Problem solving by search

Tomáš Svoboda

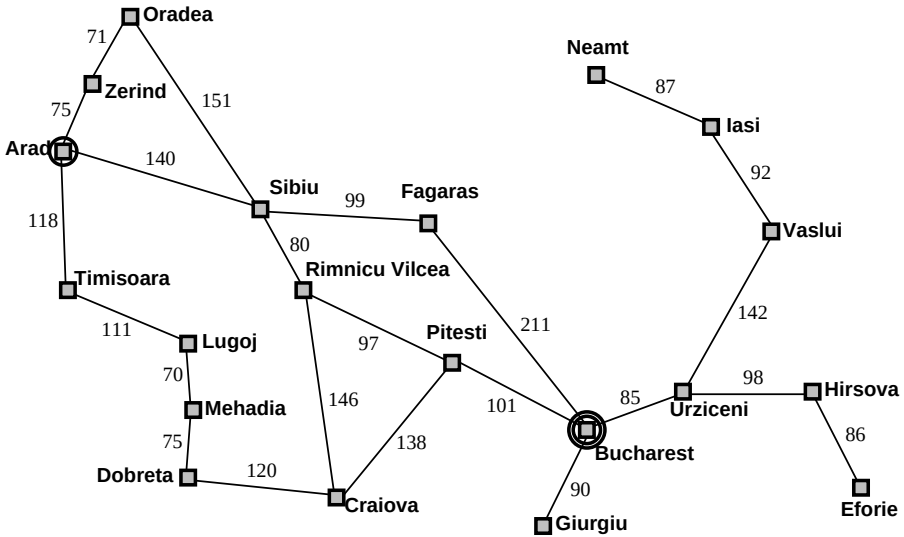
Vision for Robots and Autonomous Systems, Center for Machine Perception
Department of Cybernetics,
Faculty of Electrical Engineering, Czech Technical University in Prague

February 27, 2019

Outline

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

Example: Romania



Ok, start with a simple one, almost everybody knows about the navigation - path planning problem. Waze, Garmin, ...

Can you think about more problems?

Example: Romania

Goal:

be in Bucharest

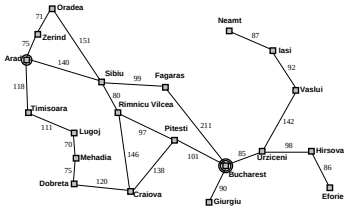
Problem formulation:

states: position in a city (cities)

actions: drive between cities

Solution:

Sequence of cities (path)



Classical problem from the Book [1], we use it, too.

Example: Romania

Goal:

be in Bucharest

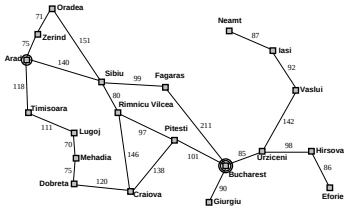
Problem formulation:

states: position in a city (cities)

actions: drive between cities

Solution:

Sequence of cities (path)



Classical problem from the Book [1], we use it, too.

Example: Romania

Goal:

be in Bucharest

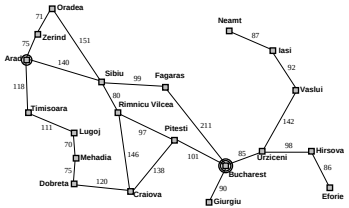
Problem formulation:

states: position in a city (cities)

actions: drive between cities

Solution:

Sequence of cities (path)



Classical problem from the Book [1], we use it, too.

Example: Romania

Goal:

be in Bucharest

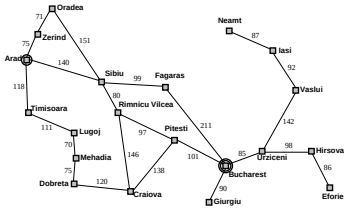
Problem formulation:

states: position in a city (cities)

actions: drive between cities

Solution:

Sequence of cities (path)



Classical problem from the Book [1], we use it, too.

Example: Romania

Goal:

be in Bucharest

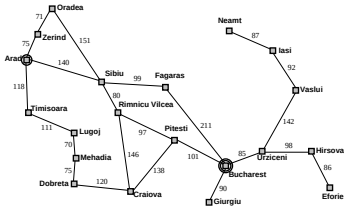
Problem formulation:

states: position in a city (cities)

actions: drive between cities

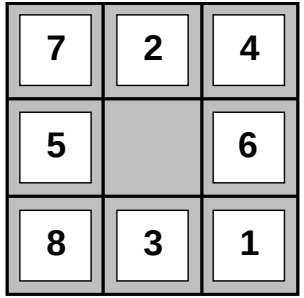
Solution:

Sequence of cities (path)

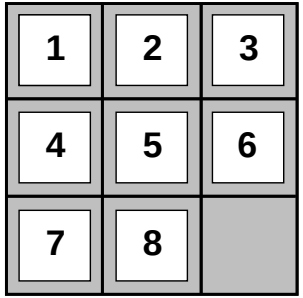


Classical problem from the Book [1], we use it, too.

Example: The 8-puzzle



Start State

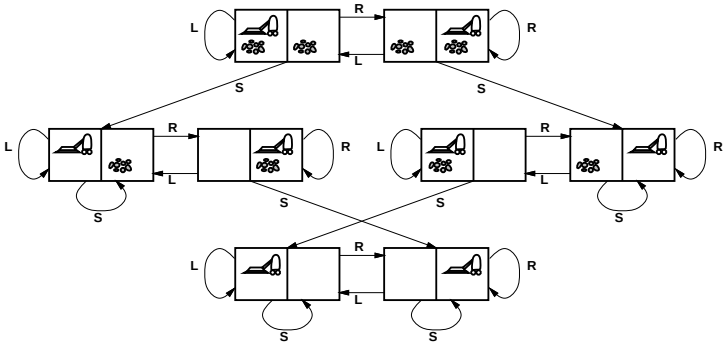


Goal State

states?
actions?
solution?
cost?

Also known as $n-1$ puzzle.

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?

We will use the terminology throught the next 5-6 lectures; also for Markov (Sequential) Decision Processes, Reinforcement Learning

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?

We will use the terminology throught the next 5-6 lectures; also for Markov (Sequential) Decision Processes, Reinforcement Learning

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?

We will use the terminology throught the next 5-6 lectures; also for Markov (Sequential) Decision Processes, Reinforcement Learning

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?

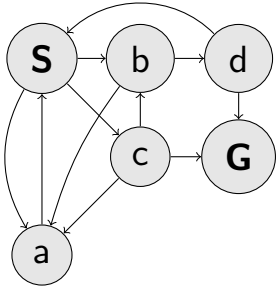
We will use the terminology throught the next 5-6 lectures; also for Markov (Sequential) Decision Processes, Reinforcement Learning

State Space Graphs

State space graph: a representation of a search problem

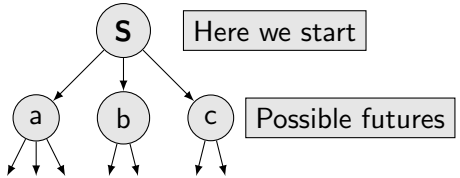
- Graph Nodes – states – are abstracted world configurations
- Arcs represent action results
- Goal test – a set of goal nodes

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



Formalizing a real world problem – (creating) state space graph – could be a problem of itself. I put creating into brackets is it may be also infinite.

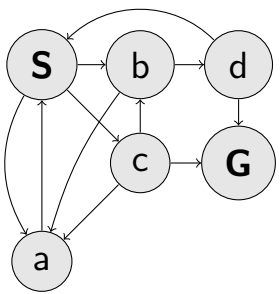
Search Trees



- 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

- What if decision about an action, repeats ...
- Nodes in the search tree are not the same as the nodes in the state space graph.

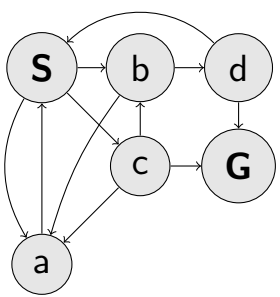
State Space Graphs vs. Search Trees

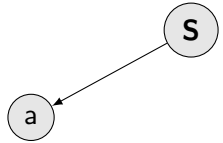


How big is the search tree?

- There could be multiple search trees, depending ot the algorithm.
- A search tree can be much bigger than the state space.
- Both items will be discussed next.

State Space Graphs vs. Search Trees

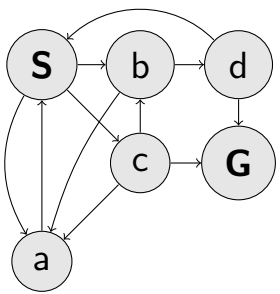


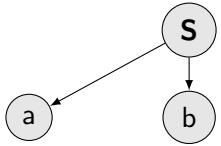


How big is the search tree?

- There could be multiple search trees, depending ot the algorithm.
- A search tree can be much bigger than the state space.
- Both items will be discussed next.

State Space Graphs vs. Search Trees

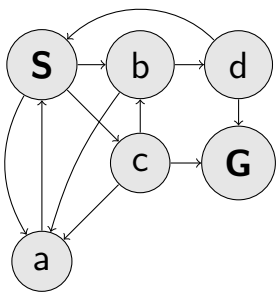


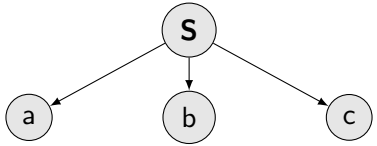


How big is the search tree?

- There could be multiple search trees, depending ot the algorithm.
- A search tree can be much bigger than the state space.
- Both items will be discussed next.

State Space Graphs vs. Search Trees

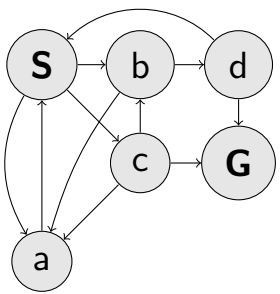


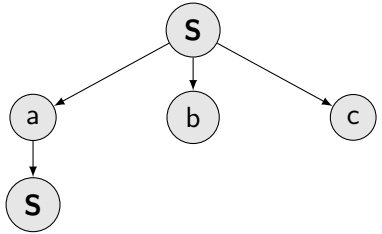


How big is the search tree?

- There could be multiple search trees, depending ot the algorithm.
- A search tree can be much bigger than the state space.
- Both items will be discussed next.

State Space Graphs vs. Search Trees

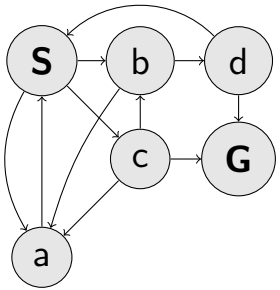


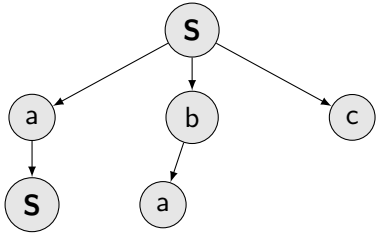


How big is the search tree?

- There could be multiple search trees, depending ot the algorithm.
- A search tree can be much bigger than the state space.
- Both items will be discussed next.

State Space Graphs vs. Search Trees

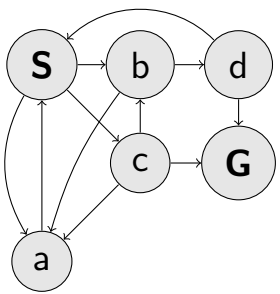


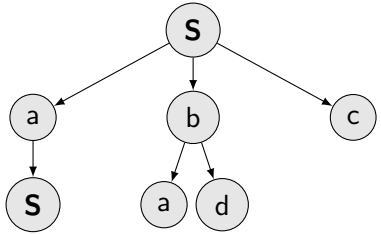


How big is the search tree?

- There could be multiple search trees, depending ot the algorithm.
- A search tree can be much bigger than the state space.
- Both items will be discussed next.

State Space Graphs vs. Search Trees

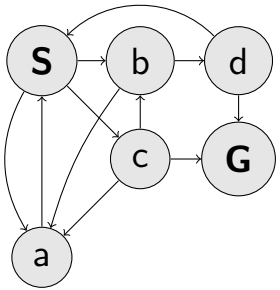


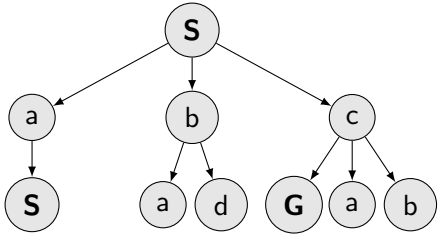


How big is the search tree?

- There could be multiple search trees, depending ot the algorithm.
- A search tree can be much bigger than the state space.
- Both items will be discussed next.

State Space Graphs vs. Search Trees

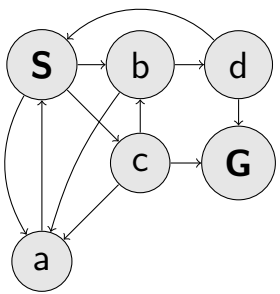


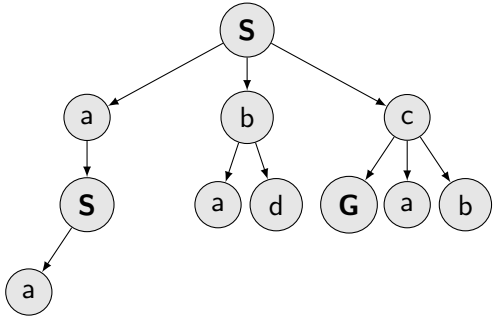


How big is the search tree?

- There could be multiple search trees, depending ot the algorithm.
- A search tree can be much bigger than the state space.
- Both items will be discussed next.

State Space Graphs vs. Search Trees

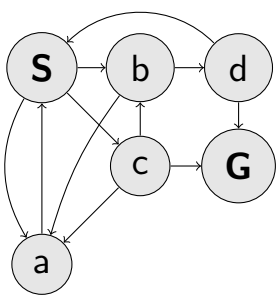


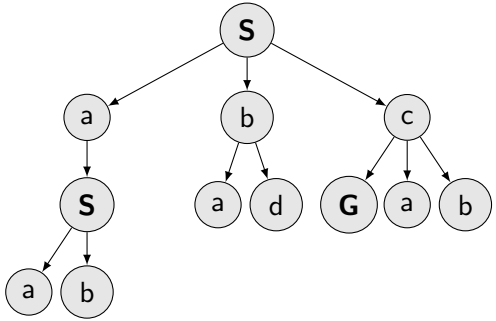


How big is the search tree?

- There could be multiple search trees, depending ot the algorithm.
- A search tree can be much bigger than the state space.
- Both items will be discussed next.

State Space Graphs vs. Search Trees

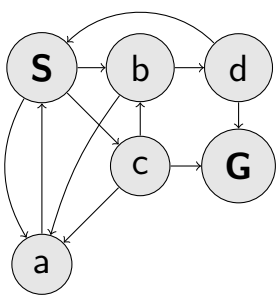


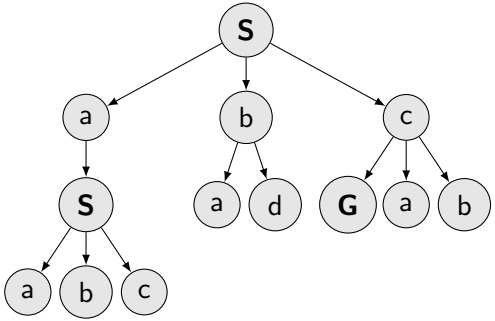


How big is the search tree?

- There could be multiple search trees, depending ot the algorithm.
- A search tree can be much bigger than the state space.
- Both items will be discussed next.

State Space Graphs vs. Search Trees

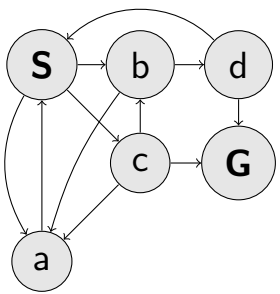


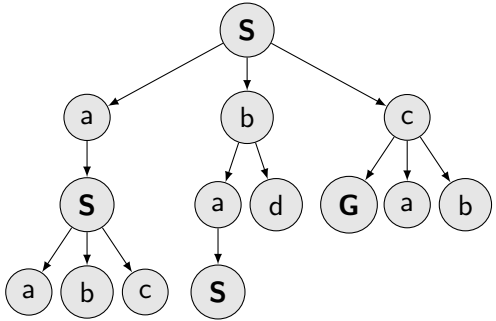


How big is the search tree?

- There could be multiple search trees, depending ot the algorithm.
- A search tree can be much bigger than the state space.
- Both items will be discussed next.

State Space Graphs vs. Search Trees

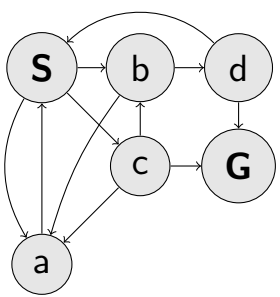


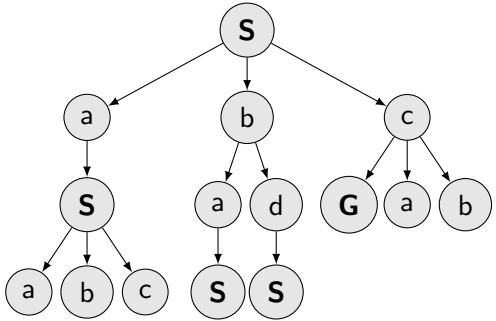


How big is the search tree?

- There could be multiple search trees, depending ot the algorithm.
- A search tree can be much bigger than the state space.
- Both items will be discussed next.

State Space Graphs vs. Search Trees

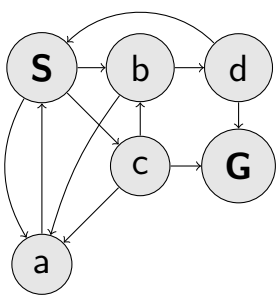


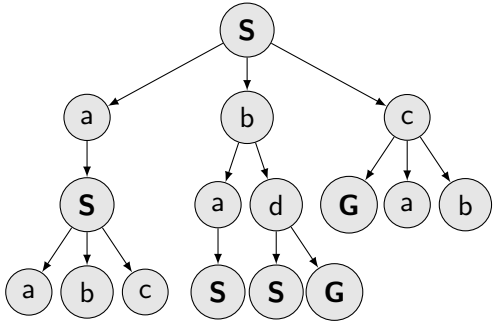


How big is the search tree?

- There could be multiple search trees, depending ot the algorithm.
- A search tree can be much bigger than the state space.
- Both items will be discussed next.

State Space Graphs vs. Search Trees



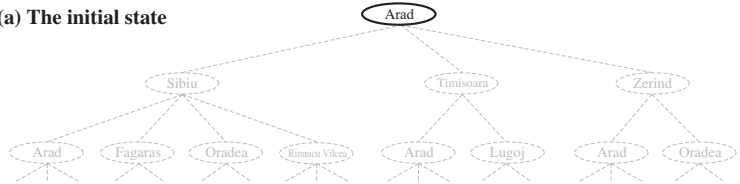


How big is the search tree?

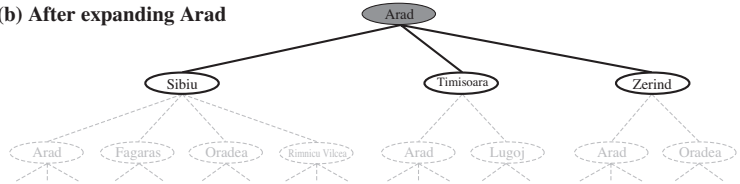
- There could be multiple search trees, depending ot the algorithm.
- A search tree can be much bigger than the state space.
- Both items will be discussed next.

Search tree for Romania

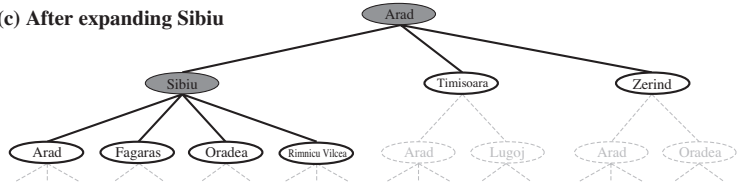
(a) The initial state



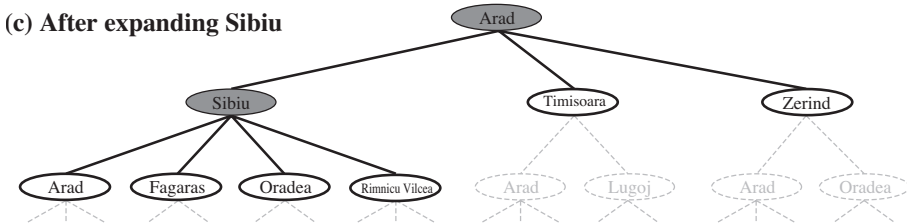
(b) After expanding Arad



(c) After expanding Sibiu

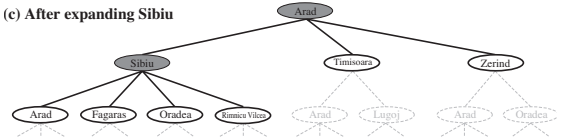


Search elements



- 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

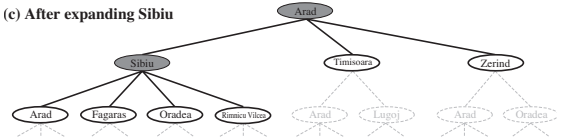
end if

Expand the node and add the resulting nodes to the tree

end loop

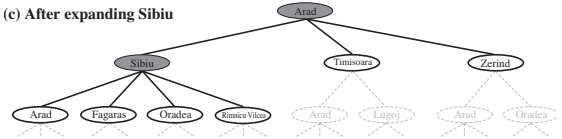
end function

Tree search algorithm



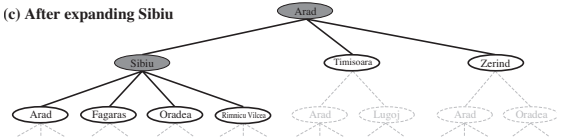
function TREE_SEARCH(problem) **return** a solution or failure
initialize by using the initial state of the problem
loop
if no candidates for expansion **then return** failure
else choose a leaf node for expansion
end if
if the node contains a goal state **then return** the solution
end if
Expand the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

Tree search algorithm



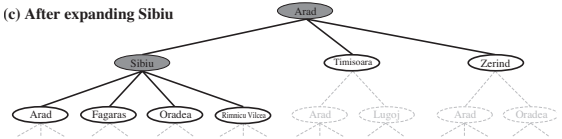
function TREE_SEARCH(problem) **return** a solution or failure
initialize by using the initial state of the problem
loop
if no candidates for expansion **then return** failure
else choose a leaf node for expansion
end if
if the node contains a goal state **then return** the solution
end if
Expand the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

Tree search algorithm



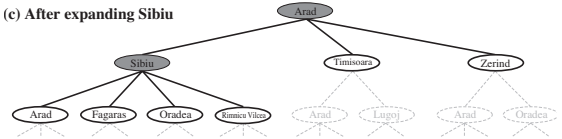
function TREE_SEARCH(problem) **return** a solution or failure
  initialize by using the initial state of the problem
  loop
    if no candidates for expansion **then return** failure
    else choose a leaf node for expansion
    end if
    if the node contains a goal state **then return** the solution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

Tree search algorithm



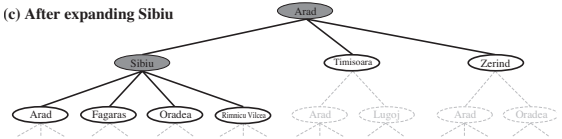
function TREE_SEARCH(problem) **return** a solution or failure
  initialize by using the initial state of the problem
  loop
    if no candidates for expansion **then return** failure
    else choose a leaf node for expansion
    end if
    if the node contains a goal state **then return** the solution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

Tree search algorithm



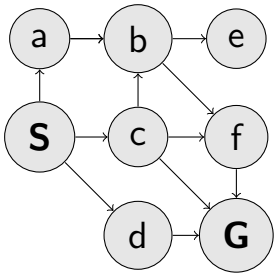
function TREE_SEARCH(problem) **return** a solution or failure
  initialize by using the initial state of the problem
  loop
    if no candidates for expansion **then return** failure
    else choose a leaf node for expansion
    end if
    if the node contains a goal state **then return** the solution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

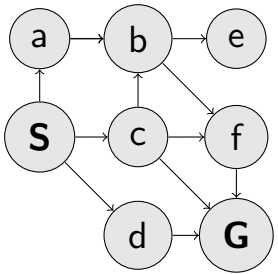
Example of a tree search

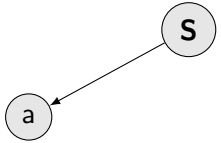


Which nodes to *explore*?
What are the properties of a strategy/algorithm?

Before going to the next slide, think about algorithms. What properties of an algorithm would you want?

Example of a tree search

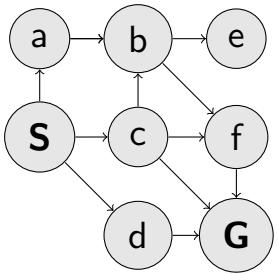


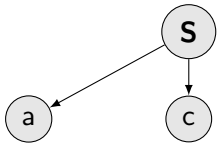


Which nodes to *explore*?
What are the properties of a strategy/algorithm?

Before going to the next slide, think about algorithms. What properties of an algorithm would you want?

Example of a tree search

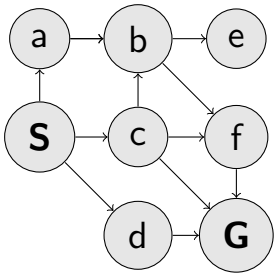


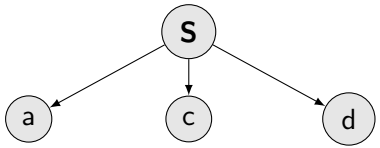


Which nodes to *explore*?
What are the properties of a strategy/algorithm?

Before going to the next slide, think about algorithms. What properties of an algorithm would you want?

Example of a tree search

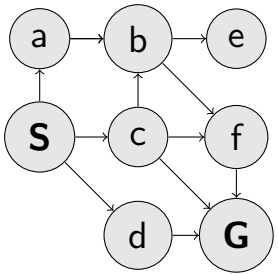


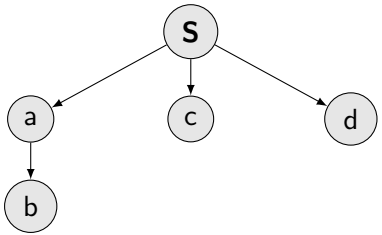


Which nodes to *explore*?
What are the properties of a strategy/algorithm?

Before going to the next slide, think about algorithms. What properties of an algorithm would you want?

Example of a tree search

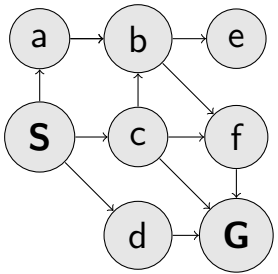


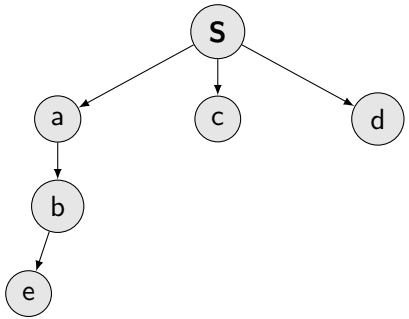


Which nodes to *explore*?
What are the properties of a strategy/algorithm?

Before going to the next slide, think about algorithms. What properties of an algorithm would you want?

Example of a tree search

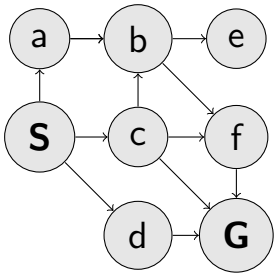


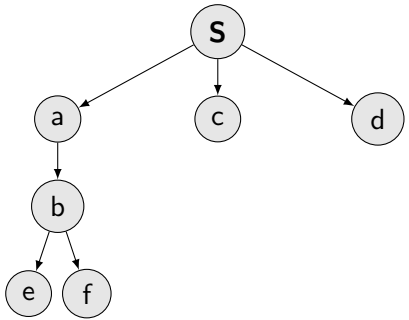


Which nodes to *explore*?
What are the properties of a strategy/algorithm?

Before going to the next slide, think about algorithms. What properties of an algorithm would you want?

Example of a tree search

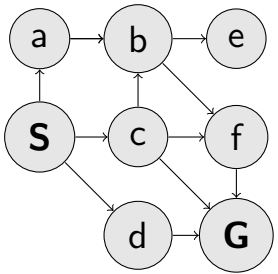


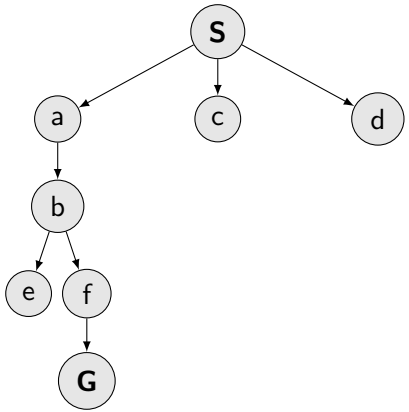


Which nodes to *explore*?
What are the properties of a strategy/algorithm?

Before going to the next slide, think about algorithms. What properties of an algorithm would you want?

Example of a tree search

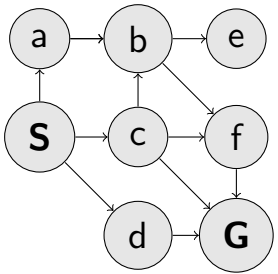


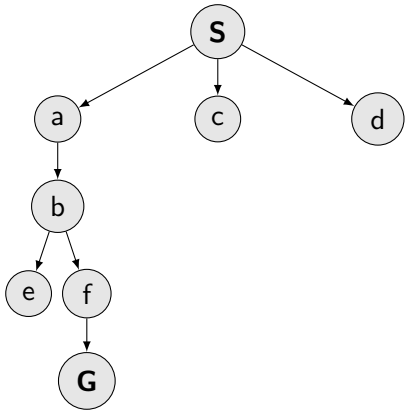


Which nodes to *explore*?
What are the properties of a strategy/algorithm?

Before going to the next slide, think about algorithms. What properties of an algorithm would you want?

Example of a tree search



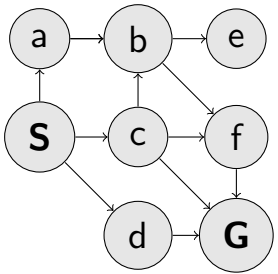


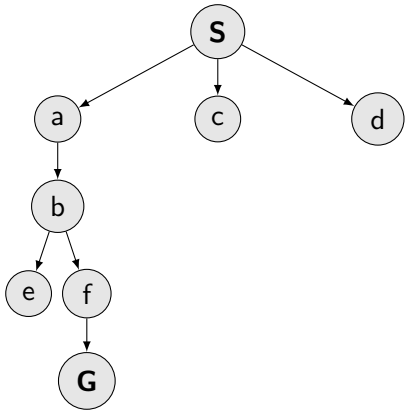
Which nodes to *explore*?

What are the properties of a strategy/algorithm?

Before going to the next slide, think about algorithms. What properties of an algorithm would you want?

Example of a tree search





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

Before going to the next slide, think about algorithms. What properties of an algorithm would you want?

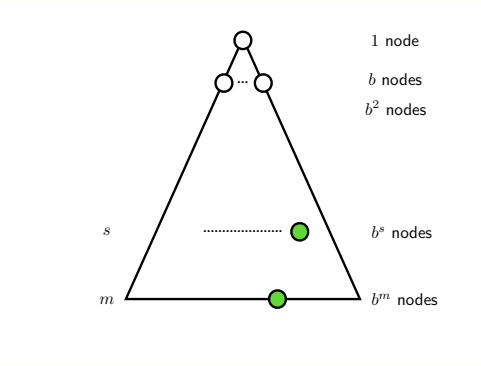
Search (algorithm) properties

- 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 tree? What are tree parameters?

Draw a (symbolic–think about a triangle) sketch of a (search) tree. It may grow upwards or downwards. How would you characterize/parametrize *size* of a 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?



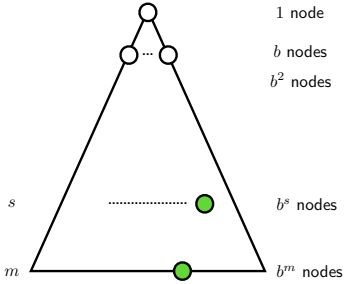
Search (algorithm) properties

- 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 tree? What are tree parameters?

Draw a (symbolic–think about a triangle) sketch of a (search) tree. It may grow upwards or downwards. How would you characterize/parametrize *size* of a 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?



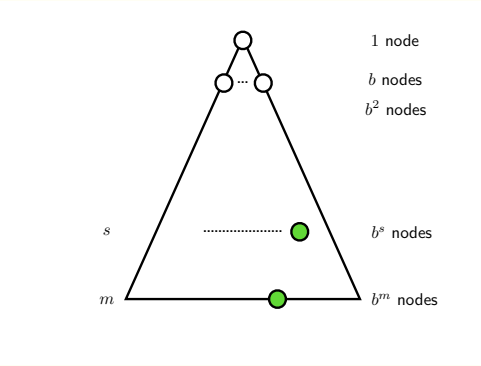
Search (algorithm) properties

- 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 tree? What are tree parameters?

Draw a (symbolic–think about a triangle) sketch of a (search) tree. It may grow upwards or downwards. How would you characterize/parametrize *size* of a 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?



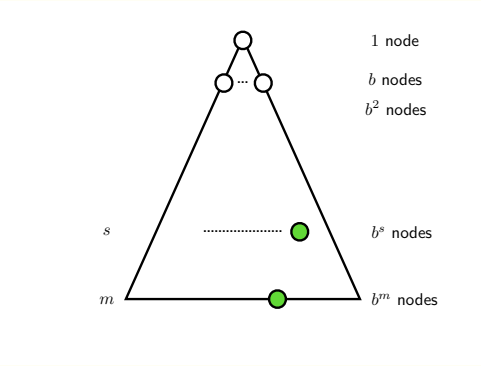
Search (algorithm) properties

- 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 tree? What are tree parameters?

Draw a (symbolic–think about a triangle) sketch of a (search) tree. It may grow upwards or downwards. How would you characterize/parametrize *size* of a 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?



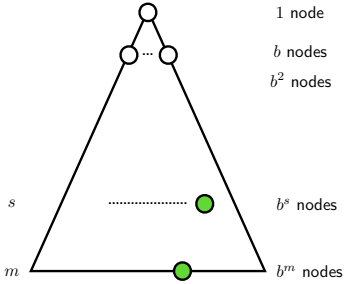
Search (algorithm) properties

- 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 tree? What are tree parameters?

Draw a (symbolic–think about a triangle) sketch of a (search) tree. It may grow upwards or downwards. How would you characterize/parametrize *size* of a 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?



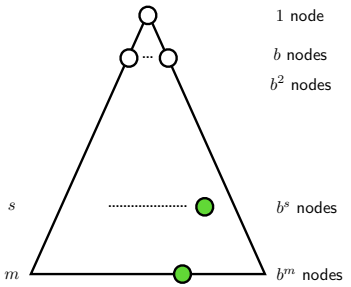
Search (algorithm) properties

- 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 tree? What are tree parameters?

Draw a (symbolic–think about a triangle) sketch of a (search) tree. It may grow upwards or downwards. How would you characterize/parametrize *size* of a 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?



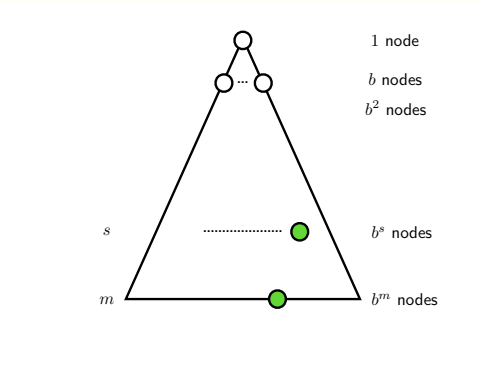
Search (algorithm) properties

- 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 tree? What are tree parameters?

Draw a (symbolic–think about a triangle) sketch of a (search) tree. It may grow upwards or downwards. How would you characterize/parametrize *size* of a 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?



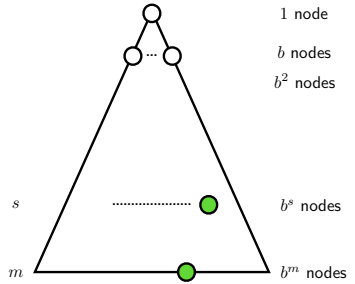
Search (algorithm) properties

- 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 tree? What are tree parameters?

Draw a (symbolic–think about a triangle) sketch of a (search) tree. It may grow upwards or downwards. How would you characterize/parametrize *size* of a 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?



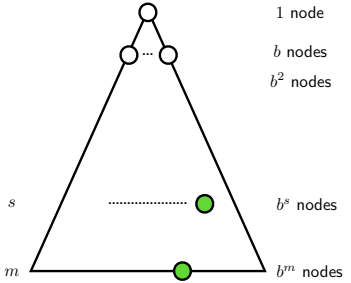
Search (algorithm) properties

- 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 tree? What are tree parameters?

Draw a (symbolic–think about a triangle) sketch of a (search) tree. It may grow upwards or downwards. How would you characterize/parametrize *size* of a 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?



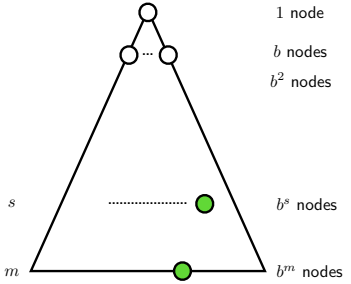
Search (algorithm) properties

- 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 tree? What are tree parameters?

Draw a (symbolic–think about a triangle) sketch of a (search) tree. It may grow upwards or downwards. How would you characterize/parametrize *size* of a 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?

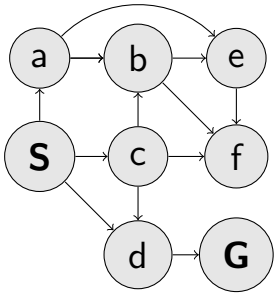


Strategies

How to traverse/build a search tree?

It is perhaps worth to remember that the search tree is built as the algorithm goes. Or better said, the tree is a human friendly representaion of the machine run.

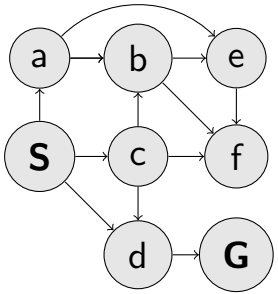
Depth-First Search (DFS)

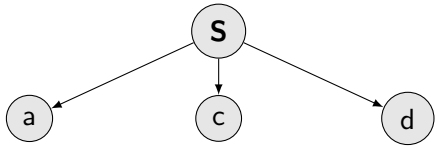


What are the DFS properties (complete, optimal, time, space)?

- In animation, we will do the expansion step at once.
- What is the *frontier* - set of nodes, waiting to be expanded?
- When to stop the search?
- Thinking about optimality, what is the best solution we seek?

Depth-First Search (DFS)

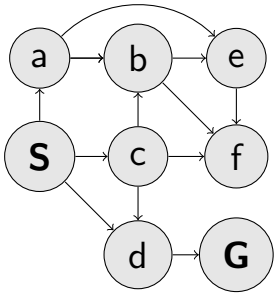


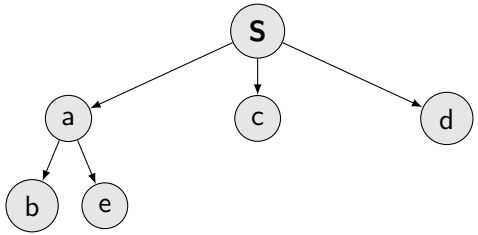


What are the DFS properties (complete, optimal, time, space)?

- In animation, we will do the expansion step at once.
- What is the *frontier* - set of nodes, waiting to be expanded?
- When to stop the search?
- Thinking about optimality, what is the best solution we seek?

Depth-First Search (DFS)

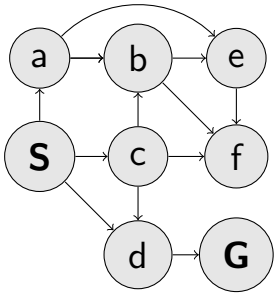


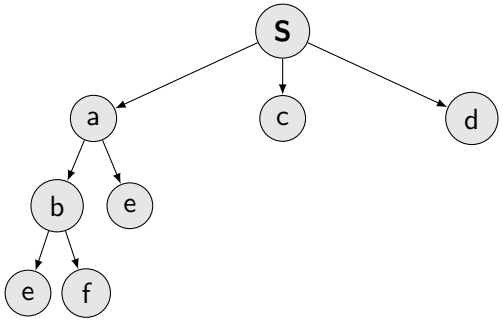


What are the DFS properties (complete, optimal, time, space)?

- In animation, we will do the expansion step at once.
- What is the *frontier* - set of nodes, waiting to be expanded?
- When to stop the search?
- Thinking about optimality, what is the best solution we seek?

Depth-First Search (DFS)

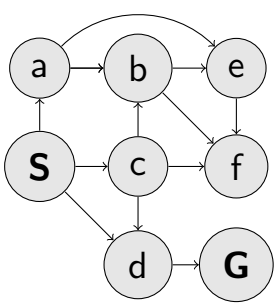


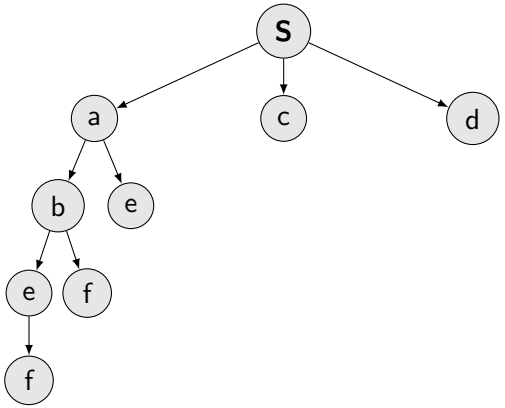


What are the DFS properties (complete, optimal, time, space)?

- In animation, we will do the expansion step at once.
- What is the *frontier* - set of nodes, waiting to be expanded?
- When to stop the search?
- Thinking about optimality, what is the best solution we seek?

Depth-First Search (DFS)

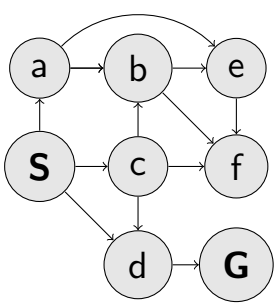


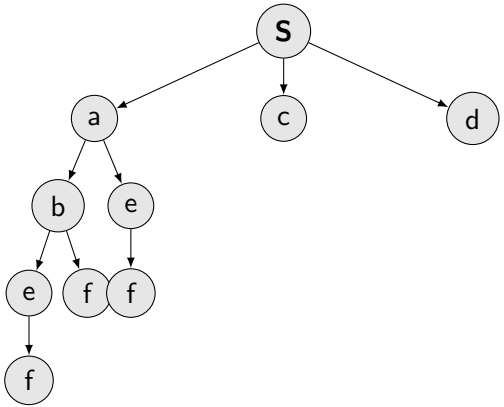


What are the DFS properties (complete, optimal, time, space)?

- In animation, we will do the expansion step at once.
- What is the *frontier* - set of nodes, waiting to be expanded?
- When to stop the search?
- Thinking about optimality, what is the best solution we seek?

Depth-First Search (DFS)

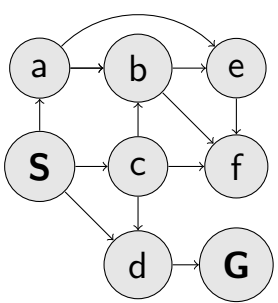


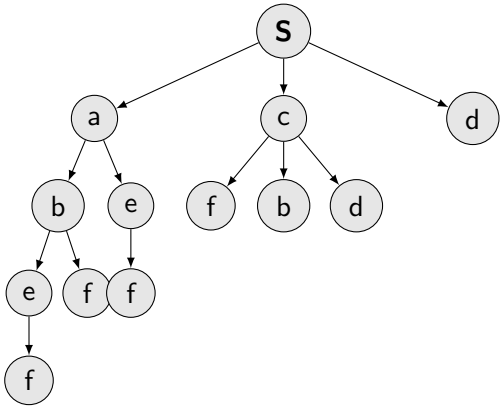


What are the DFS properties (complete, optimal, time, space)?

- In animation, we will do the expansion step at once.
- What is the *frontier* - set of nodes, waiting to be expanded?
- When to stop the search?
- Thinking about optimality, what is the best solution we seek?

Depth-First Search (DFS)

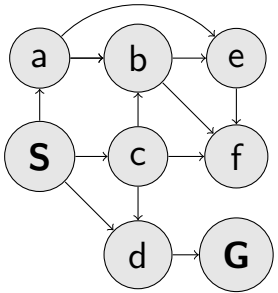


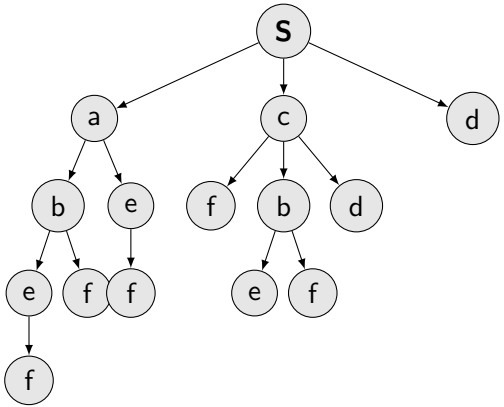


What are the DFS properties (complete, optimal, time, space)?

- In animation, we will do the expansion step at once.
- What is the *frontier* - set of nodes, waiting to be expanded?
- When to stop the search?
- Thinking about optimality, what is the best solution we seek?

Depth-First Search (DFS)

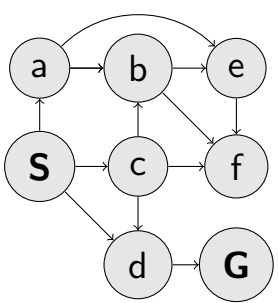


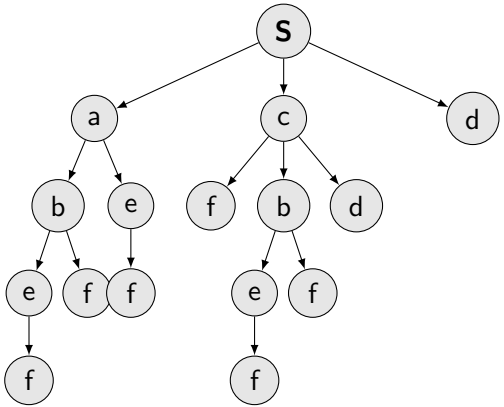


What are the DFS properties (complete, optimal, time, space)?

- In animation, we will do the expansion step at once.
- What is the *frontier* - set of nodes, waiting to be expanded?
- When to stop the search?
- Thinking about optimality, what is the best solution we seek?

Depth-First Search (DFS)

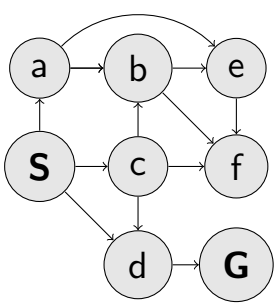


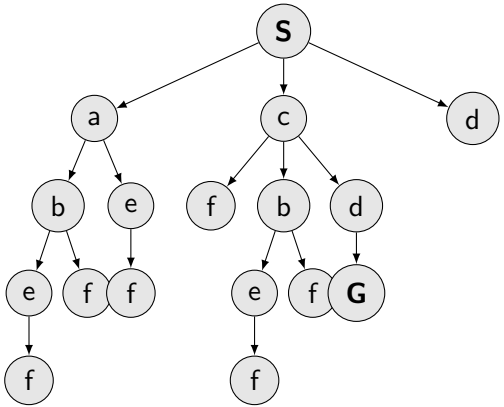


What are the DFS properties (complete, optimal, time, space)?

- In animation, we will do the expansion step at once.
- What is the *frontier* - set of nodes, waiting to be expanded?
- When to stop the search?
- Thinking about optimality, what is the best solution we seek?

Depth-First Search (DFS)

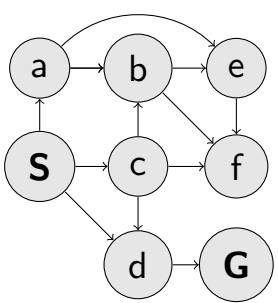


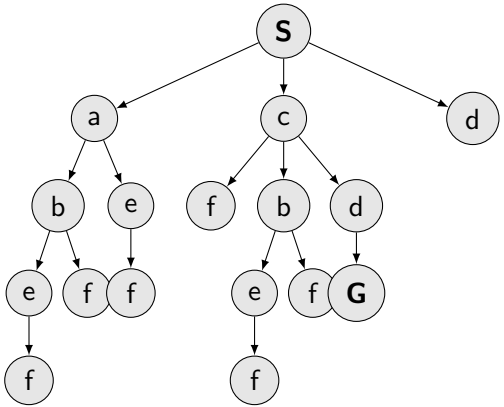


What are the DFS properties (complete, optimal, time, space)?

- In animation, we will do the expansion step at once.
- What is the *frontier* - set of nodes, waiting to be expanded?
- When to stop the search?
- Thinking about optimality, what is the best solution we seek?

Depth-First Search (DFS)



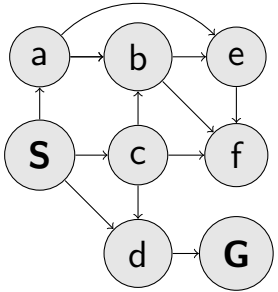


What are the DFS properties (complete, optimal, time, space)?

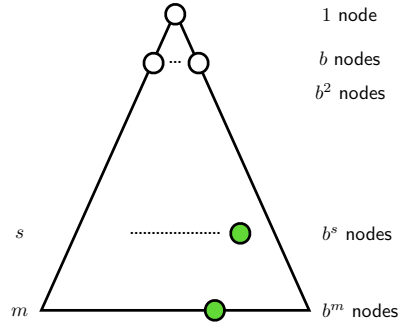
- In animation, we will do the expansion step at once.
- What is the *frontier* - set of nodes, waiting to be expanded?
- When to stop the search?
- Thinking about optimality, what is the best solution we seek?

DFS properties

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

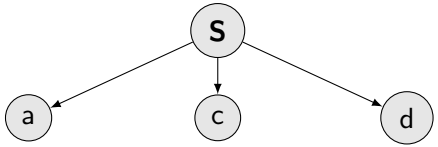


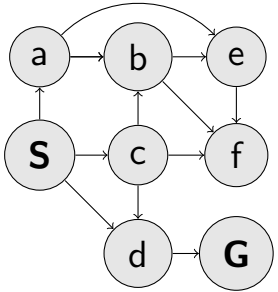
- Time, can process the whole tree: b^m
- Space, only the path so far: bm
- Completness: m may be ∞ hence, not in general
- Optimality: No! It just takes the first solution found.



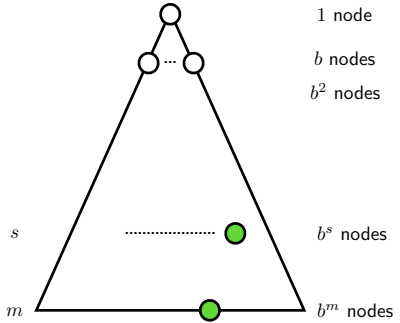
DFS properties

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



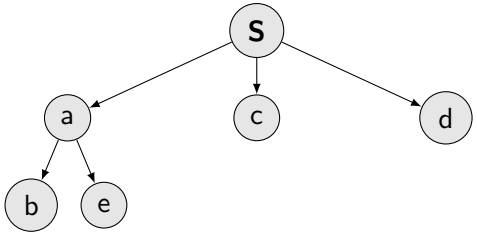


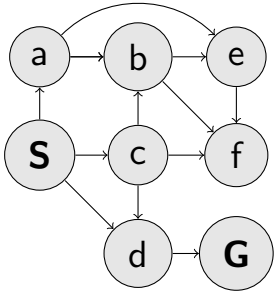
- Time, can process the whole tree: b^m
- Space, only the path so far: bm
- Completness: m may be ∞ hence, not in general
- Optimality: No! It just takes the first solution found.



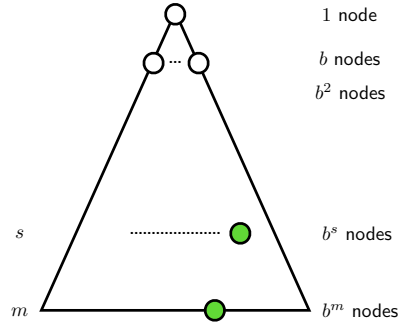
DFS properties

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



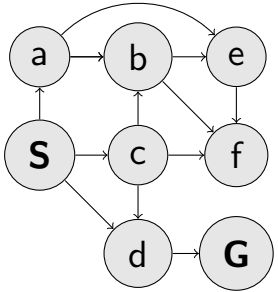


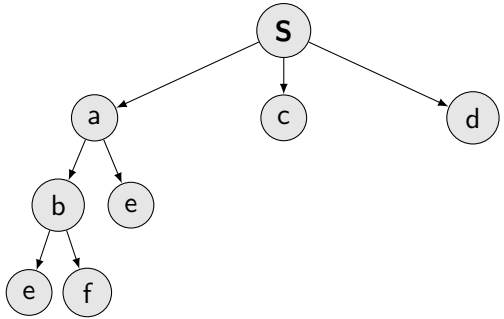
- Time, can process the whole tree: b^m
- Space, only the path so far: bm
- Completness: m may be ∞ hence, not in general
- Optimality: No! It just takes the first solution found.



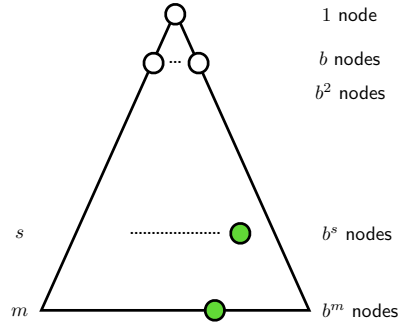
DFS properties

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



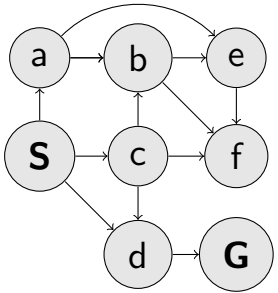


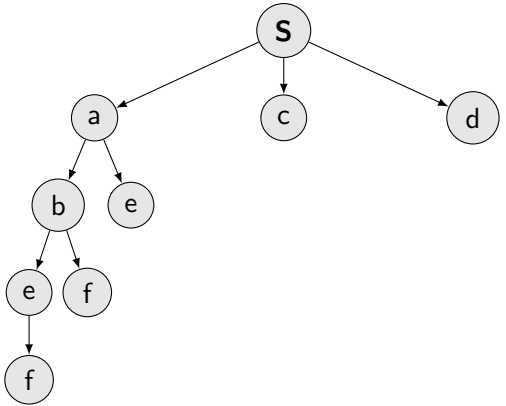
- Time, can process the whole tree: b^m
- Space, only the path so far: bm
- Completness: m may be ∞ hence, not in general
- Optimality: No! It just takes the first solution found.



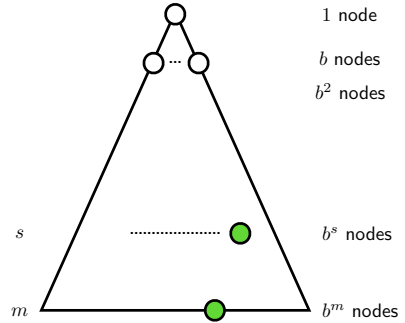
DFS properties

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



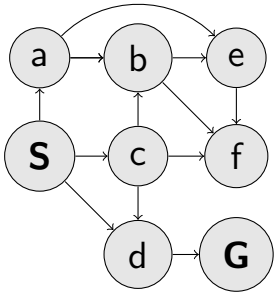


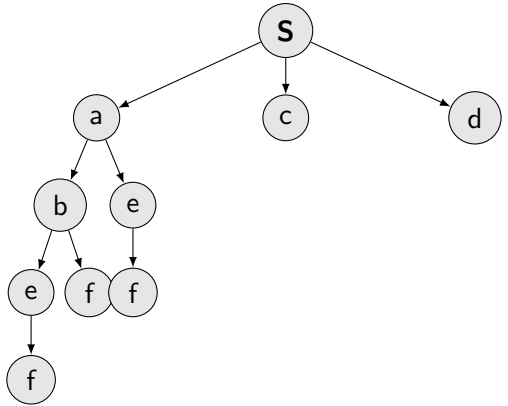
- Time, can process the whole tree: b^m
- Space, only the path so far: bm
- Completness: m may be ∞ hence, not in general
- Optimality: No! It just takes the first solution found.



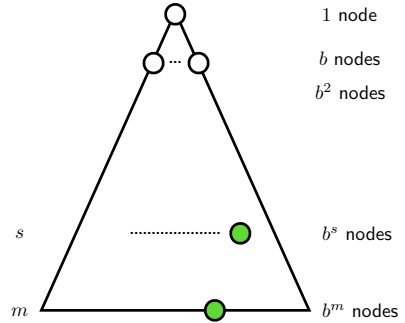
DFS properties

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



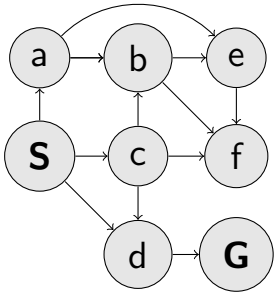


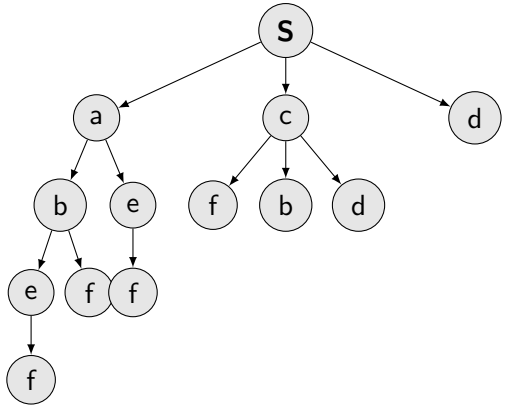
- Time, can process the whole tree: b^m
- Space, only the path so far: bm
- Completness: m may be ∞ hence, not in general
- Optimality: No! It just takes the first solution found.



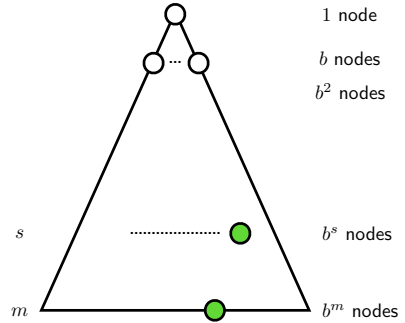
DFS properties

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



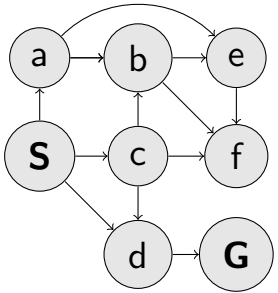


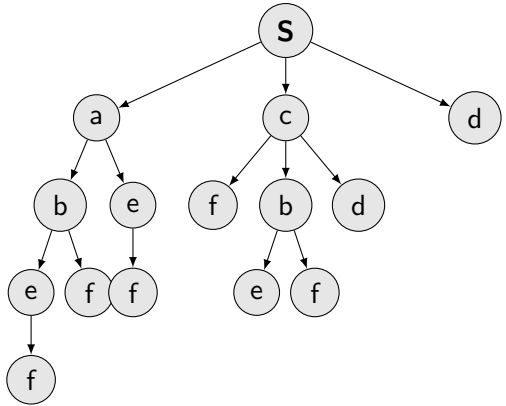
- Time, can process the whole tree: b^m
- Space, only the path so far: bm
- Completness: m may be ∞ hence, not in general
- Optimality: No! It just takes the first solution found.



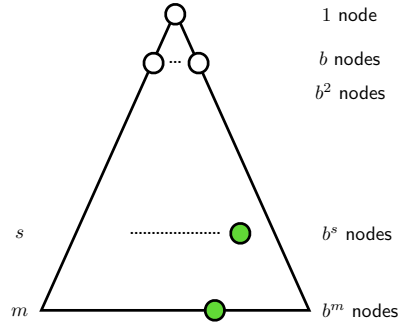
DFS properties

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



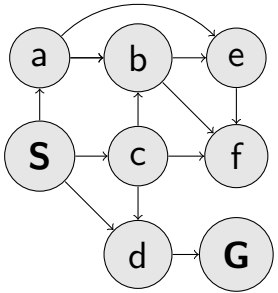


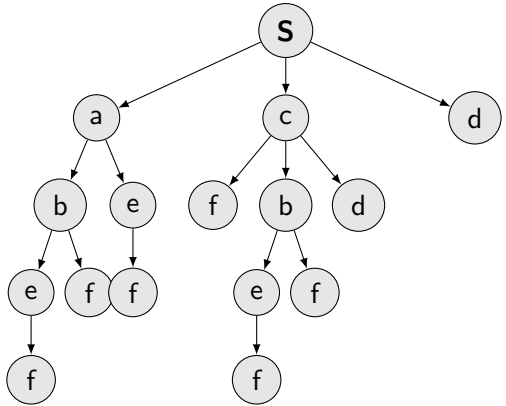
- Time, can process the whole tree: b^m
- Space, only the path so far: bm
- Completness: m may be ∞ hence, not in general
- Optimality: No! It just takes the first solution found.



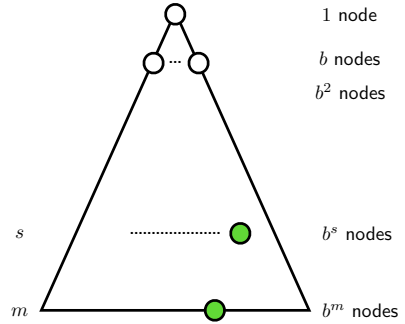
DFS properties

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



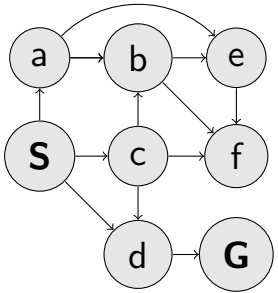


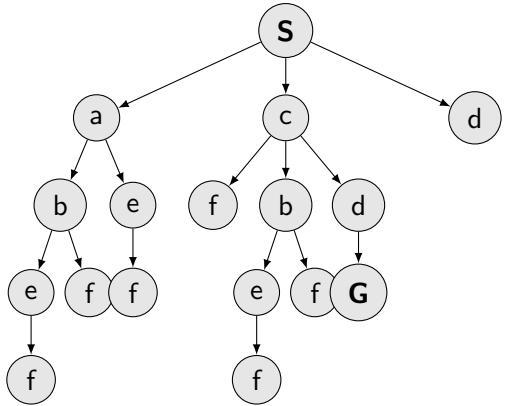
- Time, can process the whole tree: b^m
- Space, only the path so far: bm
- Completness: m may be ∞ hence, not in general
- Optimality: No! It just takes the first solution found.



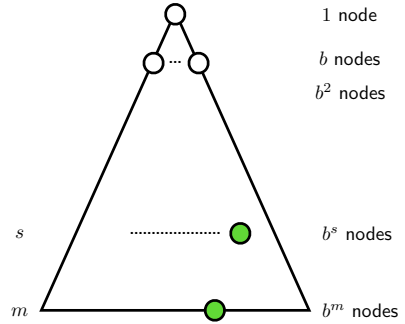
DFS properties

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

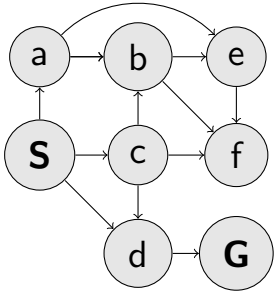




- Time, can process the whole tree: b^m
- Space, only the path so far: bm
- Completness: m may be ∞ hence, not in general
- Optimality: No! It just takes the first solution found.

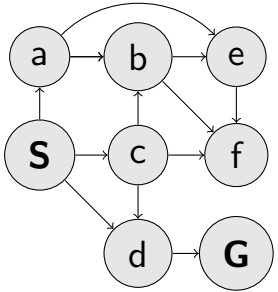


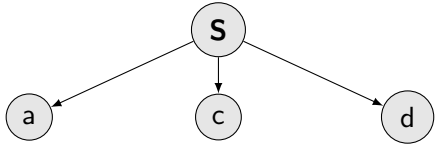
Breadth-First Search (BFS)



What are the BFS properties?

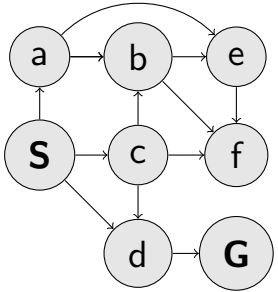
Breadth-First Search (BFS)

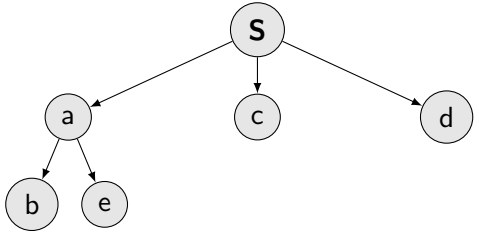




What are the BFS properties?

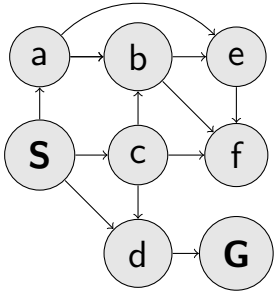
Breadth-First Search (BFS)

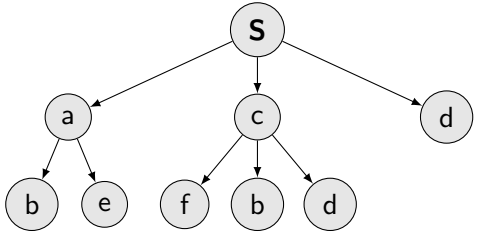




What are the BFS properties?

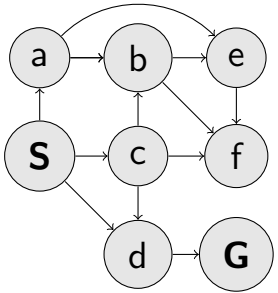
Breadth-First Search (BFS)

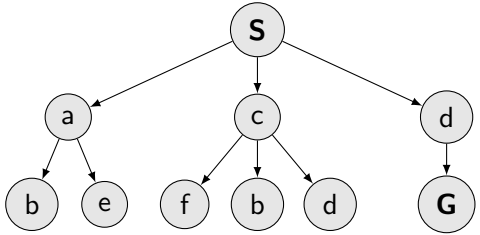




What are the BFS properties?

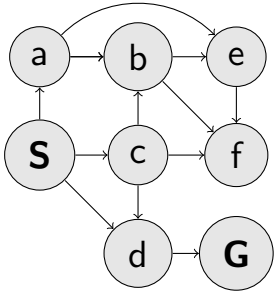
Breadth-First Search (BFS)

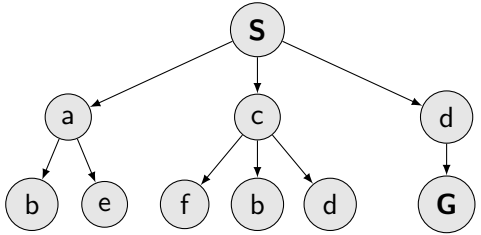




What are the BFS properties?

Breadth-First Search (BFS)

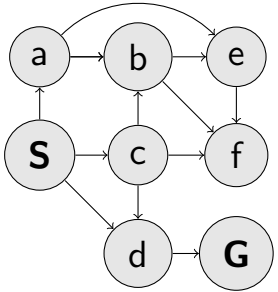




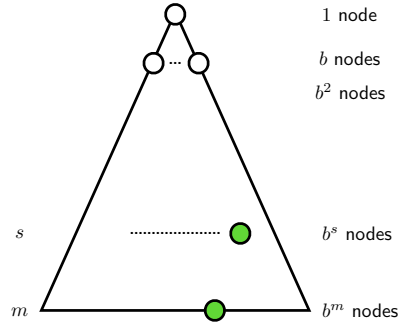
What are the BFS properties?

BFS properties

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

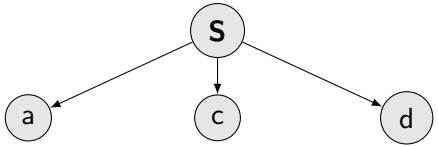


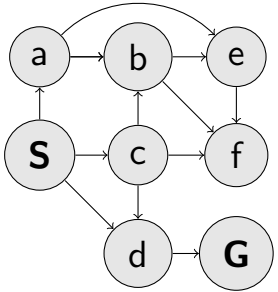
- Time, can process the whole tree until s: b^s, well actually $b + b^2 + b^3 + \cdots + b^s$ but the last layer vastly dominates. Try some calculations for various b.
- Space, all the frontier: b^s
- Completness: Yes!
- Optimality, it does not miss the shallowest solution, hence if all the transition costs are 1: Yes!



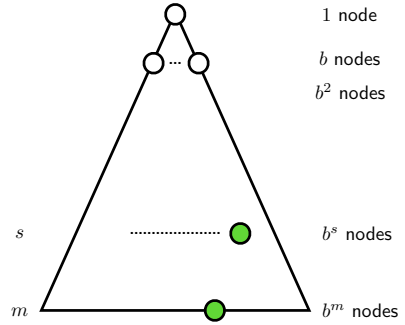
BFS properties

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



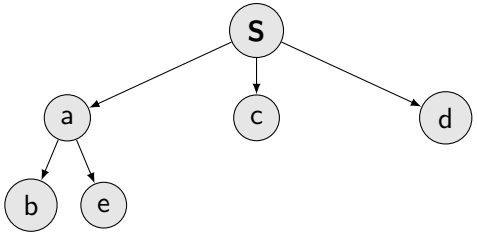


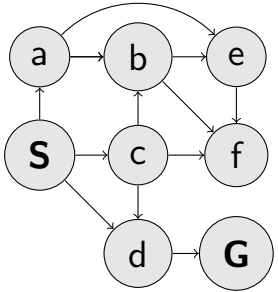
- Time, can process the whole tree until s: b^s, well actually $b + b^2 + b^3 + \cdots + b^s$ but the last layer vastly dominates. Try some calculations for various b.
- Space, all the frontier: b^s
- Completness: Yes!
- Optimality, it does not miss the shallowest solution, hence if all the transition costs are 1: Yes!



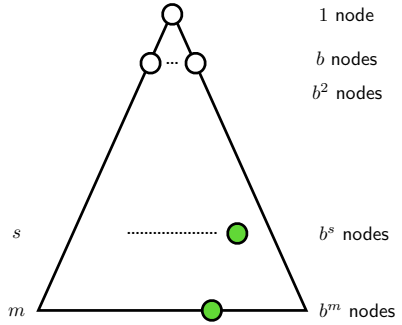
BFS properties

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



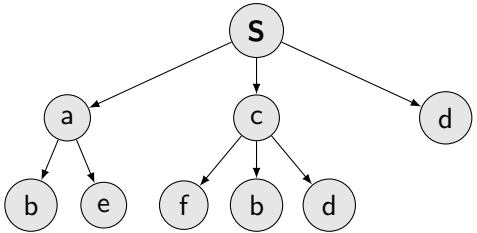


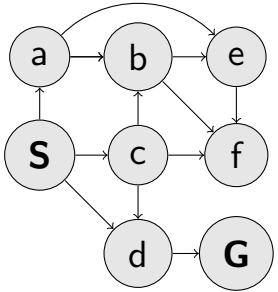
- Time, can process the whole tree until s: b^s, well actually $b + b^2 + b^3 + \cdots + b^s$ but the last layer vastly dominates. Try some calculations for various b.
- Space, all the frontier: b^s
- Completness: Yes!
- Optimality, it does not miss the shallowest solution, hence if all the transition costs are 1: Yes!



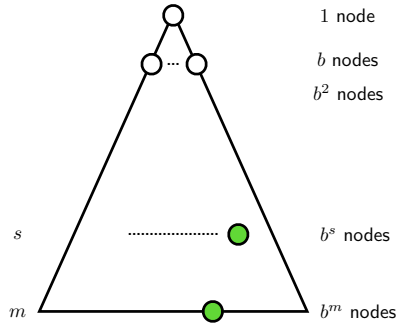
BFS properties

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



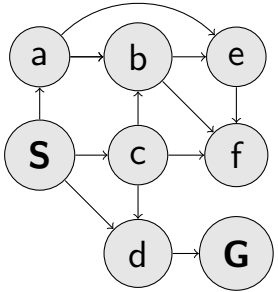


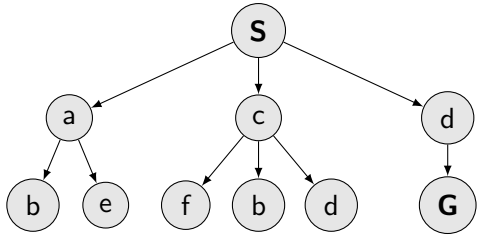
- Time, can process the whole tree until s: b^s, well actually $b + b^2 + b^3 + \cdots + b^s$ but the last layer vastly dominates. Try some calculations for various b.
- Space, all the frontier: b^s
- Completness: Yes!
- Optimality, it does not miss the shallowest solution, hence if all the transition costs are 1: Yes!



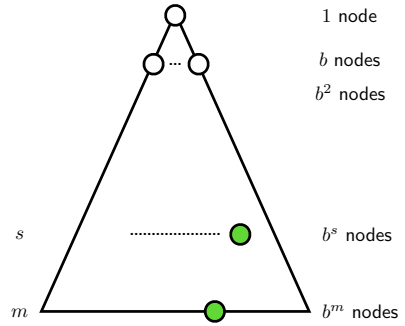
BFS properties

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





- Time, can process the whole tree until s: b^s, well actually $b + b^2 + b^3 + \cdots + b^s$ but the last layer vastly dominates. Try some calculations for various b.
- Space, all the frontier: b^s
- Completness: Yes!
- Optimality, it does not miss the shallowest solution, hence if all the transition costs are 1: Yes!



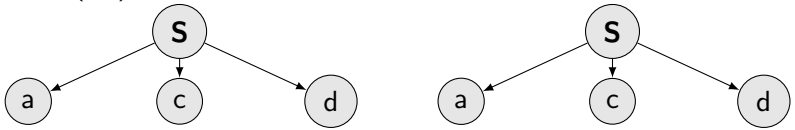
DFS vs BFS

What are (dis)advantages of the individual strategies?

- Do not overfit your thinking!
- Draw for yourself a different graph and contruct appropriate trees.
- Not everything is visible from the animations.
- Draw a comparison tables.

DFS vs BFS

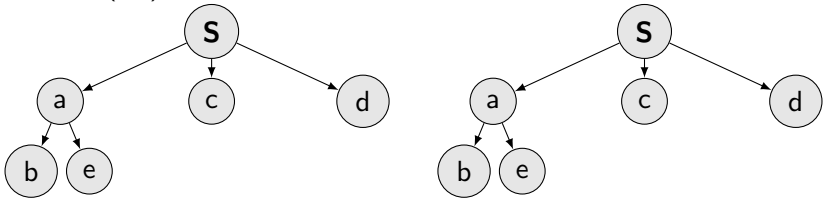
What are (dis)advantages of the individual strategies?



- Do not overfit your thinking!
- Draw for yourself a different graph and contruct appropriate trees.
- Not everything is visible from the animations.
- Draw a comparison tables.

DFS vs BFS

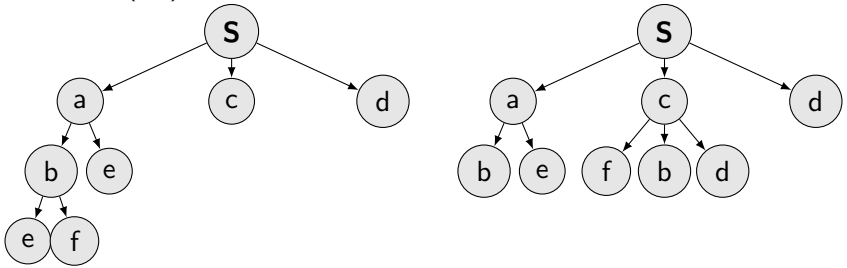
What are (dis)advantages of the individual strategies?



- Do not overfit your thinking!
- Draw for yourself a different graph and contruct appropriate trees.
- Not everything is visible from the animations.
- Draw a comparison tables.

DFS vs BFS

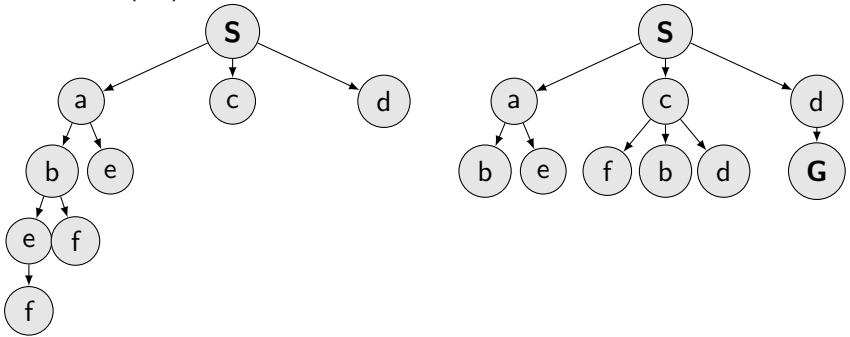
What are (dis)advantages of the individual strategies?



- Do not overfit your thinking!
- Draw for yourself a different graph and contruct appropriate trees.
- Not everything is visible from the animations.
- Draw a comparison tables.

DFS vs BFS

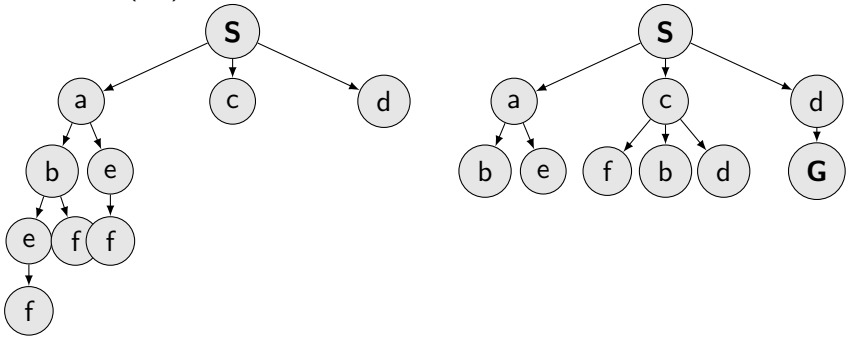
What are (dis)advantages of the individual strategies?



- Do not overfit your thinking!
- Draw for yourself a different graph and contruct appropriate trees.
- Not everything is visible from the animations.
- Draw a comparison tables.

DFS vs BFS

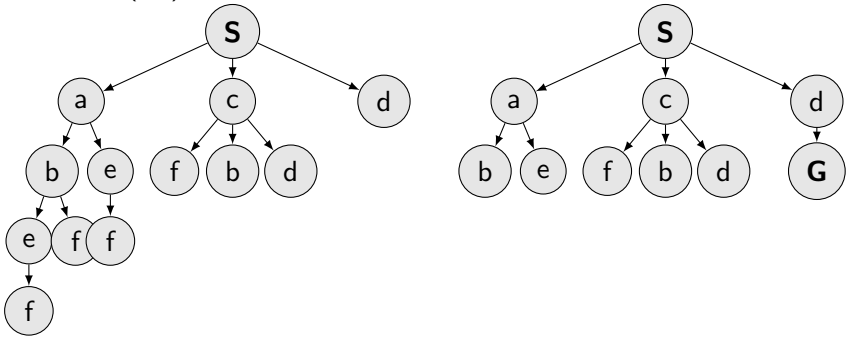
What are (dis)advantages of the individual strategies?



- Do not overfit your thinking!
- Draw for yourself a different graph and contruct appropriate trees.
- Not everything is visible from the animations.
- Draw a comparison tables.

DFS vs BFS

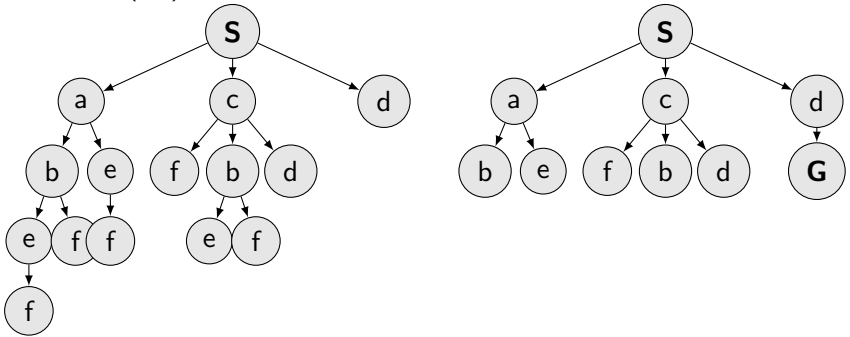
What are (dis)advantages of the individual strategies?



- Do not overfit your thinking!
- Draw for yourself a different graph and contruct appropriate trees.
- Not everything is visible from the animations.
- Draw a comparison tables.

DFS vs BFS

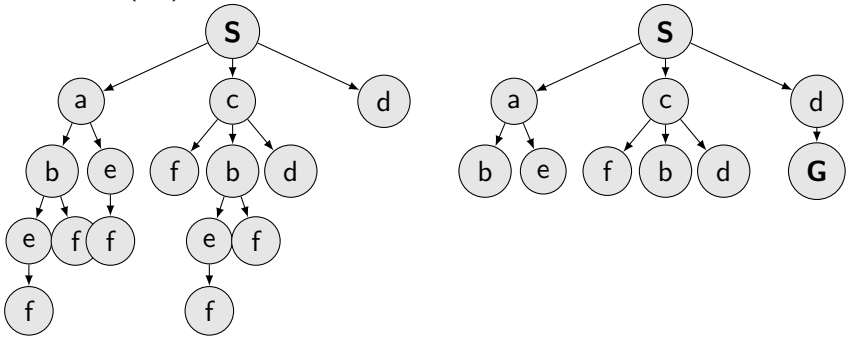
What are (dis)advantages of the individual strategies?



- Do not overfit your thinking!
- Draw for yourself a different graph and contruct appropriate trees.
- Not everything is visible from the animations.
- Draw a comparison tables.

DFS vs BFS

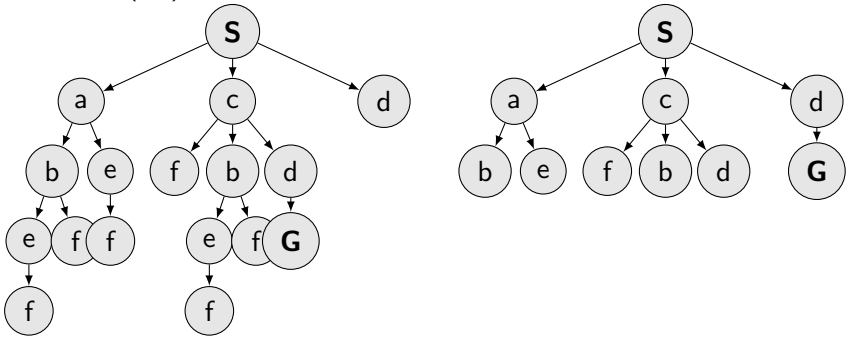
What are (dis)advantages of the individual strategies?



- Do not overfit your thinking!
- Draw for yourself a different graph and contruct appropriate trees.
- Not everything is visible from the animations.
- Draw a comparison tables.

DFS vs BFS

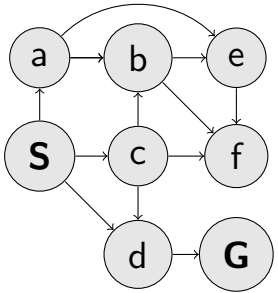
What are (dis)advantages of the individual strategies?



- Do not overfit your thinking!
- Draw for yourself a different graph and contruct appropriate trees.
- Not everything is visible from the animations.
- Draw a comparison tables.

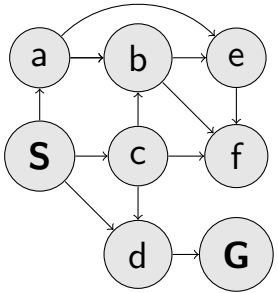
DFS with limited depth, `maxdepth=2`

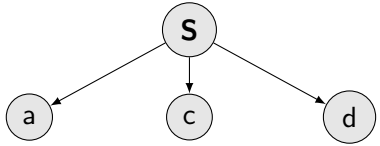
Do not follow nodes with `depth` $>$ `maxdepth`



DFS with limited depth, `maxdepth=2`

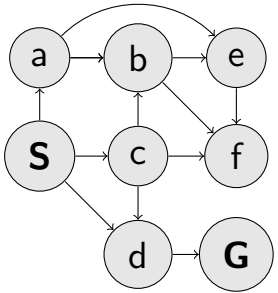
Do not follow nodes with `depth` $>$ `maxdepth`

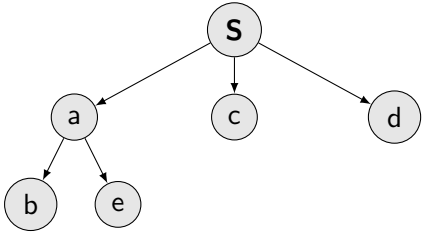




DFS with limited depth, `maxdepth=2`

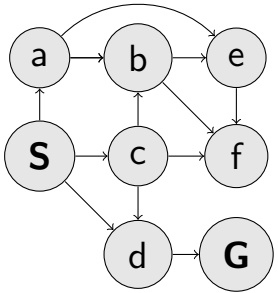
Do not follow nodes with `depth` $>$ `maxdepth`

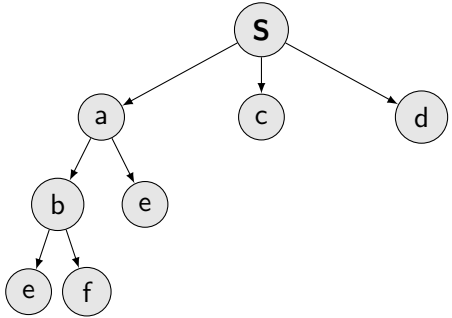




DFS with limited depth, `maxdepth=2`

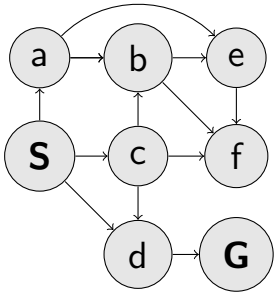
Do not follow nodes with `depth` > `maxdepth`

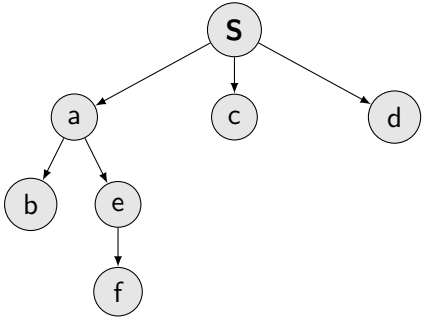




DFS with limited depth, `maxdepth=2`

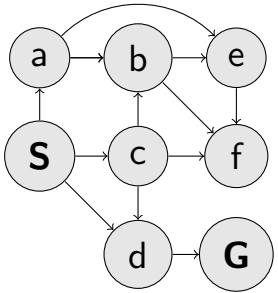
Do not follow nodes with `depth` $>$ `maxdepth`

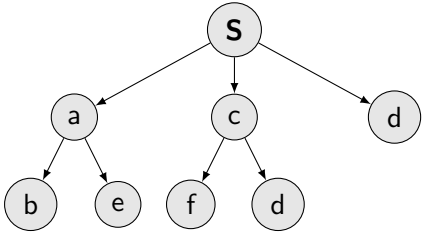




DFS with limited depth, `maxdepth=2`

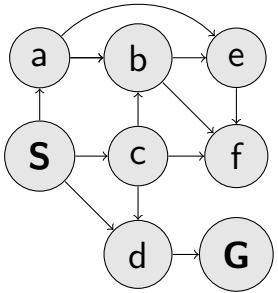
Do not follow nodes with `depth` $>$ `maxdepth`

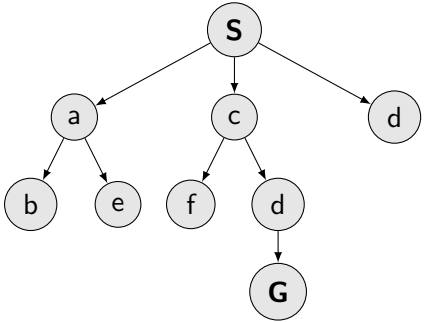




DFS with limited depth, `maxdepth=2`

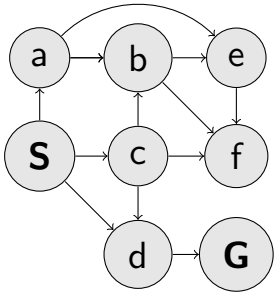
Do not follow nodes with `depth` > `maxdepth`

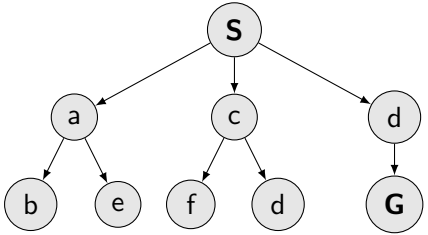




DFS with limited depth, `maxdepth=2`

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

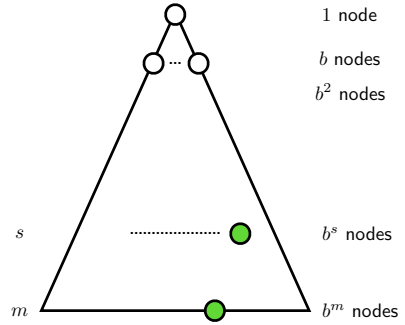
Is it not a terrible waste to forget everything between steps?

Really, how much do we repeat/waste? Compare number fo nodes generated ID-DFS vs. BFS:

$$N(\text{ID-DFS}) = (s)b + (s-1)b^2 + (s-2)b^3 + \cdots + (1)b^s$$

$$N(\text{BFS}) = b + b^2 + b^3 + \cdots + b^s$$

Try some calculations for various s and b.



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

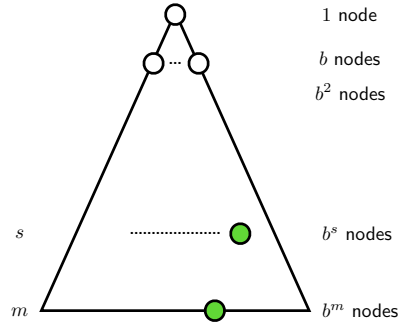
Is it not a terrible waste to forget everything between steps?

Really, how much do we repeat/waste? Compare number fo nodes generated ID-DFS vs. BFS:

$$N(\text{ID-DFS}) = (s)b + (s-1)b^2 + (s-2)b^3 + \cdots + (1)b^s$$

$$N(\text{BFS}) = b + b^2 + b^3 + \cdots + b^s$$

Try some calculations for various s and b.



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

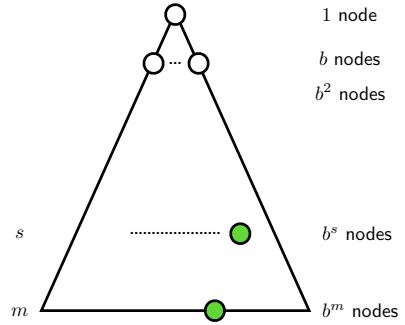
Is it not a terrible waste to forget everything between steps?

Really, how much do we repeat/waste? Compare number fo nodes generated ID-DFS vs. BFS:

$$N(\text{ID-DFS}) = (s)b + (s-1)b^2 + (s-2)b^3 + \cdots + (1)b^s$$

$$N(\text{BFS}) = b + b^2 + b^3 + \cdots + b^s$$

Try some calculations for various s and b.



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

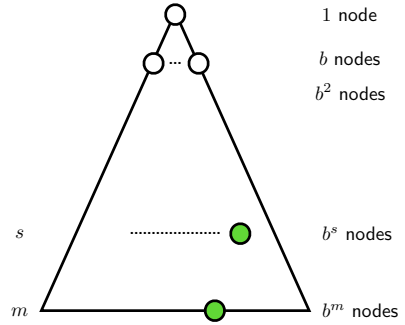
Is it not a terrible waste to forget everything between steps?

Really, how much do we repeat/waste? Compare number fo nodes generated ID-DFS vs. BFS:

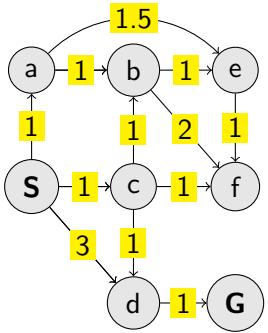
$$N(\text{ID-DFS}) = (s)b + (s-1)b^2 + (s-2)b^3 + \cdots + (1)b^s$$

$$N(\text{BFS}) = b + b^2 + b^3 + \cdots + b^s$$

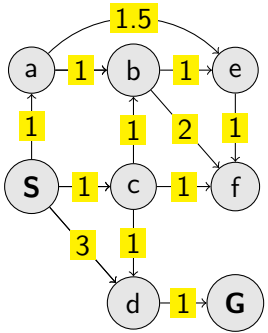
Try some calculations for various s and b.

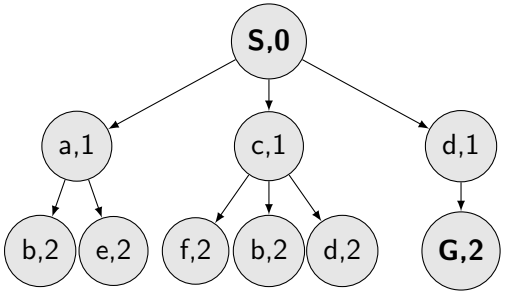


Cost sensitive search



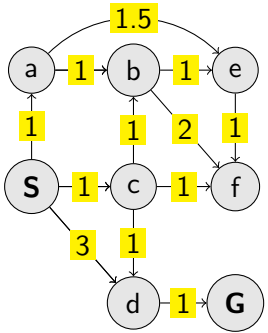
Cost sensitive search

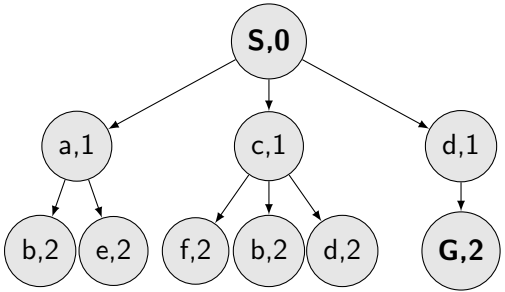




- In BFS, DFS, node ±depth was the node-value.
- How was the depth actually computed?
- How to evaluate nodes with path cost?

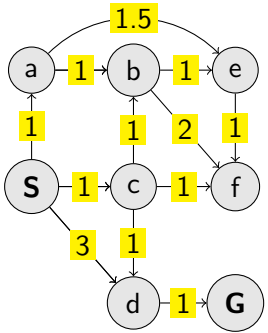
Cost sensitive search

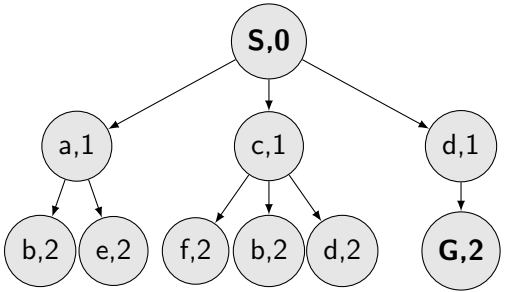




- In BFS, DFS, node ±depth was the node-value.
- How was the depth actually computed?
- How to evaluate nodes with path cost?

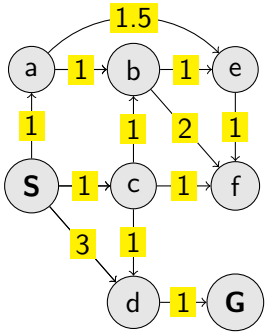
Cost sensitive search





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

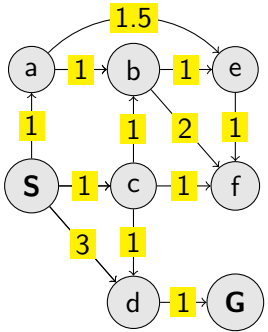
Uniform Cost Search (UCS)

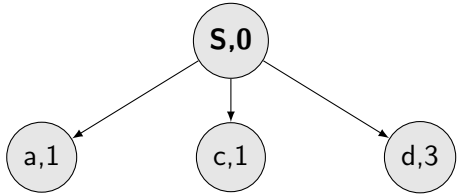


S,0

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

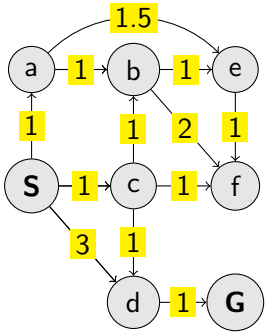
Uniform Cost Search (UCS)

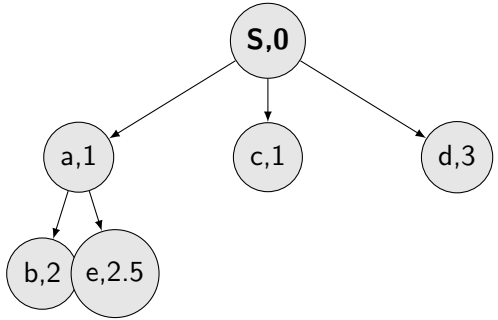




When to check the goal (and stop) the search? When visiting or expanding the node?

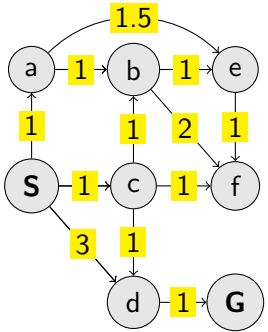
Uniform Cost Search (UCS)

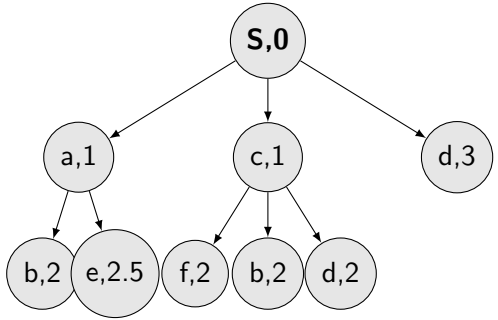




When to check the goal (and stop) the search? When visiting or expanding the node?

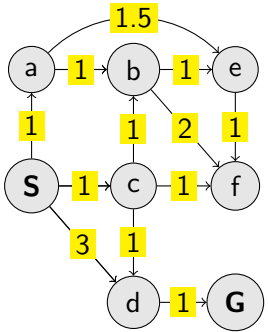
Uniform Cost Search (UCS)

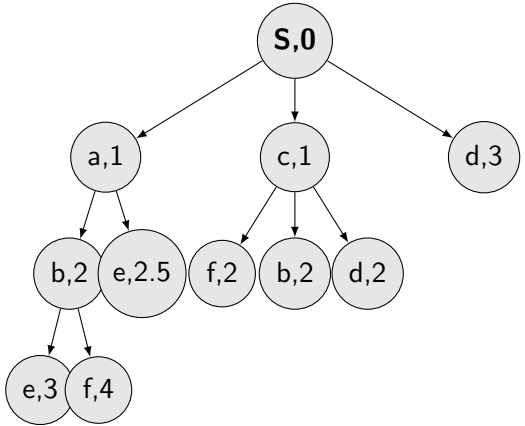




When to check the goal (and stop) the search? When visiting or expanding the node?

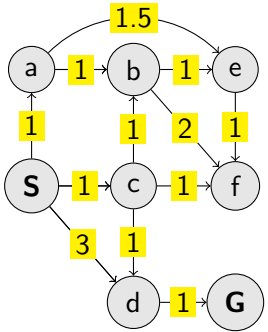
Uniform Cost Search (UCS)

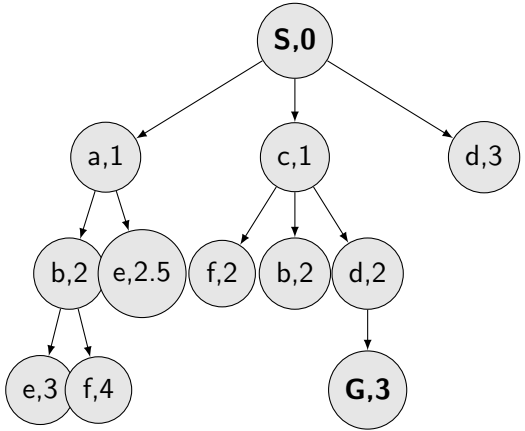




When to check the goal (and stop) the search? When visiting or expanding the node?

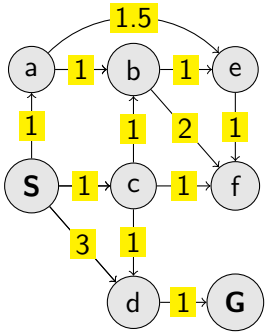
Uniform Cost Search (UCS)

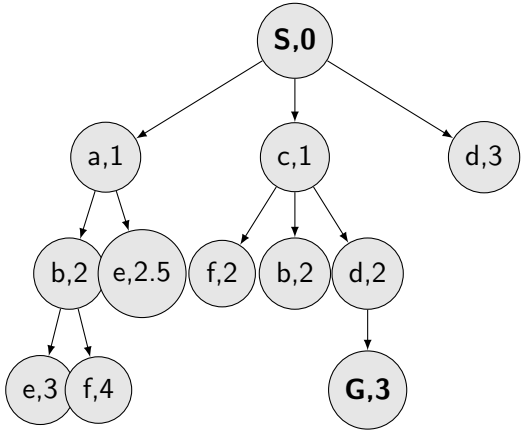




When to check the goal (and stop) the search? When visiting or expanding the node?

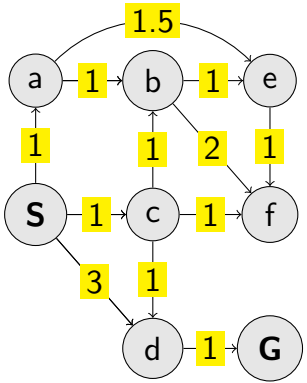
Uniform Cost Search (UCS)



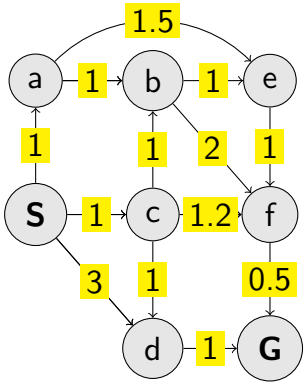


When to check the goal (and stop) the search? When visiting or expanding the node?

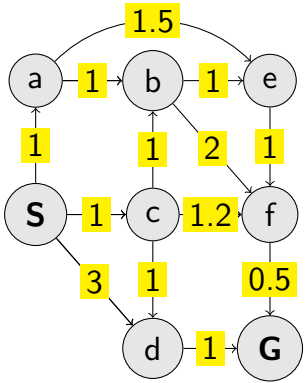
When to stop, when visiting or expanding?



When to stop, when visiting or expanding?

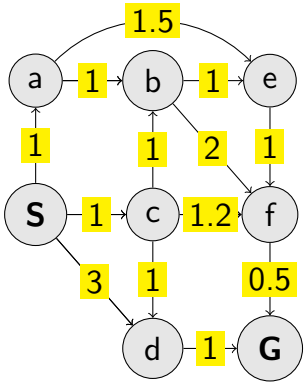


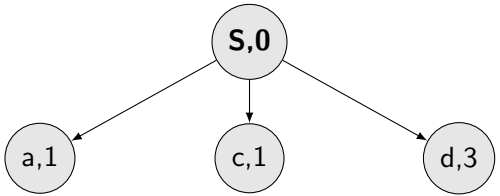
When to stop, when visiting or expanding?



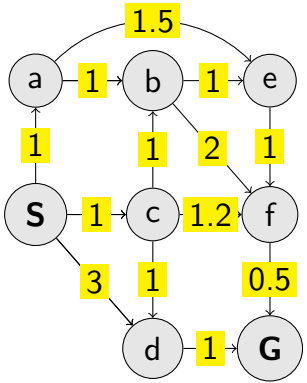
S,0

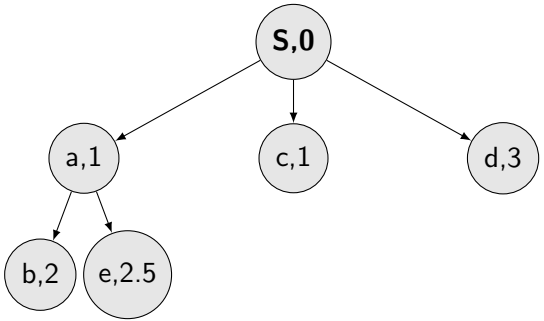
When to stop, when visiting or expanding?



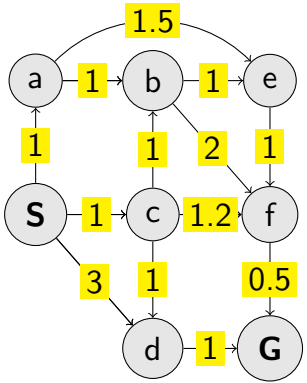


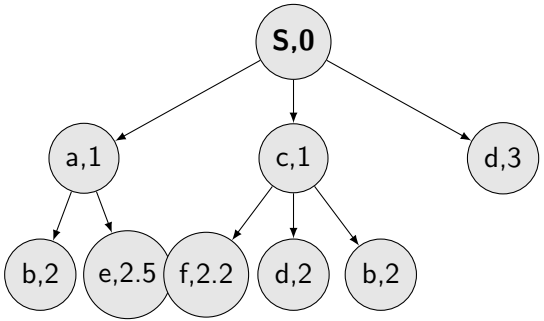
When to stop, when visiting or expanding?



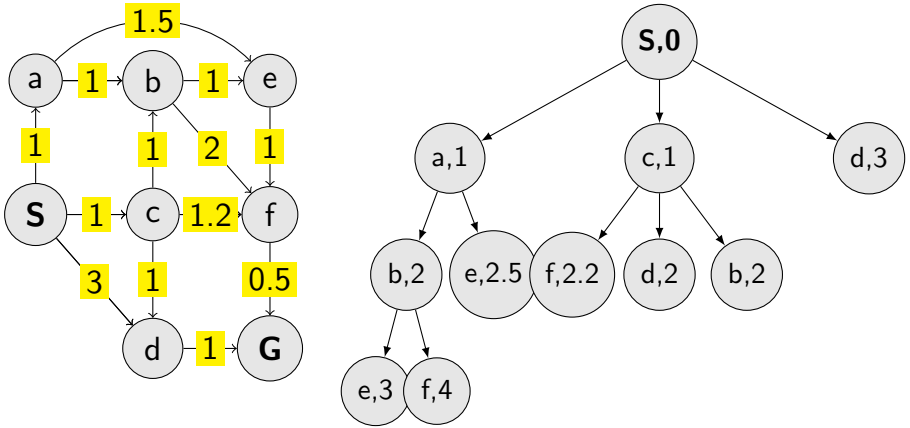


When to stop, when visiting or expanding?

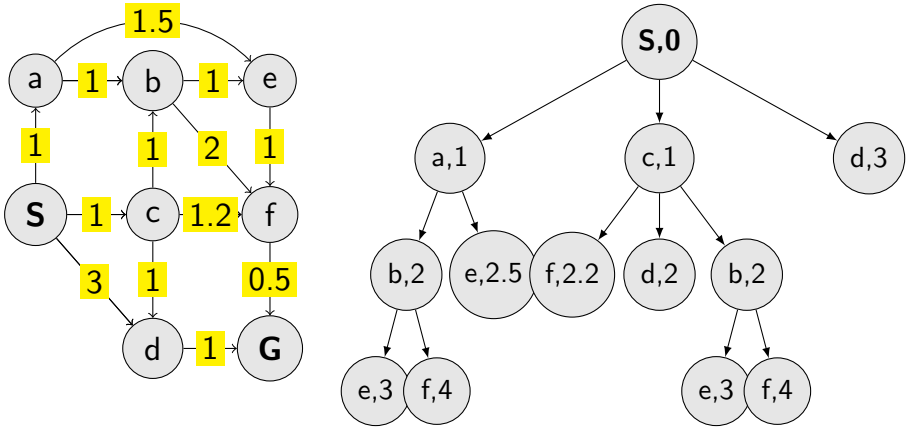




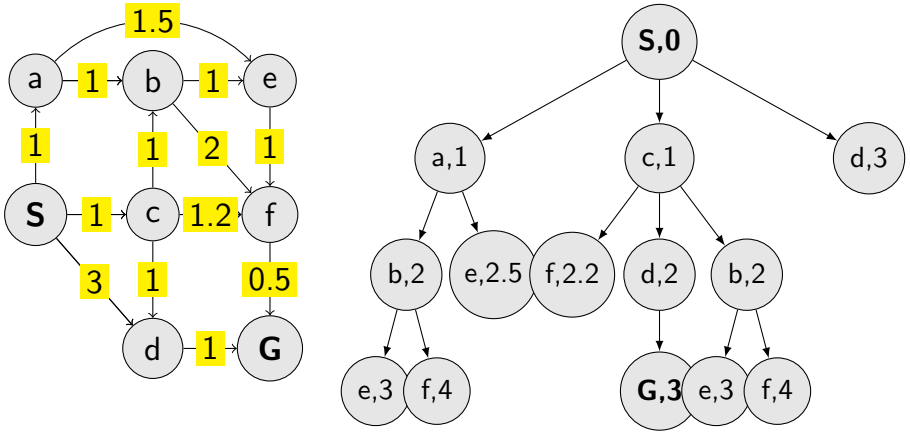
When to stop, when visiting or expanding?



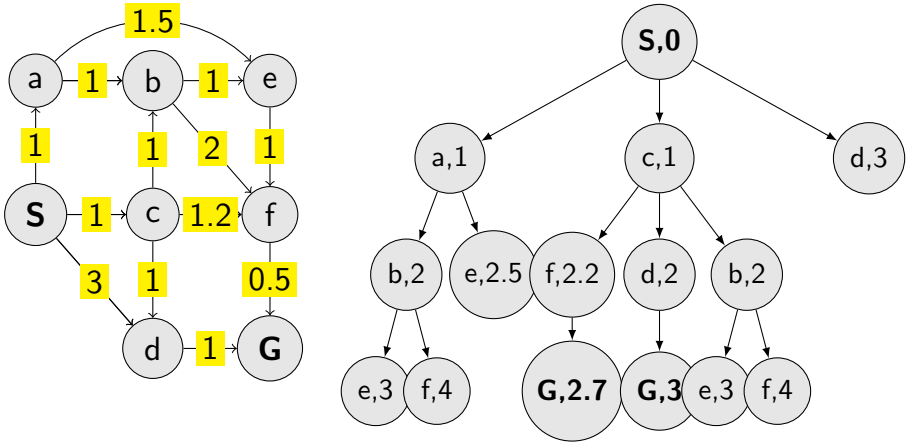
When to stop, when visiting or expanding?



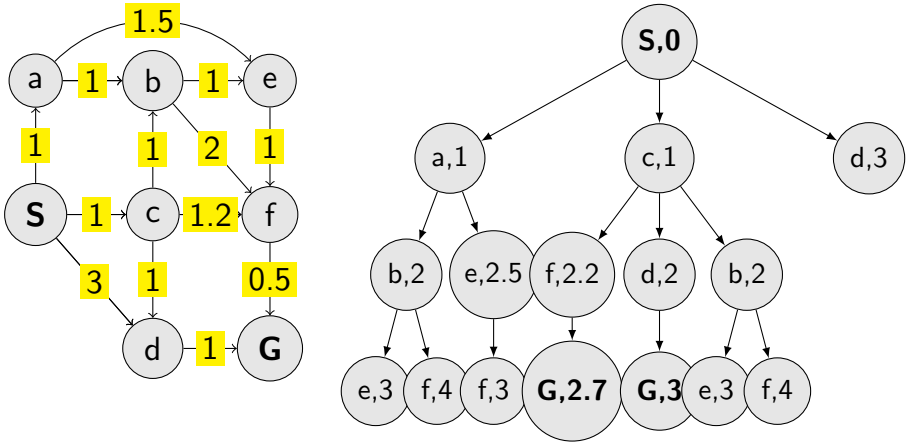
When to stop, when visiting or expanding?



When to stop, when visiting or expanding?



When to stop, when visiting or expanding?

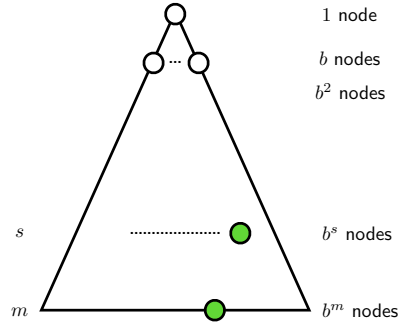


UCS properties

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

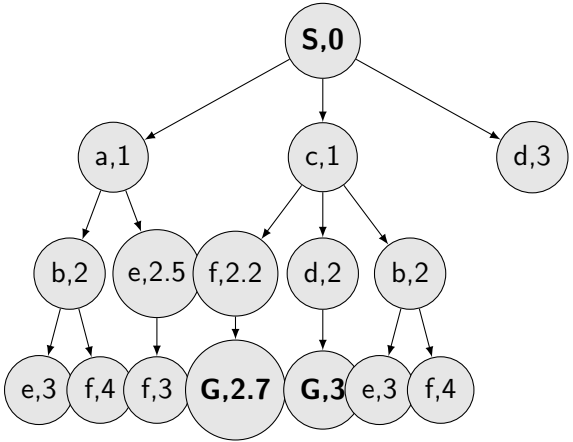
Solution cost C^*, transition cost at least ϵ. Effective depth, roughly C^*/ϵ.

- Time: $b^{C^*/\epsilon}$
- Space: $b^{C^*/\epsilon}$
- Completness: Yes!
- Optimality: Yes! Why?



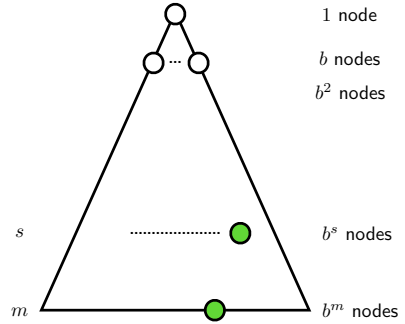
UCS properties

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

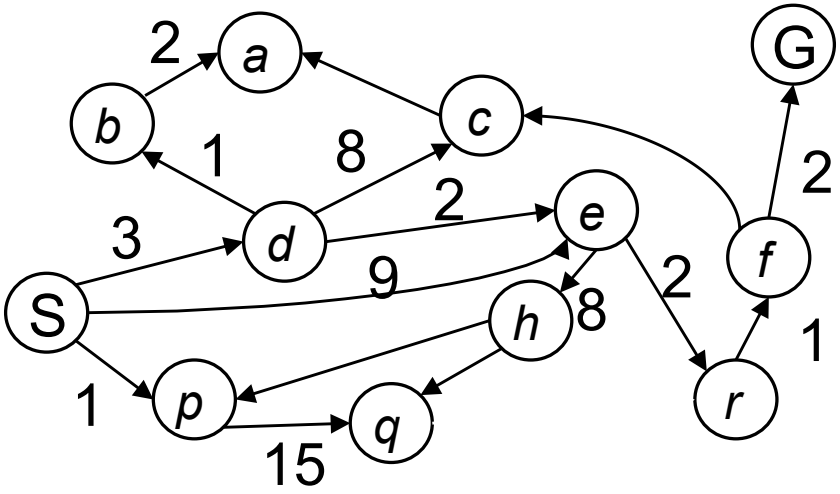


Solution cost C^*, transition cost at least ϵ. Effective depth, roughly C^*/ϵ.

- Time: $b^{C^*/\epsilon}$
- Space: $b^{C^*/\epsilon}$
- Completness: Yes!
- Optimality: Yes! Why?



Example: Graph with costs



Try it on paper, mark which nodes are in frontier, mark lines of equal cost.

Programming a Tree Search

Infrastructure for (tree) search algorithms

What should a tree node `n` now?

- `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` now?

- `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 contruct the path?

References

Some figures if from [1].

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