TARC
TUNKU ABDUL RAHMAN UNIVERSITY COLLEGE

Artificial Intelligence

Chap 4 – Informed Search

"I climb all this way, and you tell me *THAT'S* the meaning of life?!"

Today's Aims

- Define and evaluate informed (heuristic) search
- Heuristic Search Techniques
  - Hill climbing
  - Best-first Search
  - A* Search

Heuristic

- Rules for choosing those branches in a state space that are most likely to lead to an acceptable problem solution.

Example

Notes

- 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

With heuristic

- The agent is informed which branch(es) in a state space that is(are) most likely to lead to an acceptable problem solution.

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) = estimated distance of state *n* from goal

= number of tiles at the right place.

* h(n) determines the heuristic cost of a state

Initial:

h = 4

2	8	3
1	6	4
7		5

Final:

h = 8

1	2	3
8		4
7	6	5

Goal

Hill Climbing

- The simplest way in heuristic search
- 2 types:
  - Simple Hill Climbing
  - Steepest Ascent Hill Climbing

Limitations of Hill Climbing

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

Limitations of Hill Climbing

- Hill climbing may encounter **local maximum**, which are
  1. Foothills,
  2. Plateau, and
  3. ridges

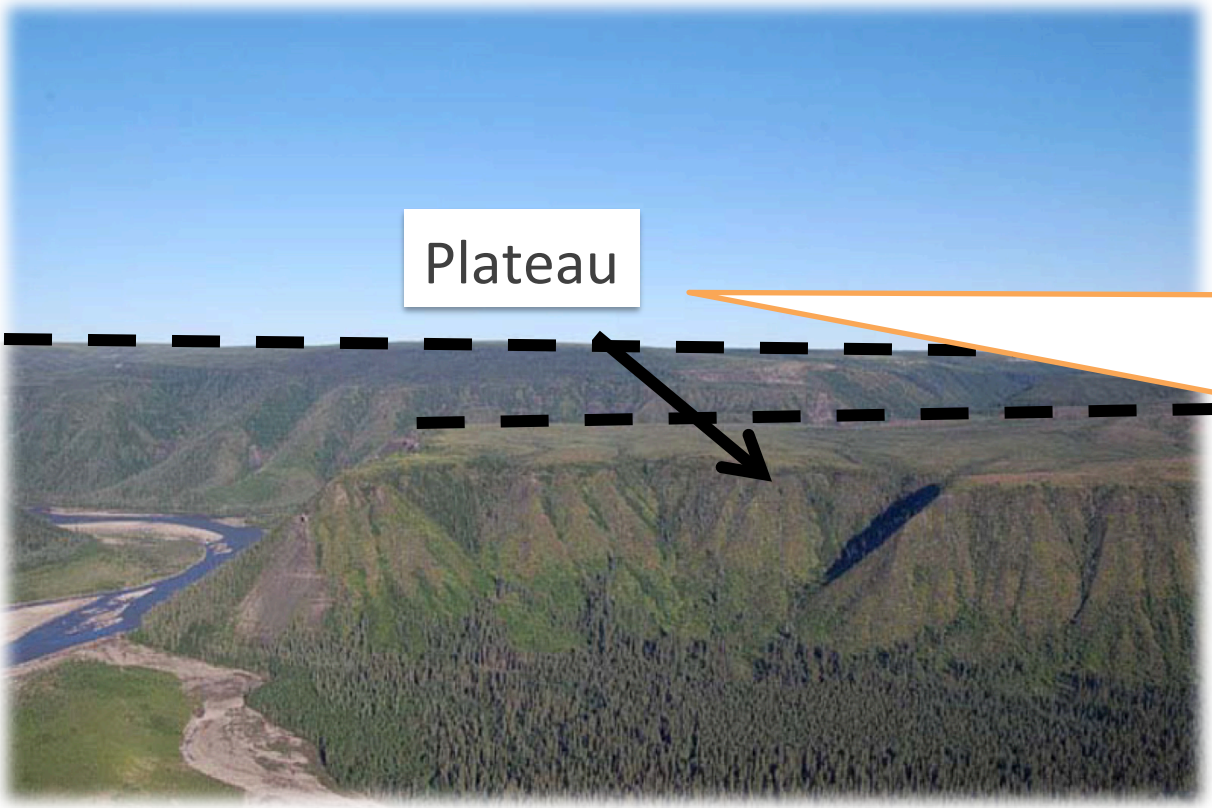
Foothill

Foothill - a state that has no better move.

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

Plateau

An area of state space where the heuristic values are equal.

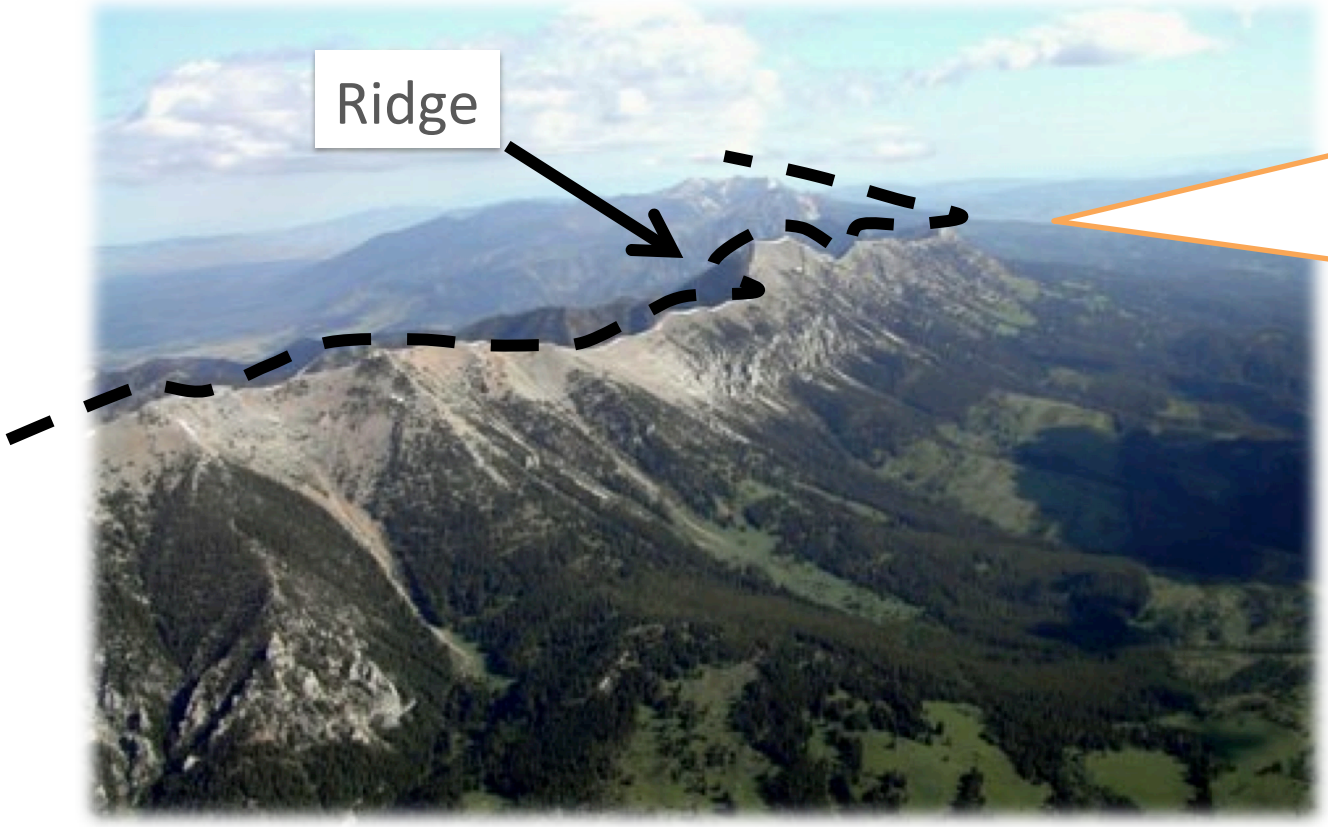


Steepest accent hill climber will not know which direction to follow

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

Ridge

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



the ridge may slope very slowly toward the peak making it a long route.

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

Let the heuristic function, h(n), is given as follows:

h(n) = estimated distance of state *n* from goal

= number of tiles at the right place.

* h(n) determines the heuristic cost of a state

Initial:

h = 4

2	8	3
1	6	4
7		5

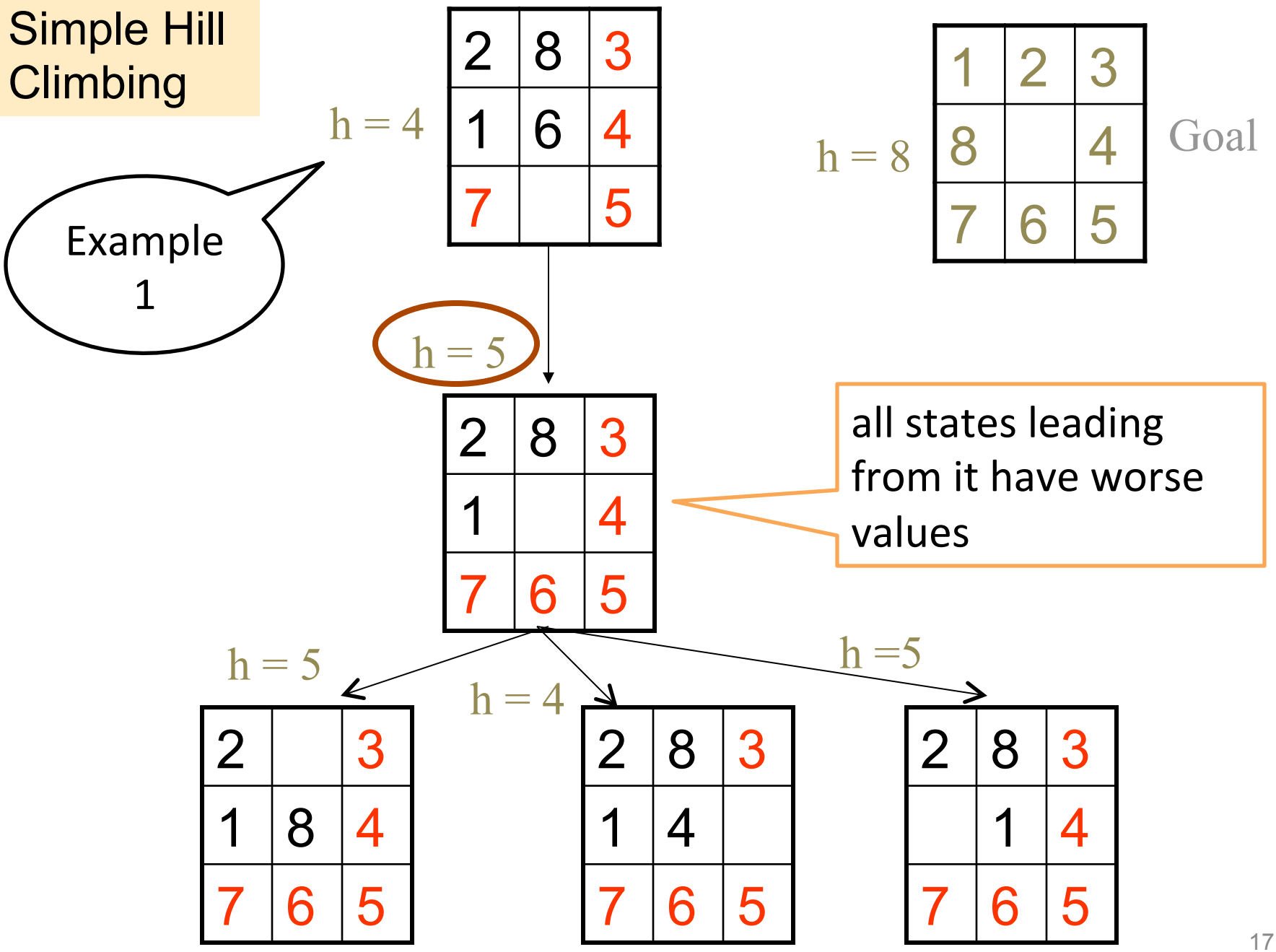
Final:

h = 8

1	2	3
8		4
7	6	5

Goal

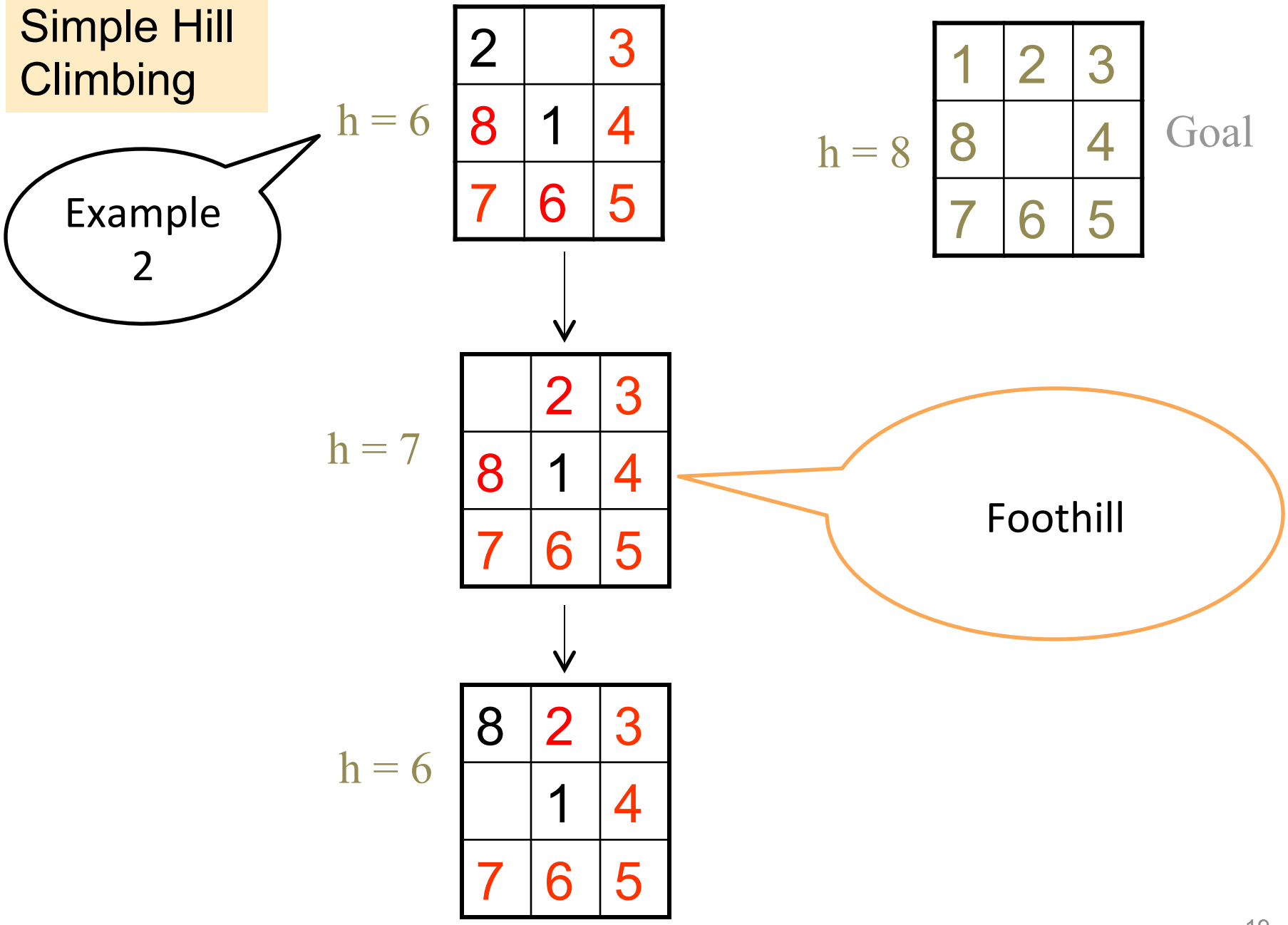
Simple Hill Climbing



Ops! We're stuck!

- Simple hill climbing has faced a dead end.
- The child nodes are worse than the parent node.
- This problem is known as **foothill**.

Simple Hill Climbing



Steepest Ascend Hill Climbing

- Here, we attempt to improve on the previous Hill Climbing algorithm
  - Given initial state perform the following

Step 1 : *Evaluate the initial state. If it is goal state then exit else make the current state as initial state*

Step 2 : *Repeat these steps until a solution is found or current state does not change*

i. Let 'target' be a state such that any successor of the current state will be better than it;

ii. for each operator that applies to the current state

a. apply the new operator and create a new state

b. evaluate the new state

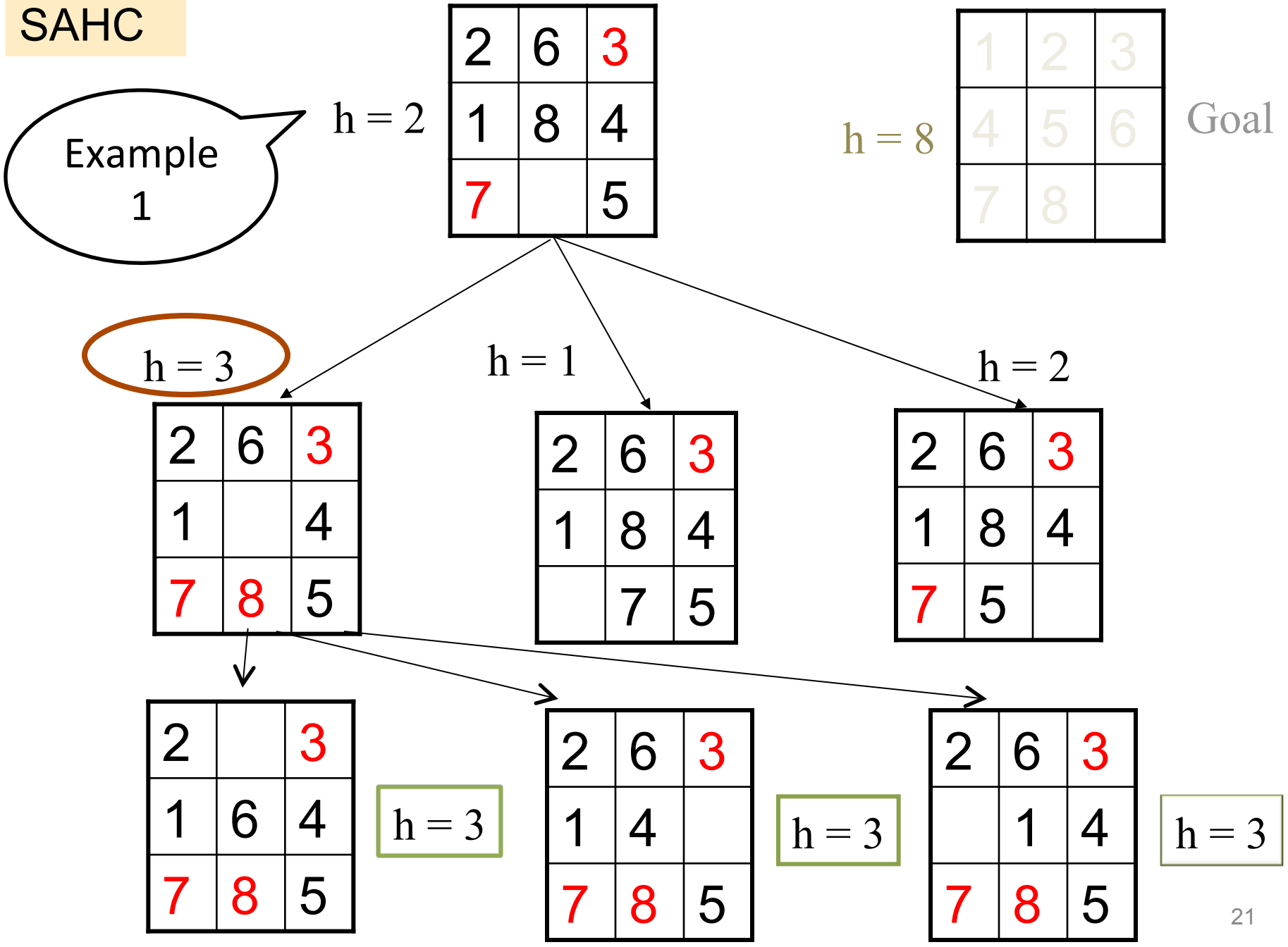
c. if this state is goal state then quit else compare with 'target'

d. if this state is better than 'target', set this state as 'target'

e. if target is better than current state set current state to Target

Step 3 : *Exit*

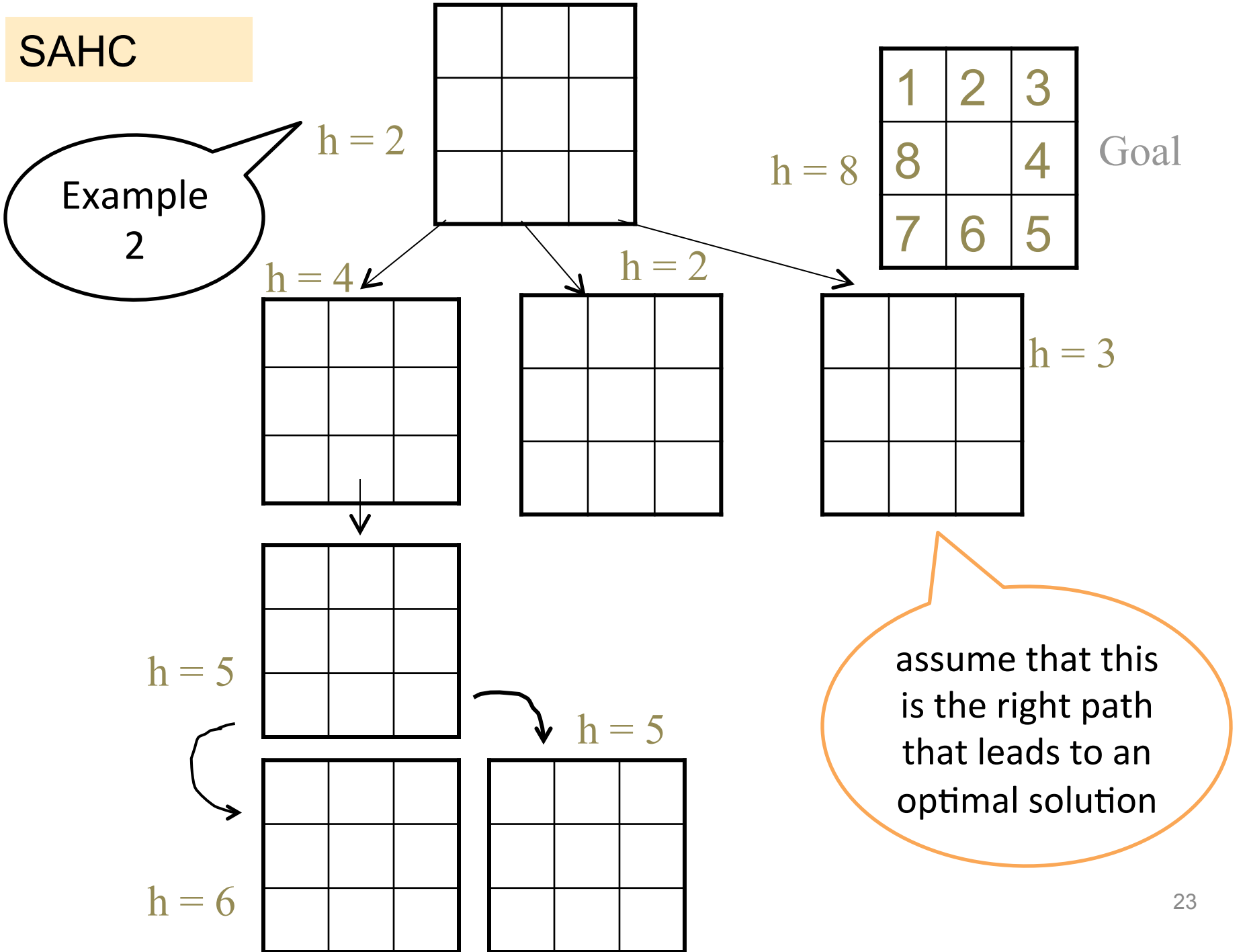
SAHC


Example 1
h = 2
2 6 3
1 8 4
7 5
h = 8
1 2 3
4 5 6
7 8
Goal
h = 3
h = 1
h = 2
2 6 3
1 4
7 8 5
2 6 3
1 8 4
7 5
2 6 3
1 8 4
7 5
2 3
1 6 4
7 8 5
h = 3
2 6 3
1 4
7 8 5
h = 3
2 6 3
1 4
7 8 5
h = 3

Ops! We're stuck again!

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

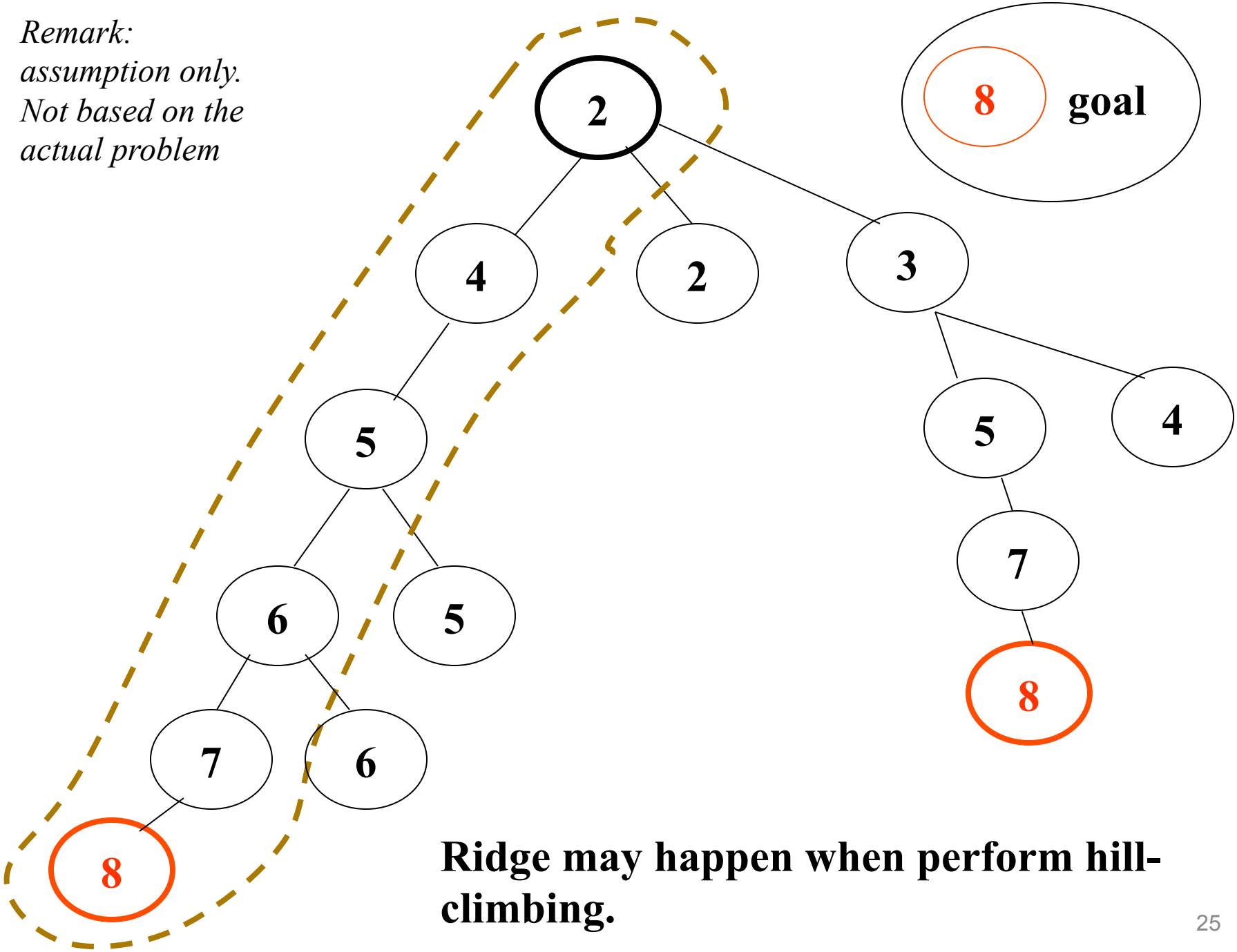
SAHC


Example 2
h = 2
h = 8
1 2 3
8 4
7 6 5
Goal
h = 4
h = 2
h = 3
h = 5
h = 5
h = 6
assume that this is the right path that leads to an optimal solution

Ops! It took the wrong path!

- The actual path that the search took in Example 2 leads to a sub-optimal solution (longer path).
- The problem is known as **ridge**.

Remark:
assumption only.
Not based on the actual problem



Ridge may happen when perform hill-climbing.

Conclusion

- Hill climbing is not always effective
- Heuristic function not always perfect
- inefficient in a large, rough problem space
- Useful if combined with other methods

Best First Search

Recall

- Depth-first search
  - without traverse all the branches.
- Breadth-first search
  - Not trapped into a dead-end path.

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

- Set *Open* := [Start] and *closed* := [ ]
- Loop until open := [ ]
  - Remove the LEFTMOST node *X* from *open*
  - If *X = goal* then return the path from *Start* to *X*
  - Else generate children of *X*
  - For each child of *X*
    - If the child is on *open* or *closed*
      - Assign the child its heuristic value
      - Add the child to open
    - Else if the child is on open
      - If the child was reached by a shorter path
        - Give the child a state on open the shorter path

…CONT NEXT PAGE

Best-first Search Algorithm -CONT

- Else if the child is already on closed
  - If the child was reached by a shorter path
    - Remove the state from closed
    - Add the child to open
- Put X on closed
- Sort states on open by heuristic merit (best leftmost)

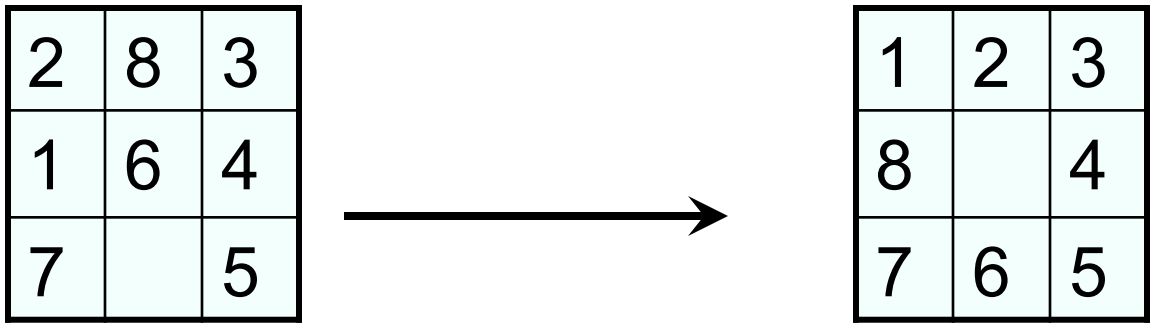
Return fail

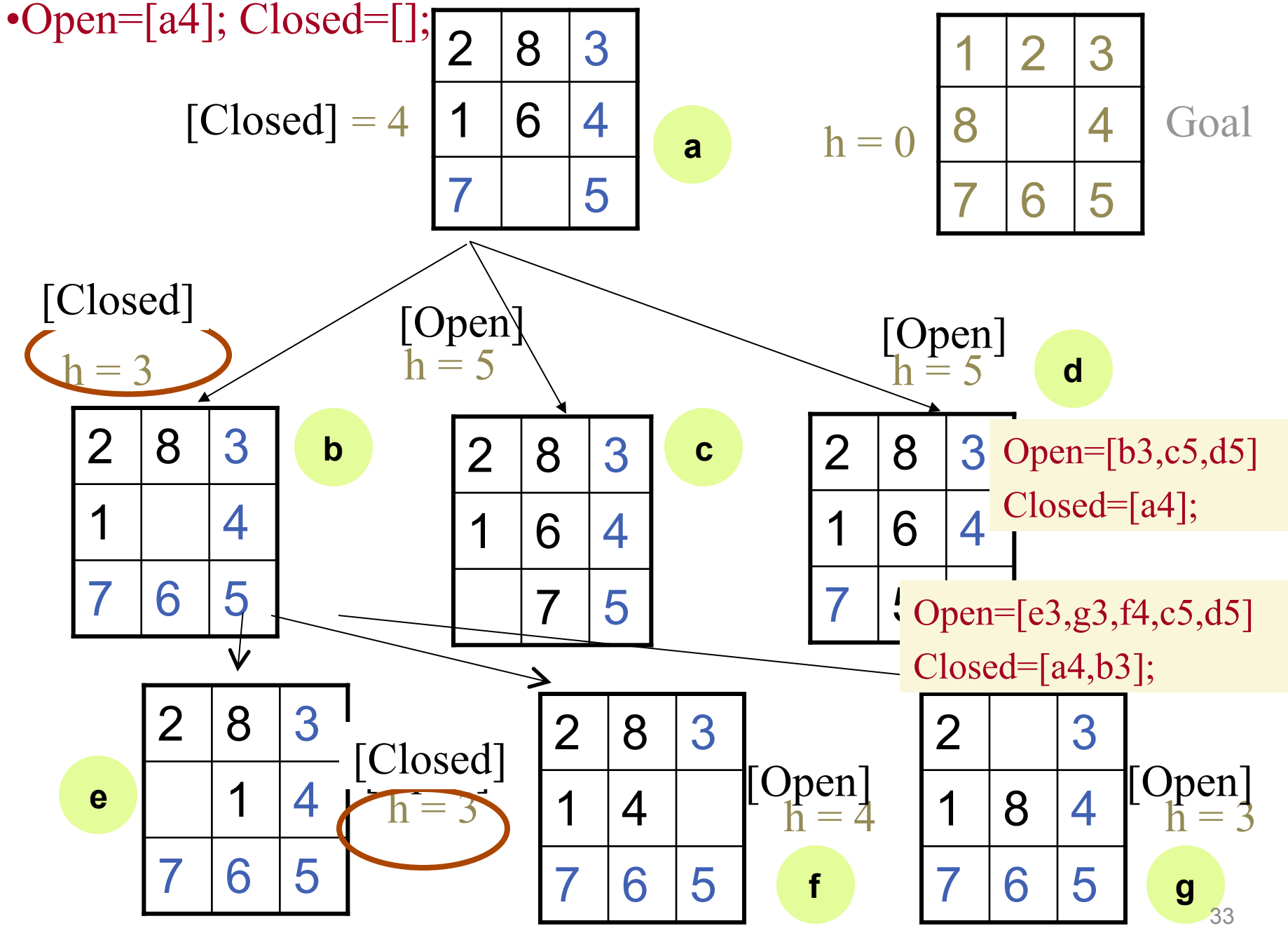
After loop till open = [ ]

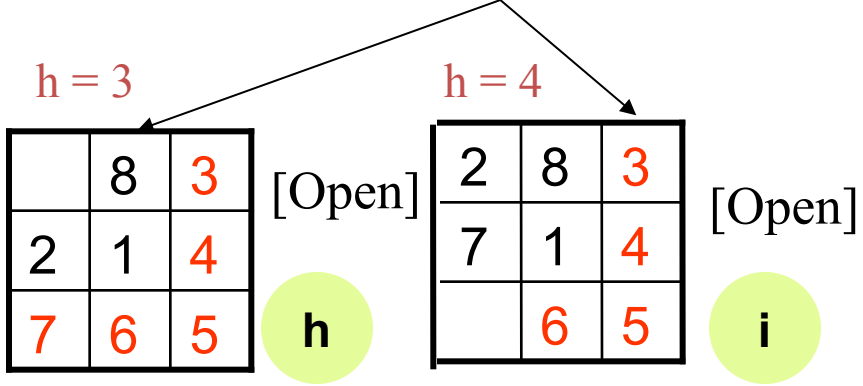
Back to exercise

heuristic

- Let:
- h(n) = estimated distance of state n from goal
- = number of tiles at the wrong place.
- * h(n) = heuristic cost
- Initial: Final:

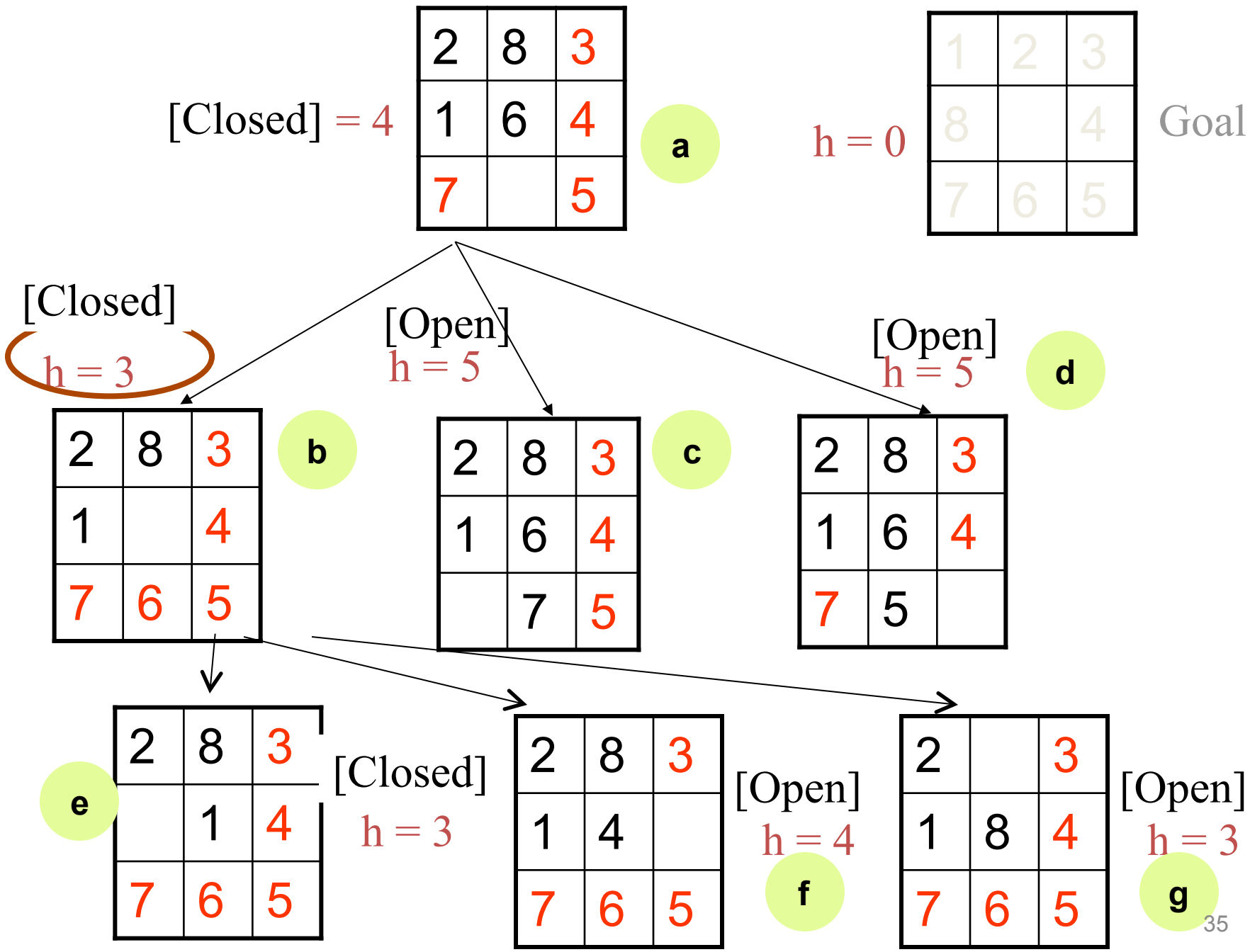


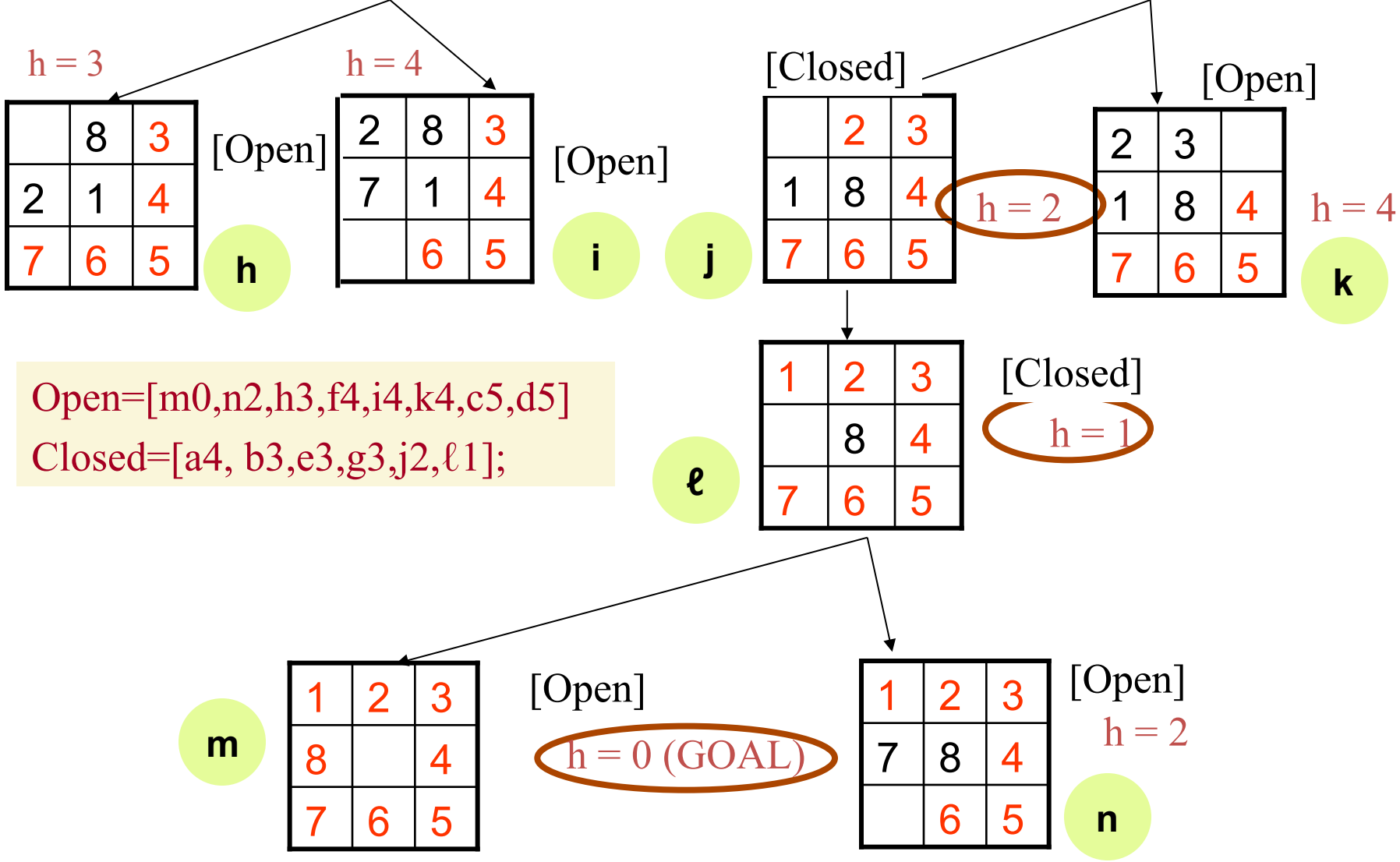

•Open=[a4]; Closed=[];
[Closed] = 4
a
h = 0
Goal
[Closed]
h = 3
[Open]
h = 5
[Open]
h = 5
b
c
d
Open=[b3,c5,d5]
Closed=[a4];
Open=[e3,g3,f4,c5,d5]
Closed=[a4,b3];
e
[Closed]
h = 3
[Open]
h = 4
f
[Open]
h = 3
g



Open=[g3,h3,f4,i4,c5,d5]

Closed=[a4,b3,e3]

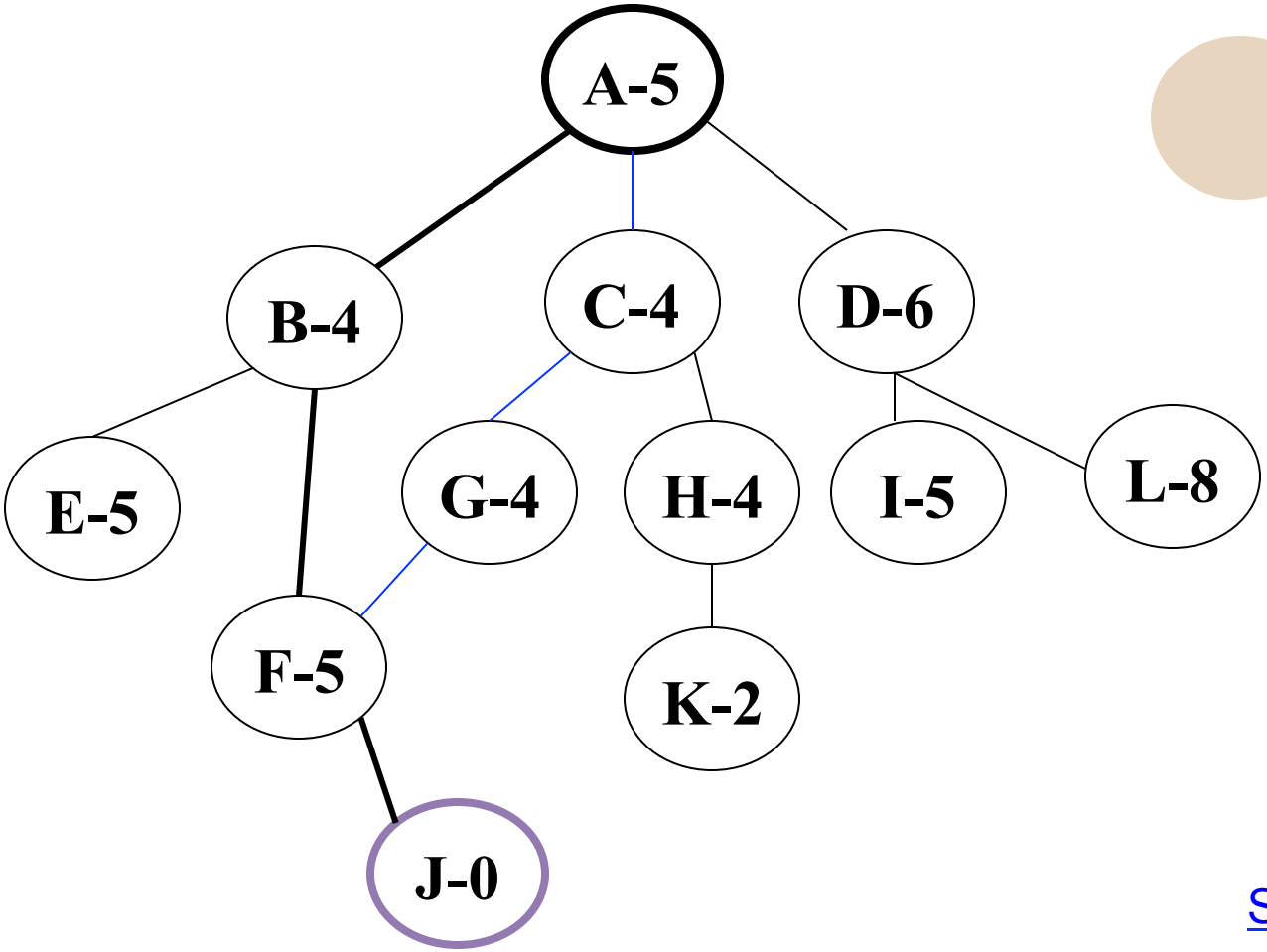

[Closed] = 4
2 8 3
1 6 4
7 5
a
h = 0
1 2 3
8 4
7 6 5
Goal
[Closed]
h = 3
[Open]
h = 5
[Open]
h = 5
d
2 8 3
1 4
7 6 5
b
2 8 3
1 6 4
7 5
c
2 8 3
1 6 4
7 5
e
2 8 3
1 4
7 6 5
[Closed]
h = 3
2 8 3
1 4
7 6 5
[Open]
h = 4
f
2 3
1 8 4
7 6 5
[Open]
h = 3
g


h = 3
h = 4
[Closed]
[Open]
[Open]
[Open]
h = 2
h = 4
[Closed]
h = 1
[Open]
h = 0 (GOAL)
[Open]
h = 2
h
i
j
k
ℓ
m
n
Open=[m0,n2,h3,f4,i4,k4,c5,d5]
Closed=[a4, b3,e3,g3,j2,ℓ1];

Open & Closed List

- Open=[a4]; Closed=[];
- Open=[b3,c5,d5]; Closed=[a4];
- Open=[e3,g3,f4,c5,d5]; Closed=[a4,b3];
- Open=[g3,h3,f4,i4,c5,d5];Closed=[a4,b3,e3];
- Open=[j2,h3,f4,i4,k4,c5,d5]; Closed=[a4,b3,e3,g3];
- Open=[ℓ1,h3,f4,i4,k4,c5,d5];
  Closed=[a4, b3,e3,g3,j2];
- Open=[m0,n2,h3,f4,i4,k4,c5,d5];
  Closed=[a4, b3,e3,g3,j2,ℓ1];
- Evaluate m0; the solution is found; return path = a->b->g->j->ℓ->m

Exercise



See algorithm

Trace

- Open=[a5]; Closed=[];
- Eval A-5: Open=[b4,c4,d6]; Closed=[a5];
- Eval B-4: Open=[c4,e5,f5,d6]; Closed=[b4,a5];
- Eval C-4: Open=[g4,h4,e5,f5,d6];Closed=[c4,b4,a5];
- Eval G-4: Open=[h4,e5,f5*,d6]; Closed=[g4,c4,b4,a5];
- Eval H-4: Open=[k2, e5,f5*,d6]; Closed=[h4, g4,c4,b4,a5];
- Eval K-2: Open=[e5,f5*,d6]; Closed=[k2,h4, g4,c4,b4,a5];
- Eval E-5: Open=[f5*,d6]; Closed=[e5,k2,h4, g4,c4,b4,a5];
- Eval F-5: Open=[j0,d6]; Closed=[f5*,e5,k2,h4, g4,c4,b4,a5];
- Eval j-0: Goal is found! Return path = a->b->f->j
- * f5 was reached by B-4

A* Search (Luger, pg 139)

Heuristic Function:

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

  where:

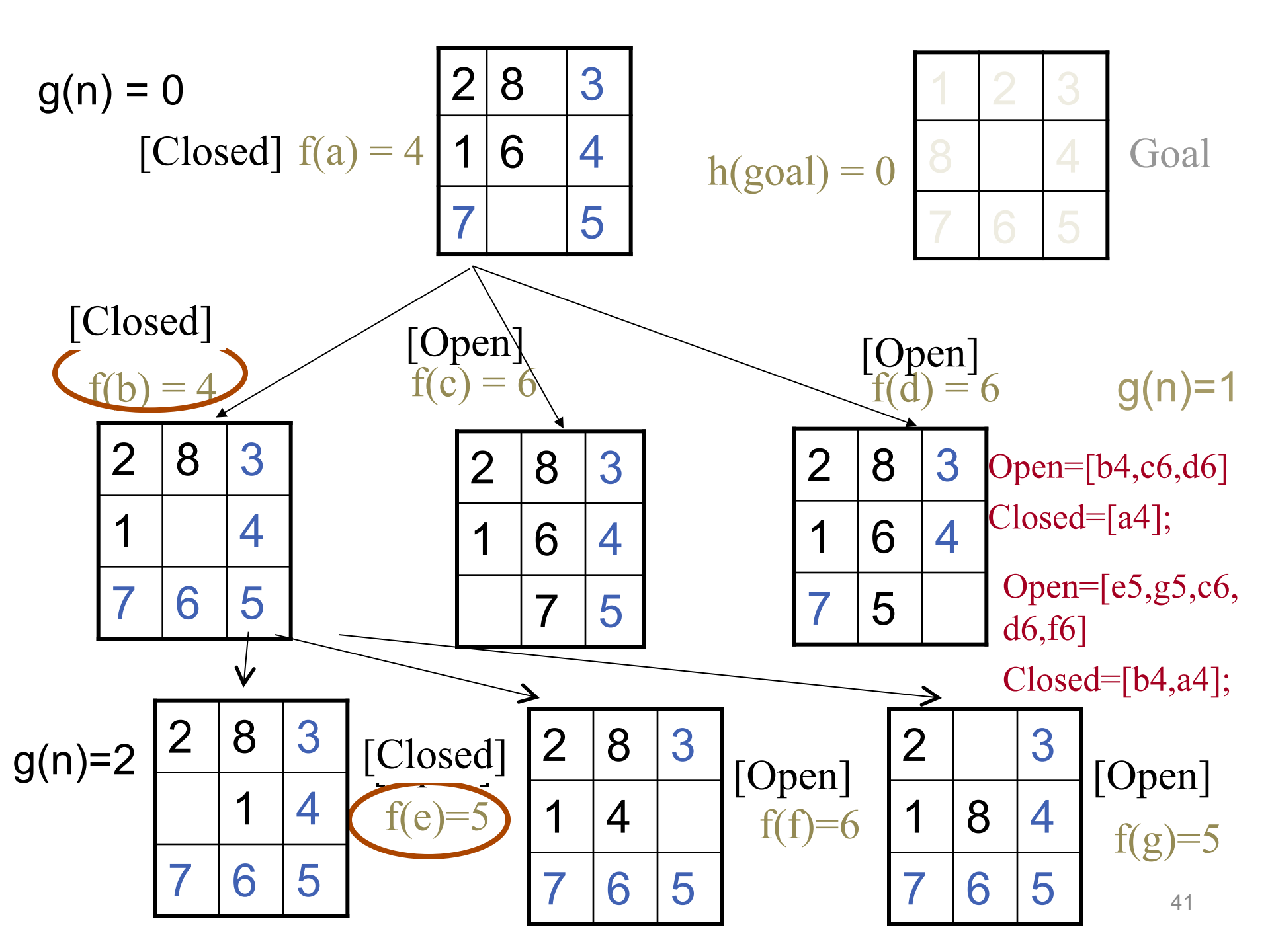
  g(n) = distance n from the start state and

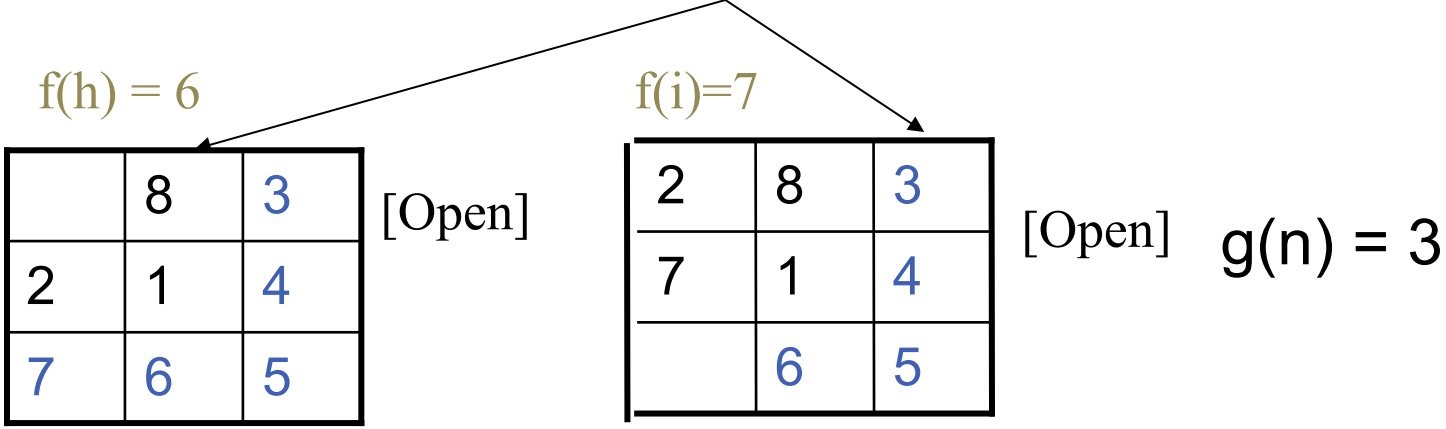
  g(start state) = 0

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

  h(goal) = 0

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


g(n) = 0
[Closed] f(a) = 4
h(goal) = 0
Goal
[Closed]
f(b) = 4
[Open]
f(c) = 6
[Open]
f(d) = 6
g(n)=1
Open=[b4,c6,d6]
Closed=[a4];
Open=[e5,g5,c6,
d6,f6]
Closed=[b4,a4];
g(n)=2
[Closed]
f(e)=5
[Open]
f(f)=6
[Open]
f(g)=5



	8	3
2	1	4
7	6	5

2	8	3
7	1	4
	6	5

Open=[g5,c6,d6,f6,h6, i7]
Closed=[e5,b4,a4];

The same process will be iterated until the goal is found

Differences between SAHC and BFS

SA Hill-climbing

- 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.

Best-first search

- 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

Next Week

Knowledge representation