CSC3206 Artificial Intelligence

Uninformed Search

Lecture Outline

Uninformed search

Breadth-first search · Depth first search · Uniform-cost search

Uninformed search

Uninformed search

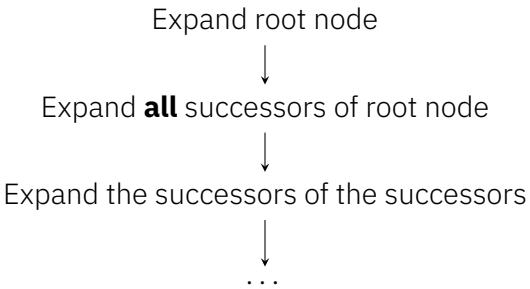
- also known as blind search.
- uses strategies with no additional information beyond the problem definition.

Uninformed search

Uninformed search algorithms

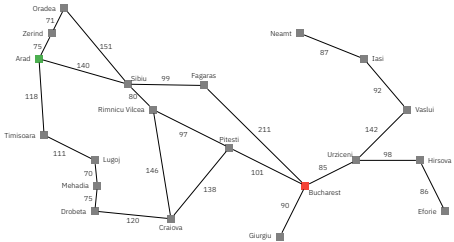
- Breadth-first search (BFS)
- Depth-first search (DFS)
- Uniform-cost search (UCS)

Breadth-first search



Breadth-first search

Nick's problem in Romania

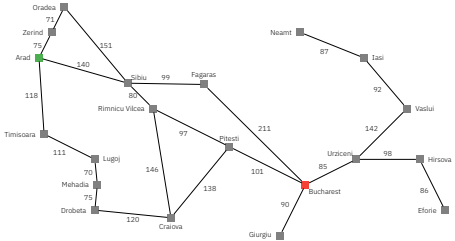


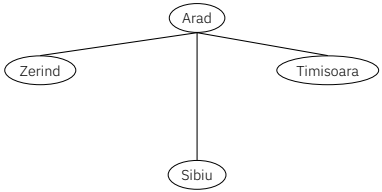
Arad

Nick is currently at *Arad*, therefore we start the search tree from *Arad*. This node is called the **root node**.

Breadth-first search

Nick's problem in Romania





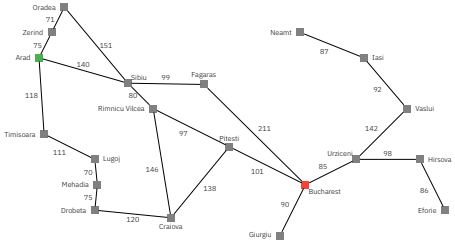
After expanding *Arad*, three **children nodes** are generated. *Arad* is the **parent node** of these three children nodes.

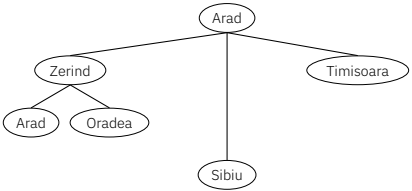
The nodes with no children are called the **leaf nodes**.

The set of leaf nodes will be added to the **frontier**, i.e. the list of nodes to be expanded later.

Breadth-first search

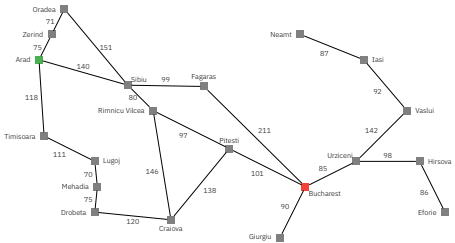
Nick's problem in Romania

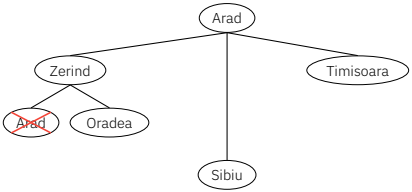




Breadth-first search

Nick's problem in Romania



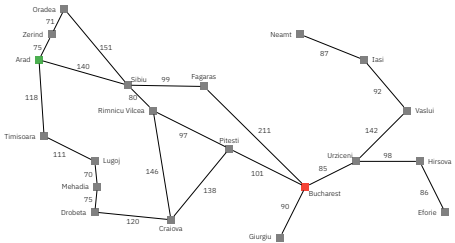


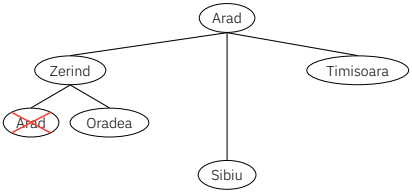
Remove *Arad* as child node of *Zerind* due to loopy path.

Loopy path is an example of **redundant paths**. Redundant paths exist when there is more than one way to get to a state in the search tree.

Breadth-first search

Nick's problem in Romania

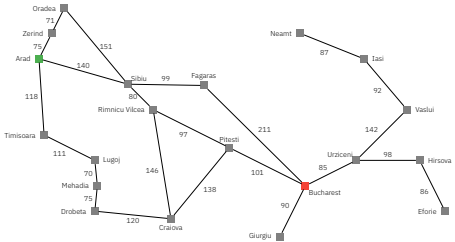


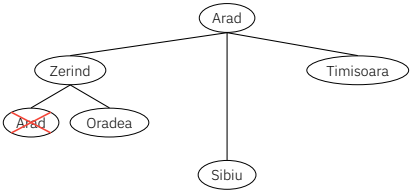


In breadth-first search, which node should we expand? Oradea or Sibiu?

Breadth-first search

Nick's problem in Romania

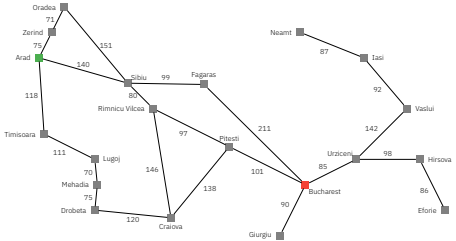


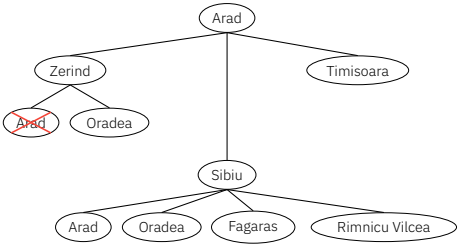


The answer is… Sibiu

Breadth-first search

Nick's problem in Romania

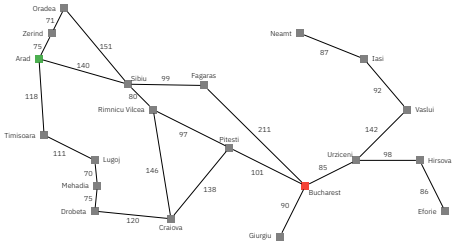


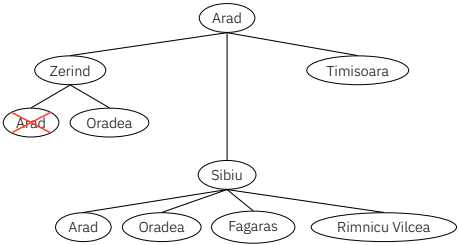


The answer is… Sibiu

Breadth-first search

Nick's problem in Romania



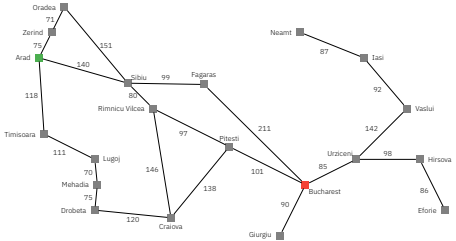


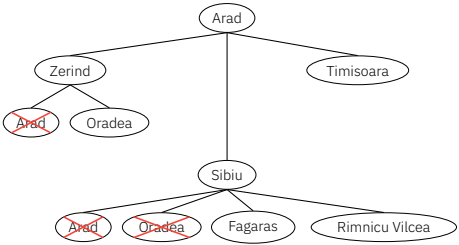
The answer is… Sibiu

- The shallowest unexpanded node is chosen for expansion
- FIFO queue for the frontier

Breadth-first search

Nick's problem in Romania

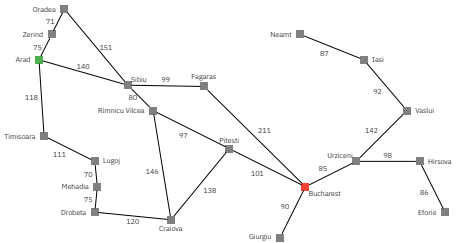


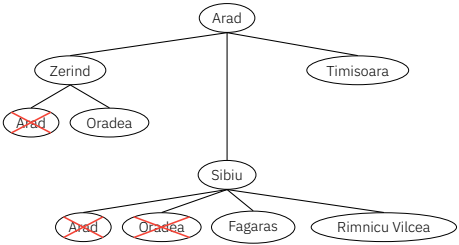


Why are the children nodes, *Arad* and *Oradea*, of *Sibiu* are removed?

Breadth-first search

Nick's problem in Romania



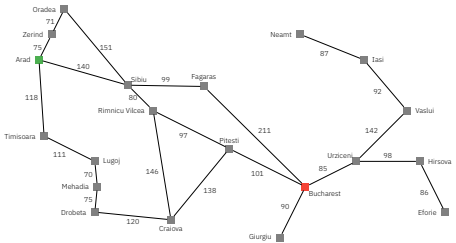


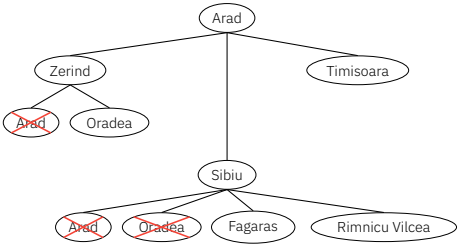
Why are the children nodes, *Arad* and *Oradea*, of *Sibiu* are removed?

Arad is removed as it creates a loopy path.

Breadth-first search

Nick's problem in Romania





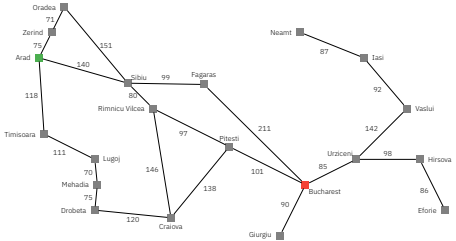
Why are the children nodes, *Arad* and *Oradea*, of *Sibiu* are removed?

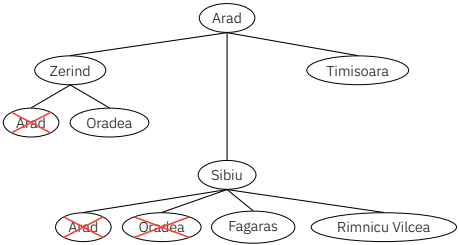
Arad is removed as it creates a loopy path.

Oradea is removed as it creates a redundant path.

Breadth-first search

Nick's problem in Romania

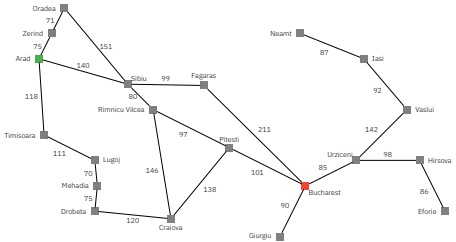


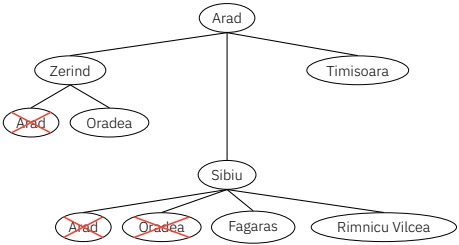


Why is *Oradea* child node of *Sibiu* instead of *Oradea* child node of *Zerind* is removed? They are of the same depth, isn't it?

Breadth-first search

Nick's problem in Romania



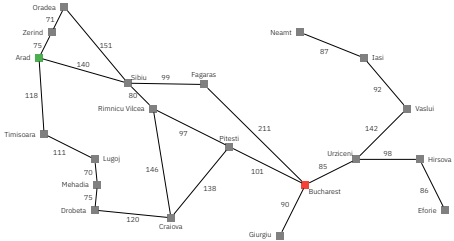


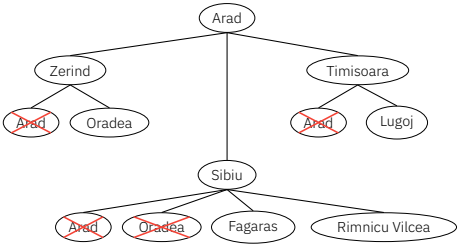
Why is *Oradea* child node of *Sibiu* instead of *Oradea* child node of *Zerind* is removed? They are of the same depth, isn't it?

As *Oradea* child node of *Zerind* came first, it's the node to be kept.

Breadth-first search

Nick's problem in Romania

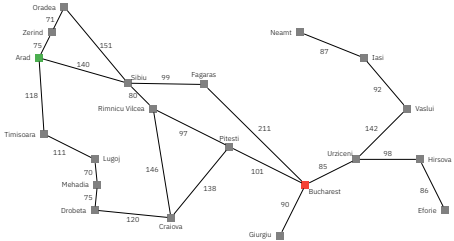


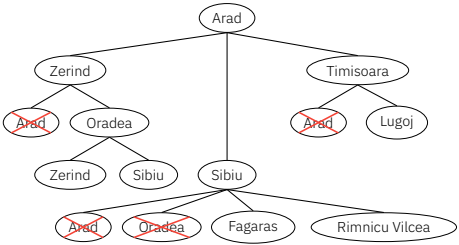


Continue to expand…

Breadth-first search

Nick's problem in Romania

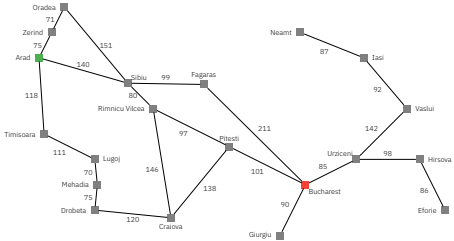


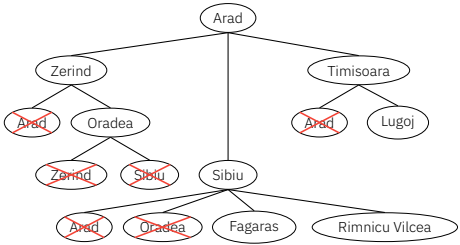


Continue to expand…

Breadth-first search

Nick's problem in Romania

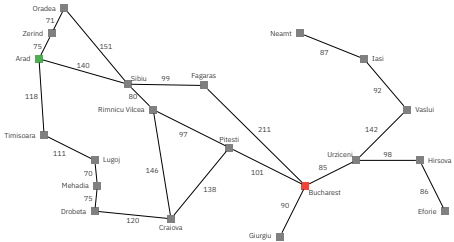


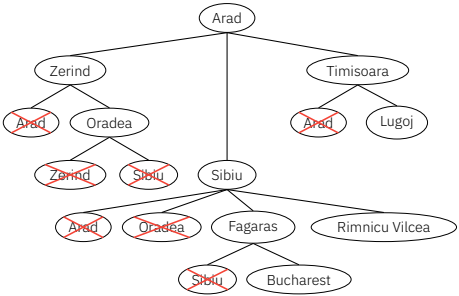


Removing children nodes that create redundant paths.

Breadth-first search

Nick's problem in Romania

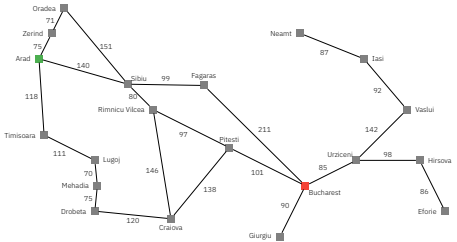


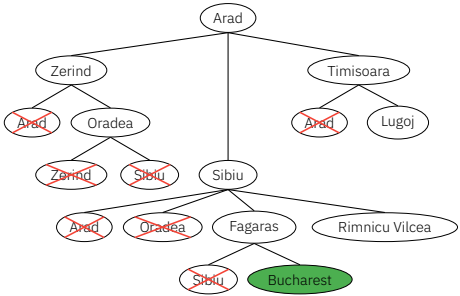


Continue to expand…

Breadth-first search

Nick's problem in Romania





The goal node is found – *Bucharest*

- The search algorithm thus now stops.
- This goal node is definitely the shallowest goal node, but not necessarily the physically nearest one.

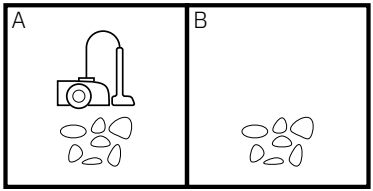
Redundant path

- Tree search
  does not remove nodes that create redundant paths
- Graph search
  remove nodes that create redundant paths

Breadth-first search

Vacuum world

Implement BFS to find the solution to clean the two rooms in the vacuum world.

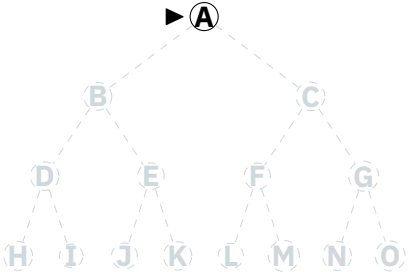


Depth-first search

Depth-first search (DFS) expands the deepest node in the current frontier of the search tree.

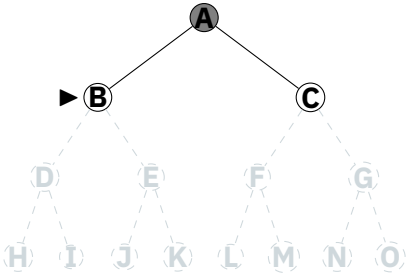
Depth-first search

Illustrating DFS on a binary tree with 3 steps, i.e. each node can be expanded to two children nodes, and the nodes at depth 3 have no descendants, with **M** be the goal node.



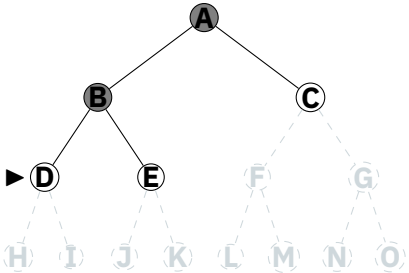
Depth-first search

Illustrating DFS on a binary tree with 3 steps, i.e. each node can be expanded to two children nodes, and the nodes at depth 3 have no descendants, with **M** be the goal node.



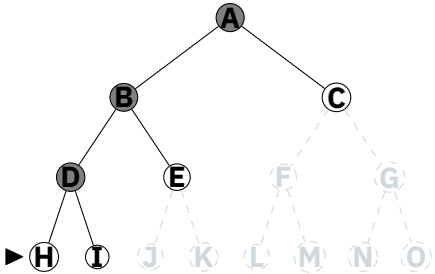
Depth-first search

Illustrating DFS on a binary tree with 3 steps, i.e. each node can be expanded to two children nodes, and the nodes at depth 3 have no descendants, with **M** be the goal node.



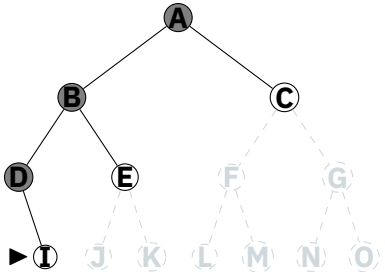
Depth-first search

Illustrating DFS on a binary tree with 3 steps, i.e. each node can be expanded to two children nodes, and the nodes at depth 3 have no descendants, with **M** be the goal node.



Depth-first search

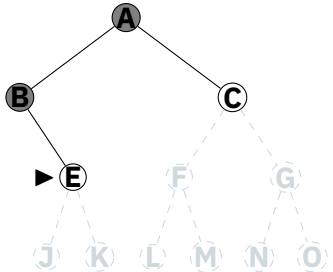
Illustrating DFS on a binary tree with 3 steps, i.e. each node can be expanded to two children nodes, and the nodes at depth 3 have no descendants, with **M** be the goal node.



Explored nodes with no descendants in the frontier are removed from memory

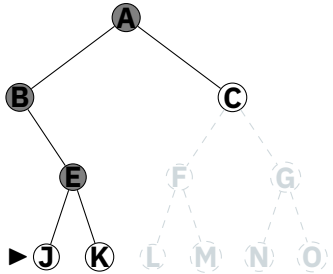
Depth-first search

Illustrating DFS on a binary tree with 3 steps, i.e. each node can be expanded to two children nodes, and the nodes at depth 3 have no descendants, with **M** be the goal node.



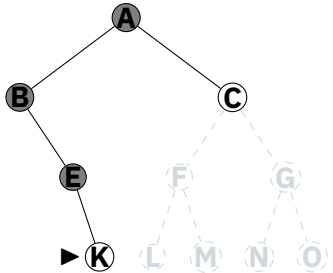
Depth-first search

Illustrating DFS on a binary tree with 3 steps, i.e. each node can be expanded to two children nodes, and the nodes at depth 3 have no descendants, with **M** be the goal node.



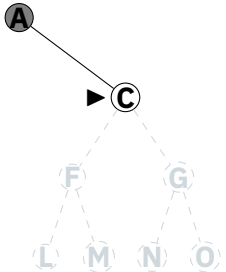
Depth-first search

Illustrating DFS on a binary tree with 3 steps, i.e. each node can be expanded to two children nodes, and the nodes at depth 3 have no descendants, with **M** be the goal node.



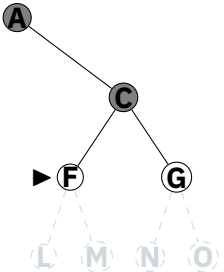
Depth-first search

Illustrating DFS on a binary tree with 3 steps, i.e. each node can be expanded to two children nodes, and the nodes at depth 3 have no descendants, with **M** be the goal node.



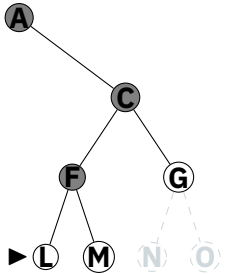
Depth-first search

Illustrating DFS on a binary tree with 3 steps, i.e. each node can be expanded to two children nodes, and the nodes at depth 3 have no descendants, with **M** be the goal node.



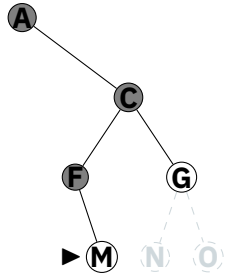
Depth-first search

Illustrating DFS on a binary tree with 3 steps, i.e. each node can be expanded to two children nodes, and the nodes at depth 3 have no descendants, with **M** be the goal node.



Depth-first search

Illustrating DFS on a binary tree with 3 steps, i.e. each node can be expanded to two children nodes, and the nodes at depth 3 have no descendants, with **M** be the goal node.



Goal node found

Depth-first search

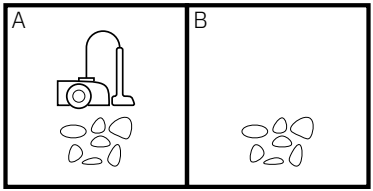
Variants

- Depth-limited search
  A depth limit is imposed. Nodes at the depth limit are treated as having no successors.
- Iterative deepening depth-first search
  Iteratively changes the depth limit in order to find the shallowest depth to reach the goal.

Depth-first search

Vacuum world

Implement DFS to find the solution to clean the two rooms in the vacuum world.



Uniform-cost search

Breadth-first search (BFS)	Uniform-cost search (UCS)
Expand on shallowest node	Expand on node with lowest path cost
FIFO queue in frontier	Priority queue based on path cost in frontier

Uniform-cost search

- Redundant path resolution
  When a redundant path exists, the path with a higher path cost is removed.
- Goal test
  The goal test (testing if a node is goal) is carried out when a node is selected for expansion rather than when it is generated.

Uniform-cost search

Frontier ordered by path cost (► indicates the next node for expansion)

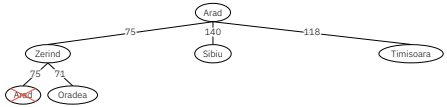
	Path cost	Path leads to the leaf node
►	0	Arad

Uniform-cost search

Frontier ordered by path cost (▶ indicates the next node for expansion)

	Path cost	Path leads to the leaf node
▶	75	Arad – Zerind
	118	Arad – Timisoara
	140	Arad – Sibiu

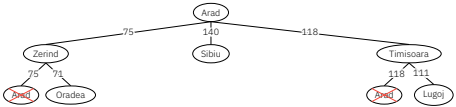
Uniform-cost search



Frontier ordered by path cost (▶ indicates the next node for expansion)

	Path cost	Path leads to the leaf node
▶	118	Arad – Timisoara
	140	Arad – Sibiu
	146	Arad – Zerind – Oradea

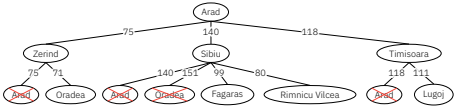
Uniform-cost search



Frontier ordered by path cost (▶ indicates the next node for expansion)

	Path cost	Path leads to the leaf node
▶	140	Arad – Sibiu
	146	Arad – Zerind – Oradea
	229	Arad – Timisoara – Lugoj

Uniform-cost search

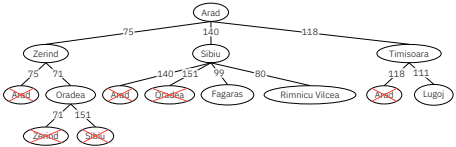


Frontier ordered by path cost (▶ indicates the next node for expansion)

	Path cost	Path leads to the leaf node
▶	146	Arad – Zerind – Oradea
	220	Arad – Sibiu – Rimnicu Vilcea
	229	Arad – Timisoara – Lugoj
	239	Arad – Sibiu – Fagaras

Arad – Sibiu – Oradea (291) is removed in favour of Arad – Zerind – Oradea (146)

Uniform-cost search

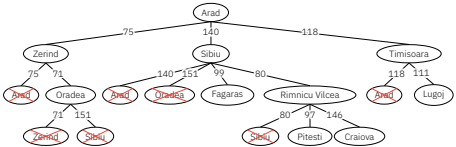


Frontier ordered by path cost (▶ indicates the next node for expansion)

	Path cost	Path leads to the leaf node
▶	220	Arad – Sibiu – Rimnicu Vilcea
	229	Arad – Timisoara – Lugoj
	239	Arad – Sibiu – Fagaras

Arad – Zerind – Oradea – Sibiu (297) is removed as Sibiu was expanded in a different path; therefore the expanded Sibiu must be on a shorter path.

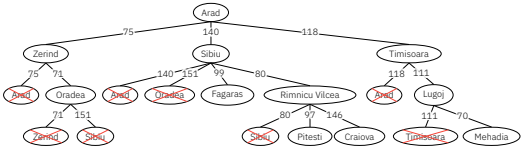
Uniform-cost search



Frontier ordered by path cost (► indicates the next node for expansion)

	Path cost	Path leads to the leaf node
►	229	Arad – Timisoara – Lugoj
	239	Arad – Sibiu – Fagaras
	317	Arad – Sibiu – Rimnicu Vilcea – Pitesti
	366	Arad – Sibiu – Rimnicu Vilcea – Craiova

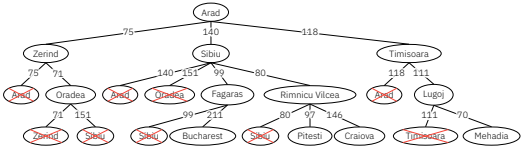
Uniform-cost search



Frontier ordered by path cost (► indicates the next node for expansion)

	Path cost	Path leads to the leaf node
►	239	Arad – Sibiu – Fagaras
	299	Arad – Timisoara – Lugoj – Mehadia
	317	Arad – Sibiu – Rimnicu Vilcea – Pitesti
	366	Arad – Sibiu – Rimnicu Vilcea – Craiova

Uniform-cost search

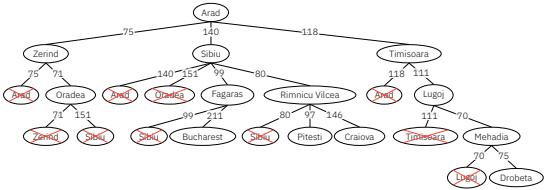


Frontier ordered by path cost (▶ indicates the next node for expansion)

	Path cost	Path leads to the leaf node
▶	299	Arad – Timisoara – Lugoj – Mehadia
	317	Arad – Sibiu – Rimnicu Vilcea – Pitesti
	366	Arad – Sibiu – Rimnicu Vilcea – Craiova
	450	Arad – Sibiu – Fagaras – Bucharest

Note that the algorithm is not stopped despite a goal node is generated.

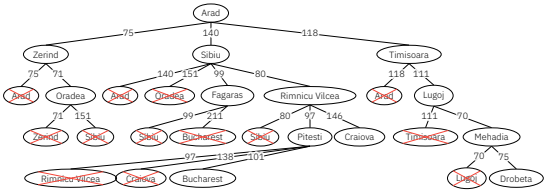
Uniform-cost search



Frontier ordered by path cost (► indicates the next node for expansion)

	Path cost	Path leads to the leaf node
►	317	Arad – Sibiu – Rimnicu Vilcea – Pitesti
	366	Arad – Sibiu – Rimnicu Vilcea – Craiova
	374	Arad – Timisoara – Lugoj – Mehadia – Drobeta
	450	Arad – Sibiu – Fagaras – Bucharest

Uniform-cost search

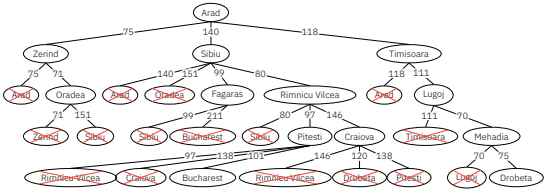


Frontier ordered by path cost (▶ indicates the next node for expansion)

	Path cost	Path leads to the leaf node
▶	366	Arad – Sibiu – Rimnicu Vilcea – Craiova
	374	Arad – Timisoara – Lugoj – Mehadia – Drobeta
	418	Arad – Sibiu – Rimnicu Vilcea – Pitesti – Bucharest

The new Craiova node (455) is removed in favour of the old Craiova node (366). The old Bucharest node (450) is removed in favour of the new Bucharest node (418).

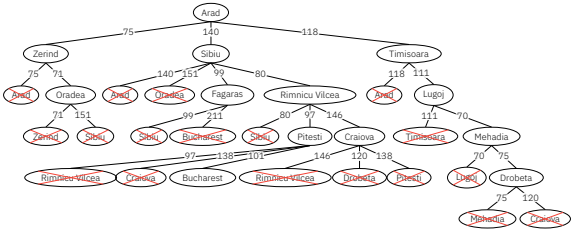
Uniform-cost search



Frontier ordered by path cost (▶ indicates the next node for expansion)

	Path cost	Path leads to the leaf node
▶	374	Arad – Timisoara – Lugoj – Mehadia – Drobeta
	418	Arad – Sibiu – Rimnicu Vilcea – Pitesti – Bucharest

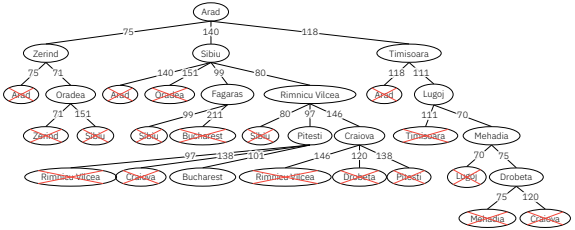
Uniform-cost search



Frontier ordered by path cost (▶ indicates the next node for expansion)

	Path cost	Path leads to the leaf node
▶	418	Arad – Sibiu – Rimnicu Vilcea – Pitesti – Bucharest

Uniform-cost search



Frontier ordered by path cost (► indicates the next node for expansion)

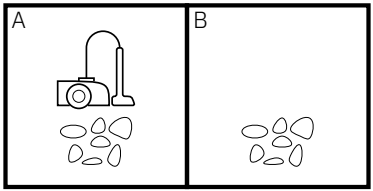
	Path cost	Path leads to the leaf node
►	418	Arad – Sibiu – Rimnicu Vilcea – Pitesti – Bucharest

As this path is selected for expansion, it is found that the leaf node to be expand is the goal node. Therefore the algorithm is stopped and this path is returned as the solution.

Uniform-cost search

Vacuum world

Implement UCS to find the solution to clean the two rooms in the vacuum world if moving left consumes 5 kJ of energy, moving right consumes 8 kJ of energy, and sucking the dirt consumes 15 kJ.



How about constraints?

Lecture Outline

Uninformed search

Breadth-first search · Depth first search · Uniform-cost search