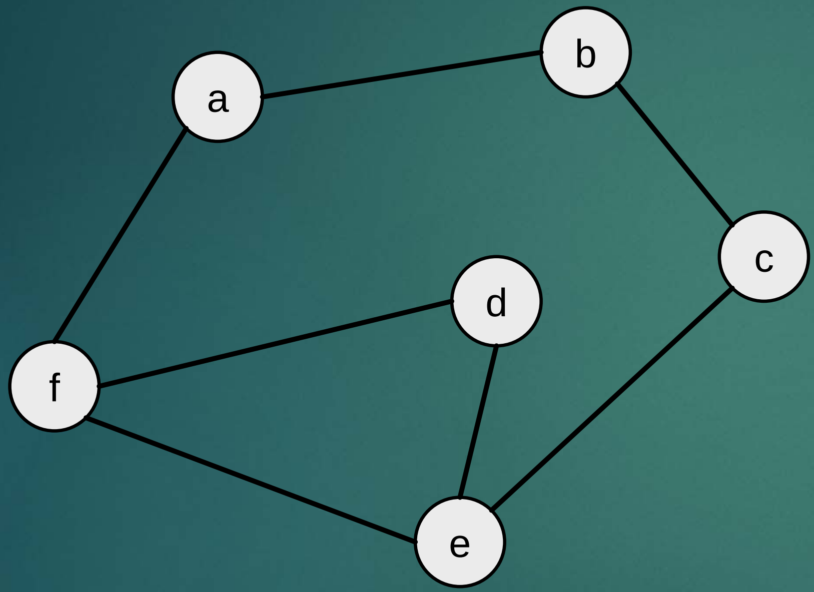
HAMILTONIAN PATH

BACKTRACKING

Hamiltonian cycle

G(V,E)

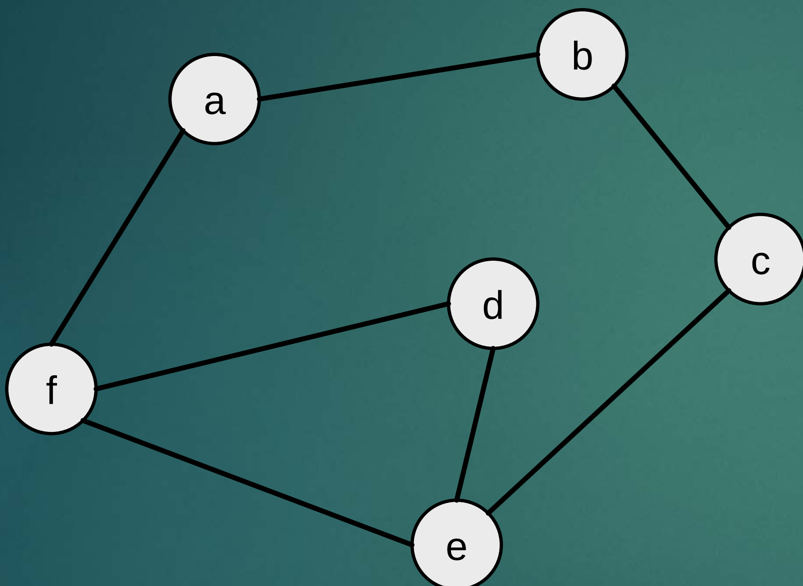
V: vertices in the graph
E: edges in the graph



Hamiltonian cycle

G(V,E)

V: vertices in the graph
E: edges in the graph



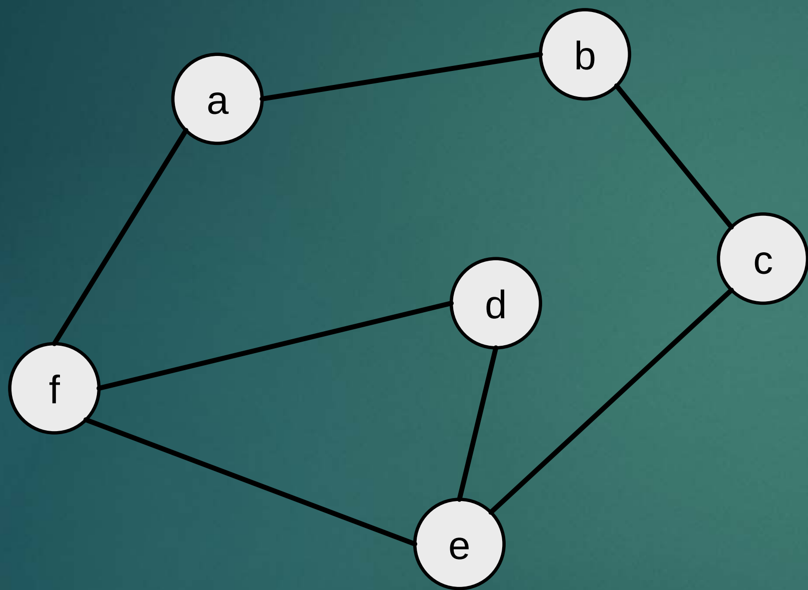
$$
\begin{array}{c|cccccc}
 & a & b & c & d & e & f \\
\hline
a & & & & & & \\
b & & & & & & \\
c & & & & & & \\
d & & & & & & \\
e & & & & & & \\
f & & & & & &
\end{array}
$$

A(i,j) = { 1 – if there is a connection between i and j ; 0 – if no connection }

Hamiltonian cycle

G(V,E)

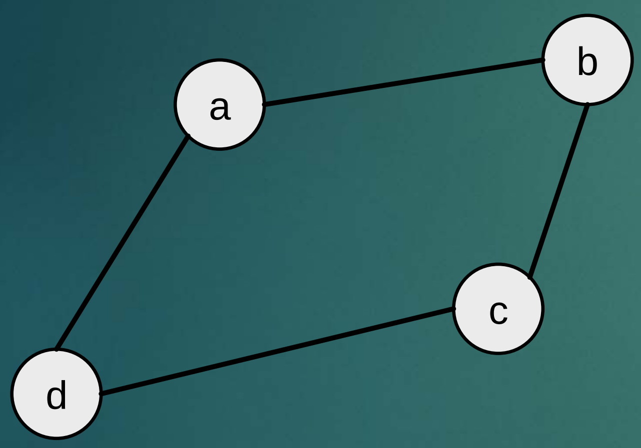
V: vertices in the graph
E: edges in the graph



$$\begin{array}{c c} & \begin{array}{cccccc} a & b & c & d & e & f \end{array} \\ \begin{array}{c} a \\ b \\ c \\ d \\ e \\ f \end{array} & \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

A(i,j) = { 1 – if there is a connection between i and j ; 0 – if no connection }

Hamiltonian path

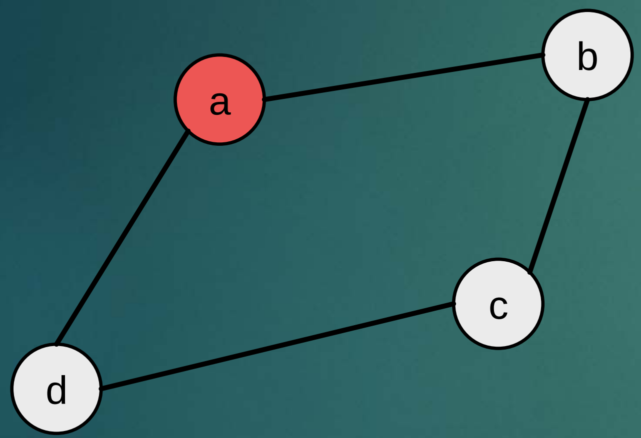


A hamiltonian path in an undirected graph is a path that visits every node exactly once !!!

Hamiltonian cycle: the first node and the last node of the path are the same vertexes

StartingPoint == EndPoint

Hamiltonian path

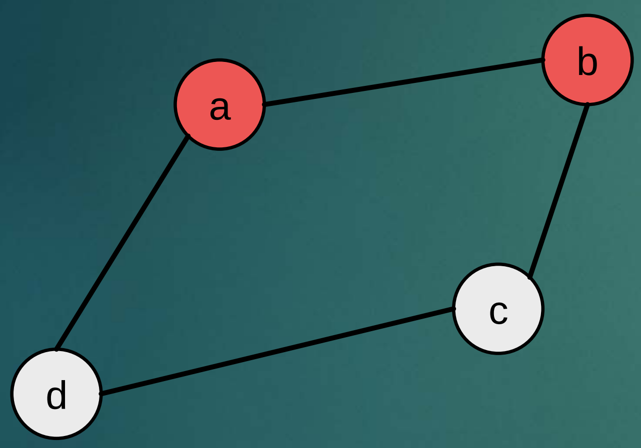


A hamiltonian path in an undirected graph is a path that visits every node exactly once !!!

Hamiltonian cycle: the first node and the last node of the path are the same vertexes

StartingPoint == EndPoint

Hamiltonian path

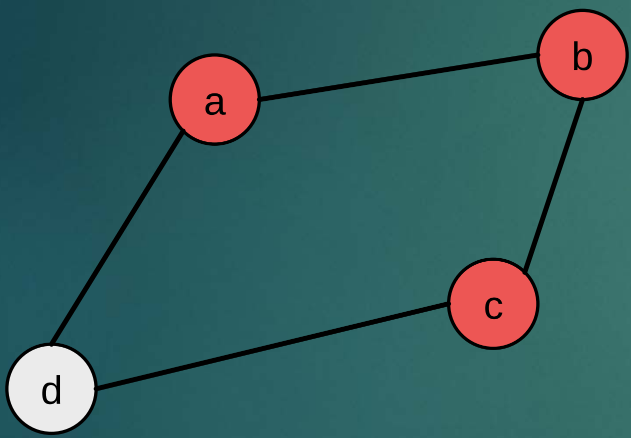


A hamiltonian path in an undirected graph is a path that visits every node exactly once !!!

Hamiltonian cycle: the first node and the last node of the path are the same vertexes

StartingPoint == EndPoint

Hamiltonian path

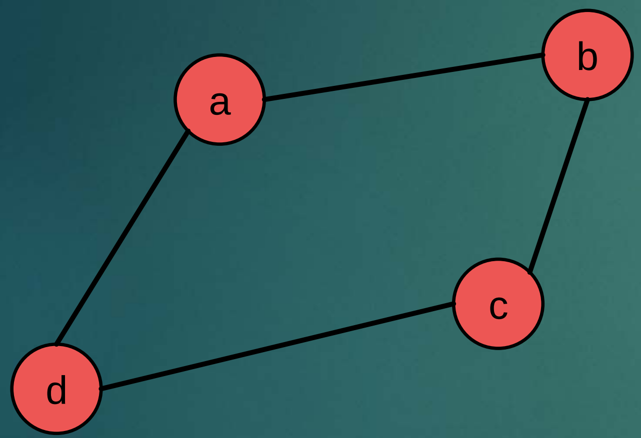


A hamiltonian path in an undirected graph is a path that visits every node exactly once !!!

Hamiltonian cycle: the first node and the last node of the path are the same vertexes

StartingPoint == EndPoint

Hamiltonian path

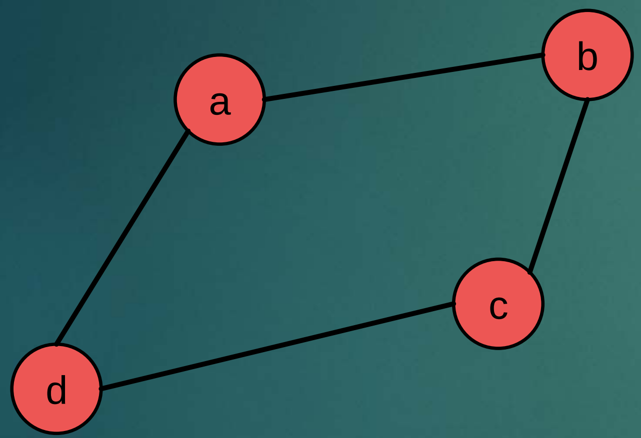


A hamiltonian path in an undirected graph is a path that visits every node exactly once !!!

Hamiltonian cycle: the first node and the last node of the path are the same vertexes

StartingPoint == EndPoint

Hamiltonian path



A hamiltonian path in an undirected graph is a path that visits every node exactly once !!!

Hamiltonian cycle: the first node and the last node of the path are the same vertexes

StartingPoint == EndPoint

A valid hamiltonian path is: **{ a b c d a }**

There may be several hamiltonian path in a given graph !!!

Hamiltonian problem

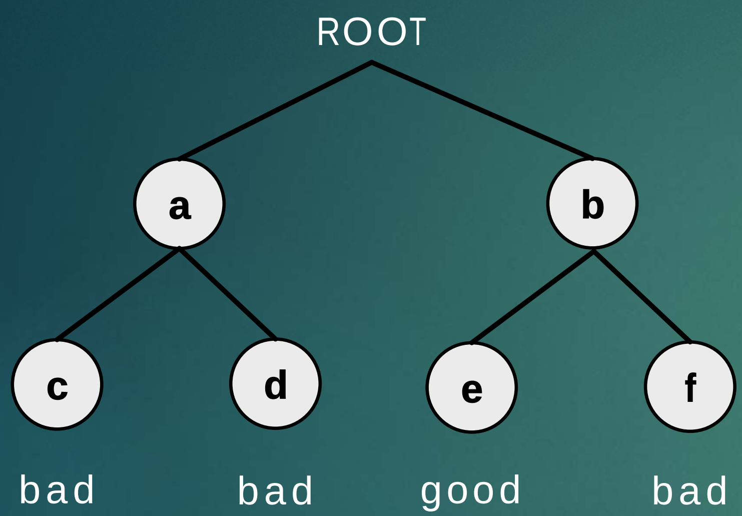
- Determining whether such paths and cycles exist in graphs is the Hamiltonian path problem
- This is an **NP-complete** complete problem !!!
- Dirac-principle: a simple graph with **N** vertices is hamiltonian if every vertex has degree **N/2** or greater (degree is the number of edges of a vertex)
- Important fact: finding Hamiltonian path is **NP-complete**, but we can decide whether such path exists in linear time complexity with topological ordering

Solutions

- Naive approach:
- Generate all possible configurations of the vertices and print a configuration that satisfies the given constraints
- Problem → if the graph has **N** vertices, there are **N!** configurations, so the „solution space” is enormous
- Very very inefficient !!!

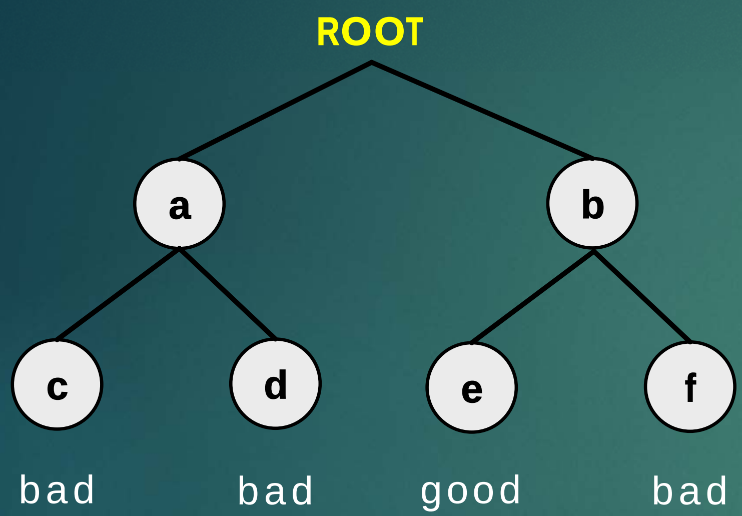
Solutions

- Constructing a tree:



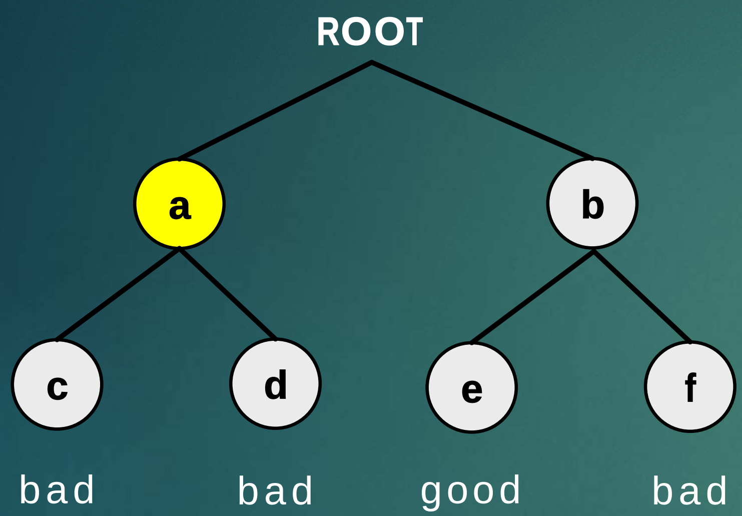
We start at the root node, and
want to get to one of the good leaves

If we get to a bad leaf: we just „backtrack"
and keep moving on by revoking
our most recent choice



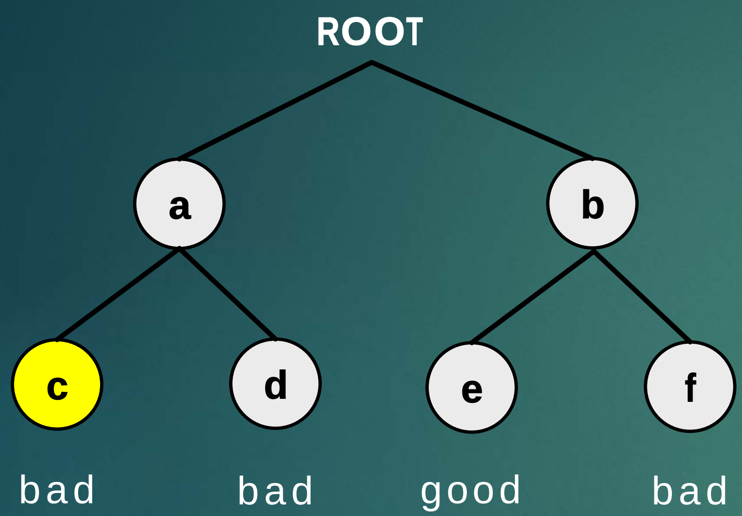
We start at the root node, and
want to get to one of the good leaves

If we get to a bad leaf: we just „backtrack”
and keep moving on by revoking
our most recent choice



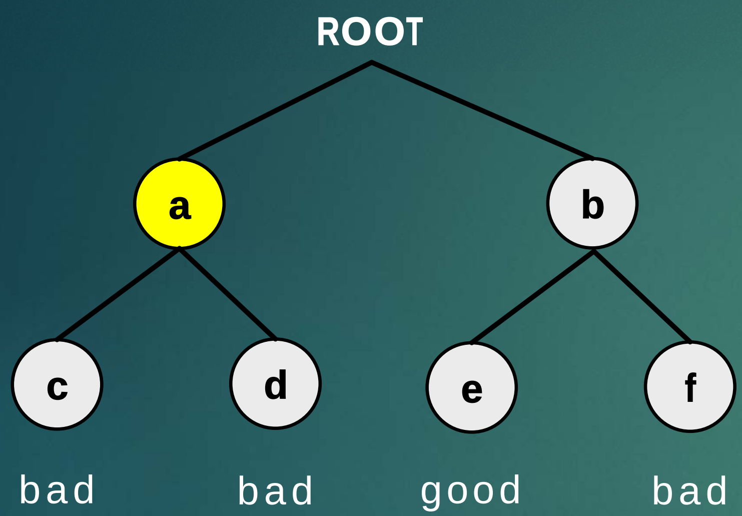
We start at the root node, and
want to get to one of the good leaves

If we get to a bad leaf: we just „backtrack”
and keep moving on by revoking
our most recent choice



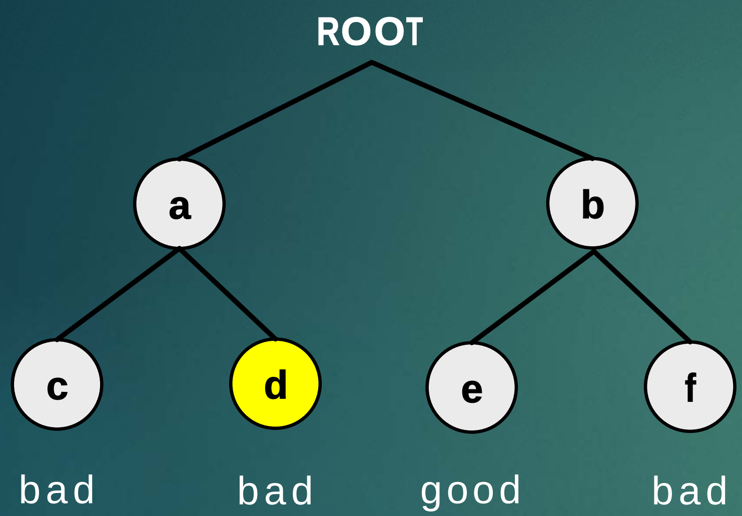
We start at the root node, and
want to get to one of the good leaves

If we get to a bad leaf: we just „backtrack”
and keep moving on by revoking
our most recent choice



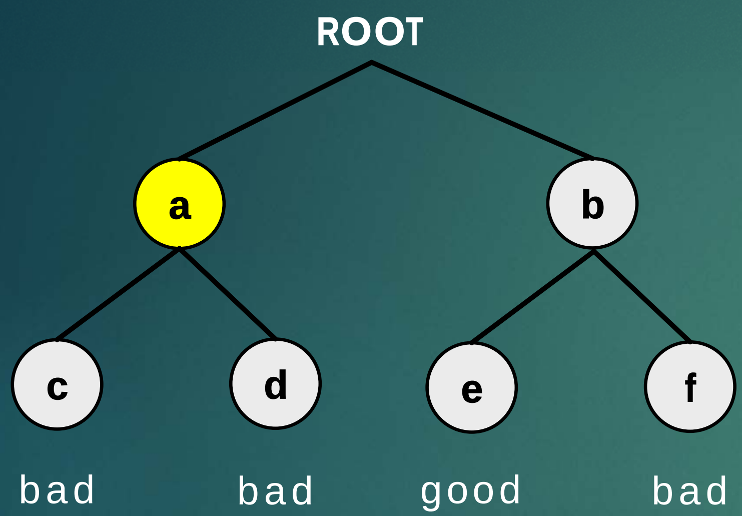
We start at the root node, and
want to get to one of the good leaves

If we get to a bad leaf: we just „backtrack"
and keep moving on by revoking
our most recent choice



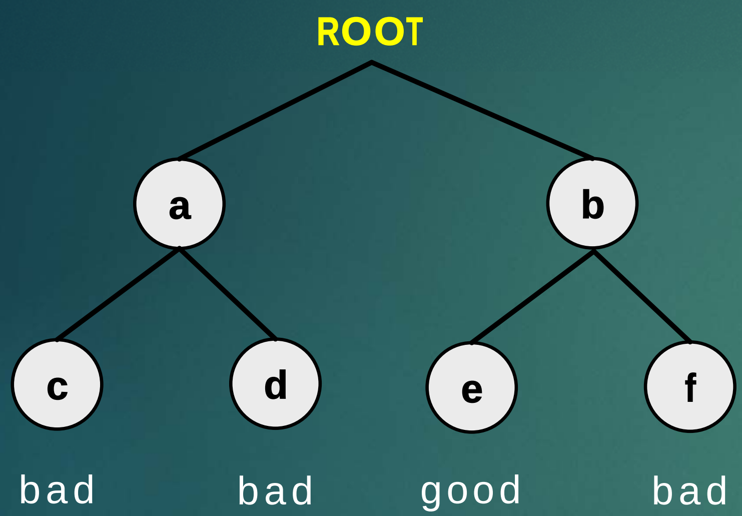
We start at the root node, and want to get to one of the good leaves

If we get to a bad leaf: we just „backtrack" and keep moving on by revoking our most recent choice



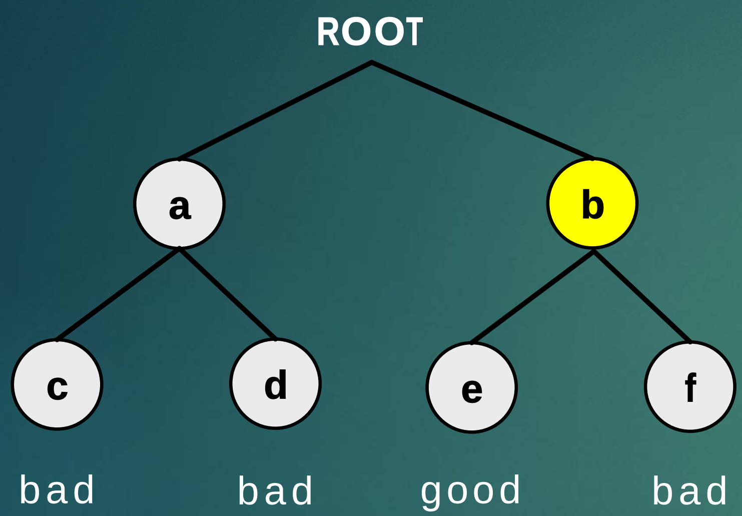
We start at the root node, and
want to get to one of the good leaves

If we get to a bad leaf: we just „backtrack"
and keep moving on by revoking
our most recent choice



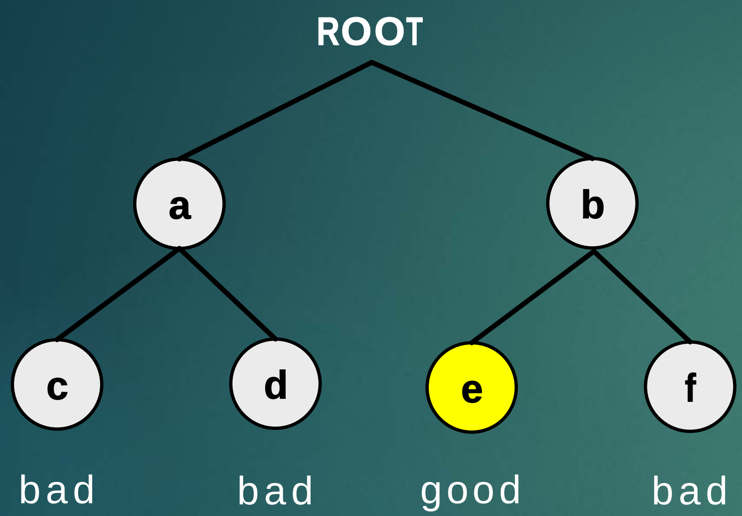
We start at the root node, and
want to get to one of the good leaves

If we get to a bad leaf: we just „backtrack”
and keep moving on by revoking
our most recent choice



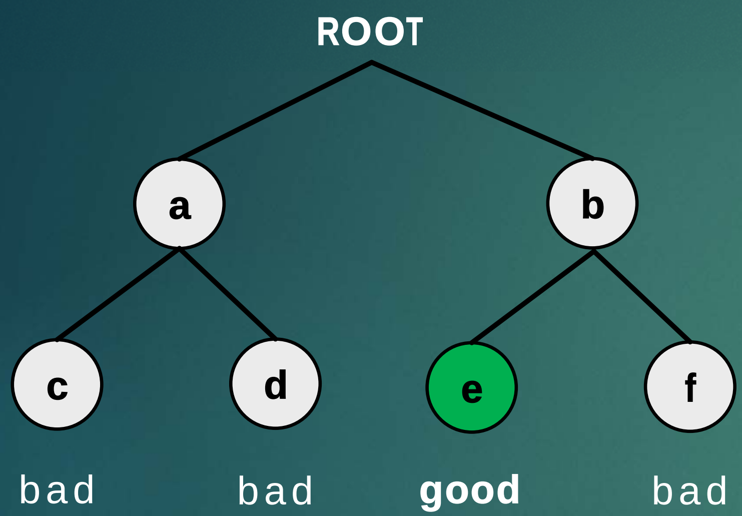
We start at the root node, and
want to get to one of the good leaves

If we get to a bad leaf: we just „backtrack”
and keep moving on by revoking
our most recent choice



We start at the root node, and
want to get to one of the good leaves

If we get to a bad leaf: we just „backtrack”
and keep moving on by revoking
our most recent choice



We start at the root node, and
want to get to one of the good leaves

If we get to a bad leaf: we just „backtrack”
and keep moving on by revoking
our most recent choice

The tree is an abstract model of the possible sequences of choices we could make
Here we do a depth-first search on the tree

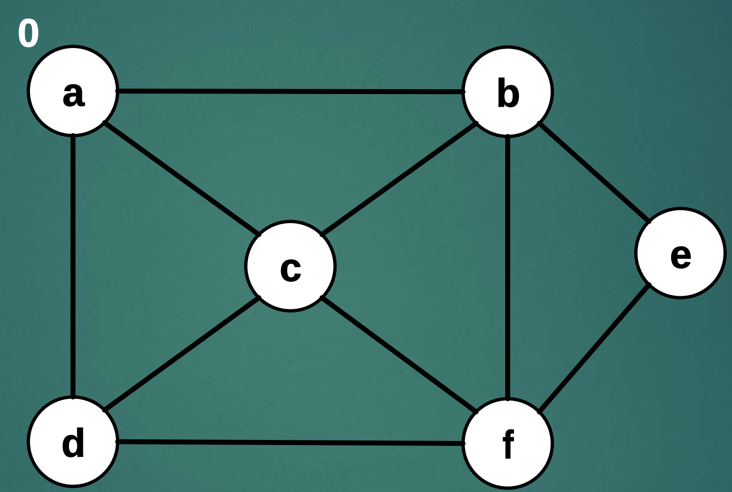
Problem: hard to construct a tree if **N** is big !!!

Solutions

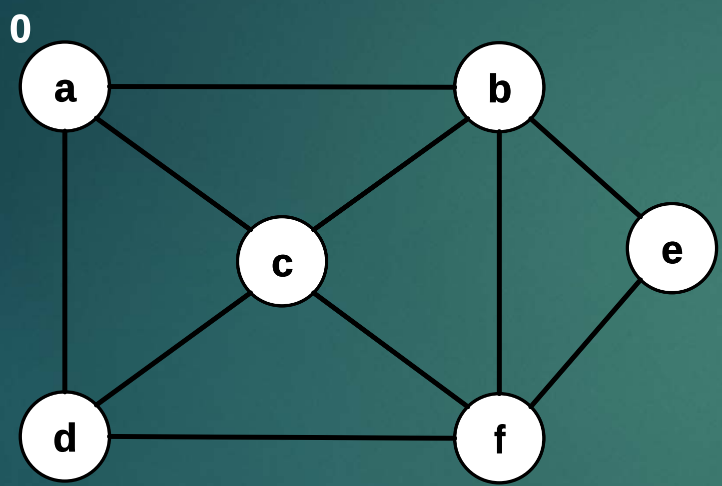
- Backtracking:
- We use recursion to solve the problem
- Create an empty path array and add vertex **0** to it as the starting vertex
- Add other vertices, starting from the vertex **1**
- Before adding a vertex, check whether it is adjacent to the previously added vertex + make sure it is not already added
- If we find such a vertex → we add the vertex as part of the solution
- If we do not find a vertex → we return false

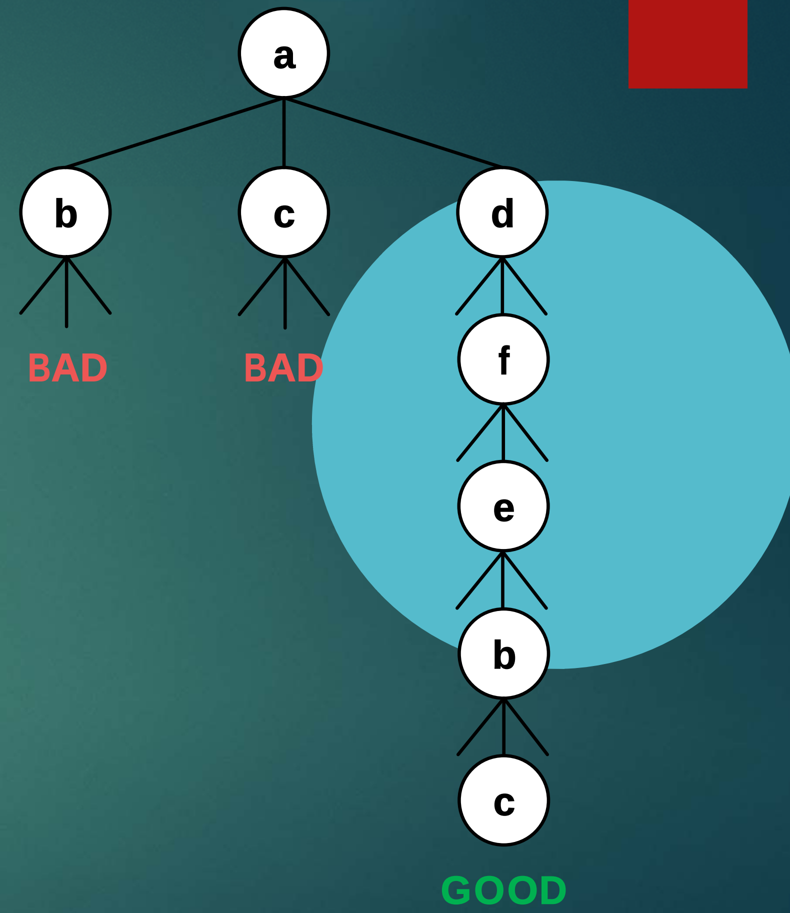
State-space tree

State-space tree

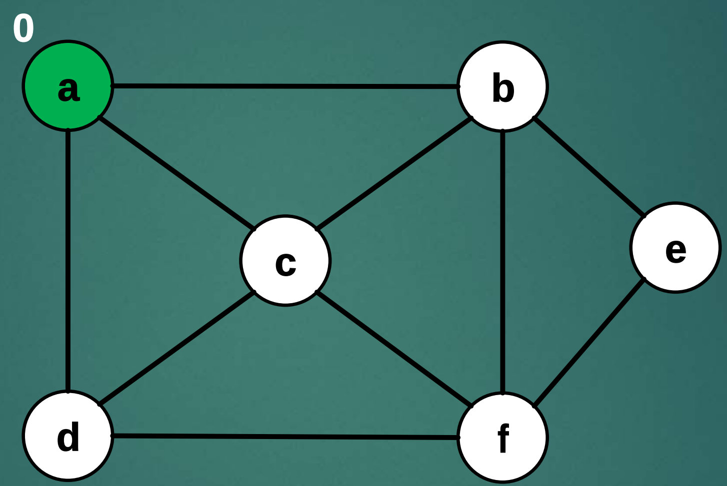


State-space tree

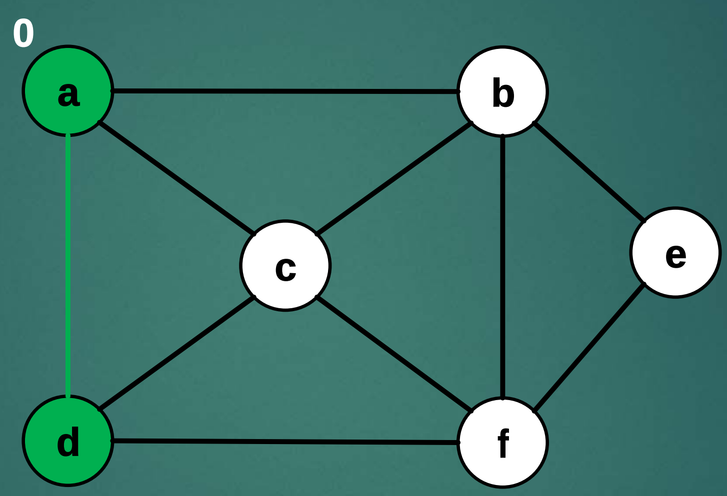




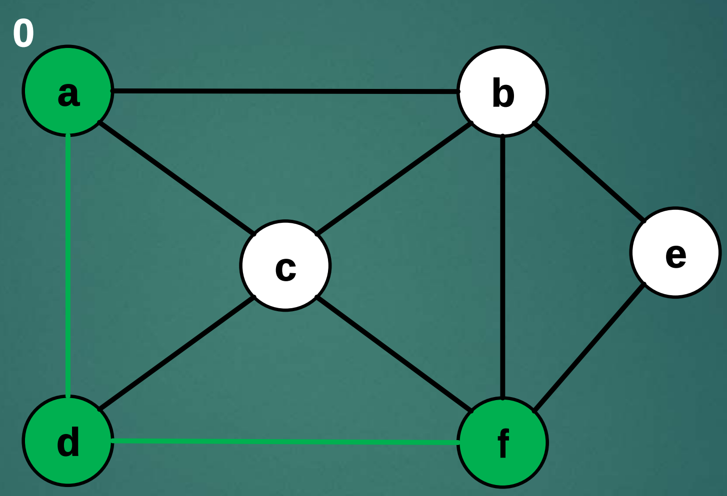
State-space tree



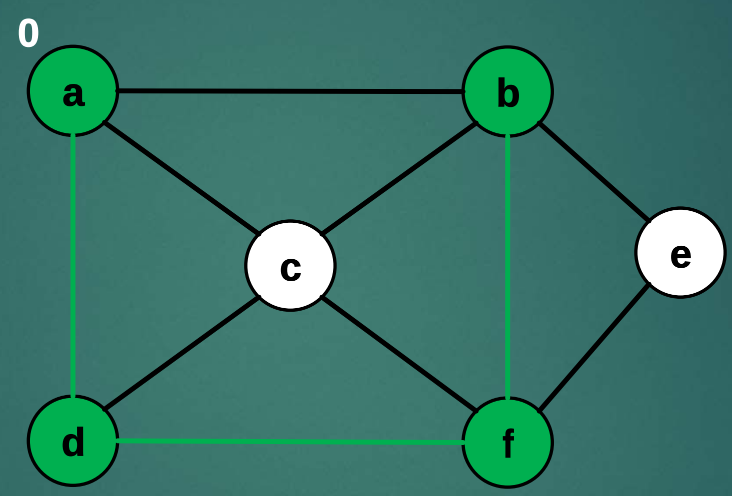
State-space tree



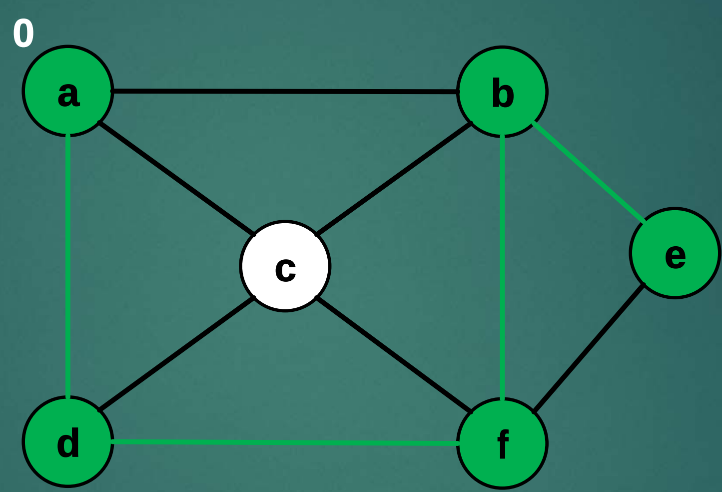
State-space tree



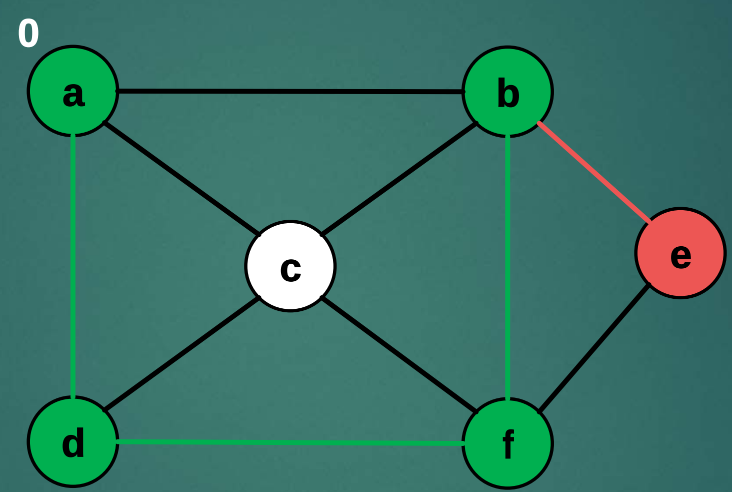
State-space tree



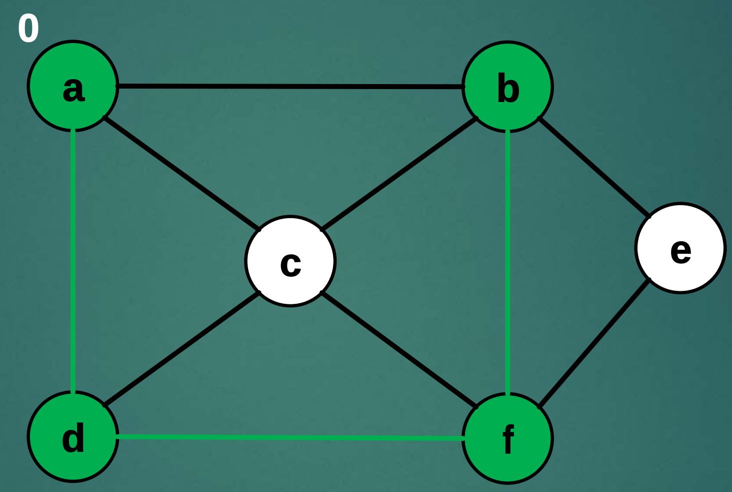
State-space tree



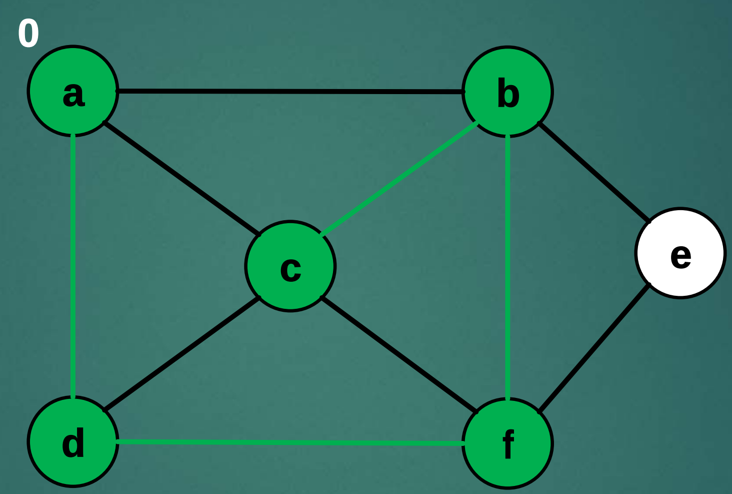
State-space tree



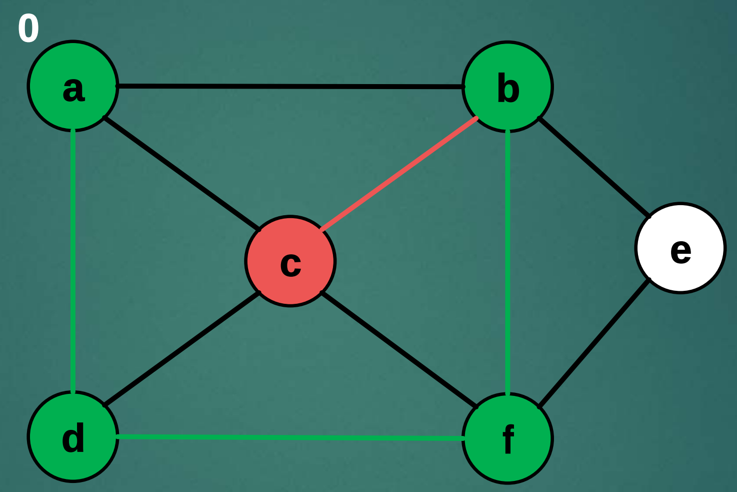
State-space tree



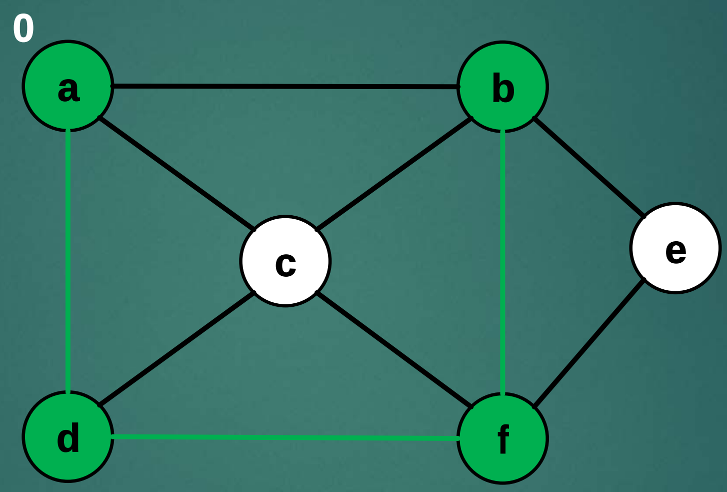
State-space tree



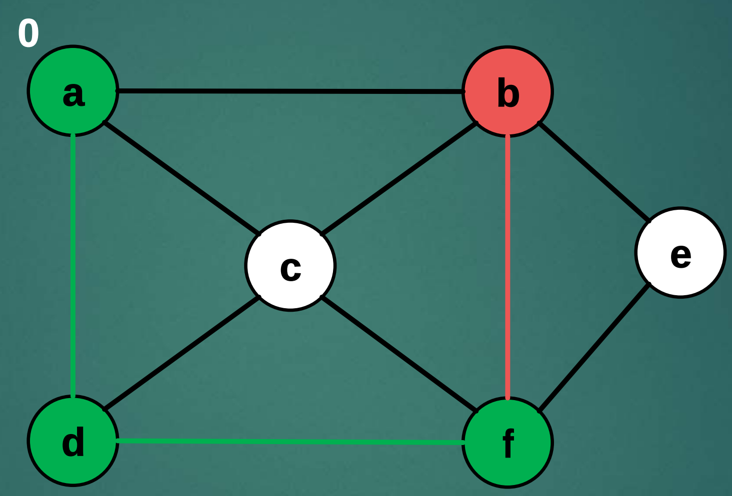
State-space tree



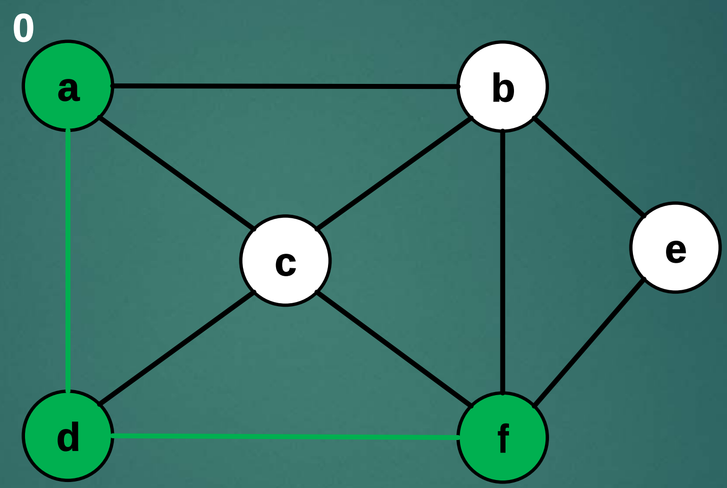
State-space tree



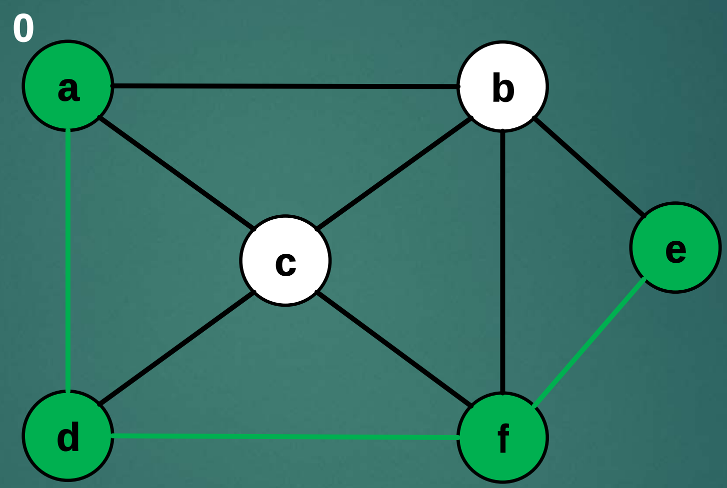
State-space tree



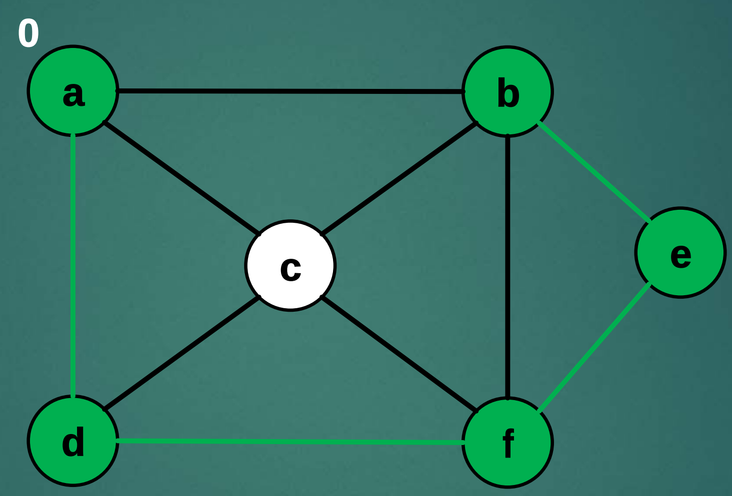
State-space tree



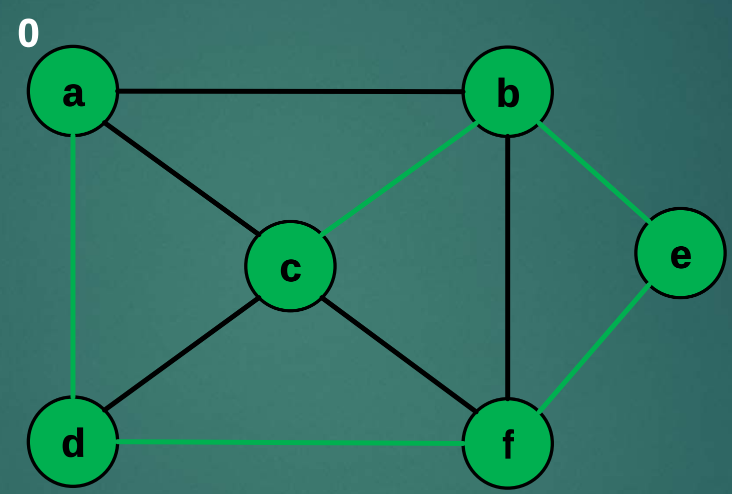
State-space tree



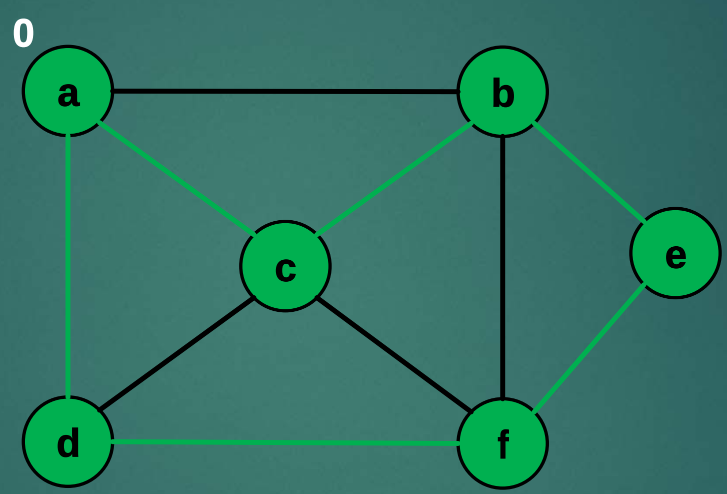
State-space tree



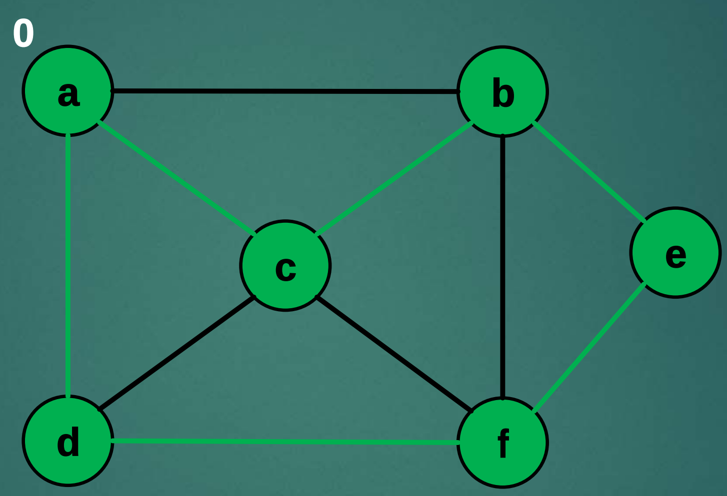
State-space tree



State-space tree



State-space tree



We have found the Hamiltonian-cycle in this graph

{ a , d , f , e , b , c , a }

State-space tree