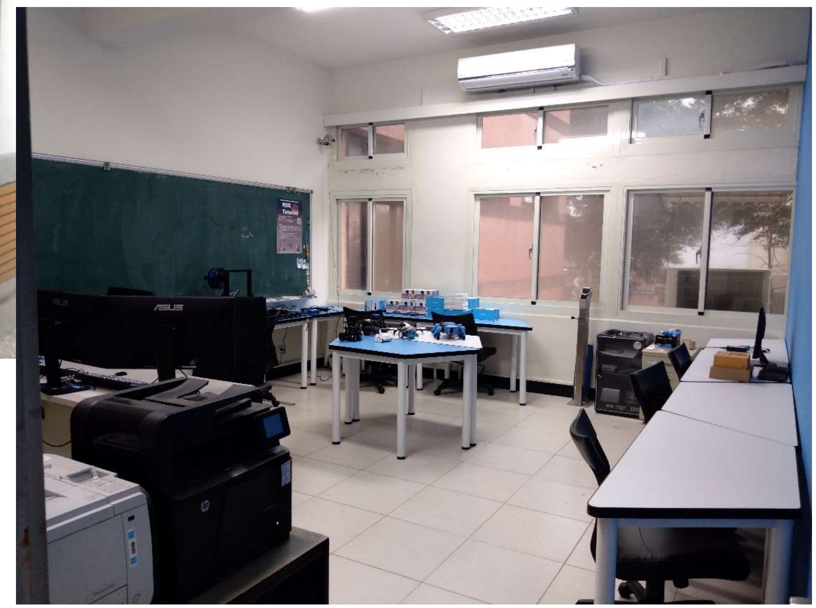
# Search—Heuristic Search

KUO-SHIH TSENG (曾國師)
DEPARTMENT OF MATHEMATICS
NATIONAL CENTRAL UNIVERSITY, TAIWAN

2020/03/10

# Course Announcement


- If you enrolled in this course, you can access to Robotics Lab (M213) via your NCU ID since this week.
- Robotics Lab has
  - Minibot X 10
  - Turtlebot3 X10
  - Bebop X8
  - PC X1
  - Notebook X2
  - Monitor X4



# Course Announcement

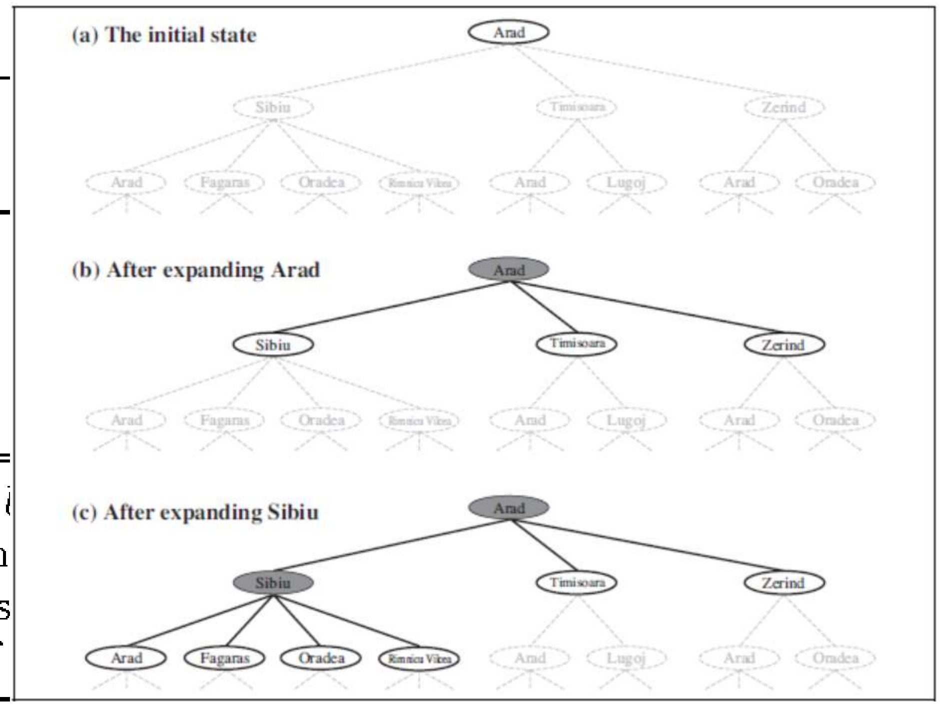
- HW1 was released on 3/02. Download it from eeclass.
- 1 programming problem and 2 theory problems.
- Deadline: 3/24(Wed.) 00:00am
- Delivery: Please update your HW to eecalss using electrical format. Compress your HW into a zip file including PDF and code.
- **Late policy**: If your HW is late for 1 day, the discount rate is 0.8. For 2 days, the discount rate is 0.8^2. and so on.
- Start to work on it this week. You have 3 weeks to work.

# Course Announcement

- DFS is not complete and optimal?

| Criterion | Breadth-First | Uniform-Cost | Depth-First |
|---|---|---|---|
| Complete? | Yes[a] | Yes[a,b] | No |
| Time | $O(b^d)$ | $O(b^{1+\lfloor C^*/\epsilon \rfloor})$ | $O(b^m)$ |
| Space | $O(b^d)$ | $O(b^{1+\lfloor C^*/\epsilon \rfloor})$ | $O(bm)$ |
| Optimal? | Yes[c] | Yes | No |

**Figure 3.21** Evaluation of tree-search strategies. $l$
of the shallowest solution; $m$ is the maximum depth
Superscript caveats are as follows: [a] complete if $b$ is
positive $\epsilon$; [c] optimal if step costs are all identical; [d] if

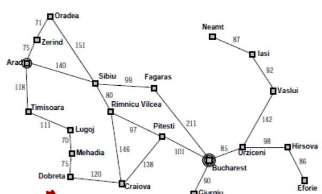


# Outline

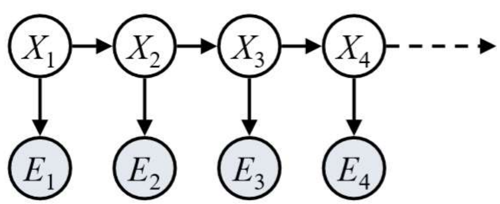
- Informed search
  - Greedy best-first search
  - A* search
- Heuristic functions
- Online search
- Learning real-time A* (LRTA*)

# Outline

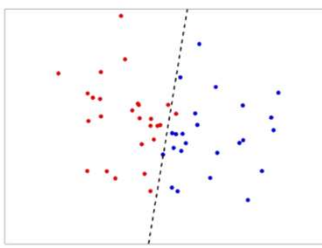
**[Problem solving]**
- Search problems
- Adversarial Search

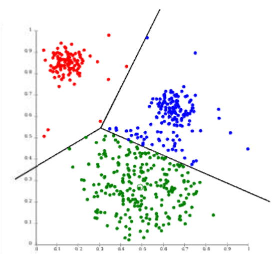


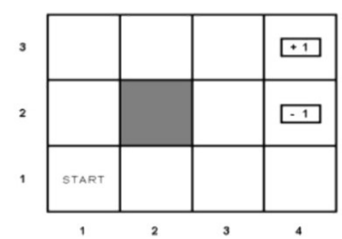

**[Perception and Uncertainty]**
- Bayes Theorem
- Bayes Filter and Smoothing




**[Learning and Decision-making]**
- Supervised learning
- Unsupervised learning
- Reinforcement learning







# Uninformed search and informed search

- Uninformed search:
  - There is no additional information beyond the given problem definition. It's also called blind search. The agent only can check if the current state is the goal state after actions.
- Informed search:
  - There is an additional information beyond the given problem definition. The agent can use this information to search more efficiently than blind search.

$$f(i) = \underbrace{g(i)}_{past} + \underbrace{h(i)}_{future}$$

$$\begin{cases} \text{Uninformed search}: f(i) = g(i) \\ \text{Heuristic search}: f(i) = g(i) + h(i) \end{cases}$$

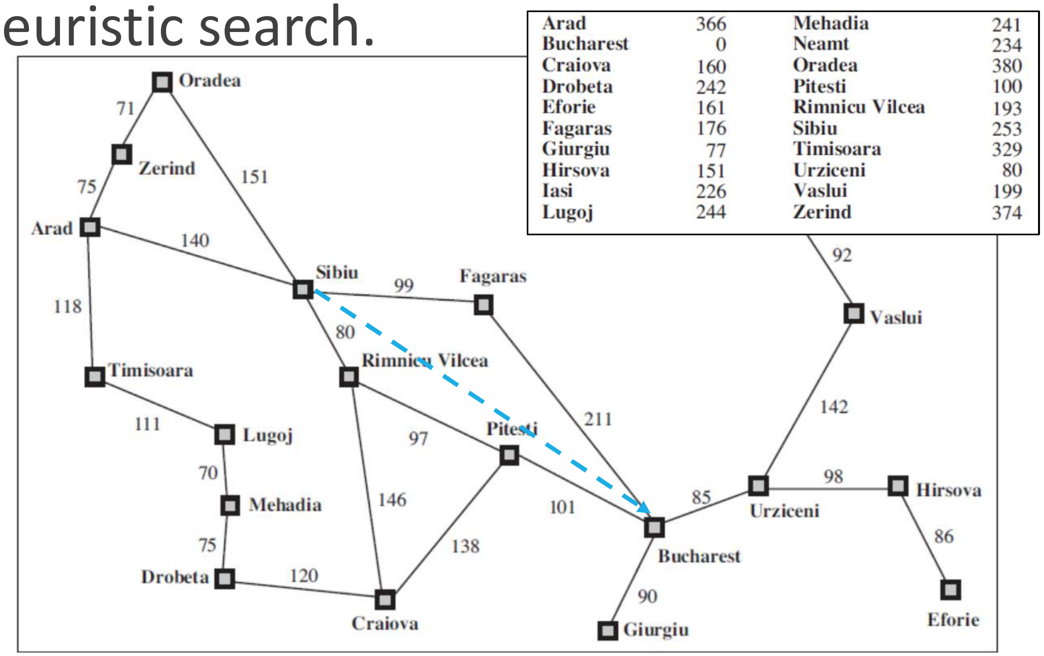
$f$ : evaluation function

$g$ : cost function

$h$ : heuristic function

# Informed search

- There is an additional information beyond the given problem definition. For example, in Romania-distance problem, the estimated cost to the goal is h(i), where i is the current node. Informed search is also called heuristic search.
  - h: Heuristic function
- Algorithms:
  - Greedy best-first search
  - A* search

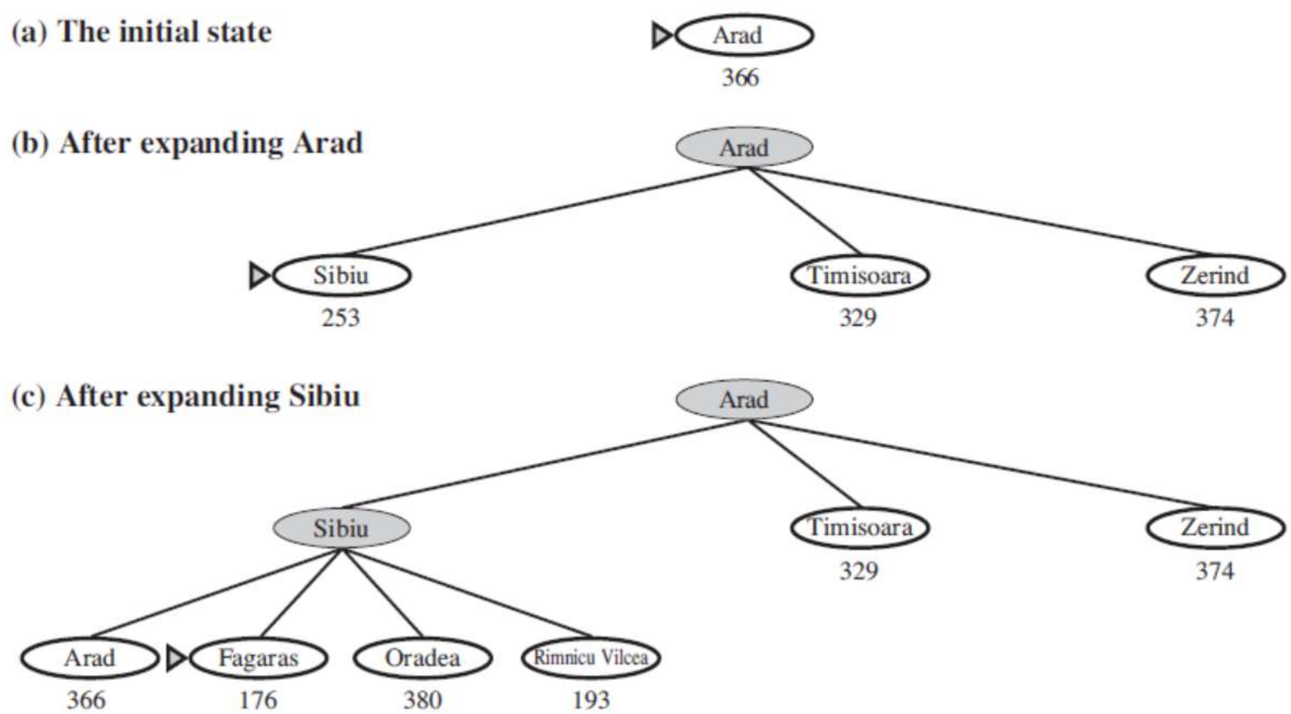



| | | | |
|---|---|---|---|
| Arad | 366 | Mehadia | 241 |
| Bucharest | 0 | Neamt | 234 |
| Craiova | 160 | Oradea | 380 |
| Drobeta | 242 | Pitesti | 100 |
| Eforie | 161 | Rimnicu Vilcea | 193 |
| Fagaras | 176 | Sibiu | 253 |
| Giurgiu | 77 | Timisoara | 329 |
| Hirsova | 151 | Urziceni | 80 |
|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

* Heuristic in Greek means “to find.”

# Informed search— Greedy best-first search

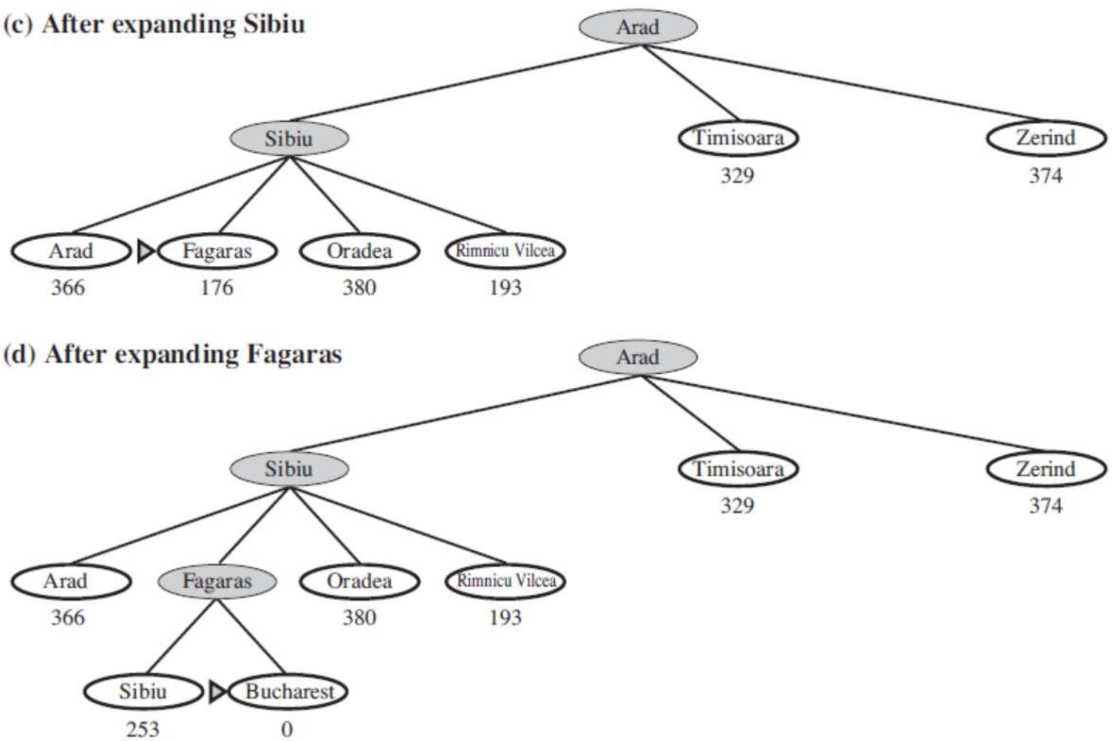
- Greedy best-first search: Expand the node with the lowest path cost from the current node to goal (heuristic function).




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

# Informed search— Greedy best-first search

- Greedy best-first search: Expand the node with the lowest path cost from the current node to goal (heuristic function).




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

# Informed search— Greedy best-first search

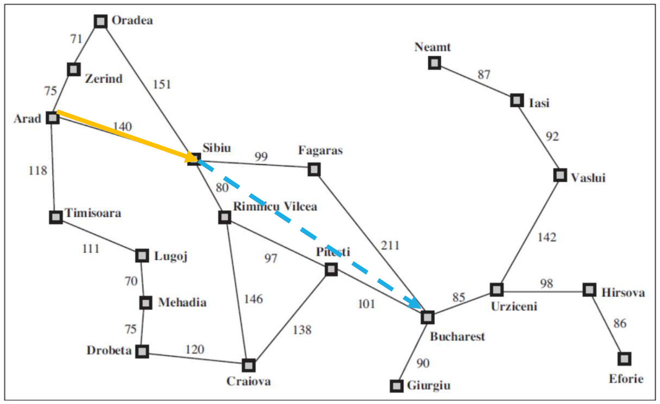
- Greedy best-first search is similar to uniform cost search but it use h function instead of g function. The data structure of greedy best-first search is **priority queue**.
- This approach is not optimal and not complete.
- UCS considers the **path cost** (g) while Greedy best-first search considers the **future cost** (h, heuristic). Could we consider both of them to find a better solution?
- How about $\text{Heuristic search}: f(i) = g(i) + h(i)$ ?

# Informed search— A*

- A*: Expand the node with the lowest evaluation cost.
- f(i)= g(i) + h(i)
- f(i): Evaluation cost
- g(i): distance from start to current node i
- h(i): ***estimated*** distance from goal to current node i

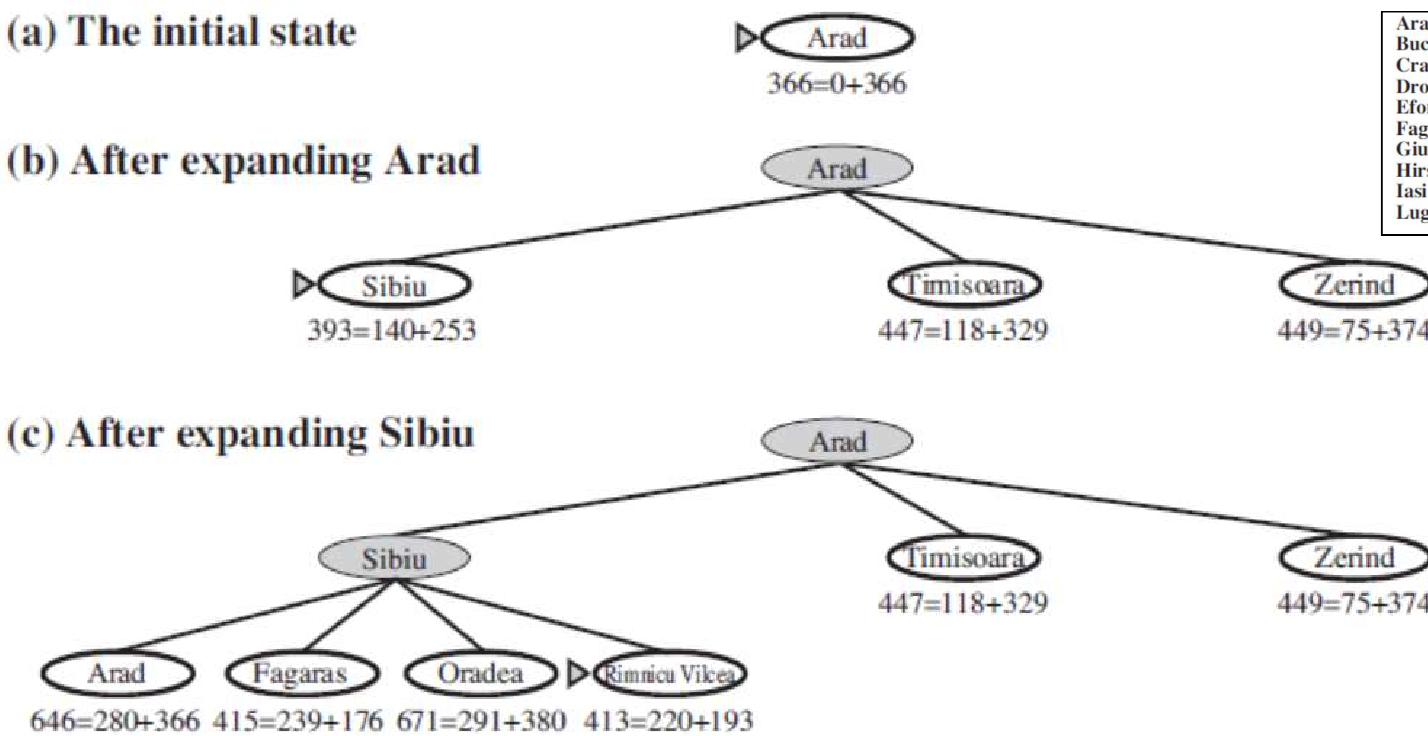
| | | | |
|---|---|---|---|
| Arad | 366 | Mehadia | 241 |
| Bucharest | 0 | Neamt | 234 |
| Craiova | 160 | Oradea | 380 |
| Drobeta | 242 | Pitesti | 100 |
| Eforie | 161 | Rimnicu Vilcea | 193 |
| Fagaras | 176 | Sibiu | 253 |
| Giurgiu | 77 | Timisoara | 329 |
| Hirsova | 151 | Urziceni | 80 |
|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

$$f(i) = \underbrace{g(i)}_{past} + \underbrace{h(i)}_{future}$$



# Informed search— A*

- A*: Expand the node with the lowest evaluation cost.

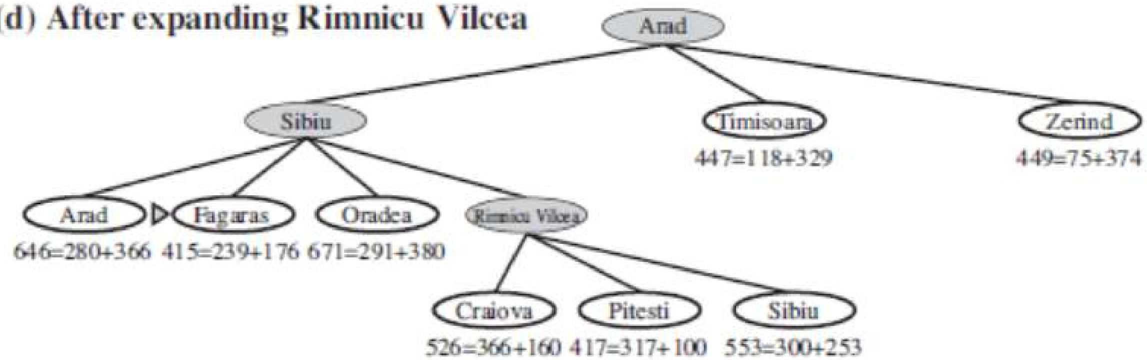



| | | | |
|---|---|---|---|
| Arad | 366 | Mehadia | 241 |
| Bucharest | 0 | Neamt | 234 |
| Craiova | 160 | Oradea | 380 |
| Drobeta | 242 | Pitesti | 100 |
| Eforie | 161 | Rimnicu Vilcea | 193 |
| Fagaras | 176 | Sibiu | 253 |
| Giurgiu | 77 | Timisoara | 329 |
| Hirsova | 151 | Urziceni | 80 |
|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

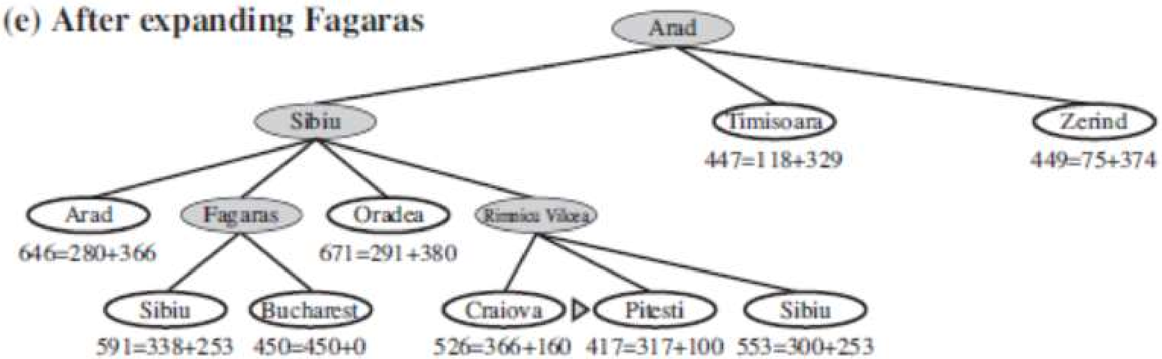
# Informed search— A*

- A*: Expand the node with the lowest evaluation cost.

(d) After expanding Rimnicu Vilcea



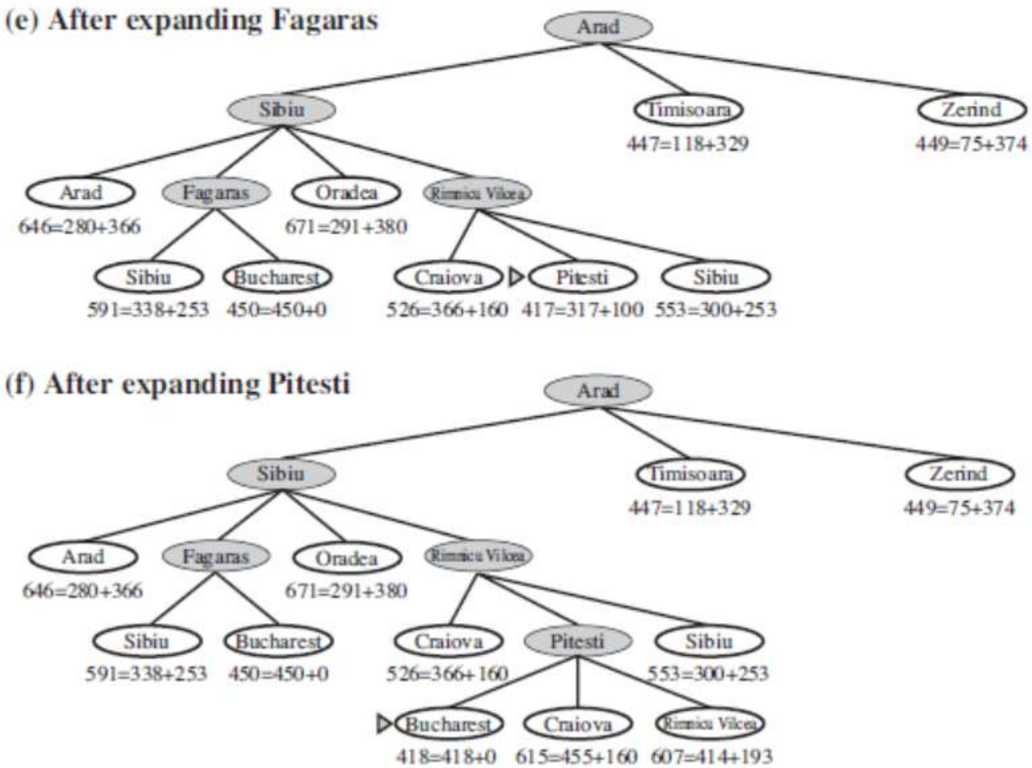

(e) After expanding Fagaras




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

# Informed search— A*

- A*: Expand the node with the lowest evaluation cost.




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

# Informed search— A*

- Conditions for optimality: Admissibility and Consistency
- Admissibility: An admissible heuristic is the one that ***never overestimates*** the cost to reach the goal.
- Consistency (monotonicity):

$$h(n) \leq c(n, a, n') + h(n')$$

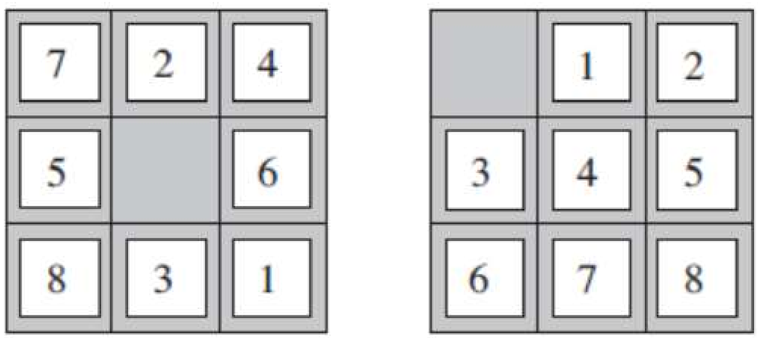
$where$ $n'$ is the next state of $n$ after action $a$

$c$ is the cost function

The proof of A* optimality can be found in AIMA P.95.

# Heuristic functions

- How to find a heuristic function is the key of heuristic search.
- Different heuristic functions could generate different search tree. We hope the number of branches is fewer.




Start State Goal State

$h_1 = \#$ of misplaced tiles

$h_2 =$ sum of the distance of the tiles from their goal positions. (Manhattan distance)

$$h_1 = 8$$

$$h_2 = 3+1+2+2+2+3+3+2 = 18$$

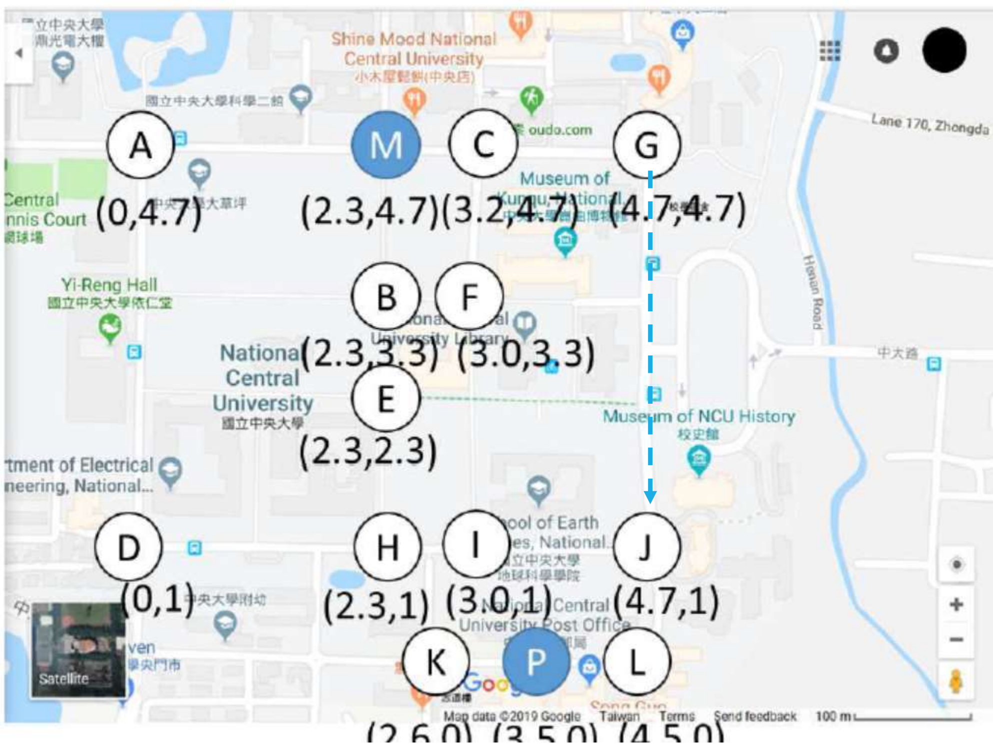
# Heuristic functions

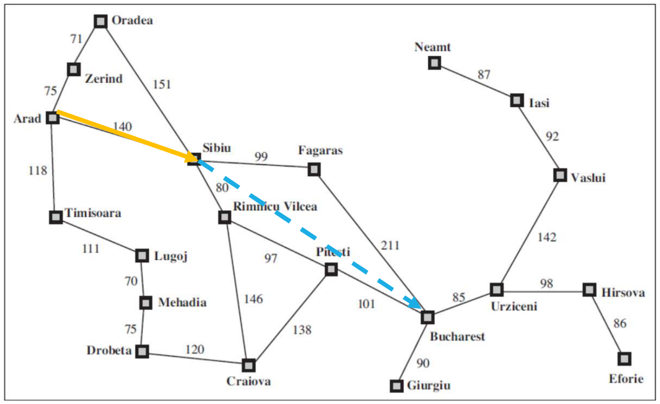
| | Search Cost (nodes generated) | | | Effective Branching Factor | | |
|---|---|---|---|---|---|---|
| $d$ | IDS | $A^*(h_1)$ | $A^*(h_2)$ | IDS | $A^*(h_1)$ | $A^*(h_2)$ |
| 2 | 10 | 6 | 6 | 2.45 | 1.79 | 1.79 |
| 4 | 112 | 13 | 12 | 2.87 | 1.48 | 1.45 |
| 6 | 680 | 20 | 18 | 2.73 | 1.34 | 1.30 |
| 8 | 6384 | 39 | 25 | 2.80 | 1.33 | 1.24 |
| 10 | 47127 | 93 | 39 | 2.79 | 1.38 | 1.22 |
| 12 | 3644035 | 227 | 73 | 2.78 | 1.42 | 1.24 |
| 14 | – | 539 | 113 | – | 1.44 | 1.23 |
| 16 | – | 1301 | 211 | – | 1.45 | 1.25 |
| 18 | – | 3056 | 363 | – | 1.46 | 1.26 |
| 20 | – | 7276 | 676 | – | 1.47 | 1.27 |
| 22 | – | 18094 | 1219 | – | 1.48 | 1.28 |
| 24 | – | 39135 | 1641 | – | 1.48 | 1.26 |

**Figure 3.29** Comparison of the search costs and effective branching factors for the ITERATIVE-DEEPENING-SEARCH and A* algorithms with $h_1$, $h_2$. Data are averaged over 100 instances of the 8-puzzle for each of various solution lengths $d$.

# Heuristic functions

- Admissibility: An admissible heuristic is the one that ***never overestimates*** the cost to reach the goal.



# Online Search

- The aforementioned search is offline search, which the agent has the environment information and searches for solutions.
- Online search is the case for unknown environments, where the robot doesn't know
  1. What the state is. $S : state \rightarrow ???$
  2. What the outcome of an action. $s, a \rightarrow ???$
- The robot needs to explore the environments and learn to find optimal actions via interaction.
- Online search is a deterministic case of *reinforcement learning*.

# Online Search

- Online search is that the robot first takes an action, then it observes the environment and compute the next action.
- The robot does trial-and-error actions to improve its solutions.

$[GIVEN]$

$A : action$

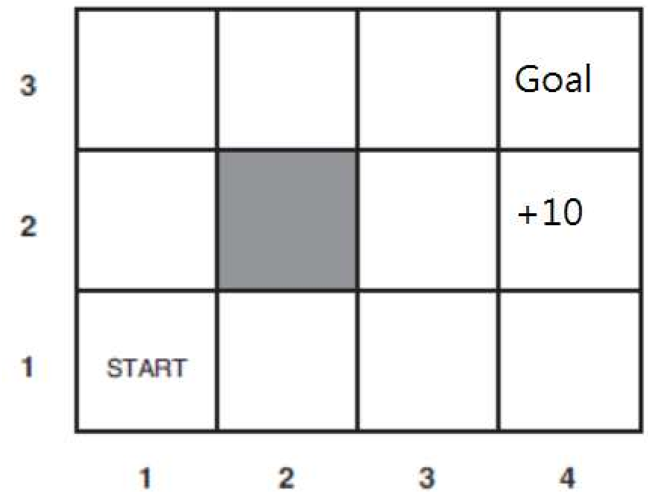
$c(s, a, s') : \text{step} - \text{cost function}$

$Goal - Test(s)$

[Find]

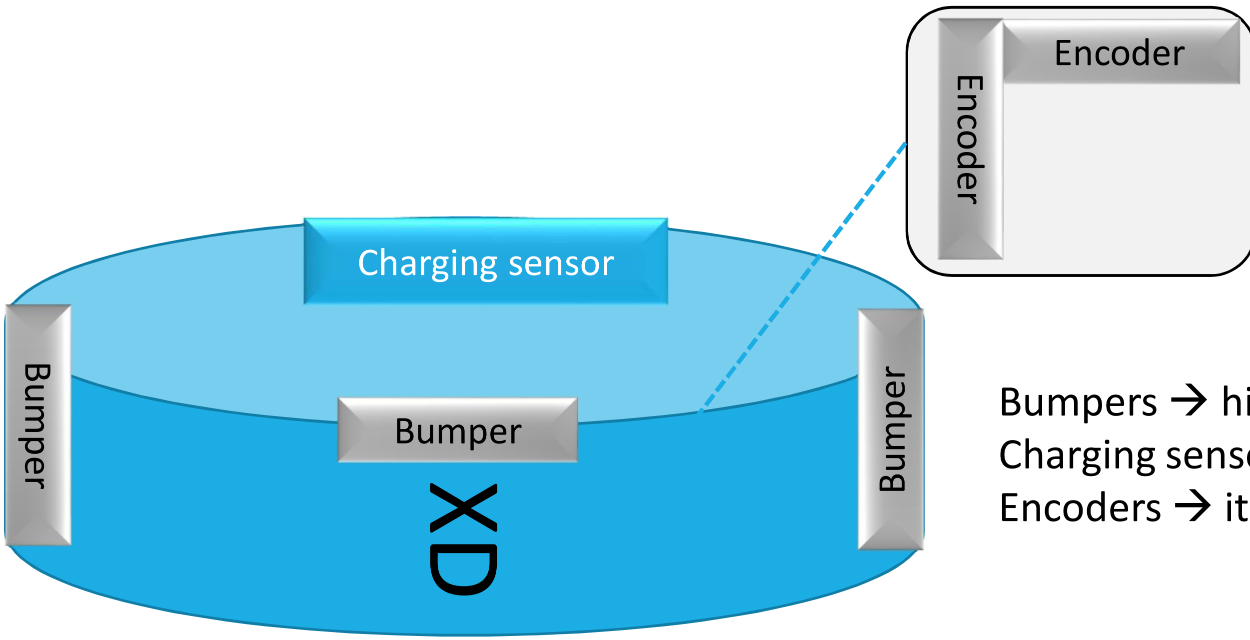
The optimal (lowest cost) path

$[Environment]$



# Online Search

- There is a cleaner robot with bumpers, charging sensors and encoders. Its goal is to move to the charging station with the lowest cost.

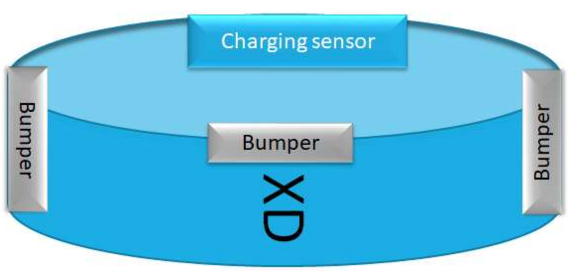


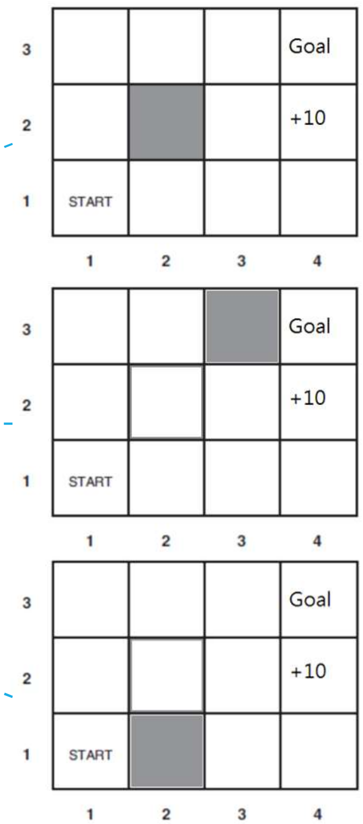

$$\Rightarrow \begin{cases} V_x \\ V_y \end{cases} \Rightarrow \begin{cases} x \\ y \end{cases}$$

Bumpers → hitting something?
Charging sensors → is charging?
Encoders → its location?

# Online Search

- The robot does trial-and-error actions to improve its solutions.
- Why does the robot need to try?




It needs to learn to work in different environments!

# Online Search

- There is a cleaner robot with bumpers, charging sensors and encoders. Its goal is to move to the charging station with the lowest cost.

$[GIVEN]$

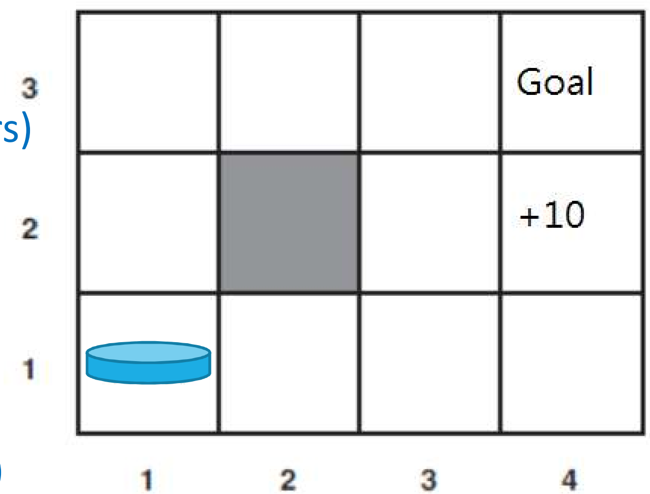
$A: action = \{left, up, right\}$

$c(s, a, s')$ : step − cost function (Encoders, bumpers)

$$\begin{cases} \text{hit wall} \rightarrow +2 \\ \text{hit } (4,2) \rightarrow +10 \\ others \rightarrow +1 \end{cases}$$

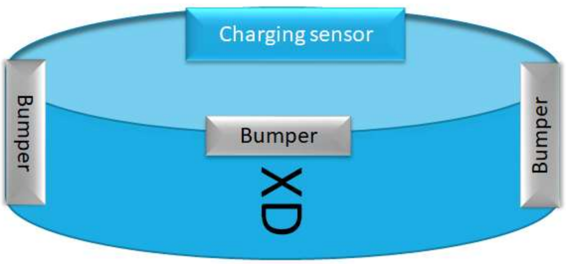
$Goal - Test(s)$ (charging sensors)

$[Environment]$

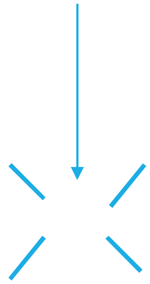


# Online Search

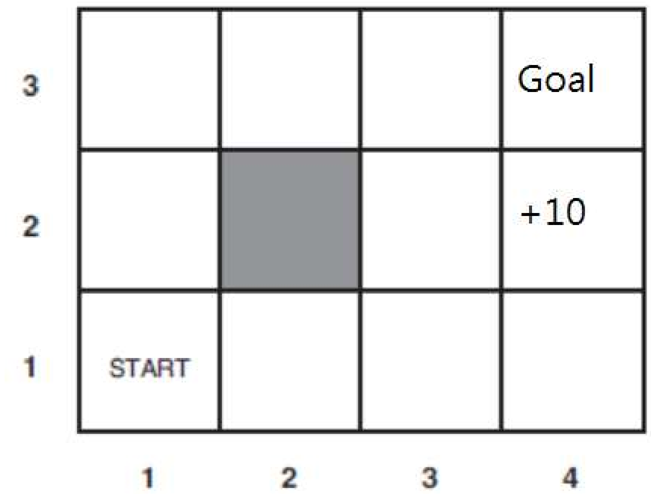
- There is a cleaner robot with bumpers, charging sensors  and encoders. Its goal is to move to the charging station with the lowest cost.




This is what the robot sees!

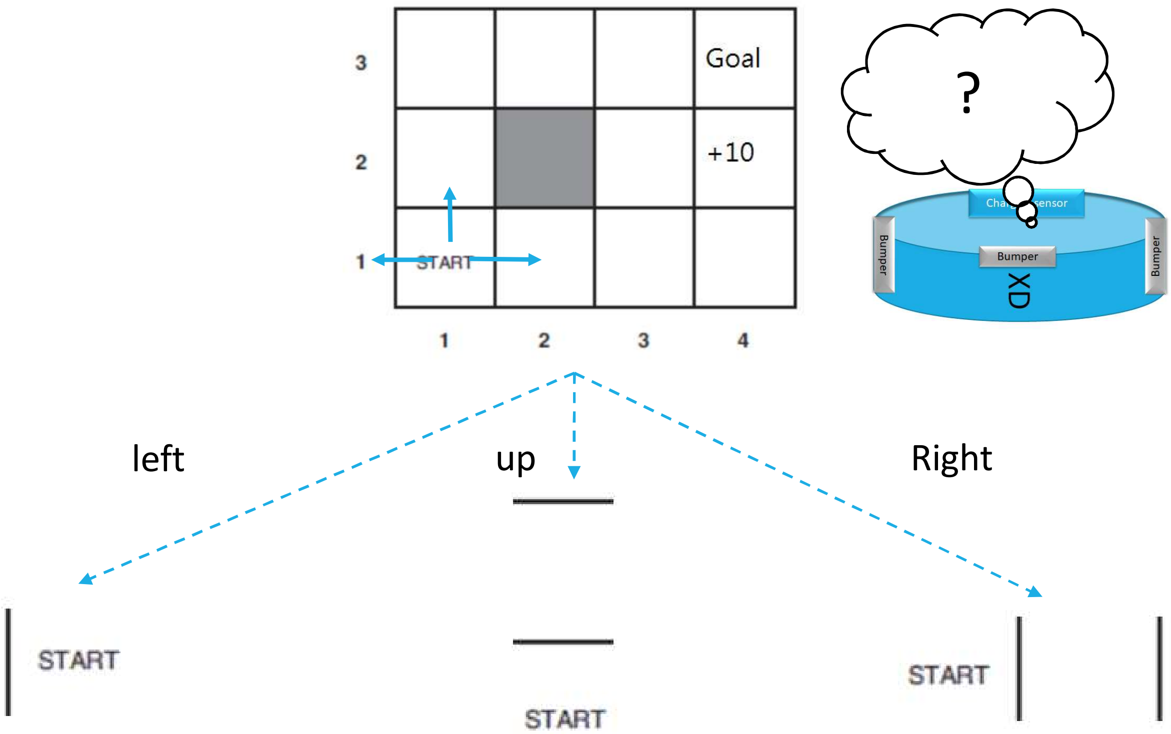


This is what you see!



# Online Search

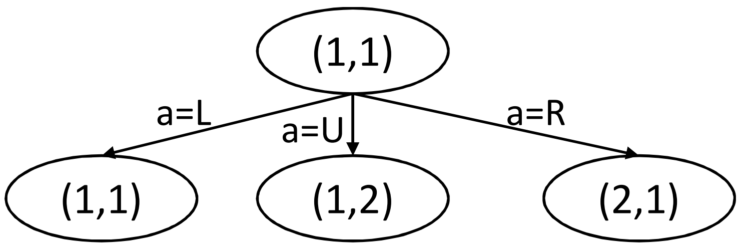
- Illustration of the robot’s world




Bumpers → hitting something?
Charging sensors → is charging?
Encoders → its location?

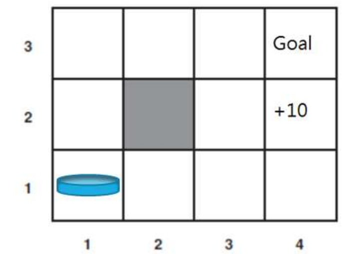
# Online Search

- The robot has to learn the cost via sequential actions.




$c(s,a,s') = 2 \quad c(s,a,s') = 1 \quad c(s,a,s') = 1$

(This is just an illustration of offline search. In online search, the robot cannot know the children nodes without experience.)




We want to apply A* to online search. The robot needs to ***remember*** its experience.

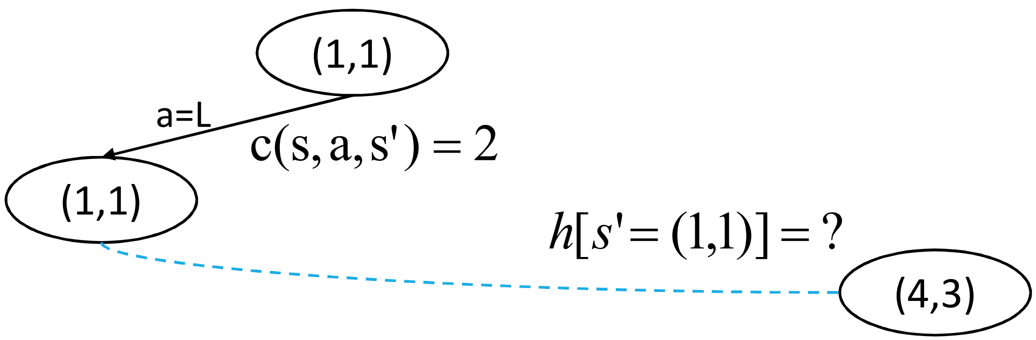
$$f(i) = \underbrace{g(i)}_{past} + \underbrace{h(i)}_{future}$$

$$h : \text{manhattan distance}$$

If it's a offline search. We can build a tree or graph and then apply A* to find the optimal actions.

# LRTA*

- Let’s think about how to revise A* for online search
- For offline search, the robot has the environment information. However, for online search, the robot doesn’t know how large the environment is. Hence, the robot needs to adopt a memory efficiency way – Markov chain.
- The state (s’) at time t+1 only depends on the state (s) at time t.

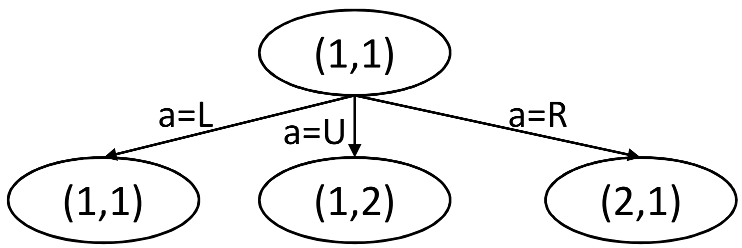



$$f(i) = \underbrace{g(i)}_{past} + \underbrace{h(i)}_{future}$$

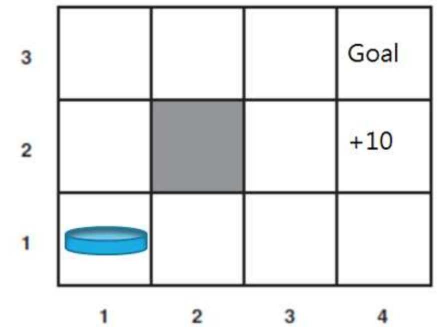
$$\mathrm{H}(s) = \underbrace{c(s,a,s')}_{past} + \underbrace{h(s')}_{future}$$

# LRTA*

- Assume the robot has a heuristic function. The robot should choose the action with the lowest value of evaluation function




$c(s,a,s')=2 \quad c(s,a,s')=1 \quad c(s,a,s')=1$




$$\mathrm{H(s)} = \min_a \left[ \mathrm{c(s,a,s')} + H(s') \right]$$

$$\mathrm{H(s)} = \min_a \begin{cases} 2+5=7 & \text{(Left)} \\ 1+4=5 & \text{(Up)} \\ 1+4=5 & \text{(Right)} \end{cases}$$

The robot should choose Up or Right. However, the assumption is the **cost** and **action outcome** are known. In online search, the robot needs to explore them!

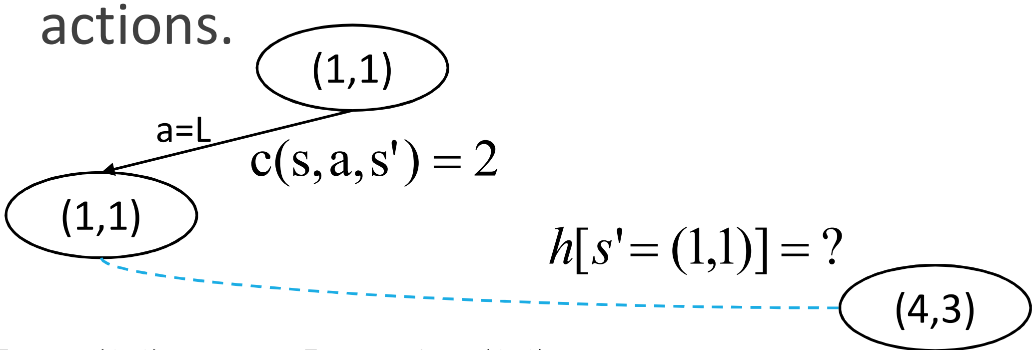
# LRTA*

- How could the robot explore the environment or act optimally based on an explored H table?
- This is a trade-off between exploration and exploitation.
- **Exploration** is to explore unknown states while **exploitation** is to utilize known knowledge.

1. ε-greedy was proposed to solve this issue. The robot will action randomly with ε probability and select best action with (1- ε) probability.
2. Another way is to explore everywhere. Set unexplored H(s) as a small value.

# LRTA*

- The robot needs to ***remember*** its past experience to improve its actions.




$$[s = (1,1), a = L] \rightarrow s' = (1,1)$$

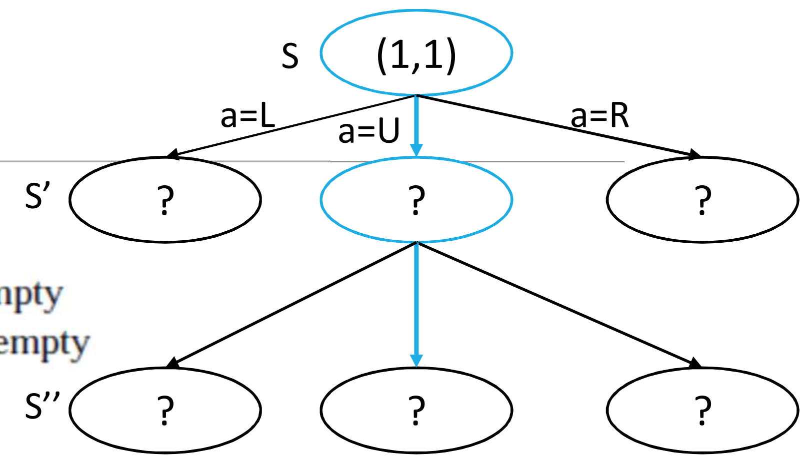
1. ***remember*** the result table.

2. ***remember*** the estimated cost table.

The result table tells the robot the next state (s') given (s,a)
while the H table tells the robot what's the estimated cost at s.
Then, the robot can learn to find a solution!
It's called LRTA*[1], which is a special case of reinforcement learning.
**LRTA* is the bridge between classic AI and modern AI.**

[1] R.E. Korf, "Real-time heuristic search," Artificial Intelligence, 1990.

# LRTA*

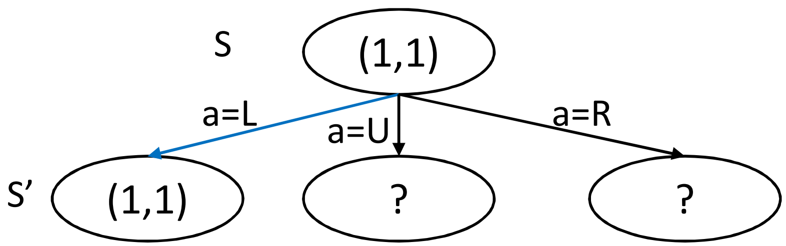


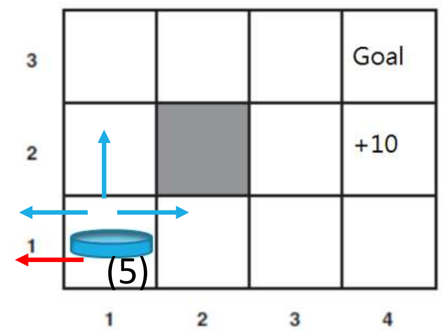

**function** LRTA*-AGENT($s'$) **returns** an action
  **inputs**: $s'$, a percept that identifies the current state
  **persistent**: *result*, a table, indexed by state and action, initially empty
    $H$, a table of cost estimates indexed by state, initially empty
    $s$, $a$, the previous state and action, initially null

  **if** GOAL-TEST($s'$) **then return** *stop*
  **if** $s'$ is a new state (not in $H$) **then** $H[s'] \leftarrow h(s')$ (for exploration)
  **if** $s$ is not null
    $result[s, a] \leftarrow s'$ → [Past] Update result table
    $H[s] \leftarrow \min_{b \in \text{ACTIONS}(s)} \text{LRTA*-COST}(s, b, result[s, b], H)$ → Update H table
  $a \leftarrow$ an action $b$ in ACTIONS($s'$) that minimizes LRTA*-COST($s'$, $b$, $result[s', b]$, $H$) → [Future] Select action
  $s \leftarrow s'$
  **return** $a$

**function** LRTA*-COST($s, a, s', H$) **returns** a cost estimate
  **if** $s'$ is undefined **then return** $h(s)$ Set heuristic as estimated cost (for exploration)
  **else return** $c(s, a, s') + H[s']$ (past cost + Heuristic )

# LRTA*




$$c(s,a,s')=2$$

$$H[s'=(1,1)]=5$$

$$s'=(1,1)\leftarrow[s=(1,1),a=L]$$

$$\begin{cases} b=L\rightarrow H[s=(1,1)]=2+5 \\ b=U\rightarrow undefined, h[s=(1,1)]=5 \\ b=R\rightarrow undefined, h[s=(1,1)]=5 \end{cases}$$

$$H[s=(1,1)]\leftarrow 5$$

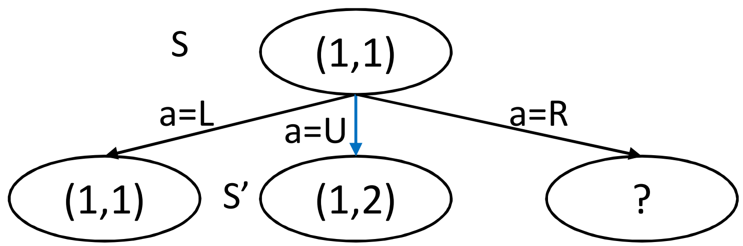
$$a\leftarrow U$$

$$s\leftarrow s'=(1,1)$$

**if** $s$ is not null
  $result[s,a]\leftarrow s'$
  $H[s]\leftarrow \min_{b\in \text{ACTIONS}(s)} \text{LRTA*-COST}(s,b,result[s,b],H)$
$a\leftarrow$ an action $b$ in ACTIONS($s'$) that minimizes LRTA*-COST($s',b,result[s',b],H$)
$s\leftarrow s'$

$$\begin{cases} b=L\rightarrow H[s''=(1,1)]=7 \\ b=U\rightarrow undefined, h[s'=(1,1)]=5 \\ b=R\rightarrow undefined, h[s'=(1,1)]=5 \end{cases}$$

# LRTA*




$$c(s,a,s')=1$$

$$H[s'=(1,2)]=4$$

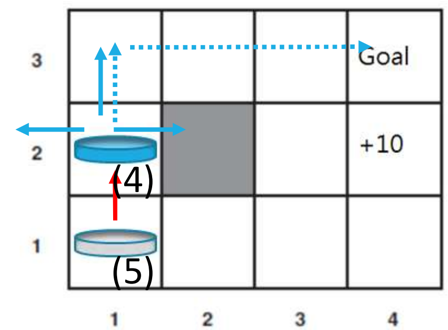
$$s'=(1,2)\leftarrow[s=(1,1),a=U]$$

$$\begin{cases} b=L\rightarrow H[s=(1,1)]=2+5 \\ b=U\rightarrow H[s=(1,2)]=1+4=5 \\ b=R\rightarrow undefined, h[s=(1,1)]=5 \end{cases}$$

$$H[s=(1,1)]\leftarrow 5$$

$$a\leftarrow L$$

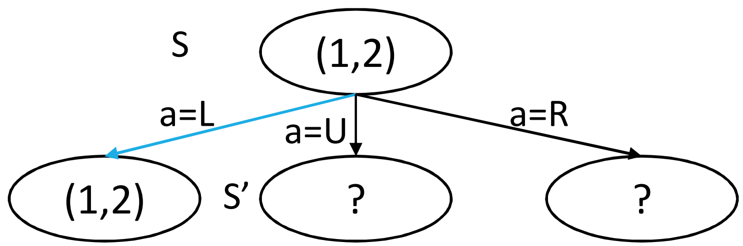
$$s\leftarrow s'=(1,2)$$

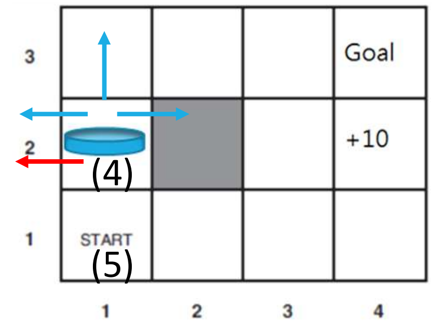



**if** $s$ is not null
  $result[s,a]\leftarrow s'$
  $H[s]\leftarrow \min_{b\in \text{ACTIONS}(s)} \text{LRTA*-COST}(s,b,result[s,b],H)$
$a\leftarrow$ an action $b$ in ACTIONS($s'$) that minimizes LRTA*-COST($s', b, result[s',b], H$)
$s\leftarrow s'$

$$\begin{cases} b=L\rightarrow undefined, h[s'=(1,2)]=4 \\ b=U\rightarrow undefined, h[s'=(1,2)]=4 \\ b=R\rightarrow undefined, h[s'=(1,2)]=4 \end{cases}$$

# LRTA*




$$c(s,a,s')=2$$

$$s'=(1,2)\leftarrow[s=(1,2),a=L]$$

$$\begin{cases} b=L\rightarrow H[s=(1,2)]=2+4 \\ b=U\rightarrow undefined, h[s=(1,2)]=4 \\ b=R\rightarrow undefined, h[s=(1,2)]=4 \end{cases}$$

$$H[s=(1,2)]\leftarrow 4$$

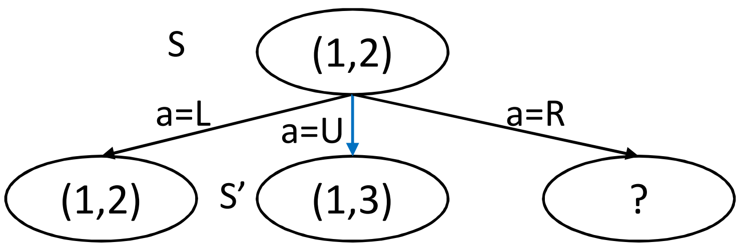
$$a\leftarrow U$$

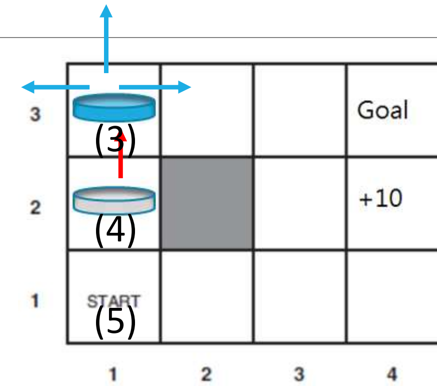
$$s\leftarrow s'=(1,2)$$

**if** $s$ is not null
  $result[s,a]\leftarrow s'$
  $H[s]\leftarrow \min\limits_{b\in \text{ACTIONS}(s)} \text{LRTA*-COST}(s,b,result[s,b],H)$
$a\leftarrow$ an action $b$ in ACTIONS$(s')$ that minimizes LRTA*-COST$(s',b,result[s',b],H)$
$s\leftarrow s'$

$$\begin{cases} b=L\rightarrow 2+h[s'=(1,2)]=6 \\ b=U\rightarrow undefined, h[s'=(1,2)]=4 \\ b=R\rightarrow undefined, h[s'=(1,2)]=4 \end{cases}$$

# LRTA*




$$c(s,a,s')=1$$

$$H[s'=(1,3)]=h[s'=(1,3)]=3$$

$$s'=(1,3)\leftarrow[s=(1,2),a=U]$$

$$\begin{cases} b=L\rightarrow H[s=(1,2)]=2+4 \\ b=U\rightarrow H[s=(1,2)]=1+4 \\ b=R\rightarrow undefined, h[s=(1,2)]=4 \end{cases}$$

$$H[s=(1,2)]\leftarrow 4$$

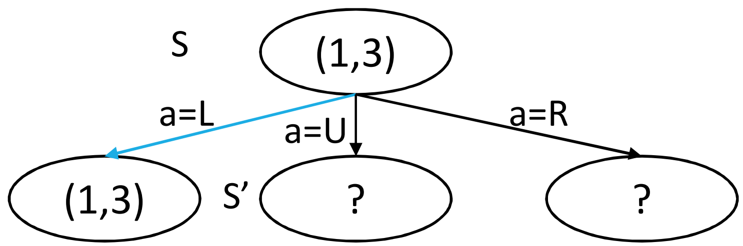
$$a\leftarrow L$$

$$s\leftarrow s'=(1,3)$$

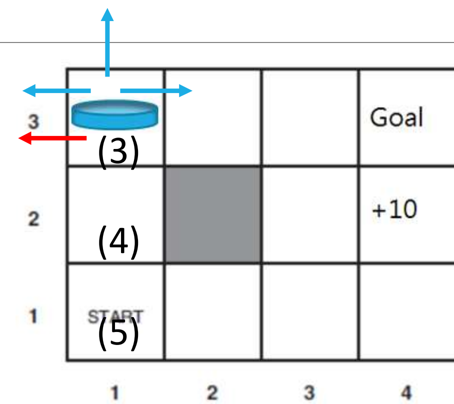
**if** $s$ **is not null**
  $result[s,a]\leftarrow s'$
  $H[s]\leftarrow \min_{b\in \text{ACTIONS}(s)} \text{LRTA*-COST}(s,b,result[s,b],H)$
$a\leftarrow$ an action $b$ in $\text{ACTIONS}(s')$ that minimizes $\text{LRTA*-COST}(s',b,result[s',b],H)$
$s\leftarrow s'$

$$\begin{cases} b=L\rightarrow undefined, h[s'=(1,3)]=3 \\ b=U\rightarrow undefined, h[s'=(1,3)]=3 \\ b=R\rightarrow undefined, h[s'=(1,3)]=3 \end{cases}$$

# LRTA*




$$c(s,a,s')=2$$




$$s'=(1,3)\leftarrow[s=(1,3),a=L]$$

$$\begin{cases} b=L\rightarrow H[s=(1,3)]=2+3 \\ b=U\rightarrow undefined, h[s=(1,3)]=3 \\ b=R\rightarrow undefined, h[s=(1,3)]=3 \end{cases}$$

$$H[s=(1,3)]\leftarrow 3$$

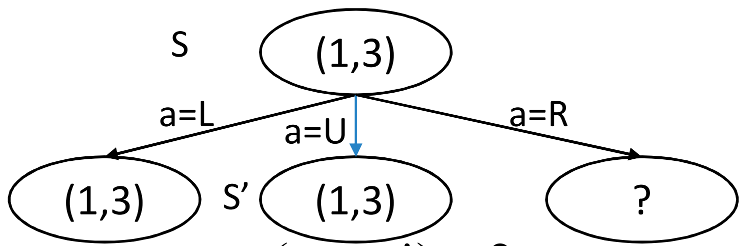
$$a\leftarrow U$$

$$s\leftarrow s'=(1,3)$$

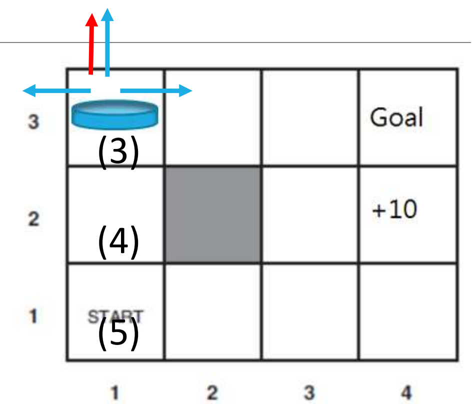
**if** $s$ is not null
  $result[s,a]\leftarrow s'$
  $H[s]\leftarrow \min_{b\in\text{ACTIONS}(s)} \text{LRTA*-COST}(s,b,result[s,b],H)$
$a\leftarrow$ an action $b$ in ACTIONS($s'$) that minimizes LRTA*-COST($s',b,result[s',b],H$)
$s\leftarrow s'$

$$\begin{cases} b=L\rightarrow 2+h[s'=(1,3)]=5 \\ b=U\rightarrow undefined, h[s'=(1,3)]=3 \\ b=R\rightarrow undefined, h[s'=(1,3)]=3 \end{cases}$$

# LRTA*




$$c(s,a,s')=2$$




$$s'=(1,3)\leftarrow[s=(1,3),a=U]$$

$$\begin{cases} b=L\rightarrow H[s=(1,3)]=2+3 \\ b=U\rightarrow H[s=(1,3)]=2+3 \\ b=R\rightarrow undefined, h[s=(1,3)]=3 \end{cases}$$

$$H[s=(1,3)]\leftarrow 3$$

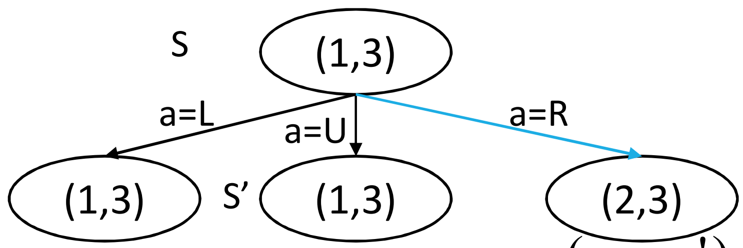
$$a\leftarrow R$$

$$s\leftarrow s'=(1,3)$$

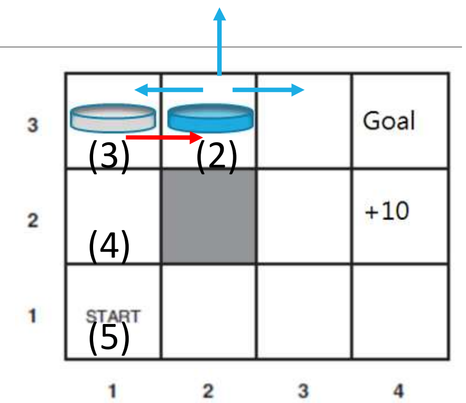
**if** $s$ is not null
  $result[s,a]\leftarrow s'$
  $H[s]\leftarrow \min_{b\in \text{ACTIONS}(s)} \text{LRTA*-COST}(s,b,result[s,b],H)$
$a\leftarrow$ an action $b$ in ACTIONS$(s')$ that minimizes LRTA*-COST$(s',b,result[s',b],H)$
$s\leftarrow s'$

$$\begin{cases} b=L\rightarrow 2+h[s'=(1,3)]=5 \\ b=U\rightarrow 2+h[s'=(1,3)]=5 \\ b=R\rightarrow undefined, h[s'=(1,3)]=3 \end{cases}$$

# LRTA*




$$c(s,a,s')=1$$




$$H[s'=(2,3)]=h[s'=(2,3)]=2$$

$$s'=(2,3)\leftarrow[s=(1,3),a=R]$$

$$\begin{cases} b=L\rightarrow H[s=(1,3)]=2+3 \\ b=U\rightarrow H[s=(1,3)]=2+3 \\ b=R\rightarrow H[s=(1,3)]=1+2 \end{cases}$$

$$H[s=(1,3)]\leftarrow 3$$

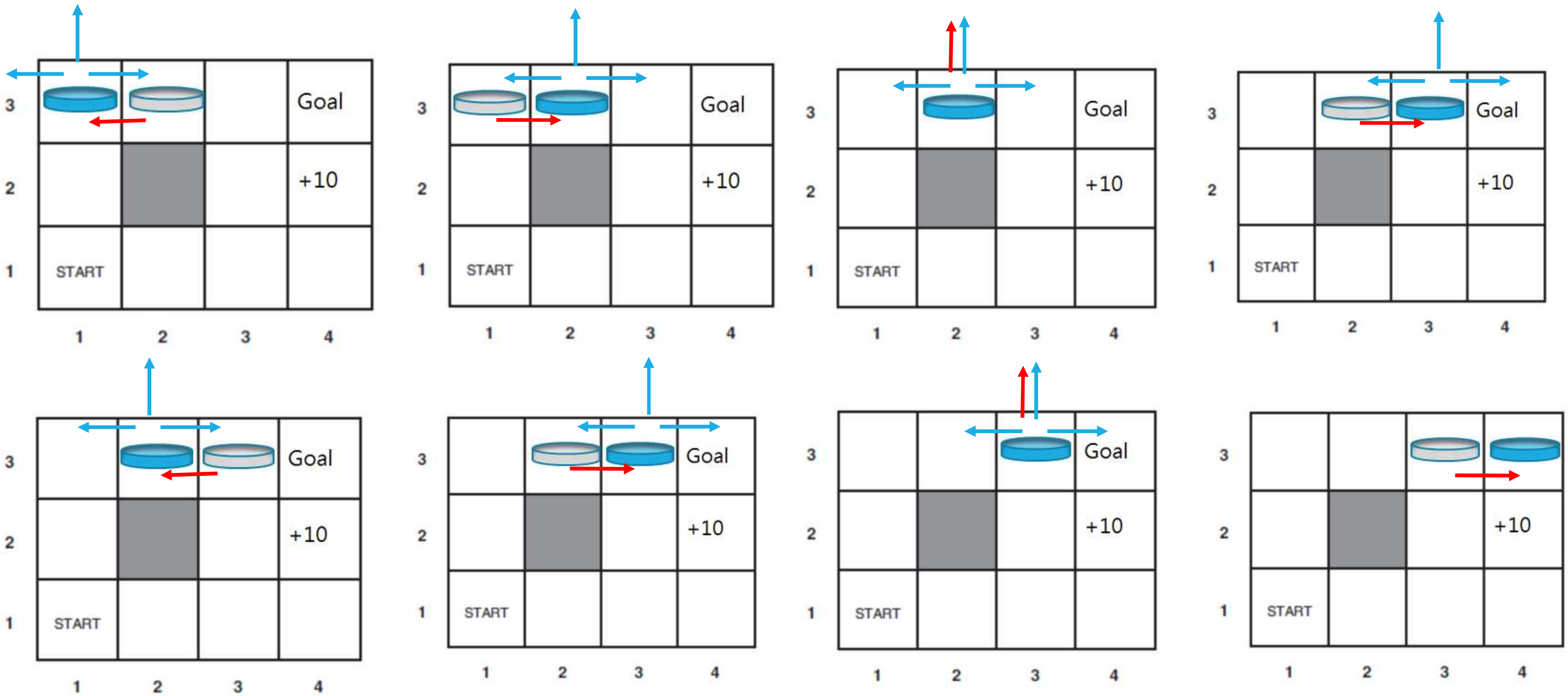
$$a\leftarrow L$$

$$s\leftarrow s'=(2,3)$$

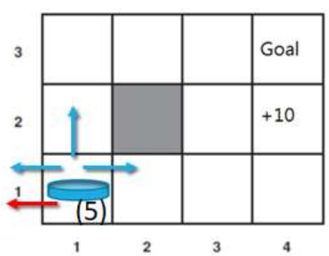
**if** $s$ **is not null**
  $result[s,a] \leftarrow s'$
  $H[s] \leftarrow \min_{b \in \text{ACTIONS}(s)} \text{LRTA*-COST}(s, b, result[s,b], H)$
$a \leftarrow$ an action $b$ in $\text{ACTIONS}(s')$ that minimizes $\text{LRTA*-COST}(s', b, result[s',b], H)$
$s \leftarrow s'$

$$\begin{cases} b=L\rightarrow undefined, h[s'=(2,3)]=2 \\ b=U\rightarrow undefined, h[s'=(2,3)]=2 \\ b=R\rightarrow undefined, h[s'=(2,3)]=2 \end{cases}$$

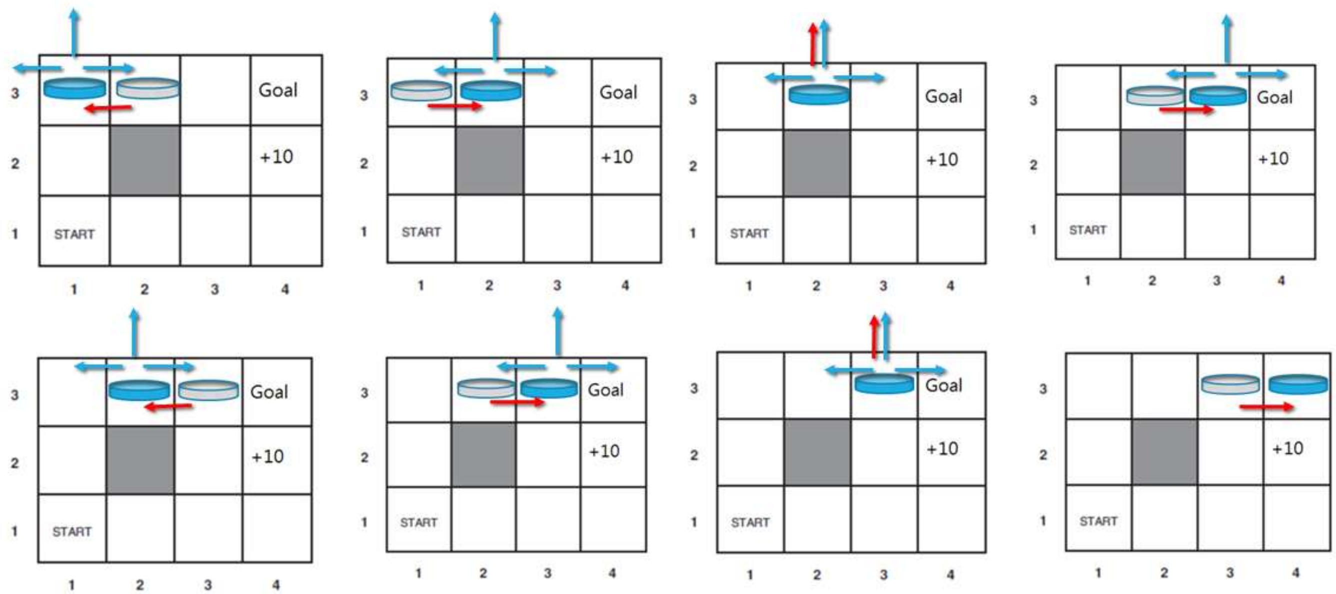
# LRTA*



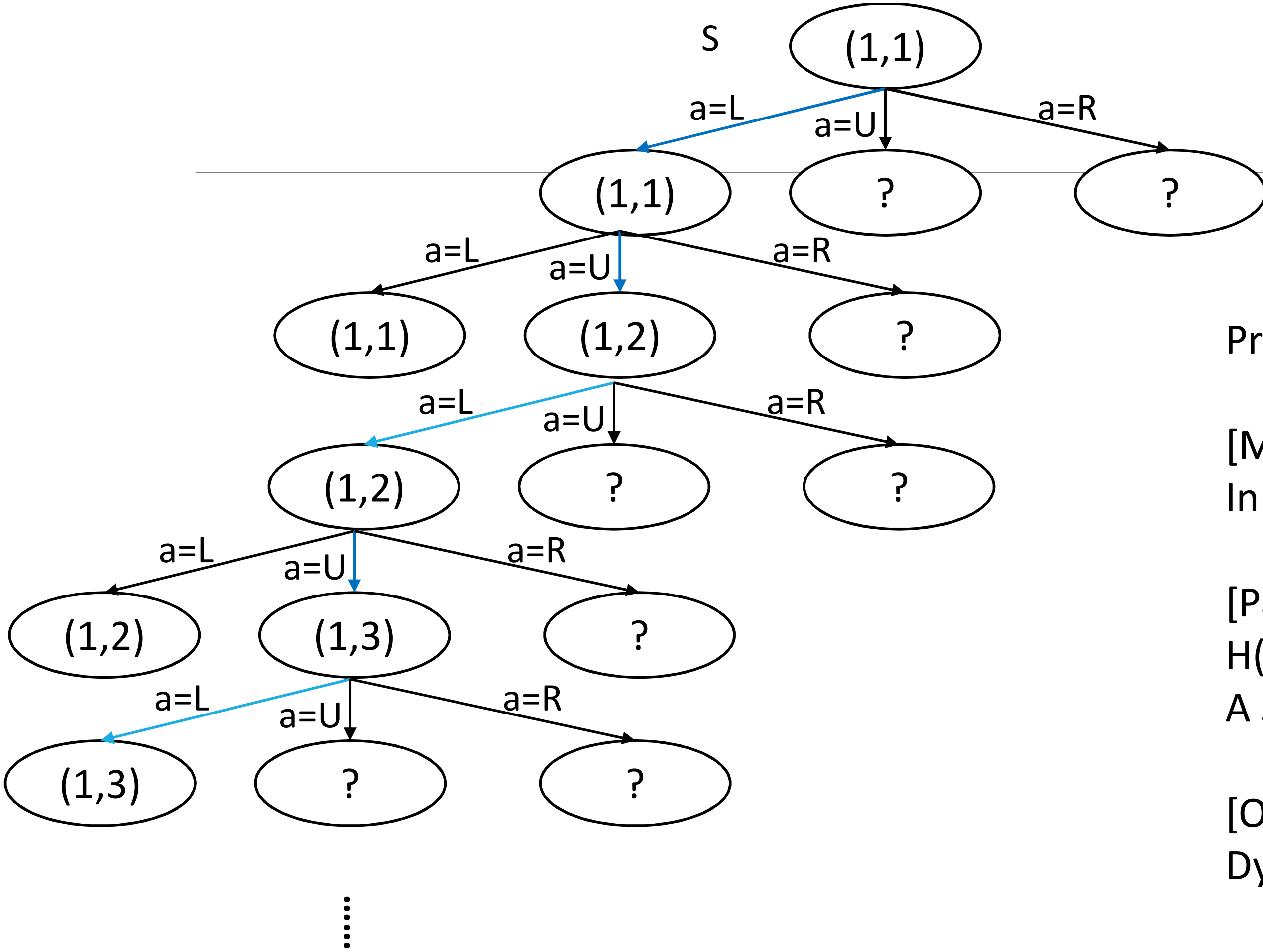
# LRTA*




....




This is episode #1.
The robot built the H table and results table.
The robot can try more episodes to improve its actions.
Or use more exploration actions to find more possible solutions.




Properties of LRTA*

[Markov Chain]
In LRTA*, the s’ only depends on s and a.

[Past + Future cost]
H(s)=c(s,a,s’)+h(s’)
A special case of Bellman equation

[Optimality]
Dynamic programming?

# LRTA*

- Recalled optimality of A*.
- Conditions for optimality: Admissibility and Consistency
- Admissibility: An admissible heuristic is the one that ***never overestimates*** the cost to reach the goal.
- Consistency (monotonicity):

$$h(n) \leq c(n, a, n') + h(n')$$

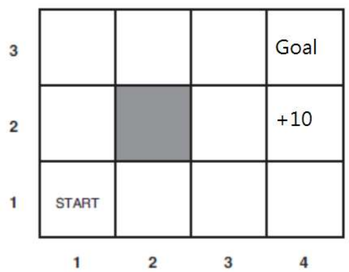
$where\ n'$ is the next state of $n$ after action $a$

$c$ is the cost function

# LRTA* and Reinforcement Learning

- LRTA*

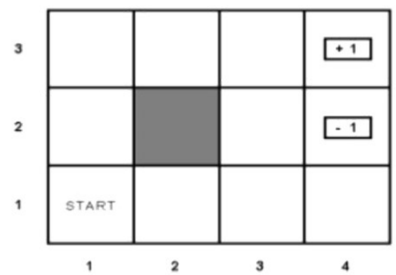
Deterministic action




$$s, a \rightarrow s'$$

MDP (RL)

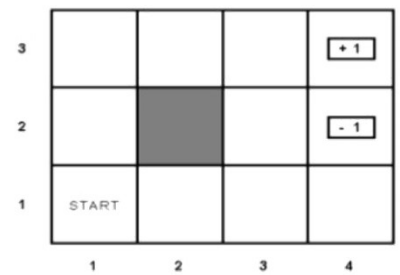
Probabilistic actions




$$P(s' | s, a)$$

POMDP

Probabilistic actions and states




$$P(s' | s, a), P(s)$$

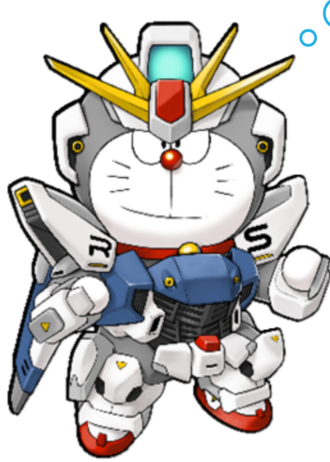
L2: Uninformed search
L3: Heuristic search
L4: Adversarial search
L5: Bayes theorem
L6: Bayes theorem over time

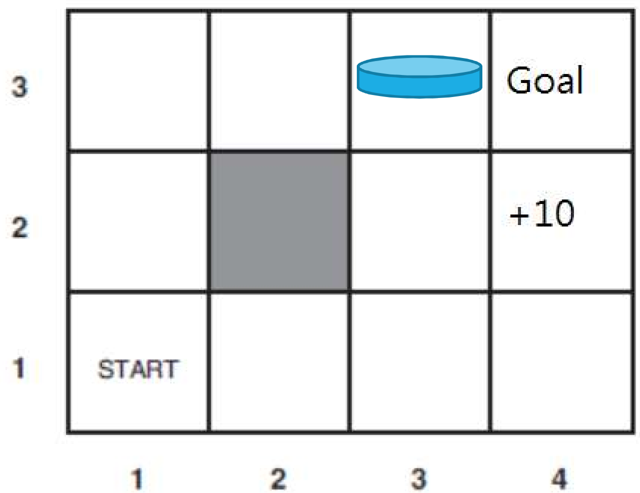
(LRTA*)

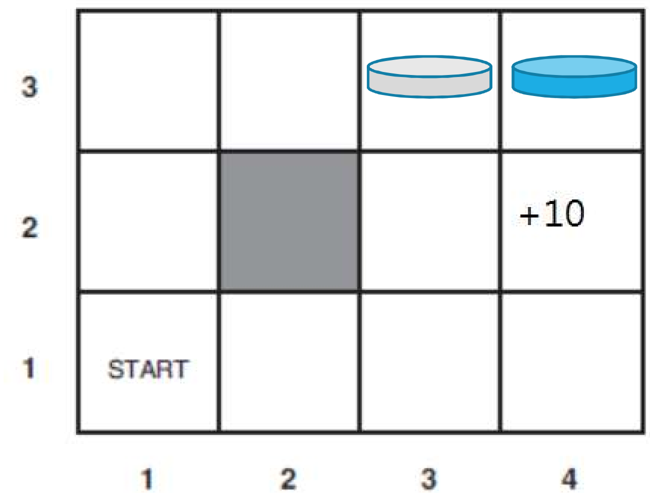
L7: MDP
L9: Reinforcement learning
L10: GP and LWPR
L11: Naïve Bayes and Perceptron
L12: Adaboost
L13: Deep learning and DRL

L8: POMDP

# Q&A




Goal
+10
START
3
2
1
1
2
3
4


+10
START
3
2
1
1
2
3
4