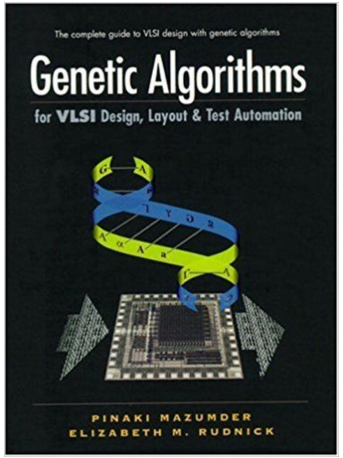


Search Tree Examples: Tic Tac Toe



Search Problems: Real World Examples

- Route Finding: Using Google Map or MapQuest
- Droid Navigation: Amazon Prime Air
- VLSI Layout: locate millions of component and connections on a chip to meet design constraints
- Social Media: Slack introduces a new search feature powered by artificial intelligence
- Automatic Assembling: find the order to assemble parts of an object (e.g. cars,)
- Pharmaceutical: Protein design, search for a sequence of amino acids in a 3D protein to cure some disease



Depth First Search Algorithm

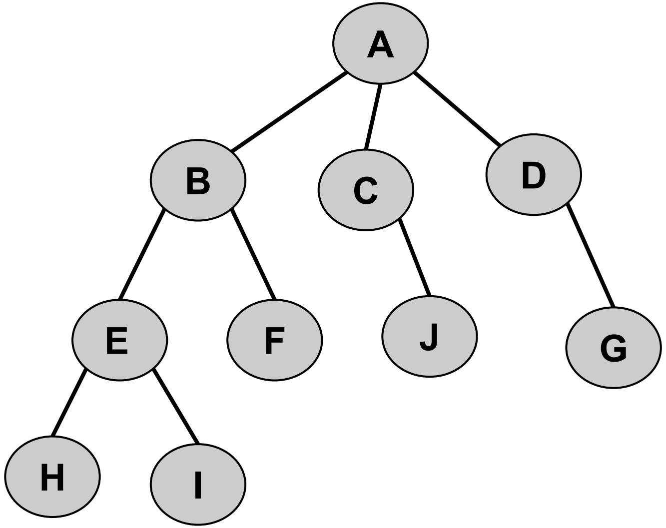
Uninformed Search Algorithm: Depth First Search

What is Depth First Search?

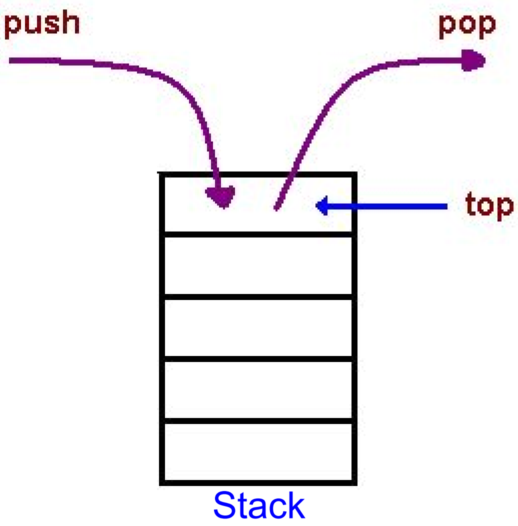
- A a general algorithm for graph traversal (searching tree or graph data structure)
- Works on directed and undirected graphs
- Implemented using a data structure called stack.
- Time Complexity:
  - $O(|V|+|E|)$ traversed without repetition
  - $O(b^d)$ in implicit graph (where b is the branching factor and d is the depth)
- Space Complexity: $O(|V|)$

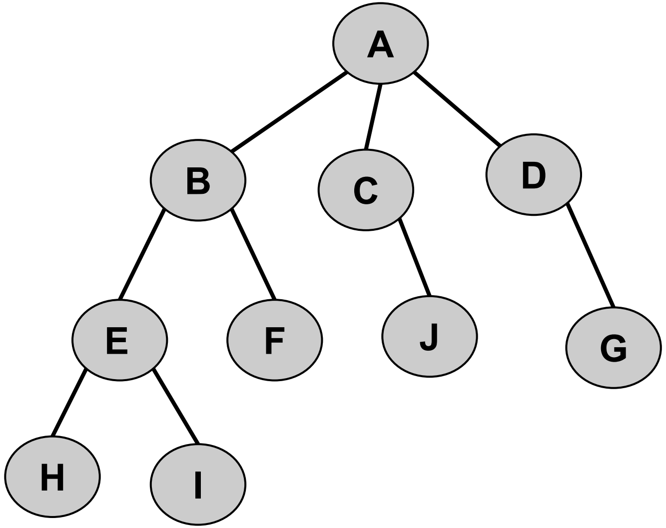
Depth First Search: How it works?

- Go forward (in depth) as long as it is possible, if not then backtrack
- Backtrack means you reached a dead-end (e.g. leaf node in a search tree)
- We need to keep track of visited (explored) nodes, so we do not visited a node infinite times.



Depth First Search: In Action





Visited Node

output:

Depth First Search: Pseudo Code

```
def DFS(G,v):
    # let S be a stack
    for u in V of G:
        visited[u] = False
    S = stack()
    S.push(v)
    while S.isEmpty() == False:
        v = S.pop
        if v not in visited:
            visited[v] = True
            for w in neighbours of v and visited[w] is False:
                S.push(w)
```

Depth First Search in AI

The Depth-first algorithm is not complete, (will not always find a solution) why??

Depth First Search in AI

- When should we use it?
    - Space (storage) is restricted;
    - Many solutions exist, perhaps with long path lengths, where nearly all paths lead to a solution;
    - The order of the neighbors of a node are added to the stack can be tuned so that solutions are found on the first try.
- When we should not use it?
    - It is possible to get caught in infinite paths; this occurs when the graph is infinite or when there are cycles in the graph; or
    - solutions exist at shallow depth, because in this case the search may look at many long paths before finding the short solutions.

Constraint Satisfaction Problems

- What is a Constraint Satisfaction Problems (CSPs)?
    - are mathematical problems where one must find states or objects that satisfy a number of constraints or conditions
- Formally a **CSP** is defined by a triple (V, D, C):
    - V is a set of variables
    - D is a domain of values
    - C is a set of constraints
- A **constraint** is a combination of valid values for the variables.

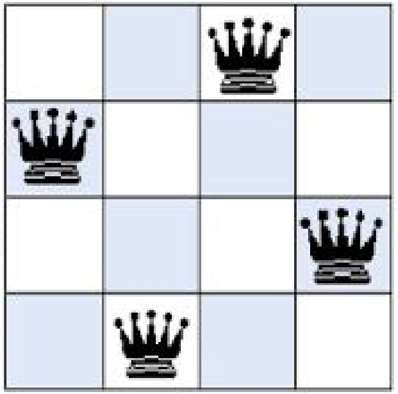
Constraint Satisfaction Problems (cont)

- A state of the problem is defined by an **assignment** of values to some or all of the variables.
- A **solution** to a CSP is an assignment that satisfies all the constraints
- Examples:
  - N-Queens ??
  - 8-Puzzels ??
  - Map Coloring ??
  - Cryptarithmetic (**Verbal arithmetic**) ??

N-Queens as a CSP

N-Queens: *Given an* ***n x n*** *chessboard, arrange n queens so that none is attacking another.*

- *Variables:* Q_i *for each row i of the board*
- *Domain: {1, 2, 3, …, n} for position in row*
- *Constraints: ??*



Solving Constraint Satisfaction Problems

- One common method to solve CSPs is using Depth First Search
- Avoid using a Naive DFS algorithm by applying the following techniques:
  - Backtracking
  - Forward Checking
  - Constraint Propagation

DFS Improvements

- Consider only actions that will not violate any constraints.
- Predict valid actions ahead
- Do not explore branches or paths that obviously will not lead to a solution
- Use a controlled procedure to for variables and values assignments

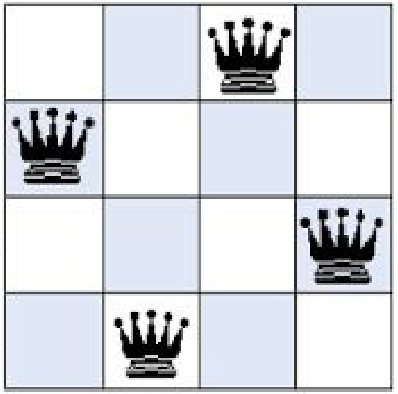
DFS Improvements:

- Backtracking
  - After each action we need to check for the goal and the constraints
  - When there is no valid action backtrack to the previous node (state)
- Forward Checking
  - Keep Track of available legal values for unassigned variables
  - Terminate when the solution is found
- Constraint Propagation
  - Update the values in the domain based on the last action

Depth First Search and N-Queens

4-Queens: *Given an **4 x 4** chessboard, arrange n queens so that none is attacking another.*

- *States: 43680 => (16 x 15 x 14 x 13)*
- *Initial State: the board is empty (no queen on the board)*
- *Actions: Add or move a queen to any empty square.*
- *Goal Test: 4 queens the board with none attacked*



Depth First Search and N-Queens

What does a node in the search tree represent?

How can we validate an actions?

How can we test for goal state?

What should we do when we reach a deadend?

Appendix Graph Data Structure

What is a Graph (data structure)?

- Graph: A graph **G** is a pair of sets **(V, E)**, where **V** is the set of vertices and **E** is the set of edges, connecting the pairs of vertices.
- Vertex: Each node of the graph is represented as a vertex.
- Edge: Edge represents a path between two vertices or a line between two vertices.
- Adjacency: Two node or vertices are adjacent if they are connected to each other through an edge.
- Path: Path represents a sequence of edges between the two vertices.

What is a Graph (data structure)?

- Directed Graph: In the directed graph, each edge is defined by ordered pair of vertices.
- Non-Directed Graph: In the undirected graph, each edge is defined by unordered pair of vertices
- Connected graph: In the connected path, there is a path from every vertex to every other vertex.
- Non Connected Graph: In the non-connected graph, path does not exist from any vertex to any other vertex.

What is a Graph (data structure)?

- Weighted Graph: In the weighted graph, some weight is attached to the edge.
- Tree: is considered as a special case of graph. It is also termed as a minimally connected graph.
  - Every tree can be considered as a graph, but every graph cannot be considered as a tree.
  - Self-loops and circuits are not available in the tree as in the case of graphs.

Questions