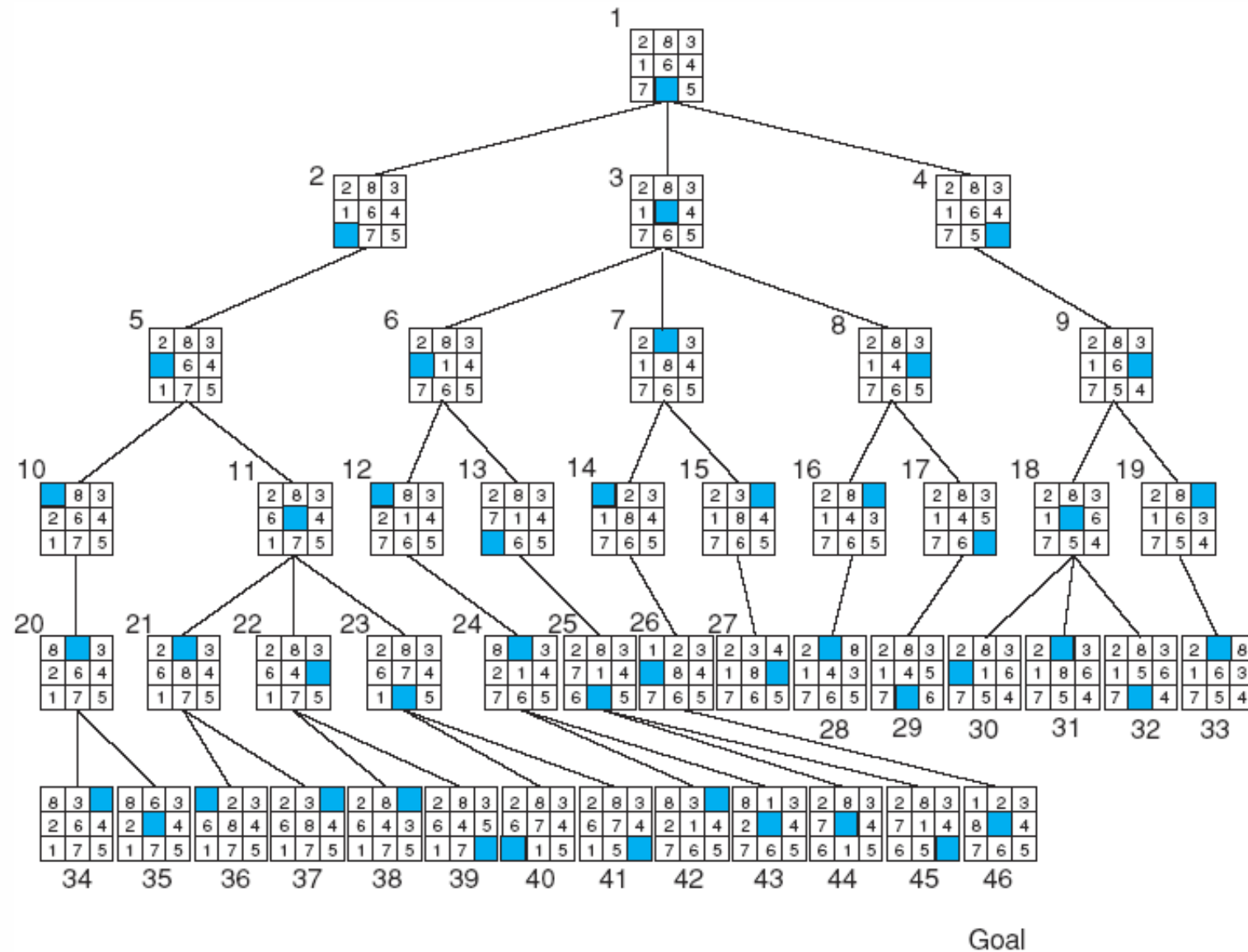# Informed Search

Heuristics Search

Recap: Search

- Search problem:
  - States
  - Actions and costs
  - Start state and goal test

- Search algorithm:
  - Systematically builds a search tree
  - Chooses an ordering of the fringe (unexplored nodes)
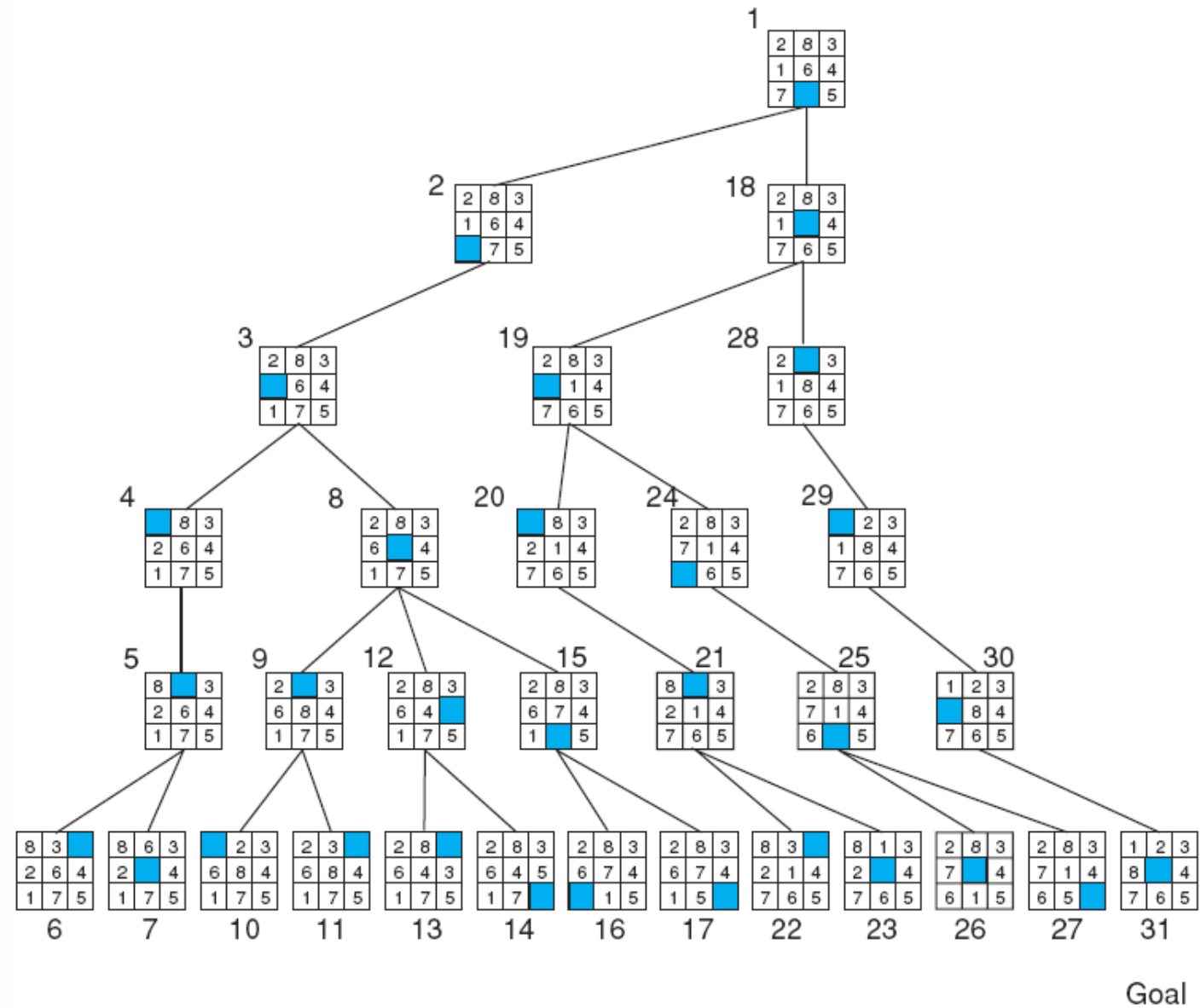 - Optimal: finds least-cost plans

Today

- Informed Search
  - Heuristics
  - Greedy Search
  - A* Search

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

Goal

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

Goal

Using problem specific knowledge to aid searching

- With knowledge, one can search the state space as if he was given "hints" when exploring a tree.
  - Heuristic information in search = Hints

- Leads to dramatic speed up in efficiency.

Search only in this subtree!!
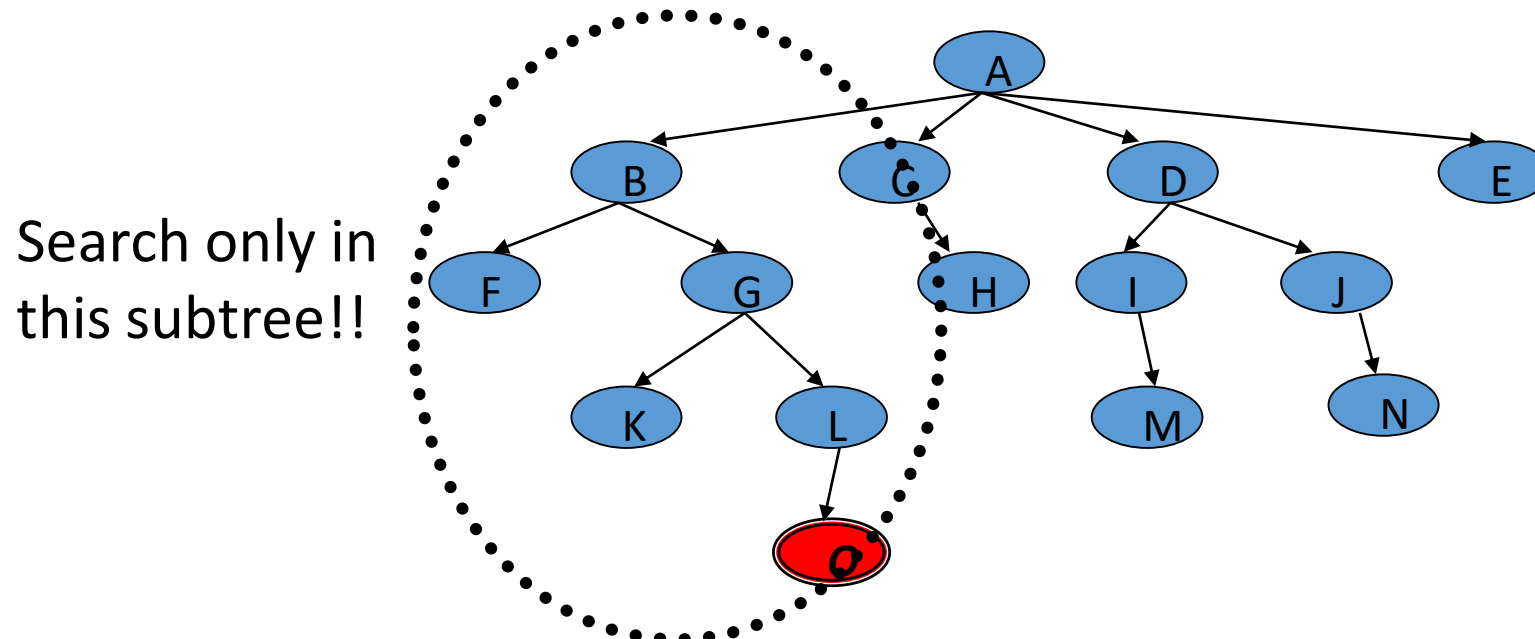
What are Heuristics?

- Heuristics are know-hows obtained through a lot of experiences.
- Heuristics often enable us to make decisions quickly without thinking deeply about the reasons.
- The more experiences we have, the better the heuristics will be.

When Heuristics are used?

AI problem solvers employ heuristics in two basic situations:

1. A **problem may not have an exact solution** because of inherent ambiguities in the problem statement or available data.

   - Medical diagnosis: Given symptoms may have several possible causes;
   - Doctors use heuristics to choose the most likely diagnosis and formulate a treatment plan.

2. A problem may have an exact solution, but the **computational cost of finding it may be prohibitive**.

   - Chess: state space growth is combinatorically explosive, with the number of possible states increasing with the depth of the search.
   - Heuristics attack this complexity by guiding the search along the most "promising" path through the space.

Why heuristic functions work?

- **Uninformed Search** : at most b choices at each node and a depth of d at the goal node
  - In the worst case, search around $O(b^d)$ nodes before finding a solution (Exponential Time Complexity).

- **Heuristics** improve the efficiency of search algorithms by reducing the effective branching factor from b to (ideally) a lower constant b^* such that $1 \leq b^* \ll b$

- Guide search ***towards the goal*** instead of ***all over the place***

Start   Goal

Informed

Start   Goal

Uninformed
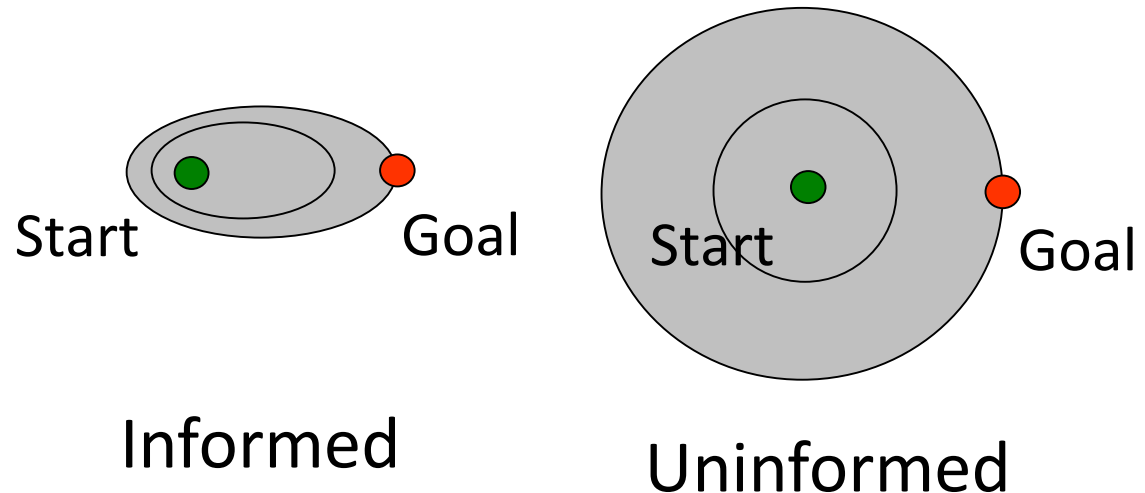
Evaluation / Heuristic Function

- Evaluation function *f(n) = g(n) + h(n)*
  - *g(n)* = exact cost so far to reach n
  - *h(n)* = estimated cost to goal from n
  - *f(n)* = estimated total cost of cheapest path through n to goal

- Special cases:
  - Uniform Cost Search: $f(n) = g(n)$
  - Greedy (best-first) Search: $f(n) = h(n)$
 - A* Search: $f(n) = g(n) + h(n)$

How it works?

- Idea: use heuristic function for each node
  - that estimates desirability
- Expands most desirable unexpanded node
- **Implementation:**
  - fringe is a queue sorted in order of desirability
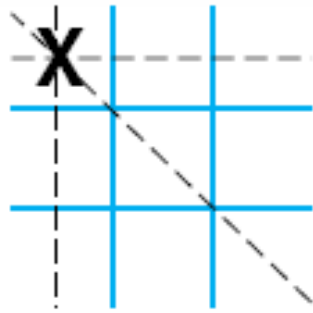
Heuristic Search

- A heuristic is:
  - A function that *estimates* how close a state is to a goal
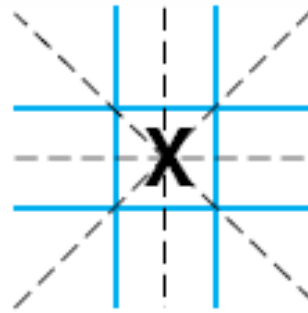 - Designed for a particular search problem

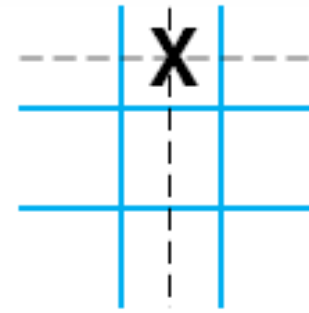First three levels of the tic-tac-toe state space reduced by symmetry
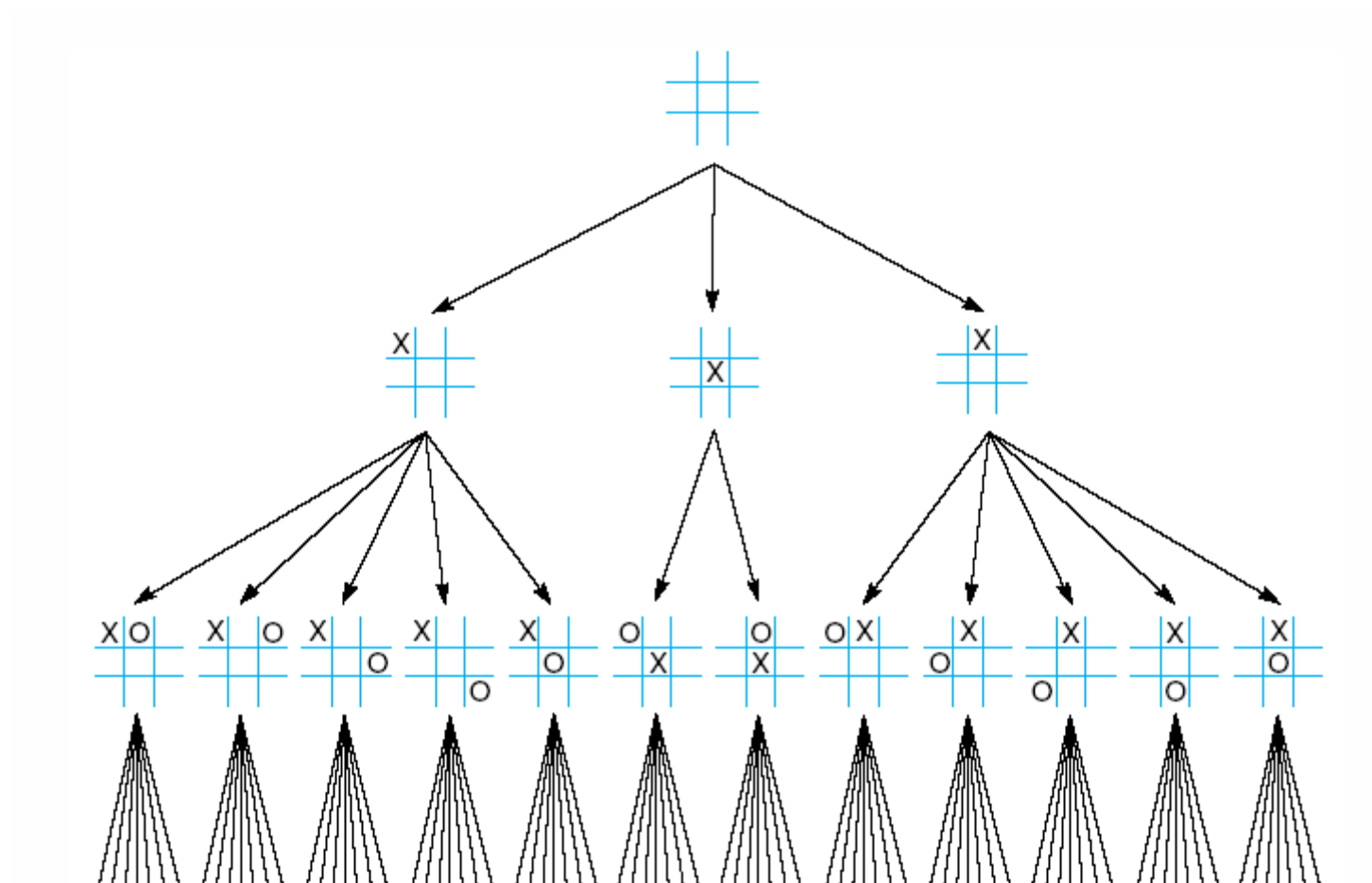
The "most wins" heuristic applied to the first children in tic-tac-toe.
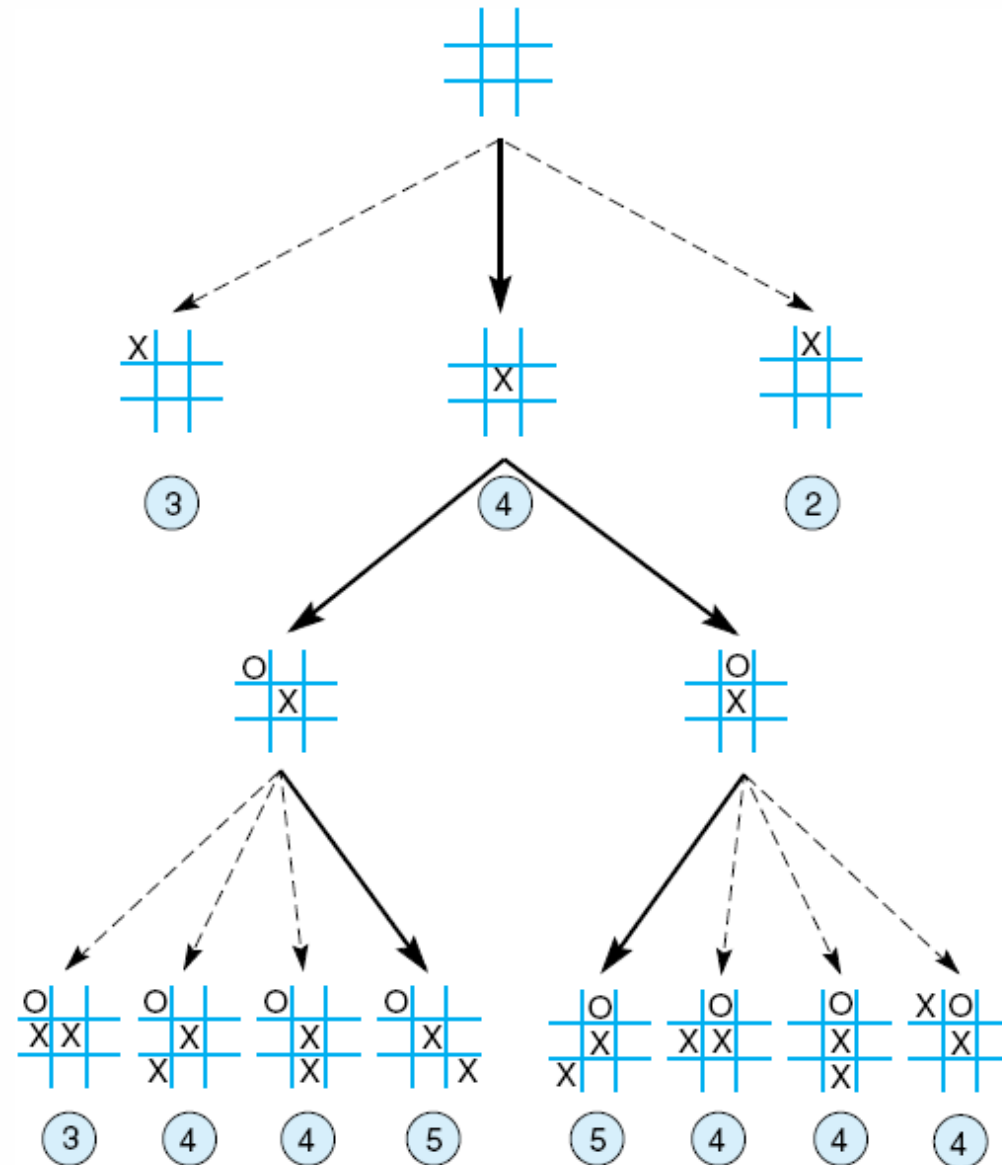

| Three wins through a corner square | Four wins through the center square | Two wins through a side square |

Heuristically reduced state space for tic-tac-toe.

8–Puzzle: Heuristics?

Start State   Goal State

- h1: The number of misplaced tiles.
- h2: The sum of the distances of the tiles from their goal positions.

Current State

1	2	3
4	5	6
7		8

Goal State

1	2	3
4	5	6
7	8	

h1: The number of misplaced tiles (not including the blank)

1	2	3
4	5	6
7		8

Current State

1	2	3
4	5	6
7	8	

Goal State

- *Notation:* $f(n) = h(n)$
- *h(current state) = 1*
- Because this state has one misplaced tile.

- Only "**8**" is misplaced
- So the heuristic function evaluates to 1.
- The heuristic is *telling* us, that it *thinks* a solution might be available in just 1 more move.

h2: The sum of the distances of tiles from goal positions

Current State

3	2	8
4	5	6
7	1	

•The Manhattan Distance (not including the blank)

Goal State

1	2	3
4	5	6
7	8	

3	➡	3

2 spaces

	⬅	8
	⬇	
	8	

3 spaces

1	⬅	
	⬆	
	1	

3 spaces

Total 8

In this case, only the "3", "8" and "1" tiles are misplaced, by 2, 3, and 3 squares respectively, so the heuristic function evaluates to 8.
In other words, the heuristic is *telling* us, that it *thinks* a solution is available in just 8 more moves.

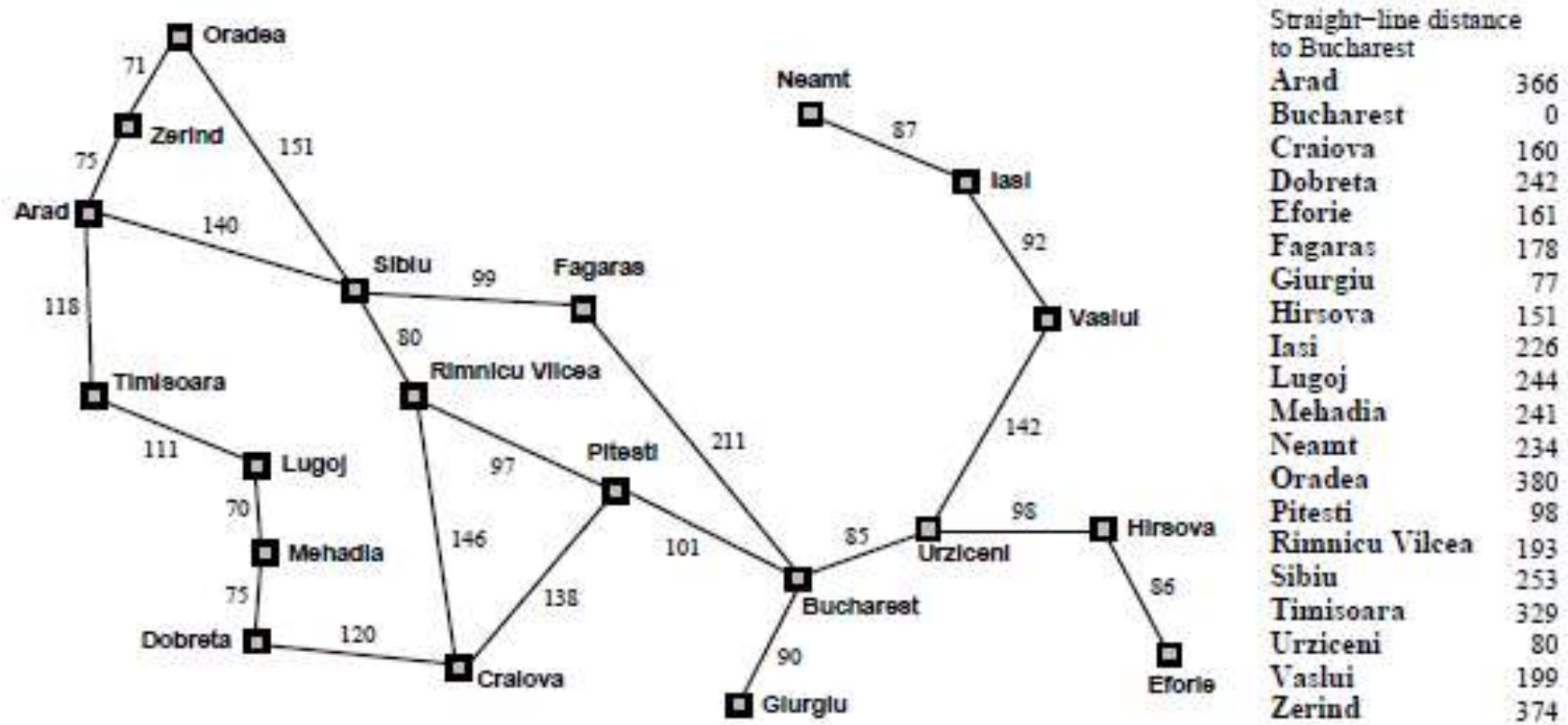*Notation: $f(n) = h(n)$,*   *h(current state) = 8*

Greedy Best First Search
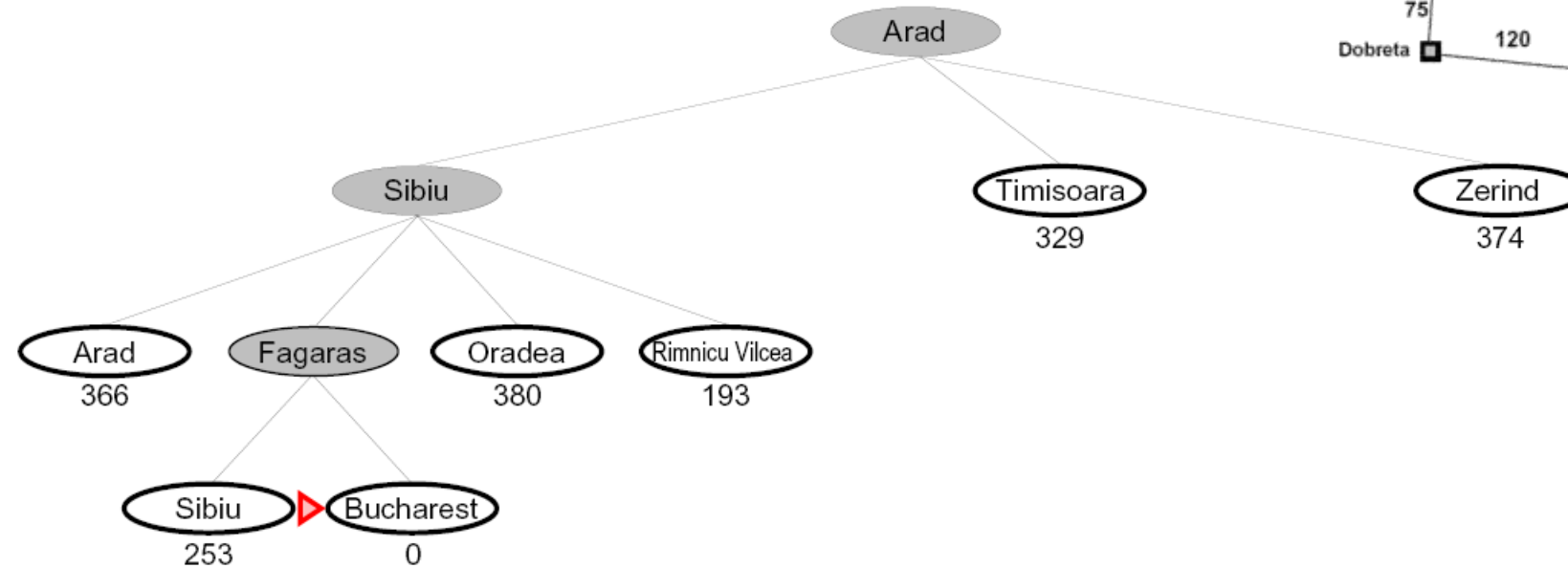
eval-fn: $f(n) = h(n)$

Greedy (Best-First) Search

- Idea: Expand node with the smallest estimated cost to reach the goal.

- Use heuristic function $f(n) = h(n)$

- **Best-first search** tries to expand the node that is closest to the goal, on the grounds that this is likely to lead to a solution quickly.
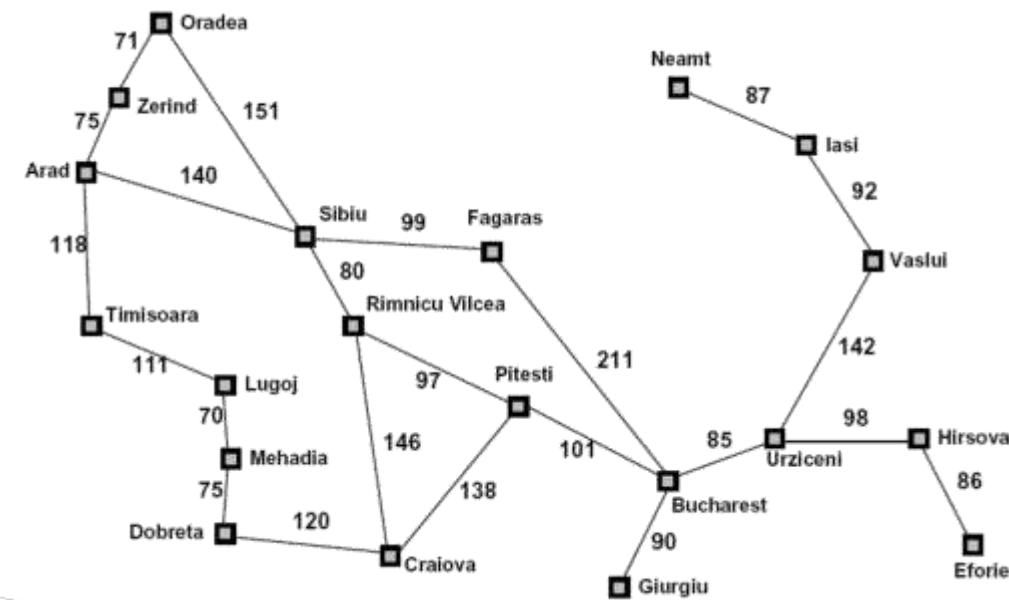
- *Choose path with low heuristic value*

Straight-line distance to Bucharest

City	Distance
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

h_{SLD} (n) = straight line distance between n and goal state, given coordinates of each city

Greedy Search

- Expand the node that seems closest…

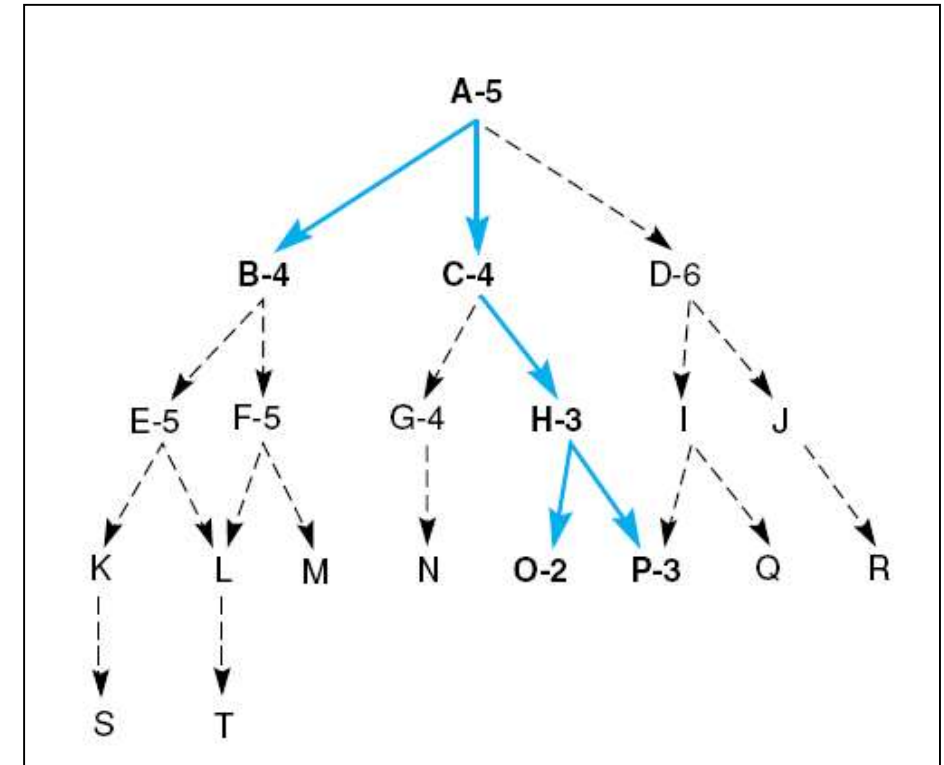- What can go wrong?

Greedy Search

- Expand the node that seems closest...(order frontier by h)
- What can possibly go wrong?
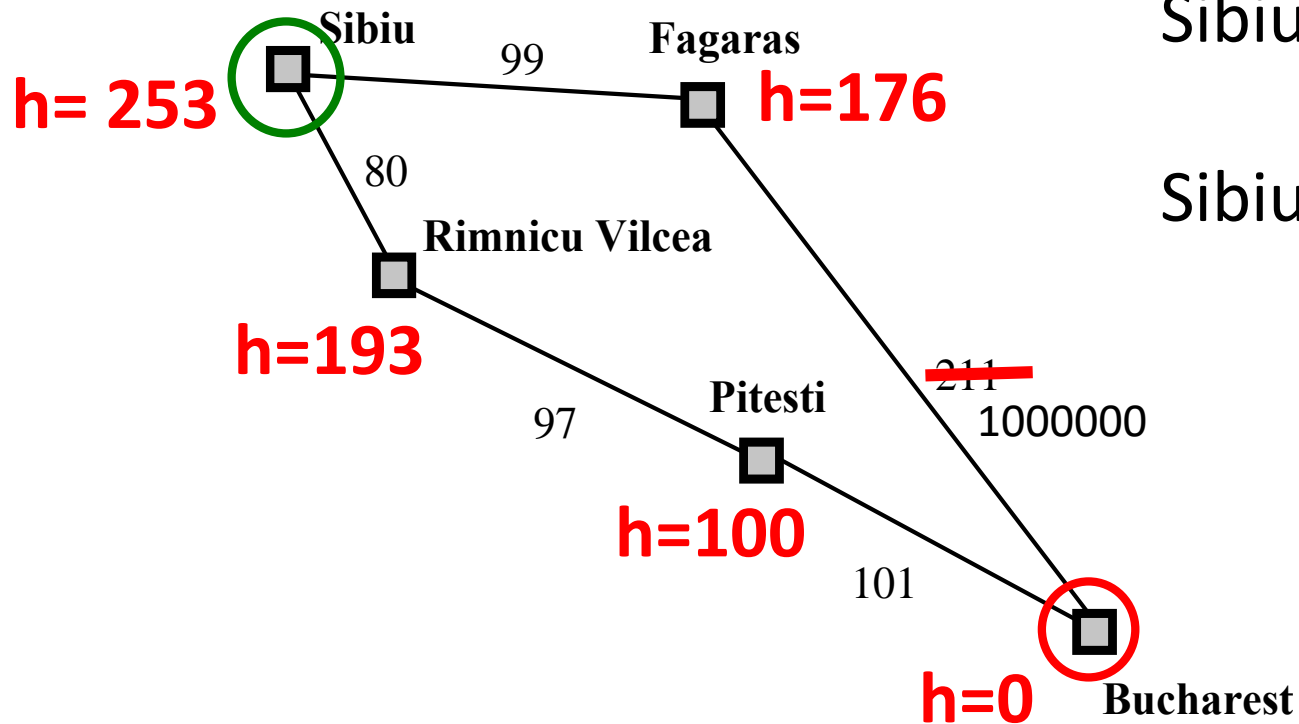
Sibiu-Fagaras-Bucharest =
99+211 = **310**

Sibiu-Rimnicu Vilcea-Pitesti-Bucharest =
80+97+101 = **278**

Best First Search

1. open = [A5]; closed = [ ]

2. evaluate A5; open = [B4,C4,D6]; closed = [A5]

3. evaluate B4; open = [C4,E5,F5,D6]; closed = [B4,A5]

4. evaluate C4; open = [H3,G4,E5,F5,D6]; closed = [C4,B4,A5]

5. evaluate H3; open = [O2,P3,G4,E5,F5,D6]; closed = [H3,C4,B4,A5]

6. evaluate O2; open = [P3,G4,E5,F5,D6]; closed = [O2,H3,C4,B4,A5]
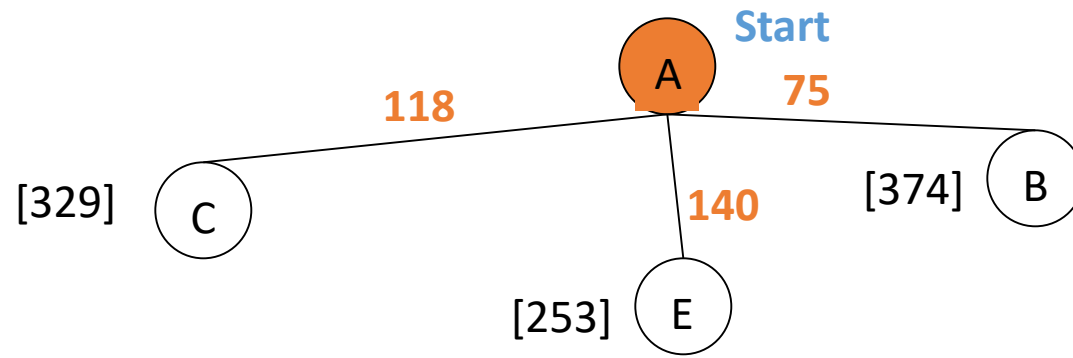
7. evaluate P3; the solution is found!

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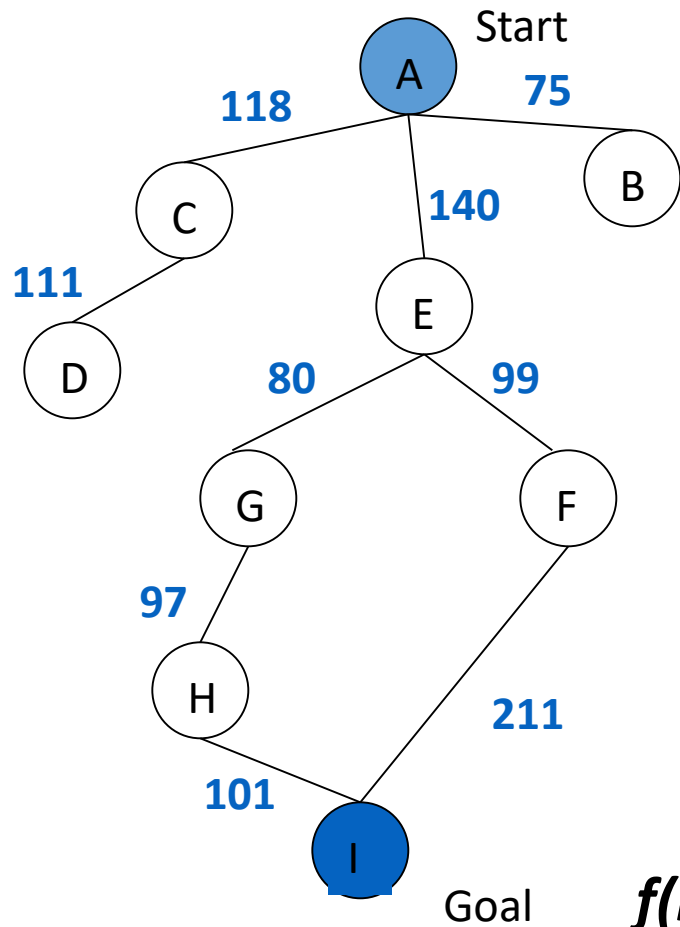
State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	**0**

$f(n) = h(n)$ = straight-line distance heuristic

Greedy Search: Tree Search
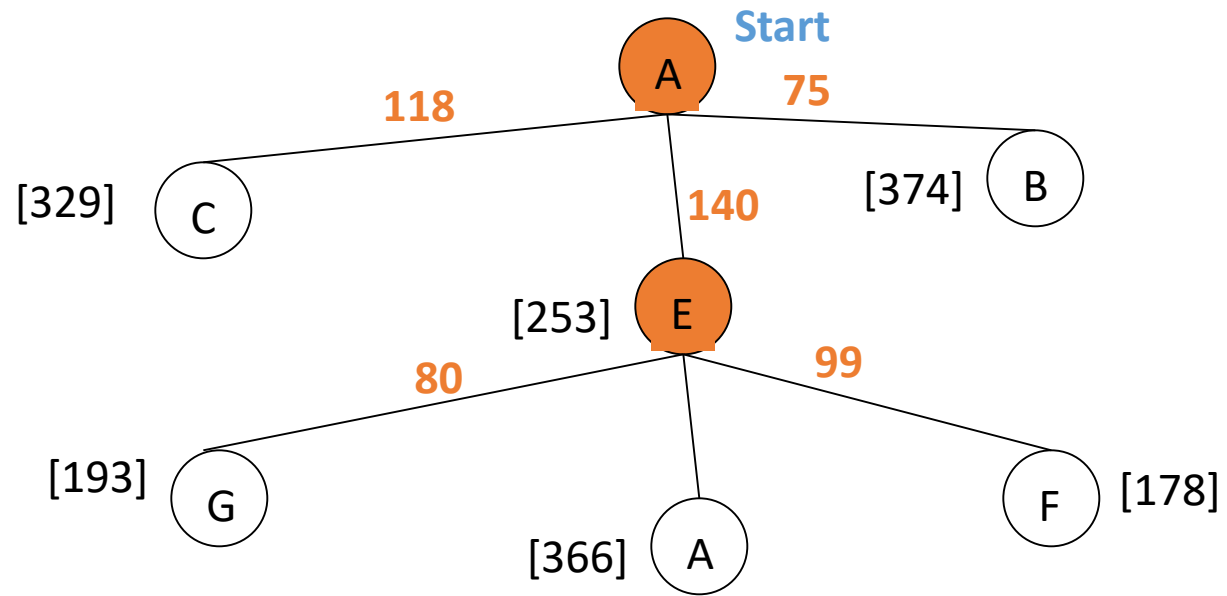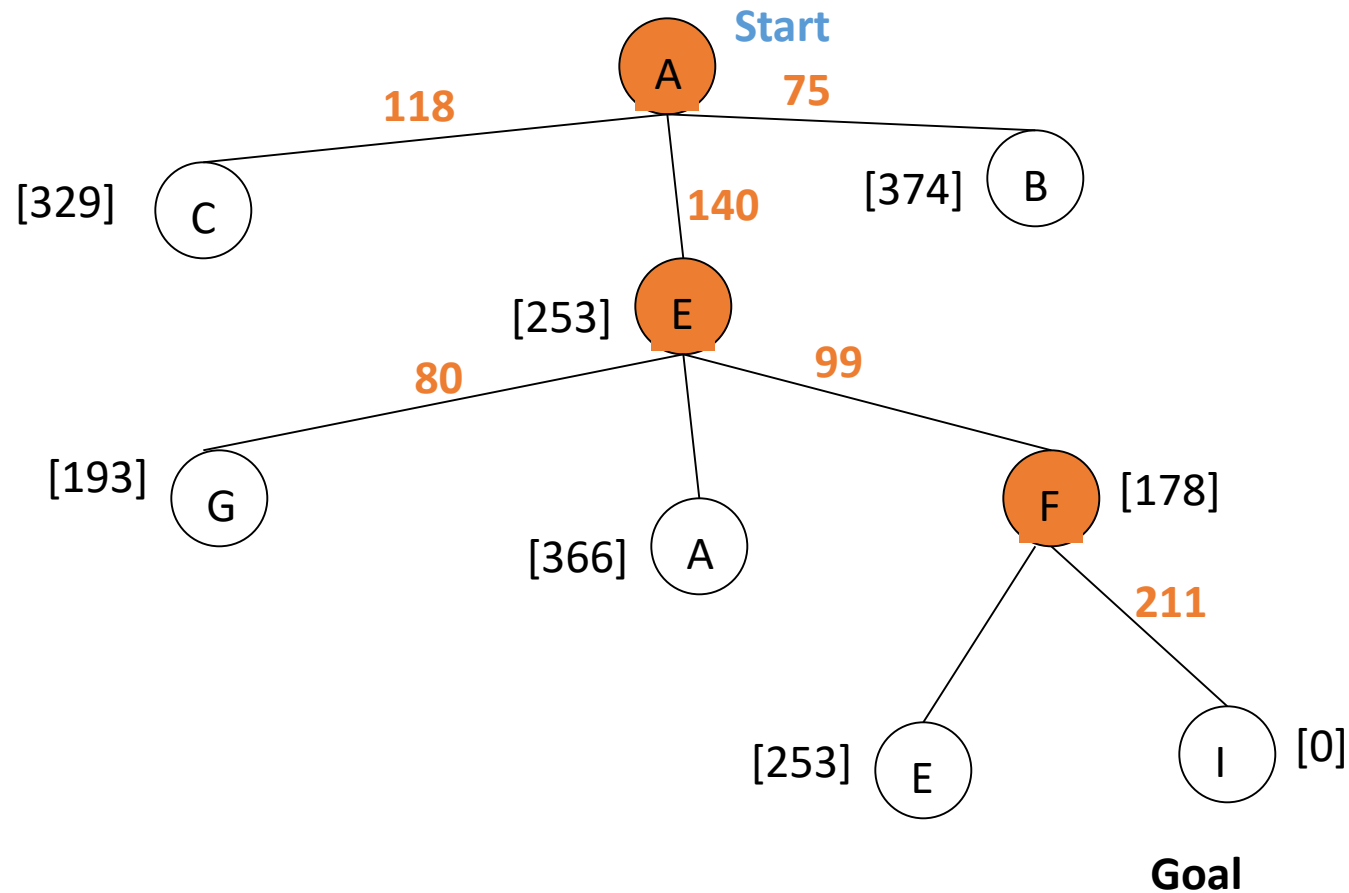
Start

(A)

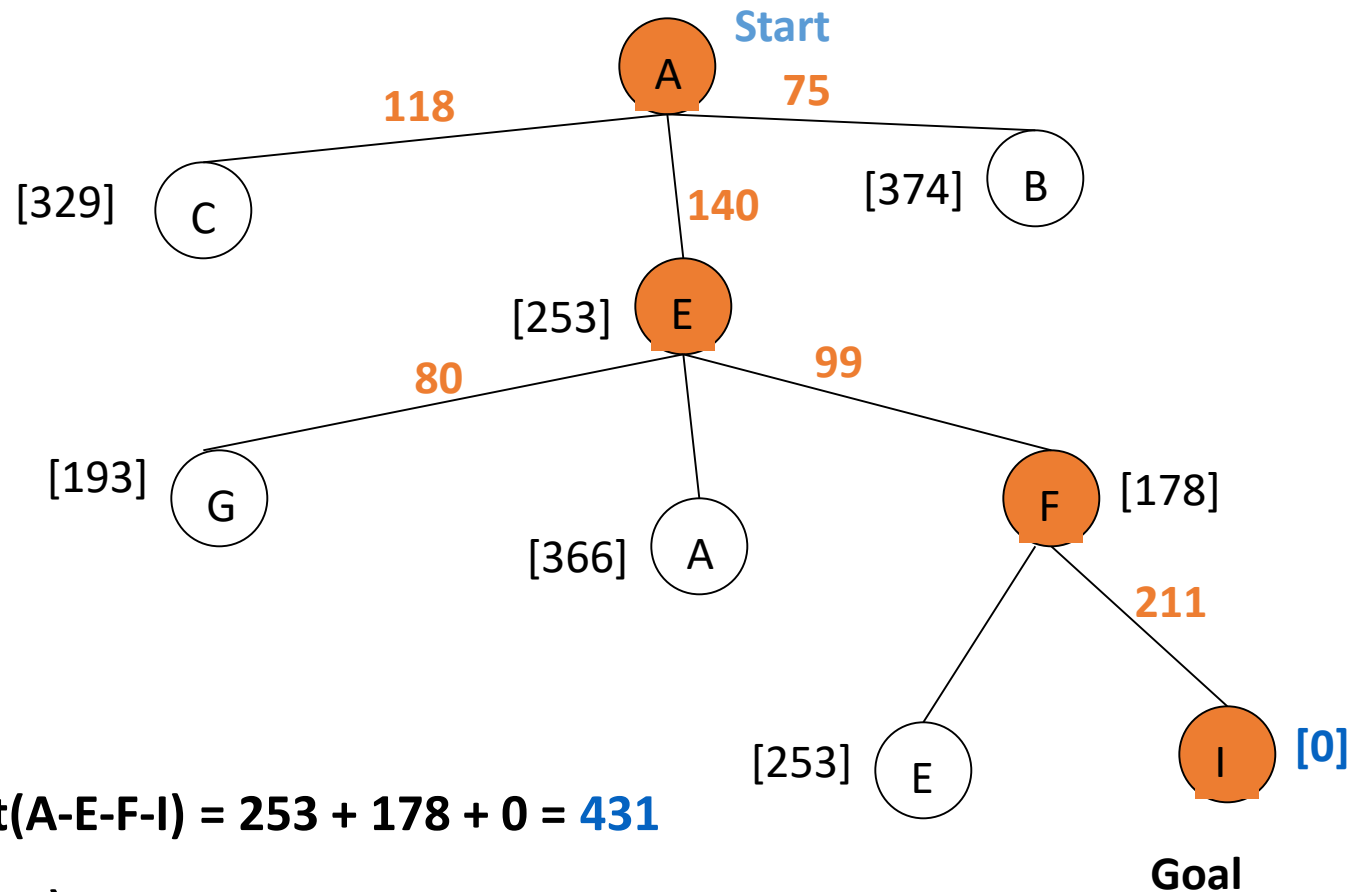# Greedy Search: Tree Search

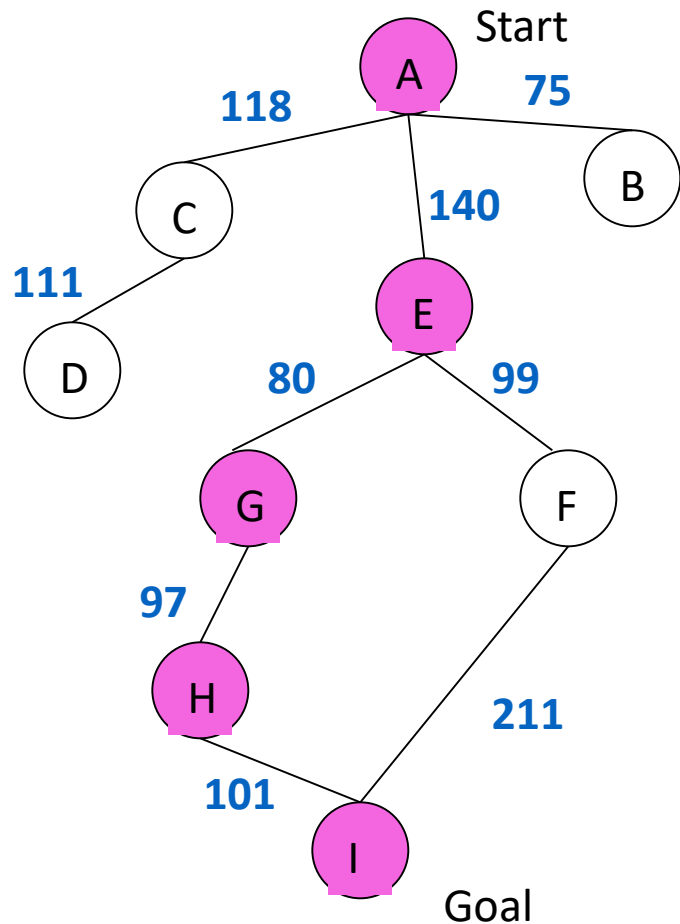# Greedy Search: Tree Search

Greedy Search: Tree Search

Greedy Search: Tree Search

Path cost(A-E-F-I) = 253 + 178 + 0 = **431**

dist(A-E-F-I) = 140 + 99 + 211 = **450**
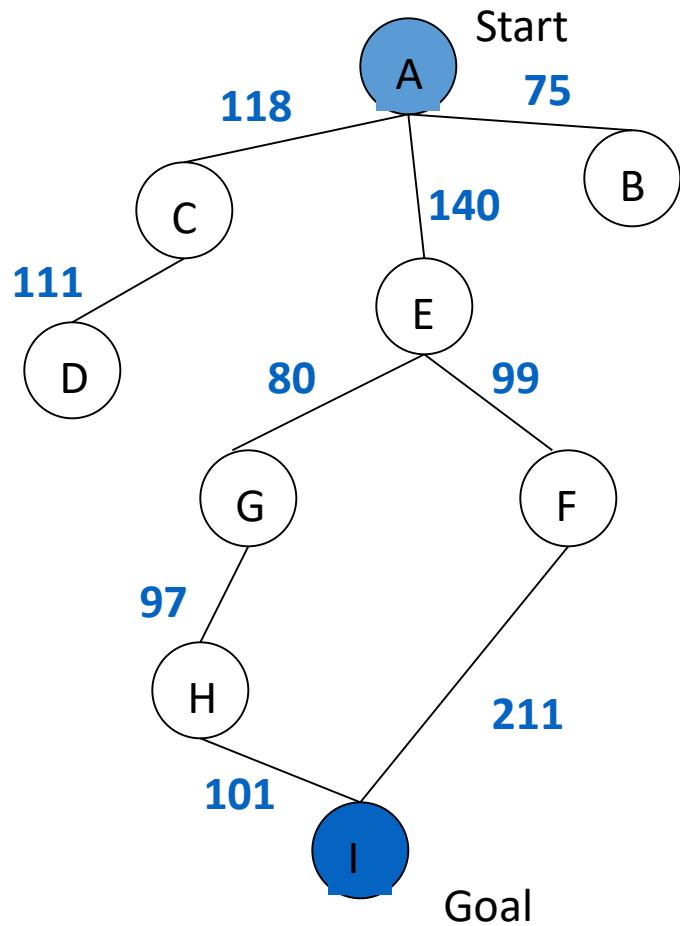
Greedy Search: Optimal?

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

Not optimal!

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

dist(A-E-G-H-I) =140+80+97+101=418

Greedy Search: Complete ?

State	Heuristic: h(n)
A	366
B	374
** C	**250**
D	244
E	253
F	178
G	193
H	98
I	0

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

Greedy Search: Tree Search

Start

(A)

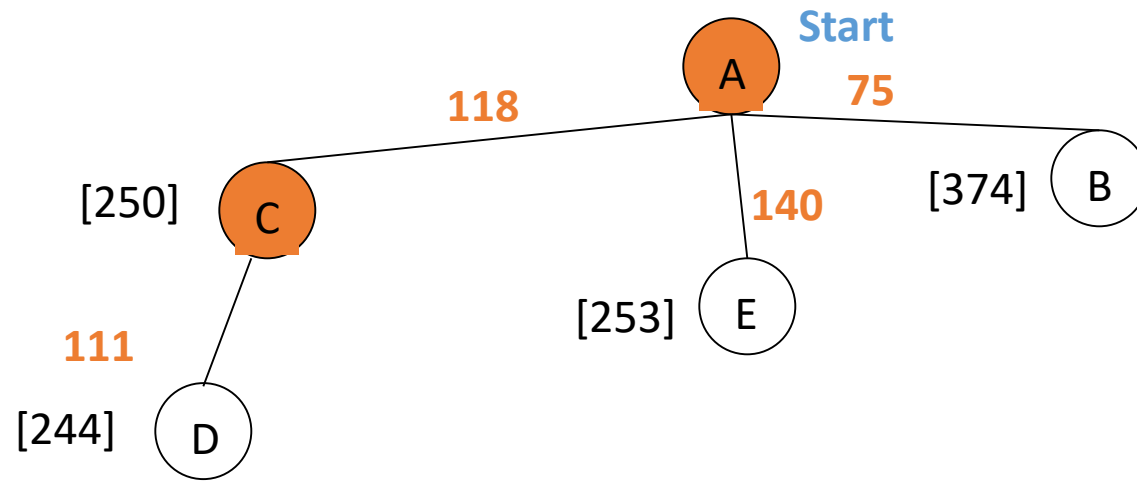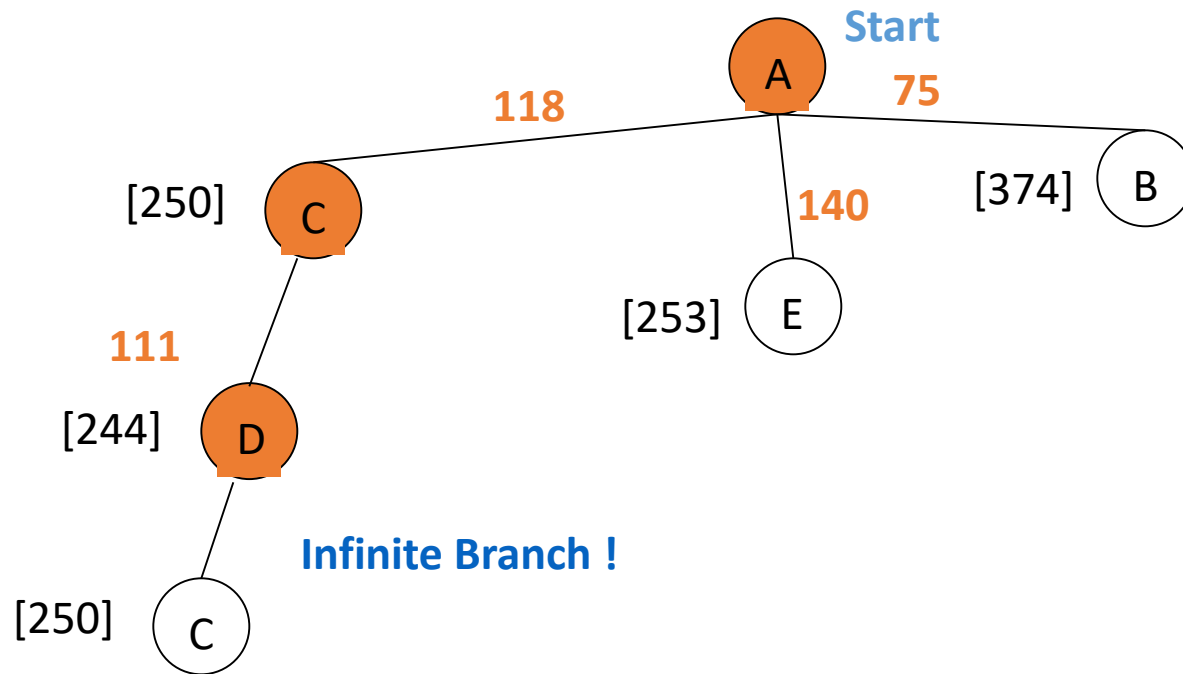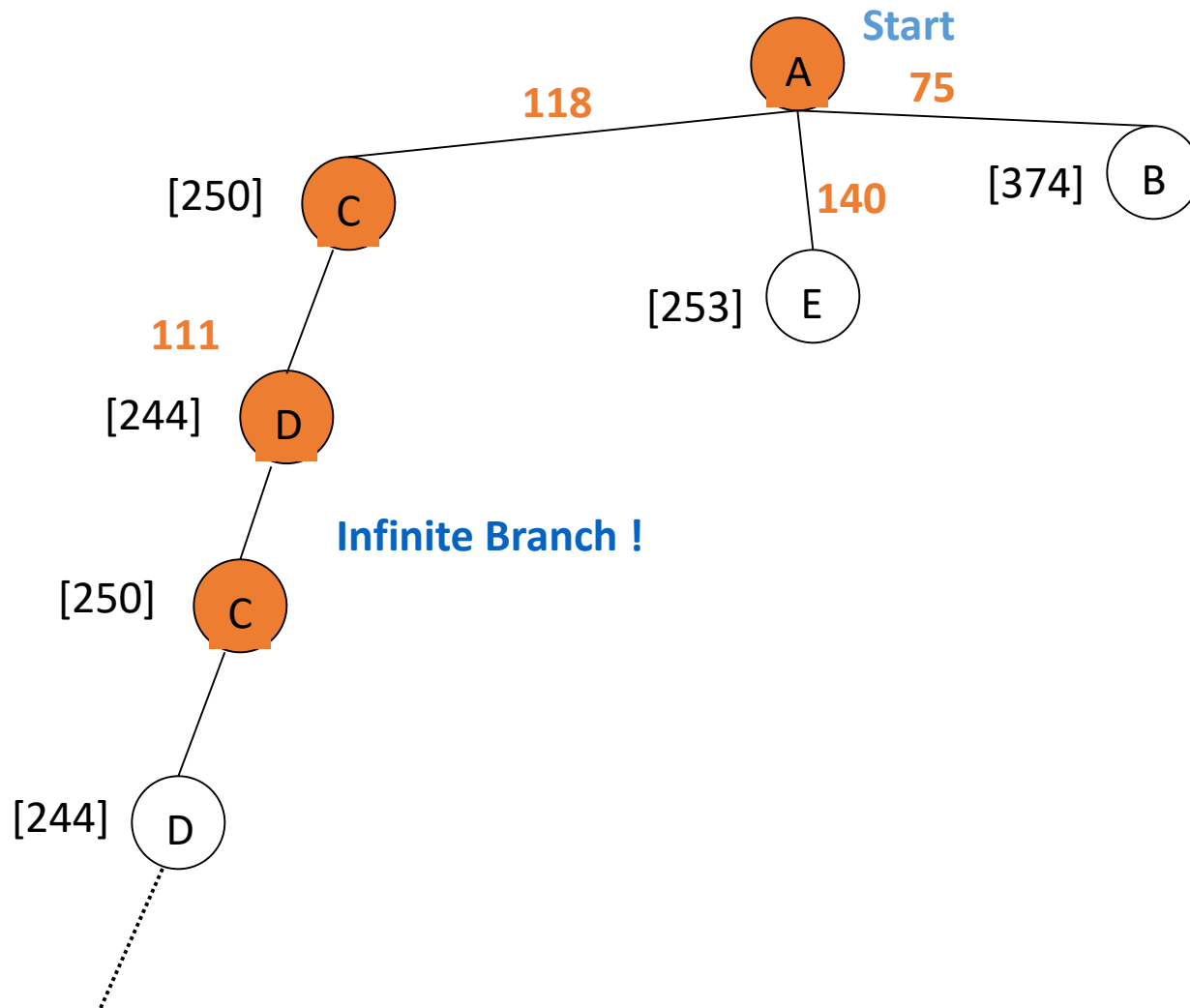# Greedy Search: Tree Search
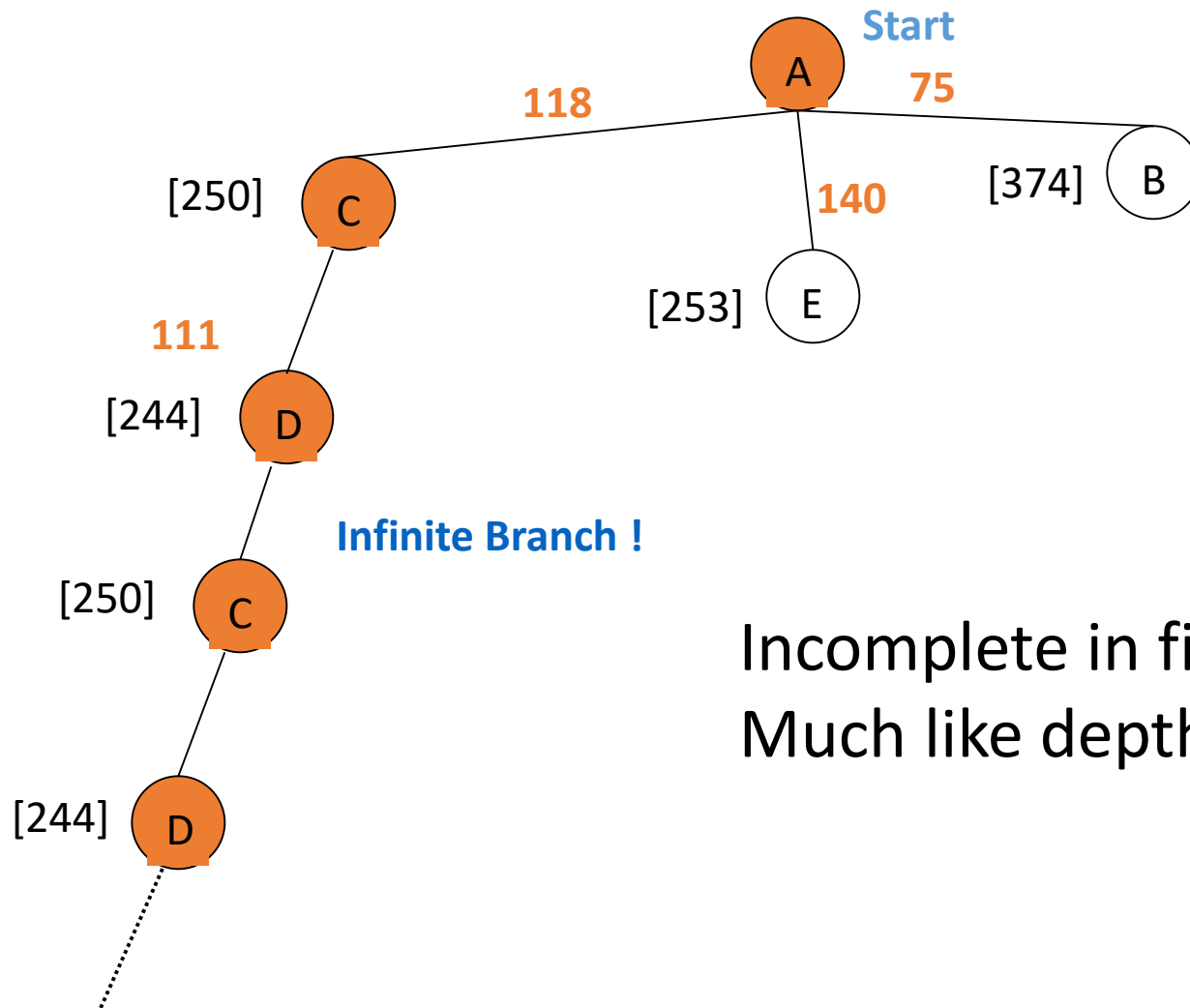
Greedy Search: Tree Search

Greedy Search: Tree Search

Greedy Search: Tree Search

Greedy Search: Tree Search

Start

A

118   75

[250] C   [374] B

140

[253] E

111

[244] D

Infinite Branch !

[250] C
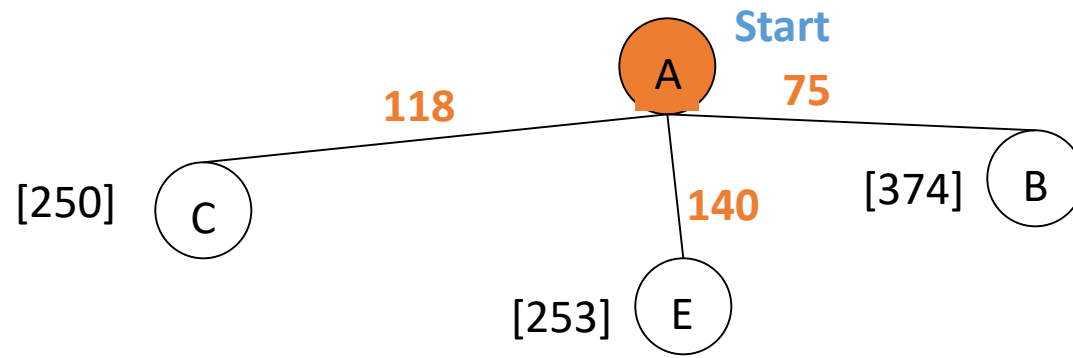
Incomplete in finite state space
Much like depth first search

[244] D

Greedy Search: Time and Space Complexity ?

Start

A

118   75

C   B

140

111   E

D   80   99

G   F

97

H   211

101

I

Goal
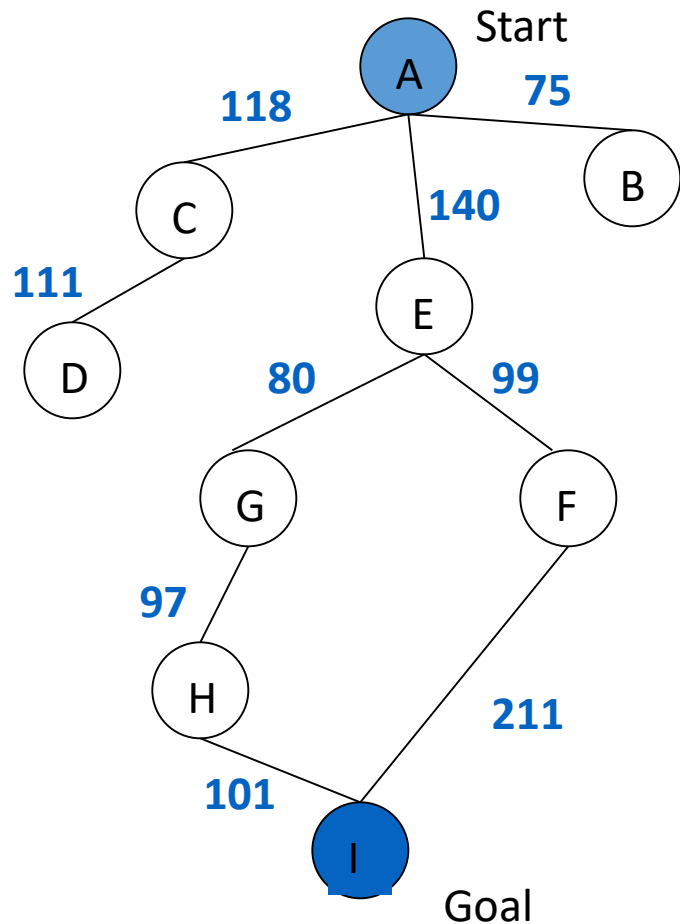
- Greedy search is not optimal.

- Greedy search is incomplete

- In the worst case, the Time and Space Complexity of Greedy Search are both $O(b^m)$

- Where b is the branching factor and m the maximum path length
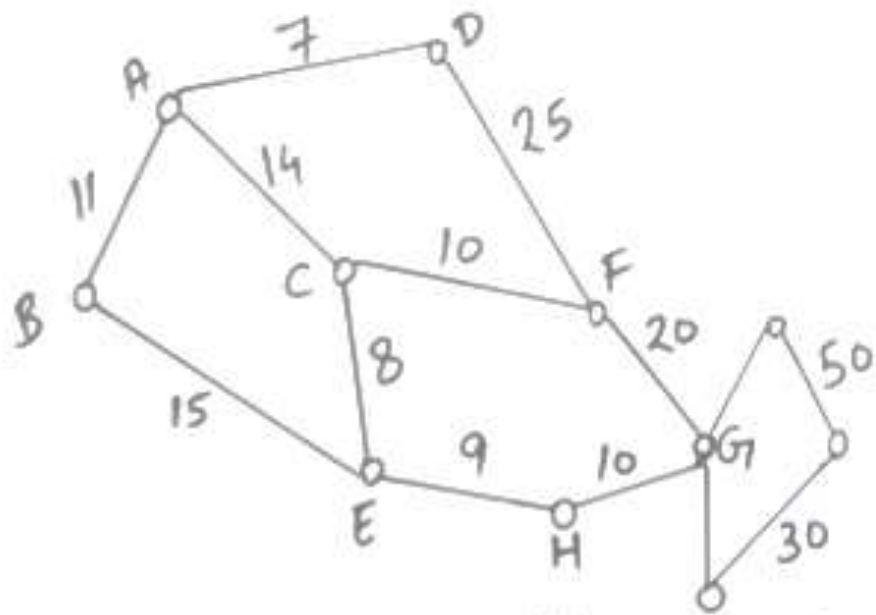
Straight line dist.

A → G = 40
B → G = 32
C → G = 25
D → G = 35
E → G = 19
F → G = 17
H → G = 10
G → G = 0

	Frontier (Greedy)	Expand	Explored
1	(A,40)	A	Empty
2	(A-B,32)(A-C,25)(A-D,35)	C	A
3	(A-B,32)(A-D,35) (A-C-E,19)(A-C-F,17)	F	A,C
4	(A-B,32)(A-D,35) (A-C-E,19)(A-C-F-G,0)	G	A,C,F
5	(A-C-F-G,0) Goal Found		
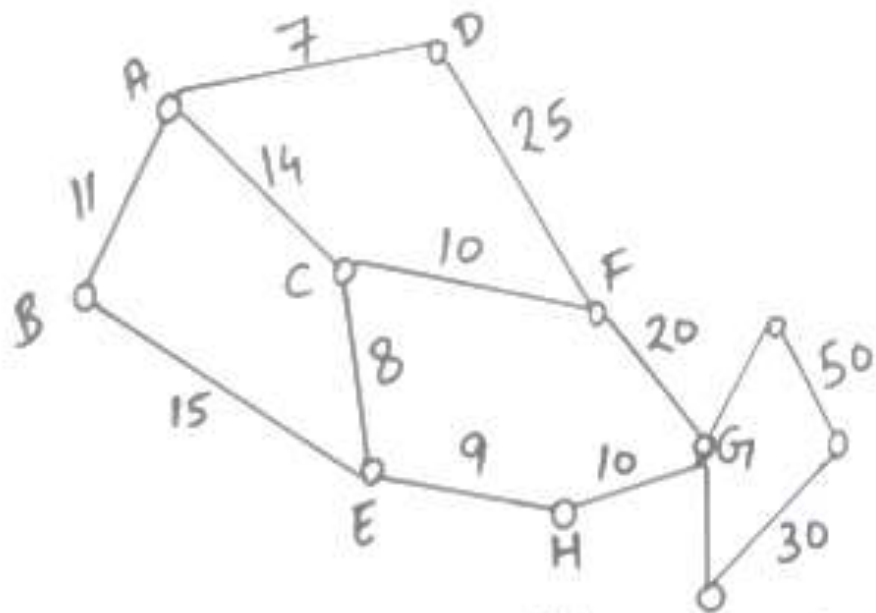
Straight line dist.

A → G = 40
B → G = 32
C → G = 25
D → G = 35
E → G = 19
F → G = 17
H → G = 10
G → G = 0

	Frontier (Greedy)	Expand	Explored
1	(A,40)	A	Empty
2	(A-C,25) (A-B,32)(A-D,35)	C	A
3	(A-B,32)(A-D,35)(A-C-F,17)(A-C-E,19)	F	A,C
4	(A-B,32)(A-D,35)(A-C-F-G,0)(A-C-E,19)	G	A,C,F
5	(A-C-F-G) Goal Found (Cost: 44)		A,C,F,G

	Frontier (UCS)	Expand	Explored
1	(A,0)	A	
2	(A-B,11)(A-C,14)(A-D,7)	D	A
3	(A-B,11)(A-C,14)(A-D-F,32)	B	A,D
4	(A-B-E,26)(A-C,14)(A-D-F,32)	C	A,D,B
5	~~(A-B-E,26)~~(A-C-F,24)(A-C-E,22) ~~(A-D-F,32)~~	E	A,D,B,C
6	~~(A-B-E,26)~~(A-C-F,24)(A-C-E-H,31) ~~(A-D-F,32)~~	F	A,D,B,C,E
7	~~(A-B-E,26)~~(A-C-F-G,44)(A-C-E-H,31) ~~(A-D-F,32)~~	H	A,D,B,C,E, F
8	~~(A-B-E,26)~~(A-C-F-G,44)(A-C-E-H-G,41) ~~(A-D-F,32)~~	G	A,D,B,C,E, F,H
9	A-C-E-H-G,41 Goal Found	G	A,D,B,C,E, F,H,G

Heuristics

- Problem:
  - A heuristic is only an informed guess of the next step to be taken in solving a problem.
  - It is often based on experience or intuition.
  - Heuristics use limited information, such as knowledge of the present situation or descriptions of states currently on the open list.
  - They are not always able to predict the exact behavior of the state space farther along in the search.
  - A heuristic can lead a search algorithm to a suboptimal solution or fail to find any solution at all. This is an **inherent limitation** of heuristic search.