Problem solving by search

Tomáš Svoboda

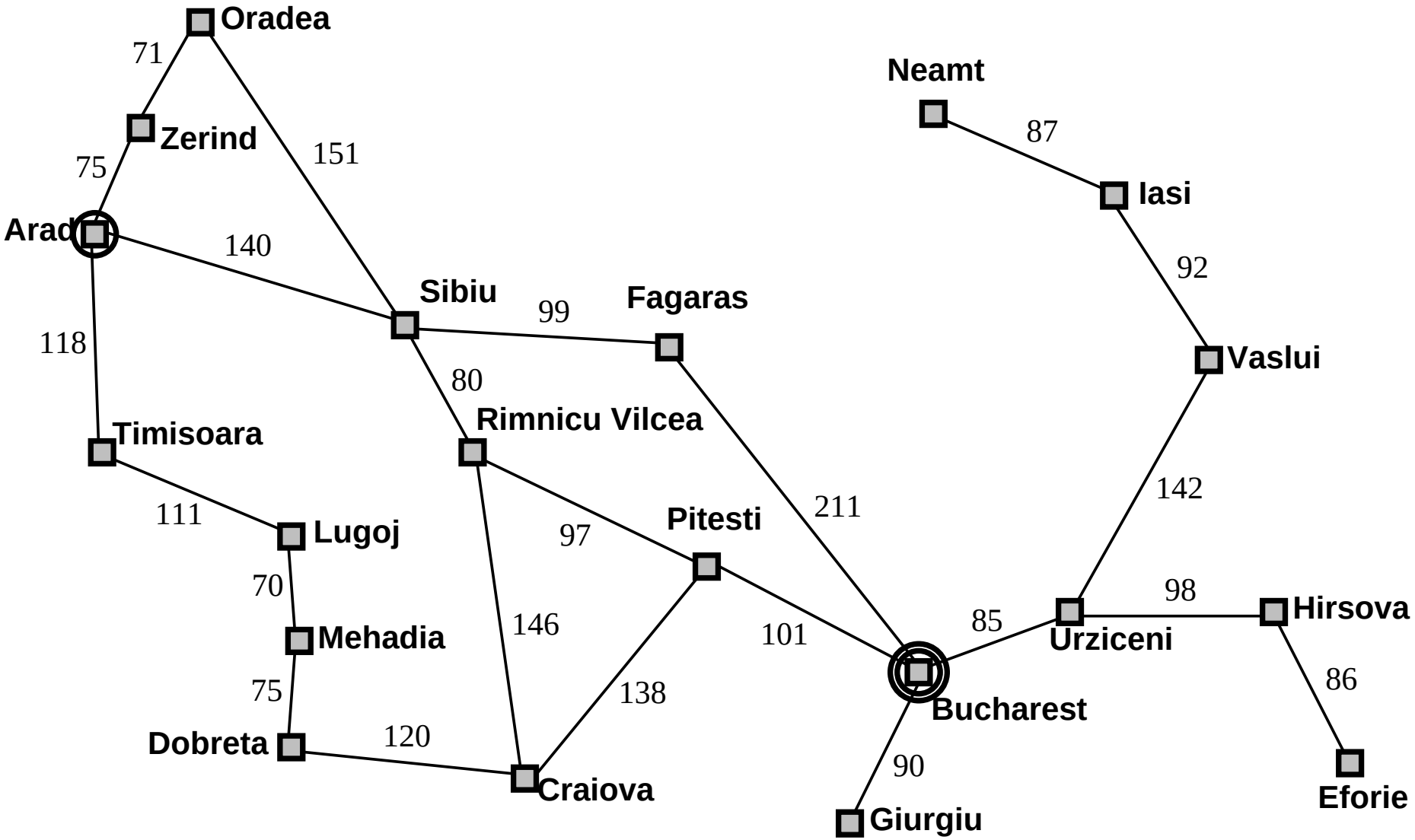
Department of Cybernetics, Vision for Robots and Autonomous Systems

March 5, 2018

Outline

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

Example: Romania



Example: Romania

Goal:

be in Bucharest

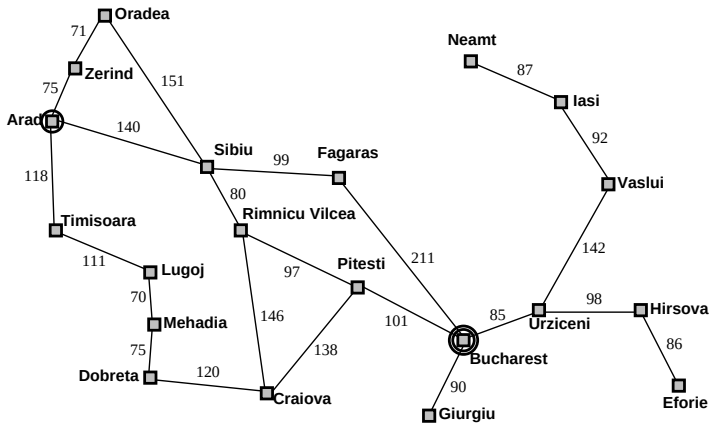
Problem formulation:

states: position in a city (cities)

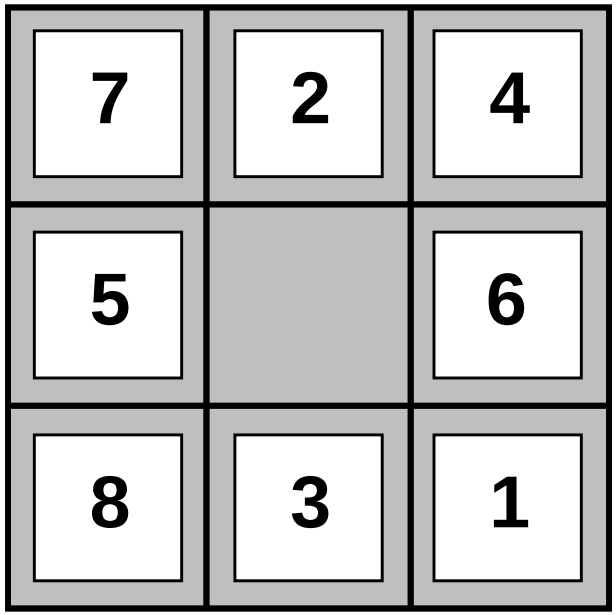
actions: drive between cities

Solution:

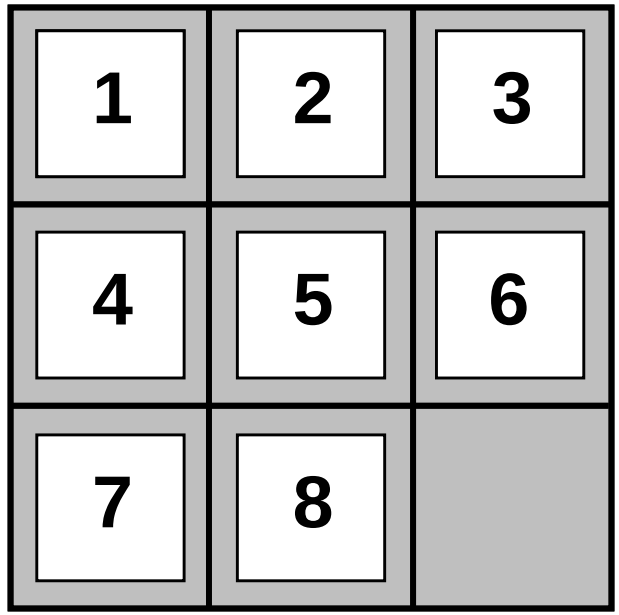
Sequence of cities (path)



Example: The 8-puzzle



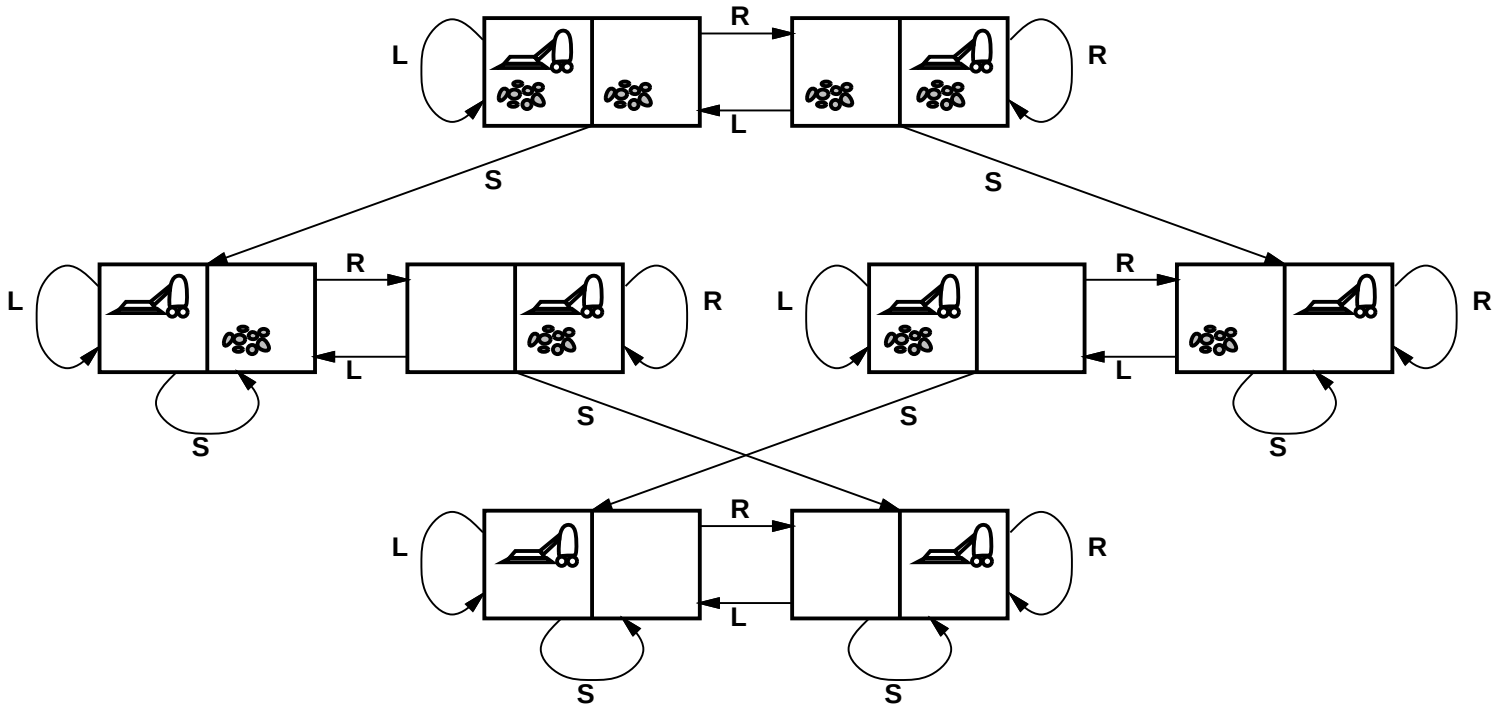
Start State



Goal State

states?
actions?
solution?
cost?

Example: Vacuum cleaner



states?

actions?

solution?

cost?

A Search Problem

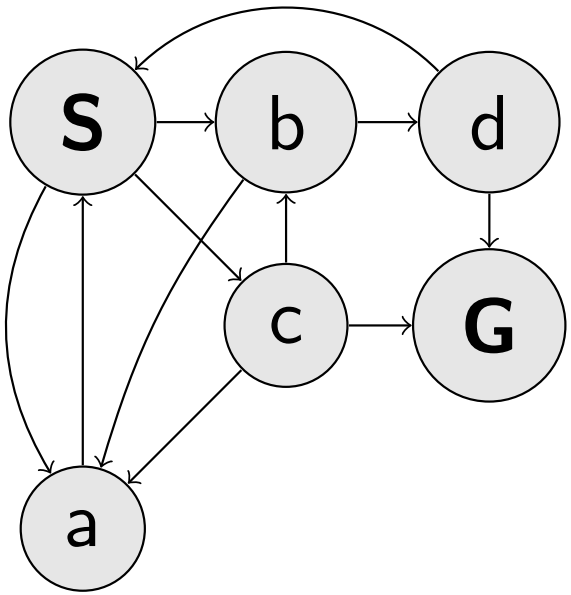
- State space (including Start/Initial state): position, board configuration,
- Actions: drive to, Up, Down, Left ...
- Transition model: Given state and action return state (and cost)
- Goal test: Are we done?

State Space Graphs

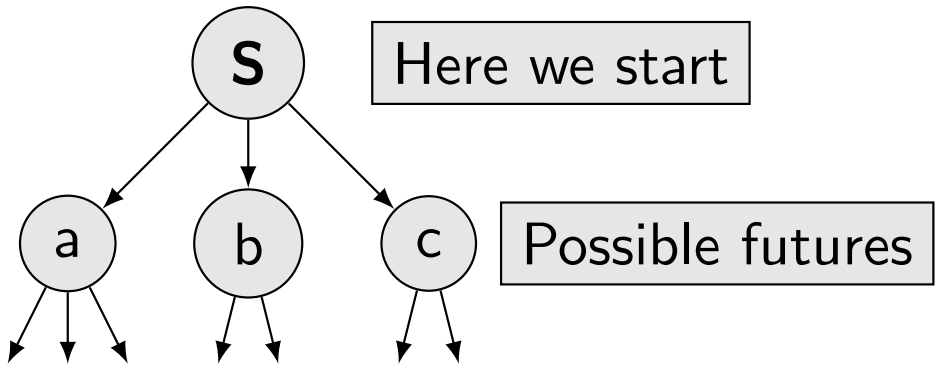
State space graph: a representation of a search problem

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

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



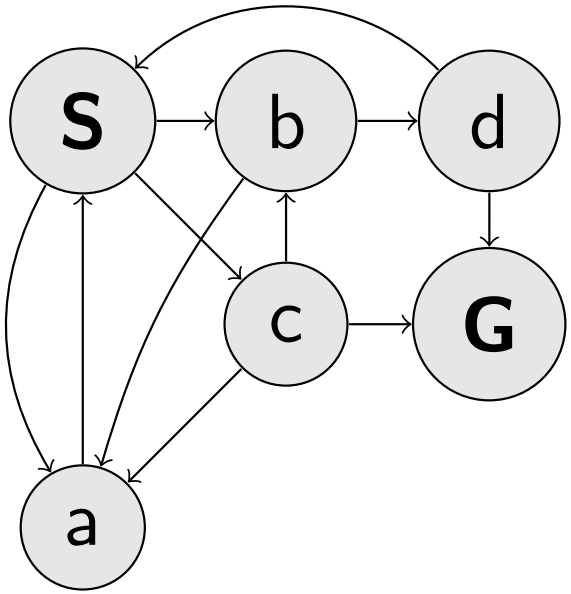
Search Trees

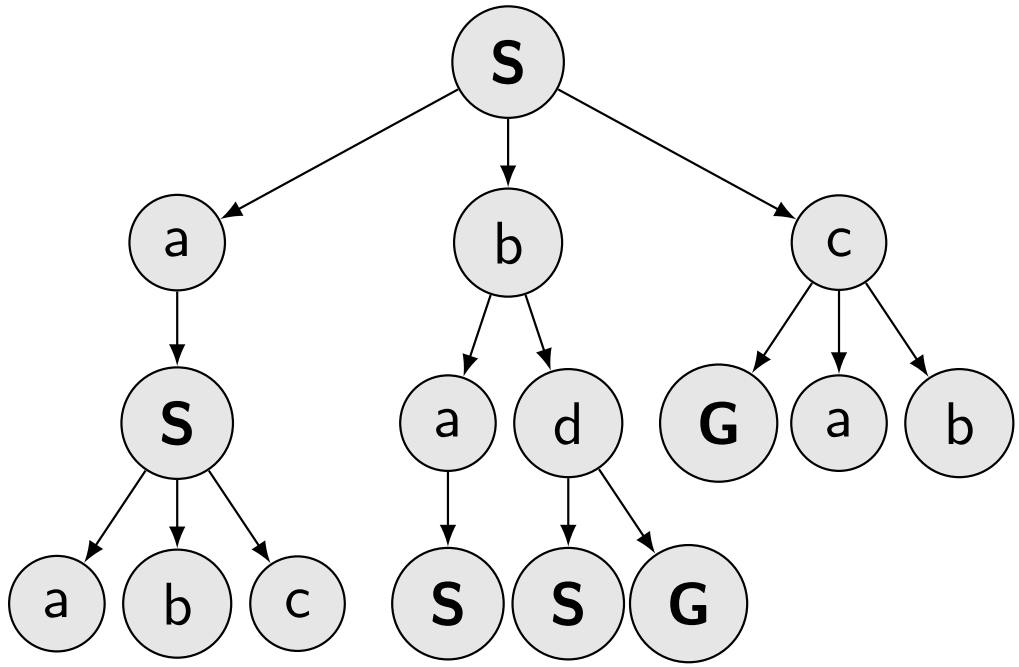


- A "what if" tree of plans and their outcomes
- Start node is the root
- Children are successors
- Nodes show states, but correspond to *plans* that achieve those states

What does the last item mean, actually?

State Space Graphs vs. Search Trees

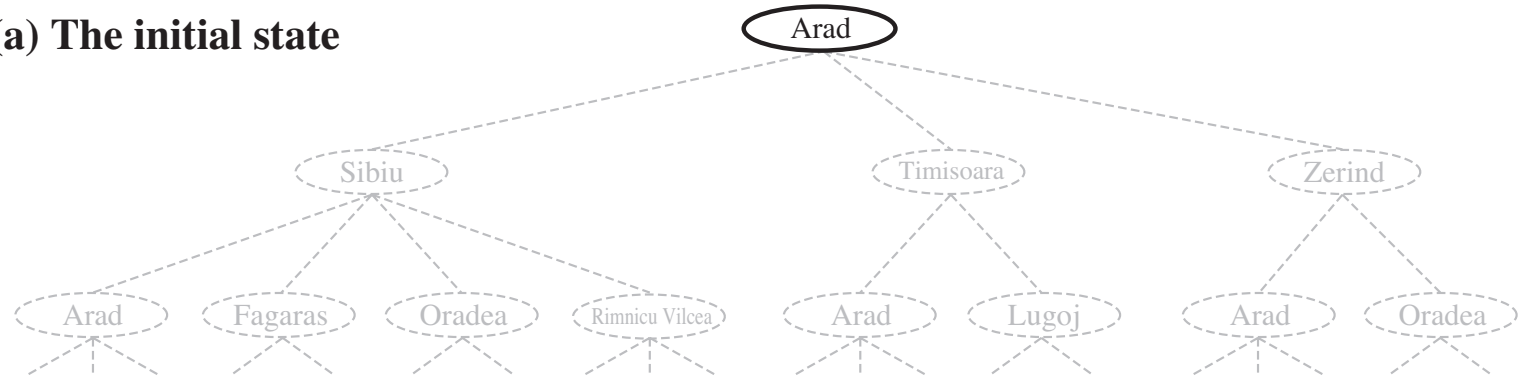




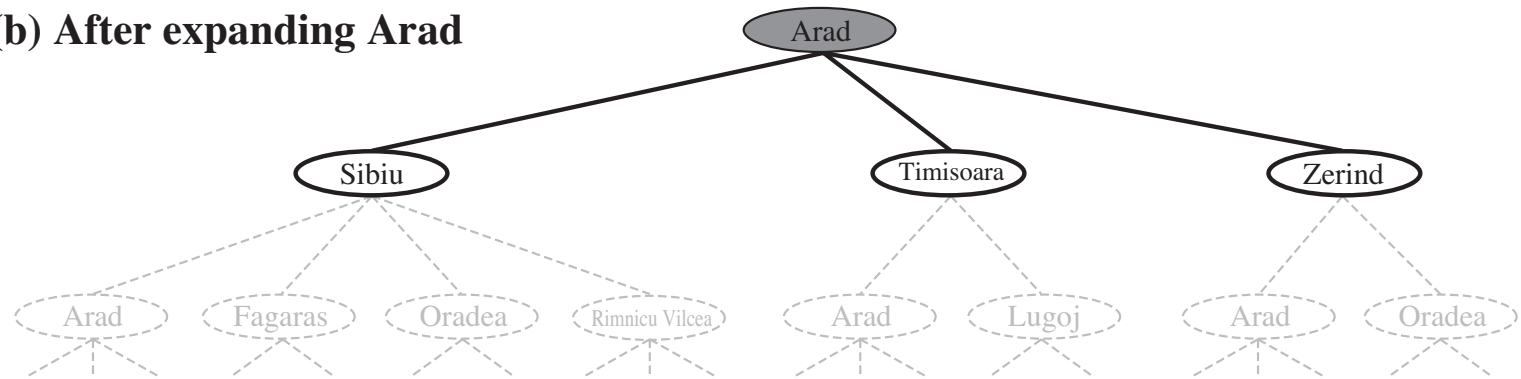
How big is the search tree?

Search tree for Romania

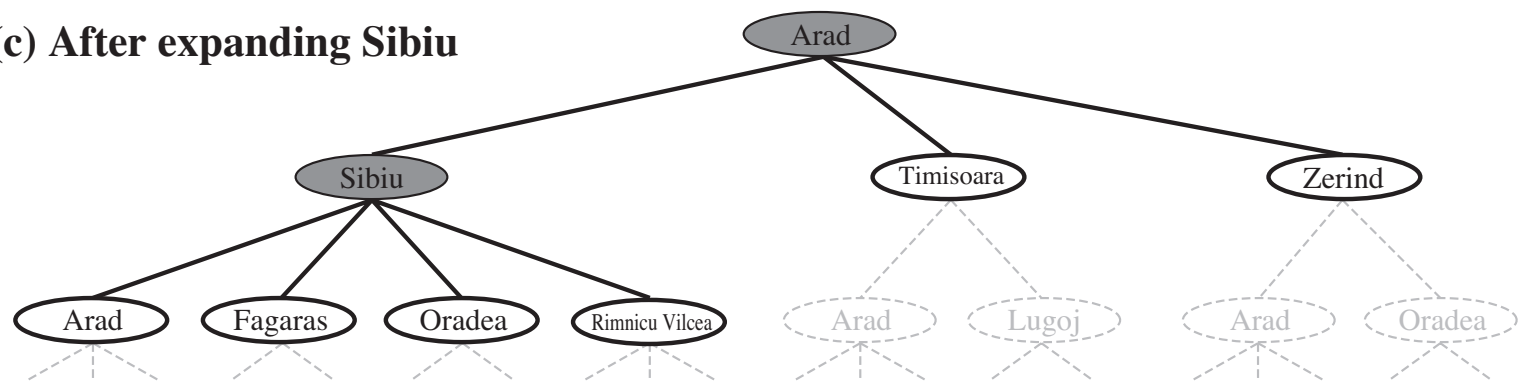
(a) The initial state



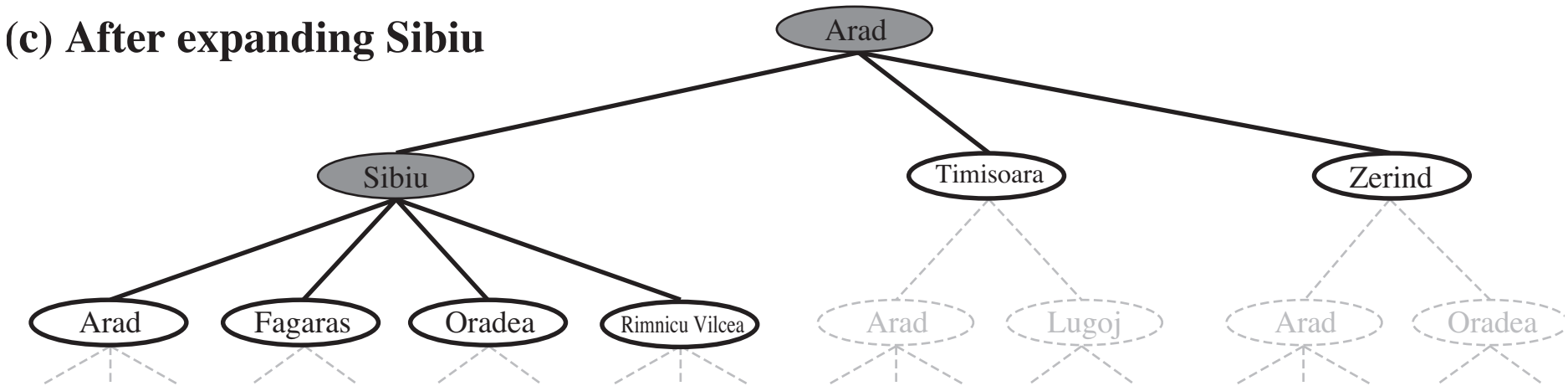
(b) After expanding Arad



(c) After expanding Sibiu

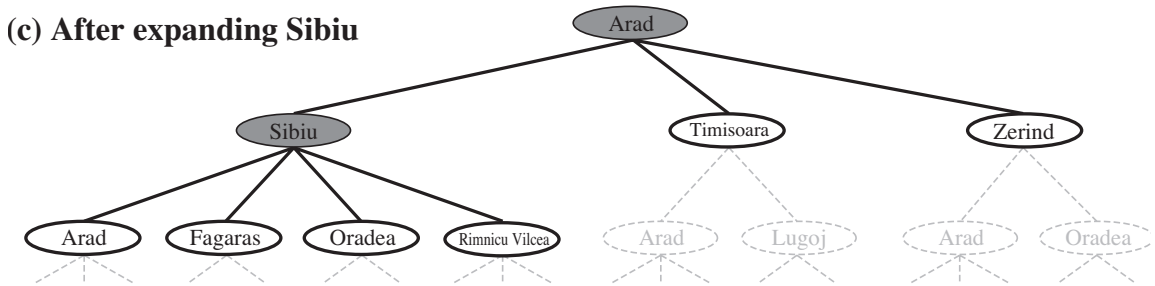


Search elements



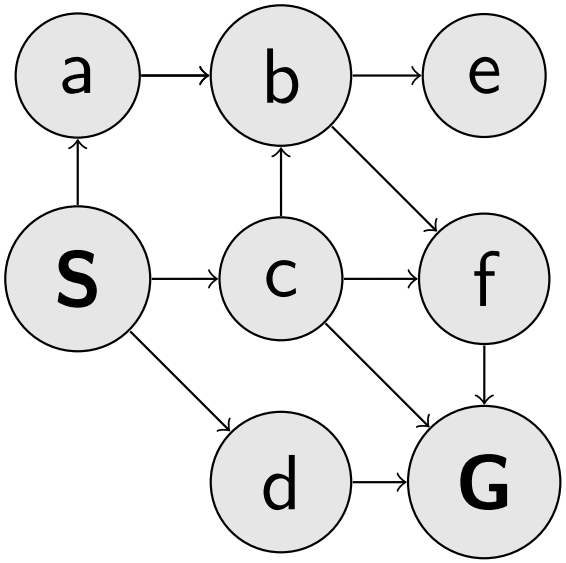
- Expand plans - possible ways (tree nodes).
- Manage/Maintan fringe (or frontier) of plans under consideration.
- Expand new nodes *wisely(?).*

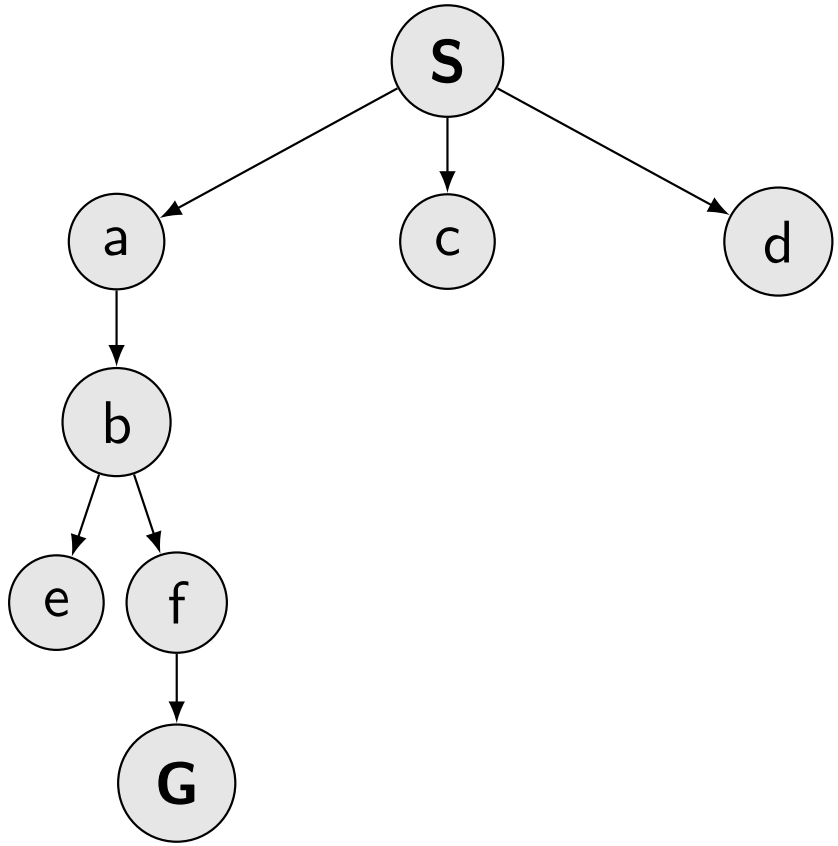
Tree search algorithm



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

Example of a tree search





Which nodes to *explore*?

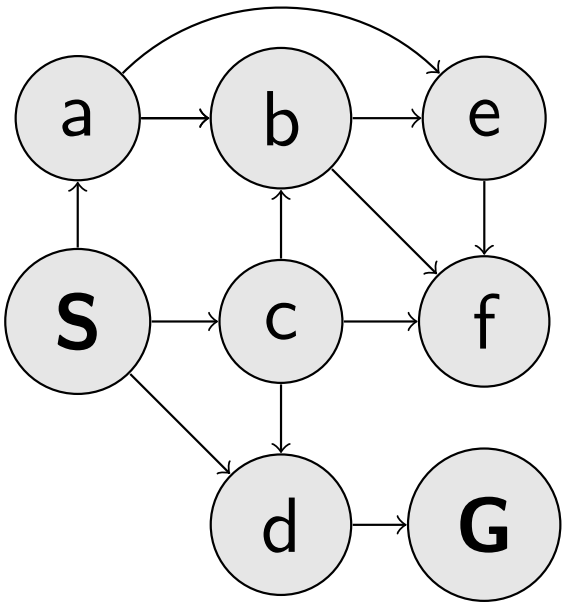
What are the properties of a strategy/algorithm?

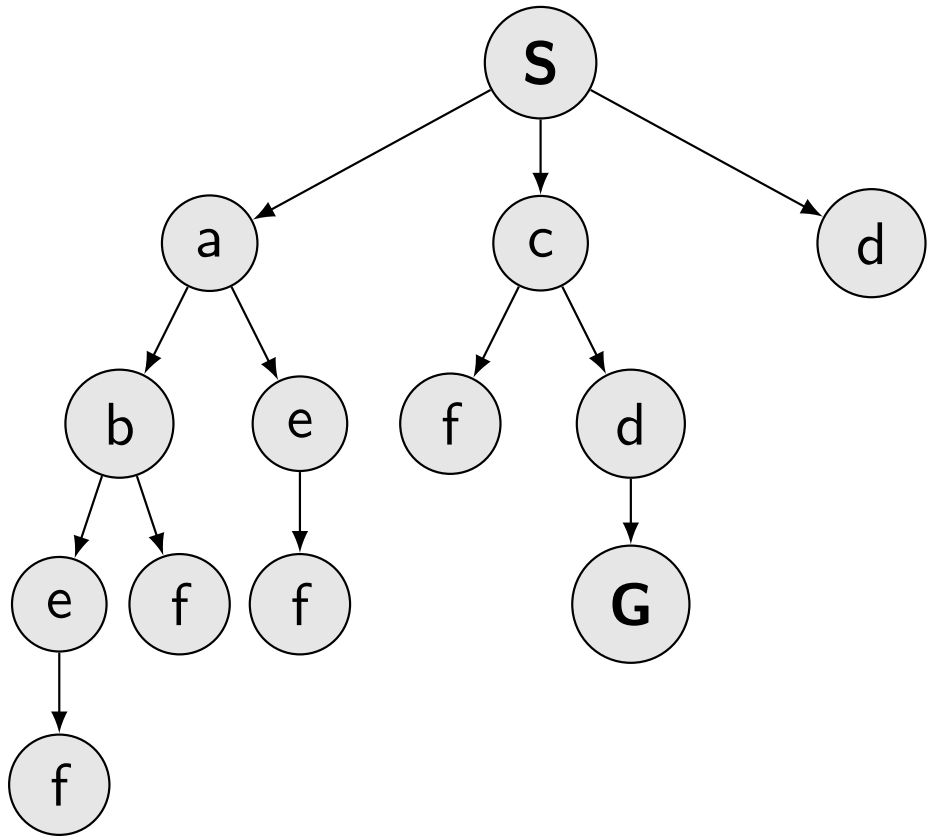
Search (algorithm) properties

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

How many nodes in a tree? What are tree parameters?

Depth-First Search (DFS)



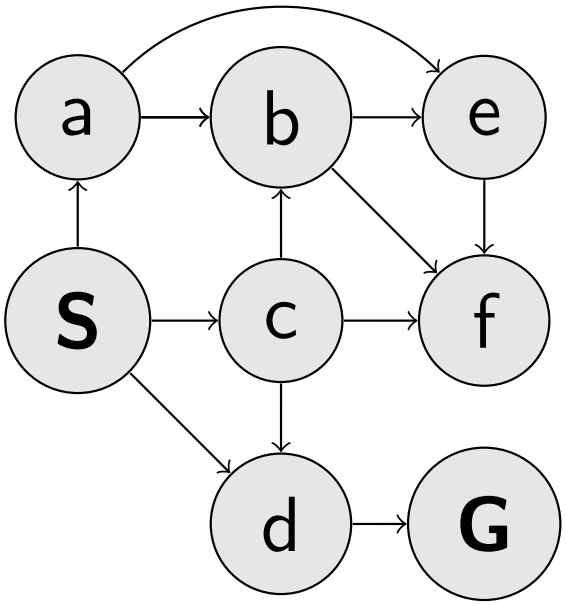


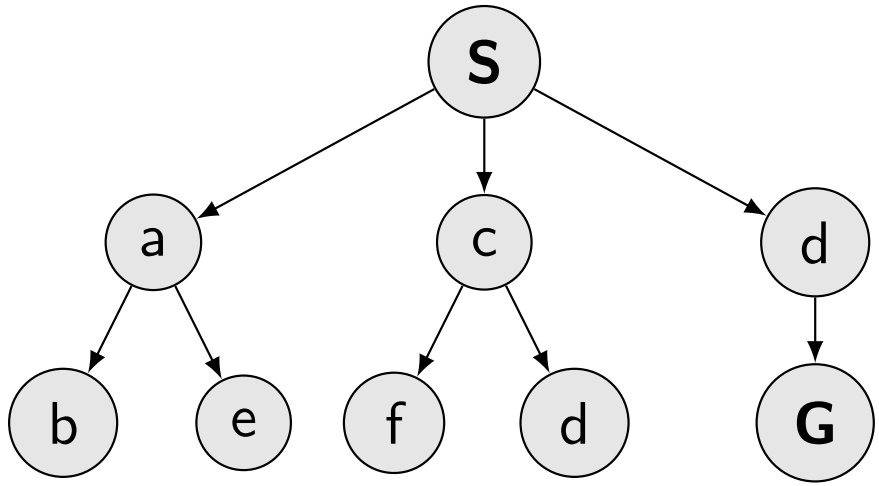
What are the DFS properties?

DFS properties

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

Breadth-First Search (BFS)



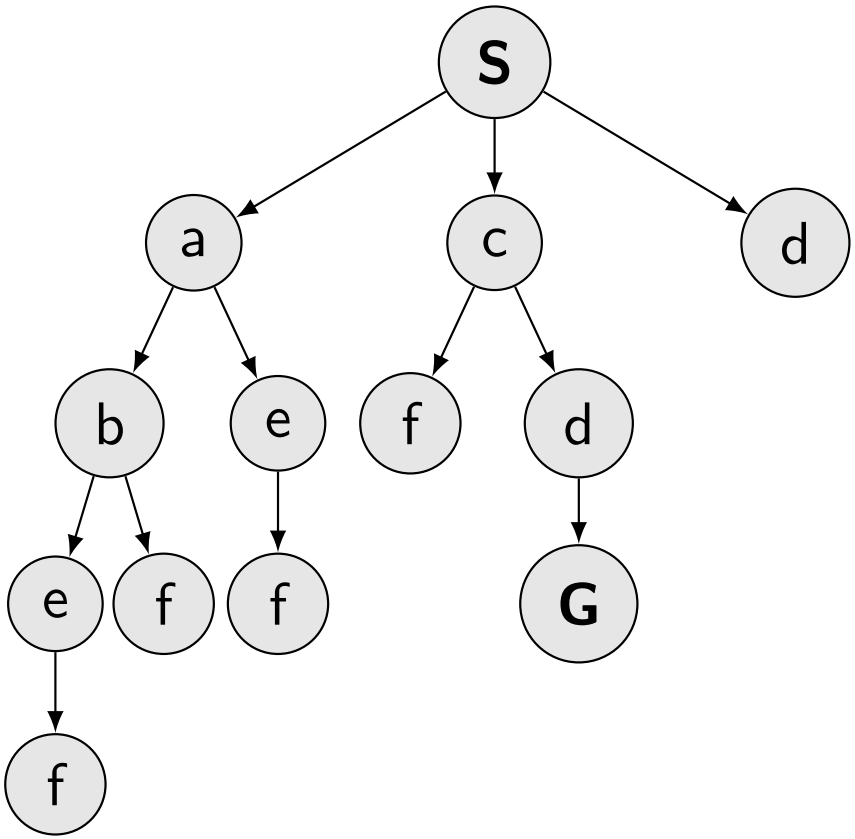


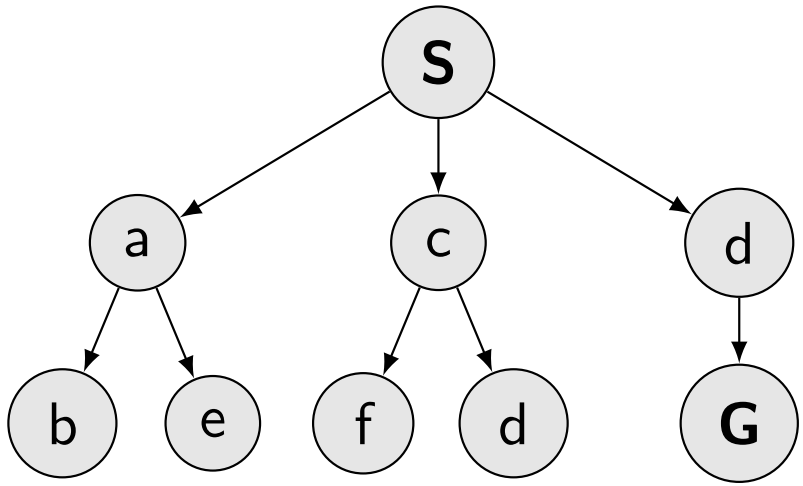
What are the BFS properties?

BFS properties

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

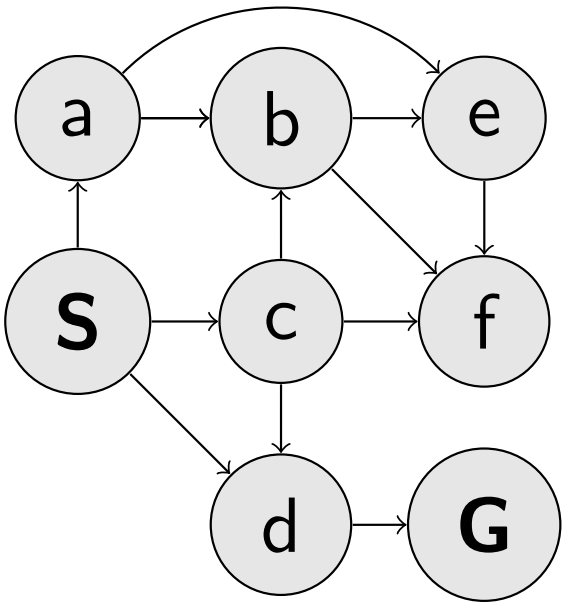
DFS vs BFS

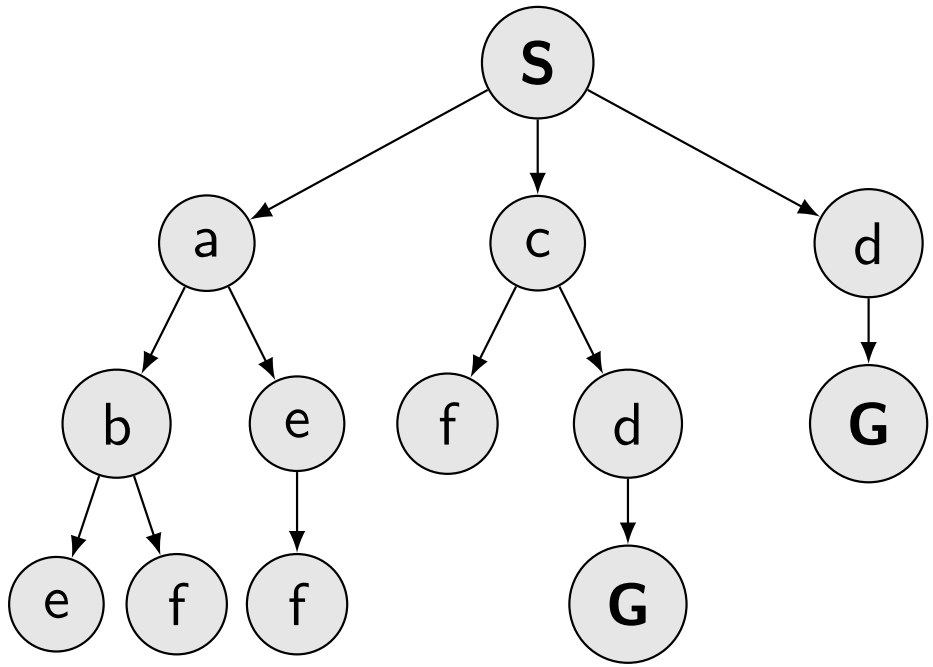




DFS with limited depth, `maxdepth=2`

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



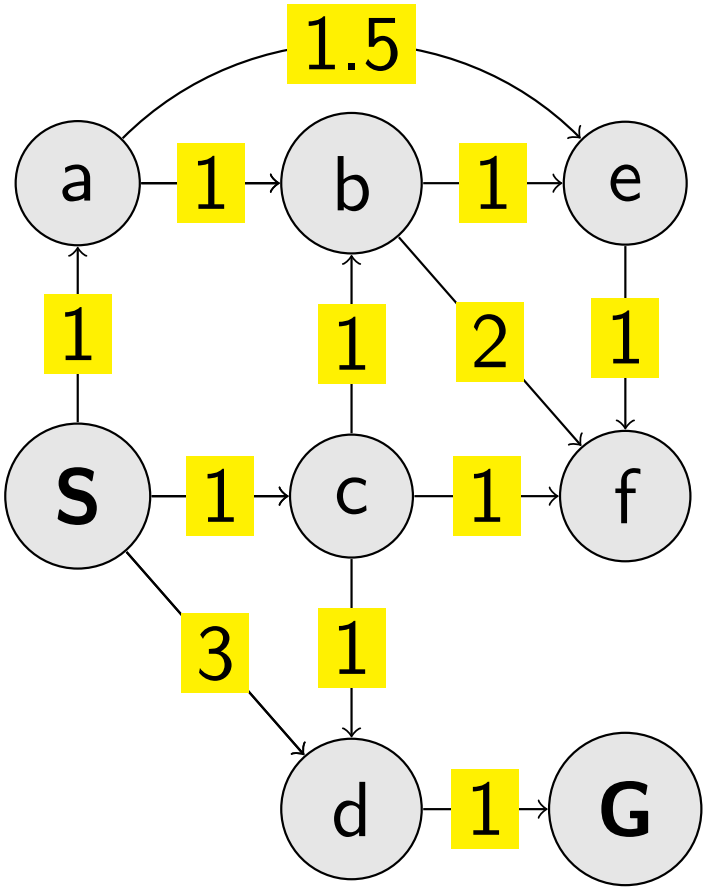


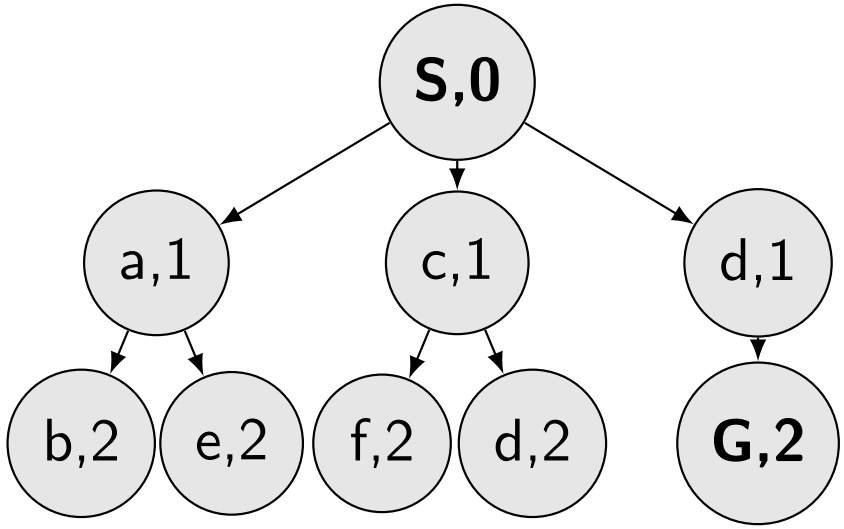
Iterative deepening DFS (ID-DFS)

- 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?

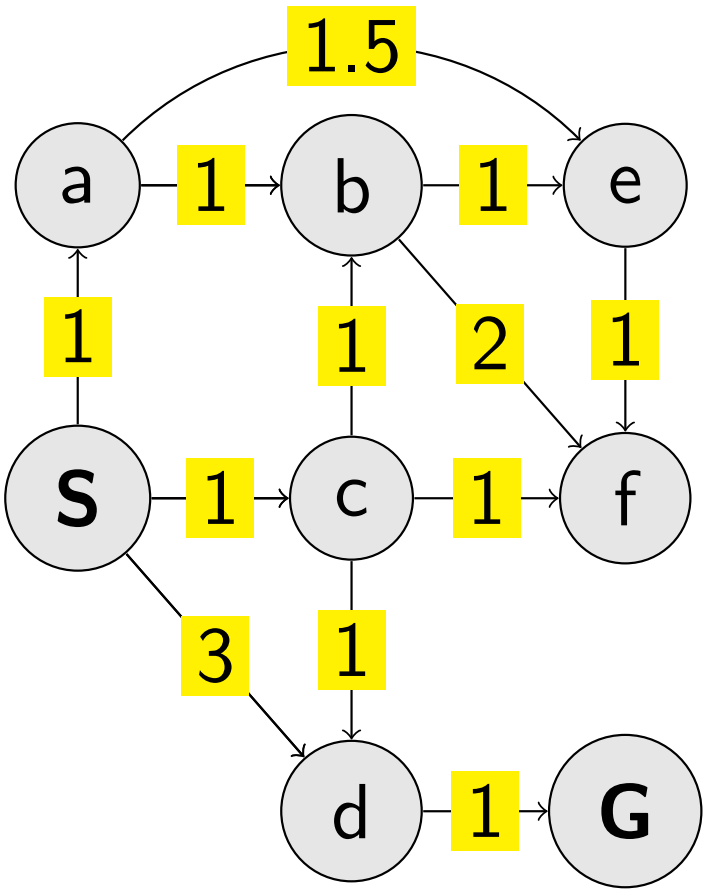
Cost sensitive search

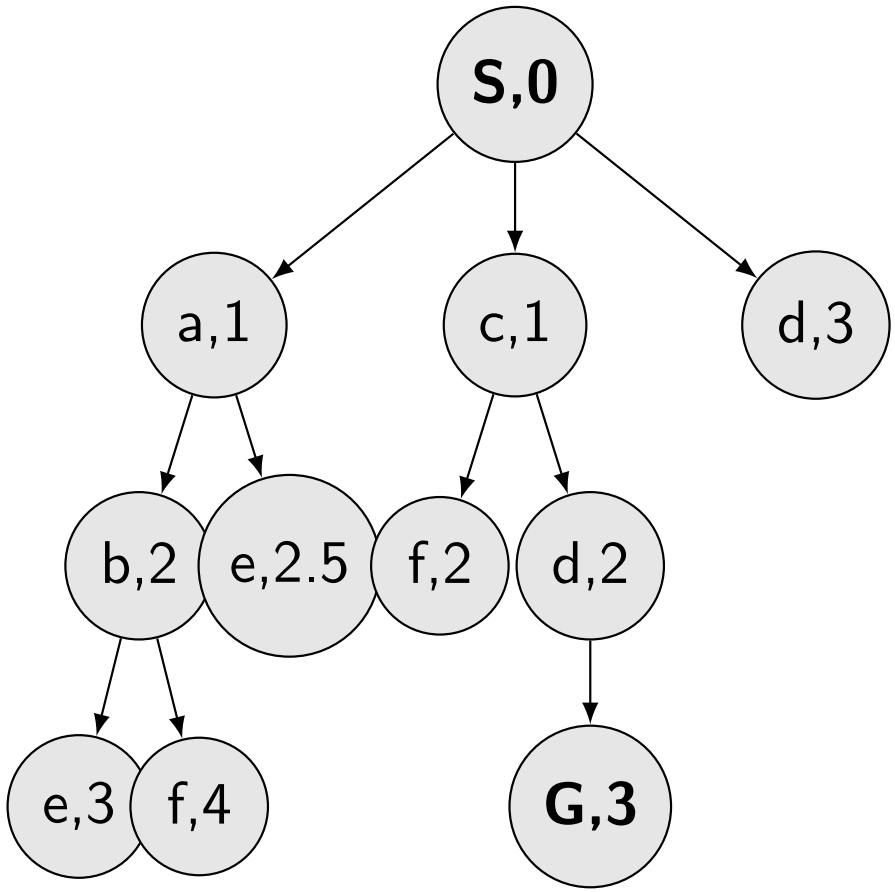




- In BFS, DFS, node ±depth was the node-value.
- How was the depth actually computed?
- How to evaluate nodes with path cost?

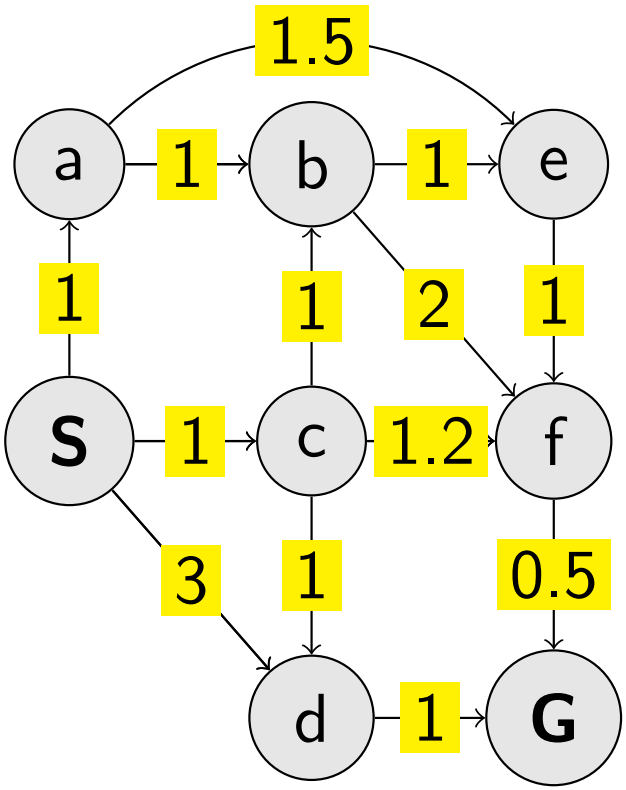
Uniform Cost Search (UCS)

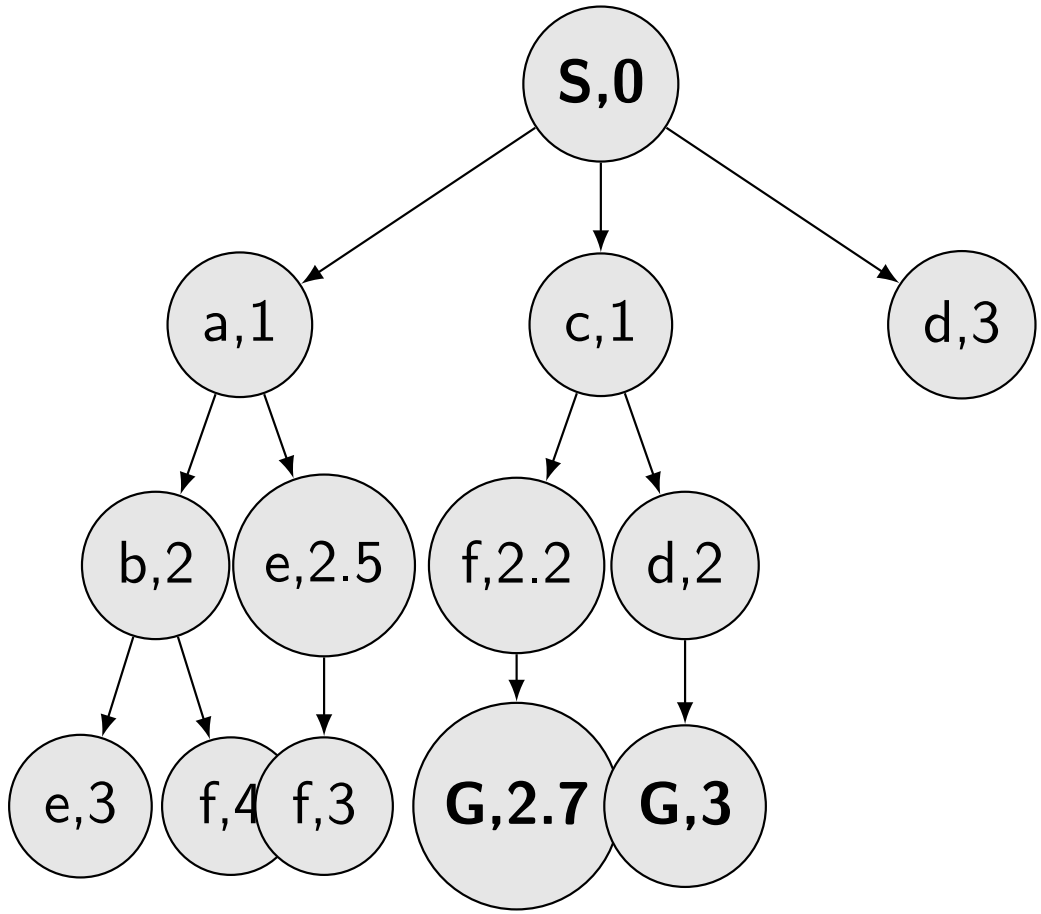




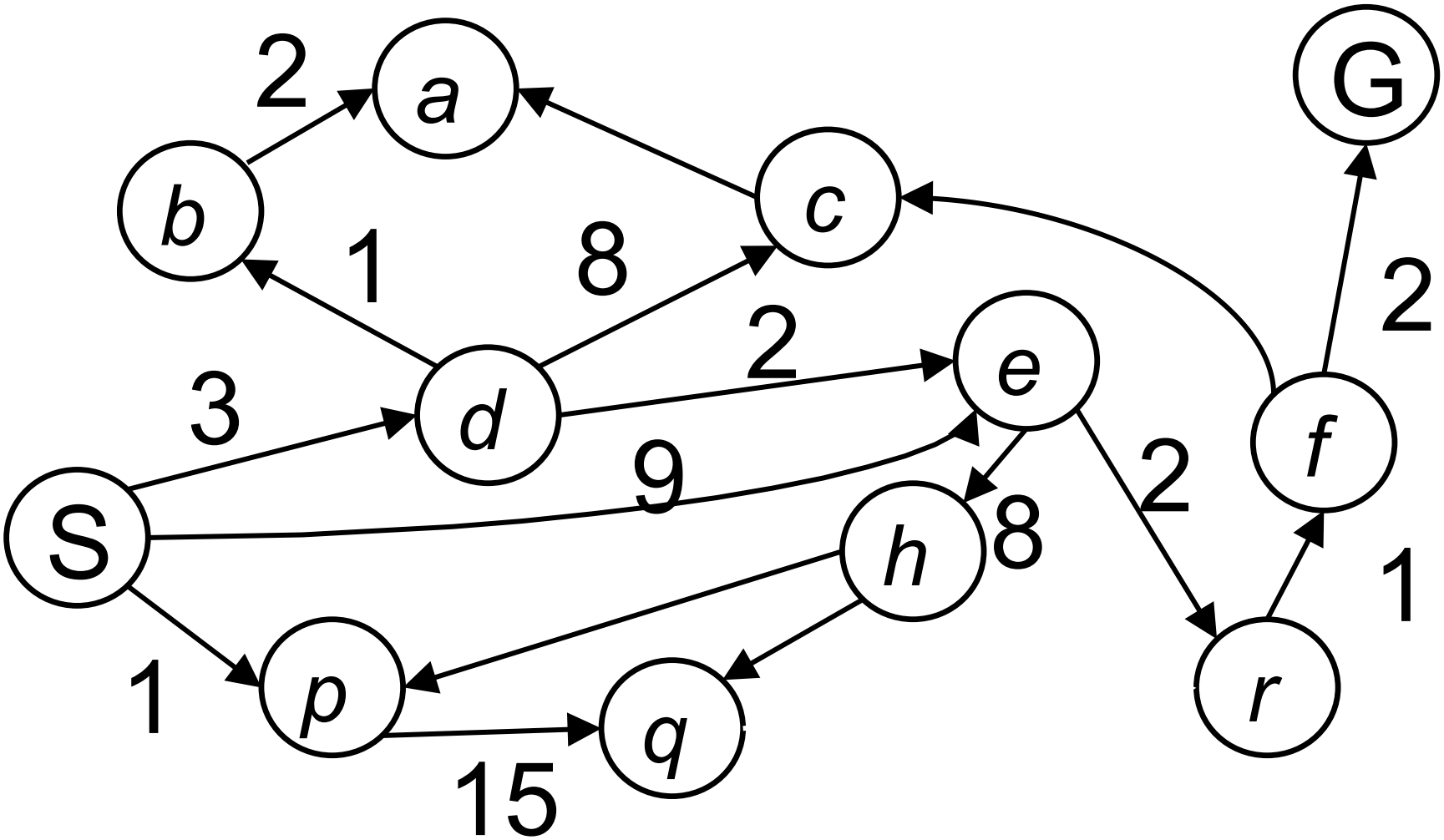
When to check the goal (and stop) the search? When visiting or expanding the node?

When to stop, when visiting or expanding?





Example: Graph with costs



UCS properties

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 ...

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?