Search

COMP3411: Artificial Intelligence

UNSW SYDNEY

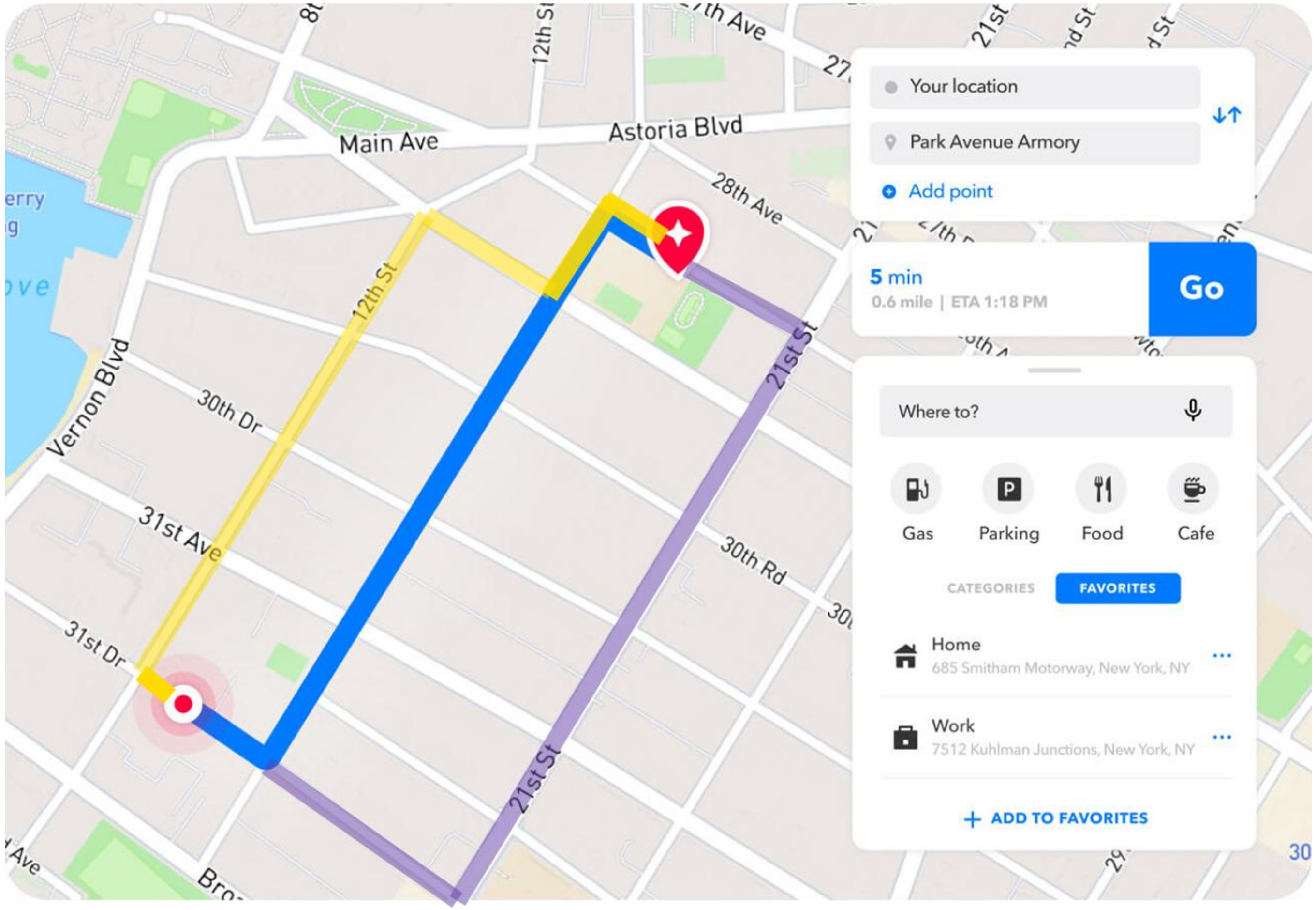
Search

Path 1:
Cost $15, takes 20 mins

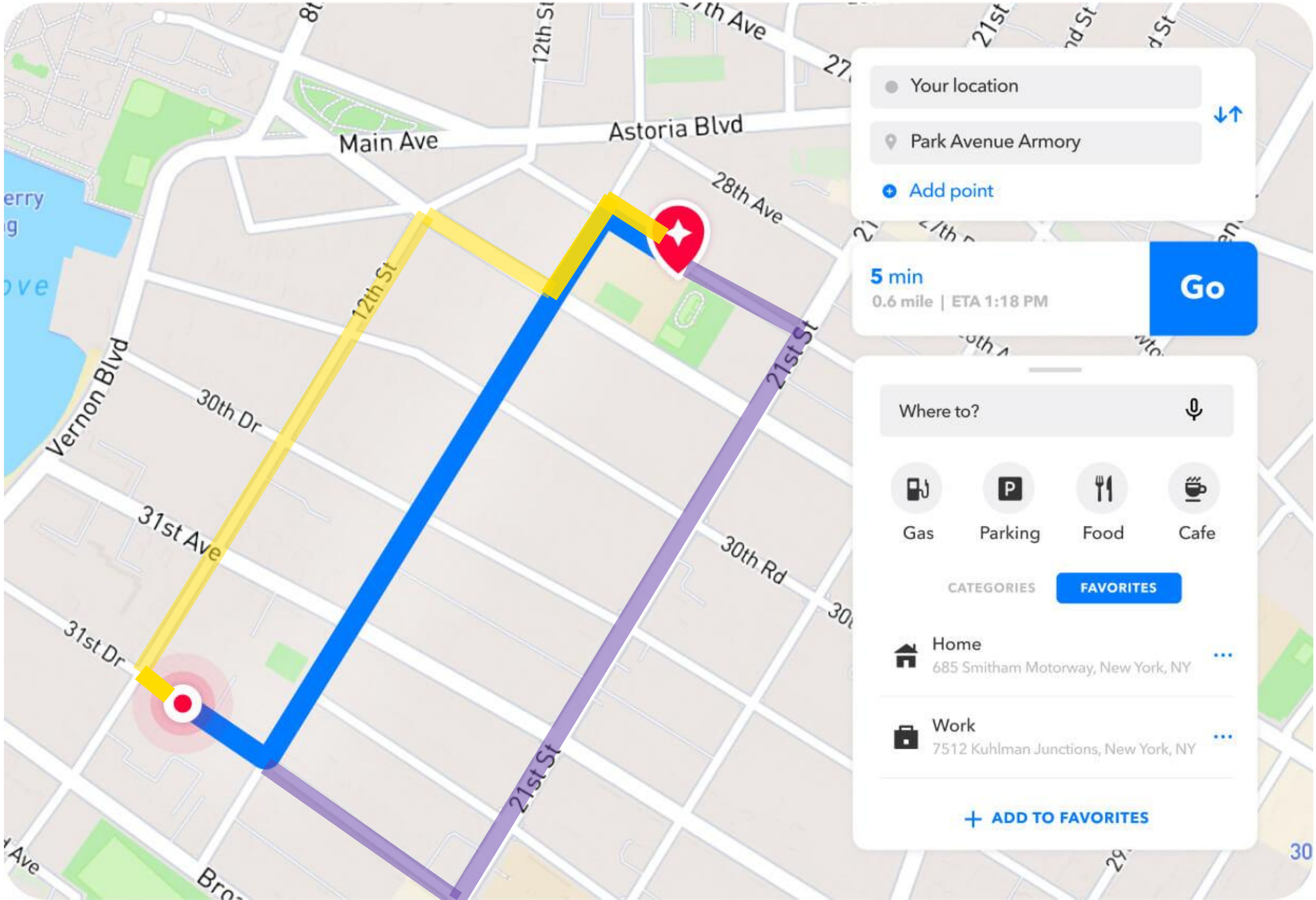
Path 2:
Cost $20, takes 15 mins

Path 3:
Cost $20, takes 20 mins, but you can pick up your friend living in 21st St.

Which one is the best path?



Search


Your location
Park Avenue Armory
Add point
5 min
0.6 mile | ETA 1:18 PM
Go
Where to?
Gas
Parking
Food
Cafe
CATEGORIES
FAVORITES
Home
685 Smitham Motorway, New York, NY
Work
7512 Kuhlman Junctions, New York, NY
ADD TO FAVORITES
Main Ave
Astoria Blvd
28th Ave
12th St
30th Dr
31st Ave
31st Dr
Vernon Blvd
30th Rd
21st St

What is Search and why it is needed

> *In which we see how an agent can find a sequence of actions that achieves its goals when no single action will do.*
>
> - Peter Norvig and Stuart J. Russel
> Artificial Intelligence: A Modern Approach

When is Search Needed?

Motion Planning

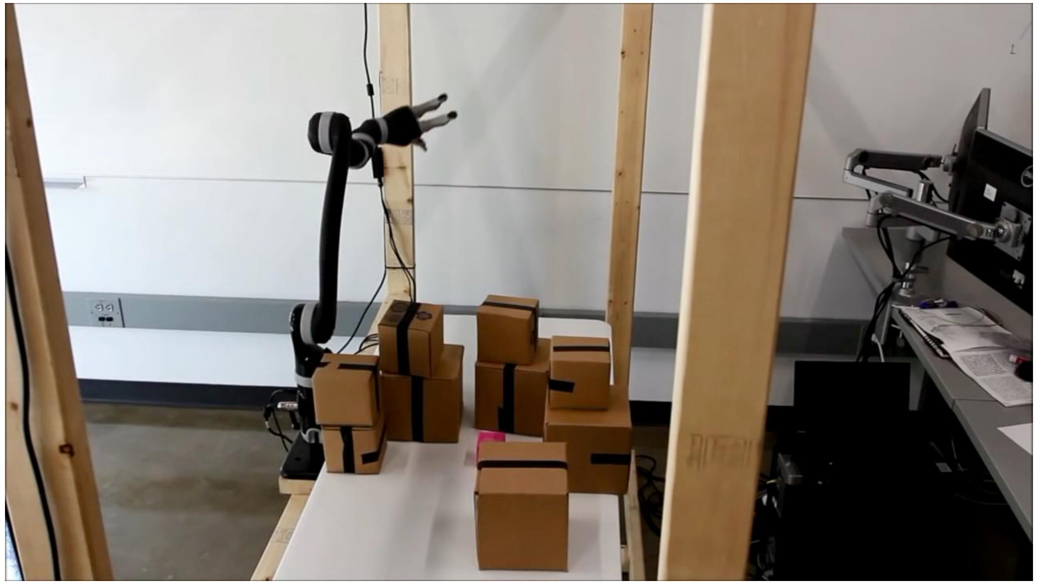
Navigation

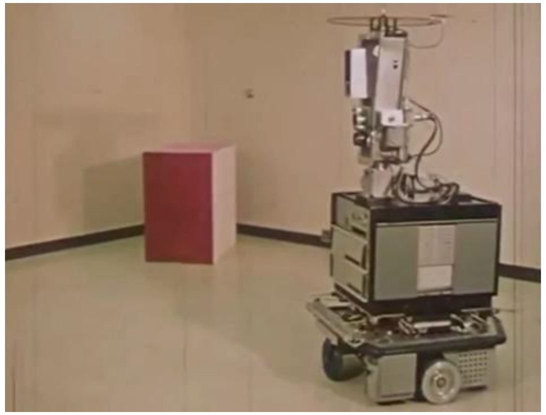
Speech and Natural Language

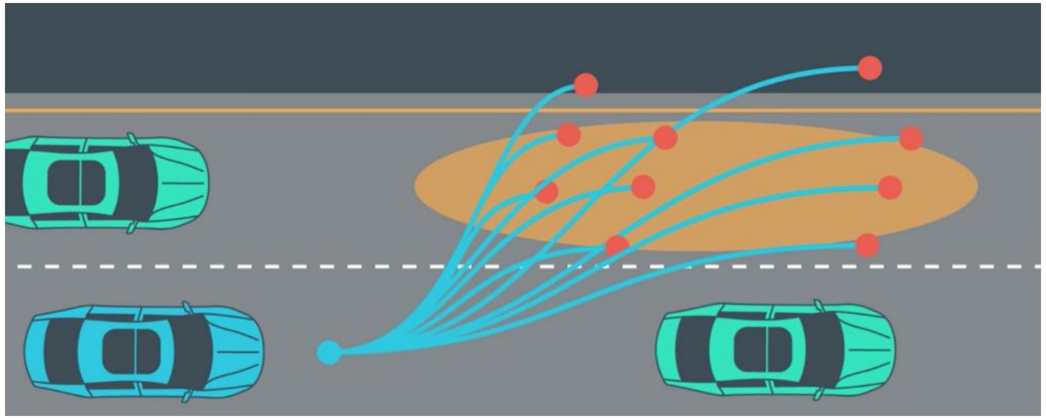
Task Planning

Machine Learning

Game Playing





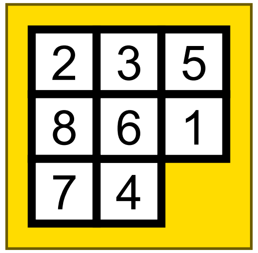


Converting real-world problem into machine understandable problem

Let’s formalize it!

A problem can be described by

- state (S)



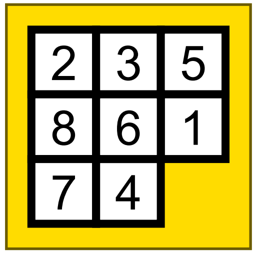

```
[[2,3,5],
 [8,6,1],
 [7,4,0]]
```

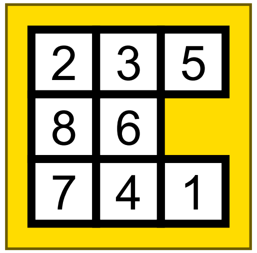
- action (A) {“up”, “left”}

- Transition Function do: $S \times A \rightarrow S$

Let’s formalize it

Transition Function

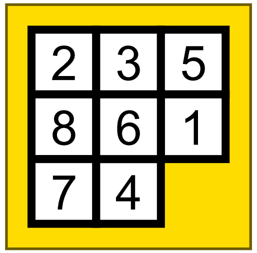
do (

, “down, 1”) → 

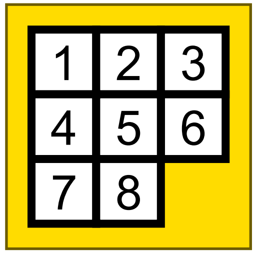
Let's formalize it

Special states:

- Initial state: s_0

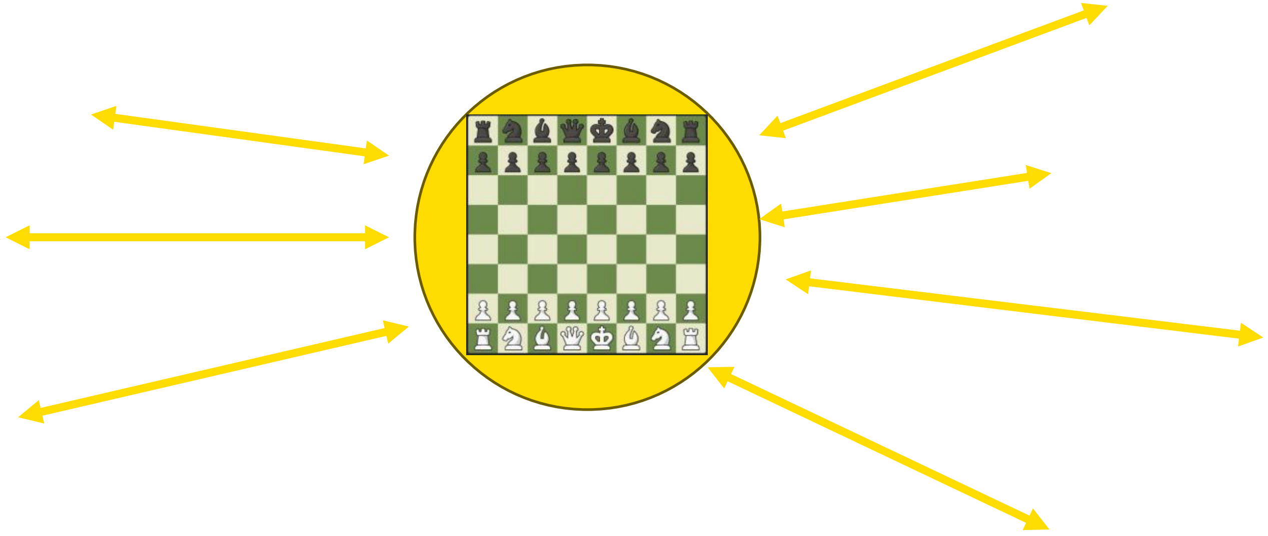


- Terminal states: $Z \subset S$



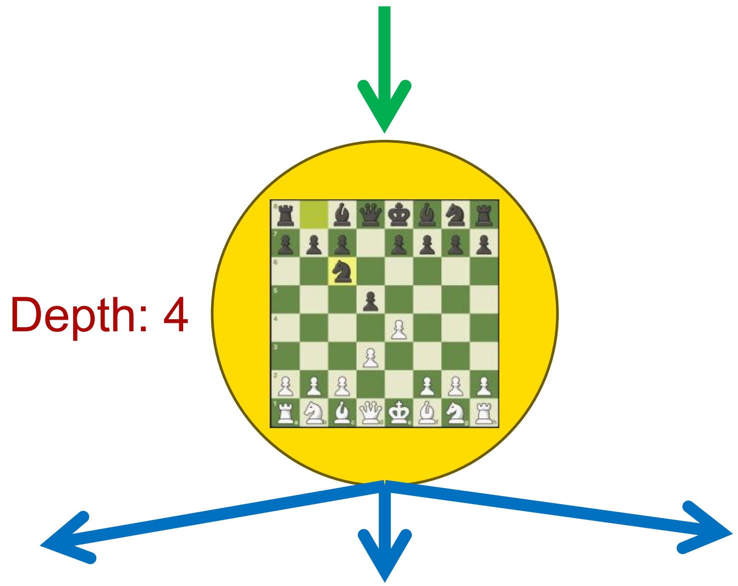
State Space in Graph

- Node in Graph is a data structure which contains a state and other related information such as legal actions from the state



State Space in Tree

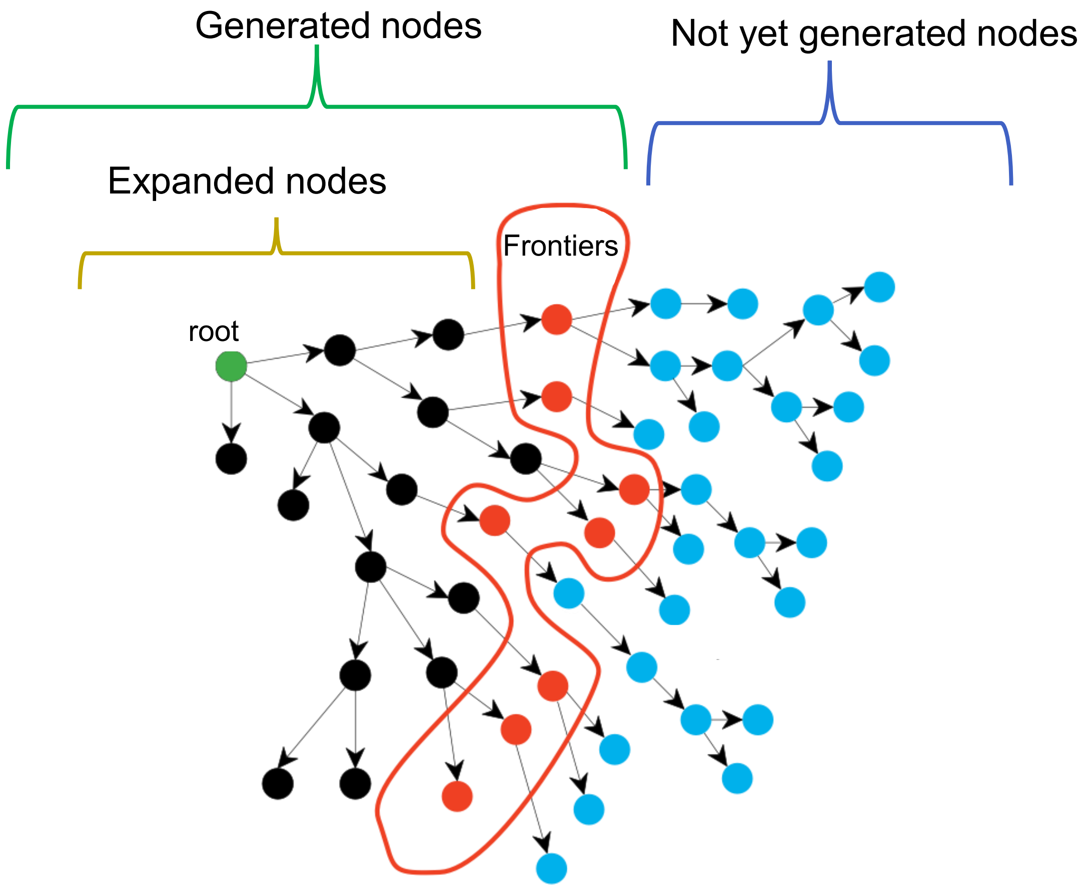
- Node in Tree is a data structure which contains a state and other related information such as children nodes, parent node and depth



	Search Algorithms Covered at this Lecture
Uninformed Search	Breath First Search (BFS)
	Depth First Search (DFS)
	Iterative Deepening Search (IDDFS)
	Uniform Cost Search (UCS)
Informed Searches	Greedy Best-First Search
	A* Search

Problem Solving by Graph Searching

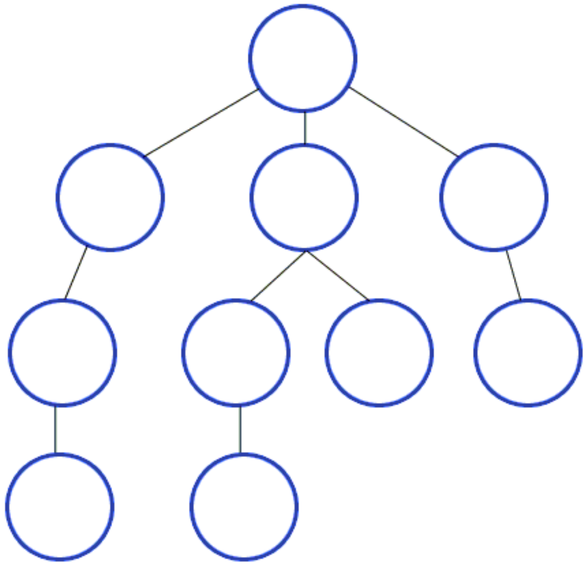
- **Expanded nodes**: green and black
- **Frontiers**: red
- **Generated:** green, black and red
- **Not yet generated nodes:** blue



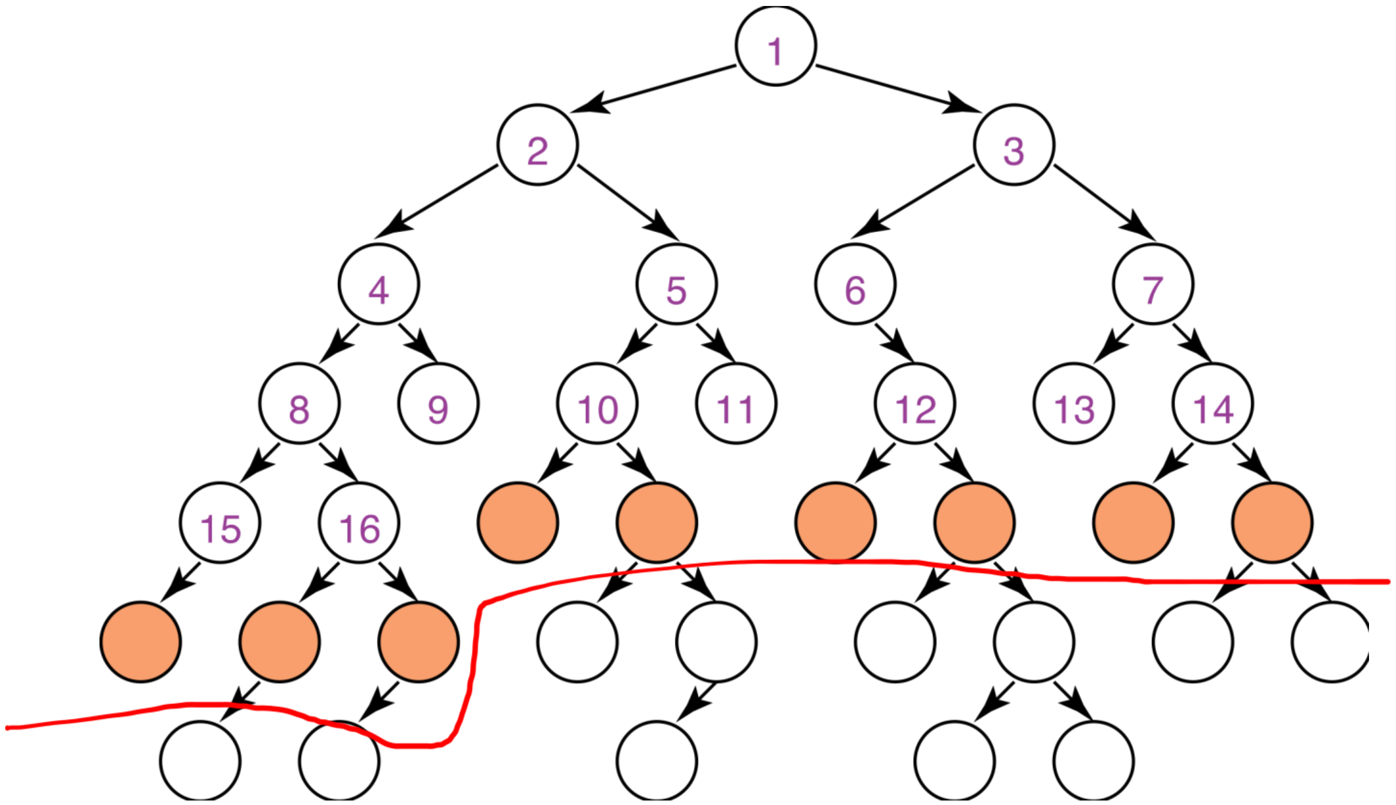
Search strategy differ in the way they expand the frontier

Uninformed Search	Search Algorithms Covered at this Lecture
	Breath First Search (BFS)
	Depth First Search (DFS)
	Iterative Deepening Search (IDDFS)
	Uniform Cost Search (UCS)
Informed Searches	Greedy Best-First Search
	A* Search

Breath First Search (BFS)



Breadth-first Search Frontier

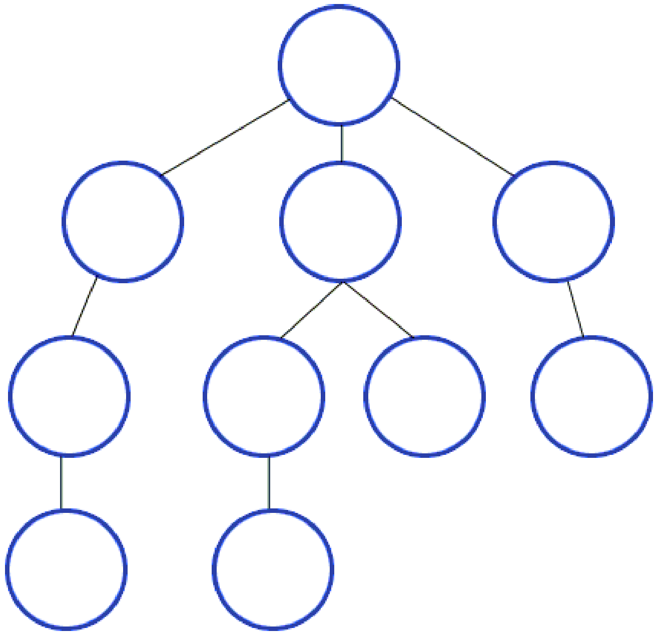


Breadth-First Search

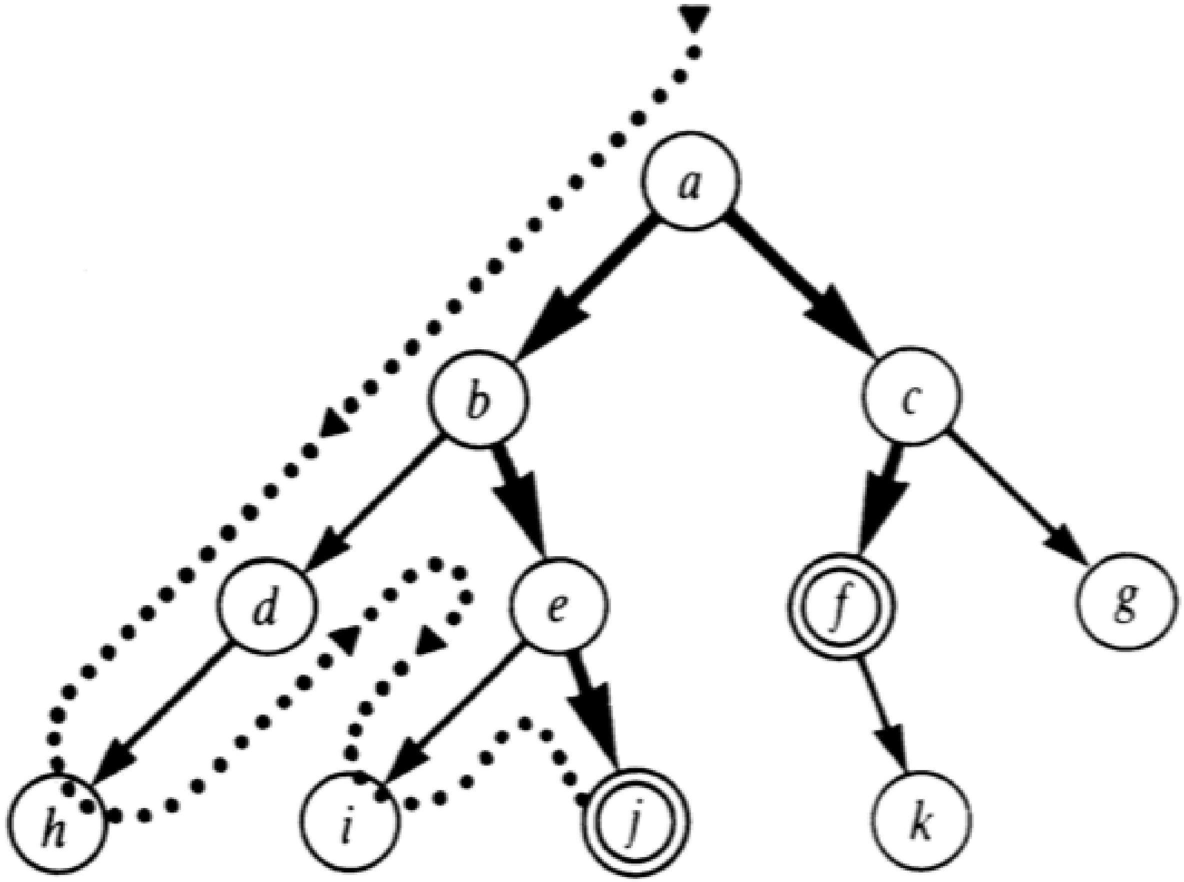
- ✓ All nodes are expanded at a same depth in the tree before any nodes at the next level are expanded
- ✓ Can be implemented by using a queue to store frontier node
  - ✓put newly generated successors at end of queue
- ✓ Finds the shallowest path first

Uninformed Search	Search Algorithms Covered at this Lecture
	Breath First Search (BFS)
	Depth First Search (DFS)
	Iterative Deepening Search (IDDFS)
	Uniform Cost Search (UCS)
Informed Searches	Greedy Best-First Search
	A* Search

Depth First Search (DFS)



Depth-first Search - DFS



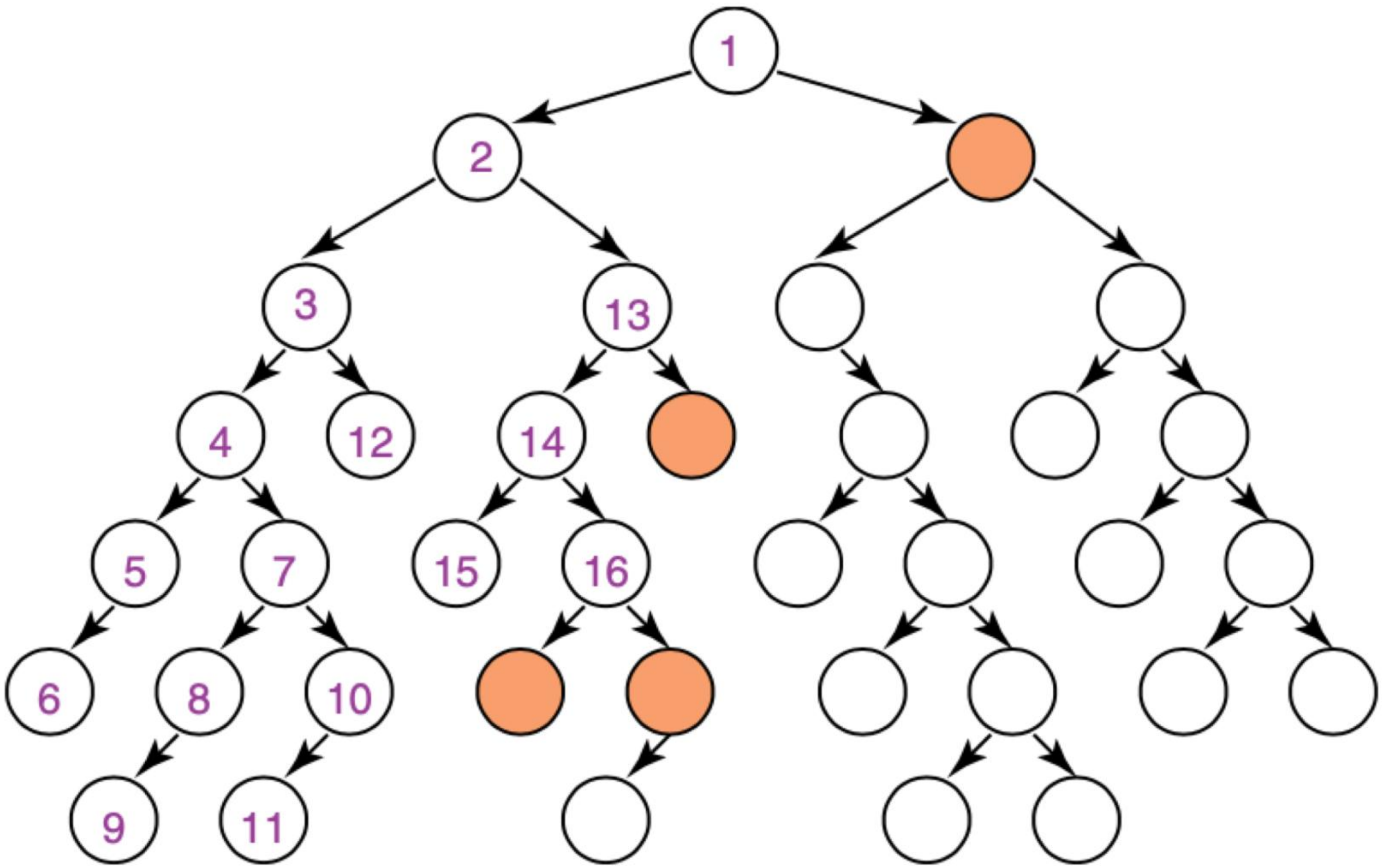
Depth First Search

Expand one node at the deepest level reached so far

Implementation:

- Implement the frontier as a stack, i.e. insert newly generated states at the front of the open list (frontier)
- Can be implemented by recursive function calls

Depth-first Search Example

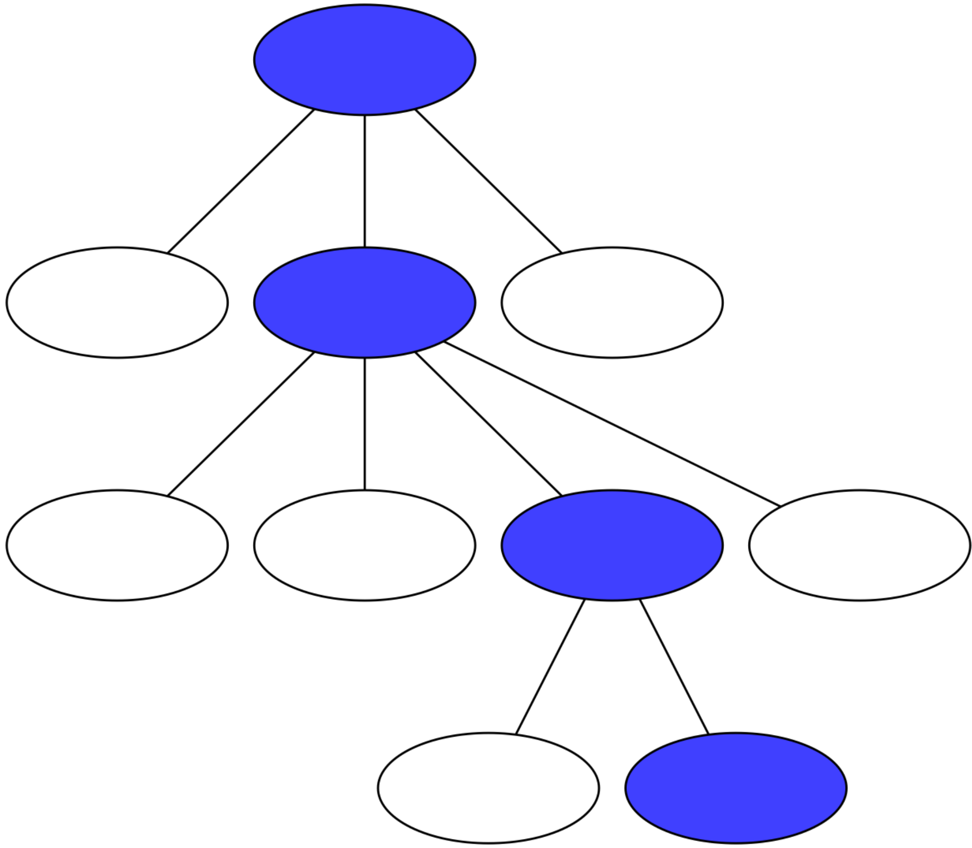


Depth First Search

- At any point depth-first search stores single path from root to leaf.

- Stop when node with goal state is expanded

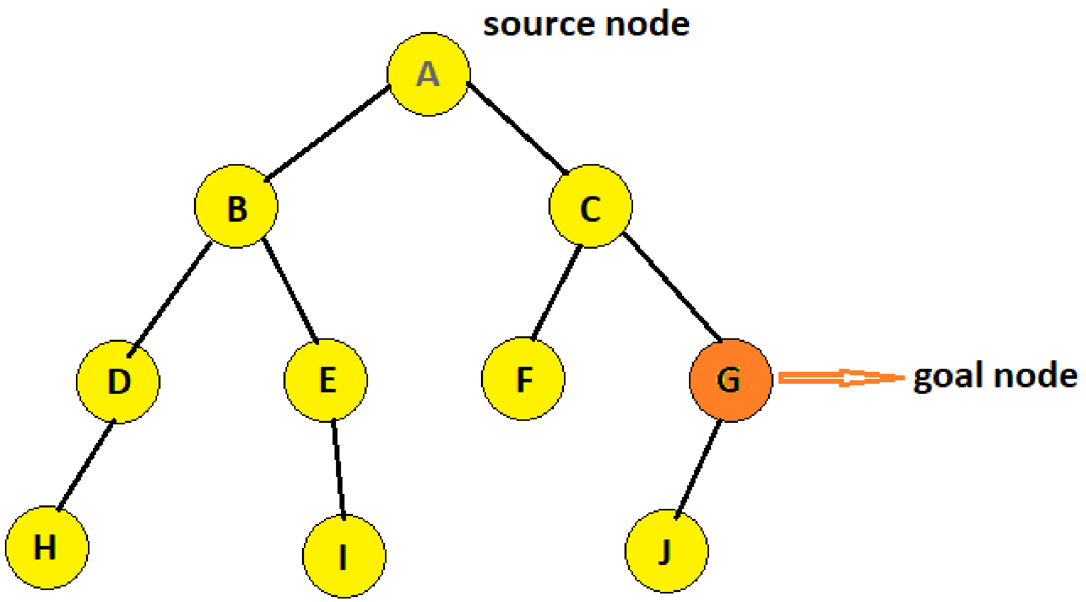
- Include check that state has not already been explored **along a path – cycle checking**



	Search Algorithms Covered at this Lecture
Uninformed Search	Breath First Search (BFS)
	Depth First Search (DFS)
	Iterative Deepening Search (IDDFS)
	Uniform Cost Search (UCS)
Informed Searches	Greedy Best-First Search
	A* Search

Iterative Deepening Depth-First Search (IDDFS)

- IDDFS calls DFS for different depths starting from an initial value. In every call, DFS is restricted from going beyond given depth.

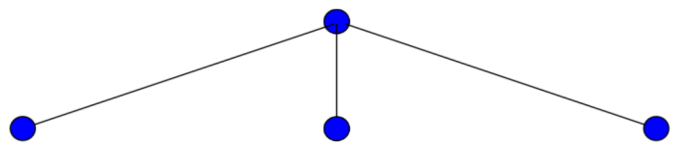


- Iteration 0: A
- Iteration 1: A -> B -> C
- Iteration 2: A -> B -> D -> E -> C -> F -> G

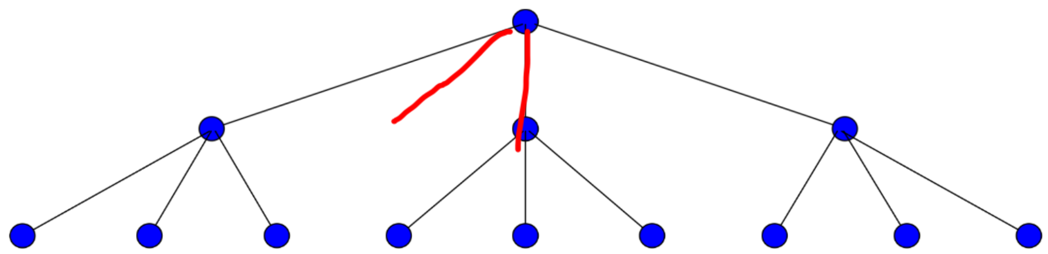
Iterative Deepening Search

- Iterative deepening: Try all possible depth bounds in turn.
- Combines depth-first and breadth-first search.
- Does a series of depth-limited depth-first searches to depth 1, 2, 3, etc.
- Early states will be expanded multiple times, but that might not matter too much because most of the nodes are near the leaves.

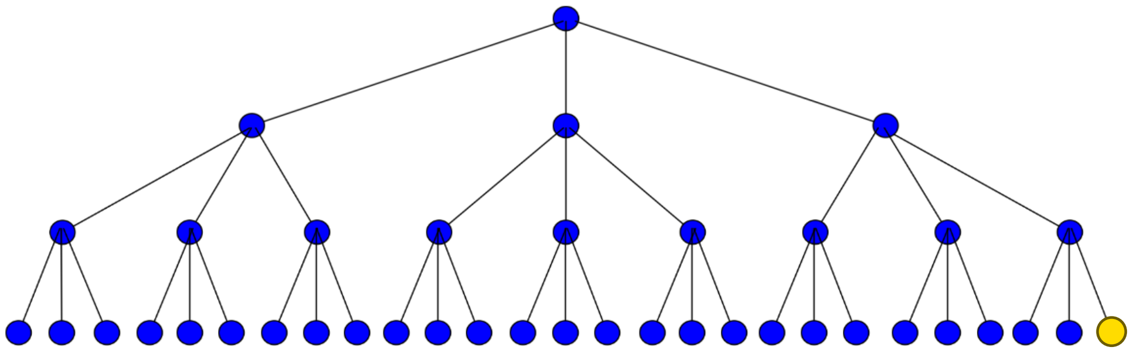
Iterative Deepening Search



Iterative Deepening Search



Iterative Deepening Search



	Search Algorithms Covered at this Lecture
Uninformed Search	Breath First Search (BFS)
	Depth First Search (DFS)
	Iterative Deepening Search (IDDFS)
	Uniform Cost Search (UCS)
Informed Searches	Greedy Best-First Search
	A* Search

Uniform-Cost Search, $f(n) = g(n)$

Sometimes transitions from one node to another have a cost

Cost of a **path** is the sum of the costs of its arcs:

$$cost(\langle n_0, \cdots, n_k \rangle) = \sum_{i=1}^{k} cost(\langle n_{i-1}, n_i \rangle)$$

An optimal solution has minimum cost

Delivery robot example:

- cost of arc may be resources (e.g., time, energy) required to execute action represented by the arc
- aim is to reach goal using least resources

Uniform-Cost Search, $f(n) = g(n)$

Expand root first, then expand **least-cost** unexpanded node

- implemented by treating the frontier as a priority queue ordered by the cost function

$$cost(\langle n_0, \cdots, n_k \rangle) = \sum_{i=1}^{k} cost(\langle n_{i-1}, n_i \rangle)$$

Uniform-Cost Search, $f(n) = g(n)$

- Reduces to breadth-first search when all actions have same cost
- Finds the cheapest goal provided path cost is monotonically increasing along each path (i.e. no negative-cost steps)

$\boldsymbol{f(n) = g(n)}$

$f(n)$: the estimated total cost through node n

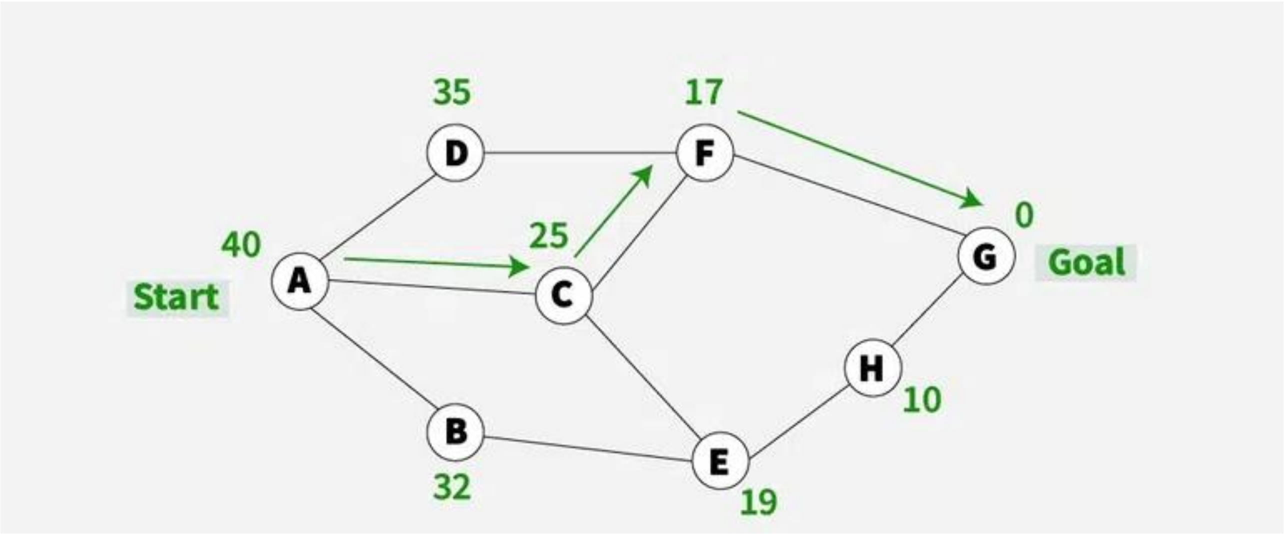
$g(n)$: the cost of the path from the current node to the next node n.

Uninformed Search	Search Algorithms Covered at this Lecture
	Breath First Search (BFS)
	Depth First Search (DFS)
	Iterative Deepening Search (IDDFS)
	Uniform Cost Search (UCS)
Informed Searches	**Greedy Best-First Search**
	A* Search

Uniformed vs Informed Search

- Uninformed - keeps searching until it stumbles on goal
  - No domain knowledge
- Informed - searches (**a.k.a heuristics**) in direction of best guess to goal
  - Uses domain knowledge

Greedy Best-First Search, $f(n) = h(n)$



Greedy Best-First Search, $f(n) = h(n)$

Always select node closest to goal according to heuristic function. $h(n)$ is the estimated cost for a current node to goal.

- $h(n) = 0$ if n is a goal state

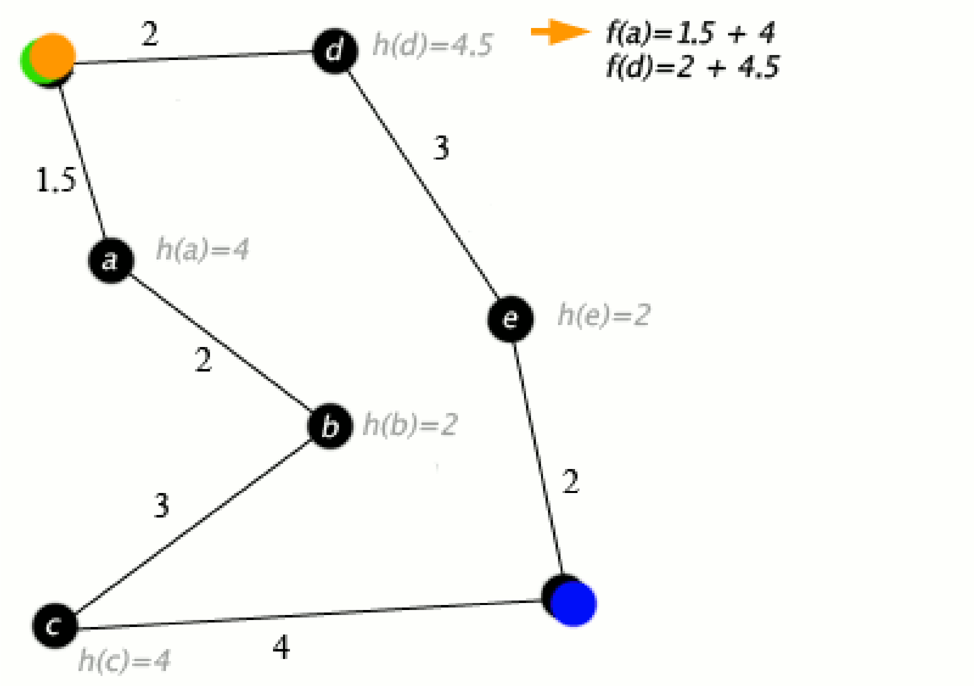
Frontier is a **priority queue** ordered by h.

"Greedy" algorithm takes "best" node first.

$\boldsymbol{f(n) = h(n)}$
$f(n)$: the estimated total cost through node n
$h(n)$: the **heuristic** estimate of the cost from n to the goal.

	Search Algorithms Covered at this Lecture
Uninformed Search	Breath First Search (BFS)
	Depth First Search (DFS)
	Iterative Deepening Search (IDDFS)
	Uniform Cost Search (UCS)
Informed Searches	Greedy Best-First Search
	A* Search

A* Search, $f(n) = g(n) + h(n)$




Uniform-Cost Search
Greedy Search
A* Search

A* Search, $f(n) = g(n) + h(n)$

Use both cost of path generated and estimate to goal to order nodes on the frontier

$\boldsymbol{f(n) = g(n) + h(n)}$
$f(n)$: the estimated total cost through node n
$g(n)$: the cost of the path from the start node to the current node n.
$h(n)$: the heuristic estimate of the cost from n to the goal.

Order priority queue using function $f(n) = g(n) + h(n)$

Summary of Informed Search

- Informed search makes use of problem-specific knowledge to guide progress of search
- This can lead to a significant improvement in performance
- Greedy Search tries to minimise cost from current node *n* to the goal.
- A* combines the advantages of Uniform-Cost Search and Greedy Search