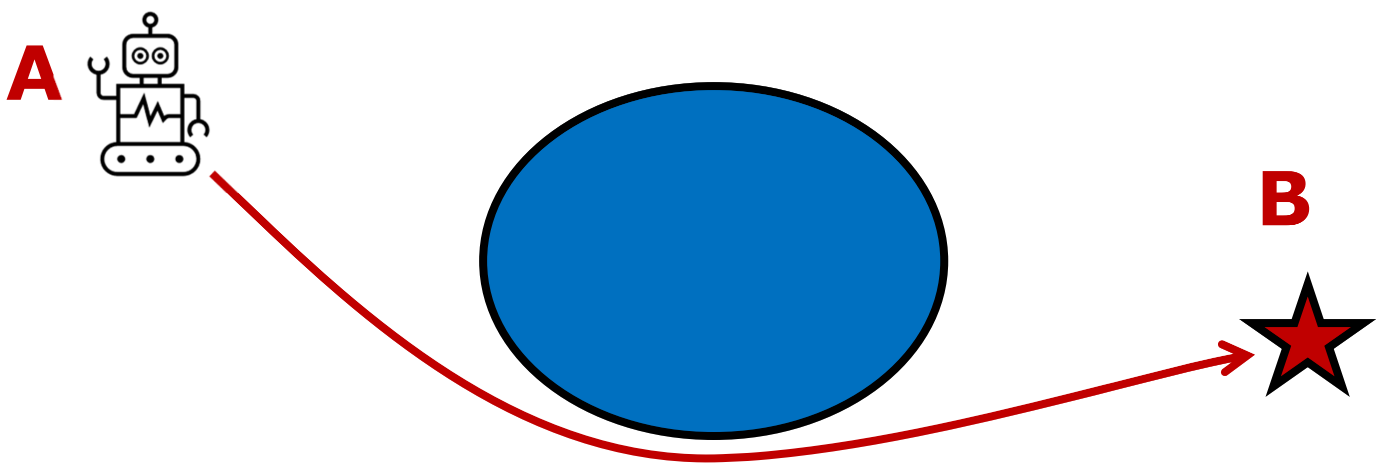
4 – Search-based Methods

Dr. Marija Popović

UNIVERSITÄT BONN

Review: What is Planning?

- Find a sequence of valid configurations to move a robot from point A to point B – *how*?
- **Classical path planning**: What is the shortest geometric path?



Review: Planning Methods

- Geometric
- Potential field
- Search-based
- Sampling-based
- Trajectory
- Bioinspired

Review: Planning Methods

- Geometric
- Potential field
- Search-based
- Sampling-based
- Trajectory
- Bioinspired

Problem Statement

- **Given**:
  - Discrete representation of the environment
  - Robot model
  - Start and goal configurations
- Find a sequence of configurations to move the robot from start to goal
- Exclude uncertainty first

- **Search-based**: Explore the environment systematically under given rules

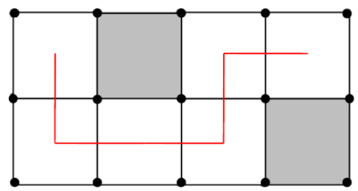
Environment

- Discrete representations often appear in the form of **graph**
- A graph is an ordered pair $G = (V, E)$
  - V is a set of vertices
  - E is a set of edges

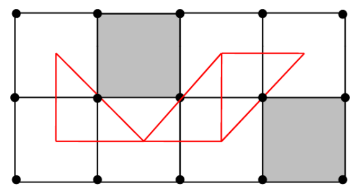
$$E \subseteq \{\{x, y\} \mid x, y \in V; x \neq y\}$$

- **Grid map**: Special case of graphs

4 neighbours



8 neighbours



https://cw.fel.cvut.cz/old/_media/courses/b4m36uir/lectures/b4m36uir-lec04-handout.pdf

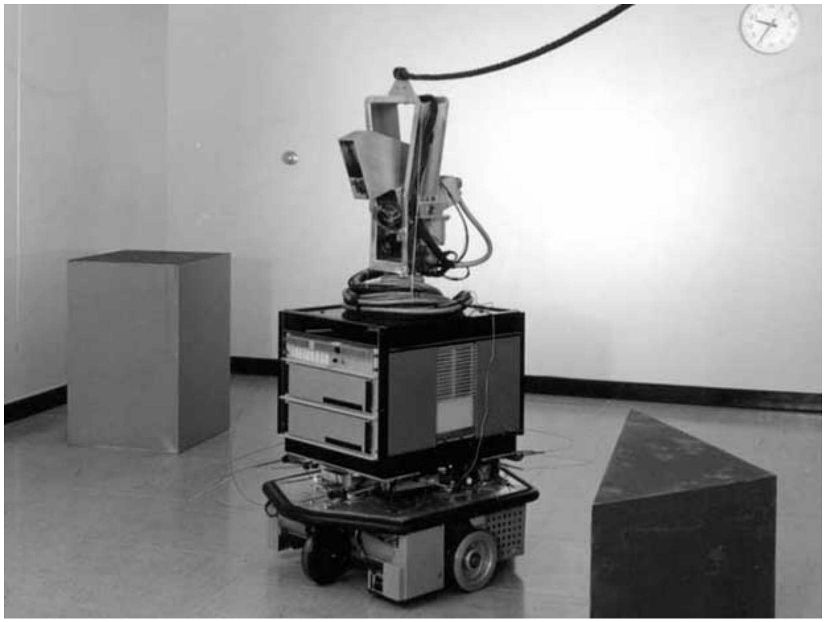
Environment

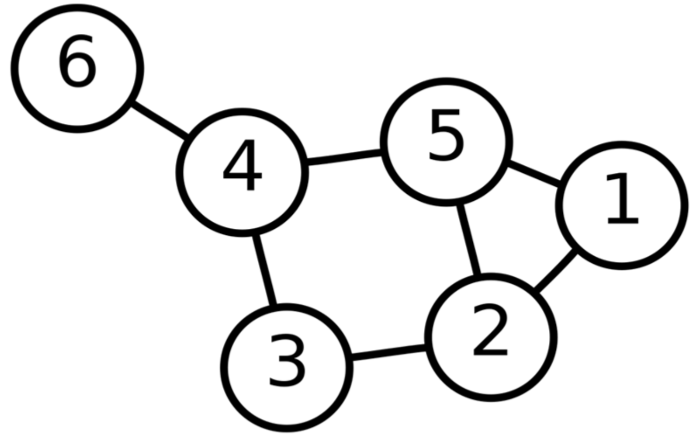
- Discrete representations often appear in the form of **graph**
- A graph is an ordered pair $G = (V, E)$
  - V is a set of vertices
  - E is a set of edges

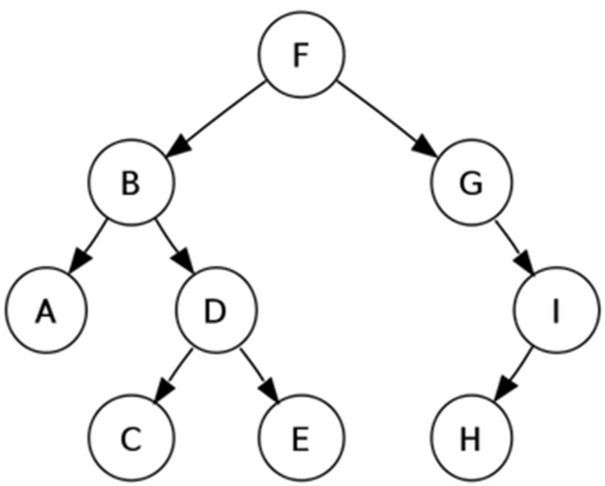
$$E \subseteq \{\{x, y\} \mid x, y \in V; x \neq y\}$$

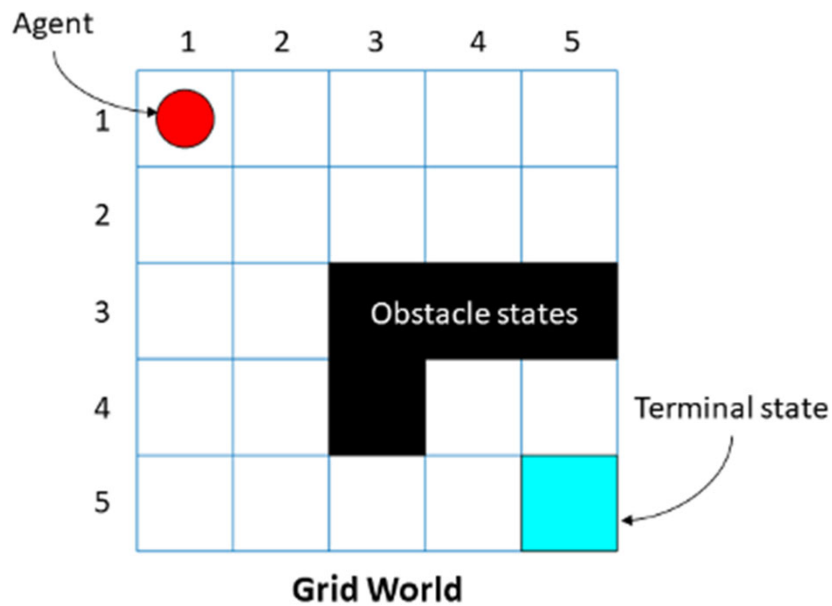
- **Grid map**: Special case of graphs
- **Tree**: Minimally connected graph which must be connected and free from loops

Environment





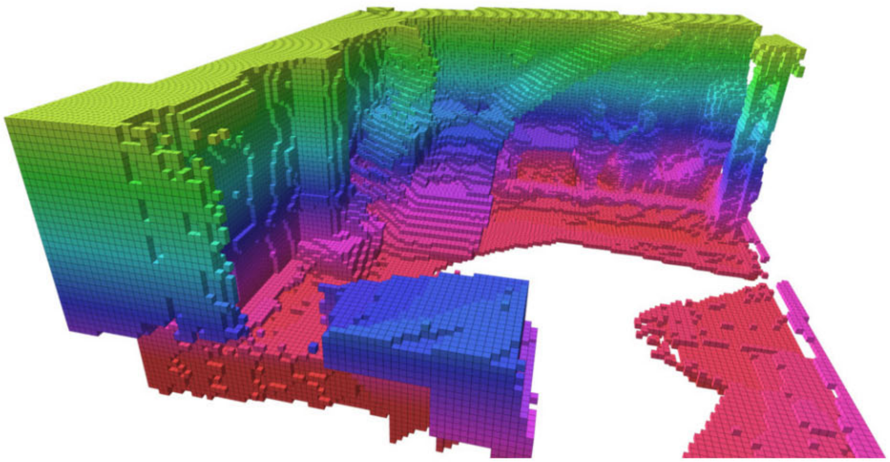




https://lifeboat.com/blog/2017/02/straight-out-of-sci-fi-shakey-was-the-first-mobile-robot-built-with-ai

Environment

- In 3D: Octomap



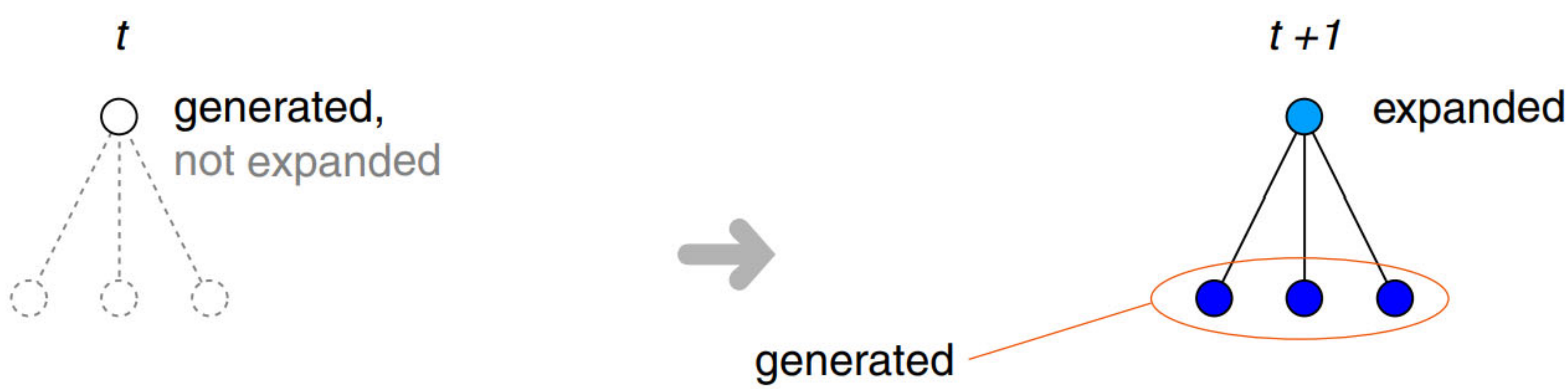
Hornung et al. (2013), “OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees,” in: Autonomous Robots.
Dai et al. (2020), “Fast Frontier-based Information-driven Autonomous Exploration with an MAV,” in: IEEE ICRA. pp.9570-9576.

Criteria

- **Completeness**: The algorithm can always find a solution when a solution exists
  - Resolution completeness
- **Optimality**: The solution is the best one of all possible solutions in terms of pre-defined cost
- **Time complexity**: Computational burden of the search algorithm
- **Space complexity**: Memory needed to perform the search algorithm

General Approach

- Starting with an **initial state**
- Repeatedly **expand** a state by generating its successors
- Stop when a **goal state** is expanded
- Or **all reachable states** considered



https://webis.de/downloads/lecturenotes/search/unit-en-basic-search1.pdf

Terminology

- **Parent node**: Predecessor node, through which the current node is reached
- **Open list**: The collection of nodes that are neighbours of expanded nodes – candidates for the next expansion
- **Closed list**: The collection of expanded nodes – will not be considered again

Search Algorithms

- **Uninformed search**
  - Only the problem definition is available
  - No further information about the domain
  - Expand the search “blindly” and “brutally”
  - Examples: breadth-first, depth-first, uniform cost, Dijkstra's algorithm

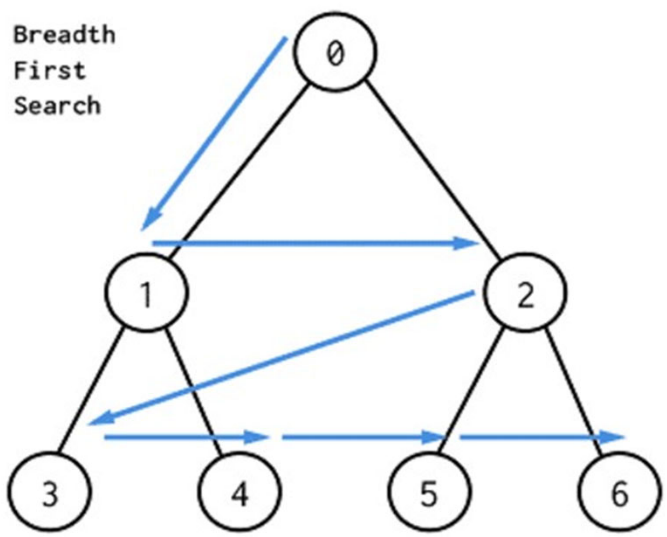
Search Algorithms

- **Informed search**
  - Further information through heuristic
  - Can say that one node is “more promising” than another
  - Directional
  - Examples: greedy best-first, A*, D*, D* Lite

Uninformed Search

Breadth-First Search (BFS)

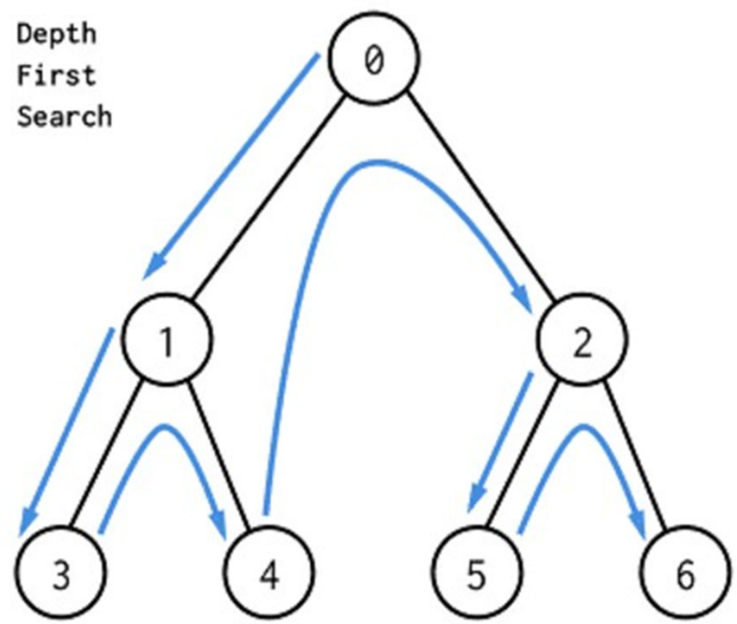
- Search along the breadth
- Complete
- Optimal if edge costs are equal and non-negative
- Time and space complexity $O(b^d)$



(*b*: branching factor; *d*: distance from start node; *m*: max. tree depth)

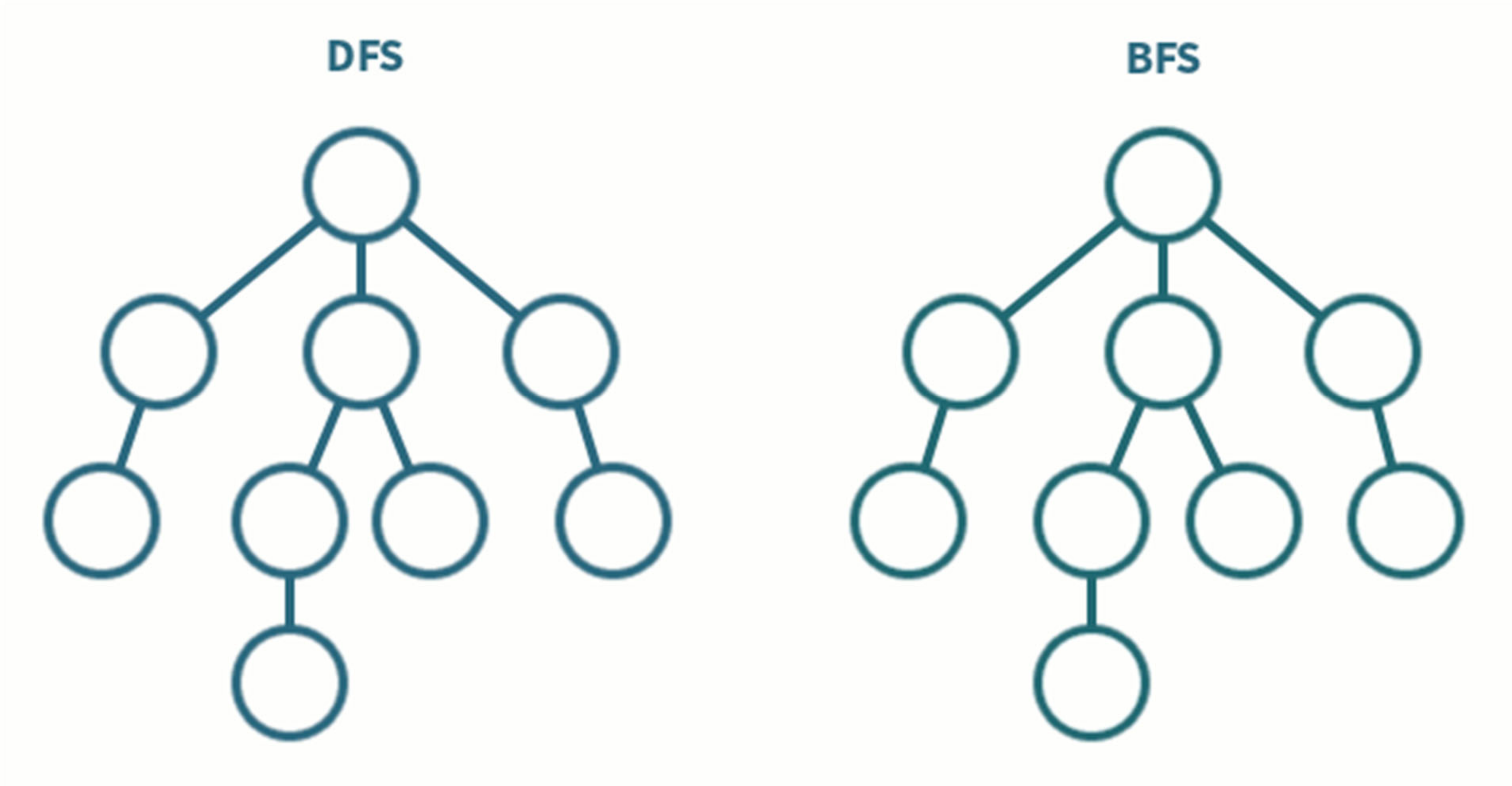
Depth-First Search (DFS)

- Search along the depth
- Not complete if depth is infinite
- Not optimal
- Time complexity: $O(b^m)$
- Space complexity: $O(bm)$



(b: branching factor; d: distance from start node; m: max. tree depth)

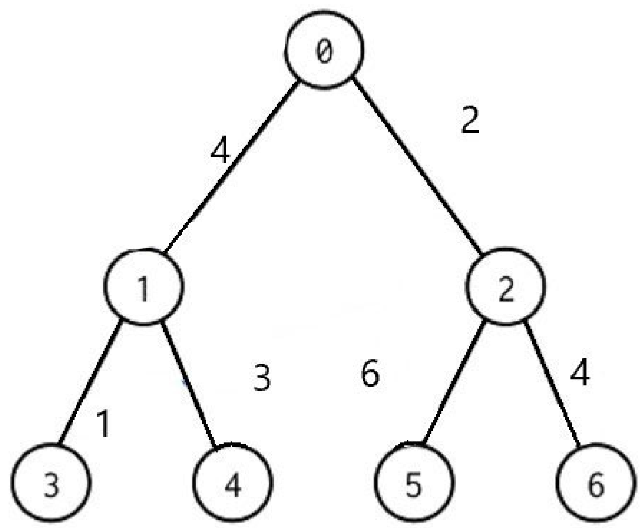
BFS vs. DFS



https://medium.com/@kenny.hom27/breadth-first-vs-depth-first-tree-traversal-in-javascript-48df2ebfc6d1

Uniform Cost Search (UCS)

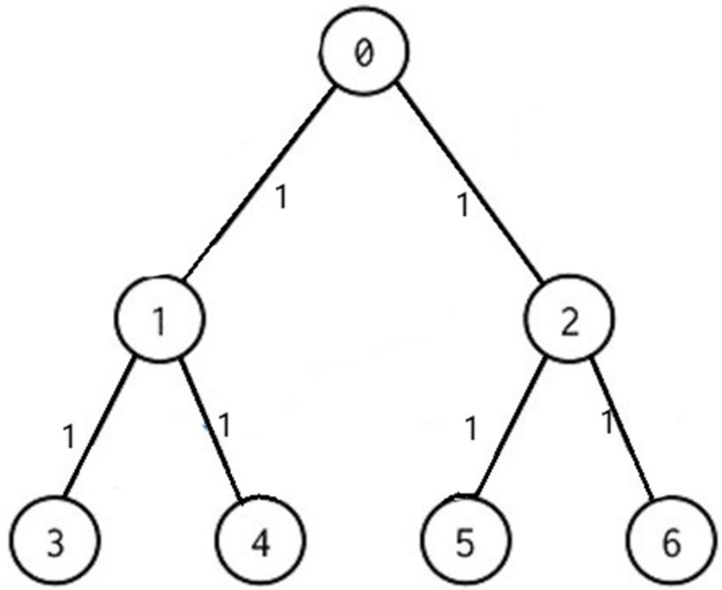
- Nodes are treated uniformly (Uniform) but expansion has different costs (Cost)
- At each step, expand the node with minimal accumulated cost $g(n)$
- Complete and optimal
- Time and space complexity: $O(b^{C/e})$

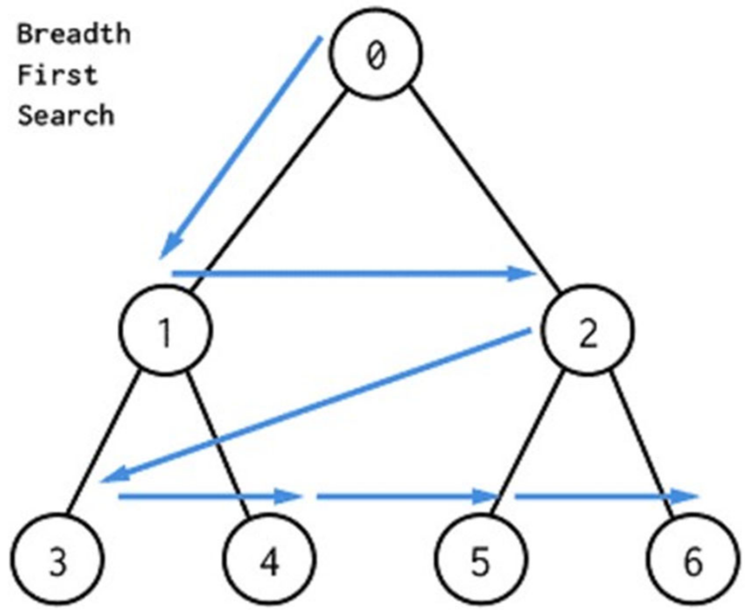


(*b*: branching factor; *C*: solution cost; *e*: min. edge cost)

UCS vs. BFS

- With uniform cost per expansion, UCS reduces to BFS

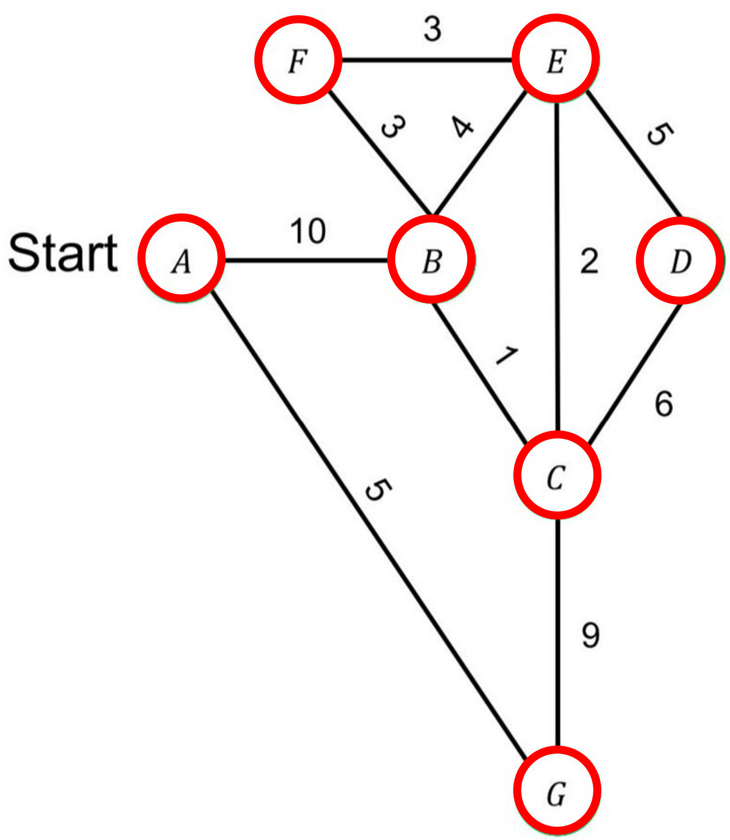




Dijkstra's Algorithm

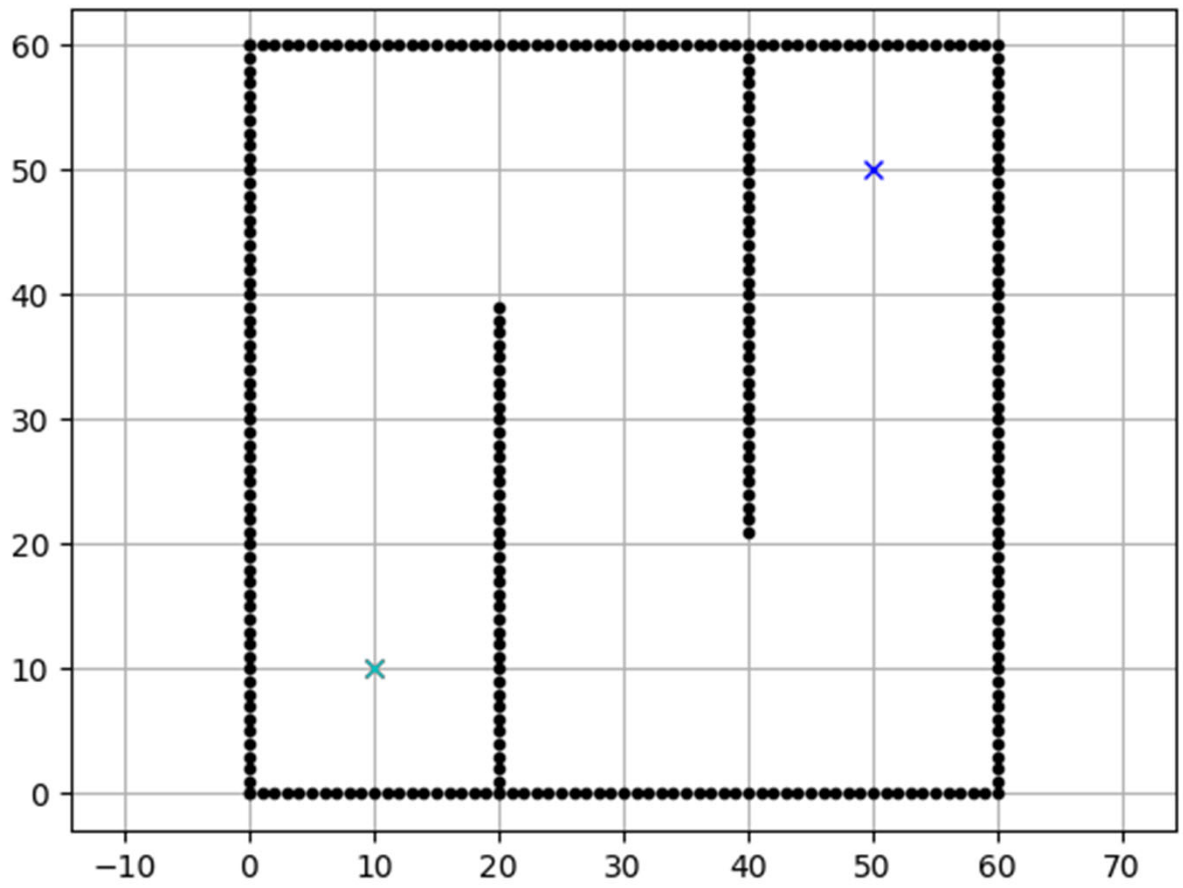
- Same expansion rules as UCS
  - Expand node with min. $g(n)$
- Find the shortest paths from the initial node to **all** other nodes = expand the full search tree
- If the search stops after the goal is reached, Dijkstra's algorithm becomes UCS
- Same time complexity, higher space complexity than UCS

Dijkstra's Algorithm



Closed list	Aktiv	A	B	C	D	E	F	G
		0	∞	∞	∞	∞	∞	∞
	A	0	10	∞	∞	∞	∞	5
A	G	0	10	14	∞	∞	∞	5
A, G	B	0	10	11	∞	14	13	5
A, G, B	C	0	10	11	17	13	13	5
A, G, B, C	E	0	10	11	17	13	13	5
A, G, B, C, E	F	0	10	11	17	13	13	5
A, G, B, C, E, F	D	0	10	11	17	13	13	5

Dijkstra's Algorithm



Informed Search

Heuristic

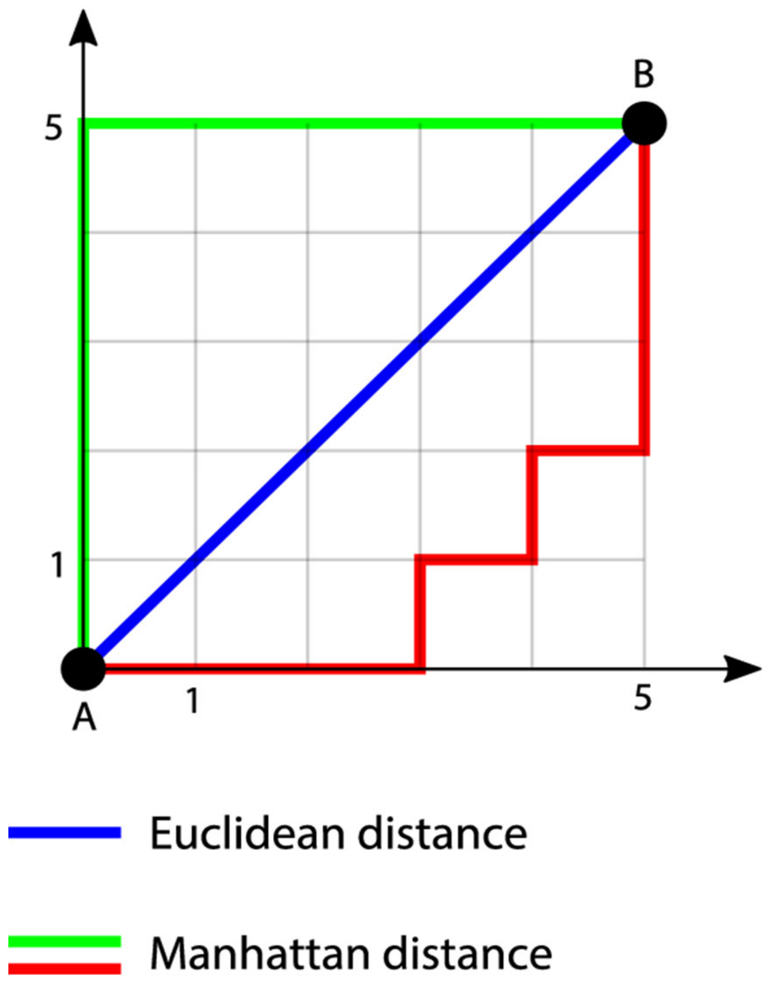
- **Heuristic** $h(n)$: Estimates the cost from the current node to the goal
- A good heuristic tells us how “promising” a node is → focuses and accelerates the search

- **Admissibility condition**: Heuristic *never overestimates* the true cost to the goal.

$$h(n) \leq h^*(n)$$

Heuristic

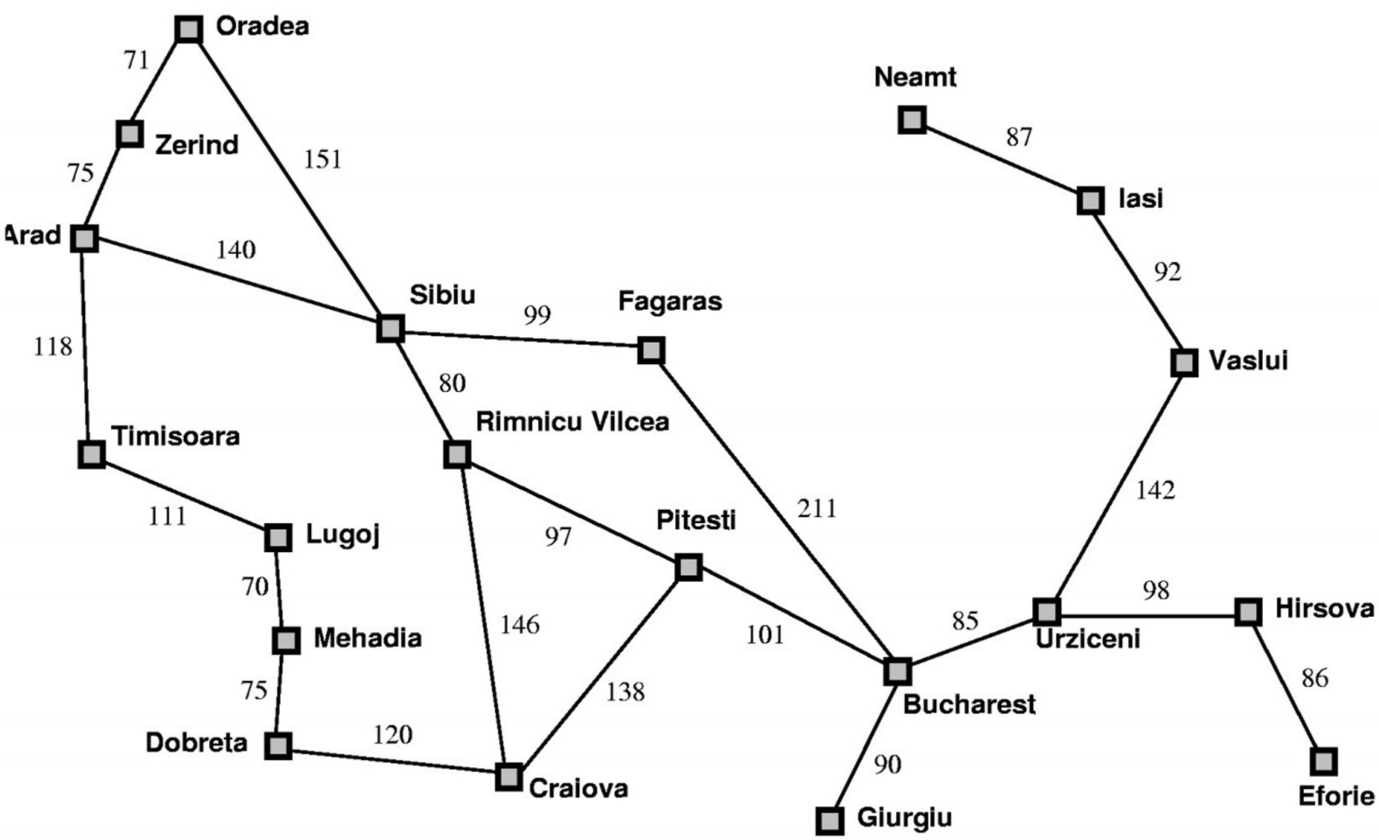
- In grid maps, a heuristic can be the distance to the goal
- Example metrics:
  - Are these heuristics always admissible?



Greedy Best-First Search

- Expand node with minimal heuristic value $h(n)$, as it appears to be closest to the goal
- Does not consider accumulated path cost
- Can be very fast in simple environments
- Vulnerable to local minima traps
- Neither complete nor optimal
- Highly depends on the quality of the heuristic

Greedy Best-First Search

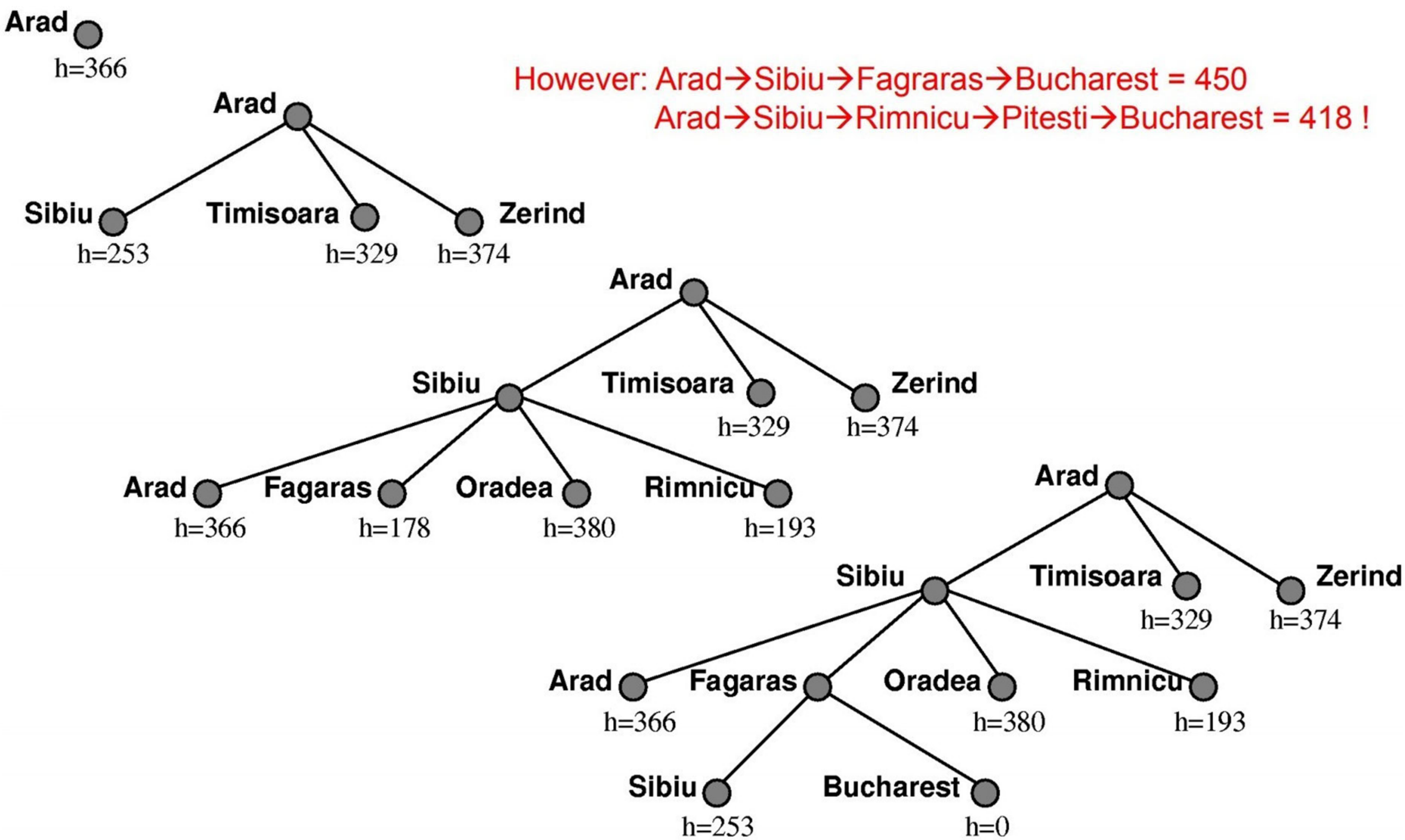


Straight–line distance to Bucharest

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

Greedy Best-First Search

However: Arad→Sibiu→Fagraras→Bucharest = 450
Arad→Sibiu→Rimnicu→Pitesti→Bucharest = 418 !



Recap: UCS

- Only considers the accumulated cost to a node

$$f(n) = g(n)$$

Total estimated cost of cheapest solution through n

Actual accumulated cost to reach n from start node

UCS

Recap: Greedy Best-First Search

- Only considers the heuristic value

$$f(n) = h(n)$$

Total estimated cost of cheapest solution through n

Heuristic: estimated cost to reach goal node from n

Greedy

A*

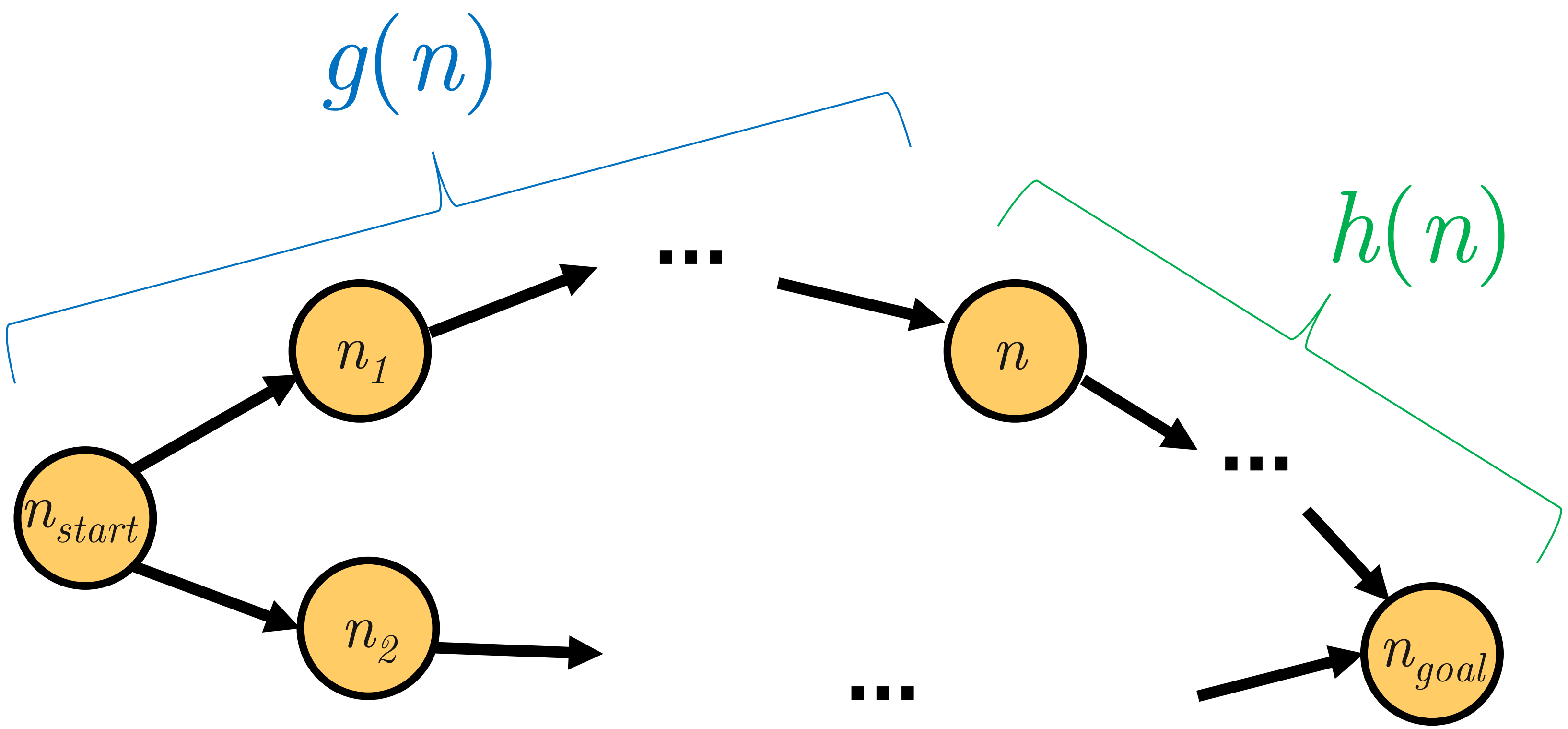
- Combines UCS and greedy best-first search
- Considers **both** the accumulated cost and the heuristic value

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

Total estimated cost of cheapest solution through n	Actual accumulated cost to reach n from start node	Heuristic: estimated cost to reach goal node from n

A*

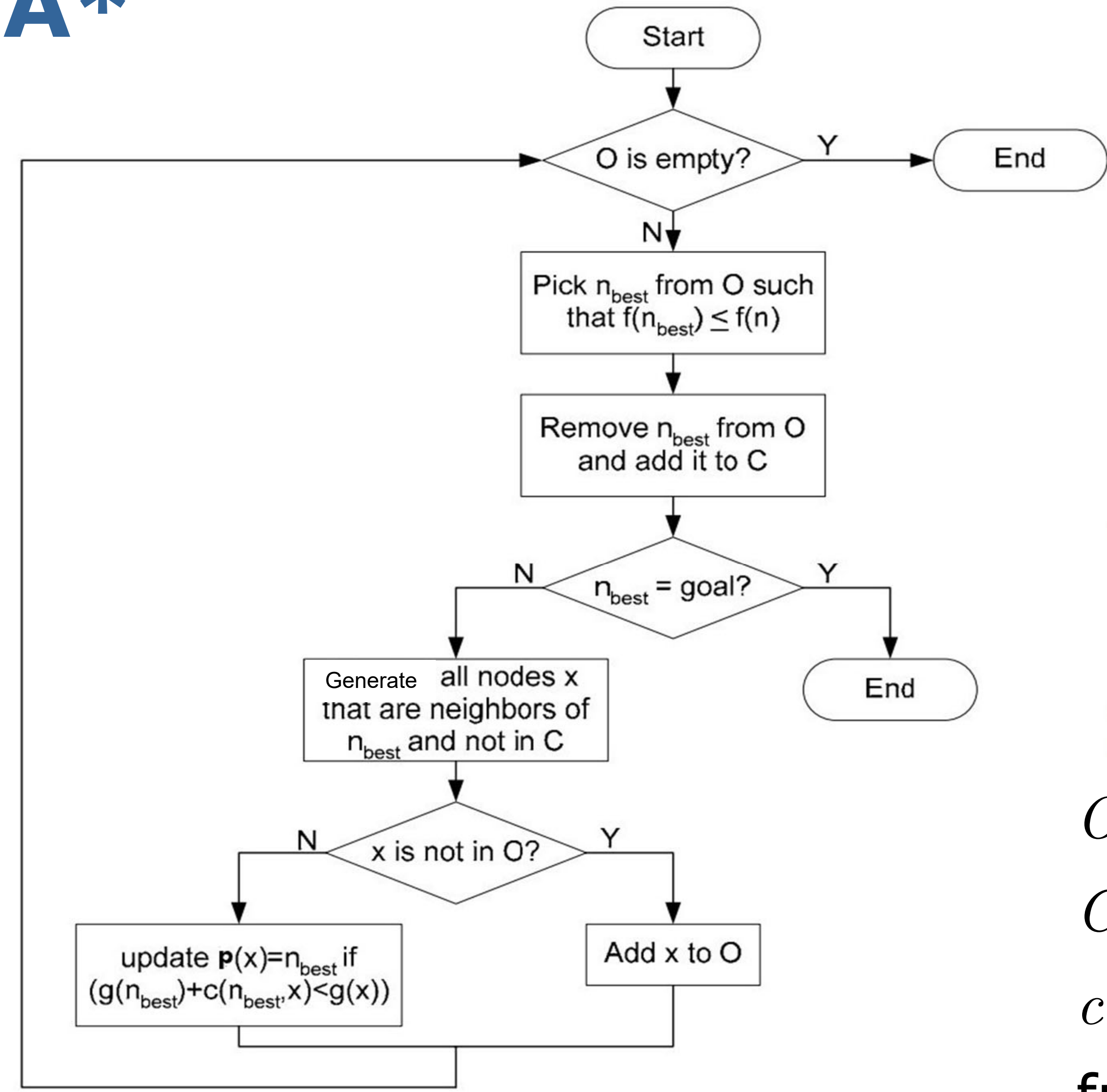
- Combines UCS and greedy best-first search
- Considers **both** the accumulated cost and the heuristic value



A* Heuristic

- A* needs an **admissible** heuristic
- Optimal and complete
- The lower the heuristic, the more nodes A* expands (A* with $h(n) = 0$ is UCS)
- If the heuristic is overestimated, the result is suboptimal but the search is faster
- If the heuristic is optimal, the search will follow the best path. Can we get an optimal heuristic?

A*

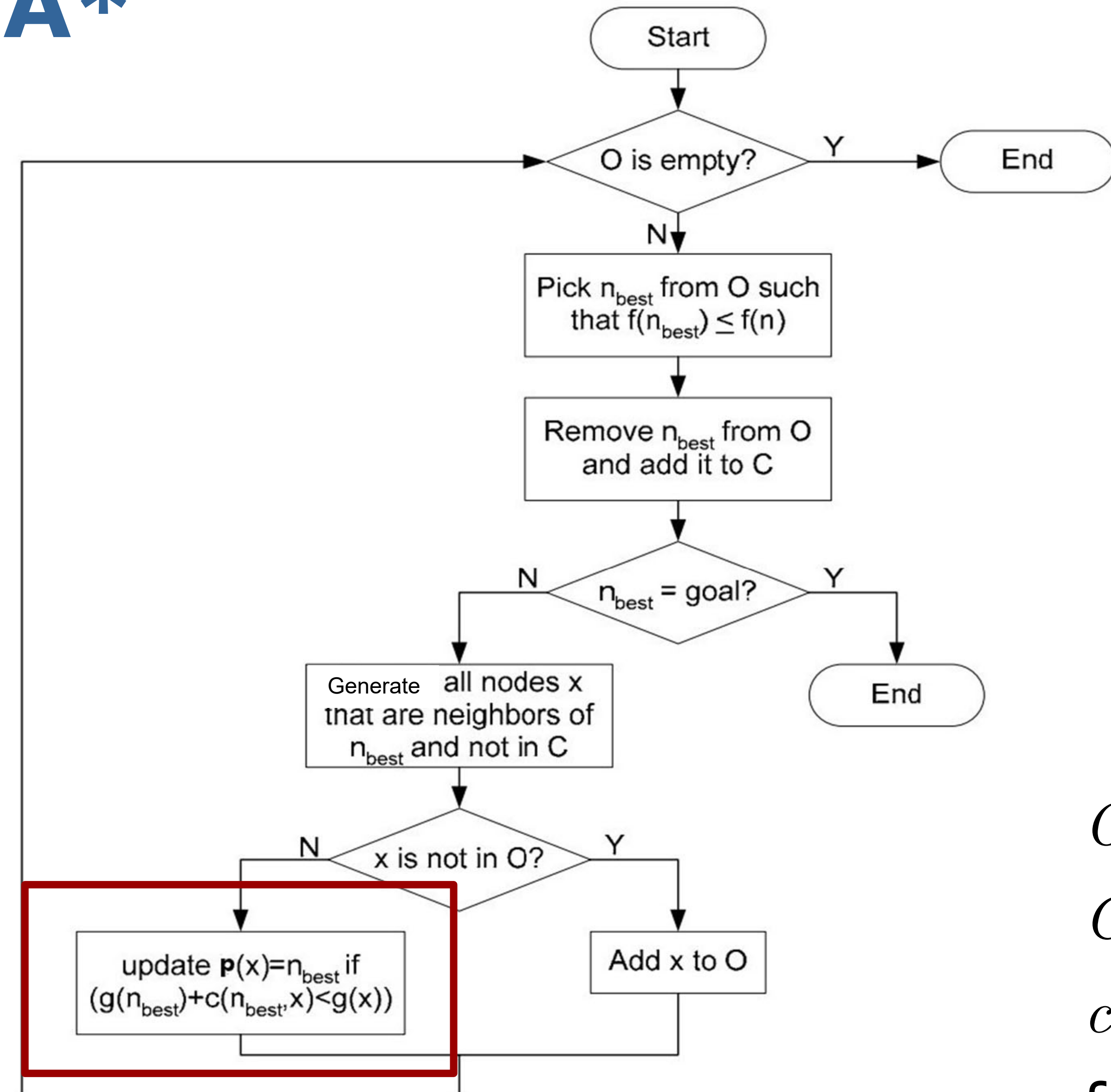


O: Open list

C: Closed list

$c(n_1, n_2)$: Edge cost from n_1 to n_2

A*

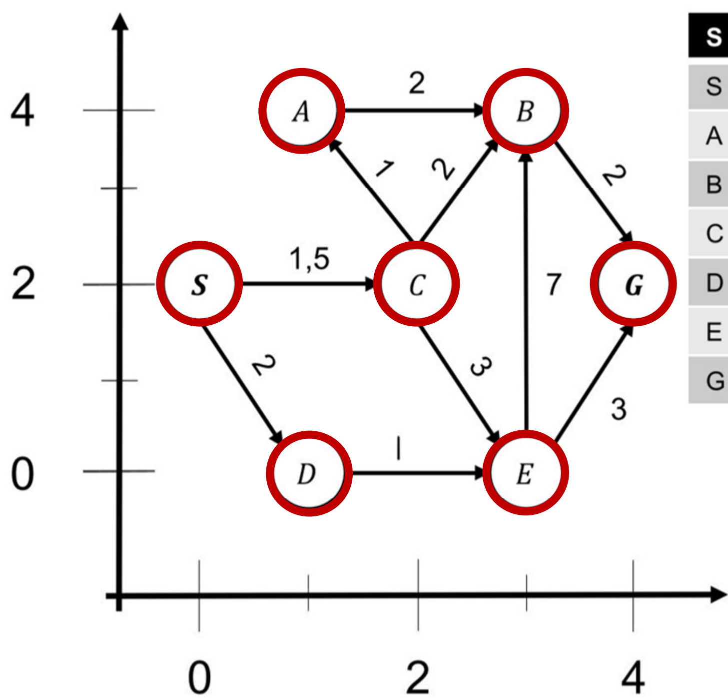


O: Open list

C: Closed list

$c(n_1,n_2)$: Edge cost from n_1 to n_2

A*



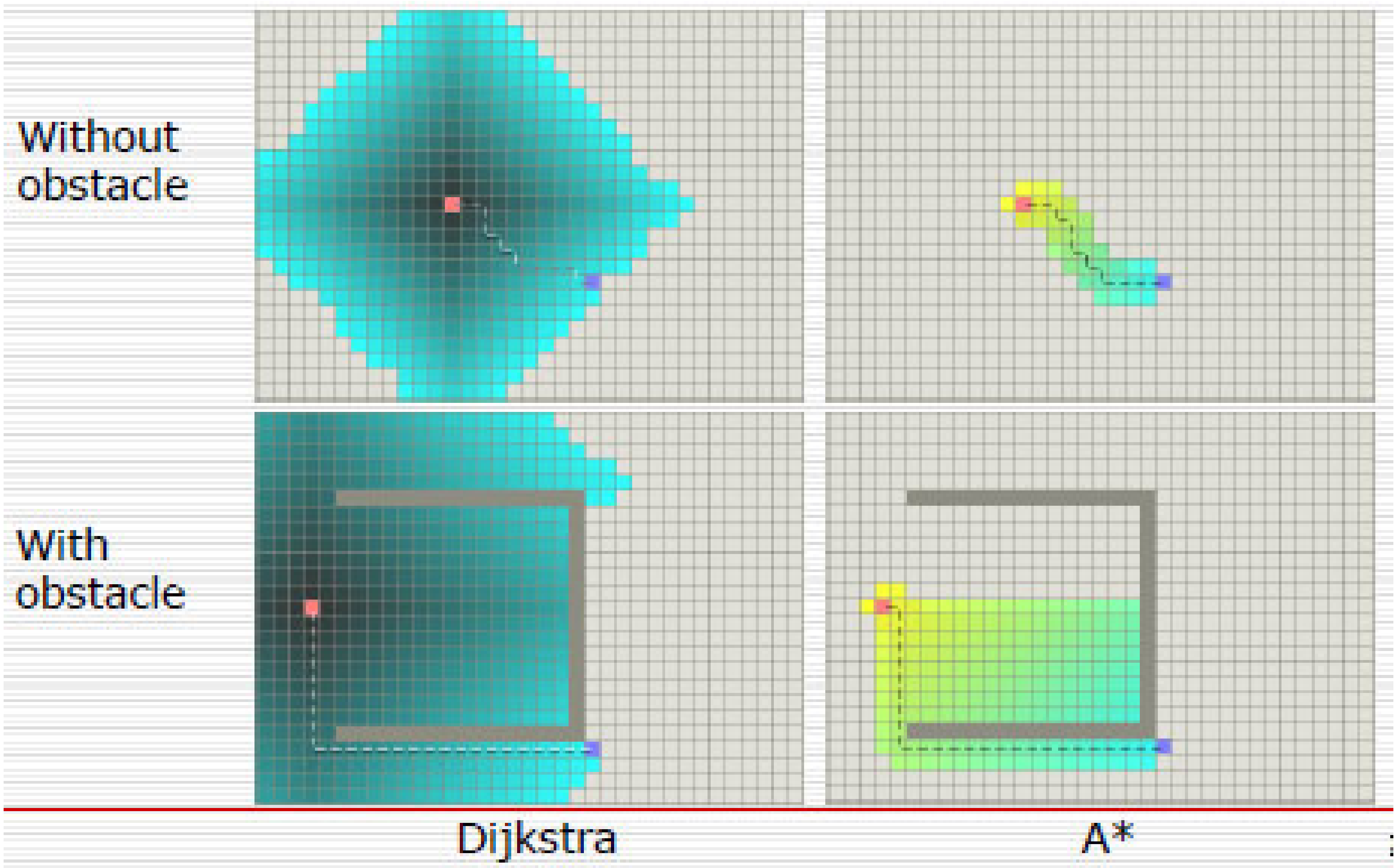
Heuristics

S	H
S	4
A	3,1
B	2,2
C	2
D	3,1
E	2,2
G	0

Total costs

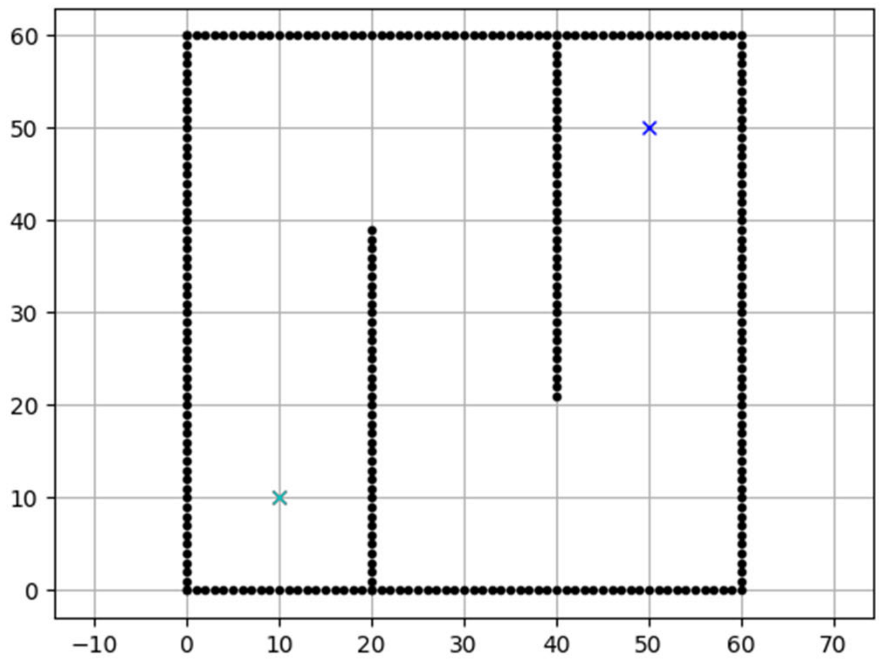
S				
0+4				
S-C	S-D			
3,5	5,1			
S-C-A	S-C-B	S-C-E	S-D	
(1,5+1)+3,1	(1,5+2)+2,2	(1,5+3)+2,2	5,1	
S-C-A	S-C-B	S-C-E	S-D-E	
5,6	5,7	6,7	(2+1)+2,2	
S-C-A	S-C-B	S-C-E	S-D-E-B	S-D-E-G
5,6	5,7	6,7	(3+7)+2,2	(3+3)+0
S-C-A-B	S-C-B	S-C-E	S-D-E-B	S-D-E-G
(2,5+2)+2,2	5,7	6,7	12,2	6
S-C-A-B	S-C-B-G	S-C-E	S-D-E-B	S-D-E-G
6,7	5,5	6,7	12,2	6

A* vs. Dijkstra's Algorithm

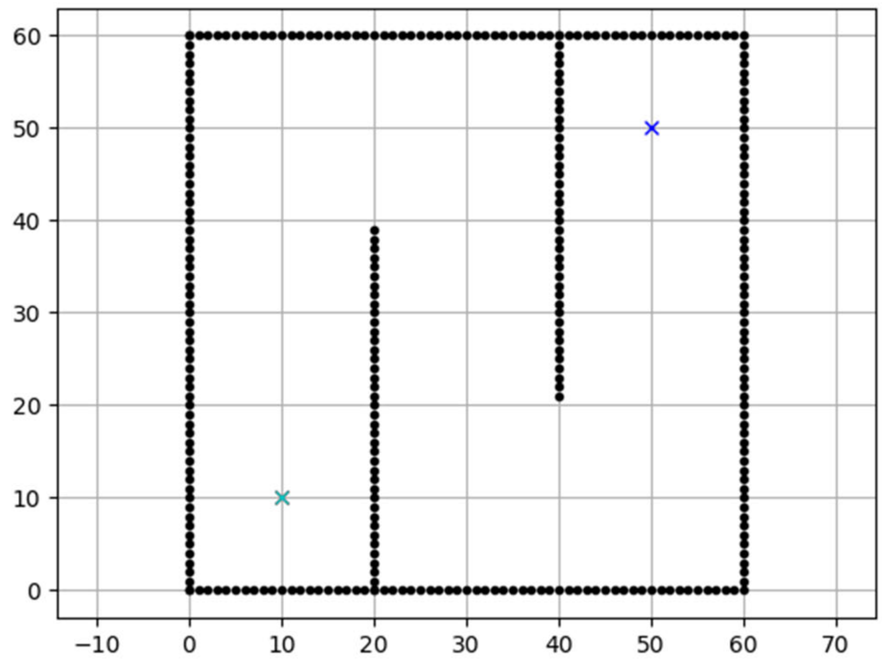


https://towardsdatascience.com/can-self-driving-car-think-6c9e8d939d60

A* vs. Dijkstra's Algorithm



A*

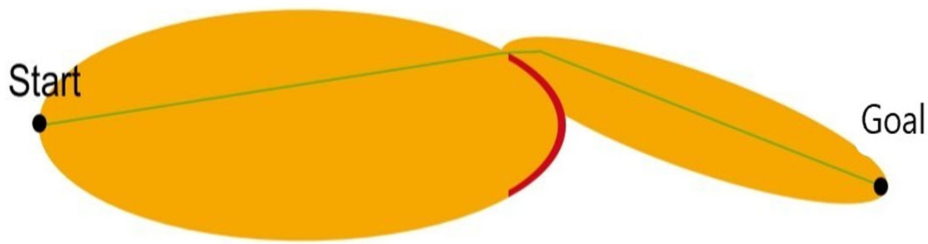


Dijkstra

Weighted A*

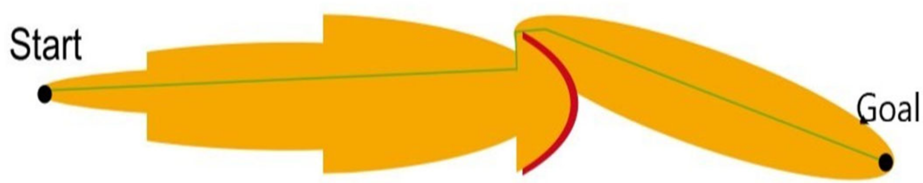
- Trade off optimality for speed
- Expand nodes based on a **weighting**:
  - Static: $f(n) = g(n) + \varepsilon h(n)$, $\varepsilon > 1$
  - Dynamic: $f(n) = g(n) + (1 + \varepsilon w(n))h(n)$, where $\varepsilon > 1$ and $w(n)$ is a variable that decreases as the search goes deeper
- Introduce bias towards nodes that appear to be closer towards the goal

Weighted A*



A*

Static weighting A*



Dynamic weighting A*

Optimal Heuristic

- Dijkstra's Algorithm can find the optimal (shortest) path from one node to all the others
- What if Dijkstra's Algorithm is performed starting from the goal?
- This gives us the optimal heuristic to the goal

→ Not feasible in practice

Search in Dynamic Environments

UNIVERSITÄT BONN

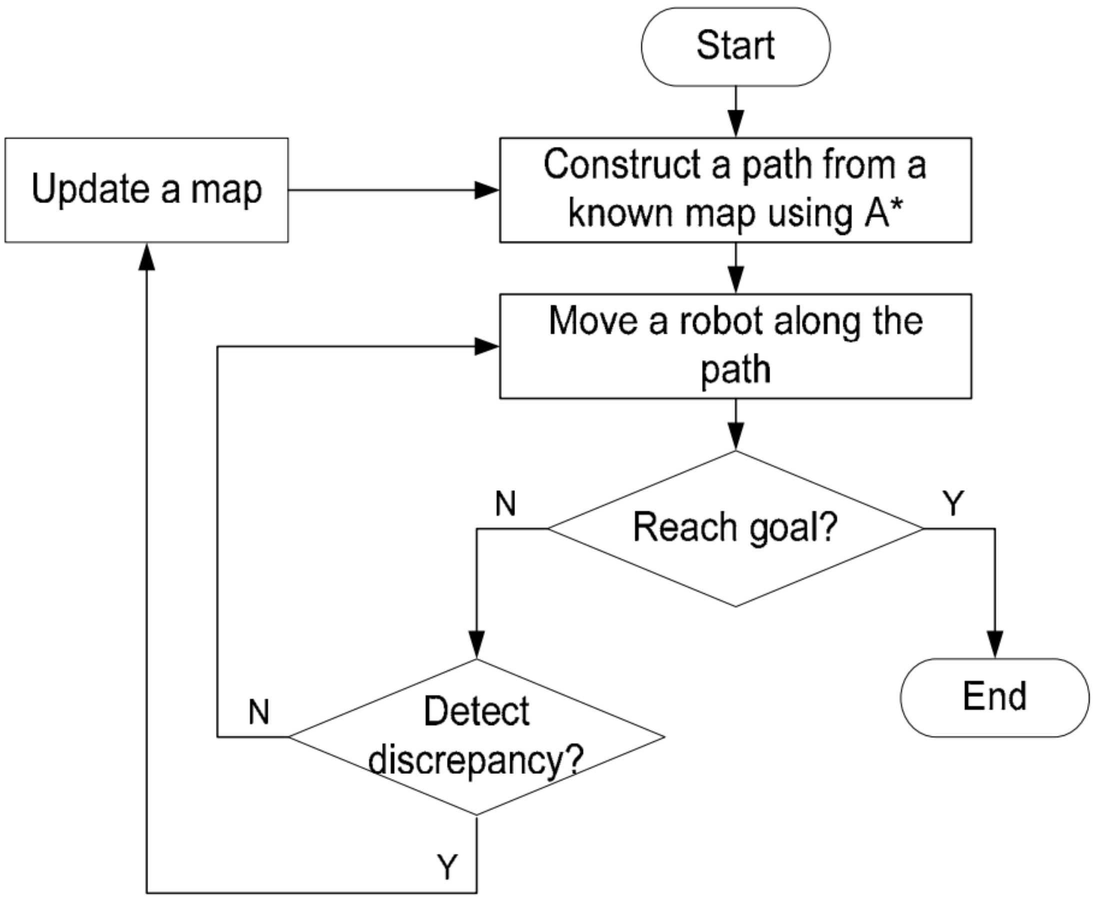
A* in Dynamic Environments

- What if the environment is dynamic?

- Heuristic generated by Dijkstra's Algorithm
- Still admissible with dynamic obstacles

https://www.programmersought.com/article/65933548761/

A* Replanner – Unknown Map



- Optimal
- Inefficient, impractical in large environments

https://www.cs.cmu.edu/~motionplanning/lecture/AppH-astar-dstar_howie.pdf

D* Lite

- **D*** stands for **dynamic A***
- D* **Lite** is a simplified version of D*
- Designed for dynamic or partially known environments
- Replans **online** by allowing edge costs to change during the search

D* Lite

- Plans from **goal** to **start** (reverse A*)
- Keeps track of **two** scores for each node:
  - G-score: accumulated cost $g(n)$ from the **goal**
  - RHS-score: one-step lookahead

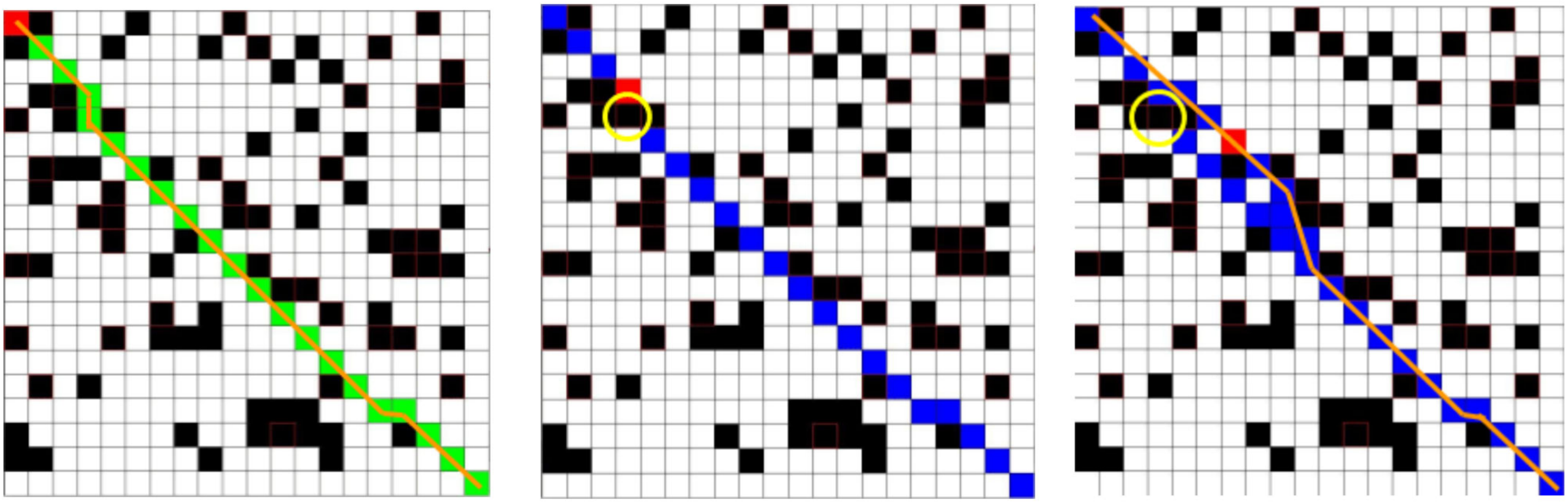
$$rhs(u) = \min_{s' \in Succ(u)}(c(u, s') + g(s'));$$

- Compare G-score and RHS-score to detect inconsistencies
- Total estimated cost:

$$\min(g(s), rhs(s)) + h(s_{start}, s)$$

https://www.cs.cmu.edu/~motionplanning/lecture/AppH-astar-dstar_howie.pdf

D* Lite



Summary

- Overview of search-based methods
- Uninformed search
  - BFS, DFS, UCS, Dijkstra's Algorithm
- Informed search methods
  - Heuristics, A*
- Search in dynamic environments
  - A* replanner, D* Lite

Further Reading

- Animation: Rohith | Pathfinding Visualizer (rohithaug.github.io)
- Introduction (stanford.edu)
- Informed Search Algorithms in AI – Javatpoint
- Dijkstra’s shortest path algorithm in a grid | by Roman Kositski | Mar, 2021 | Level Up Coding (gitconnected.com)
- A* Search and Dijkstra's Algorithm: A Comparative Analysis (cse442-17f.github.io)
- Microsoft PowerPoint - AppH-astar-dstar_howie.ppt (cmu.edu)
- Dstar Lite: An Optimal Algorithm for Robotics Pathfinding - NHSJS

Thank you for your attention