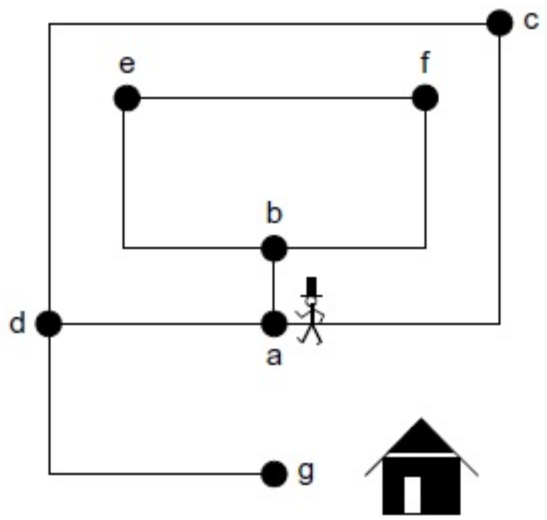
CSCI-564
Constraint Processing and Heuristic Search

Lecture 4 – Uninformed Search

Recap

- A multitude of problem can be formalized as state space problem P
- It conducts to a graph representation
- When the state space is large, we need a solution to explore the graph.



Searching

- Usually finding a solution correspond to finding a shortest path in the graph.
  - Or a minimum cost path.

- We can distinguish two types of graph search.
  - Explicit search
  - Implicit search

What is the difference?

Search

- Explicit graph search:
  - It assumes that the graph structure of the problem is accessible.
  - Meaning that you know perfectly the problem.

- Implicit graph search:
  - Nodes are generated iteratively and expanded without access to the unexplored part of the graph.

For small state space problem, it is possible to represent the entire graph.

Uninformed search

- In an implicit graph search:
  - The graph representation is not available at the beginning
  - In each iteration, a node is expanded by generating all adjacent nodes
    - If you apply all the actions allowed!
  - Nodes explored remains known.
  - In path from one node to another, each node need to be explored once.

- With uninformed search algorithm you don't have any estimation on the actions you perform.

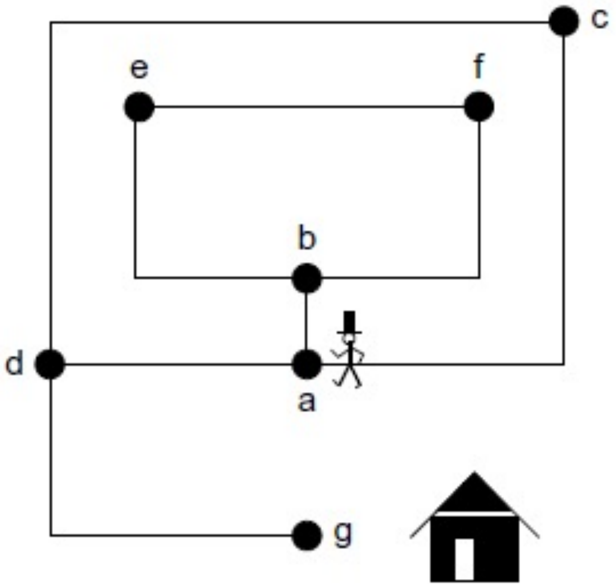
Uninformed Search

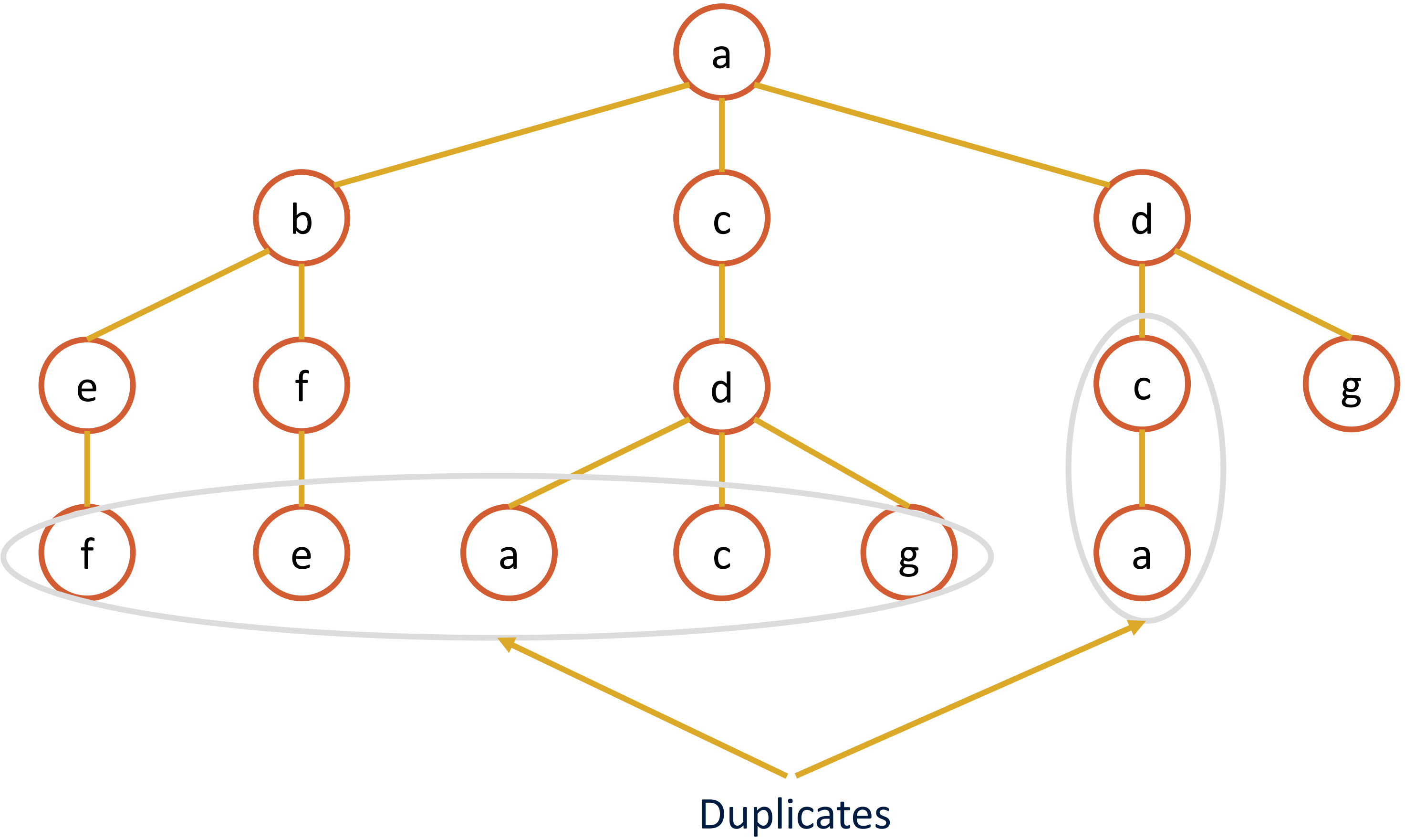
- The set of reached nodes can be divided into two sets:
  - The set of expanded nodes (or closed nodes).
  - The set of generated nodes (or open nodes), the set of nodes that are not expanded yet.

- Definition (Search Tree):
  - The search tree of the underlying problem is the set of all explicitly generated paths from the root node to a generated node.

The search tree is not the graph problem representation.

Example





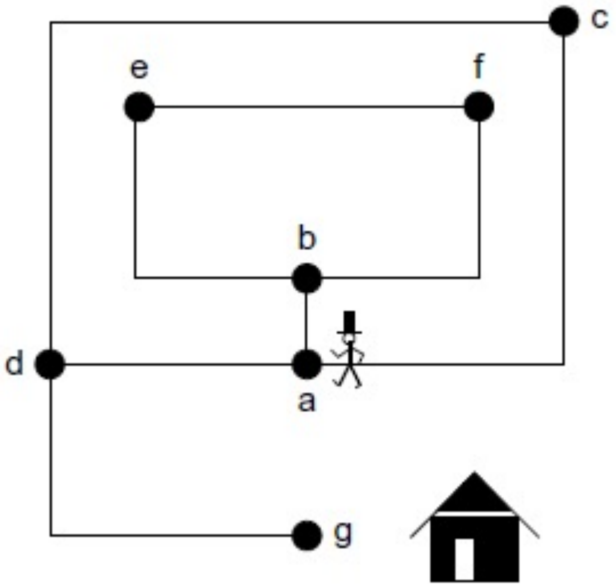
Uninformed search

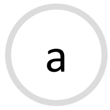
- In a tree-structured problem, a node can only be reached on a single path.
- For finite acyclic graphs, the search tree can be exponentially larger than the original problem.
- For cyclic graphs, the search tree can be infinite.

Depth-First Search

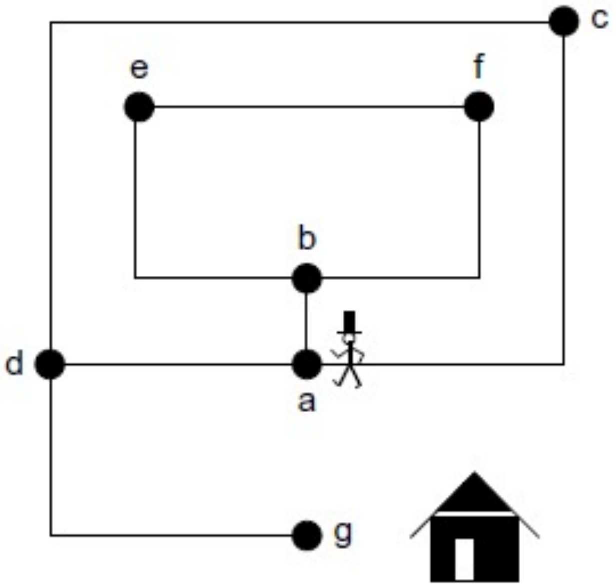
- Each step greedily generates a successor of the last visited node
- Unless it has no successor, in which case it backtracks to the parent and explores another node not explored.

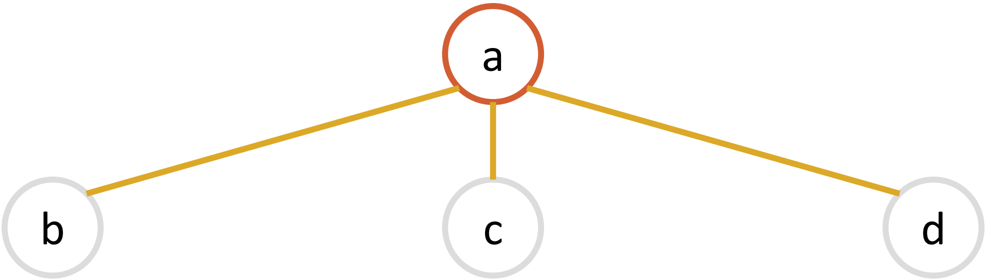
Depth-First Search



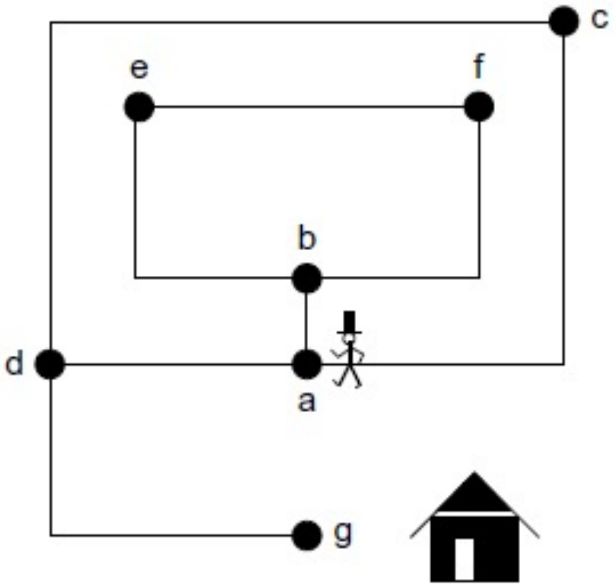


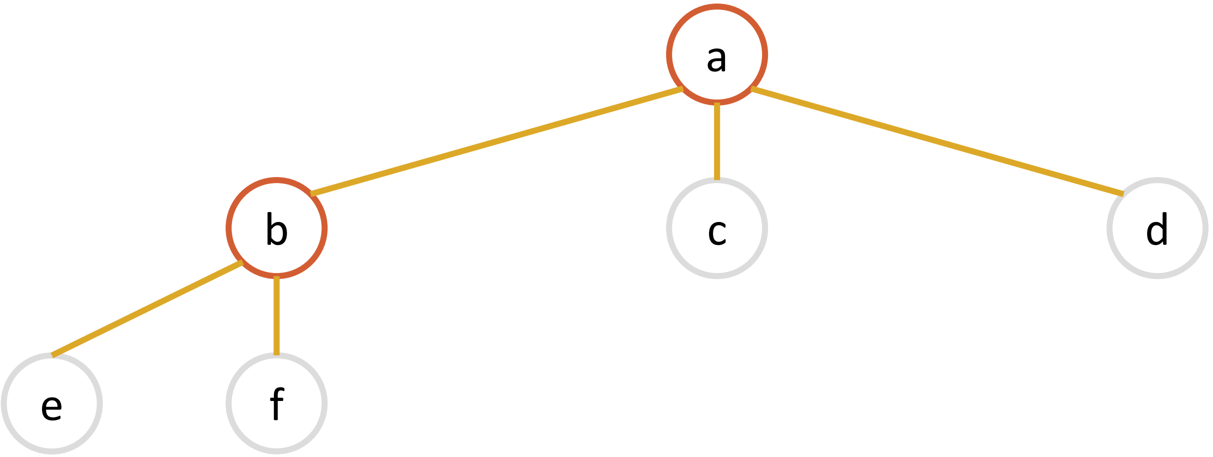
Depth-First Search





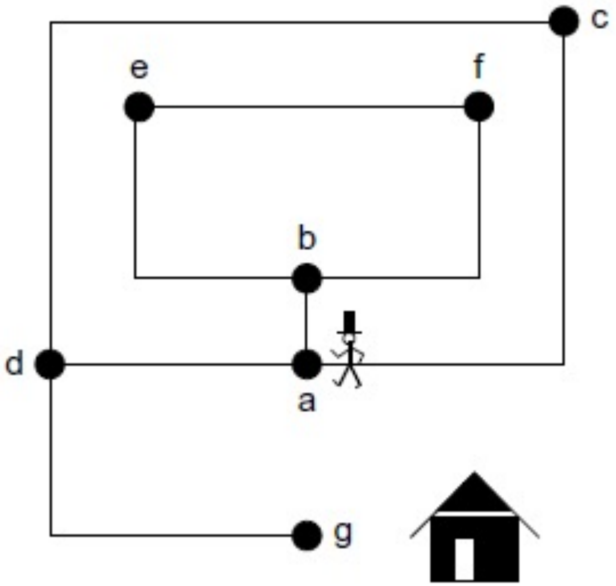
Depth-First Search

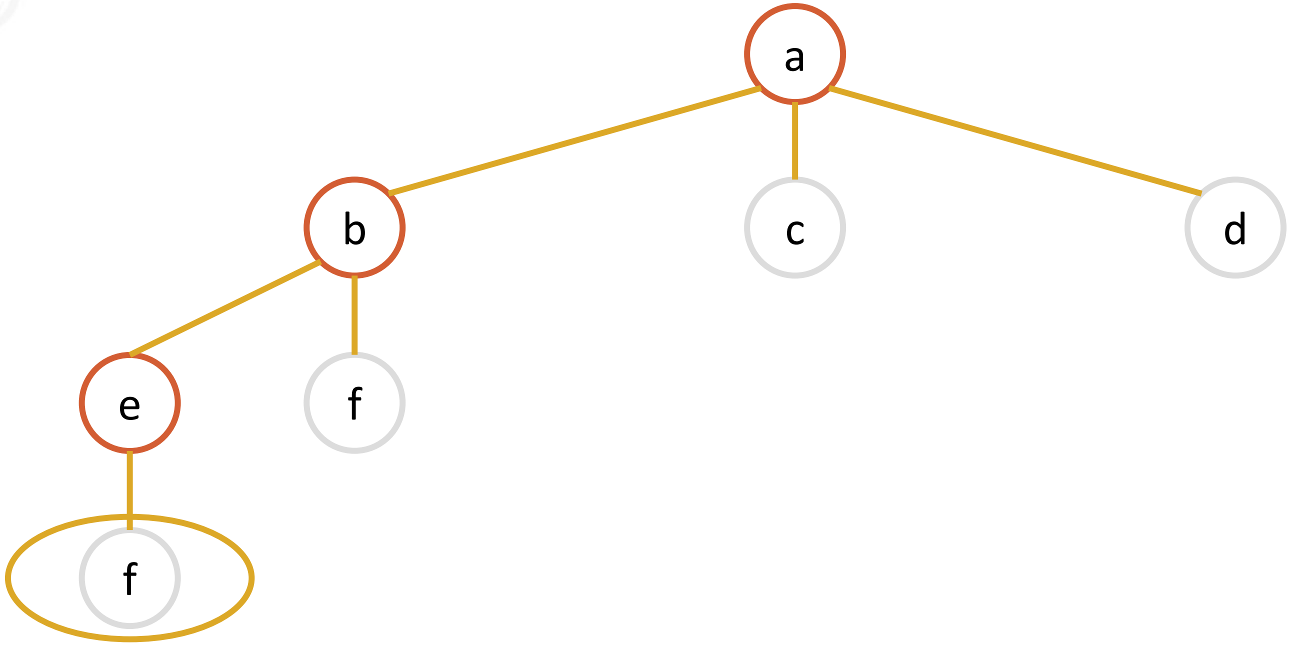




StFX

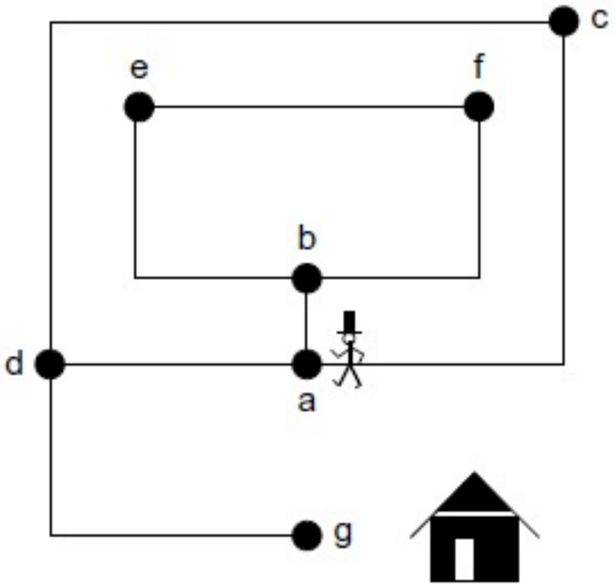
Depth-First Search

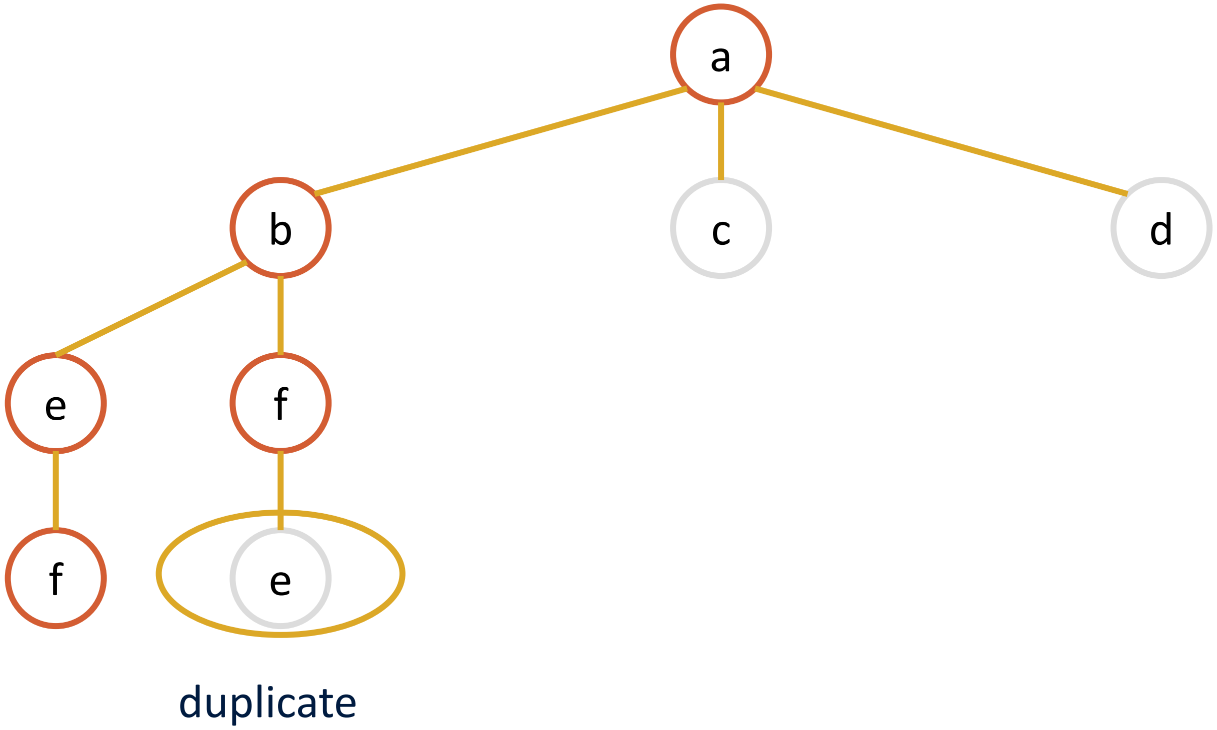




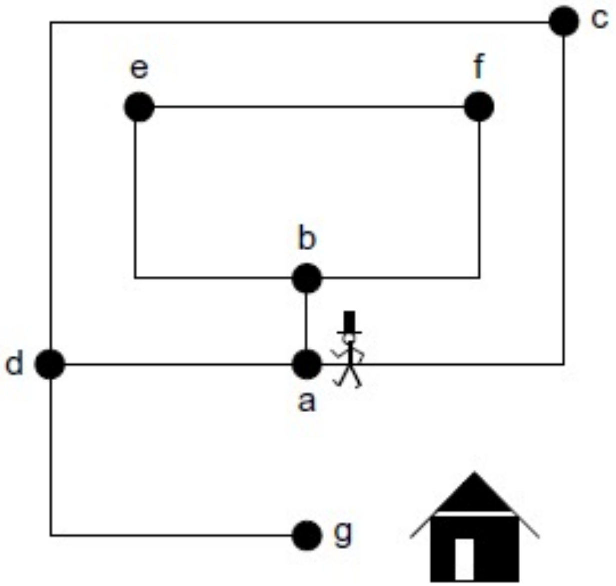
duplicate

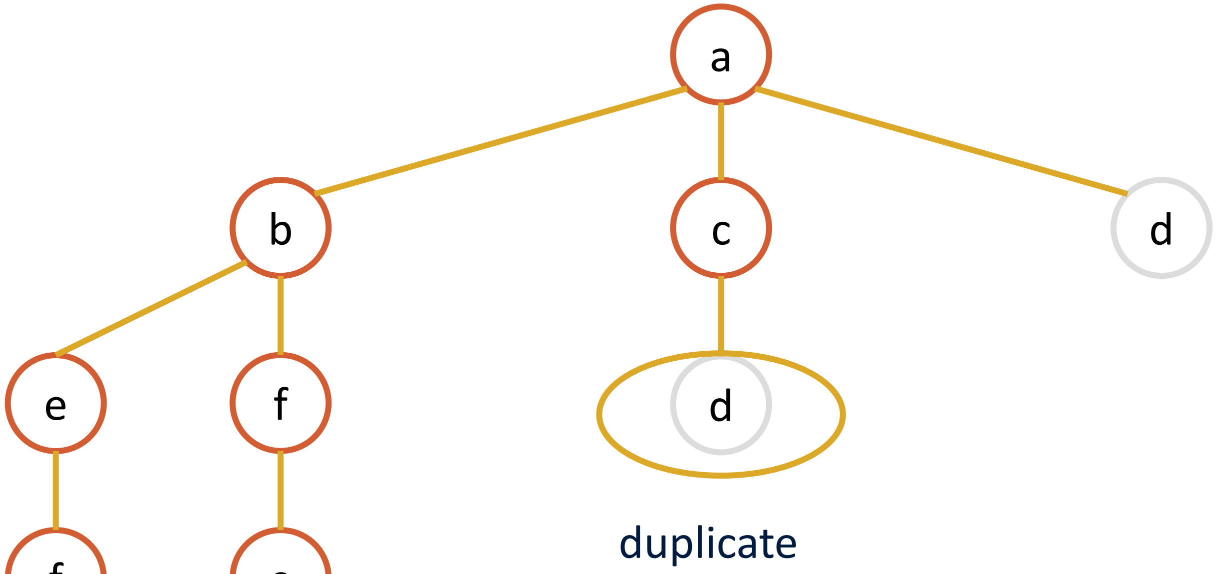
Depth-First Search



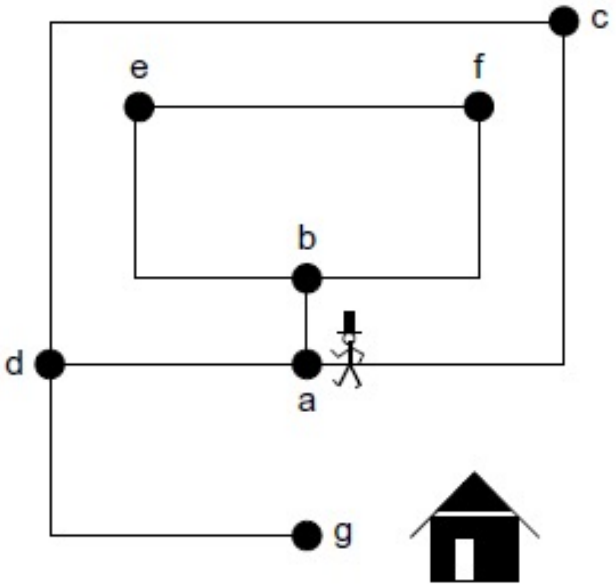


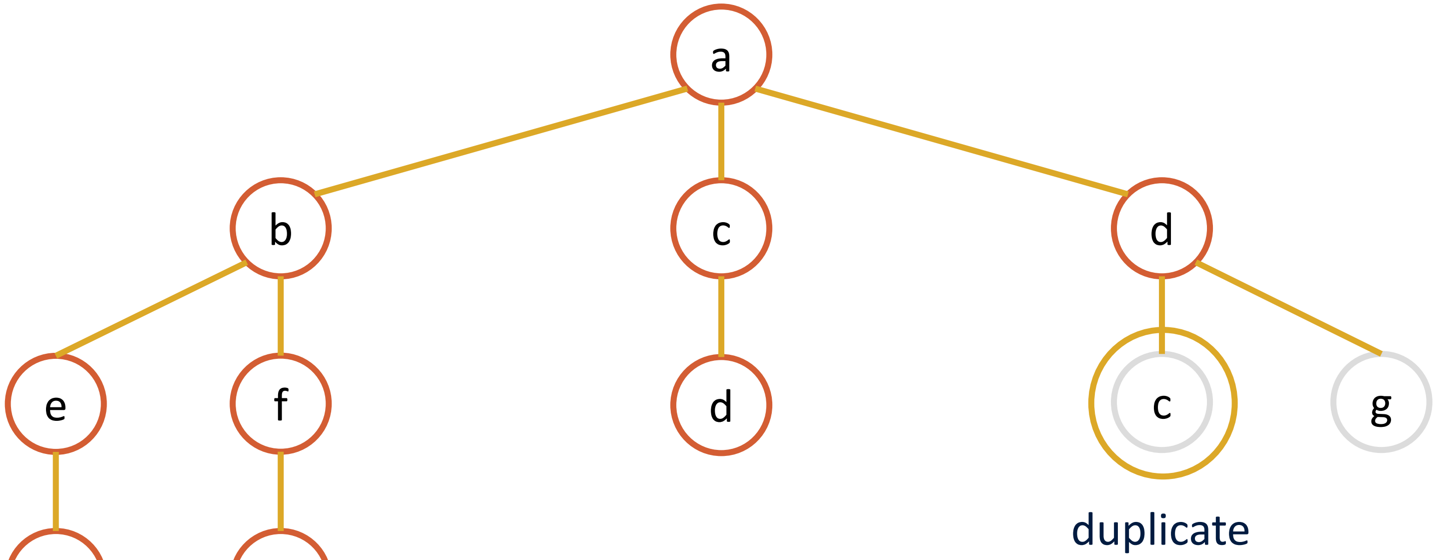
Depth-First Search



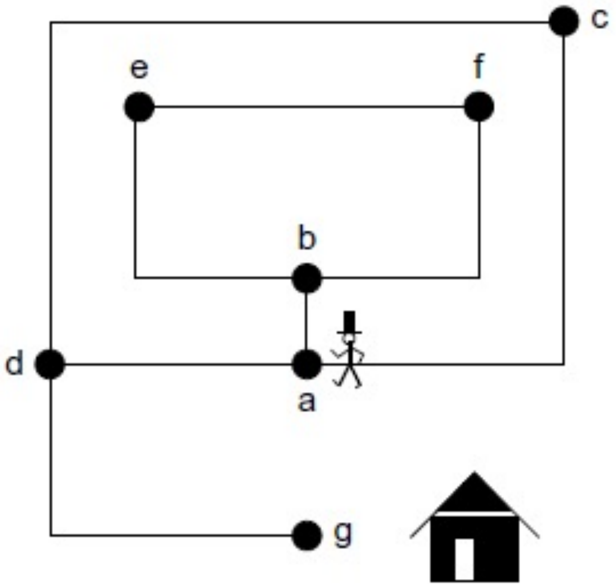


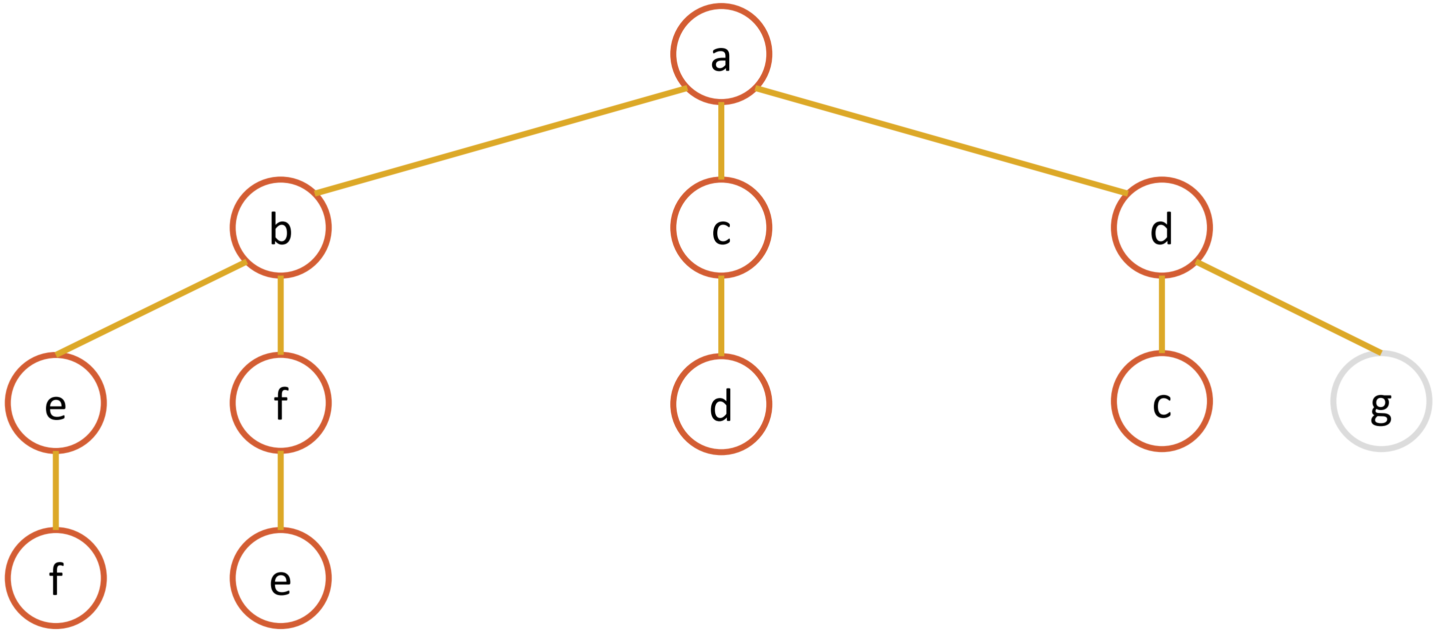
Depth-First Search



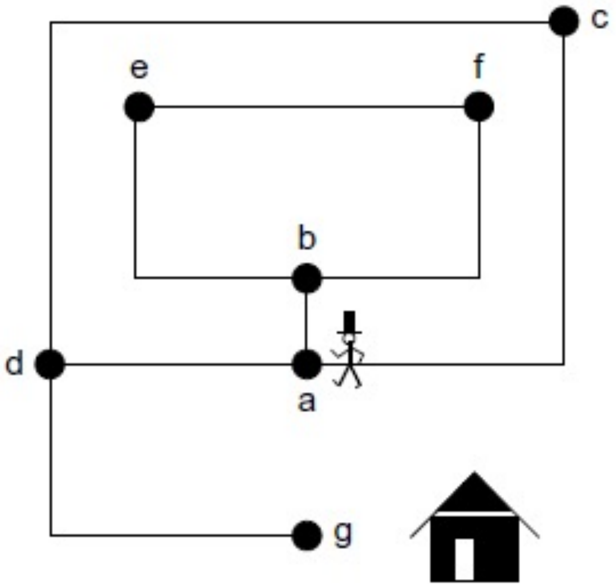


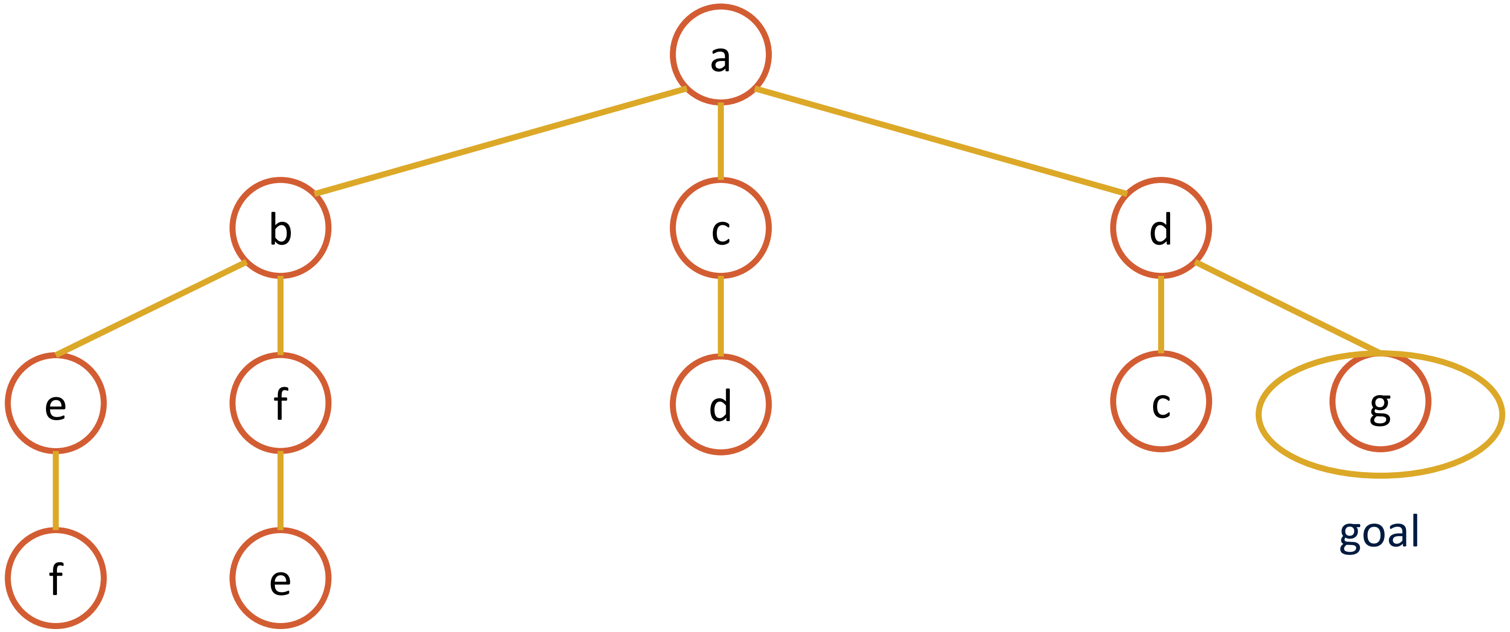
Depth-First Search





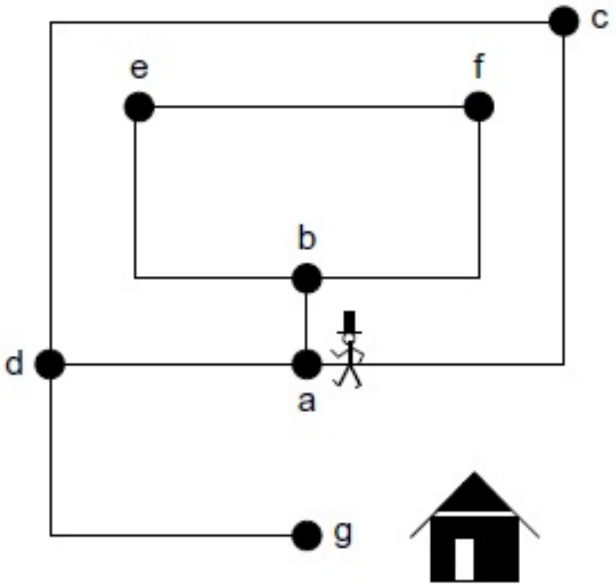
Depth-First Search

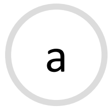




Breadth-First Search

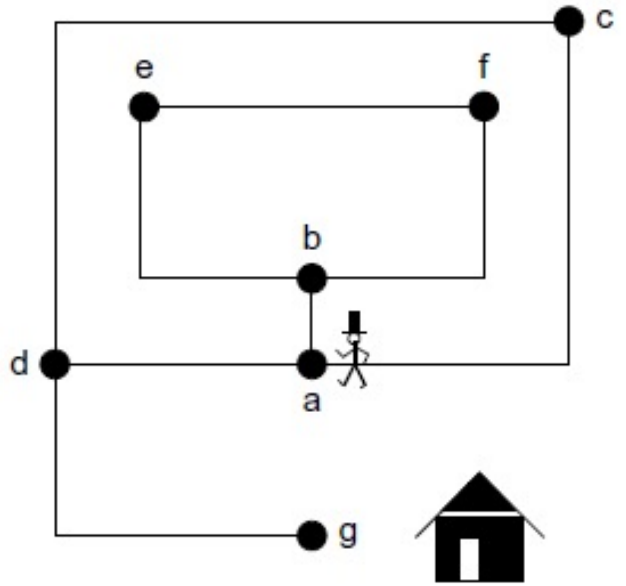
- The tree is searched layer by layer.

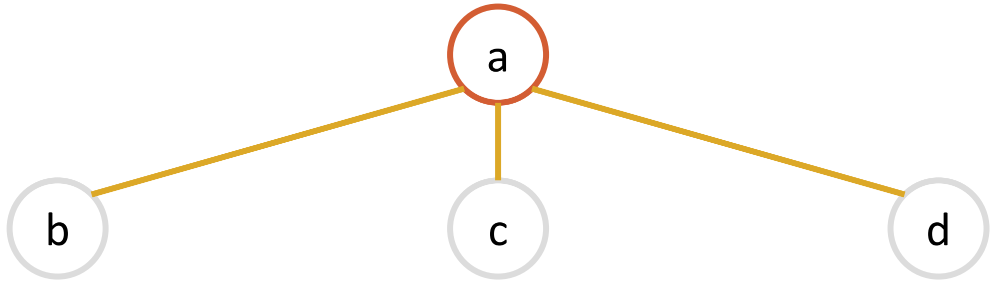




Breadth-First Search

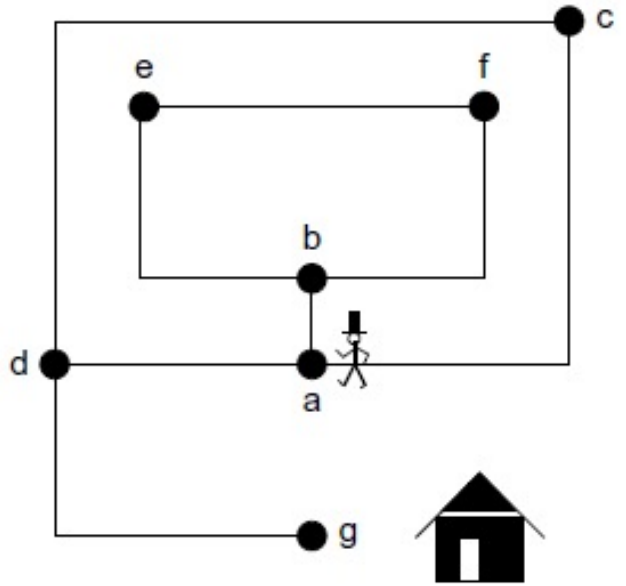
- The tree is searched layer by layer.

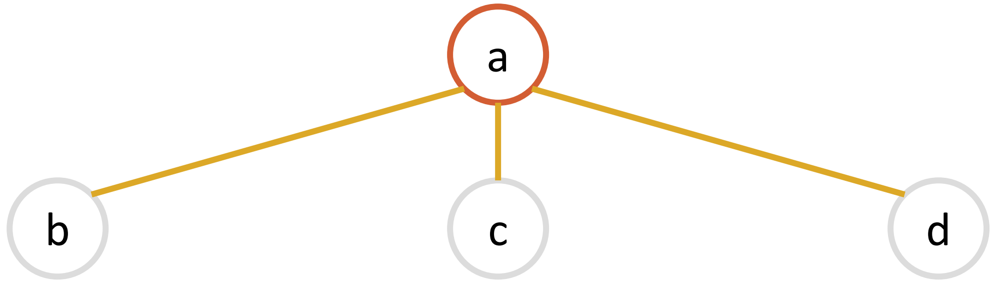




Breadth-First Search

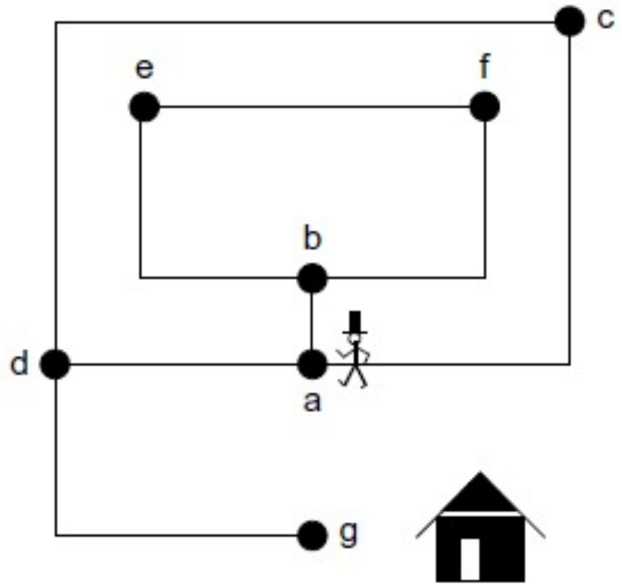
- The tree is searched layer by layer.

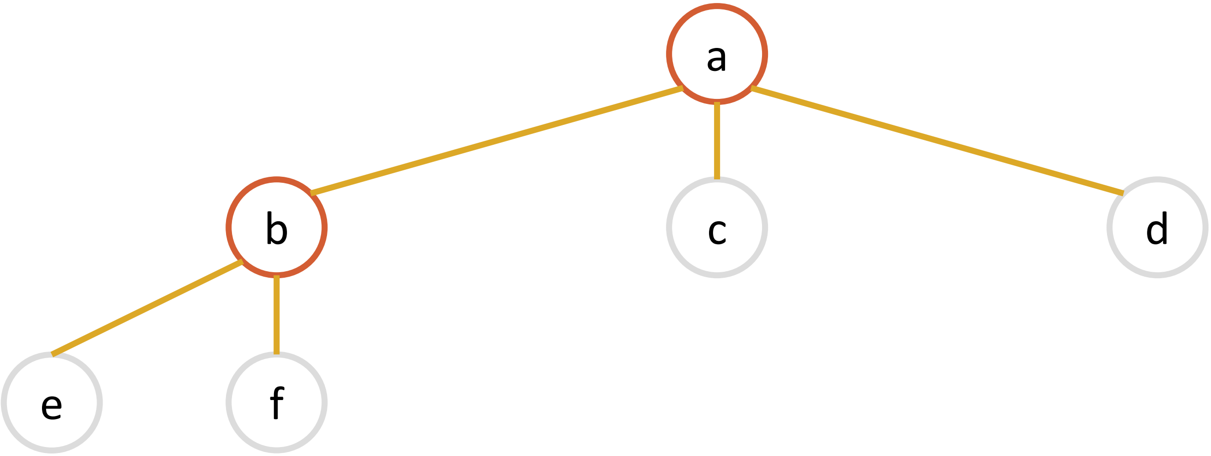




Breadth-First Search

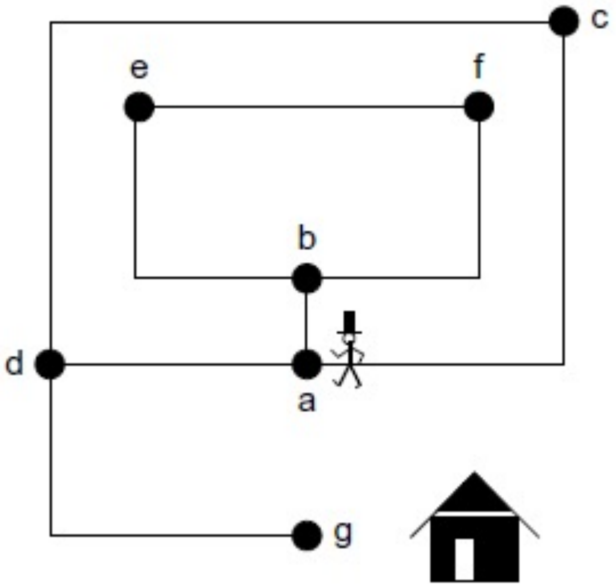
- The tree is searched layer by layer.

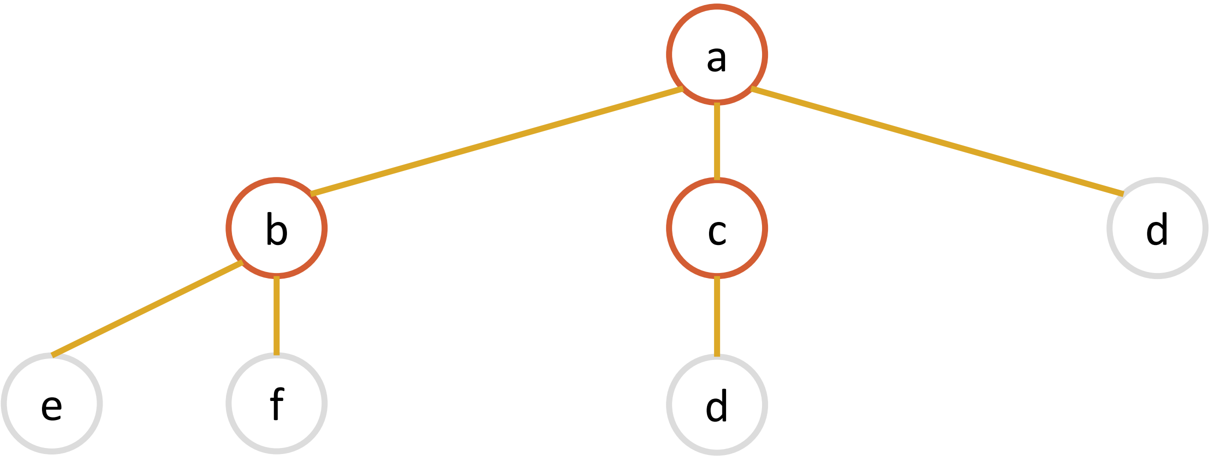




Breadth-First Search

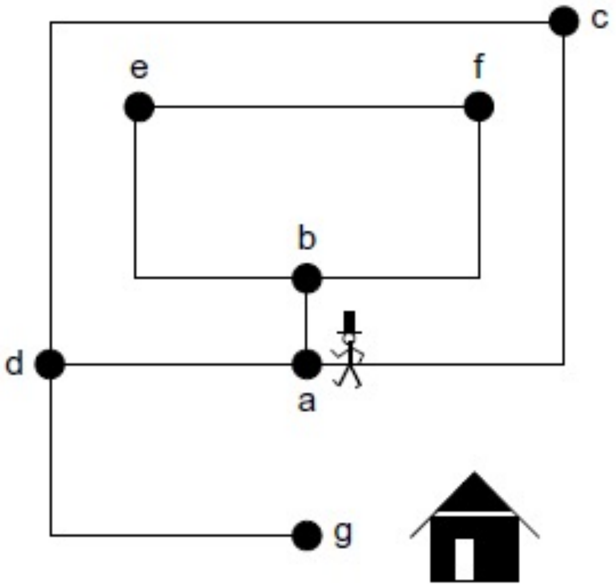
- The tree is searched layer by layer.

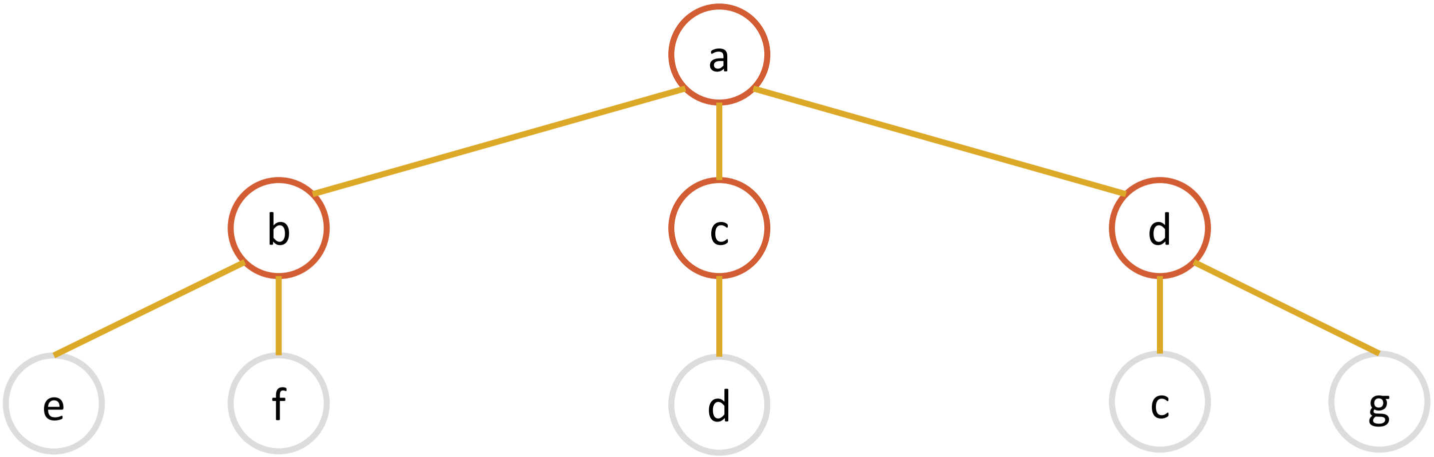




Breadth-First Search

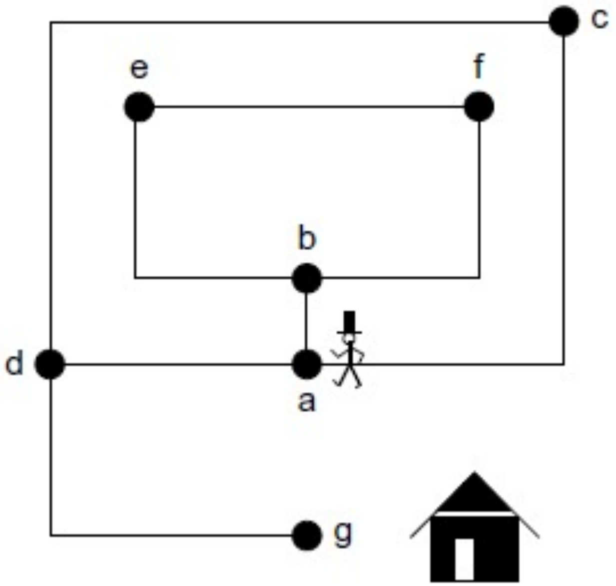
- The tree is searched layer by layer.

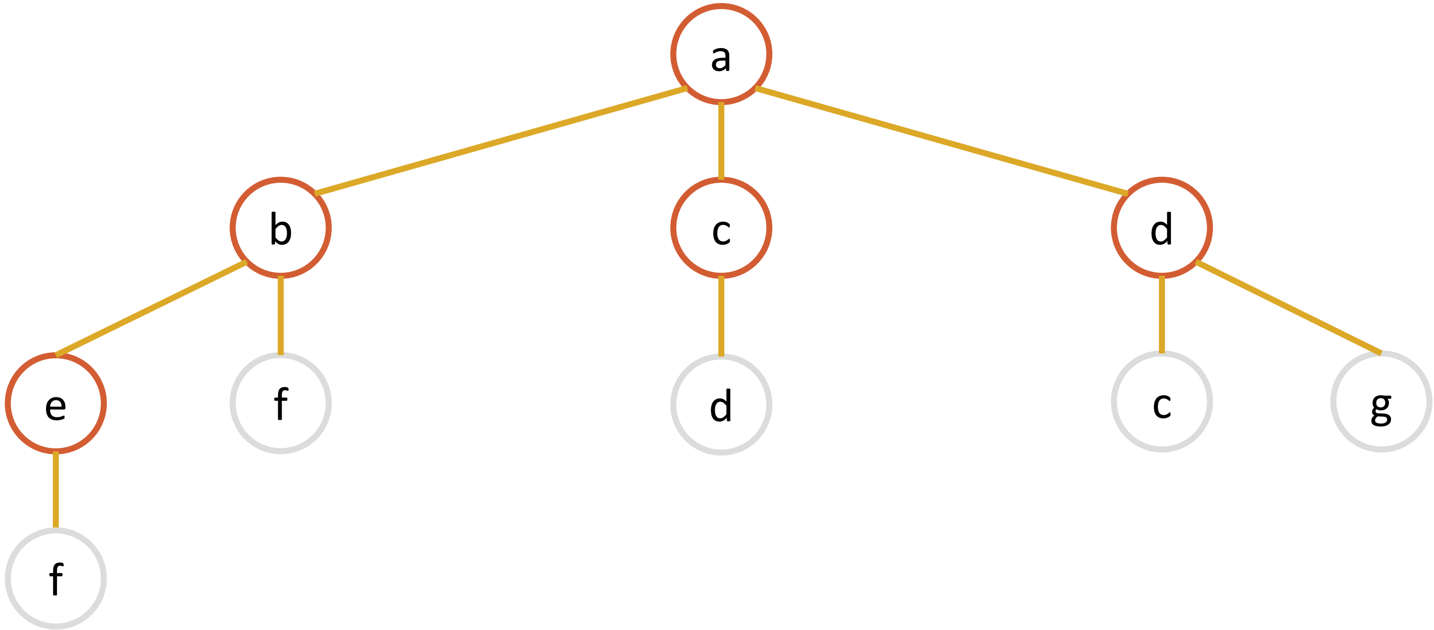




Breadth-First Search

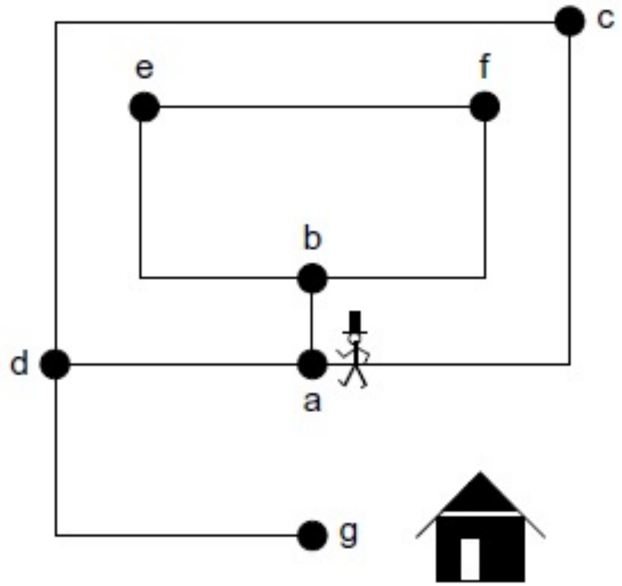
- The tree is searched layer by layer.

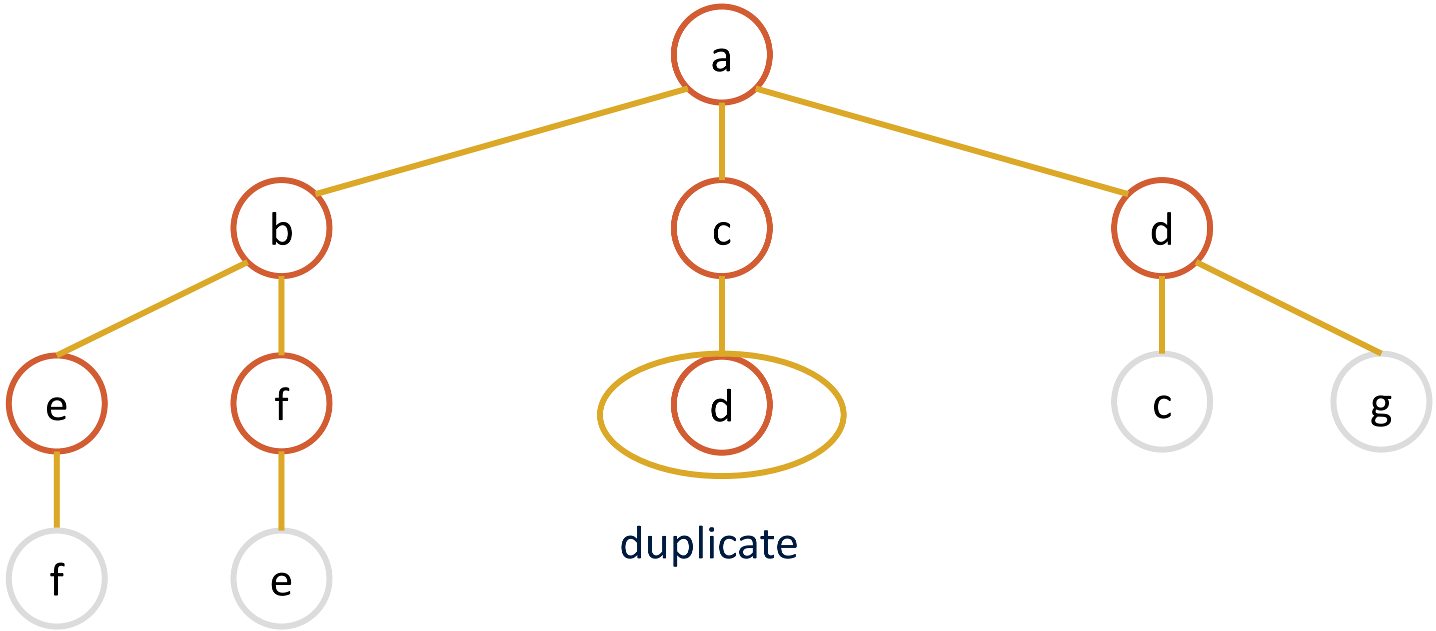




Breadth-First Search

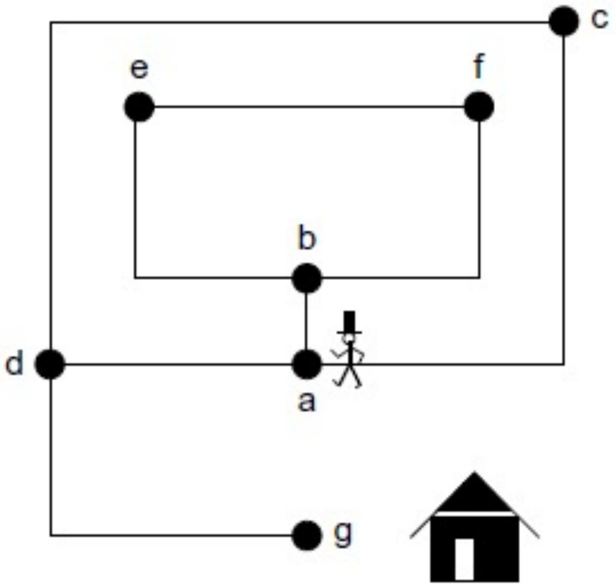
- The tree is searched layer by layer.

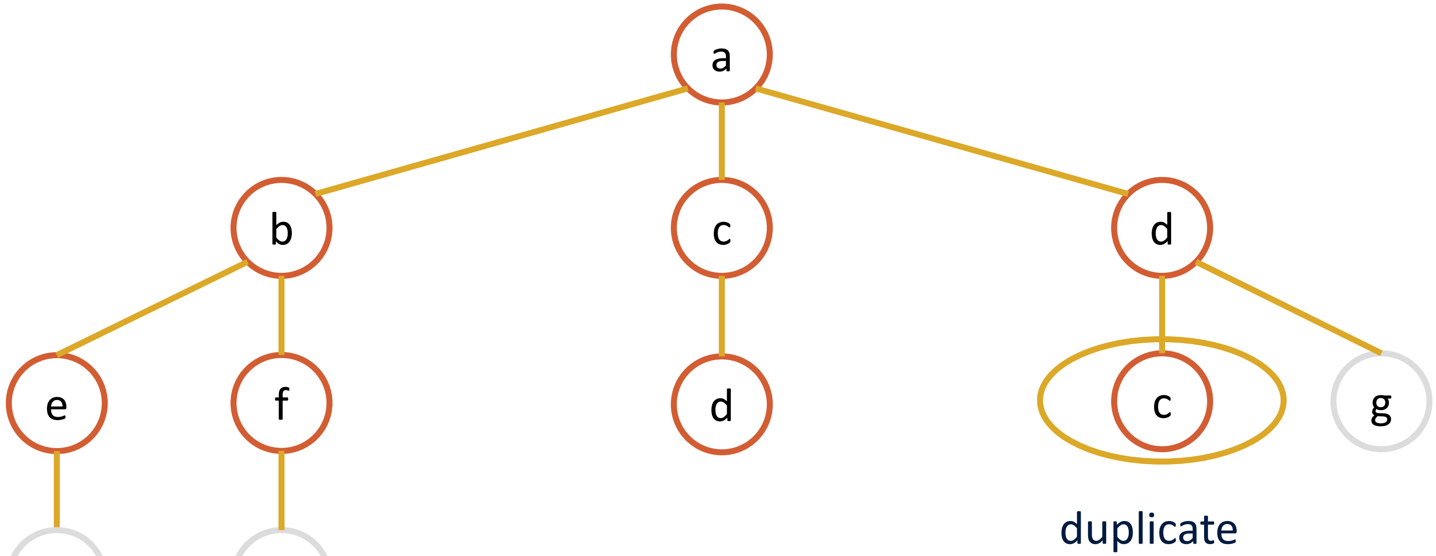




Breadth-First Search

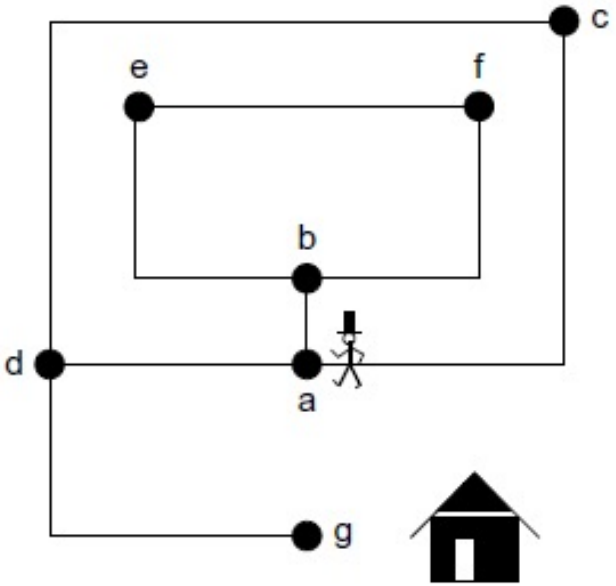
- The tree is searched layer by layer.

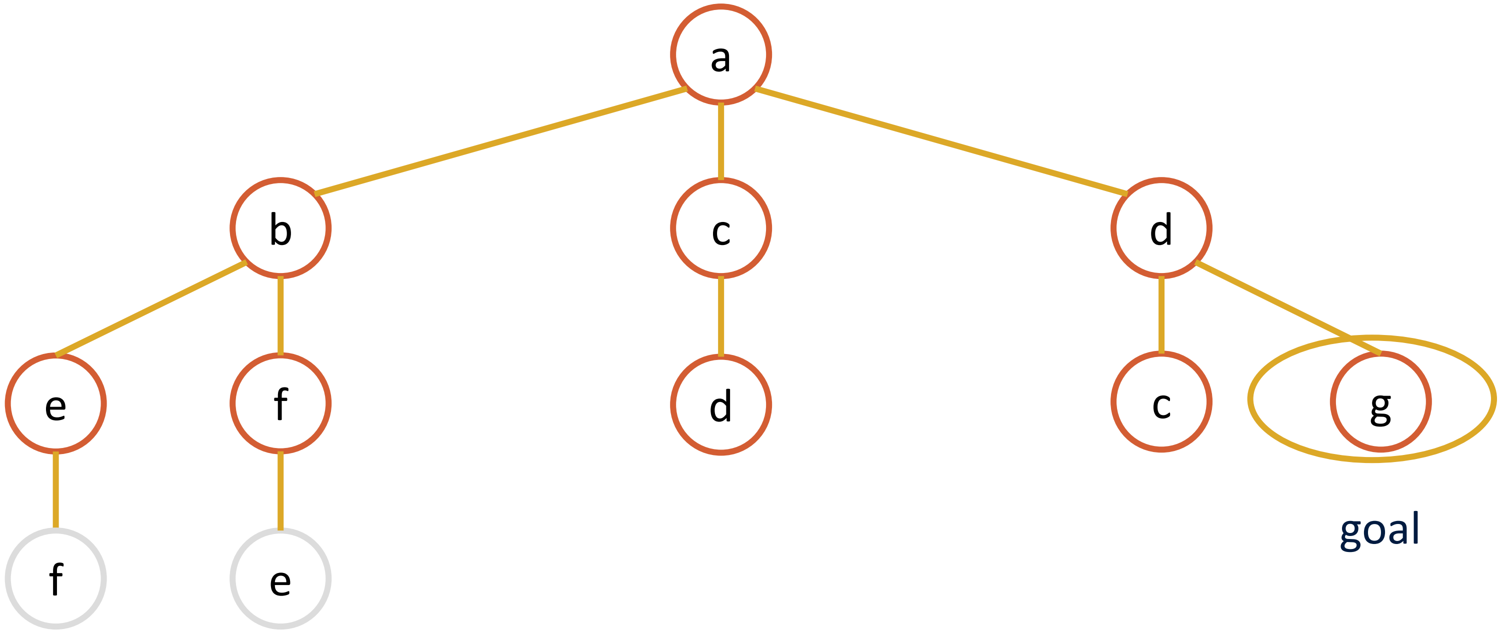




Breadth-First Search

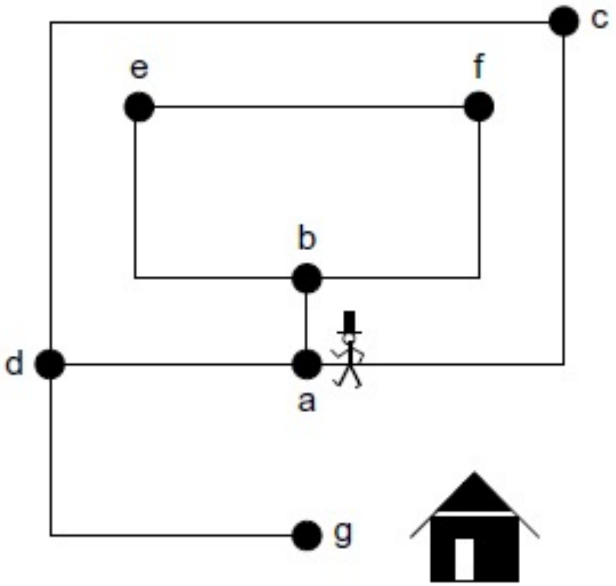
- The tree is searched layer by layer.

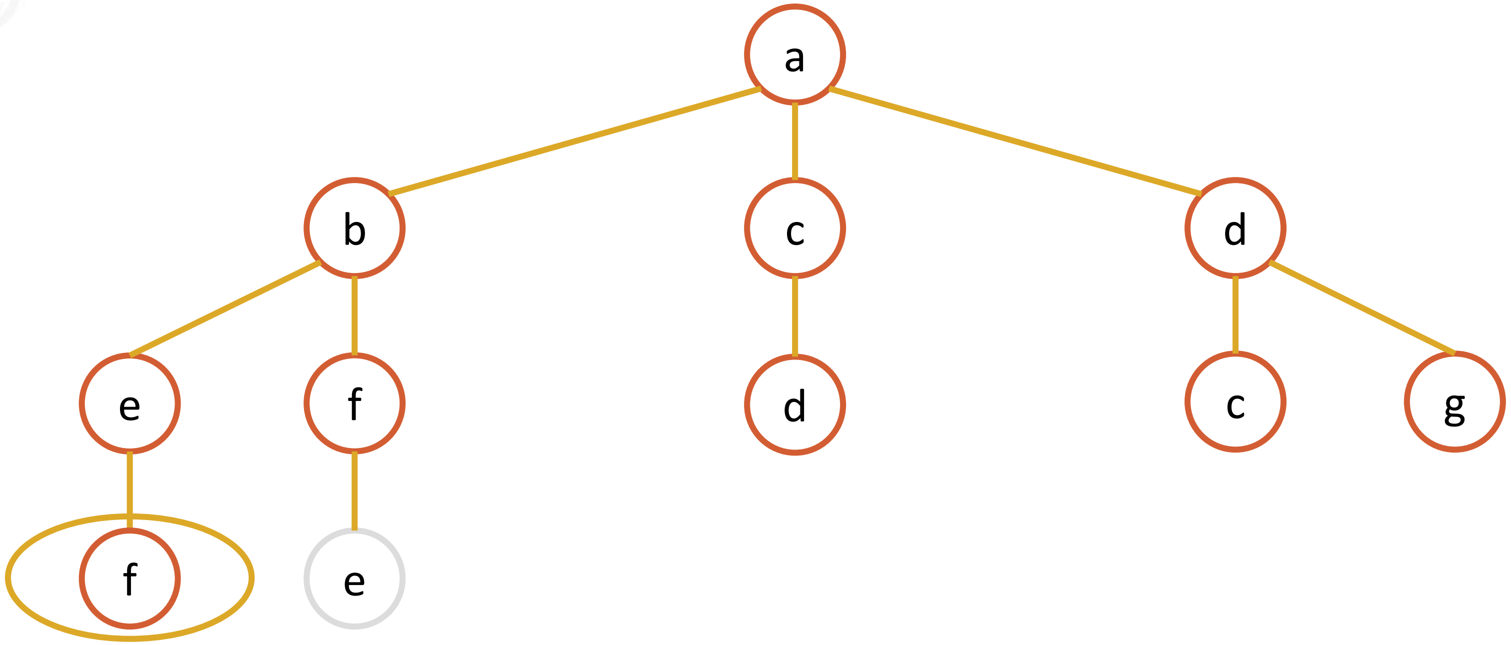




Breadth-First Search

- The tree is searched layer by layer.

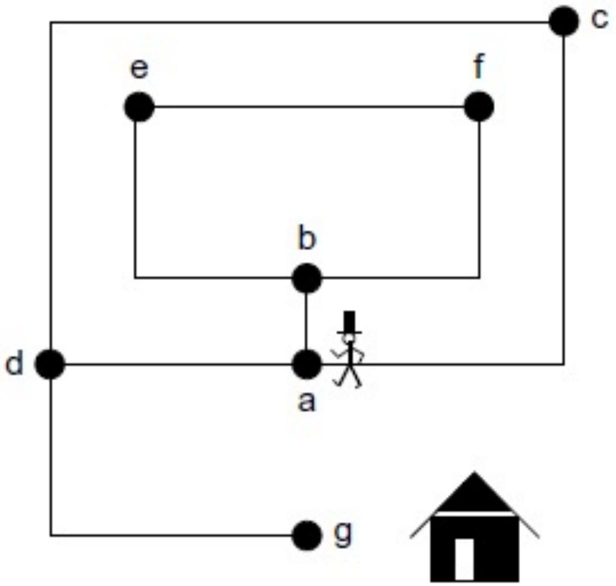


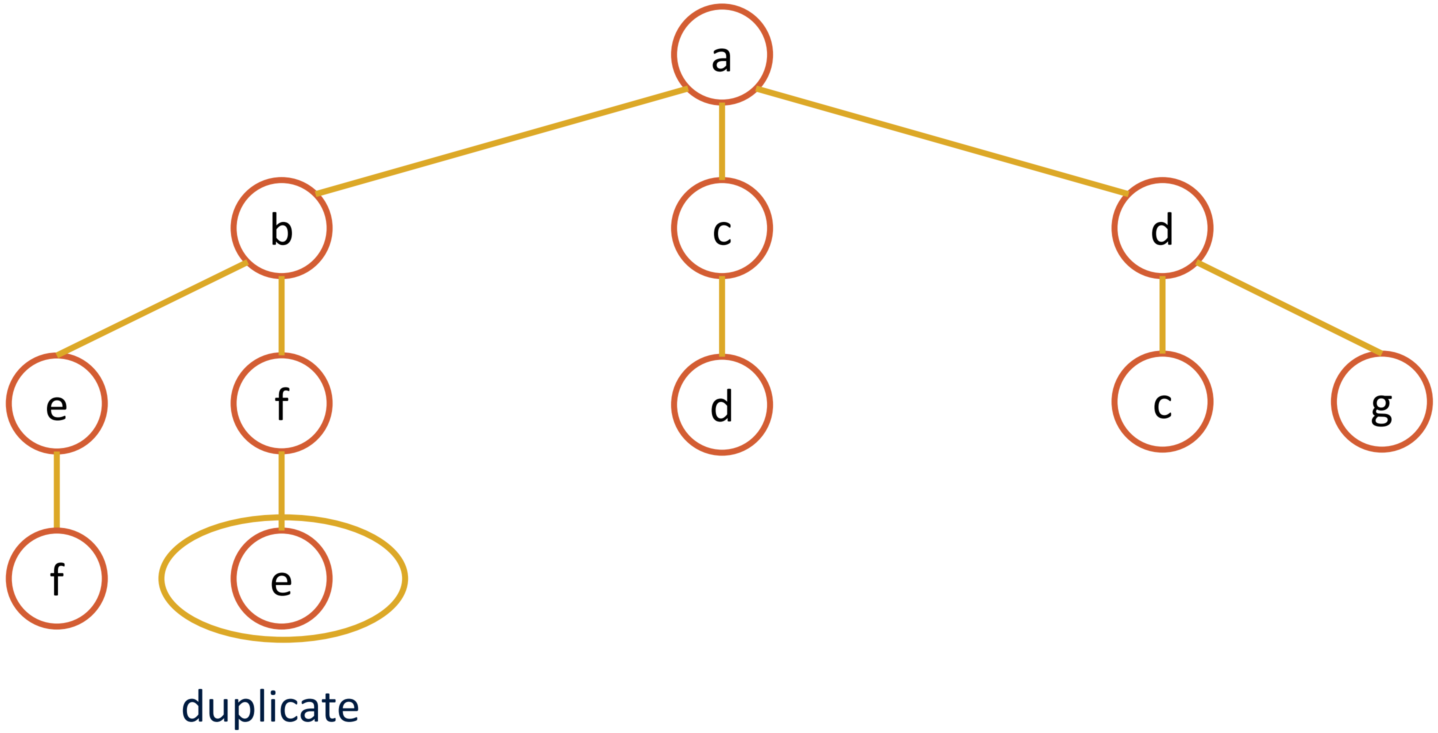


duplicate

Breadth-First Search

- The tree is searched layer by layer.



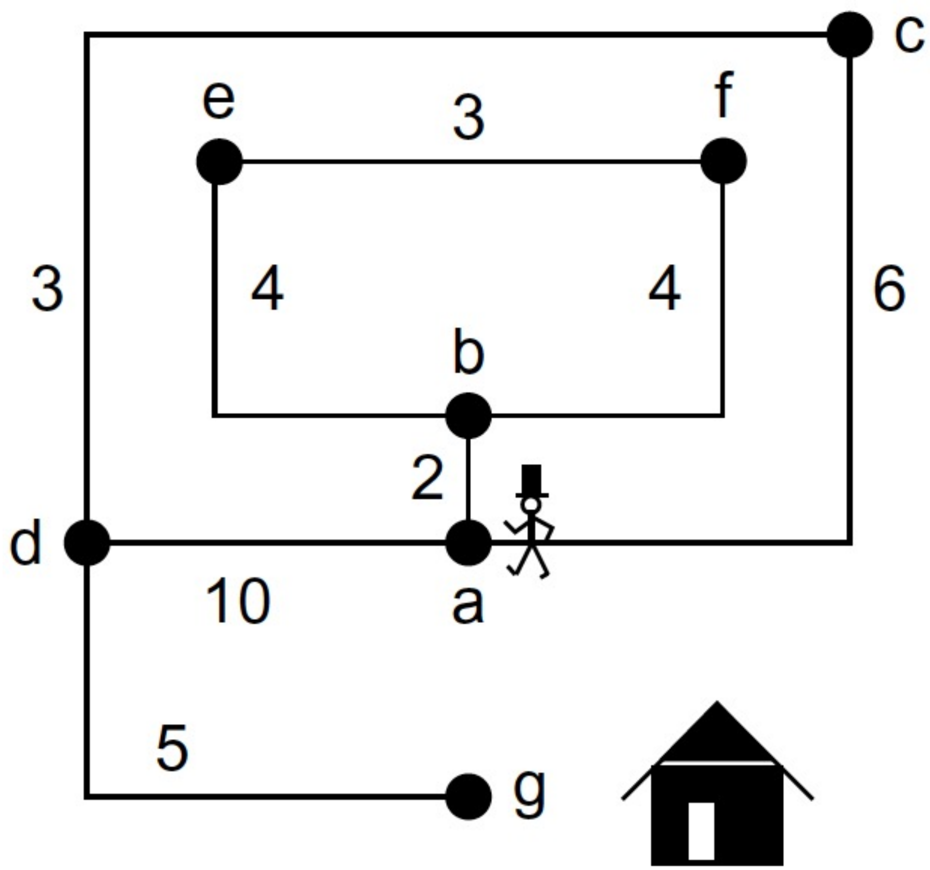


BFS is optimal

StFX

Weighted graphs

- BFS loose its optimality when the graph is not uniformly weighted.



Dijsktra algorithm

- Dijkstra propose a greedy search algorithm based on the principle of optimality.
- Definition:
  - An optimal path has the property that whatever the initial conditions and control variables (choices) over some initial period, the control (or decision variables) chosen over the remaining period must be optimal for the remaining problem, with the node resulting from the early decisions taken to be the initial condition.

What does it mean?

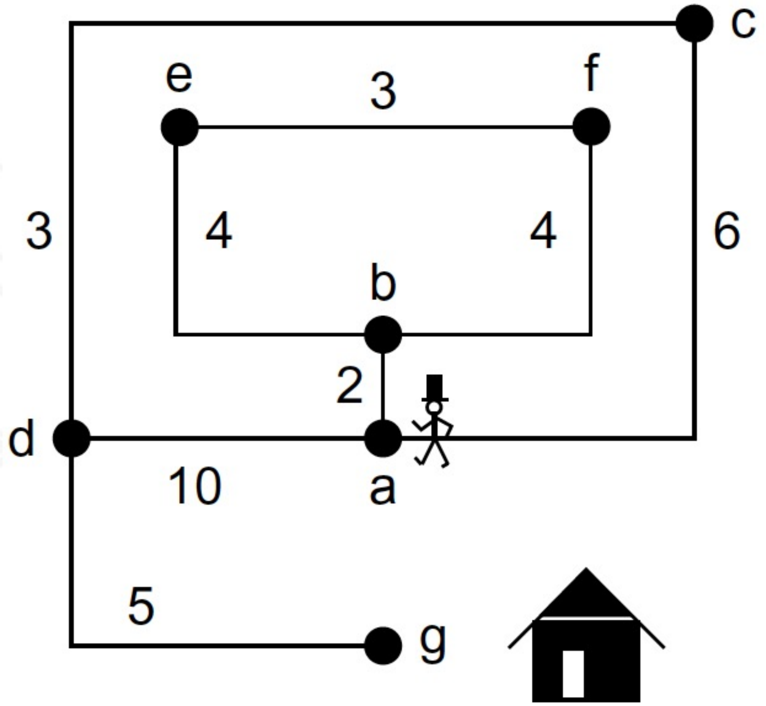
Dijsktra algorithm

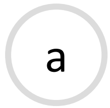
- Dijkstra propose a greedy search algorithm based on the principle of optimality.
- From there, we have the following formula:

$$\delta(s, v) = \min_{v \in Succ(u)} \{\delta(s, u) + w(u, v)\}$$

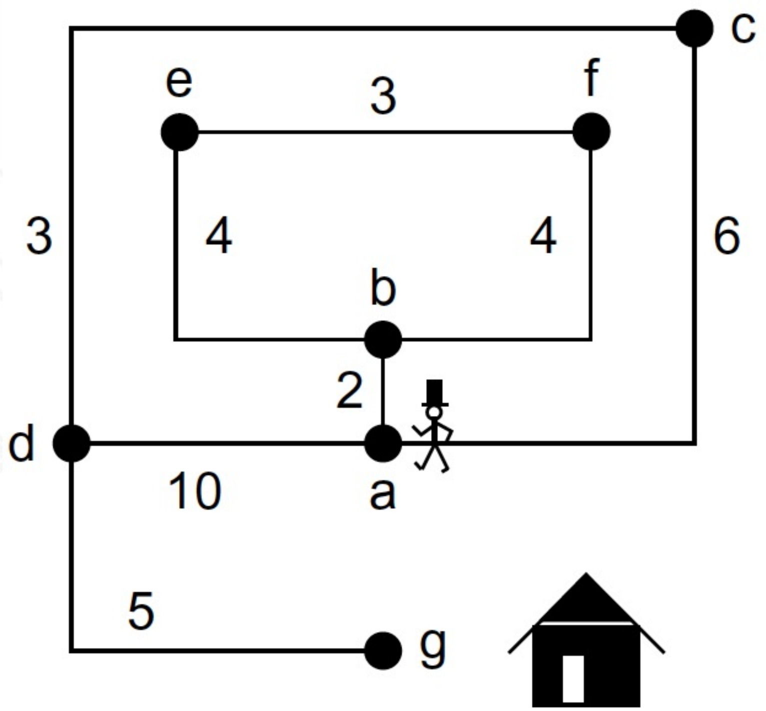
- The minimum distance from s to v is equal to the minimum of the sum of the distance from s to a predecessor u of v, plus the edge weight between u and v.
    - Implies that any subpath of an optimal path is itself optimal

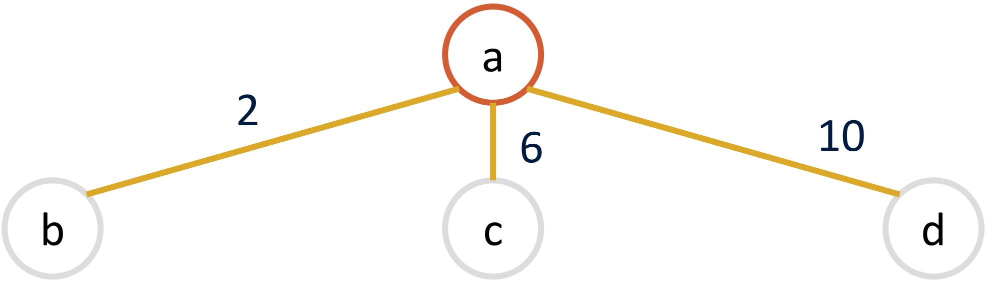
Dijsktra algorithm



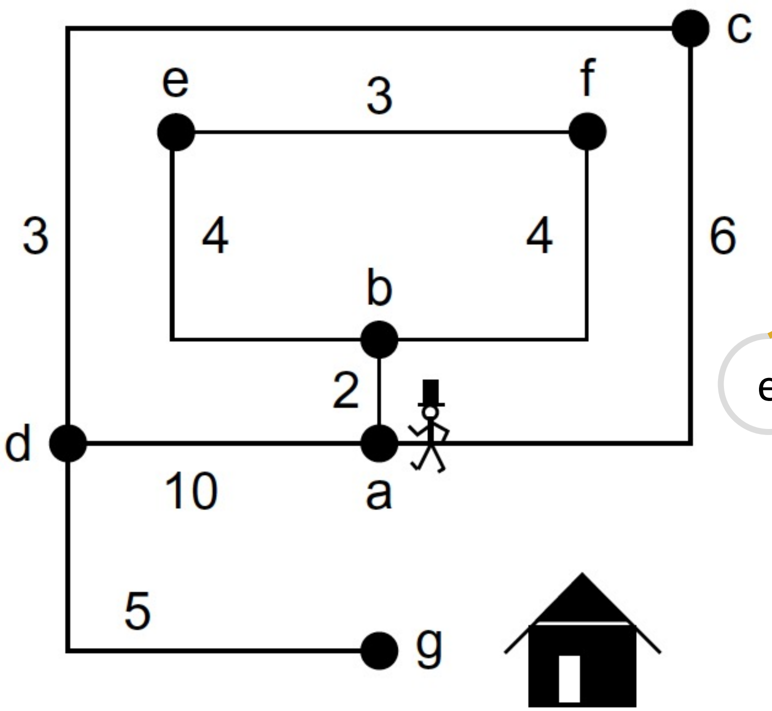


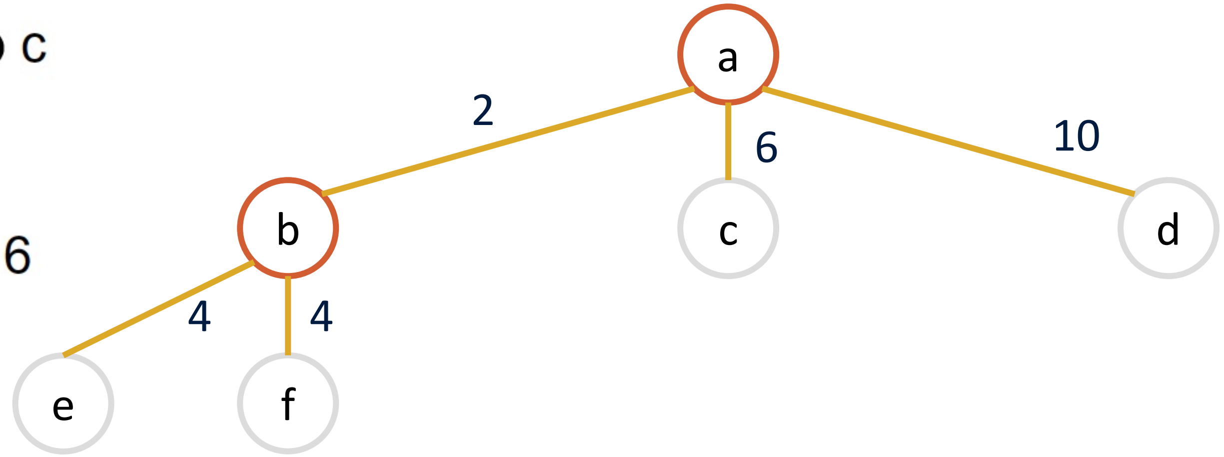
Dijsktra algorithm





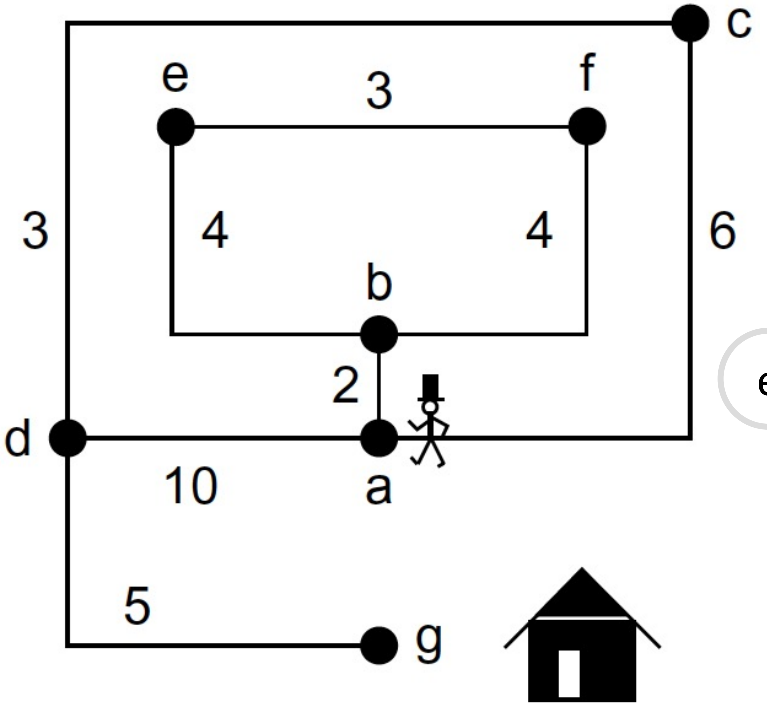
Dijsktra algorithm

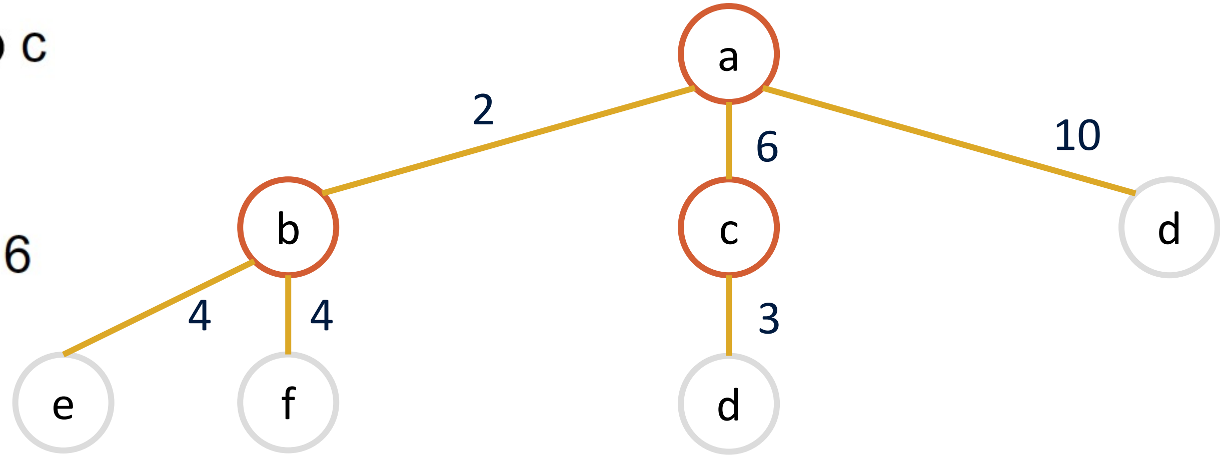




StFX

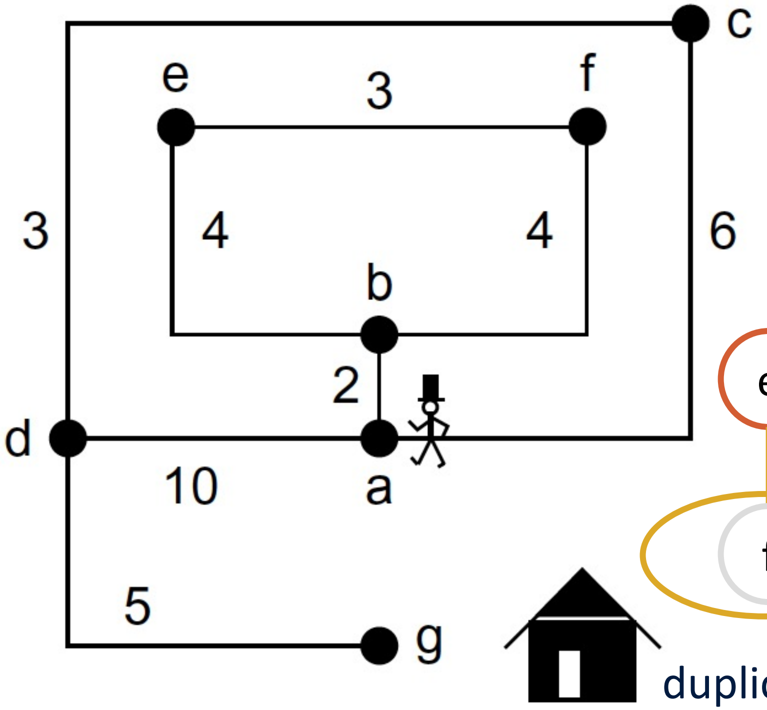
Dijsktra algorithm

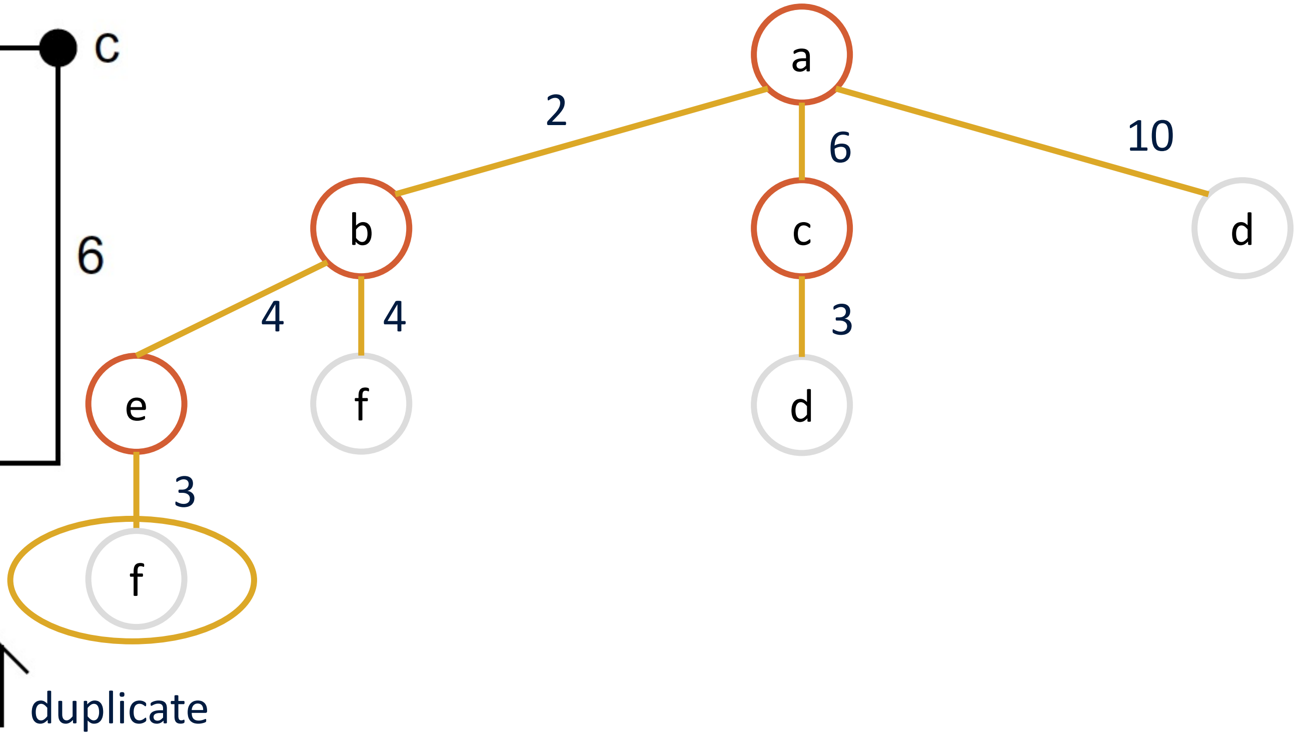




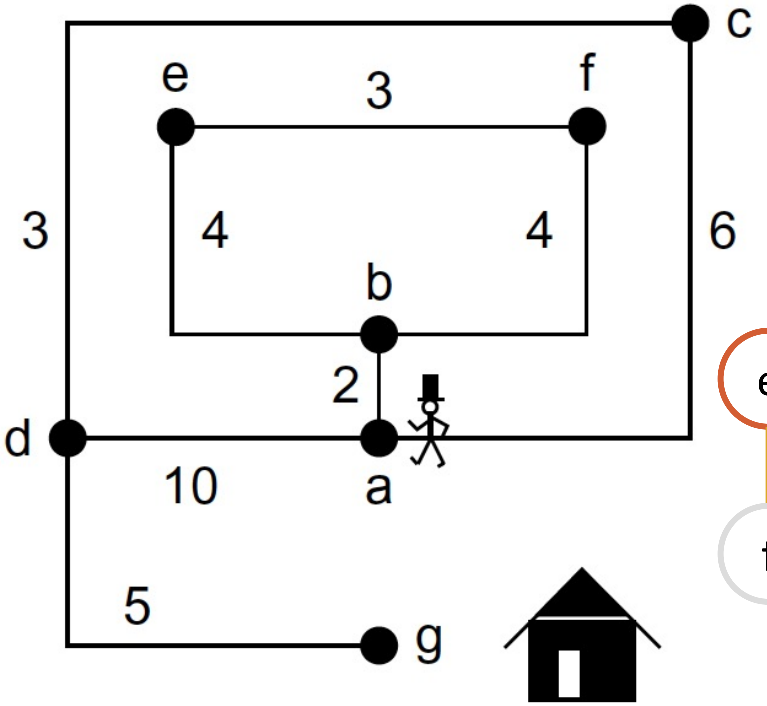
e and f were also possible

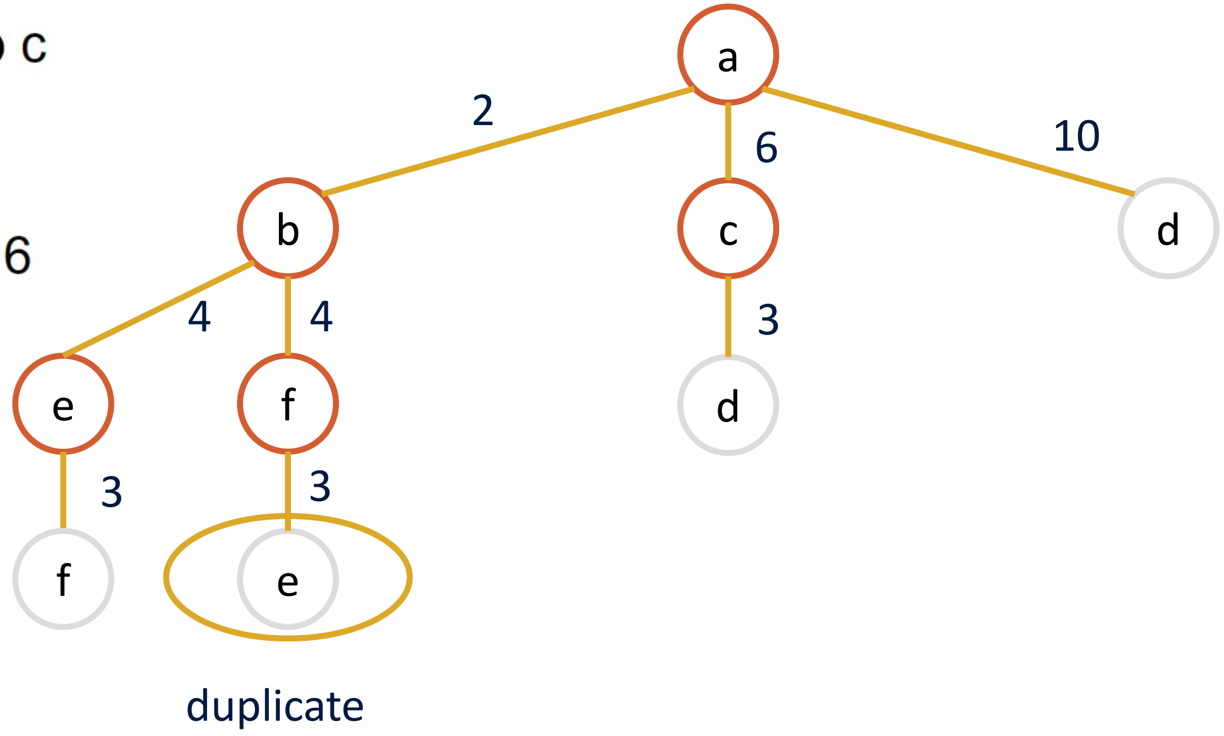
Dijsktra algorithm





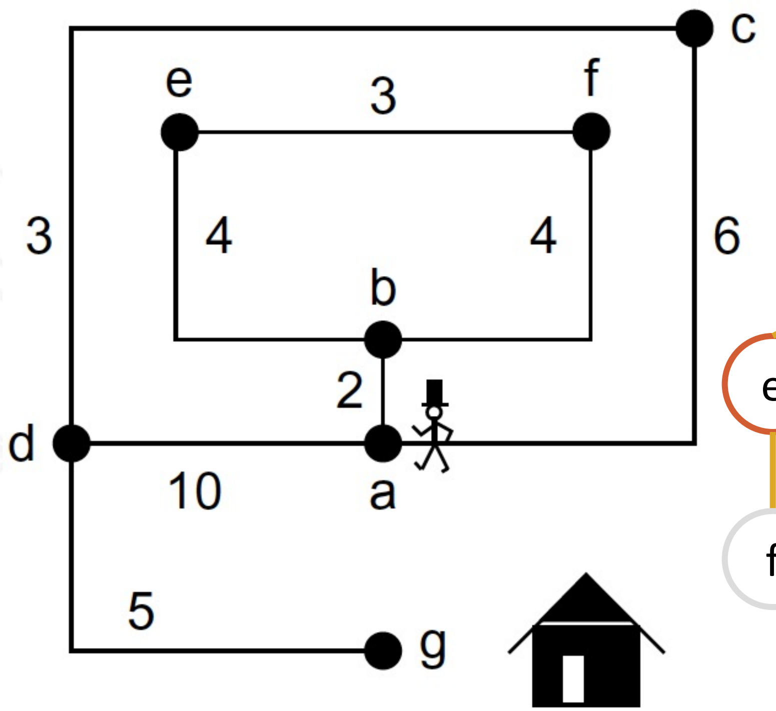
Dijsktra algorithm

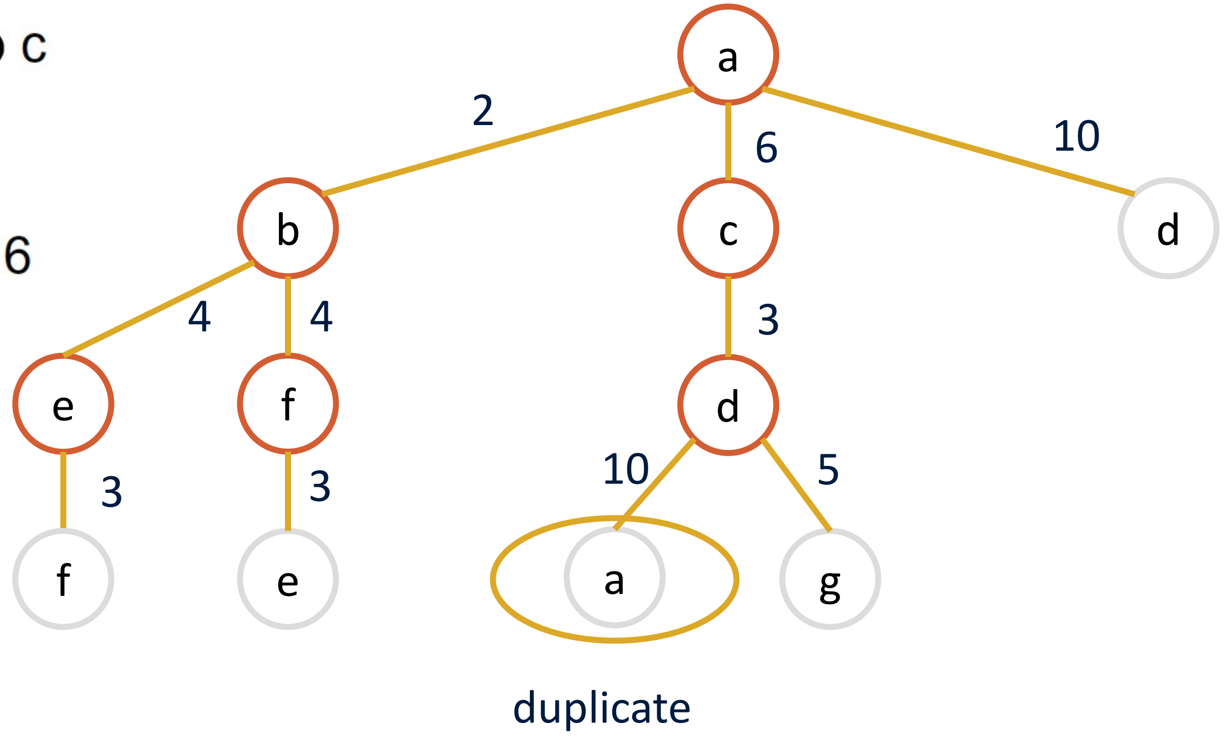




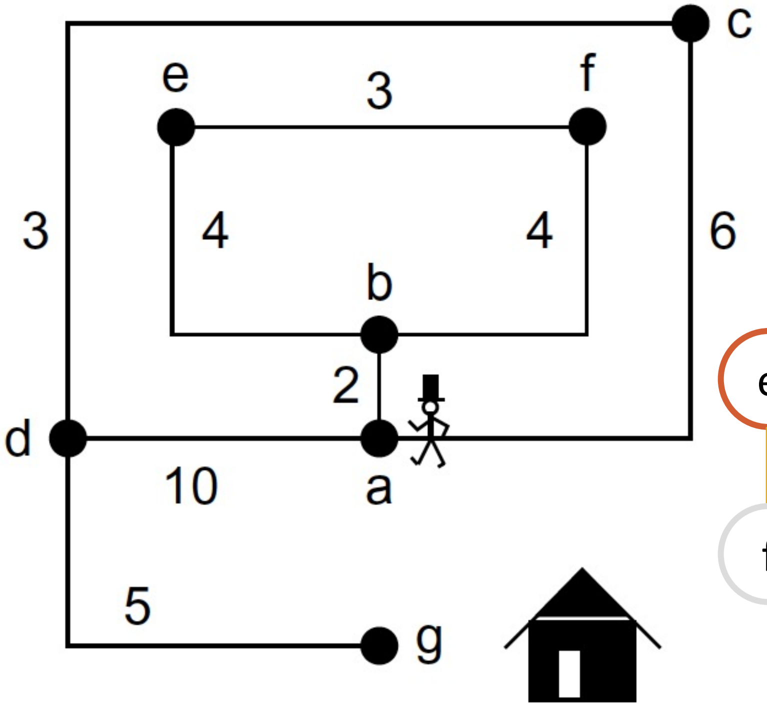
StFX

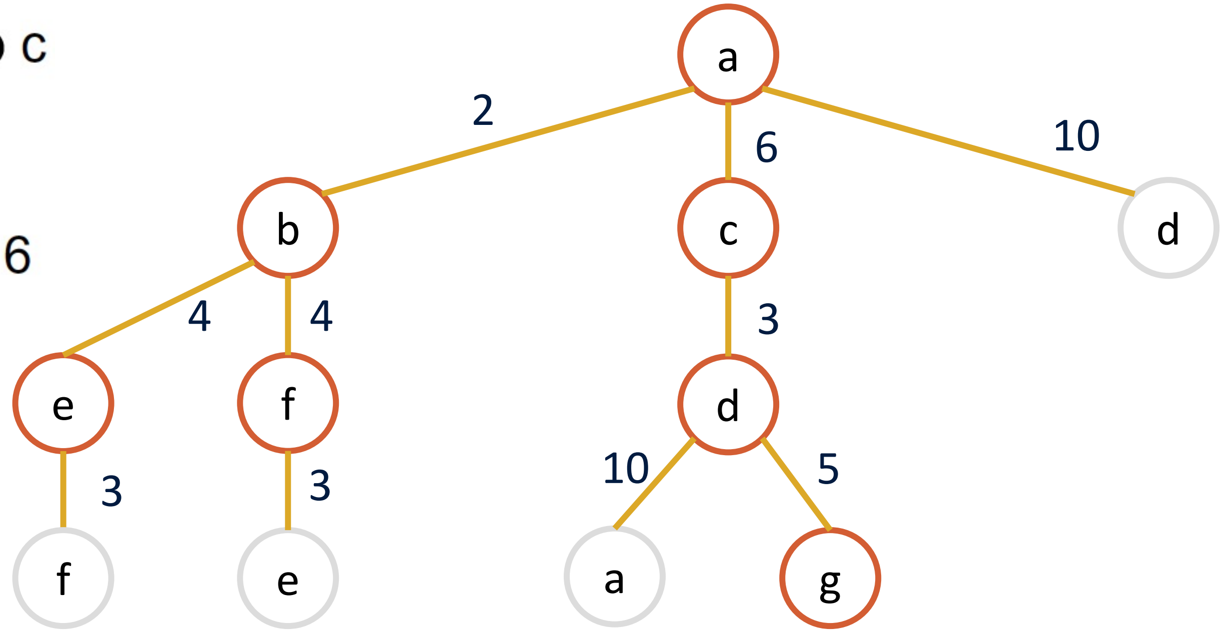
Dijsktra algorithm





Dijsktra algorithm





We stop there. Why?

Because we know that we already have the optimal solution