Problem solving by search

Tomáš Svoboda and Matěj Hoffmann

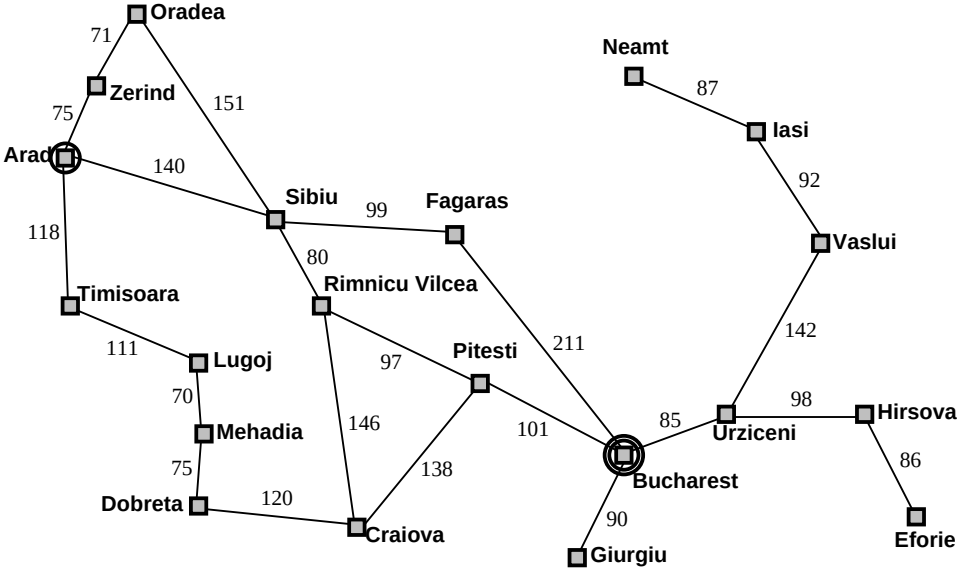
Vision for Robots and Autonomous Systems, Center for Machine Perception
Department of Cybernetics
Faculty of Electrical Engineering, Czech Technical University in Prague

May 23, 2022

Outline

- Search problem.
- State space graphs.
- Search trees.
- Strategies: which tree branches to choose?
- Strategy/Algorithm properties.
- Programming infrastructure.

Example: Traveling in Romania



Example: Map of Romania

Goal:

be in Bucharest

Problem formulation:

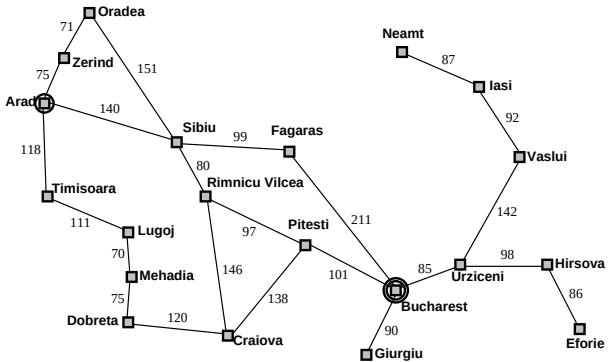
states: position in a city (cities)

actions: drive between cities

Solution:

Sequence of cities (path)

(action sequence [2])



Example: Map of Romania

Goal:

be in Bucharest

Problem formulation:

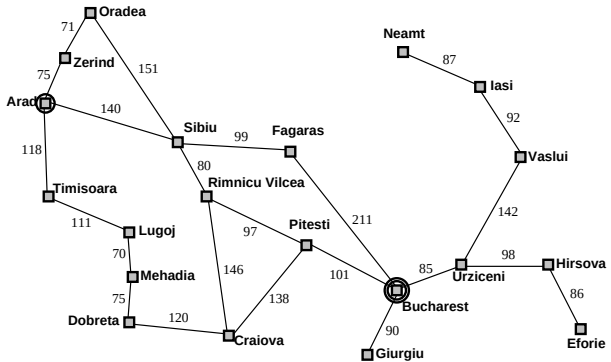
states: position in a city (cities)

actions: drive between cities

Solution:

Sequence of cities (path)

(action sequence [2])



Example: Map of Romania

Goal:

be in Bucharest

Problem formulation:

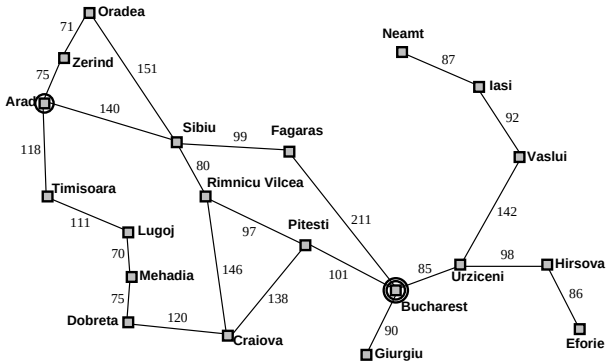
states: position in a city (cities)

actions: drive between cities

Solution:

Sequence of cities (path)

(action sequence [2])



Example: Map of Romania

Goal:

be in Bucharest

Problem formulation:

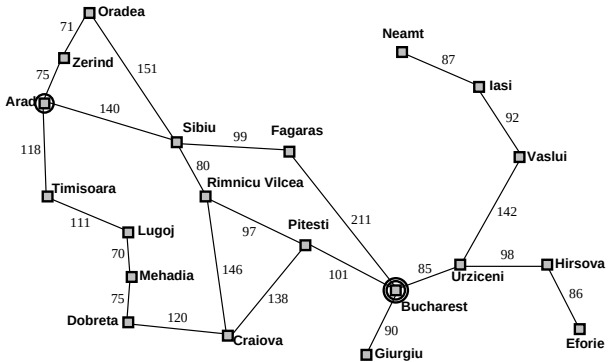
states: position in a city (cities)

actions: drive between cities

Solution:

Sequence of cities (path)

(action sequence [2])



Example: Map of Romania

Goal:

be in Bucharest

Problem formulation:

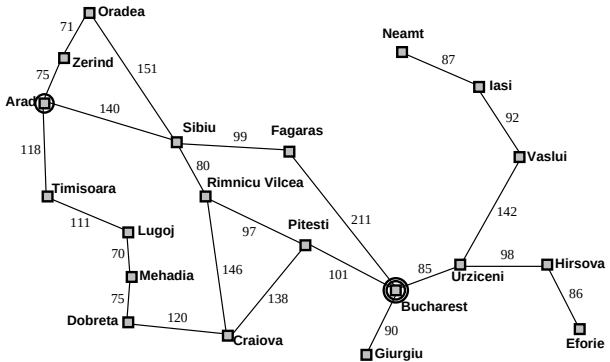
states: position in a city (cities)

actions: drive between cities

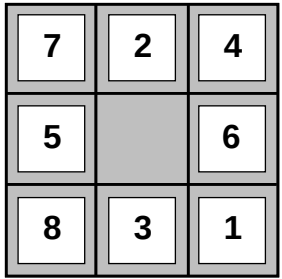
Solution:

Sequence of cities (path)

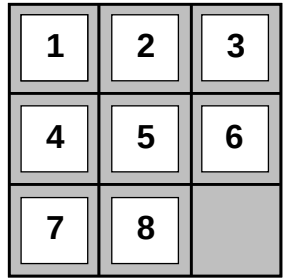
(action sequence [2])



Example: The 8-puzzle



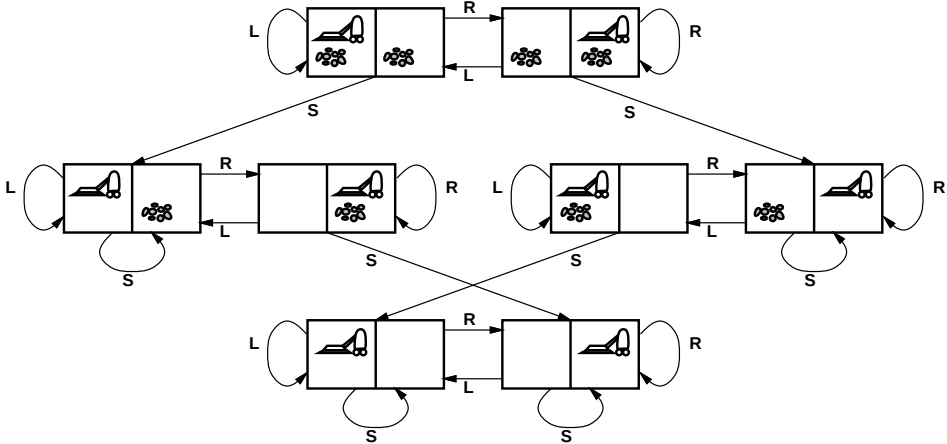
Start State



Goal State

states?
actions?
solution?
cost?

Example: Vacuum cleaner



states?
actions?
solution?
cost?

A Search Problem

- State space (including Start/Initial state): position, board configuration,
- Actions : drive to, Up, Down, Left ...
- Transition model : Given state and action return state (and cost)
- Goal test : Are we done?

A Search Problem

- State space (including Start/Initial state): position, board configuration,
- Actions : drive to, Up, Down, Left ...
- Transition model : Given state and action return state (and cost)
- Goal test : Are we done?

A Search Problem

- State space (including Start/Initial state): position, board configuration,
- Actions : drive to, Up, Down, Left ...
- Transition model : Given state and action return state (and cost)
- Goal test : Are we done?

A Search Problem

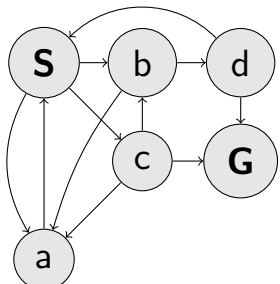
- State space (including Start/Initial state): position, board configuration,
- Actions : drive to, Up, Down, Left ...
- Transition model : Given state and action return state (and cost)
- Goal test : Are we done?

State Space Graphs

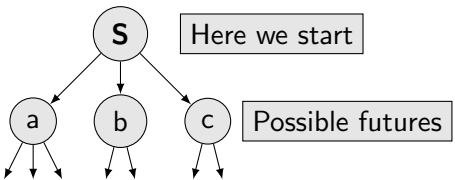
State space graph: a representation of a search problem

- Graph Nodes – states – are abstracted world configurations
- Arcs represent action results
- Goal test – a set of goal nodes

Each state occurs only *once* in a state (search) space.

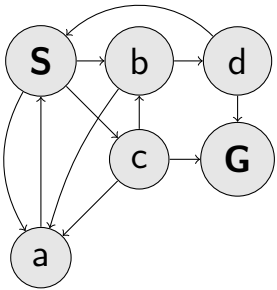


Search Trees



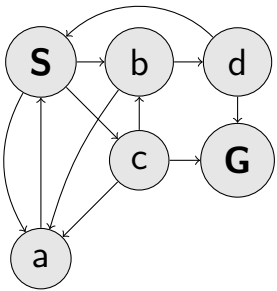
- A "what if" tree of plans and their outcomes
- Start node is the root
- Children are successors
- Nodes show/contains states, but correspond to *plans* that achieve those states

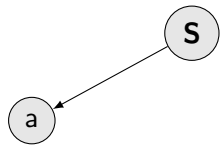
State Space Graphs vs. Search Trees



How big is the search tree?

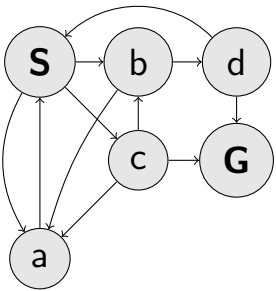
State Space Graphs vs. Search Trees

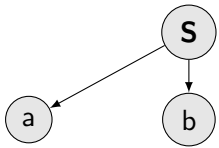




How big is the search tree?

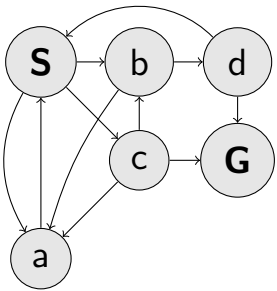
State Space Graphs vs. Search Trees

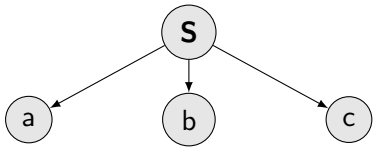




How big is the search tree?

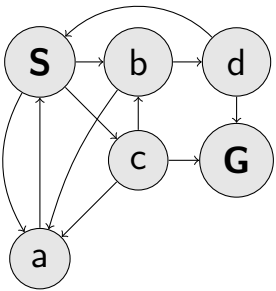
State Space Graphs vs. Search Trees

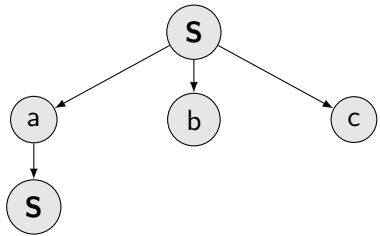




How big is the search tree?

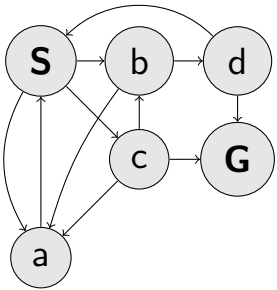
State Space Graphs vs. Search Trees

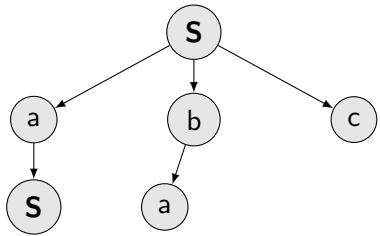




How big is the search tree?

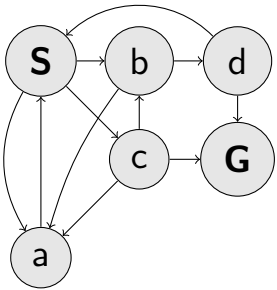
State Space Graphs vs. Search Trees

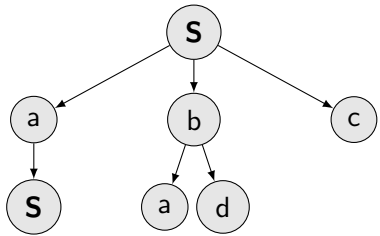




How big is the search tree?

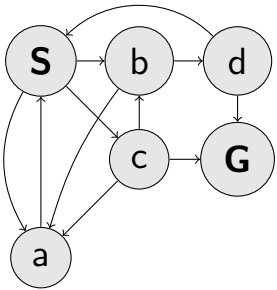
State Space Graphs vs. Search Trees

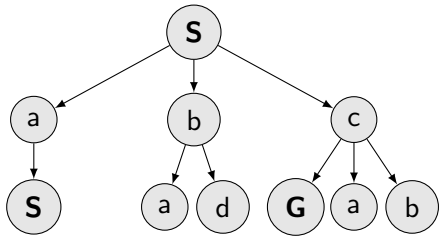




How big is the search tree?

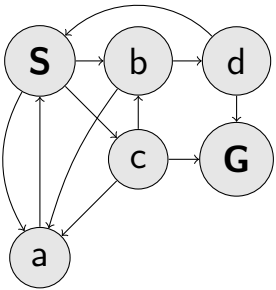
State Space Graphs vs. Search Trees

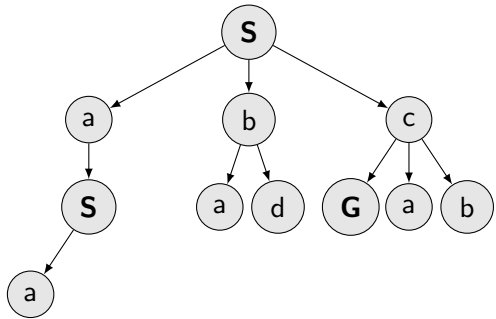




How big is the search tree?

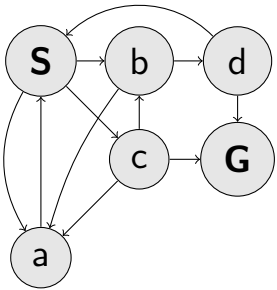
State Space Graphs vs. Search Trees

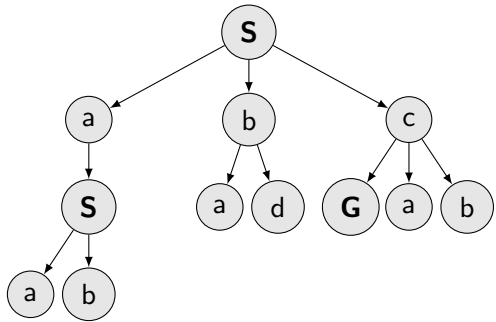




How big is the search tree?

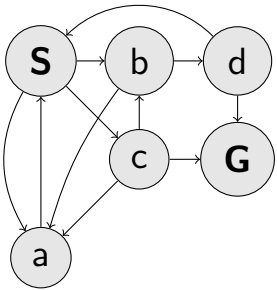
State Space Graphs vs. Search Trees

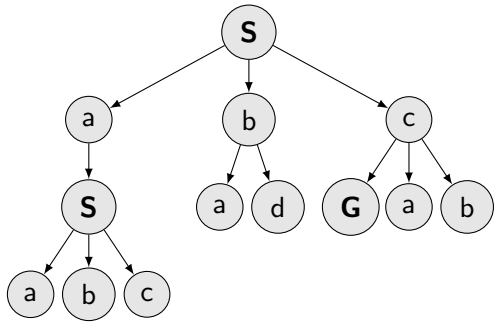




How big is the search tree?

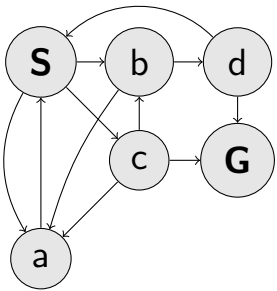
State Space Graphs vs. Search Trees

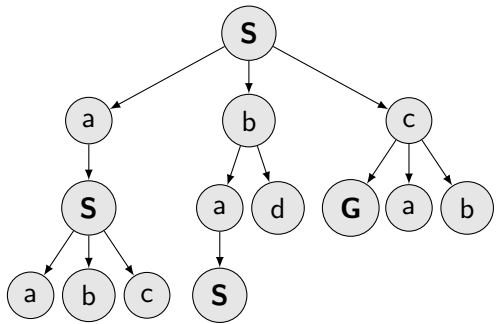




How big is the search tree?

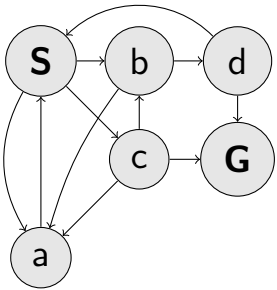
State Space Graphs vs. Search Trees

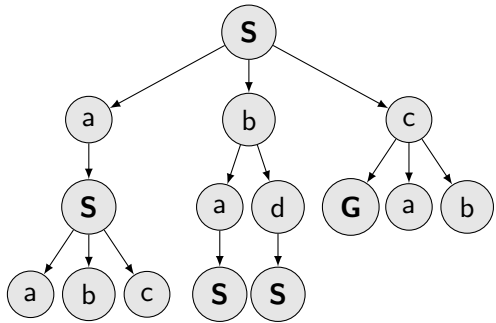




How big is the search tree?

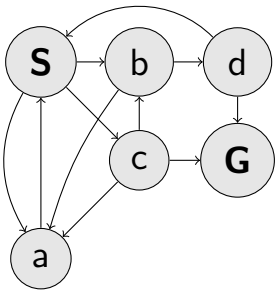
State Space Graphs vs. Search Trees

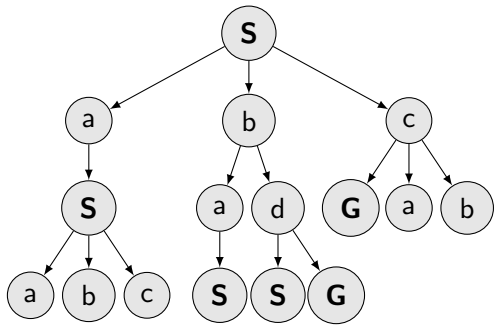




How big is the search tree?

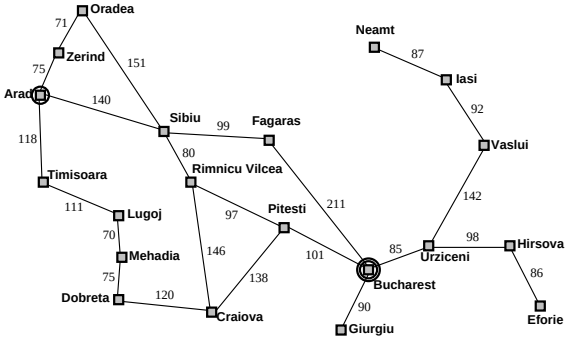
State Space Graphs vs. Search Trees

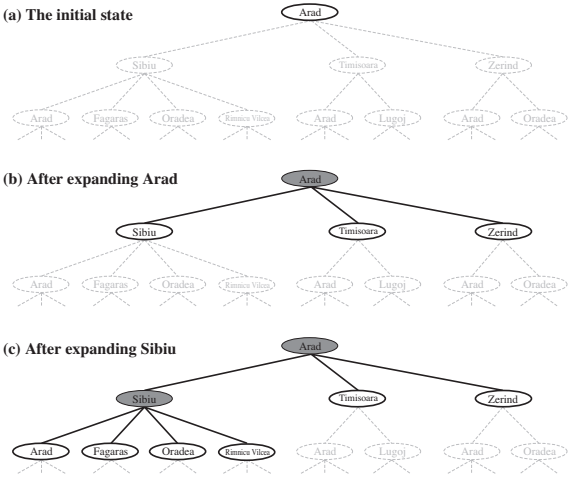




How big is the search tree?

From problem/transition graph to search tree (Romania)

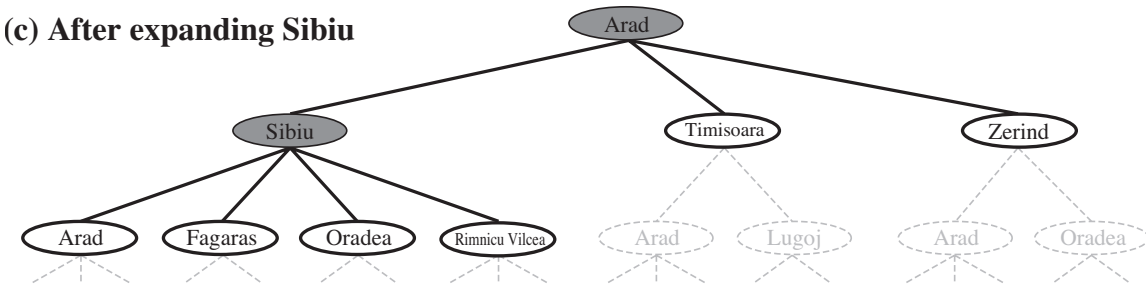




Problem/transition graph is revealed incrementally.

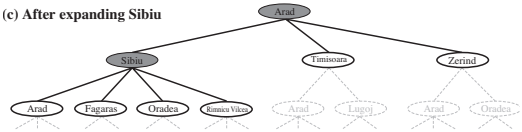
The revealing strategy can be visualized as a search tree.

Search elements - unvisited, dead, alive states



- Expand plans - possible ways (tree nodes).
- Manage/Maintain fringe (or frontier) of plans under consideration.
- Expand new nodes *wisely(?).*

Tree search algorithm



function TREE_SEARCH(problem) **return** a solution or failure

initialize by using the initial state of the problem

loop

if no candidates for expansion **then return** failure

else choose a leaf node for expansion

end if

if the node contains a goal state **then return** the solution

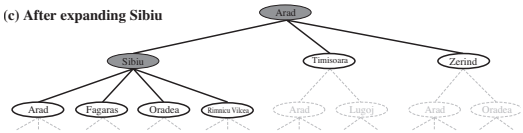
end if

Expand the node and add the resulting nodes to the tree

end loop

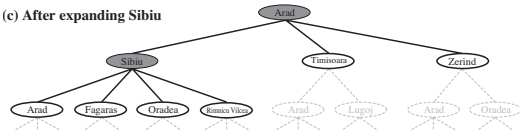
end function

Tree search algorithm



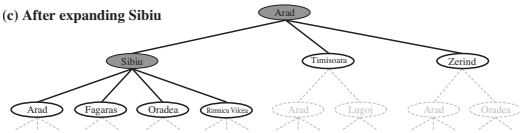
function TREE_SEARCH(problem) **return** a solution or failure
initialize by using the initial state of the problem
loop
if no candidates for expansion **then return** failure
else choose a leaf node for expansion
end if
if the node contains a goal state **then return** the solution
end if
Expand the node and add the resulting nodes to the tree
end loop
end function

Tree search algorithm



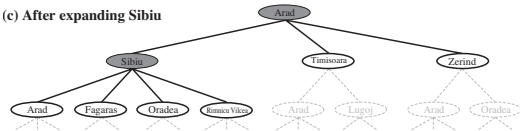
function TREE_SEARCH(problem) **return** a solution or failure
  initialize by using the initial state of the problem
  loop
    if no candidates for expansion **then return** failure
    else choose a leaf node for expansion
    end if
    if the node contains a goal state **then return** the solution
    end if
    Expand the node and add the resulting nodes to the tree
  end loop
end function

Tree search algorithm



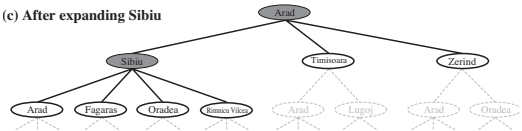
function TREE_SEARCH(problem) **return** a solution or failure
  initialize by using the initial state of the problem
  loop
    if no candidates for expansion **then return** failure
    else choose a leaf node for expansion
    end if
    if the node contains a goal state **then return** the solution
    end if
    Expand the node and add the resulting nodes to the tree
  end loop
end function

Tree search algorithm



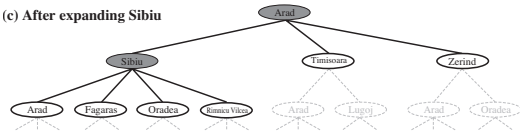
function TREE_SEARCH(problem) **return** a solution or failure
  initialize by using the initial state of the problem
  loop
    if no candidates for expansion **then return** failure
    else choose a leaf node for expansion
    end if
    if the node contains a goal state **then return** the solution
    end if
    Expand the node and add the resulting nodes to the tree
  end loop
end function

Tree search algorithm



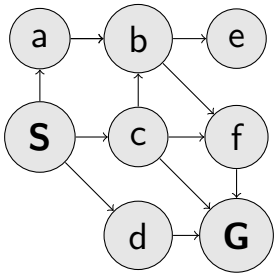
function TREE_SEARCH(problem) **return** a solution or failure
  initialize by using the initial state of the problem
  loop
    if no candidates for expansion **then return** failure
    else choose a leaf node for expansion
    end if
    if the node contains a goal state **then return** the solution
    end if
    Expand the node and add the resulting nodes to the tree
  end loop
end function

Tree search algorithm



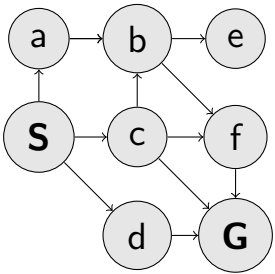
function TREE_SEARCH(problem) **return** a solution or failure
    initialize by using the initial state of the problem
    loop
        if no candidates for expansion **then return** failure
        else choose a leaf node for expansion
        end if
        if the node contains a goal state **then return** the solution
        end if
        Expand the node and add the resulting nodes to the tree
    end loop
end function

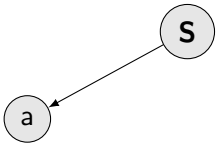
Example of a tree search



Which nodes to *explore*?
What are the properties of a strategy/algorithm?

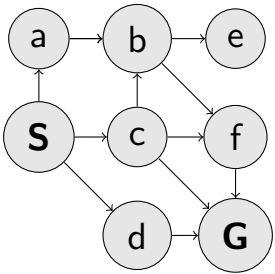
Example of a tree search

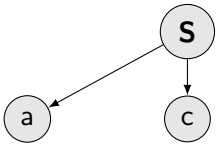




Which nodes to *explore*?
What are the properties of a strategy/algorithm?

Example of a tree search

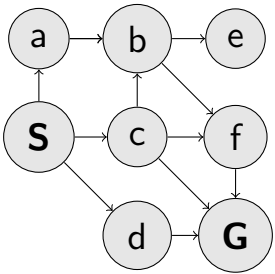


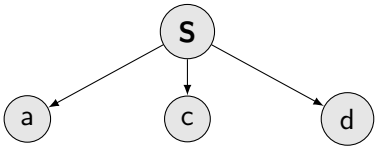


Which nodes to *explore*?

What are the properties of a strategy/algorithm?

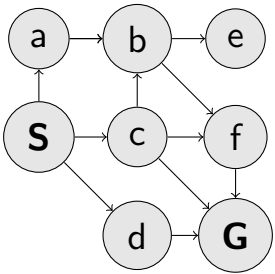
Example of a tree search

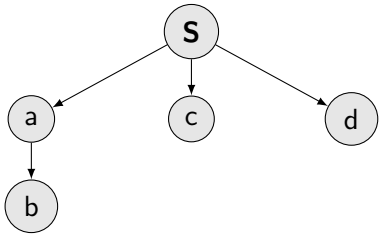




Which nodes to *explore*?
What are the properties of a strategy/algorithm?

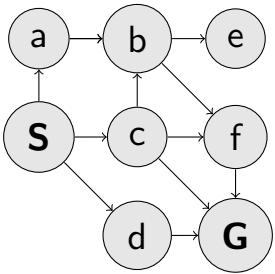
Example of a tree search

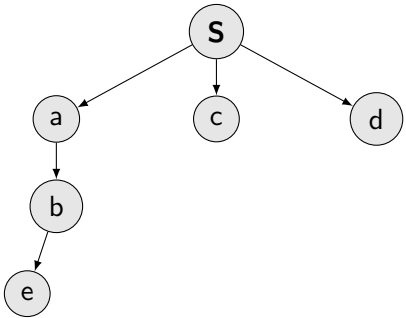




Which nodes to *explore*?
What are the properties of a strategy/algorithm?

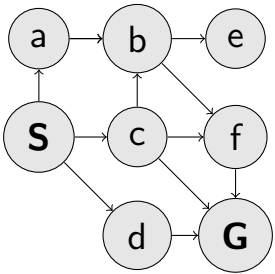
Example of a tree search

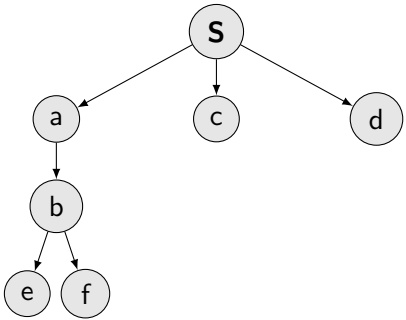




Which nodes to *explore*?
What are the properties of a strategy/algorithm?

Example of a tree search

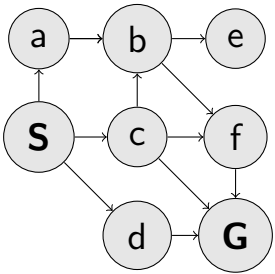


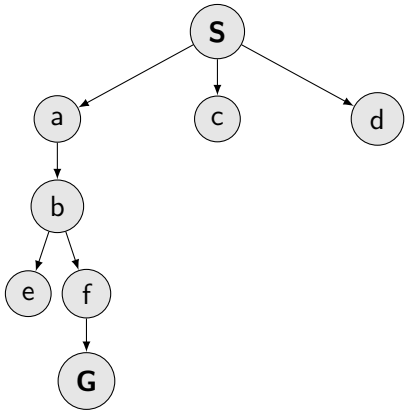


Which nodes to *explore*?

What are the properties of a strategy/algorithm?

Example of a tree search

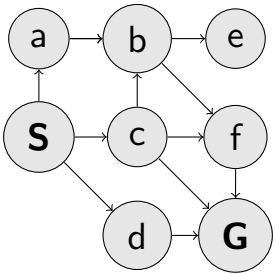


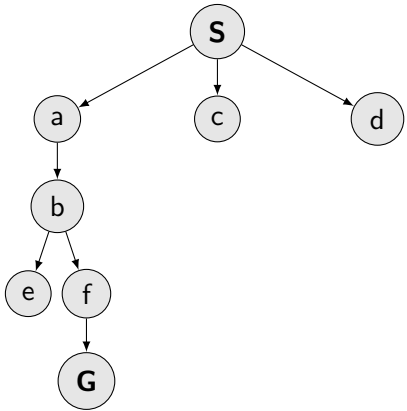


Which nodes to *explore*?

What are the properties of a strategy/algorithm?

Example of a tree search

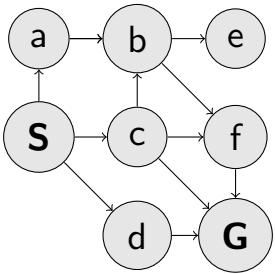


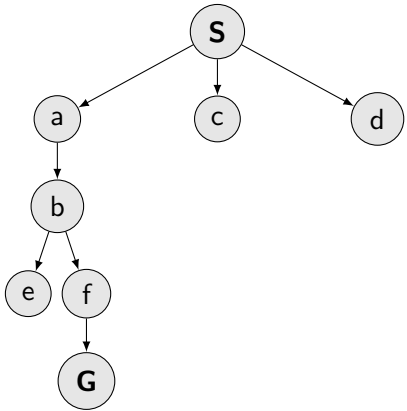


Which nodes to *explore*?

What are the properties of a strategy/algorithm?

Example of a tree search





Which nodes to *explore*?
What are the properties of a strategy/algorithm?

Search (algorithm) properties

- Guaranteed to find a solution (if exists)? Complete?
- Guaranteed to find the least cost path? Optimal?
- How many steps - an operation with a node? Time complexity?
- How many nodes to remember? Space/Memory complexity?

How many nodes in a (search) tree? What are tree parameters?

Search (algorithm) properties

- Guaranteed to find a solution (if exists)? Complete?
- Guaranteed to find the least cost path? Optimal?
- How many steps - an operation with a node? Time complexity?
- How many nodes to remember? Space/Memory complexity?

How many nodes in a (search) tree? What are tree parameters?

Search (algorithm) properties

- Guaranteed to find a solution (if exists)? Complete?
- Guaranteed to find the least cost path? Optimal?
- How many steps - an operation with a node? Time complexity?
- How many nodes to remember? Space/Memory complexity?

How many nodes in a (search) tree? What are tree parameters?

Search (algorithm) properties

- Guaranteed to find a solution (if exists)? Complete?
- Guaranteed to find the least cost path? Optimal?
- How many steps - an operation with a node? Time complexity?
- How many nodes to remember? Space/Memory complexity?

How many nodes in a (search) tree? What are tree parameters?

Search (algorithm) properties

- Guaranteed to find a solution (if exists)? Complete?
- Guaranteed to find the least cost path? Optimal?
- How many steps - an operation with a node? Time complexity?
- How many nodes to remember? Space/Memory complexity?

How many nodes in a (search) tree? What are tree parameters?

Search (algorithm) properties

- Guaranteed to find a solution (if exists)? Complete?
- Guaranteed to find the least cost path? Optimal?
- How many steps - an operation with a node? Time complexity?
- How many nodes to remember? Space/Memory complexity?

How many nodes in a (search) tree? What are tree parameters?

Search (algorithm) properties

- Guaranteed to find a solution (if exists)? Complete?
- Guaranteed to find the least cost path? Optimal?
- How many steps - an operation with a node? Time complexity?
- How many nodes to remember? Space/Memory complexity?

How many nodes in a (search) tree? What are tree parameters?

Search (algorithm) properties

- Guaranteed to find a solution (if exists)? Complete?
- Guaranteed to find the least cost path? Optimal?
- How many steps - an operation with a node? Time complexity?
- How many nodes to remember? Space/Memory complexity?

How many nodes in a (search) tree? What are tree parameters?

Search (algorithm) properties

- Guaranteed to find a solution (if exists)? Complete?
- Guaranteed to find the least cost path? Optimal?
- How many steps - an operation with a node? Time complexity?
- How many nodes to remember? Space/Memory complexity?

How many nodes in a (search) tree? What are tree parameters?

Search (algorithm) properties

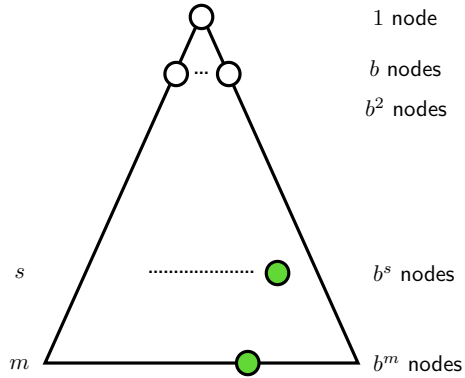
- Guaranteed to find a solution (if exists)? Complete?
- Guaranteed to find the least cost path? Optimal?
- How many steps - an operation with a node? Time complexity?
- How many nodes to remember? Space/Memory complexity?

How many nodes in a (search) tree? What are tree parameters?

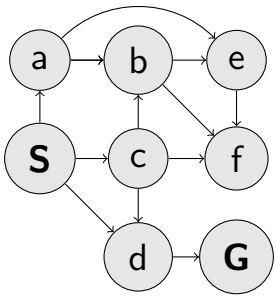
Strategies

How to traverse/build a search tree?

- Depth of the tree d.
- Max-Depth of the tree m. Can be ∞.
- Branching factor b.
- s denotes the shallowest Goal .
- How many nodes in the whole tree?

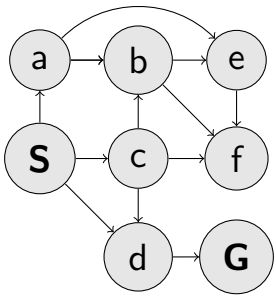


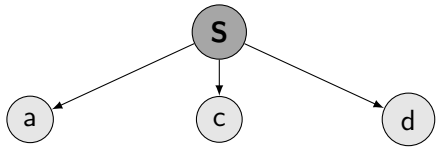
Depth-First Search (DFS)



What are the DFS properties (complete, optimal, time, space)?

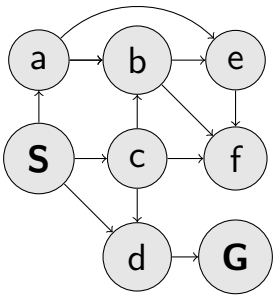
Depth-First Search (DFS)

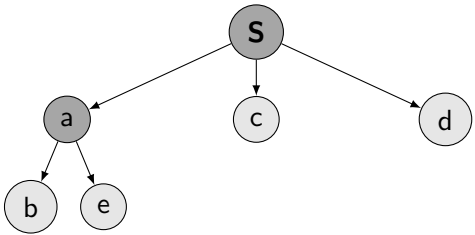




What are the DFS properties (complete, optimal, time, space)?

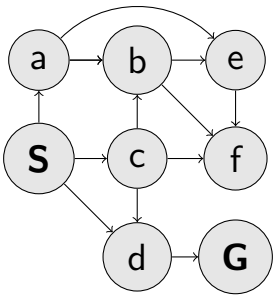
Depth-First Search (DFS)

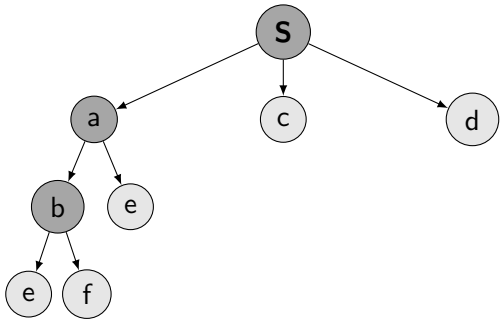




What are the DFS properties (complete, optimal, time, space)?

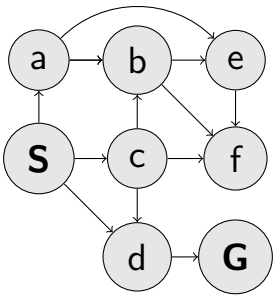
Depth-First Search (DFS)

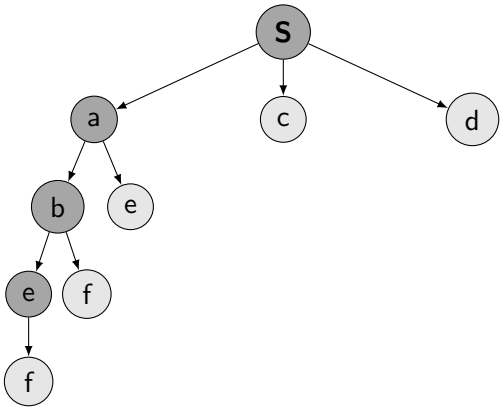




What are the DFS properties (complete, optimal, time, space)?

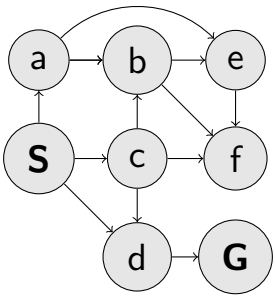
Depth-First Search (DFS)

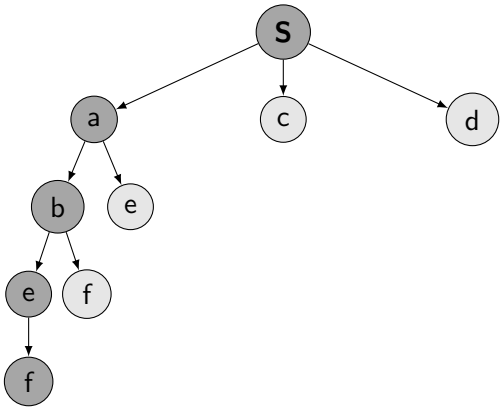




What are the DFS properties (complete, optimal, time, space)?

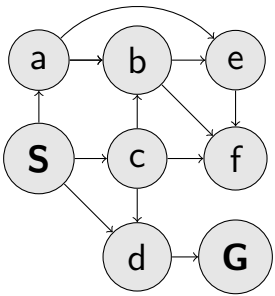
Depth-First Search (DFS)

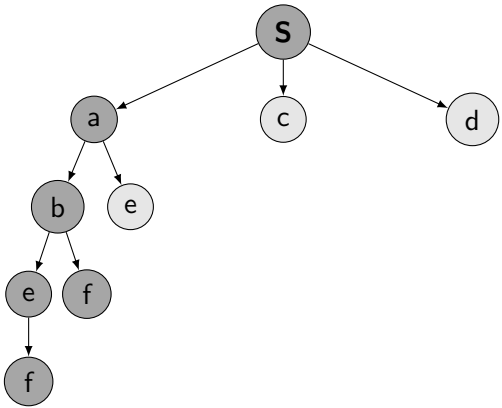




What are the DFS properties (complete, optimal, time, space)?

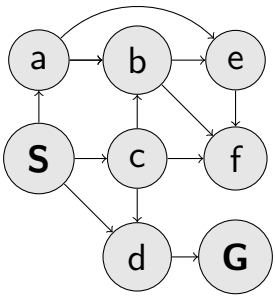
Depth-First Search (DFS)

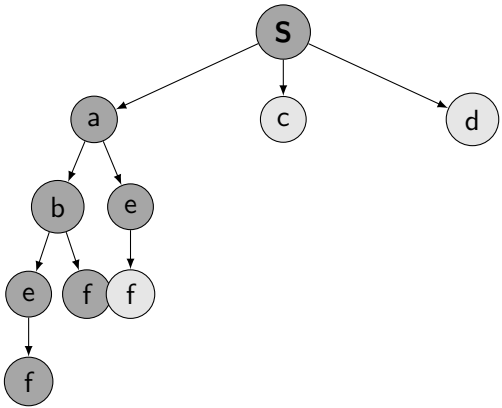




What are the DFS properties (complete, optimal, time, space)?

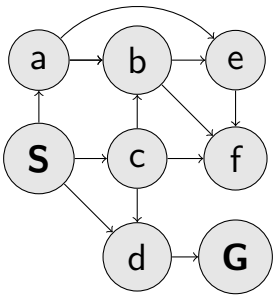
Depth-First Search (DFS)

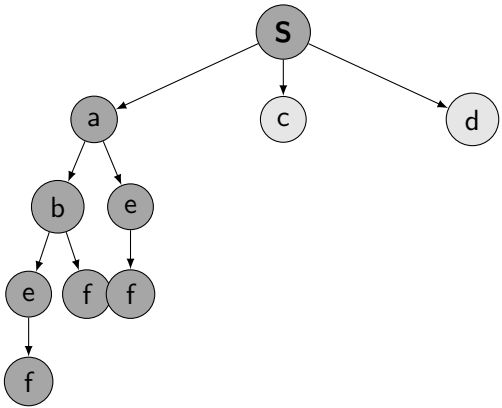




What are the DFS properties (complete, optimal, time, space)?

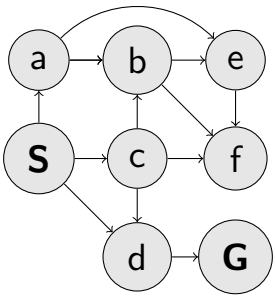
Depth-First Search (DFS)

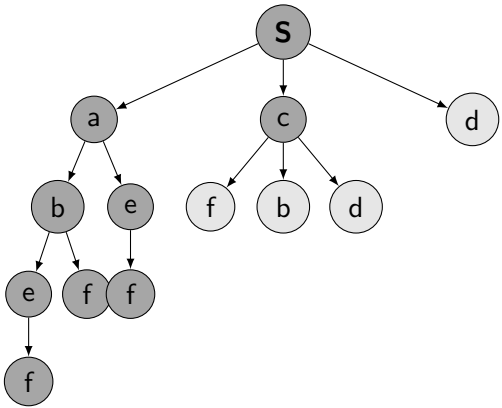




What are the DFS properties (complete, optimal, time, space)?

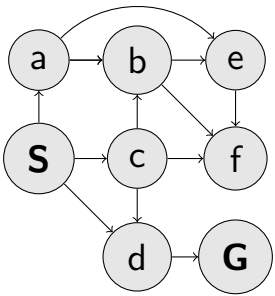
Depth-First Search (DFS)

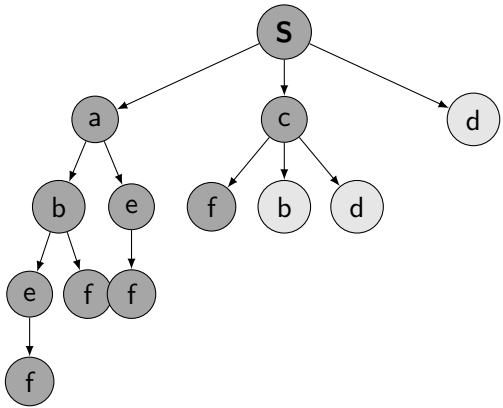




What are the DFS properties (complete, optimal, time, space)?

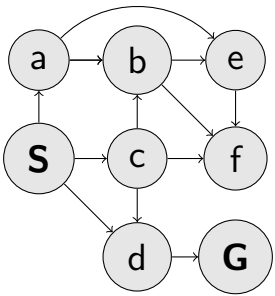
Depth-First Search (DFS)

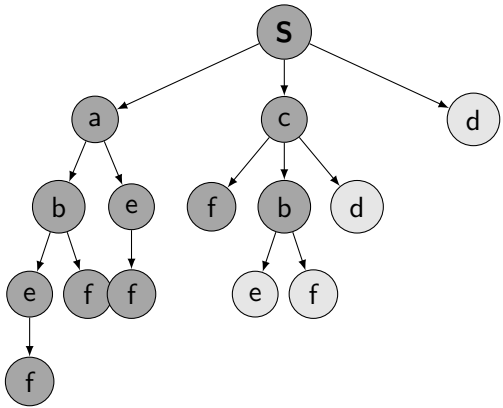




What are the DFS properties (complete, optimal, time, space)?

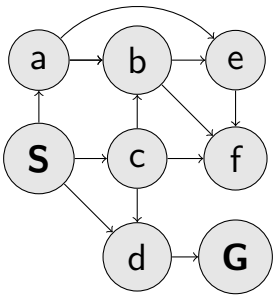
Depth-First Search (DFS)

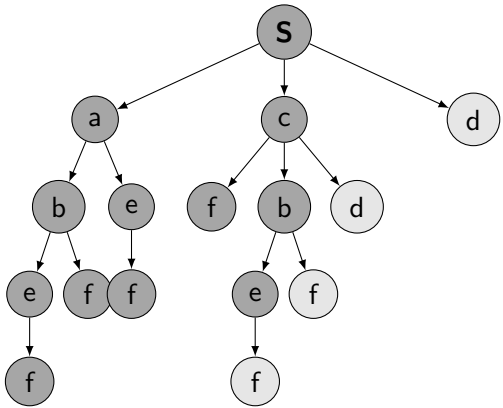




What are the DFS properties (complete, optimal, time, space)?

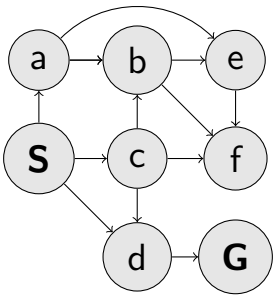
Depth-First Search (DFS)

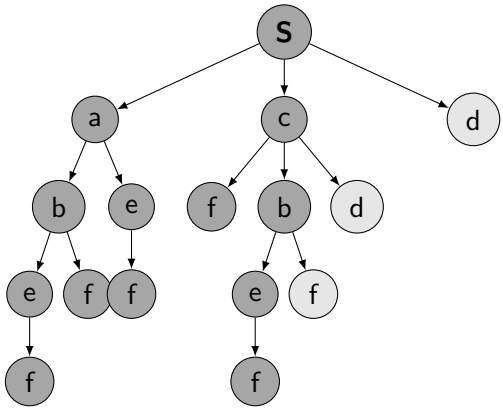




What are the DFS properties (complete, optimal, time, space)?

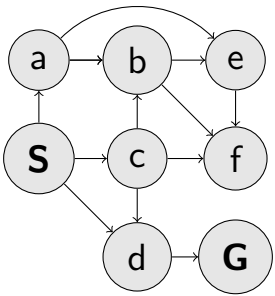
Depth-First Search (DFS)

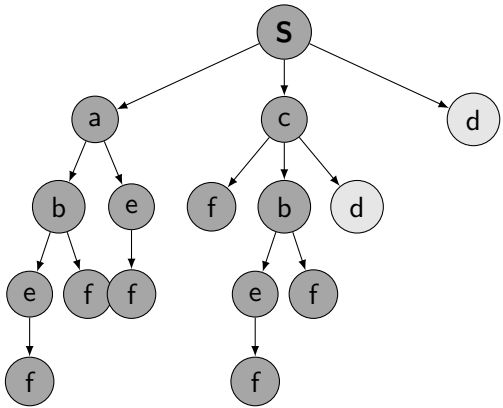




What are the DFS properties (complete, optimal, time, space)?

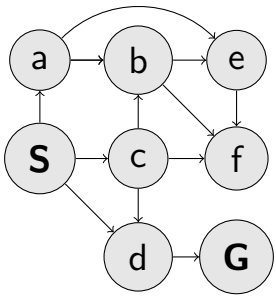
Depth-First Search (DFS)

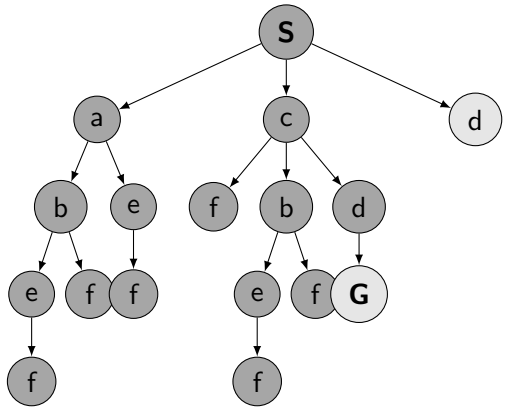




What are the DFS properties (complete, optimal, time, space)?

Depth-First Search (DFS)





What are the DFS properties (complete, optimal, time, space)?

DFS properties

- Time complexity?
- Space complexity?
- Complete?
- Optimal?

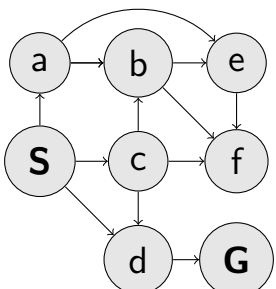
b, m, s, Time complexity?

A $\mathcal{O}(bm)$

B $\mathcal{O}(b^m)$

C $\mathcal{O}(m^b)$

D ∞



DFS properties

- Time complexity?
- Space complexity?
- Complete?
- Optimal?

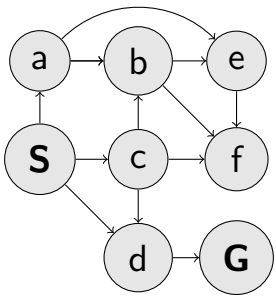
b, m, s, Space complexity?

A $\mathcal{O}(bm)$

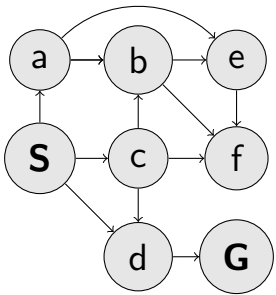
B $\mathcal{O}(b^m)$

C $\mathcal{O}(m^b)$

D ∞

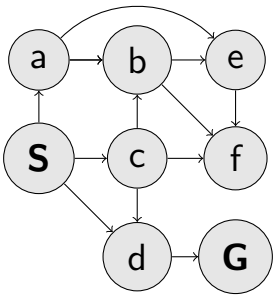


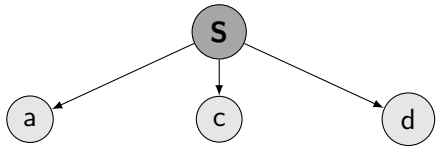
Breadth-First Search (BFS)



What are the BFS properties?

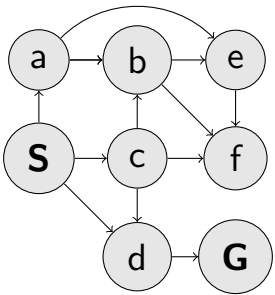
Breadth-First Search (BFS)

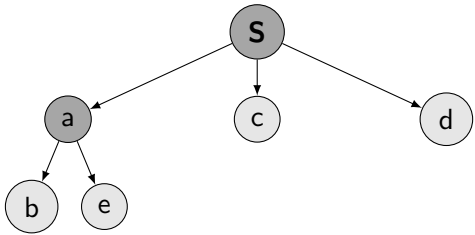




What are the BFS properties?

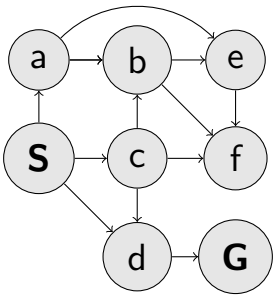
Breadth-First Search (BFS)

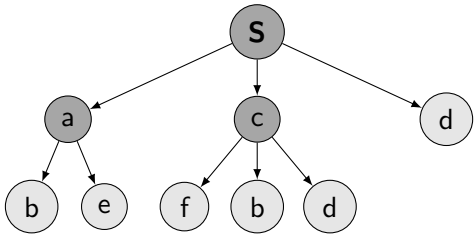




What are the BFS properties?

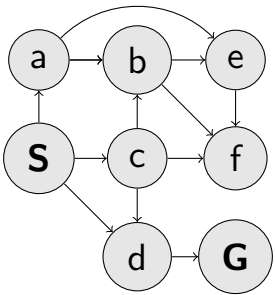
Breadth-First Search (BFS)

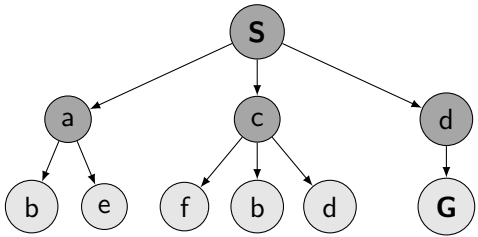




What are the BFS properties?

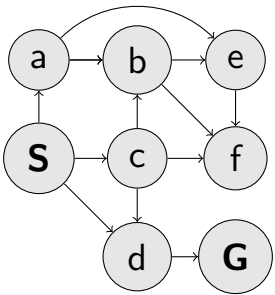
Breadth-First Search (BFS)

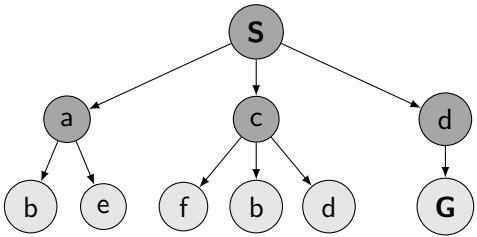




What are the BFS properties?

Breadth-First Search (BFS)





What are the BFS properties?

BFS properties

- Time complexity?
- Space complexity?
- Complete?
- Optimal?

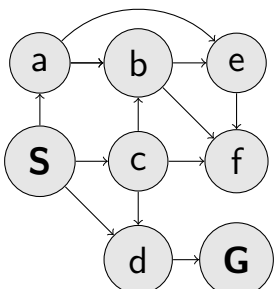
b, m, s, Time complexity?

A $\mathcal{O}(bm)$

B $\mathcal{O}(b^m)$

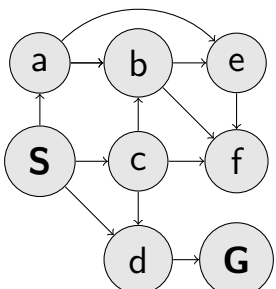
C $\mathcal{O}(m^b)$

D $\mathcal{O}(b^s)$



BFS properties

- Time complexity?
- Space complexity?
- Complete?
- Optimal?



b, m, s, Space complexity?

A $\mathcal{O}(bm)$

B $\mathcal{O}(b^m)$

C $\mathcal{O}(m^b)$

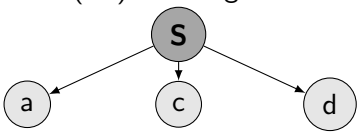
D $\mathcal{O}(b^s)$

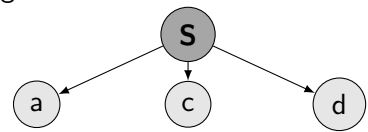
DFS vs BFS

What are (dis)advantages of the individual strategies?

DFS vs BFS

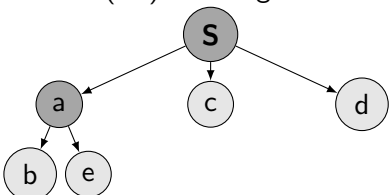
What are (dis)advantages of the individual strategies?

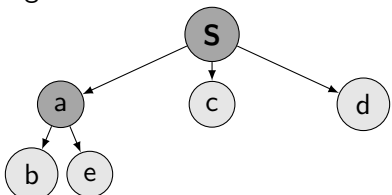




DFS vs BFS

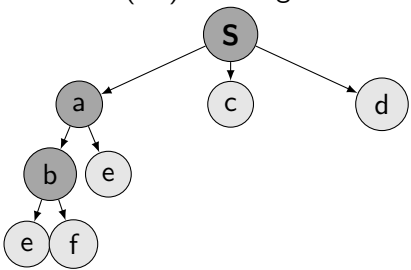
What are (dis)advantages of the individual strategies?

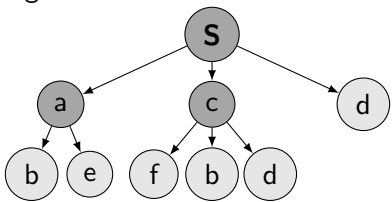




DFS vs BFS

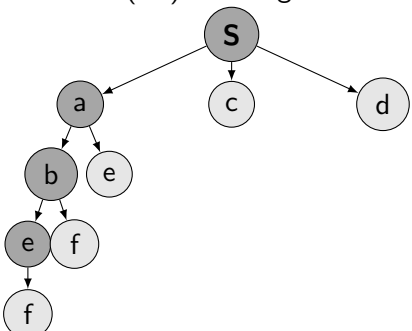
What are (dis)advantages of the individual strategies?

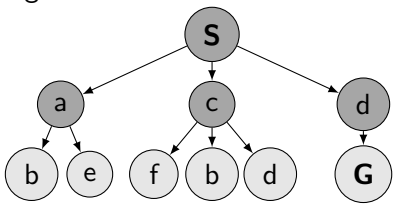




DFS vs BFS

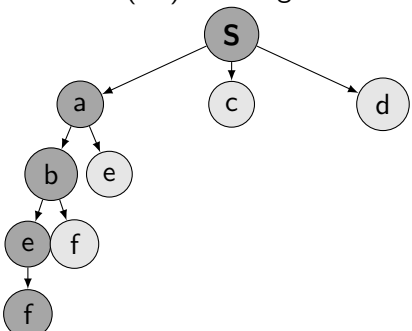
What are (dis)advantages of the individual strategies?

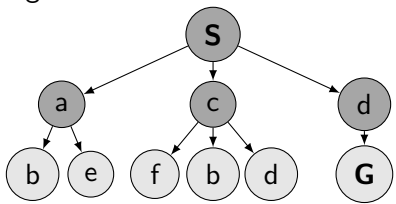




DFS vs BFS

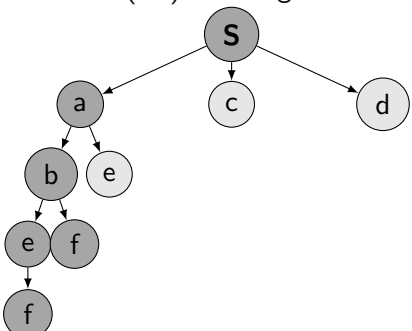
What are (dis)advantages of the individual strategies?

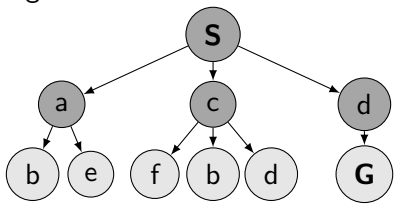




DFS vs BFS

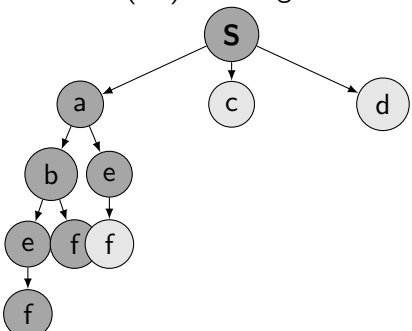
What are (dis)advantages of the individual strategies?

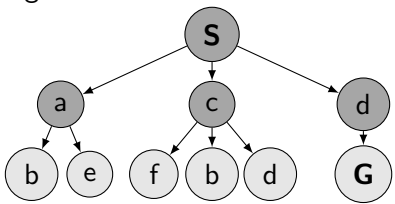




DFS vs BFS

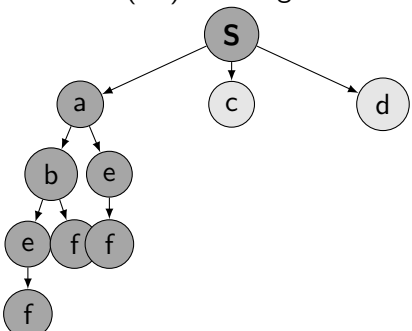
What are (dis)advantages of the individual strategies?

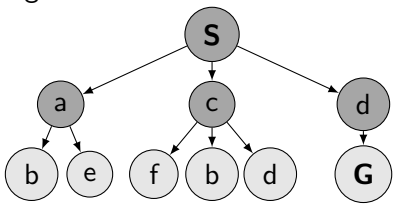




DFS vs BFS

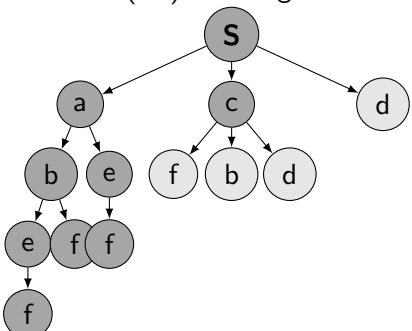
What are (dis)advantages of the individual strategies?

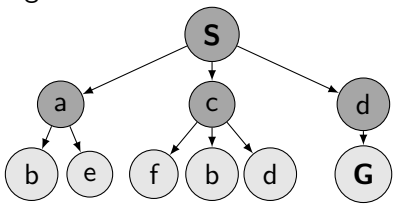




DFS vs BFS

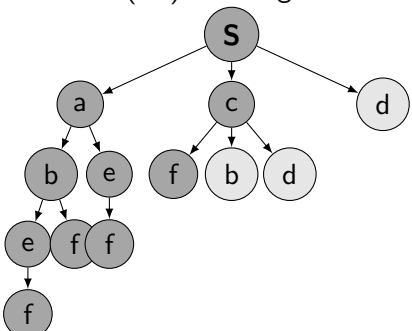
What are (dis)advantages of the individual strategies?

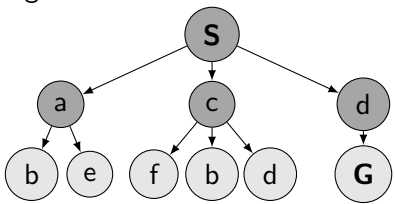




DFS vs BFS

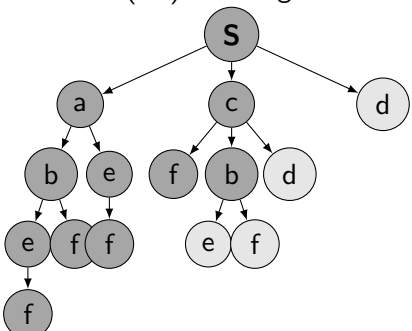
What are (dis)advantages of the individual strategies?

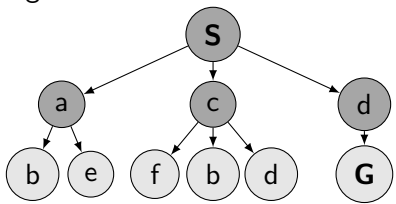




DFS vs BFS

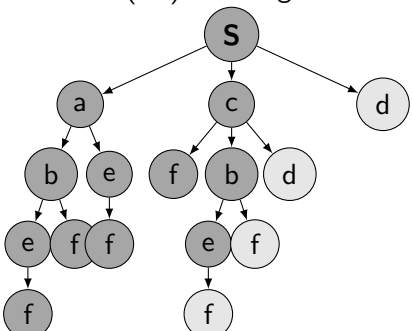
What are (dis)advantages of the individual strategies?

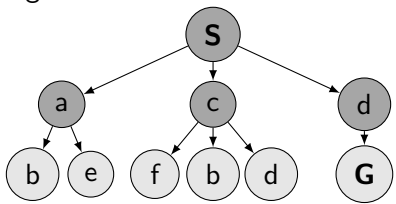




DFS vs BFS

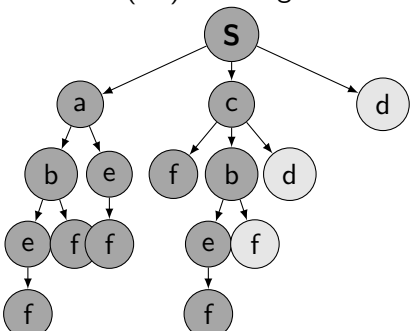
What are (dis)advantages of the individual strategies?

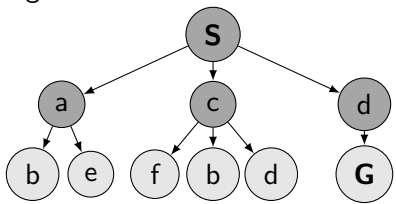




DFS vs BFS

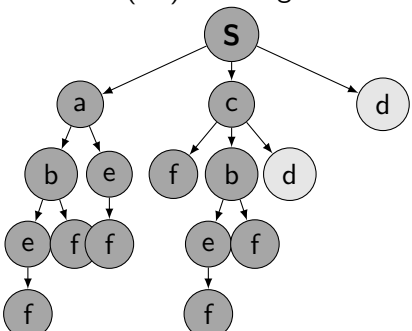
What are (dis)advantages of the individual strategies?

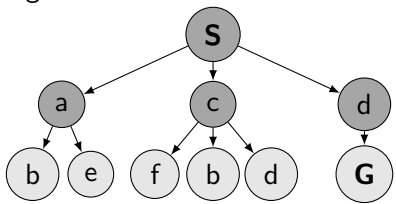




DFS vs BFS

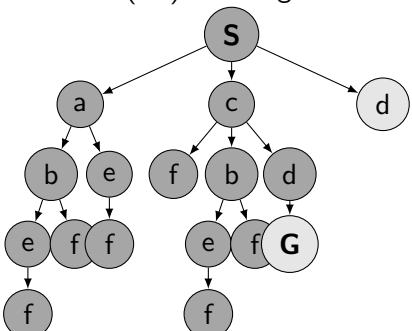
What are (dis)advantages of the individual strategies?

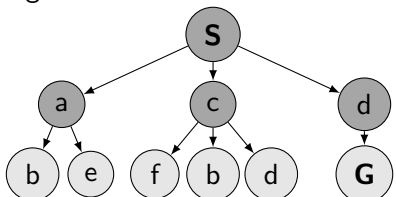




DFS vs BFS

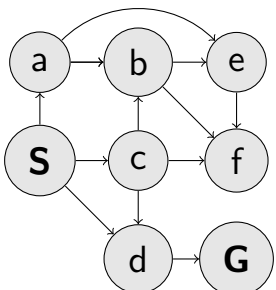
What are (dis)advantages of the individual strategies?





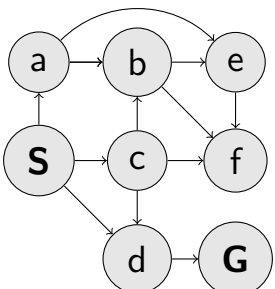
DFS with limited depth, `maxdepth=2`

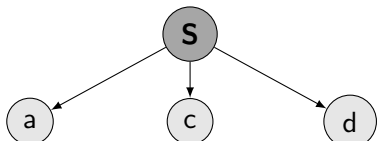
Do not follow nodes with `depth` $>$ `maxdepth`



DFS with limited depth, `maxdepth=2`

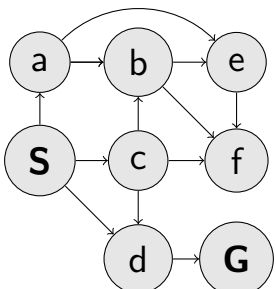
Do not follow nodes with `depth` > `maxdepth`

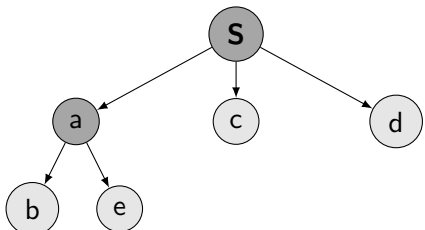




DFS with limited depth, `maxdepth=2`

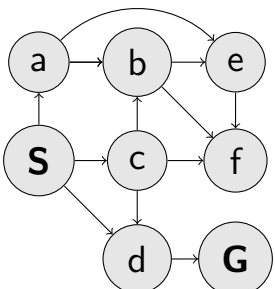
Do not follow nodes with `depth` > `maxdepth`

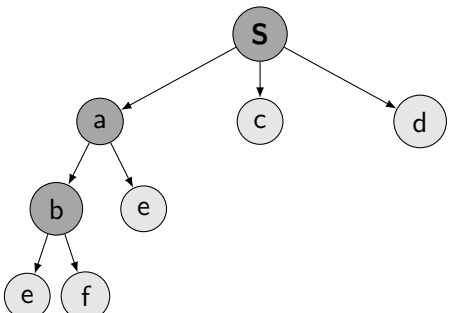




DFS with limited depth, `maxdepth=2`

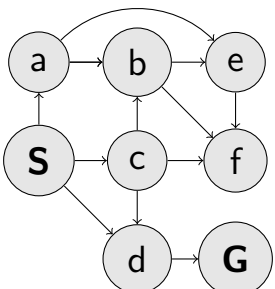
Do not follow nodes with `depth` > `maxdepth`

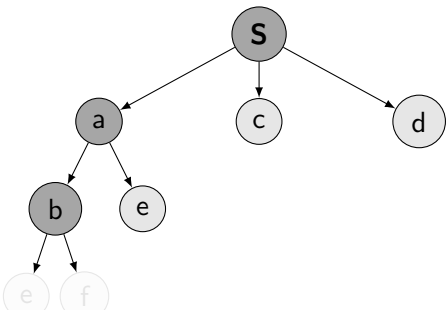




DFS with limited depth, `maxdepth=2`

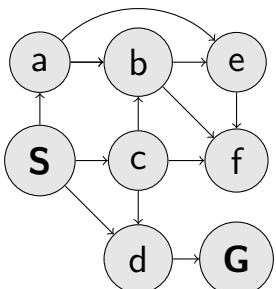
Do not follow nodes with `depth` > `maxdepth`

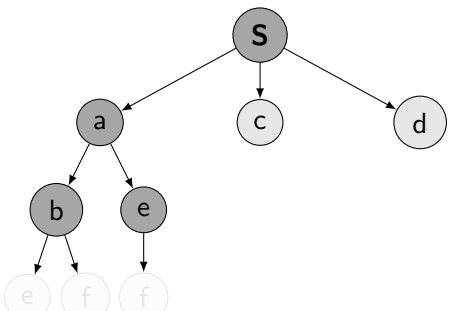




DFS with limited depth, `maxdepth=2`

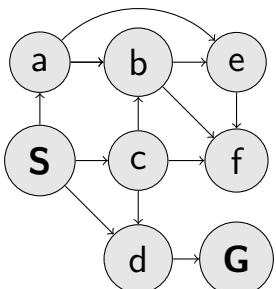
Do not follow nodes with `depth` > `maxdepth`

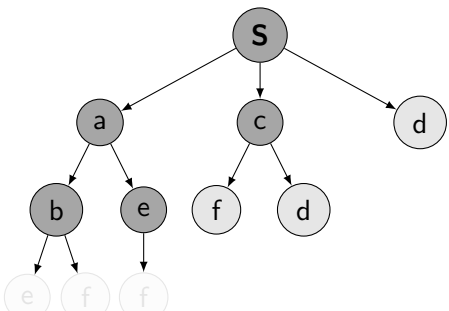




DFS with limited depth, `maxdepth=2`

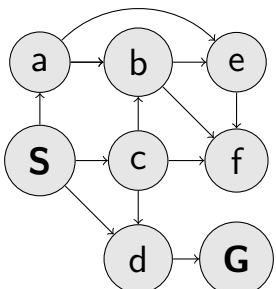
Do not follow nodes with `depth` > `maxdepth`

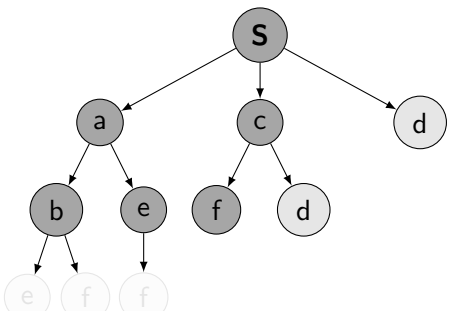




DFS with limited depth, `maxdepth=2`

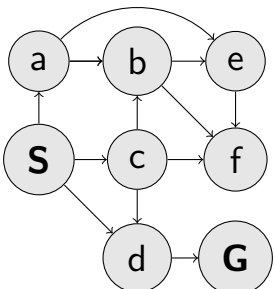
Do not follow nodes with `depth` > `maxdepth`

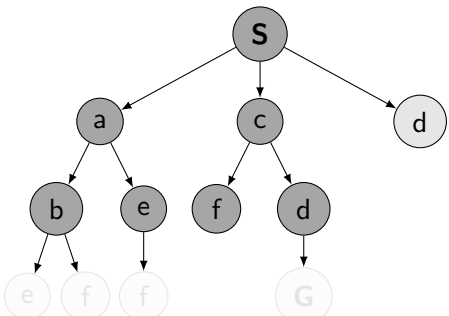




DFS with limited depth, `maxdepth=2`

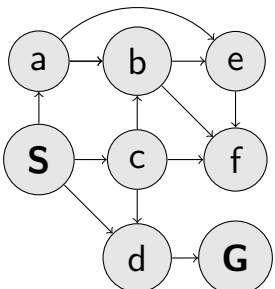
Do not follow nodes with `depth` > `maxdepth`

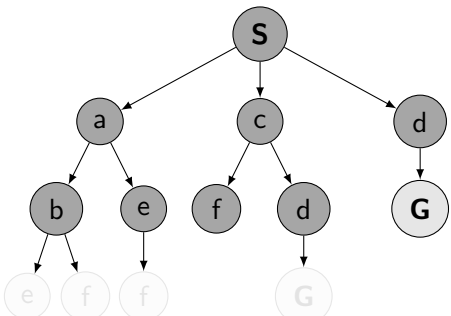




DFS with limited depth, `maxdepth=2`

Do not follow nodes with `depth` > `maxdepth`





Iterative deepening DFS (ID-DFS)

- Start with `maxdepth = 1`
- Perform DFS with limited depth. Report success or failure.
- If failure, forget everything, increase `maxdepth` and repeat DFS

Is it not a terrible waste to forget everything between steps?

Iterative deepening DFS (ID-DFS)

- Start with `maxdepth = 1`
- Perform DFS with limited depth. Report success or failure.
- If failure, forget everything, increase `maxdepth` and repeat DFS

Is it not a terrible waste to forget everything between steps?

Iterative deepening DFS (ID-DFS)

- Start with `maxdepth = 1`
- Perform DFS with limited depth. Report success or failure.
- If failure, forget everything, increase `maxdepth` and repeat DFS

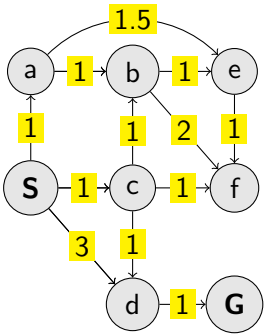
Is it not a terrible waste to forget everything between steps?

Iterative deepening DFS (ID-DFS)

- Start with `maxdepth = 1`
- Perform DFS with limited depth. Report success or failure.
- If failure, forget everything, increase `maxdepth` and repeat DFS

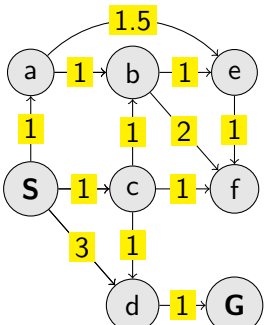
Is it not a terrible waste to forget everything between steps?

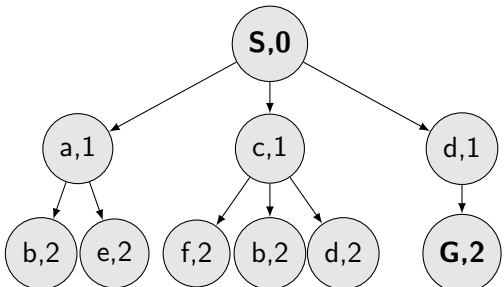
Cost sensitive search



- In BFS, DFS, node ±depth was the node-value.
- How was the depth actually computed?
- How to evaluate nodes with path cost?

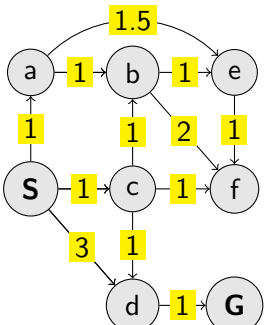
Cost sensitive search

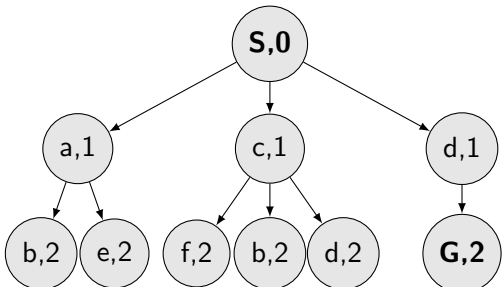




- In BFS, DFS, node $\pm$depth was the node-value.
- How was the depth actually computed?
- How to evaluate nodes with path cost?

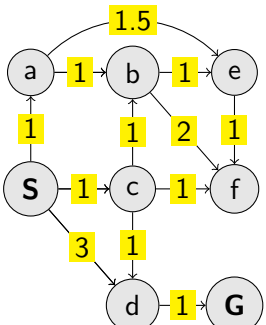
Cost sensitive search

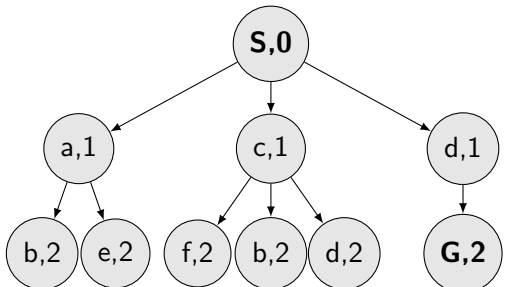




- In BFS, DFS, node $\pm$depth was the node-value.
- How was the depth actually computed?
- How to evaluate nodes with path cost?

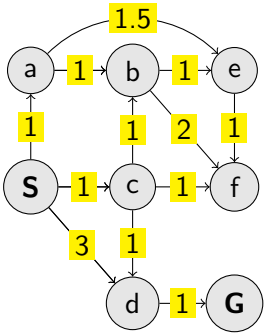
Cost sensitive search





- In BFS, DFS, node $\pm$depth was the node-value.
- How was the depth actually computed?
- How to evaluate nodes with path cost?

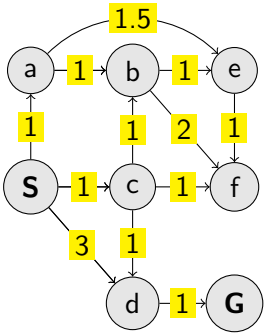
Uniform Cost Search (UCS)

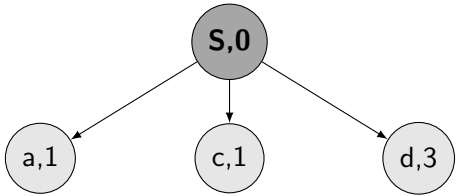


S,0

When to check the goal (and stop) the search? When visiting or expanding the node?

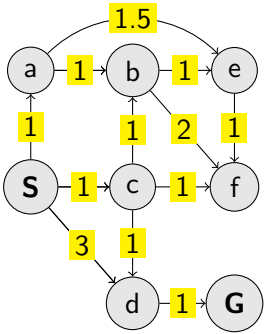
Uniform Cost Search (UCS)

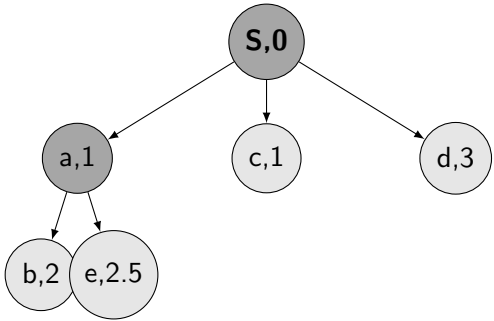




When to check the goal (and stop) the search? When visiting or expanding the node?

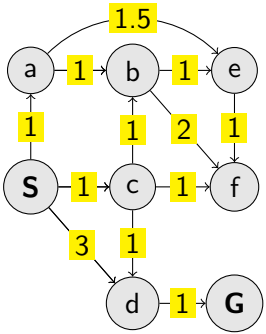
Uniform Cost Search (UCS)

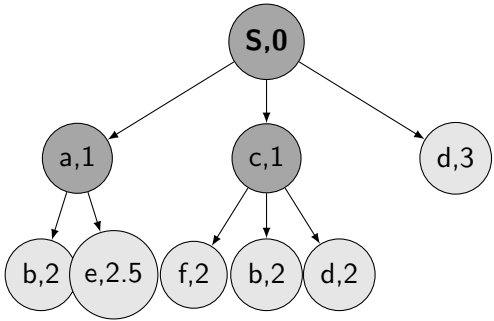




When to check the goal (and stop) the search? When visiting or expanding the node?

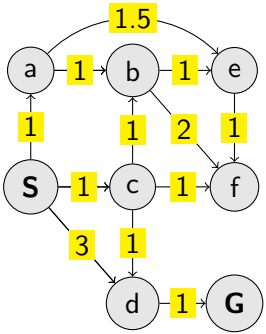
Uniform Cost Search (UCS)

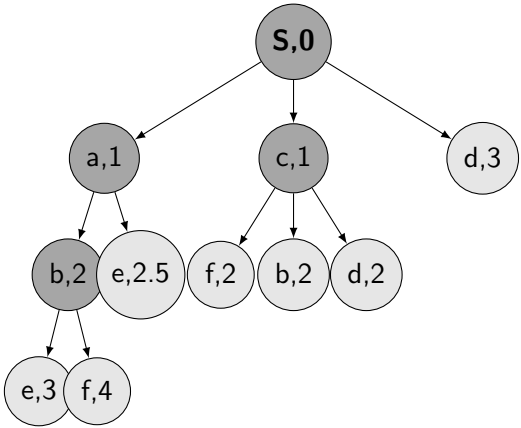




When to check the goal (and stop) the search? When visiting or expanding the node?

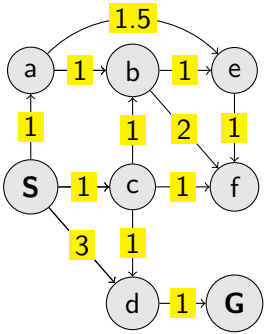
Uniform Cost Search (UCS)

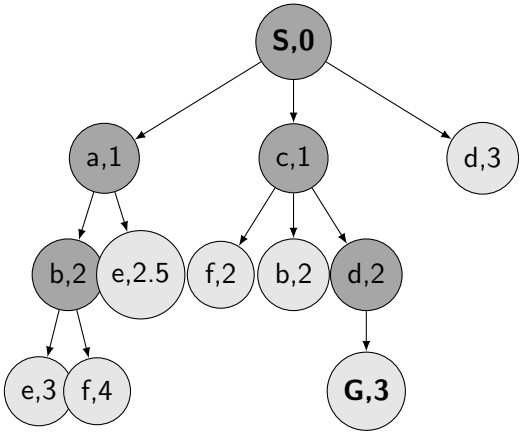




When to check the goal (and stop) the search? When visiting or expanding the node?

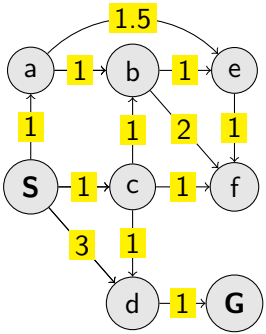
Uniform Cost Search (UCS)

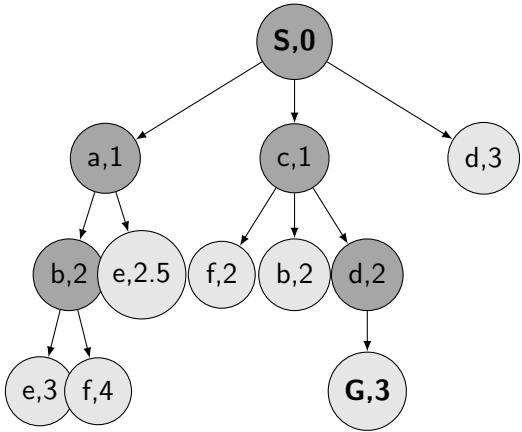




When to check the goal (and stop) the search? When visiting or expanding the node?

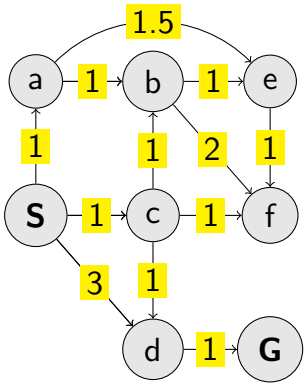
Uniform Cost Search (UCS)



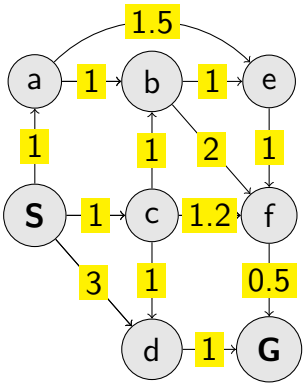


When to check the goal (and stop) the search? When visiting or expanding the node?

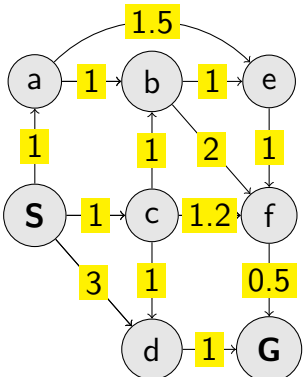
When to stop, when visiting or expanding?



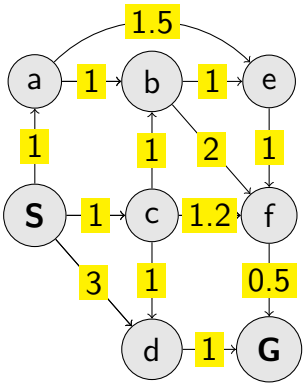
When to stop, when visiting or expanding?

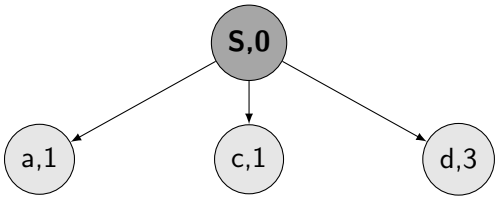


When to stop, when visiting or expanding?

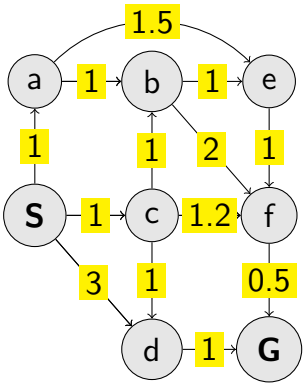


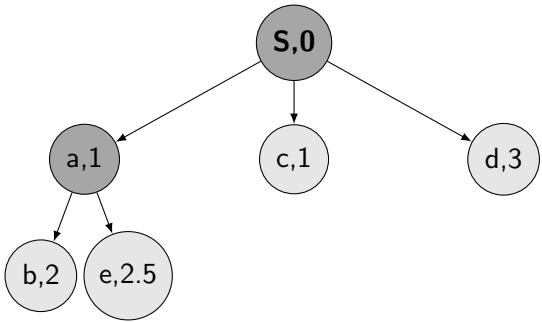
When to stop, when visiting or expanding?



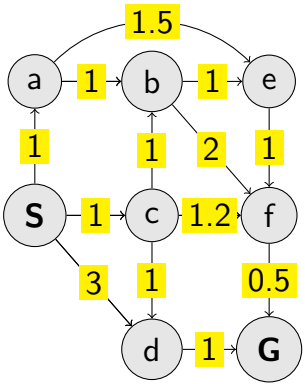


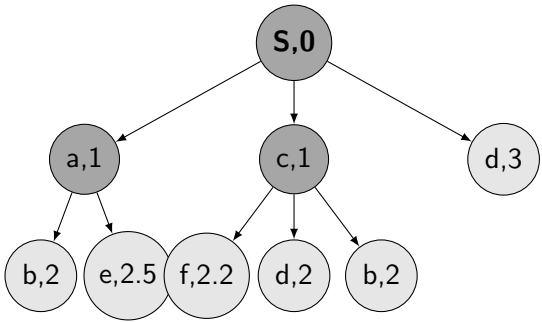
When to stop, when visiting or expanding?



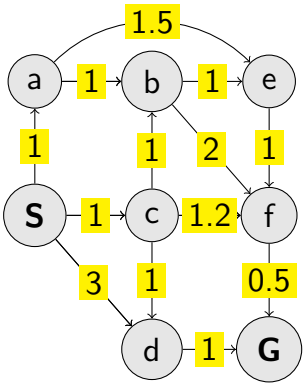


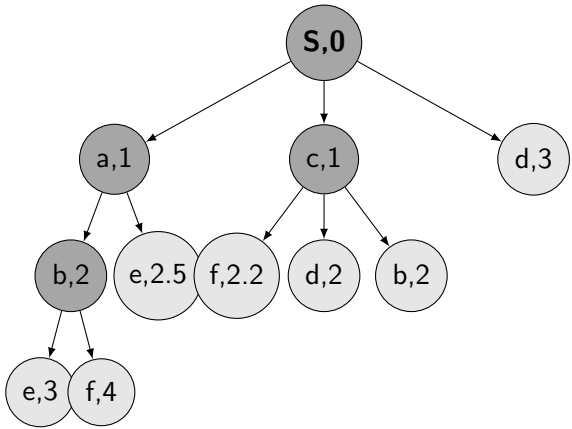
When to stop, when visiting or expanding?



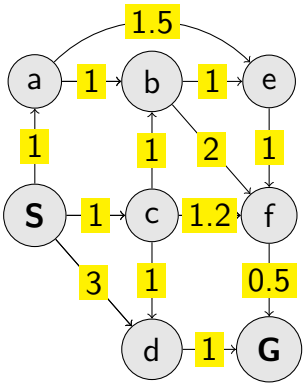


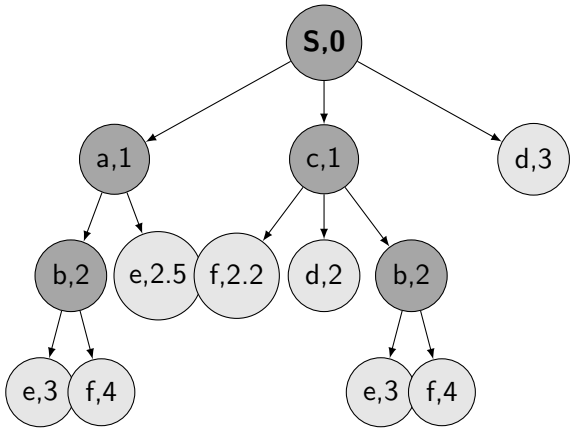
When to stop, when visiting or expanding?



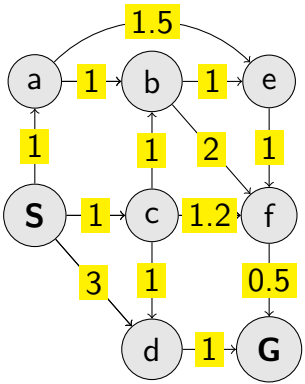


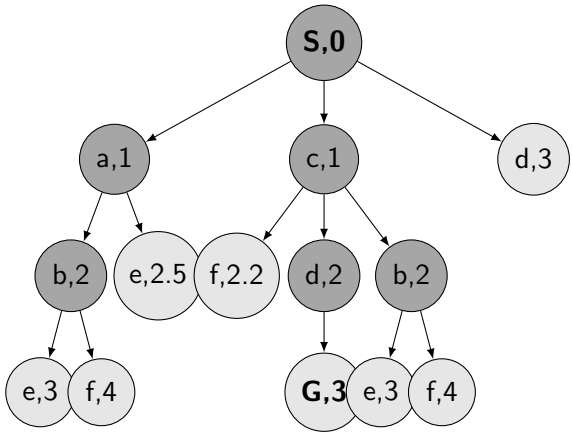
When to stop, when visiting or expanding?



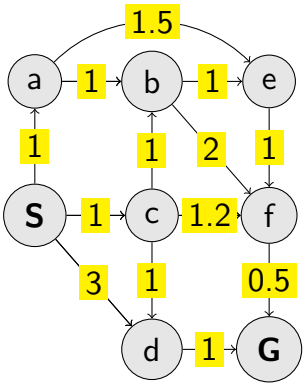


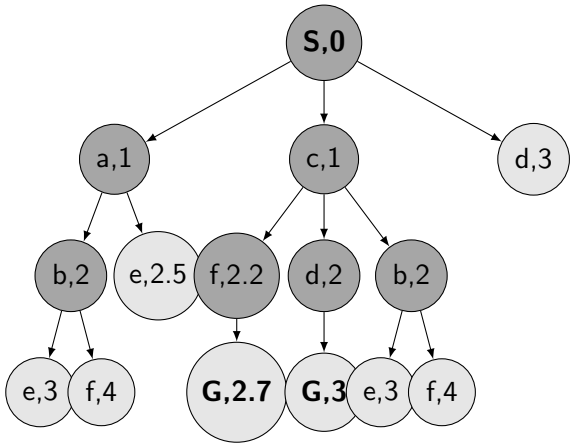
When to stop, when visiting or expanding?



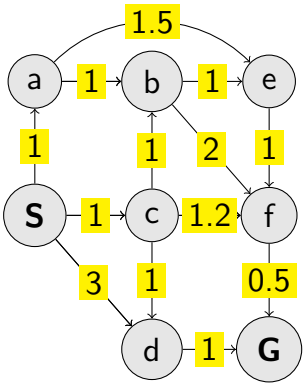


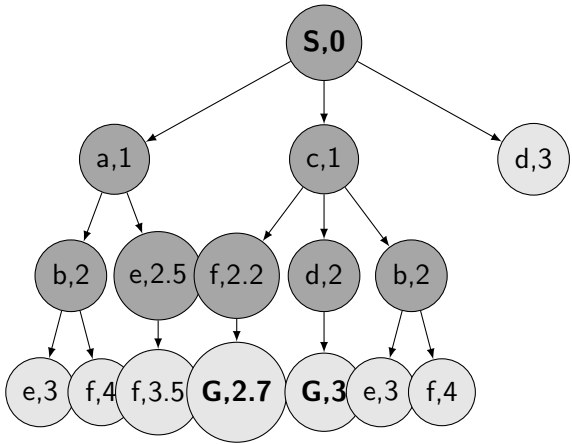
When to stop, when visiting or expanding?



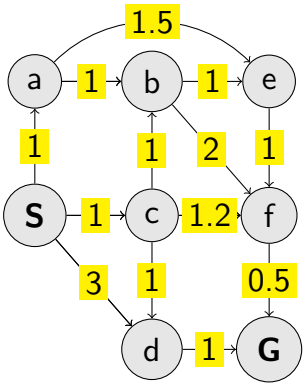


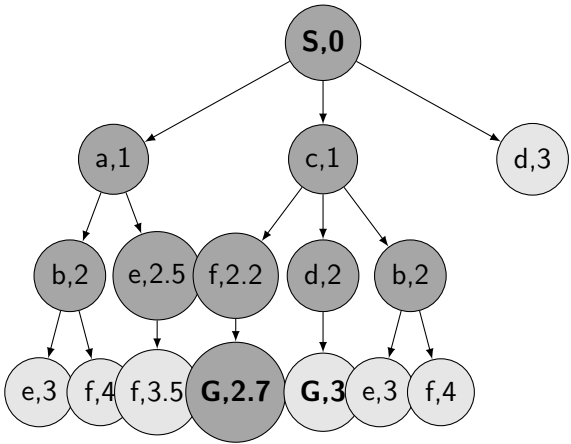
When to stop, when visiting or expanding?





When to stop, when visiting or expanding?



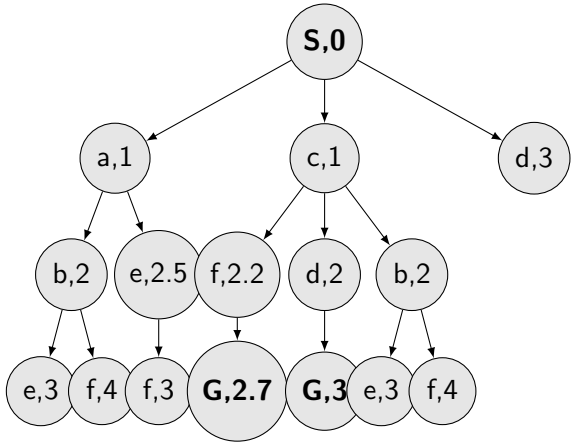


UCS properties

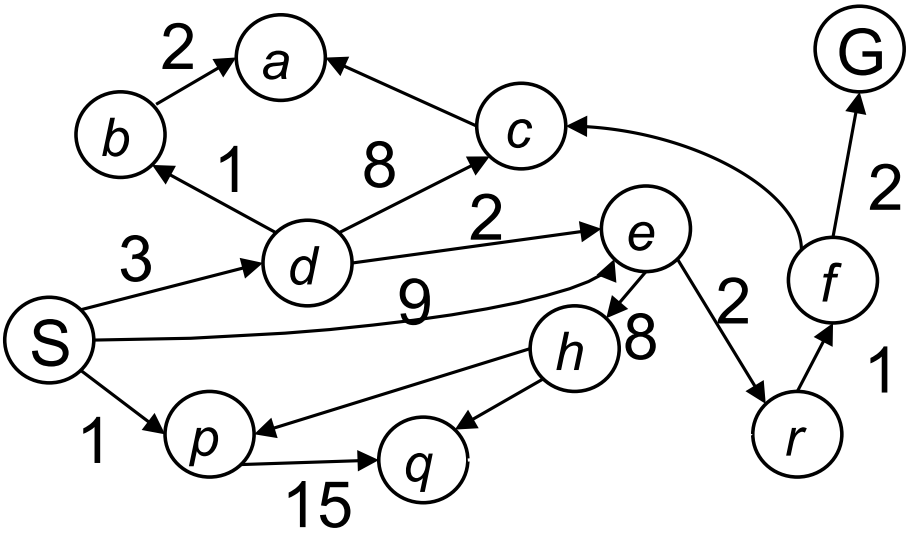
- Time complexity?
- Space complexity?
- Complete?
- Optimal?

UCS properties

- Time complexity?
- Space complexity?
- Complete?
- Optimal?



Example: Graph with costs



Infrastructure for (tree) search algorithms

What should a tree node `n` know?

- `n.state`
- `n.parent`
- `n.pathcost`

Perhaps we may add something later, if needed . . .

Infrastructure for (tree) search algorithms

What should a tree node `n` know?

- `n.state`
- `n.parent`
- `n.pathcost`

Perhaps we may add something later, if needed ...

How to organize nodes?

The Python examples are just suggestions, ...

- A dynamically linked structure (`list()`).
- Add a node (`list.insert(node)`).
- Take a node and remove from the structure (`node=list.pop()`).
- Check the Python modules `heapq`[1] and `queue`[2] for inspiration.

[1] https://docs.python.org/3.5/library/heapq.html
[2] https://docs.python.org/3.5/library/queue.html

What is the solution?

- We stop when Goal is reached.
- How do we construct the path?

References, further reading

Some figures if from [2]. Chapter 2 in [1] provides a compact/dense intro into search algorithms.

[1] Steven M. LaValle.
Planning Algorithms.
Cambridge, 1st edition, 2006.
Online version available at: http://planning.cs.uiuc.edu.

[2] Stuart Russell and Peter Norvig.
Artificial Intelligence: A Modern Approach.
Prentice Hall, 3rd edition, 2010.
http://aima.cs.berkeley.edu/.