State Space Search

- Many problems can be represented as a set of states and a set of rules of how one state is transformed to another.
- The problem is how to reach a particular goal state, starting from an initial state and using the state traversing rules.

Search Classification

- Some path
  - Depth first
  - Breadth first
  - Hill climbing
  - Beam search
  - Best first

- Optimal path
  - British museum
  - Branch and bound
  - Dynamic programming
  - A*

- Games
  - Minimax
  - Alpha-beta pruning
  - Progressive deepening
  - Heuristic pruning

Search Classification

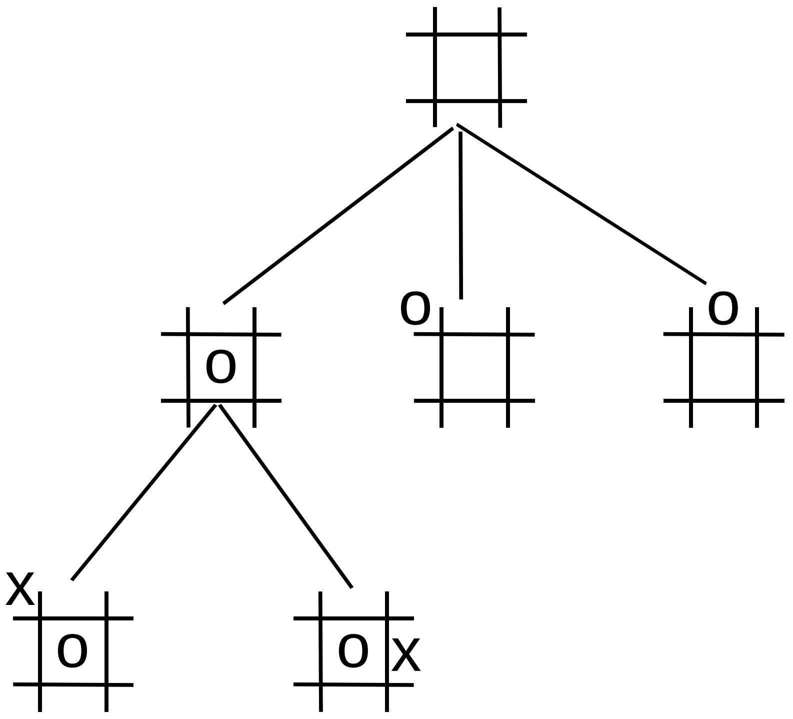
- Strategies for finding a goal. No concern of the cost associated with it.
    - No informomation of the search space (e.g. depth first)
    - Some information on the structure of the search space such as estimates to the distance to the goal (e.g. best first)

- Strategies for finding a minimum cost path to the goal (e.g. branch and bound)

- Strategies for finding a goal in presence of adversaries such as game playing (e.g. A-B pruning)

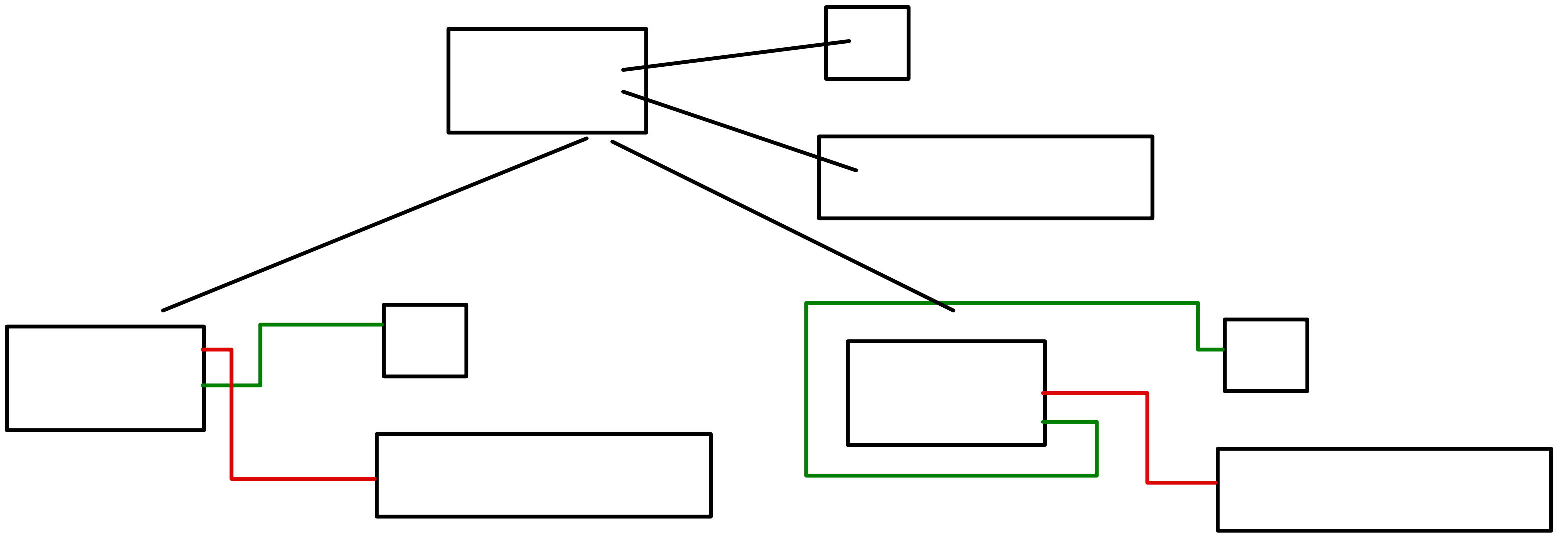
State Space Representation Example

- Games: tic-tac-toe, chess, etc

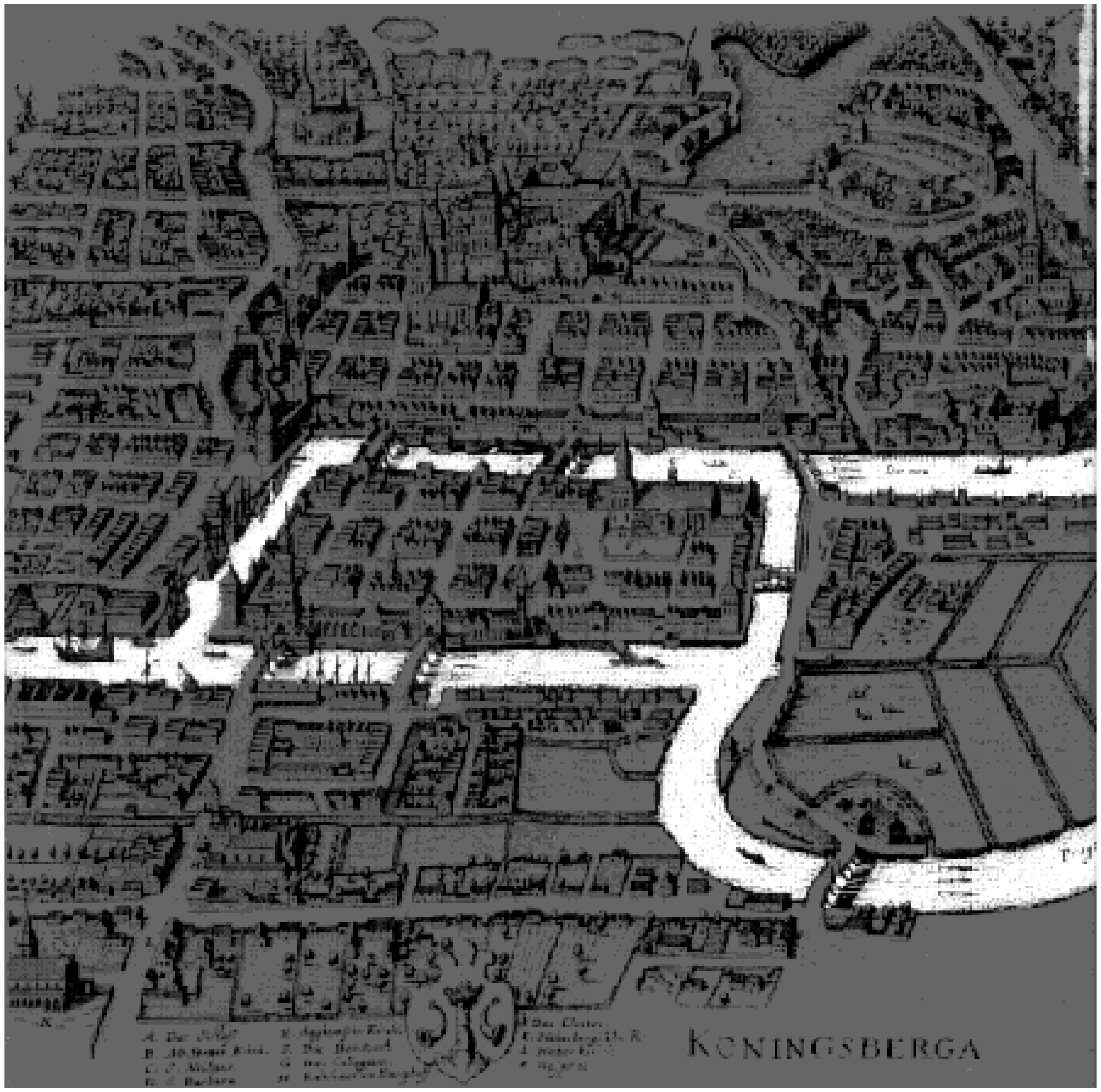


State Space Representation Example

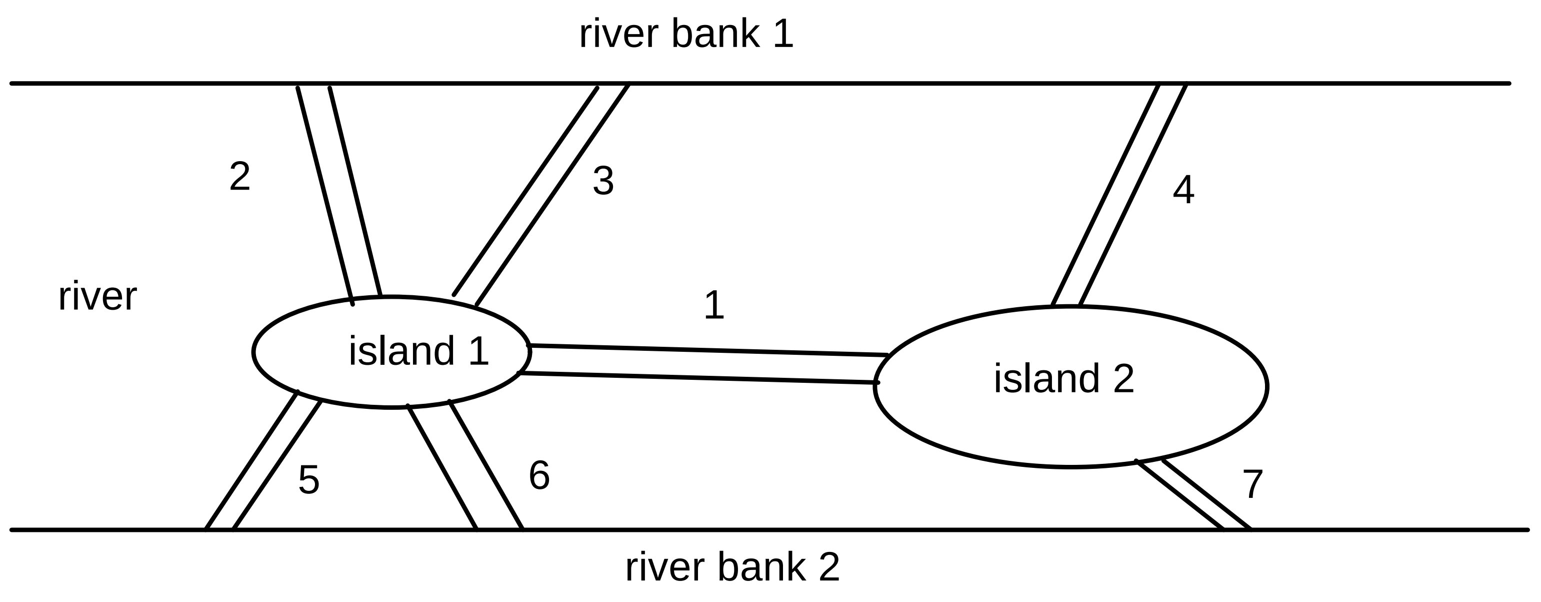
- VLSI routing



The City of Königsberg



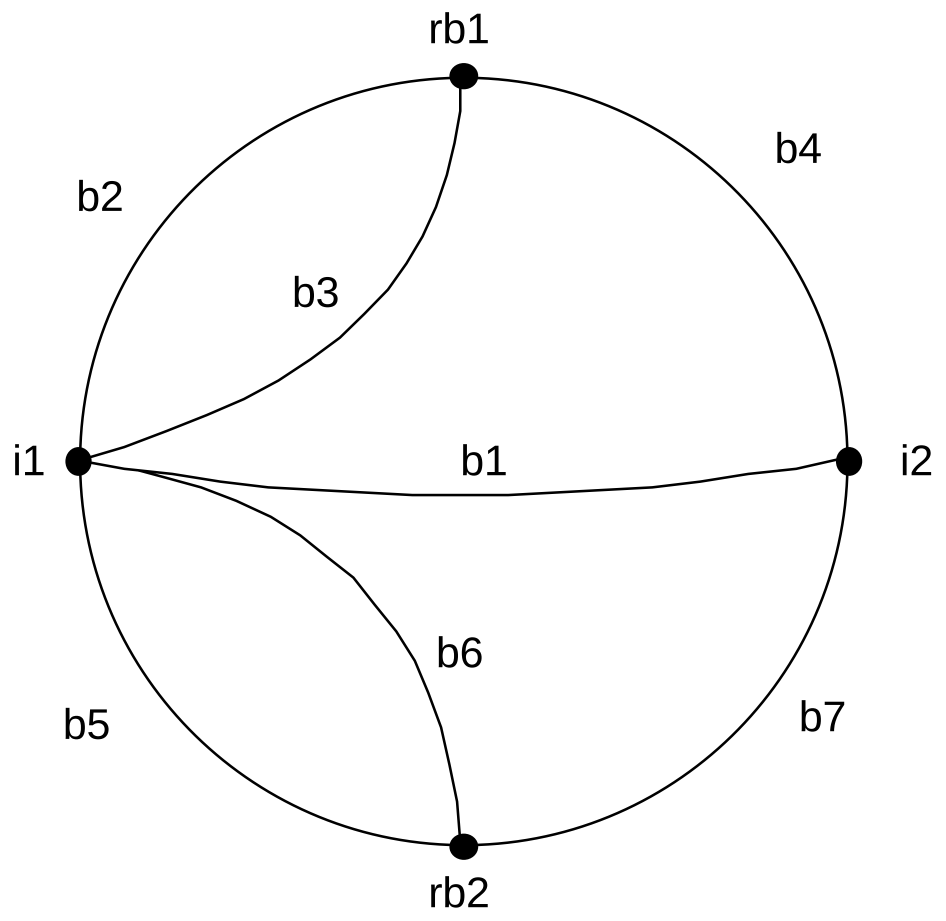
A Mathematical Problem: Euler’s Graph Theory



There are 2 river banks, a river, 2 islands, and 7 bridges.

Q: Is there a walk around the city that crosses each bridge exactly once?

Graph of the Königsberg Bridge System



A solution exists if there is either zero or two nodes with odd degree.

Representation of the Königsberg Bridge System

- Predicate calculus representations

connect (i1,i2,b1)	connect (i2,i1,b1)
connect (rb1,i1,b2)	connect (i1,rb1,b2)
connect (rb1,i1,b3)	connect (i1,rb1,b3)
connect (rb1,i2,b4)	connect (i2,rb1,b4)
connect (rb2,i1,b5)	connect (i1,rb2,b5)
connect (rb2,i1,b6)	connect (i1,rb2,b6)
connect (rb2,i2,b7)	connect (i2,rb2,b7)

- How to know the degree of a node?
- Graphs are convenient in representing many problems.

Graph Theory: Definitions

- A graph $G = (N,E)$ consists of a set of nodes N and a set of edges E. The set of edges $E \subseteq N \times N$ defines how the nodes are connected.
- A directed graph has edges which incorporate direction. That is, if $e = (a,b) \in E$, $e' = (b,a)$ is not necessarily in E.
- In a directed graph if $(a,b) \in E$ (i.e., a is connected to b) then a is the parent of b, while b is a child of a.
- A rooted graph has a unique node N_s (the root) from which all paths in the graph originate.
  - The root has no parents.
  - A leaf node has no children.

Graph Theory: Definitions (cont'd)

- An ordered sequence of nodes [n_1, n_2, ..., n_k] where n_i is the parent of n_{i+1}, i = 1, 2, 3..., k-1, is called a <u>path</u>.
  - A path that contains any node more than once is said to contain a <u>loop</u> or a <u>cycle</u>.
- Two nodes in a graph are <u>connected</u> if there exists a path between them.
- The <u>degree</u> of a node is the number of edges incident to that node.
- A graph is <u>regular</u> if all its nodes have the same degree.
- The degree of a regular graph is the degree of its nodes.
- A <u>tree</u> is a graph where there is a unique path between any pair of nodes.
- The edges in a tree are called <u>branches</u>.
  - The <u>branching factor</u> of a node in a tree is the number of its children.

State Space Search: Definitions

- A state space is represented by a four-tuple [*N*,*E*,*S*,*G*].
- *N* is the set of nodes of the graph, correspond to the states in a problem-solving process.
- *E* is the set of edges between nodes, corresponding to the steps in a problem-solving process.
- *S*, a nonempty subset of *N*, contains the start/initial state(s) of the problem.
- *G*, a nonempty subset of *N*, contains the goal state(s) of the problem.
- The states in *G* are described using either:

  1. A measurable property of the states encountered in the search.

  2. A property of the path developed in the search.

Tic-Tac-Toe Partial State Space

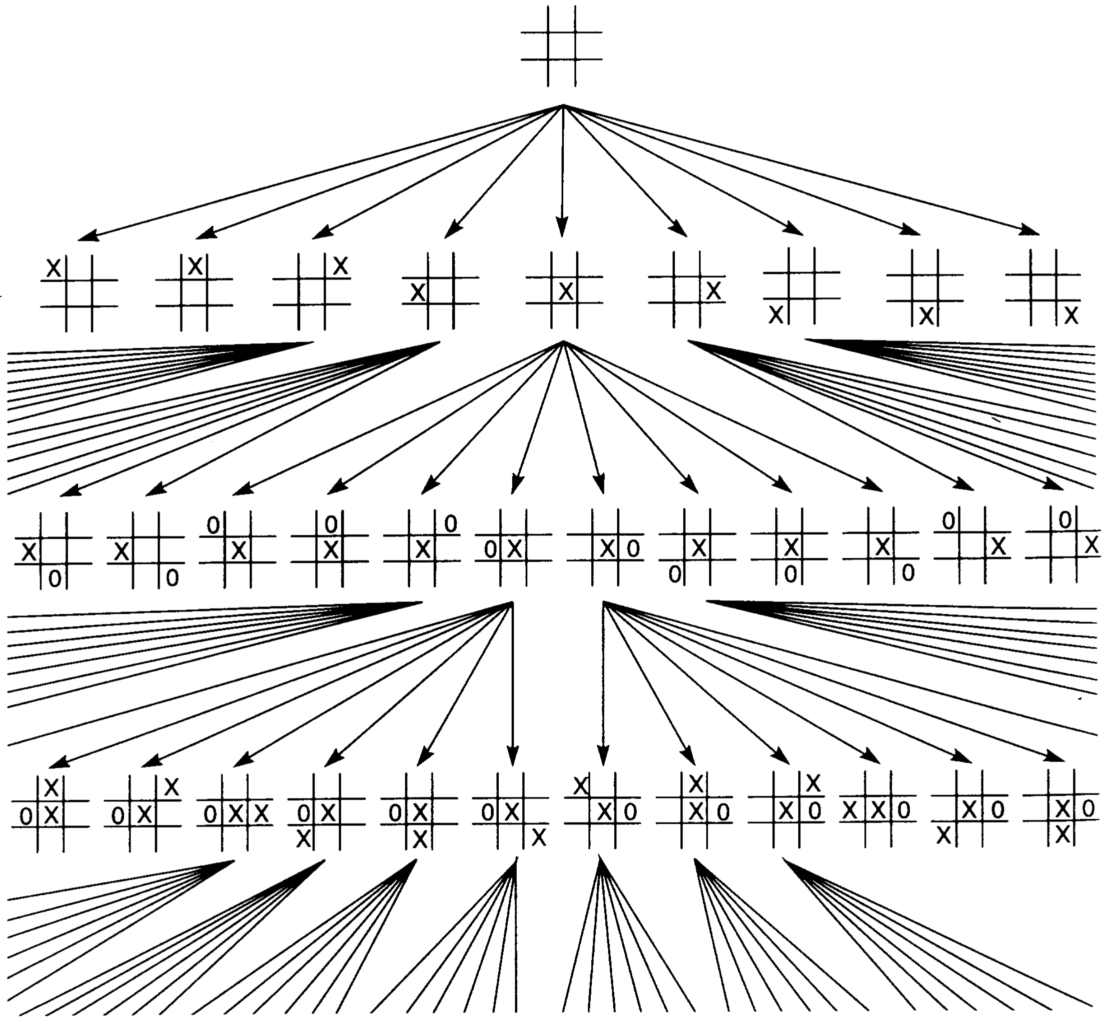


Figure II.5 Portion of the state space for tic-tac-toe.

The 15-Puzzle and the 8-Puzzle

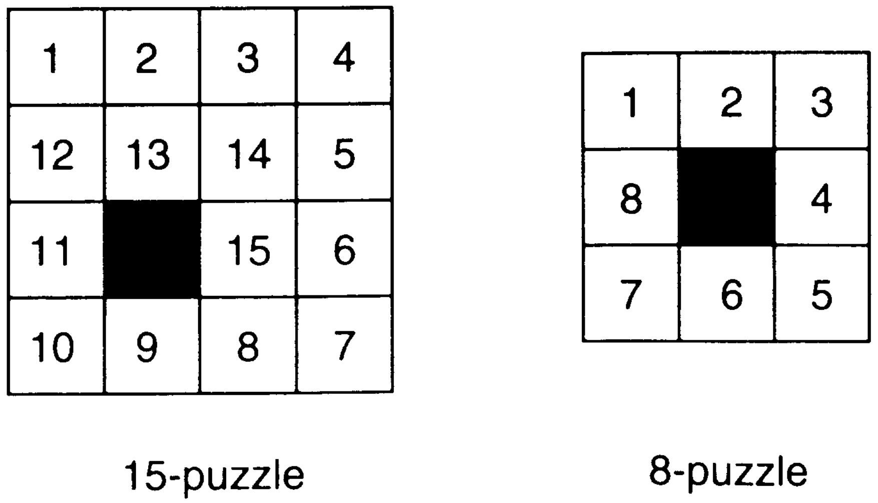


Figure 3.5 The 15-puzzle and the 8-puzzle.

8-Puzzle Partial State Space

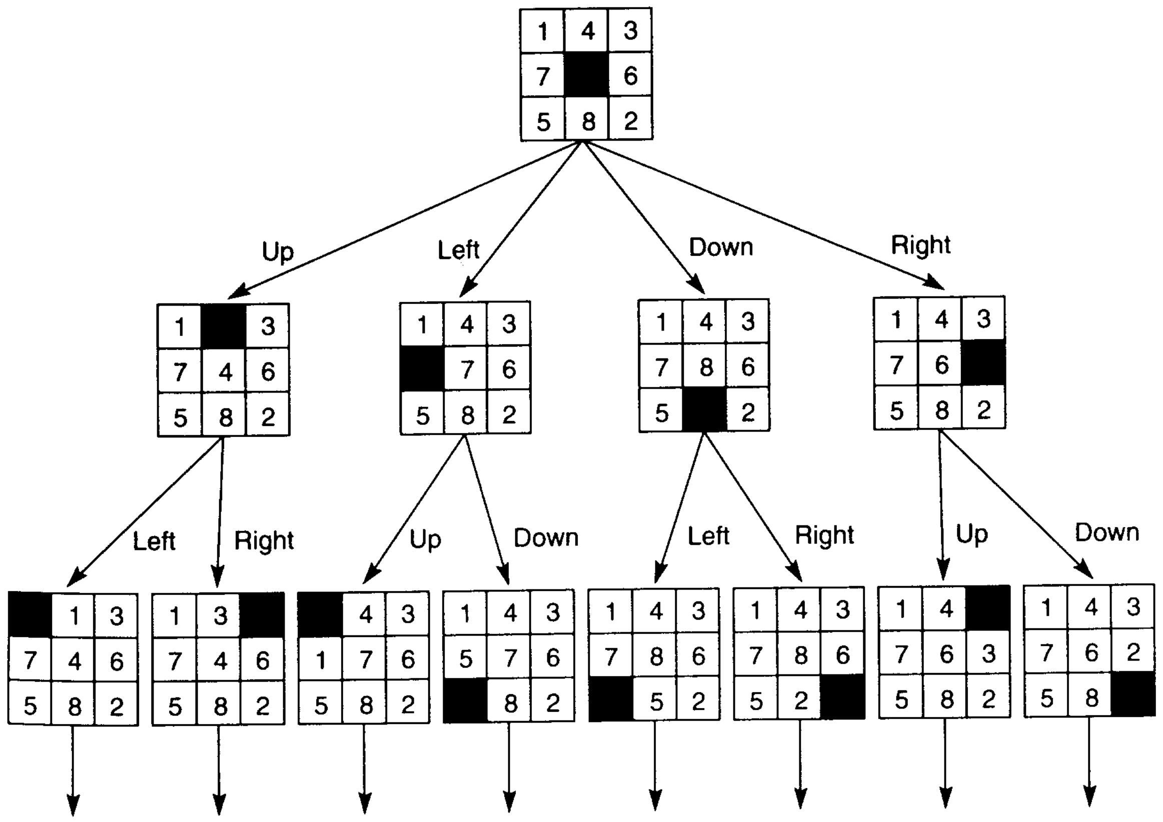


Figure 3.6 State space of the 8-puzzle generated by "move blank" operations.

The Travelling Salesman Problem

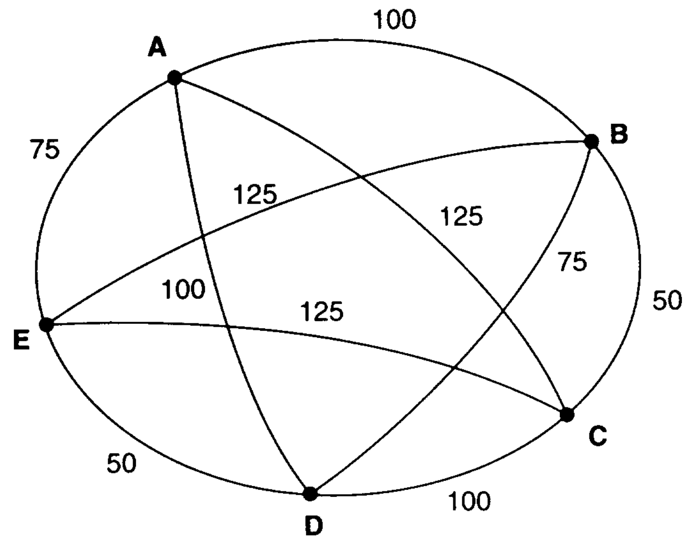


Figure 3.7 An instance of the traveling salesperson problem.

Find the shortest path
- **visit all cities**
- **return home**

Travelling Salesman Partial State Space

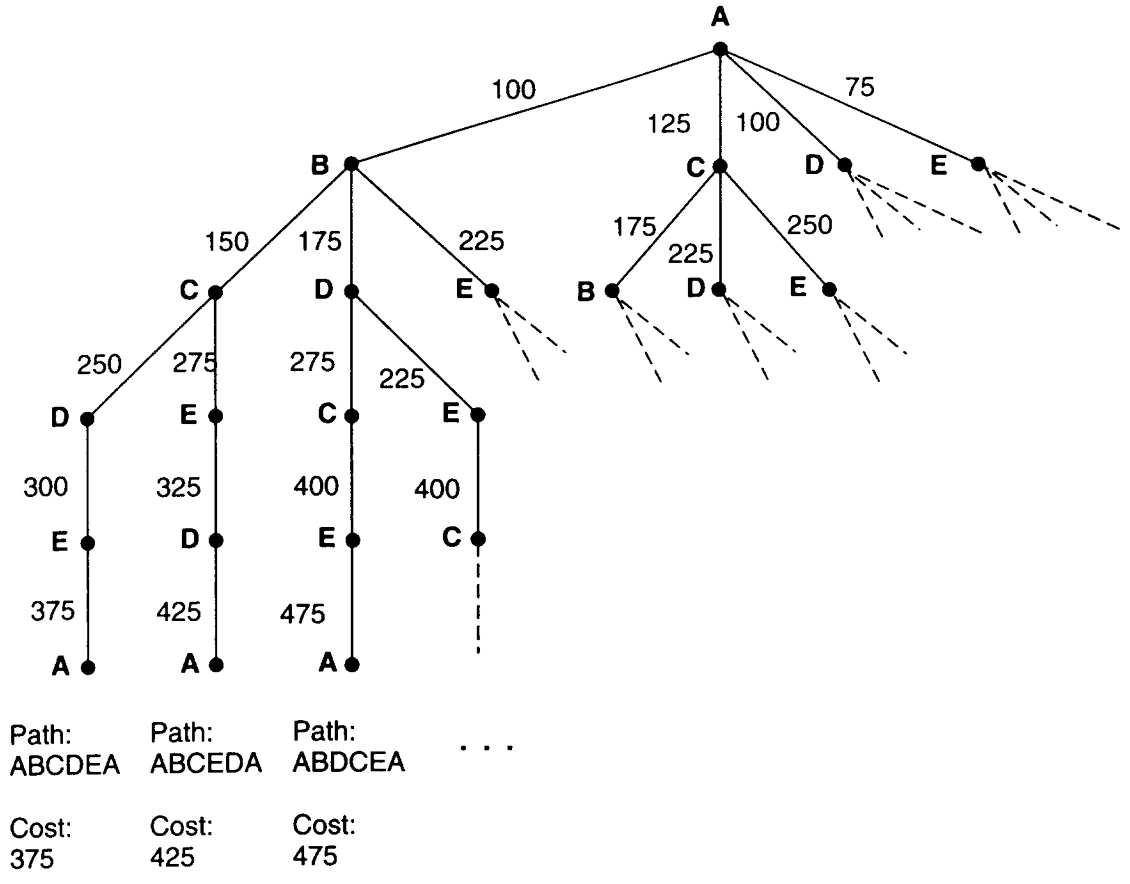


Figure 3.8 Search of the traveling salesperson problem. Each arc is marked with the total weight of all paths from the start node (A) to its endpoint.

Search Algorithms

- A <u>solution path</u> is a path from a node in *S* to a node in *G*.
- A <u>search algorithm</u> finds a solution path through the problem space.
- Multiple paths to a state can lead to cycles in a solution path that prevent the algorithm from reaching a goal.
- General graph search algorithms must detect and eliminate loops from potential solution paths.
- A tree search can gain efficiency by eliminating this test and its overhead.
- For search purpose, one can convert a general graph to a tree by eliminating all possible loops.
- On any path in the tree (from the root to the present node), do not include a node twice.

Search Strategy

- Measuring problem-solving performance:
  - does it find a solution at all?
  - is it a good solution (one with a low path cost)?
  - what is the search cost associated with the time and memory required to find a solution?

- The total cost of the search is the sum of the path cost and the search cost.

- Search Strategy is evaluated in four criteria:
  - completeness
  - time complexity
  - space complexity
  - optimality/admissibility

- Informed search (heuristic search) versus uninformed search (blind search)

Backtracking

- A problem solver or search algorithm must consider one or more paths through the state space until it reaches a goal node.
- Backtracking is a systematic technique to explore nodes in the state space.
- A path might lead to a goal node or a ‘dead end’ (i.e., the leaf node is not a goal node).
- Upon reaching a dead end, a search algorithm backtracks to examine unexplored nodes for possible solution path.

Depth-First Search

- In depth first, a node is expanded and then one of its children is chosen to be expanded next. This continues until the goal is found or a backtrack is needed. If a dead end is found at a node, one of its siblings is chosen to be expanded next.

1. Form a one-element queue consisting of the root node.
2. Until the queue is empty or the goal has been reached, determine if the first element of the queue is the goal node.

    2a. If the first element of the queue is not the goal node, remove the first element from the queue and add the first element's children, if any, to the <u>front</u> of the queue.

The 8-Puzzle: Depth-First Search

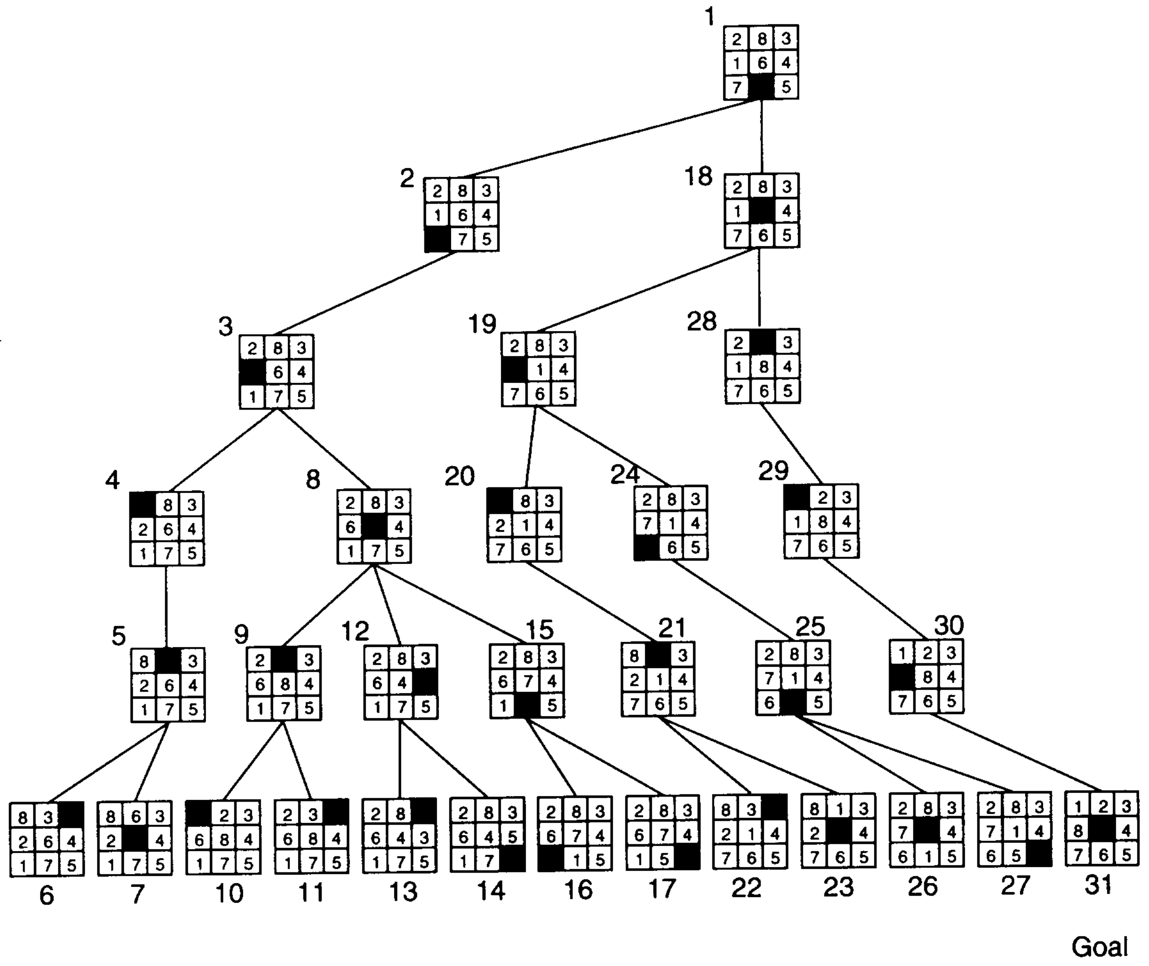


Figure 3.17 Depth-first search of the 8-puzzle with a depth bound of 5.

Breadth-First Search

- In breadth-first, all the siblings of a node are explored before their children are expanded.

1. Form a one-element queue consisting of the root node.
2. Until the queue is empty or the goal has been reached, determine if the first element in the queue is the goal node.

   2a. If the first element is not the goal node, remove the first element from the queue and add the first element's children, if any, to the back of the queue.

The 8-Puzzle: Breadth-First Search

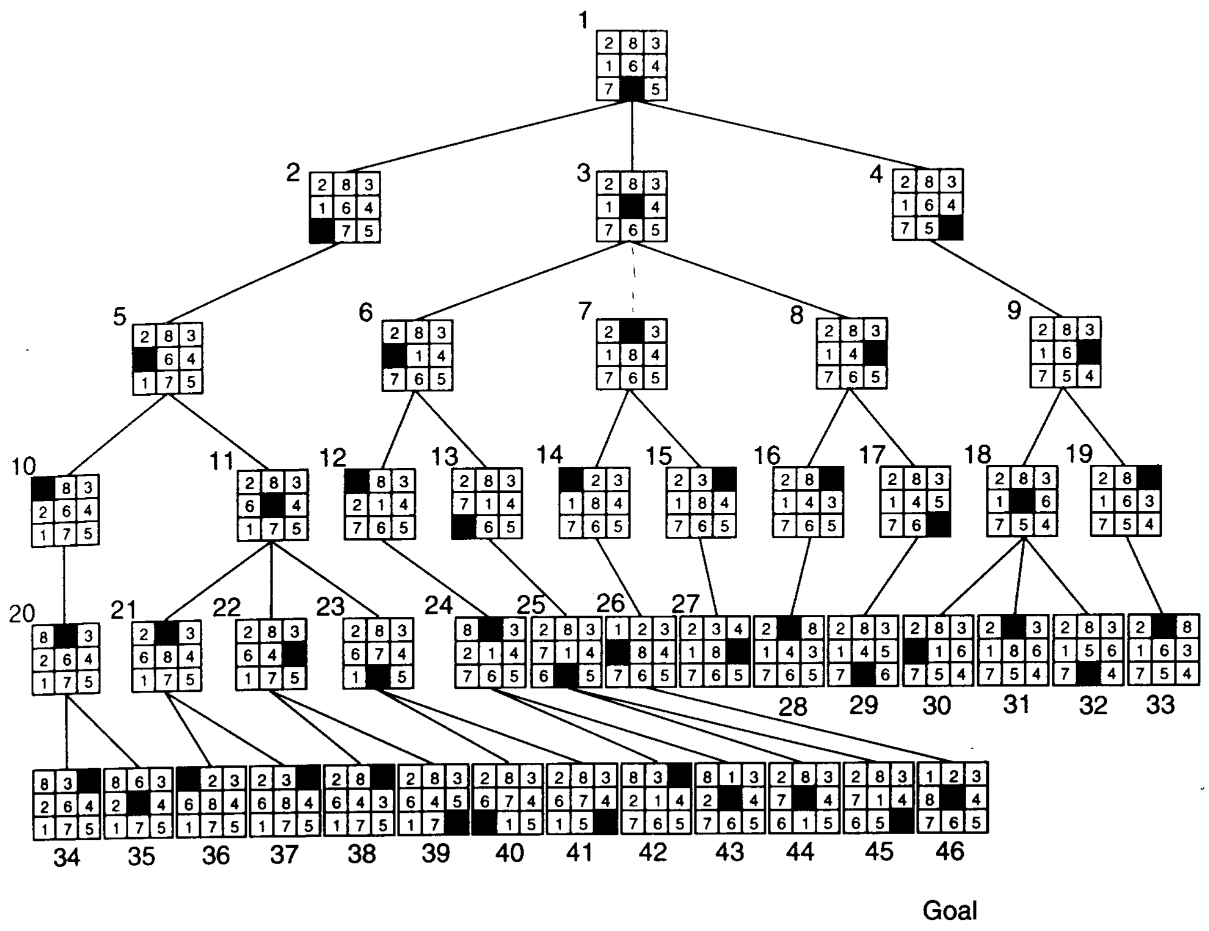


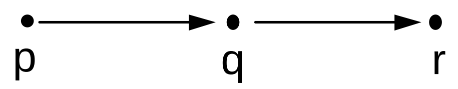
Figure 3.15 Breadth-first search of the 8-puzzle, showing order in which states were removed from open.

Breadth-First versus Depth-First

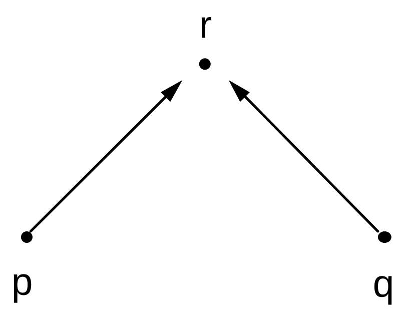
- Breadth-First Search
  - since all the nodes at level n are examined before nodes at level n+1, it always finds the shortest path to a goal node
  - efficient if the goal node is to be found at shallow depth in a deep tree
  - inefficient if the goal node is deep in the tree (space utilization, i.e., the number of elements in the queue, is an exponential function of the level)
- Depth-First Search
  - can be trapped in a deep tree, missing shorter paths to the goal node
  - efficient space utilization as the number of elements in the queue is a linear function of the length of the path, or the level of the node)
- Which is better? Depends on
  - the length of the solution path
  - the branching factor
  - shortest path desirability
  - space resource availability

Predicate Calculus: State Space Representation

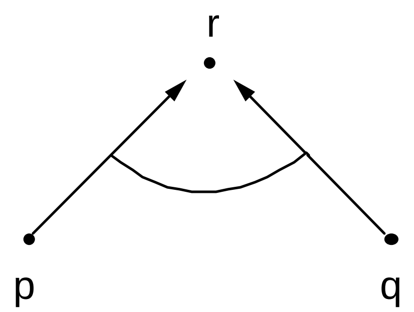
- A directed graph for (p⇒q and q⇒r)



- And/Or graph for (p∨q⇒r)



- And/Or graph for (p∧q⇒r)



Propositional Calculus Example

- Description given:

  $a \wedge b \Rightarrow d$

  $a \wedge c \Rightarrow e$

  $b \wedge d \Rightarrow f$

  $f \Rightarrow g$

  $a \wedge e \Rightarrow h$

- Current problem (given facts):

  a
  b
  c

Data-Driven and Goal-Driven Search

- Data-Driven Reasoning
  - takes the facts of the problem
  - applies rules and legal moves
  - produces new facts that eventually lead to a goal
  - also called forward chaining
- Goal-Driven Reasoning
  - focuses on the goal
  - finds the rules that could produce the goal
  - chains backward through subgoals to the given facts
  - also called backward chaining

Goal-Driven or Data-Driven

Goal-Driven search is recommended if:

- A goal or hypothesis is given in the problem statement or can easily be formulated.
- There is a large number of rules that match the facts of the problem and thus produce an increasing number of conclusions or goals.
- Problem data are not given but must be acquired by the problem solver.

Data-Driven search is recommended if:

- All or most of the data are given in the initial problem statement.
- There is a large number of potential goals, but there are only a few ways to use the facts and given information of the particular problem.
    - It is difficult to form a goal or hypothesis.

Direction of Reasoning

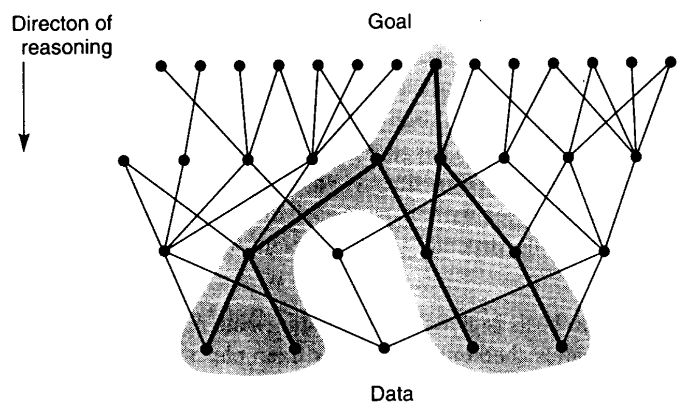


Figure 3.10 State space in which goal-directed search effectively prunes extraneous search paths.

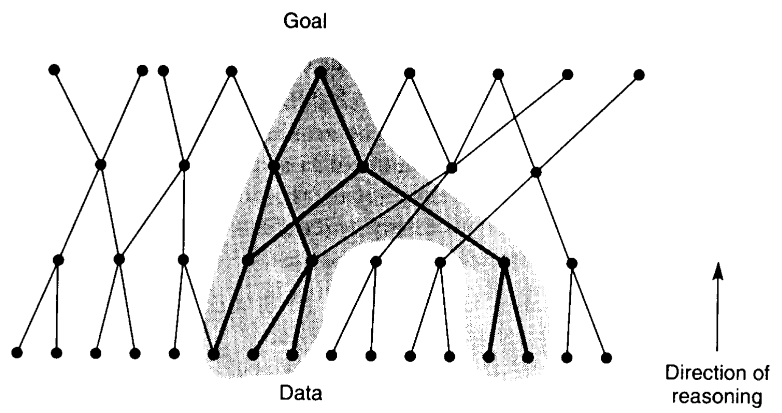


Figure 3.11 State space in which data-directed search prunes irrelevant data and their consequents and determines one of a number of possible goals.

Financial Advisor: And/Or Graph Search

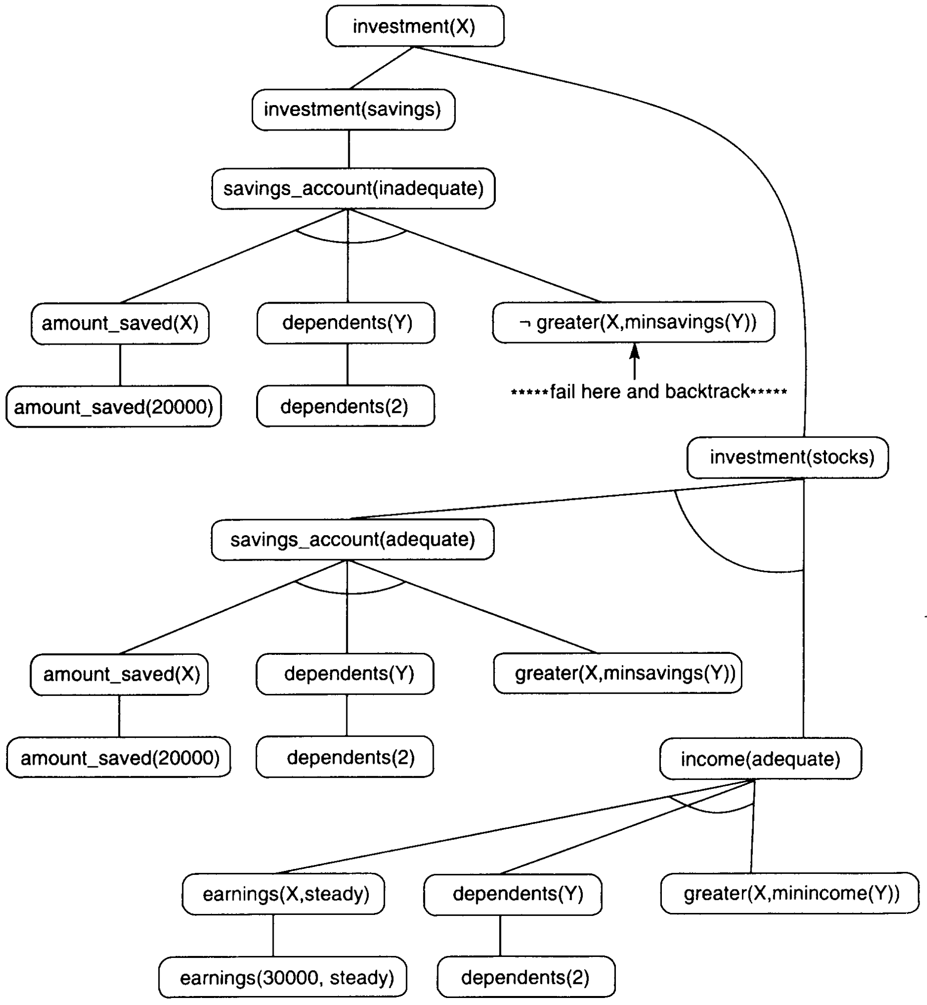


Figure 3.24 And/or graph searched by the financial advisor.