CS361 ***(Artificial Intelligence)***

Lecture 4
Problem Solving as Search
(Blind/Uninformed vs. Heuristic/Informed Strategies)

Dr. Hala Abdel-Galil & Dr. Amr S. Ghoneim
(Computer Science Dept.)

Helwan University
Fall 2019

Lecture is based on its counterparts in the following courses:

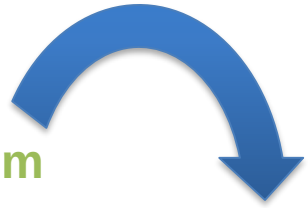
- *Artificial Intelligence*, **University of Illinois at Urbana-Champaign**

Resources for this lecture

- This lecture covers the following chapters/sections:
  - Chapter 3 (Structures & Strategies for State Space Search; sections 3.2, and 3.3) and Chapter 4 (Heuristic Search) *from* George F. Luger, "Artificial Intelligence: Structures and strategies for complex problem solving, " Sixth edition (2009), Pearson Education Limited.

Outline

- Search: Basic idea
  - Search tree
    - Tree Search Algorithm Outline
    - Handling repeated states
  - Backtracking Search
    - Backtracking Algorithm Data Structure
    - Backtracking Algorithm
- Blind vs. Heuristic Strategies
  - Blind Strategies
  - Depth-First Strategy
    - Depth-Limited Strategy
    - Comparison of Blind Search Strategies
  - Repeated States
    - Avoiding Repeated States
  - Uniform-Cost Strategy
  - Best-First Search



- Heuristic Function
  - Robot Navigation
  - Examples of Evaluation function
  - 8-Puzzle
  - Reasoning Representation
    - Propositional Calculus: Example (1)
  - And/ Or Graph
    - Propositional Calculus: Example (2)
    - Predicate Calculus Example
- More on Heuristic Search & Functions
  - Symmetry Reduction
  - Heuristic Reduction
    - Hill Climbing Strategy
    - Best-First Search
    - 8-Puzzle Heuristics

Recap: Search

Given:

- **Initial state**
- **Actions**
- **Transition model**
- **Goal state**
- **Path cost**

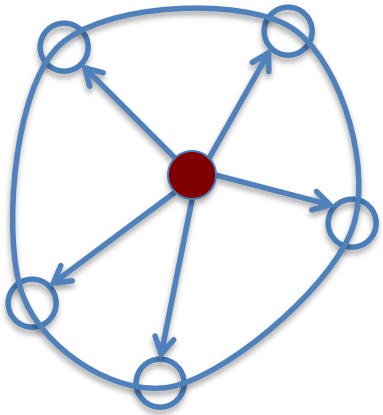
How do we find the optimal solution?

Recap: Search: Basic idea

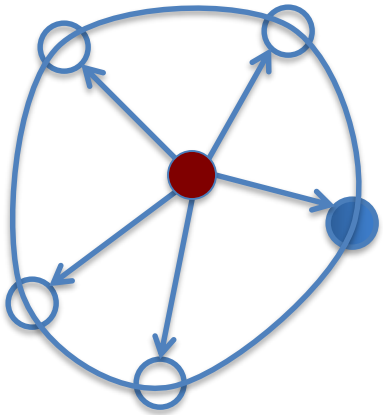
- Let's begin at the start state and **expand** it by making a list of all possible successor states.
- Maintain a **frontier** or a list of unexpanded states.
- At each step, pick a state from the frontier to expand.
- Keep going until you reach a goal state.
- Try to expand as few states as possible.

Recap: Search: Basic idea

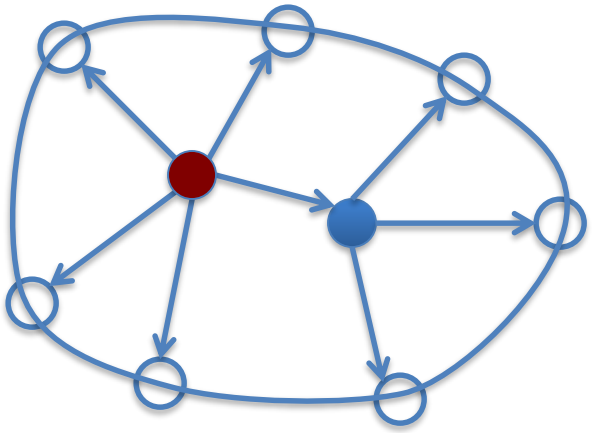
Recap: Search: Basic idea



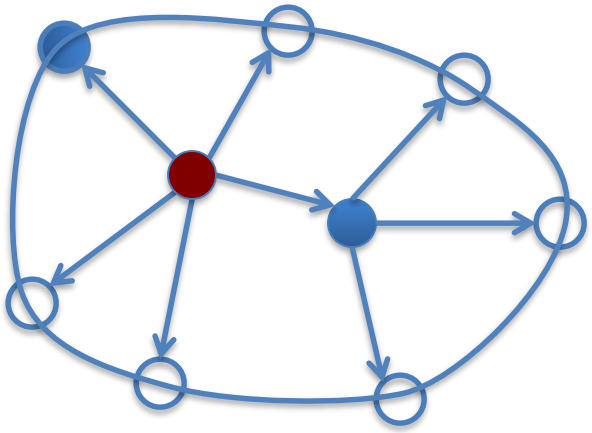
Recap: Search: Basic idea



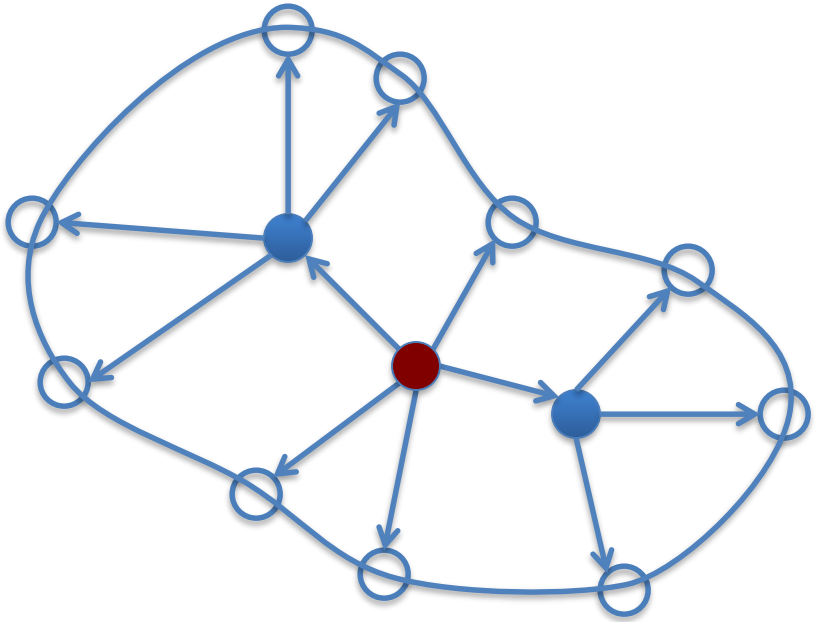
Recap: Search: Basic idea



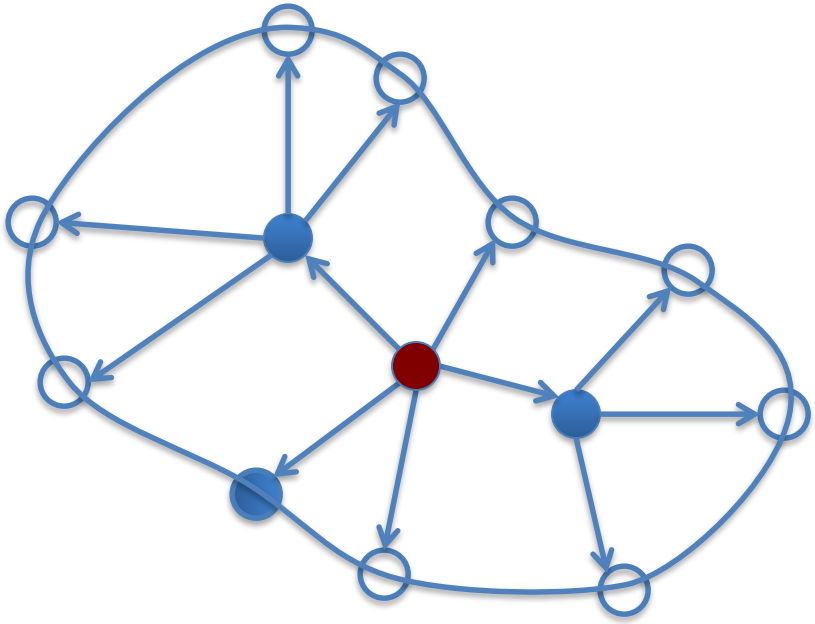
Recap: Search: Basic idea



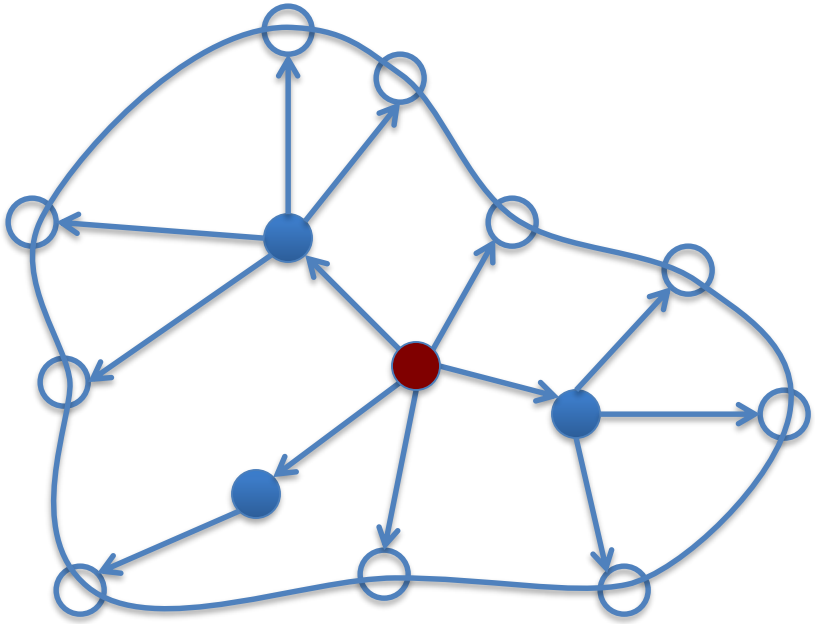
Recap: Search: Basic idea



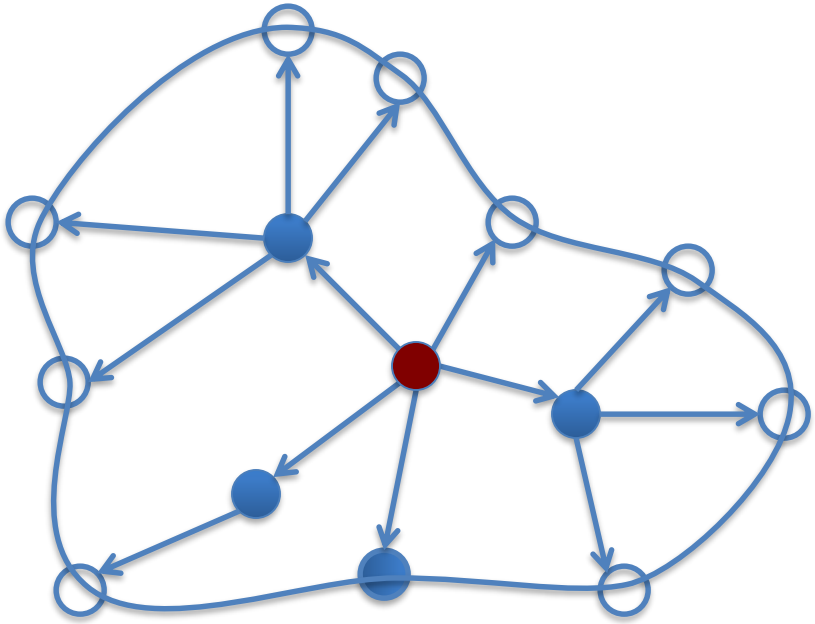
Recap: Search: Basic idea



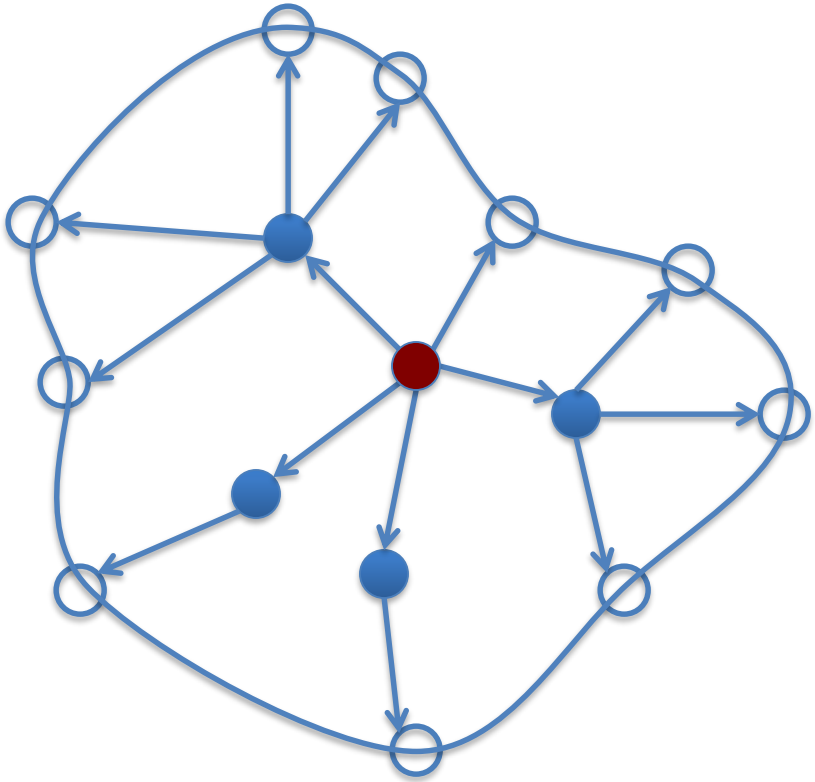
Recap: Search: Basic idea



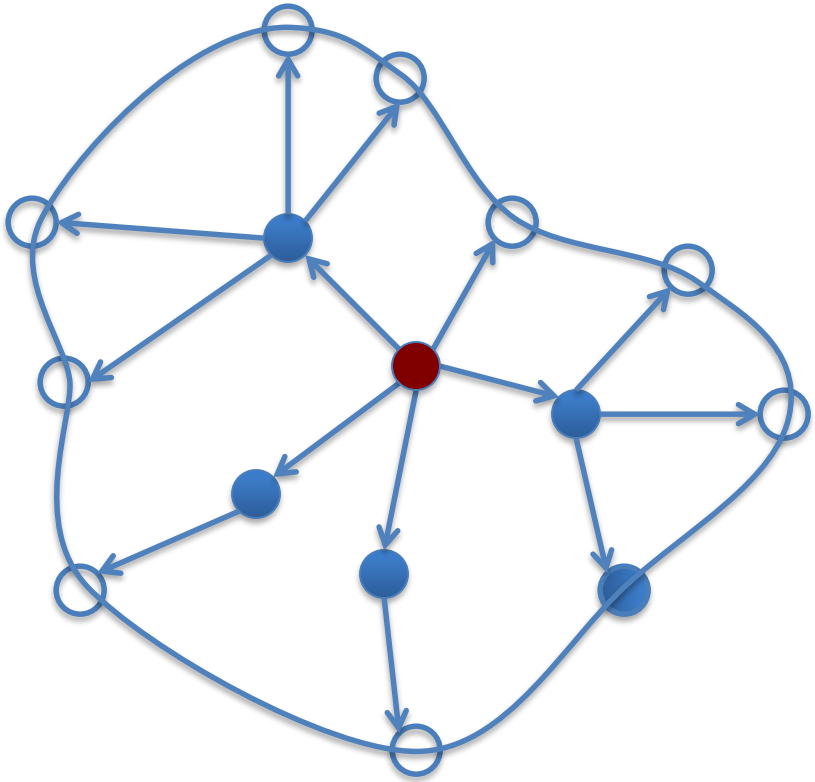
Recap: Search: Basic idea



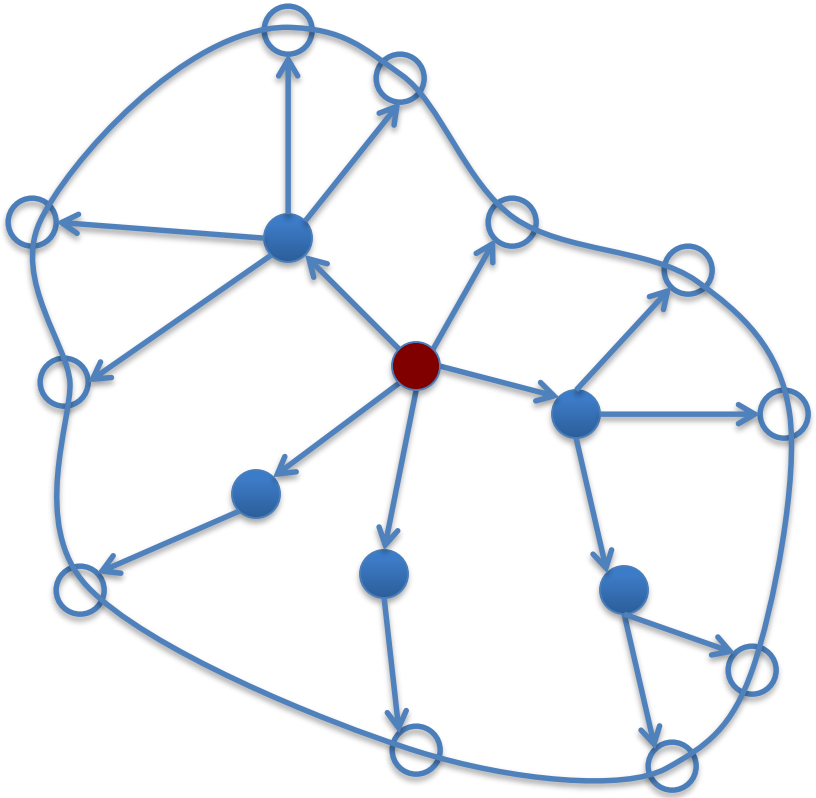
Recap: Search: Basic idea



Recap: Search: Basic idea



Recap: Search: Basic idea



Recap: Search tree

"What if" tree of sequences of actions and outcomes

- When we are searching, we are not acting in the world, merely "thinking" about the possibilities

The root node corresponds to the starting state

The children of a node correspond to the **successor states** of that node's state

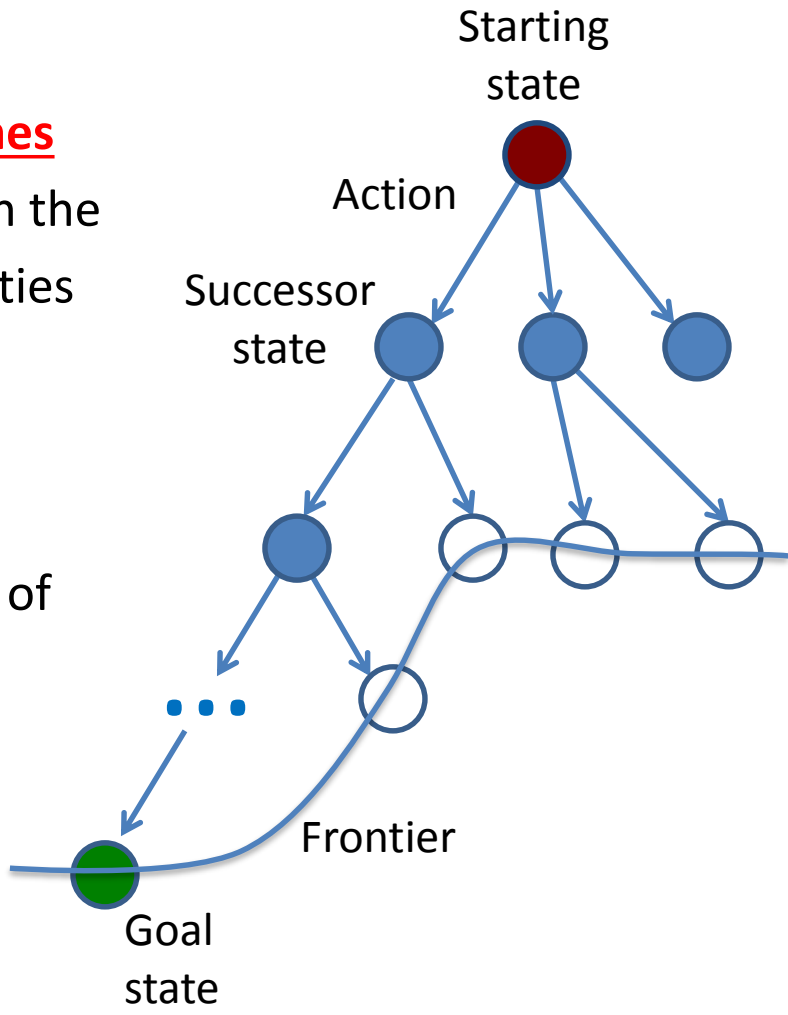
A path through the tree corresponds to a sequence of actions

- A solution is a path ending in the goal state

Nodes vs. **states**

A state is a representation of the world, while a node is a data structure that is part of the search tree

Node has to keep pointer to parent, path cost, possibly other info



Recap: Tree Search Algorithm Outline

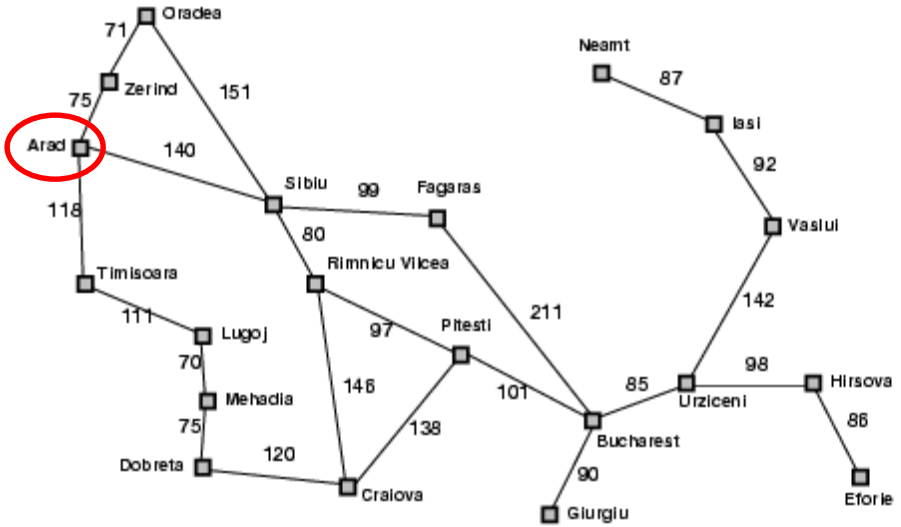
Initialize the **frontier** using the **starting state.**

While the frontier is not empty:

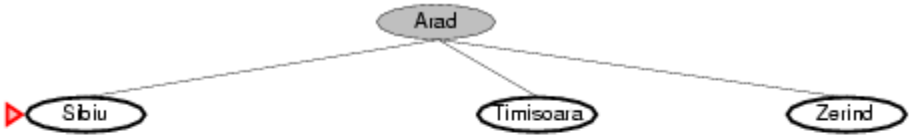
- Choose a frontier node according to **search strategy** and take it off the frontier.
- If the node contains the **goal state**, return solution.
- Else **expand** the node and add its children to the frontier.

Recap: Tree Search .. An example

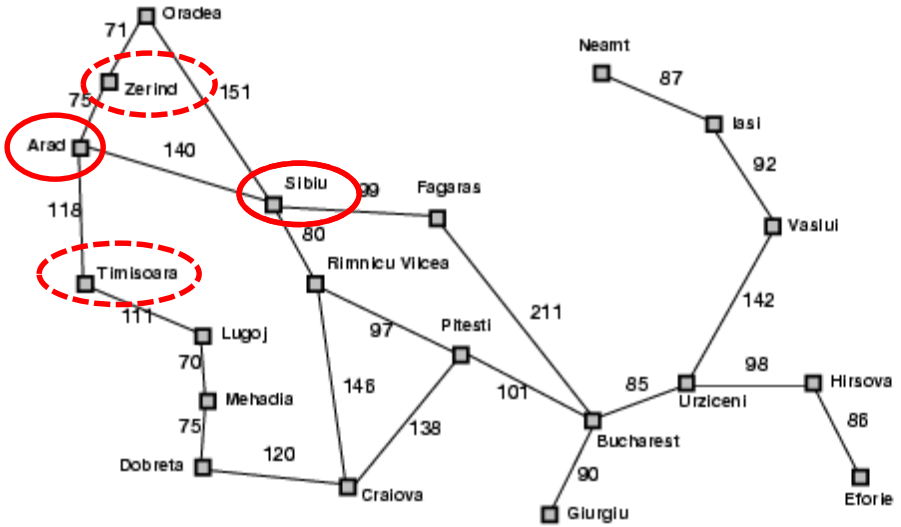
Start: Arad
Goal: Bucharest



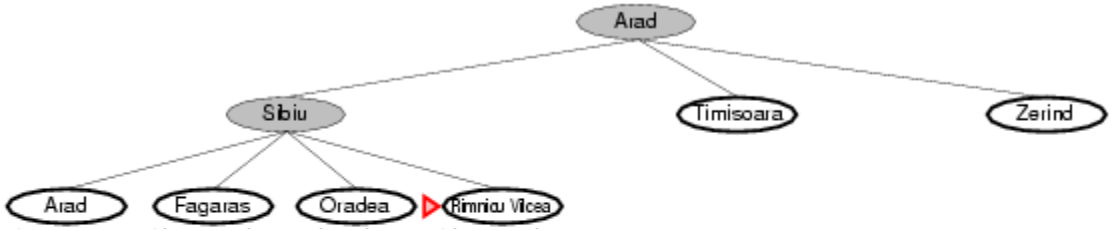
Recap: Tree Search .. An example



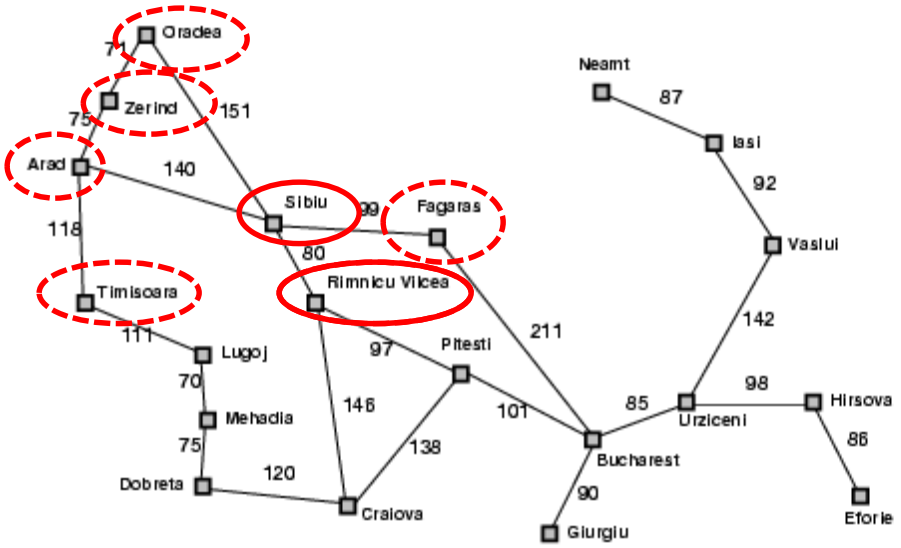
Start: Arad
Goal: Bucharest



Recap: Tree Search .. An example



Start: Arad
Goal: Bucharest



Recap: Handling Repeated States

Initialize the **frontier** using the **starting state**
While the frontier is not empty
    Choose a frontier node according to **search strategy** and take it off the frontier
    If the node contains the **goal state**, return solution
    Else **expand** the node and add its children to the frontier

To handle repeated states:
    Every time you expand a node, add that state to the **explored set**; do not put explored states on the frontier again
    Every time you add a node to the frontier, check whether it already exists with a higher path cost, and if yes, replace that node with the new one.

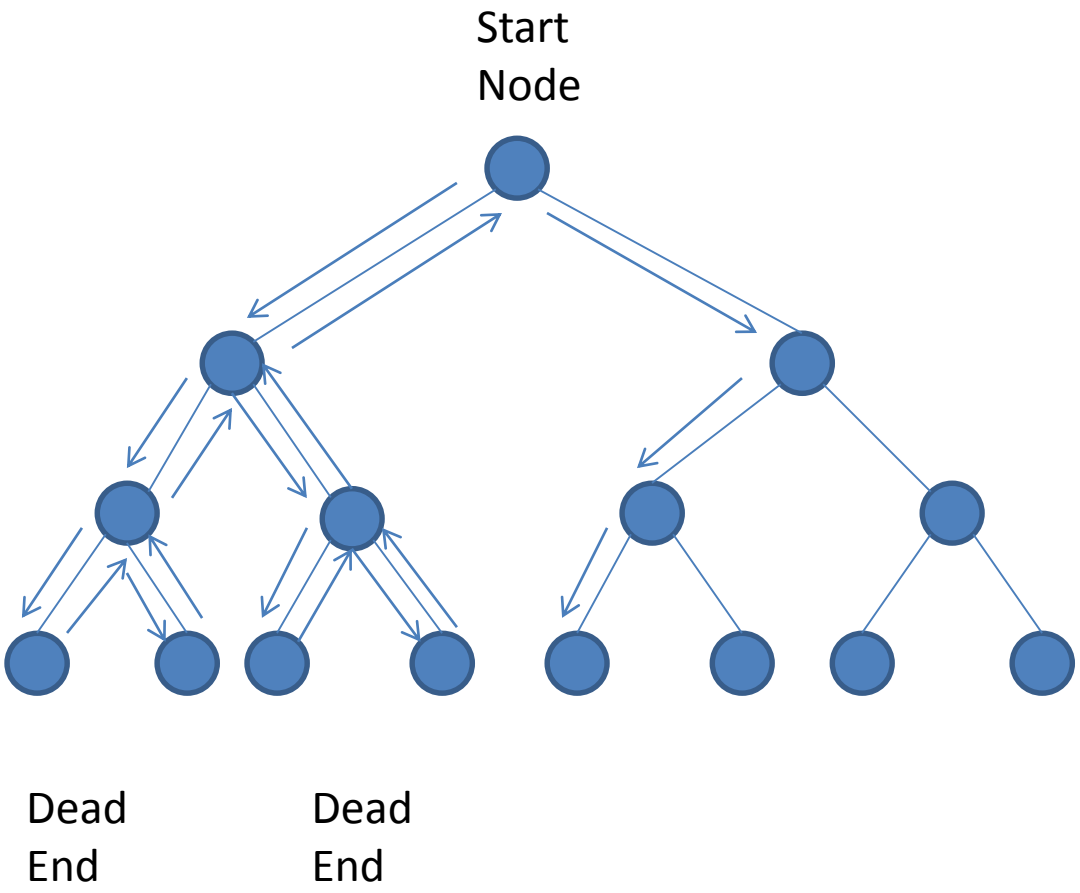
Recap: Backtracking Search

"Backtracking is a technique for systematically trying all paths through a state space"

It begins at the start state and pursues a path until:

- Finding a goal, then quit and return the solution path.
- Finding a dead end, then backtrack to the most recent unexamined node and continue down one of its branches.

Recap: Backtracking Search



Recap: Backtracking Algorithm Data Structure

State List (SL): lists the states in the current path being tried. If the goal is found, then it contains the solution path.

New State List (NSL): contains nodes a waiting evaluation.

Dead Ends (DE): lists states whose descendants have failed to contain a goal node.

Current State (CS): a state currently under consideration.

Recap: Backtracking Algorithm

```
Function backtrack;
Begin
        SL:=[Start]; NSL:=[Start]; DE:= Start;
        while NSL# [ ] do
        begin
                if CS=goal(or meets goal description) then return(SL);
                if CS has no children(excluding nodes already on DE, SL, and NSL)
                        then begin
                                while SL is not empty and CS = the first element of SL do
                                        begin
                                          add CS to DE;
                                          remove first element from SL;
                                          remove first element from NSL;
                                           CS:= first element of NLS;
                                        end;
                                        add CS to SL;
                                end
                        else begin
                                place children of CS on NSL; (except nodes on DE, SL, or NSL)
                                CS:= first element of NSL;
                                add CS to SL;
                        end;
        end;
        return FAIL;
end;
```

Blind vs. Heuristic Strategies

Blind (or un-informed) strategies do not exploit any of the information contained in a state.

Heuristic (or informed) strategies exploits such information to assess that one node is “more promising” than another.

Blind Strategies

Breadth-first

Bidirectional

Depth-first

Depth-limited

Iterative deepening

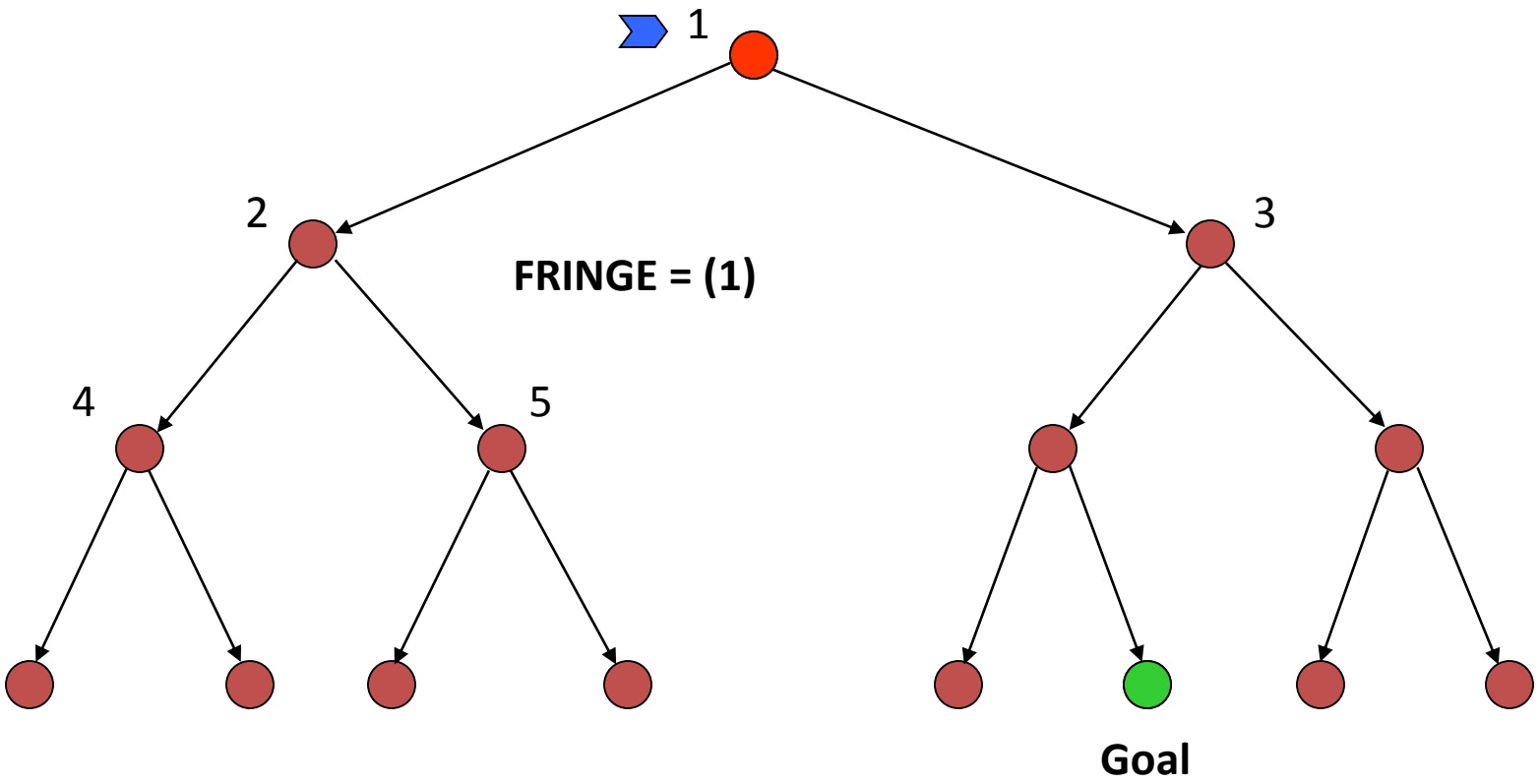
Step cost = 1

Uniform-Cost

Step cost = c(action) $\geq \varepsilon > 0$

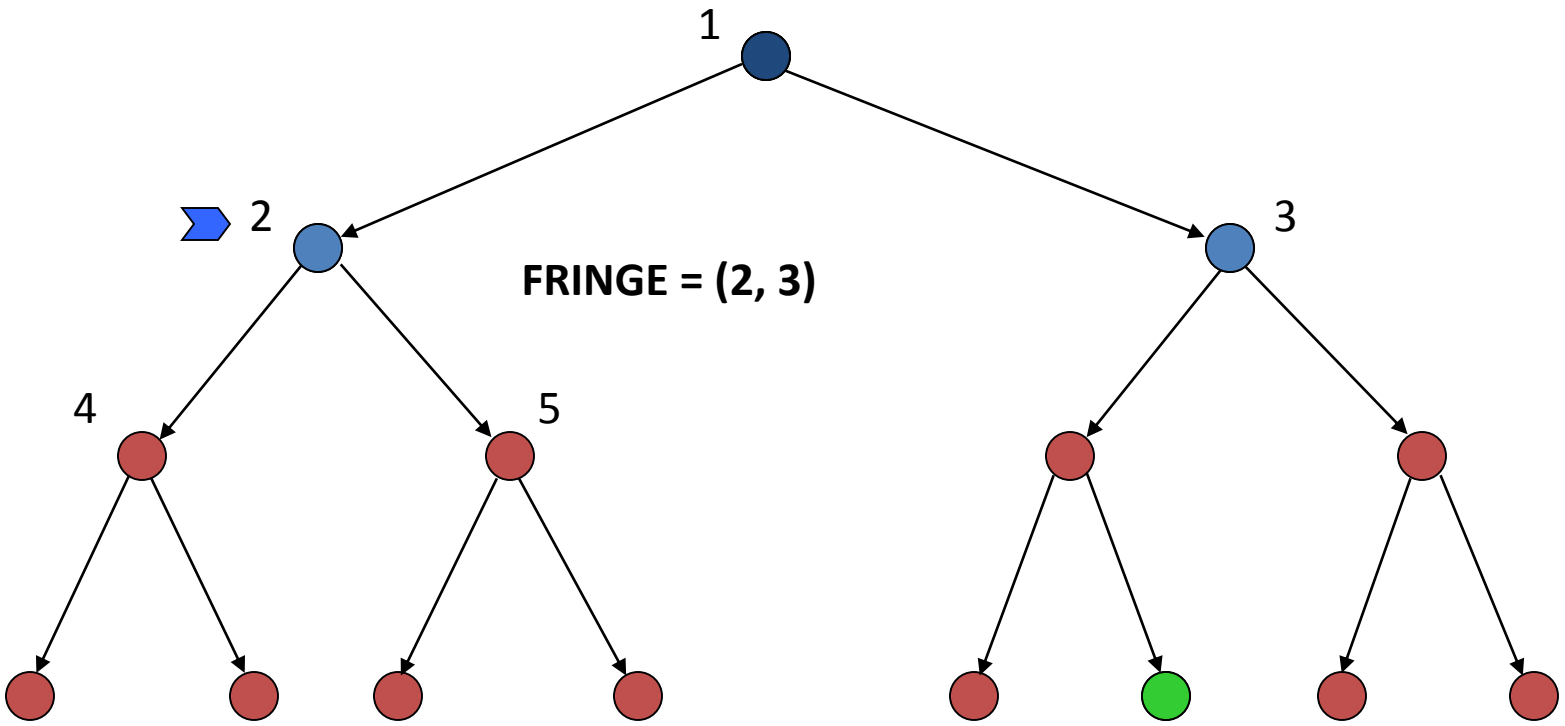
Depth-First Strategy

New nodes are inserted at the front of FRINGE



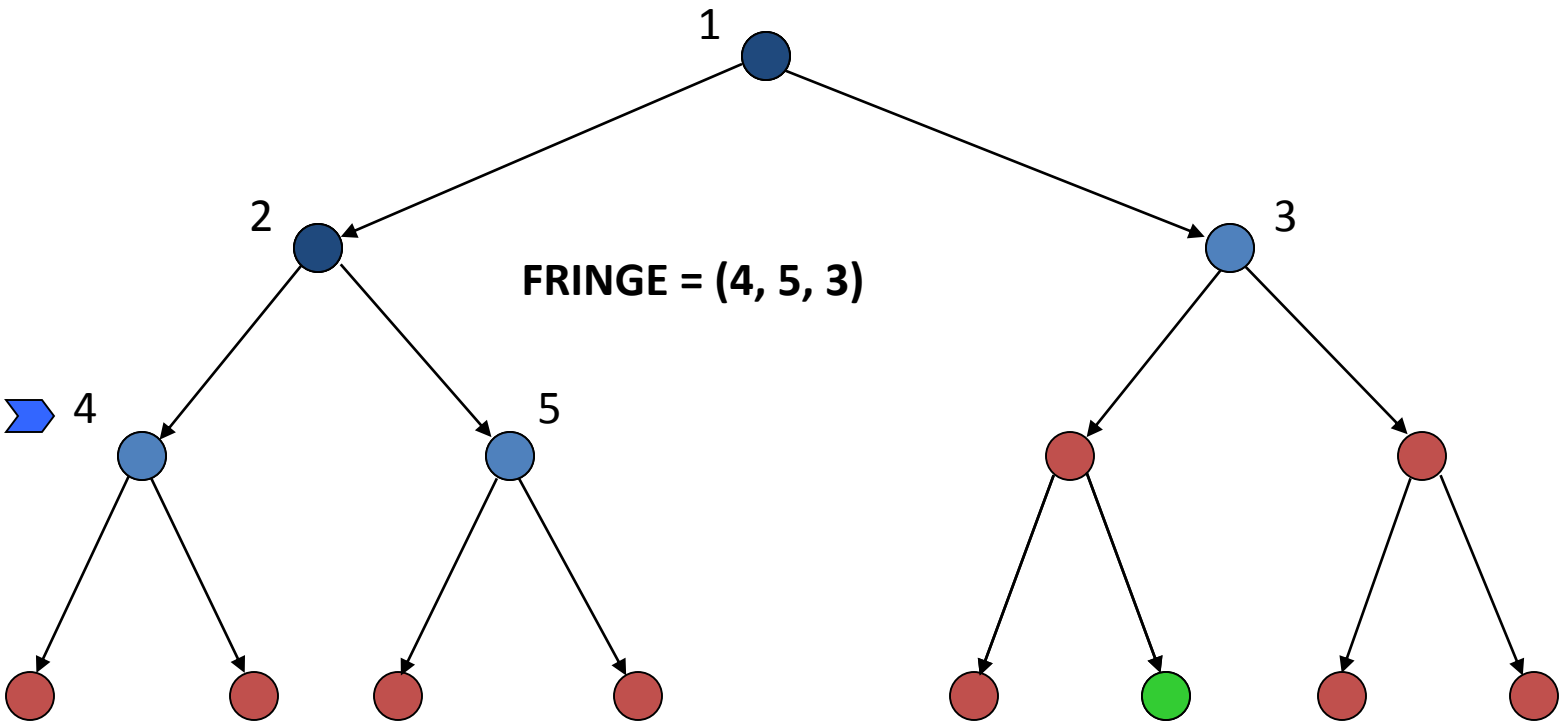
Depth-First Strategy

New nodes are inserted at the front of FRINGE



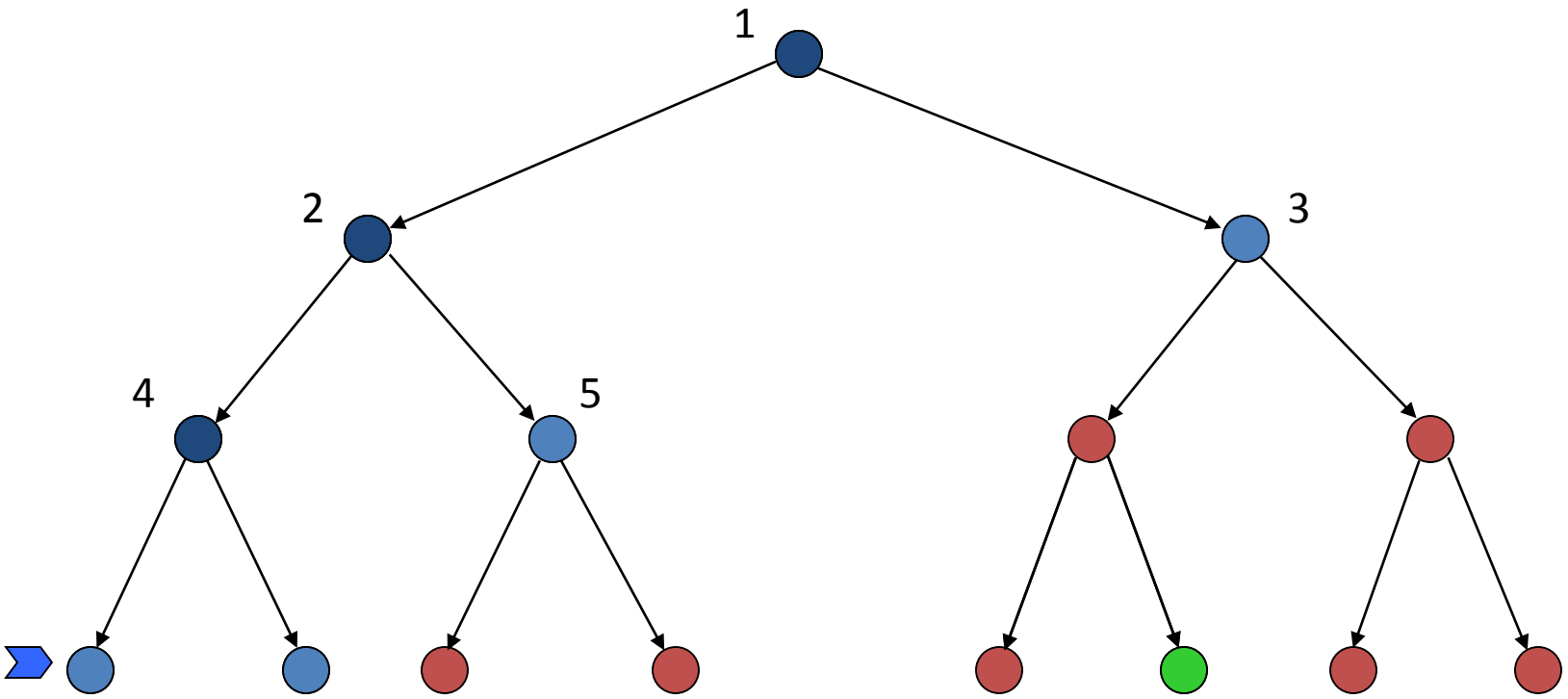
Depth-First Strategy

New nodes are inserted at the front of FRINGE



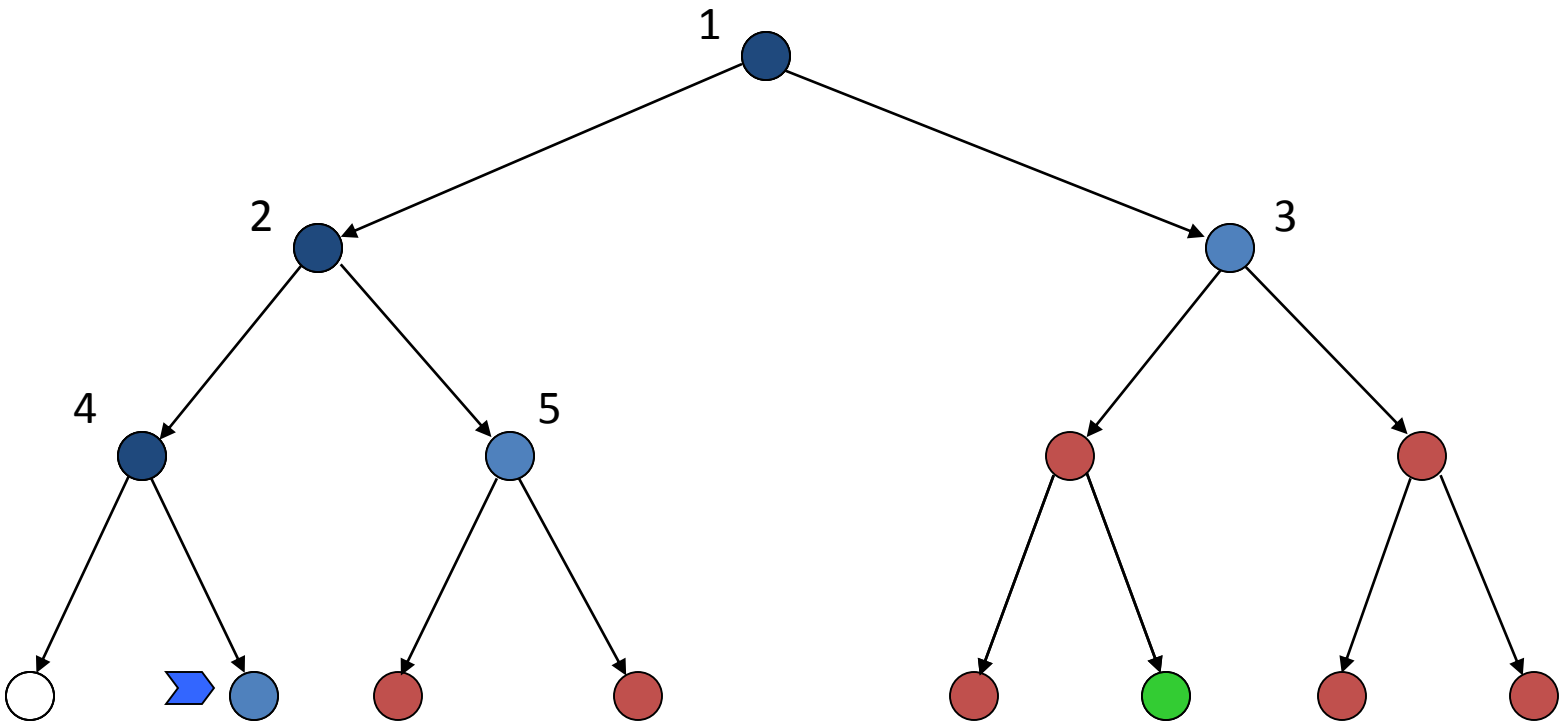
Depth-First Strategy

New nodes are inserted at the front of FRINGE



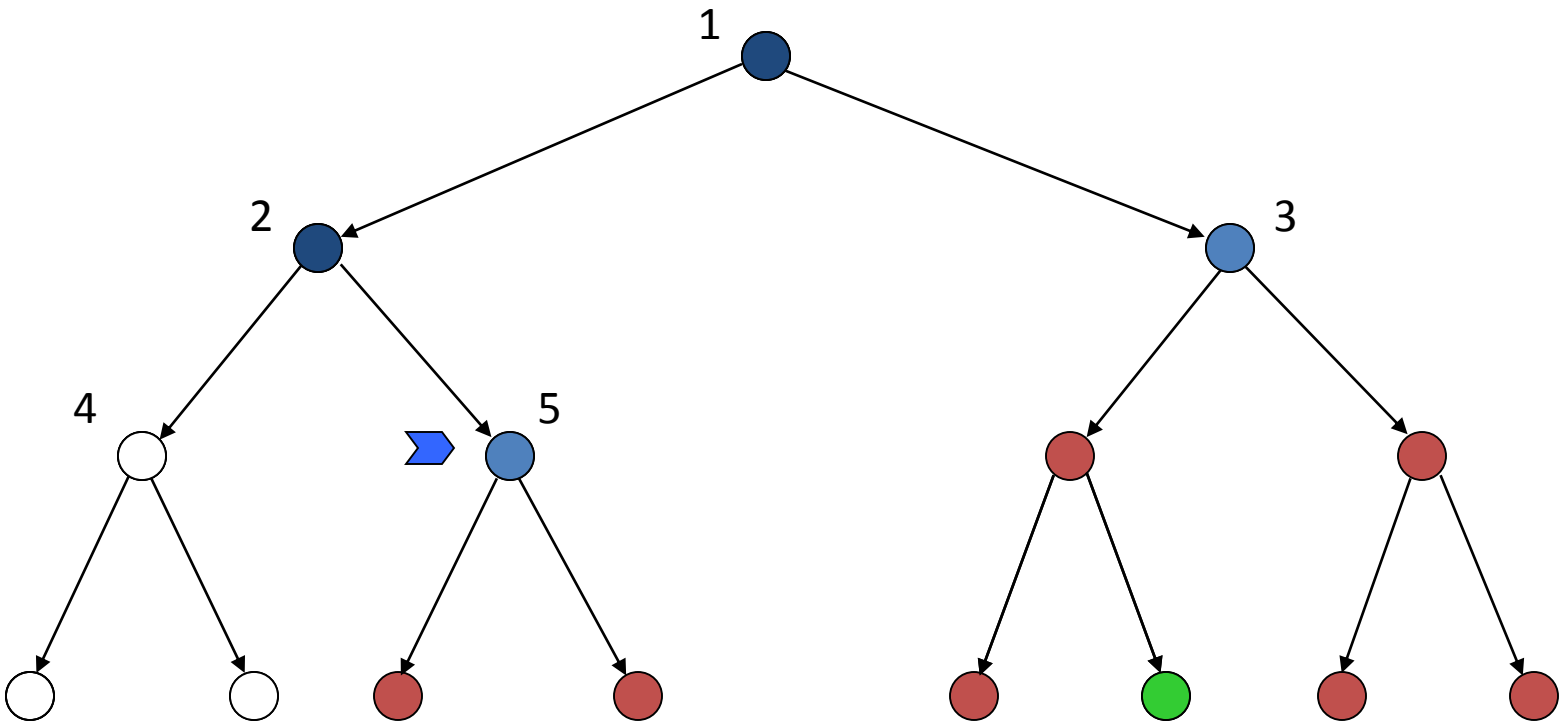
Depth-First Strategy

New nodes are inserted at the front of FRINGE



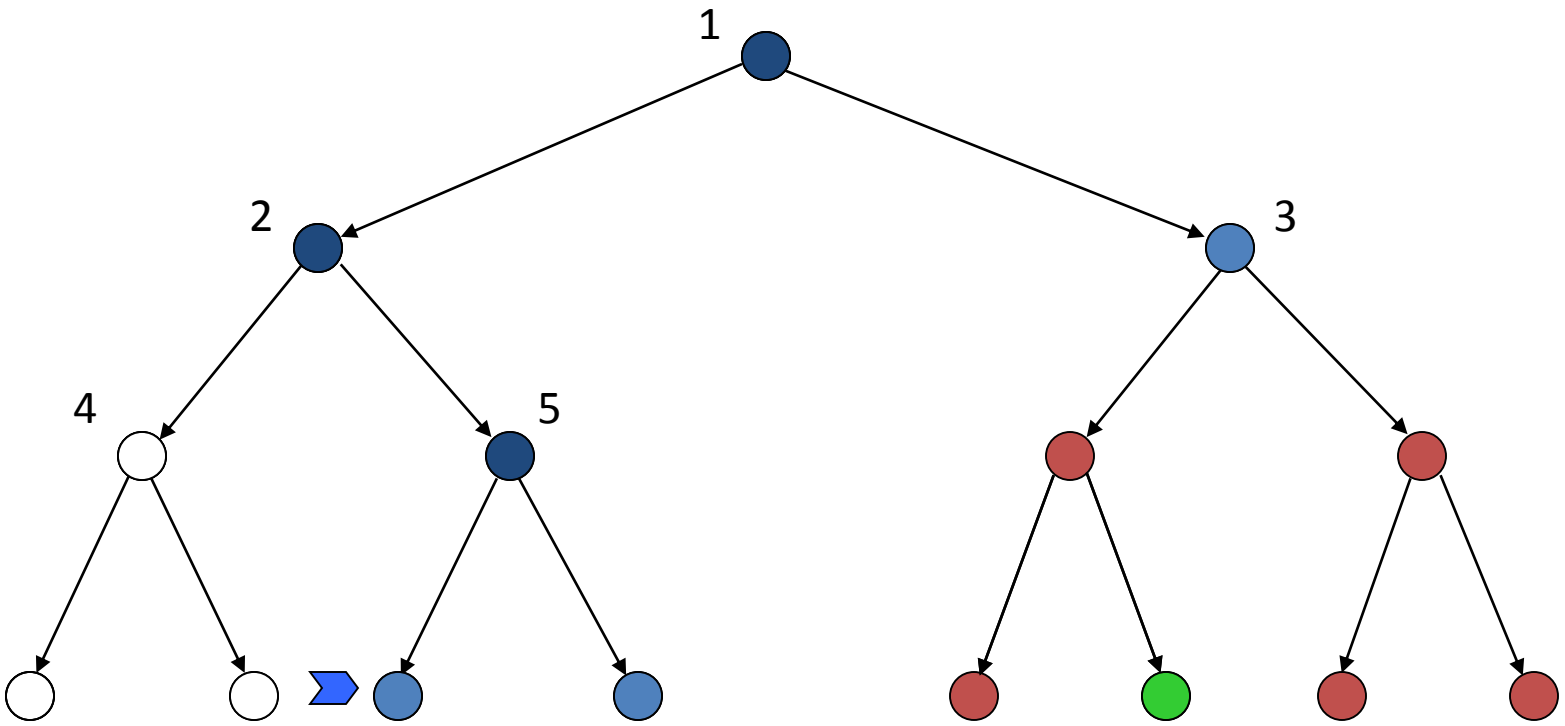
Depth-First Strategy

New nodes are inserted at the front of FRINGE



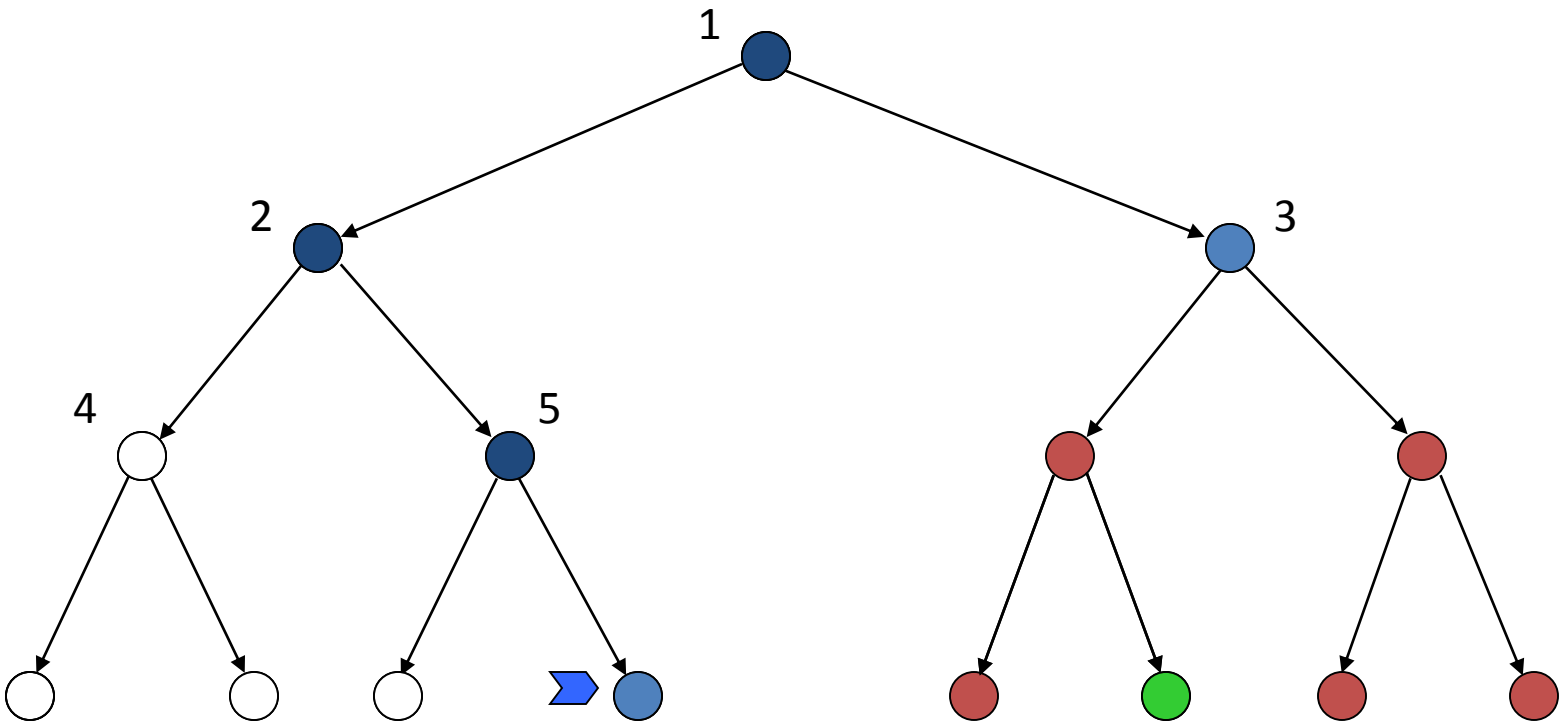
Depth-First Strategy

New nodes are inserted at the front of FRINGE



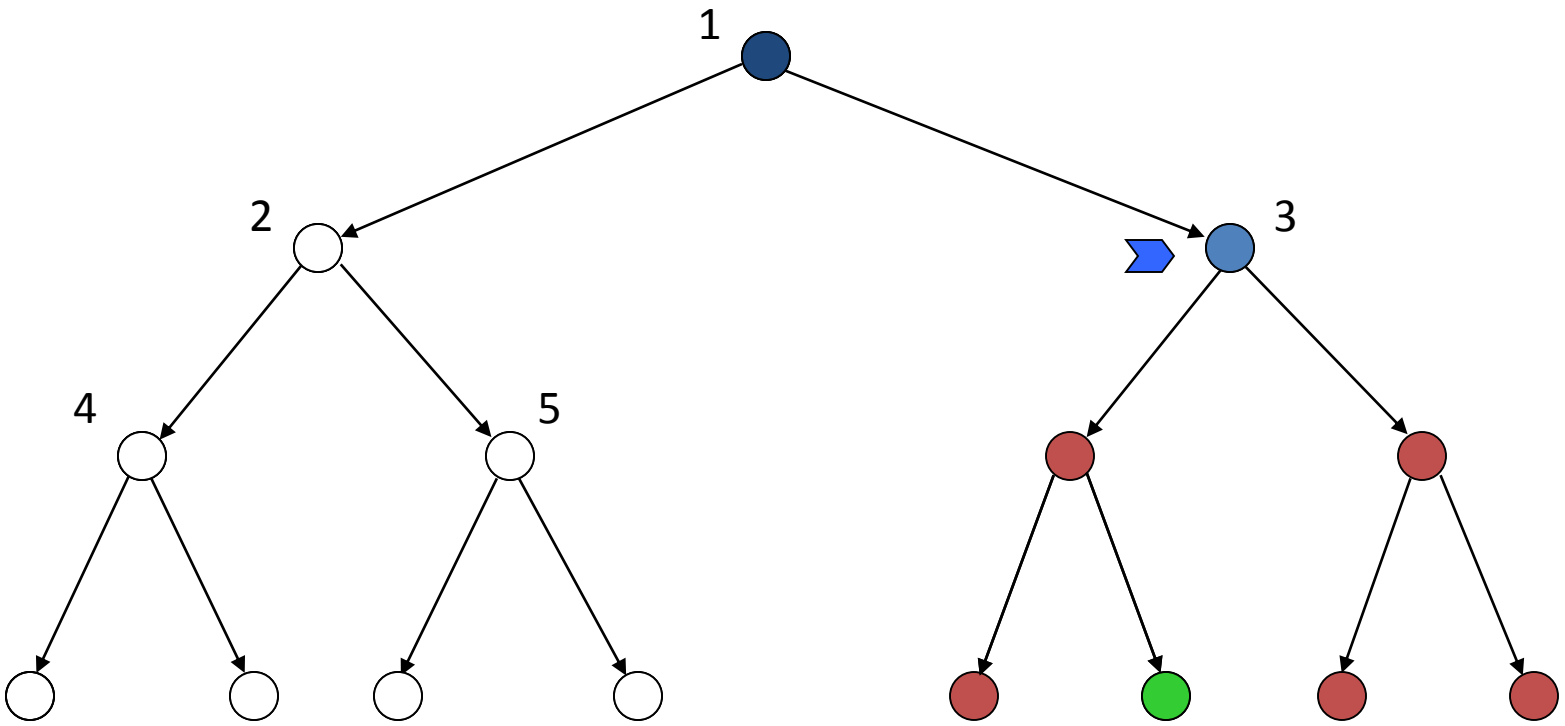
Depth-First Strategy

New nodes are inserted at the front of FRINGE



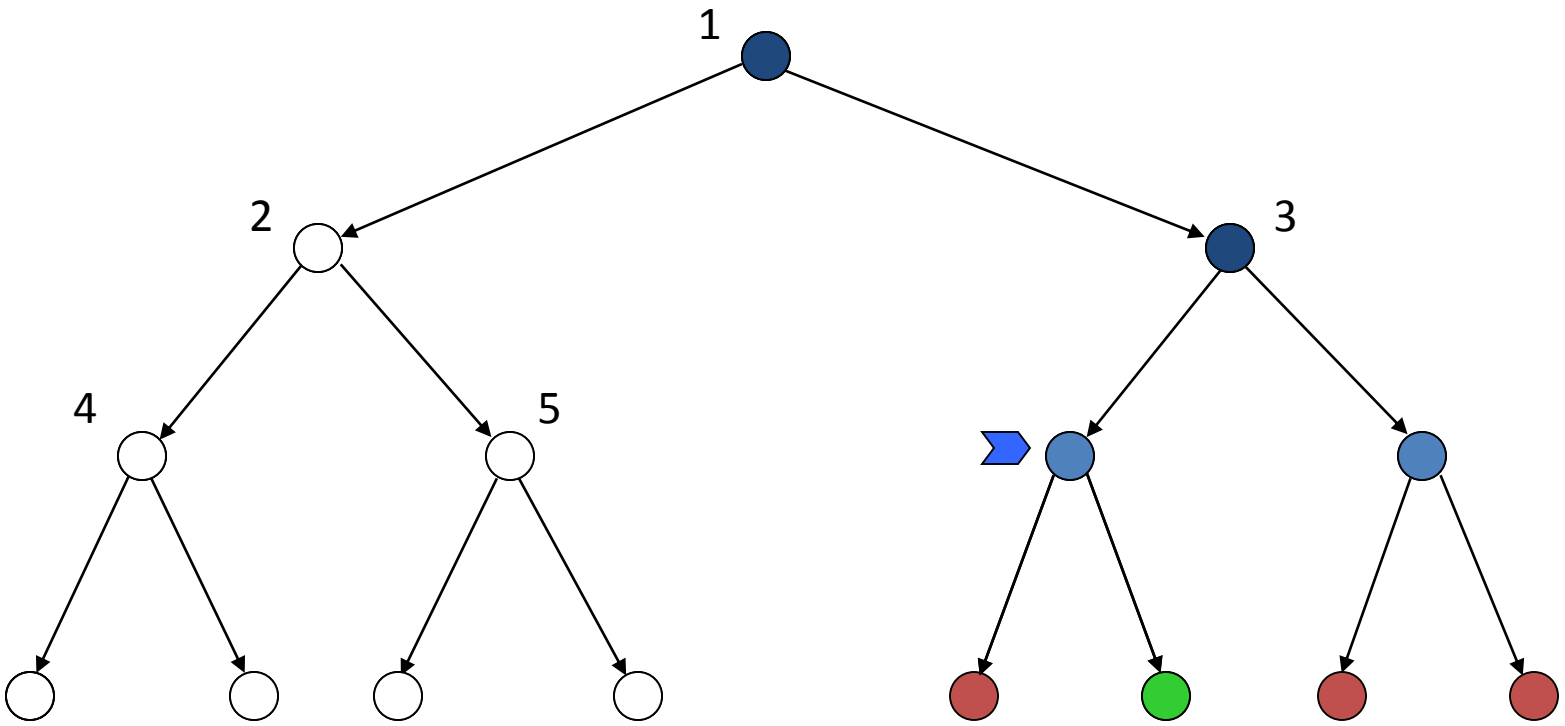
Depth-First Strategy

New nodes are inserted at the front of FRINGE



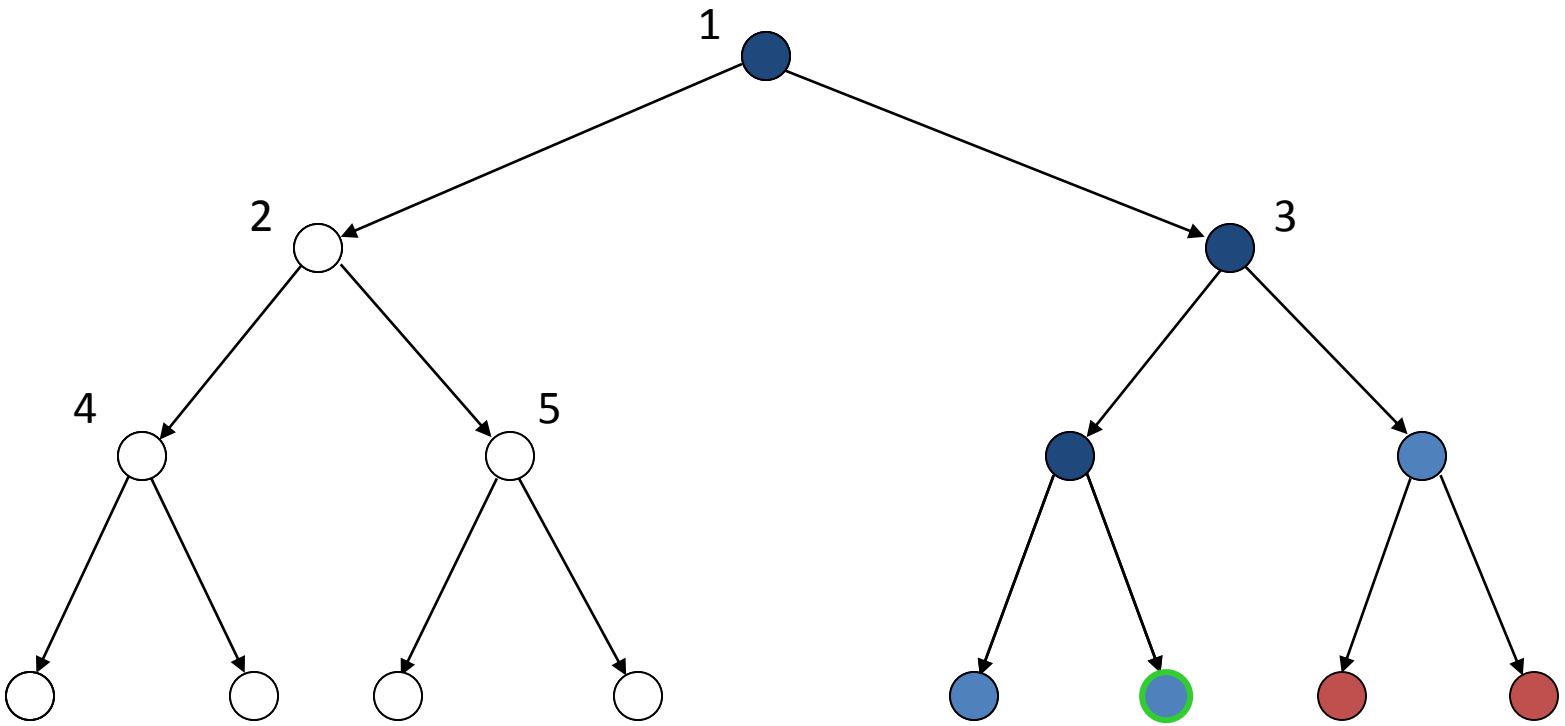
Depth-First Strategy

New nodes are inserted at the front of FRINGE



Depth-First Strategy

New nodes are inserted at the front of FRINGE



Depth-Limited Strategy

Depth-first with depth cutoff k (maximal depth below which nodes are not expanded)

Three possible outcomes:

- Solution
- Failure (no solution)
- Cutoff (no solution within cutoff)

Iterative Deepening Strategy

Repeat for k = 0, 1, 2, …:

Perform depth-first with depth cutoff k

Repeated States

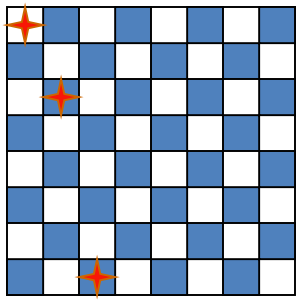
No

Few

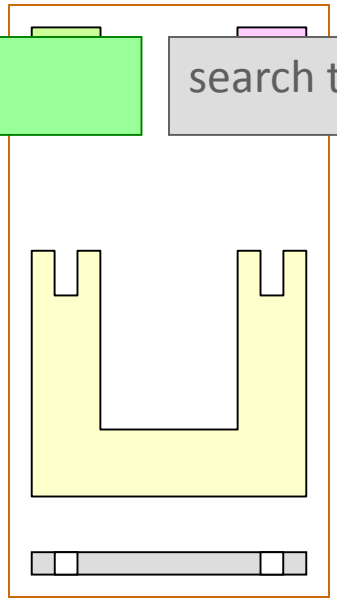
Many

search tree is finite

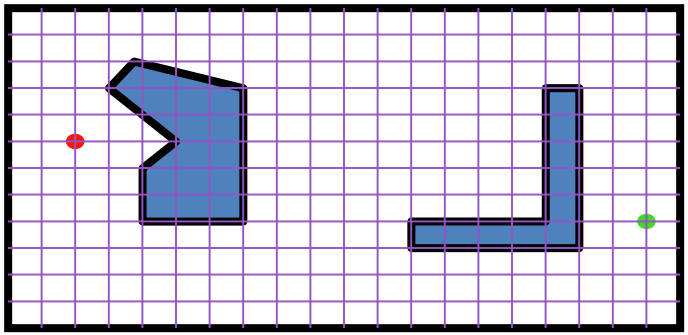
search tree is infinite



8-queens



assembly planning



8-puzzle and robot navigation

Avoiding Repeated States

Requires comparing state descriptions.

For example, Breadth-first strategy:

- Keep track of all generated states.
- If the state of a new node already exists, then discard the node.

Avoiding Repeated States

Depth-first strategy:

Solution 1:

- Keep track of all states associated with nodes in current path
- If the state of a new node already exists, then discard the node

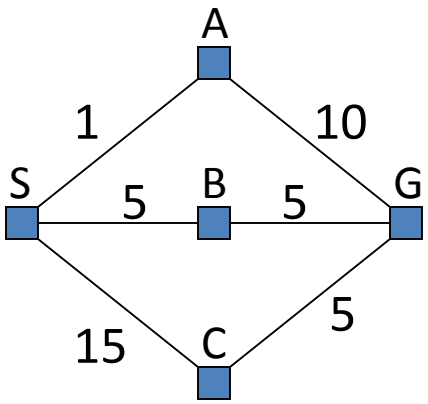
Solution 2:

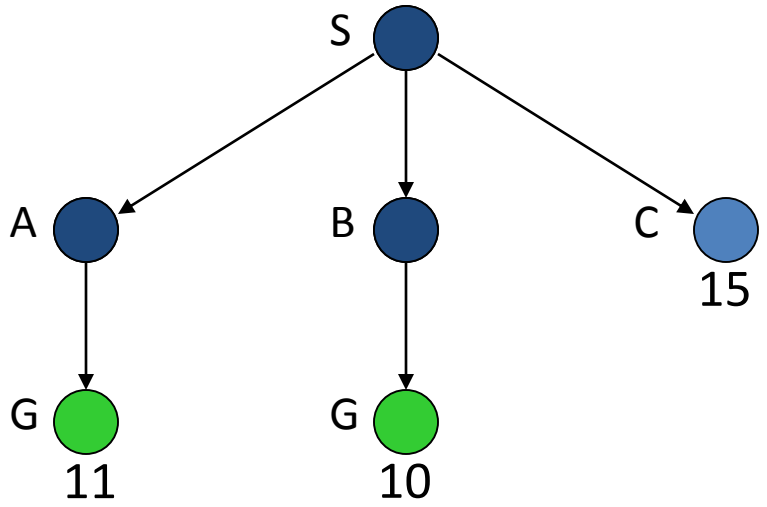
- Keep track of all states generated so far
- If the state of a new node has already been generated, then discard the node

Uniform-Cost Strategy

- Each step has some cost $\geq \varepsilon > 0$.
- The cost of the path to each fringe node N is

  $g(N) = \sum$ costs of all steps.
- The goal is to generate a solution path of minimal cost.
- The queue FRINGE is sorted in increasing cost.





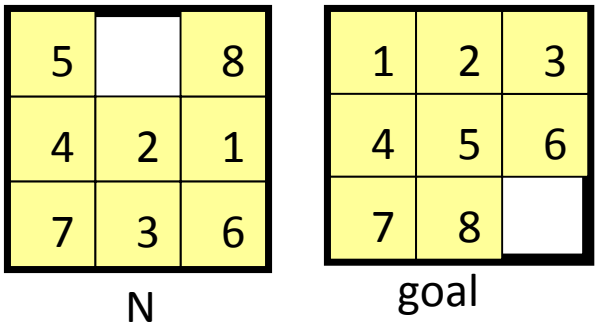
Best-First Search

- Define a function:
  f : node N → real number f(N)
  called the evaluation function, whose value depends on the contents of the state associated with N
- Order the nodes in the fringe in increasing values of f(N)
- f(N) can be any function you want, but will it work?

Heuristic Function

Function h(N) that estimate the cost of the cheapest path from node N to goal node.

Example: 8-puzzle

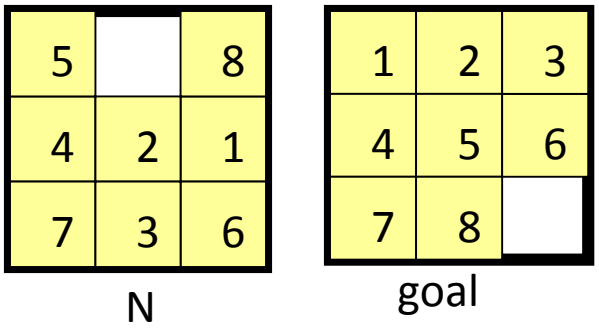


h(N) = number of misplaced tiles
= 6

Heuristic Function

Function h(N) that estimate the cost of the cheapest path from node N to goal node.

Example: 8-puzzle

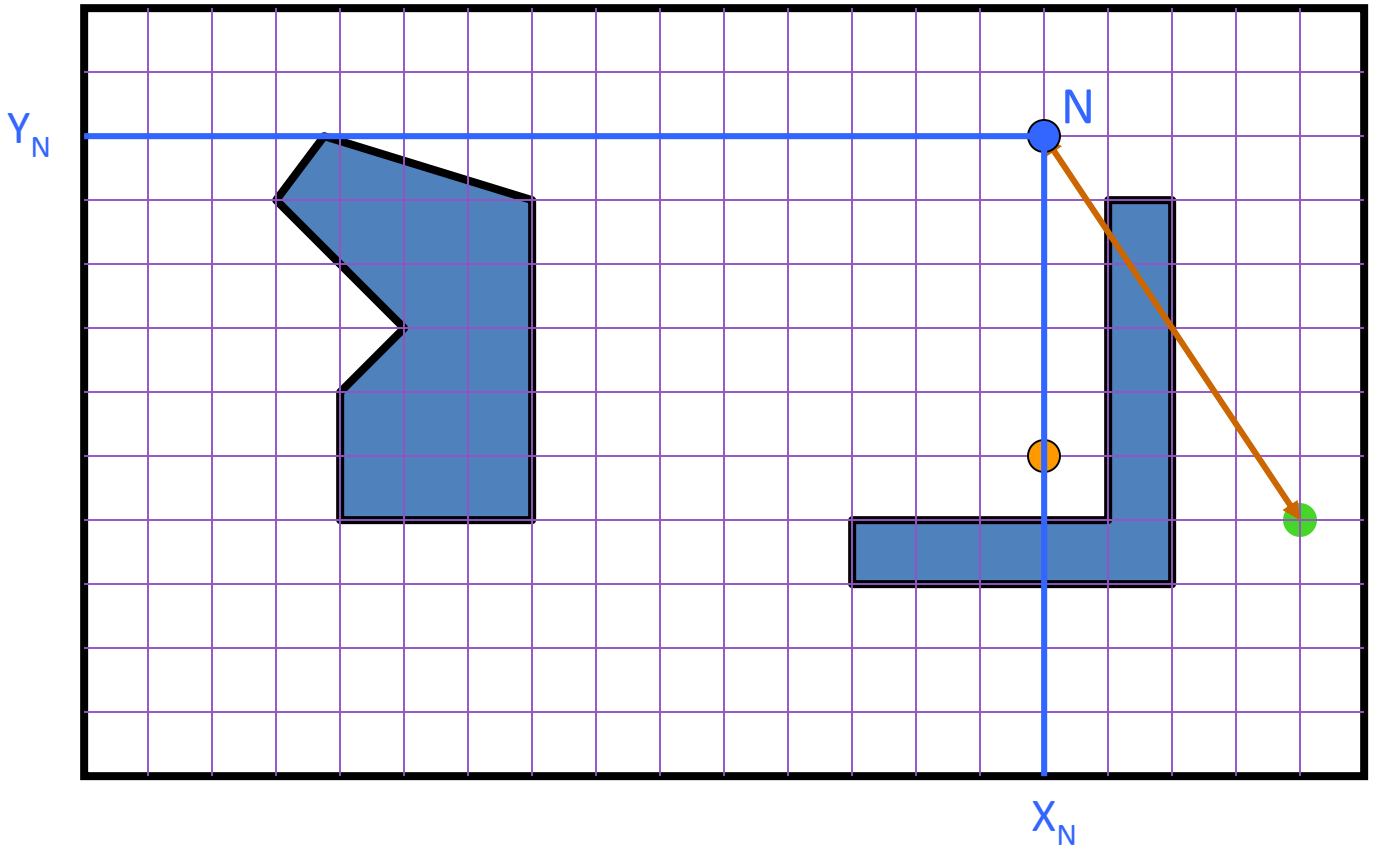


h(N) = sum of the distances of every tile to its goal position

= 2 + 3 + 0 + 1 + 3 + 0 + 3 + 1

= 13

Robot Navigation



h(N) = Straight-line distance to the goal

$$= [(X_g - X_N)^2 + (Y_g - Y_N)^2]^{1/2}$$

Examples of Evaluation function

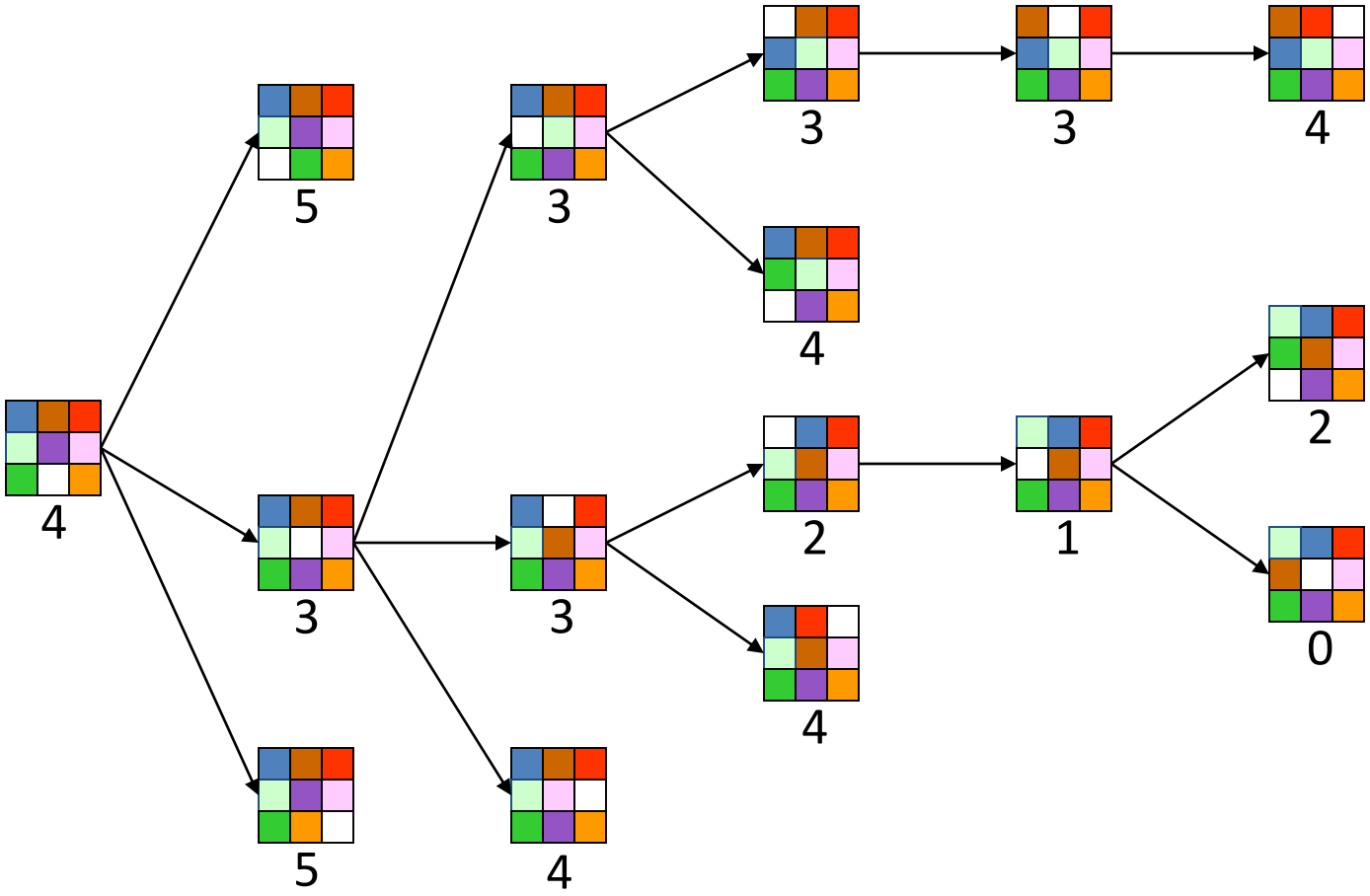
Let g(N) be the cost of the best path found so far between the initial node and N

f(N) = h(N) → greedy best-first

f(N) = g(N) + h(N)

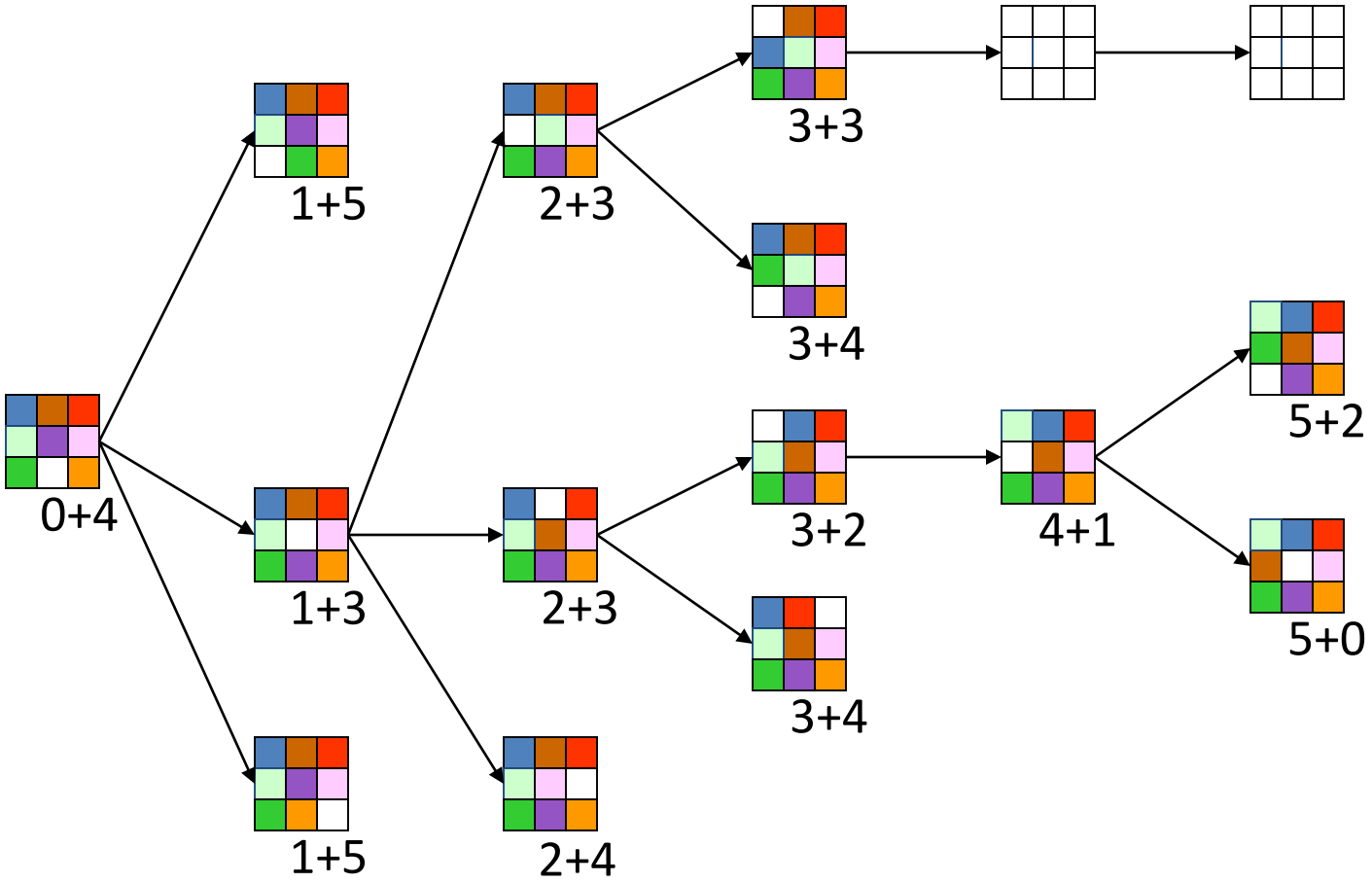
8-Puzzle

f(N) = h(N) = number of misplaced tiles



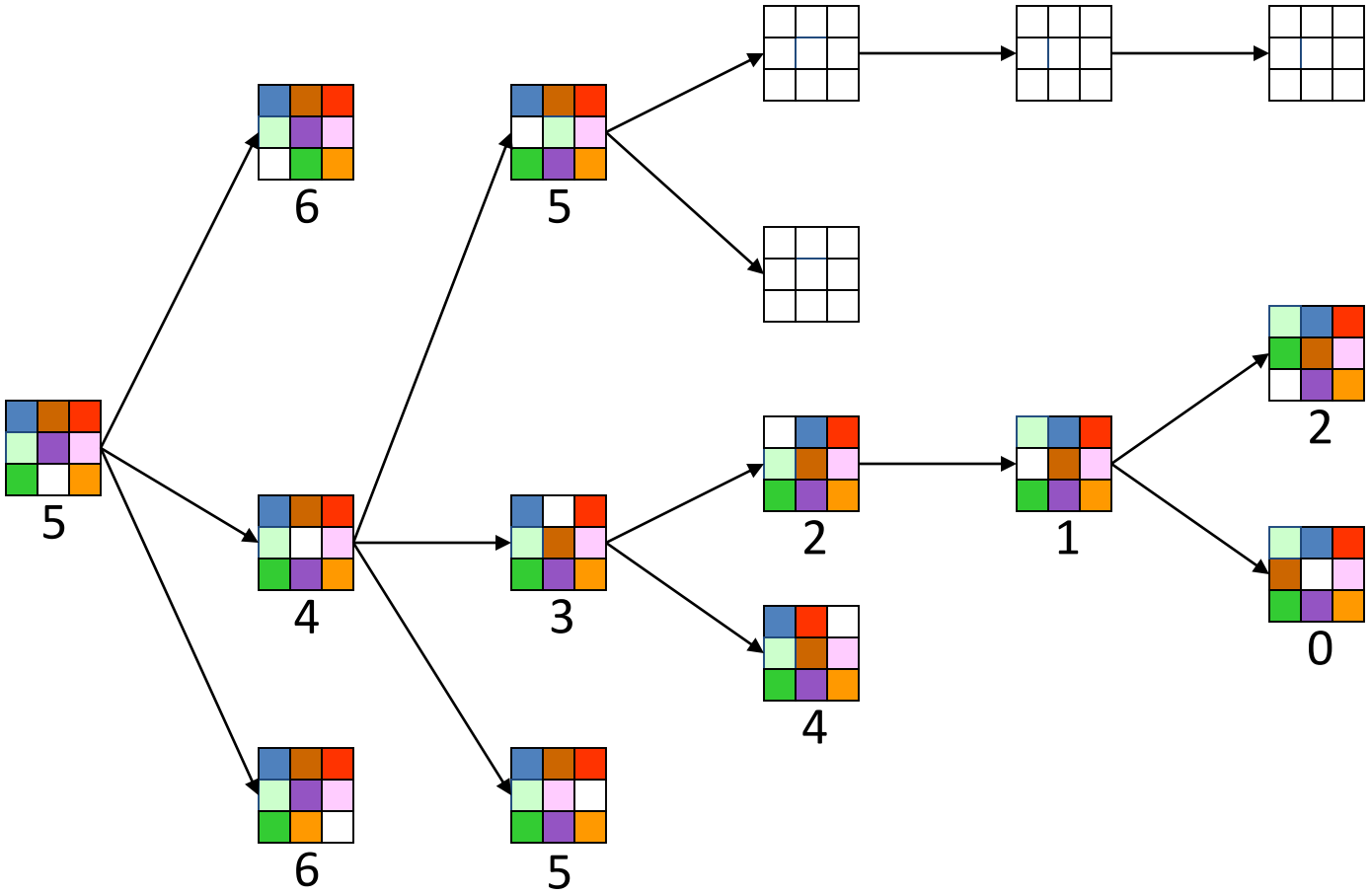
8-Puzzle

f(N) = g(N) + h(N)
with h(N) = number of misplaced tiles



8-Puzzle

$f(N) = h(N) = \sum$ distances of tiles to goal



Reasoning Representation

- Reasoning with propositional and predicate calculus can be represented as state space graphs.
- Propositions and predicates are represented as nodes in the graph and inference rules create the arcs between that nodes.
- Search is used to determine that a particular expression is a logical consequence of a given set of assertions.
- Different search strategies can be applied to this representation and can searching the space of inferences in either a depth-first or breadth-first search.

Propositional Calculus: Example (1)

$q \Longrightarrow p$

$r \Longrightarrow p$

$v \Longrightarrow q$

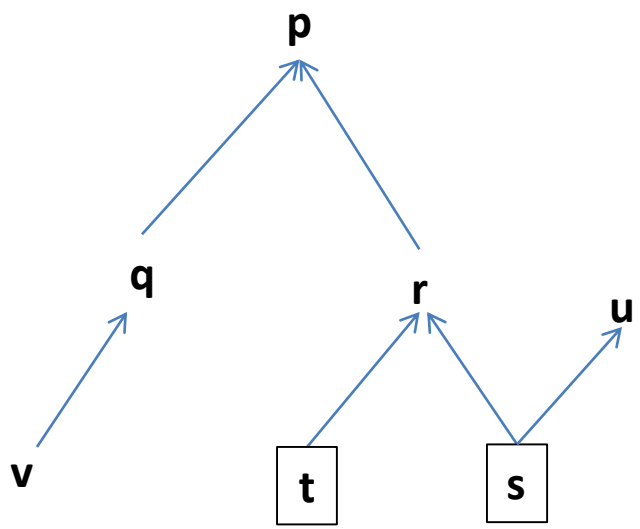
$s \Longrightarrow r$

$t \Longrightarrow r$

$s \Longrightarrow u$

s

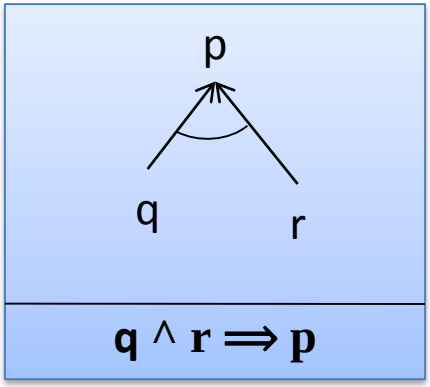
t

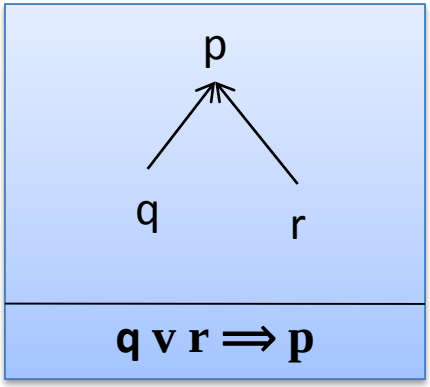


And/Or Graph

It is an extension to the basic graph model to represent the logical operator **and** and **or.**

- If the premises of an implication are connected by an ^ operator, they are called **and** nodes in the graph and the arcs from this nodes are joined by a curved link.
- If the premises are connected by an v operator, they are regarded as **or** nodes in the graph and the arcs from this nodes to their parent are not so connected.





Propositional Calculus: Example (2)

a

b

c

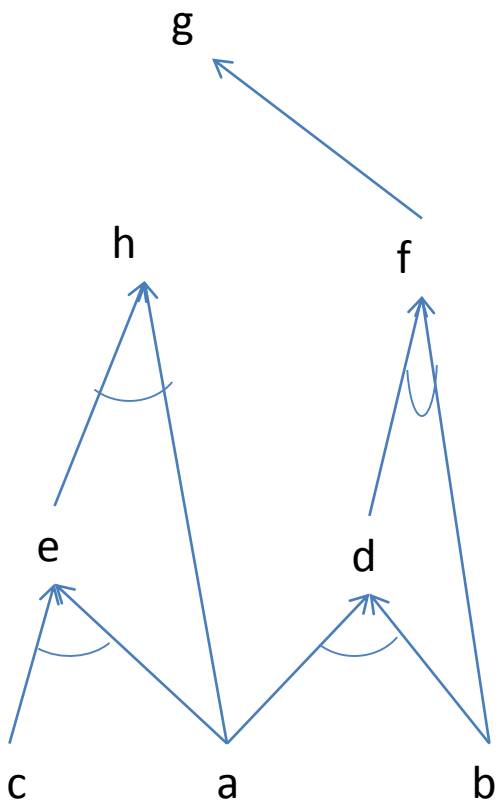
$a \wedge b \Longrightarrow d$

$a \wedge c \Longrightarrow e$

$b \wedge d \Longrightarrow f$

$f \Longrightarrow g$

$a \wedge e \Longrightarrow h$

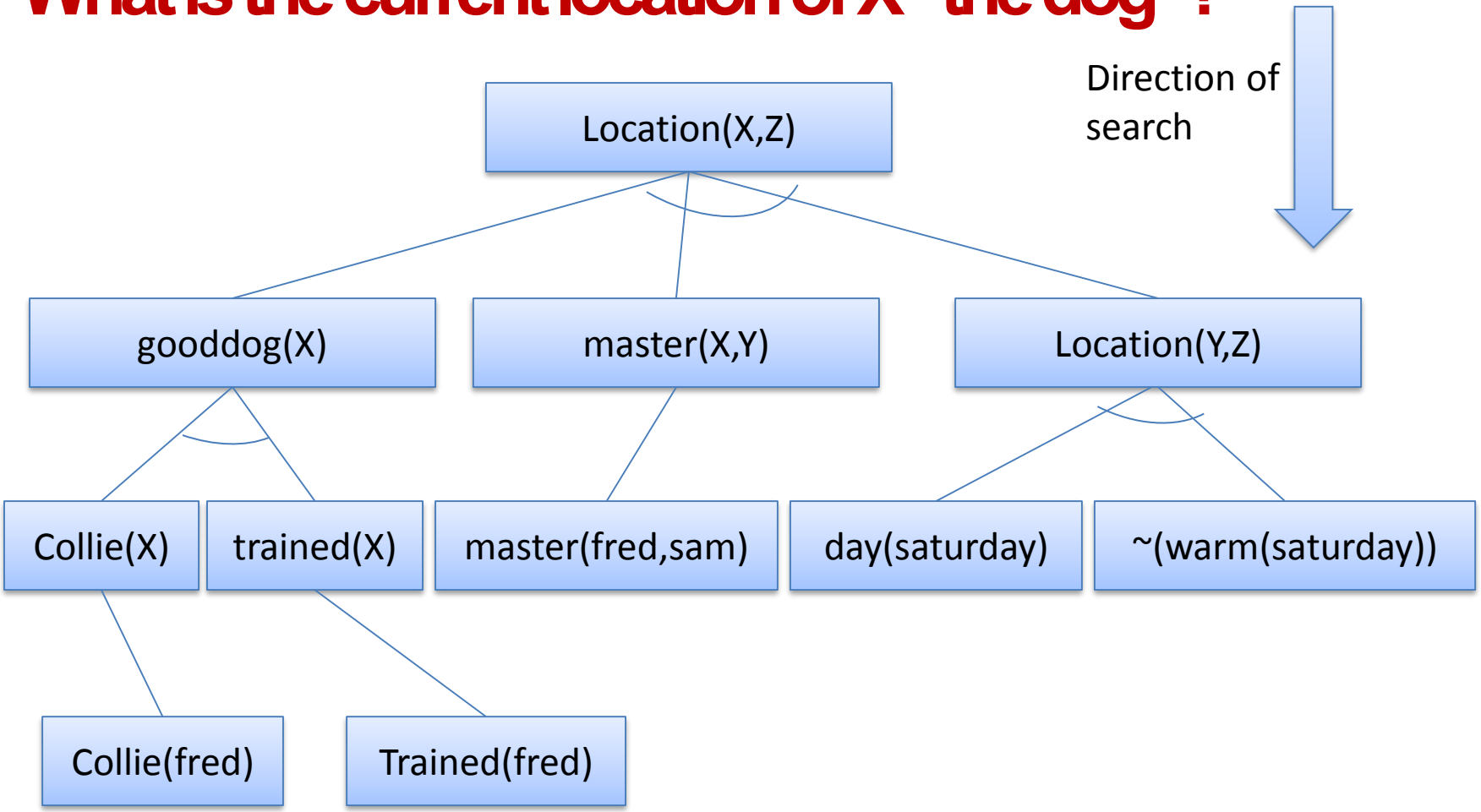


Predicate Calculus Example

1. collie(fred). Fred is a collie.
2. master(fred,sam). Sam is a fred's master.
3. day(saturday). This day is a Saturday.
4. ~(warm(saturday)). This Saturday is not warm.
5. trained(fred). Fred is trained.
6. ∀ X[spaniel(X)v(collie(X)^ trained(X)) ⟹ gooddog(X)].
   Spaniels are good dogs and so are trained collies
7. ∀(X,Y,Z)(gooddog(X) ^ master(X,Y) ^ location(Y,Z) ⟹ location(X,Z)).
   If a dog is a good dog and has a master then he will be with his master.
8. (day(saturday) ^ warm(staurday)) ⟹ location(sam,park).
   If it is Saturday and warm, then sam is at the park
9. day(saturday) ^ ~warm(saturday) ⟹ location(sam,museum).
   If it is Saturday & not warm, then sam is at the museum

Predicate Calculus Example

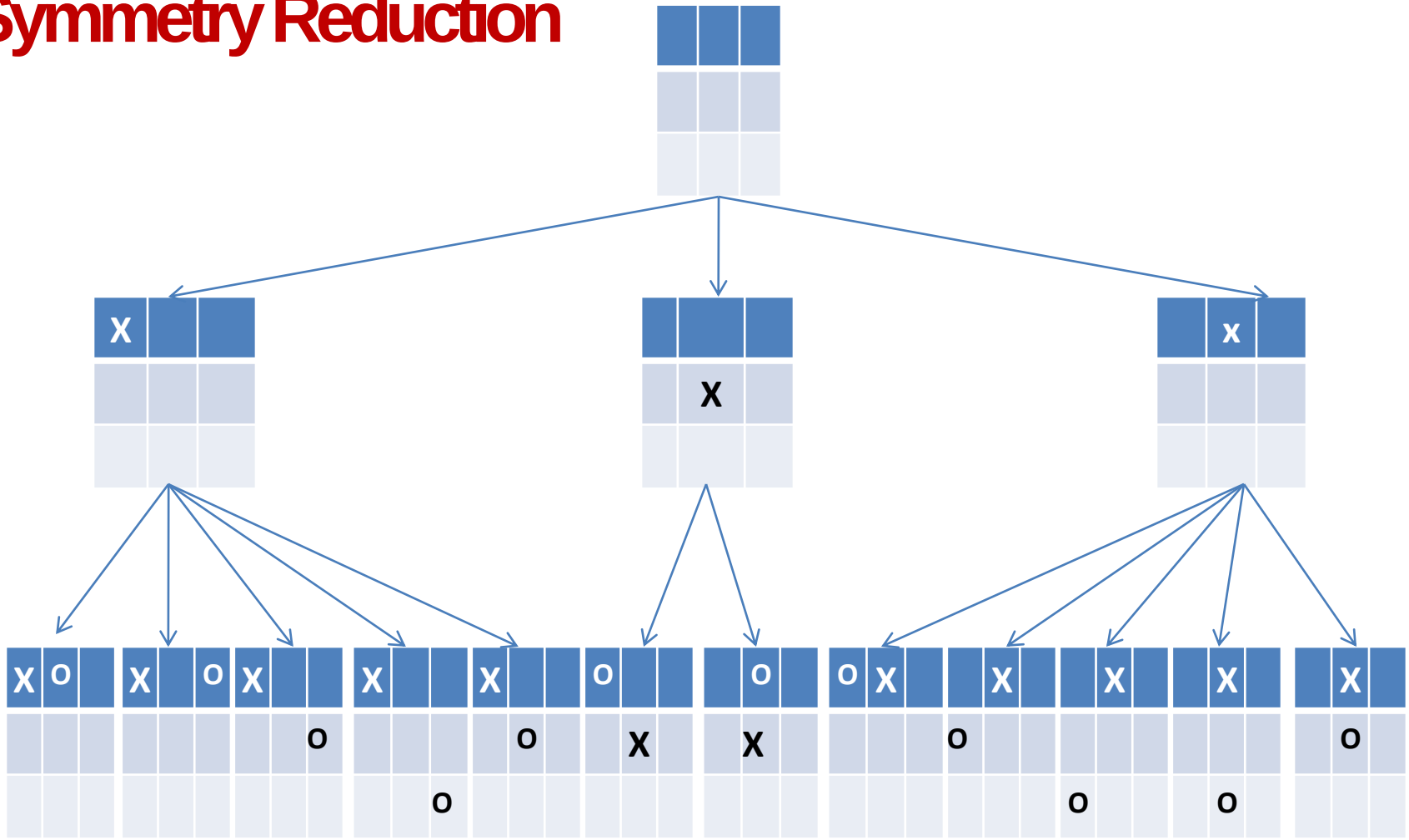
What is the current location of X "the dog"?



Substitution = {fred/X, sam/Y, museum/Z}

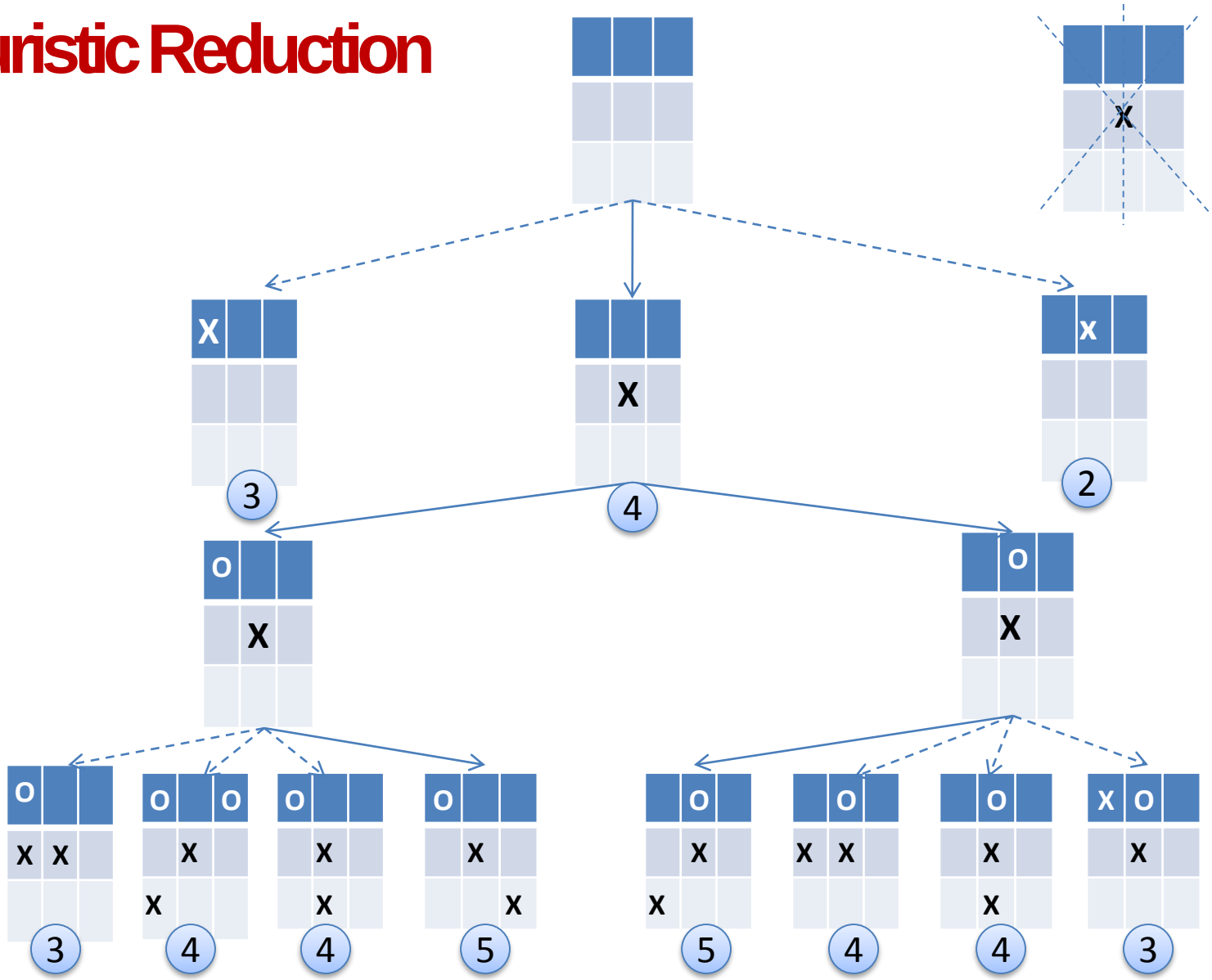
More on Heuristic Search & Functions

Symmetry Reduction



More on Heuristic Search & Functions

Heuristic Reduction



More on Heuristic Search & Functions

Hill Climbing Strategy

- "Method of blind mountain climbers to go uphill along the steepest possible path until no farther to go"
- Expand to the current state and evaluate its children. The best child is selected for further expansion until reaching a state that is better than any of its children.
- It fails to find a solution in case of the reaching state is not a goal but just a local maximum state.
- The strategy can be used effectively if the evaluation function if sufficiently informative to distinguish between local and global maximum.

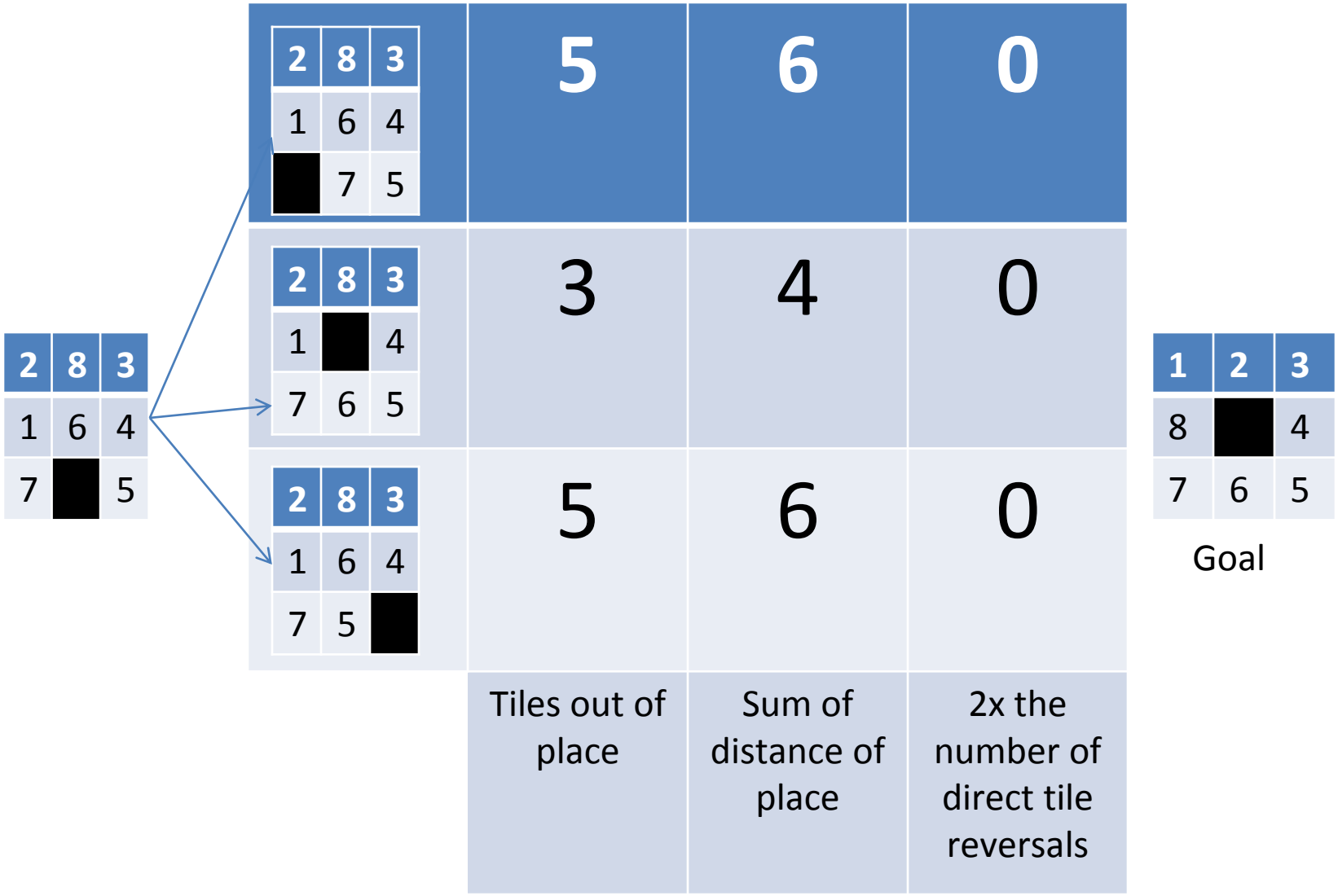
More on Heuristic Search & Functions

Best-First Search

- Use open list to keep track of the current edge of the search and closed list to record states already visited.
- Order the states on open list according to some heuristic estimate of their closeness to a goal and the most promising state is considered in each iteration.
- The best state to be examined in each iteration may be from any level of the state space.
- Open sorted list is represented as a **priority queue**.

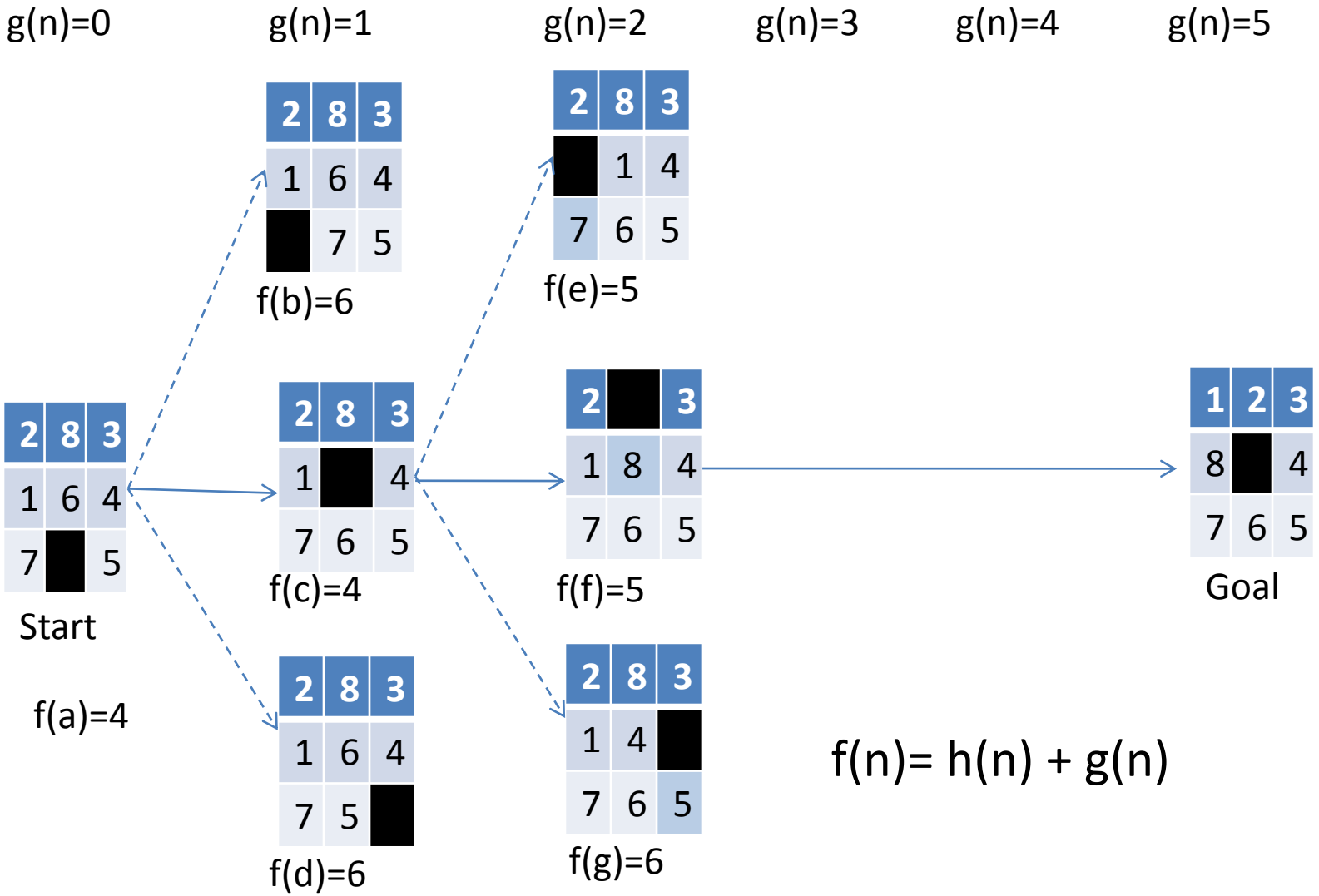
More on Heuristic Search & Functions

8-Puzzle Heuristics



More on Heuristic Search & Functions

8-Puzzle Heuristics



Thanks! … *Questions?*

AI: History and Applications
Sheet # 1

No. Of Questions: 3 **No. Of Pages: 1**

<u>Question 1:</u> Describe briefly:

a) The Turing Test "imitation game". *(Illustrate through drawing)*
b) The Total Turing Test.
c) Systems that "Act/Behave Rationally", and Systems that "Act/Behave Humanly".
d) "Weak AI Hypothesis" versus the "Strong AI Hypothesis".
e) The two most fundamental concerns of AI researchers.
f) Intelligent Agents.

<u>Question 2:</u> Give the scientific term for each of the following statements:

a) The branch of computer science that is concerned with the automation of intelligent behavior.
b) A problem-solving technique that systematically explores a space of *problem states*.
c) Systems that are constructed by obtaining knowledge from a human expert and coding it into a form that a computer may apply to similar problems.
d) Models that parallel the structure of neurons in the human brain and used to build intelligent programs.
e) Algorithms that evolve new problem solutions from components of previous solutions using specific operators such as crossover and mutation.

<u>Question 3:</u> Criticize Turing's criteria for computer software being "intelligent"; What is Searle's thought experiment *(the Chinese Room Argument)*?

With our best wishes;
Dr. Hala Abdel-Galil, & Dr. Amr S. Ghoneim

Propositional Calculus
Sheet # 2

No. Of Questions: 6 **No. Of Pages: 2**

Question 1: An AI representation language must:

a. Handle qualitative knowledge.
b. Allow knowledge to be inferred from a set of facts and rules.
c. Allow representation of general principles as well as specific situations.
d. Capture complex semantic meaning.
e. Allow for meta-level reasoning.

Describe each of the stated requirements briefly.

Question 2: Fill in the first three columns of the table with all possible interpretations in the domain {*A, B, C*}:

A	B	C	S1

Now, consider the following sentence, S1: $(A \vee B) \rightarrow (\neg B \wedge (C \vee A))$
In the column labeled "S1", place a mark next to each interpretation in which S1 holds *(is true)*.

Question 3: Using truth tables, prove the – *following* – identities:

- $\neg(\neg P) \equiv P$
- $(P \vee Q) \equiv (\neg P \rightarrow Q)$
- The contrapositive law: $(P \rightarrow Q) \equiv (\neg Q \rightarrow \neg P)$
- De Morgan's law: $\neg (P \vee Q) \equiv (\neg P \wedge \neg Q)$ **and** $\neg (P \wedge Q) \equiv (\neg P \vee \neg Q)$
- The commutative laws: $(P \wedge Q) \equiv (Q \wedge P)$ **and** $(P \vee Q) \equiv (Q \vee P)$
- The associative law: $((P \wedge Q) \wedge R) \equiv (P \wedge (Q \wedge R))$
- The associative law: $((P \vee Q) \vee R) \equiv (P \vee (Q \vee R))$
- The distributive law: $P \vee (Q \wedge R) \equiv (P \vee Q) \wedge (P \vee R)$
- The distributive law: $P \wedge (Q \vee R) \equiv (P \wedge Q) \vee (P \wedge R)$

Question 4: The logical operator "↔" is read "if and only if." P↔Q is defined as being equivalent to (P→Q)∧(Q→P). Based on this definition, show that P↔Q is logically equivalent to (P∨Q)→(Q∧P):

a. By using truth tables.
b. By a series of substitutions using the identities on page 51.

Question 5: Prove that implication is transitive in the propositional calculus, that is,

$$((P \rightarrow Q) \wedge (Q \rightarrow R)) \rightarrow (P \rightarrow R).$$

Question 6: Here is a sentence in propositional logic:

$$(A \rightarrow (B \vee (C \wedge D))) \leftrightarrow (B \vee C)$$

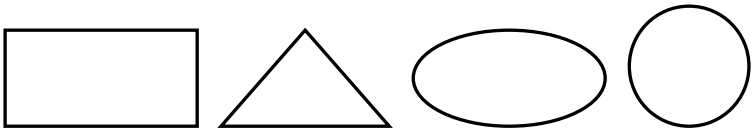
Does it hold given the interpretation $i = \{A = t; B = f, C = t, D = f\}$? If so, give an interpretation in which it does not hold. If not, give an interpretation in which it does hold.

With our best wishes;
Dr. Hala Abdel-Galil, & Dr. Amr S. Ghoneim

Predicate Calculus
Sheet # 3

No. Of Questions: 6 **No. Of Pages: 2**

<u>Question 1:</u> Determine whether each of the following sentences holds or fails given the interpretation below:

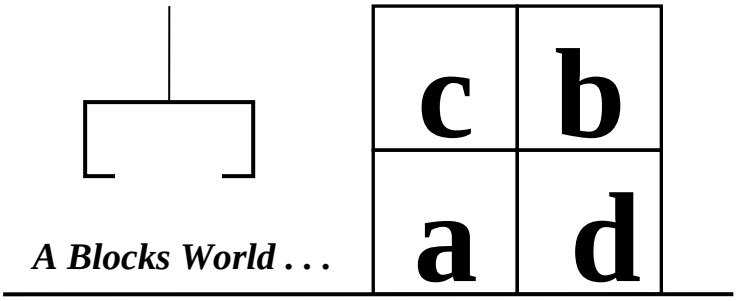


The Real World

- U = { □, △, ⬭, ○ }
- Constants: fred
- Predicates: above[2], circle[1], oval[1], square[1]
- Function: hat[1]
- I(fred) = △
- I(above) = {< □, △ >, < ○, ⬭ >}
- I(circle) = {< ○ >}
- I(oval) = {< ⬭ >, < ○ >}
- I(square) = {< △ >}
- I(hat) = {< □, △ >, < ⬭, ○ >, < □, □ >, < ○, ○ >}

a. ∀X, above(X, fred)
b. ∀X, above(X, Fred) → Square(X)
c. ∃X∀Y, circle(Y) → above(Y, X)

<u>Question 2:</u> Solve the following problem:



Considering the given "blocks world", write its predicate calculus describing *which blocks are on the table, which are on other blocks, which blocks are clear, the status of the robot arm and its operations*. In addition, introduce two general rules, the first rule is to describe which

block is clear, the second is to describe the operation of stacking one block on top of another.

Question 3: Represent the following English sentences in predicate calculus:

a) Some people like anchovies.
b) Hany is taller than any of other students.
c) Nobody likes taxes.
d) All software developers are intelligent.
e) If a ≤ b, and b ≤ c, then a ≤ c.
f) John or David will come.
g) Jumbo is an elephant, but quite intelligent.
h) There are problems that have no solutions.

Question 4: Give the scientific term for each of the following statements:

a) A quantifier that indicates that a sentence is true for all values of the variable.
b) A quantifier that indicates that a sentence is true for at least one value in the domain.

Question 5: Find the most general unifier for each of the following pairs of sentences:

S1	S2	Unifier
h(bob)	h(X)	
eq(f(bob), alice)	eq(X, Y)	
p(f(X), bob)	p(alice, X)	
eq(f(f(bob)), f(bob))	eq(f(X), X)	
p(Y, bob)	p(f(bob), X)	

Question 6: Write the following English statements in first-order logic using these predicates. You may assume that everything in the world has already been constrained to be a person:

Predicate	Meaning
h(X)	X is an heir.
m(X)	X is male.
p(Y, X)	Y is in the Trump family and is the parent of X.
eq(X, Y)	X and Y are equal.

a. A person is an heir if and only if he or she is male and has a parent from the Trump family who is an heir.
b. A person only has one parent from the Trump family.
c. If two people are equal and one of them is an heir, the other is an heir.
d. Equal is symmetric.

With our best wishes;
Dr. Hala Abdel-Galil, & Dr. Amr S. Ghoneim

Structures & Strategies for State Space Search
Sheet # 4

No. Of Questions: 6 **No. Of Pages: 3**

<u>Question 1:</u> Define the following terms:

a. State Space Graph.
b. Exhaustive Search.
c. Heuristics.
d. Path.
e. Rooted Graph.
f. Tree.

<u>Question 2:</u> Describe using drawing the Kongsberg problem "Euler Tour."

<u>Question 3:</u> Determine whether goal-driven or data-driven search would be preferable for solving each of the following problems. Justify your answer.

a. You have met a person who claims to be your distant cousin, with a common ancestor named John Doe. You would like to verify her claim.
b. Another person claims to be your distant cousin. He doesn't know the common ancestor's name but knows that it was no more than eight generations back. You would like to either find this ancestor or determine that she didn't exist.
c. A theorem prover for plane geometry.

<u>Question 4:</u> Answer the following Questions:

a. Discuss briefly the techniques used to reduce the search complexity when solving the *Traveling Salesman Problem*.

b. The following is a problem which can be solved using state-space search techniques: The Cannibals and Missionaries problem: "*Three cannibals and three missionaries come to a crocodile infested river. There is a boat on their side that can be used by either one or two persons. If cannibals outnumber the missionaries at any time, the cannibals eat the missionaries. How can they use the boat to cross the river so that all missionaries survive?*" Formalize the problem in terms of state-space search. You should:

- Suggest a suitable representation for the problem state.
- State what the initial and final states are in this representation.

- State the available operators/rules for getting from one state to the next, giving any conditions on when they may be applied.
- Draw the first two levels of the directed state-space graph for the given problem.

c. Briefly discuss the advantages and disadvantages of depth and breadth first search. What sort of problem is each appropriate for?

d. How does the use of a *closed* node list reduce the amount of search required in graph search?

e. Referencing *4.b*, Solve the following problem "*A farmer with his dog, rabbit and lettuce come to the east side of a river they wish to cross. There is a boat at the river's edge, but of course only the farmer can row. The boat can only hold two things (including the rower) at any one time. If the dog is ever left alone with the rabbit, the dog will eat it. Similarly, if the rabbit is ever left alone with the lettuce, the rabbit will eat it. How can the farmer get across the river so that all four characters arrive safely on the other side?*"

f. The state space of many two-player games (e.g. chess, checkers, and tic-tac-toe) can conveniently be represented as And/Or trees. Why? Give an example, stating which nodes are **And** and which are **Or**.

<u>Question 5:</u> Represent the following rules in first-order predicate calculus:

- Employees working on the HERMES or PEGASUS project work the latest hours.
- Employees working the latest hours are always the last ones out of the building.
- Whoever likes Van Gogh and was one of the last people out of the building must have stolen the painting.
- All New Yorkers and Californians like Van Gogh.

Show a goal-driven and/or search tree - based on your predicate-calculus rules - that proves that Willie Slick, a New Yorker working on the HERMES project, stole the painting**.**

<u>Question 6:</u> Consider the following NL "Natural Language" Grammar:

s → np, vp.
np → art, noun.
vp → verb.
noun → [cat].
noun → [dog].
art → [the].
art → [a].
verb → [jumps].

verb → [sings].
verb → [likes].

i. Sketch the parse tree for the sentence "the cat jumps".
ii. Add a rule to the grammar so that it will accept "the cat likes the dog".
iii. How would you prevent "the cat likes" being accepted?
iv. Add rule(s) and dictionary entries to the grammar so that it will accept "the cat on the mat jumps".

With our best wishes;
Dr. Hala Abdel-Galil & Dr. Amr S. Ghoneim