SOLVING PROBLEMS BY SEARCHING

Nguyễn Thị Hải Bình, PhD

Email: nth.binh@hutech.edu.vn

Contents

1. Introduction

2. State Space

3. Blind Search

4. Heuristic Search

5. Optimal Search

6. Exercises

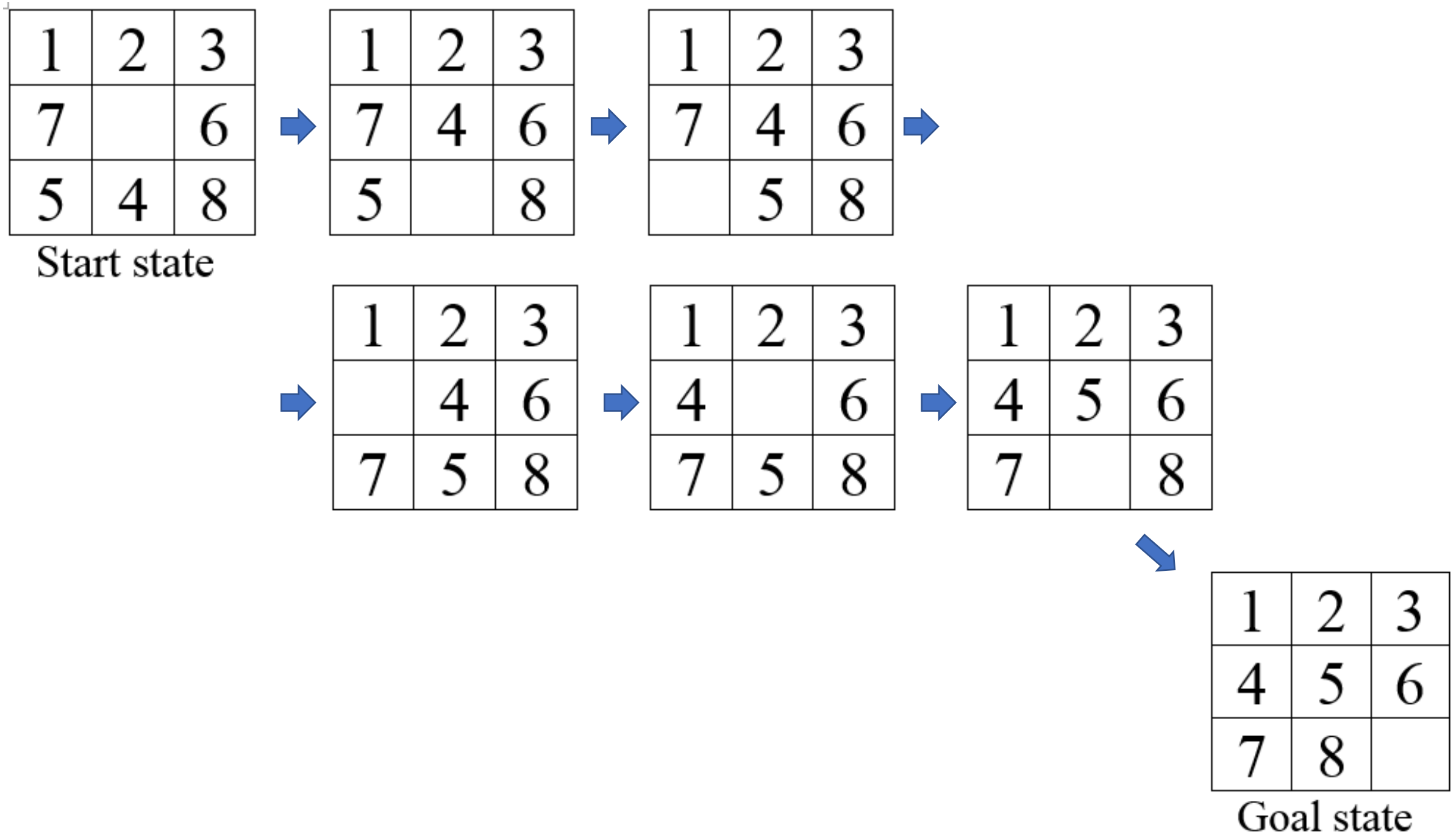
Example: 8-puzzle

1	2	3
7		6
5	4	8

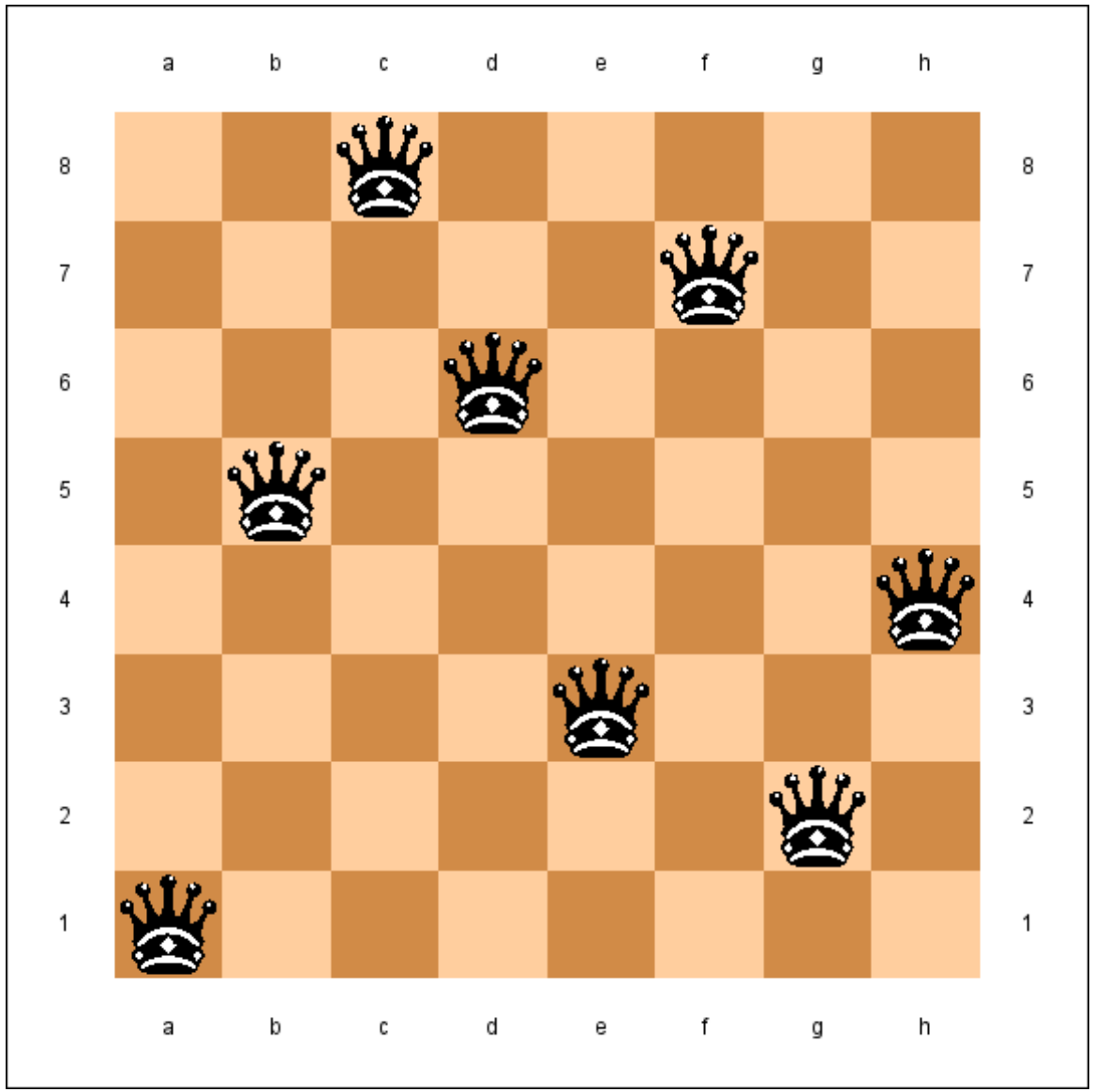
Start state

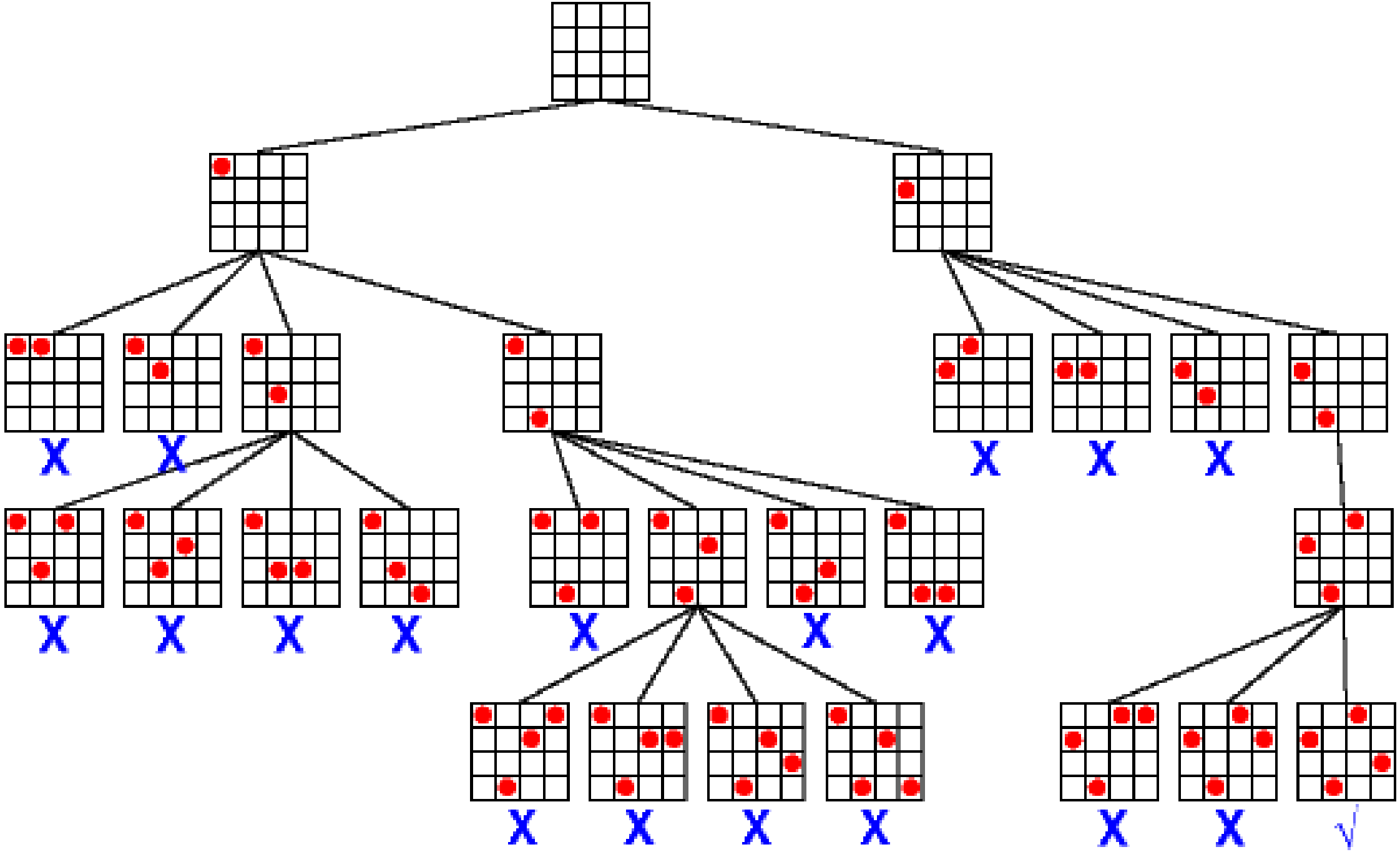
1	2	3
4	5	6
7	8	

Goal state


1 2 3
7 6
5 4 8
Start state
1 2 3
7 4 6
5 8
1 2 3
7 4 6
5 8
1 2 3
4 6
7 5 8
1 2 3
4 6
7 5 8
1 2 3
4 5 6
7 8
1 2 3
4 5 6
7 8
Goal state

Example: 8 queen problem




X
X
X
X
X
X
X
X
X
X
X
X
X
X
X
√

Examples

- Route finding
- Travelling salesman problem
- VLSI layout (**Very-large-scale integration**)
- Robot navigation
- Web searching

Solving Problems by Searching

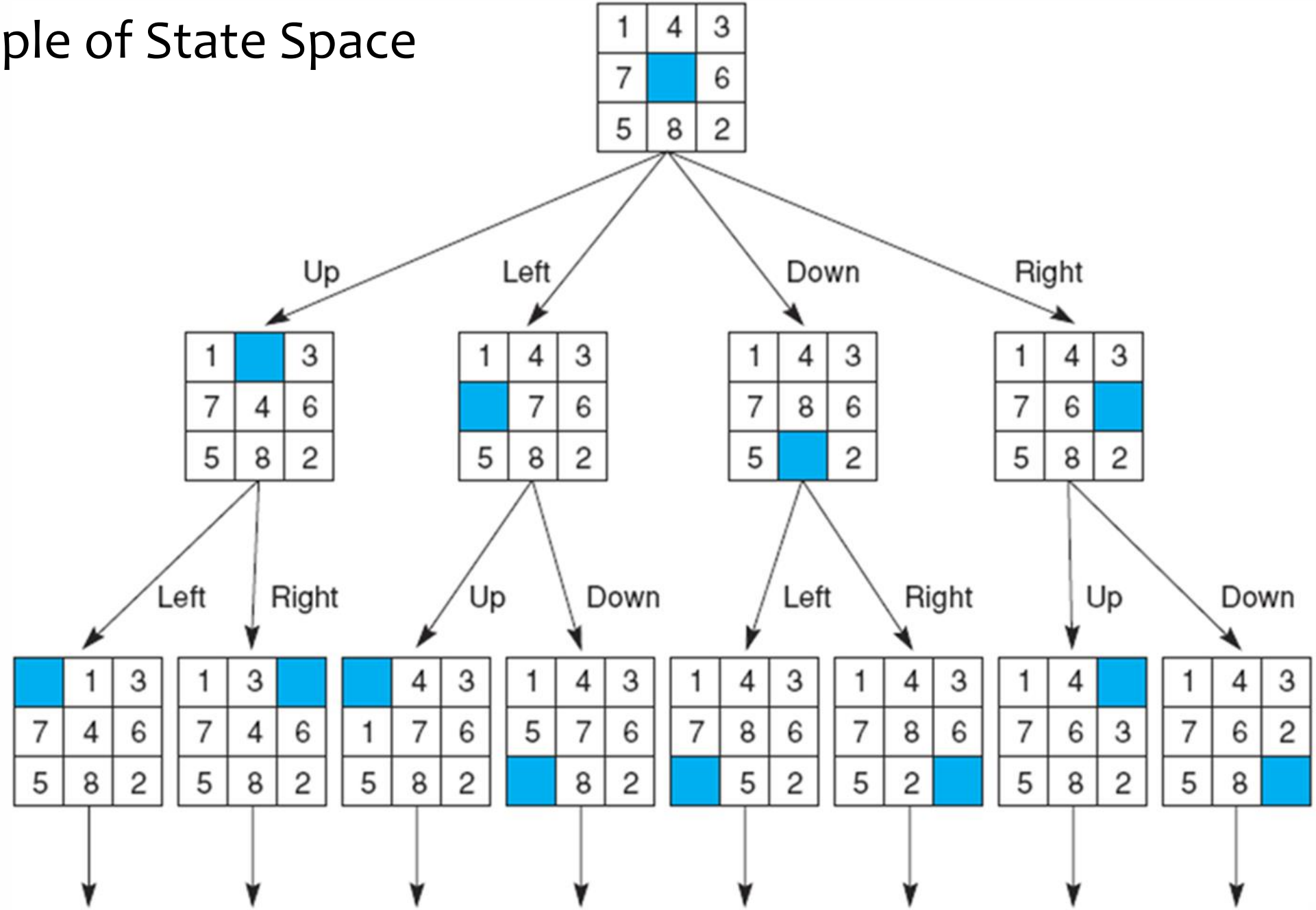
- Searching algorithms are used to solve AI problems.

- **Step to solve problem by searching:**
  - Step 1. Specify state space
  - Step 2. Apply searching algorithms to solve the problem

State Space

- A search problem consists of:
  - A State Space: Set of all possible states where you can be.
  - A Start State: The state from where the search begins.
  - Goal States (or A Goal Test - A function that looks at the current state returns whether or not it is the goal state)
- The Solution to a search problem is a sequence of actions, called the plan that transforms the start state to the goal state.
- This plan is achieved through search algorithms.

Example of State Space



Represent Problem in State Space

We must define:

- State definition.
- Start state.
- Operators.
- Set of goal states.

Example: State space of 8-puzzle

- State = positions of cells.
- Operators = Movement of blank cell
  - Include: Left, Right, Up, and Down
- Start state.
- Goal state.

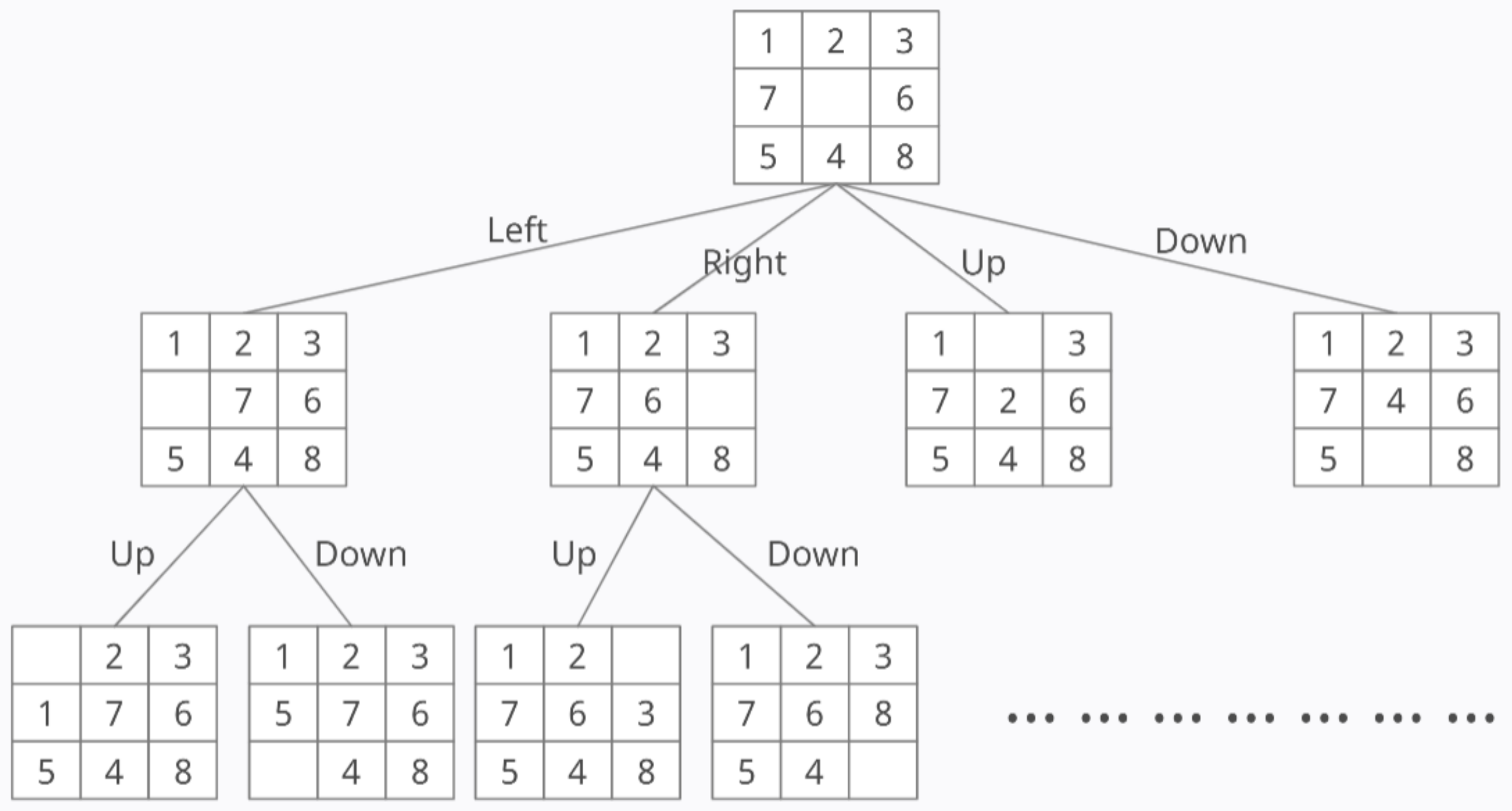
1	2	3
7		6
5	4	8

Start state

1	2	3
4	5	6
7	8	

Goal state

A part of state space



Example: State space of 8-Queen

- State = ?
- Operators = ?
- Start state = ?
- Goal states = ?

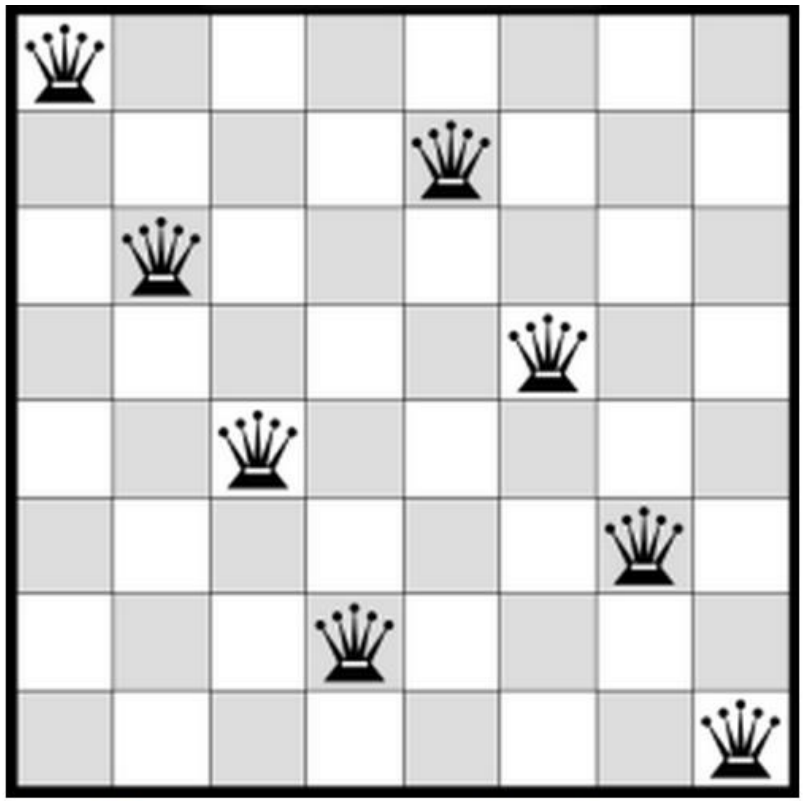
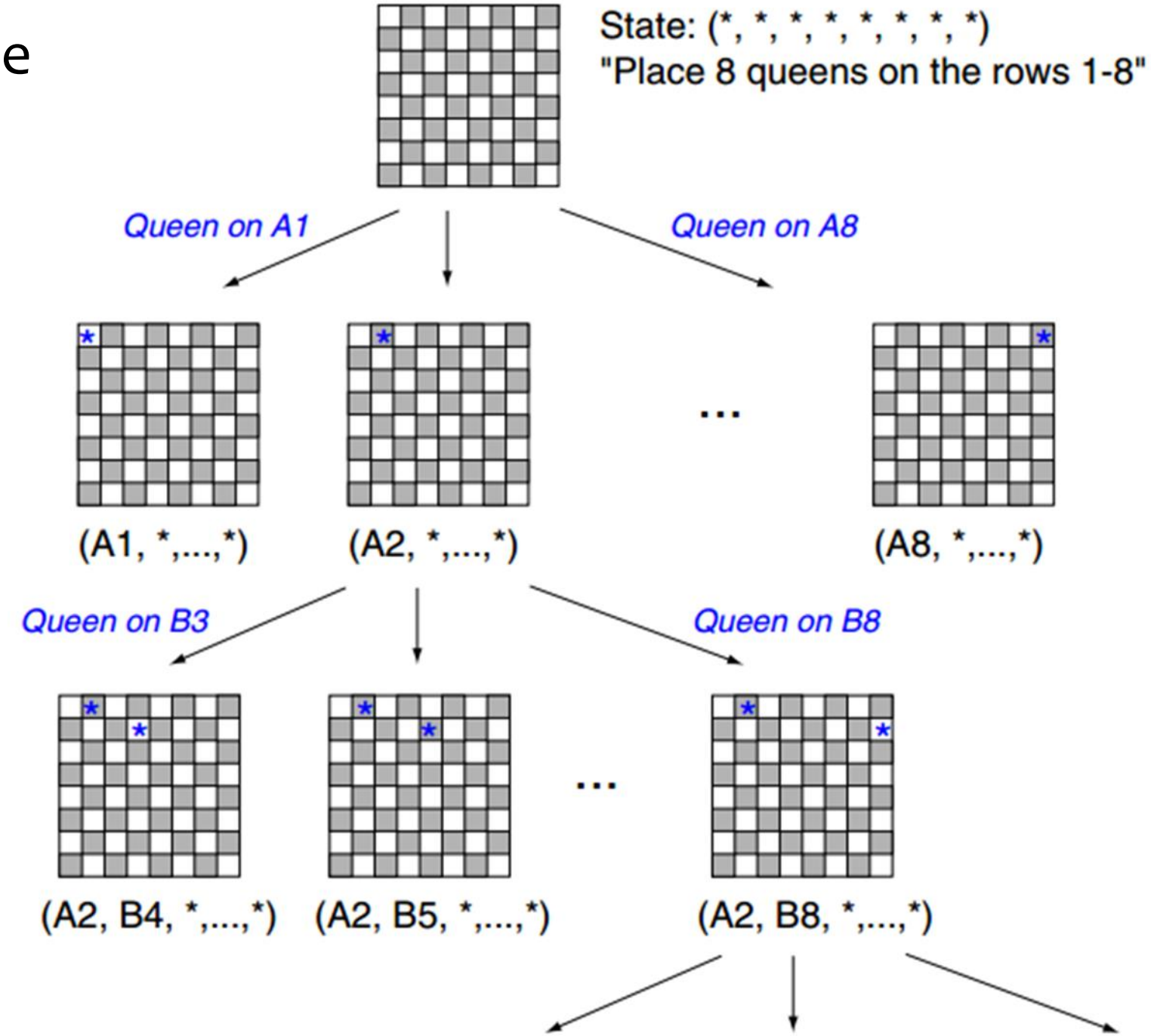
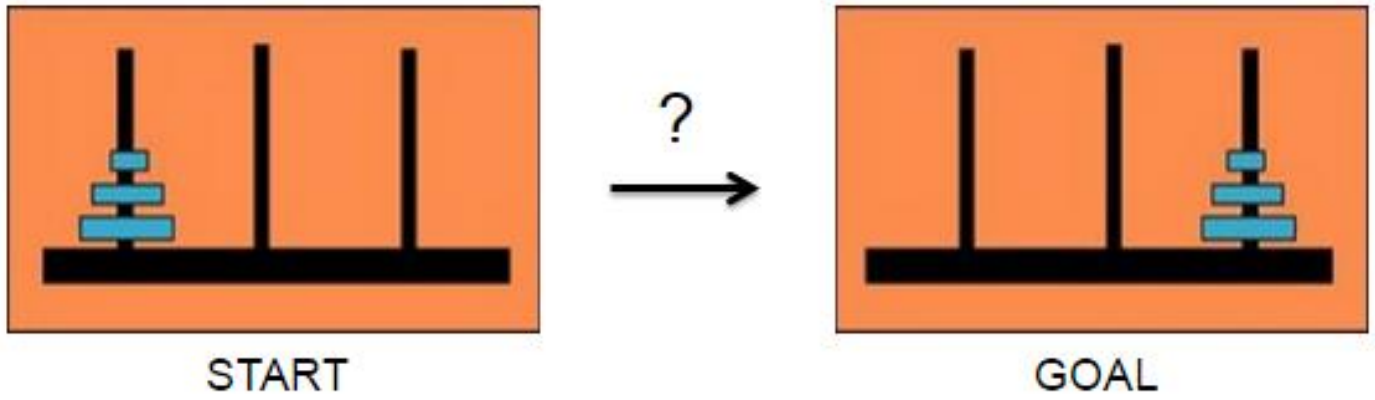


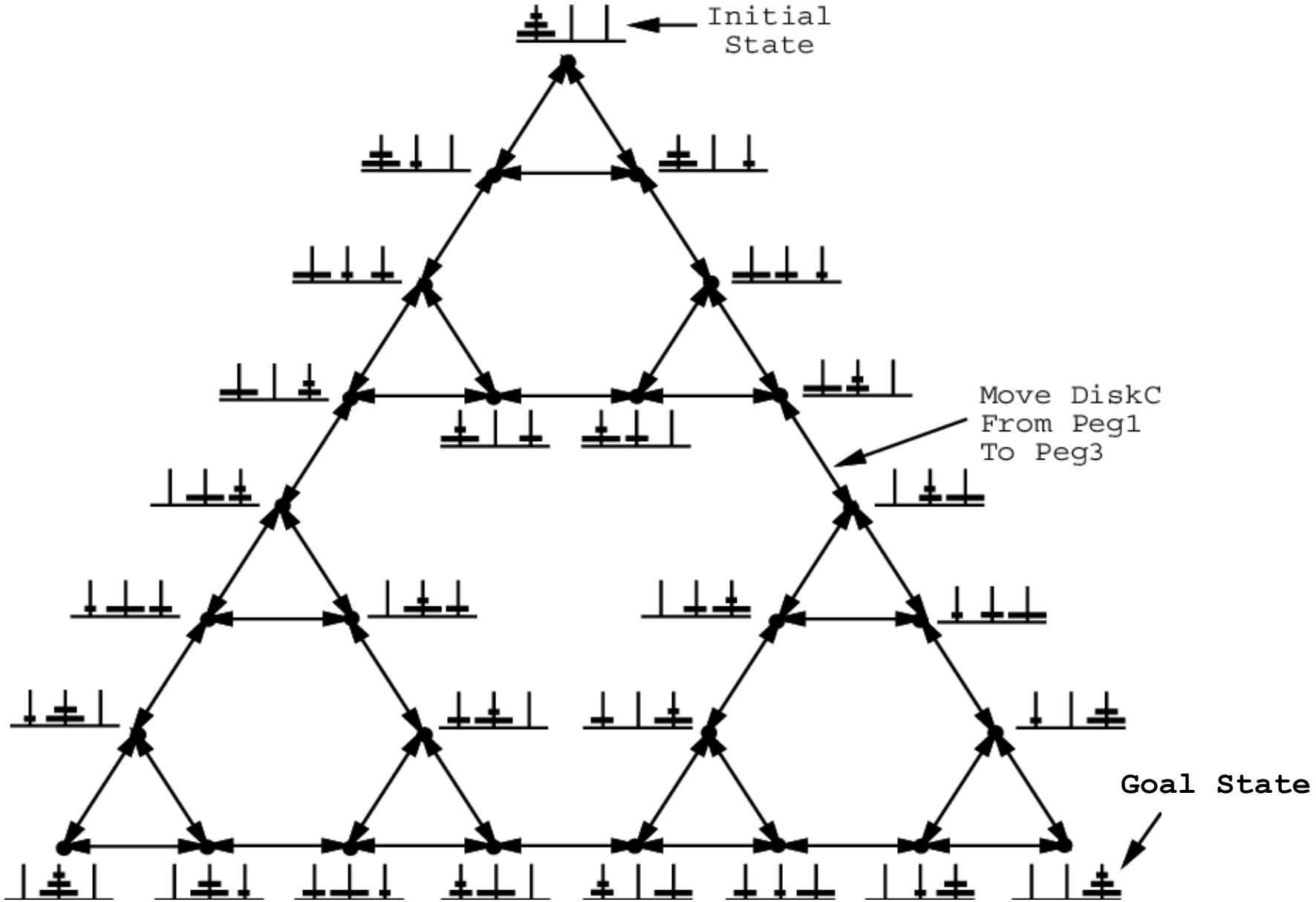
Figure 1: Almost a solution of the 8-queens problem

A part of state space



Example: State space of Hanoi Tower problem





Knoblock, Craig. (2000). Abstracting the Tower of Hanoi. Working Notes of AAAI-90 Workshop on Automatic Generation of Approximations and Abstractions.

SEARCHING ALGORITHMS

Searching Algorithms

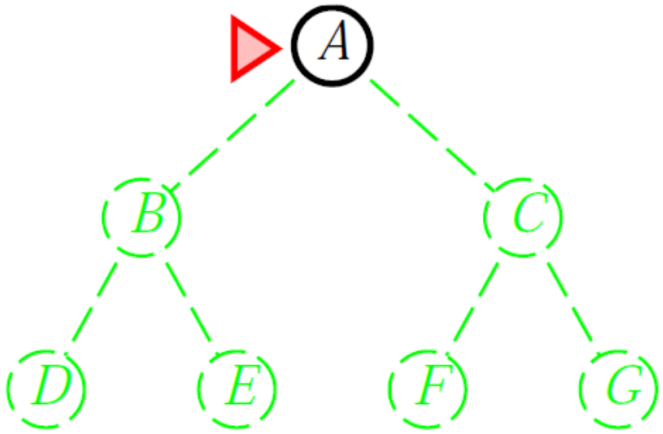
- **Blind Search**
  - Breadth First Search
  - Depth First Search
  - Limited Depth First Search
  - Iterative Depth First Search
- **Heuristic Search**
  - Best First Search
  - Hill Climbing
- **Optimal Search**
  - A^T Search
  - A^{KT} Search
  - A* Search
  - Branch and Bound Search
- **Adversarial Search**
  - Minimax
  - Alpha-beta prunning

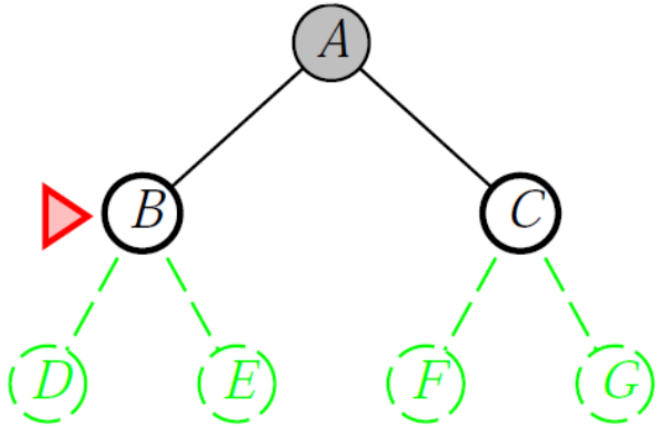
BLIND SEARCH

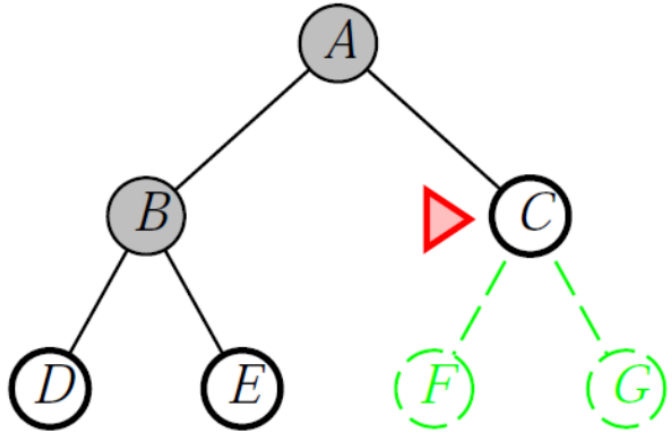
Searching Algorithms

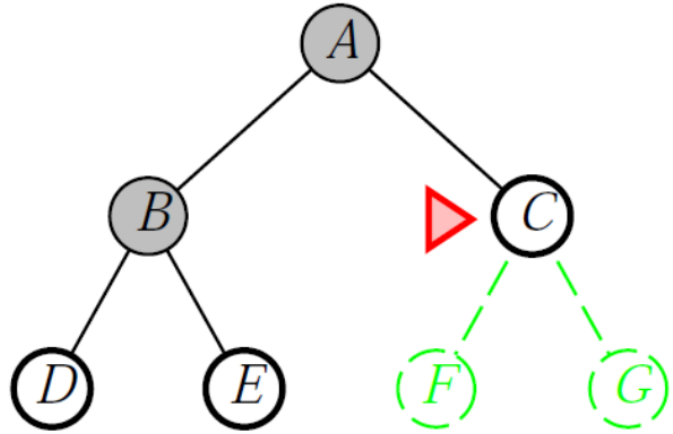
- **Blind Search**
  - Breadth First Search
  - Depth First Search
  - Limited Depth First Search
  - Iterative Depth First Search
- **Heuristic Search**
  - Best First Search
  - Hill Climbing
- **Optimal Search**
  - A^T Search
  - A^{KT} Search
  - A* Search
  - Branch and Bound Search
- **Adversarial Search**
  - Minimax
  - Alpha-beta prunning

Breath First Search – BFS









Breath First Search – BFS

Procedure Breadth_First_Search

begin

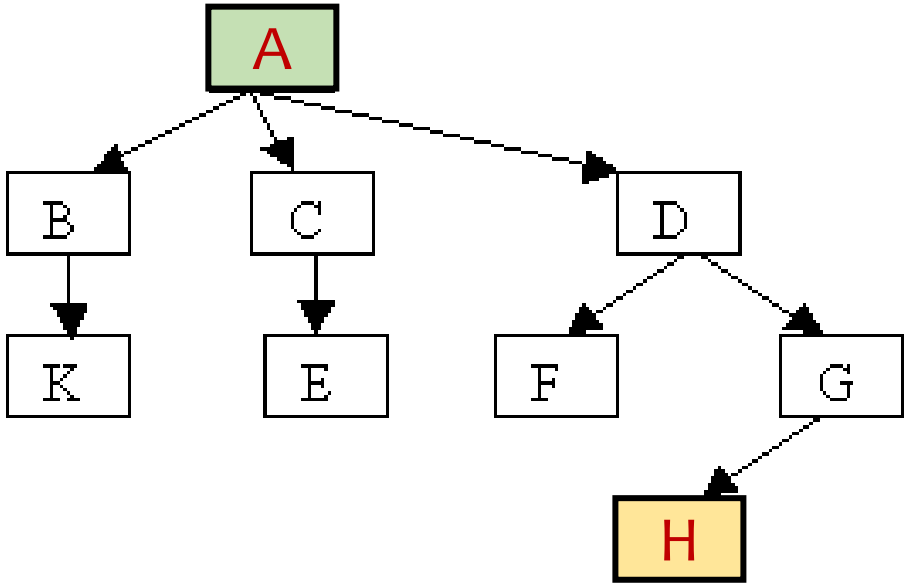
1. L = Empty Queue; L.enqueue(start_state)
2. **loop do**
   - 2.1 **if** $L = \emptyset$ **then**
     {searching fail; stop}
   - 2.2 u ← L.dequeue()
   - 2.3 **if** u is GOAL **then**
     {searching success; stop}
   - 2.4 **for** all neighbors v of u **do**
     {L.enqueue(v);
     father(v)=u};

end;

BFS – Example 1

Start state: A

Goal state: H



Steps of BFS:

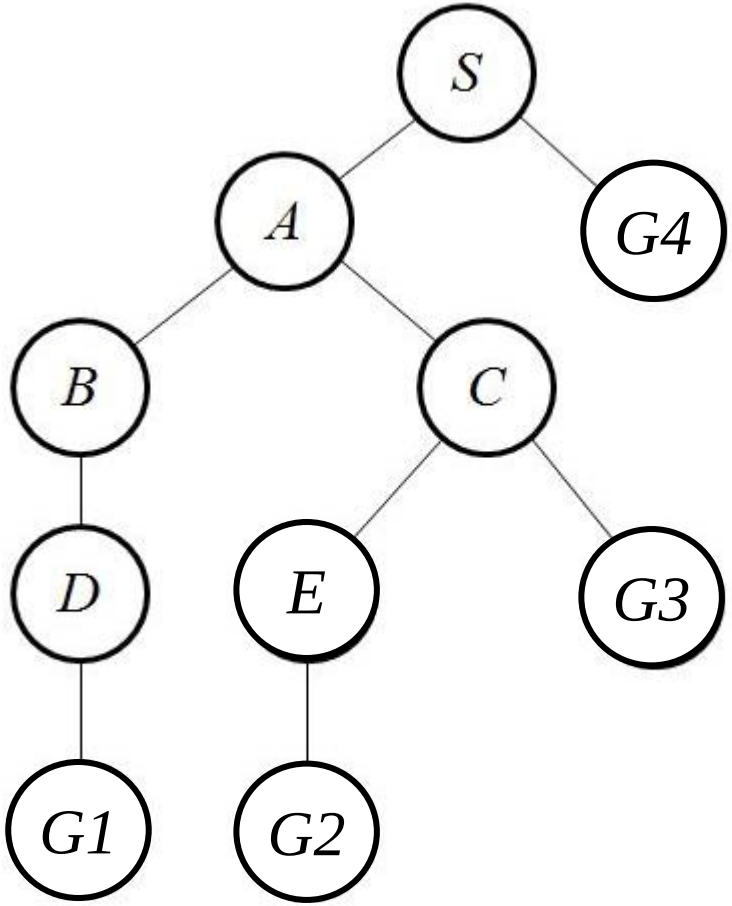
- Initialization: L = {A}

u	Neighbors of u	L

BFS - Example 2

Start state: S

Goal states: {G1, G2, G3, G4}



Steps of BFS:

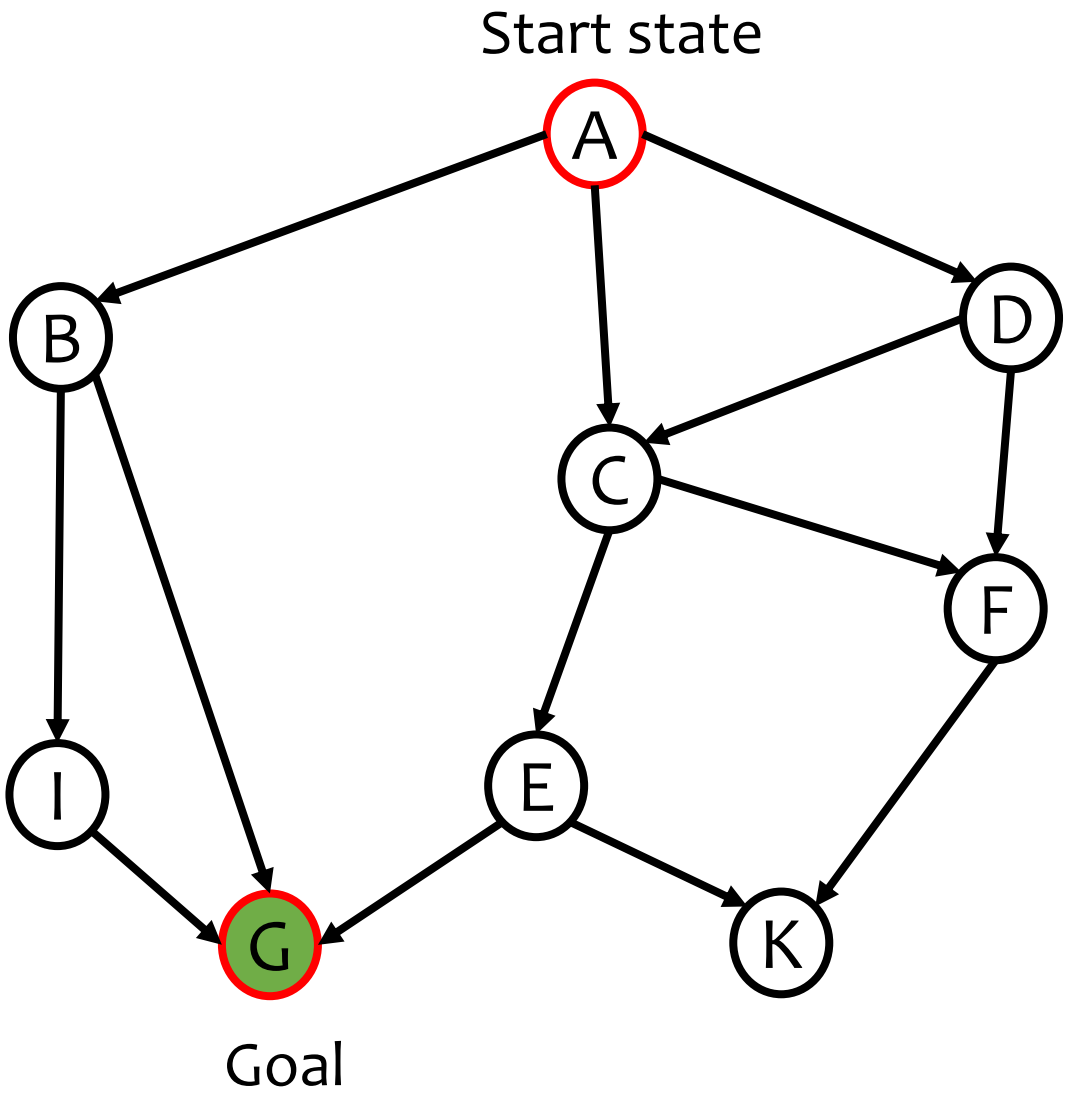
- Initialization: L = {S}

u	Neighbors of u	L

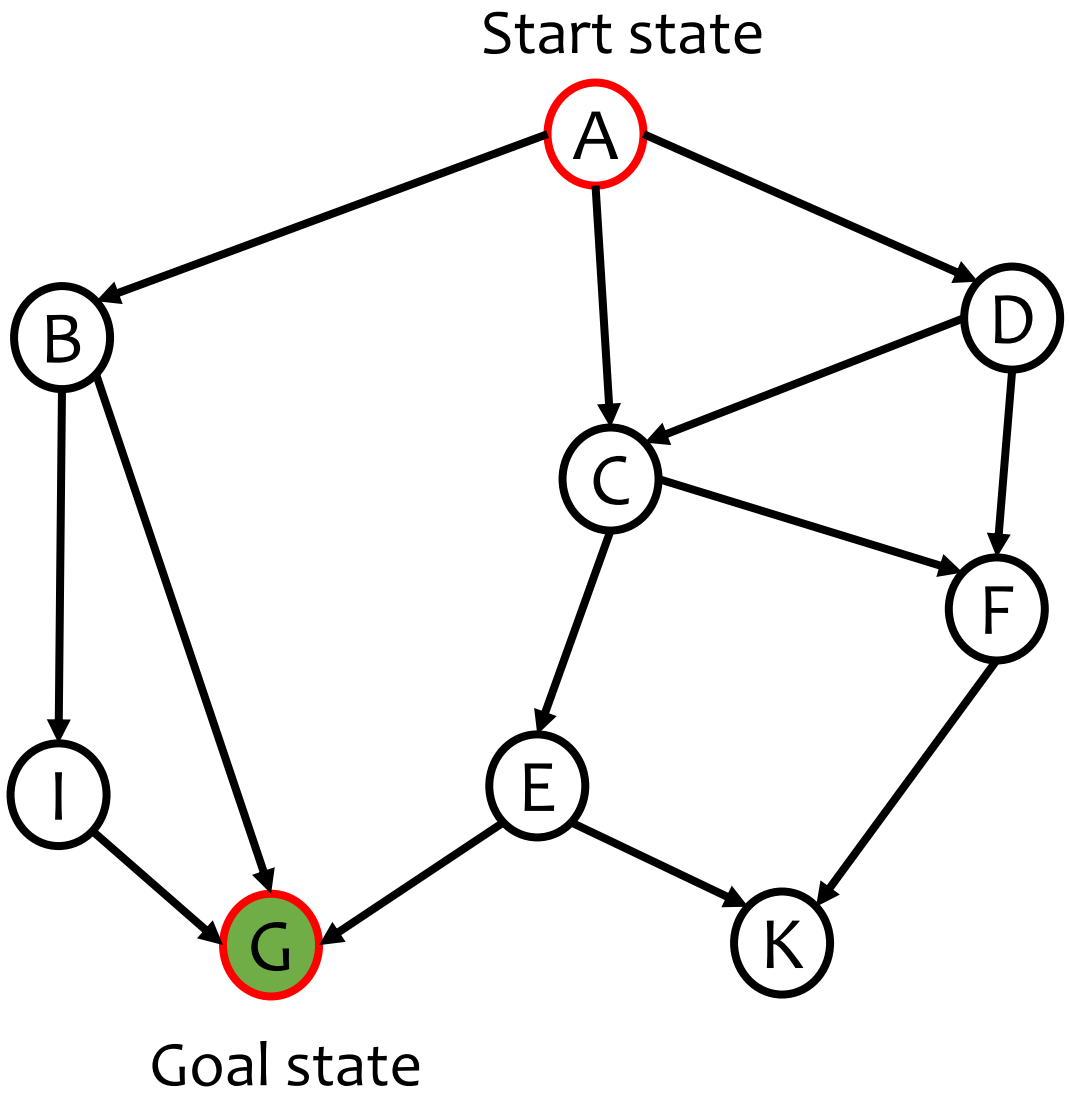
Properties of BFS

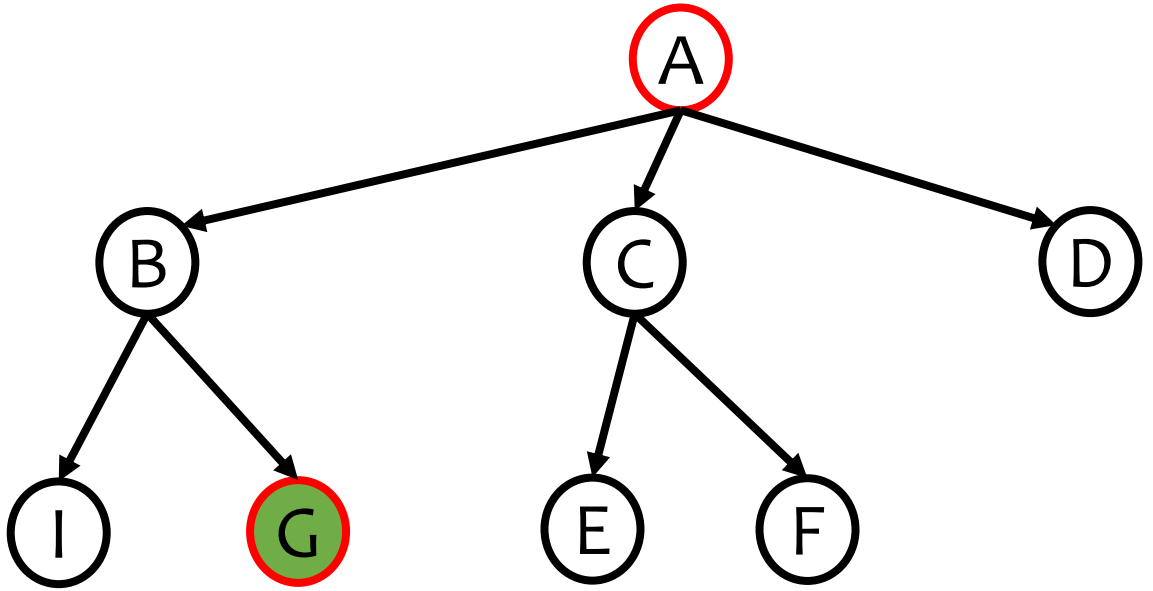
- Always find a path if the path exist.
- The found path is optimal (if cost = 1 per step)
- Time complexity = $1 + b + b^2 + \ \ldots + b^d = O(b^d)$
  - b = maximum branching factor
  - d = maximum path length
- Memory complexity = $1 + b + b^2 + \ \ldots + b^d = O(b^d)$

Repeat State



u	Neighbors of u	L





Search tree if using BFS

BFS – Example 3: 8-puzzle problem

- State = position of cells
- Operators = movements of blank cell
  - Including: Up, Left, Right, Down

- Start state:

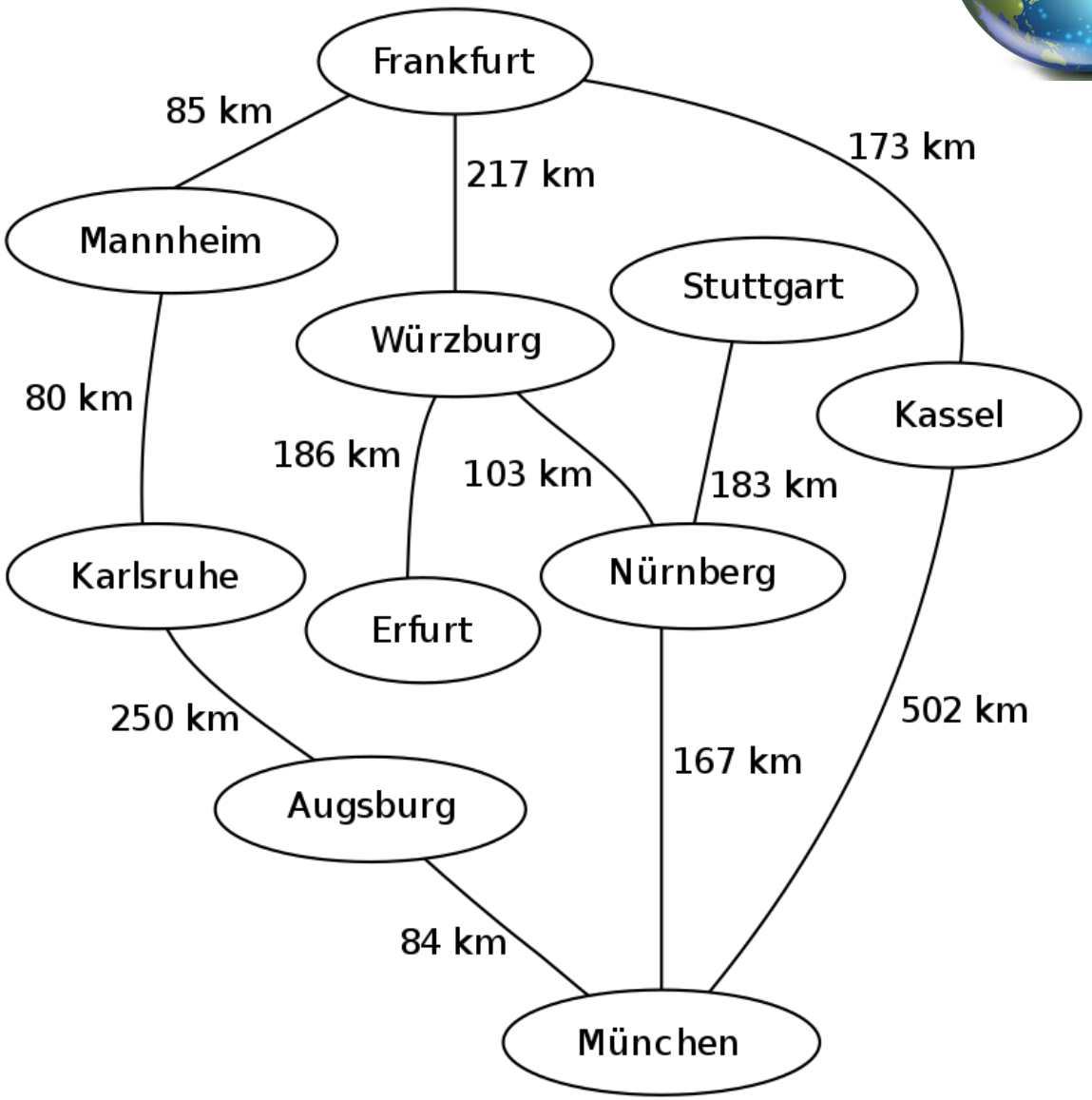
1	2	5
3	4	
6	7	8

- Goal state:

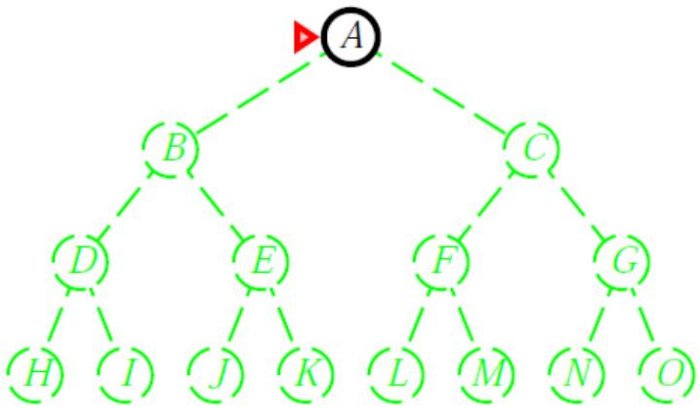
	1	2
3	4	5
6	7	8

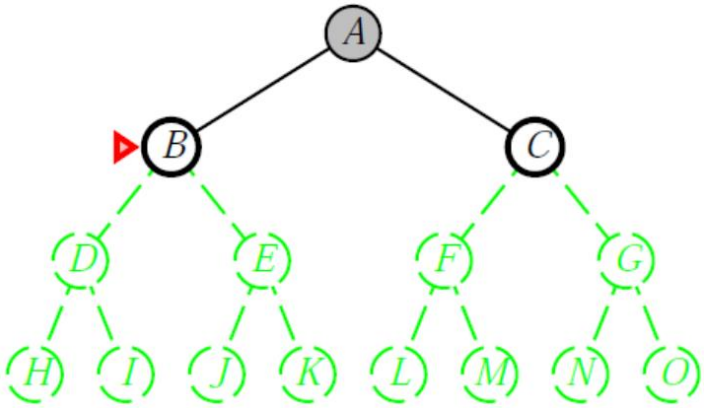
BFS – Example 4: Map finding

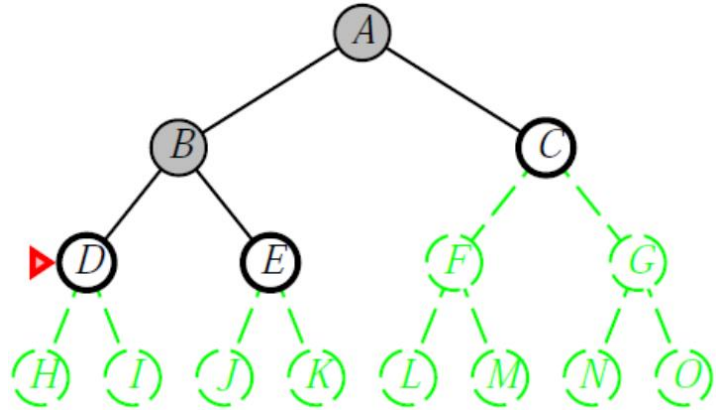
- Given a simple map of Southern Germany.
- Find the road from Frankfurt to München.

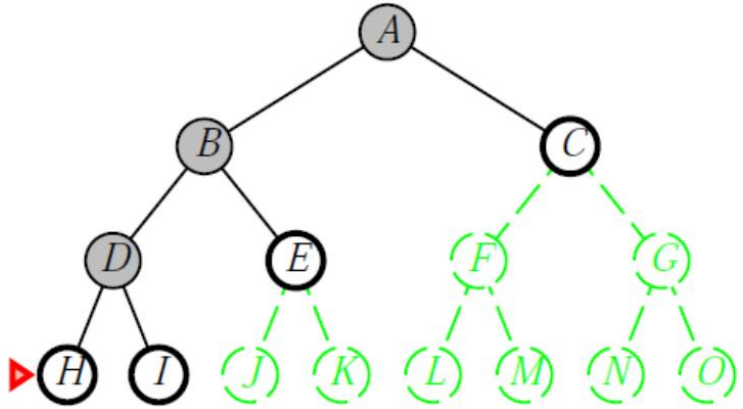


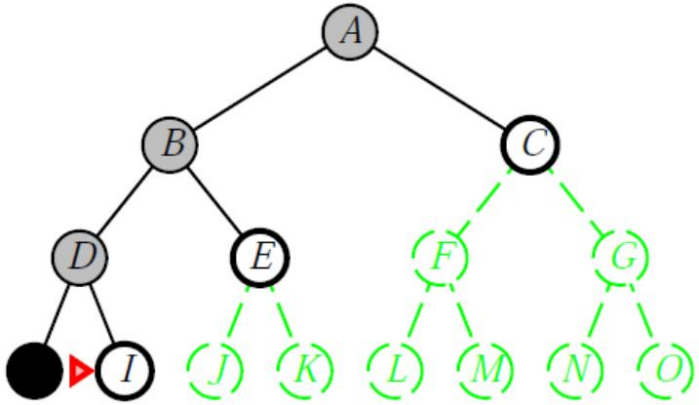
Depth First Search – DFS











Depth First Search – DFS

Procedure Depth_First_Search

begin

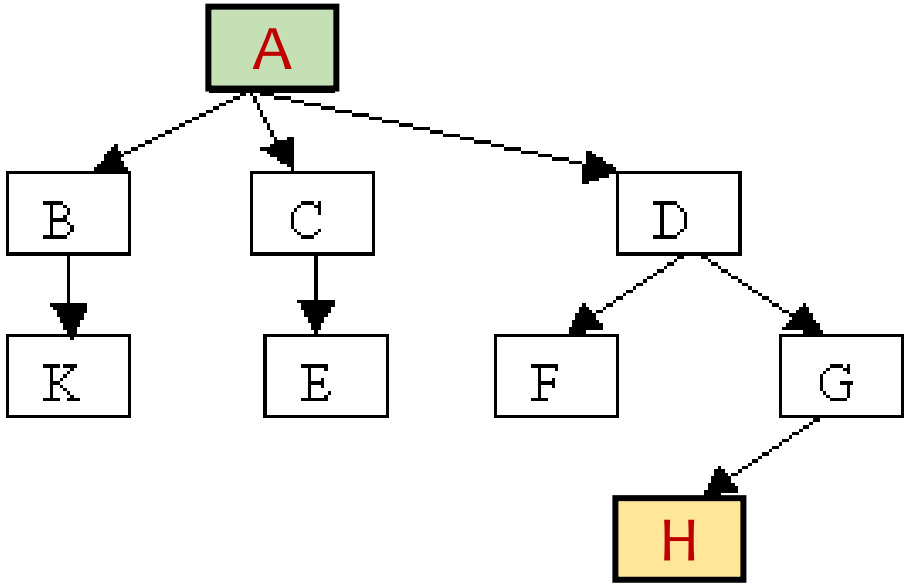
1. L = Empty Stack; L.push(start_state)
2. **loop do**
   - 2.1 **if** $L = \emptyset$ **then**
     {searching fail; stop};
   - 2.2 u ← L.pop()
   - 2.3 **if** u is GOAL **then**
     {searching success; stop};
   - 2.4 **for** all neighbors v of u **do**
     {L.push(v);
     father(v)=u};

end;

DFS – Example 1

Start state: A

Goal state: H



Steps of DFS:

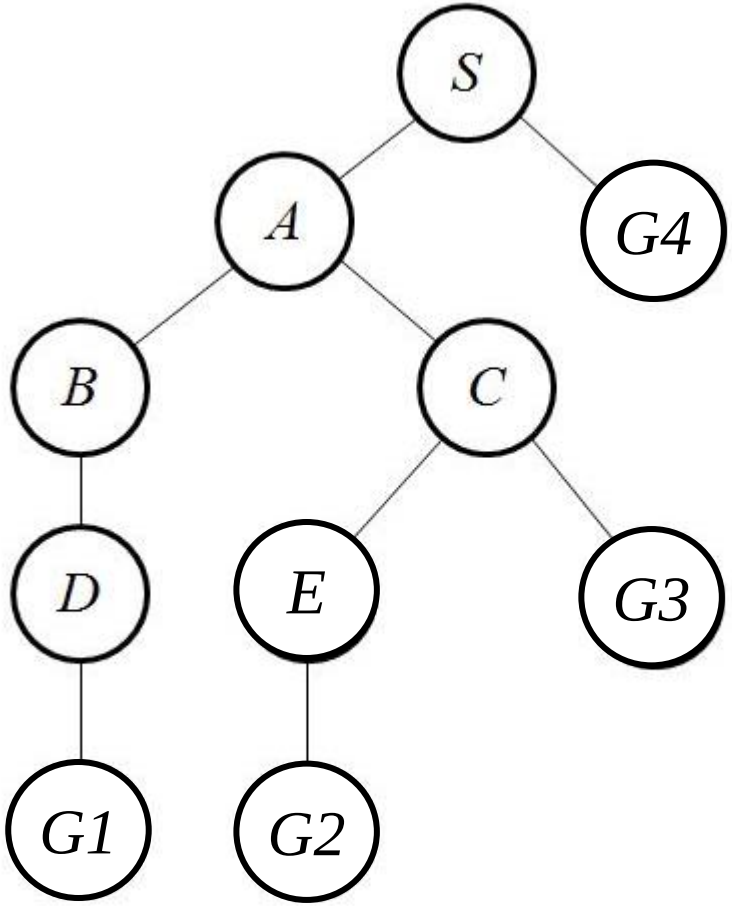
- Initialization: L = {A}

u	Neighbors of u	L

DFS - Example 2

Start state: S

Goal states: {G1, G2, G3, G4}



Steps of DFS:

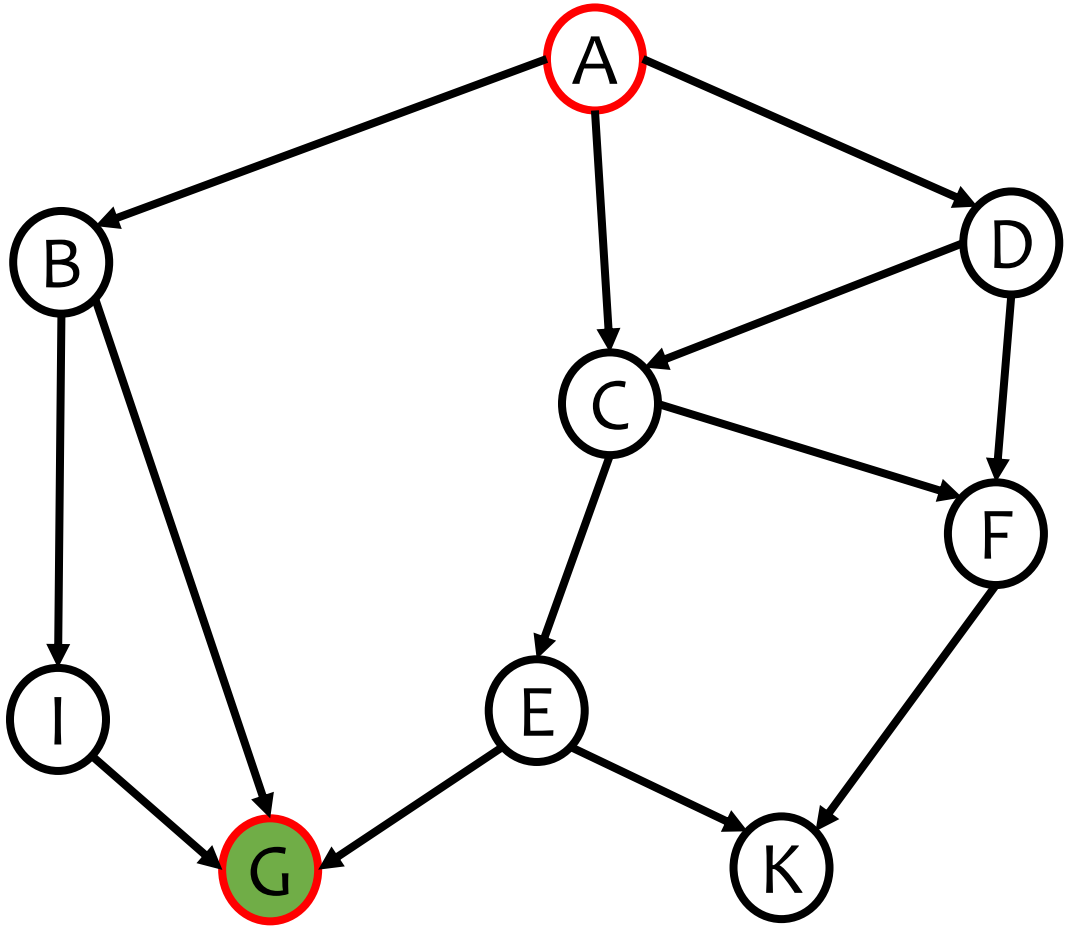
- Initialization: L = {S}

u	Neighbors of u	L

DFS – Example 3

Start state: S

Goal state: G



u	Neighbors of u	L

DFS – Example 4: 8-puzzle

- State = position of cells
- Operators = movements of blank cell
  - Including: Up, Left, Right, Down

- Start state:

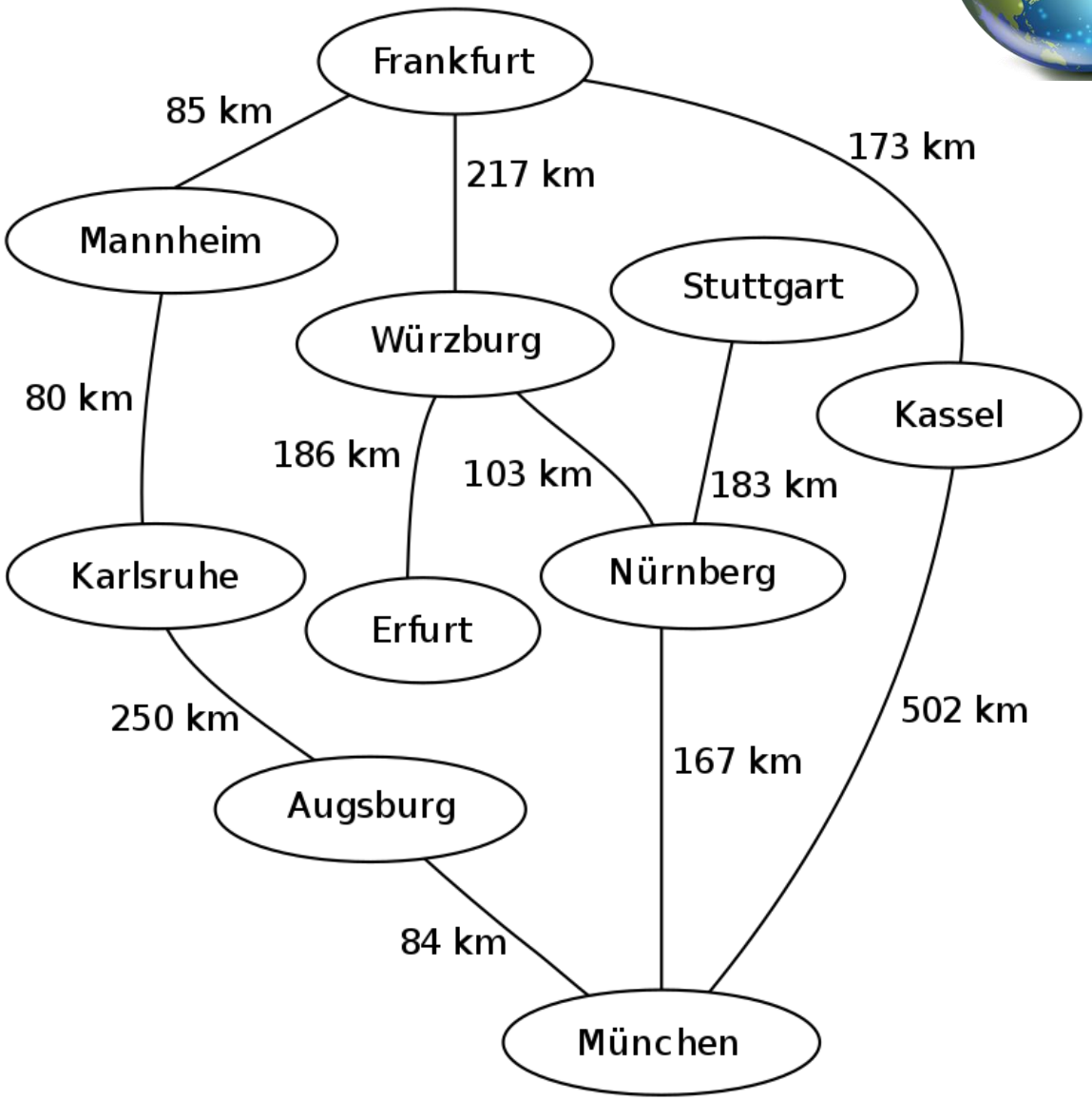
1	2	5
3	4	
6	7	8

- Goal state:

	1	2
3	4	5
6	7	8

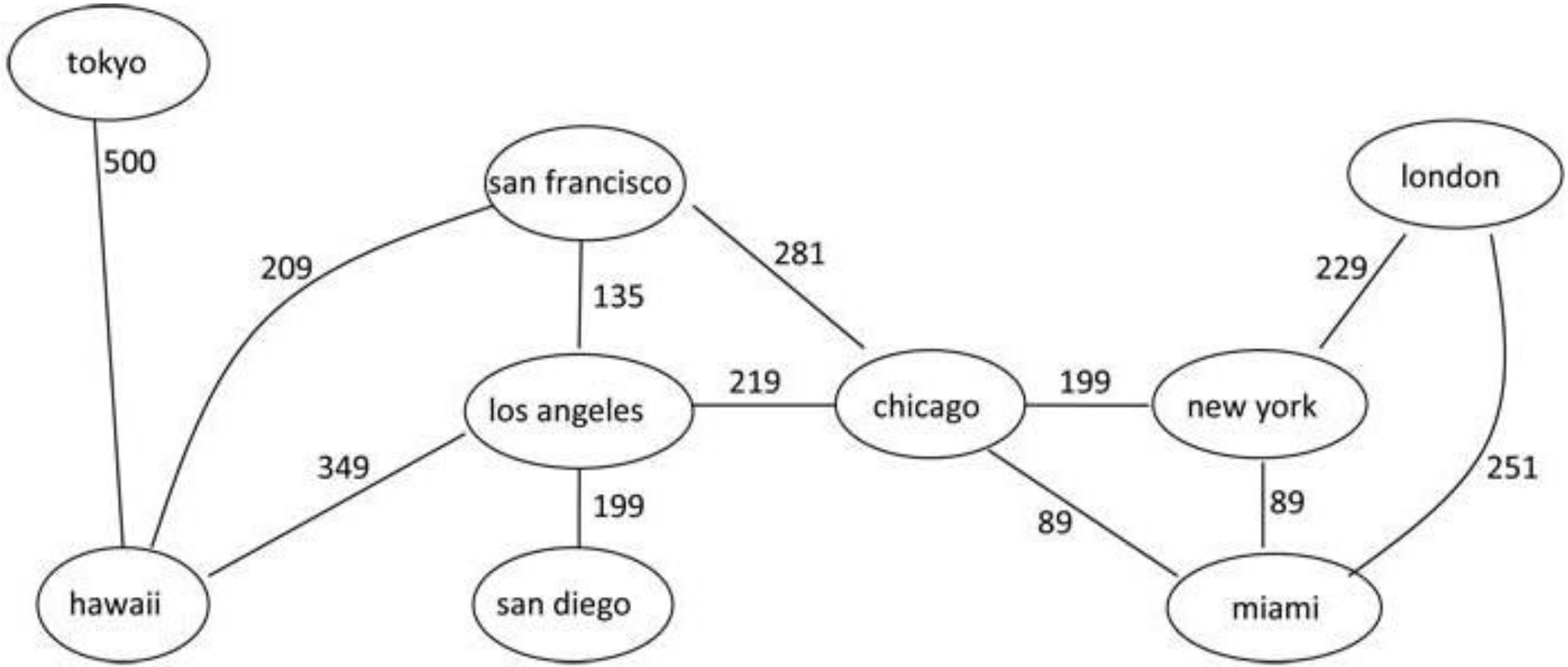
DFS – Example 5

- Given a simple map of Southern Germany.
- Find the road from Frankfurt to München.

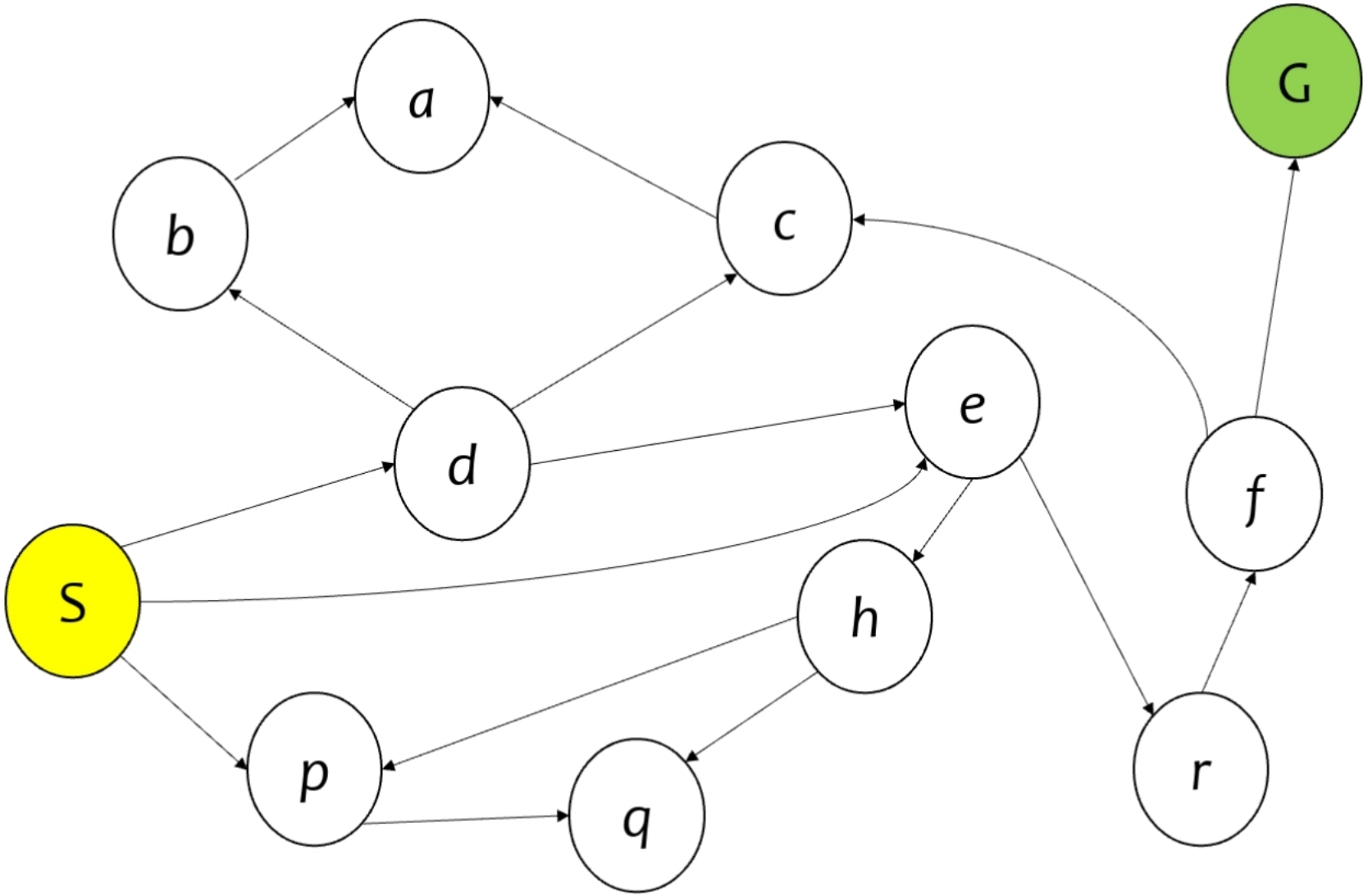


EXERCISES

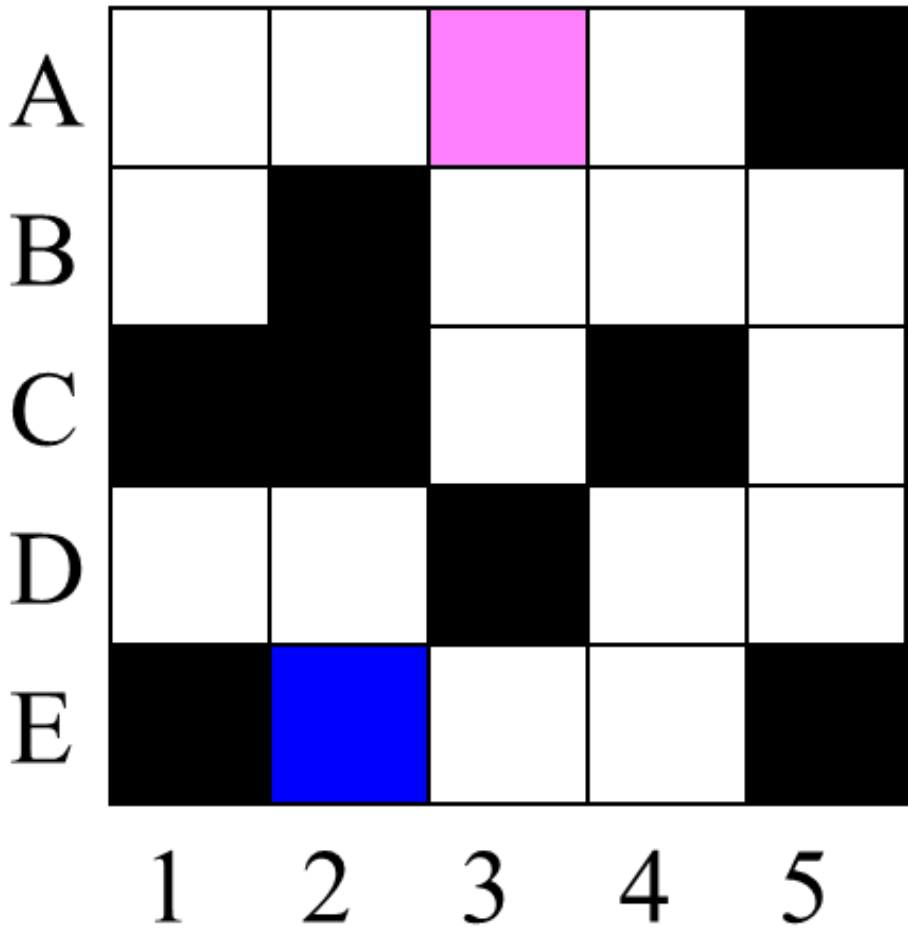
1. From Tokyo to London
2. From Hawaii to New York



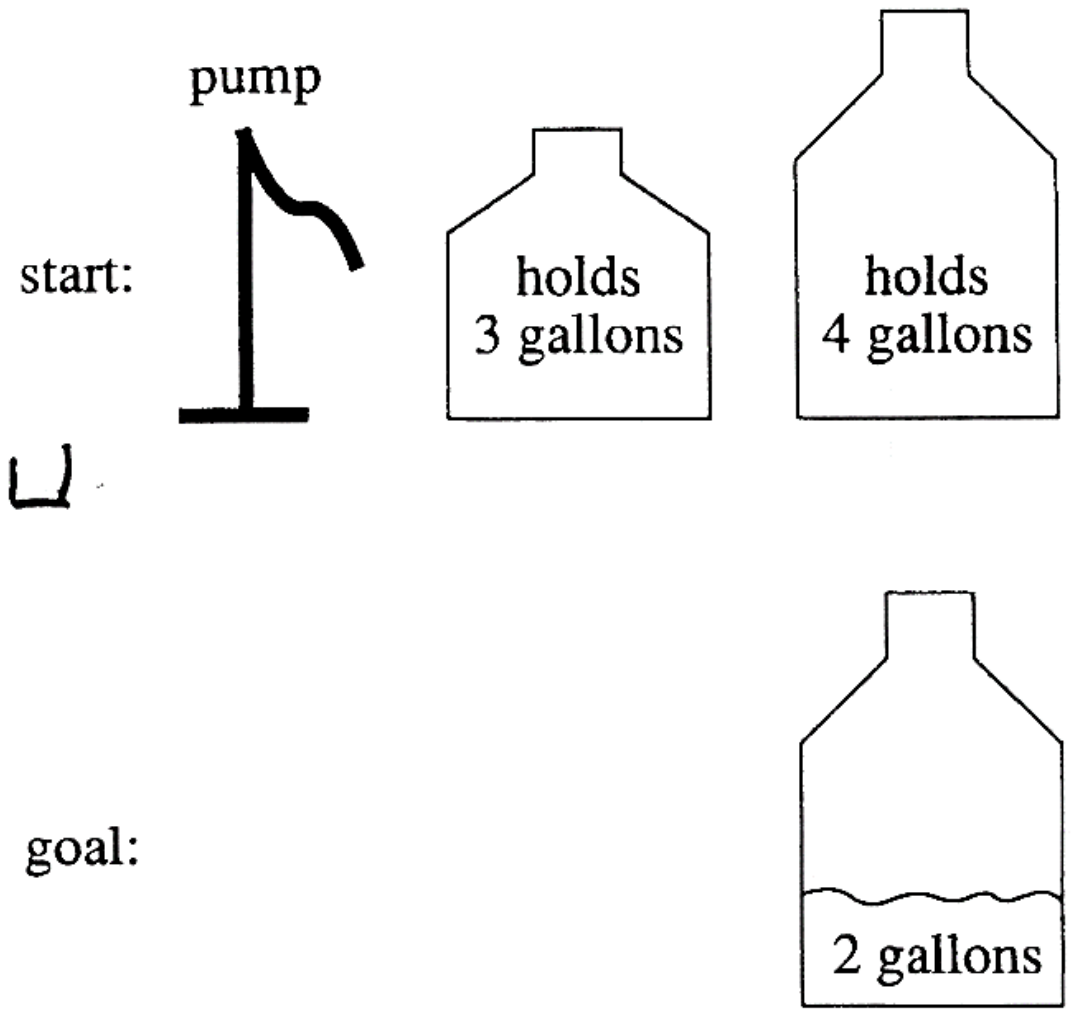
From S to G



Find the path from A3 to E2. Assumption: robot can move to left, right, or down.



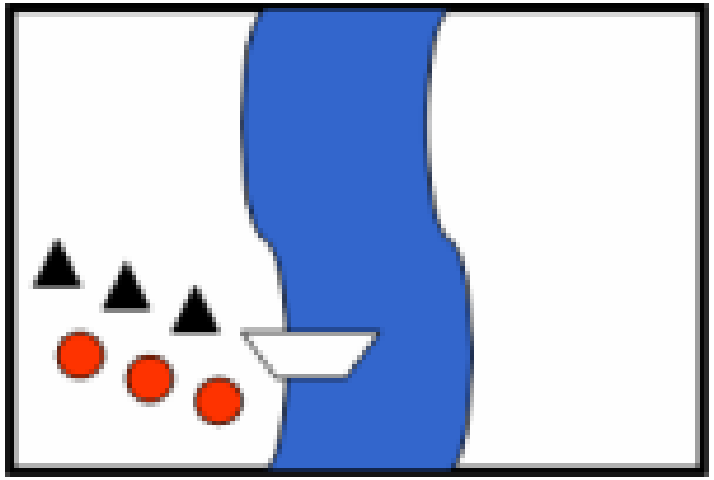
Water jug problem



Missionaries and Cannibals problem

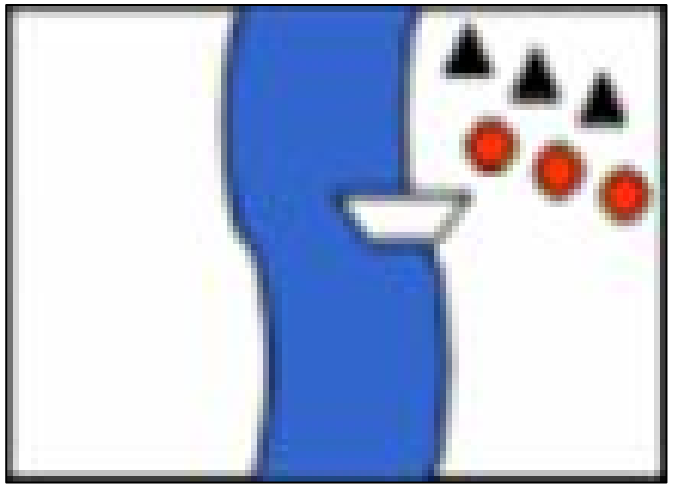
On one bank of a river are three missionaries and three cannibals. There is one boat available that can hold up to two people and that they would like to use to cross the river. If the cannibals ever outnumber the missionaries on either of the river's banks, the missionaries will get eaten.

How can the boat be used to safely carry all the missionaries and cannibals across the river?



Start state

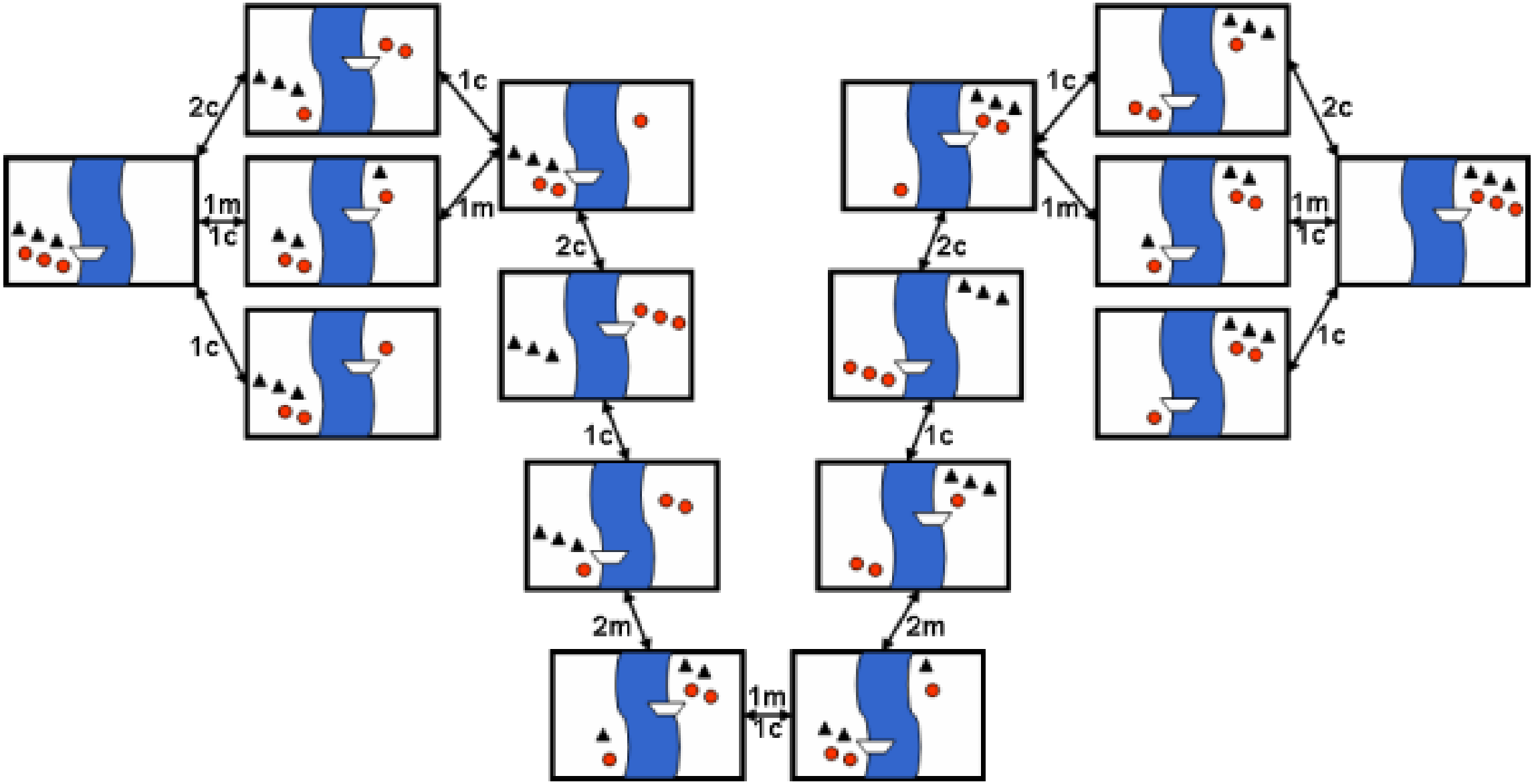
Black triangles represent missionaries.
Red circles represent cannibals.



Goal state

5 possible actions:

- One missionary crossing
- One cannibal crossing
- Two missionaries crossing
- Two cannibals crossing
- One missionary and one cannibal crossing



http://www.aiai.ed.ac.uk/~gwickler/missionaries.html

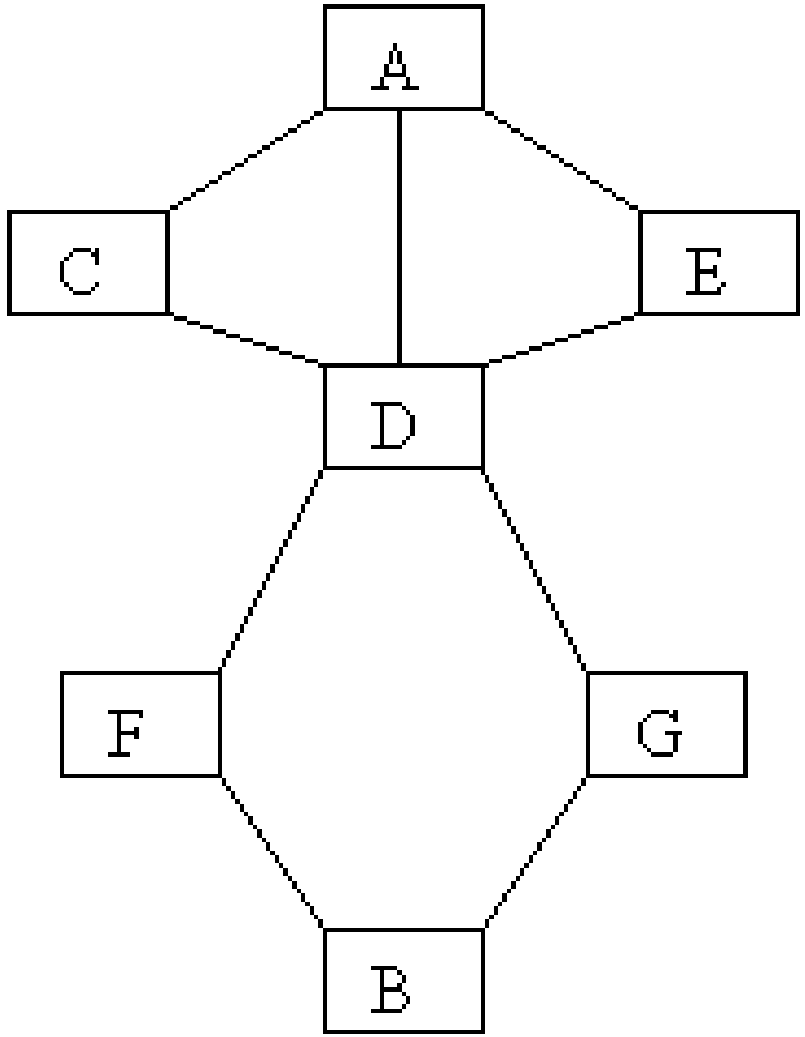
HEURISTIC SEARCH

Searching Algorithms

- Blind Search
  - Breadth First Search
  - Depth First Search
  - Limited Depth First Search
  - Iterative Depth First Search
- Heuristic Search
  - Best First Search
  - Hill Climbing
- Optimal Search
  - A^T Search
  - A^{KT} Search
  - A* Search
  - Branch and Bound Search
- Adversarial Search
  - Minimax
  - Alpha-beta prunning

Heuristic Search

- Example: find the path from A to B
  - If we know that the straight-line distance between C and B is less than the straight-line distance between D and B and between E and B.
  - How should we go?

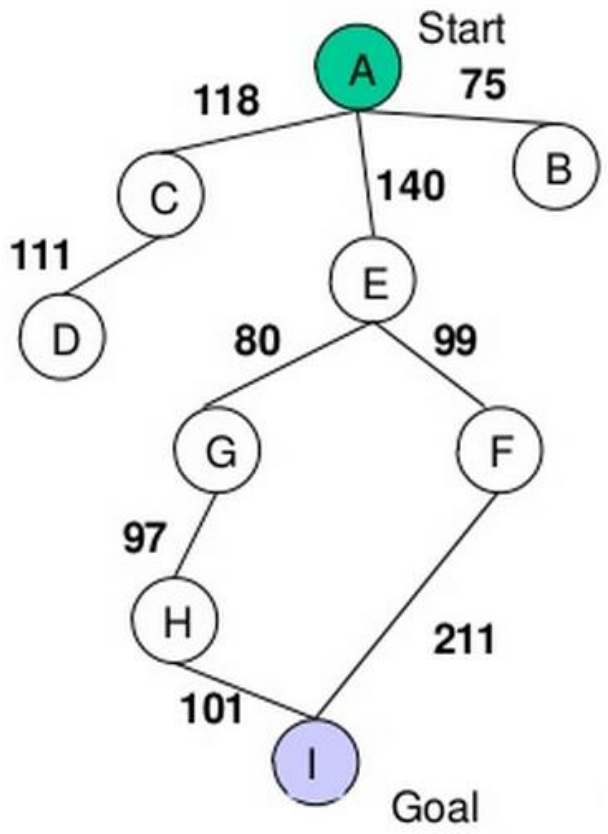


Heuristic function

- Heuristic function *h(n)*
  - Estimated distance from the state *n* to the goal.
  - *h(n)* = 0 → ???
- How to define a heuristic function?

Example of heuristic functions

- Heuristic function = straight-line distance between a city to the destination city.

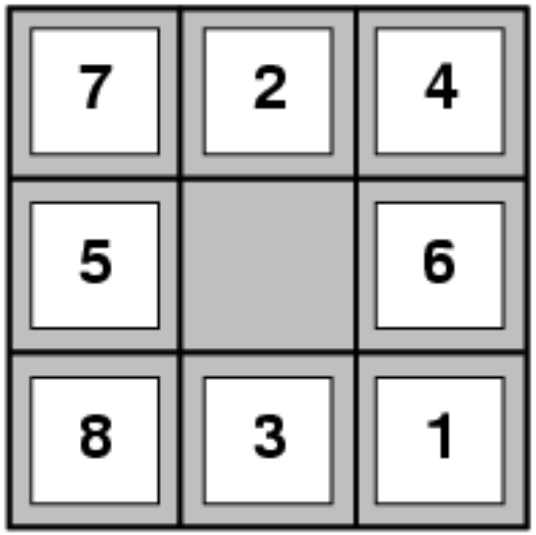


State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0

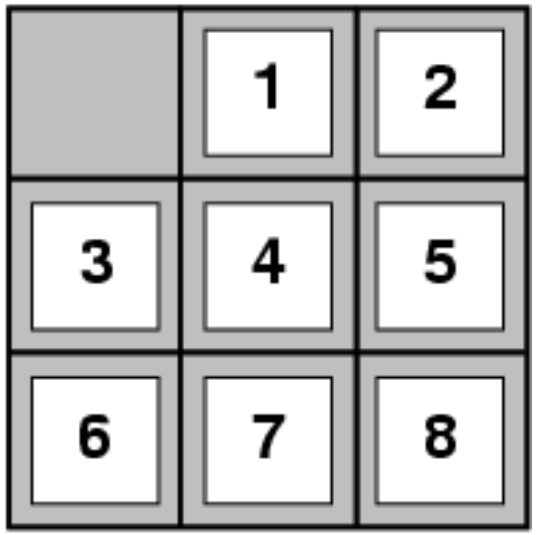
***h* (*n*) = straight-line distance heuristic**

Example of heuristic functions

- $h_1(n)$ = number of misplaced tiles (not include the blank)
- $h_2(n)$ = total Manhattan distance (the sum of the distances of the tiles from their goal positions)



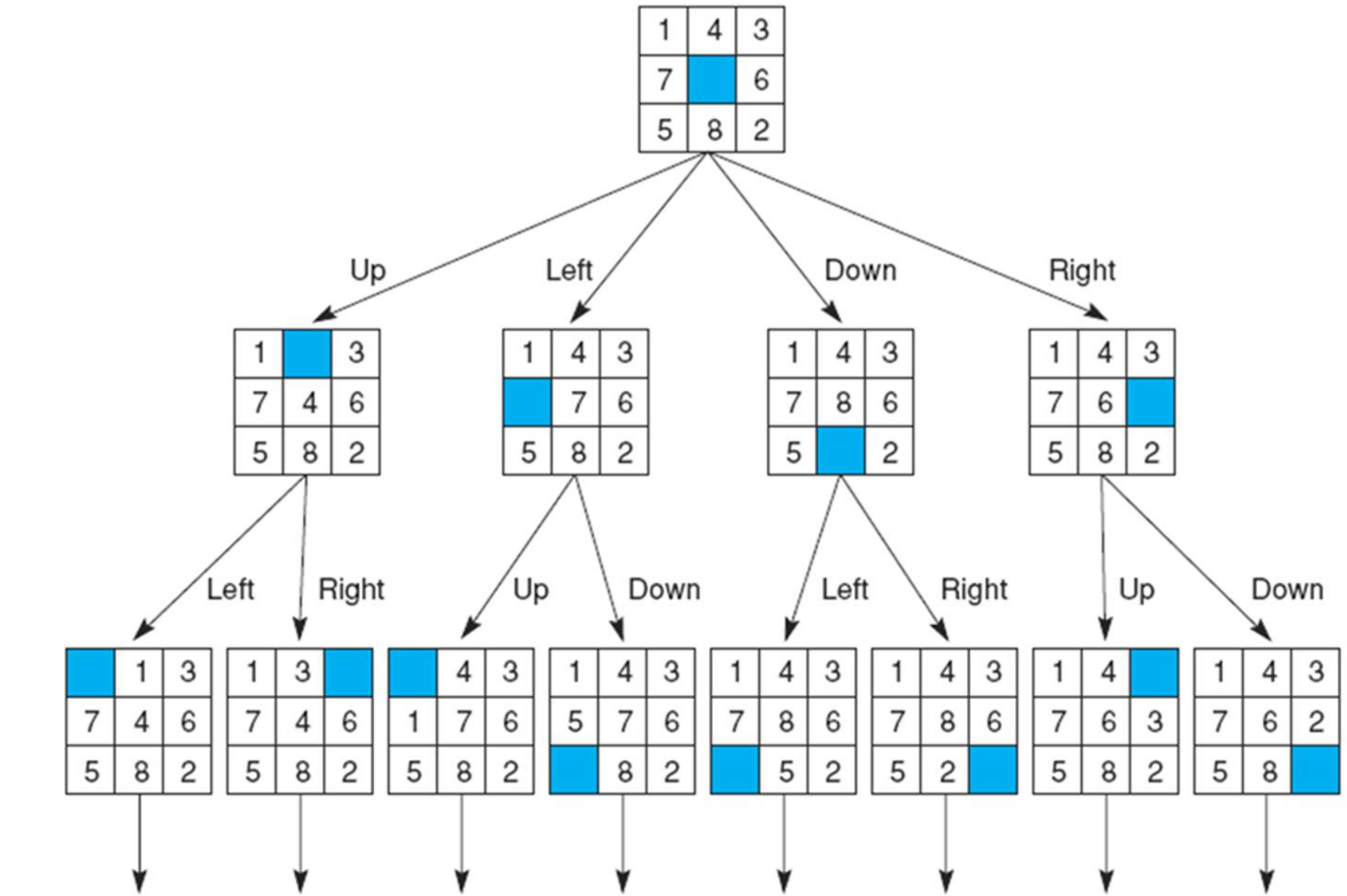
Start State



Goal State

h_1(start_state) = ?
h_2(start_state) = ?

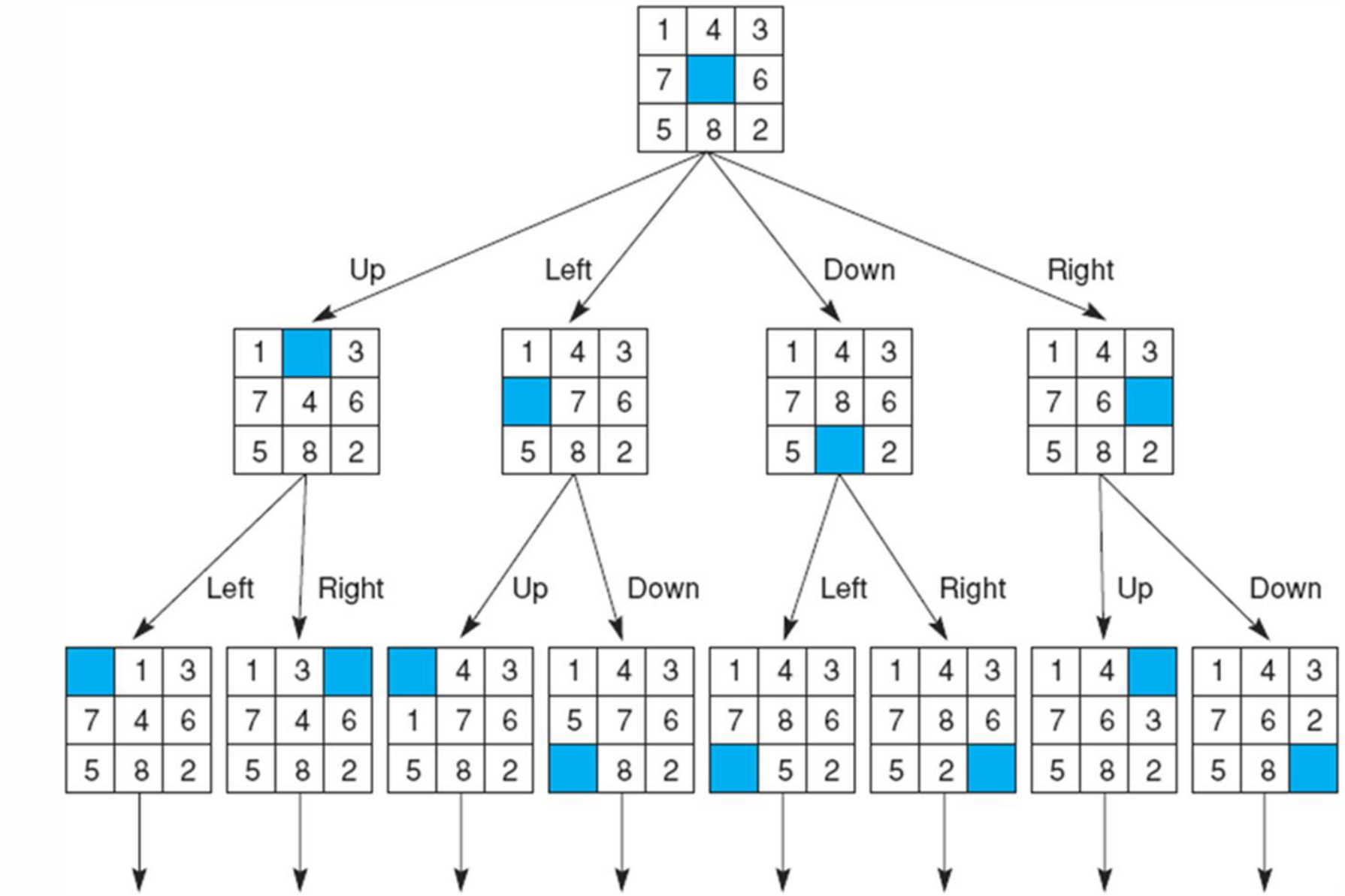
Compute h_1



1	2	3
4	5	6
7	8	

Goal state

Compute h_2

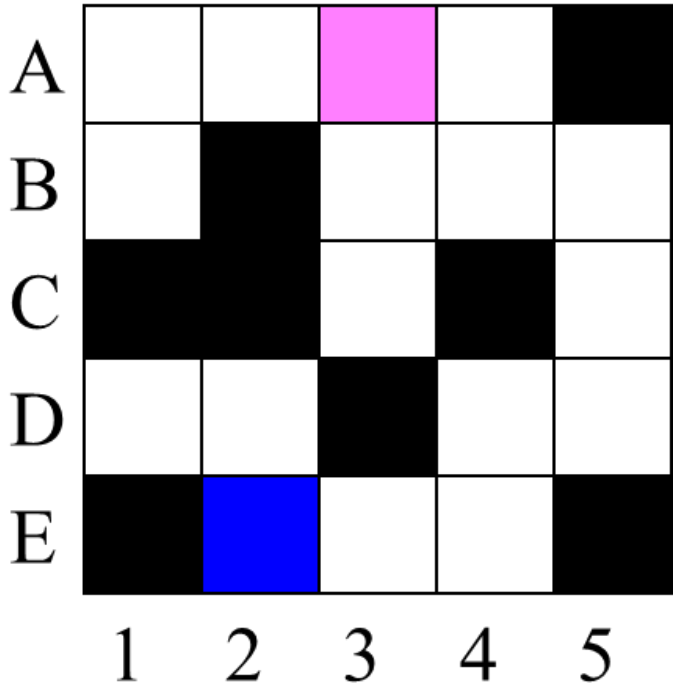


1	2	3
4	5	6
7	8	

Goal state

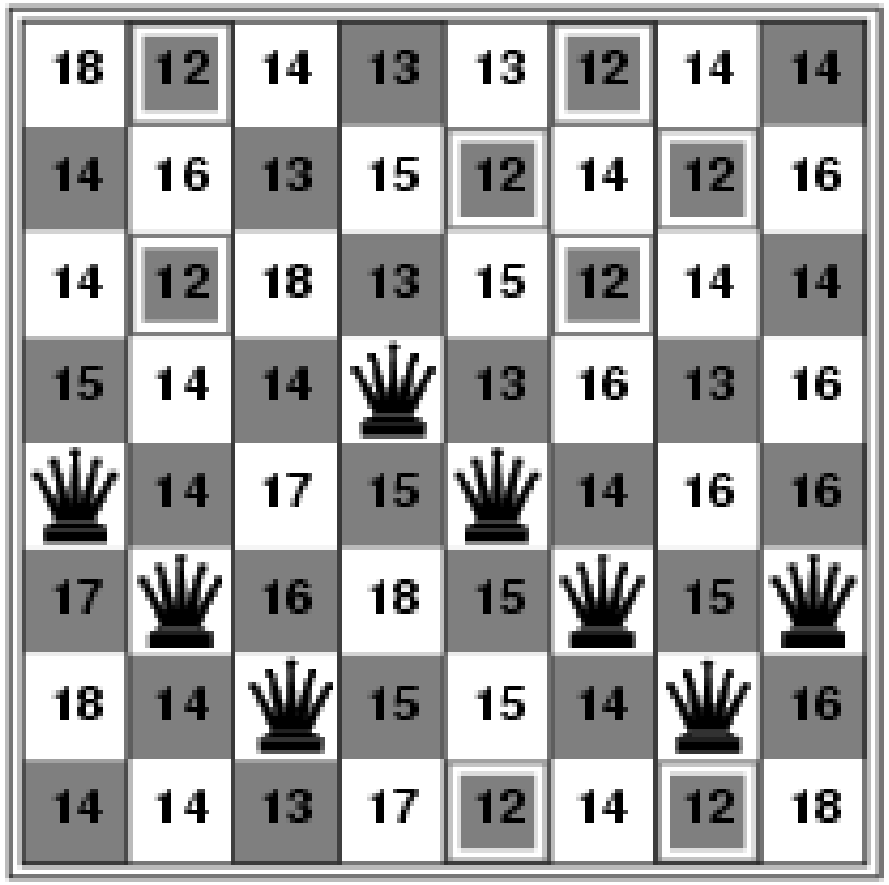
Example of heuristic functions

- **Problem:** To get from square **A3** to square **E2**, one step at a time, avoiding obstacles (black squares).
- **Operators:** (in order)
  - `go_left`(n)
  - `go_down`(n)
  - `go_right`(n)
  - each operator costs 1.
- **Heuristic** = ? → Manhattan distance

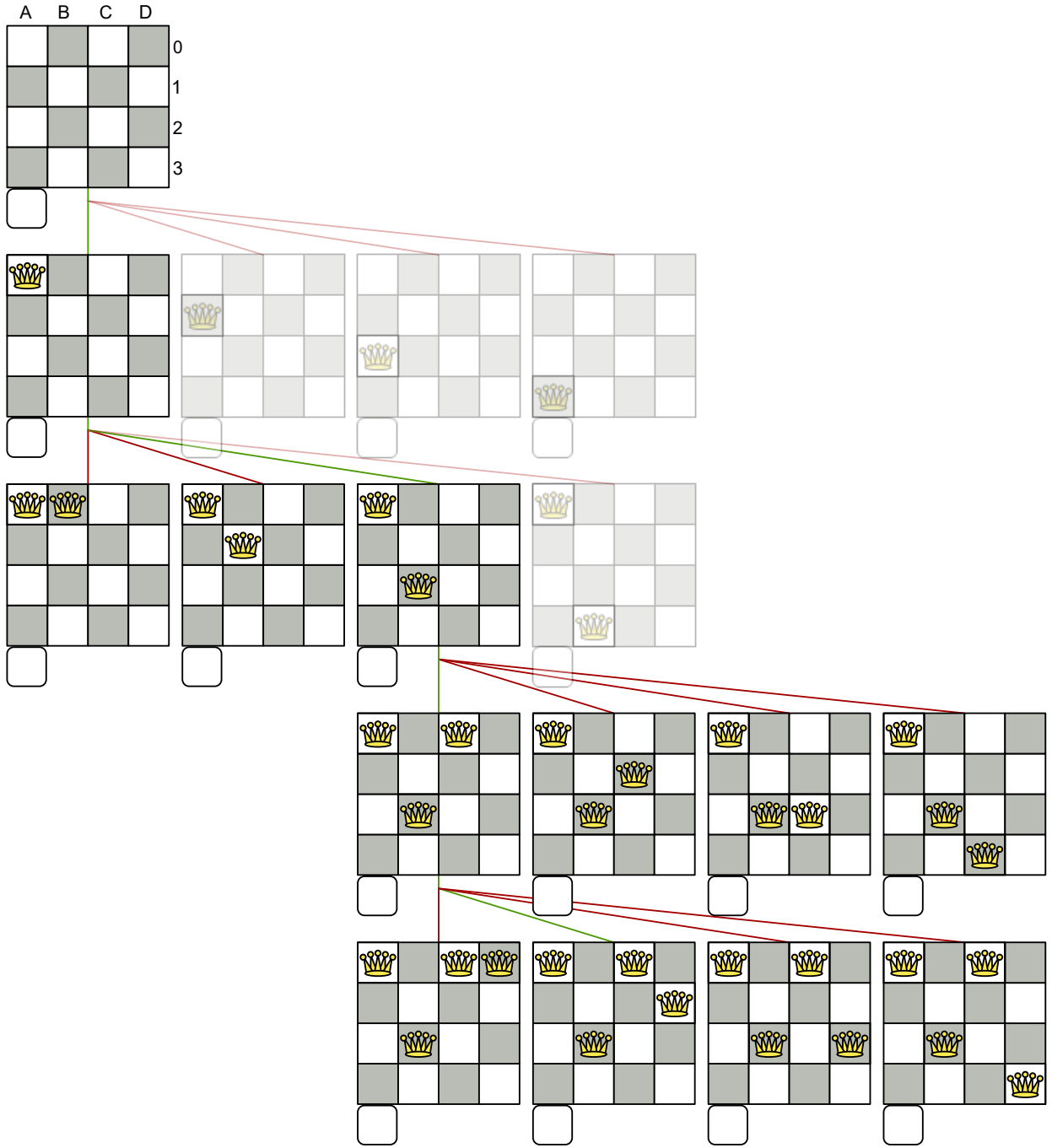


Example of heuristic functions

- Heuristic function = number of pairs of queens that are attacking each other, either directly or indirectly



4-queens: compute the values of heuristics function



Best First Search

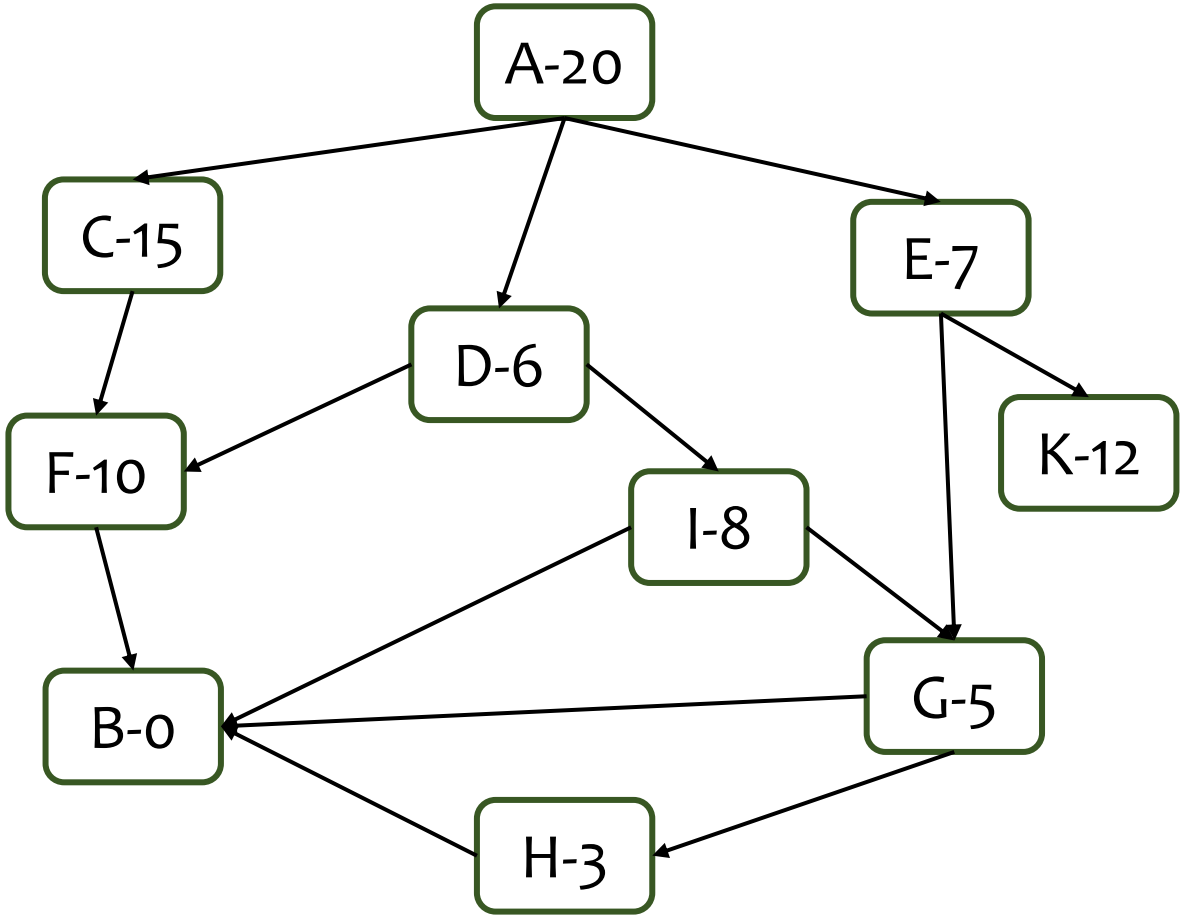
- Breath first search with the guidance of heuristic function.
- Select the state which has the smallest heuristic function value.

Procedure BestFirstSearch
begin
1. Initialize: L = {Start_state}
2. loop do
    2.1. if L = ∅ then
        {searching fail; stop};
    2.2. u ← L.getFirst();
    2.3. if u is GOAL then
        {searching success; stop};
    2.4. for all neighbors v of u do
        Insert v into L.
        Sort L in ascending order by the heuristic function values.

end;

Best First Search – Example 1

- Start state: A
- Goal state = ?



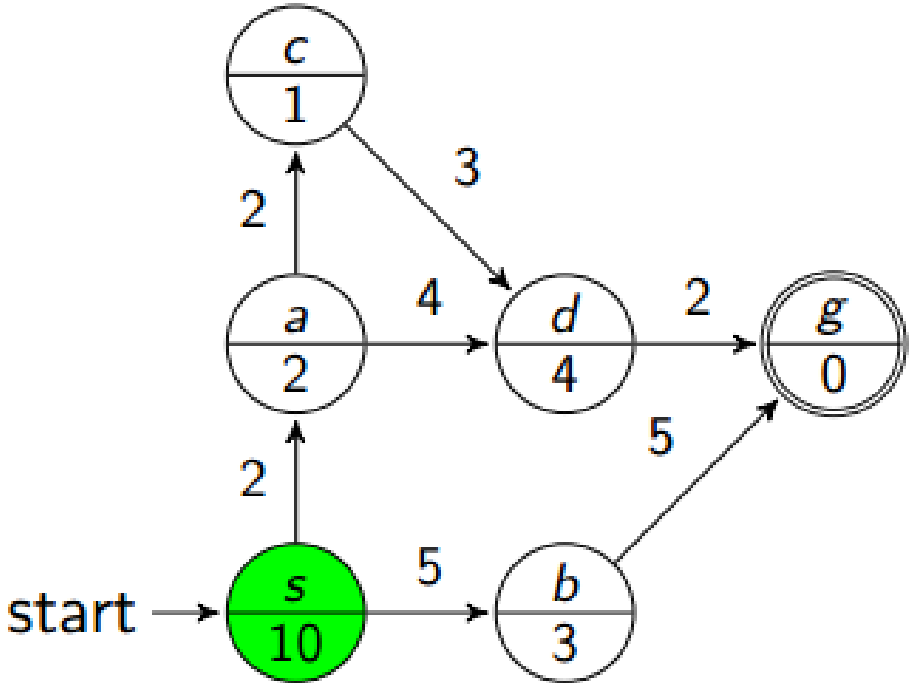
Steps of Best First Search

- Intialization: L = {A}

u	Neighbors of u	L

Best First Search – Example 2

- Start state: s
- Goal state = ?

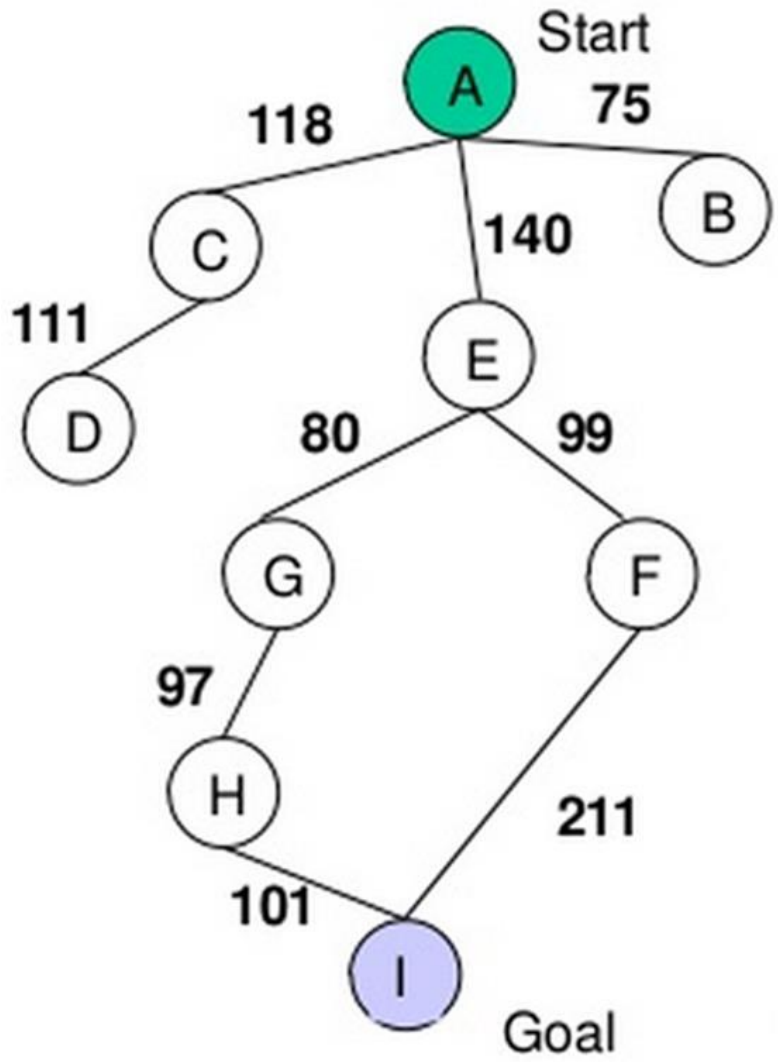


Steps of Best First Search

- Intialization: L = {s}

u	Neighbors of u	L

Best First Search – Example 3



State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0

***h* (*n*) = straight-line distance heuristic**

Best First Search – Example 4

- State = position of cells
- Operators = movements of blank cell
  - Including: Up, Left, Right, Down
- Heuristic function = Manhattan distance

- Start state:

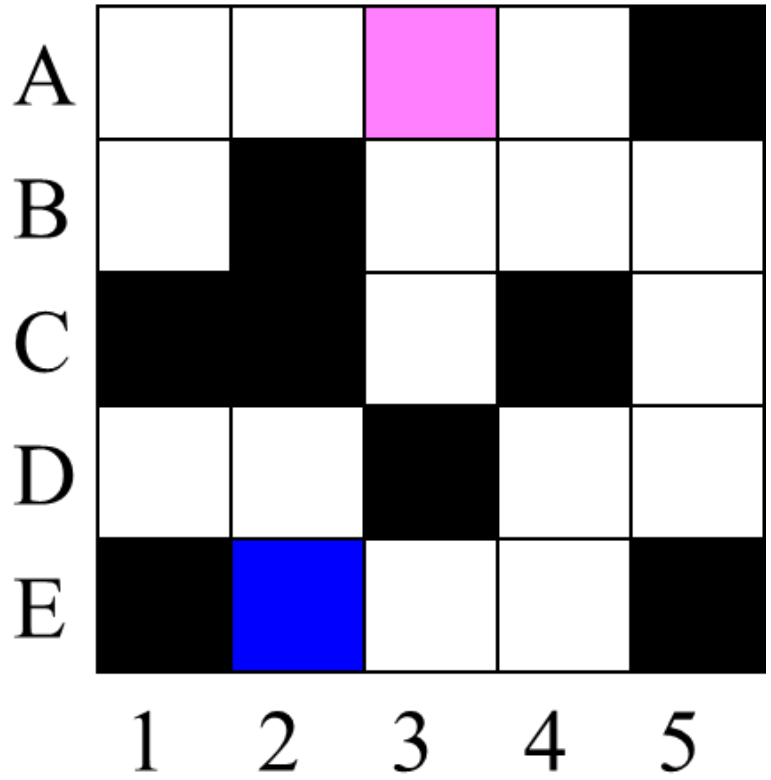
1	2	5
3	4	
6	7	8

- Goal state:

	1	2
3	4	5
6	7	8

Best First Search – Example 5

- **Problem:** To get from square **A3** to square **E2**, one step at a time, avoiding obstacles (black squares).
- **Operators:** (in order)
  - `go_left`(n)
  - `go_down`(n)
  - `go_right`(n)
  - each operator costs 1.
- **Heuristic function = Manhattan distance**



Hill Climbing Search

- Depth first search with the guidance of heuristic function.

Procedure HillClimbing

begin

1. Initialization: L = {start_state}
2. loop do

    2.1 if L = ∅ then

        {searching fail; stop};

    2.2 u ← L.getFirst();

    2.3 if u is GOAL then

        {searching success; stop};

    2.4 for all neighbors v of u do

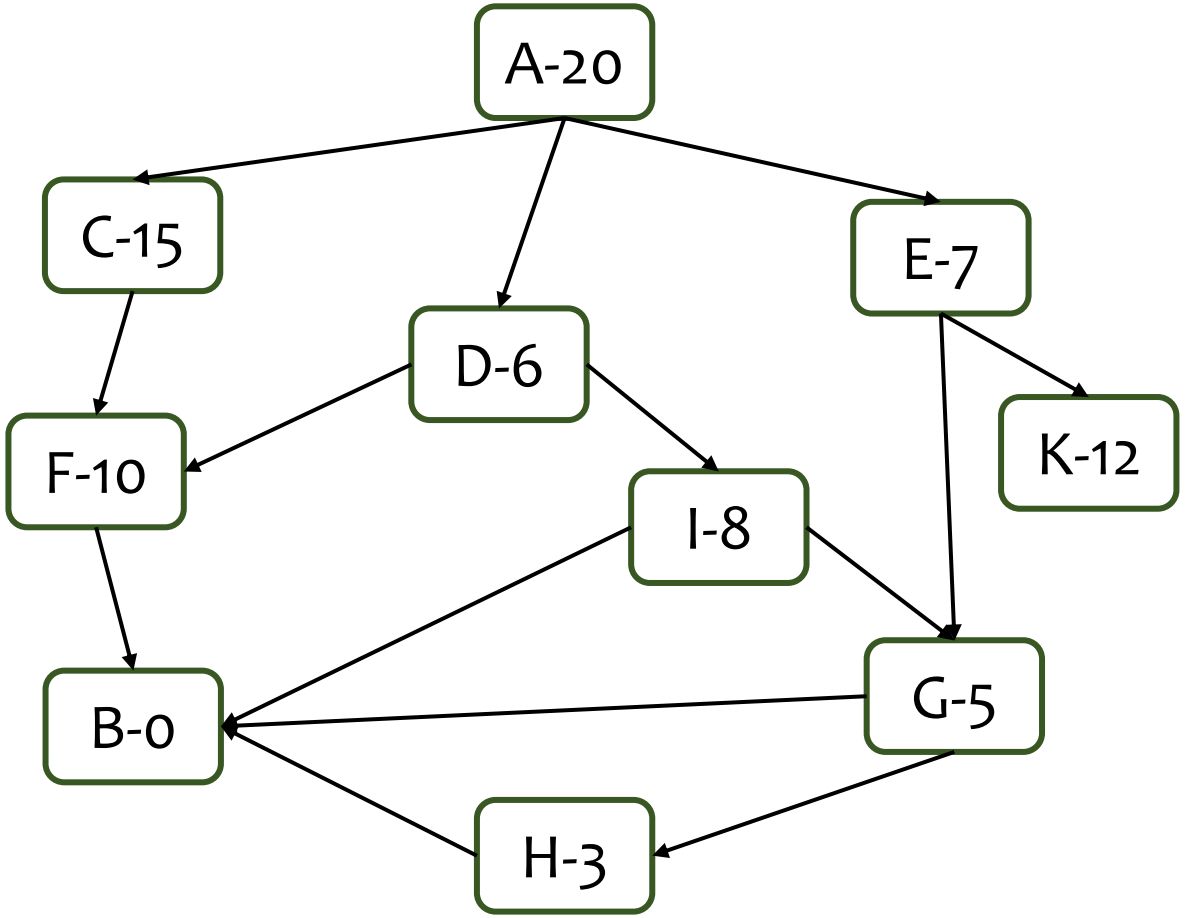
        Insert v to L1; sort L1 in ascending order by the heuristic function values.

    2.5 Insert L1 into the beginning of L

end;

Hill Climbing Search – Example 1

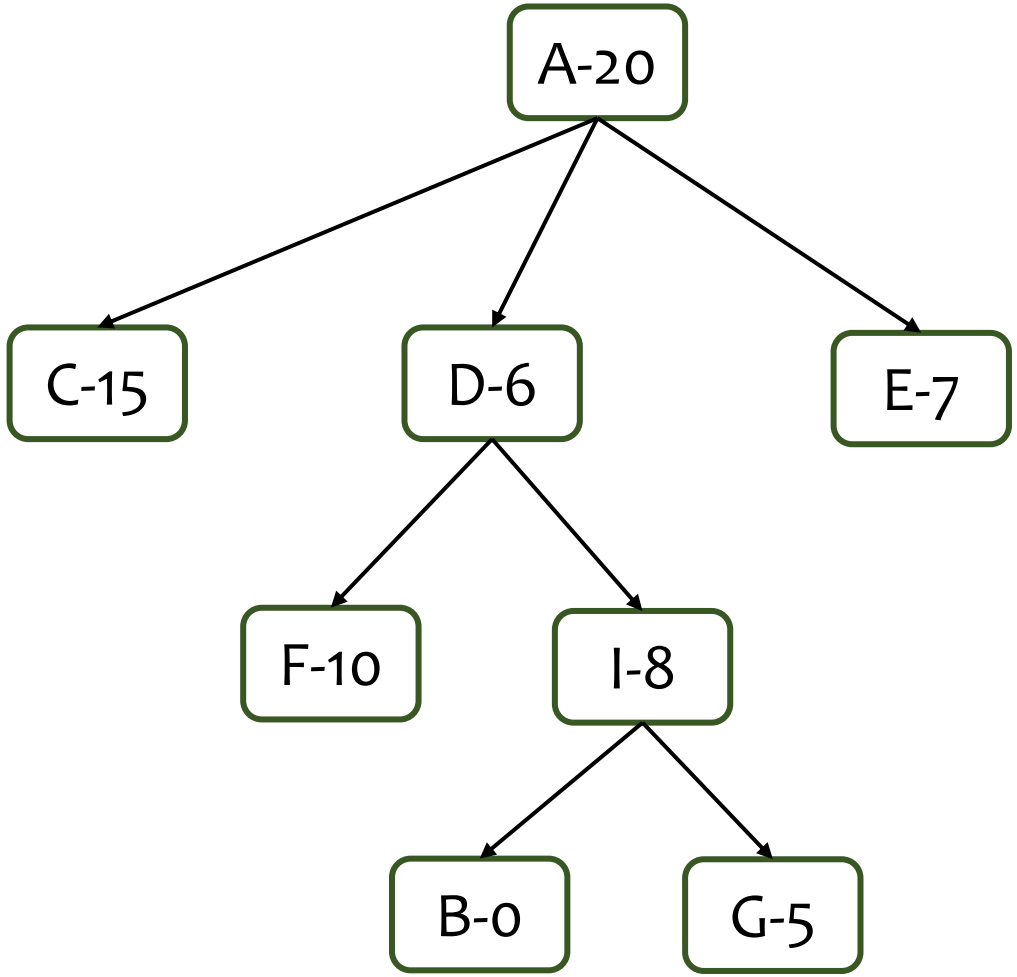
- Start state: A
- Goal state = ?



L = {A}

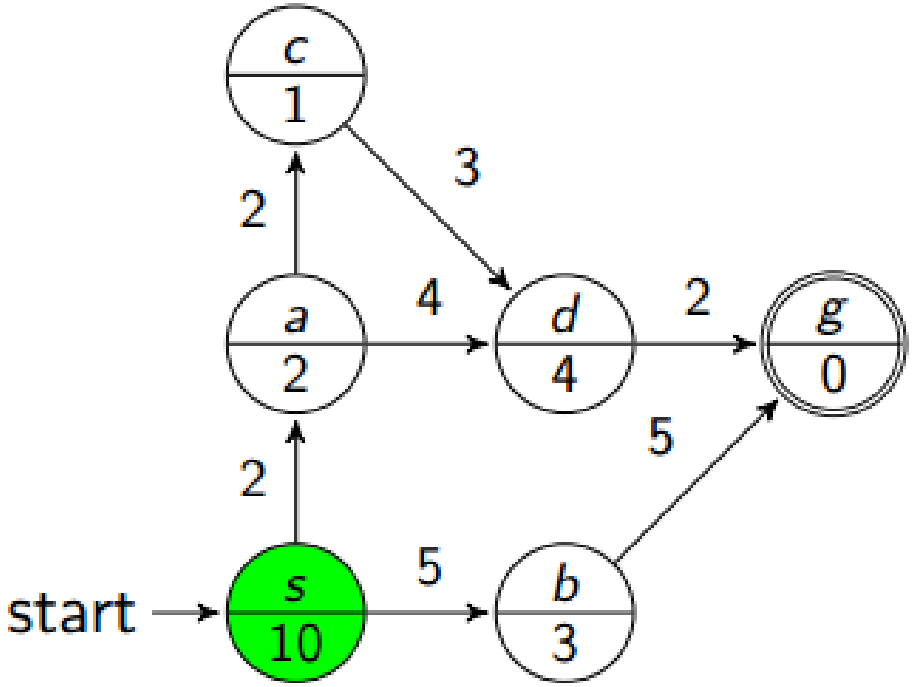
u	Neighbors of u	L

Search tree generated



Hill Climbing Search – Example 2

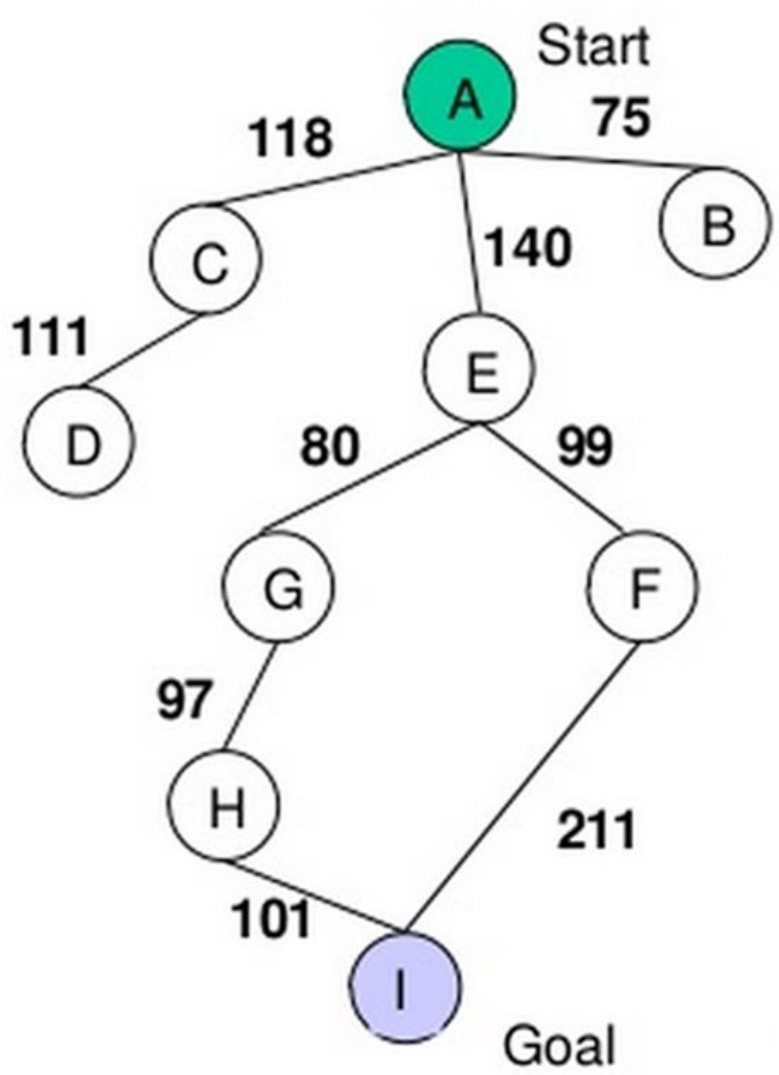
- Start state: s
- Goal state = ?



L = {s}

u	Neighbors of u	L

Hill Climbing Search – Example 3



State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0

***h* (*n*) = straight-line distance heuristic**

Hill Climbing Search – Example 4

- State = position of cells
- Operators = movements of blank cell
  - Including: Up, Left, Right, Down
- Heuristic function = Manhattan distance

- Start state:

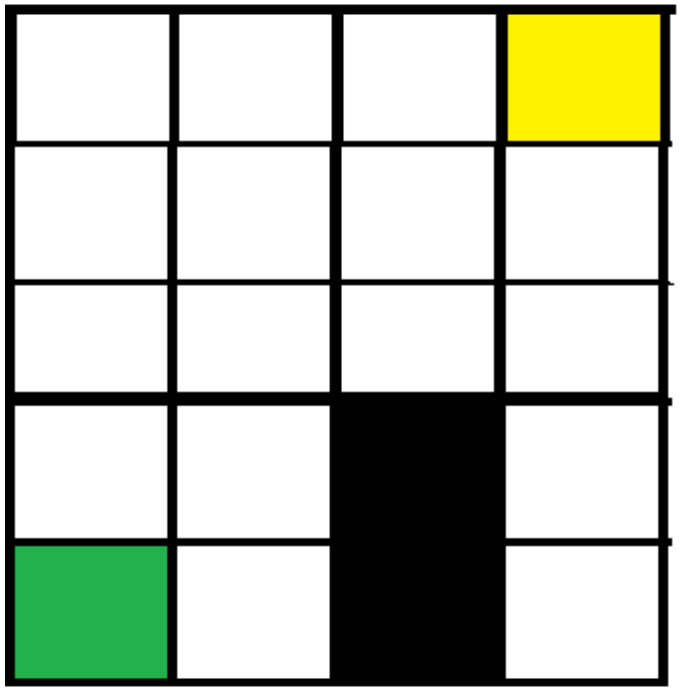
1	2	5
3	4	
6	7	8

- Goal state:

	1	2
3	4	5
6	7	8

Hill Climbing Search – Example 5

- **Problem**: To get from the green square to the yellow square, one step at a time, avoiding obstacles (black squares).
- **Operators**: The robot can move in 8 directions.
- **Heuristic function = Manhattan distance or Euclidean distance.**



OPTIMAL SEARCH

Searching Algorithms

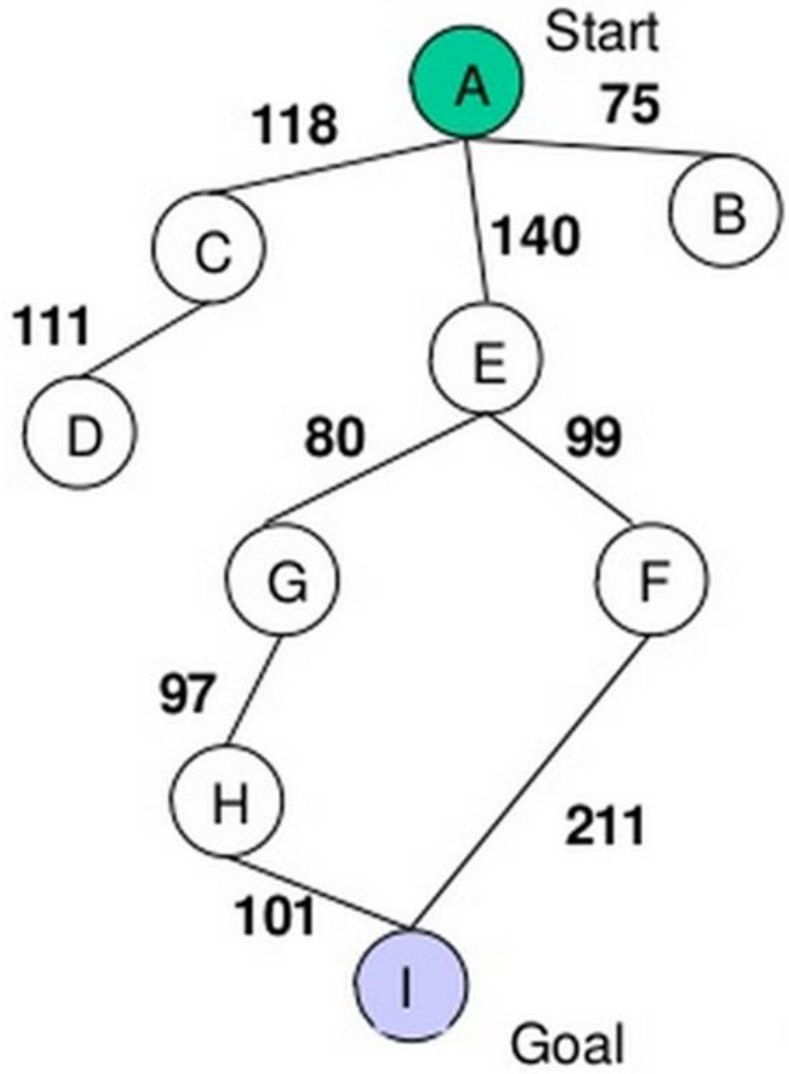
- Blind Search
  - Breadth First Search
  - Depth First Search
  - Limited Depth First Search
  - Iterative Depth First Search
- Heuristic Search
  - Best First Search
  - Hill Climbing
- Optimal Search
  - A^T Search
  - A^{KT} Search
  - A* Search
  - Branch and Bound Search
- Adversarial Search
  - Minimax
  - Alpha-beta prunning

Shortest Path Finding

- Given a state space.
- Assumption: cost to change from u to v = k(u,v)>=0.
- Find the shortest path from the start state to the goal state, so that the total cost is minimal.
- Examples?

Shortest Path Finding

- Solution
  - Before?
  - Now:
    - A*
    - Branch and bound



A* Search

- Best first search with a heuristic function, namely f(u)
- f(u) = g(u) + h(u)
  - g(u) = cost of the shortest path from the start state to u.
  - h(u) = minimal estimated cost to move from u to the goal state.

Procedure A*

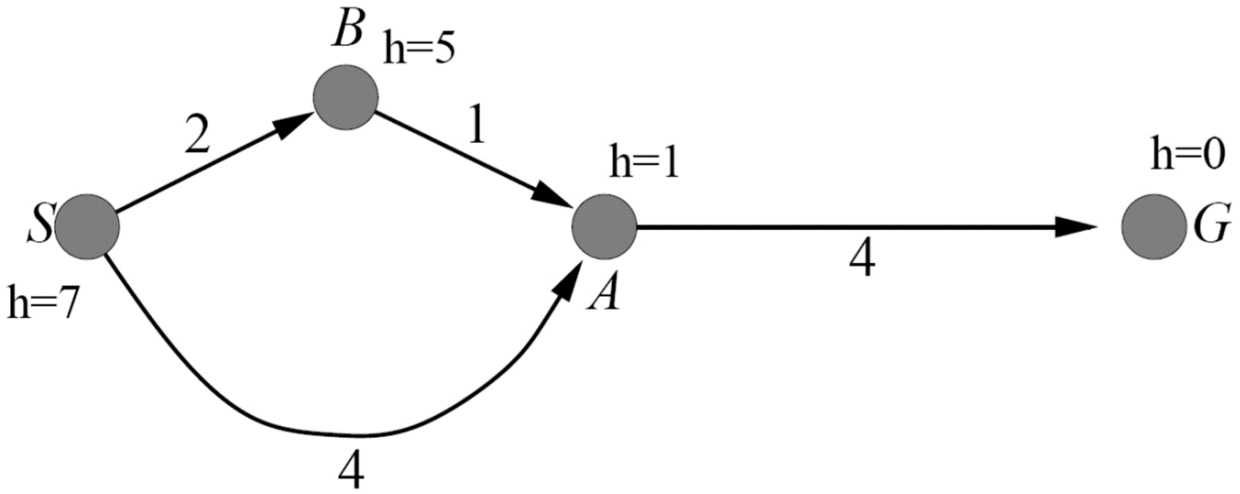
begin

1. Initialize: L = {Start_state}
2. loop do
   2.1. if L = Ø then
   {searching fail; stop};
   2.2. u ← L.getFirst();
   2.3. if u is GOAL then
   {searching success; stop};
   2.4 for all neighbors v of u do
   g(v) = g(u) + k(u,v);
   f(v) = g(v) + h(v);
   Insert v into L; Sort L in ascending order by the values of f;

end;

A* – Example 1

- Start state: S
- Goal state = ?



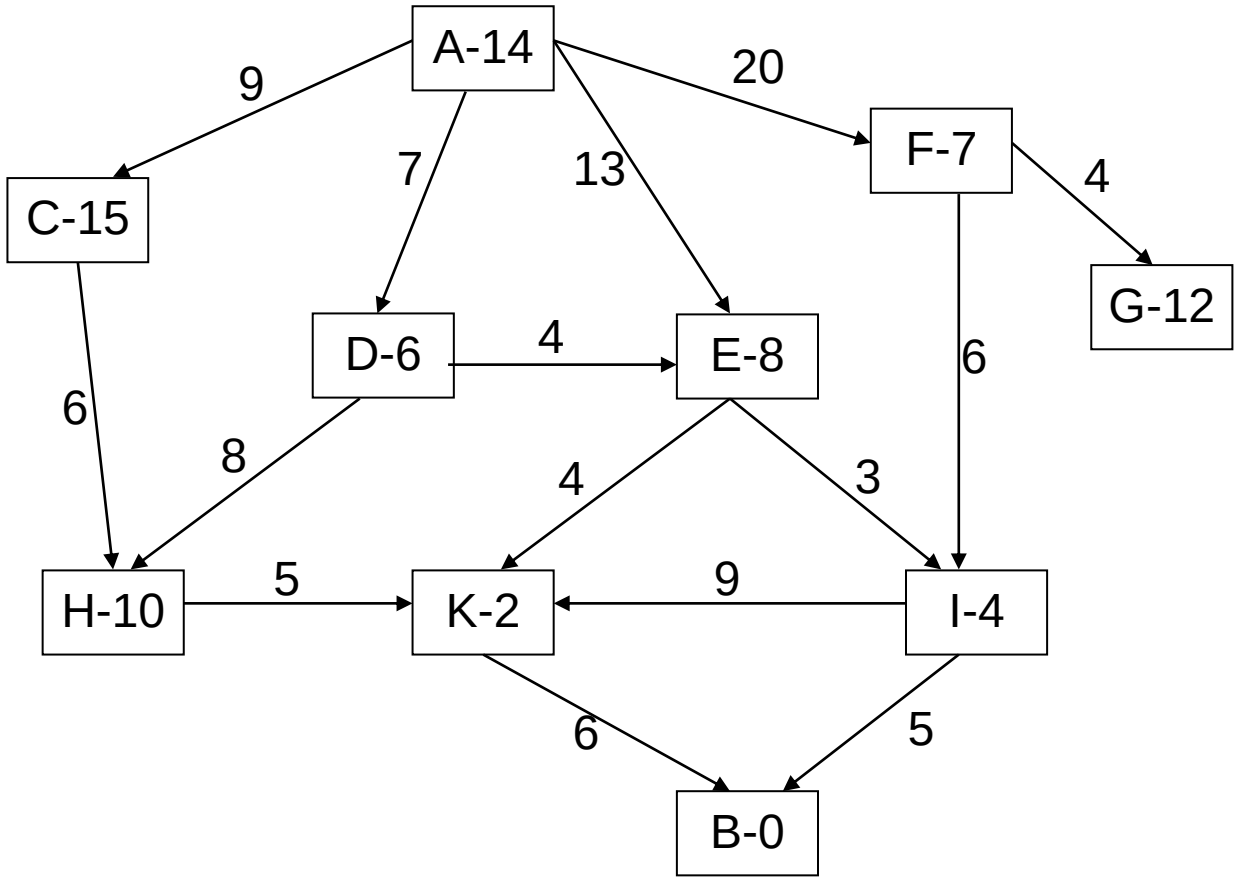
Steps of A*

- Initialization: L = {S}

u	Neighbors of u	L

A* – Example 2

- Start state: A
- Goal state = ?



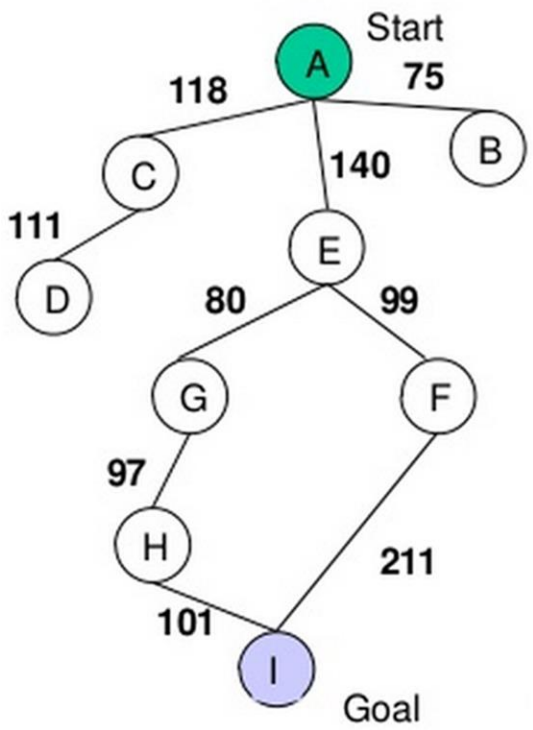
Steps of A*

- Initialization: L = {A}

u	Neighbors of u	L

A* – Example 3

- Start state: A
- Goal state = ?



State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0

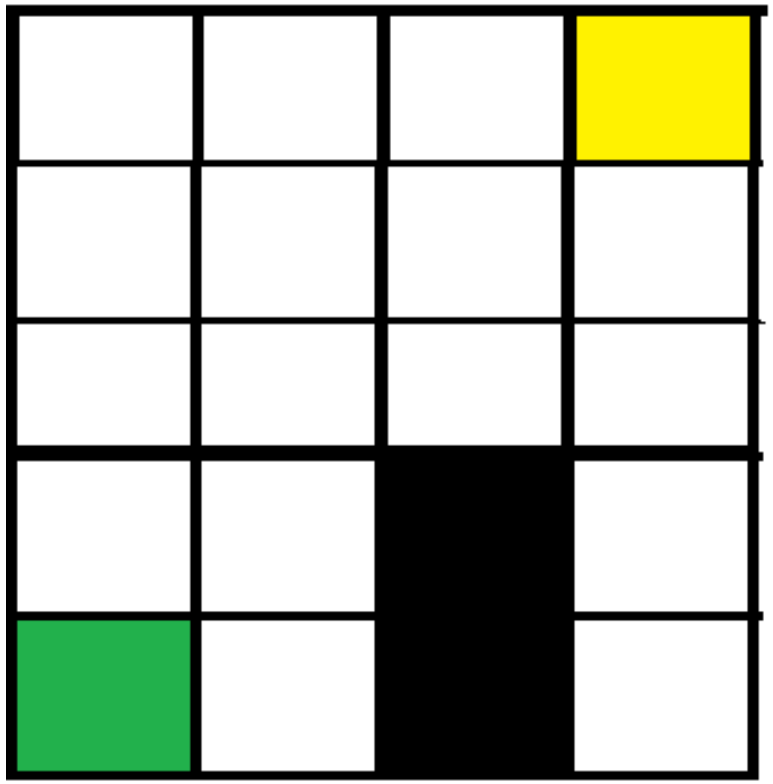
Steps of A*

- Initialization: L = {A}

u	Neighbors of u	L

A*– Example 4

- **Problem:** To get from the green cell to the yellow cell, one step at a time, avoiding obstacles (black squares).
- **Operators:** The robot can move in 8 directions.
- **g(n) = the cost that has been accrued in reaching the cell.**
  - **Cost to move from cell A to cell B = Euclidean_distance(A, B)**
- **h(n) = Manhattan_distance(n, goal_cell)**
- **f(n) = ?**



Branch and Bound Search

- Hill climbing search with a heuristic function, namely f(u)
- f(u) = g(u) + h(u)
  - g(u) = cost of the shortest path from the start state to u.
  - h(u) = minimal estimated cost to move from u to the goal state.

Procedure BranchAndBound

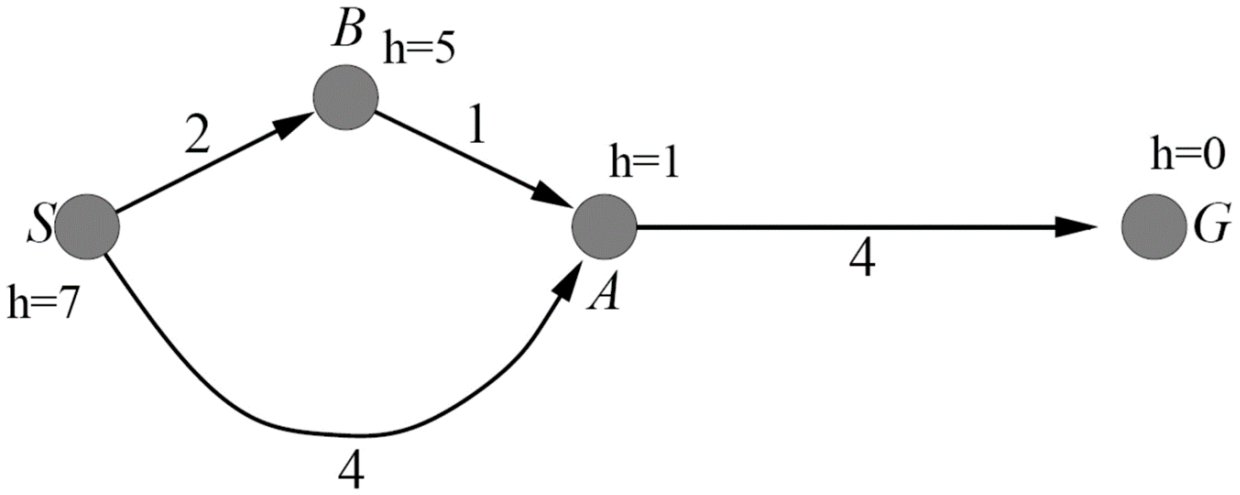
begin

1. Initialize: L = {start_state}; cost = ∞
2. loop do
   - 2.1 if L = ∅ then {searching fail; stop};
   - 2.2 u ← L.getFirst();
   - 2.3 if u is GOAL then
     - if g(u)<=cost then {cost = g(u); go to 2.1}
   - 2.4. if f(u)>cost then go to 2.1
   - 2.5 for all neighbors v of u do
     - {g(v) = g(u) + k(u,v); f(v) = g(v) + h(v);
     - Insert v into L1; sort L1 ascending by the values of f};
   - 2.6 Insert L into the beginning of L

end;

Branch and Bound– Example 1

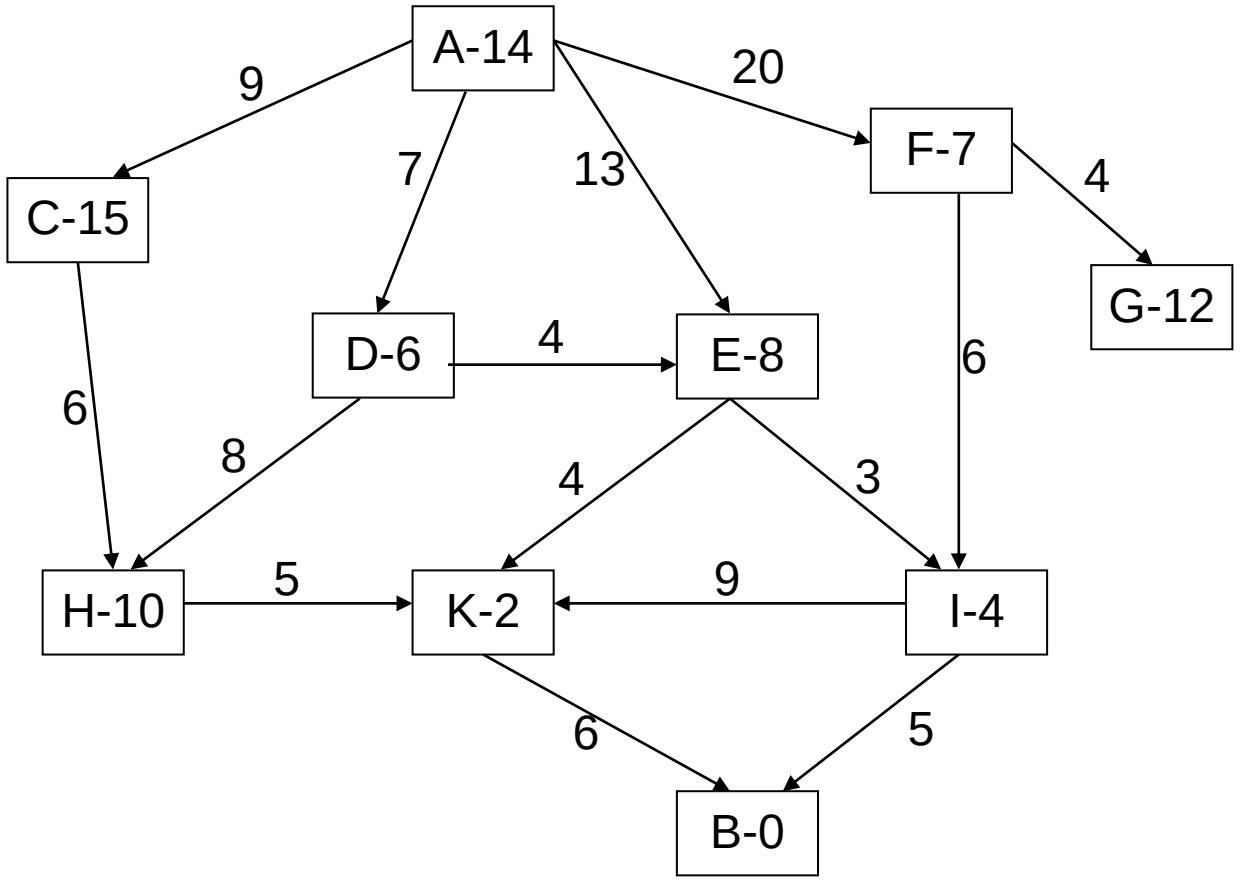
- Start state: S
- Goal state = ?



u	Neighbors of u	L

Branch and Bound– Example 2

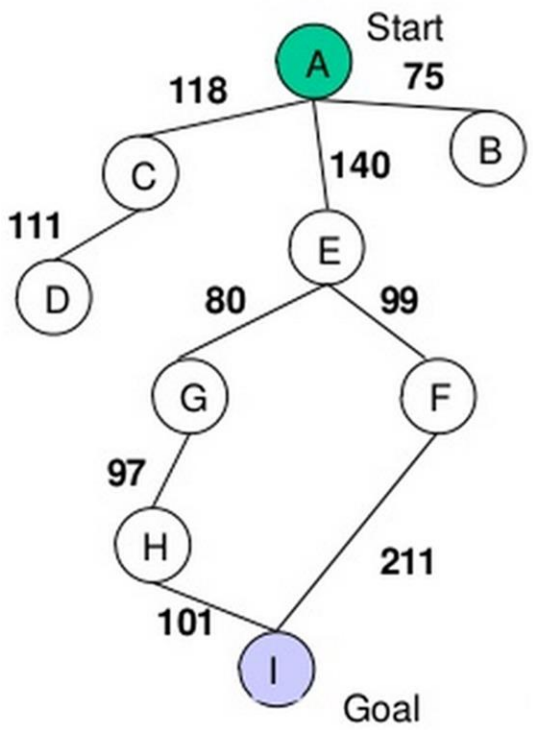
- Start state: A
- Goal state = ?



u	Neighbors of u	L

Branch and Bound– Example 3

- Start state: A
- Goal state = ?



State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0

u	Neighbors of u	L

Branch and Bound– Example 4

- **Problem:** To get from the green cell to the yellow cell, one step at a time, avoiding obstacles (black squares).
- **Operators:** The robot can move in 8 directions.
- **g(n) = the cost that has been accrued in reaching the cell.**
  - **Cost to move from cell A to cell B = Euclidean_distance(A, B)**
- **h(n) = Manhattan_distance(n, goal_cell)**
- **f(n) = ?**