E190Q – Lecture 15
Autonomous Robot Navigation

Instructor: Chris Clark

Semester: Spring 2014

Figures courtesy of Probabilistic Robotics (Thrun et. Al.)

Control Structures
Planning Based Control

Prior Knowledge

Operator Commands

Localization

Cognition

Perception

Motion Control
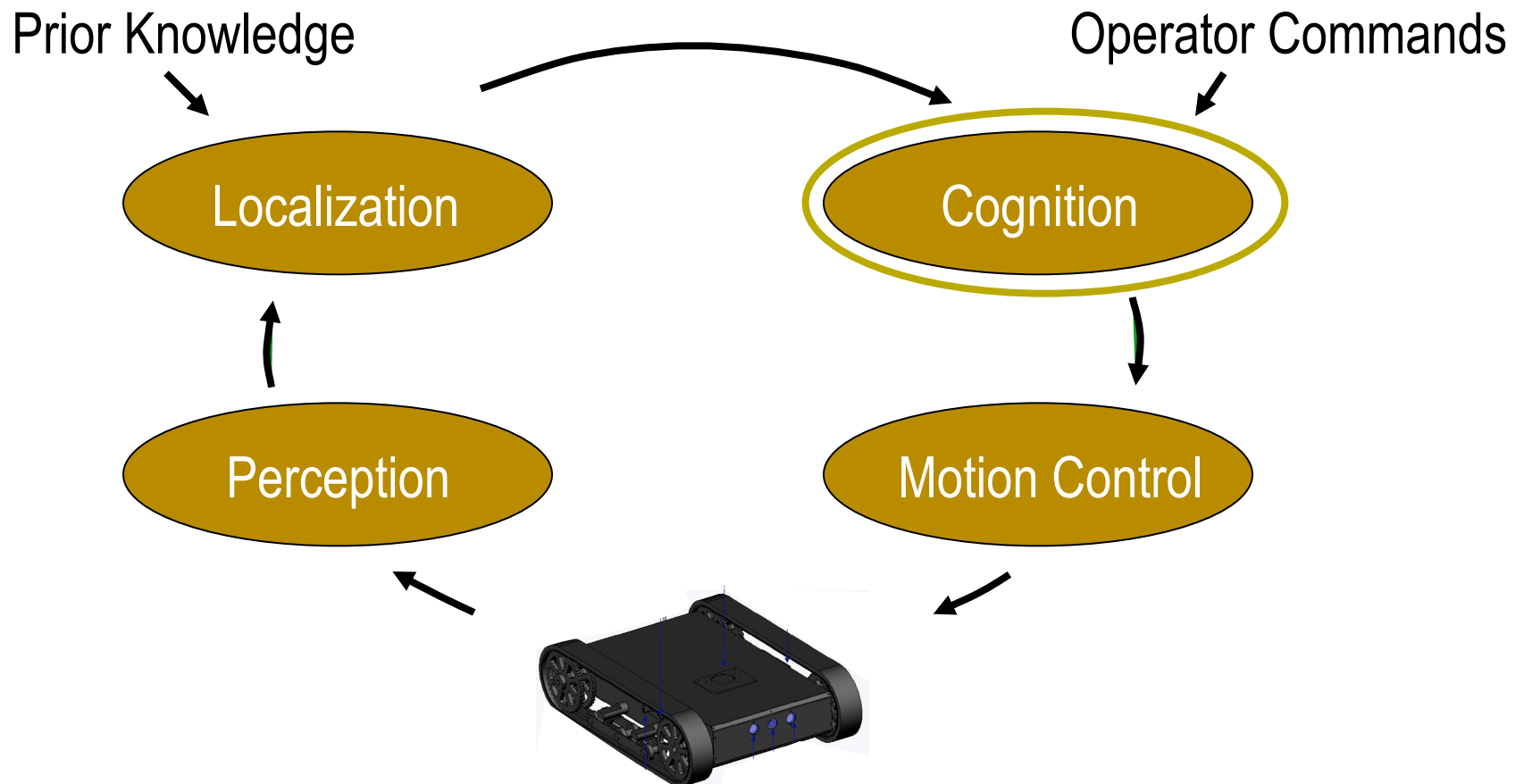
MP: Outline

1. Multi-Query PRMs
2. Graph Search
3. Artificial Potential Fields

MP: Outline

1. Multi-Query PRMs
2. Graph Search
3. Artificial Potential Fields

Multi-Query PRMs

- Multi-Query Strategy
  1. Learning Phase:
     - Generate the PRM with two steps:
       - Construction
       - Expansion
  2. Query Phase:
     - Connect start and goal configurations to PRM
     - Perform a graph search to find path

Multi-Query PRMs

local path

free space

milestone

m_b

m_g

[Kavraki, Svetska, Latombe,Overmars, 95]
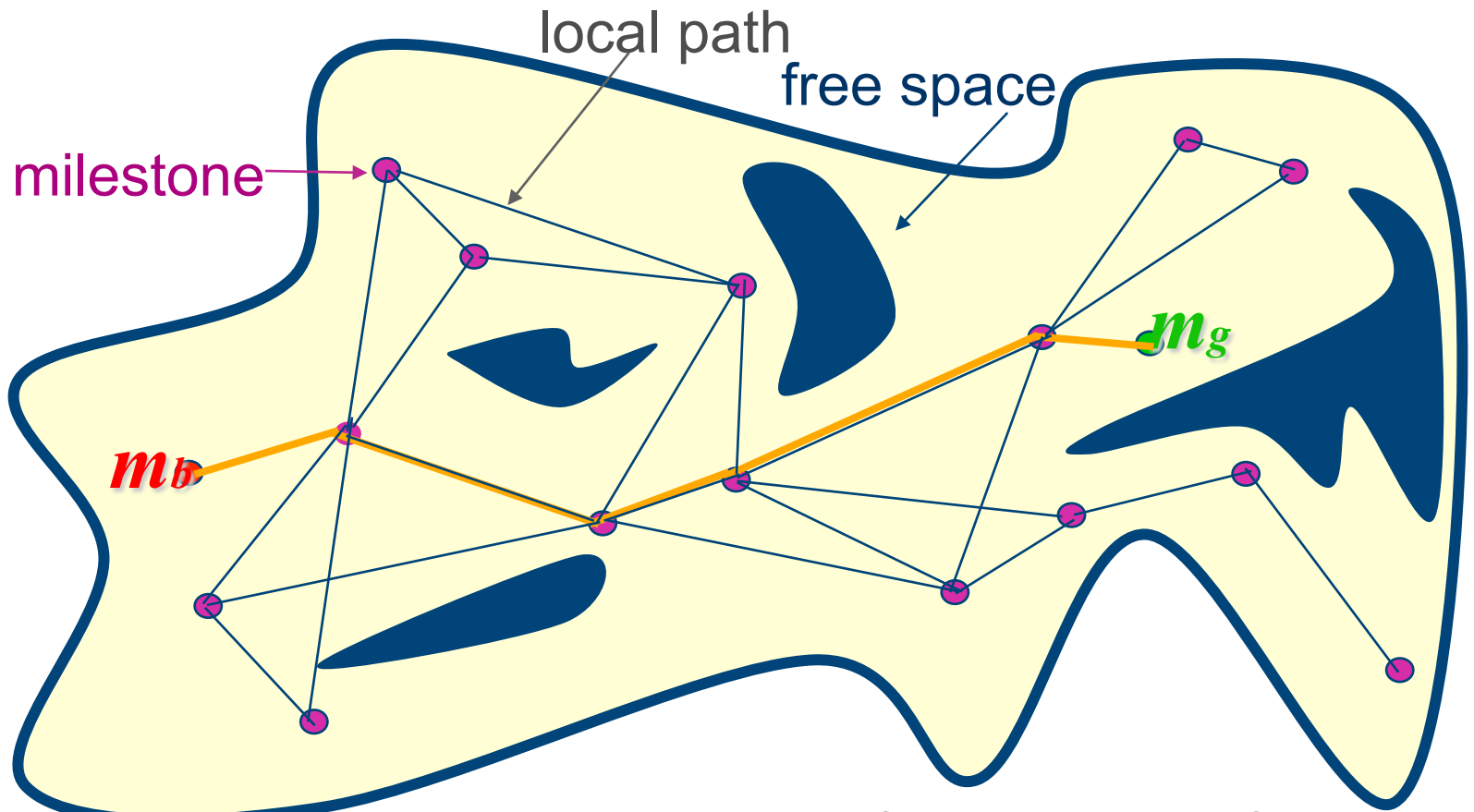
Multi-Query PRMs

- Nomenclature

  $R=(\ \mathbf{N},\ \mathbf{E}\ )$ RoadMap

  $\mathbf{N}$ Set of Nodes

  $\mathbf{E}$ Set of edges

  c Configuration

  e edge

Multi-Query PRMs Learning Phase

- Construction Step Algorithm

  Start with empty $R=(\mathbf{N}, \mathbf{E})$

  while (not done)

  {

      Generate a random free config c and add to $\mathbf{N}$

      Choose a subset $\mathbf{N_c}$ of candidate neighbors around c from $\mathbf{N}$

      Try to connect c to each node in $\mathbf{N_c}$ with local planner in the order of increasing distance from c

      Add the edge found to $\mathbf{E}$

  }

Multi-Query PRMs Learning Phase

- Construction Step
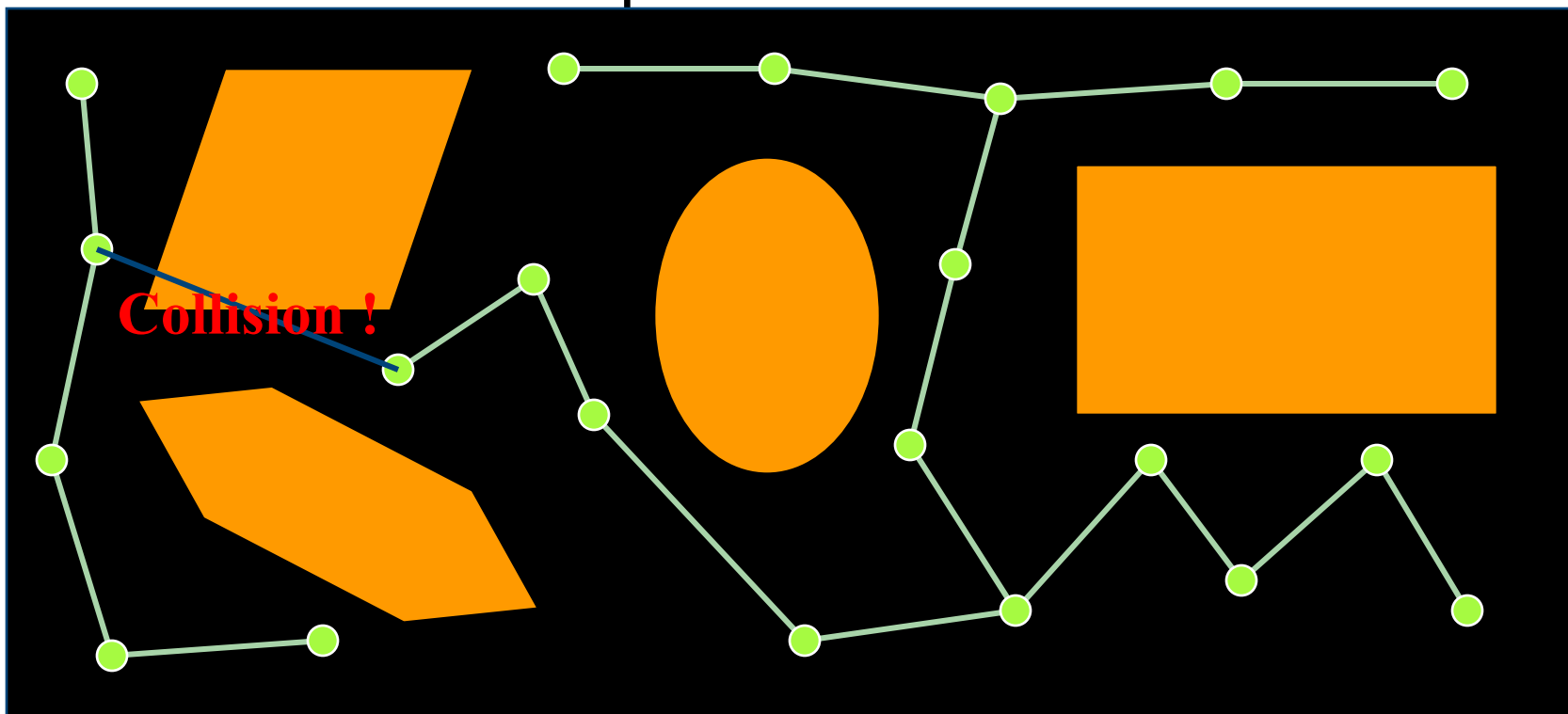
Collision !

Multi-Query PRMs
Learning Phase

- Local Planner
  - Used to connect two nodes.
  - Must contain collision-check.
  - For good performance, the LP must be:
    1. Deterministic - Eliminates the need for storing local plans.
    2. Fast - To ensure quick planning queries.

Multi-Query PRMs Learning Phase

- Expansion Step
  1. Find the nodes in 'difficult' regions using <u>heuristic weight function</u> $w(c)$
  2. Expand c using <u>random-bounce walks</u>
  3. Repeat as necessary

Multi-Query PRMs
Learning Phase

- Expansion Step
  - Several options to define weight function $w(c)$
    - Inversely proportional to the "number of nodes within some predefined distance from c"
    - Inversely proportional to the "distance from c to the nearest connected component not containing c"
    - Proportional to the "failure ratio of the local planner"

Multi-Query PRMs
Learning Phase

- Expansion Step

1. Loop
   1. Pick a random direction of motion in C-space
   2. Move in the direction until an obstacle is hit
   3. Check for connection with another node
   4. Repeat until the path can be connected to another node

Multi-Query PRMs Learning Phase
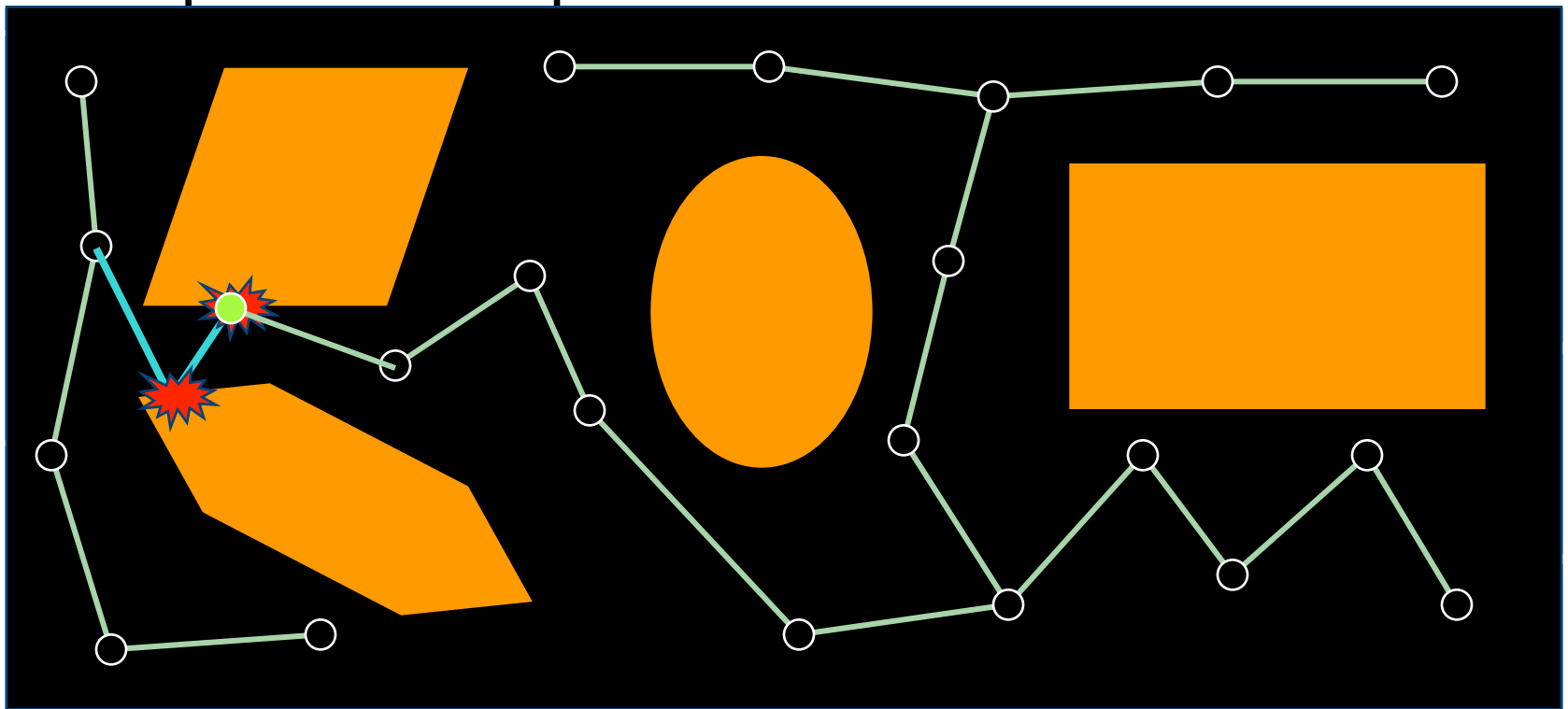
- Expansion Step

Multi-Query PRMs
Learning Phase

- Expansion Step

  1. Loop
     1. Pick a random direction of motion in C-space
     2. Move in the direction until an obstacle is hit
     3. Check for connection with another node
     4. Repeat until the path can be connected to another node
  2. Store the final config n and the edge (c, n) in R
  3. Store the computed path (non-deterministic)
  4. Record that n belongs to the same connected component as c

Multi-Query PRMs
Query Phase

- Query Phase Algorithm

  1. Given the start and goal configurations s and g, calculate feasible paths P_s and P_g to the nodes $\tilde{s}$ and $\tilde{g}$ on the roadmap (w/ LP)

  2. Calculate the path P from s to g using the roadmap and a **tree search** planner

Multi-Query PRMs
Query Phase

Courtesy of C. Allocco

Probabilistic Road Maps

- Two Tenets:
  1. Checking sampled configurations and connections between samples for collision can be done efficiently.

  2. A relatively small number of milestones and local paths are sufficient to capture the connectivity of the free space.

Probabilistic Road Maps: Discrete and Continous Planning

Courtesy of T. Bretl

MP: Outline

1. Multi-Query PRMs
2. Graph Search
3. Artificial Potential Fields

Graph Search

- Cell decomposition
  - Decompose the free space into simple cells and represent the connectivity of the free space by the adjacency graph of these cells
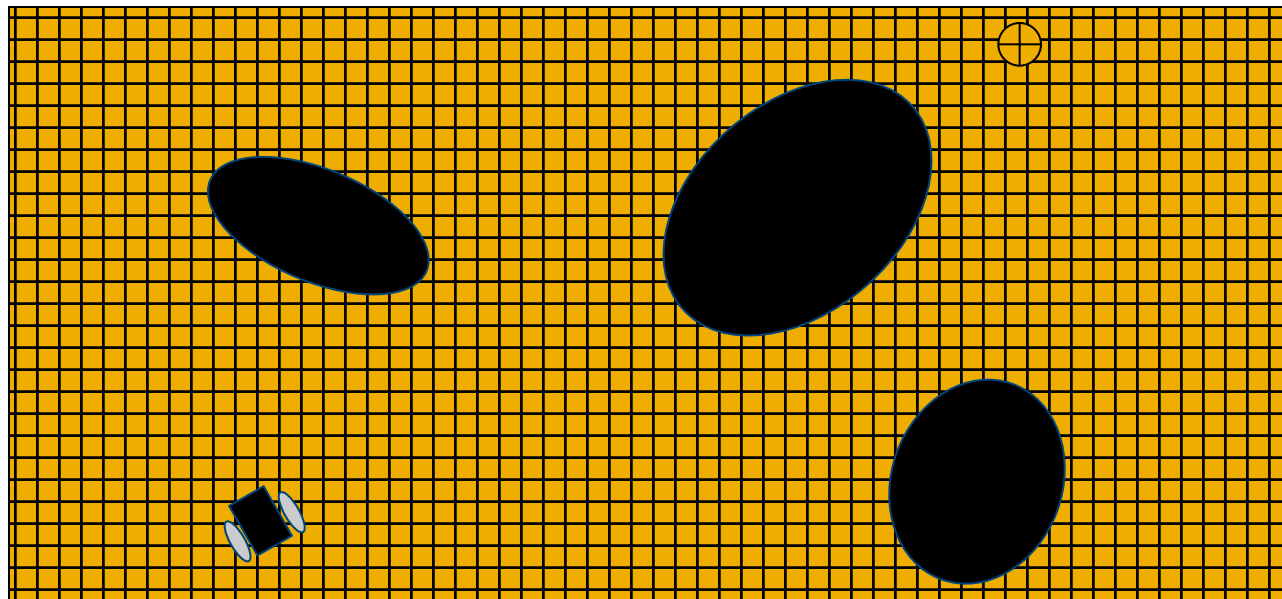
Graph Search

- Given a discretization of C, a search can be carried out using a Graph Search or gradient descent, etc.

  - Example: Find a path from D to G
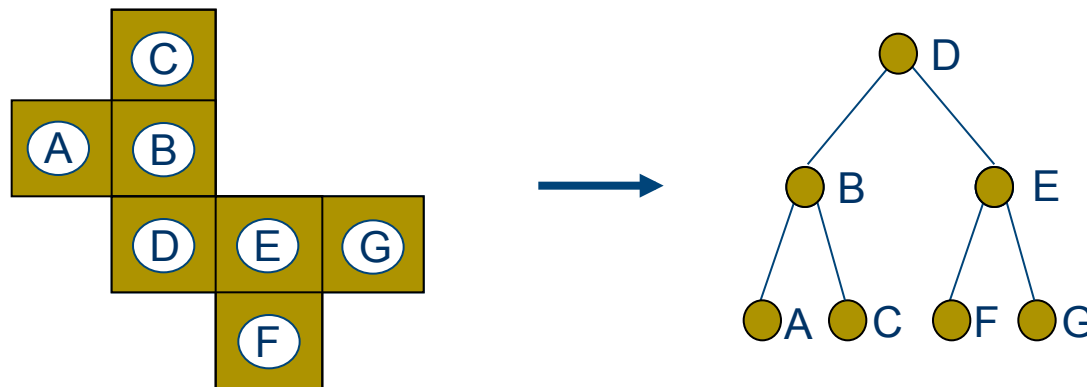
Tree Search

- Tree nomenclature:

Parent Node

Child Node
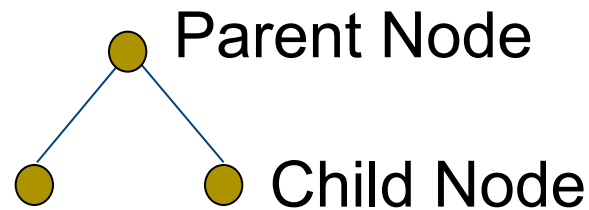
- Algorithms differ in the order in which they search the branches (edges) of the tree
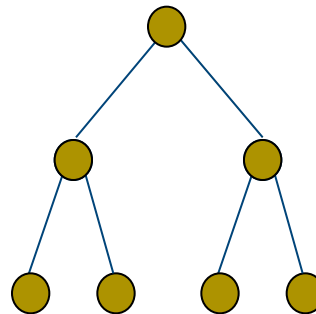
Data Structures

- The **Fringe** or Frontier is the collection of nodes waiting to be expanded.
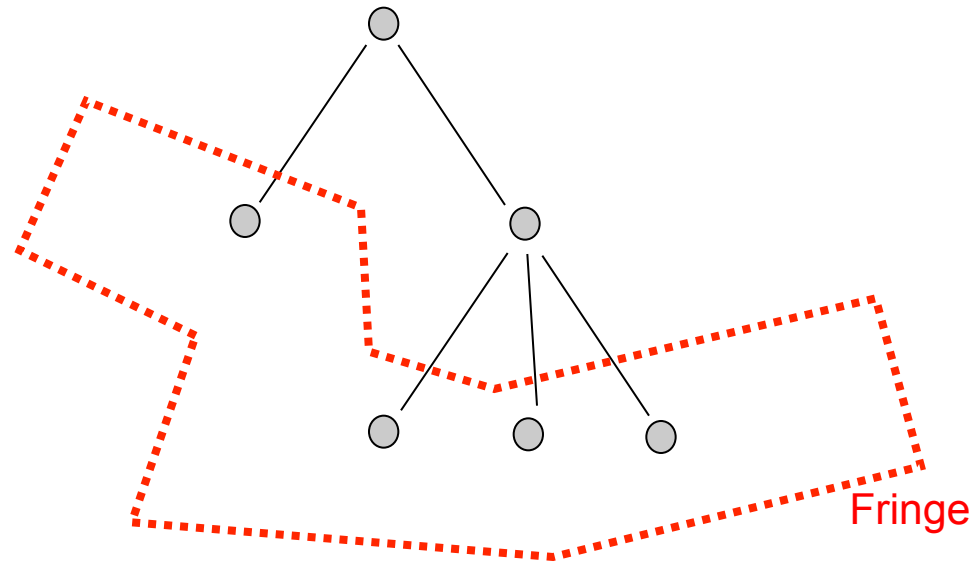
Fringe

Tree Search

- Search Algorithms
  1. Breadth First Search
  2. Depth First Search
  3. A*

Breadth-First

- All the nodes at depth d in the search tree are expanded before nodes at depth $d+1$

Breadth-First Snapshot 1
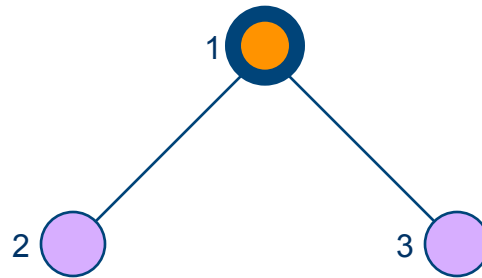
Initial
Visited
Fringe
Current
Visible
Goal

Fringe: [] + [2,3]

Breadth-First Snapshot 2
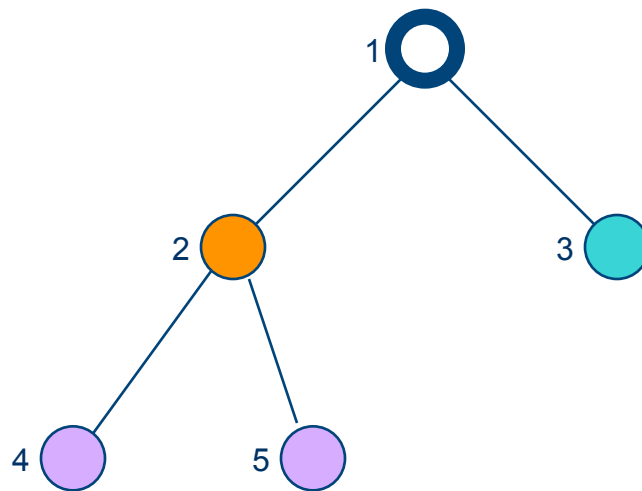
Initial

Visited

Fringe

Current

Visible

Goal

Fringe: [3] + [4,5]

Breadth-First Snapshot 3
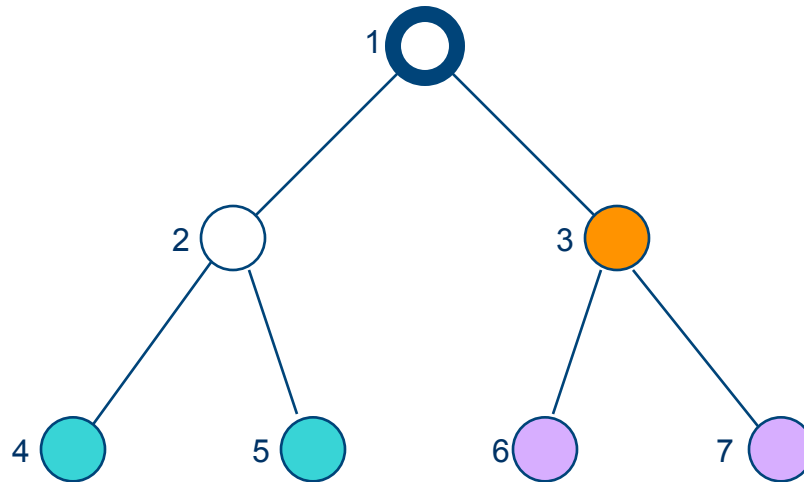
Initial
Visited
Fringe
Current
Visible
Goal

Fringe: [4,5] + [6,7]

Breadth-First Snapshot 4
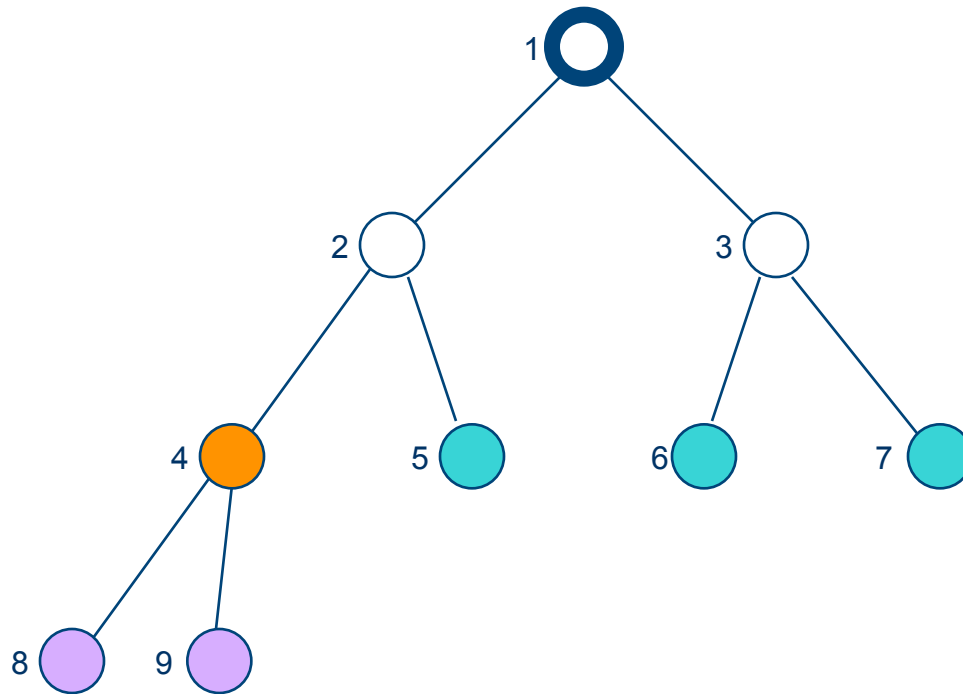

Initial
Visited
Fringe
Current
Visible
Goal

Fringe: [5,6,7] + [8,9]

Breadth-First Snapshot 5
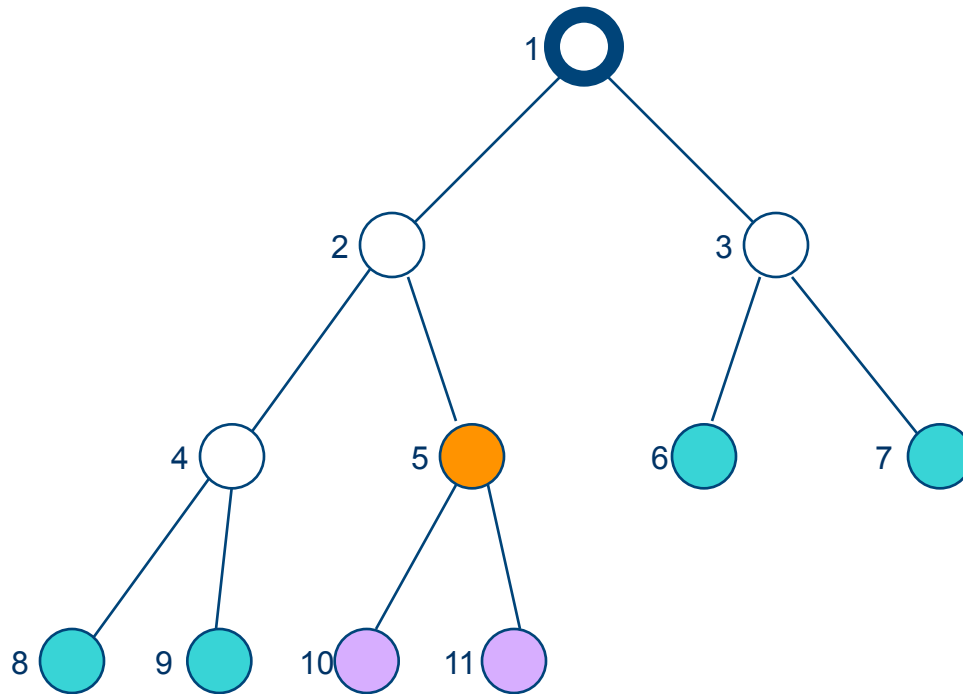
Initial ⬤

Visited ◯

Fringe ⬤

Current ⬤

Visible ⬤

Goal ⬤

Fringe: [6,7,8,9] + [10,11]

Breadth-First Snapshot 6
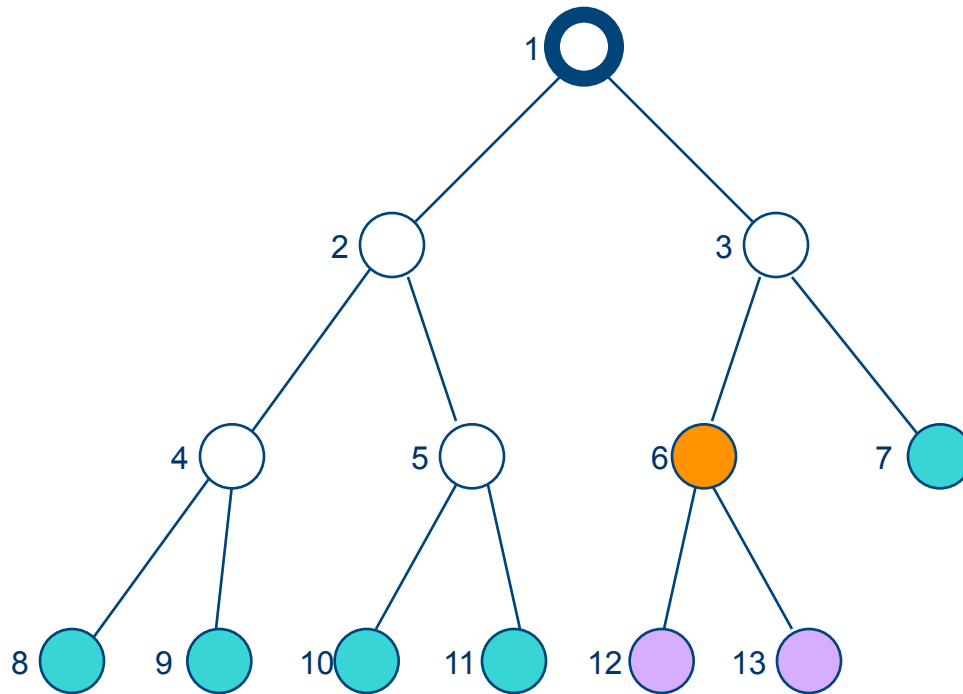
Initial
Visited
Fringe
Current
Visible
Goal

Fringe: [7,8,9,10,11] + [12,13]

Breadth-First Snapshot 7
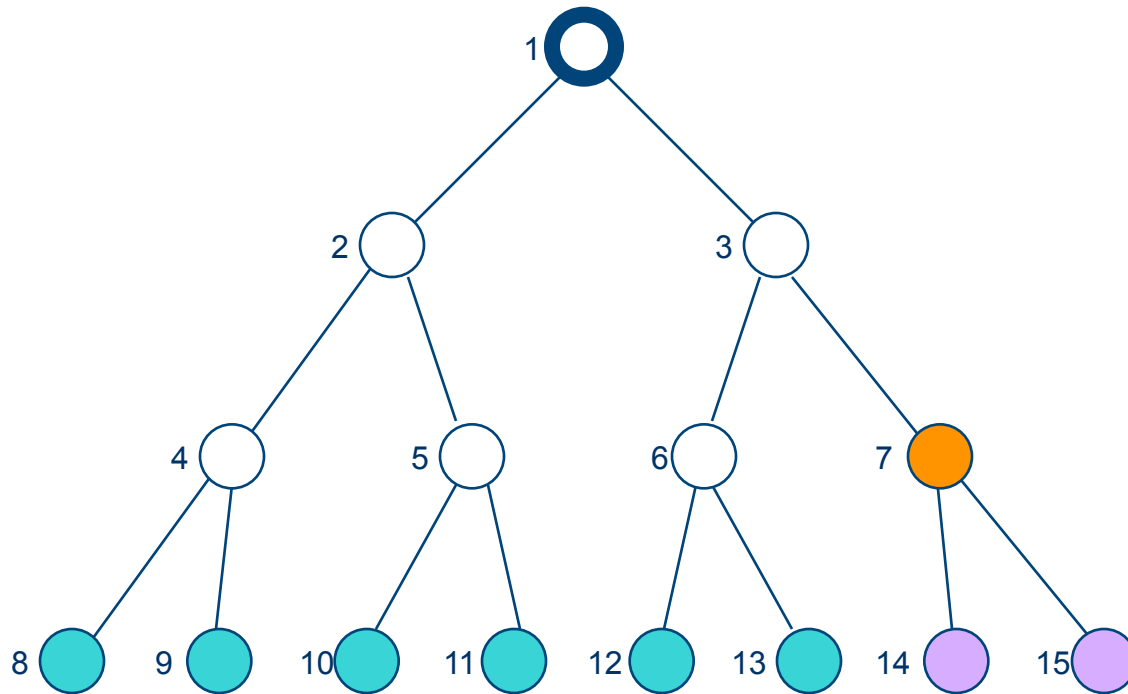
Initial ⬤

Visited ◯

Fringe 🔵

Current 🟠

Visible 🟣

Goal 🟢

Fringe: [8,9.10,11,12,13] + [14,15]

Breadth-First Snapshot 8

Initial
Visited
Fringe
Current
Visible
Goal

1
2   3
4   5   6   7
8   9   10   11   12   13   14   15
16   17
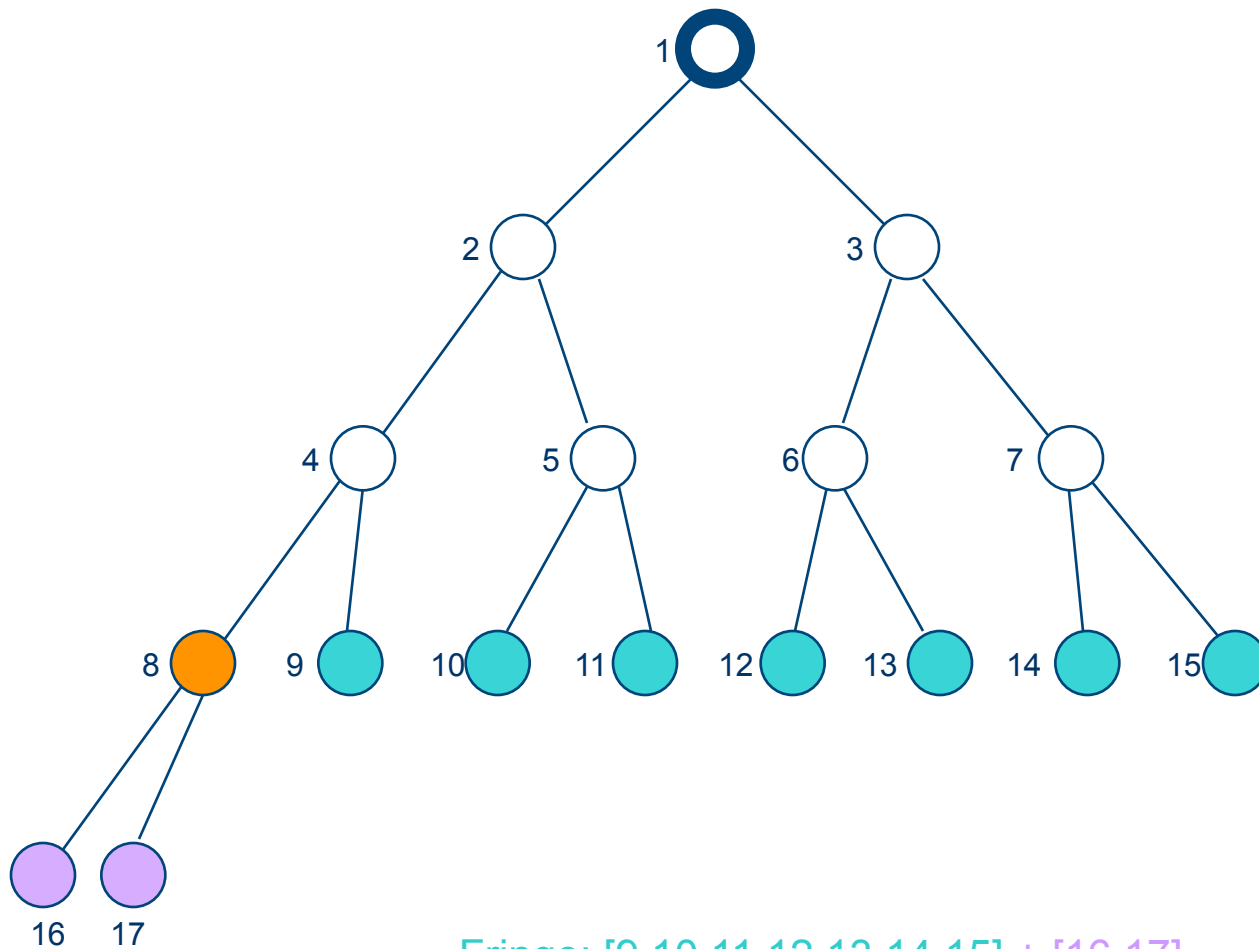
Fringe: [9,10,11,12,13,14,15] + [16,17]

Breadth-First Snapshot 9
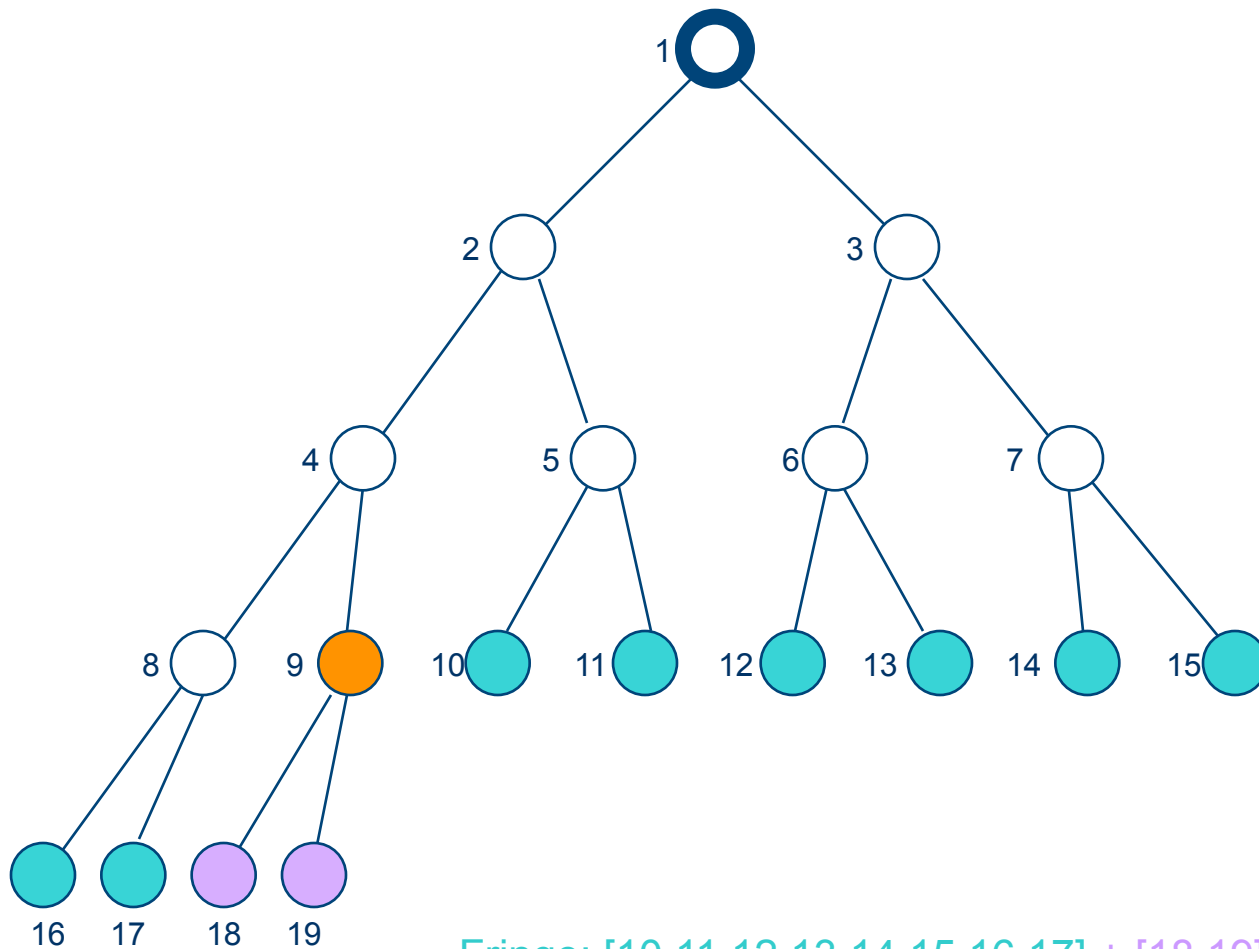
Fringe: [10,11,12,13,14,15,16,17] + [18,19]

Breadth-First Snapshot 10

Initial
Visited
Fringe
Current
Visible
Goal
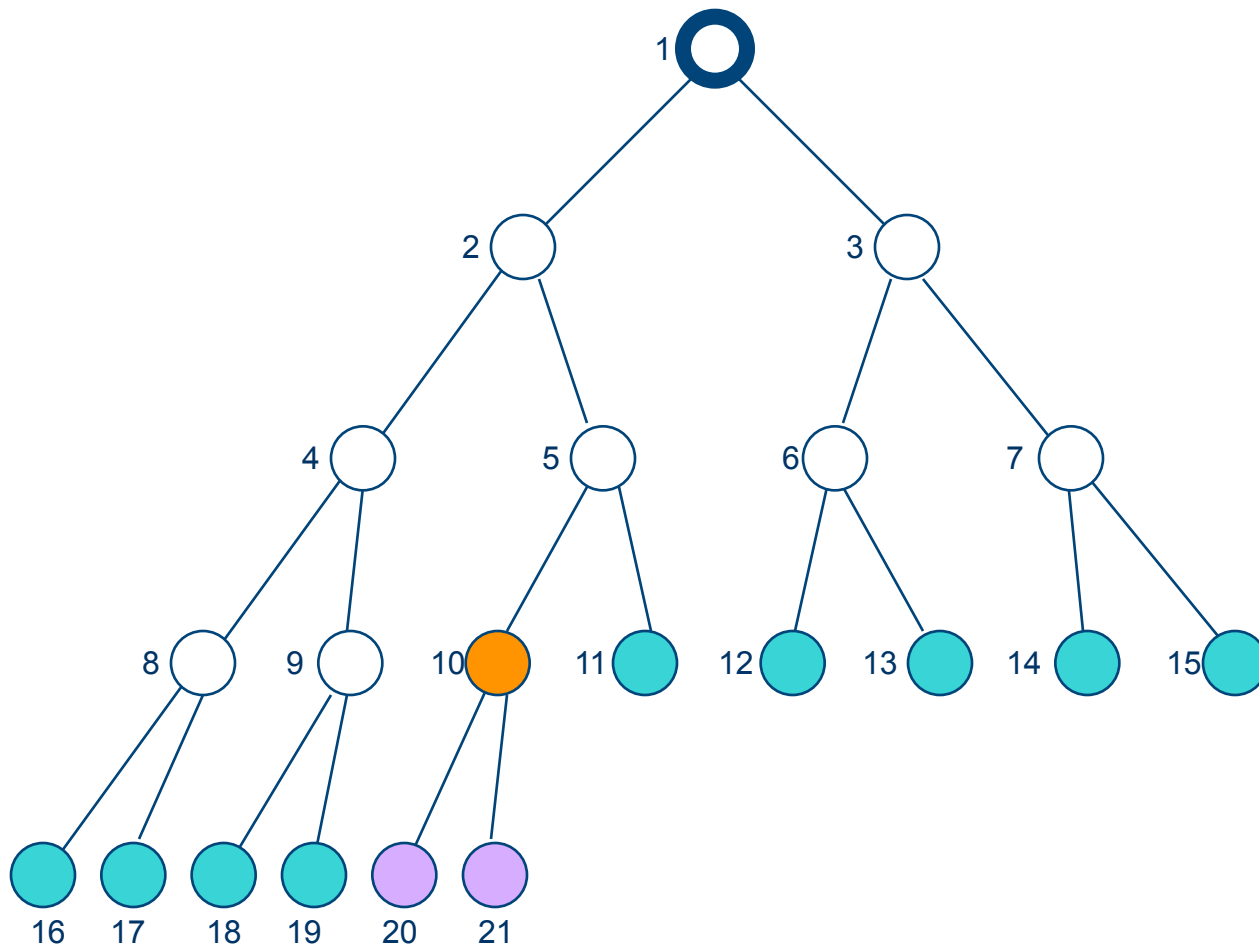
Fringe: [11,12,13,14,15,16,17,18,19] + [20,21]

Breadth-First Snapshot 11

Initial
Visited
Fringe
Current
Visible
Goal
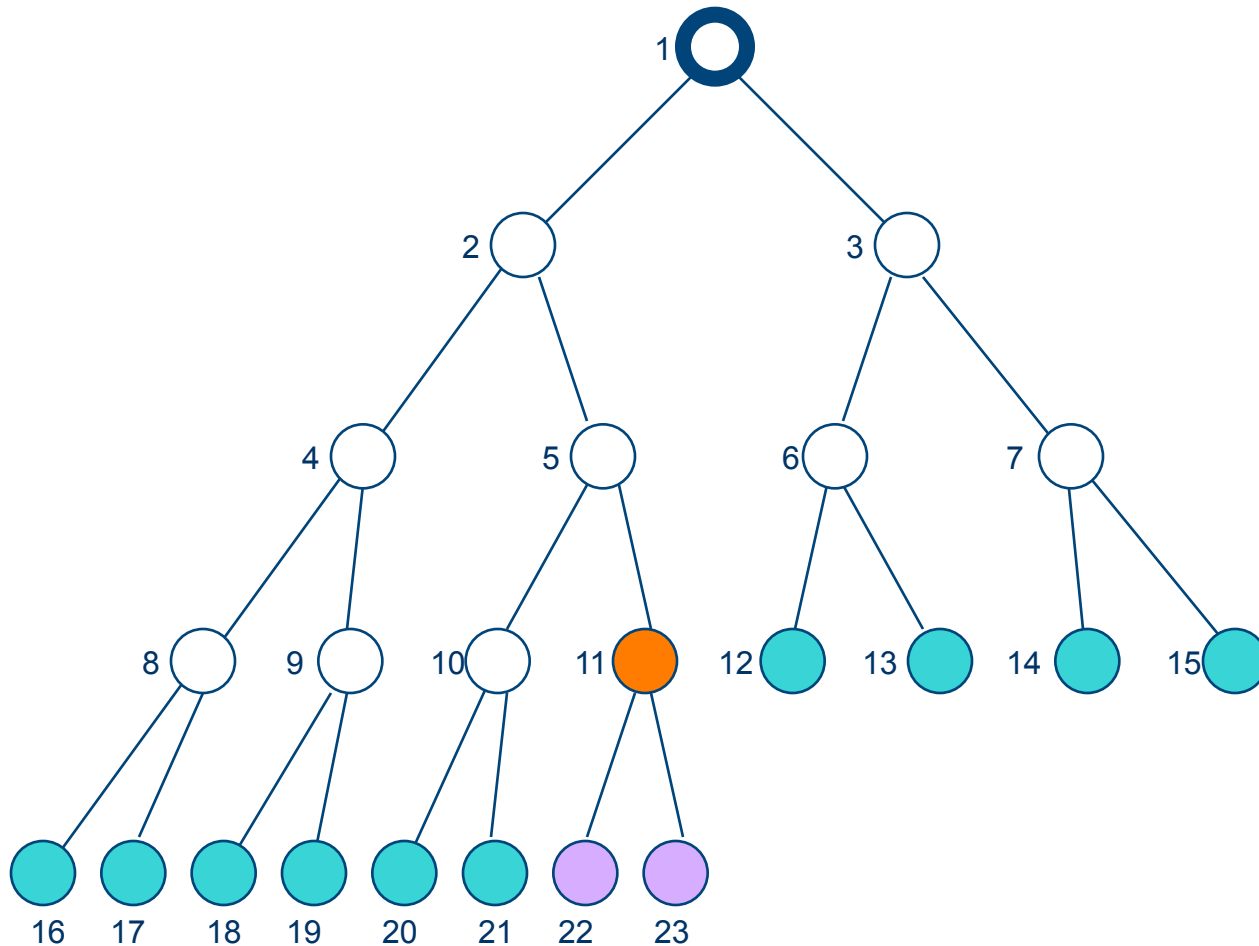
Fringe: [12, 13, 14, 15, 16, 17, 18, 19, 20, 21] + [22,23]

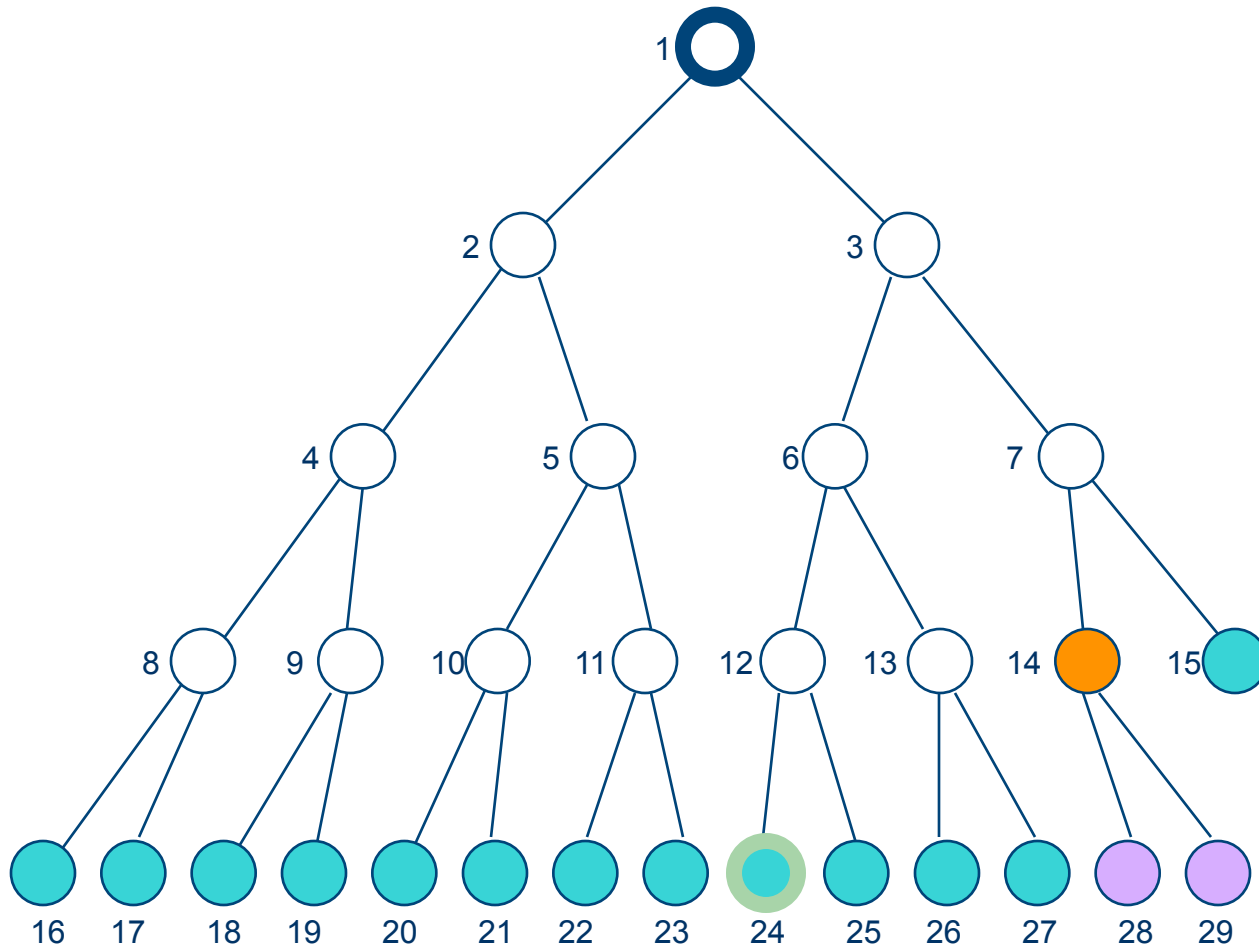# Breadth-First Snapshot 12

Initial
Visited
Fringe
Current
Visible
Goal

1

2   3

4   5   6   7

8   9   10   11   12   13   14   15

16   17   18   19   20   21   22   23   24   25

Note:
The goal node is "visible" here, but we can not perform the goal test yet.

Fringe: [13,14,15,16,17,18,19,20,21] + [22,23]

Breadth-First Snapshot 13
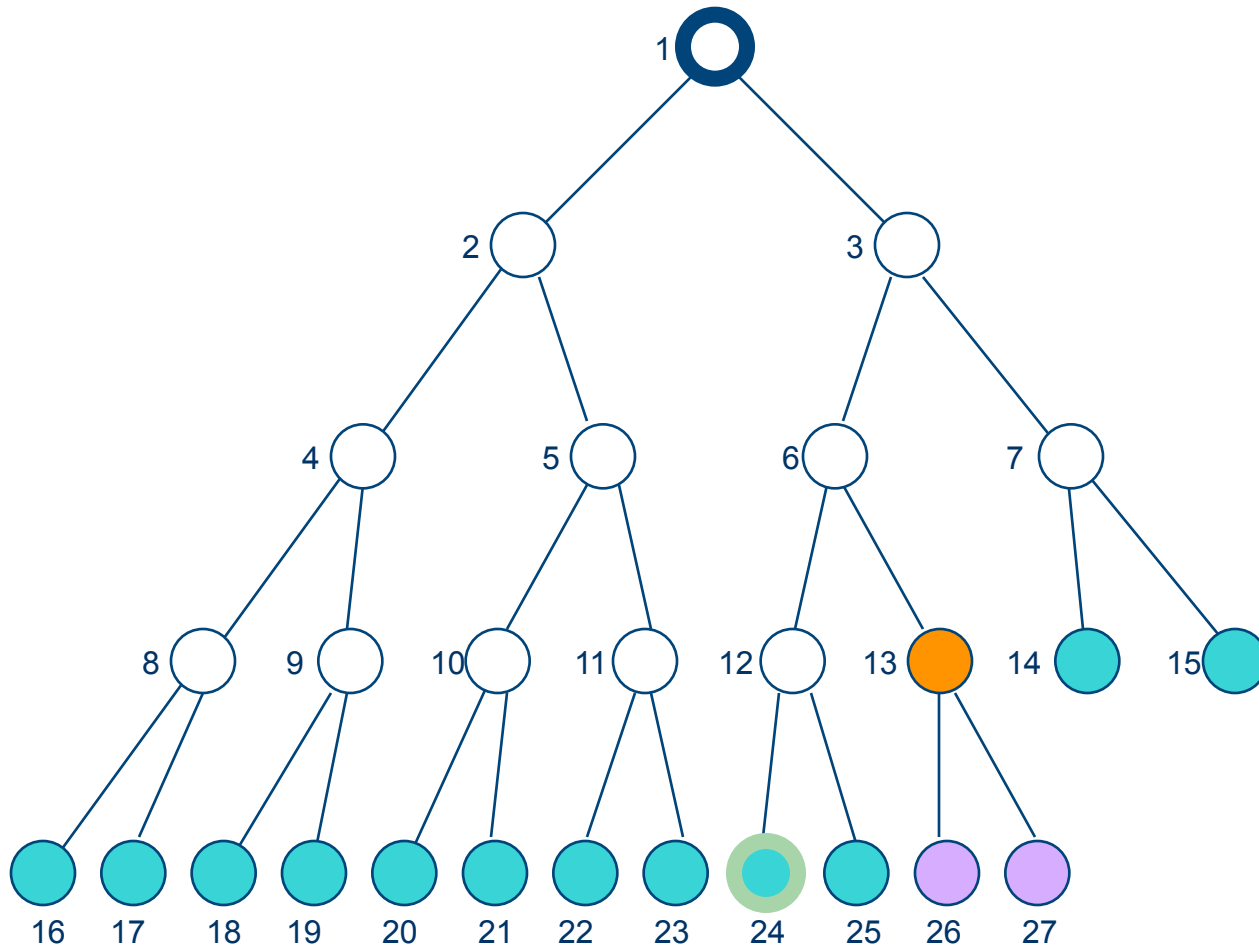
Fringe: [14,15,16,17,18,19,20,21,22,23,24,25] + [26,27]

Breadth-First Snapshot 14

Initial ●
Visited ○
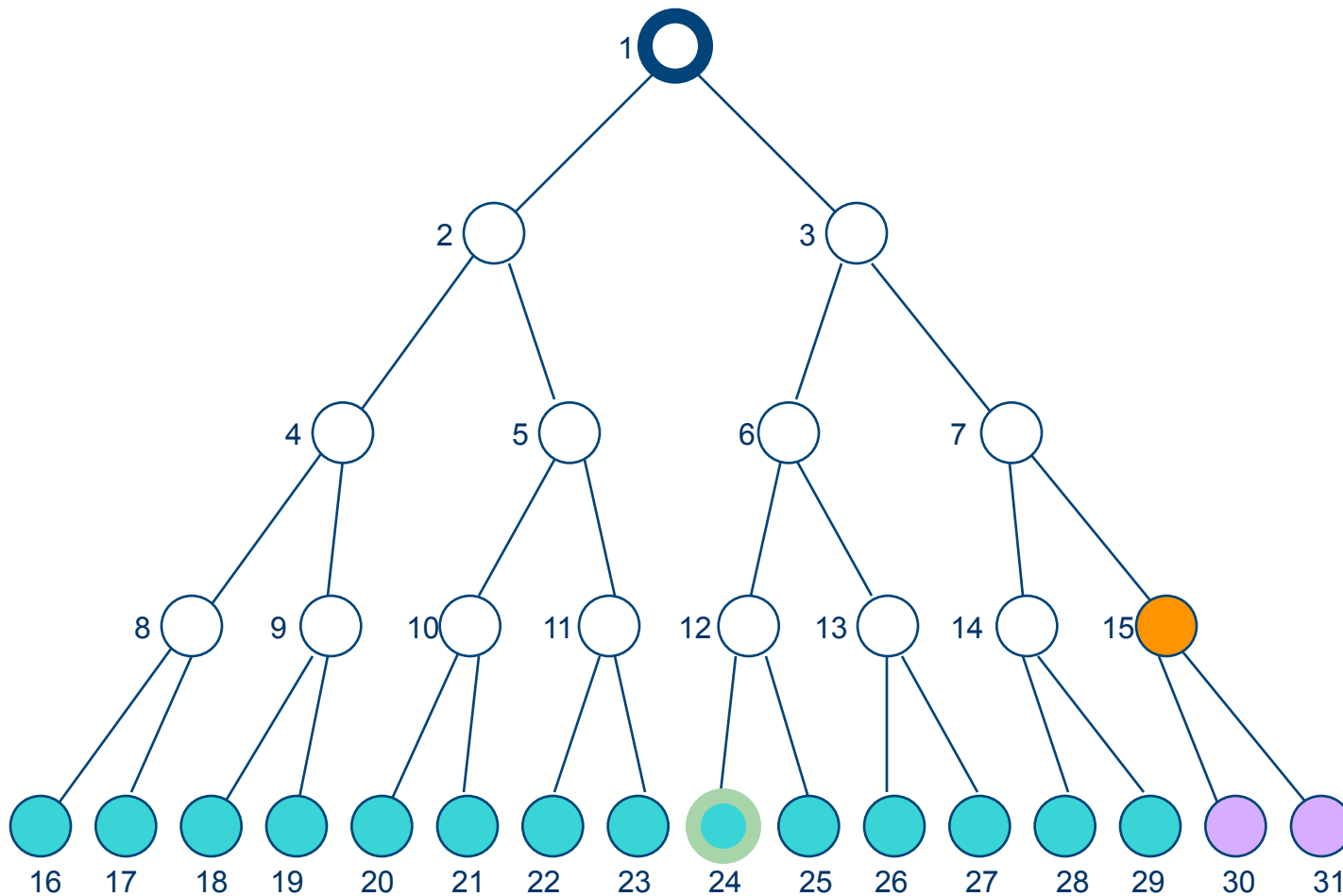Fringe ●
Current ●
Visible ●
Goal ●

Fringe: [15,16,17,18,19,20,21,22,23,24,25,26,27] + [28,29]

Breadth-First Snapshot 15

Initial

Visited

Fringe
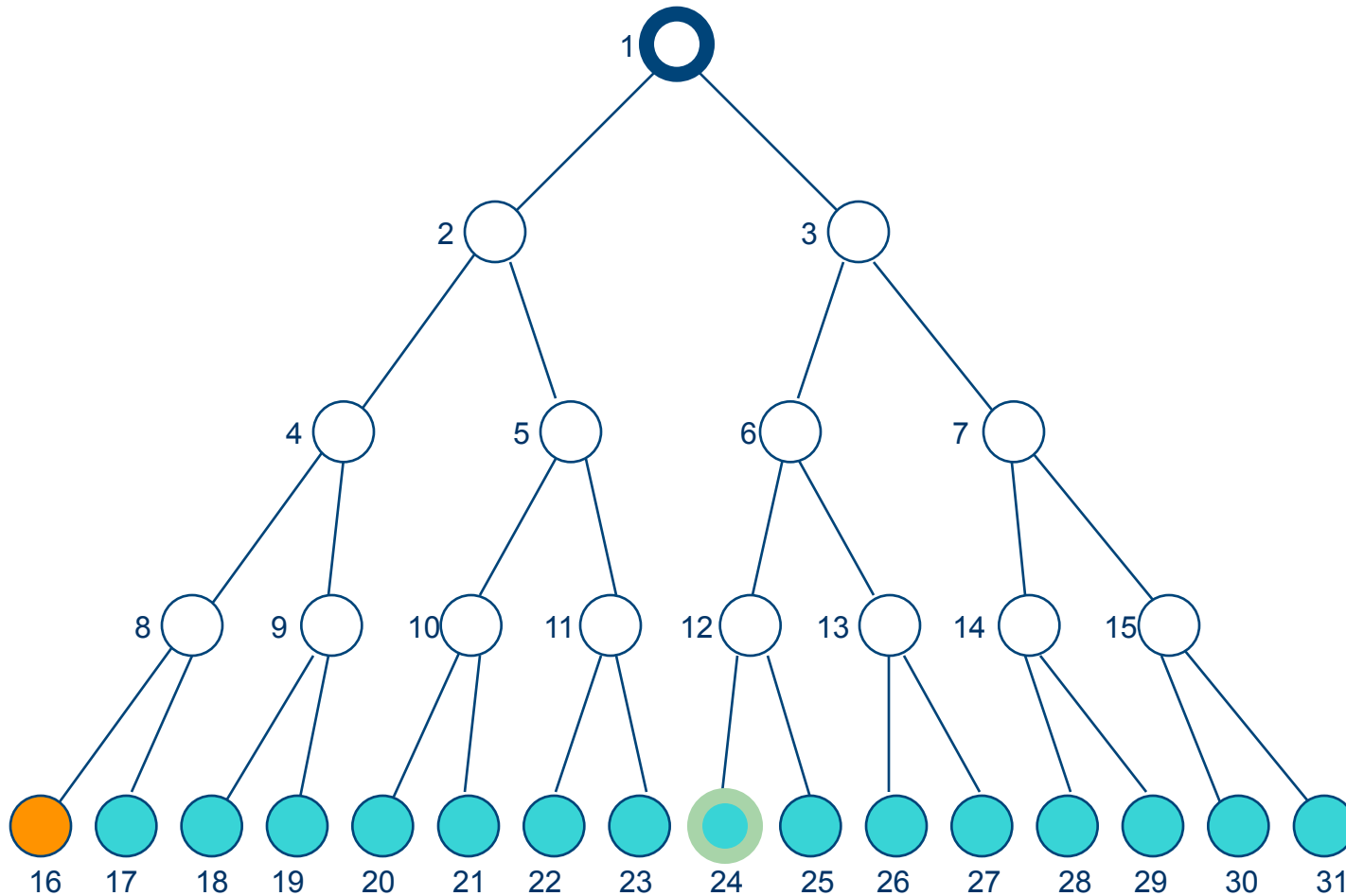
Current

Visible

Goal

Fringe: [15,16,17,18,19,20,21,22,23,24,25,26,27,28,29] + [30,31]

Breadth-First Snapshot 16


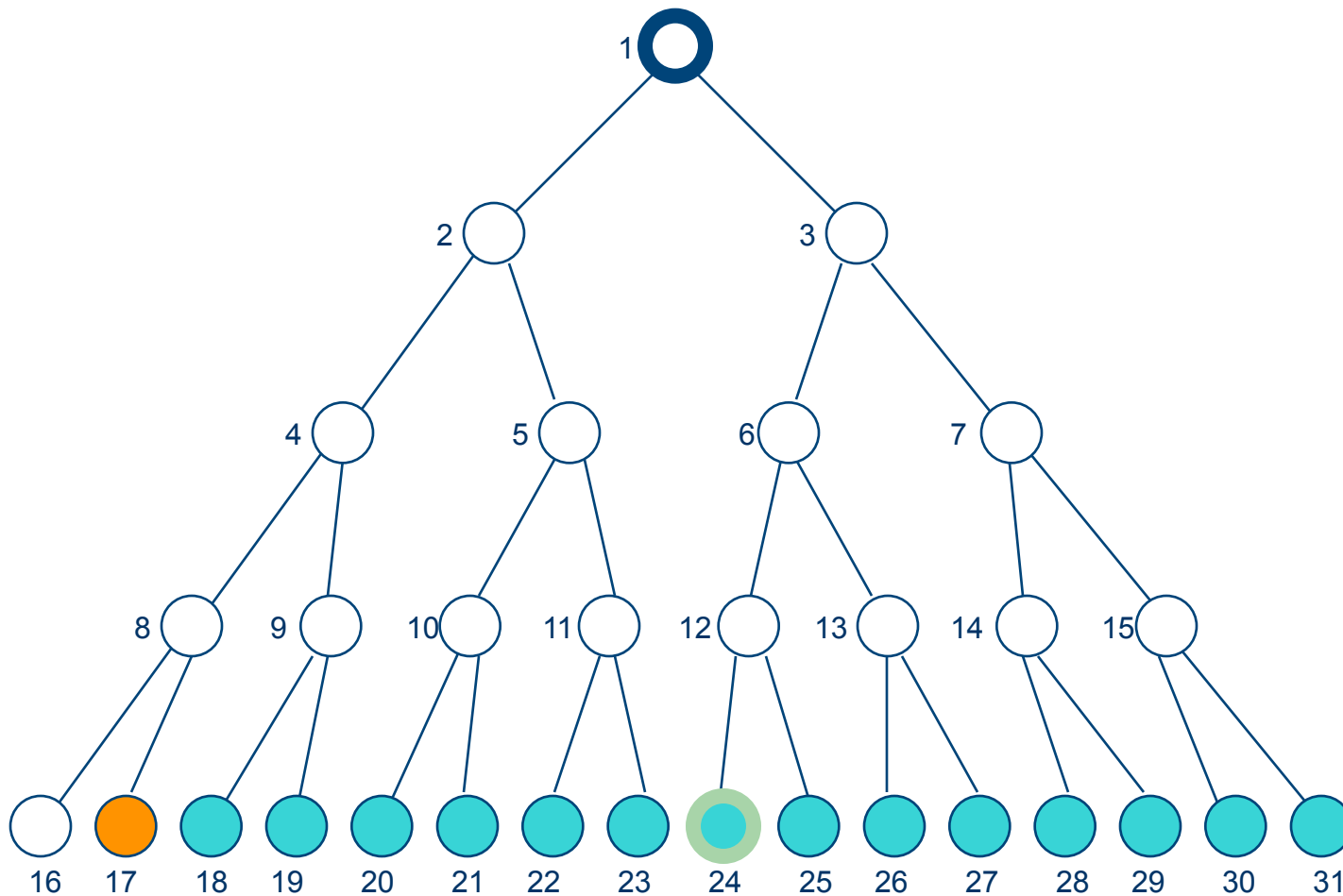Initial
Visited
Fringe
Current
Visible
Goal

Fringe: [17,18,19,20,21,22,23,24,25,26,27,28,29,30,31]

Breadth-First Snapshot 17

Initial
Visited
Fringe
Current
Visible
Goal

Fringe: [18,19,20,21,22,23,24,25,26,27,28,29,30,31]

Breadth-First Snapshot 18

Fringe: [19,20,21,22,23,24,25,26,27,28,29,30,31]

Breadth-First Snapshot 19

Initial
Visited
Fringe
Current
Visible
Goal
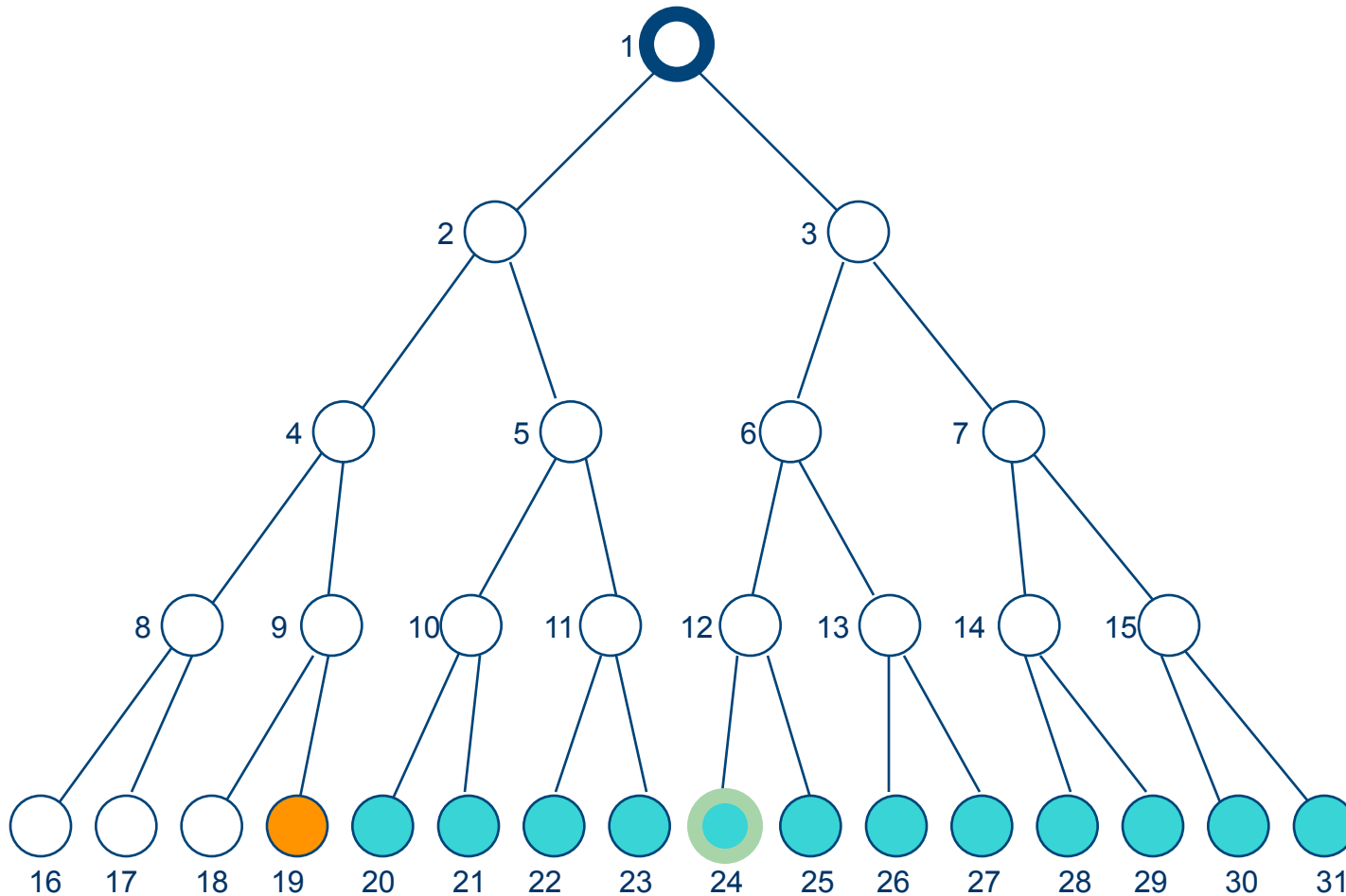
Fringe: [20,21,22,23,24,25,26,27,28,29,30,31]

Breadth-First Snapshot 20
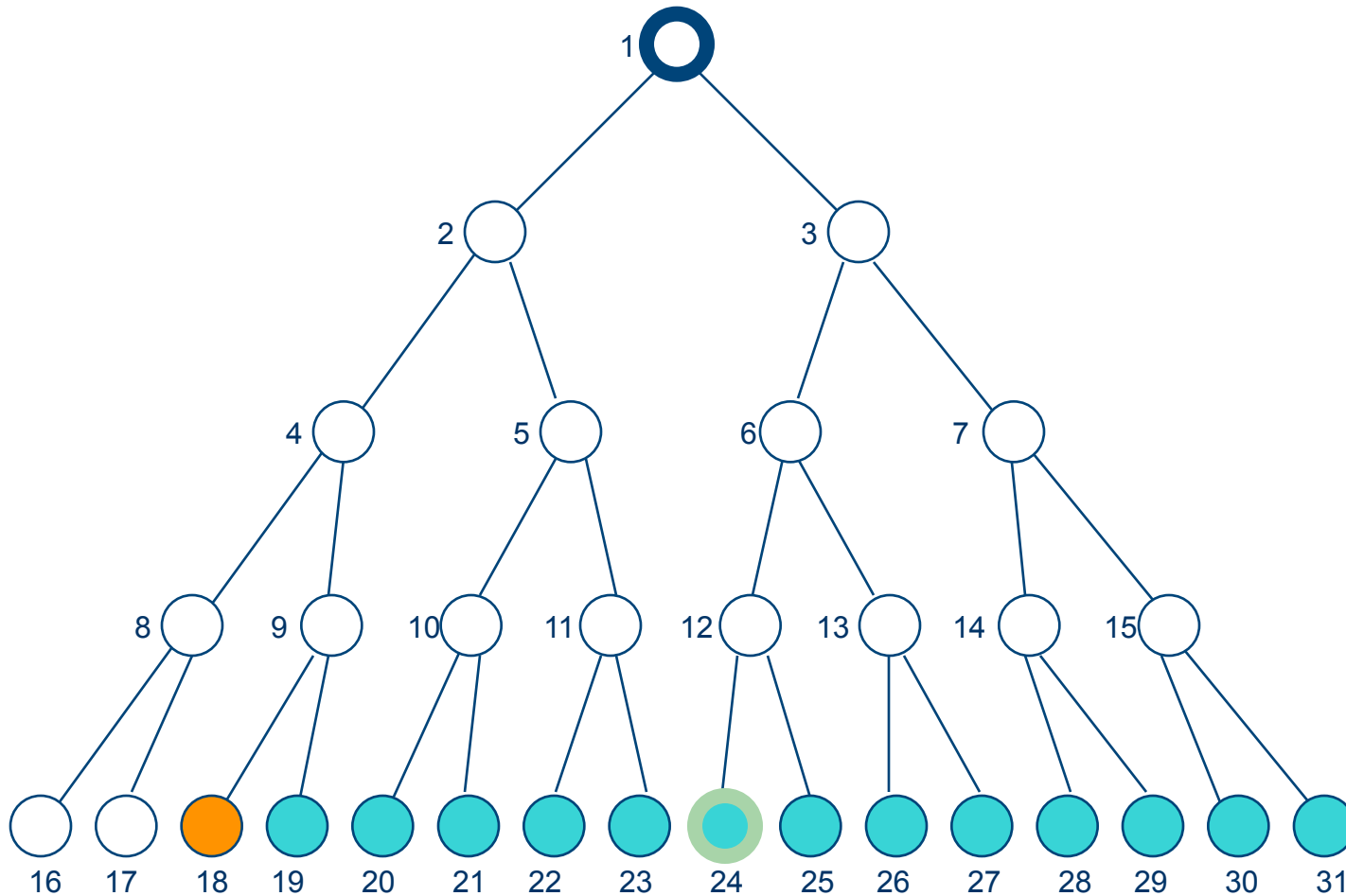
Initial
Visited
Fringe
Current
Visible
Goal
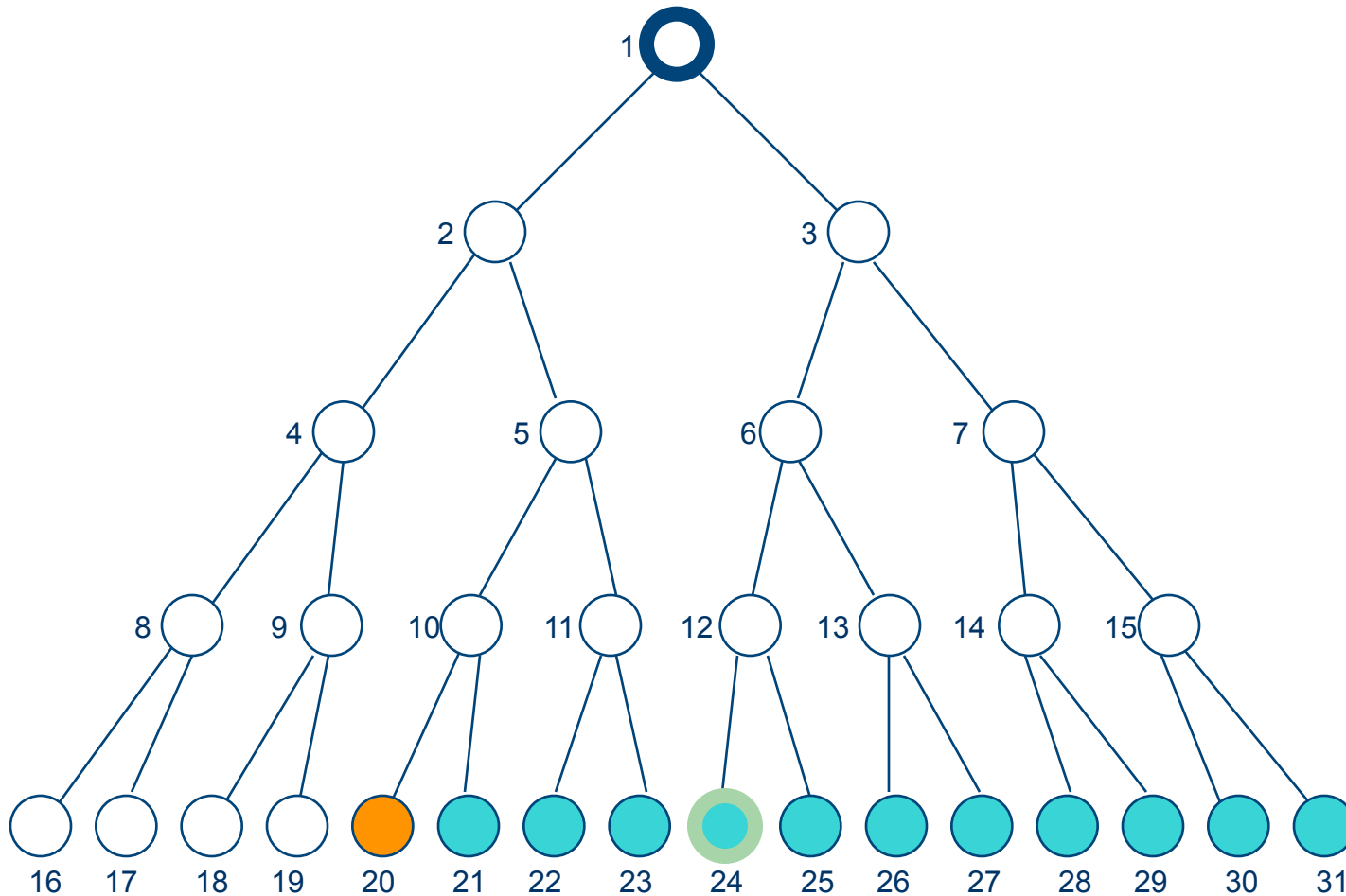
Fringe: [21,22,23,24,25,26,27,28,29,30,31]

Breadth-First Snapshot 21

1

2 3

4 5 6 7

8 9 10 11 12 13 14 15

16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
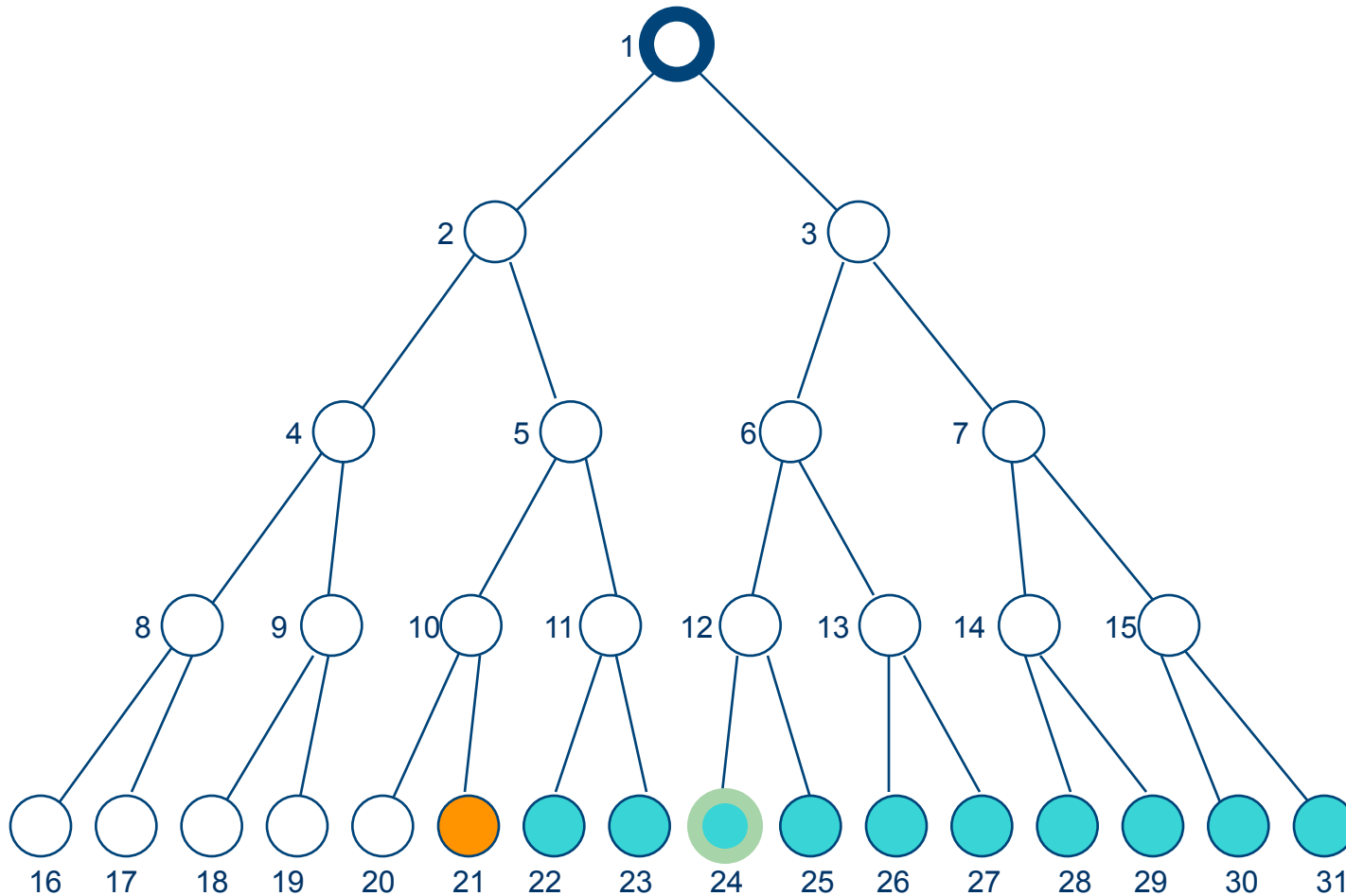
Fringe: [22,23,24,25,26,27,28,29,30,31]

Breadth-First Snapshot 22

Initial ●
Visited ○
Fringe ●
Current ●
Visible ●
Goal ●
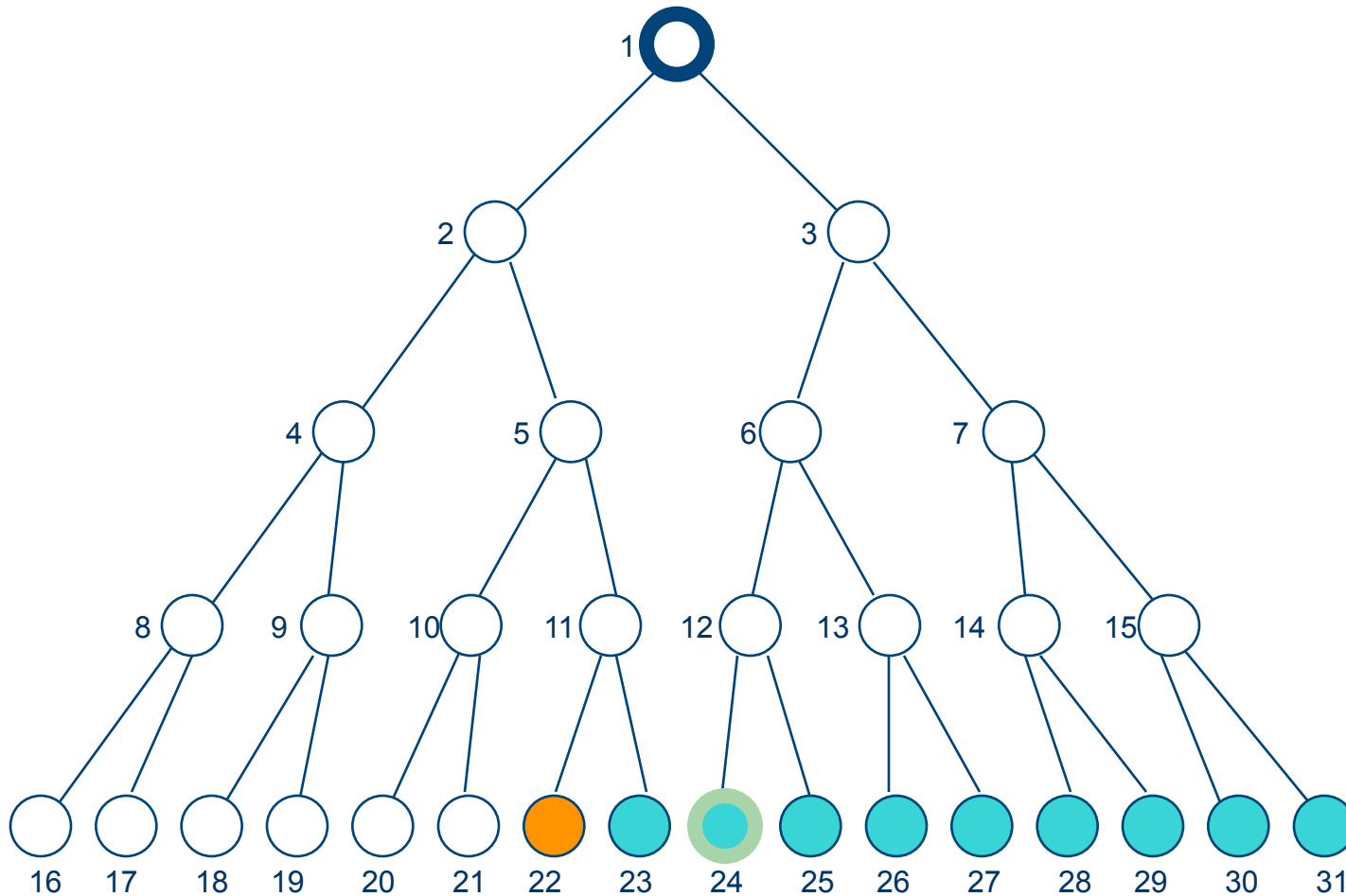
Fringe: [23,24,25,26,27,28,29,30,31]

Breadth-First Snapshot 23

Initial
Visited
Fringe
Current
Visible
Goal
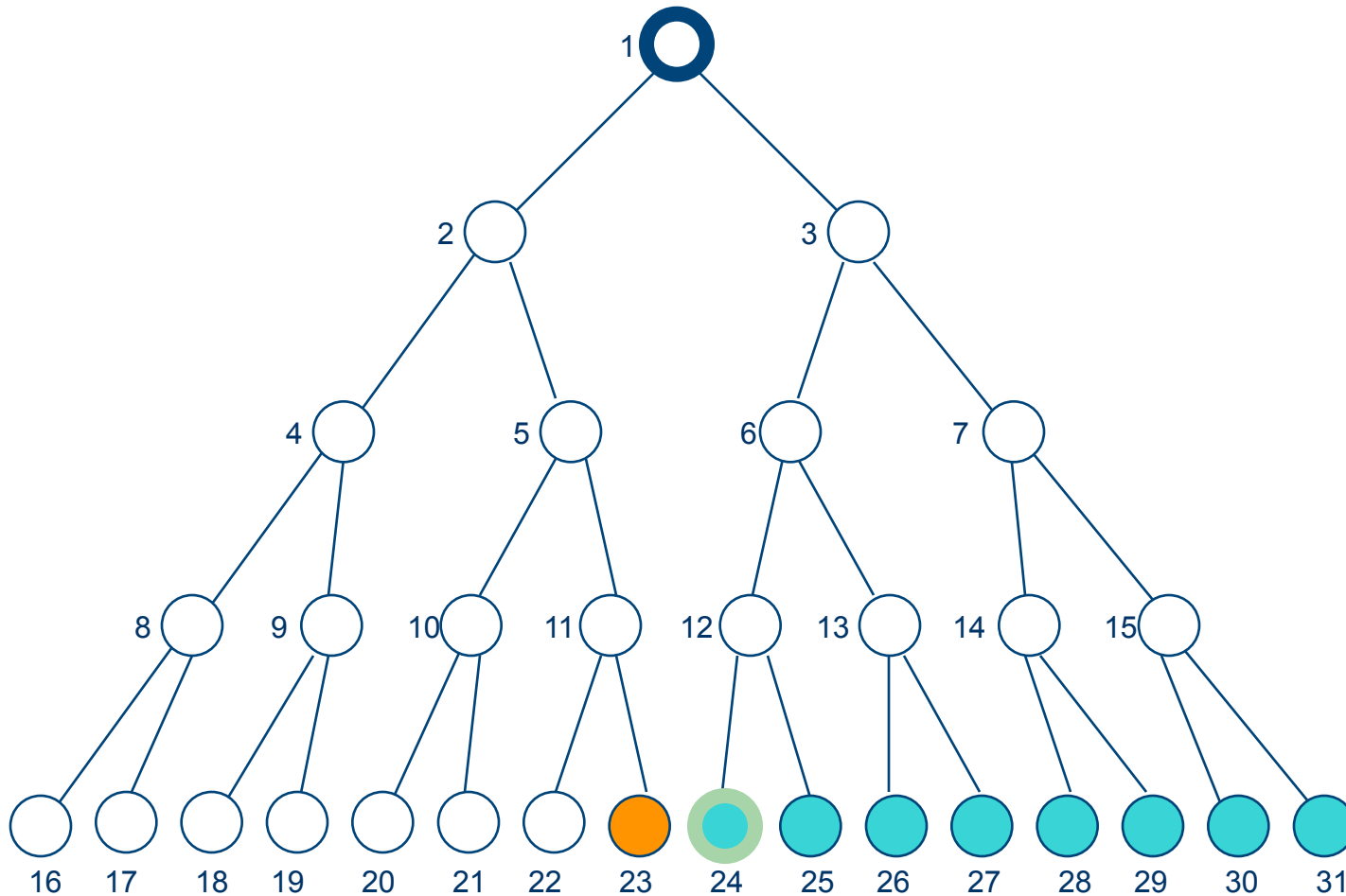
Fringe: [24,25,26,27,28,29,30,31]

Breadth-First Snapshot 24
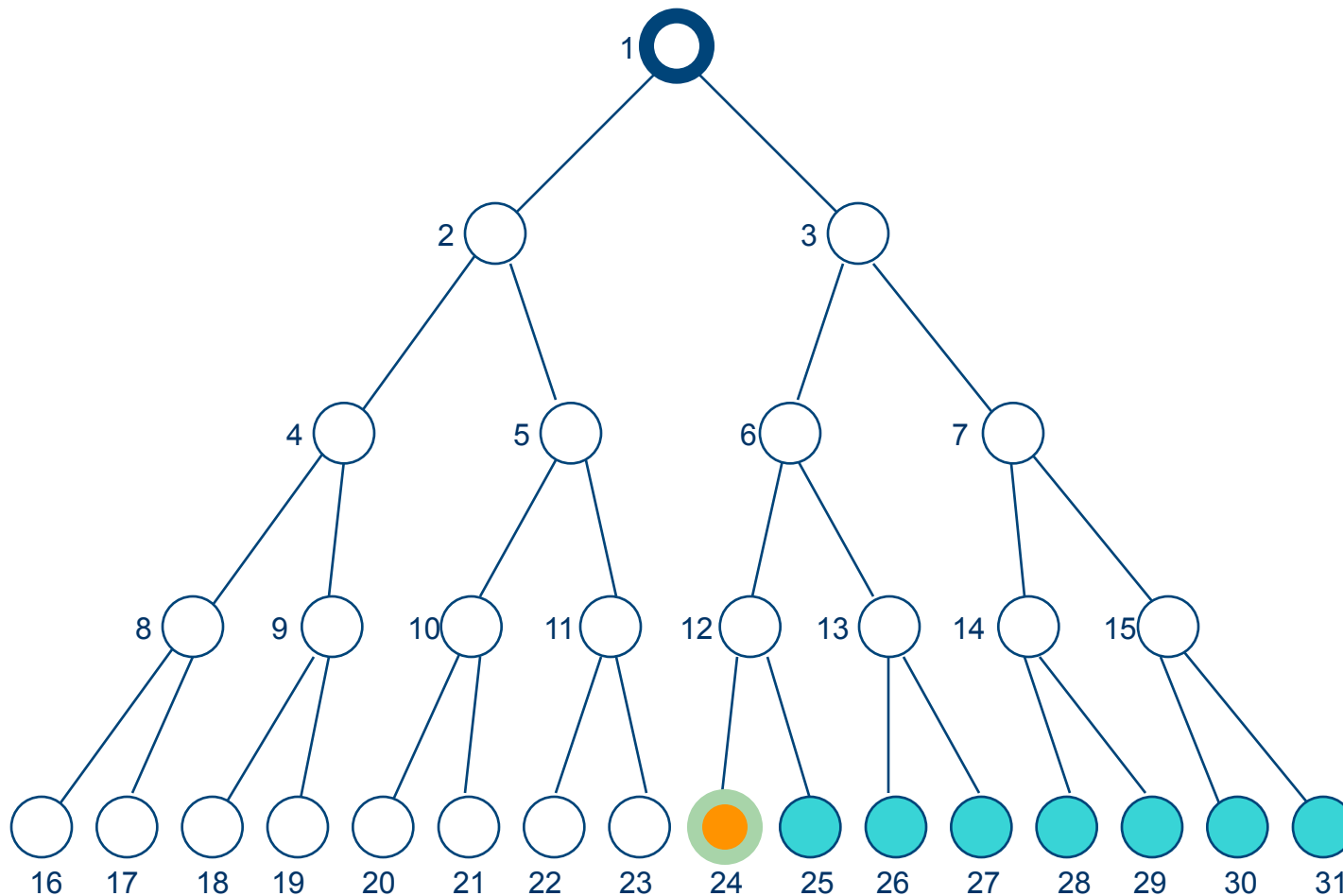
Initial ●
Visited ○
Fringe ●
Current ●
Visible ●
Goal ●

Note:
The goal test is positive for this node, and a solution is found in 24 steps.

Fringe: [25,26,27,28,29,30,31]

Breadth First Search

- Complete
- Optimal if cost is increasing with path depth.
- Computational complexity $O(b^d)$, where b is the branching factor and d is the depth
- Space (memory) complexity $O(b^d)$

Tree Search

- Search Algorithms
  1. Breadth First Search
  2. Depth First Search
  3. A*

Depth-First

- Expands one of the nodes at the deepest level of the tree

Depth-First Snapshot 1

Initial
Visited
Fringe
Current
Visible
Goal

Depth-First Snapshot 2

Initial 🔴
Visited ⚪
Fringe 🔵
Current 🟠
Visible 🟣
Goal 🟢

Depth-First Snapshot 3

Initial 🔴
Visited ⚪
Fringe 🔵
Current 🟠
Visible 🟣
Goal 🟢

Depth-First Snapshot 4
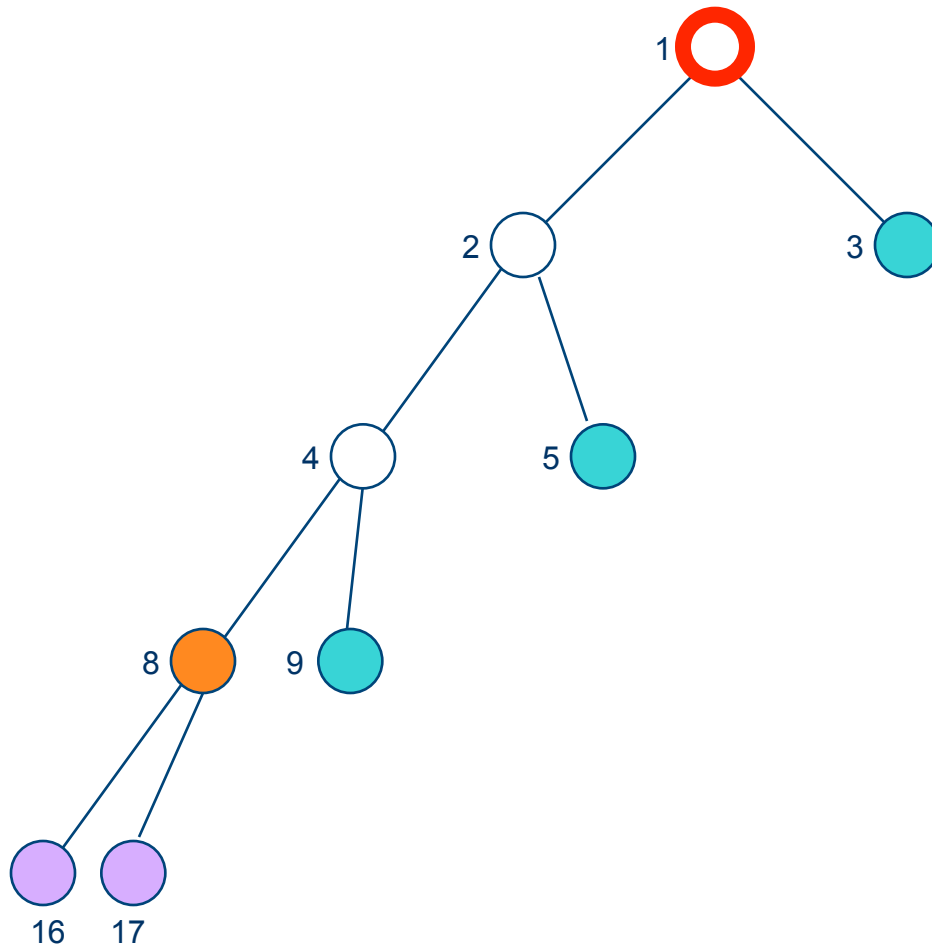
Initial ⬤
Visited ◯
Fringe ⬤
Current ⬤
Visible ⬤
Goal ⬤

Depth-First Snapshot 5
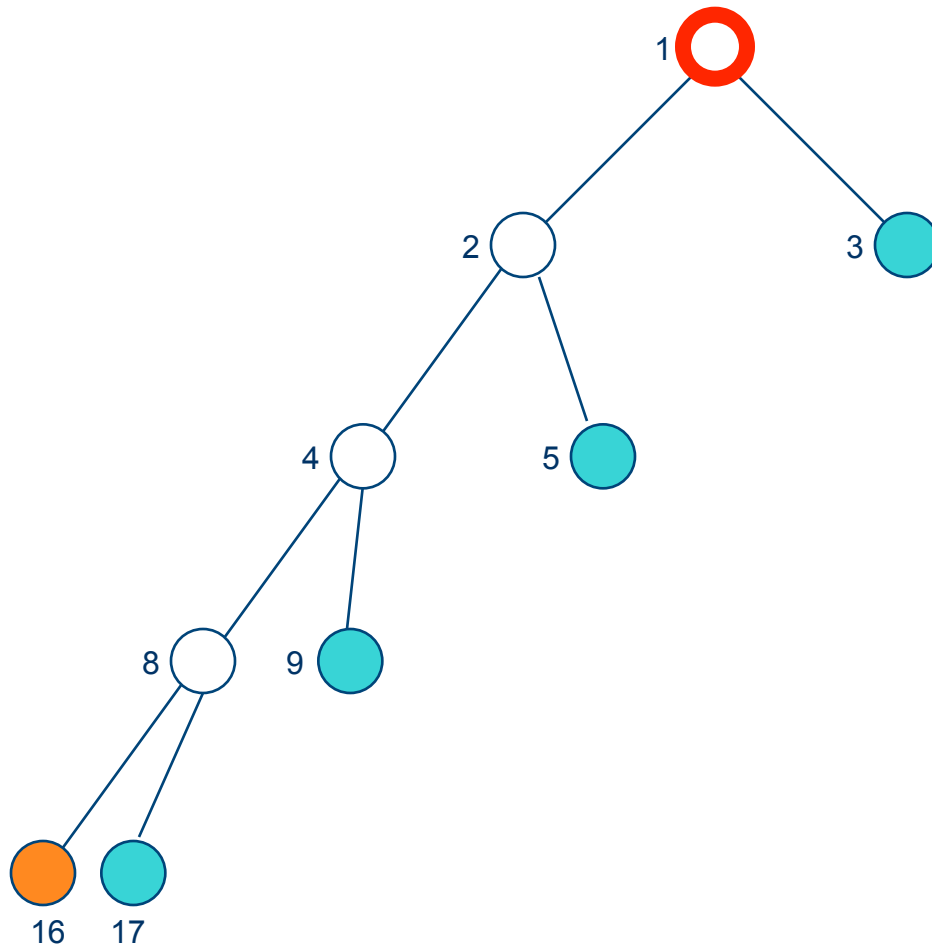
Initial 🔴
Visited ⚪
Fringe 🔵
Current 🟠
Visible 🟣
Goal 🟢

Depth-First Snapshot 6
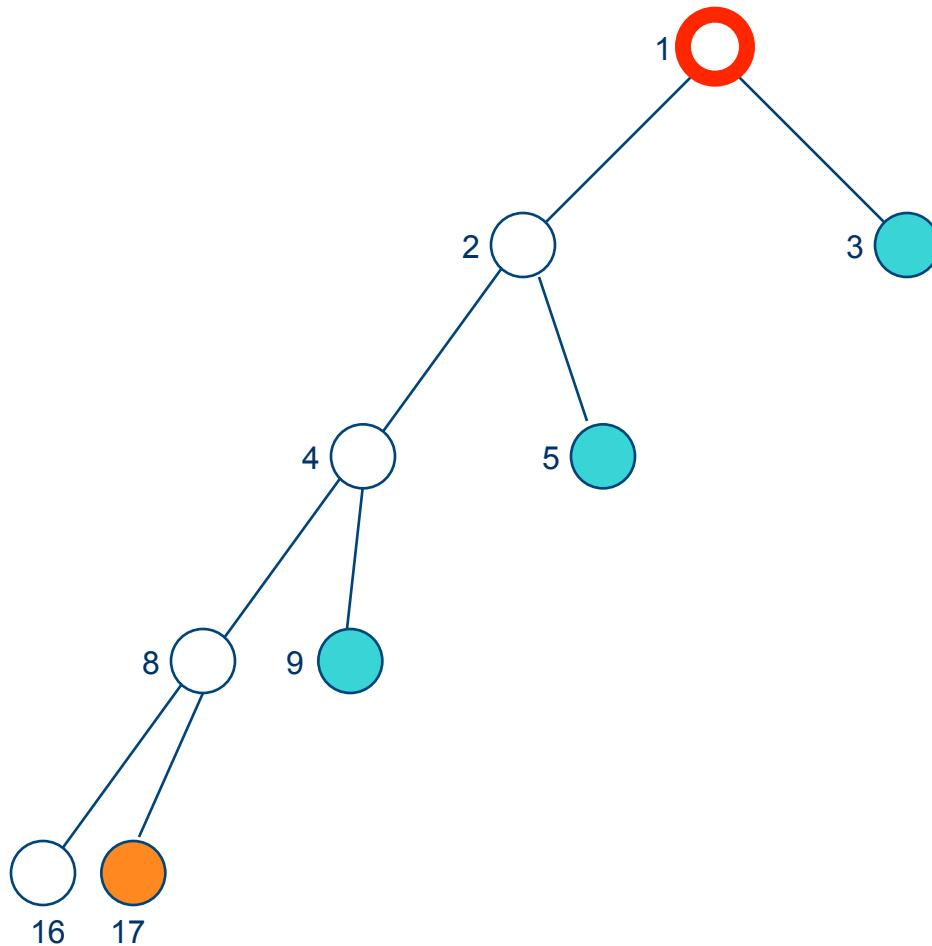
Initial 🔴
Visited ⚪
Fringe 🔵
Current 🟠
Visible 🟣
Goal 🟢

Depth-First Snapshot 7

Initial 🔴
Visited ⚪
Fringe 🔵
Current 🟠
Visible 🟣
Goal 🟢

Depth-First Snapshot 8
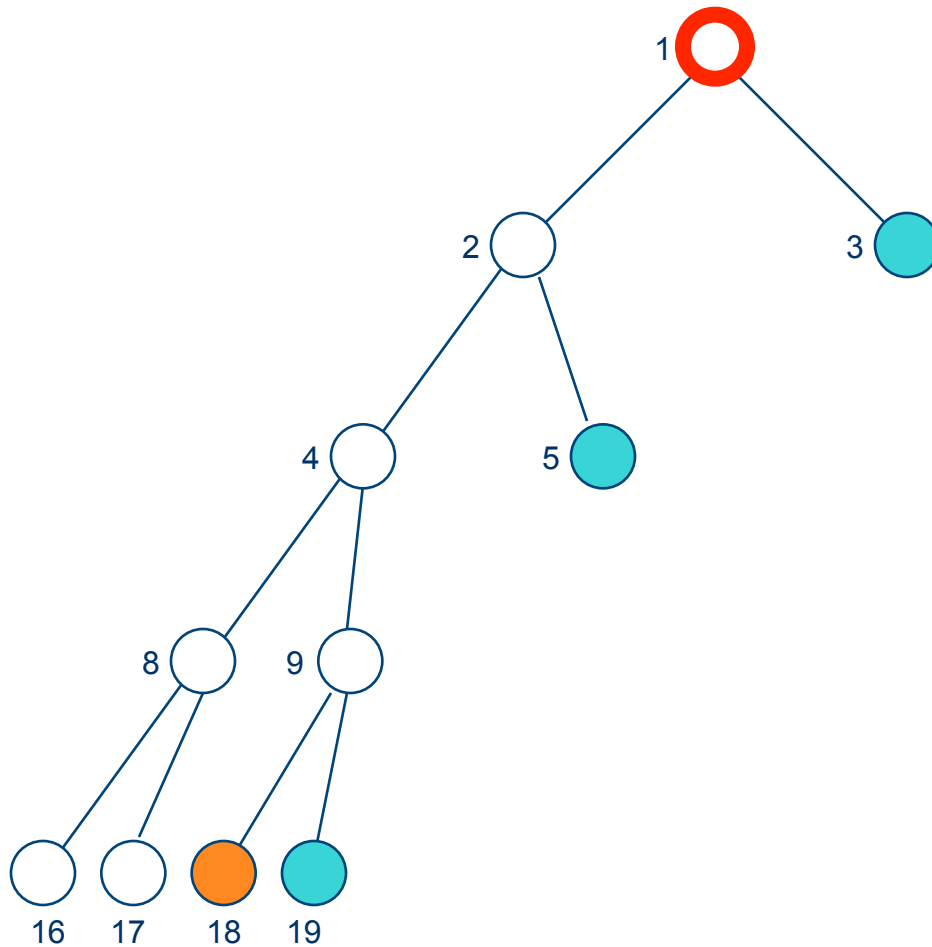
Initial ⬤
Visited ◯
Fringe ⬤
Current ⬤
Visible ⬤
Goal ⬤

Depth-First Snapshot …

Initial
Visited
Fringe
Current
Visible
Goal

Depth First Search

- Complete if finite depth
- NOT Optimal if we take first goal found
- Computational complexity $O(b^m)$, where b is the branching factor and m is the depth
- Space (memory) complexity $O(bm)$

Graph Search: Outline

- Search Algorithms
  1. Breadth First Search
  2. Depth First Search
  3. A*

Motion Planning: A* Search

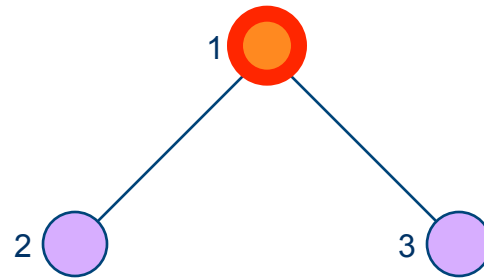- There are a set of algorithms called "Best-First Search"

  - They try to search the children of the "best" node to expand.

- A* is a best first search algorithm

  - It attempts to make the best node the one that will find the optimal solution and do so in less time.

Motion Planning:
A* Search

- A* is optimal and complete, but can take time…

    - Its complexity depends on the heuristic, but is exponential with the size of the graph.

Motion Planning:
A* Search

- We evaluate a node n for expansion based on the function:

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

- Where

$g(n)$ = path cost from the start node to n

$h(n)$ = estimated cost of the cheapest

path from node n to the goal

Motion Planning:
A* Search

- Example: Cost for one particular node

$$f(n) = g(n) + h(n)$$
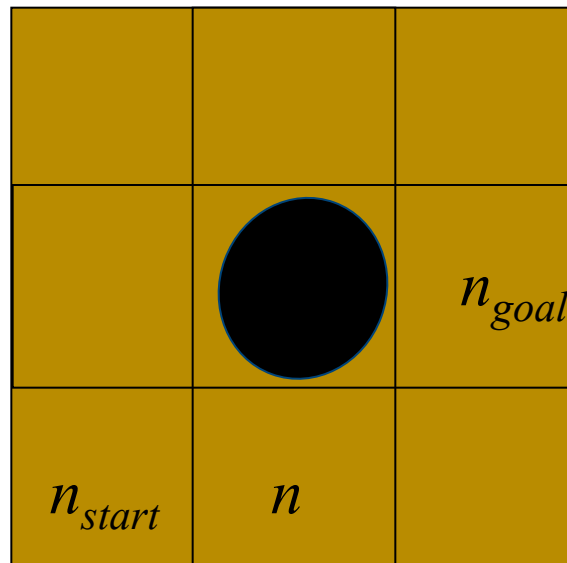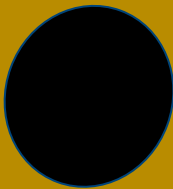
$$g(n) = 1$$
$$h(n) = \sqrt{2}$$

Motion Planning: A* Search

- Example: Cost for each node

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

g=2 h=√3	g=3 h=√2	g=4 h=1
g=1 h=2		n_{goal}
n_{start}	g=1 h=√2	g=2 h=1

Motion Planning:
A* Search

- The strategy is to expand the node with the cheapest path (lowest f).

- This is proven to be complete and optimal, if $h(n)$ is an *admissible* heuristic.

Motion Planning:
A* Search

- Here, $h(n)$ is an admissible heuristic is one that never *overestimates* the cost to the goal

  - Example: the Euclidean distance.

Motion Planning:
A* Search

- Search example: Iteration 1

$$Fringe\ set = \{\ f_1 = 2.4,\ f_2 = 3\}$$
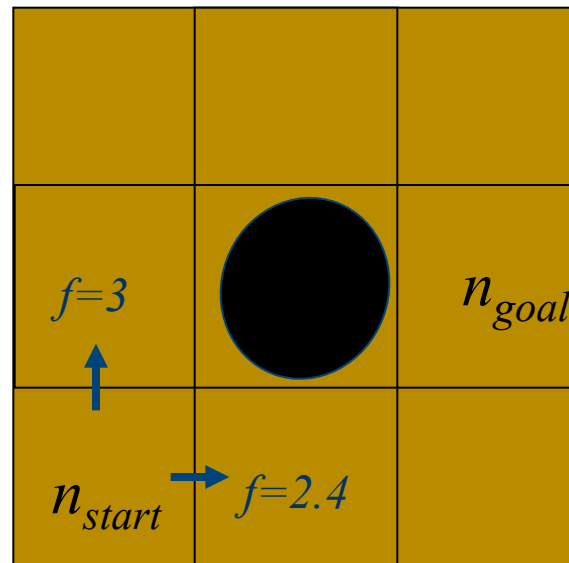
Motion Planning: A* Search

- Search example: Iteration 2

$$Fringe\ set = \{f_2 = 3, f_3 = 3\}$$
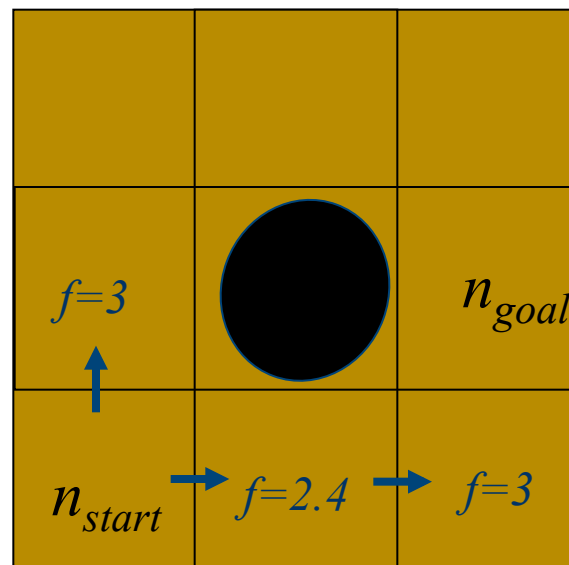
Motion Planning: A* Search

- Search example: Iteration 3

$$Fringe\ set = \{f_3 = 3, f_4 = 3.8\}$$
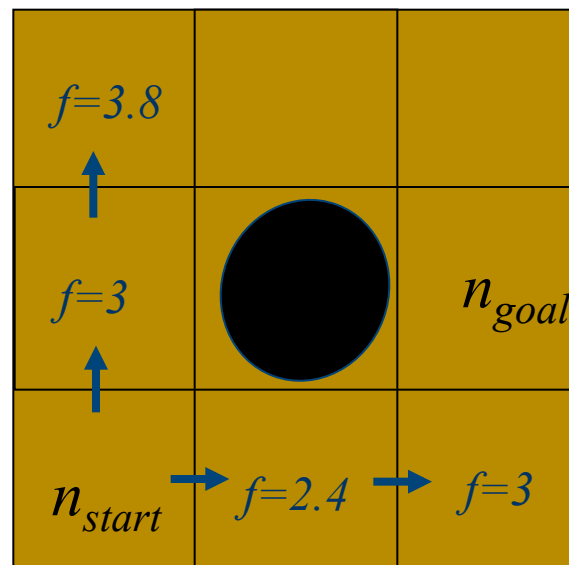
Motion Planning:
A* Search

- Search example: Iteration 4
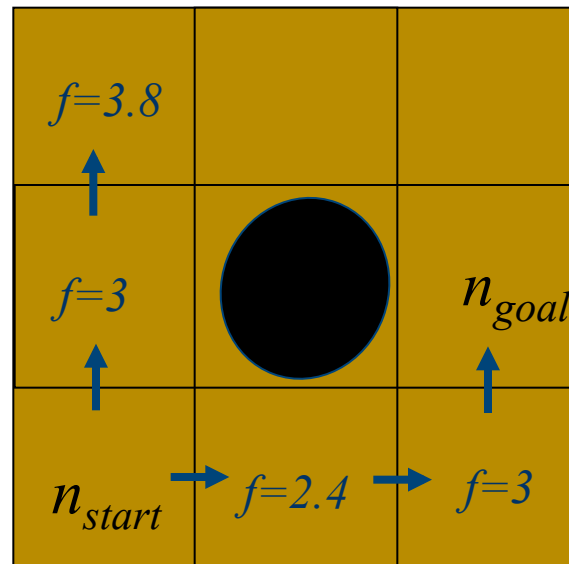
Motion Planning: Final Notes

- A * is often used as a global planner
- Planner that considers kinematic/dynamic constraints is used for local planning.

MP: Outline

1. Multi-Query PRMs
2. Graph Search
3. Artificial Potential Fields

Artificial Potential Fields

- Potential field
    - Define a function over the free space that has a global minimum at the goal configuration and follow its steepest descent
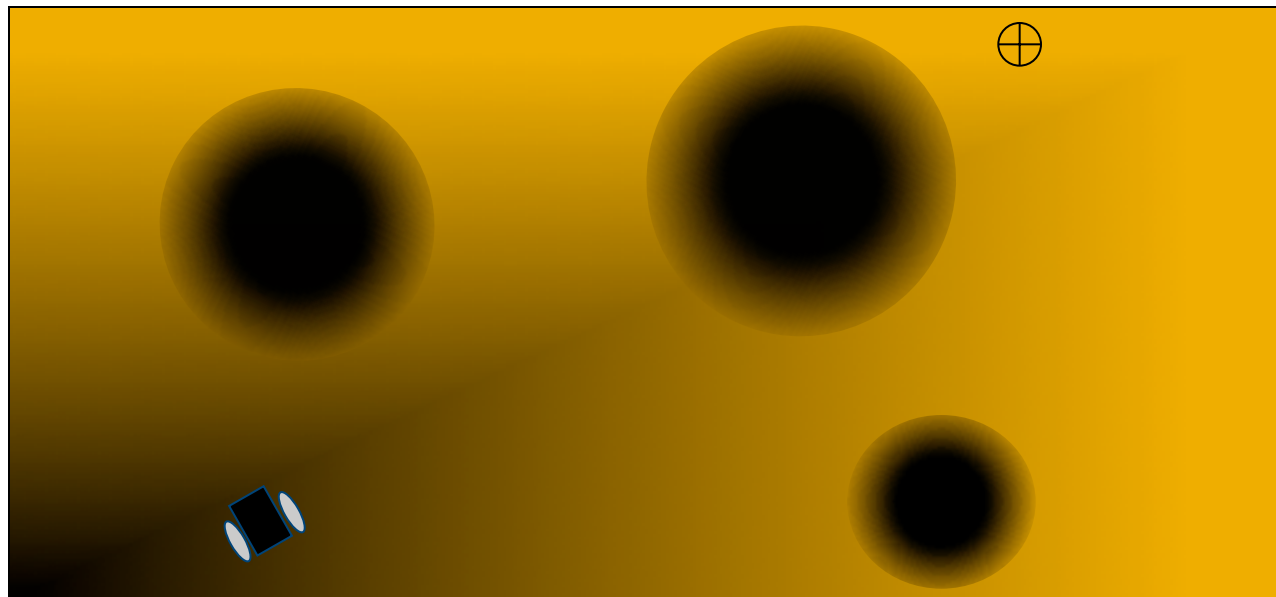
Artificial Potential Fields

- Electric Potentials
  - The electric potential V_E (J C^{-1}) created by a point charge Q, at a distance r from the charge (relative to the potential at infinity), can be shown to be

$$V_E = \frac{1}{4\pi\varepsilon_0} \frac{Q}{r}$$
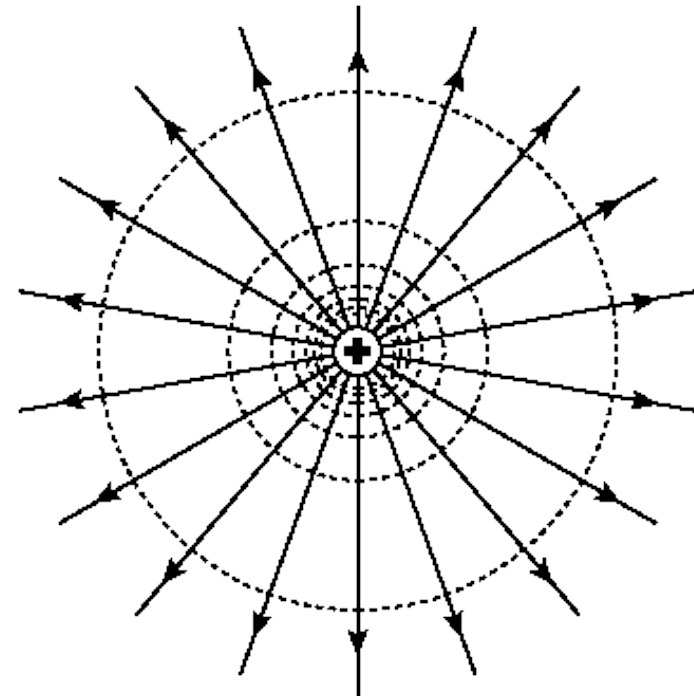
Artificial Potential Fields

- ## Electric Fields
  - The electric field intensity E is defined as the force per unit positive charge that would be experienced by a point charge
  - It is obtained by taking the negative gradient of the electric potential

$$E = -\nabla V_E$$
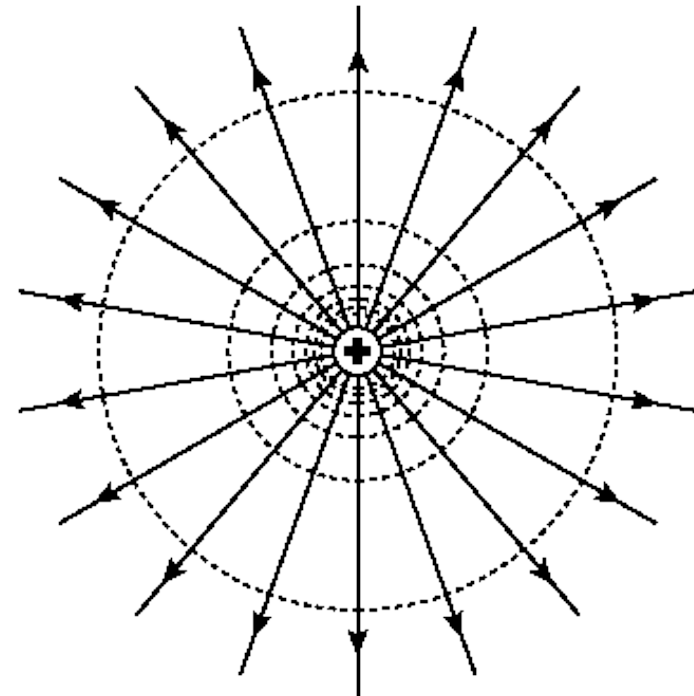
Artificial Potential Fields

- Electric Potential Fields
  - Different arrangements of charges can lead to various fields
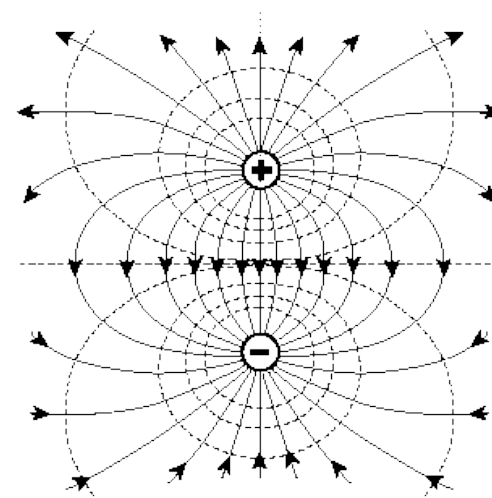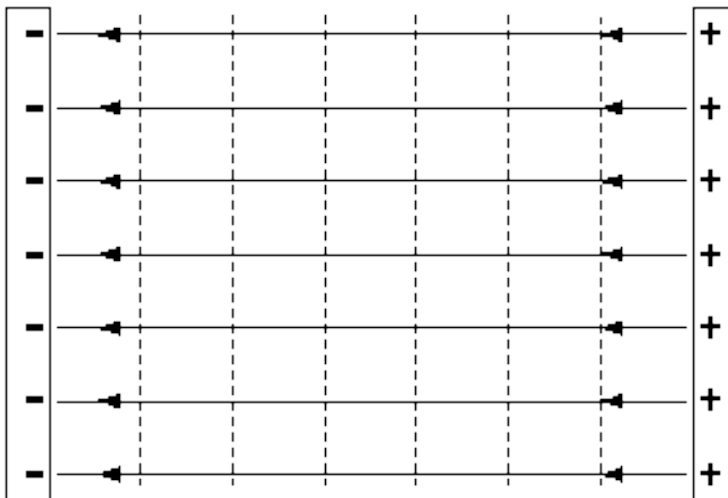
Artificial Potential Fields

- In APFs, the robot is treated as a *point under the influence* of an artificial potential field.

  - Electrical analogy: The generated robot movement is similar to an electric charge under the force of an electric field

  - Mechanical analogy: The generated robot movement is similar to a ball rolling down the hill

Artificial Potential Fields

- In APFs

  - **Goals** generates **attractive** force

  - **Obstacles** generate **repulsive** forces

Artificial Potential Fields

- For a given configuration space and desired goal, place potentials on obstacles and goals

q_{goal}

$\oplus$

q
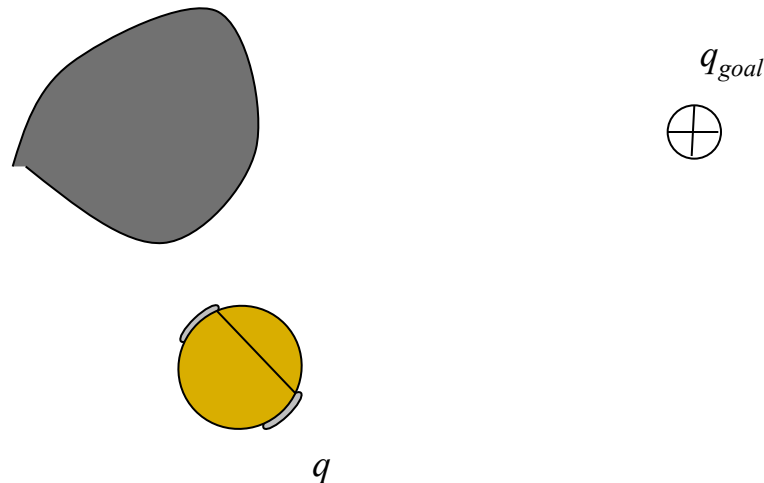
Artificial Potential Fields

- For a given configuration space and desired goal, place potentials on obstacles and goals
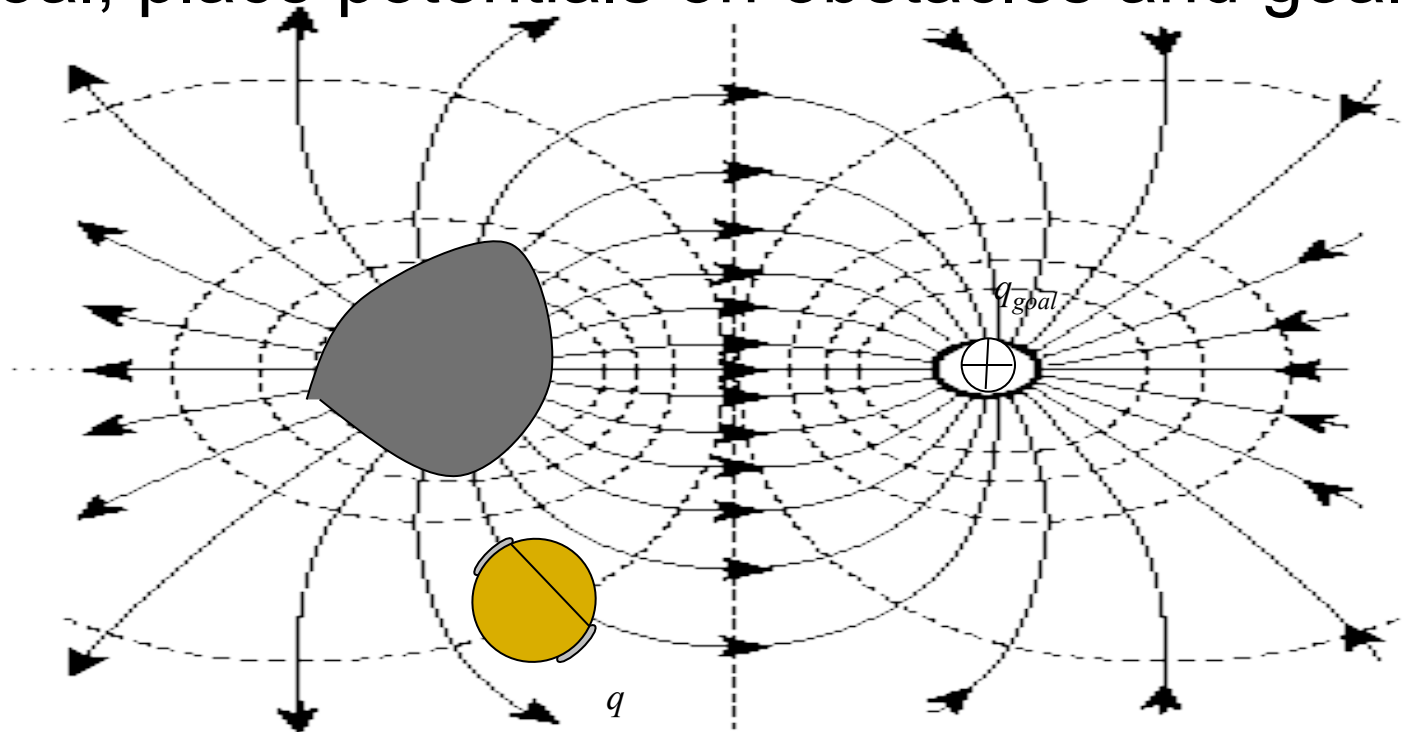
Artificial Potential Fields

- For any robot configuration q, the forces felt by the robot can be calculated to steer the robot towards the goal.
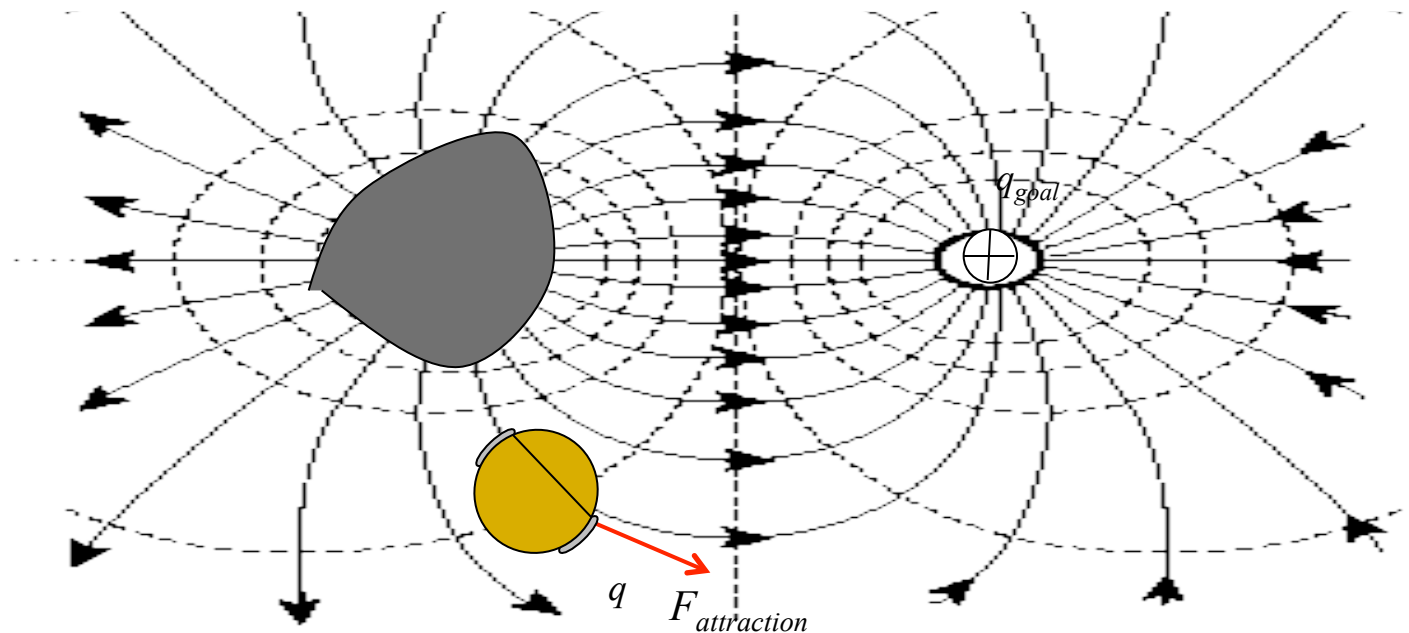
Artificial Potential Fields
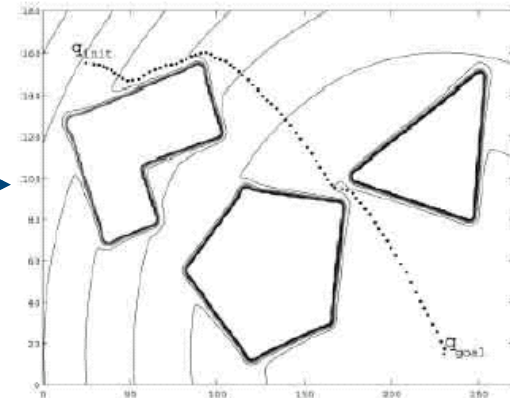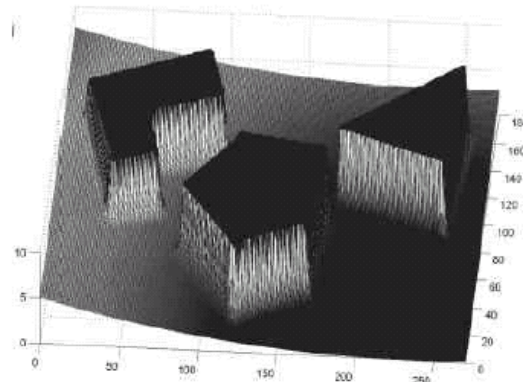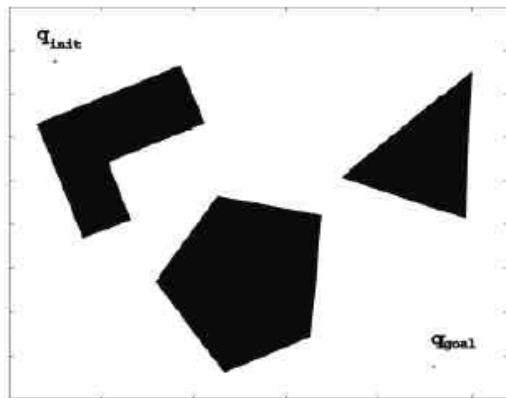
Potential Field Generation

- Given potential functions U, Generate artificial force field $F(q)$

  - Sum all potentials (repulsive and attractive).

  - Differentiate to determine forces

  - Note: functions must be differentiable

$$F(q) = -\nabla U(q)$$

$$= -\nabla U_{att}(q) - \nabla U_{rep}(q)$$

$$= \begin{bmatrix} -\delta U/\delta x \\ -\delta U/\delta y \end{bmatrix}$$

Attractive Potential Fields

- Parabolic function representing the Euclidean distance $\rho_{goal}(q) = \| q - q_{goal} \|$ to the goal.

$$U_{att}(q) = \frac{1}{2} k_{att} \rho^2_{goal}(q)$$

- Attracting force converges linearly towards 0 (goal)

$$F_{att}(q) = -\nabla U_{att}(q)$$
$$= -k_{att}(q - q_{goal})$$

Repulsive Potential Fields

- Generate a barrier around the obstacle

  - Does not influence robot if far from the obstacle

  $$U_{rep}(q) = \begin{cases} \dfrac{1}{2} k_{rep} \left[\dfrac{1}{\rho(q)} - \dfrac{1}{\rho_0} \right]^2 & if \ \ \rho(q) \le \rho_0 \\ \\ 0 & if \ \ \rho(q) > \rho_0 \end{cases}$$

  - Where $\rho(q) = || \ q - q_{obst} \ ||$ is the minimum distance to the object

Repulsive Potential Fields

- Field is positive or zero and tends to infinity as q gets closer to the object

$$F_{rep}(q) = -\nabla U_{rep}(q)$$

$$= \begin{cases} k_{rep} \left[\dfrac{1}{\rho(q)} - \dfrac{1}{\rho_0} \right] \dfrac{q - q_{obst}}{\rho^3(q)} & if \ \ \rho(q) \leq \rho_0 \\ \\ 0 & if \ \ \rho(q) > \rho_0 \end{cases}$$

91

Artificial Potential Fields

- Given current configuration of the robot q
  1. **Sum** total force **vectors** $F(q)$ generated by the potential fields.
  2. Set desired robot velocity (v, w) proportional to the force $F(q)$
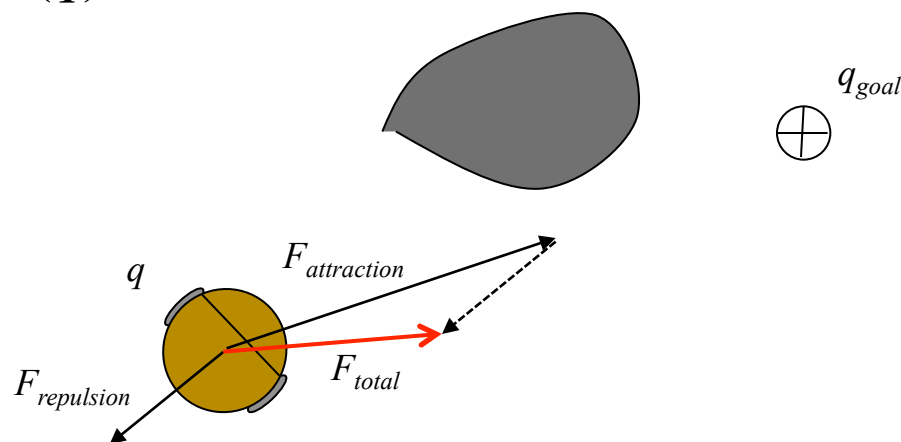
Artificial Potential Fields

- Local minimums

q_{goal}
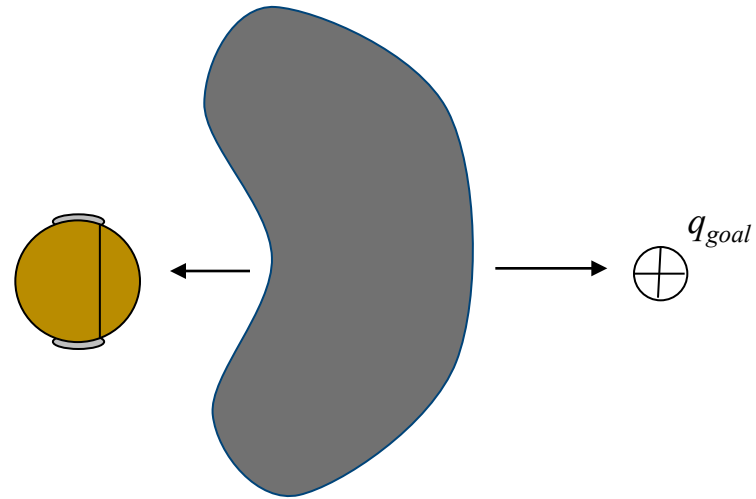
- If objects are not *convex* (i.e. concave), there exist situations where several minimal distances exist and can result in oscillations

- Not complete

93

Artificial Potential Fields

- Extended Potential Fields
  - Many modifications to potential fields have been done in order to improve completeness, optimality.
  - Example: Orientation based potentials
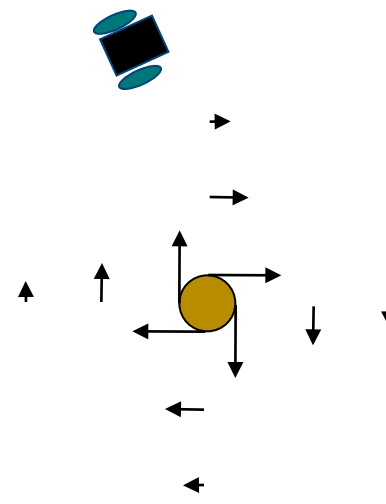   - Can increase potential depending on orientation of robot

Robot

Repulsion force

Object
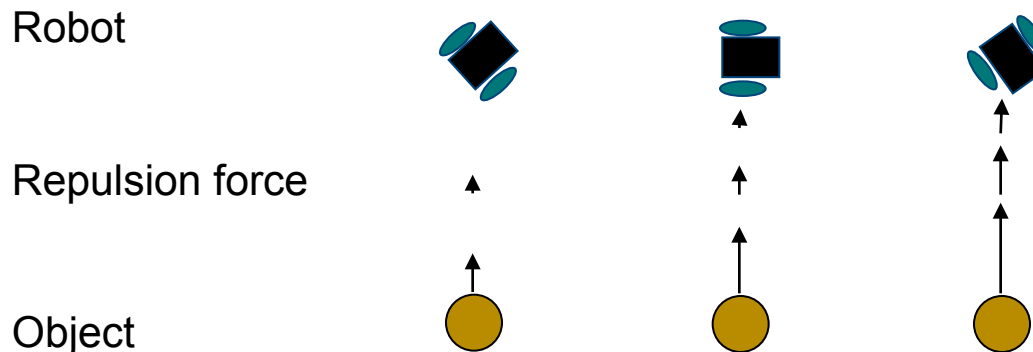
Artificial Potential Fields

- Extended Potential Fields
    - Also, can use rotational fields in one direction
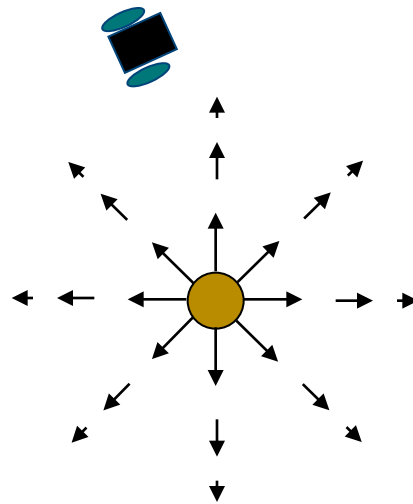
Linear source

Rotational source

Artificial Potential Fields

▪ Example:

http://www.youtube.com/watch?v=r9FD7P76zJs