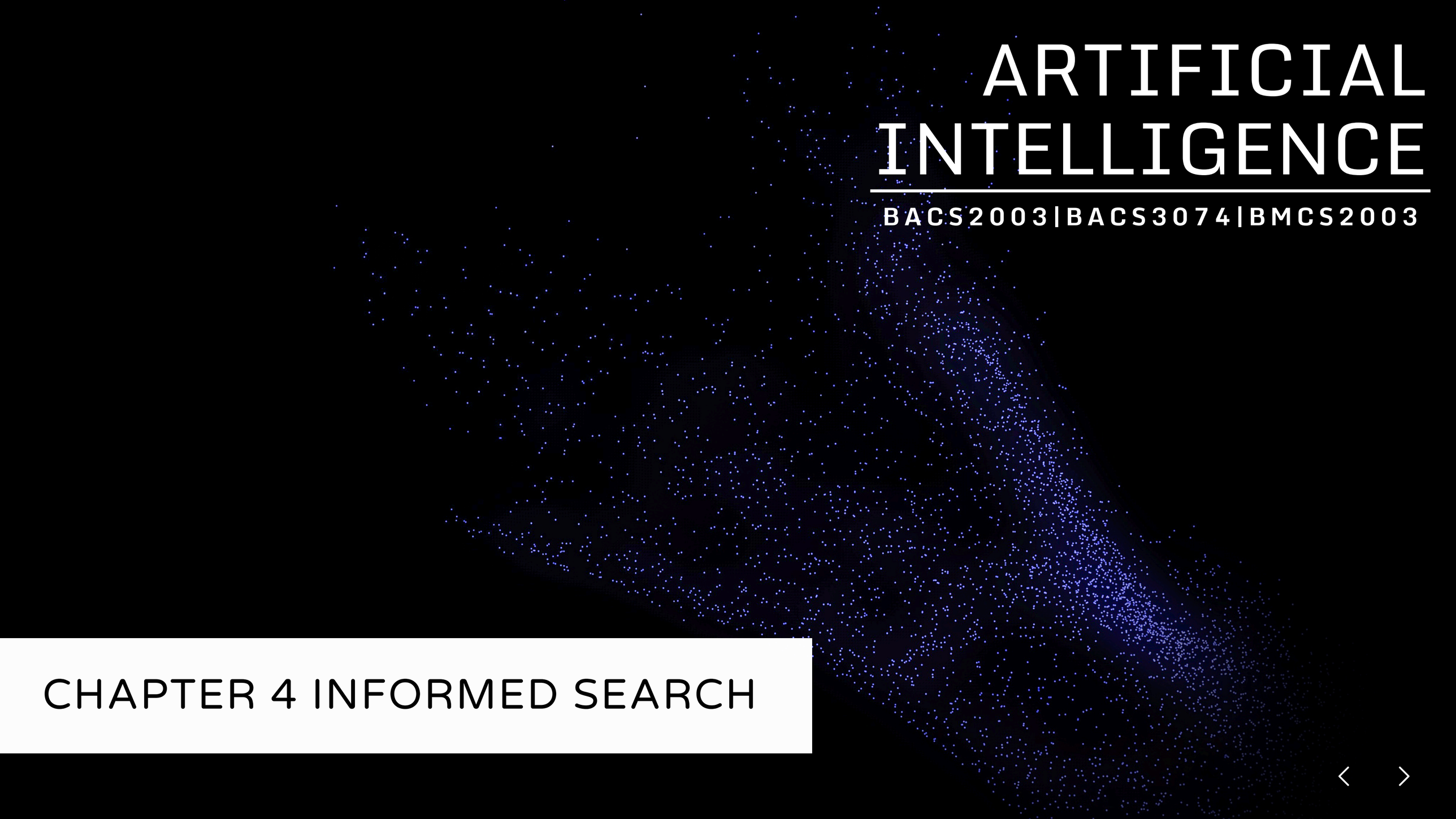
ARTIFICIAL INTELLIGENCE

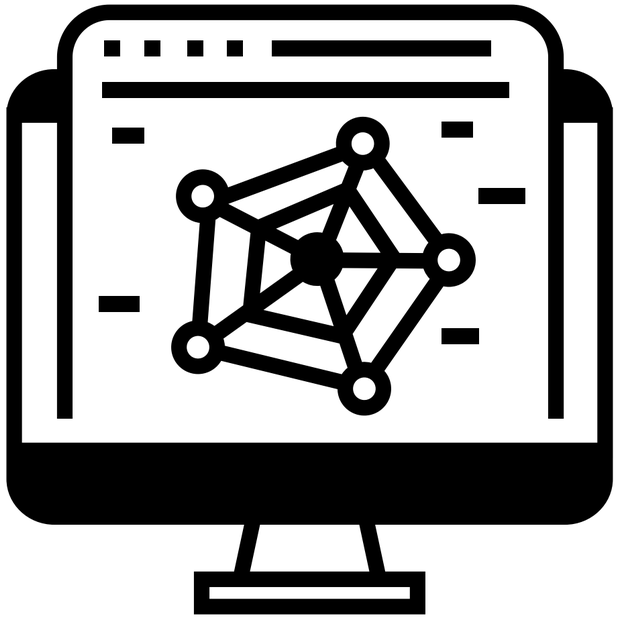
BACS2003|BACS3074|BMCS2003



CHAPTER 4 INFORMED SEARCH

OUTCOMES

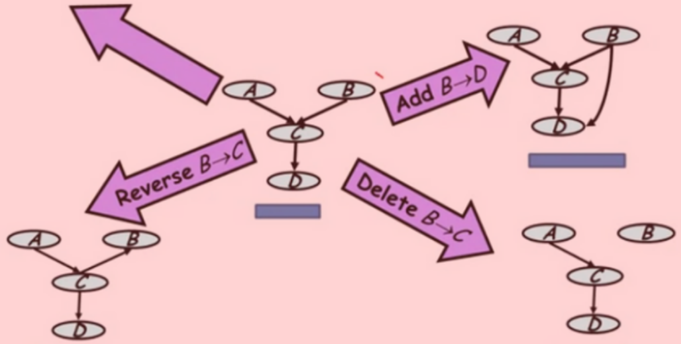
1. **Define and Evaluate Informed (Heuristic) Search**
2. **Heuristic Search Techniques**



SEARCH STRATEGIES

- UNINFORMED SEARCH / BLIND SEARCH

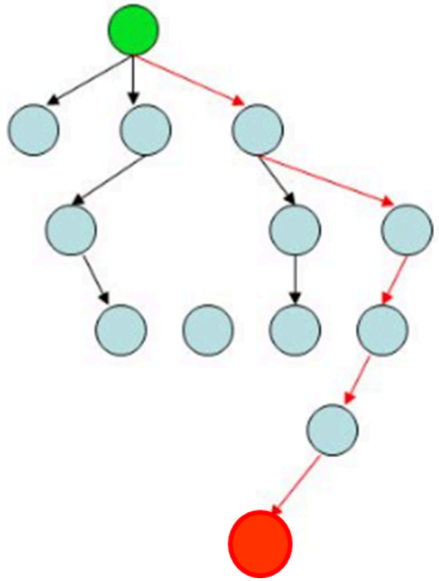
- INFORMED SEARCH / HEURISTIC SEARCH



HEURISTIC

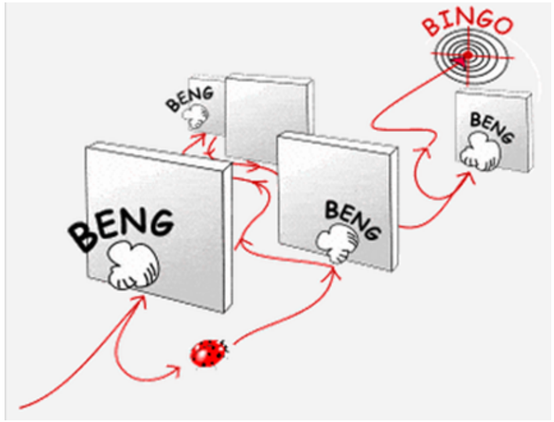
- A function that ranks alternatives at each branching step based on available information to decide which branch to follow.
- The agent is informed which branch(es) in a state space that is(are) most likely to lead to an acceptable problem solution.

Idea: be **smart** about what paths to try.



HEURISTIC

- Design a heuristic often based on experience or intuition.



- Heuristic is only an informed guess of the next step to be taken – seldom able to predict the exact behavior of the state space.
- Can lead to a suboptimal solution or fail.

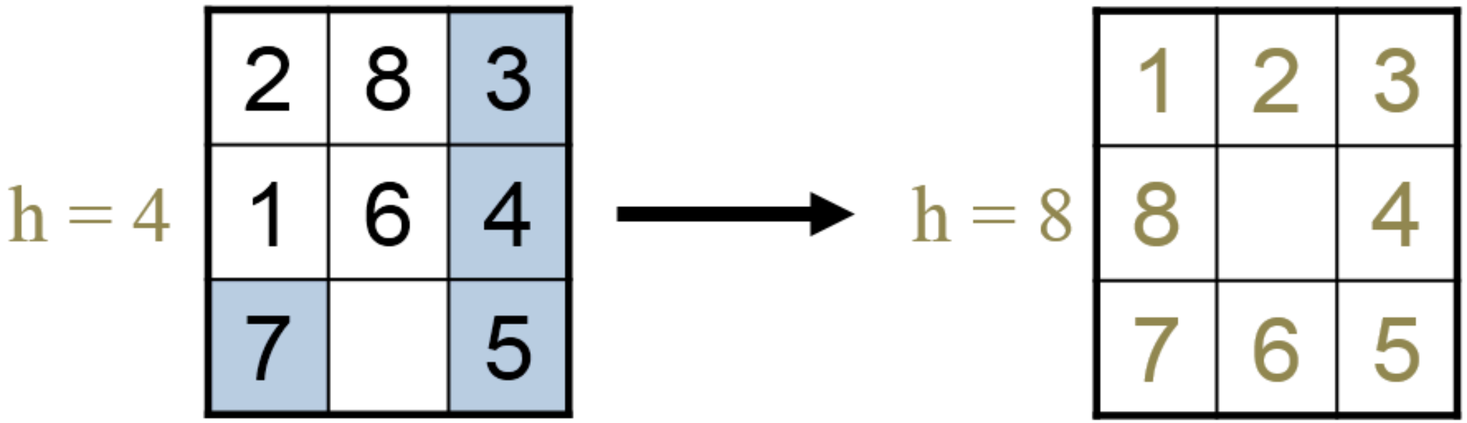
HEURISTIC FUNCTION

The heuristic function, h(n) indicates how “close” a state n is to the goal.

EXAMPLE

Let the heuristic function, h(n), is given as follows:
h(n) = number of tiles at the right place.
* h(n) determines the heuristic cost of a state

INITIAL: FINAL:



HEURISTIC SEARCH TECHNIQUES

1. Hill climbing
   - Simple Hill Climbing
   - Steepest Ascend Hill Climbing
2. Best-first Search
3. A* Search

HILL CLIMBING

The simplest way in heuristic search

- Simple Hill Climbing
- Steepest Ascend Hill Climbing

Limitations of Hill Climbing

- It is less combinatorially explosive as it searches locally rather than globally;
- possibly very inefficient and ineffective.

SIMPLE HILL CLIMBING

ALGORITHM

1.Set initial state to current
2.Loop on the following until goal is found or no more operators available

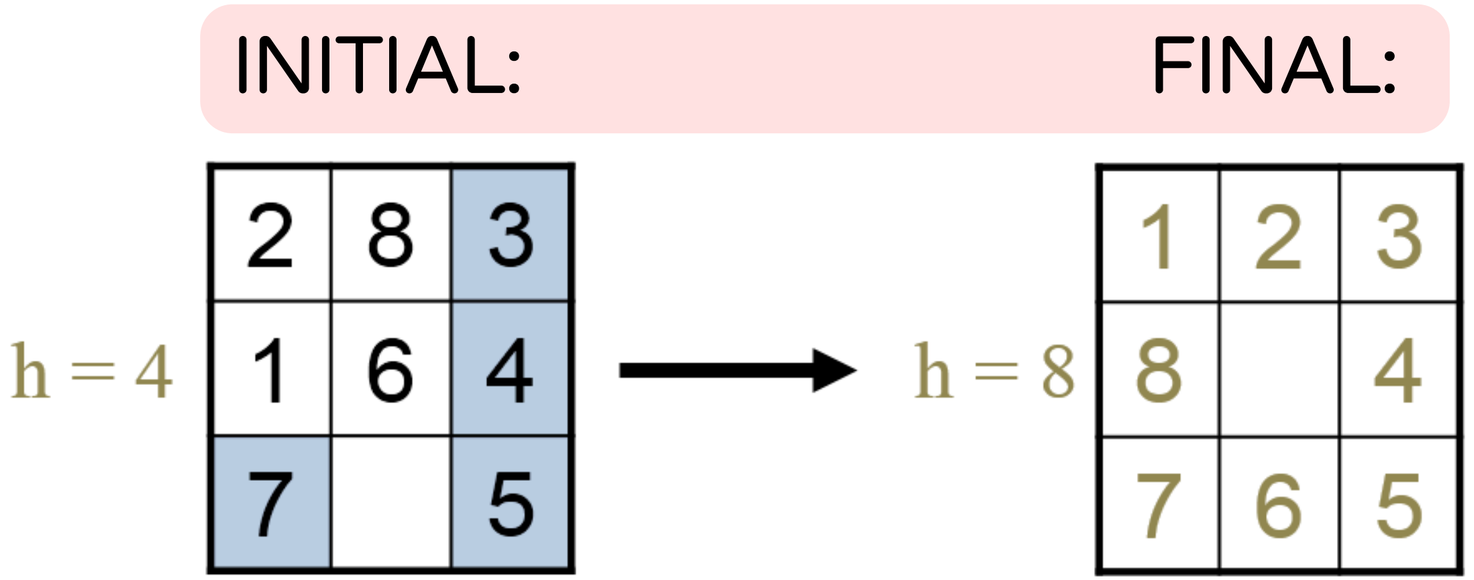
- Select an operator and apply it to create a new state
- Evaluate new state
- If new state is better than current state, perform operator making new state the current state

3.Once loop is exited, either we have found the goal or return fail

EXAMPLE

SIMPLE HILL CLIMBING

Let the heuristic function, h(n), is given as follows:
h(n) = number of tiles at the right place.
* h(n) determines the heuristic cost of a state



EXAMPLE

SIMPLE HILL CLIMBING

Let the heuristic function, h(n), is given as follows:
h(n) = number of tiles at the right place.

Sequence: Move the empty tile follows the sequence of UP,RIGHT, LEFT, DOWN
*No repeated state is allowed
*Not allowed to slide over the boundary

h = 4

2	8	3
1	6	4
7		5

h = 8

1	2	3
8		4
7	6	5

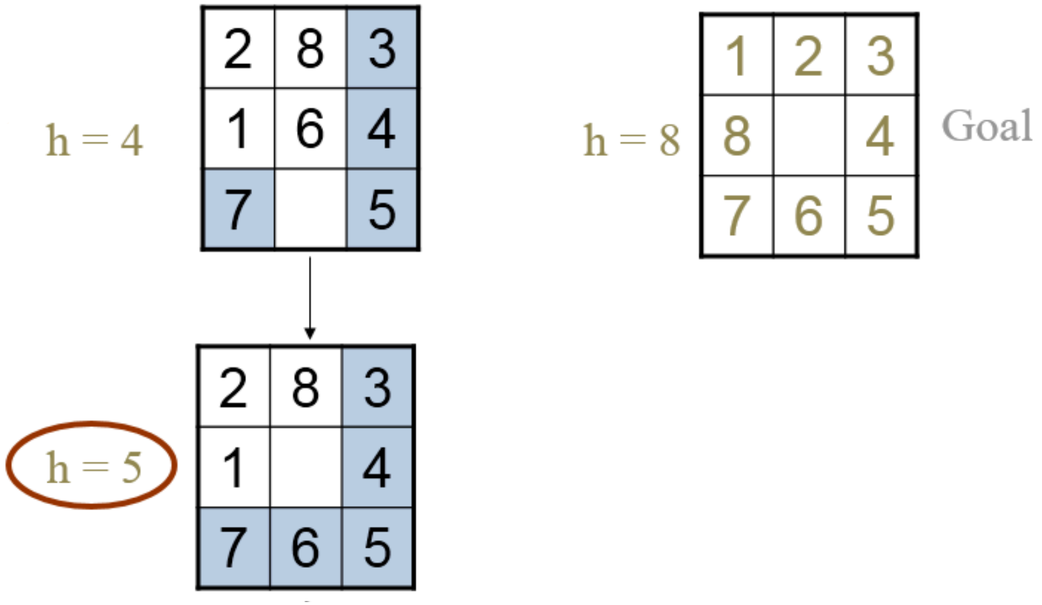
Goal

EXAMPLE

SIMPLE HILL CLIMBING

Let the heuristic function, h(n), is given as follows:
h(n) = number of tiles at the right place.

Sequence: Move the empty tile follows the sequence of UP,RIGHT, LEFT, DOWN
*No repeated state is allowed
*Not allowed to slide over the boundary

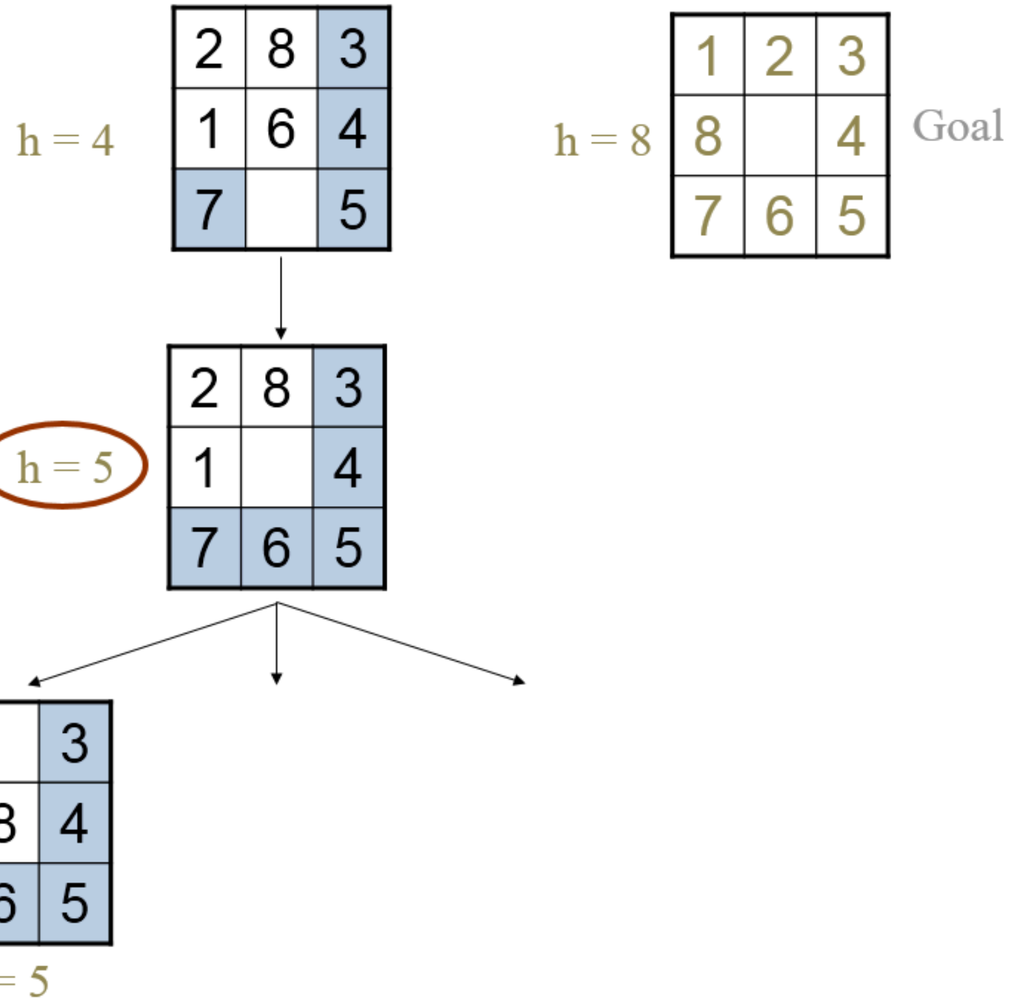


EXAMPLE

SIMPLE HILL CLIMBING

Let the heuristic function, h(n), is given as follows:
h(n) = number of tiles at the right place.

Sequence: Move the empty tile follows the sequence of UP,RIGHT, LEFT, DOWN
*No repeated state is allowed
*Not allowed to slide over the boundary

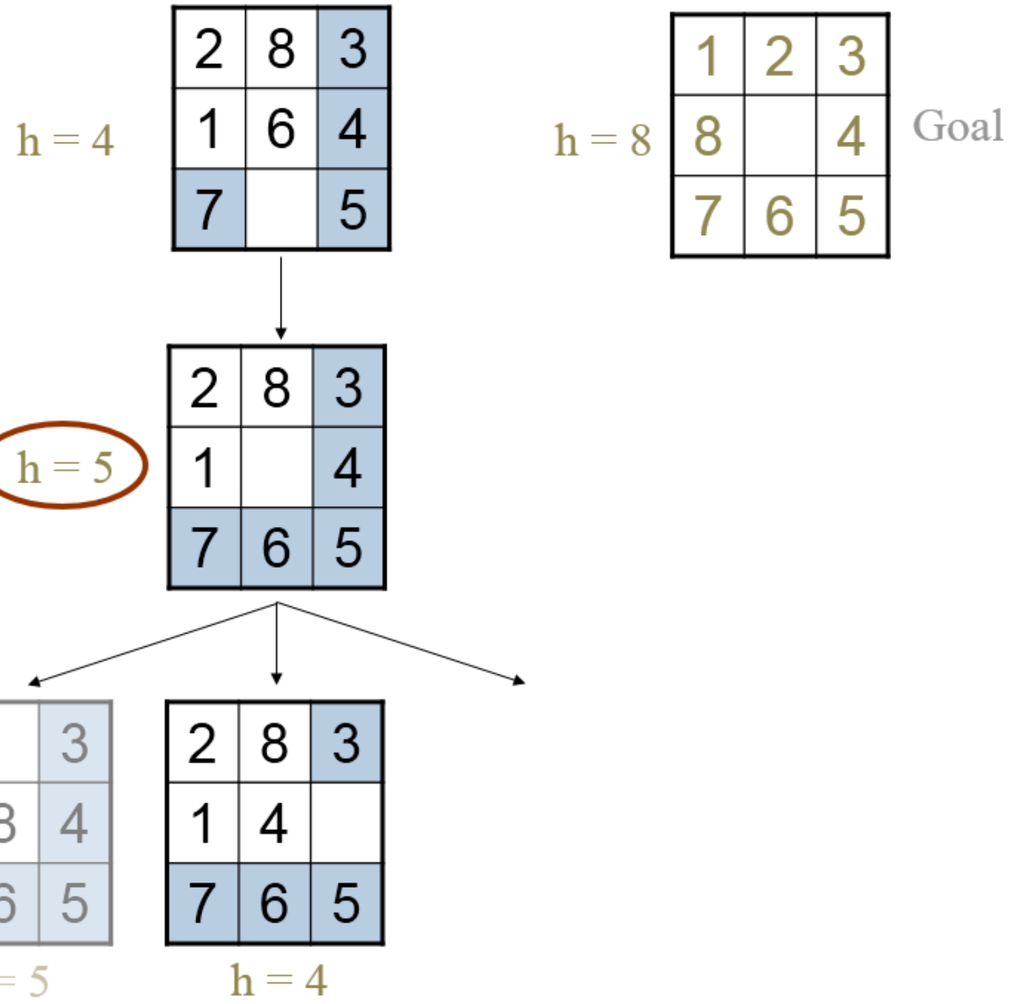


EXAMPLE

SIMPLE HILL CLIMBING

Let the heuristic function, h(n), is given as follows:
h(n) = number of tiles at the right place.

Sequence: Move the empty tile follows the sequence of UP,RIGHT, LEFT, DOWN
*No repeated state is allowed
*Not allowed to slide over the boundary

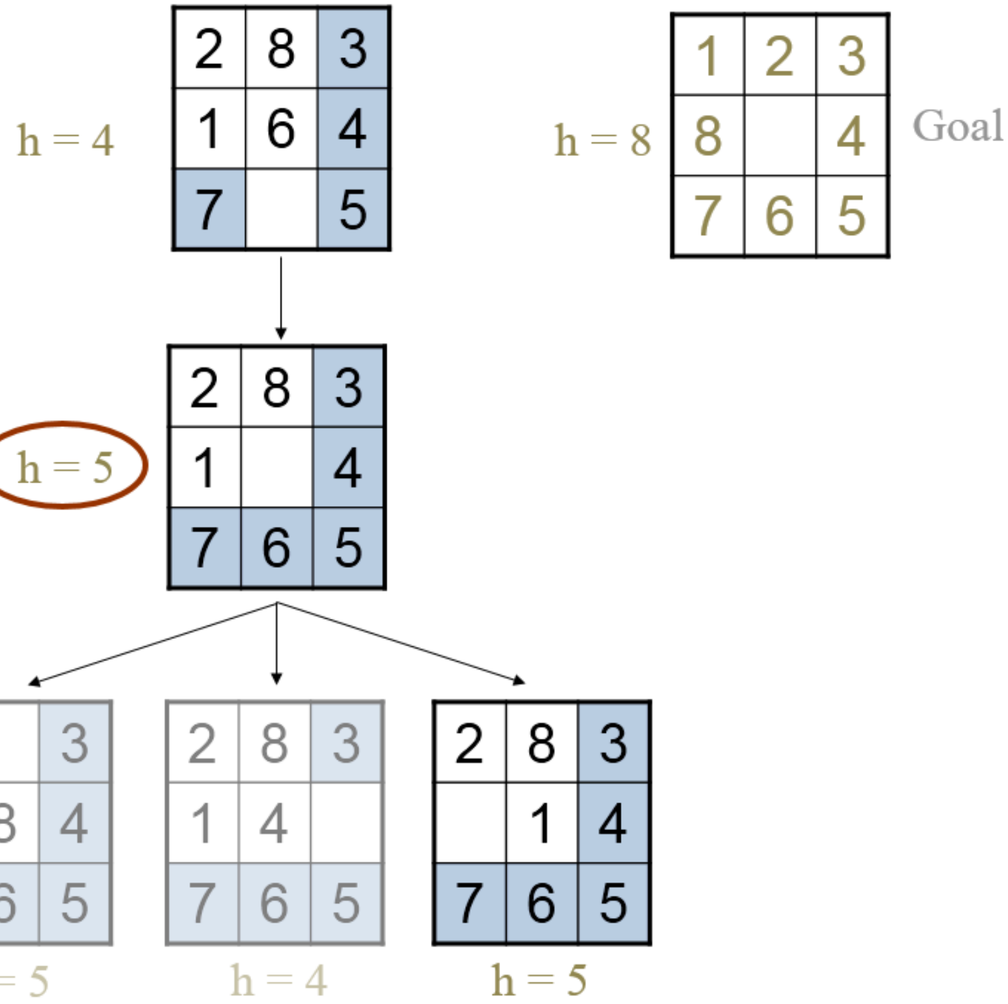


EXAMPLE

SIMPLE HILL CLIMBING

Let the heuristic function, h(n), is given as follows:
h(n) = number of tiles at the right place.

Sequence: Move the empty tile follows the sequence of UP,RIGHT, LEFT, DOWN
*No repeated state is allowed
*Not allowed to slide over the boundary

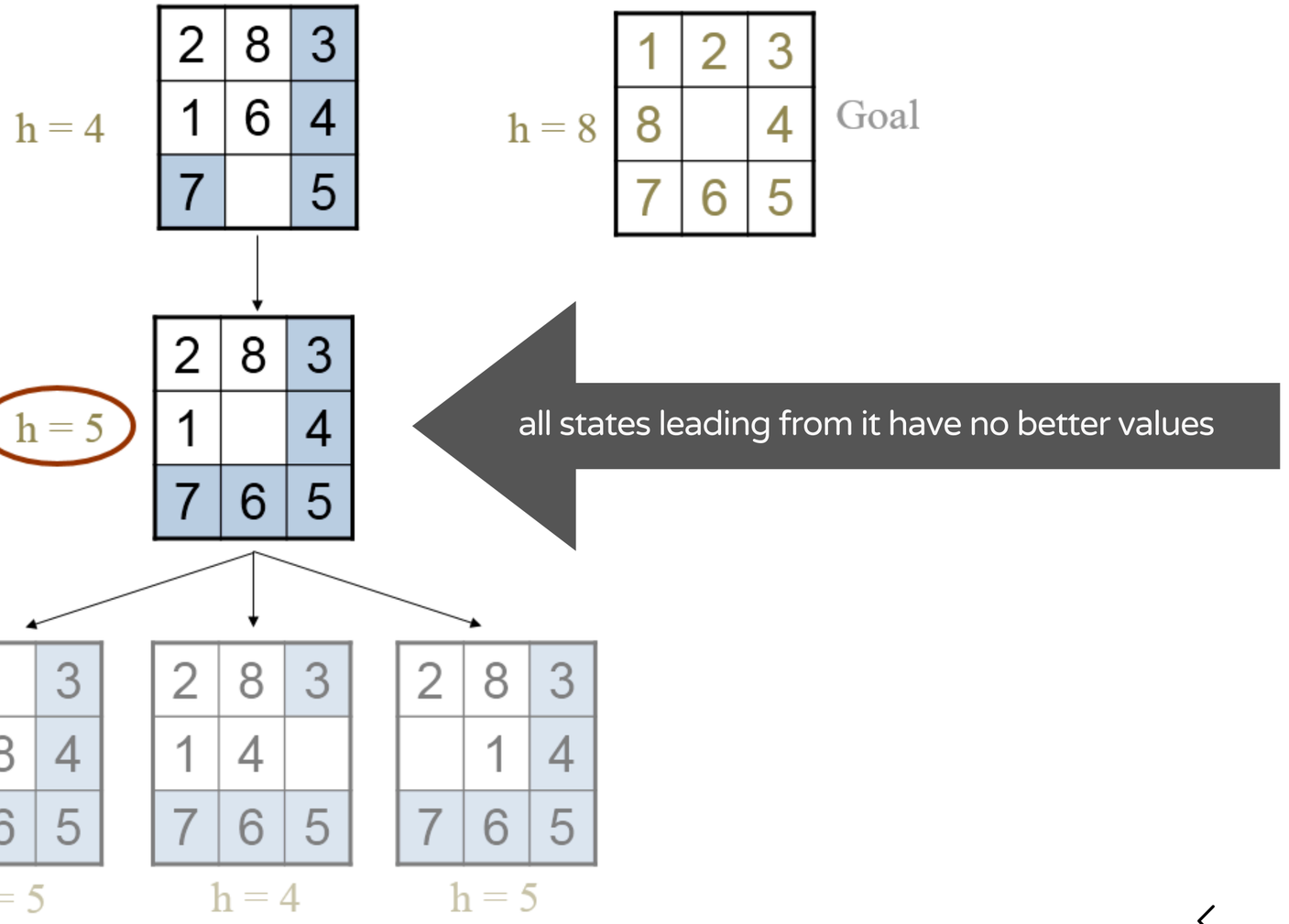


EXAMPLE

SIMPLE HILL CLIMBING

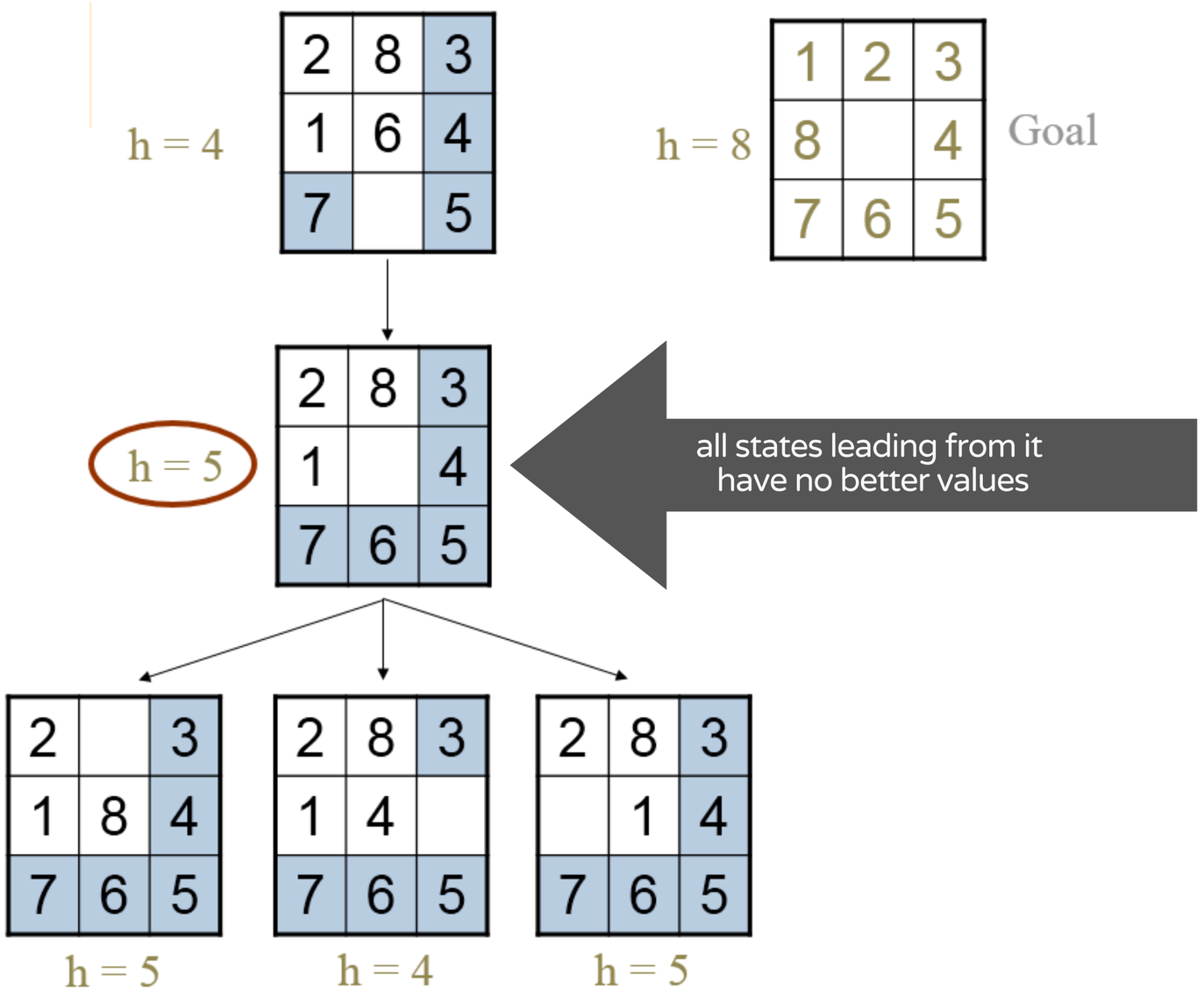
Let the heuristic function, h(n), is given as follows:
h(n) = number of tiles at the right place.

Sequence: Move the empty tile follows the sequence of UP,RIGHT, LEFT, DOWN
*No repeated state is allowed
*Not allowed to slide over the boundary



OPS! WE'RE STUCK!

SIMPLE HILL CLIMBING



- Simple hill climbing has faced a dead end.
- The child nodes are worse than the parent node.

EXAMPLE 2

SIMPLE HILL CLIMBING

Let the heuristic function, h(n), is given as follows:
h(n) = number of tiles at the right place.

Sequence: Move the empty tile follows the sequence of UP,LEFT, RIGHT, DOWN
*No repeated state is allowed
*Not allowed to slide over the boundary

h = 6

2		3
8	1	4
7	6	5

Goal
h = 8

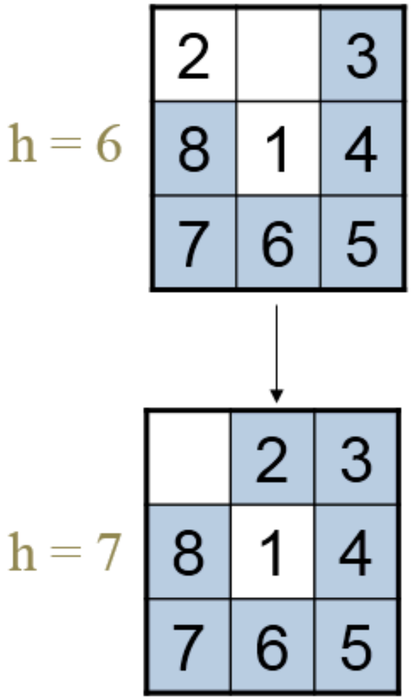
1	2	3
8		4
7	6	5

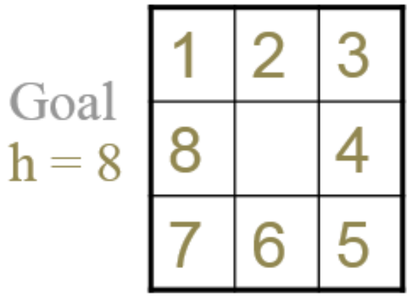
EXAMPLE 2

SIMPLE HILL CLIMBING

Let the heuristic function, h(n), is given as follows:
h(n) = number of tiles at the right place.

Sequence: Move the empty tile follows the sequence of UP,LEFT, RIGHT, DOWN
*No repeated state is allowed
*Not allowed to slide over the boundary



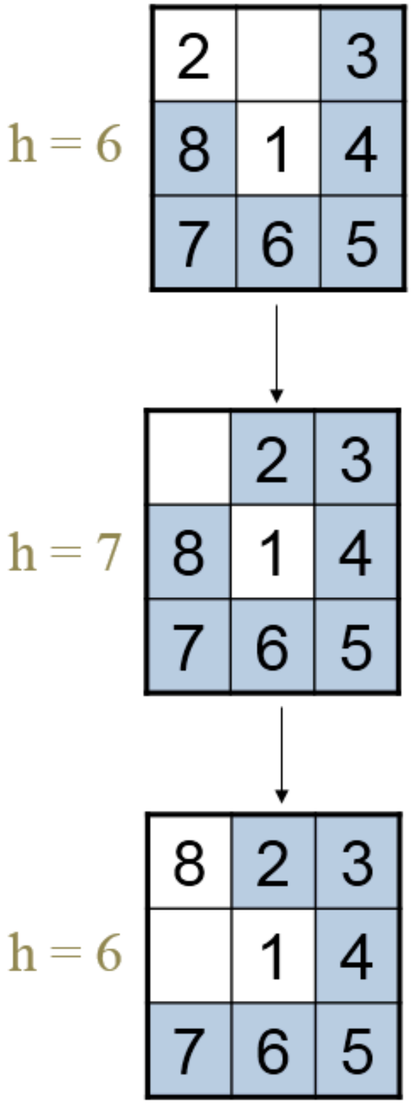


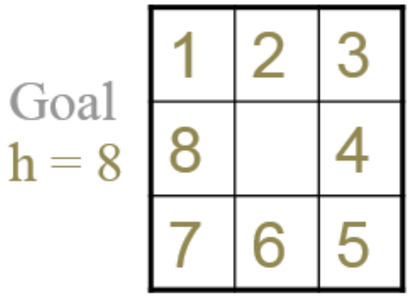
EXAMPLE 2

SIMPLE HILL CLIMBING

Let the heuristic function, h(n), is given as follows:
h(n) = number of tiles at the right place.

Sequence: Move the empty tile follows the sequence of UP,LEFT, RIGHT, DOWN
*No repeated state is allowed
*Not allowed to slide over the boundary





state leading from it
have no better value & no other possible child nodes

STEEPEST ASCEND HILL CLIMBING

Here, we attempt to improve on the previous Hill Climbing algorithm

ALGORITHM

1.Set initial state to current

2.Loop on the following until goal is found or a complete iteration occurs without change to current state

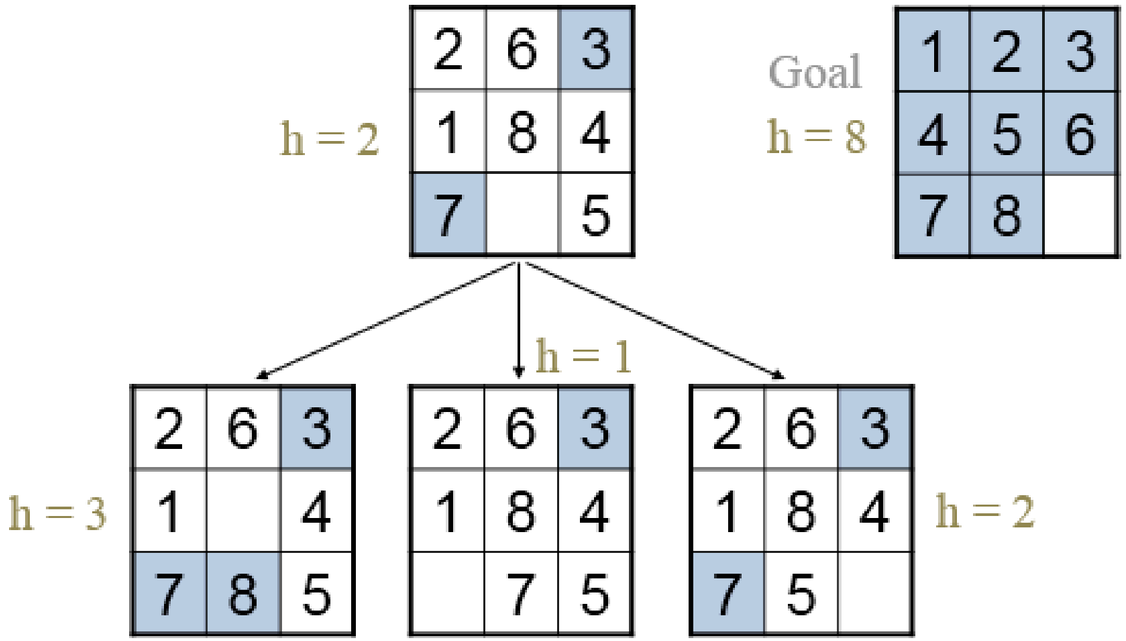
- Generate all successor states to current state
- Evaluate all successor states using heuristic
- Select the successor state that yields the highest heuristic value and perform that operator

EXAMPLE 1

STEEPEST ASCEND HILL CLIMBING

Let the heuristic function, h(n), is given as follows:
h(n) = number of tiles at the right place.

Sequence: Move the empty tile follows the sequence of UP,LEFT, RIGHT, DOWN
*No repeated state is allowed
*Not allowed to slide over the boundary

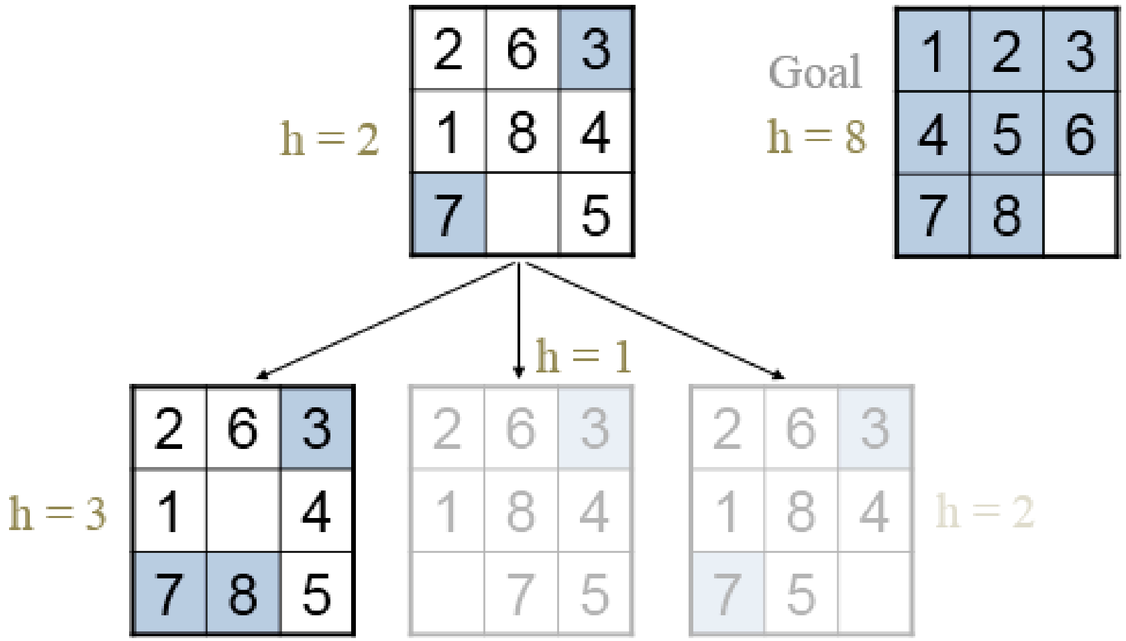


EXAMPLE 1

STEEPEST ASCEND HILL CLIMBING

Let the heuristic function, h(n), is given as follows:
h(n) = number of tiles at the right place.

Sequence: Move the empty tile follows the sequence of UP,LEFT, RIGHT, DOWN
*No repeated state is allowed
*Not allowed to slide over the boundary

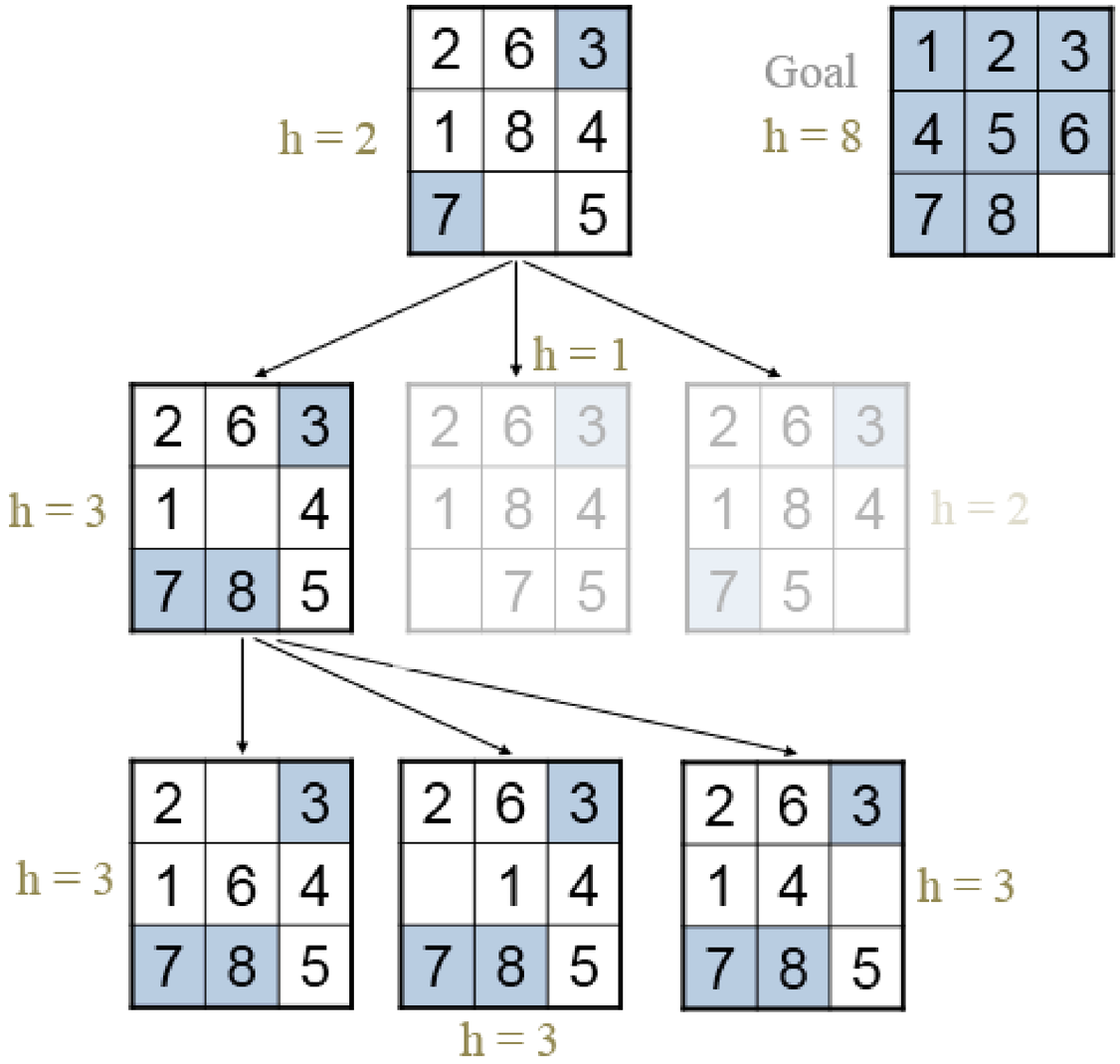


EXAMPLE 1

STEEPEST ASCEND HILL CLIMBING

Let the heuristic function, h(n), is given as follows:
h(n) = number of tiles at the right place.

Sequence: Move the empty tile follows the sequence of UP,LEFT, RIGHT, DOWN
*No repeated state is allowed
*Not allowed to slide over the boundary



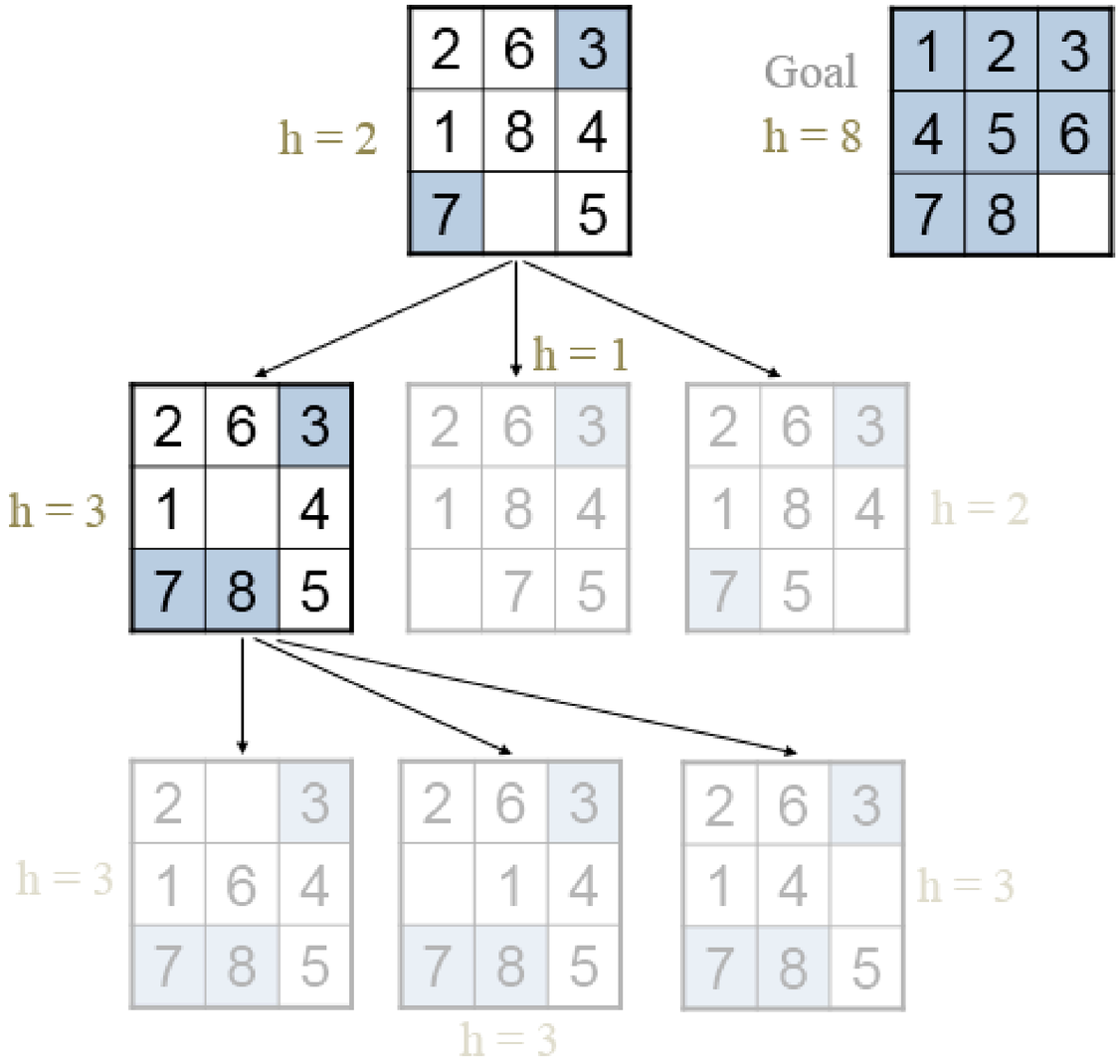
EXAMPLE 1

STEEPEST ASCEND HILL CLIMBING

Let the heuristic function, h(n), is given as follows:
h(n) = number of tiles at the right place.

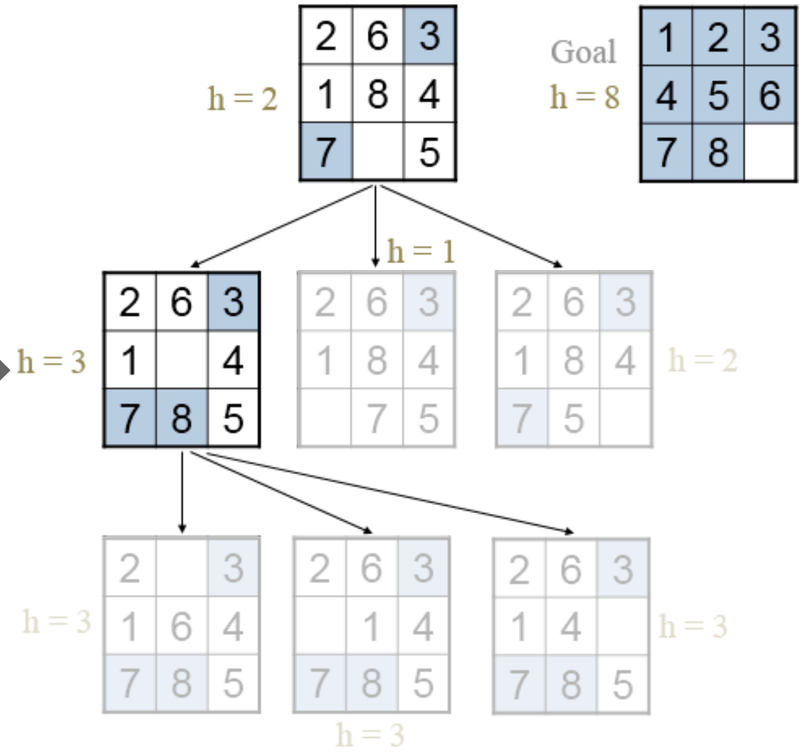
Sequence: Move the empty tile follows the sequence of UP,LEFT, RIGHT, DOWN
*No repeated state is allowed
*Not allowed to slide over the boundary

all states leading from it have equal value



OPS! WE'RE STUCK AGAIN!

STEEPEST ASCEND HILL CLIMBING

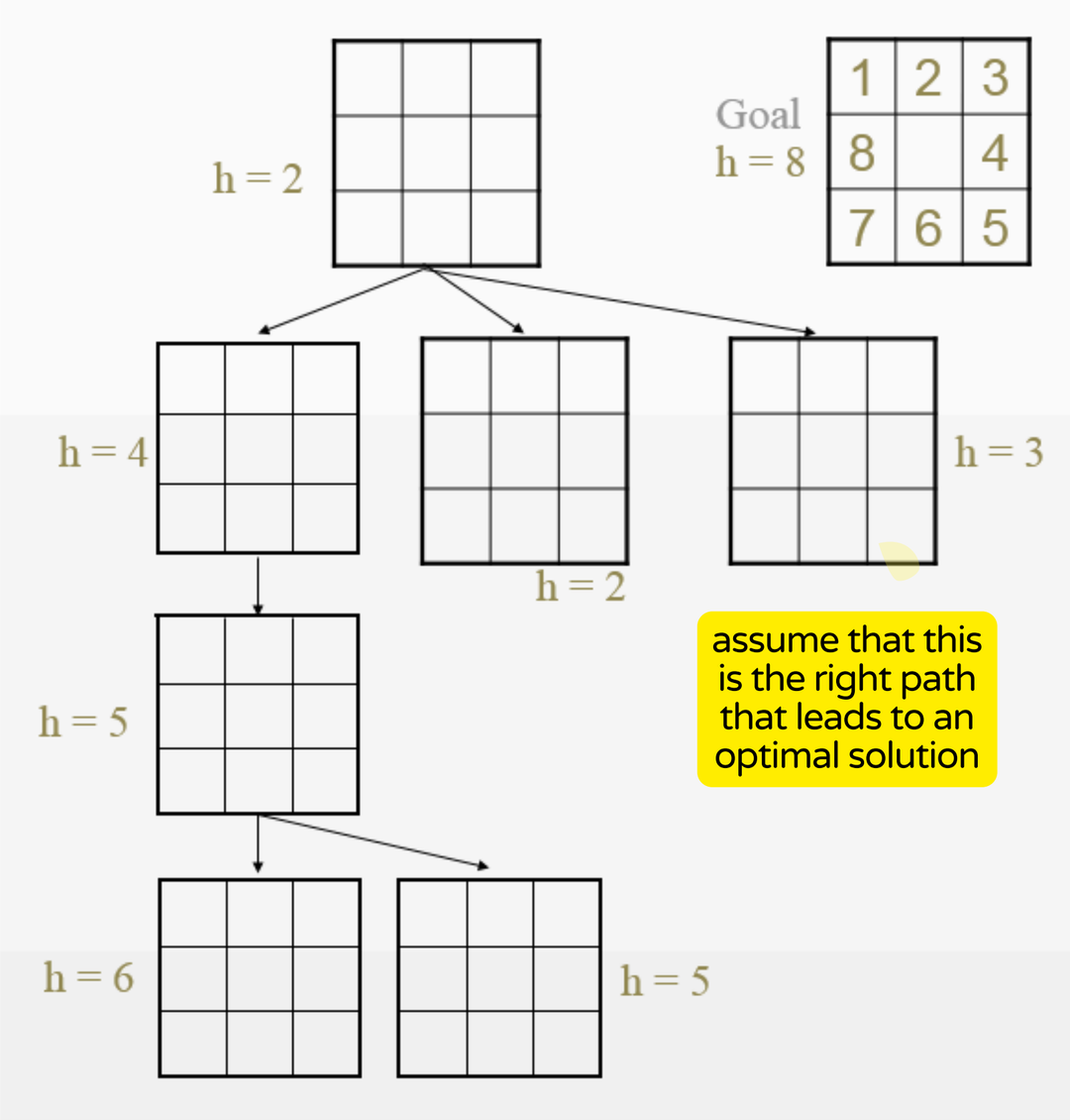


all states leading from it have equal value

- SAHC has also faced a dead end.
- All heuristic costs are the same.
- This problem is known as plateau.

OPS! IT TOOK THE WRONG PATH!

STEEPEST ASCEND HILL CLIMBING



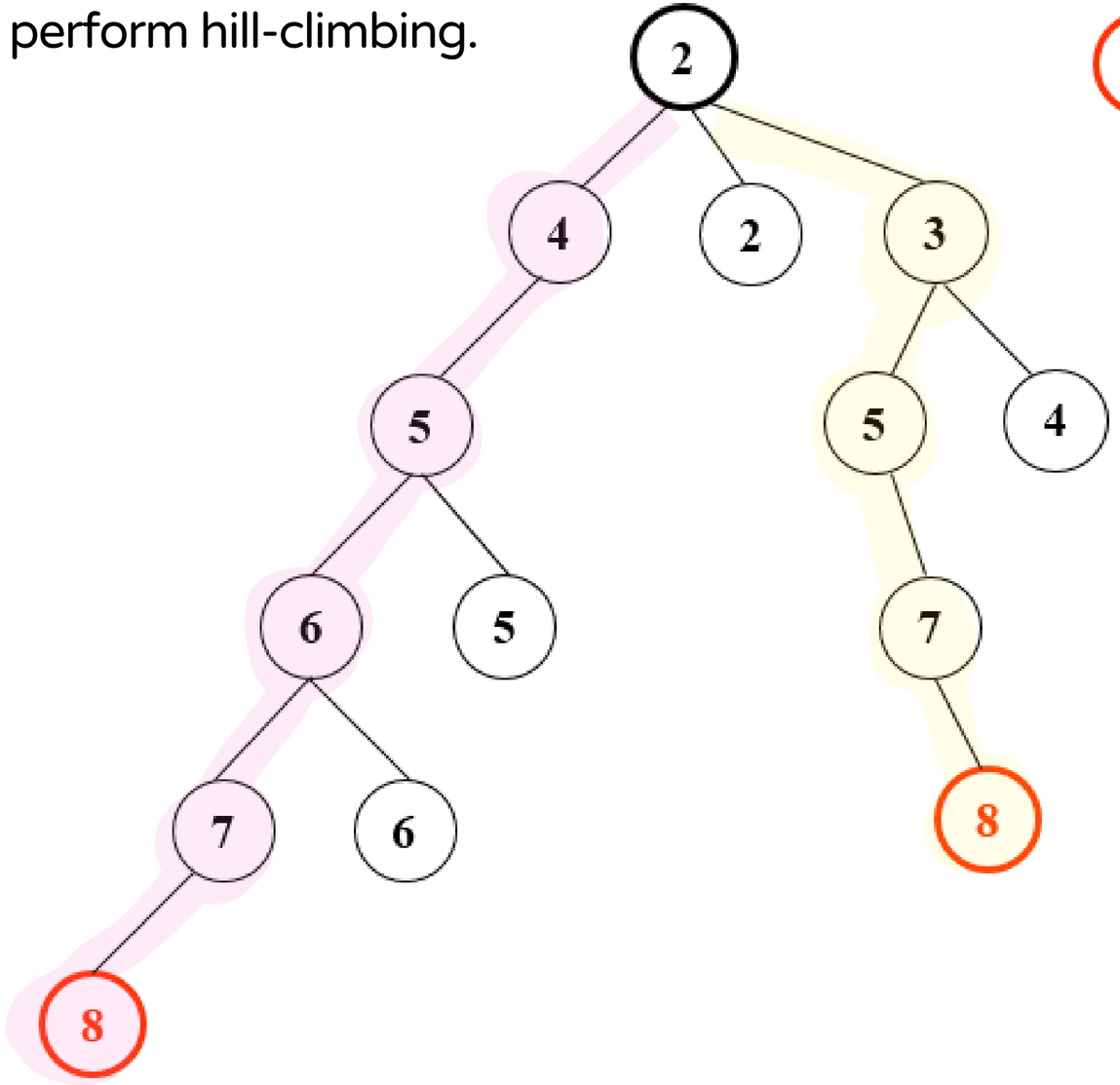
- The actual path that the search took in this example leads to a sub-optimal solution (longer path).
- The problem is known as ridge.

**Remark: assumption only. Not based on the actual problem

OPS! IT TOOK THE WRONG PATH!

STEEPEST ASCEND HILL CLIMBING

Ridge may happen when perform hill-climbing.



8 goal

optimal path

sub-optimal path

LIMITATIONS OF HILL CLIMBING

SUMMARY

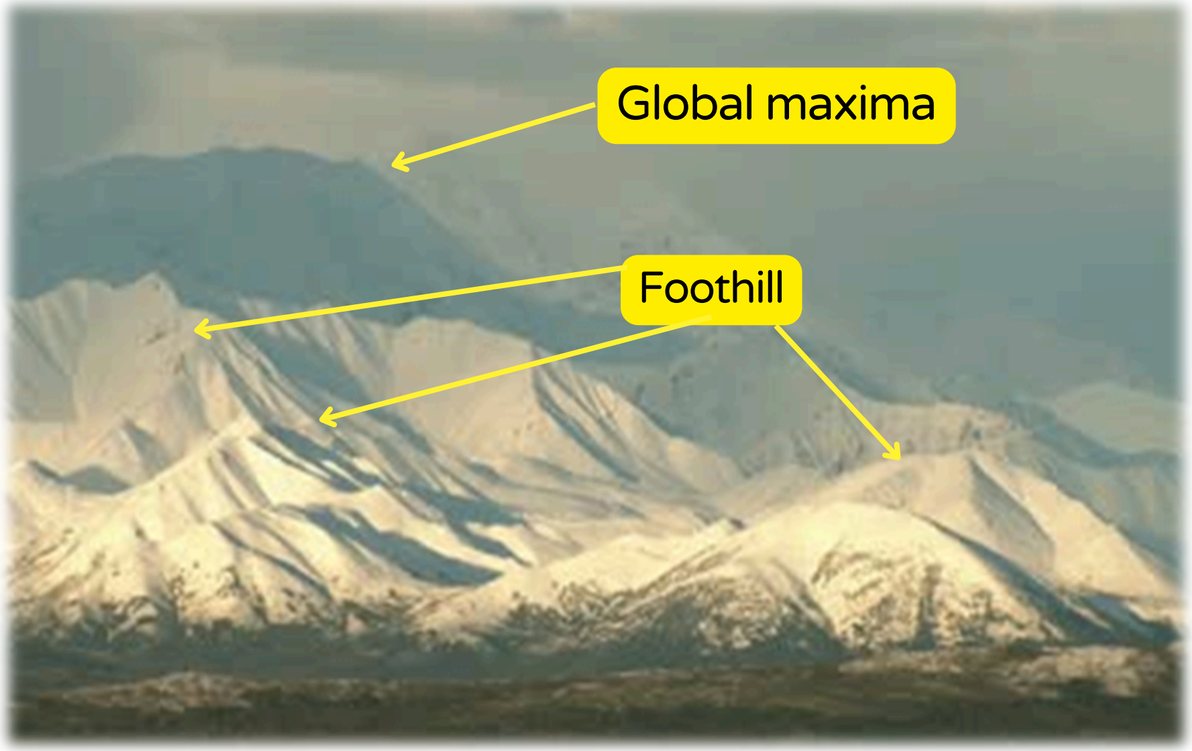
Hill climbing may encounter local maximum, which are

1. Foothills
2. Plateau
3. Ridges

LIMITATIONS OF HILL CLIMBING

Hill climbing may encounter local maximum, which are

1. Foothills
2. Plateau
3. Ridges

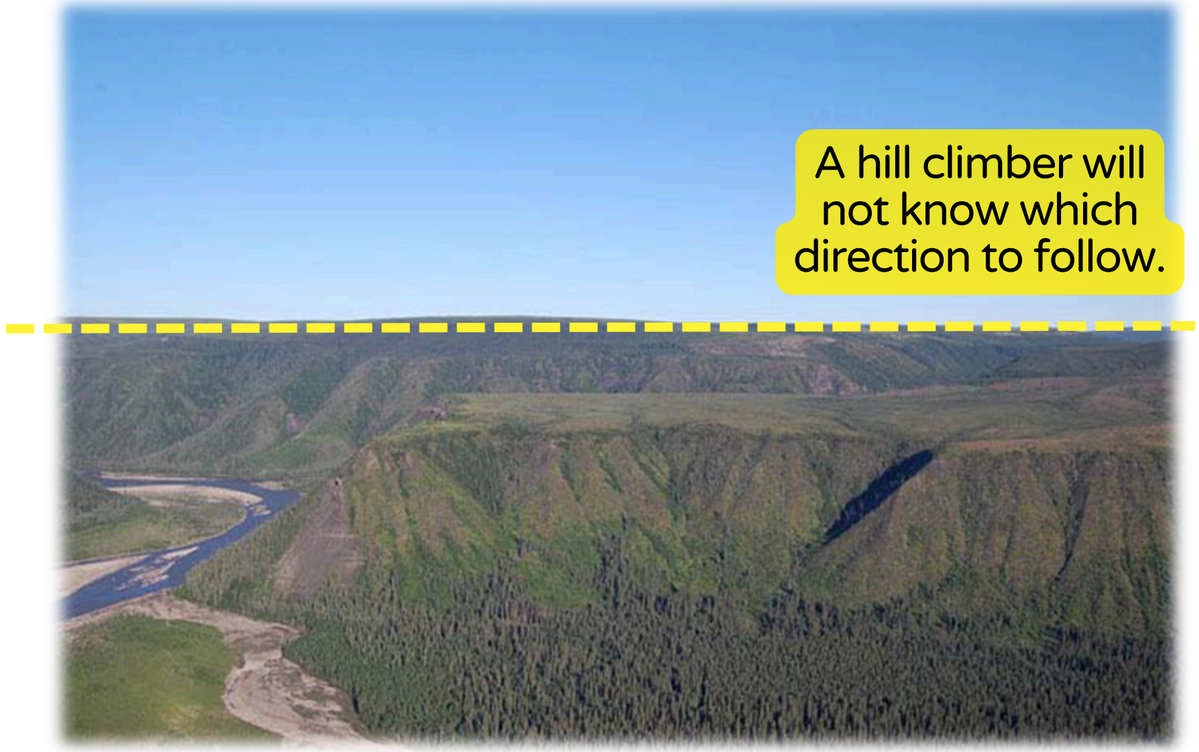


A simple hill climber will never find a solution though one exists.

LIMITATIONS OF HILL CLIMBING

Hill climbing may encounter local maximum, which are

1. Foothills
2. Plateau
3. Ridges

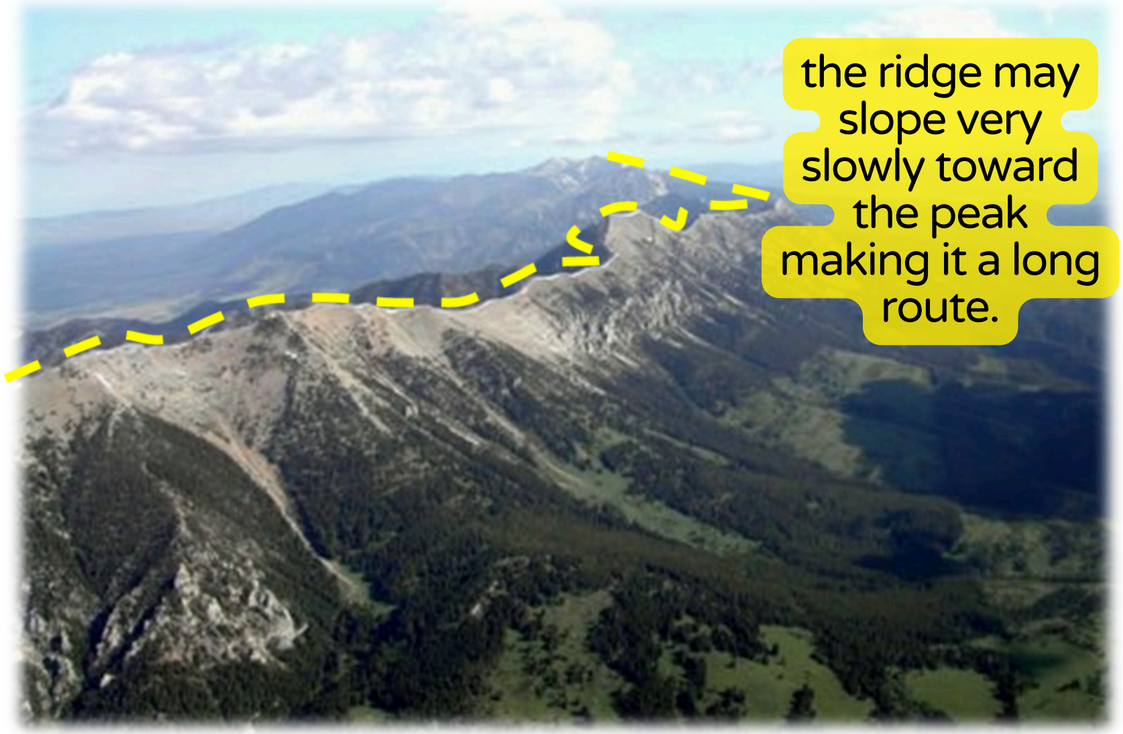


To proceed, the search algorithm may conduct a random walk.

LIMITATIONS OF HILL CLIMBING

Hill climbing may encounter local maximum, which are

1. Foothills
2. Plateau
3. Ridges



An area of state space bordered by lower values in an descending search (or higher values in ascending search)

DESIGNING A HEURISTIC FUNCTION

1. Local Heuristic
   - less combinatorially explosive (less complex)
   - possibly very inefficient and ineffective

2. Global Heuristic
   - consider the global information
   - possibly more efficient than local heuristic

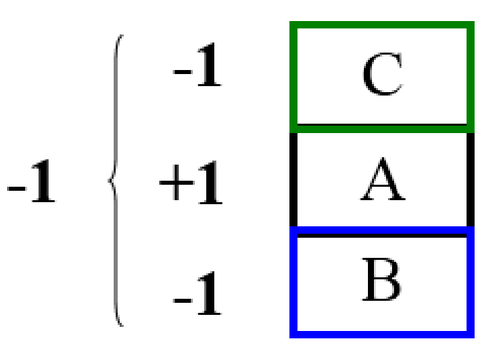
There is NO GUARANTEE that a heuristic can always produce a solution/optimal solution

LOCAL HEURISTIC

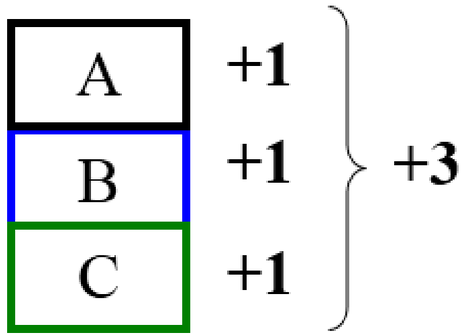
EASY TO DESIGN

Heuristic:
IF
The block is rest on the right place, Add 1.
ELSE
Minus 1.

INITIAL STATE



GOAL STATE

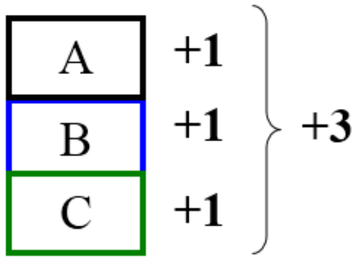


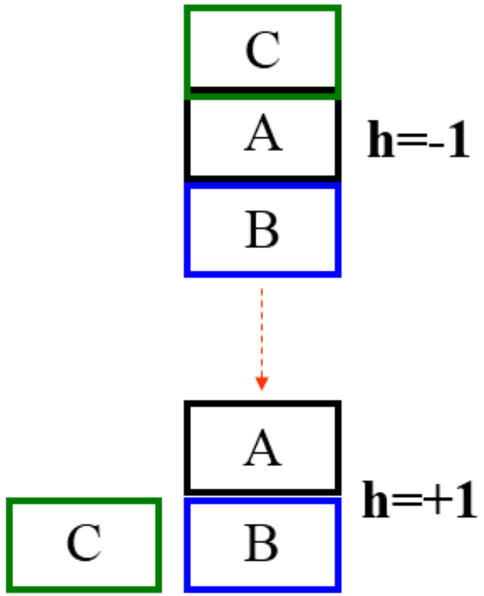
LOCAL HEURISTIC

SAHC

Heuristic:
IF
The block is rest on the right place, Add 1.
ELSE
Minus 1.

GOAL STATE



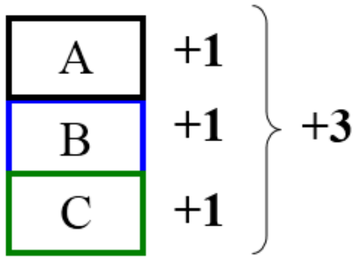


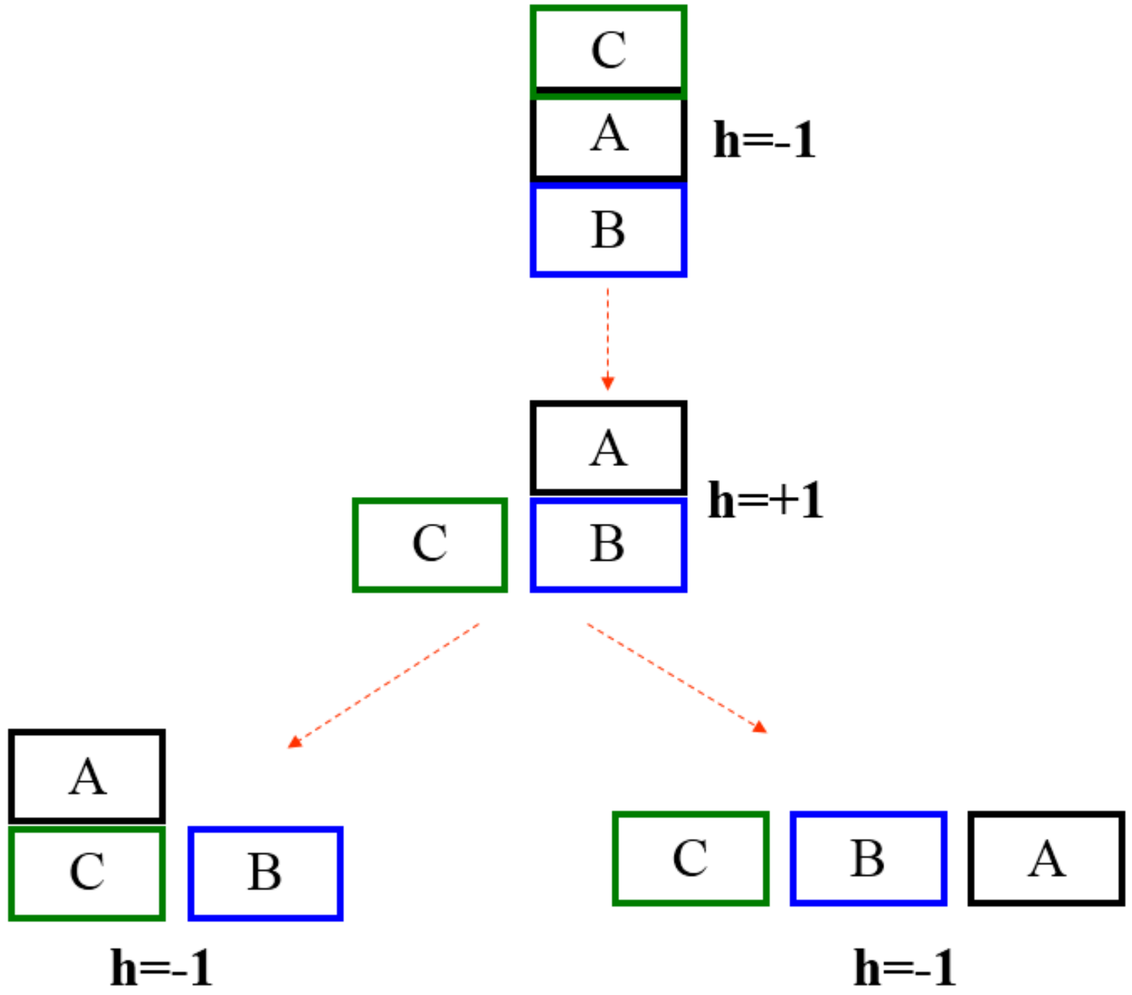
LOCAL HEURISTIC

SAHC

Heuristic:
IF
The block is rest on the right place, Add 1.
ELSE
Minus 1.

GOAL STATE



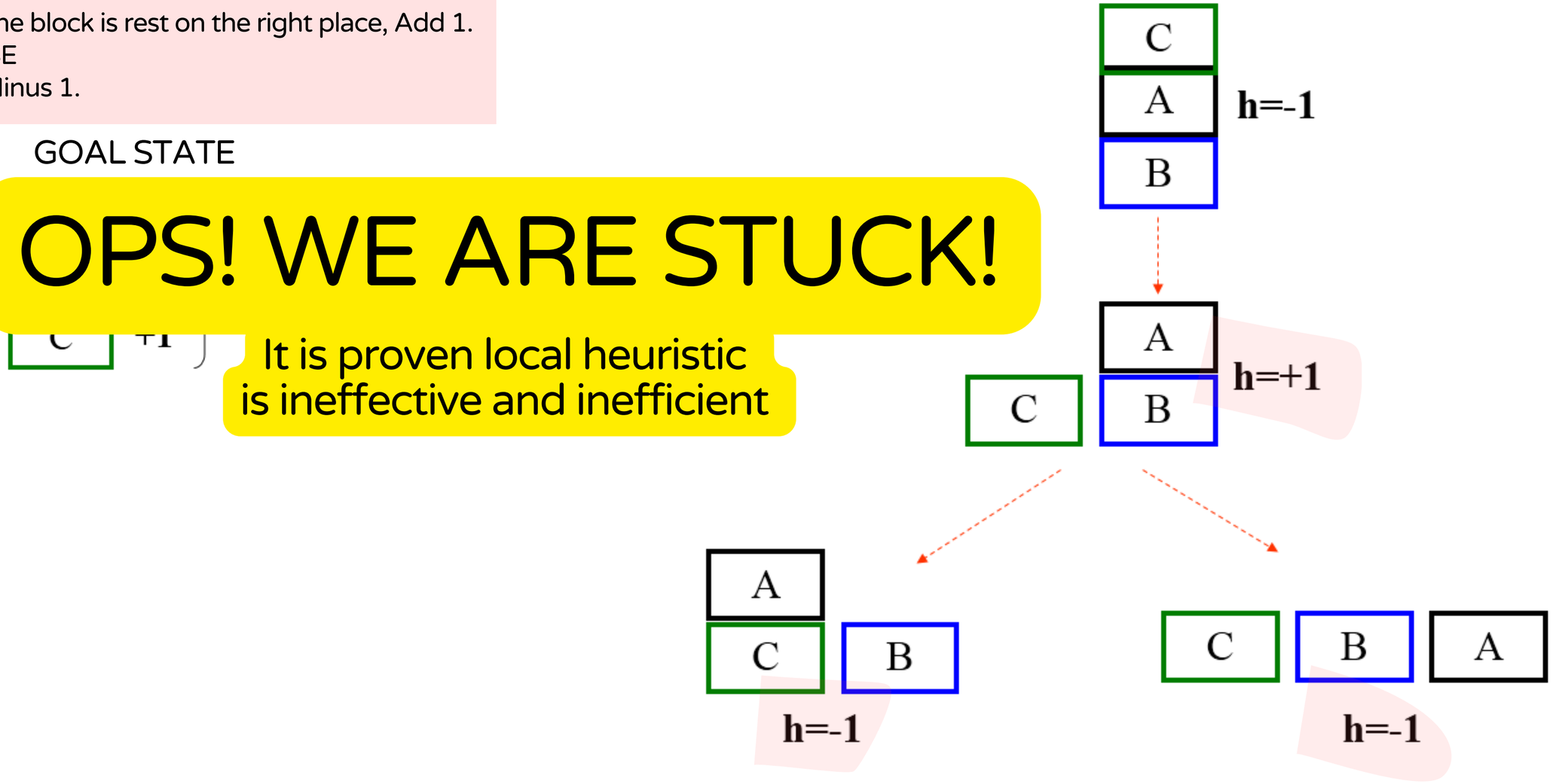


LOCAL HEURISTIC

SAHC

Heuristic:
IF
The block is rest on the right place, Add 1.
ELSE
Minus 1.

GOAL STATE



OPS! WE ARE STUCK!

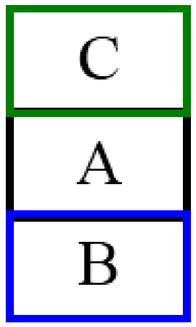
It is proven local heuristic
is ineffective and inefficient

GLOBAL HEURISTIC

Heuristic:

1.For each block that has correct support structure, + 1 for every block in the support structure.

2.For each block that has an incorrect support structure, - 1 for each block in the each block in the existing structure.



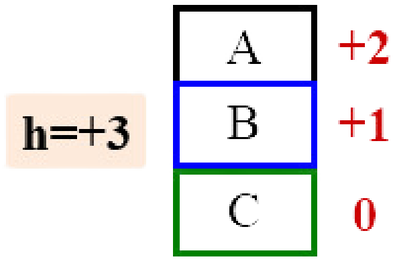
- The support for C is A & B
- The support for A is B
- The support for B is table, but the table is not a block

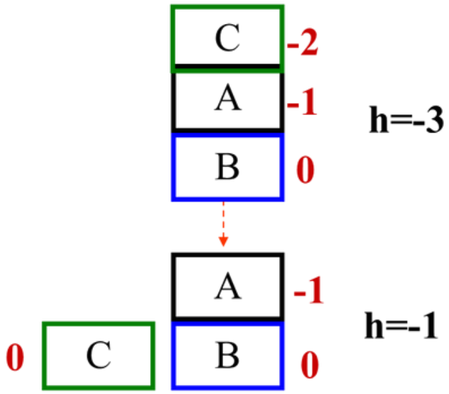
GLOBAL HEURISTIC

SIMPLE HILL CLIMBING

Heuristic:
1.For each block that has correct support structure, + 1 for every block in the support structure.
2.For each block that has an incorrect support structure, - 1 for each block in the each block in the existing structure.

GOAL STATE



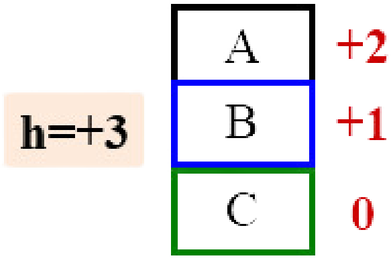


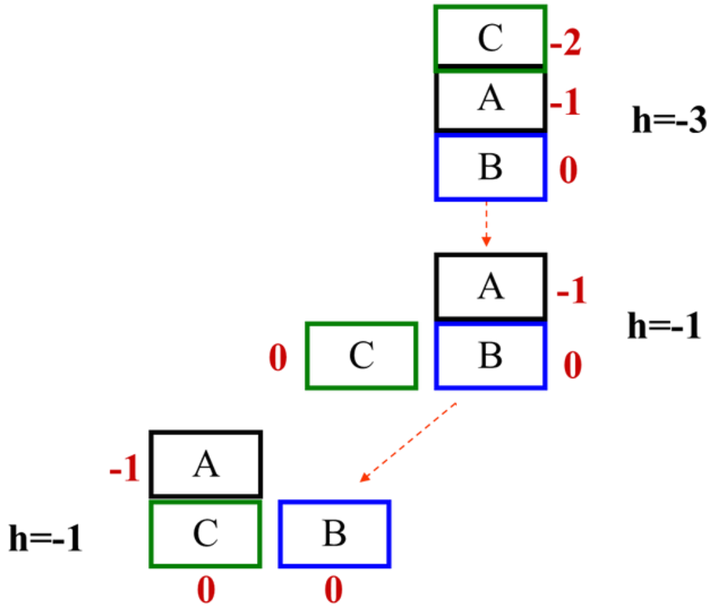
GLOBAL HEURISTIC

SIMPLE HILL CLIMBING

Heuristic:
1.For each block that has correct support structure, + 1 for every block in the support structure.
2.For each block that has an incorrect support structure, - 1 for each block in the each block in the existing structure.

GOAL STATE



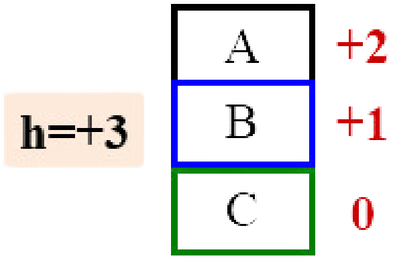


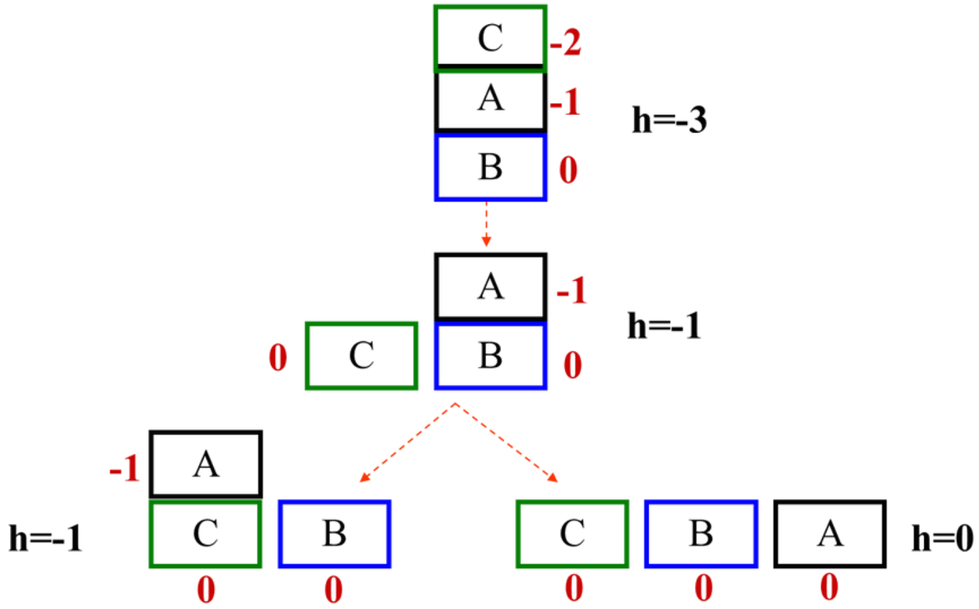
GLOBAL HEURISTIC

SIMPLE HILL CLIMBING

Heuristic:
1.For each block that has correct support structure, + 1 for every block in the support structure.
2.For each block that has an incorrect support structure, - 1 for each block in the each block in the existing structure.

GOAL STATE



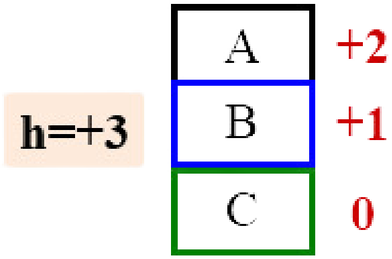


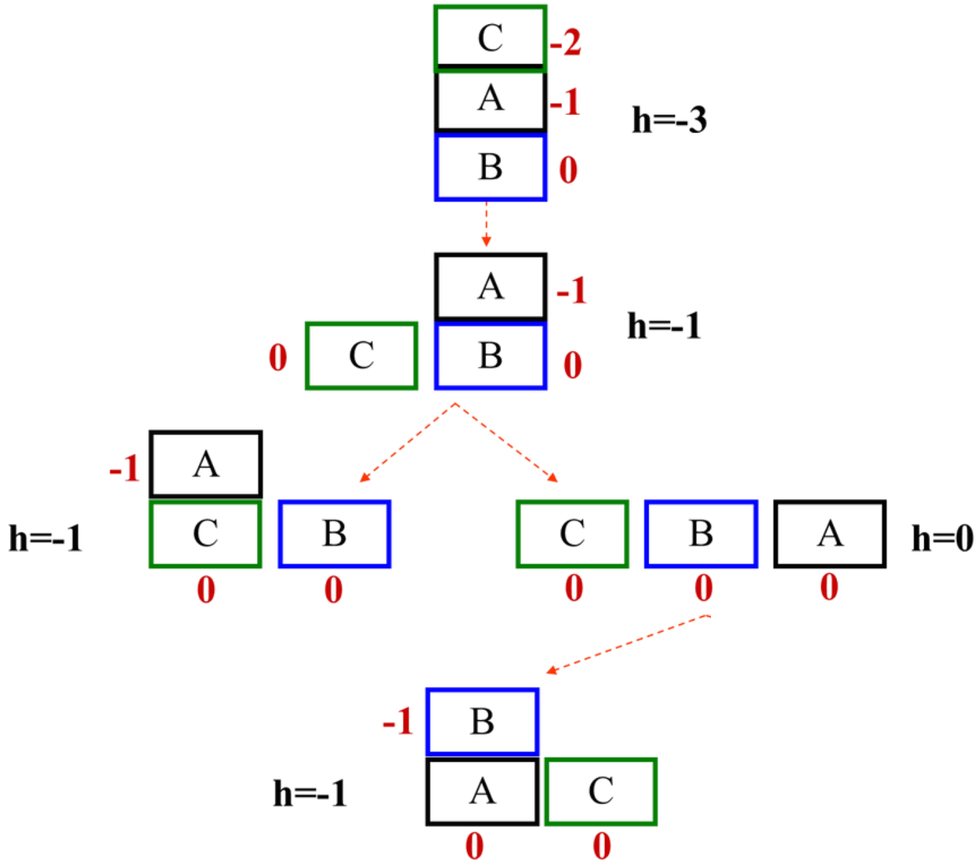
GLOBAL HEURISTIC

SIMPLE HILL CLIMBING

Heuristic:
1.For each block that has correct support structure, + 1 for every block in the support structure.
2.For each block that has an incorrect support structure, - 1 for each block in the each block in the existing structure.

GOAL STATE



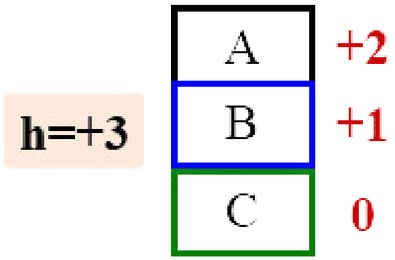


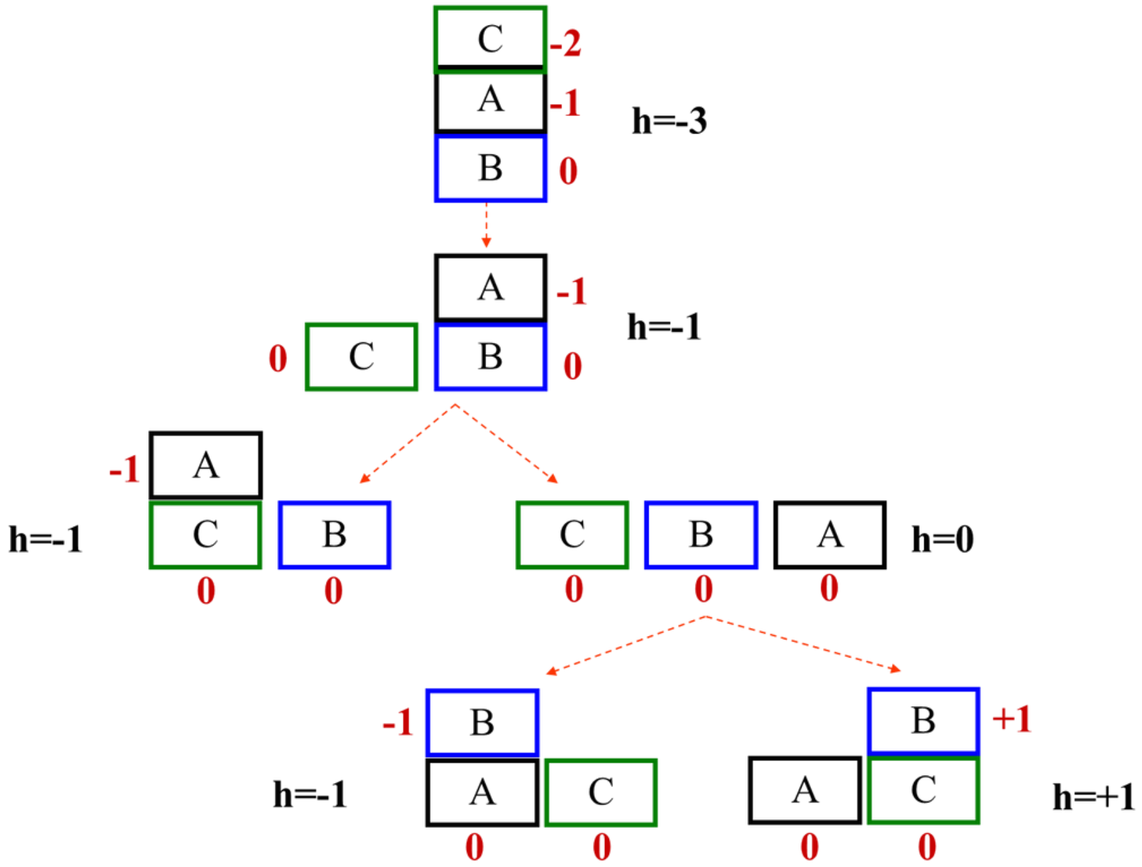
GLOBAL HEURISTIC

SIMPLE HILL CLIMBING

Heuristic:
1.For each block that has correct support structure, + 1 for every block in the support structure.
2.For each block that has an incorrect support structure, - 1 for each block in the each block in the existing structure.

GOAL STATE



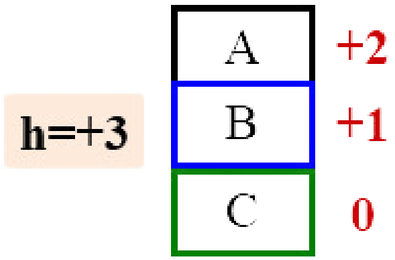


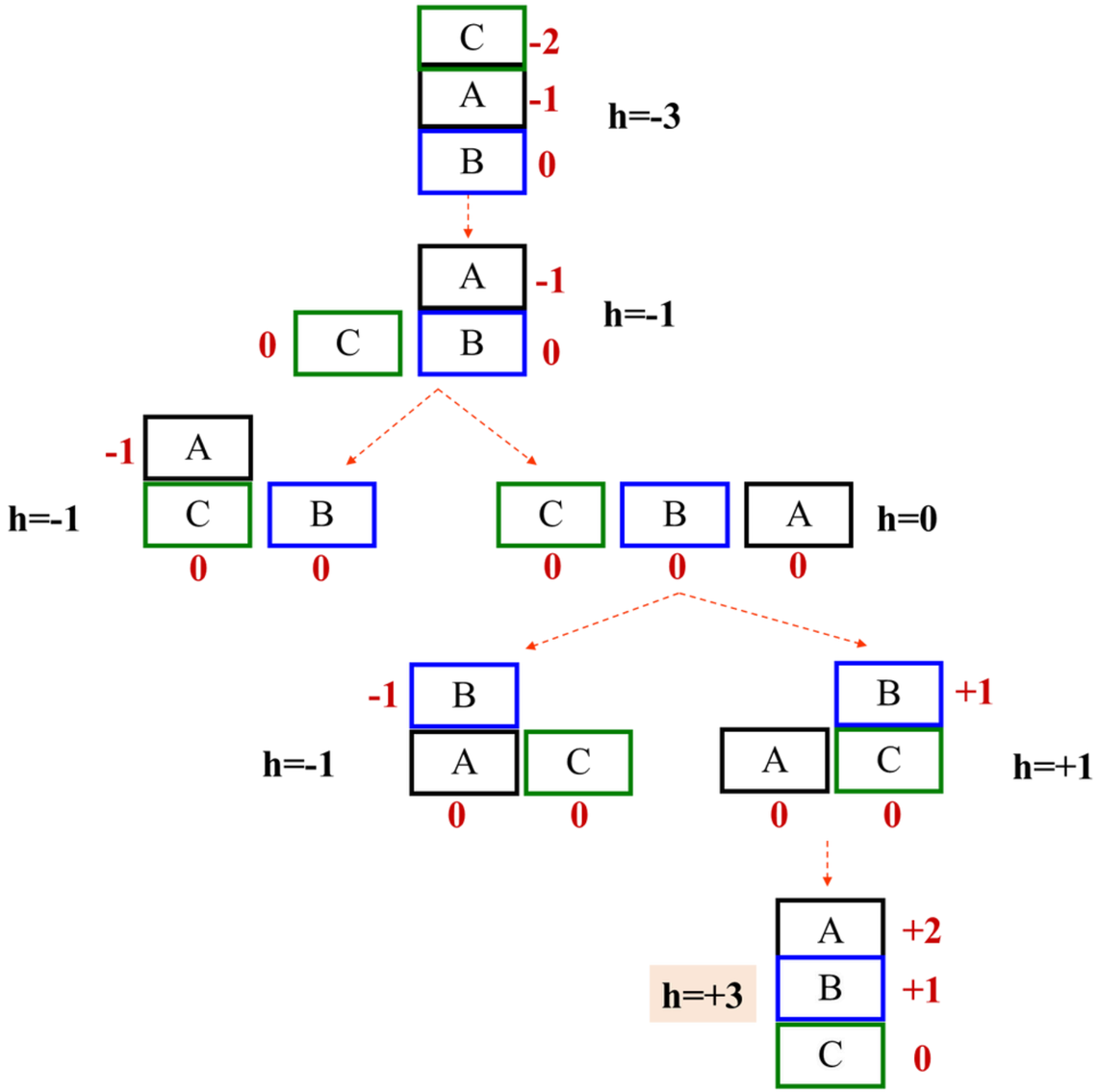
GLOBAL HEURISTIC

SIMPLE HILL CLIMBING

Heuristic:
1.For each block that has correct support structure, + 1 for every block in the support structure.
2.For each block that has an incorrect support structure, - 1 for each block in the each block in the existing structure.

GOAL STATE



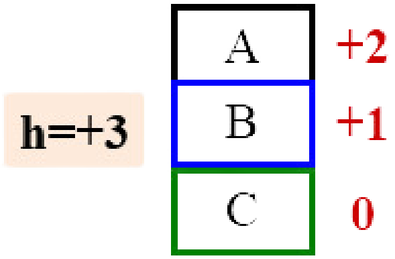


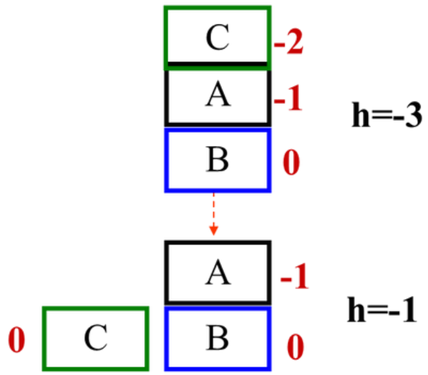
GLOBAL HEURISTIC

SA HILL CLIMBING

Heuristic:
1.For each block that has correct support structure, + 1 for every block in the support structure.
2.For each block that has an incorrect support structure, - 1 for each block in the each block in the existing structure.

GOAL STATE



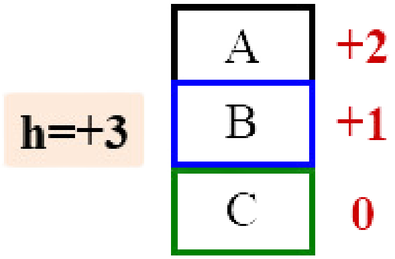


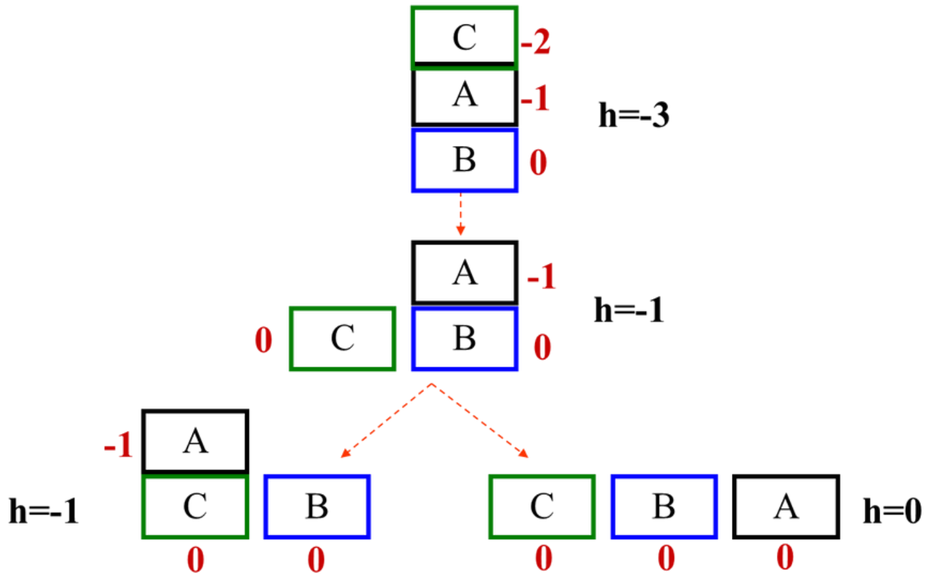
GLOBAL HEURISTIC

SA HILL CLIMBING

Heuristic:
1.For each block that has correct support structure, + 1 for every block in the support structure.
2.For each block that has an incorrect support structure, - 1 for each block in the each block in the existing structure.

GOAL STATE



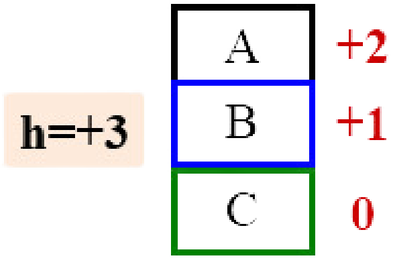


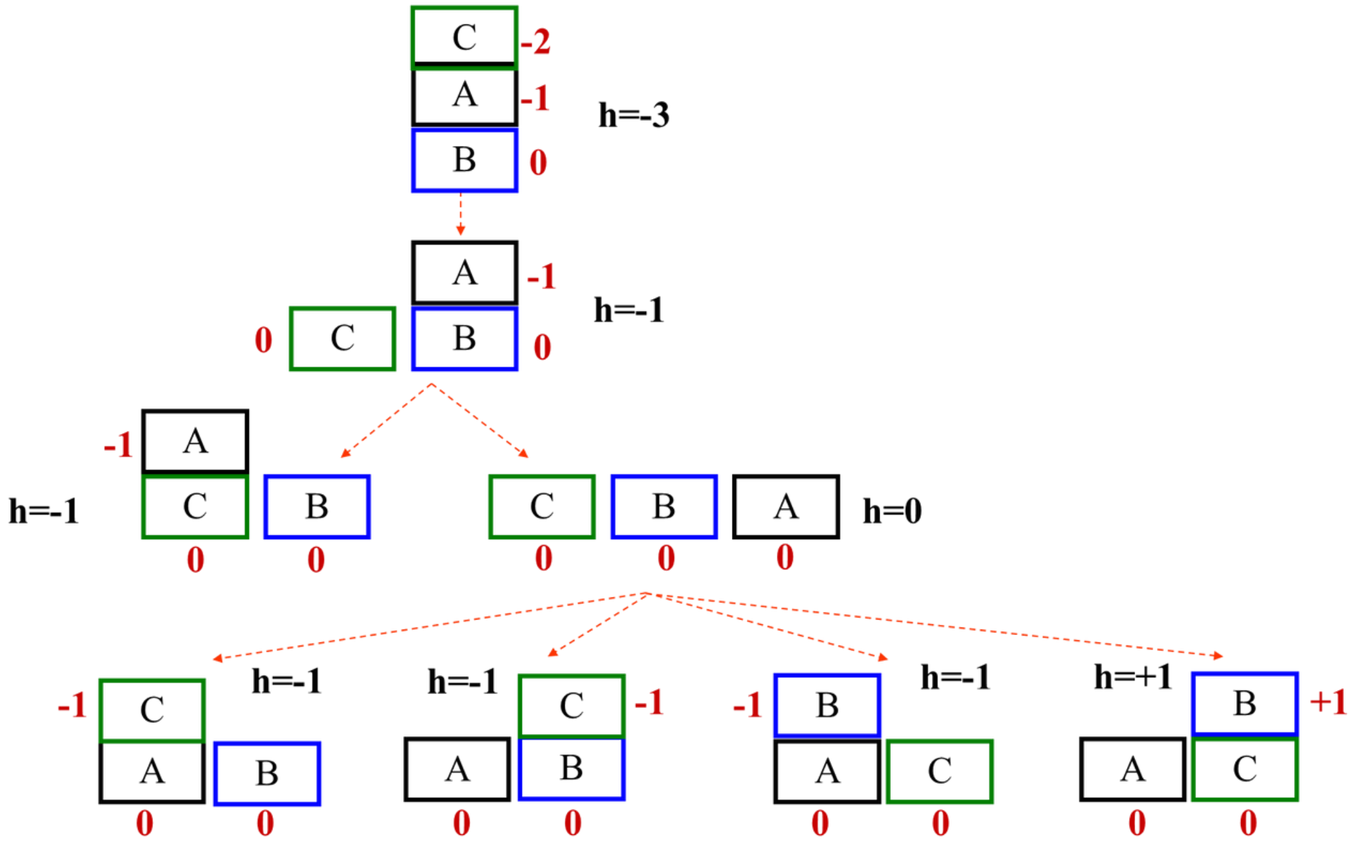
GLOBAL HEURISTIC

SA HILL CLIMBING

Heuristic:
1.For each block that has correct support structure, + 1 for every block in the support structure.
2.For each block that has an incorrect support structure, - 1 for each block in the each block in the existing structure.

GOAL STATE



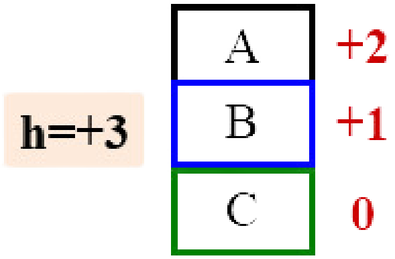


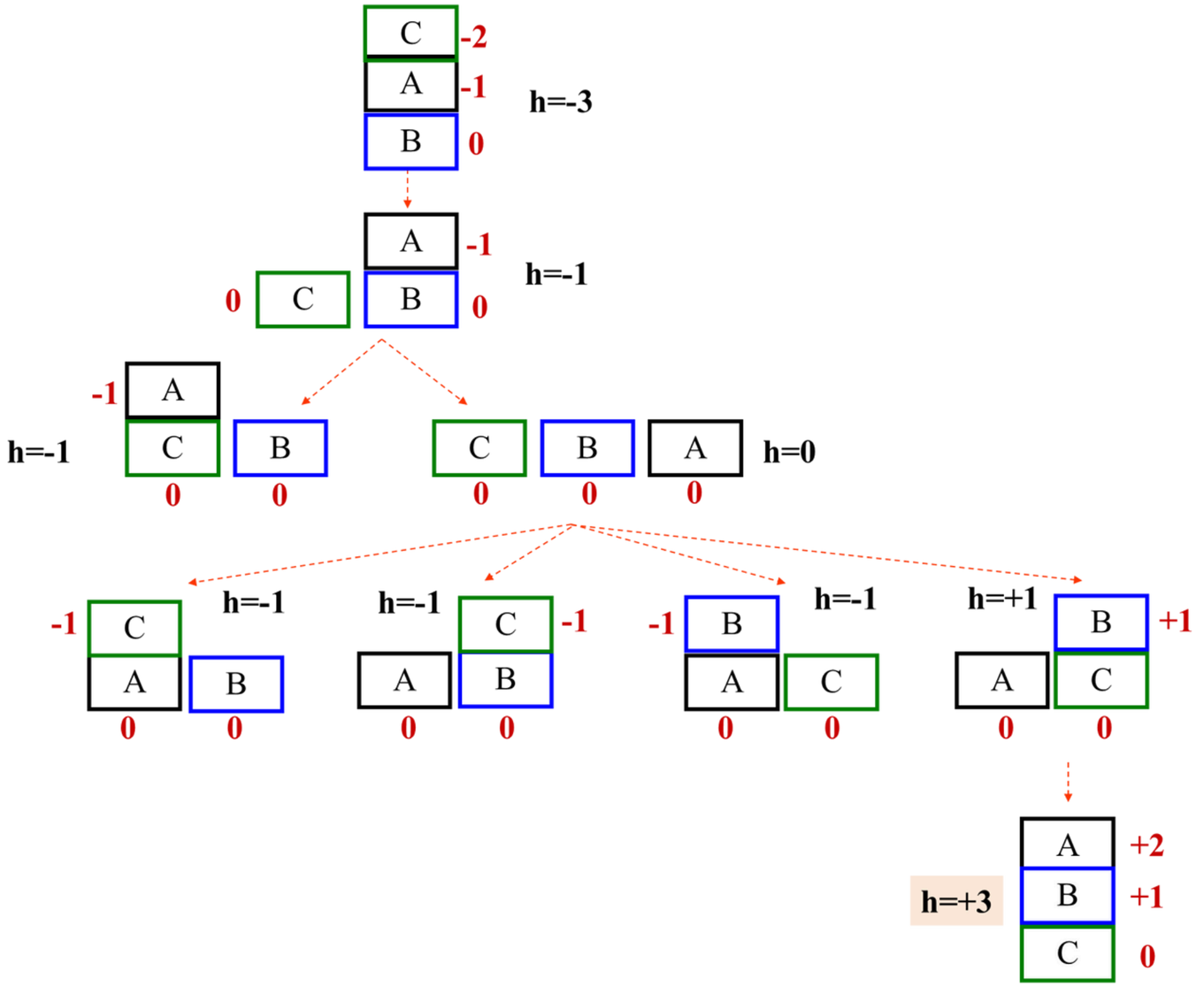
GLOBAL HEURISTIC

SA HILL CLIMBING

Heuristic:
1.For each block that has correct support structure, + 1 for every block in the support structure.
2.For each block that has an incorrect support structure, - 1 for each block in the each block in the existing structure.

GOAL STATE



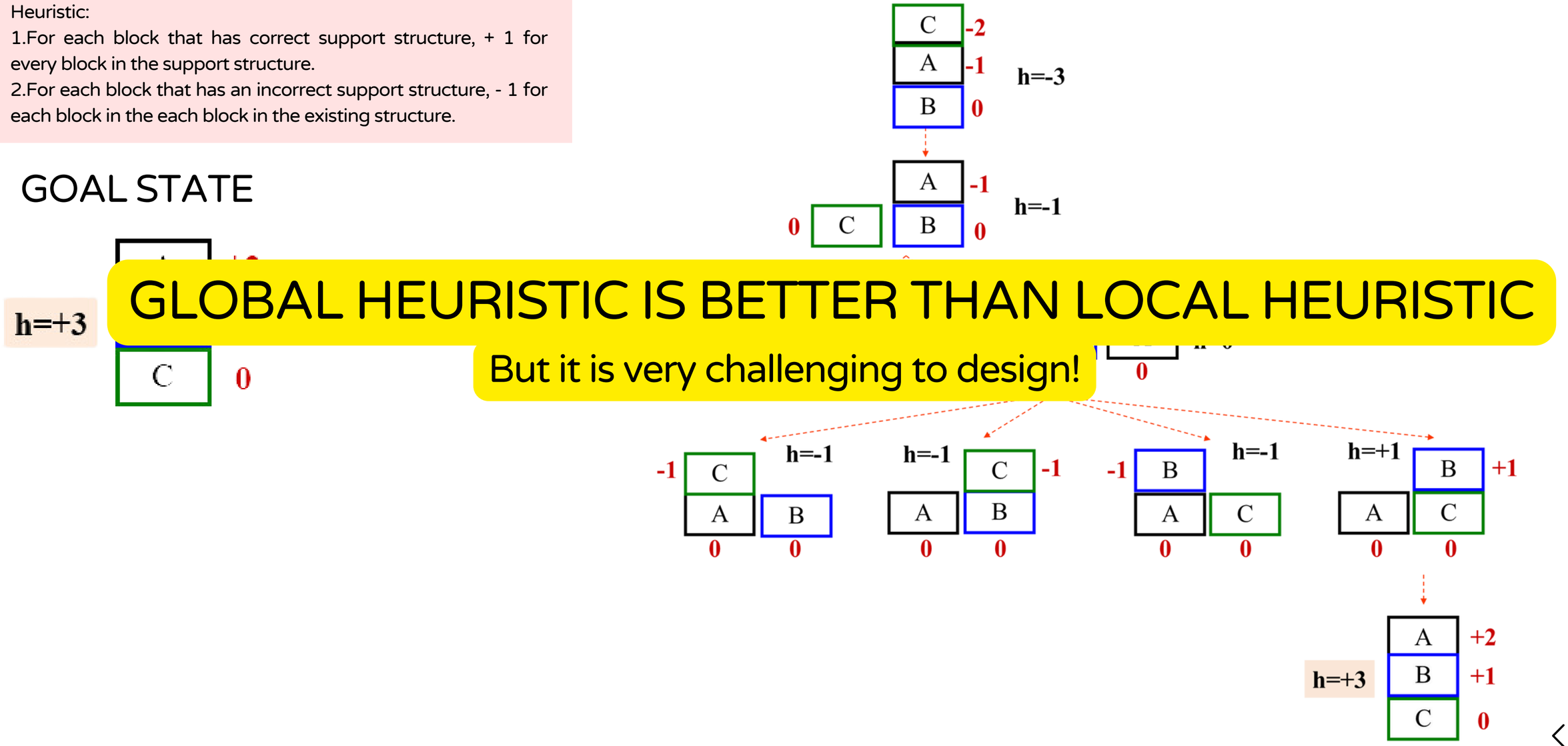


GLOBAL HEURISTIC

SA HILL CLIMBING

Heuristic:
1.For each block that has correct support structure, + 1 for every block in the support structure.
2.For each block that has an incorrect support structure, - 1 for each block in the each block in the existing structure.

GOAL STATE



GLOBAL HEURISTIC IS BETTER THAN LOCAL HEURISTIC

But it is very challenging to design!

SUMMARY HILL CLIMBING

- Hill climbing is not always effective
- Heuristic function not always perfect
- Modify the heuristic function
- Inefficient in a large, rough problem space

BEST FIRST SEARCH

- Depth-first search : without traverse all the branches & allowed to have backtracking
- Breadth-first search : not trapped into a dead-end path & guarantee optimal solution

Best-first search =
Depth-first search + Breadth-first search

BEST FIRST SEARCH

- More flexible
- Use priority queue
- Heuristic is used (to estimate the promising state/path)

BEST FIRST SEARCH

ALGORITHM

Step 1: Place the starting node into the OPEN list.

Step 2: If the OPEN list is empty, Stop and return failure.

Step 3: Remove the node n, from the OPEN list which has the lowest value of h(n) (depending on the designed heuristic function), and places it in the CLOSED list.

Step 4: Expand the node n, and generate the successors of node n.

Step 5: Check each successor of node n, and find whether any node is a goal node or not. If any successor node is goal node, then return success and terminate the search, else proceed to Step 6.

Step 6: For each successor node, algorithm checks for value of h(n), and then check if the node has been in either OPEN or CLOSED list. If the node has not been in both list, then add it to the OPEN list.

Step 7: Return to Step 2.

EXAMPLE

BEST FIRST SEARCH

h(n) = number of tiles at the wrong place.

* h(n) = heuristic cost

Move empty tile follows the sequence of UP, LEFT,RIGHT,DOWN

INITIAL

2	8	3
1	6	4
7		5

h=4

GOAL

1	2	3
8		4
7	6	5

h=0

EXAMPLE

BEST FIRST SEARCH

Move empty tile follows the sequence of UP, LEFT,RIGHT,DOWN

Open=[a4]; Closed=[]; [Open]

a

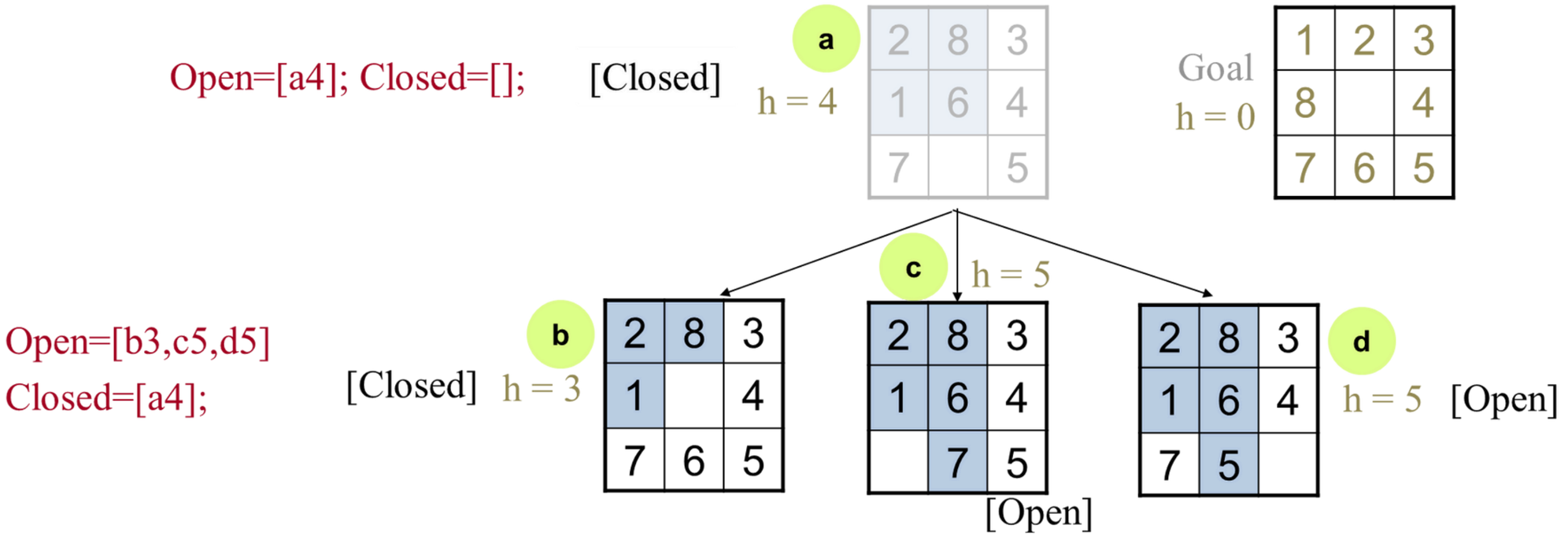
h = 4

2	8	3
1	6	4
7		5

EXAMPLE

BEST FIRST SEARCH

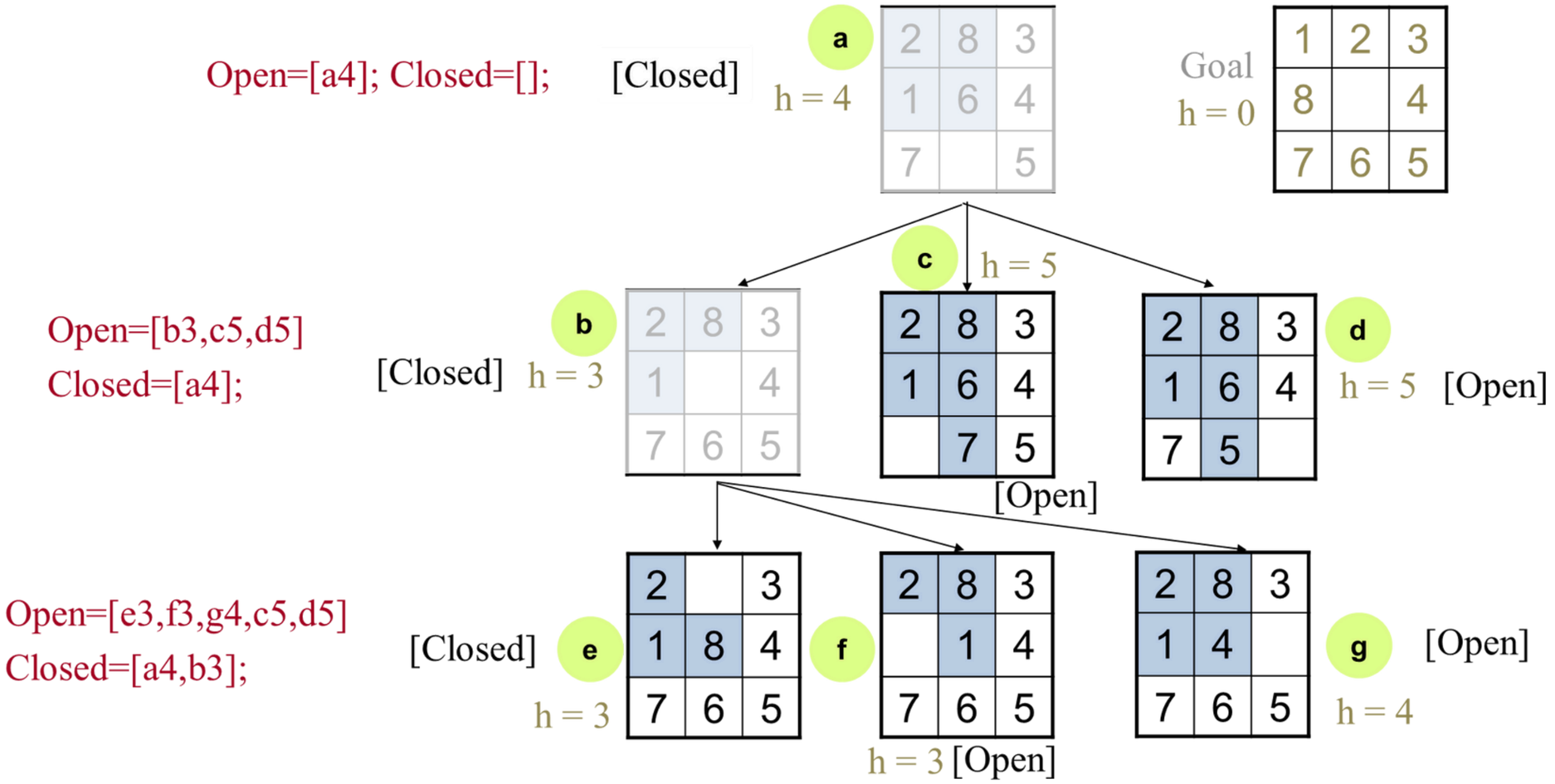
Move empty tile follows the sequence of UP, LEFT,RIGHT,DOWN



EXAMPLE

BEST FIRST SEARCH

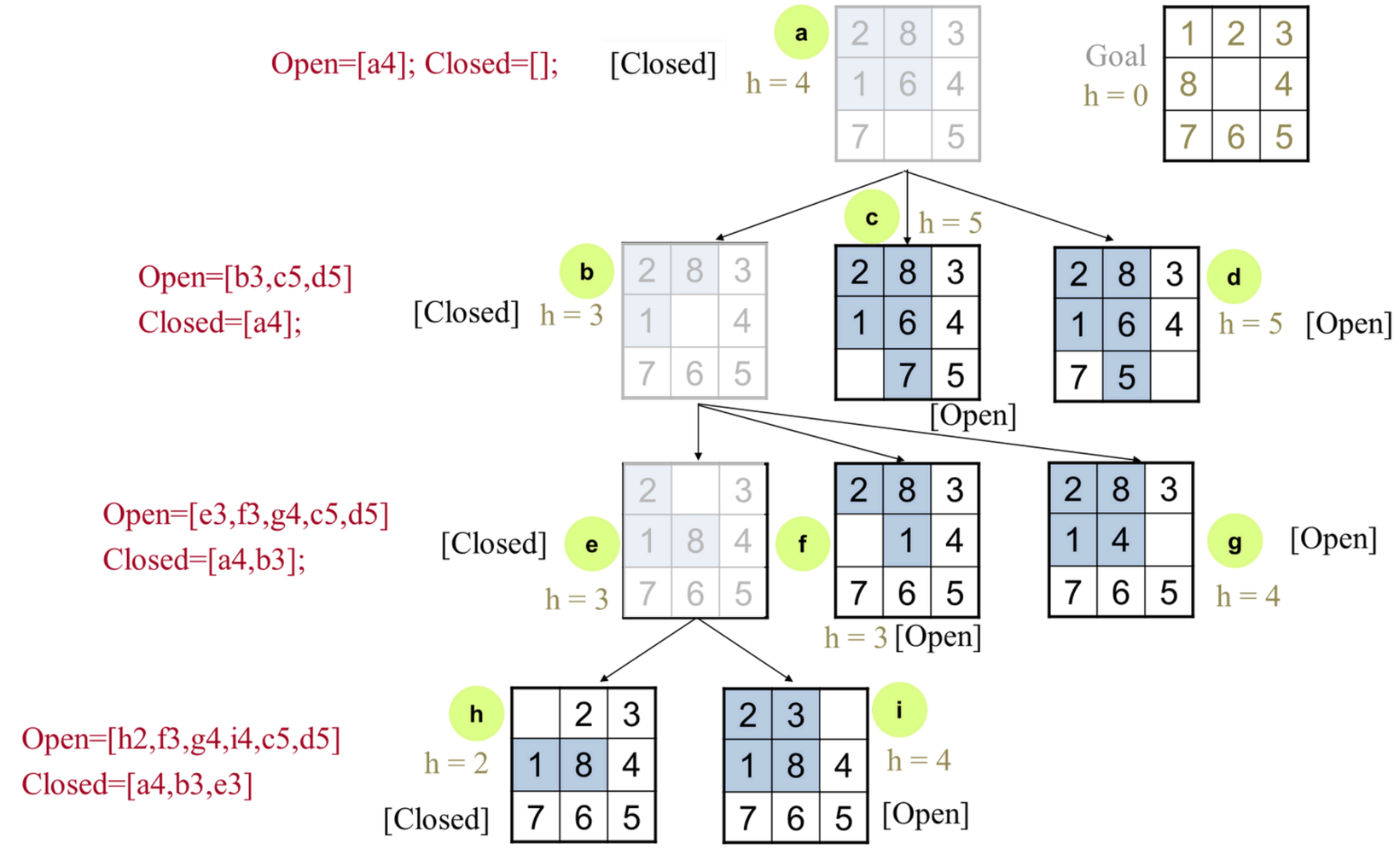
Move empty tile follows the sequence of UP, LEFT,RIGHT,DOWN



EXAMPLE

BEST FIRST SEARCH

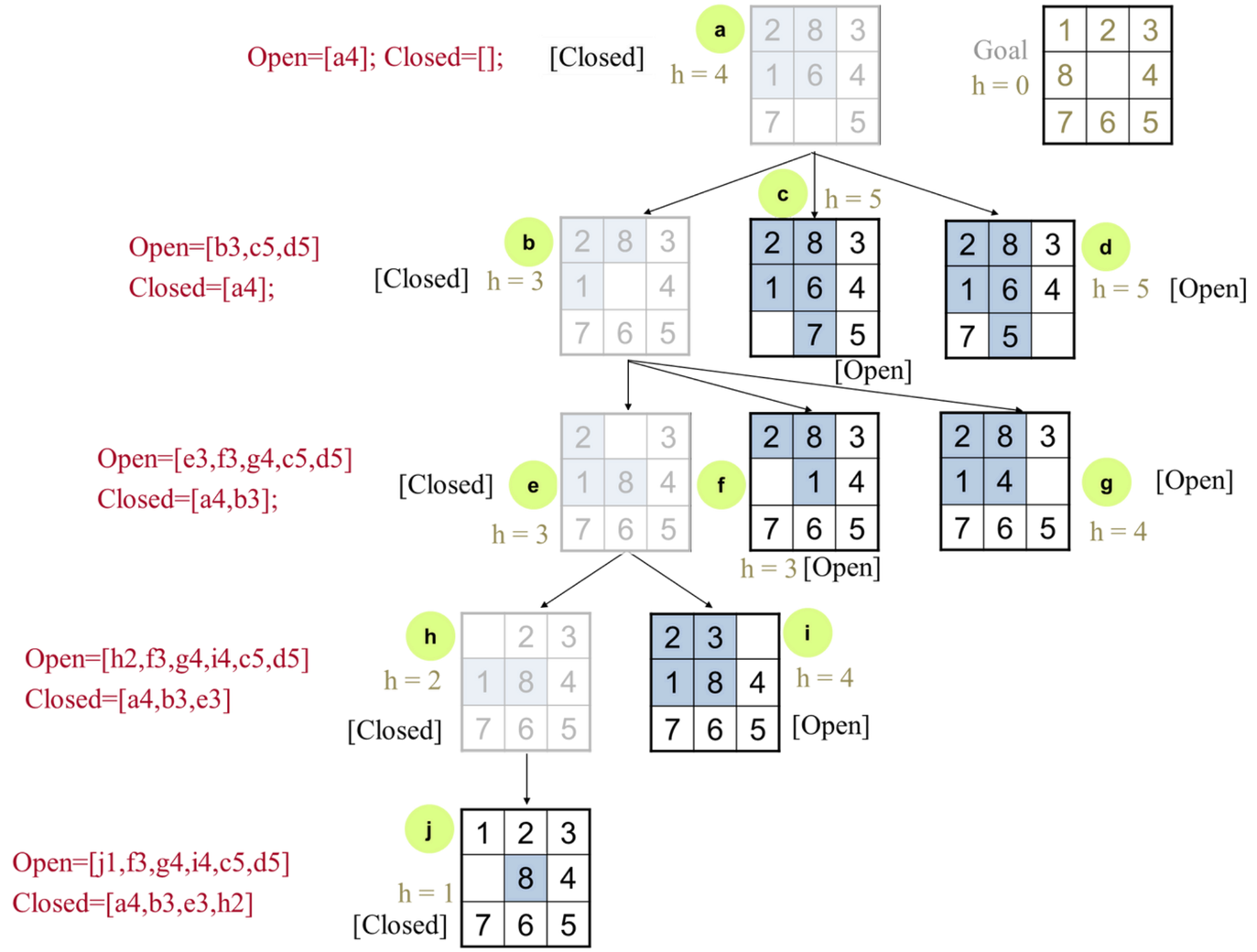
Move empty tile follows the sequence of UP, LEFT,RIGHT,DOWN



EXAMPLE

BEST FIRST SEARCH

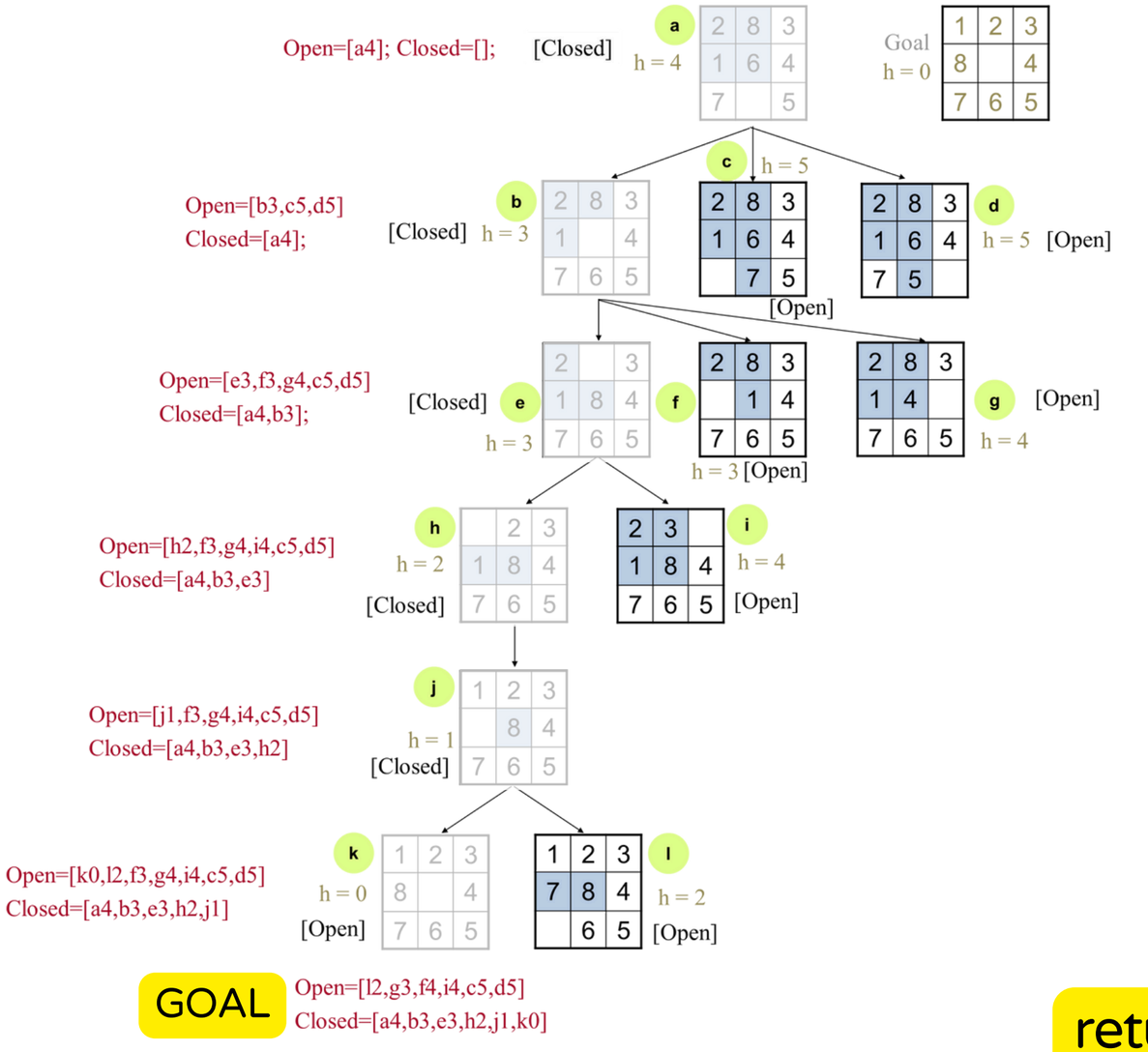
Move empty tile follows the sequence of UP, LEFT,RIGHT,DOWN



EXAMPLE

BEST FIRST SEARCH

Move empty tile follows the sequence of UP, LEFT,RIGHT,DOWN

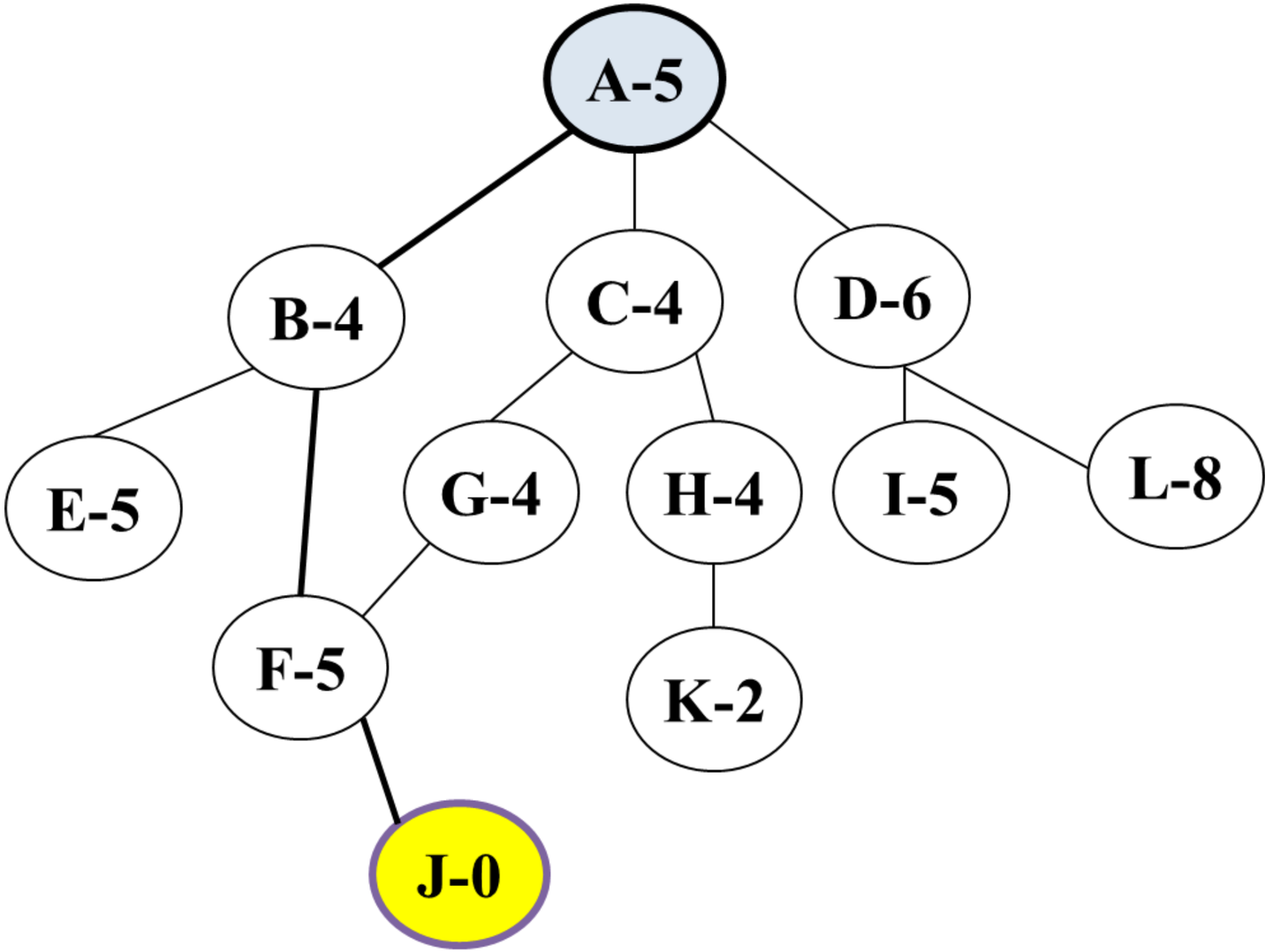


the solution is found;

return path = a->b->e->h->j->k

TRY THIS

BEST FIRST SEARCH



DIFFERENCES BETWEEN SAHC AND BFS

SA Hill-climbing	Best-first search
▶ one move is selected and all others are rejected, which will not be reconsidered. ▶ will quit if there is no better successor /children state than the current state.	▶ one move is selected, but others are still kept around so that they can be revisited later. ▶ will try on other nodes which initially was less promising

A* SEARCH

FUNCTION COST:

f(n) = g(n) + h(n)

where:

g(n) = distance n from the start state and

g(start state) = 0

h(n) = number of tiles out of place, hence

h(goal) = 0 $(h_1(n))$

OR

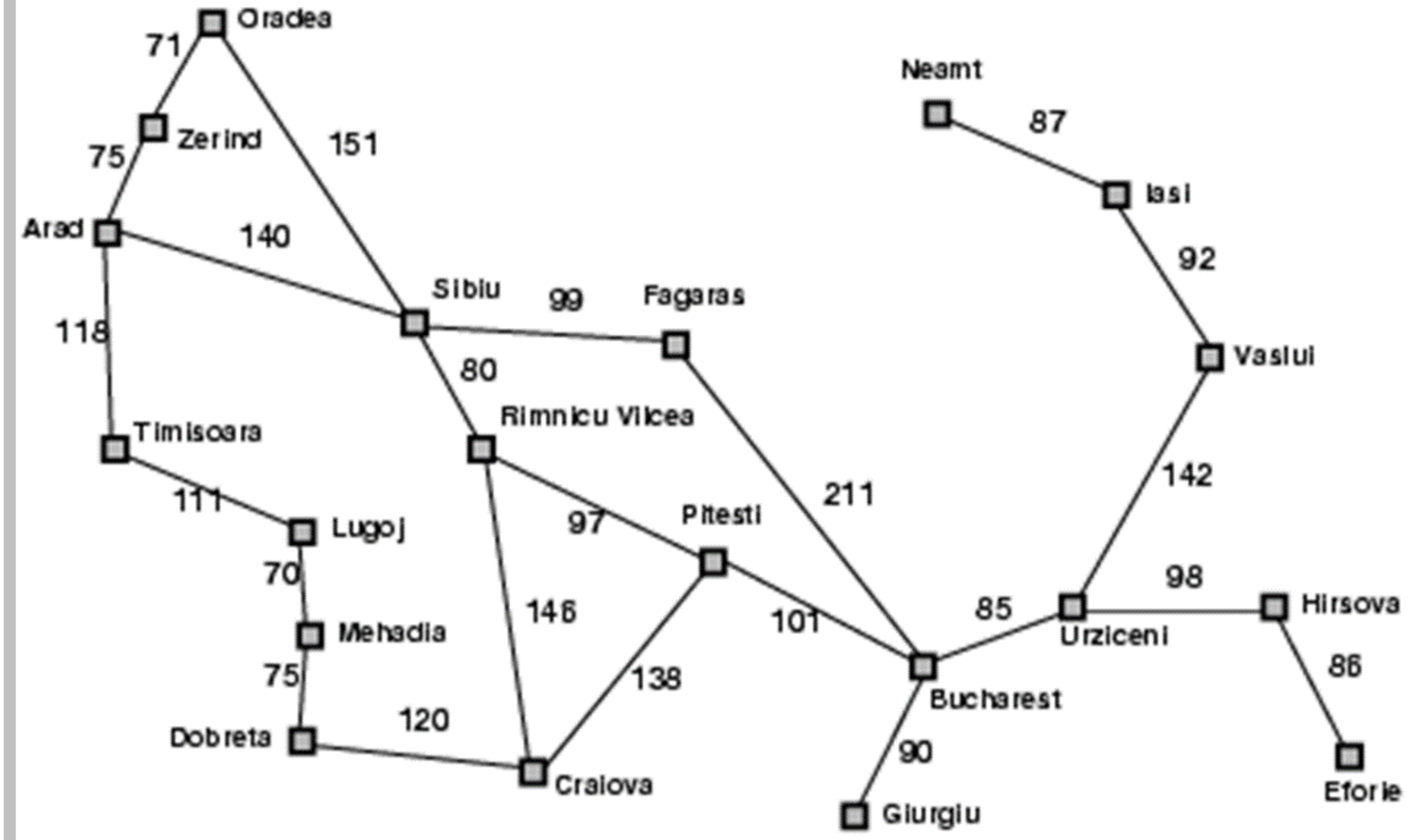
h(n) = the sum of the Manhattan distances of the tiles from their goal positions, hence

h(goal) = 0 $(h_2(n))$

look for minimum(f(n)) and h(n) = 0

EXAMPLE

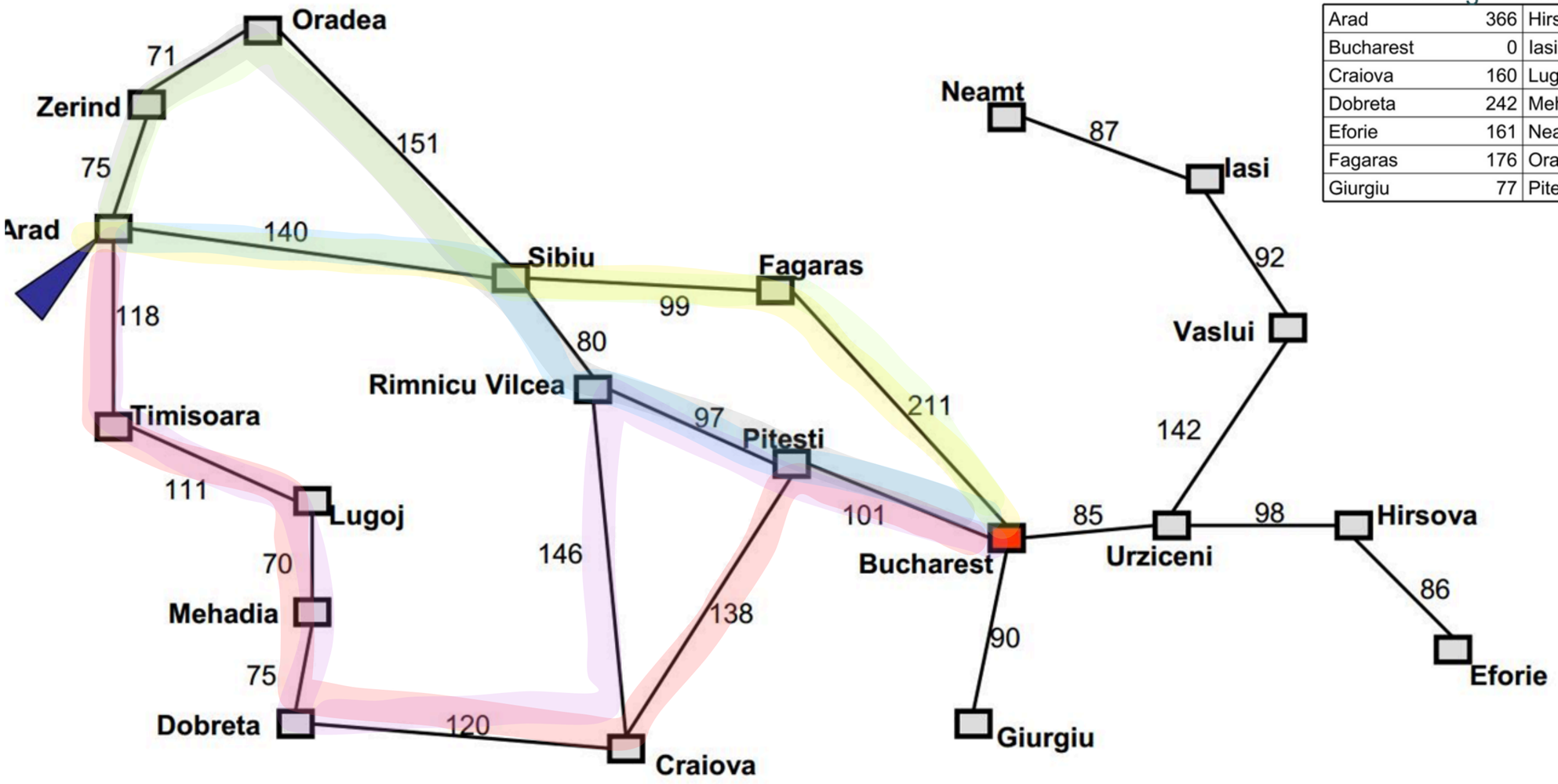
ROMANIA WITH STEP COSTS IN KM



Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

EXAMPLE

ROMANIA WITH STEP COSTS IN KM



Straight line distance to Bucharest

Arad	366	Hirsova	151	Rimnicu Vilcea	193
Bucharest	0	Iasi	226		
Craiova	160	Lugoj	244	Sibiu	253
Dobreta	242	Mehadia	241	Timisoara	329
Eforie	161	Neamt	234	Urziceni	80
Fagaras	176	Oradea	380	Vaslui	199
Giurgiu	77	Pitesti	100	Zerind	374

Heuristic Function:

$f(n) = g(n) + h(n)$

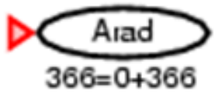
EXAMPLE

Heuristic Function:

$f(n) = g(n) + h(n)$

Open List:

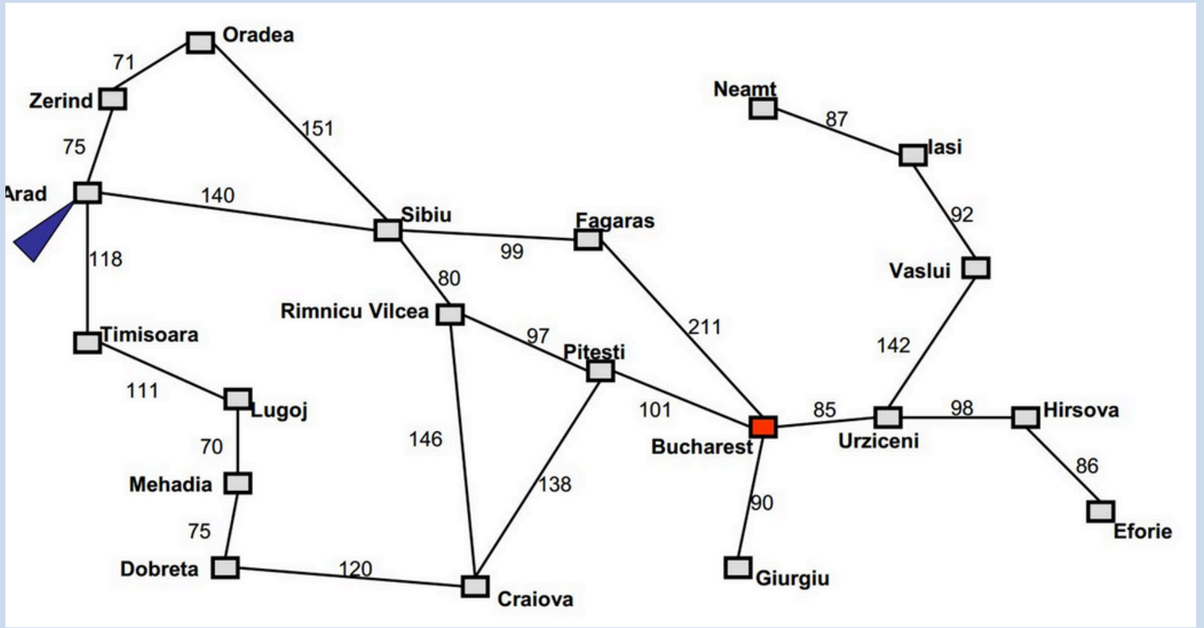
Arad



We start with our initial state Arad. We make a node and add it to the open list. Since it's the only thing on the open list, we expand the node.

Think of the open list as a priority queue (or heap) that sorts the nodes inside of it according to their g(n)+h(n) score.

Arad to Bucharest



Straight line distance to Bucharest

Arad	366	Hirsova	151	Rimnicu Vilcea	193
Bucharest	0	Iasi	226		
Craiova	160	Lugoj	244	Sibiu	253
Dobreta	242	Mehadia	241	Timisoara	329
Eforie	161	Neamt	234	Urziceni	80
Fagaras	176	Oradea	380	Vaslui	199
Giurgiu	77	Pitesti	100	Zerind	374

EXAMPLE

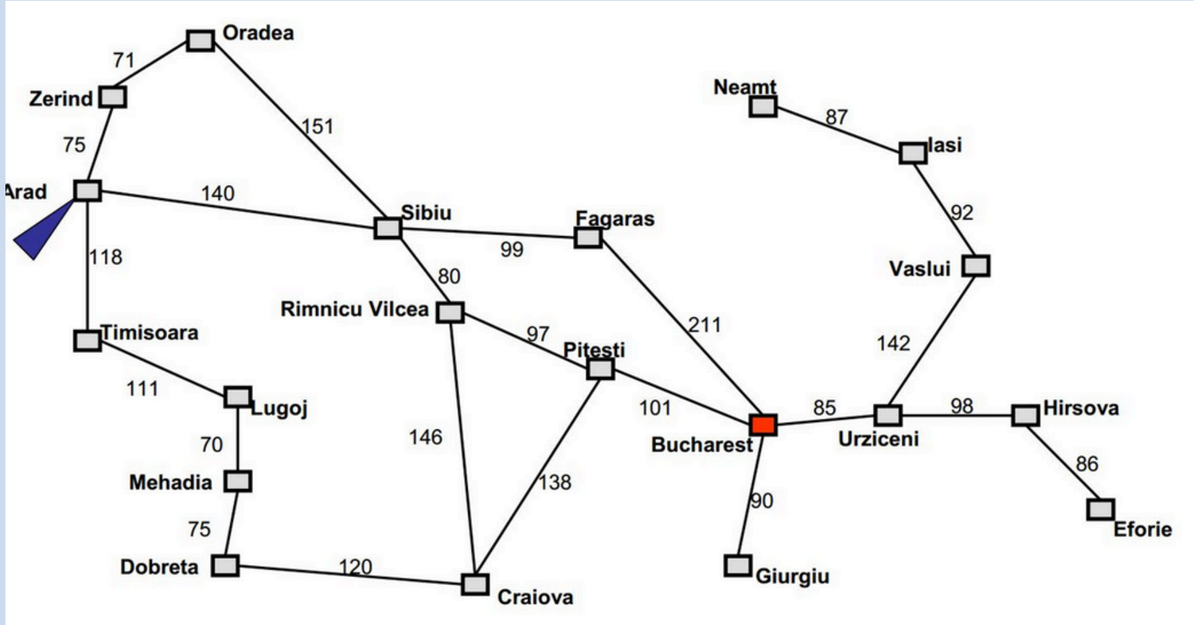
Heuristic Function:

$f(n) = g(n) + h(n)$

Open List:
Sibiu
Timisoara
Zerind

We add the three nodes we found to the open list.
We sort them according to the g(n)+h(n) calculation.

Arad to Bucharest



Straight line distance to Bucharest

Arad	366	Hirsova	151	Rimnicu Vilcea	193
Bucharest	0	Iasi	226		
Craiova	160	Lugoj	244	Sibiu	253
Dobreta	242	Mehadia	241	Timisoara	329
Eforie	161	Neamt	234	Urziceni	80
Fagaras	176	Oradea	380	Vaslui	199
Giurgiu	77	Pitesti	100	Zerind	374

EXAMPLE

Heuristic Function:

$f(n) = g(n) + h(n)$

Open List:

Rimricu Vicea

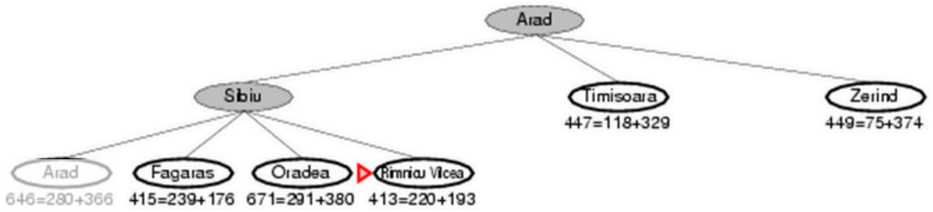
Fagaras

Timisoara

Zerind

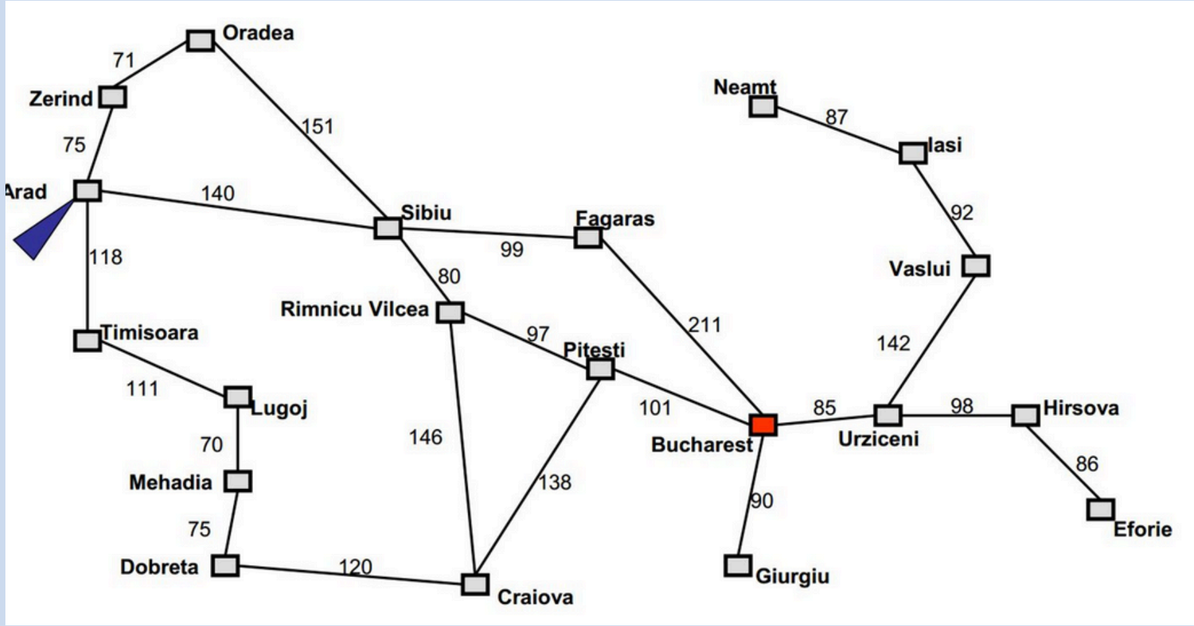
~~Arad~~

Oradea



When we expand Sibiu, we run into Arad again.But we've already expanded this node once; so, we don't add it to the open list again.

Arad to Bucharest



Straight line distance to Bucharest

Arad	366	Hirsova	151	Rimnicu Vilcea	193
Bucharest	0	Iasi	226		
Craiova	160	Lugoj	244	Sibiu	253
Dobreta	242	Mehadia	241	Timisoara	329
Eforie	161	Neamt	234	Urziceni	80
Fagaras	176	Oradea	380	Vaslui	199
Giurgiu	77	Pitesti	100	Zerind	374

EXAMPLE

Heuristic Function:

$f(n) = g(n) + h(n)$

Open List:

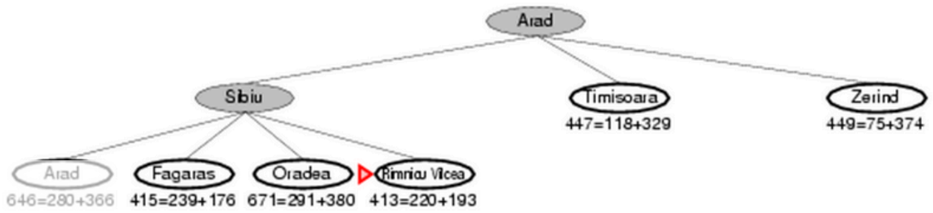
Rimricu Vicea

Fagaras

Timisoara

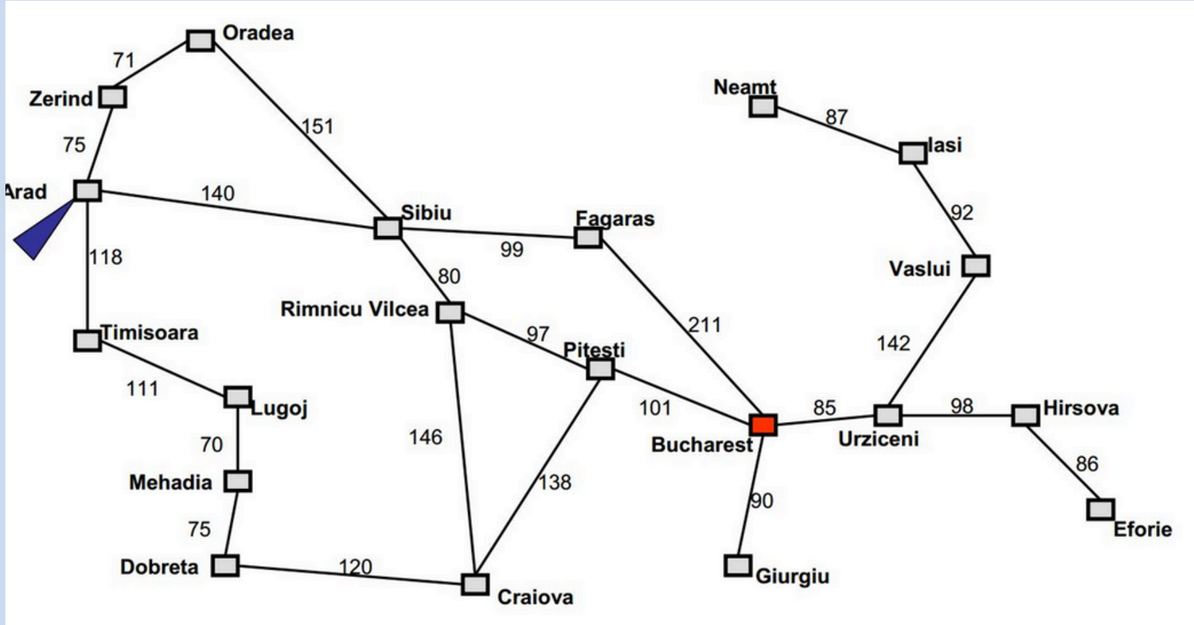
Zerind

Oradea



We see that Rimricu Vicea is at the top of the open list; so, it's the next node we will expand.

Arad to Bucharest



Straight line distance to Bucharest

Arad	366	Hirsova	151	Rimnicu Vilcea	193
Bucharest	0	Iasi	226		
Craiova	160	Lugoj	244	Sibiu	253
Dobreta	242	Mehadia	241	Timisoara	329
Eforie	161	Neamt	234	Urziceni	80
Fagaras	176	Oradea	380	Vaslui	199
Giurgiu	77	Pitesti	100	Zerind	374

EXAMPLE

Heuristic Function:

$f(n) = g(n) + h(n)$

Open List:

Fagaras

Pitesti

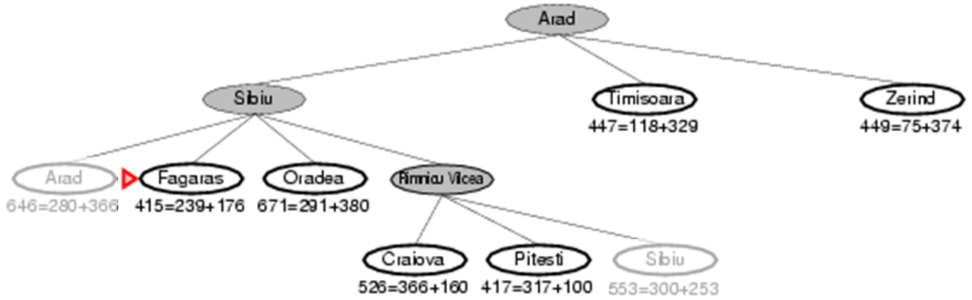
Timisoara

Zerind

Craiova

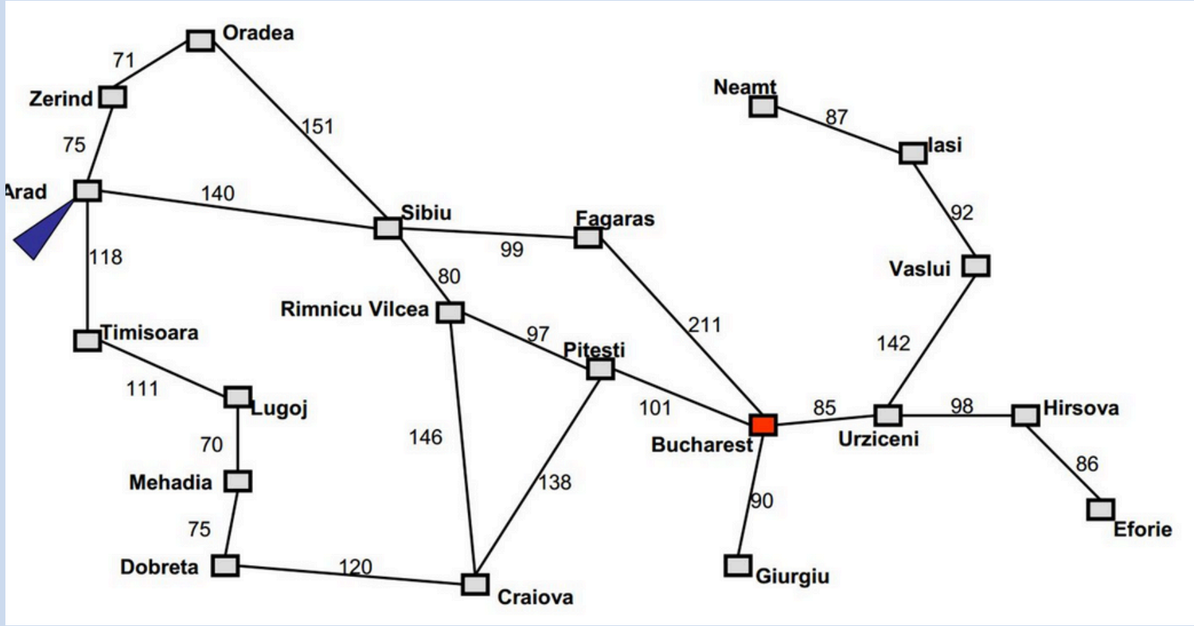
~~Sibiu~~

Oradea



When we expand Rimricu Vicea, we run into Sibiu again.
But we've already expanded this node once; so, we don't add it to the open list again.

Arad to Bucharest



Straight line distance to Bucharest

Arad	366	Hirsova	151	Rimnicu Vilcea	193
Bucharest	0	Iasi	226		
Craiova	160	Lugoj	244	Sibiu	253
Dobreta	242	Mehadia	241	Timisoara	329
Eforie	161	Neamt	234	Urziceni	80
Fagaras	176	Oradea	380	Vaslui	199
Giurgiu	77	Pitesti	100	Zerind	374

EXAMPLE

Heuristic Function:

$f(n) = g(n) + h(n)$

Open List:

Fagaras

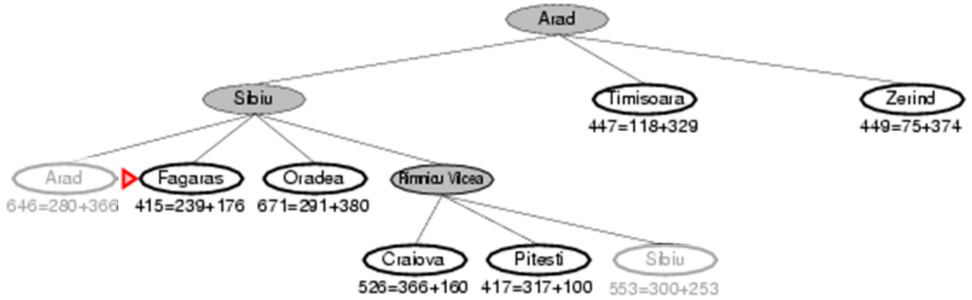
Pitesti

Timisoara

Zerind

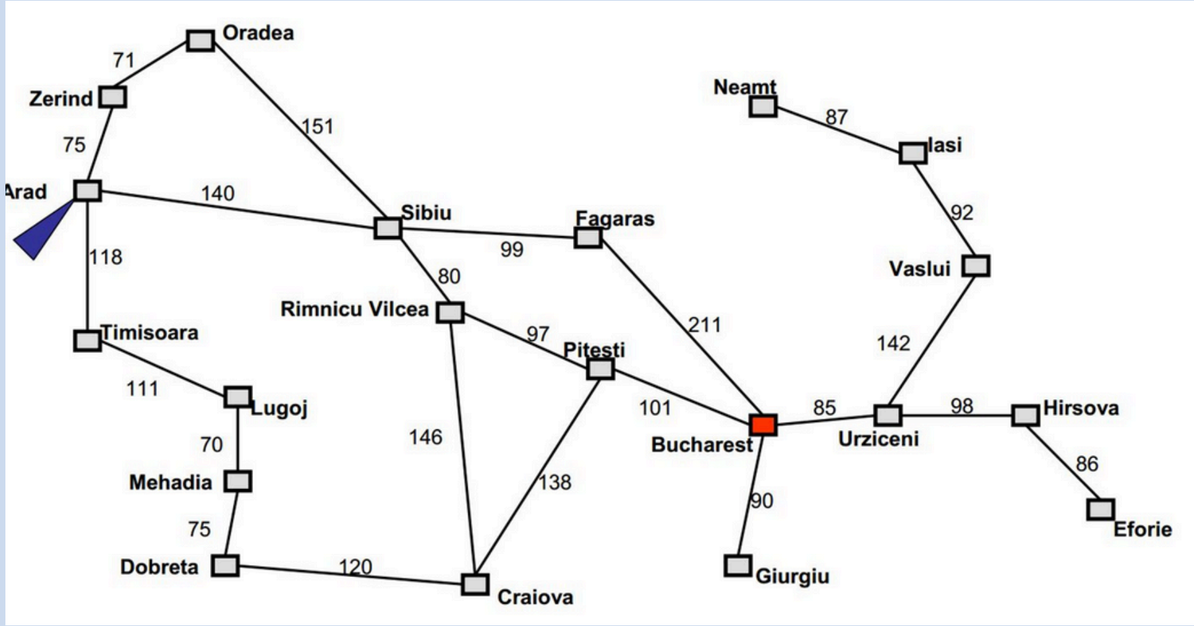
Craiova

Oradea



Fagaras will be the next node we should expand – it's at the top of the sorted open list.

Arad to Bucharest



Straight line distance to Bucharest

Arad	366	Hirsova	151	Rimnicu Vilcea	193
Bucharest	0	Iasi	226		
Craiova	160	Lugoj	244	Sibiu	253
Dobreta	242	Mehadia	241	Timisoara	329
Eforie	161	Neamt	234	Urziceni	80
Fagaras	176	Oradea	380	Vaslui	199
Giurgiu	77	Pitesti	100	Zerind	374

EXAMPLE

Heuristic Function:

$f(n) = g(n) + h(n)$

Open List:

Pitesti

Timisoara

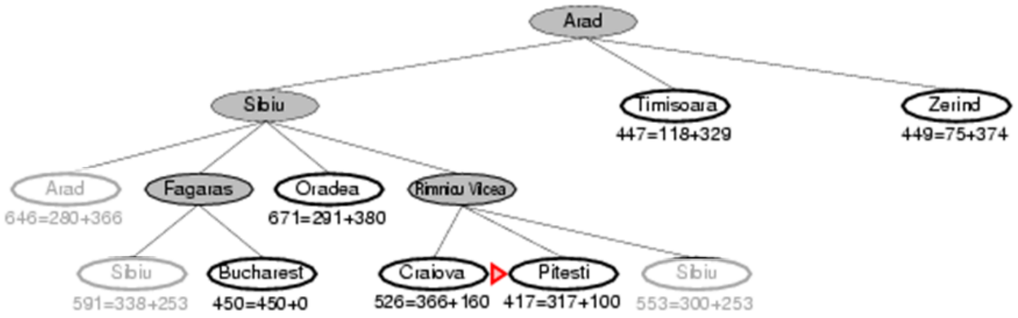
Zerind

Bucharest

Craiova

~~Sibiu~~

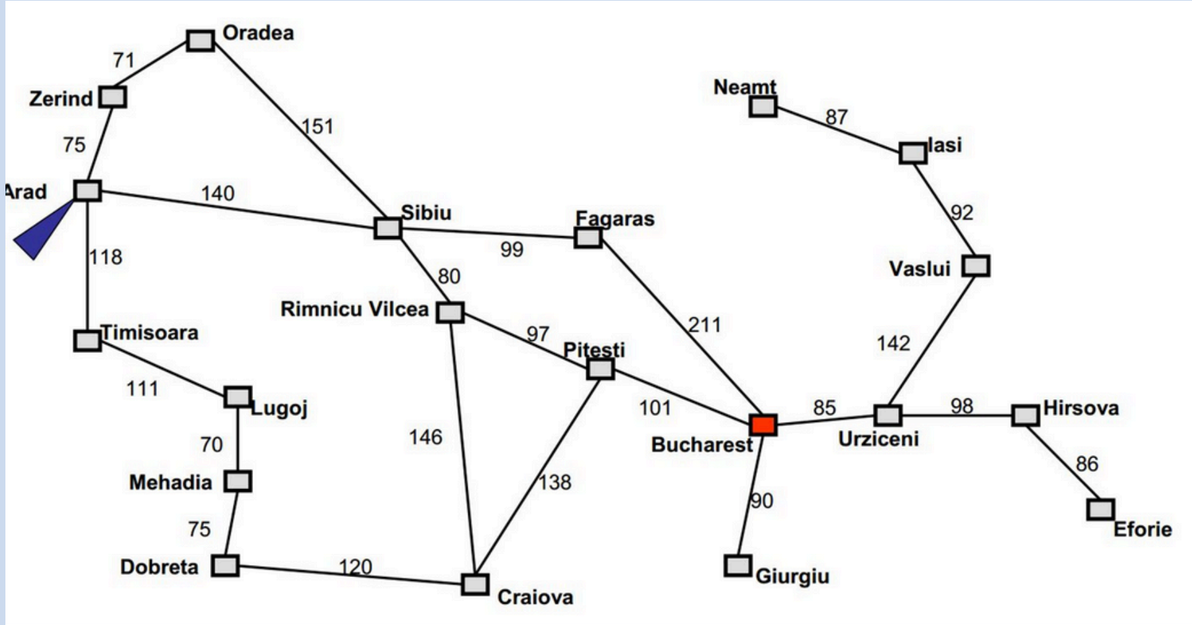
Oradea



When we expand Fagaras, we find Sibiu again. We don't add it to the open list.

We also find Bucharest, but we're not done.The algorithm doesn't end until we "expand" the goal node – it has to be at the top of the open list.

Arad to Bucharest



Straight line distance to Bucharest

Arad	366	Hirsova	151	Rimnicu Vilcea	193
Bucharest	0	Iasi	226		
Craiova	160	Lugoj	244	Sibiu	253
Dobreta	242	Mehadia	241	Timisoara	329
Eforie	161	Neamt	234	Urziceni	80
Fagaras	176	Oradea	380	Vaslui	199
Giurgiu	77	Pitesti	100	Zerind	374

EXAMPLE

Heuristic Function:

$f(n) = g(n) + h(n)$

Open List:

Pitesti

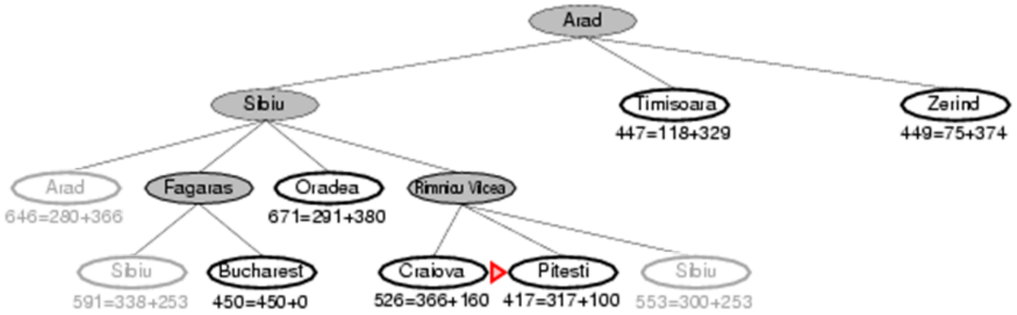
Timisoara

Zerind

Bucharest

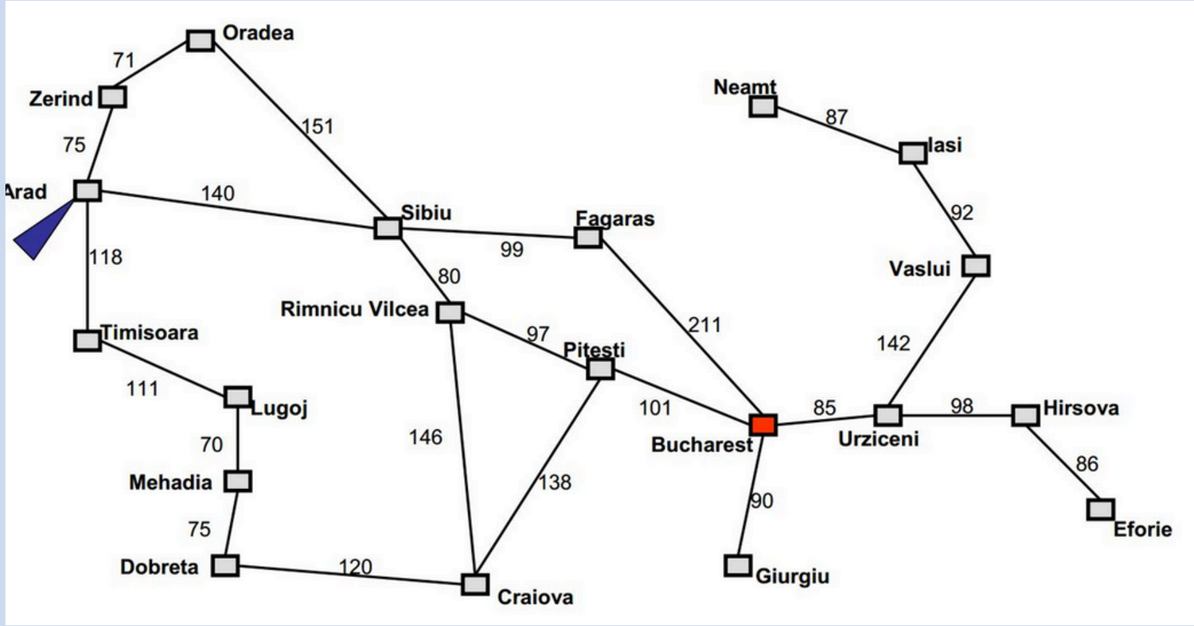
Craiova

Oradea



It looks like Pitesti is the next node we should expand.

Arad to Bucharest



Straight line distance to Bucharest

Arad	366	Hirsova	151	Rimnicu Vilcea	193
Bucharest	0	Iasi	226		
Craiova	160	Lugoj	244	Sibiu	253
Dobreta	242	Mehadia	241	Timisoara	329
Eforie	161	Neamt	234	Urziceni	80
Fagaras	176	Oradea	380	Vaslui	199
Giurgiu	77	Pitesti	100	Zerind	374

EXAMPLE

Heuristic Function:

$f(n) = g(n) + h(n)$

Open List:

Bucharest

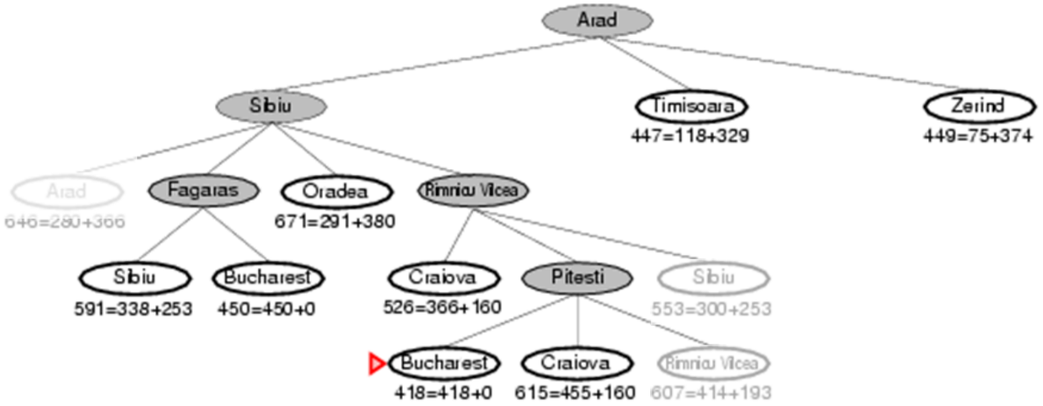
Timisoara

Zerind

Craiova

~~Rimricu Vicea~~

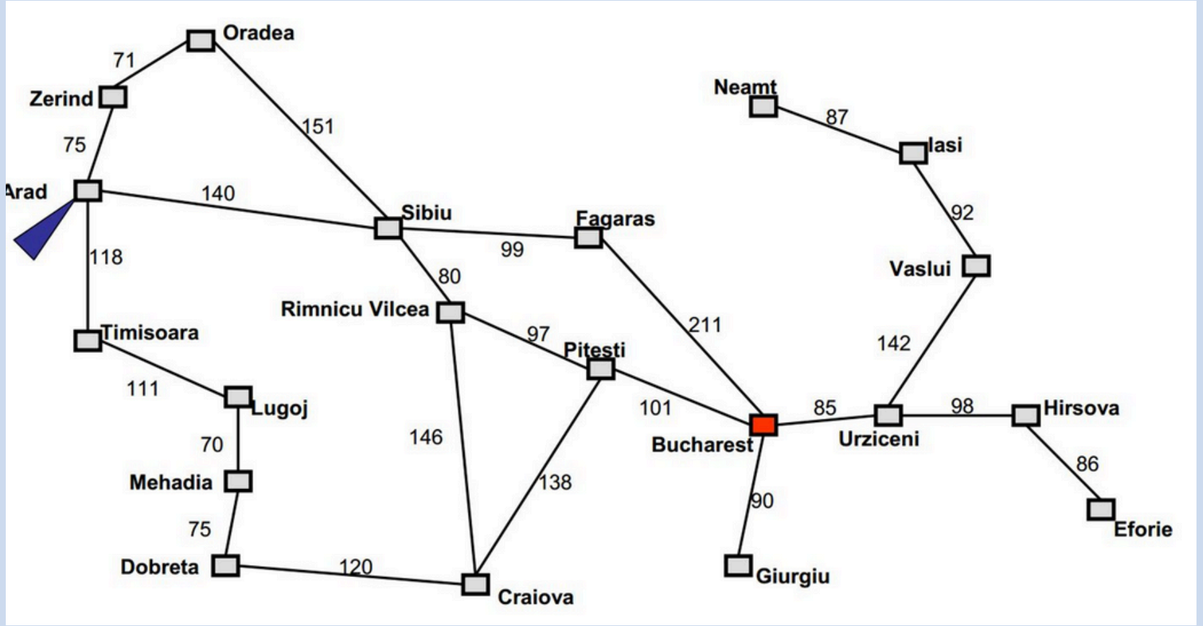
Oradea



We just found a better value for Bucharest; so, it got moved higher in the list. We also found a worse value for Craiova – we just ignore this.

And of course, we ran into Rimricu Vicea again. Since it's already been expanded once, we don't re-add it to the Open List.

Arad to Bucharest



Straight line distance to Bucharest

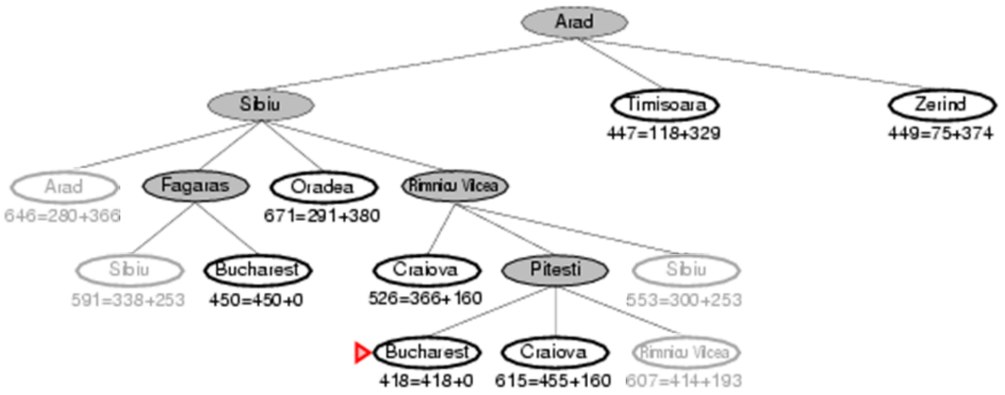
Arad	366	Hirsova	151	Rimnicu Vilcea	193
Bucharest	0	Iasi	226		
Craiova	160	Lugoj	244	Sibiu	253
Dobreta	242	Mehadia	241	Timisoara	329
Eforie	161	Neamt	234	Urziceni	80
Fagaras	176	Oradea	380	Vaslui	199
Giurgiu	77	Pitesti	100	Zerind	374

EXAMPLE

Heuristic Function:

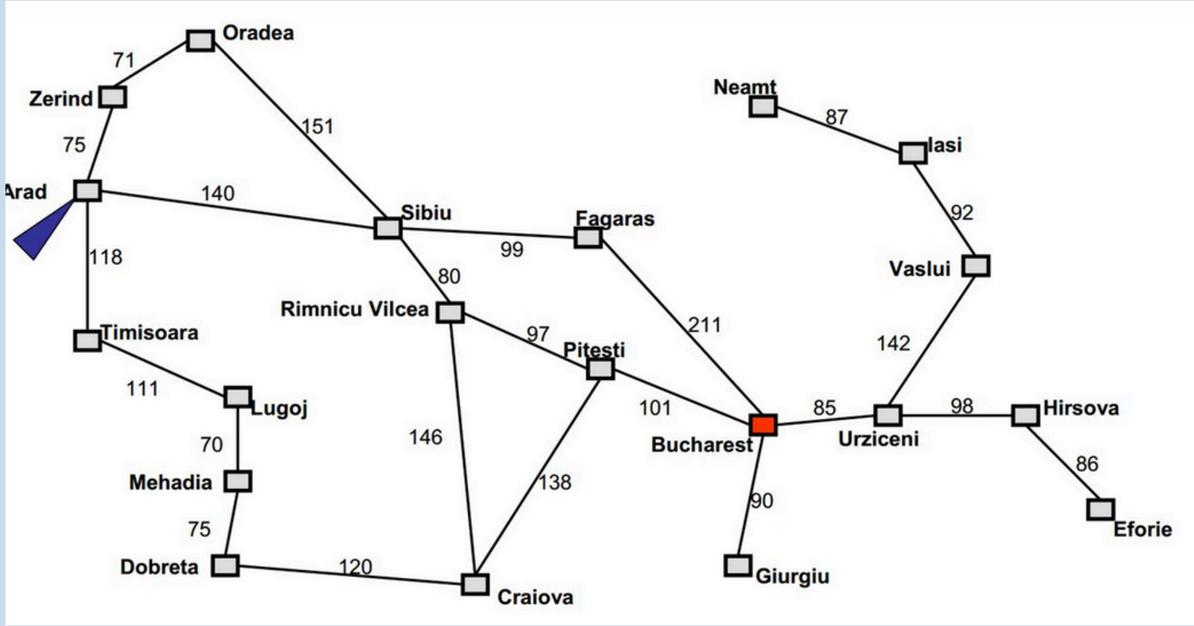
$f(n) = g(n) + h(n)$

Open List:
Bucharest
Timisoara
Zerind
Craiova
Oradea



Now it looks like Bucharest is at the top of the open list…

Arad to Bucharest



Straight line distance to Bucharest

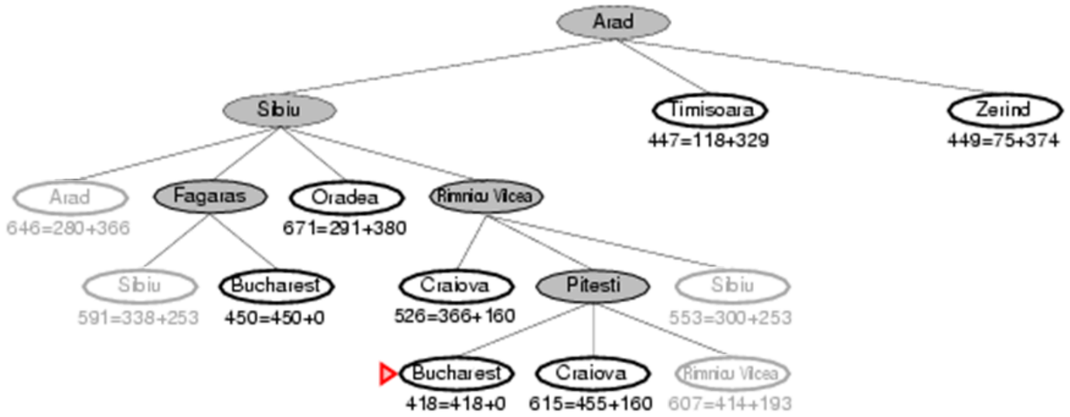
Arad	366	Hirsova	151	Rimnicu Vilcea	193
Bucharest	0	Iasi	226		
Craiova	160	Lugoj	244	Sibiu	253
Dobreta	242	Mehadia	241	Timisoara	329
Eforie	161	Neamt	234	Urziceni	80
Fagaras	176	Oradea	380	Vaslui	199
Giurgiu	77	Pitesti	100	Zerind	374

EXAMPLE

Heuristic Function:

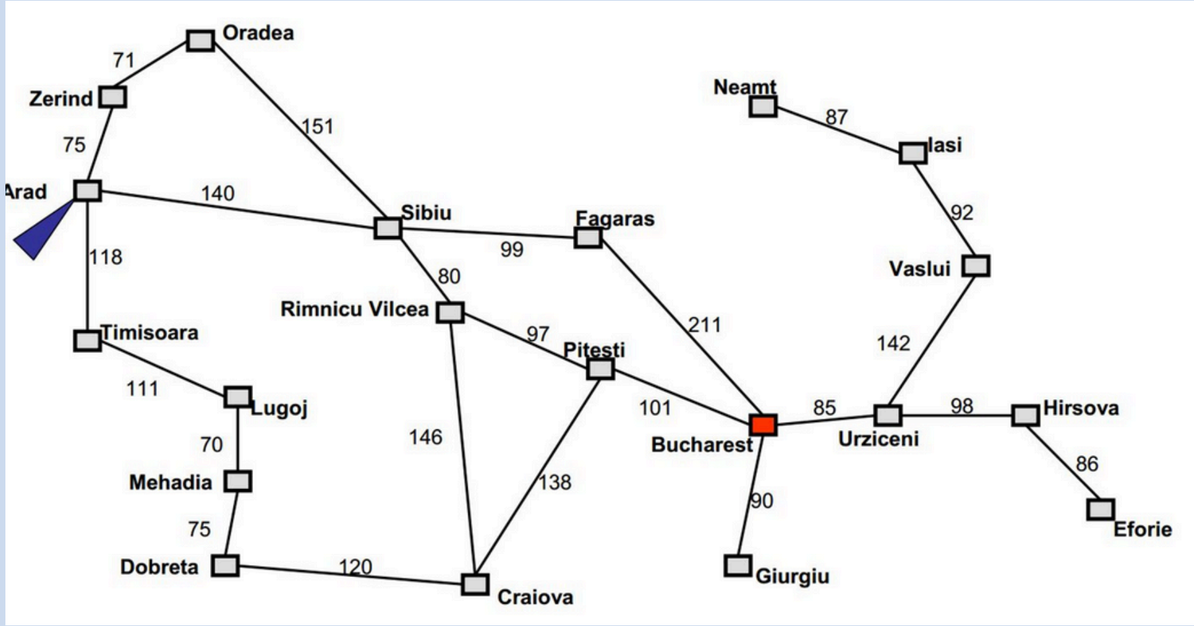
$f(n) = g(n) + h(n)$

Open List:
Bucharest
Timisoara
Zerind
Craiova
Oradea



Now we “expand” the node for Bucharest.
We’re done! (And we know the path that we’ve found is optimal.)

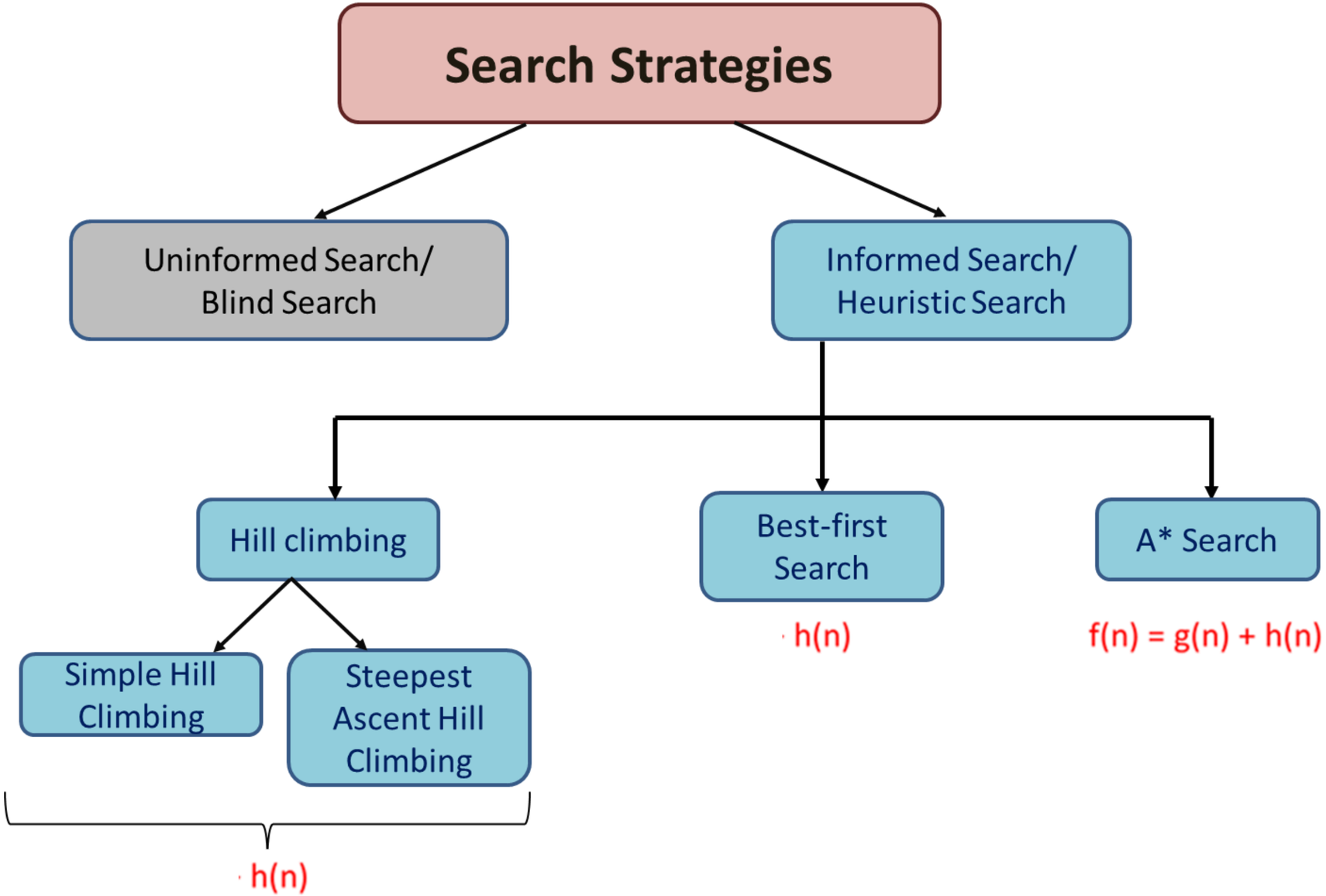
Arad to Bucharest



Straight line distance to Bucharest

Arad	366	Hirsova	151	Rimnicu Vilcea	193
Bucharest	0	Iasi	226		
Craiova	160	Lugoj	244	Sibiu	253
Dobreta	242	Mehadia	241	Timisoara	329
Eforie	161	Neamt	234	Urziceni	80
Fagaras	176	Oradea	380	Vaslui	199
Giurgiu	77	Pitesti	100	Zerind	374

SUMMARY


Search Strategies
Uninformed Search/ Blind Search
Informed Search/ Heuristic Search
Hill climbing
Best-first Search
A* Search
Simple Hill Climbing
Steepest Ascent Hill Climbing
· h(n)
· h(n)
f(n) = g(n) + h(n)

THE END

NEXT LECTURE

Knowledge representation