Search 1

Shivaram Kalyanakrishnan

Department of Computer Science and Engineering
Indian Institute of Technology Bombay

February 2023

Navigation System

How to go from IIT Bombay to Marine Drive?

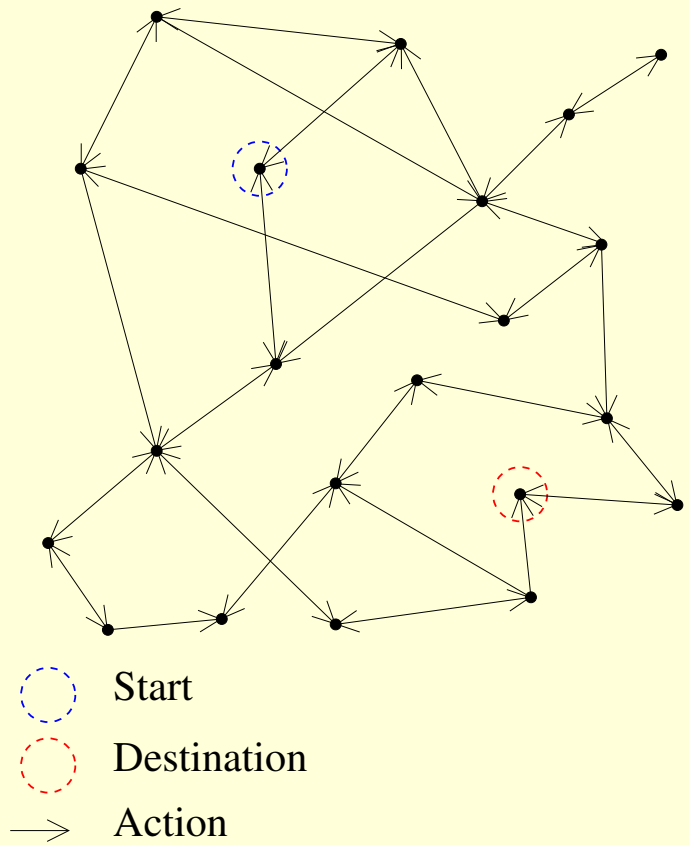
[1]

[1] https://www.flickr.com/photos/nat507/16088993607. CC image courtesy of Nathan Hughes Hamilton on Flickr licensed under CC BY 2.0.

Navigation System

How to go from IIT Bombay to Marine Drive?

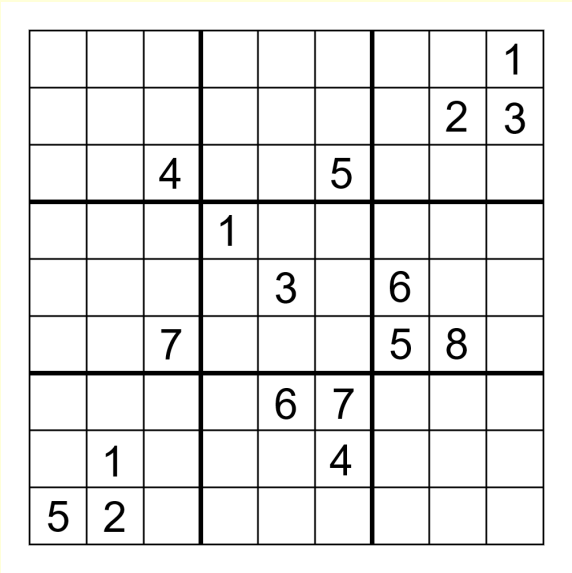
[1]



[1] https://www.flickr.com/photos/nat507/16088993607. CC image courtesy of Nathan Hughes Hamilton on Flickr licensed under CC BY 2.0.

Some Popular Puzzles

How to solve?



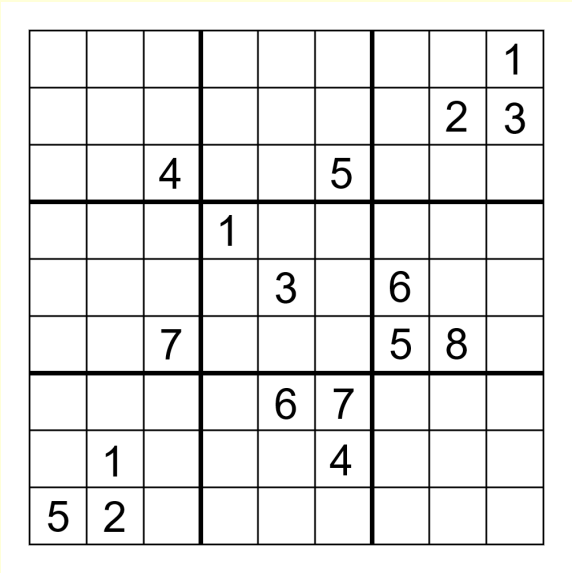
								1
							2	3
		4			5			
			1					
				3		6		
		7				5	8	
				6	7			
	1				4			
5	2							

Sudoku [1]

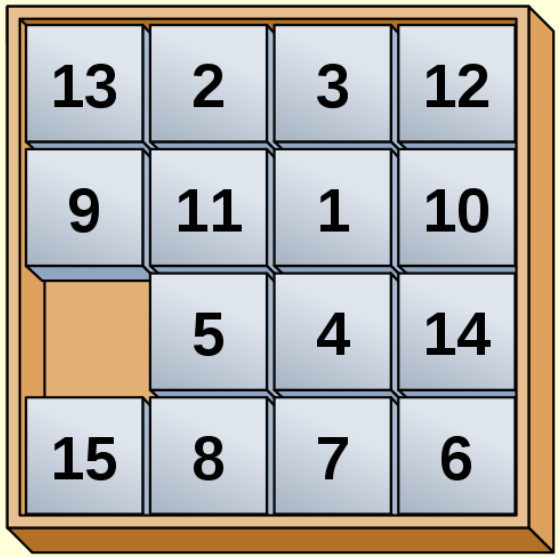
[1] `https://upload.wikimedia.org/wikipedia/commons/e/eb/Sudoku_Puzzle_%28a_symmetrical_puzzle_with_17_clues%29.png`. CC image courtesy of LithiumFlash on WikiCommons licensed under CC-BY-SA-4.0.

Some Popular Puzzles

How to solve?



Sudoku [1]



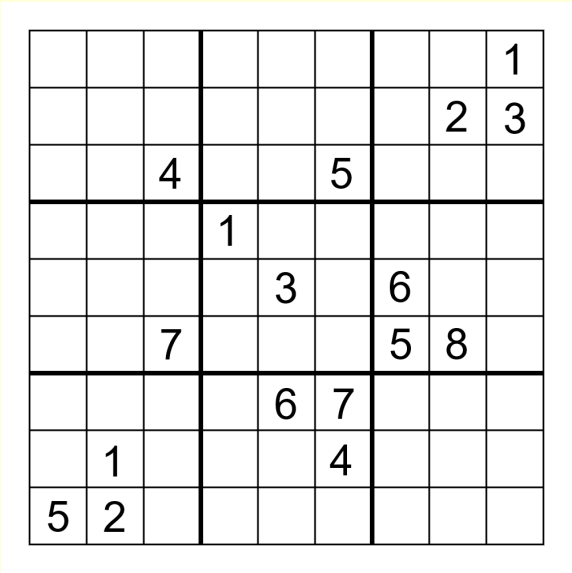
15-puzzle [2]

[1] https://upload.wikimedia.org/wikipedia/commons/e/eb/Sudoku_Puzzle_%28a_symmetrical_puzzle_with_17_clues%29.png. CC image courtesy of LithiumFlash on WikiCommons licensed under CC-BY-SA-4.0.

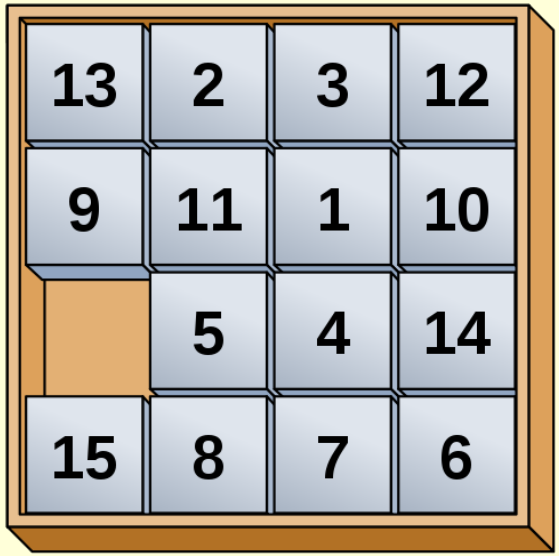
[2] https://commons.wikimedia.org/wiki/File:15-puzzle-solvable.svg. CC image courtesy of Stannic on WikiMedia Commons licensed under CC-BY-SA-3.0

Some Popular Puzzles

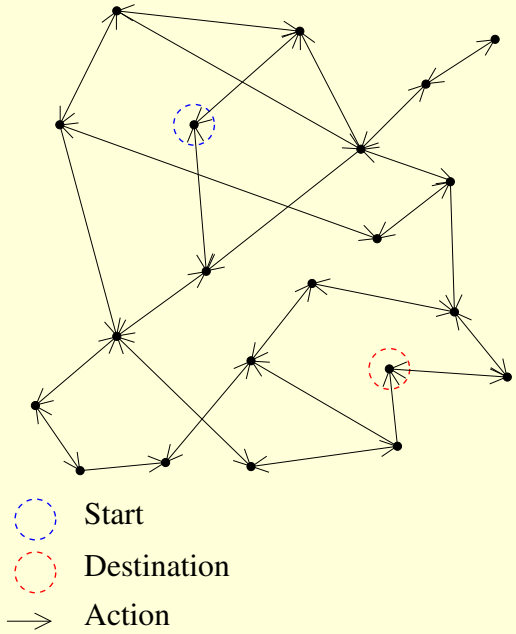
How to solve?



Sudoku [1]



15-puzzle [2]



Same abstraction?

[1] https://upload.wikimedia.org/wikipedia/commons/e/eb/Sudoku_Puzzle_%28a_symmetrical_puzzle_with_17_clues%29.png. CC image courtesy of LithiumFlash on WikiCommons licensed under CC-BY-SA-4.0.

[2] https://commons.wikimedia.org/wiki/File:15-puzzle-solvable.svg. CC image courtesy of Stannic on WikiMedia Commons licensed under CC-BY-SA-3.0

This Lecture

- Search problem instances
- Uninformed search
  - Depth-first search
  - Breadth-first search
  - Lowest-cost-first search

This Lecture

- Search problem instances
- Uninformed search
  - Depth-first search
  - Breadth-first search
  - Lowest-cost-first search

Elements of a Search Problem Instance

Elements of a Search Problem Instance

- Set of states.

Elements of a Search Problem Instance

- Set of states.
- Start state.

Elements of a Search Problem Instance

- Set of states.
- Start state.
- Set of actions available from each state.

Elements of a Search Problem Instance

- Set of states.
- Start state.
- Set of actions available from each state.
- NextState(s, a) for each state s and action a.

Elements of a Search Problem Instance

- Set of states.
- Start state.
- Set of actions available from each state.
- NextState(s, a) for each state s and action a.
- Cost(s, a) for each state s and action a. We assume Cost(s, a) ≥ 0.

Elements of a Search Problem Instance

- Set of states.
- Start state.
- Set of actions available from each state.
- NextState(s, a) for each state s and action a.
- Cost(s, a) for each state s and action a. We assume Cost(s, a) ≥ 0.
- IsGoal(s) for each state s.

Elements of a Search Problem Instance

- Set of states.
- Start state.
- Set of actions available from each state.
- NextState(s, a) for each state s and action a.
- Cost(s, a) for each state s and action a. We assume Cost(s, a) ≥ 0.
- IsGoal(s) for each state s.

Expected output: a sequence of actions, which when applied from start state:

- reaches a goal state, and
- (optionally) has minimum path-cost.

Elements of a Search Problem Instance

- Set of states.
- Start state.
- Set of actions available from each state.
- NextState(s, a) for each state s and action a.
- Cost(s, a) for each state s and action a. We assume Cost(s, a) ≥ 0.
- IsGoal(s) for each state s.

Expected output: a sequence of actions, which when applied from start state:

- reaches a goal state, and
- (optionally) has minimum path-cost.

Note: Sometimes there might be no solution!

Elements of a Search Problem Instance

- Set of states.
- Start state.
- Set of actions available from each state.
- NextState(s, a) for each state s and action a.
- Cost(s, a) for each state s and action a. We assume Cost(s, a) ≥ 0.
- IsGoal(s) for each state s.

Expected output: a sequence of actions, which when applied from start state:

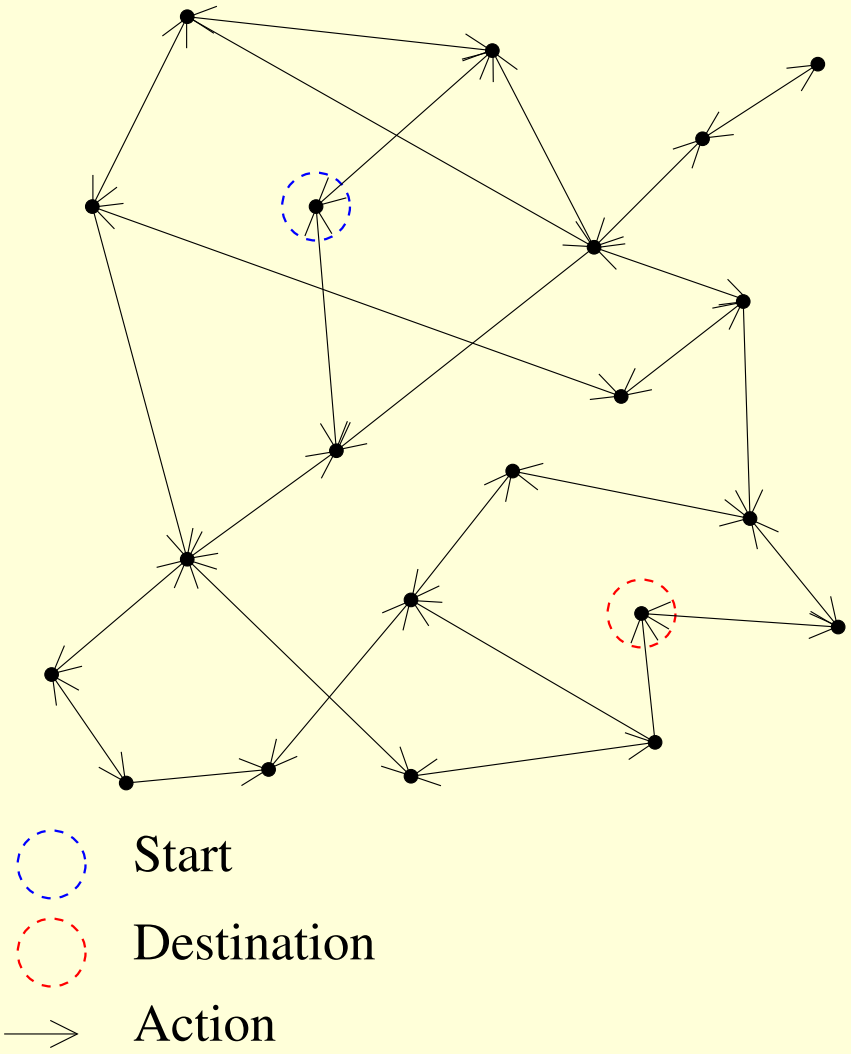
- reaches a goal state, and
- (optionally) has minimum path-cost.

Note: Sometimes there might be no solution!

Number of available actions in each state is called the branching factor b.

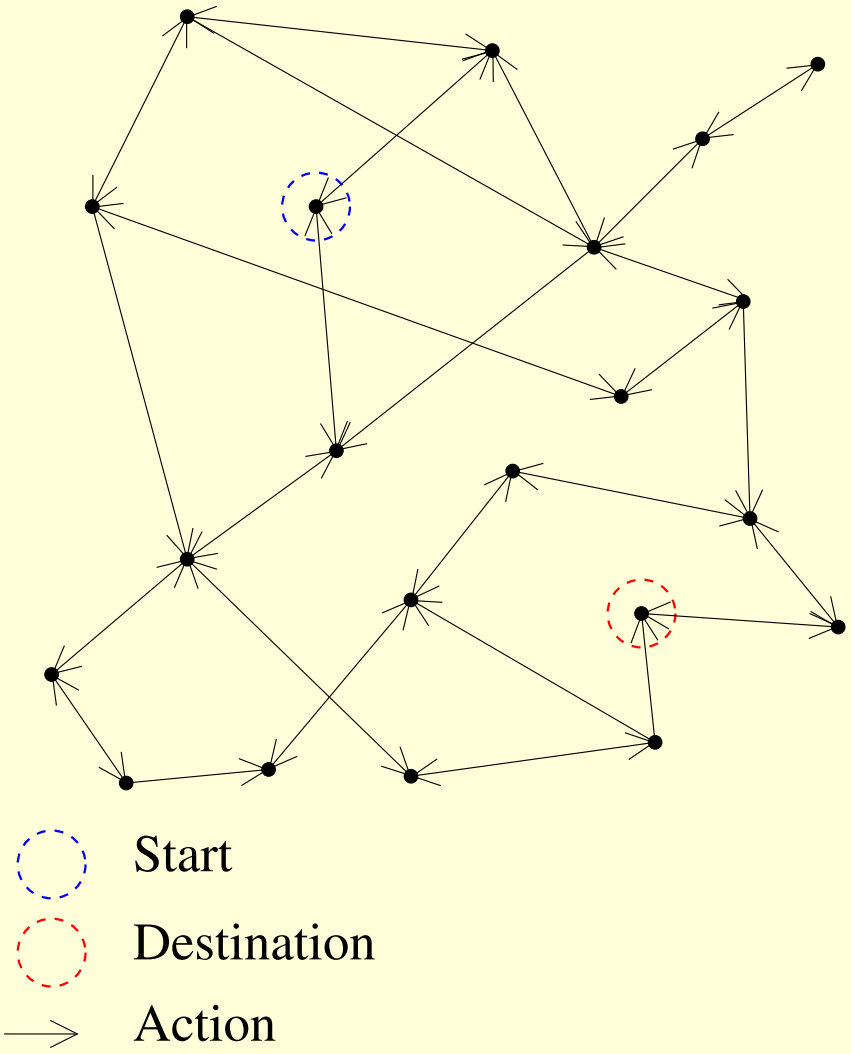
Minimum number of actions (or length of optimal path) to reach goal state is called the depth d of the search instance.

Problem Formulation: Navigation System



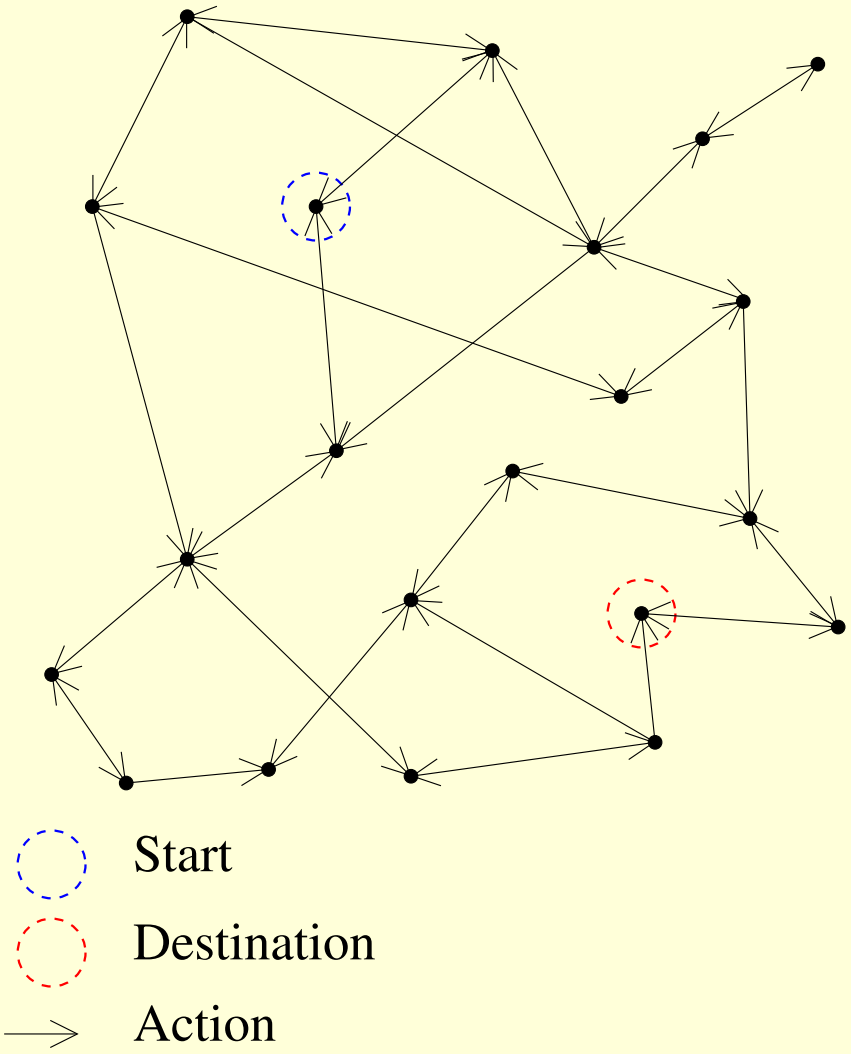
States?

Problem Formulation: Navigation System



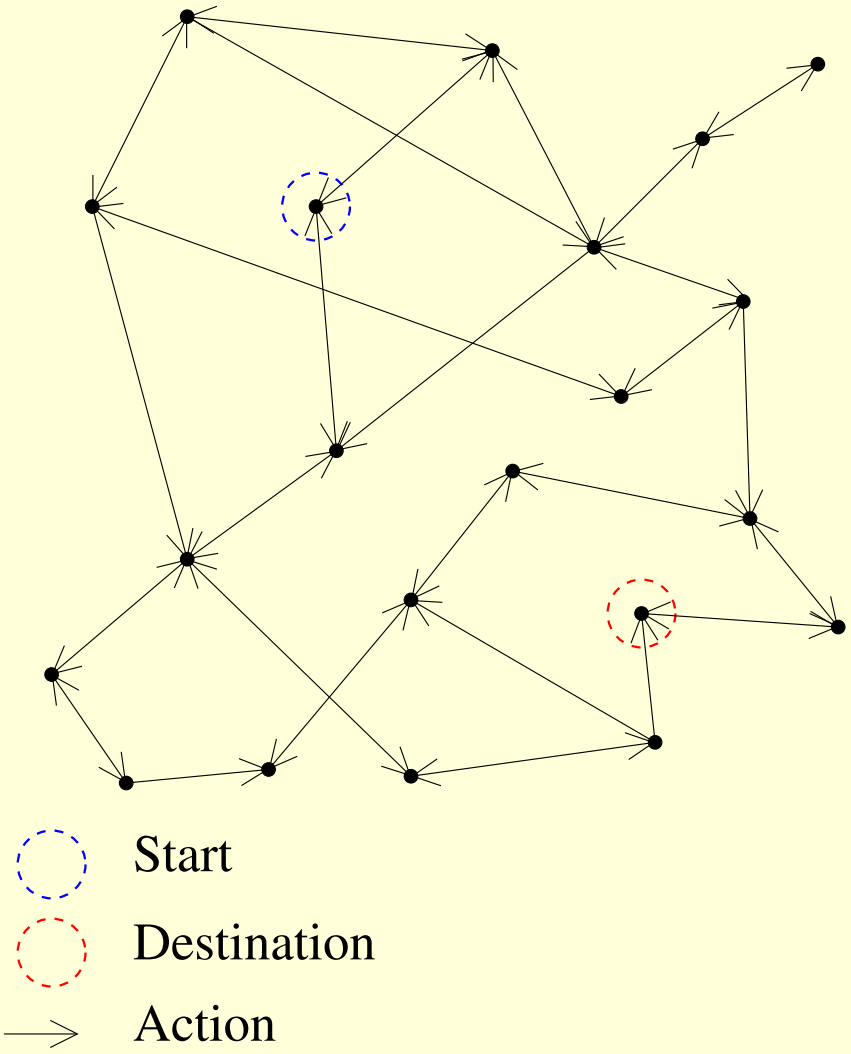
States?, start state?

Problem Formulation: Navigation System



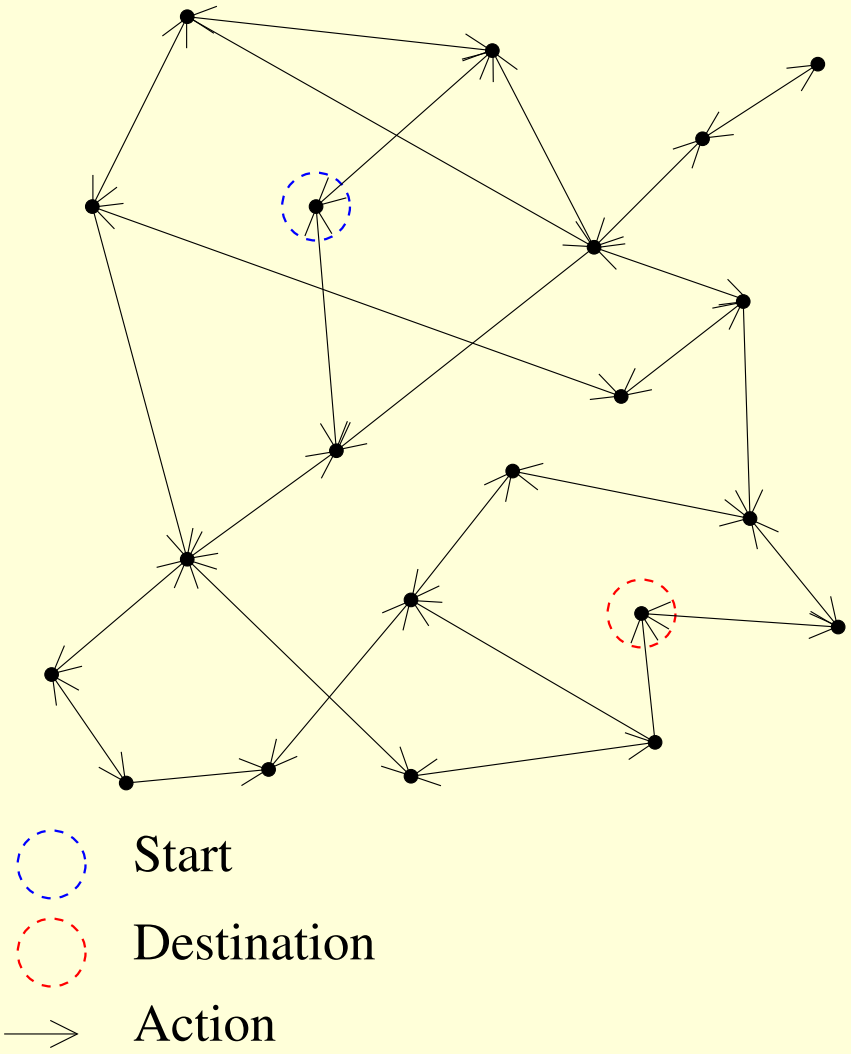
States?, start state?, actions?

Problem Formulation: Navigation System



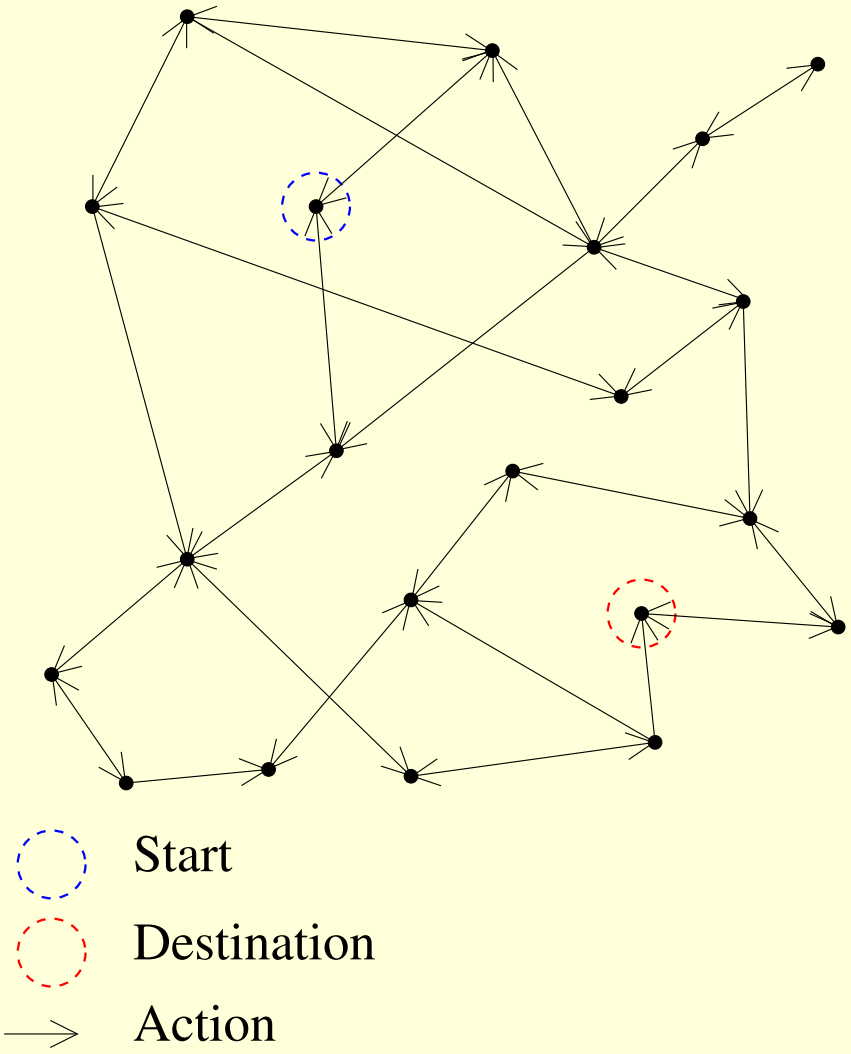
States?, start state?, actions? , NextState()?

Problem Formulation: Navigation System



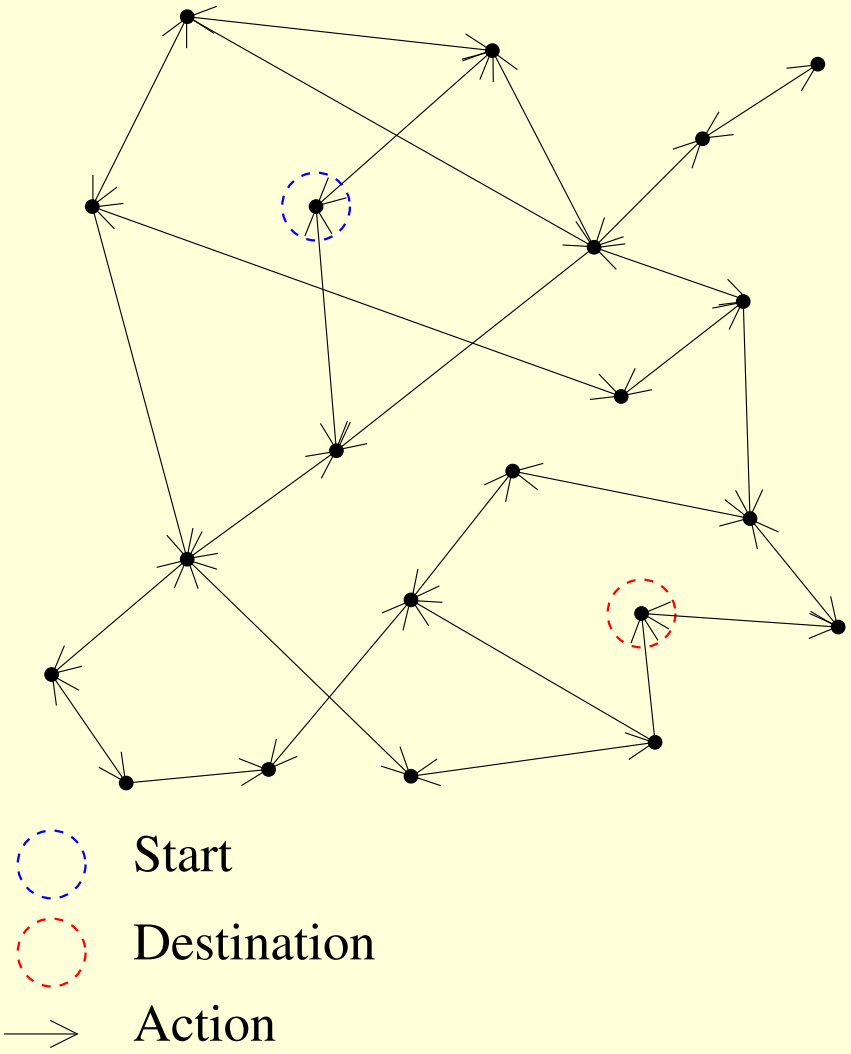
States?, start state?, actions? , NextState()?, Cost()?

Problem Formulation: Navigation System



States?, start state?, actions? , NextState()?, Cost()? , IsGoal()?

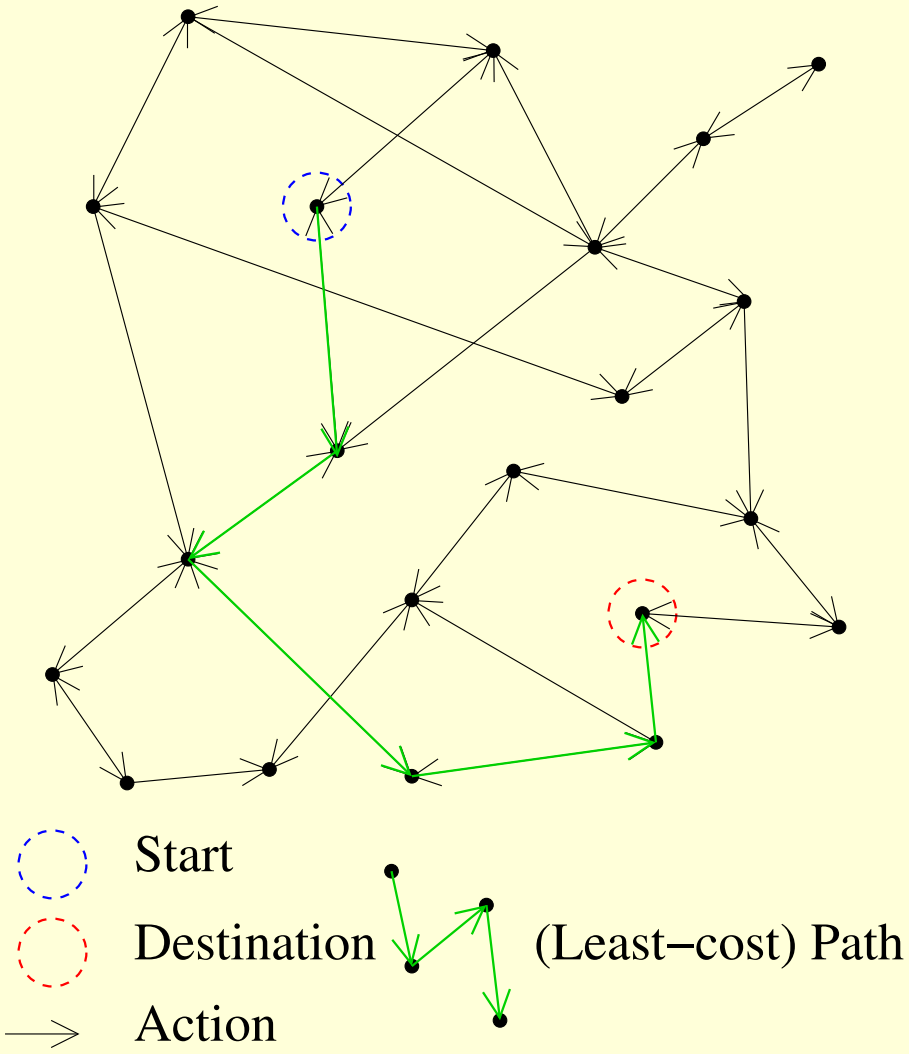
Problem Formulation: Navigation System



States?, start state?, actions? , NextState()?, Cost()? , IsGoal()?

A solver needs to find the least-cost path.

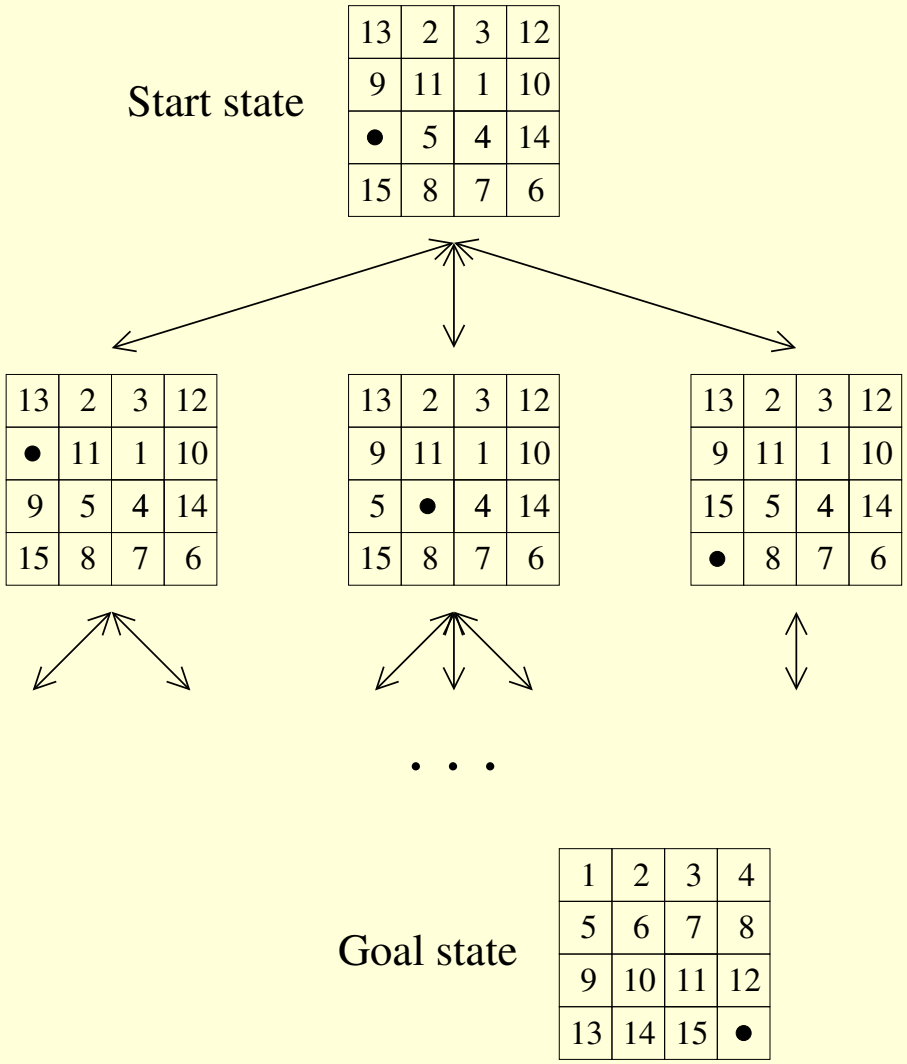
Problem Formulation: Navigation System



States?, start state?, actions? , NextState()?, Cost()? , IsGoal()?

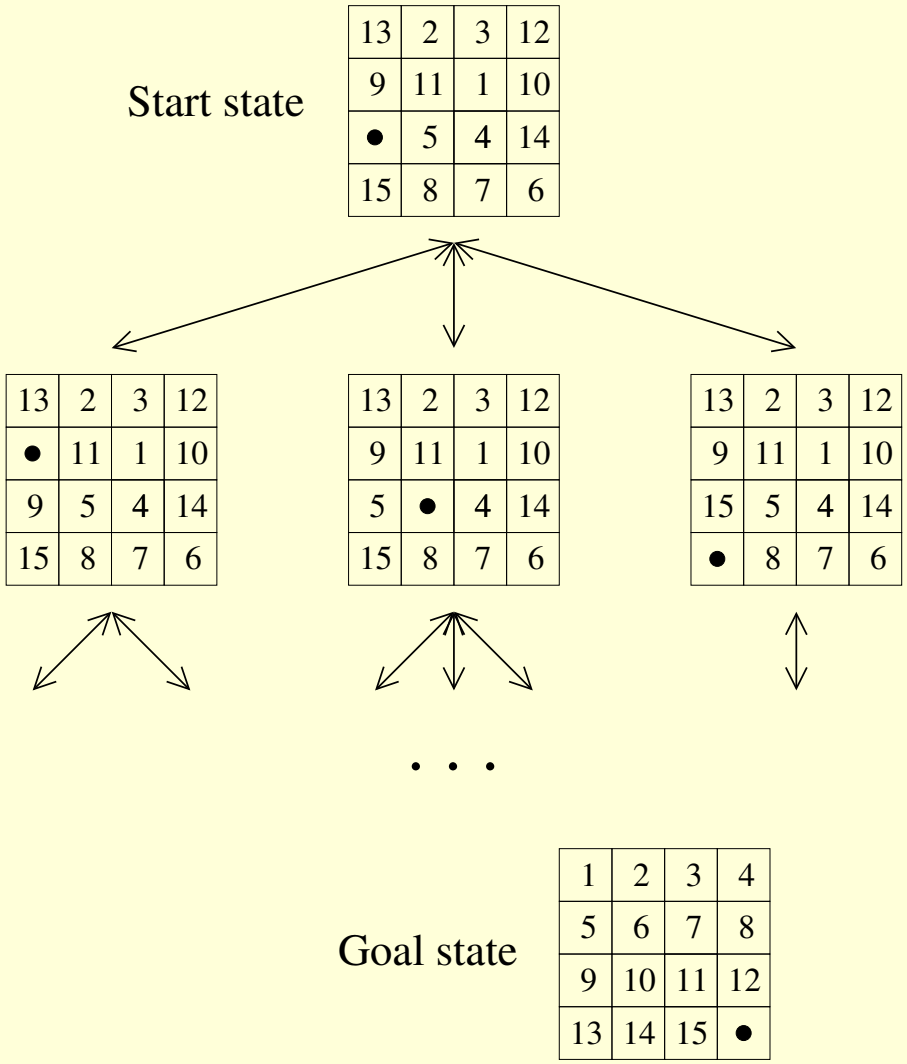
A solver needs to find the least-cost path.

Problem Formulation: 15 Puzzle



States?, start state?, actions?, NextState()?, Cost()?, IsGoal()?

Problem Formulation: 15 Puzzle



States?, start state?, actions?, NextState()?, Cost()?, IsGoal()?

A solver needs to find the shortest path to goal state.

Questions

- What are some other applications of search?

Questions

- What are some other applications of search?

  Video games (movement of characters), robotic path planning, constraint satisfaction problems/integer programming, theorem-proving, logistics.

Questions

- What are some other applications of search?

  Video games (movement of characters), robotic path planning, constraint satisfaction problems/integer programming, theorem-proving, logistics.

- Is search a type of machine learning?

Questions

- What are some other applications of search?

  Video games (movement of characters), robotic path planning, constraint satisfaction problems/integer programming, theorem-proving, logistics.

- Is search a type of machine learning?

  No! Search is an abstraction of problem solving. But note that learning is sometimes used in search.

Questions

- What are some other applications of search?

  Video games (movement of characters), robotic path planning, constraint satisfaction problems/integer programming, theorem-proving, logistics.

- Is search a type of machine learning?

  No! Search is an abstraction of problem solving. But note that learning is sometimes used in search.

- What is the main technical challenge in search?

Questions

- What are some other applications of search?

  Video games (movement of characters), robotic path planning, constraint satisfaction problems/integer programming, theorem-proving, logistics.

- Is search a type of machine learning?

  No! Search is an abstraction of problem solving. But note that learning is sometimes used in search.

- What is the main technical challenge in search?

  The large (exponentially growing) number of states in most practical tasks.

This Lecture

- Search problem instances
- Uninformed search
  - Depth-first search
  - Breadth-first search
  - Lowest-cost-first search

Generic Search Template: Pseudocode

- Primary data element is a Node, which a tuple of the form

$$(state, pathFromStartState, pathCost).$$

Generic Search Template: Pseudocode

- Primary data element is a Node, which a tuple of the form

$$(state, pathFromStartState, pathCost).$$

- At every stage of the search,
  - some states have been explored
  - some states remain unexplored, and
  - The *Frontier* is a set of nodes due for imminent expansion.

Generic Search Template: Pseudocode

- Primary data element is a Node, which a tuple of the form

$$(state, pathFromStartState, pathCost).$$

- At every stage of the search,
  - some states have been explored
  - some states remain unexplored, and
  - The *Frontier* is a set of nodes due for imminent expansion.

Frontier $\leftarrow \{Node(startState, (startState), 0)\}$.
Repeat for ever:
    Select a node n from *Frontier*.
    //Expand n.
    If $isGoal(n.state)$:
        Return n.
    For each action a available from $n.state$:
        $s \leftarrow NextState(n.state, a)$.
        $c \leftarrow Cost(n.state, a)$.
        $n' \leftarrow Node(s, n.path + (a, s), n.pathCost + c)$.
        Merge n' with *Frontier*.//Typically insertion; might allow deletions.

Generic Search Template: Pseudocode

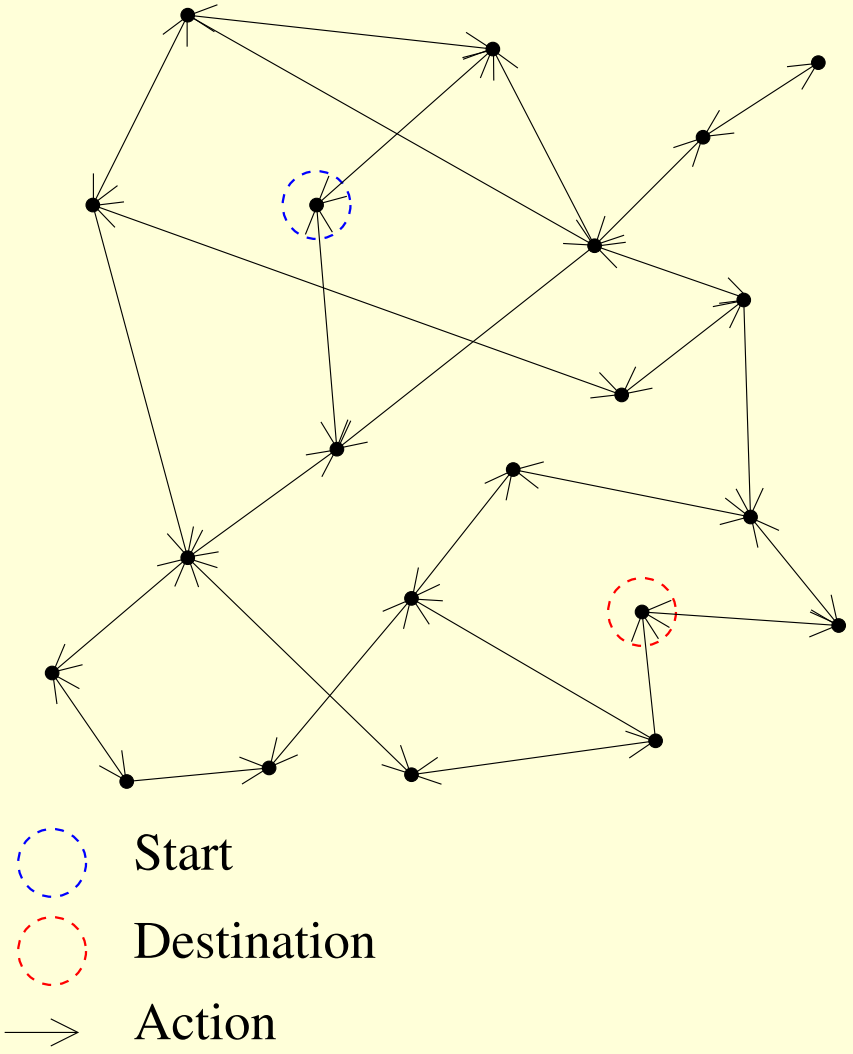
- Primary data element is a Node, which a tuple of the form

$$(state, pathFromStartState, pathCost).$$

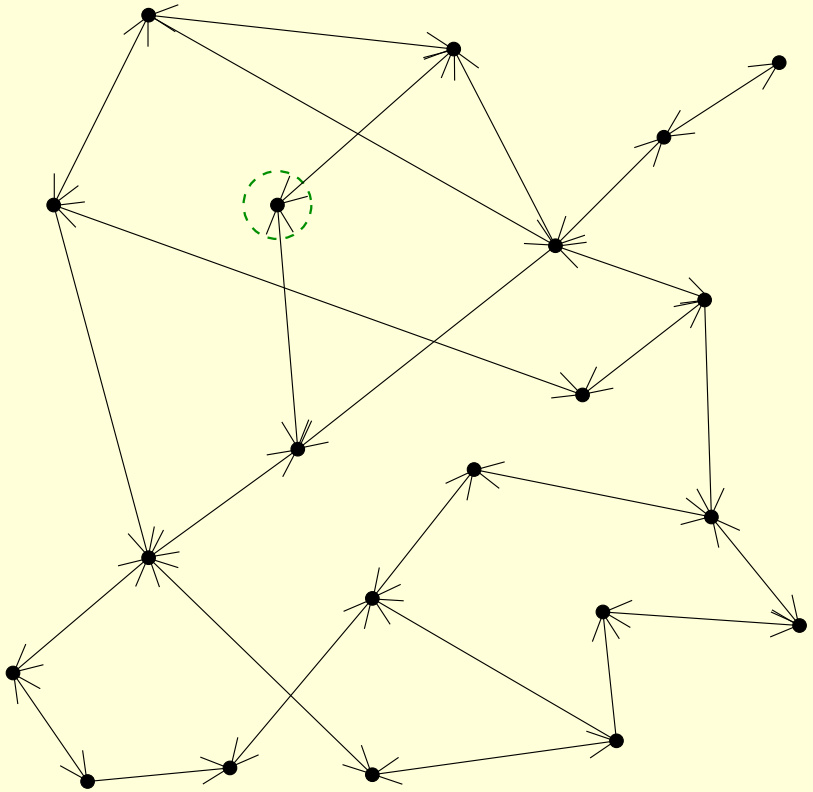
- At every stage of the search,
  - some states have been explored
  - some states remain unexplored, and
  - The *Frontier* is a set of nodes due for imminent expansion.

Frontier $\leftarrow \{Node(startState, (startState), 0)\}$.
Repeat for ever:
    Select a node n from *Frontier*.//How is this selection made?
    //Expand n.
    If $isGoal(n.state)$:
        Return n.
    For each action a available from $n.state$:
        $s \leftarrow NextState(n.state, a)$.
        $c \leftarrow Cost(n.state, a)$.
        $n' \leftarrow Node(s, n.path + (a, s), n.pathCost + c)$.
        Merge n' with *Frontier*.//Typically insertion; might allow deletions.

Generic Search Template: Illustration



Generic Search Template: Illustration

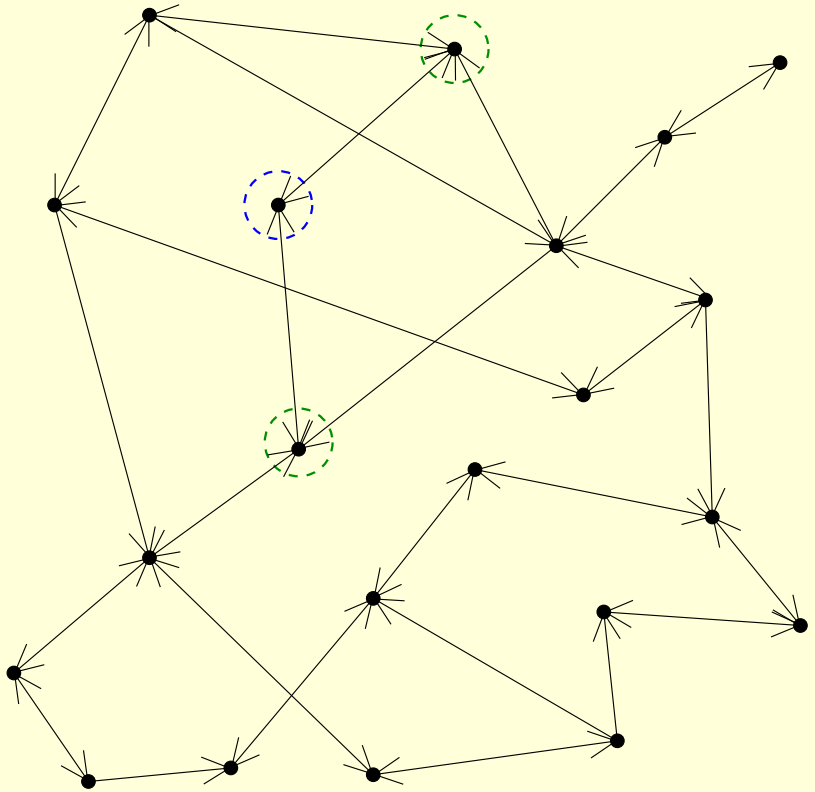


Explored

Frontier

Unexplored

Generic Search Template: Illustration

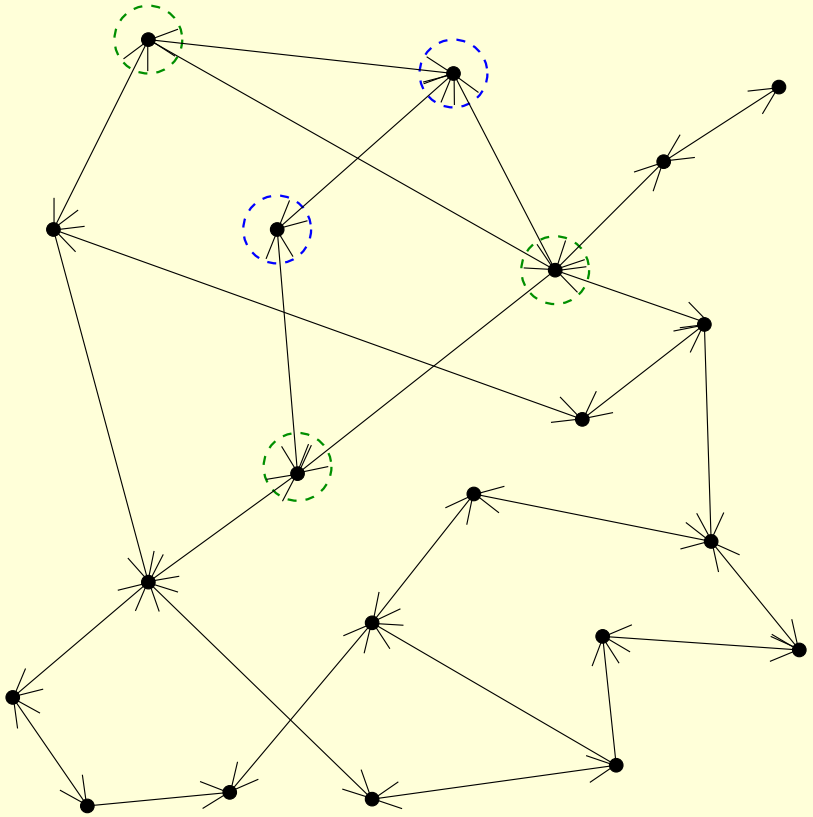


Explored

Frontier

Unexplored

Generic Search Template: Illustration

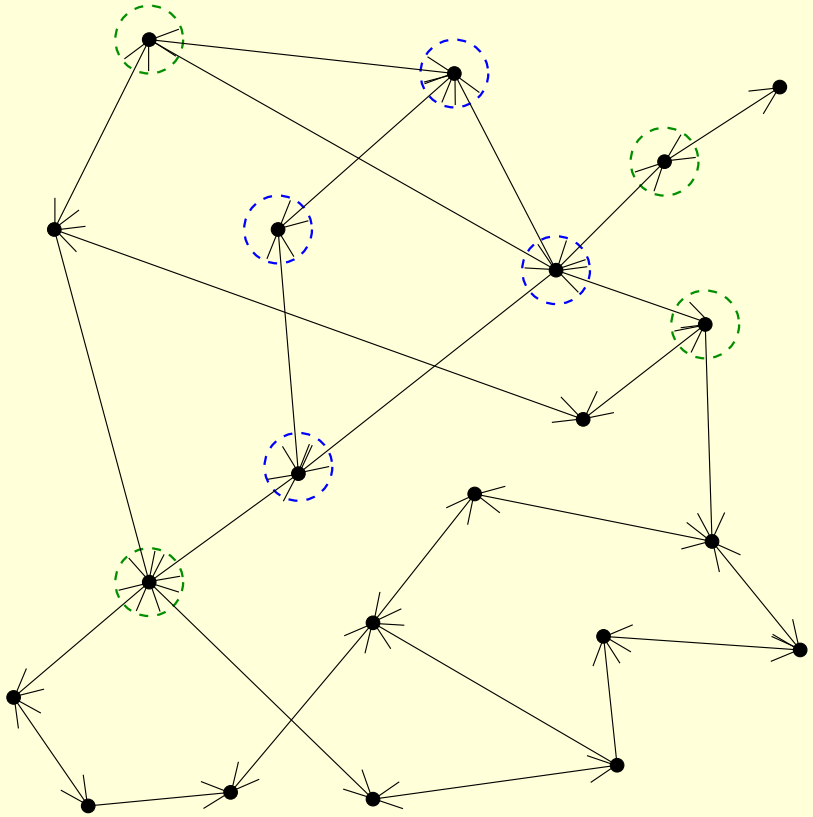


Explored

Frontier

Unexplored

Generic Search Template: Illustration

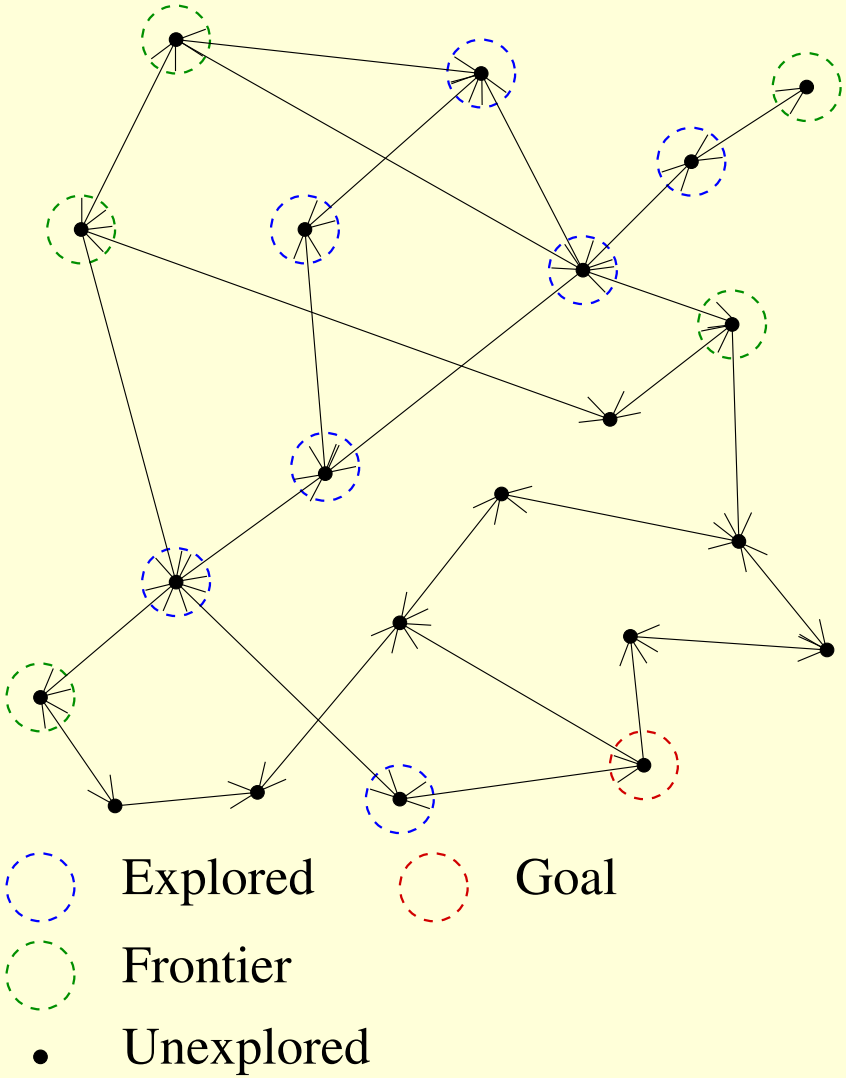


Explored

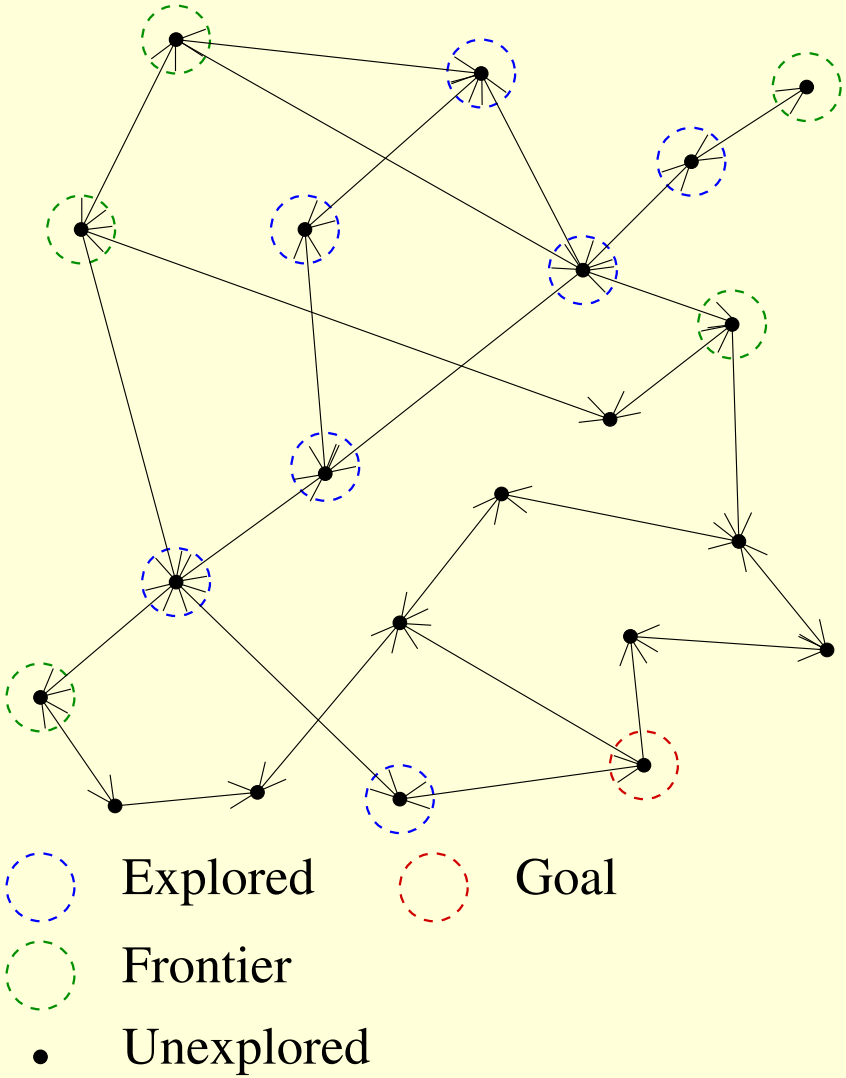
Frontier

Unexplored

Generic Search Template: Illustration



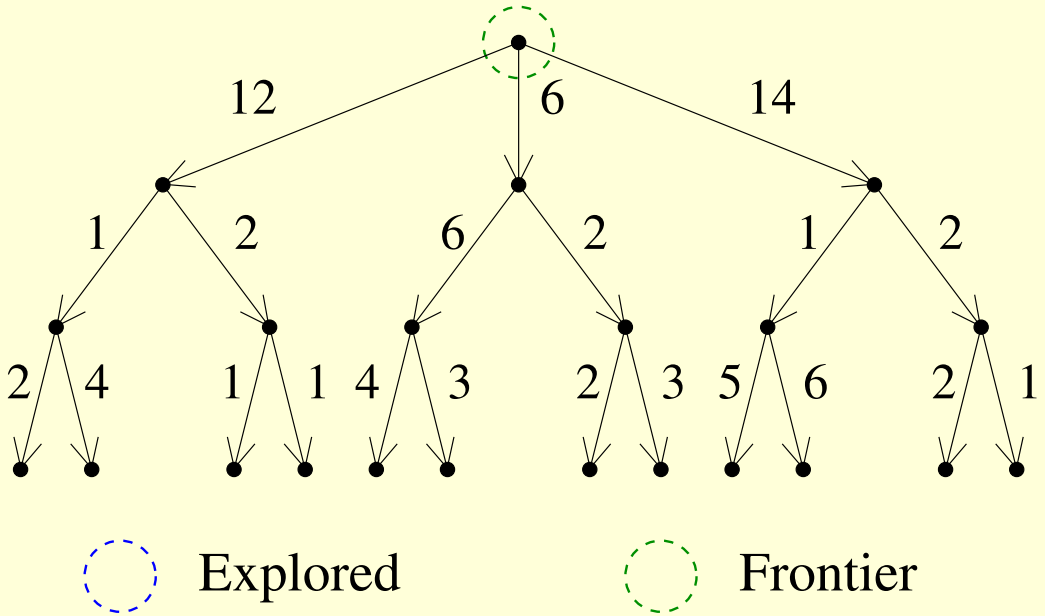
Generic Search Template: Illustration



How did we decide which frontier nodes to expand?

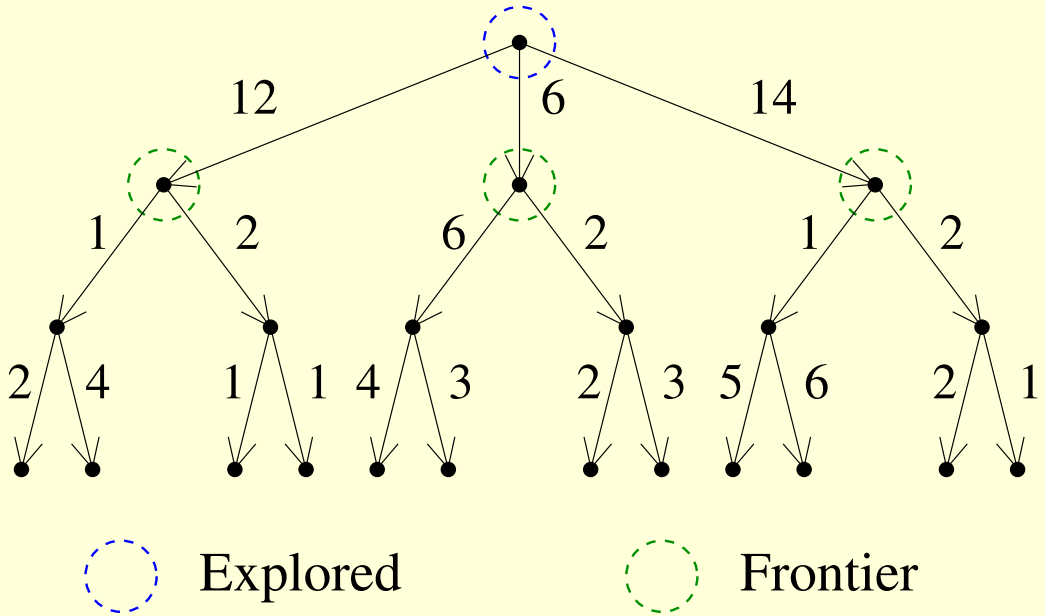
Depth-first Search (DFS)

Expand frontier node with longest path from start state.



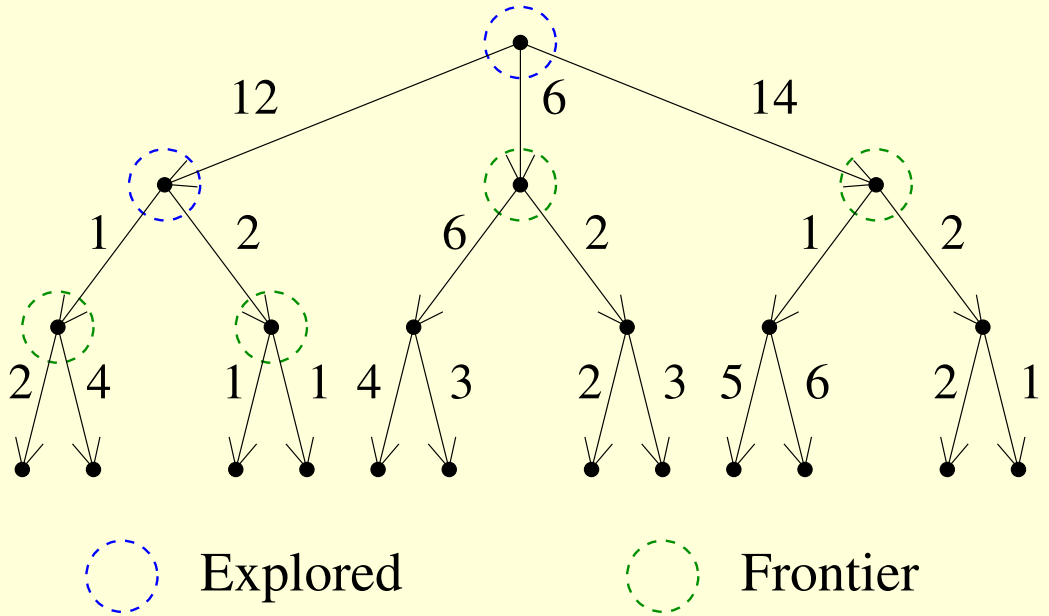
Depth-first Search (DFS)

Expand frontier node with longest path from start state.



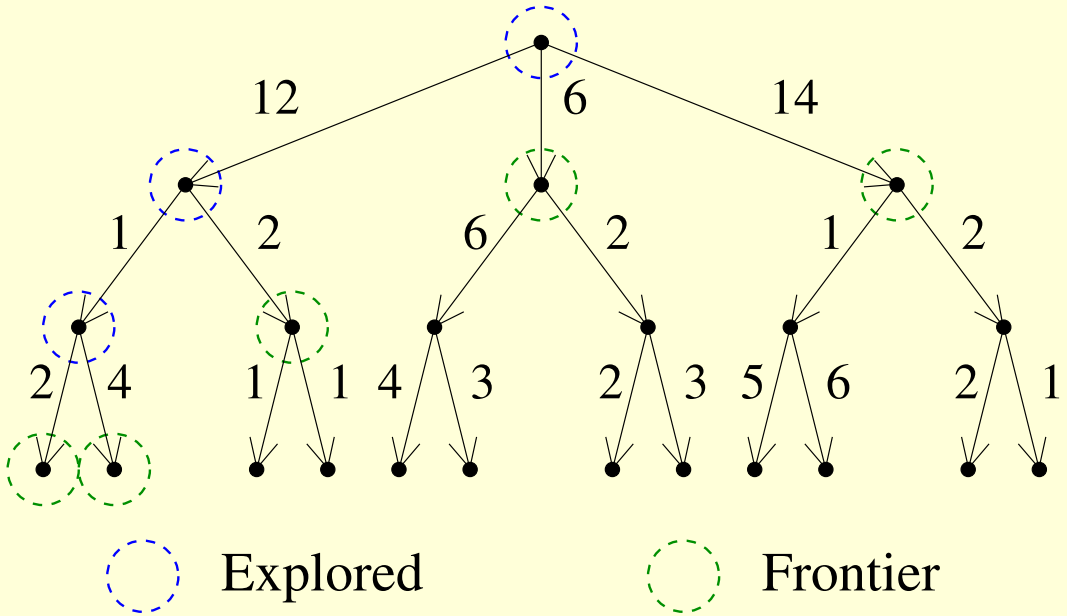
Depth-first Search (DFS)

Expand frontier node with longest path from start state.



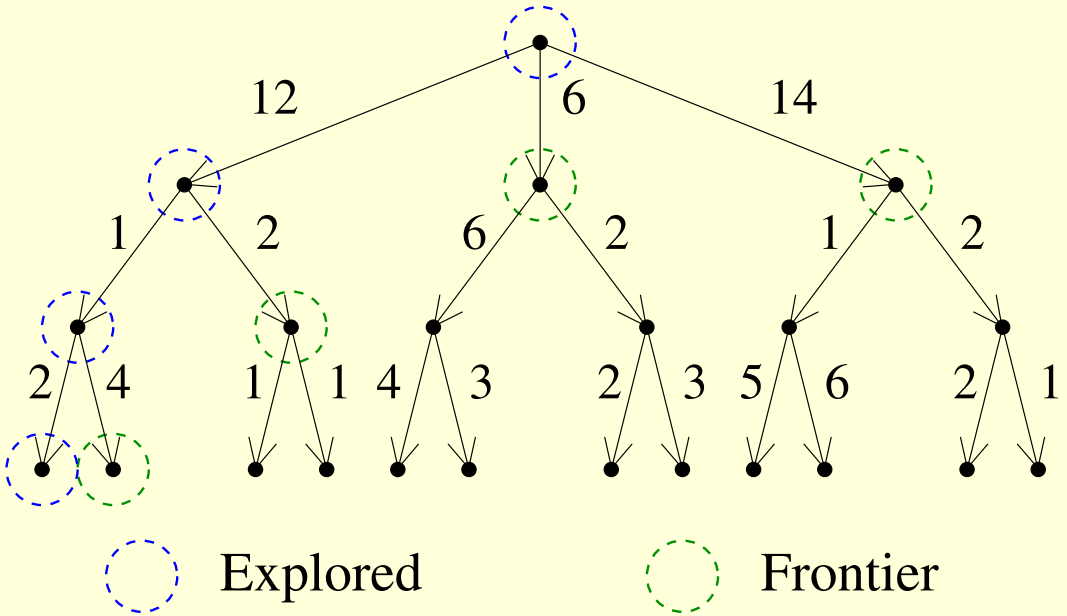
Depth-first Search (DFS)

Expand frontier node with longest path from start state.



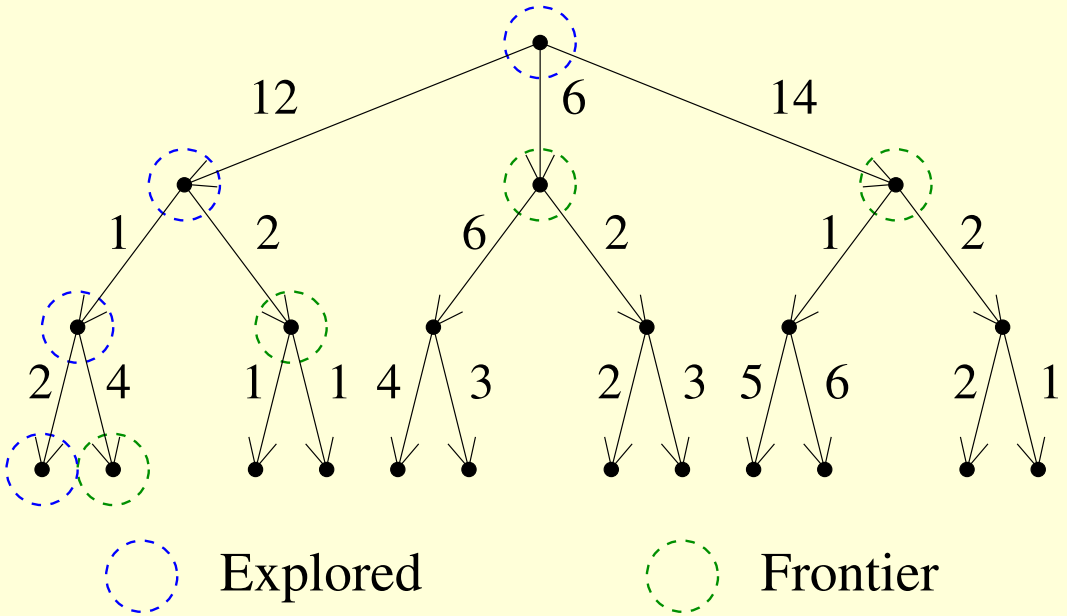
Depth-first Search (DFS)

Expand frontier node with longest path from start state.



Depth-first Search (DFS)

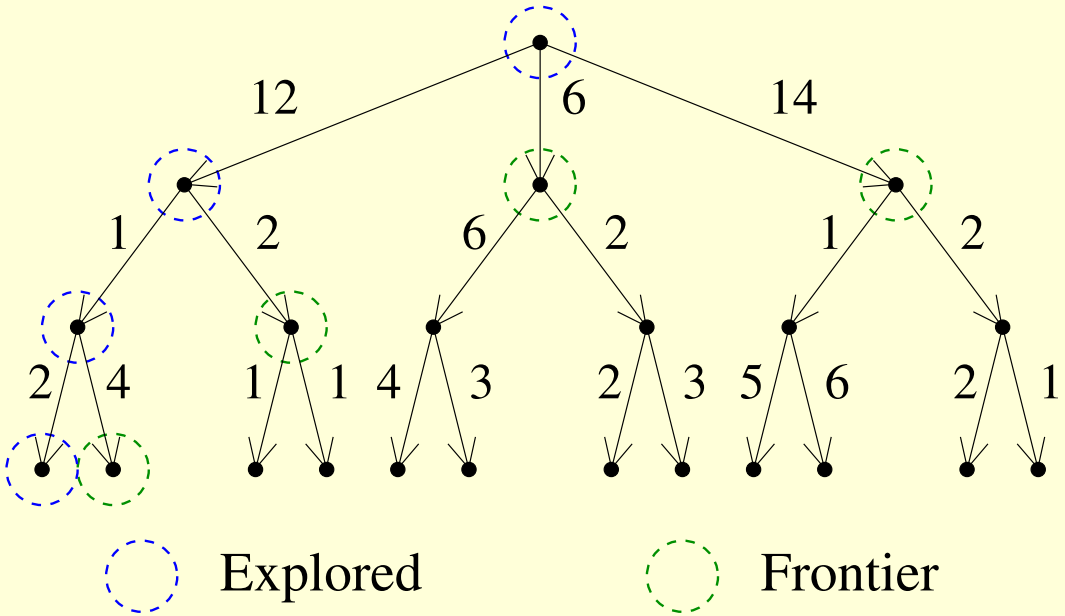
Expand frontier node with longest path from start state.



- Frontier treated like a stack (LIFO).

Depth-first Search (DFS)

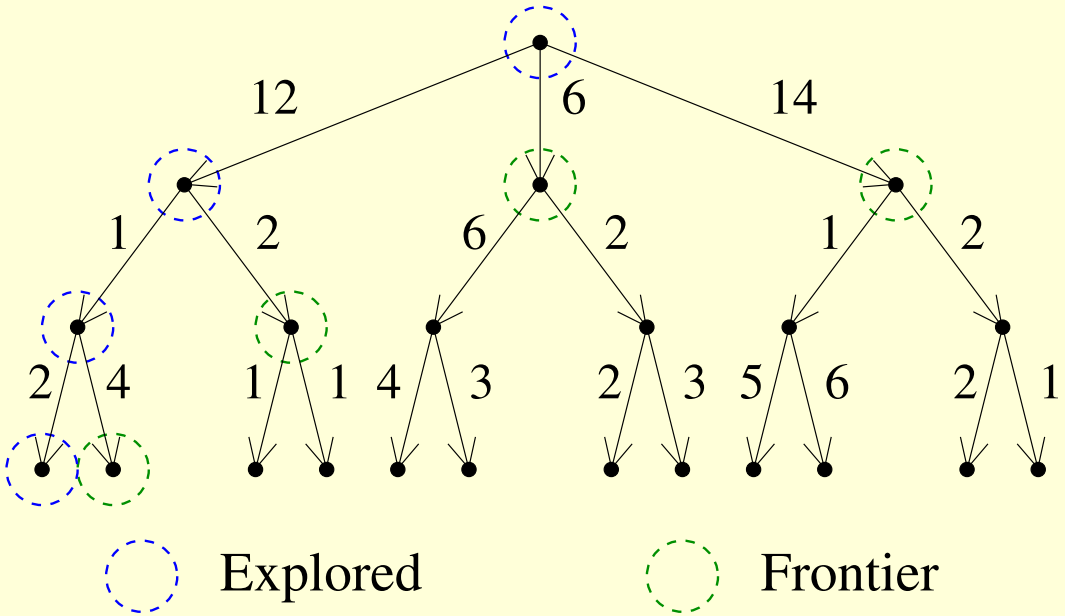
Expand frontier node with longest path from start state.



- Frontier treated like a stack (LIFO).
- No need to explicitly maintain frontier (can be constructed on-line).

Depth-first Search (DFS)

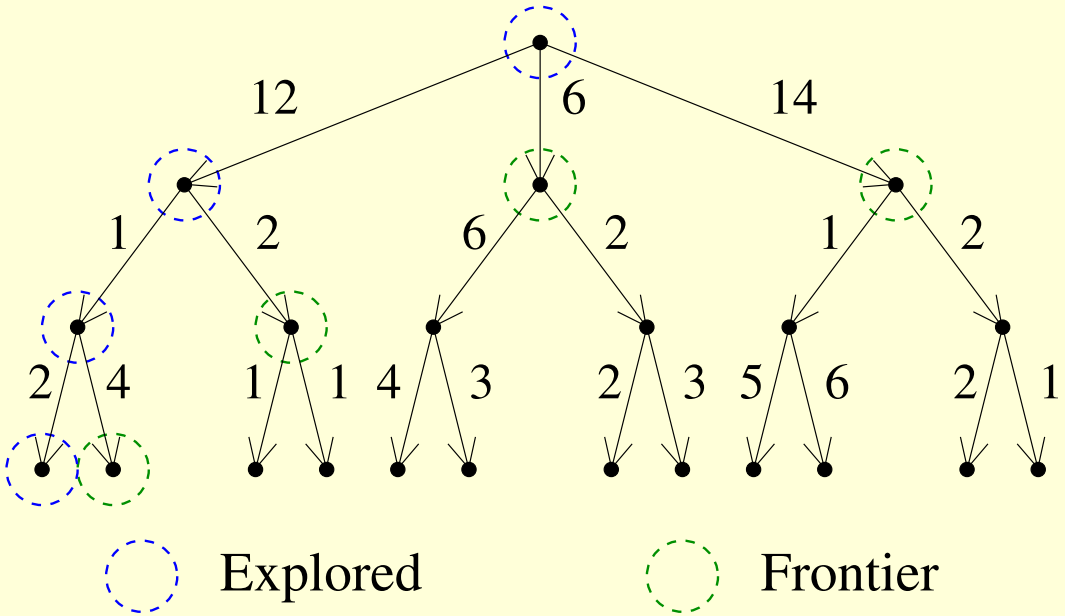
Expand frontier node with longest path from start state.



- Frontier treated like a stack (LIFO).
- No need to explicitly maintain frontier (can be constructed on-line).
- Guaranteed to terminate on finite search instances.

Depth-first Search (DFS)

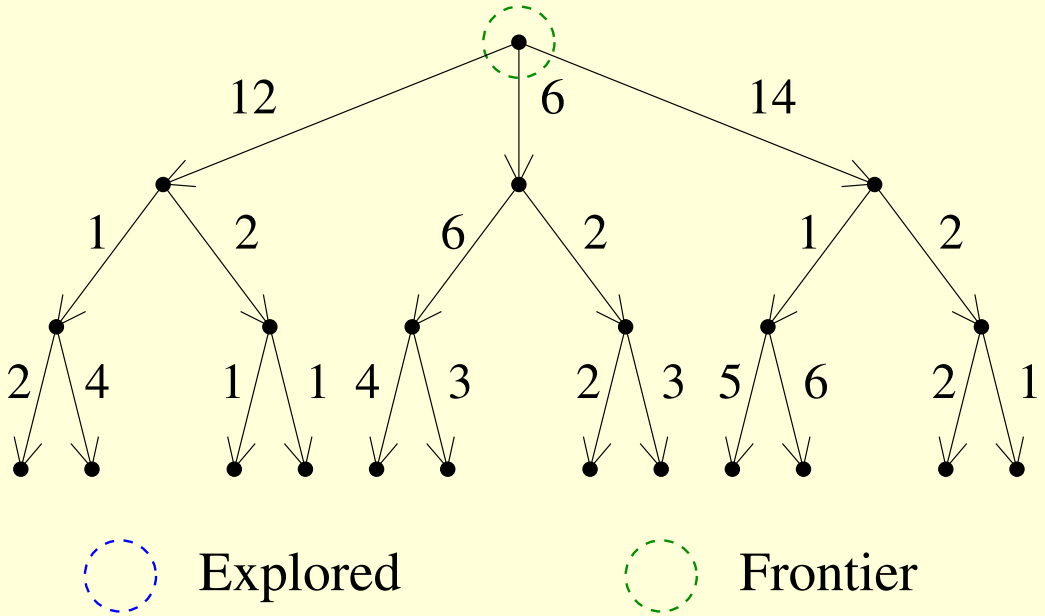
Expand frontier node with longest path from start state.



- Frontier treated like a stack (LIFO).
- No need to explicitly maintain frontier (can be constructed on-line).
- Guaranteed to terminate on finite search instances.
- Memory requirement linear in depth d.

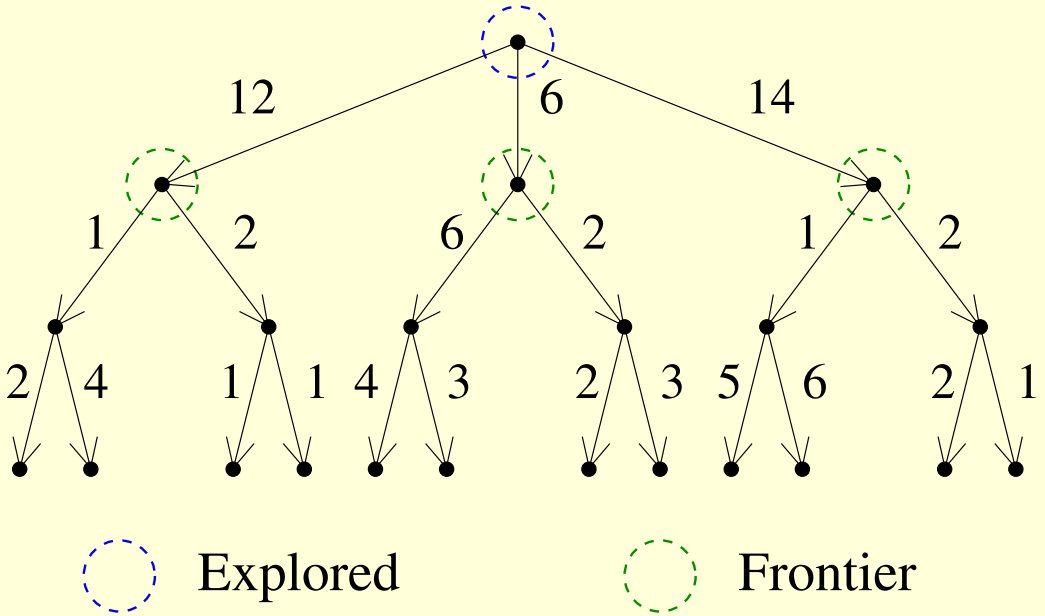
Breadth-first Search (BFS)

Expand frontier node with shortest path from start state.



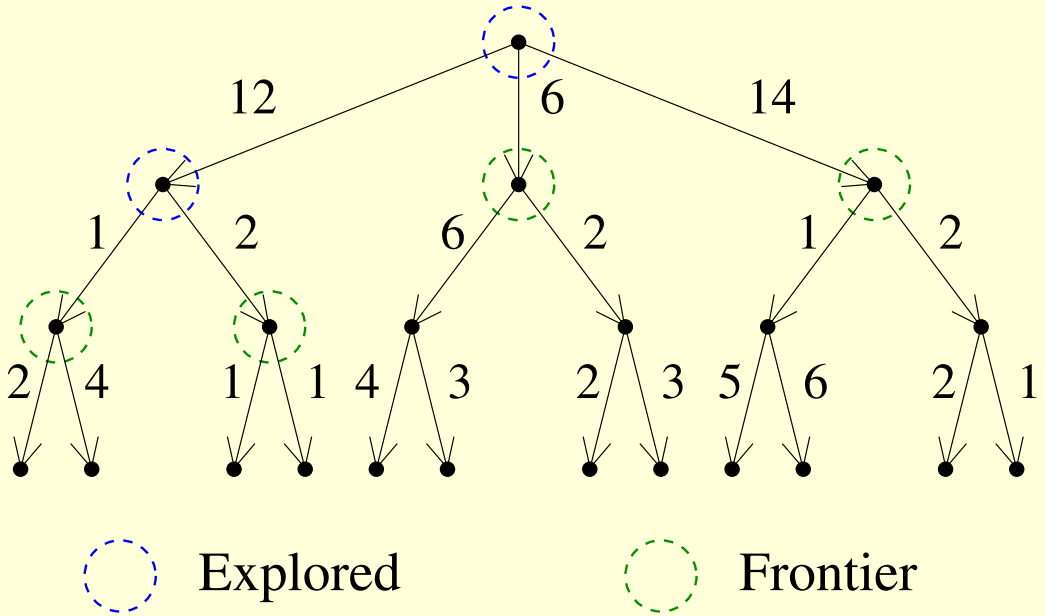
Breadth-first Search (BFS)

Expand frontier node with shortest path from start state.



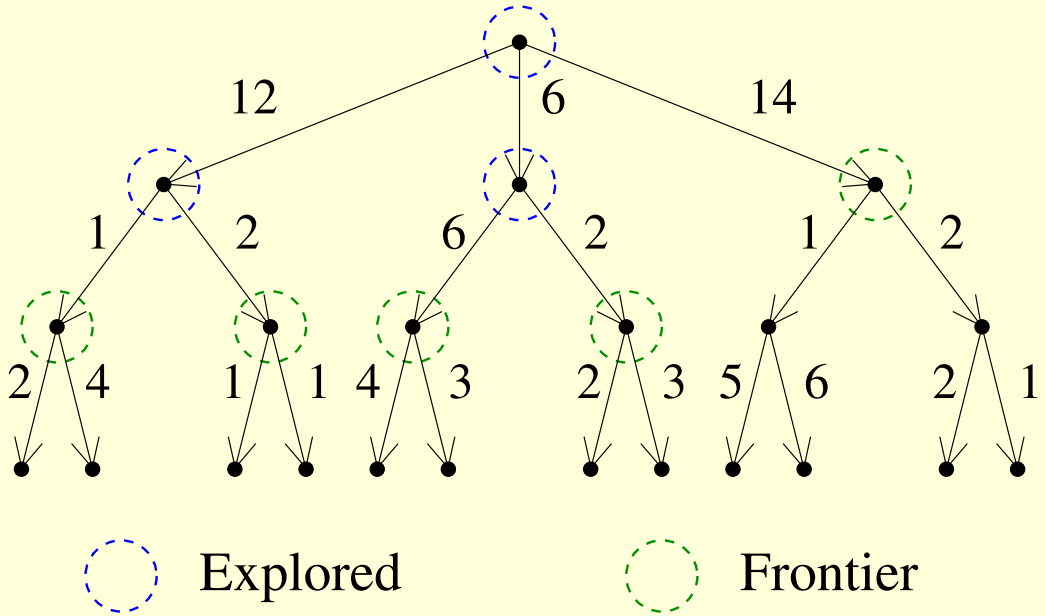
Breadth-first Search (BFS)

Expand frontier node with shortest path from start state.



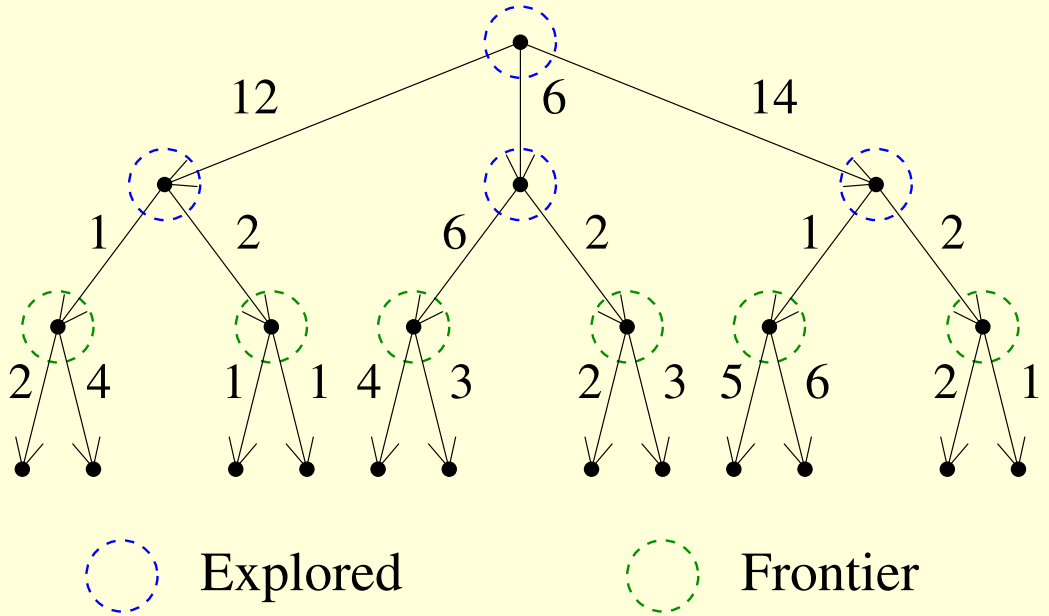
Breadth-first Search (BFS)

Expand frontier node with shortest path from start state.



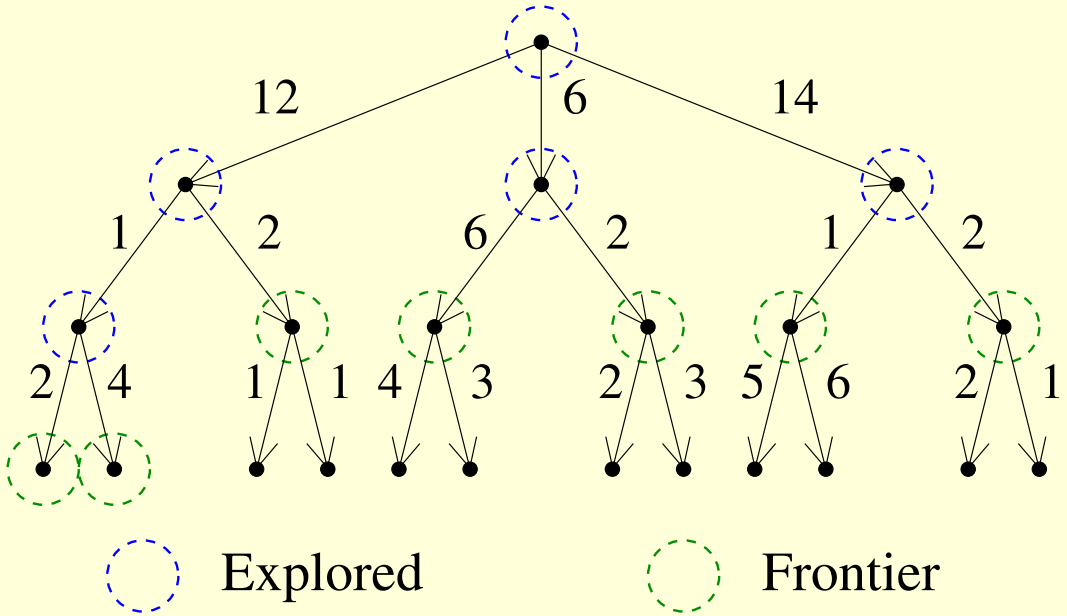
Breadth-first Search (BFS)

Expand frontier node with shortest path from start state.



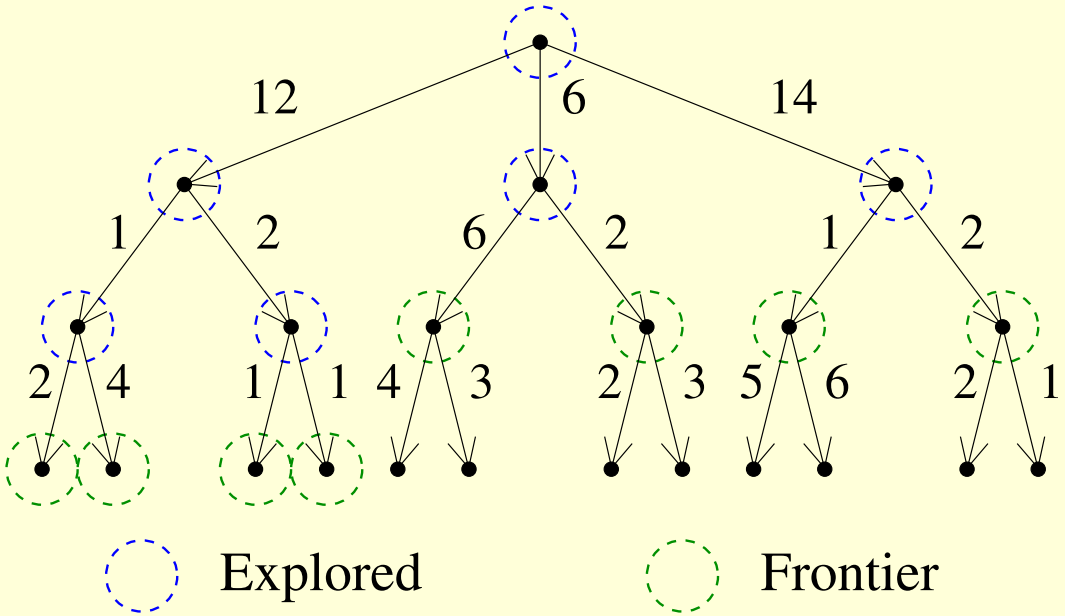
Breadth-first Search (BFS)

Expand frontier node with shortest path from start state.



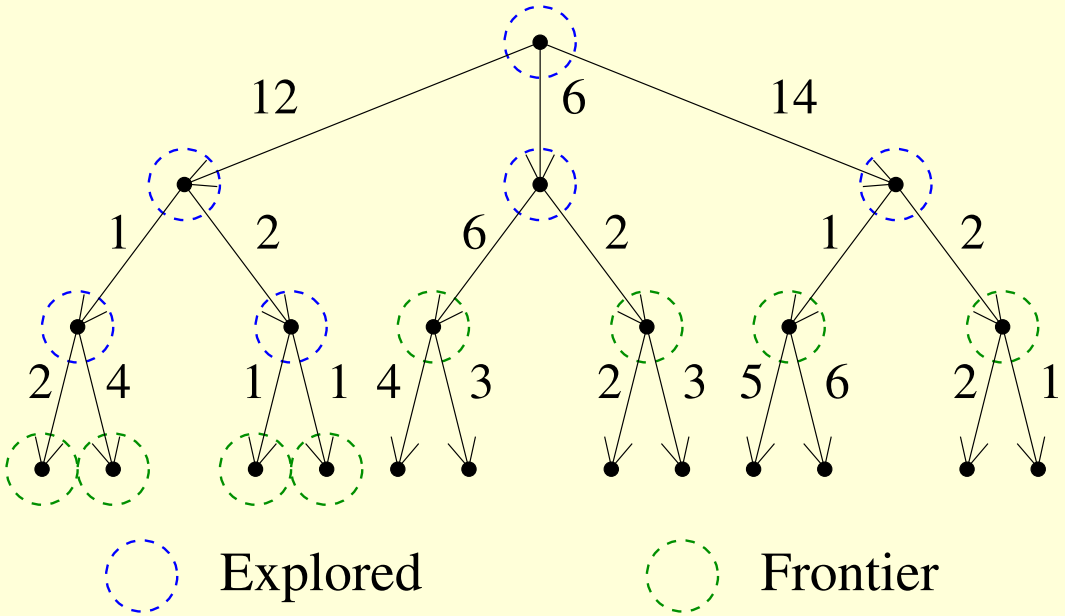
Breadth-first Search (BFS)

Expand frontier node with shortest path from start state.



Breadth-first Search (BFS)

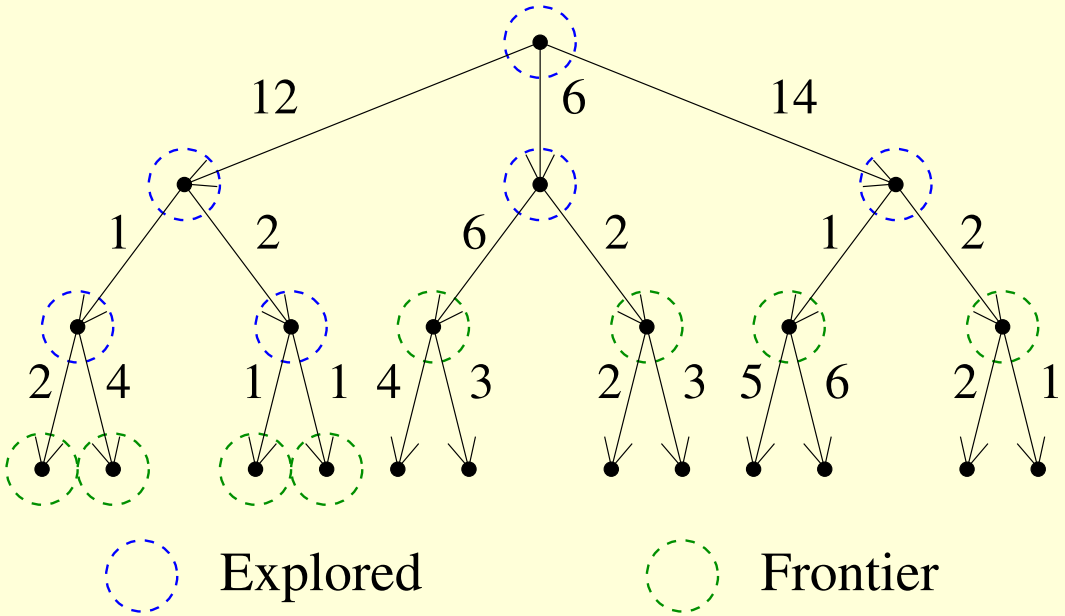
Expand frontier node with shortest path from start state.



- Frontier treated like a queue (FIFO).

Breadth-first Search (BFS)

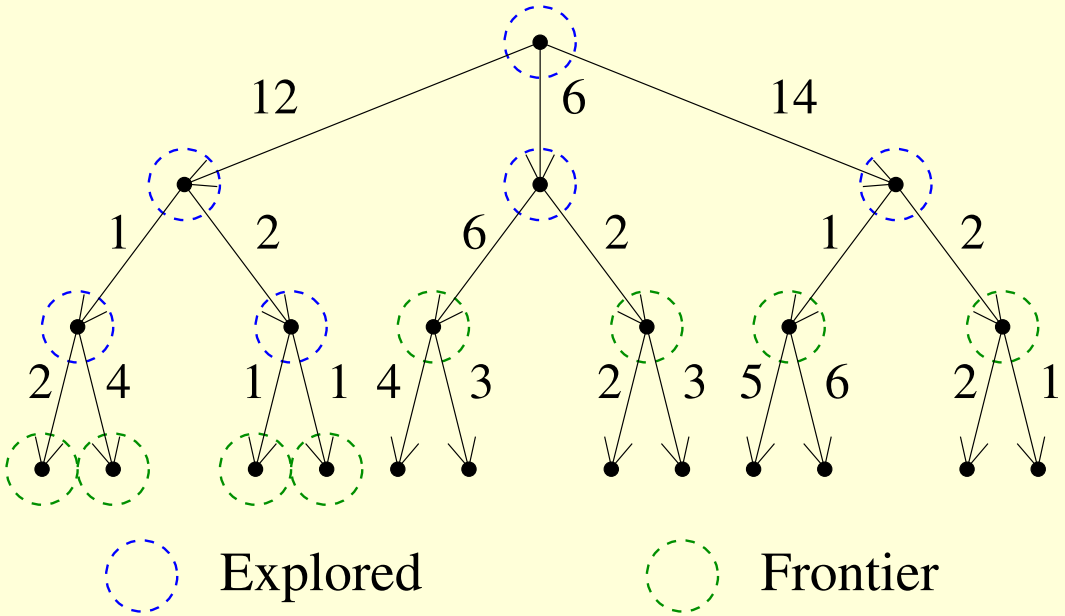
Expand frontier node with shortest path from start state.



- Frontier treated like a queue (FIFO).
- Guaranteed to terminate if search depth is finite.

Breadth-first Search (BFS)

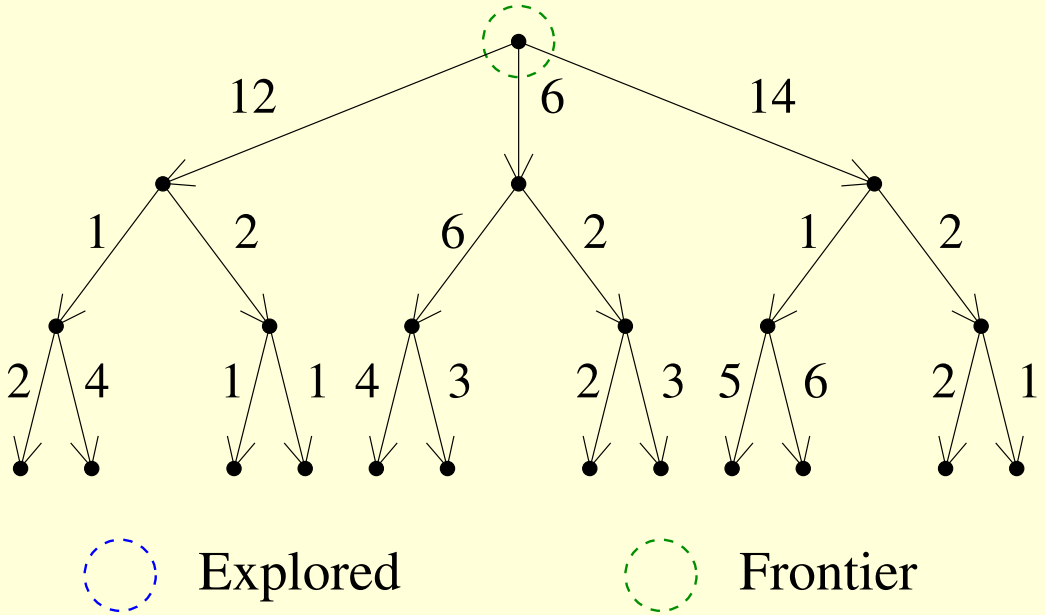
Expand frontier node with shortest path from start state.



- Frontier treated like a queue (FIFO).
- Guaranteed to terminate if search depth is finite.
- Memory requirement $O(b^d)$.

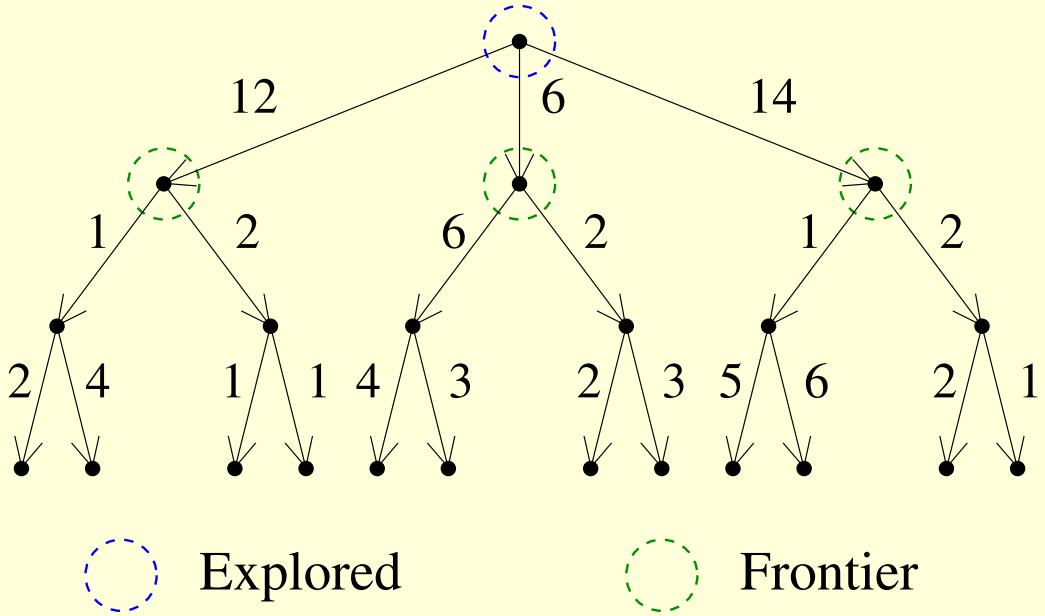
Lowest-cost-first Search (LCFS)

Expand frontier node with lowest path-cost from start state.



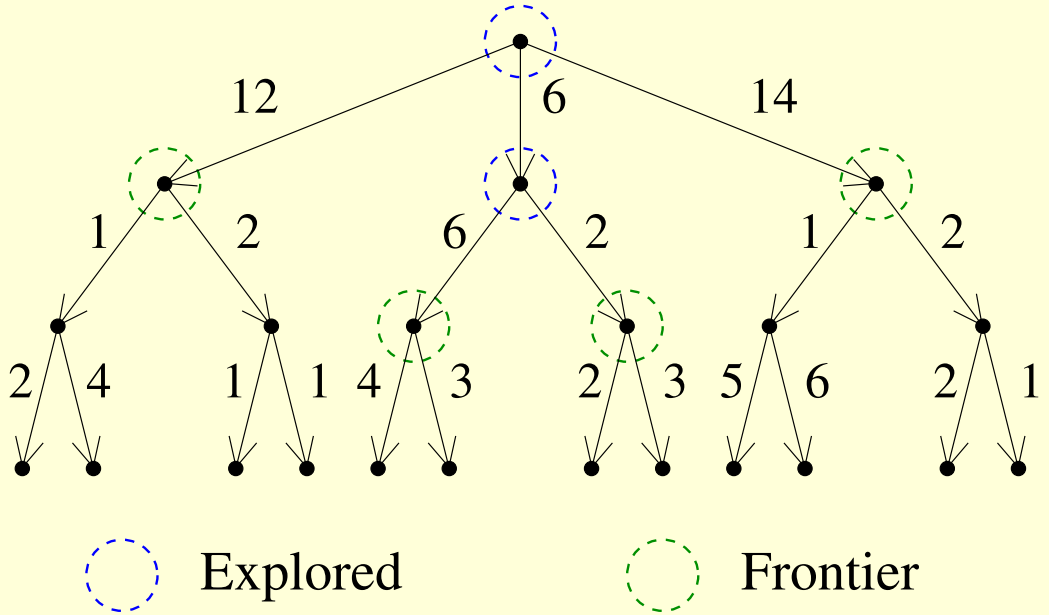
Lowest-cost-first Search (LCFS)

Expand frontier node with lowest path-cost from start state.



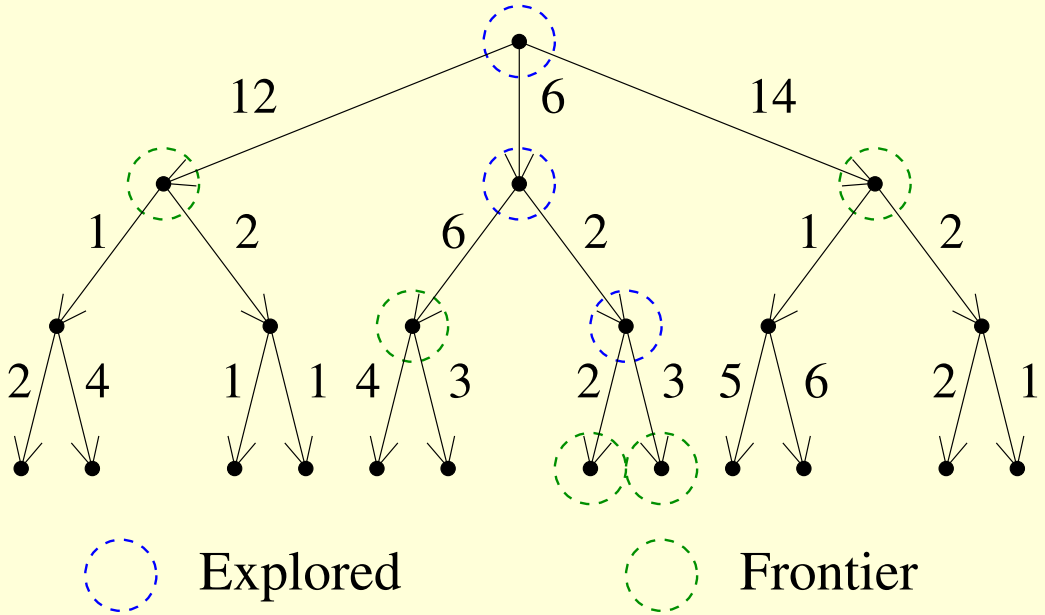
Lowest-cost-first Search (LCFS)

Expand frontier node with lowest path-cost from start state.



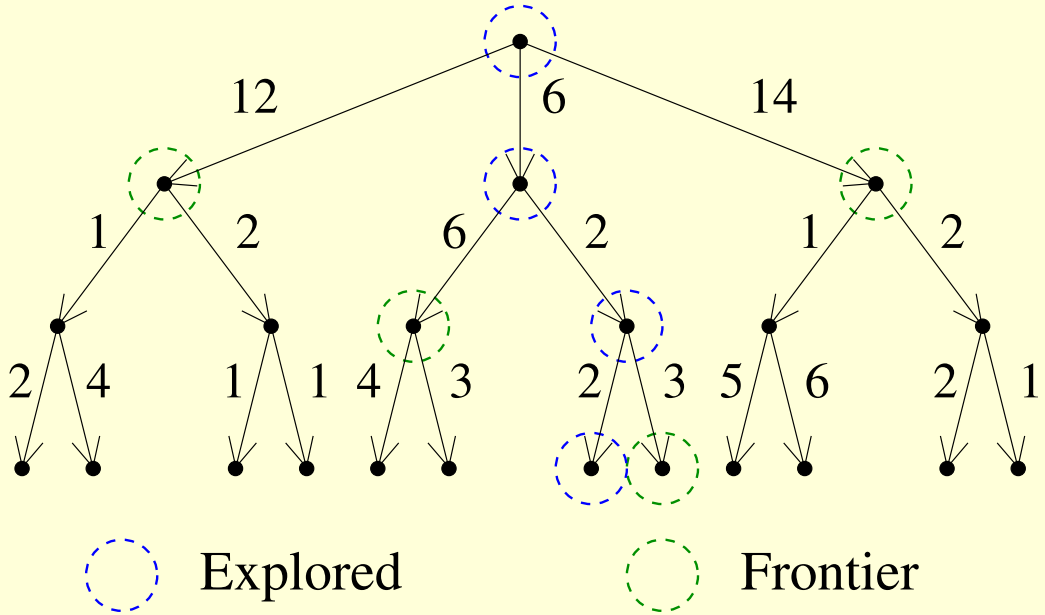
Lowest-cost-first Search (LCFS)

Expand frontier node with lowest path-cost from start state.



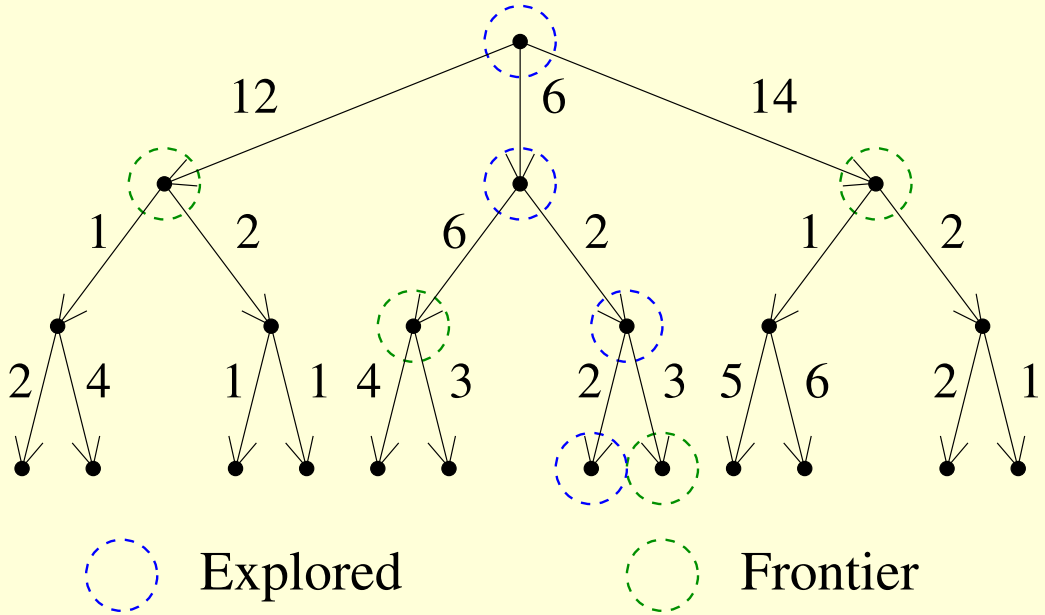
Lowest-cost-first Search (LCFS)

Expand frontier node with lowest path-cost from start state.



Lowest-cost-first Search (LCFS)

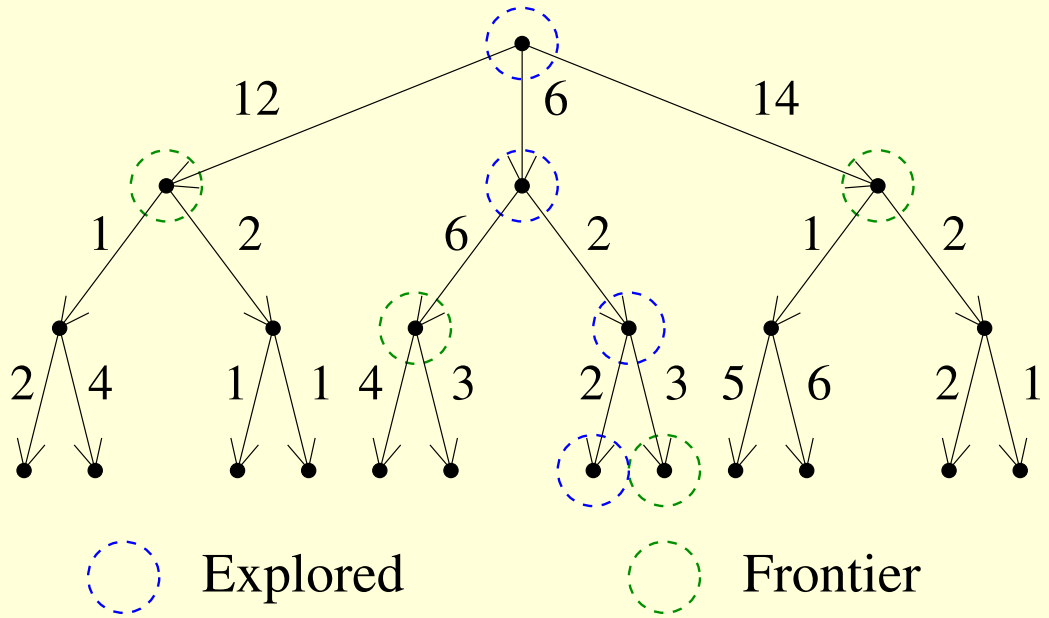
Expand frontier node with lowest path-cost from start state.



- Frontier treated like a priority queue (priority = path-cost from start state).

Lowest-cost-first Search (LCFS)

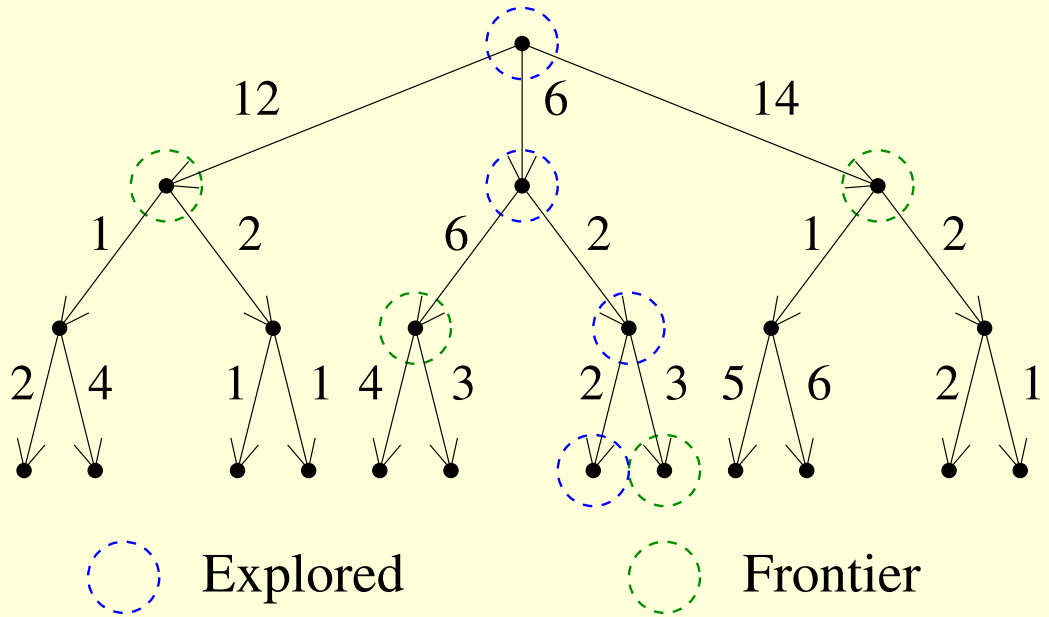
Expand frontier node with lowest path-cost from start state.



- Frontier treated like a priority queue (priority = path-cost from start state).
- Guaranteed to terminate if search depth is finite and each cost exceeds $\epsilon > 0$.

Lowest-cost-first Search (LCFS)

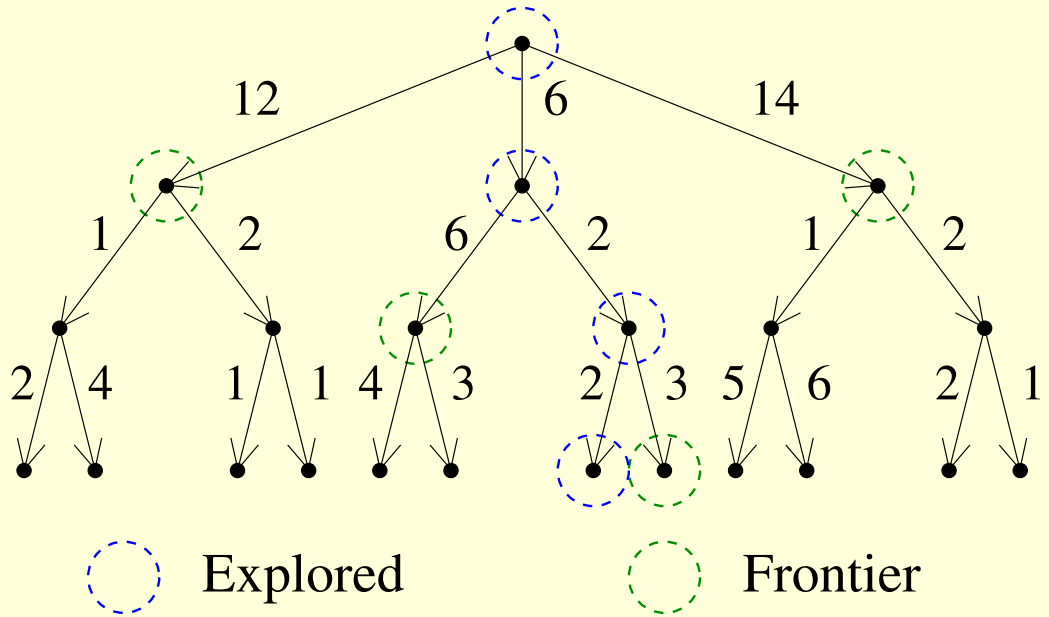
Expand frontier node with lowest path-cost from start state.



- Frontier treated like a priority queue (priority = path-cost from start state).
- Guaranteed to terminate if search depth is finite and each cost exceeds $\epsilon > 0$.
- Memory requirement depends heavily on instance.

Lowest-cost-first Search (LCFS)

Expand frontier node with lowest path-cost from start state.



- Frontier treated like a priority queue (priority = path-cost from start state).
- Guaranteed to terminate if search depth is finite and each cost exceeds $\epsilon > 0$.
- Memory requirement depends heavily on instance.
- For node n, denote path-cost from start state $g(n)$. Will need it shortly.

Questions

- I recall covering this material in a course on algorithms. Is it really AI?

Questions

- I recall covering this material in a course on algorithms. Is it really AI?

  You're right. DFS, BFS, LCFS are not considered AI. What's coming up next is!

Questions

- I recall covering this material in a course on algorithms. Is it really AI?
  You're right. DFS, BFS, LCFS are not considered AI. What's coming up next is!

- Is LCFS related to Dijkstra's shortest path algorithm?

Questions

- I recall covering this material in a course on algorithms. Is it really AI?
  You're right. DFS, BFS, LCFS are not considered AI. What's coming up next is!

- Is LCFS related to Dijkstra's shortest path algorithm?
  Yes. It's the same algorithm!

References

- Sections 3.1, 3.2, 3.3, 3.4, 3.5, **Artificial Intelligence: Foundations of Computational Agents**, David Poole and Alan Mackworth, Cambridge University Press, 2010. Available on-line at `https://artint.info/html/ArtInt.html`.