Tree search for Combinatorial Optimization

Luc Libralesso

June 14, 2020

Operations Research and Combinatorial Optimization master lecture

Goal of this lecture

- Quickly present search algorithms for optimization

Goal of this lecture

- Quickly present search algorithms for optimization
- Position algorithms you have seen in a more general context

Goal of this lecture

- Quickly present search algorithms for optimization
- Position algorithms you have seen in a more general context
- Study Tree search principles

Table of contents

Introduction: Optimization and search

Combinatorial Optimization & NP-Hardness

We want to find the best possible solution out of a finite and **huge** number of solutions.

Example: (Asymmetric) Traveling Salesman Problem

INPUT:

- graph $G = (V, A)$
- distance function $w : A \to \mathbb{R}$

Example: (Asymmetric) Traveling Salesman Problem

Input:

- graph $G = (V, A)$
- distance function $w : A \to \mathbb{R}$

Goal:

- Find a tour that visit all n cities
- Minimize the distance of selected arcs
- $n!$ possible solutions

Resolution Methods (you may already know)

- Brute Force (very bad)

Resolution Methods (you may already know)

- Brute Force (very bad)
- Branch and Bound (better)

Resolution Methods (you may already know)

- Brute Force (very bad)
- Branch and Bound (better)
- Mixed Integer Programming (LP + Branch and Bound)

Resolution Methods (you may already know)

- Brute Force (very bad)
- Branch and Bound (better)
- Mixed Integer Programming (LP + Branch and Bound)
- Constraint Programming (fixed point algorithm + Tree Search)

Resolution Methods (you may already know)

- Brute Force (very bad)
- Branch and Bound (better)
- Mixed Integer Programming (LP + Branch and Bound)
- Constraint Programming (fixed point algorithm + Tree Search)
- Meta-heuristics:
  - Local Search
  - Simulated Annealing
  - Genetic Algorithms
  - Ant Colony Optimization
  - *etc.*

Classification

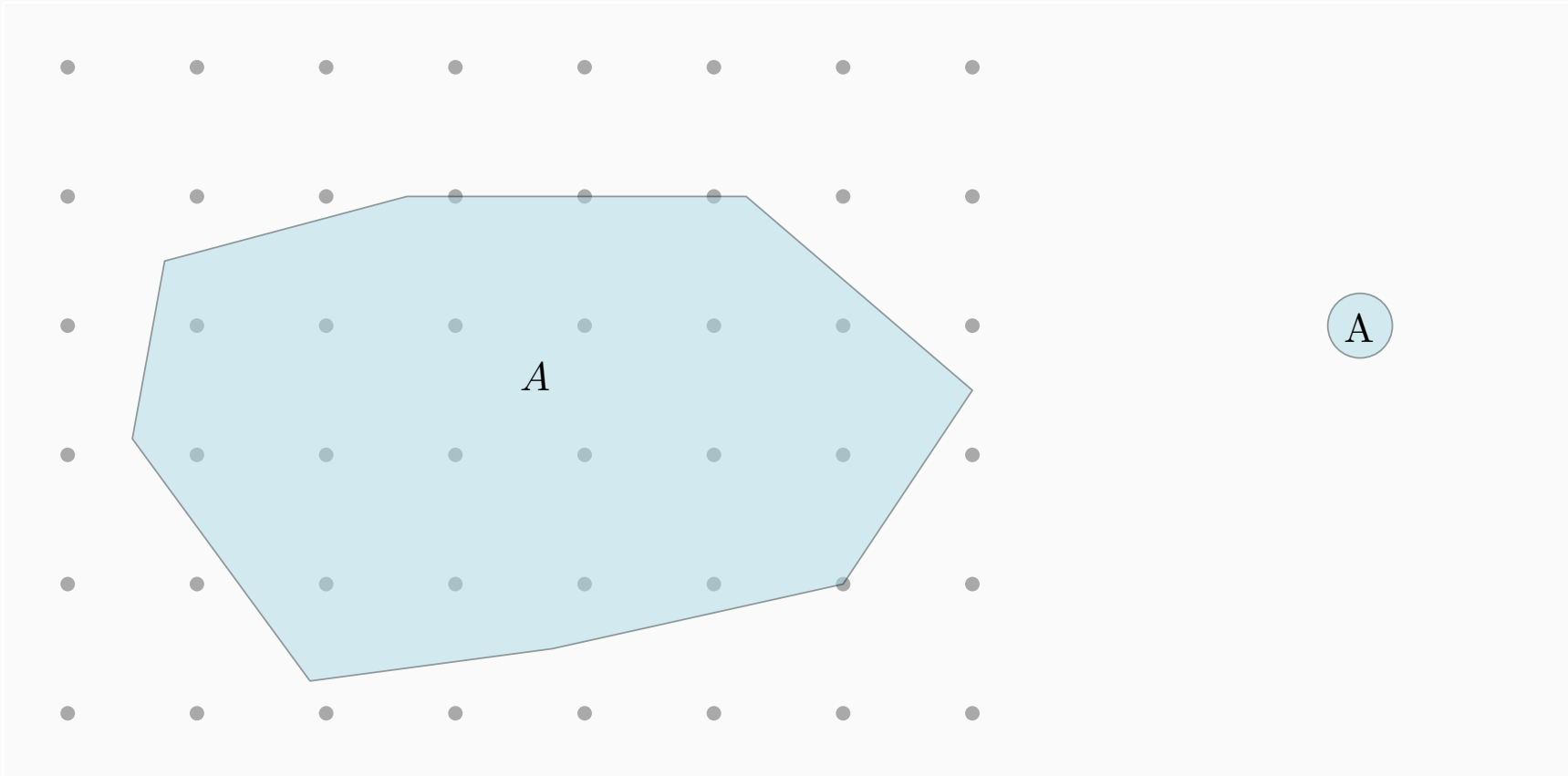
Search procedures are often labeled as:

- Tree Search
- Local Search
- Population Based Search

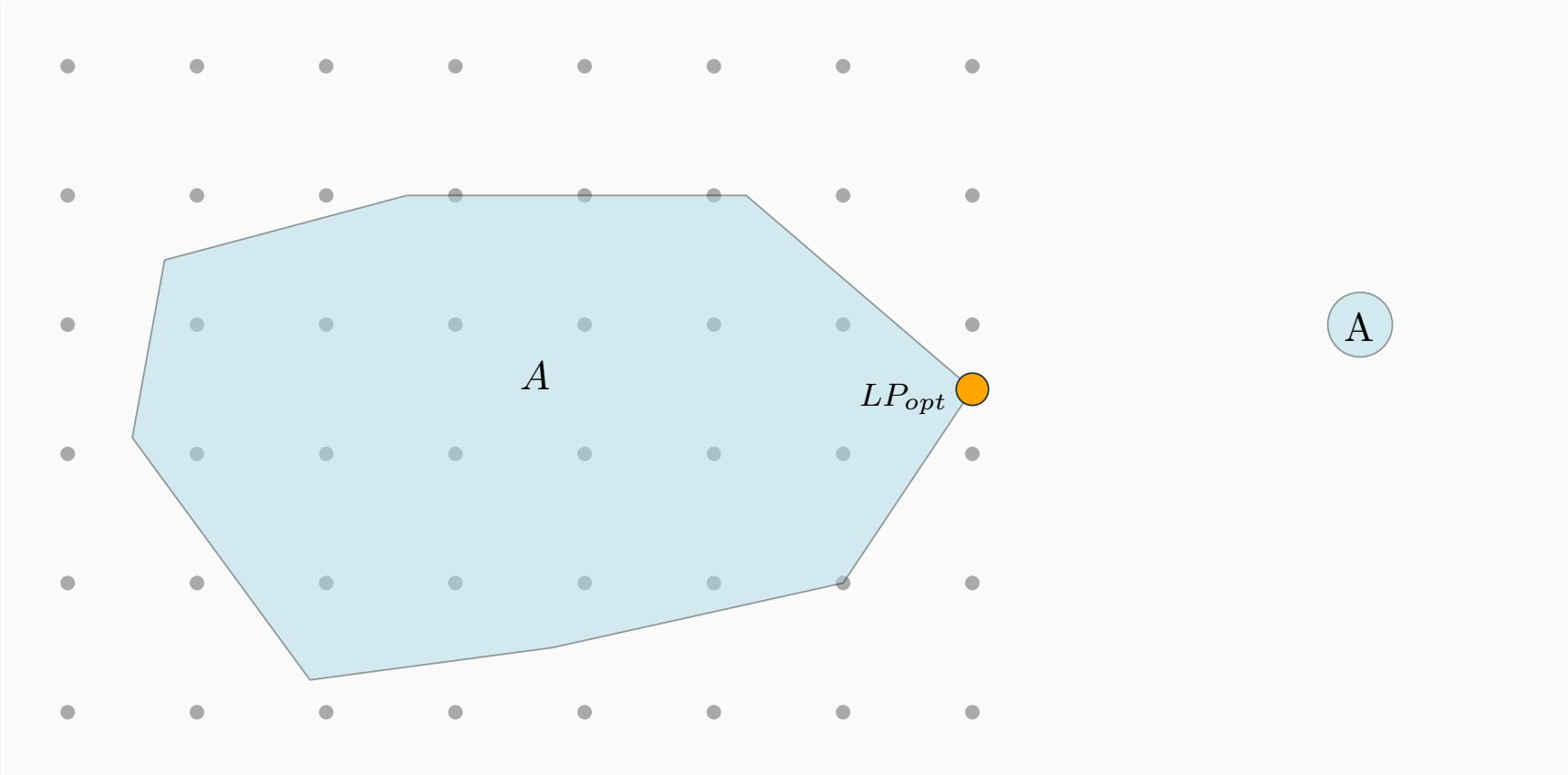
Tree Search

- usually ”constructs” solutions
- Models the problem as a tree
- Explores this tree

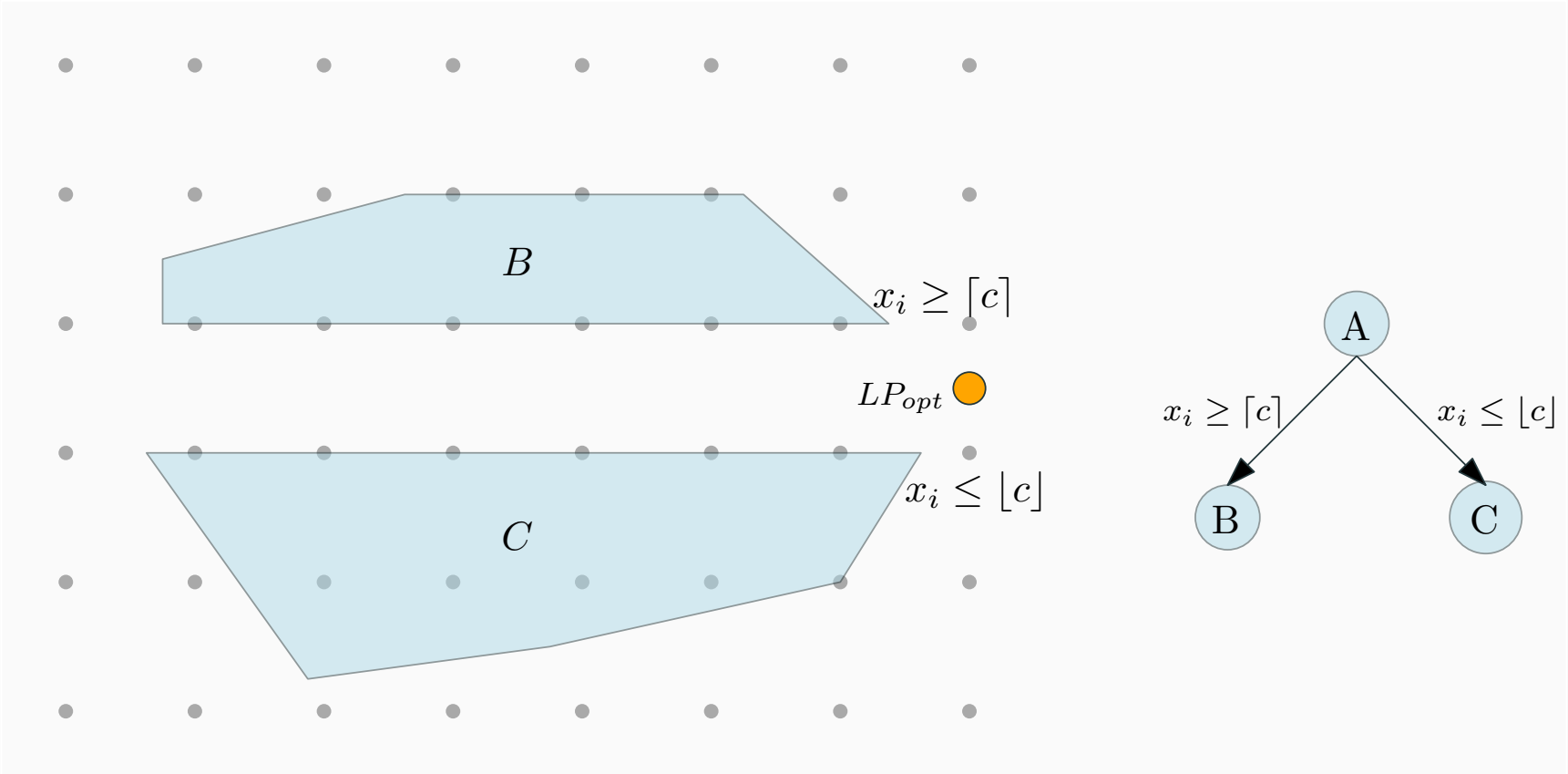
Example: Mixed Integer Programming



Example: Mixed Integer Programming



Example: Mixed Integer Programming

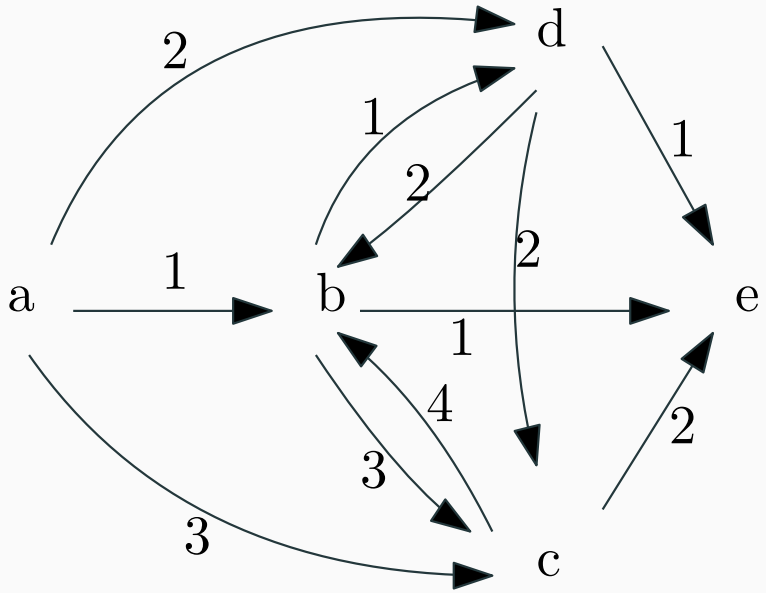


Example: Constraint Programming

- Perform at each node a domain reduction (fixed point algorithm)
- Create children by adding different constraints
  - $x_i = 5$
  - $x_i \neq 5$

Example: Dedicated Branch and Bound

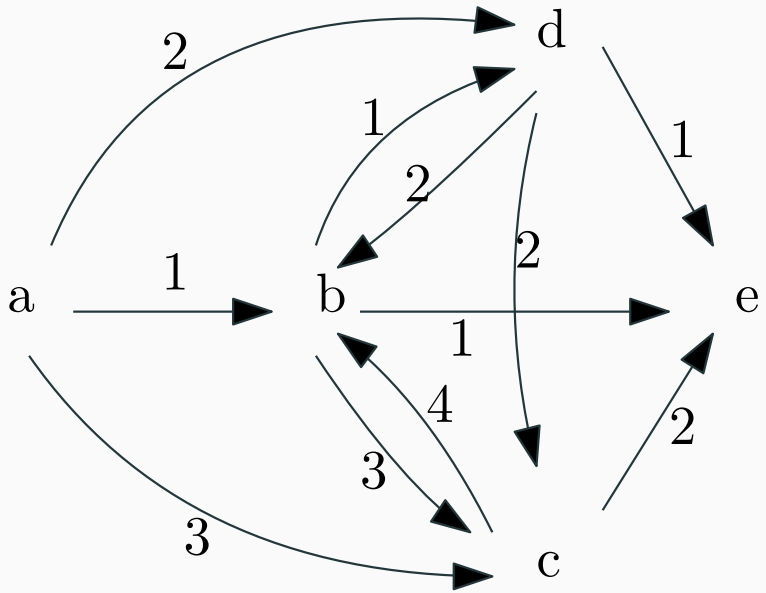
A **Sequential Ordering Problem** dedicated Branch and Bound

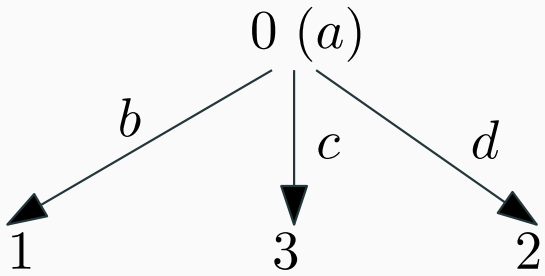


$0\ (a)$

Example: Dedicated Branch and Bound

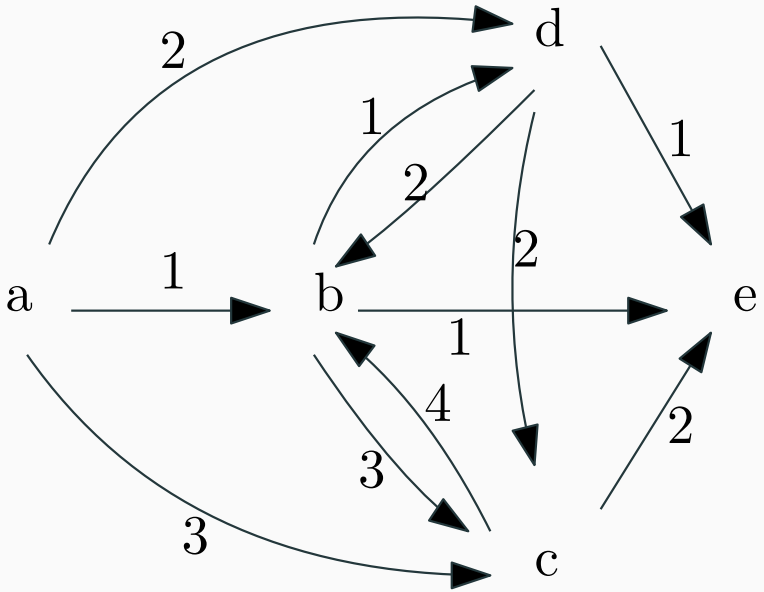
A **Sequential Ordering Problem** dedicated Branch and Bound

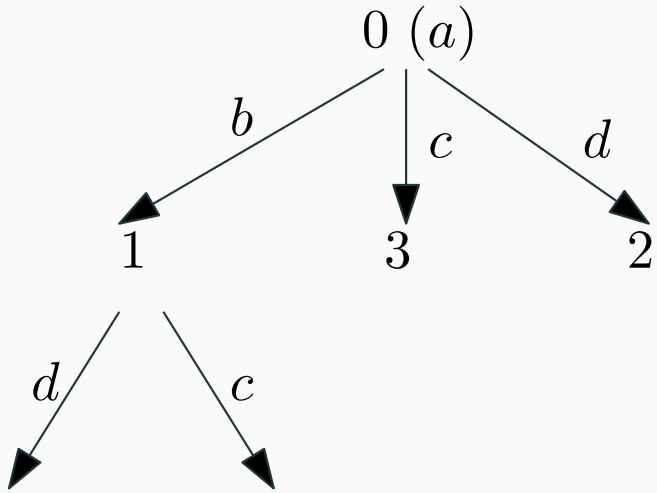




Example: Dedicated Branch and Bound

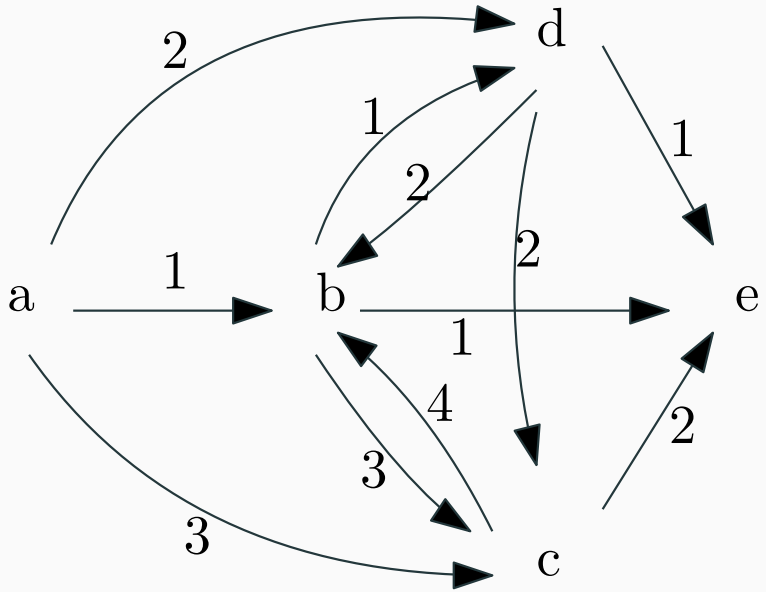
A **Sequential Ordering Problem** dedicated Branch and Bound

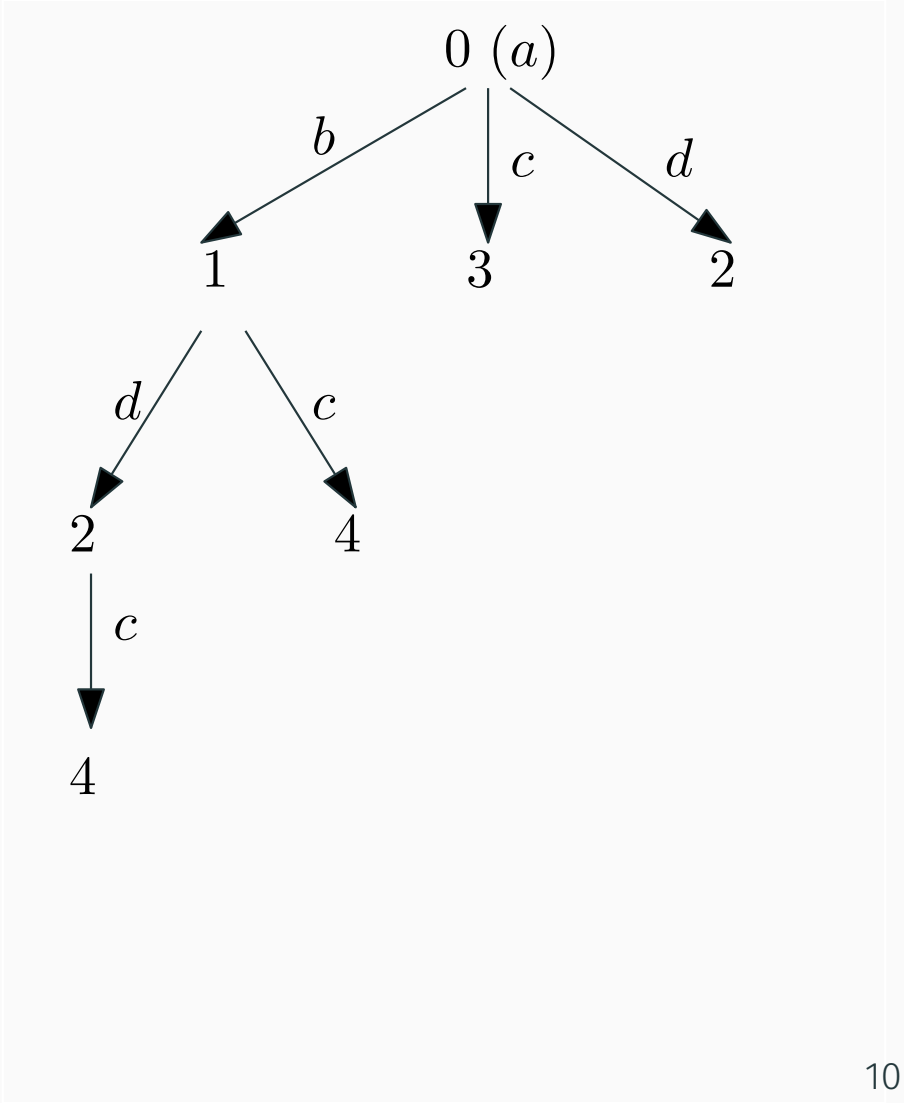




Example: Dedicated Branch and Bound

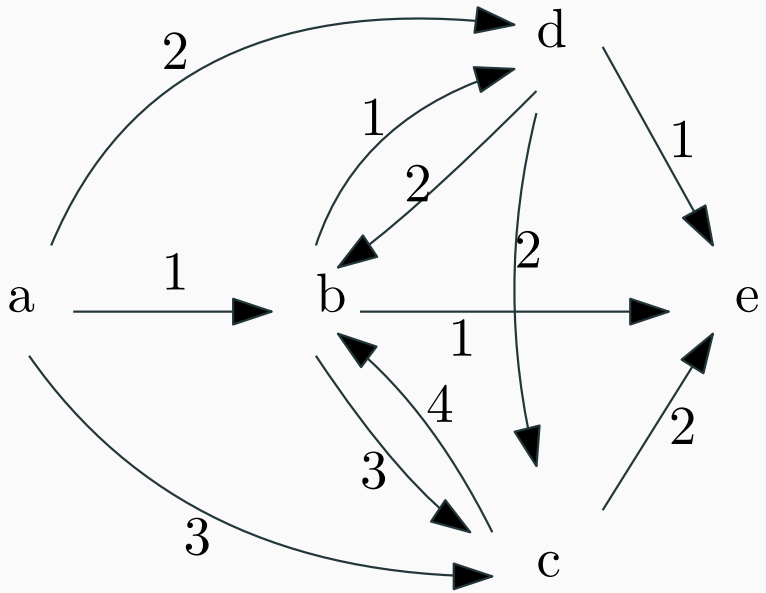
A **Sequential Ordering Problem** dedicated Branch and Bound

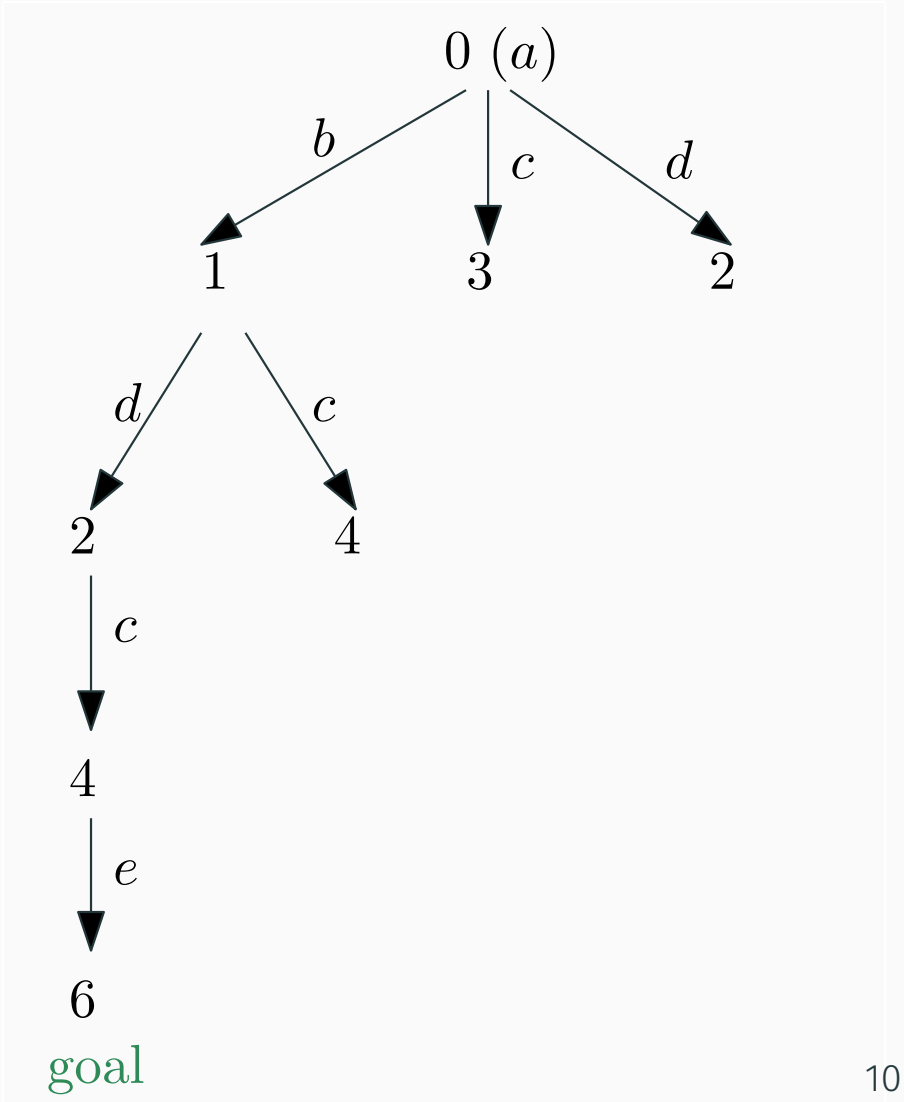




Example: Dedicated Branch and Bound

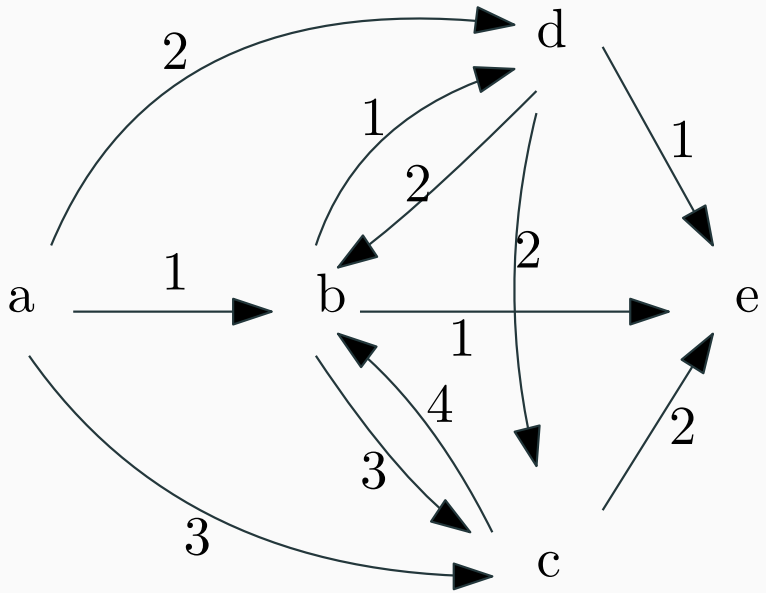
A **Sequential Ordering Problem** dedicated Branch and Bound

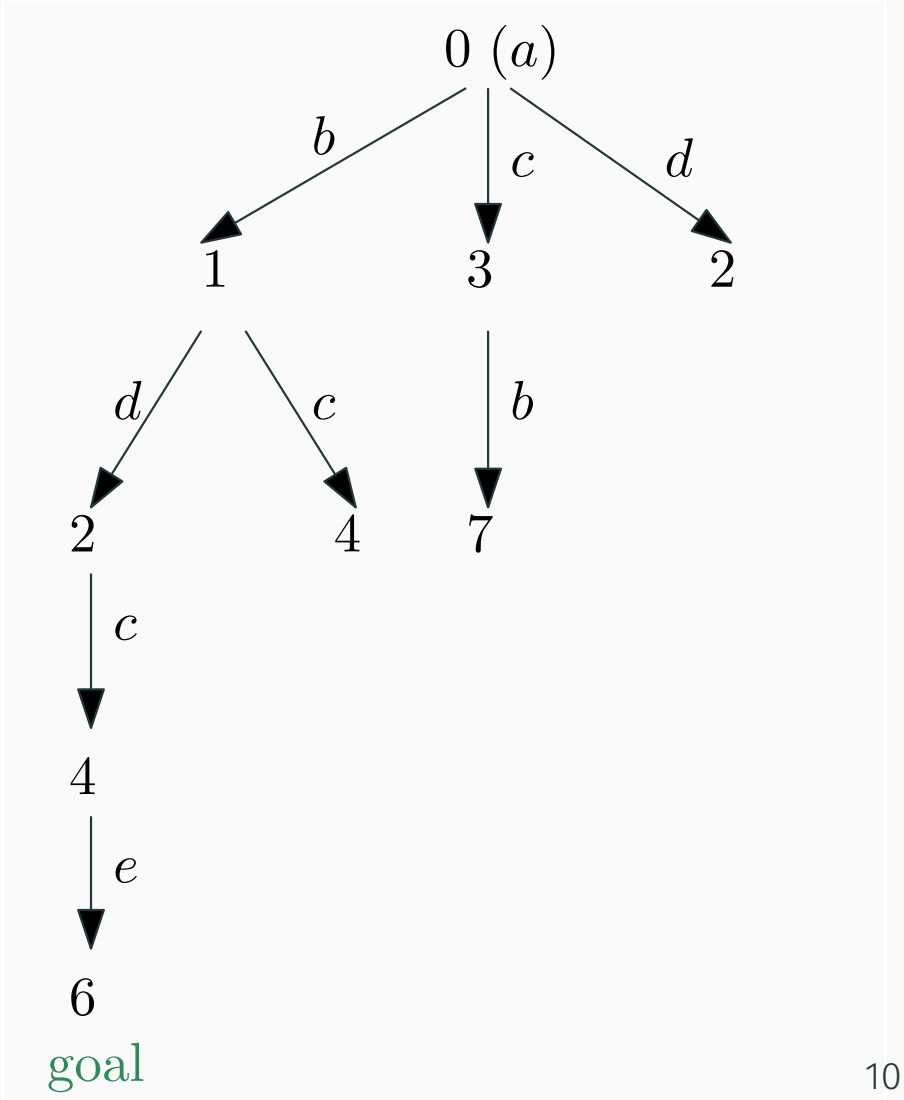




Example: Dedicated Branch and Bound

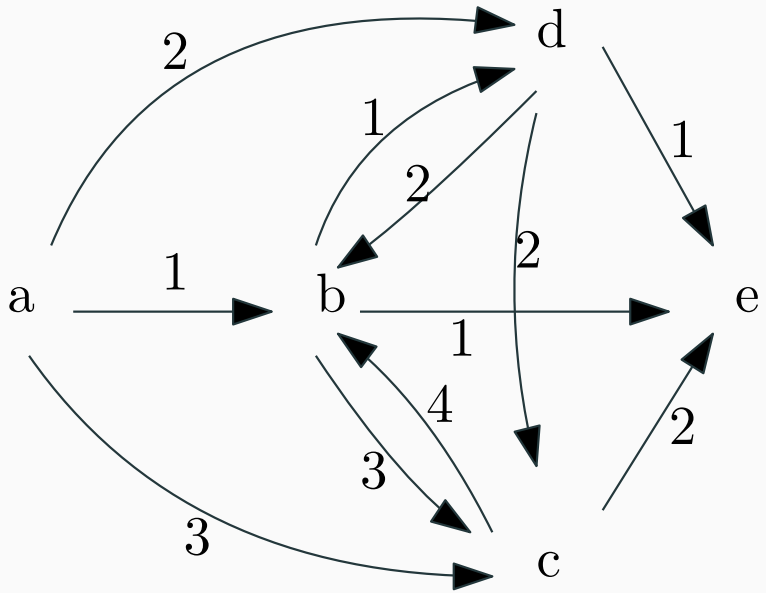
A **Sequential Ordering Problem** dedicated Branch and Bound

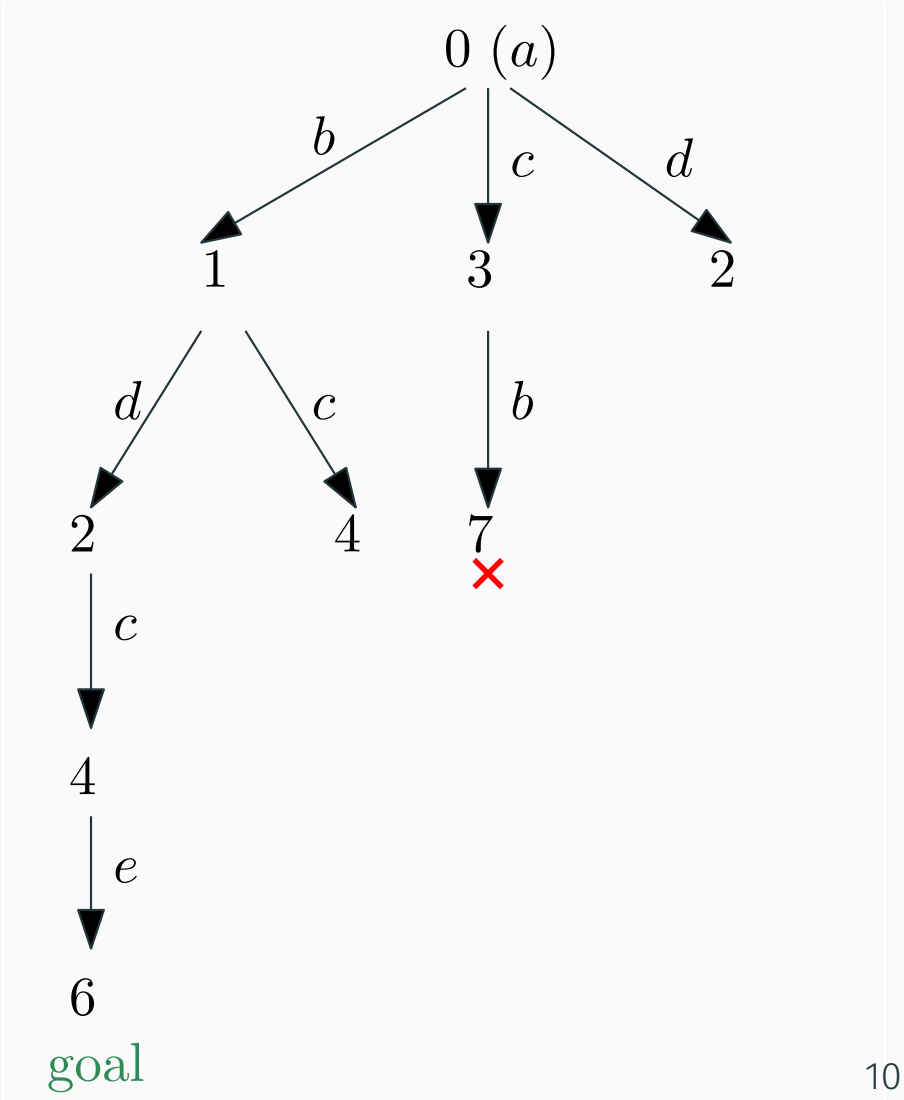




Example: Dedicated Branch and Bound

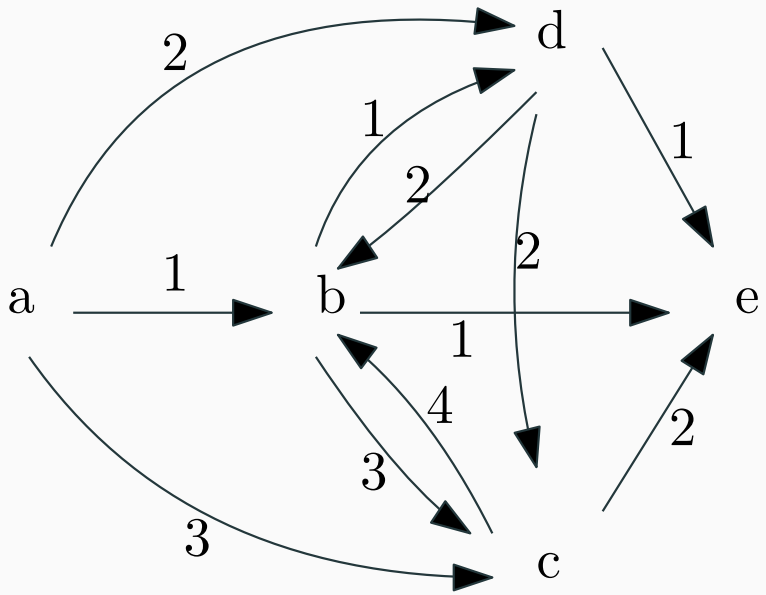
A **Sequential Ordering Problem** dedicated Branch and Bound

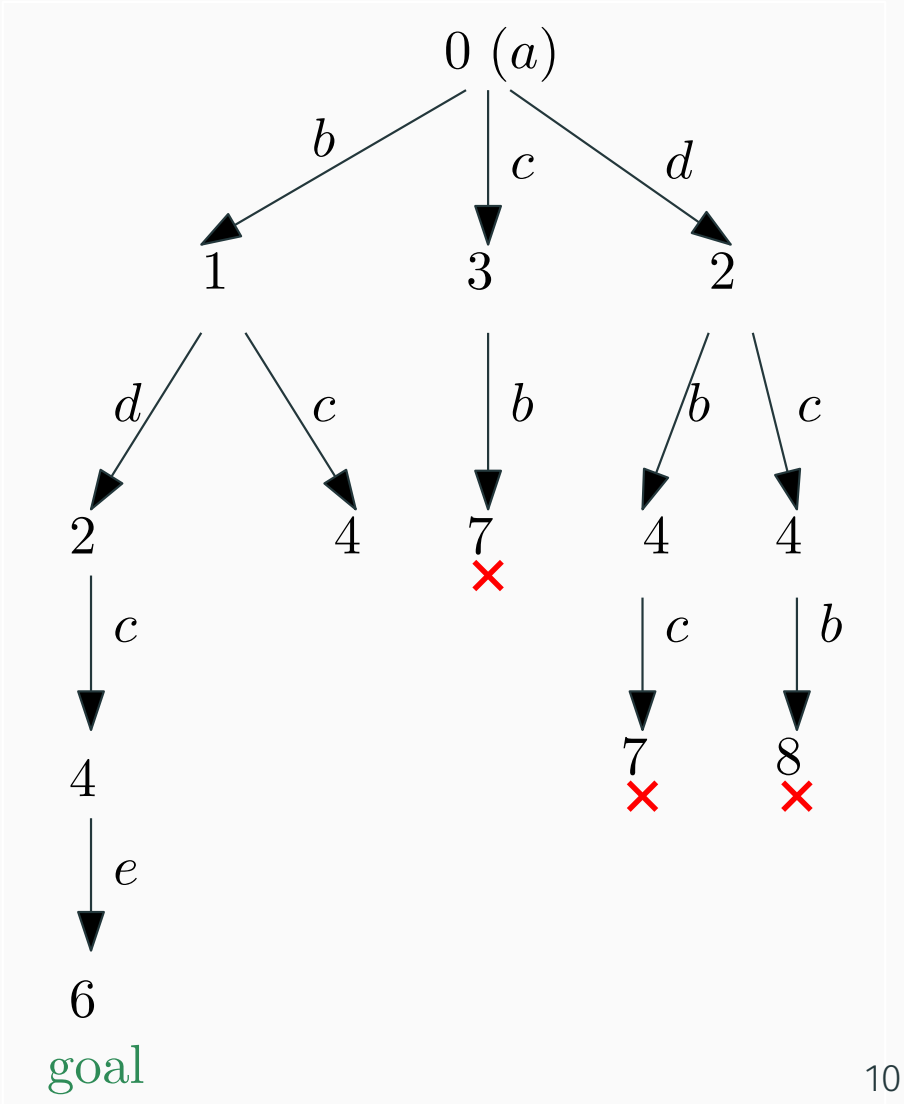




Example: Dedicated Branch and Bound

A **Sequential Ordering Problem** dedicated Branch and Bound

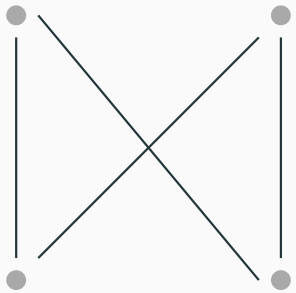




Local Search

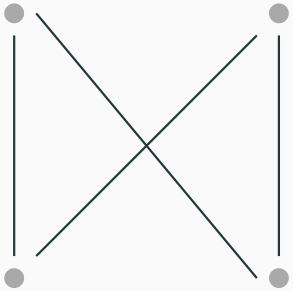
- usually improves an existing solution
- Models the problem as a graph
- Explores this graph (through a neighbourhood structure)

Local Search Example - 2-opt for the TSP

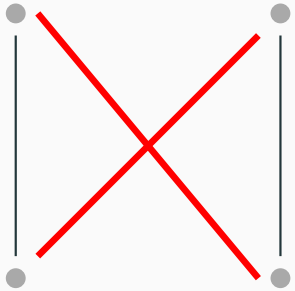


Initial solution (tour)

Local Search Example - 2-opt for the TSP

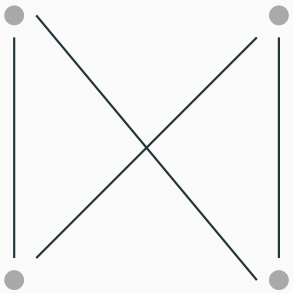


Initial solution (tour)

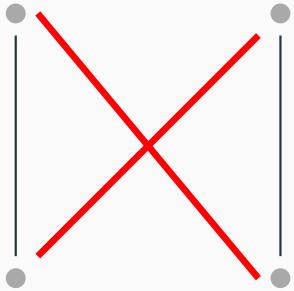


Identify edges to remove (in red)

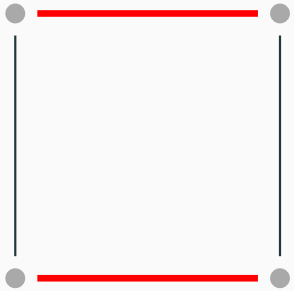
Local Search Example - 2-opt for the TSP



Initial solution (tour)



Identify edges to remove (in red)



Replace them

Population Based Search

- Consider a set of solutions (population)
- Combines promising solutions together (crossover)
- Possibly alter solutions (mutations)

Recap

	Operators	Examples
Tree Search	children, bounds	MIP, CP, (more later ...)
Local Search	neighbourhood	Tabu Search, SA ...
Population Based	crossover, mutation distance from a solution	Genetic/Evolutionary

Tree Search

Paradigm

Made of two parts:

- The Implicit Tree definition:
  - root
  - children
  - bounds (optimistic estimate)
  - isGoal
  - possibly other information (*i.e.* guides, cuts ...)

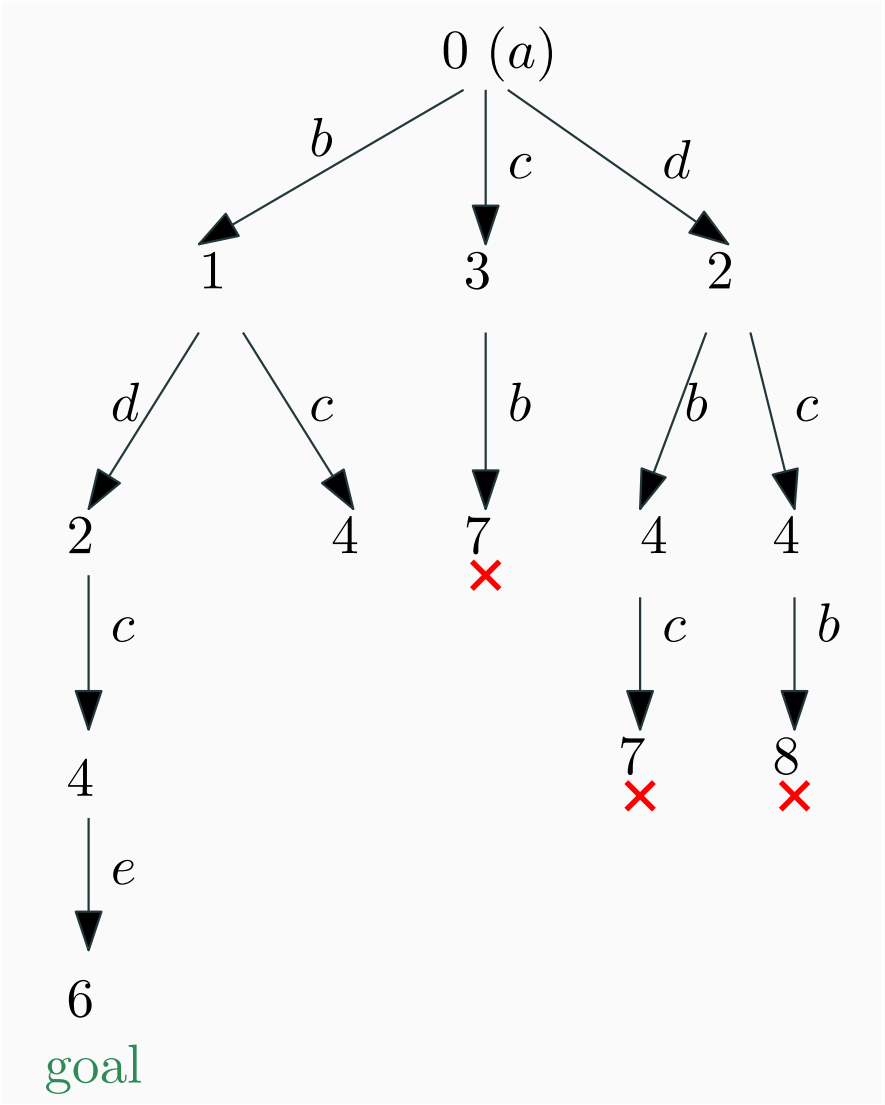
Paradigm

Made of two parts:

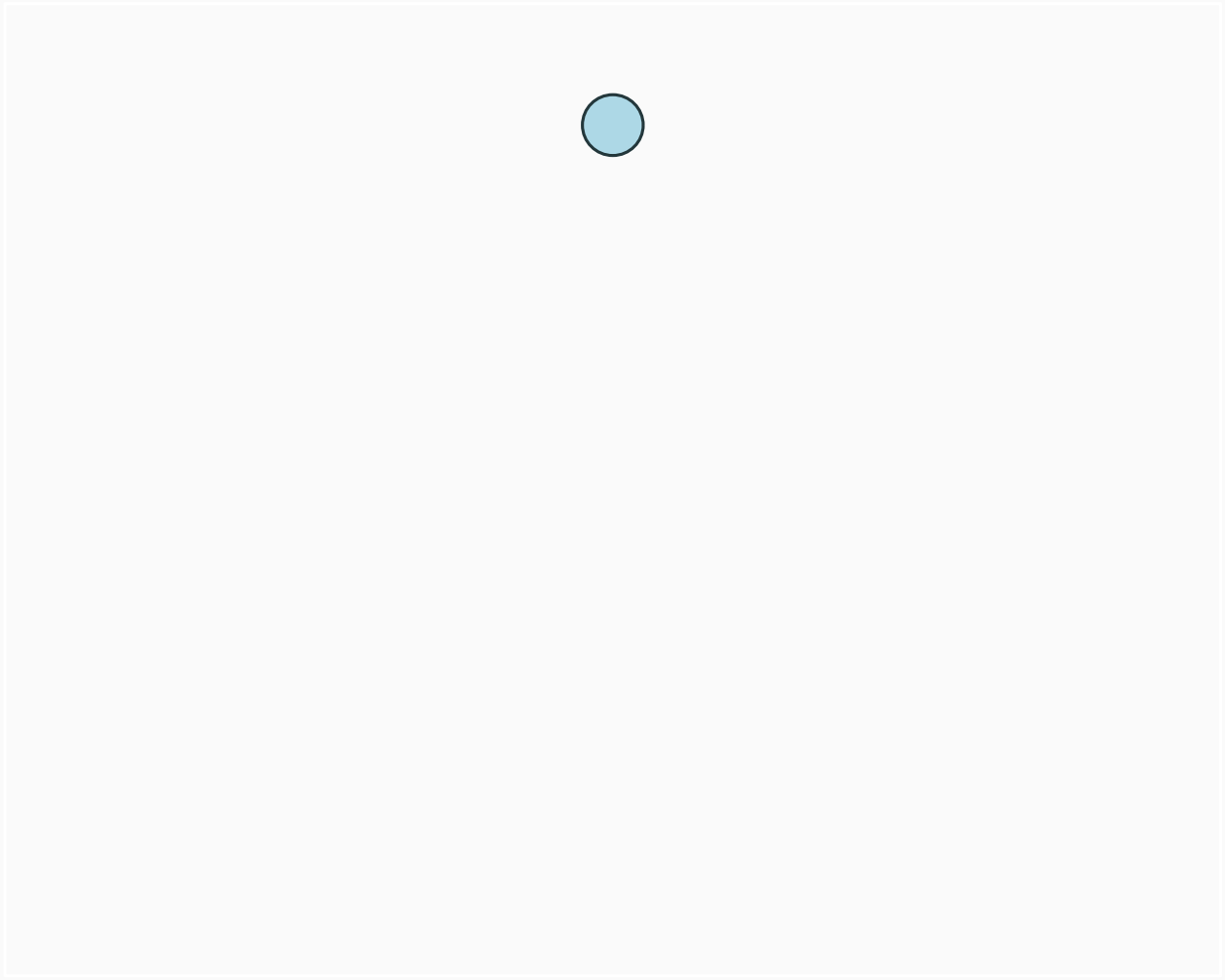
- The Implicit Tree definition:
  - root
  - children
  - bounds (optimistic estimate)
  - isGoal
  - possibly other information (*i.e.* guides, cuts ...)
- The Search Procedure (generic):
  - Depth First Search (DFS)
  - Best First Search
  - Others (discussed in a few slides)

In the next slides, we suppose an existing Implicit Tree definition and study generic search algorithms.

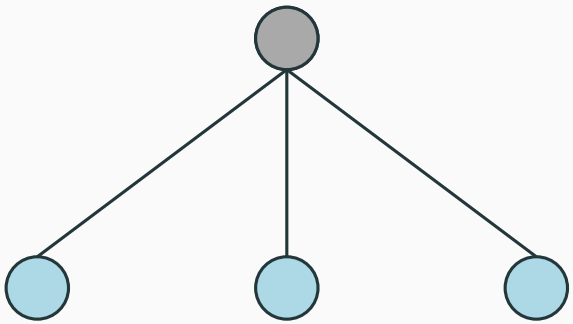
About the Tree Search Formalism: An example



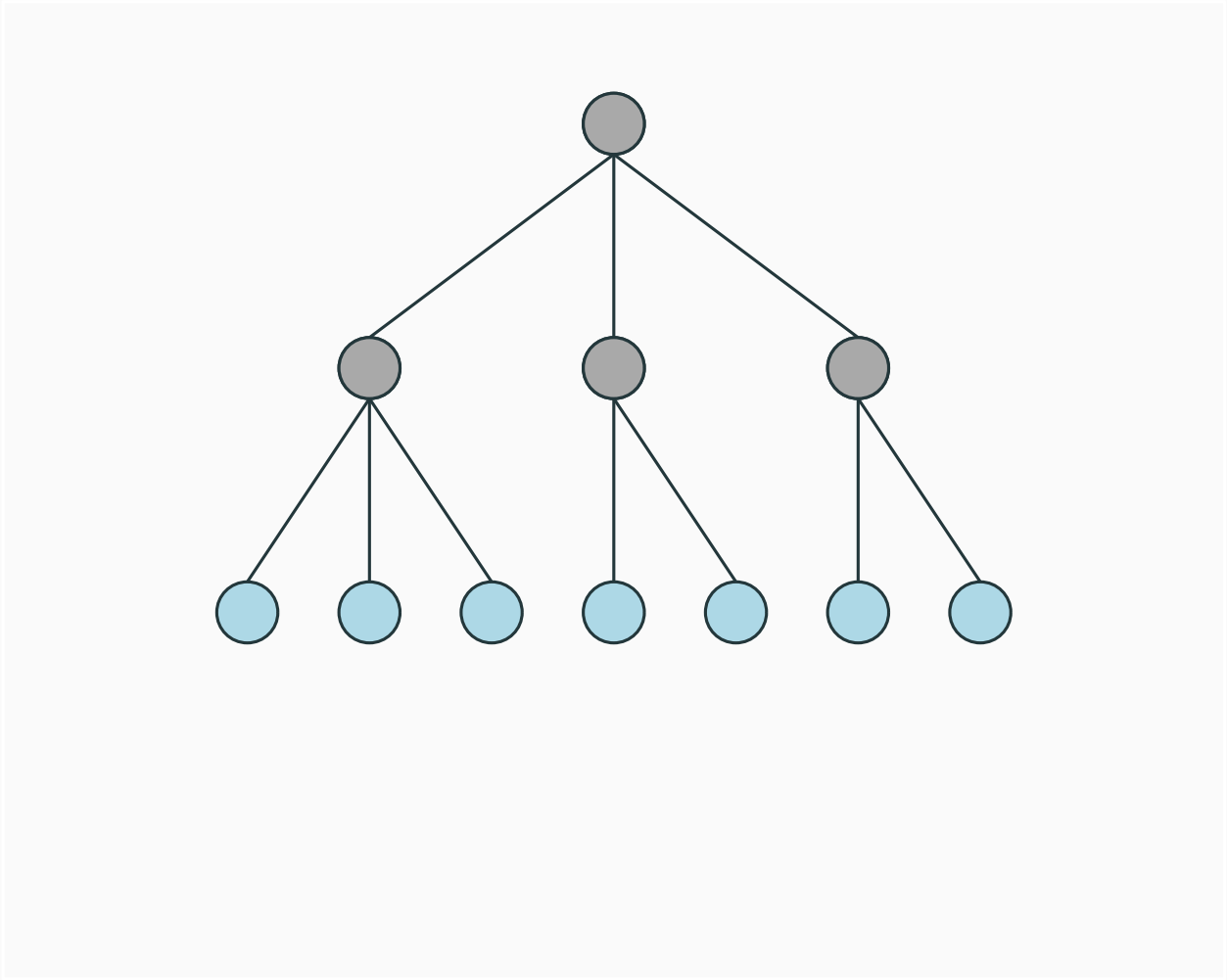
Breadth First Search



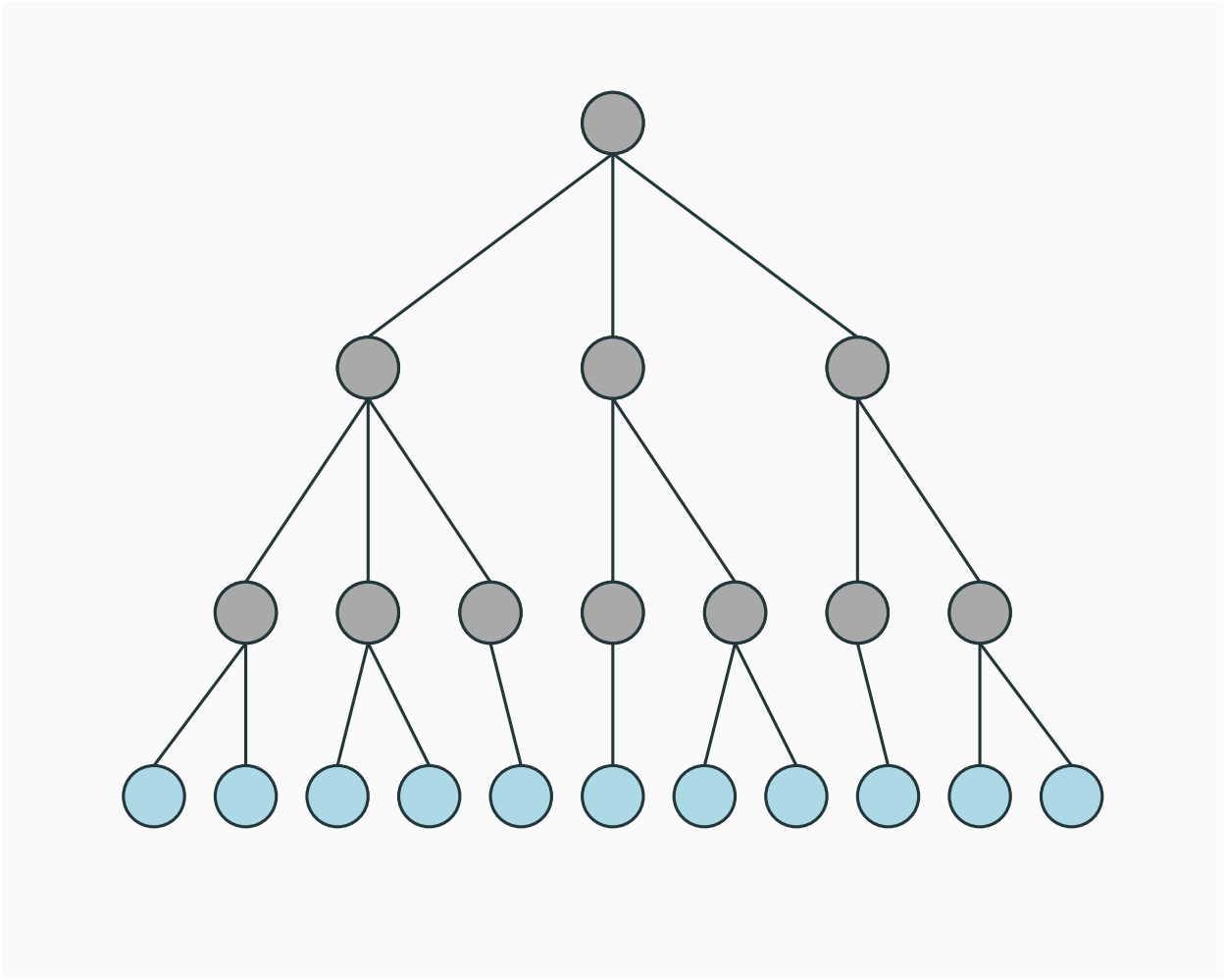
Breadth First Search



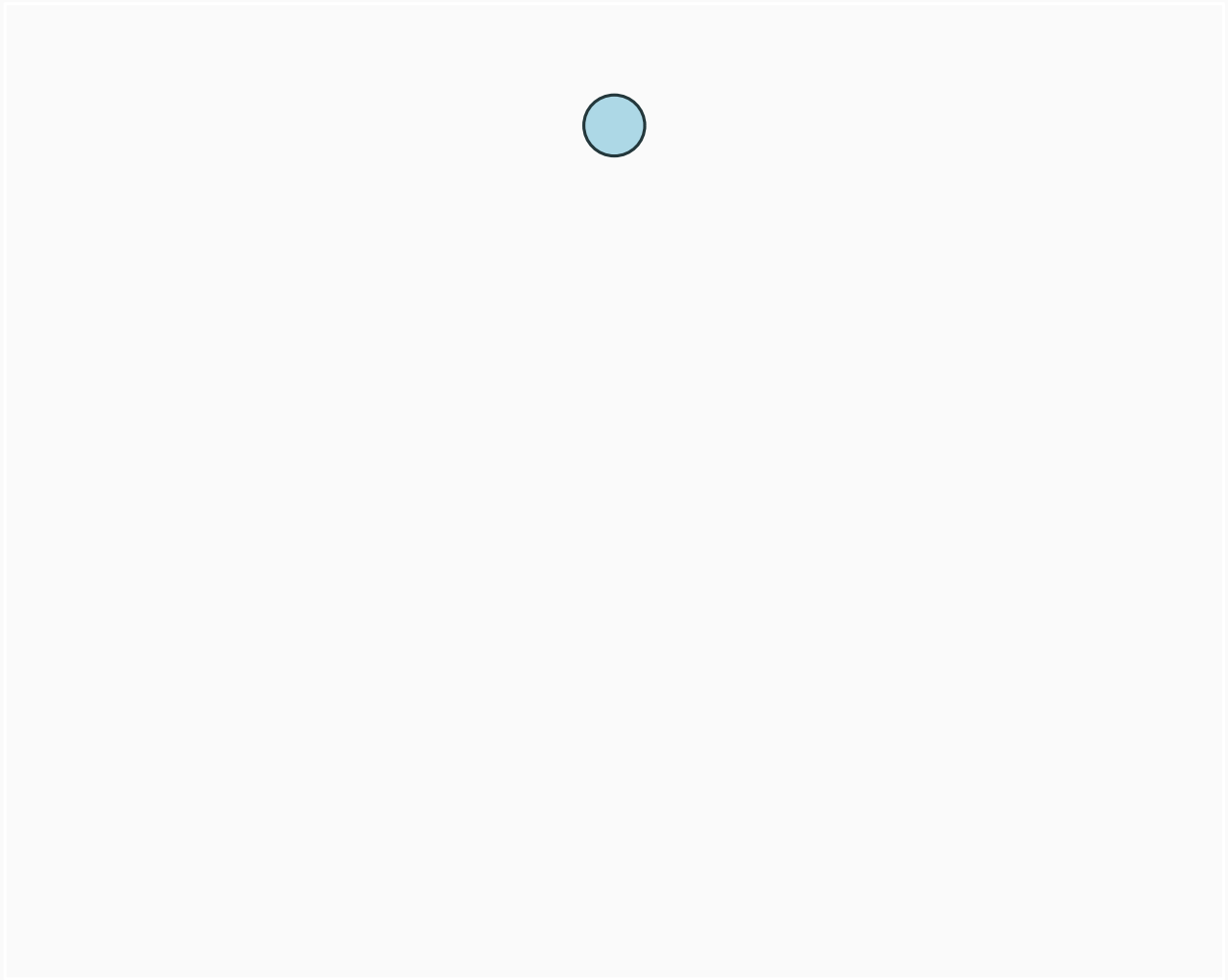
Breadth First Search



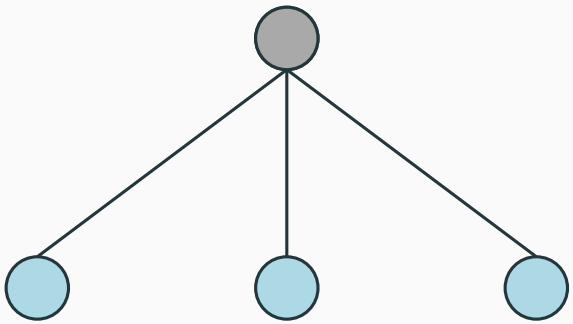
Breadth First Search



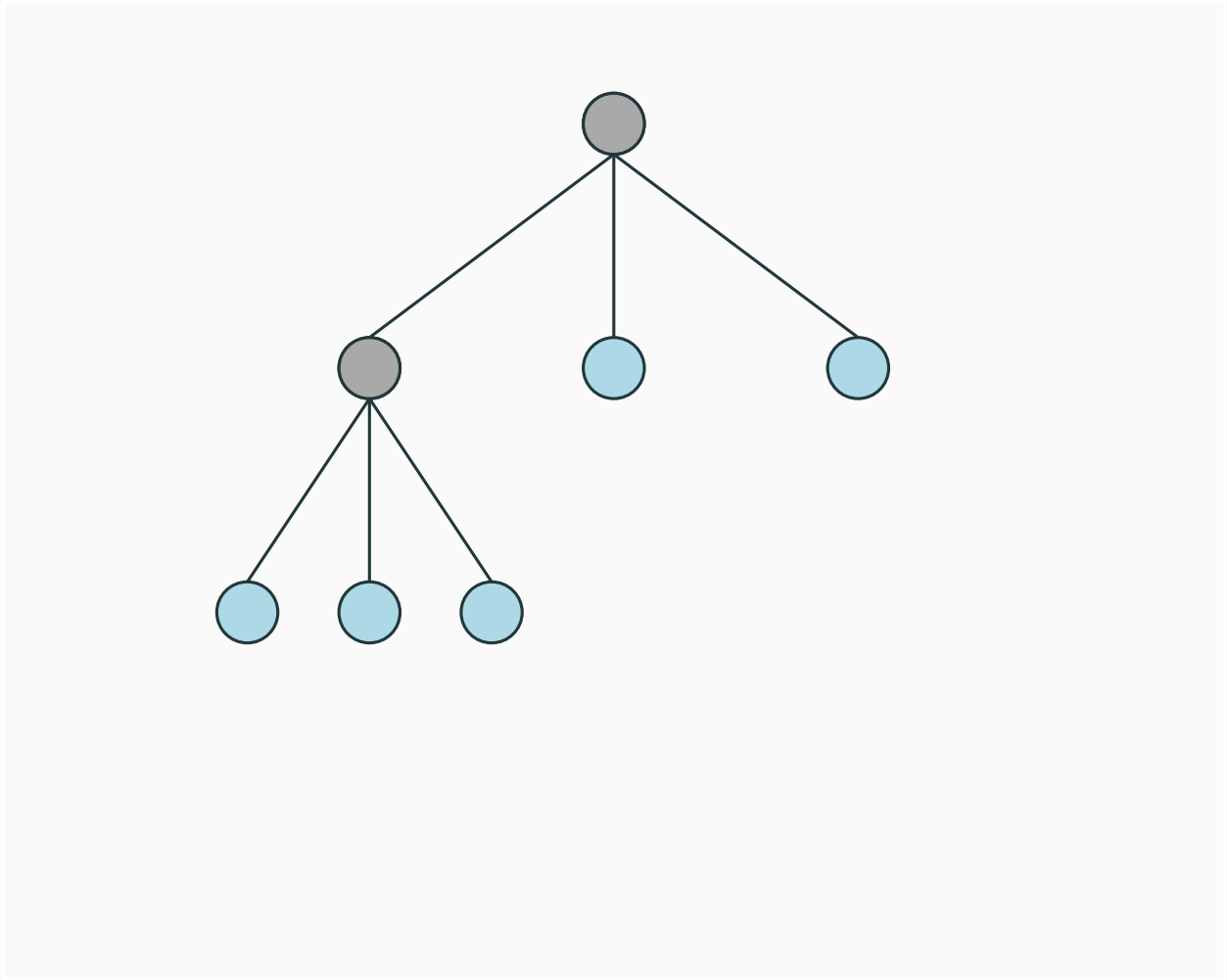
Depth First Search



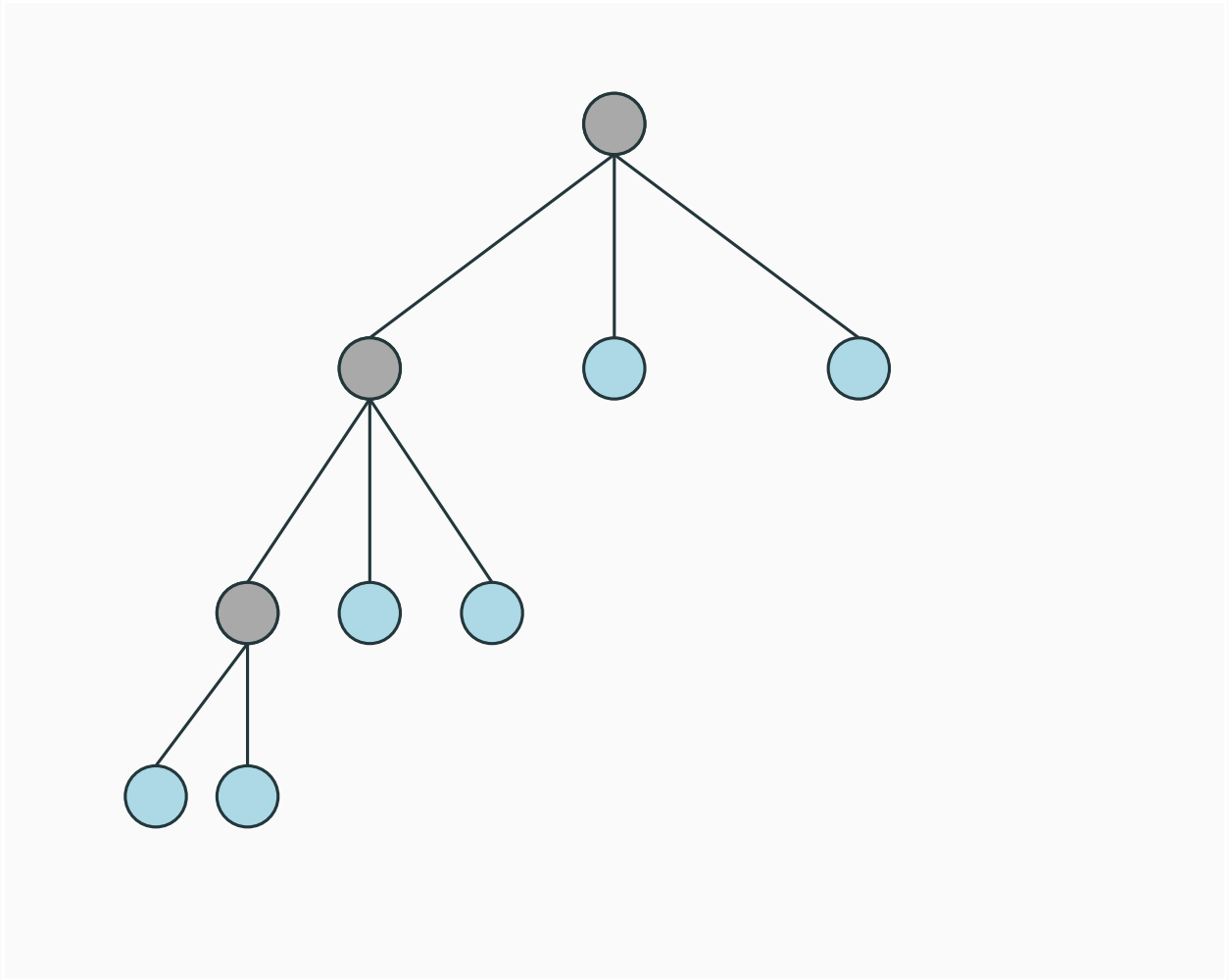
Depth First Search



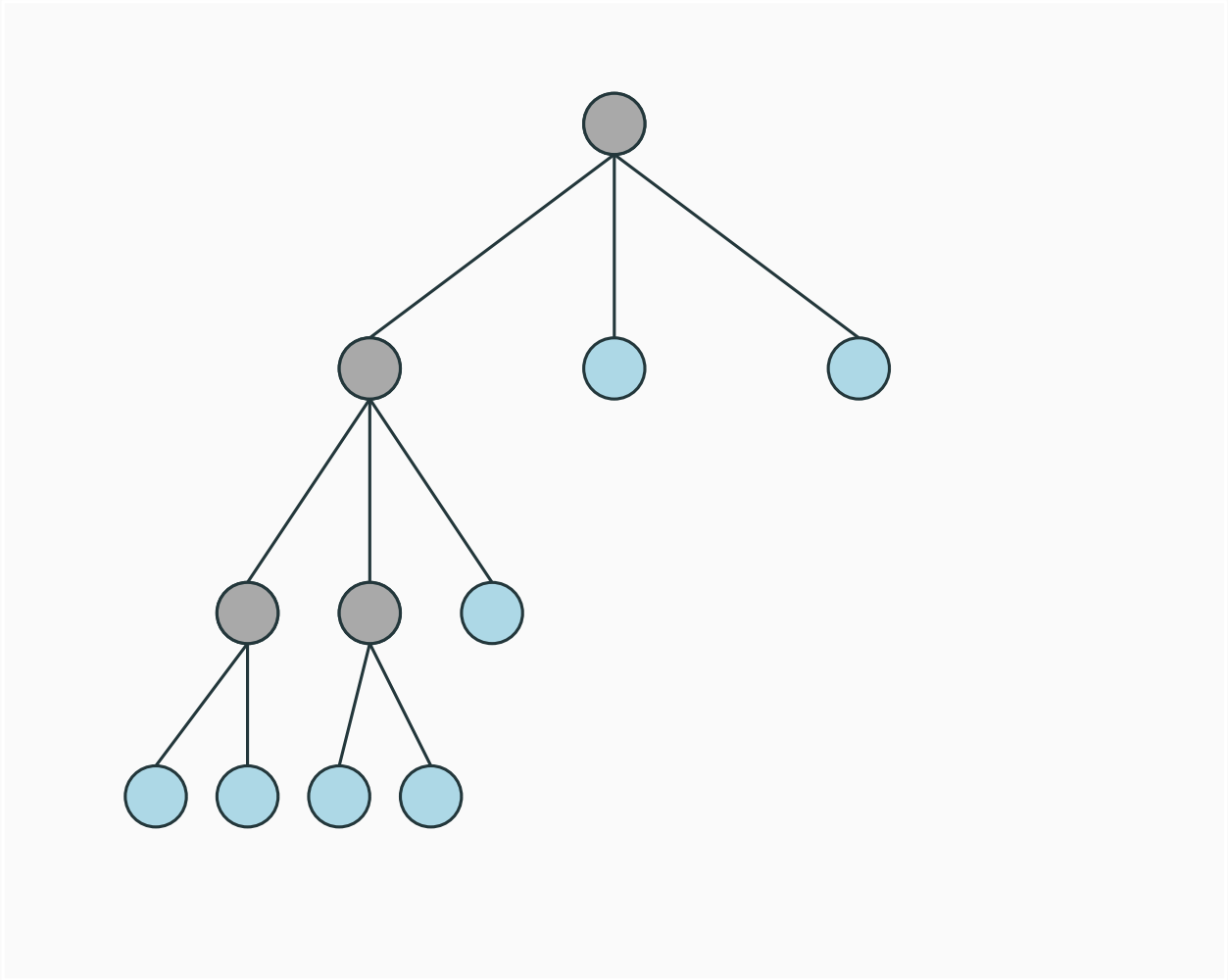
Depth First Search



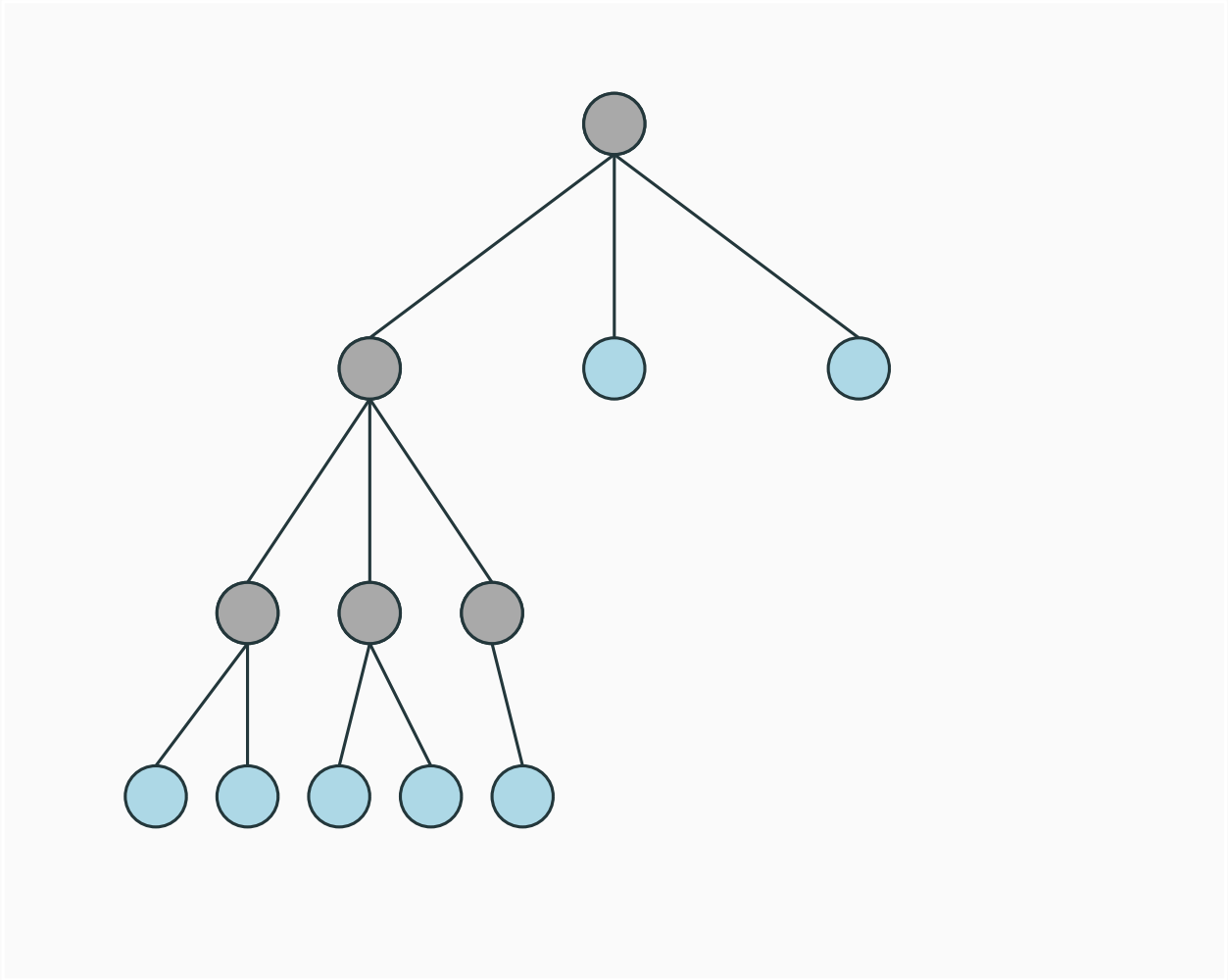
Depth First Search



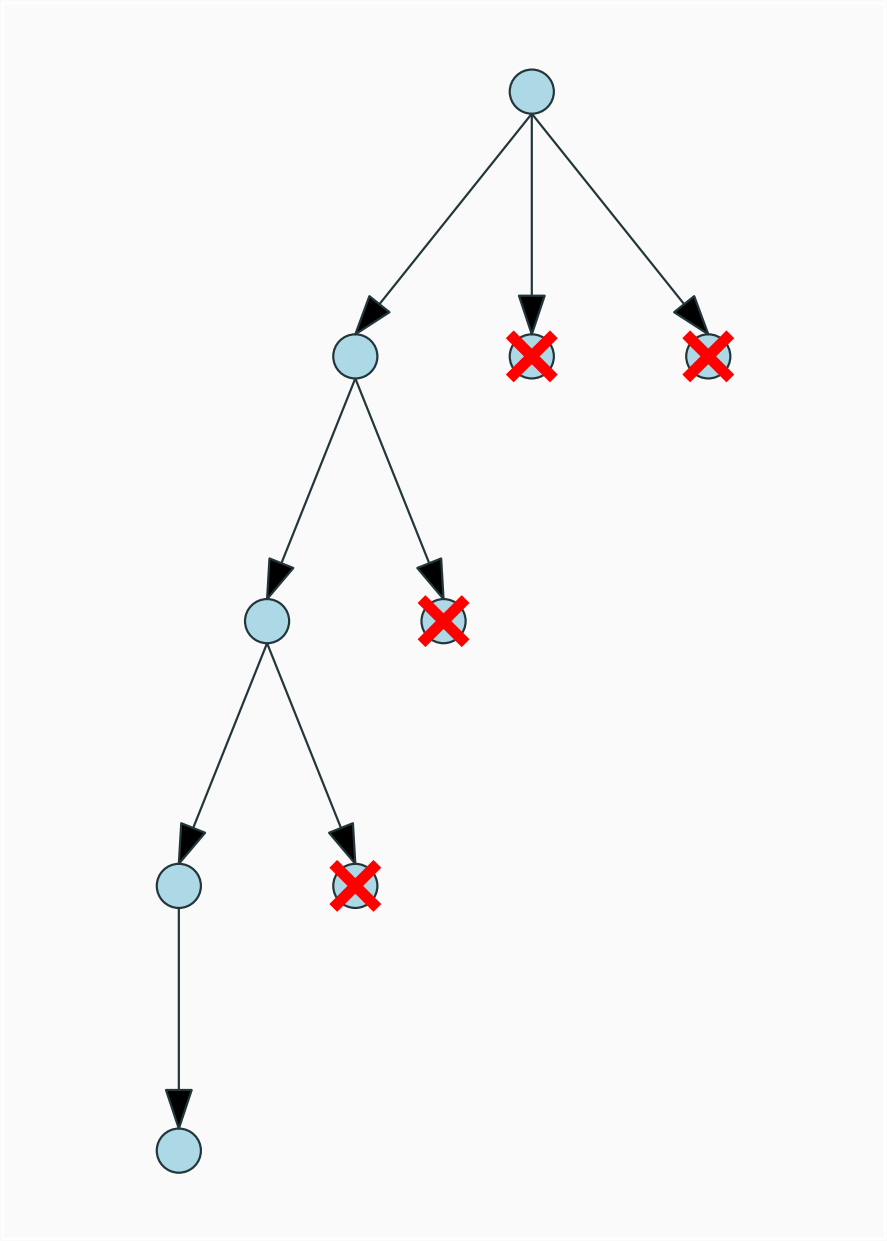
Depth First Search



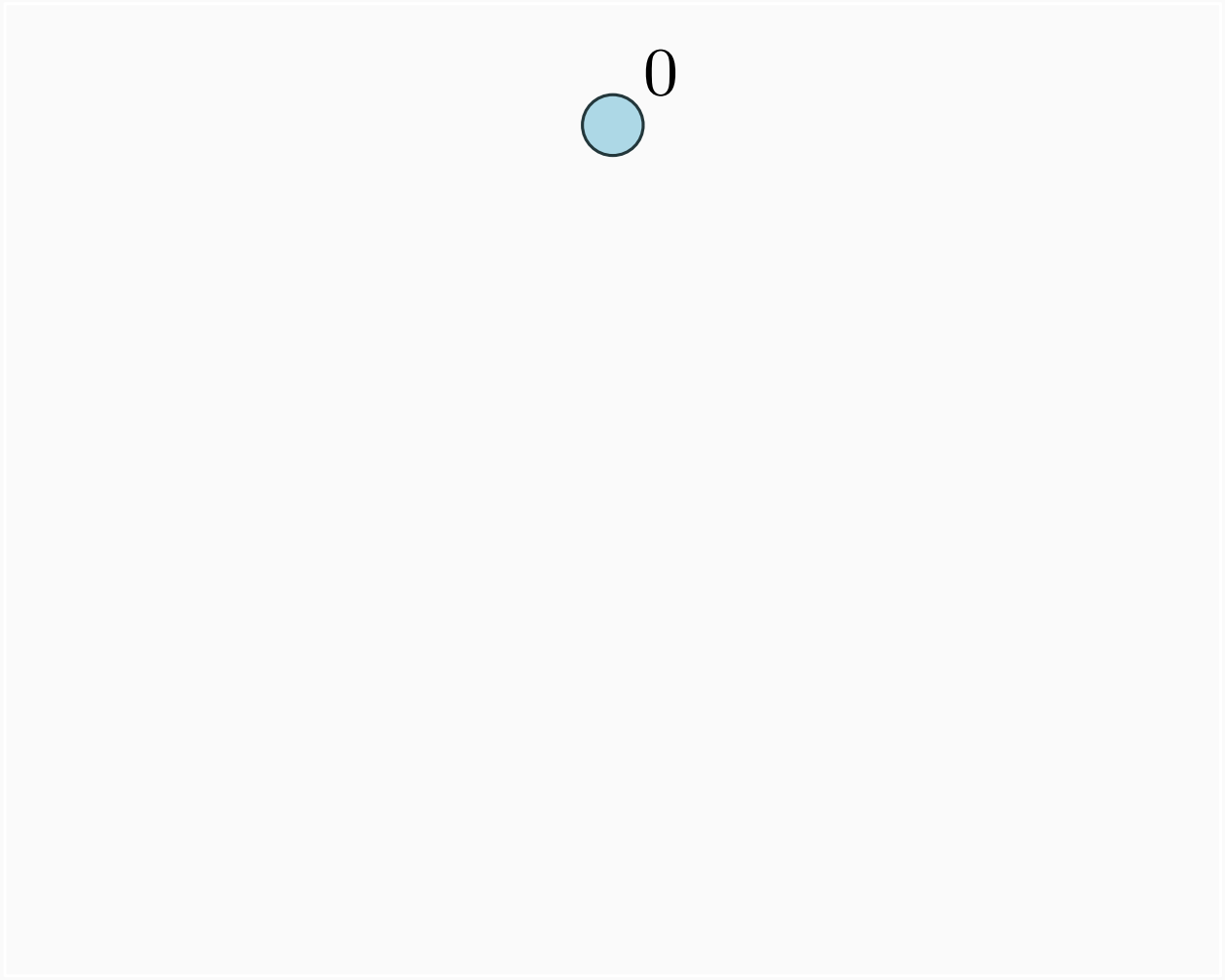
Depth First Search



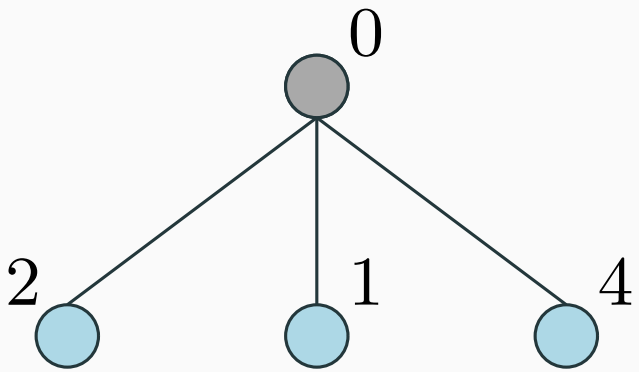
Greedy



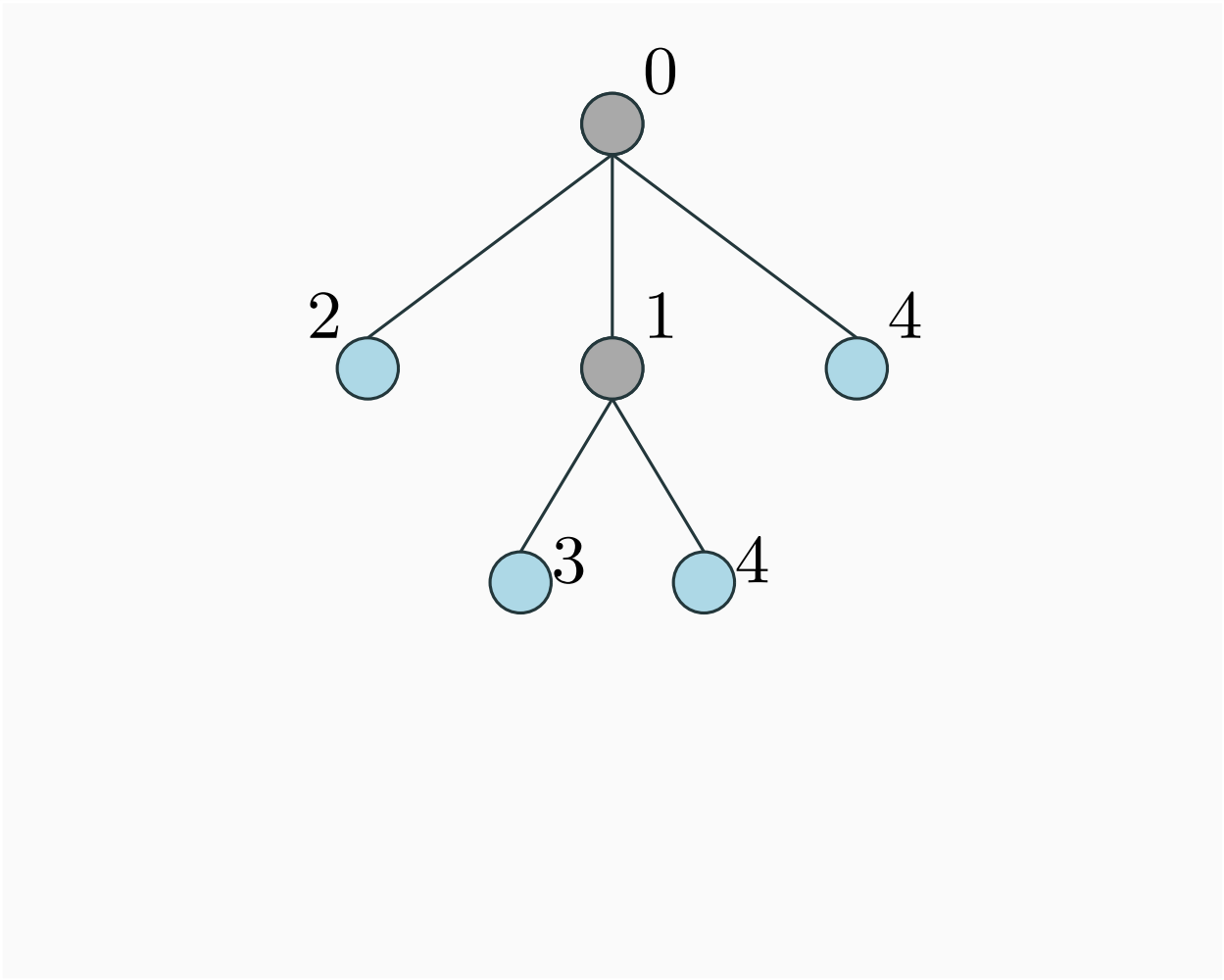
Best (bound) First / A*



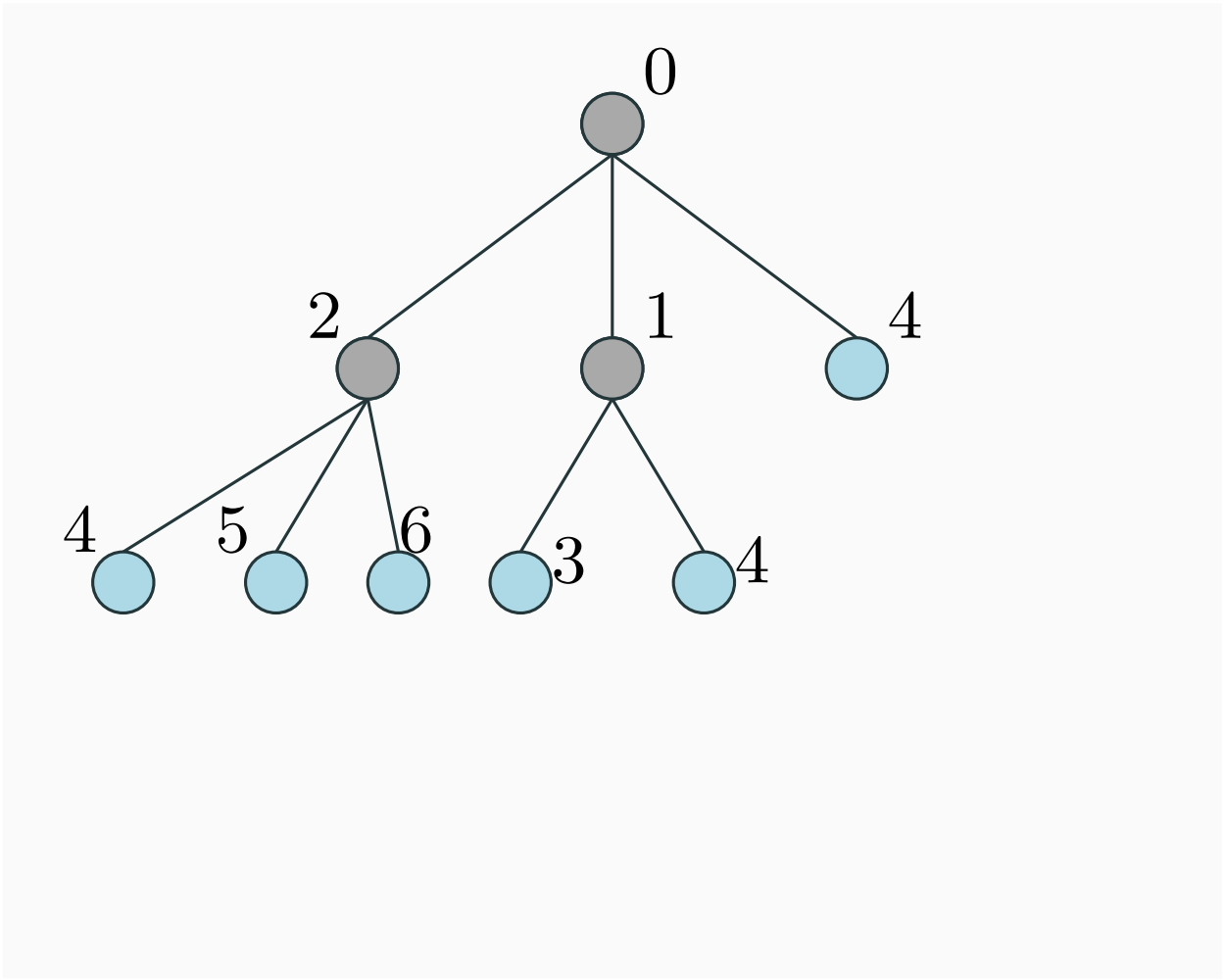
Best (bound) First / A*



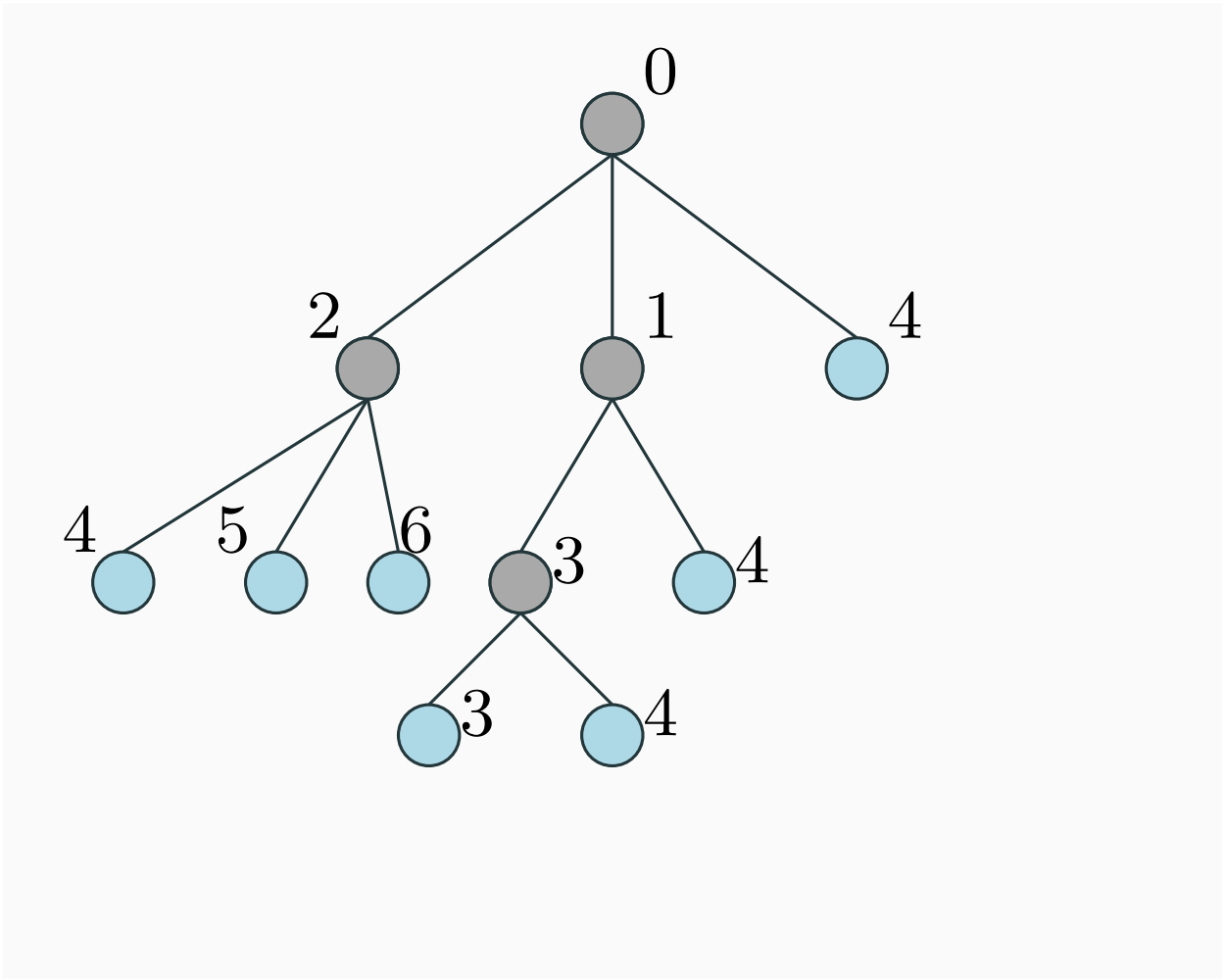
Best (bound) First / A*



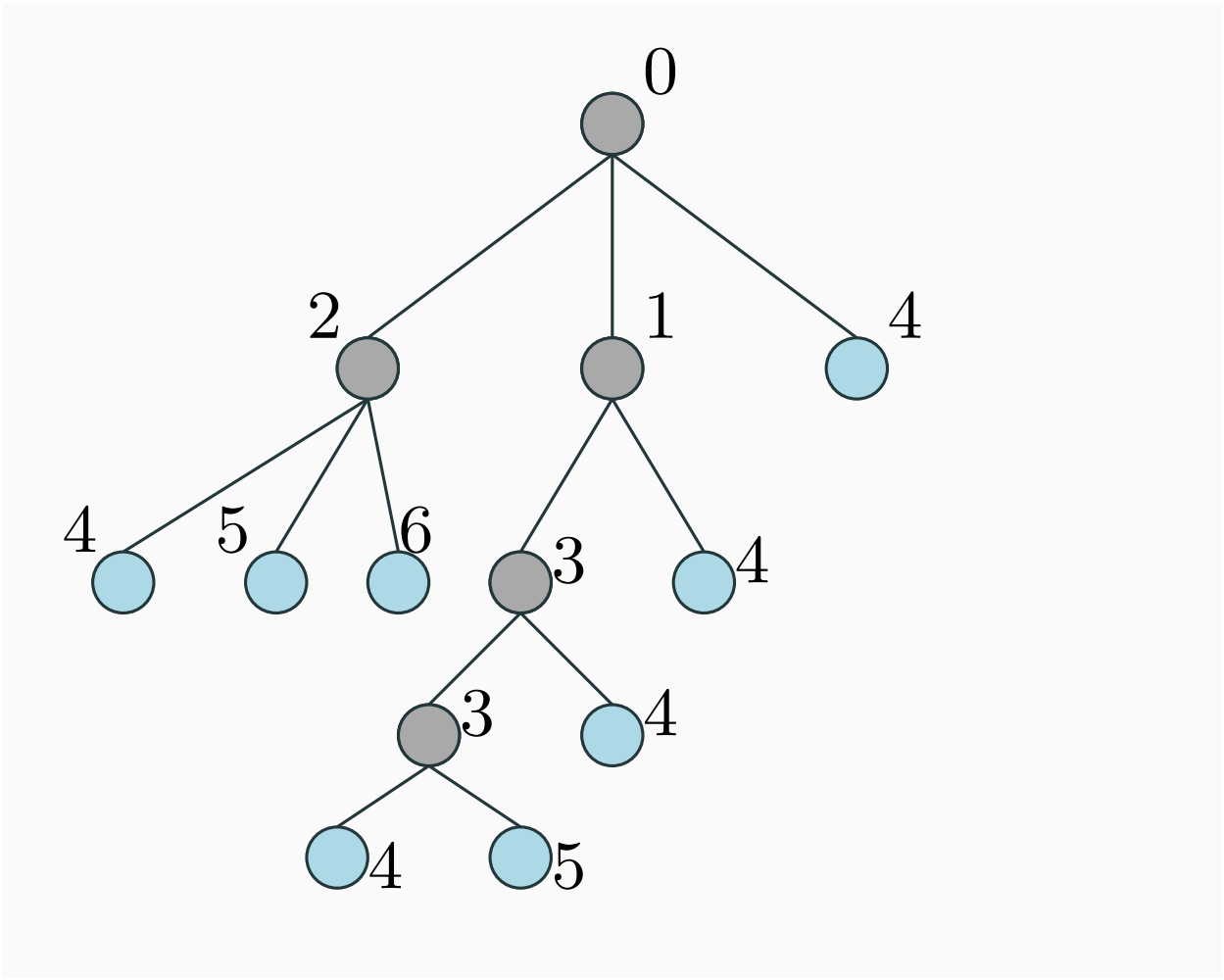
Best (bound) First / A*



Best (bound) First / A*



Best (bound) First / A*



Exercise 1 : Advantages and Drawbacks

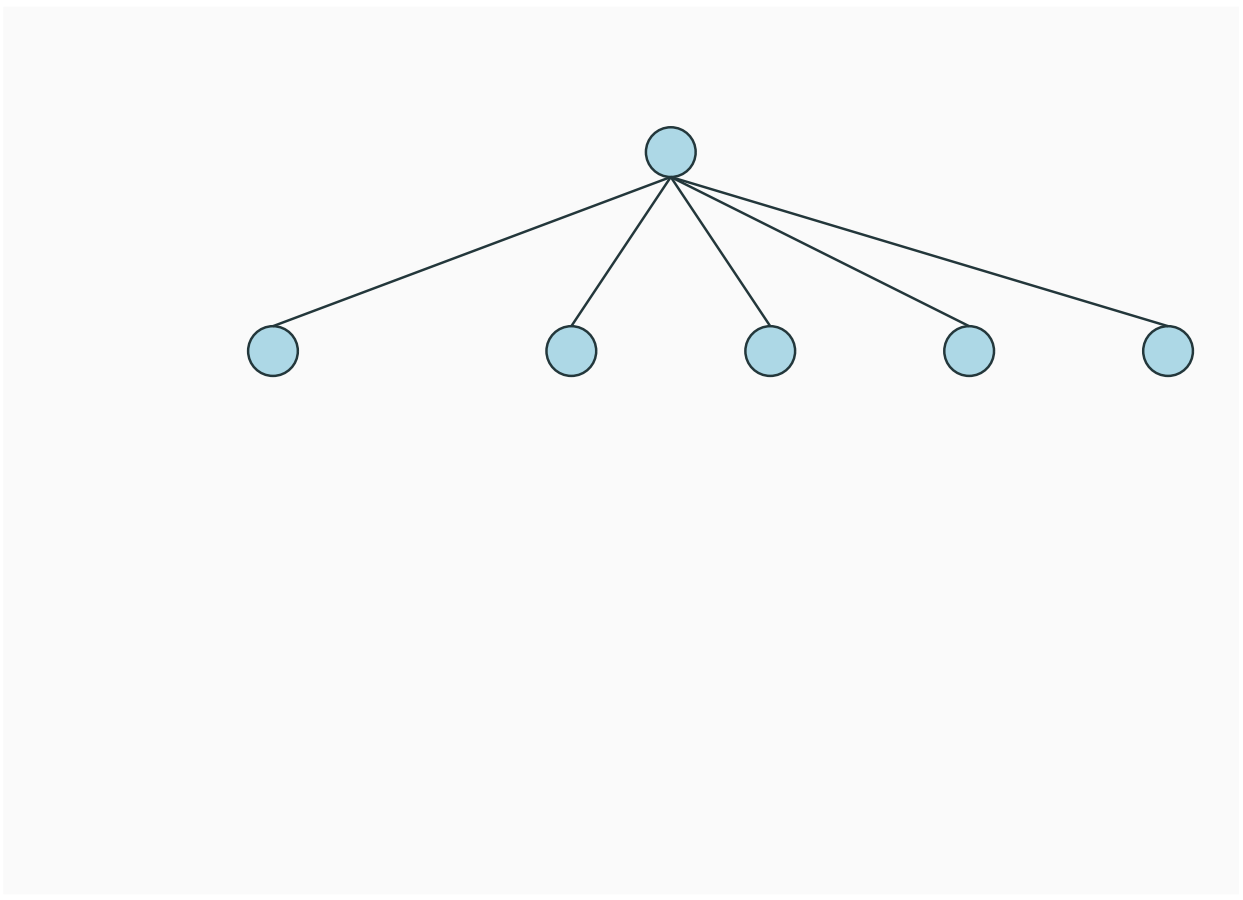
	Depth First Search	A*/Best First

Exercise 1 : Advantages and Drawbacks

	Depth First Search	A*/Best First
Pros	1. Anytime 2. Memory Bounded	1. less nodes to close the instance 2. no need of good solutions
Cons	1. requires good solutions 2. suffers from early bad decisions	1. not anytime 2. Can use too much memory

Limited Discrepancy Search (LDS) - key idea

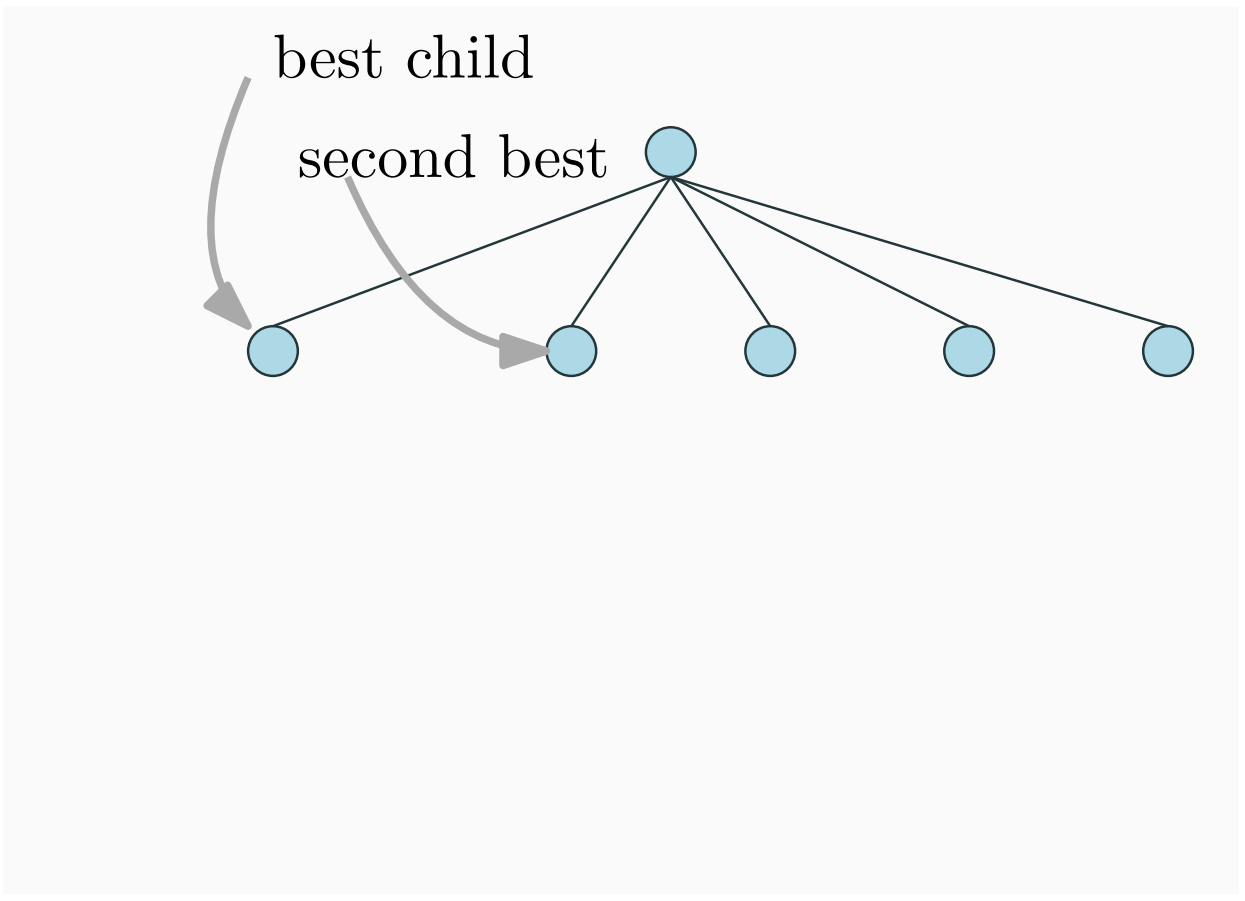
Correct DFS drawback: early bad decisions



Explore more the most promising but still keep exploring others

Limited Discrepancy Search (LDS) - key idea

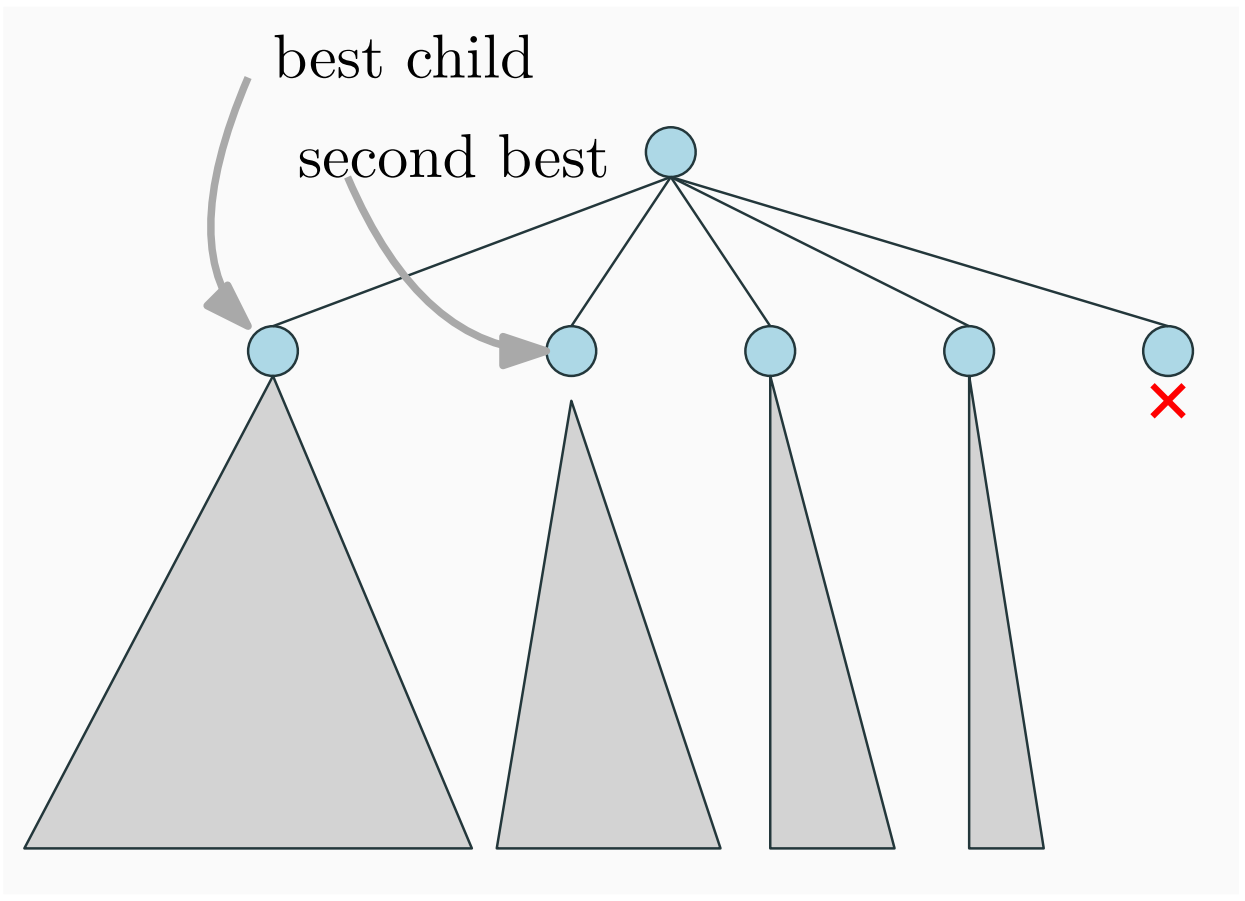
Correct DFS drawback: early bad decisions



Explore more the most promising but still keep exploring others

Limited Discrepancy Search (LDS) - key idea

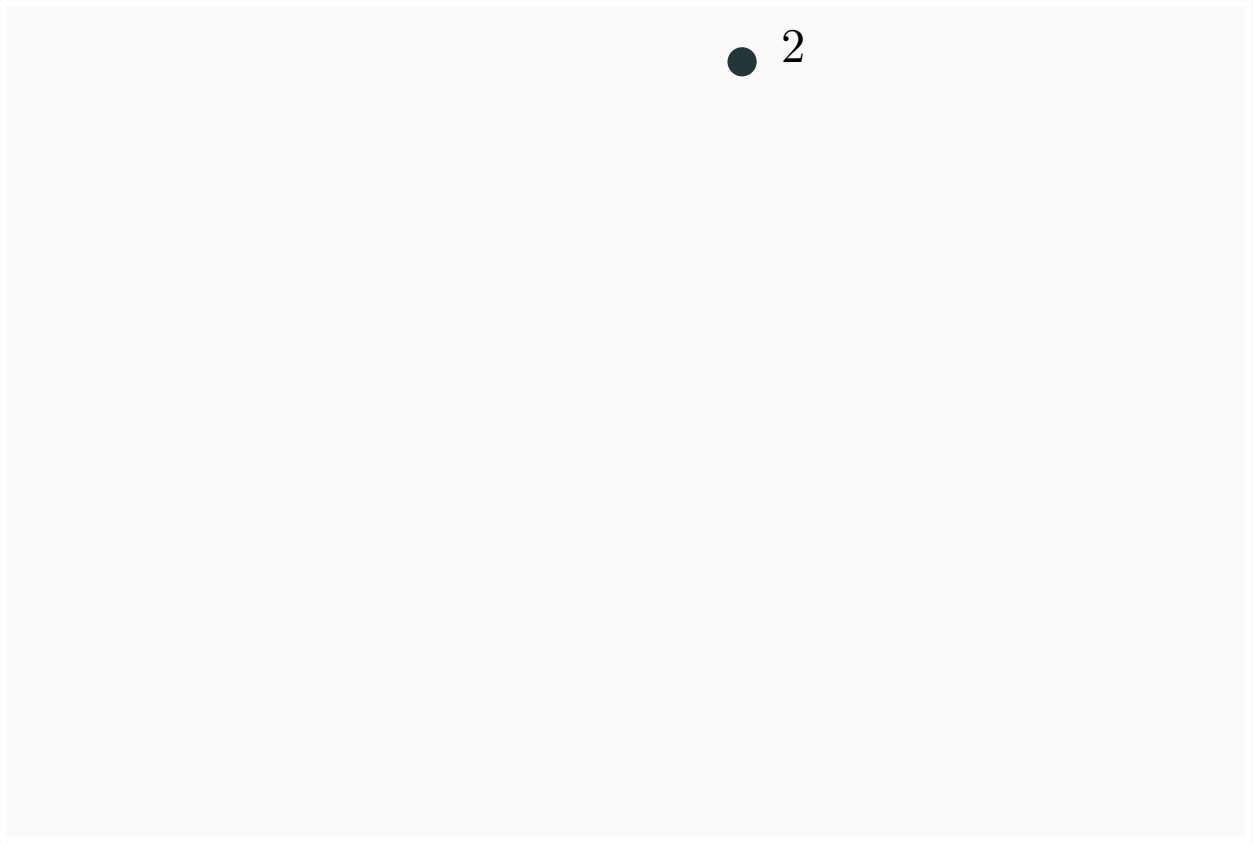
Correct DFS drawback: early bad decisions



Explore more the most promising but still keep exploring others

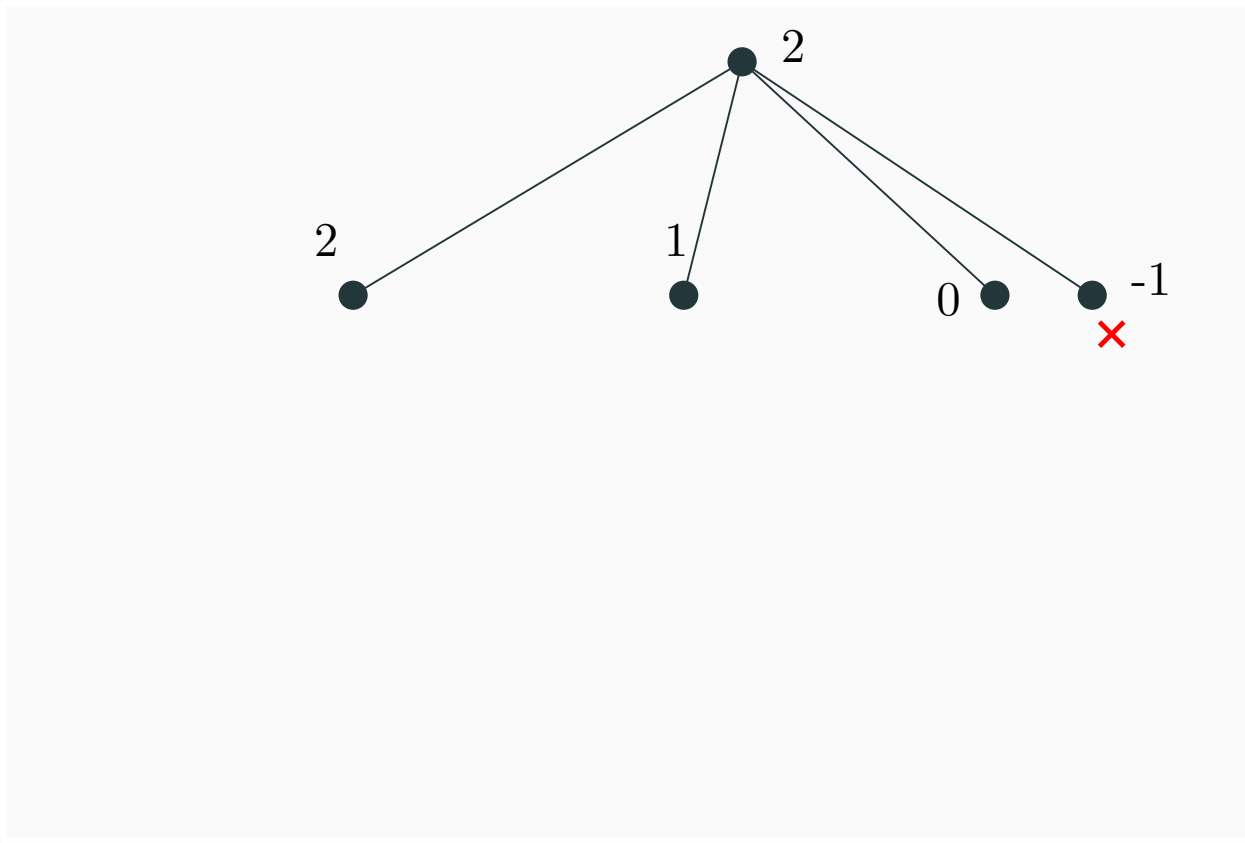
Limited Discrepancy Search (LDS) - An example: D=2

Count the number of deviations from the best child (discrepancies)



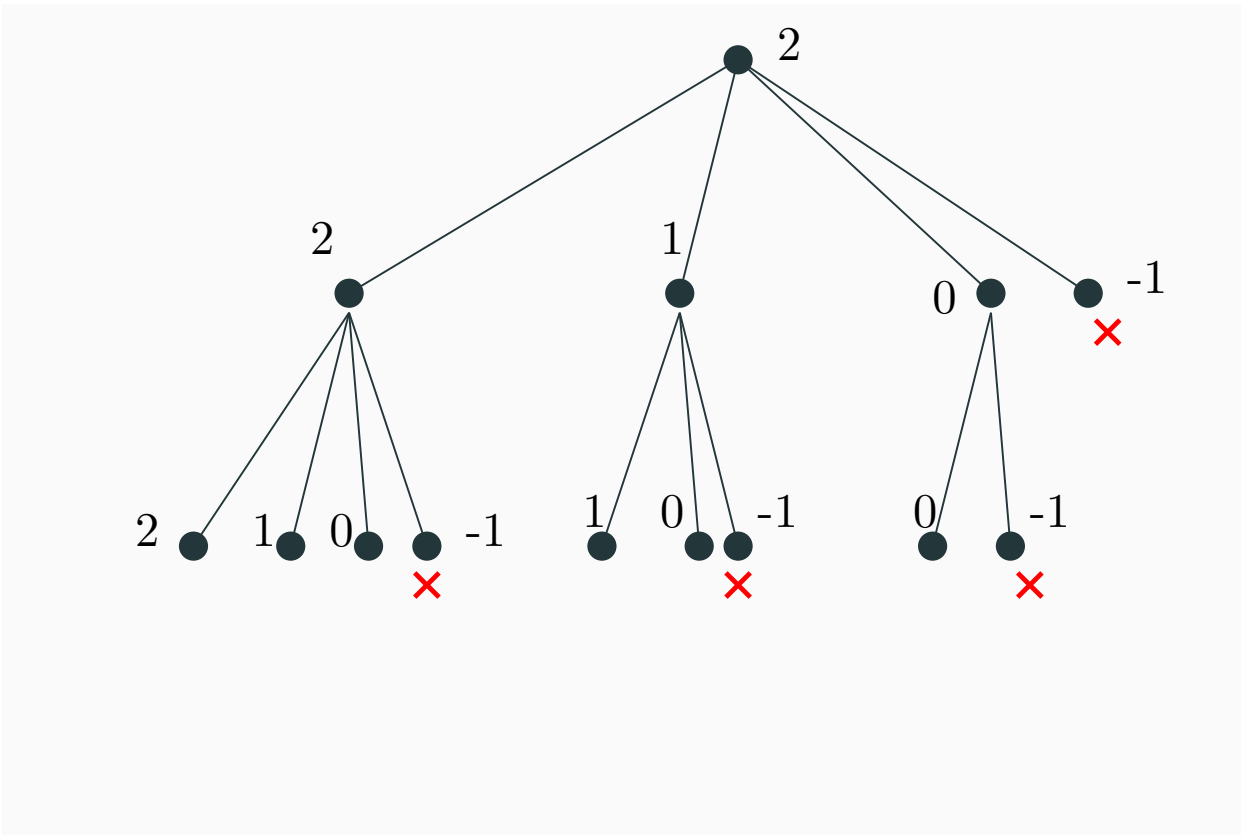
Limited Discrepancy Search (LDS) - An example: D=2

Count the number of deviations from the best child (discrepancies)



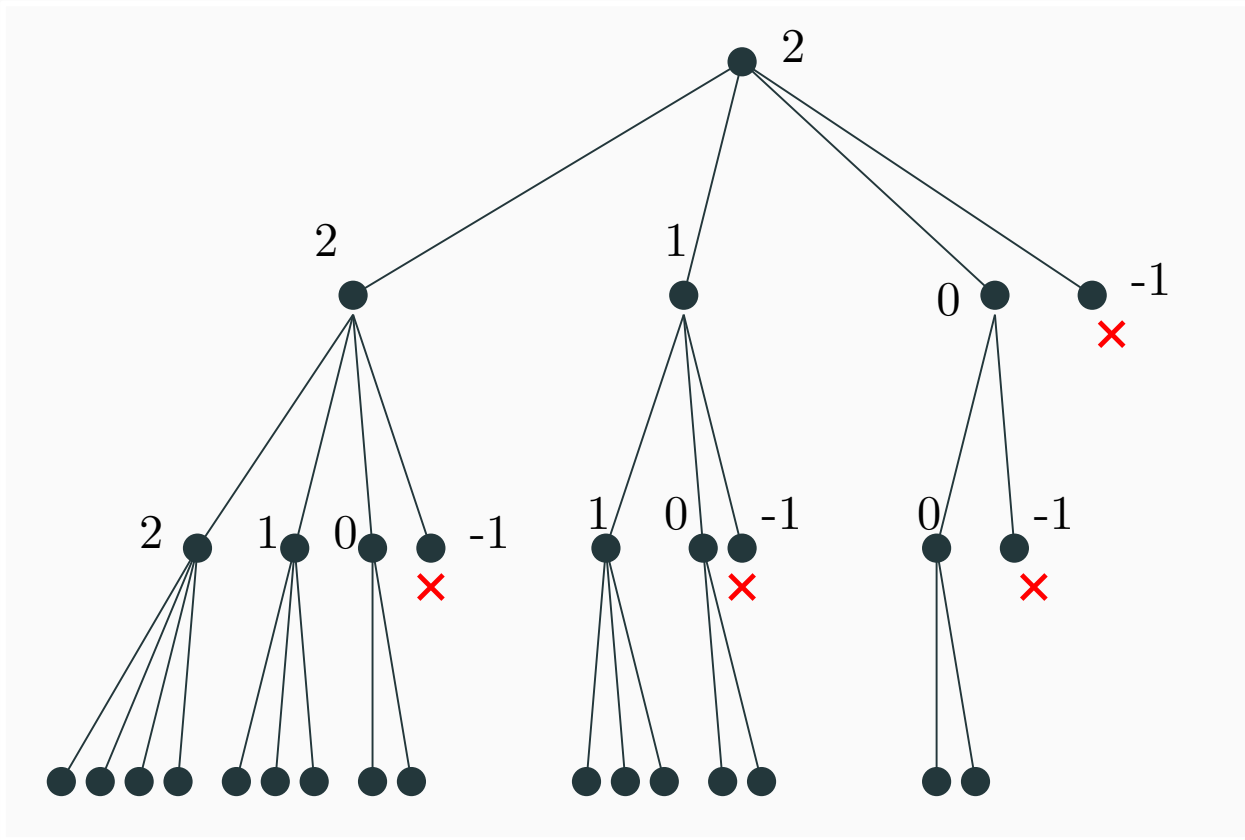
Limited Discrepancy Search (LDS) - An example: D=2

Count the number of deviations from the best child (discrepancies)

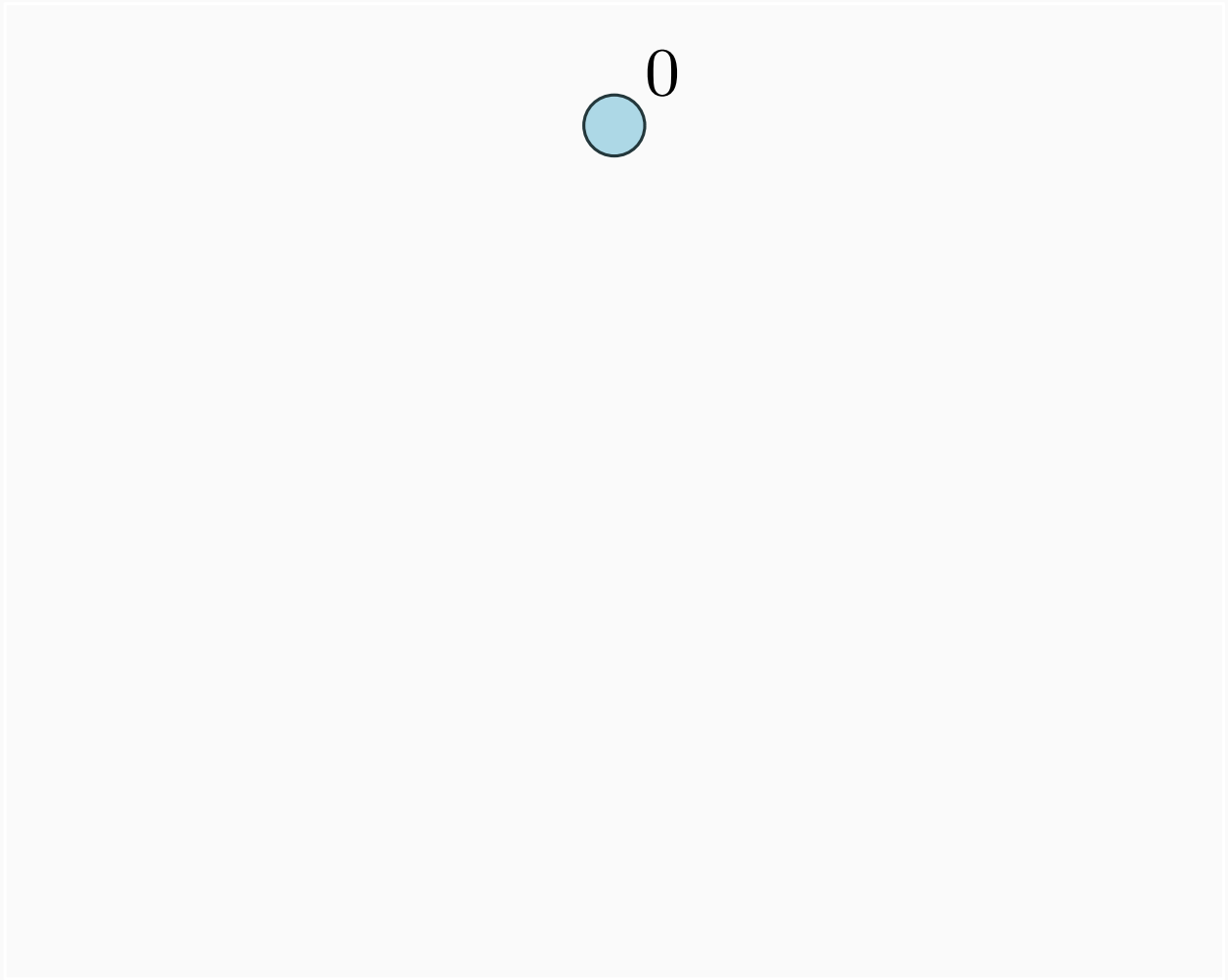


Limited Discrepancy Search (LDS) - An example: D=2

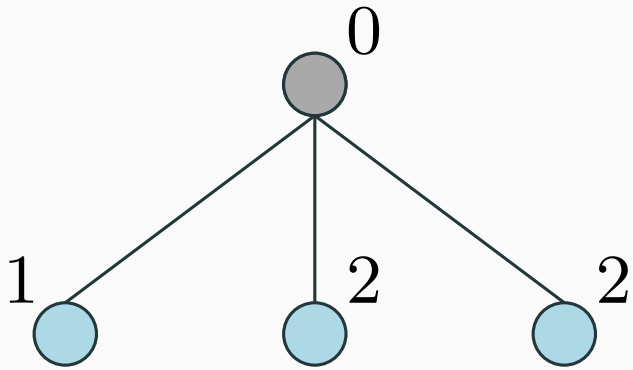
Count the number of deviations from the best child (discrepancies)



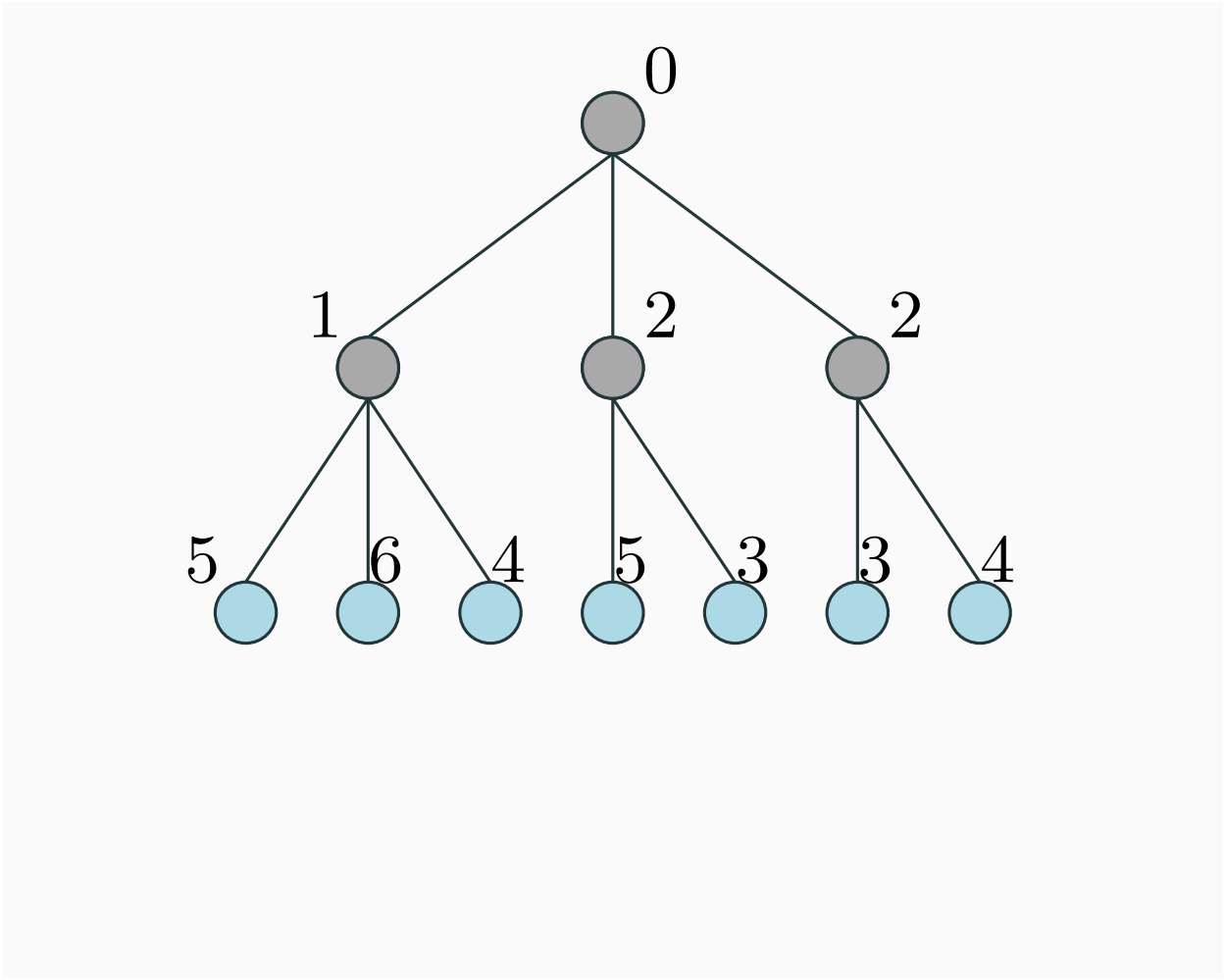
Beam Search



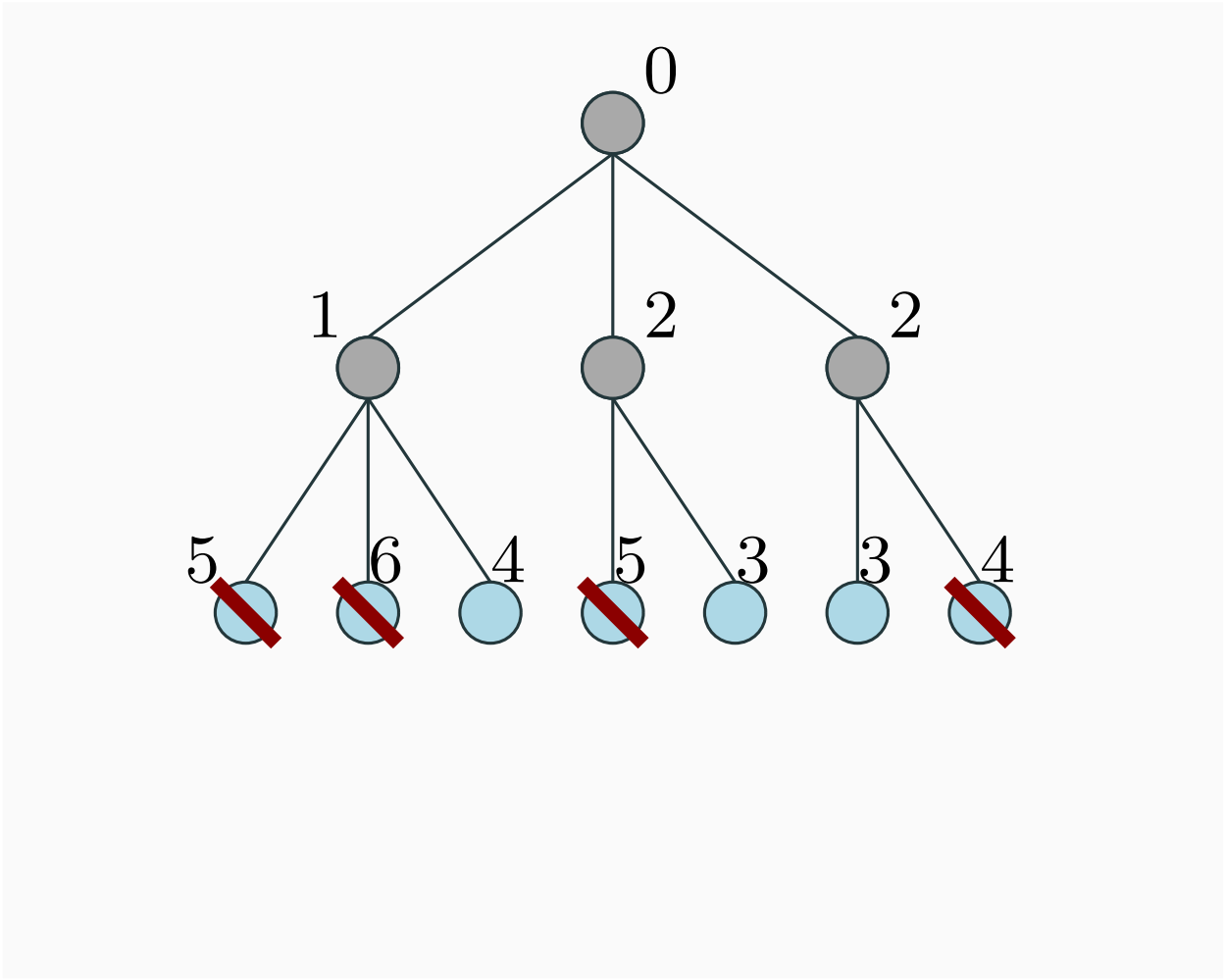
Beam Search



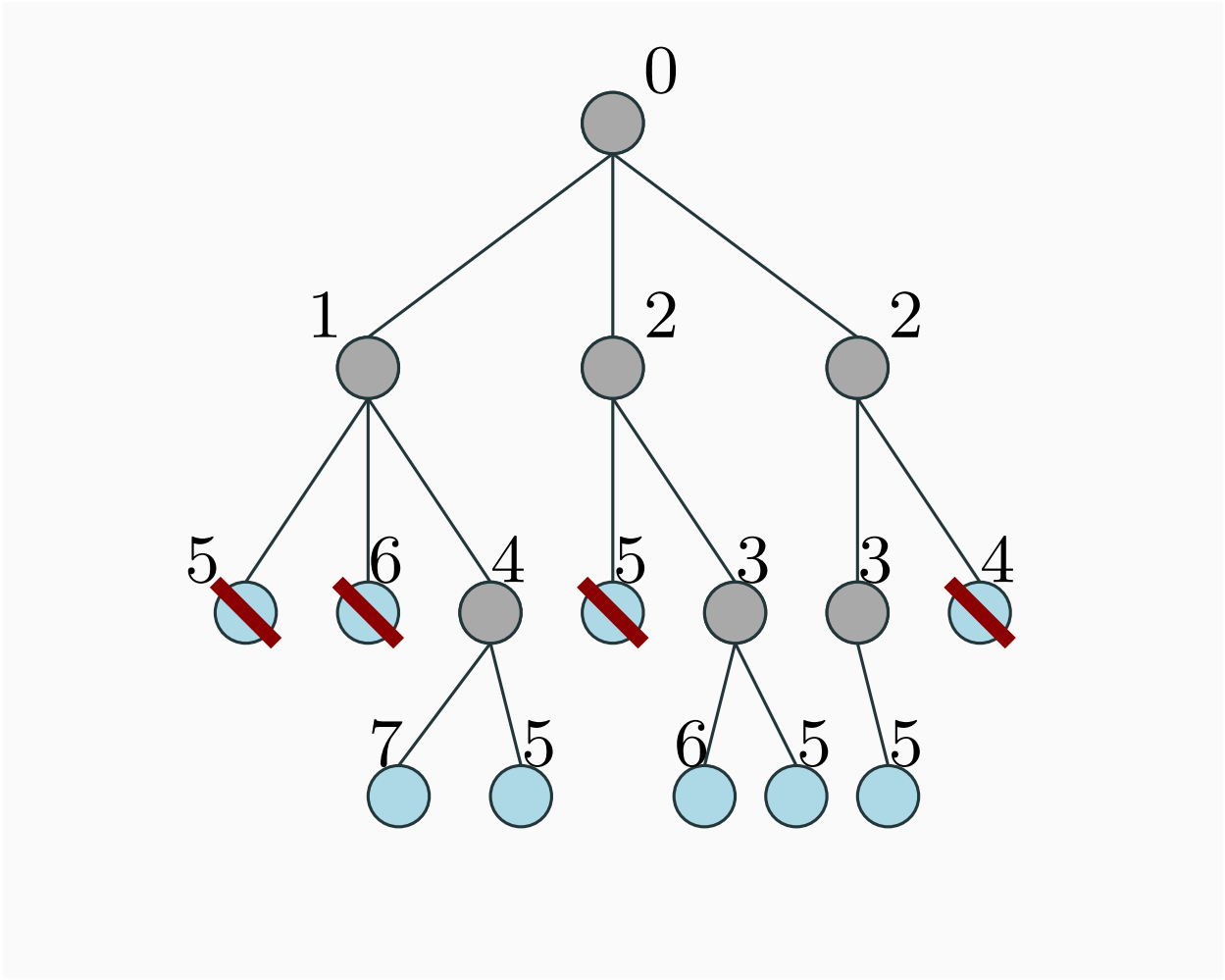
Beam Search



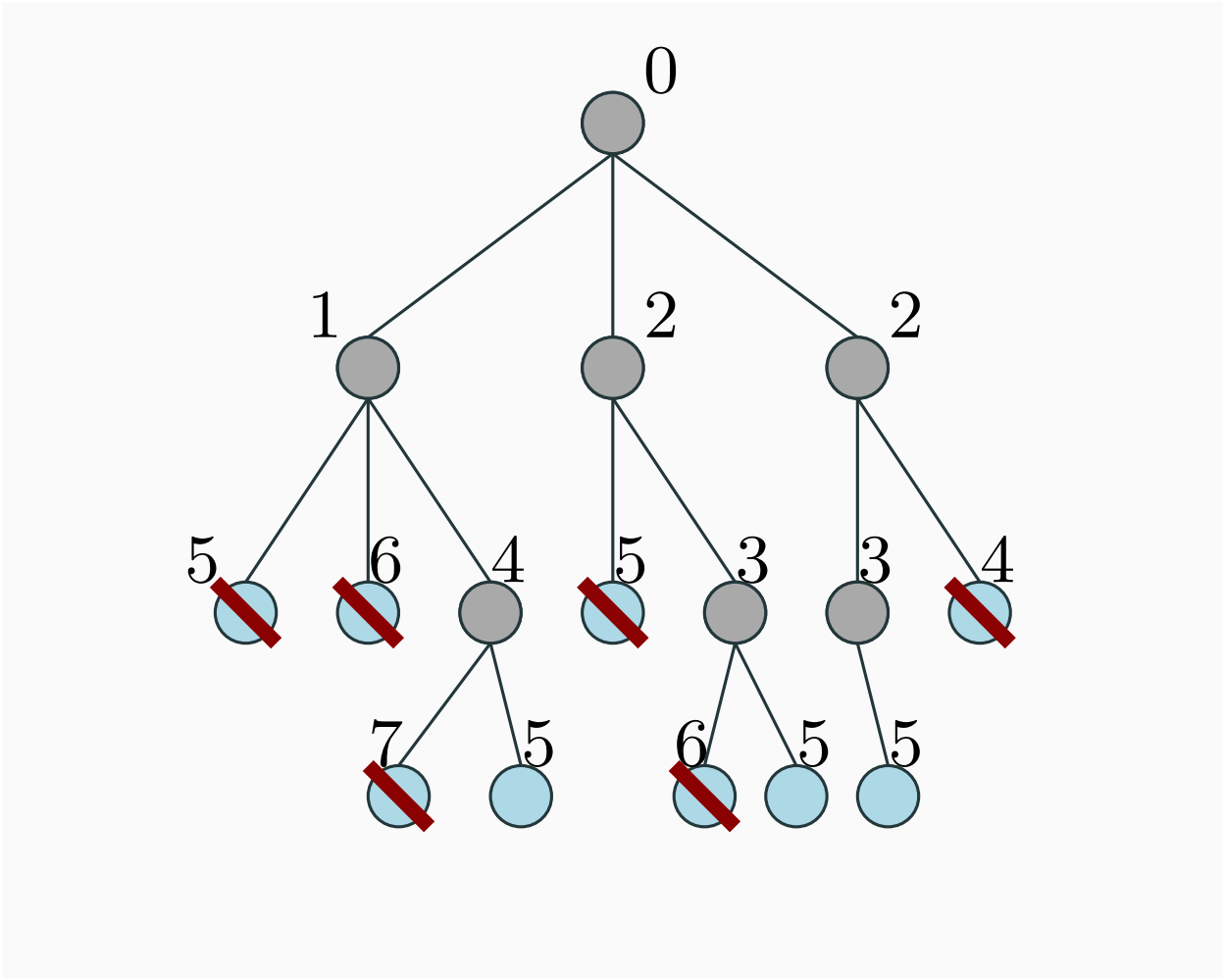
Beam Search



Beam Search



Beam Search

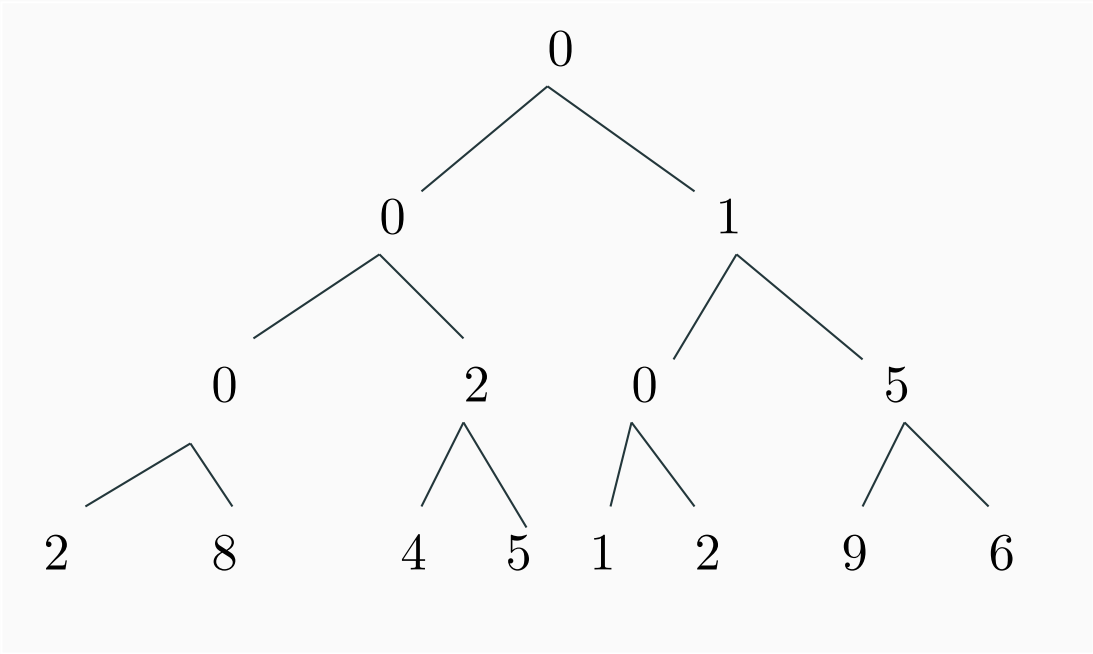


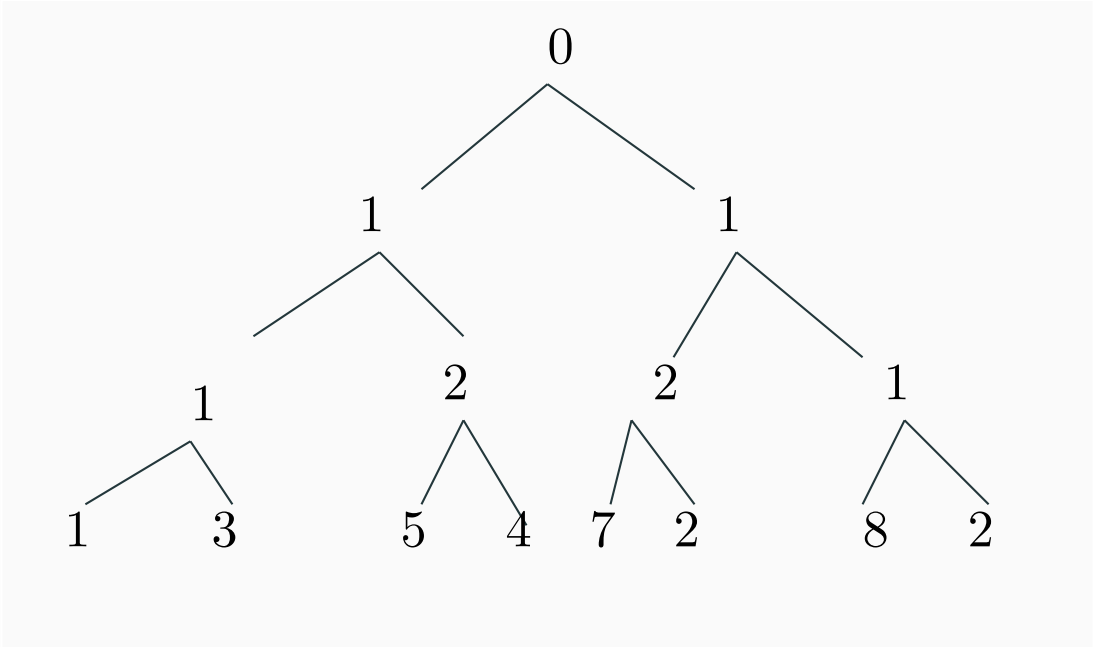
Exercise 2 - Try different tree searches

For a given Implicit tree, execute DFS, A*, (iterative) LDS, (iterative) Beam Search.

- Report the number of nodes needed to reach an optimal solution
- Report the number of nodes needed to prove optimality

Exercise 2 - Trees





What if the bound/guide is bad?

We now present (quickly) a few other tree-search algorithms useful when no good bound is available.

Branch & Greed

Key idea: Perform a probing step to guide the search

ACO

- Based on the ant metaphor
- Ants perform a probabilistic greedy
- When they find a solution, they update pheromones
- It can be seen as a form of online learning on where to find solutions

MCTS

- Uses random sampling to evaluate the potential of a given node.
- (generally) uses UCT to have a good exploration/exploitation trade-off.

Tree search for Combinatorial Optimization

Luc Libralesso

June 14, 2020

Operations Research and Combinatorial Optimization master lecture