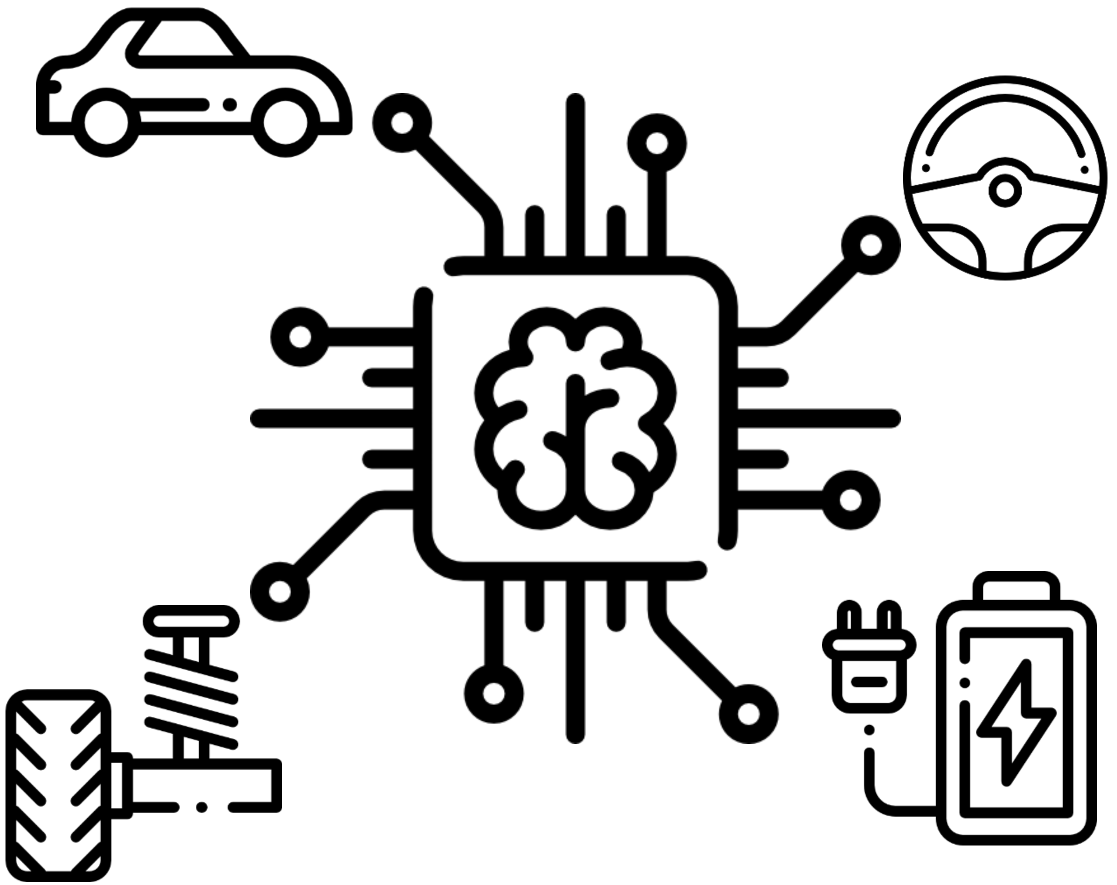
Lehrstuhl für Fahrzeugtechnik
Fakultät für Maschinenwesen
Technische Universität München

Artificial Intelligence in Automotive Technology

Johannes Betz / Prof. Dr.-Ing. Markus Lienkamp / Prof. Dr.-Ing. Boris Lohmann

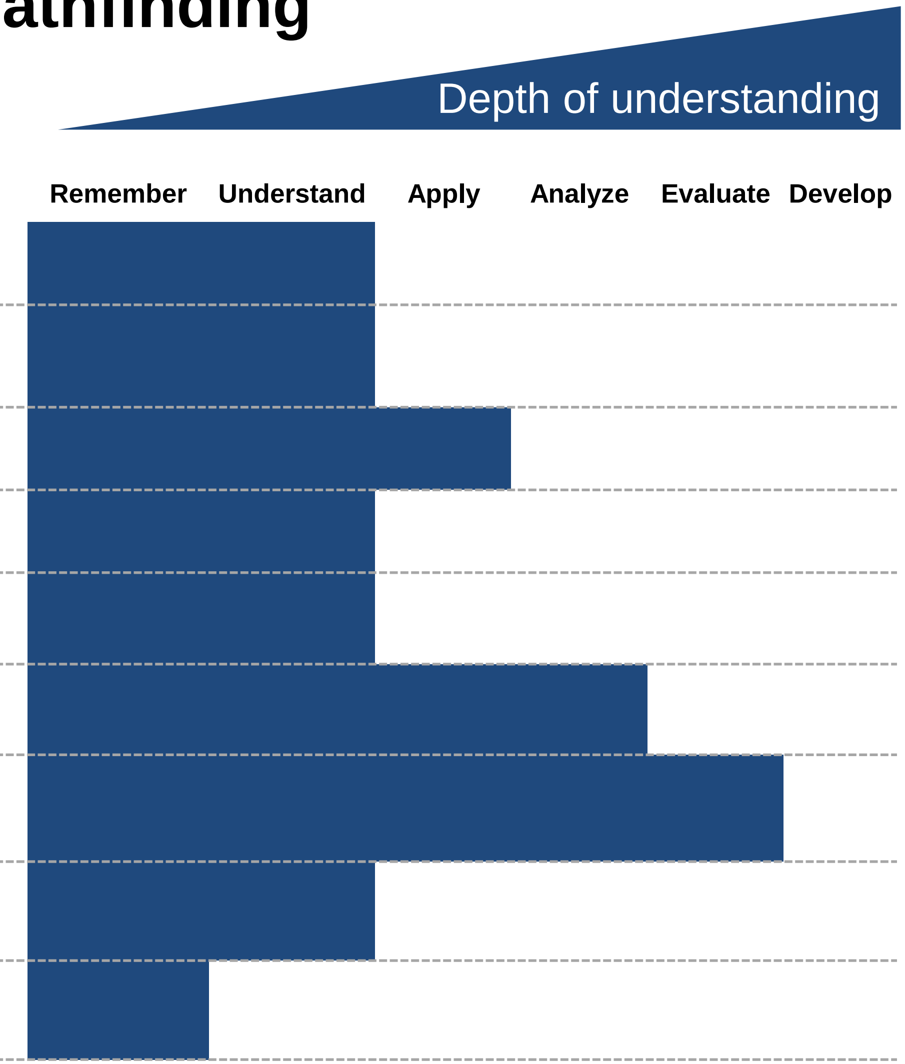


Lecture Overview

Overall Introduction for the Lecture 18.10.2018 – Betz Johannes	**6 Pathfinding: From British Museum to A*** 29.11.2018 – Lennart Adenaw	**11 Reinforcement Learning** 17.01.2019 – Christian Dengler
1 Introduction: Artificial Intelligence 18.10.2018 – Betz Johannes	**P6:** 29.11.2018 – Lennart Adenaw	**P11** 17.01.2019 – Christian Dengler
P1: 18.10.2018 – Betz Johannes	**7 Introduction: Artificial Neural Networks** 06.12.2018 – Lennart Adenaw	**12 AI-Development** 24.01.2019 – Johannes Betz
2 Perception 25.10.2018 – Betz Johannes	**P7** 06.12.2018 – Lennart Adenaw	**P12** 24.01.2019 – Johannes Betz
P2: 25.10.2018 – Betz Johannes	**8 Deep Neural Networks** 13.12.2018 – Jean-Michael Georg	**13 Free Discussion** 31.01.2019 – Betz/Adenaw
3 Supervised Learning: Regression 08.11.2018 – Alexander Wischnewski	**P8** 13.12.2018 – Jean-Michael Georg	
P3: 08.11.2018 – Alexander Wischnewski	**9 Convolutional Neural Networks** 20.12.2018 – Jean-Michael Georg	
4 Supervised Learning: Classification 15.11.2018 – Jan-Cedric Mertens	**P9** 20.12.2018 – Jean-Michael Georg	
P4: 15.11.2018 – Jan-Cedric Mertens	**10 Recurrent Neural Networks** 10.01.2019 – Christian Dengler	
5 Unsupervised Learning: Clustering 22.11.2018 – Jan-Cedric Mertens	**P10** 10.01.2019 – Christian Dengler	

Objectives for Lecture 6: Pathfinding

After the lecture you are able to…



… name and explain core elements of a navigation system

…understand how real world geometries can be represented digitally

…. generate a routable graph from a given set of nodes, ways and tags

… name and explain different pathfinding algorithms

…. split the overall routing task into sub-tasks

…. draw, use and analyze search trees

…. apply pathfinding algorithms to a mathematical graph, explain their behaviors and compare and evaluate their performance on a given graph

…. name, explain and assign typical algorithmic properties to pathfinding algorithms

…. remember real world conditions in the application of navigation systems

Pathfinding: From British Museum to A*
Johannes Betz / Prof. Dr. Markus Lienkamp / Prof. Dr. Boris Lohmann
(Lennart Adenaw, M. Sc.)

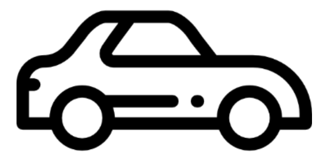
Agenda

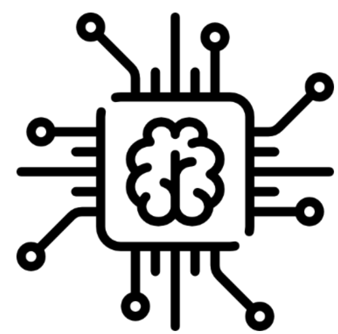
1. Chapter: Introduction
  1.1 Navigation
  1.2 Modelling Streets with Graphs

2. Chapter: Algorithms
  2.1 British Museum Algorithm
  2.2 Breadth First, Depth First, Best First
  2.3 Dijkstra, A*

3. Chapter: Application

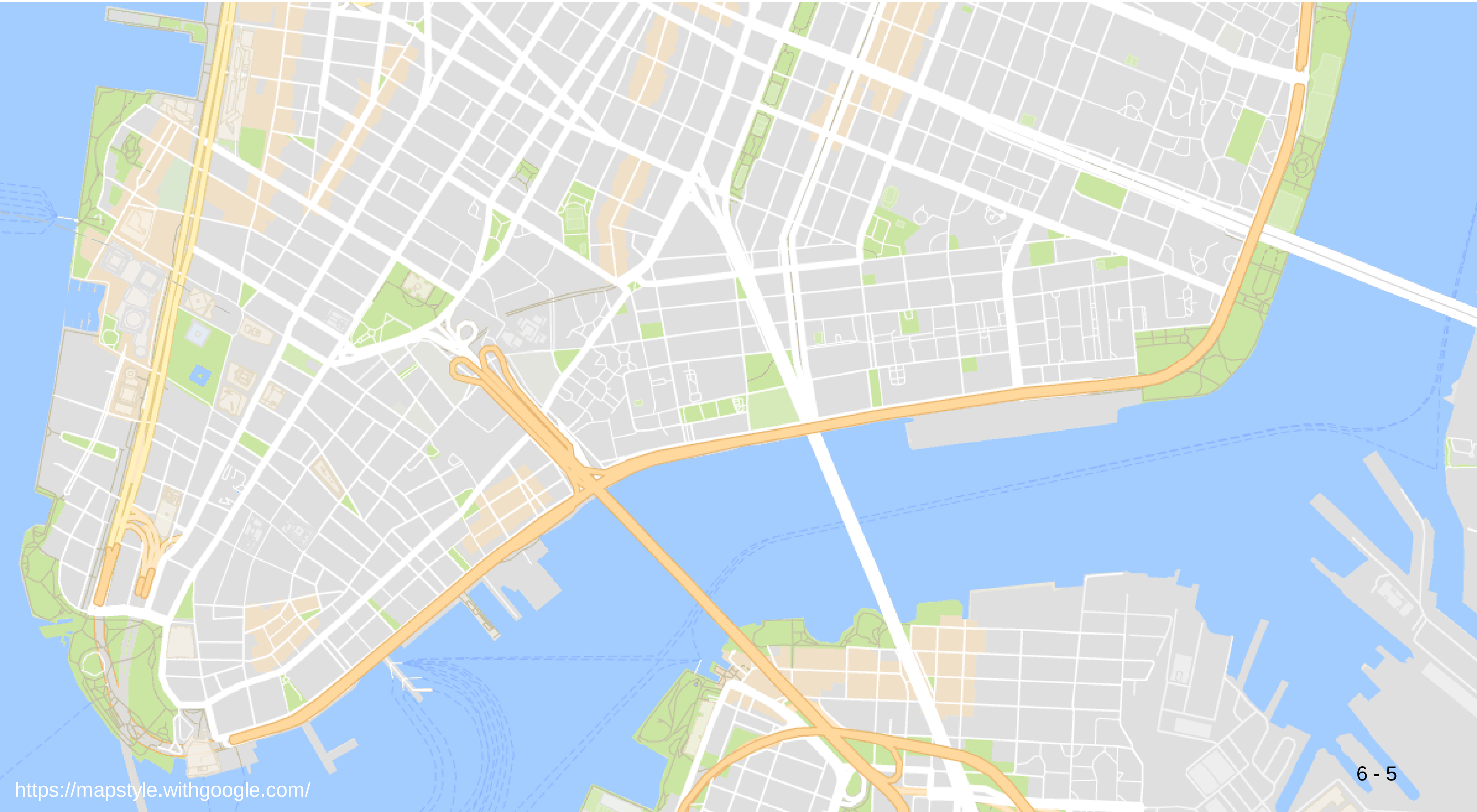
4. Chapter: Summary





Introduction – Navigation

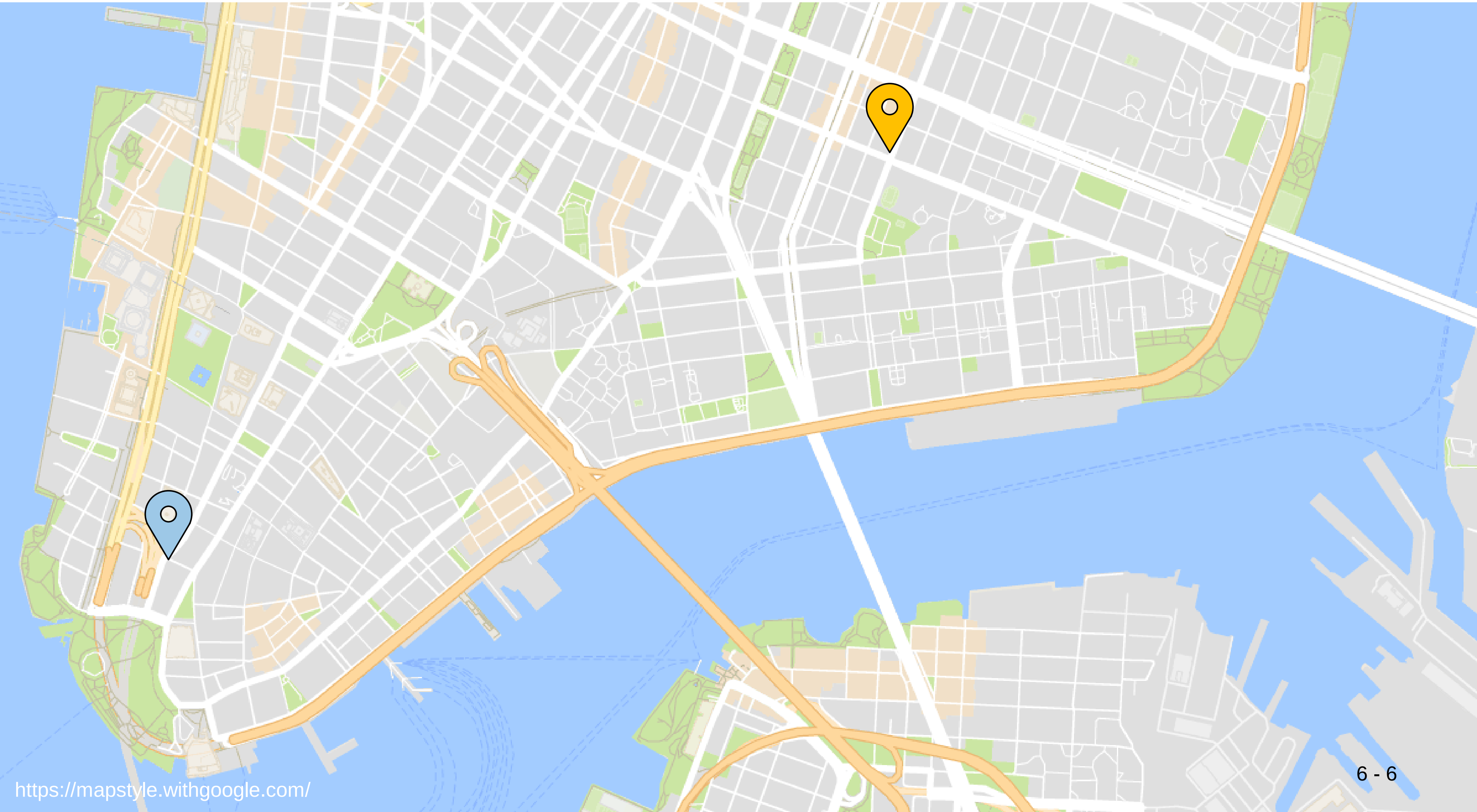
The human approach



https://mapstyle.withgoogle.com/

Introduction – Navigation

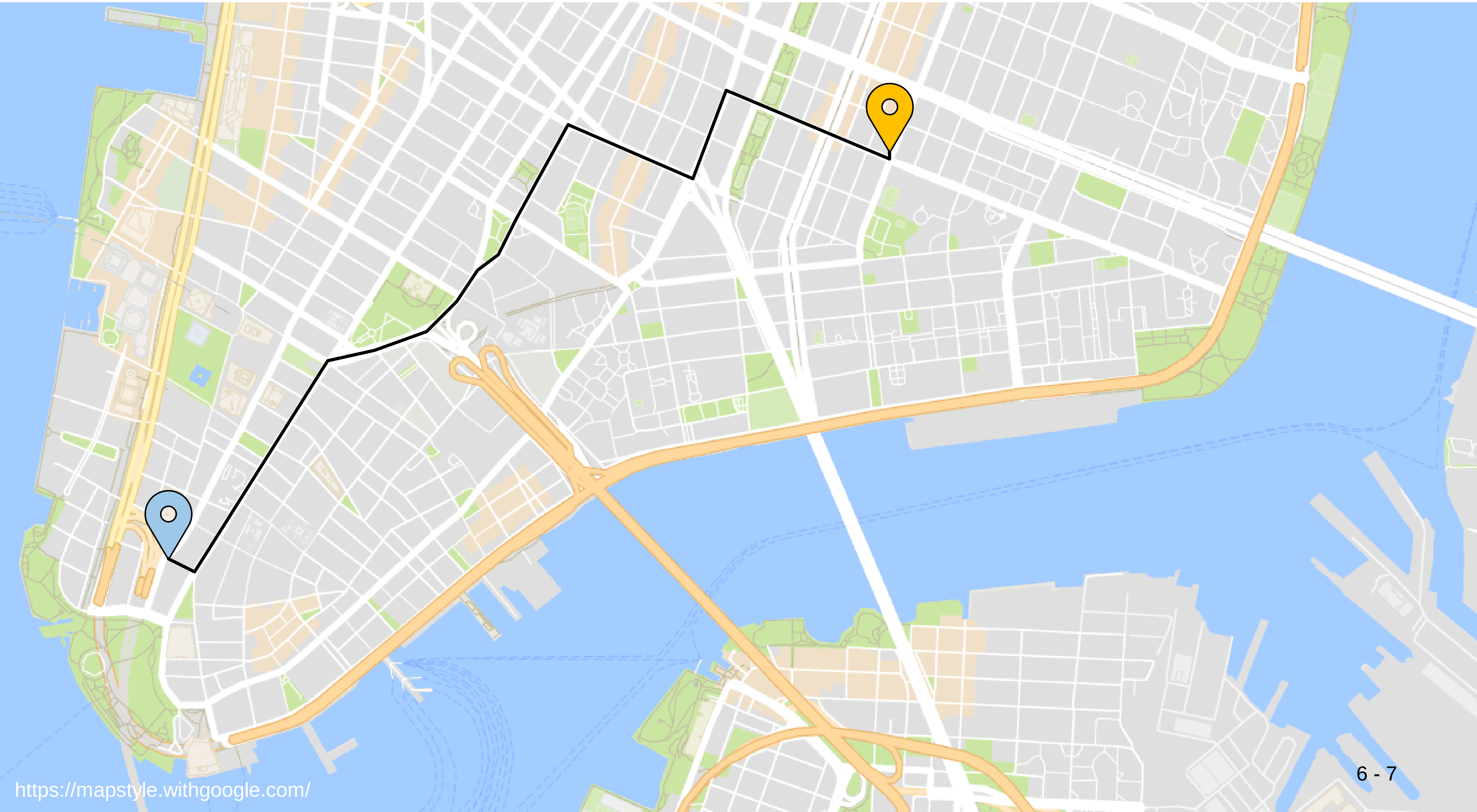
The human approach



https://mapstyle.withgoogle.com/

Introduction – Navigation

The human approach



https://mapstyle.withgoogle.com/

Additional Slides

The human approach to finding routes on a street map is a very visual one. Feasible paths are developed by visually tracing different street lines in the general direction of the target. Obstacles, dead ends and restrictions are intuitively taken into account, although – in typical routing situations – no intensive thinking or reflection is carried out. The human act of finding a path between a starting point and a destination can be considered „intelligent".

As a matter of fact, we do not understand the human approach of finding routes on maps in detail. Hence, we can not model it directly. Instead, search algorithms have evolved, solving the same problems by supposedly different means. Some popular ones of these shall be introduced in this lecture.

The question that remains is whether or not the presented algorithms can themselves be considered „intelligent" solely because they solve a problem that humans solve by means of human intelligence.

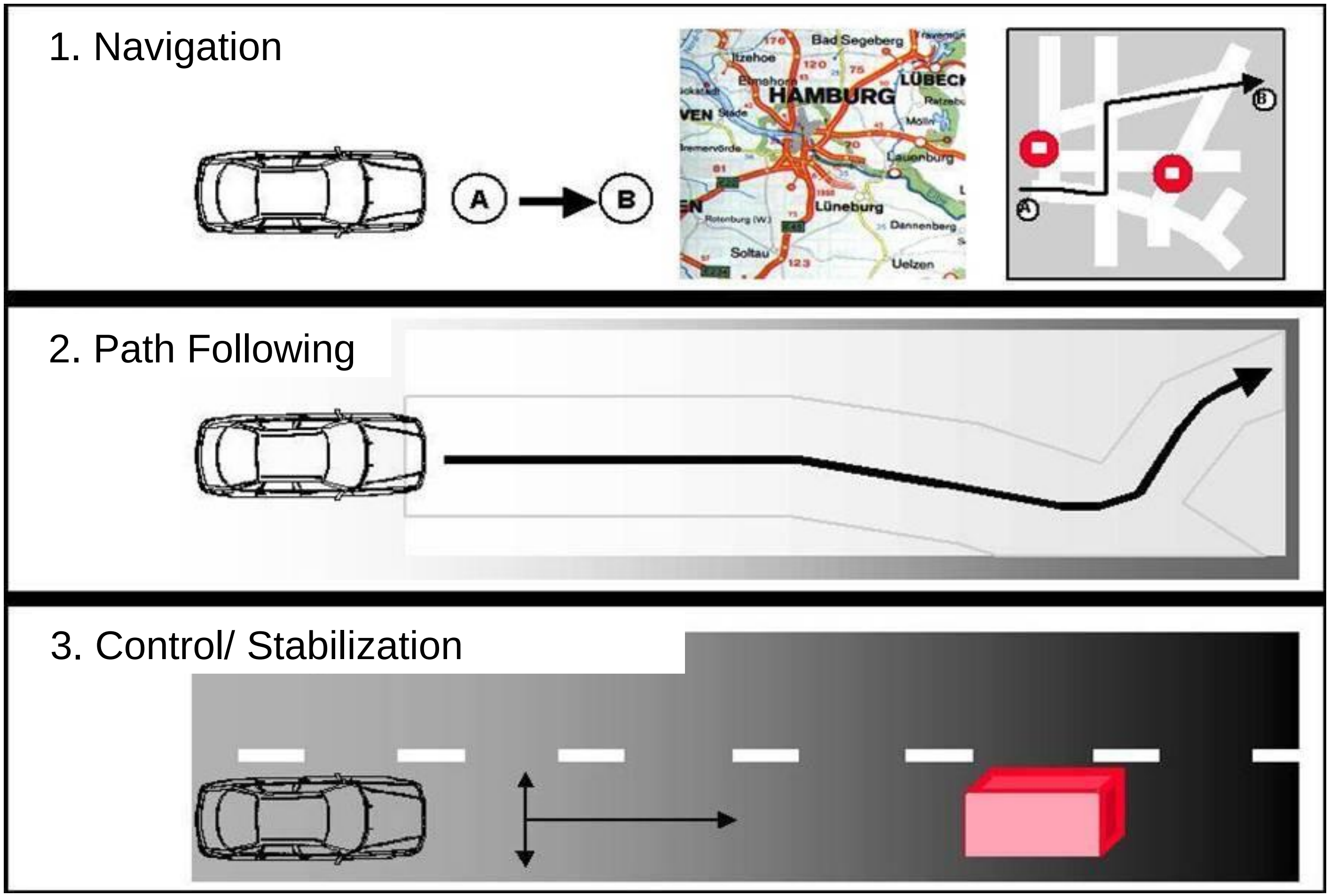
Introduction – Navigation
Definition

„The process or activity of accurately ascertaining one's position and planning and following a route."

https://en.oxforddictionaries.com/definition/navigation

Introduction – Navigation

Definition

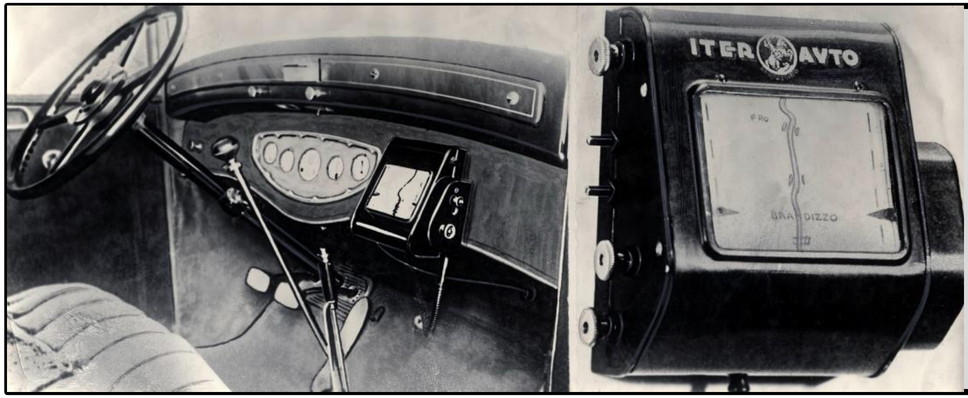


Hours to Minutes

Minutes to Seconds

Seconds to Miliseconds

Introduction – Navigation
Historical Approaches



Paper Map
Direct Transmission
„Play“ Specific Paths

1930 Iter Avto

Paper Map
Gyroscope
Scrolls Map

1981 electro Gyrocator

https://ndrive.com/brief-history-gps-car-navigation/

Introduction – Navigation
Historical Approaches

Digital Map
Dead Reckoning
Cassettes

1985 Etak Navigator

Digital Color Map
Dead Reckoning
CD-ROM

1987 Toyota CD-ROM Navigation System

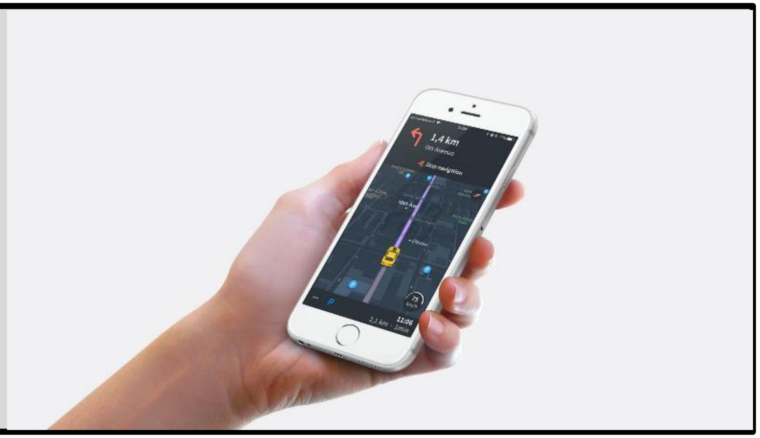
https://ndrive.com/brief-history-gps-car-navigation/

Introduction – Navigation
Historical Approaches

Digital Color Map
GPS

1990 Mazda Eunos Cosmo

GPS
Dead Reckoning
Map Matching
Speech Recognition
Dynamic Routing
Points of Interest (POI)
Modern Navigation Systems



https://ndrive.com/brief-history-gps-car-navigation/

Introduction – Navigation

What is needed for car navigation?

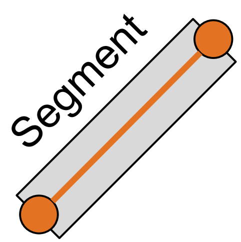
HMI	User Interaction
Navigation Engine	Routing Positioning Guidance
Data	Map Traffic POI

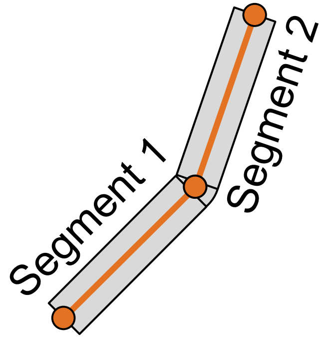
Introduction – Modelling Streets with Graphs
What needs to be modelled?

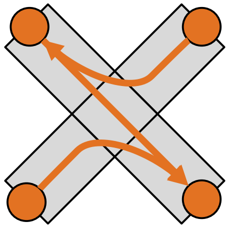
Road Geometry
- Position
- Length
- Start - End

Road Connectivity
- Intersections
- Connections
- Fly-Over

Road Attributes
- Turn Restrictions
- Speed Limits
- Type

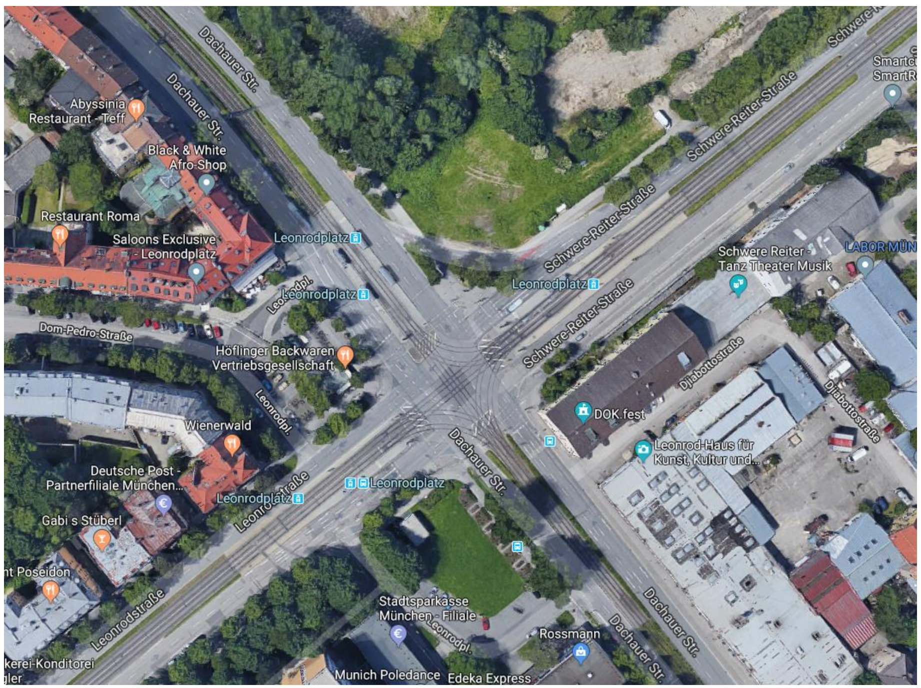






Introduction – Modelling Streets with Graphs
Data Sources

Satellite Imagery



https://www.maps.google.de

GPS Tracks

https://buy.garmin.com/de-DE/DE/p/550460

Introduction – Modelling Streets with Graphs
OSM

Open Steet Map (OSM)

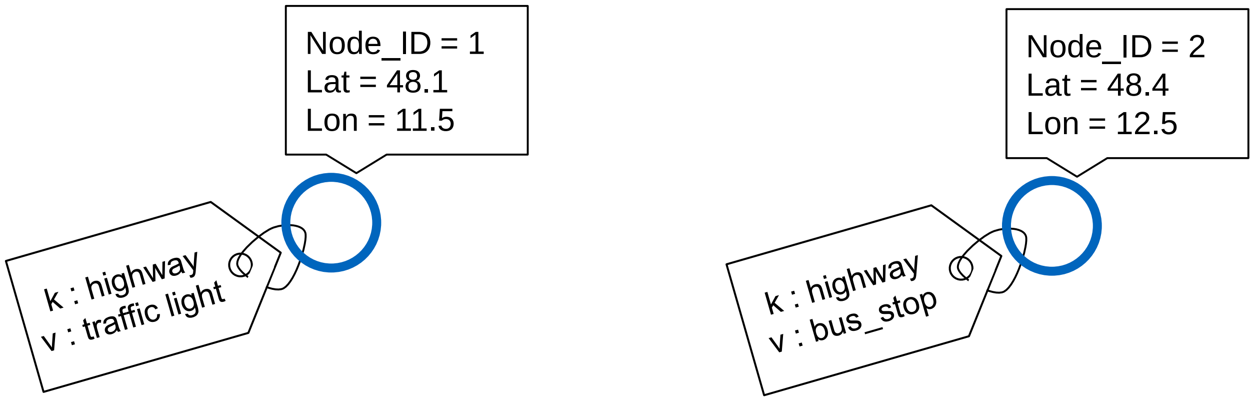
- Worldwide Map Data
- Free
- Started 2004 in London
- 1 000 000 Contributing Users
- Defines **Data Model** for Mapping

https://upload.wikimedia.org/wikipedia/commons/thumb/b/b0/Openstreetmap_logo.svg/1200px-Openstreetmap_logo.svg.png

Introduction – Modelling Streets with Graphs
OSM

Nodes

Defined by latitude, longitude and node id. Represents arbitrary locations.
Attributes specified by tags.

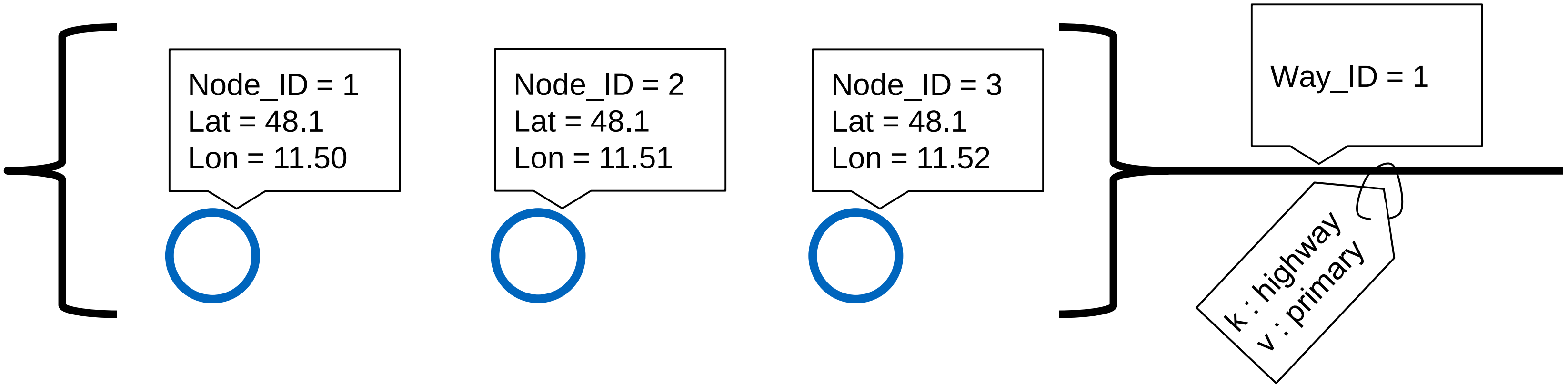


A **tag** consists of a **key** and a **value**. Tags describe features of map elements.
Keys and values are free text but conventions exist.

Introduction – Modelling Streets with Graphs
OSM

Ways

Defined by an ordered set of nodes. Groups nodes into lines.
Attributes specified by tags.

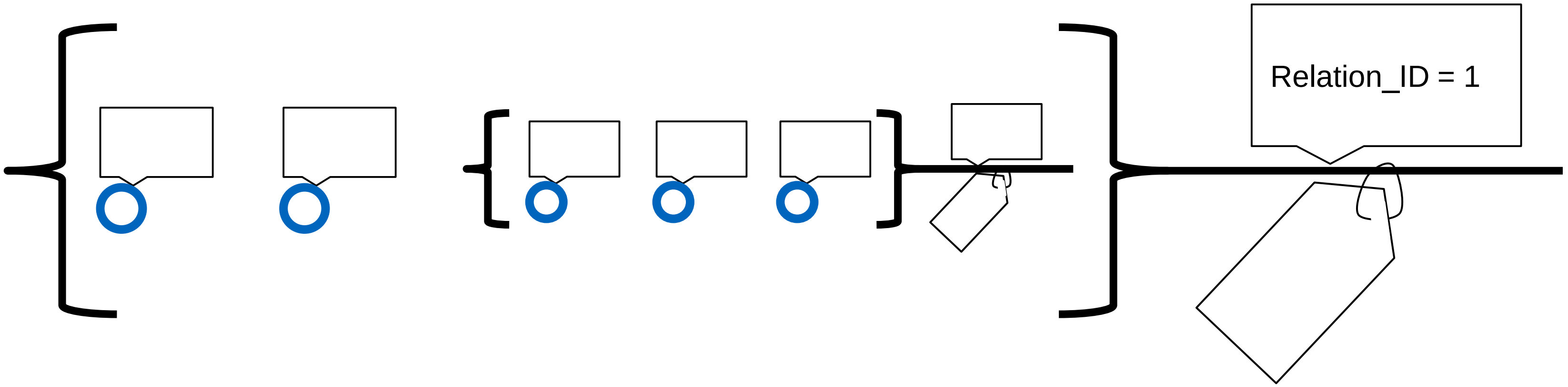


A **tag** consists of a **key** and a **value**. Tags describe features of map elements.
Keys and values are free text but conventions exist.

Introduction – Modelling Streets with Graphs
OSM

Relations

Defined by an ordered list of nodes, ways and/or relations. Defines logical or geographic relationships between other elements.

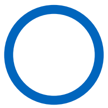


A **tag** consists of a **key** and a **value**. Tags describe features of map elements.
Keys and values are free text but conventions exist.

Introduction – Modelling Streets with Graphs

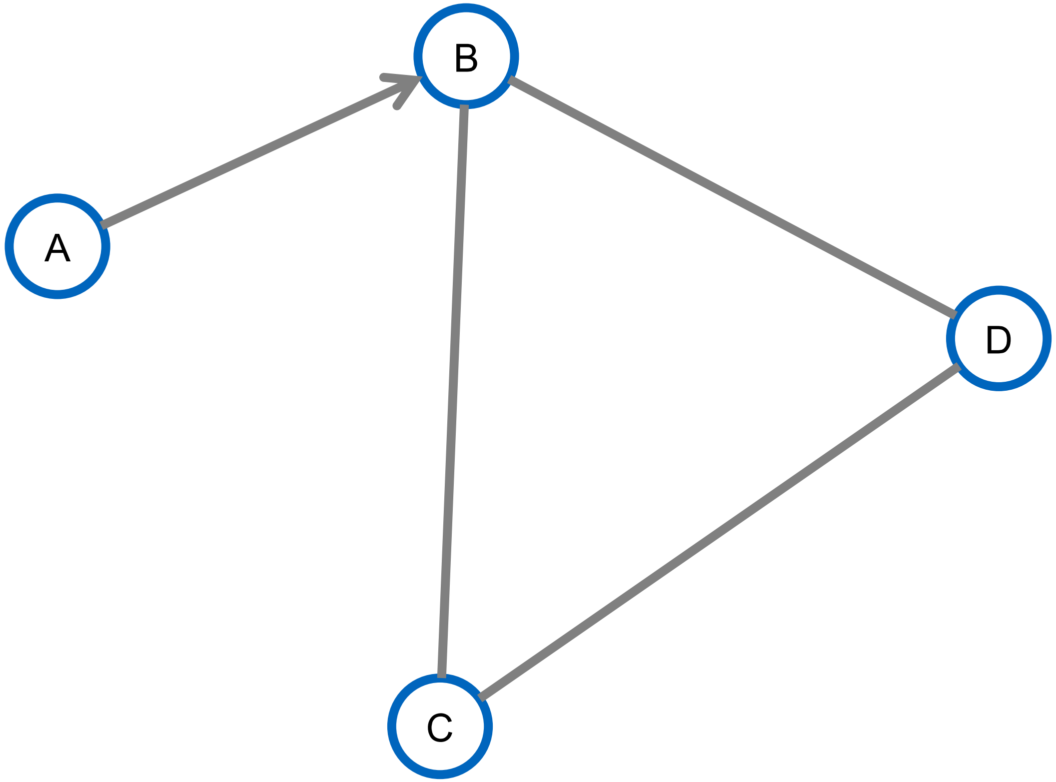
Graphs – Basic Elements

Node



Directed Edge

Undirected Edge



Introduction – Modelling Streets with Graphs

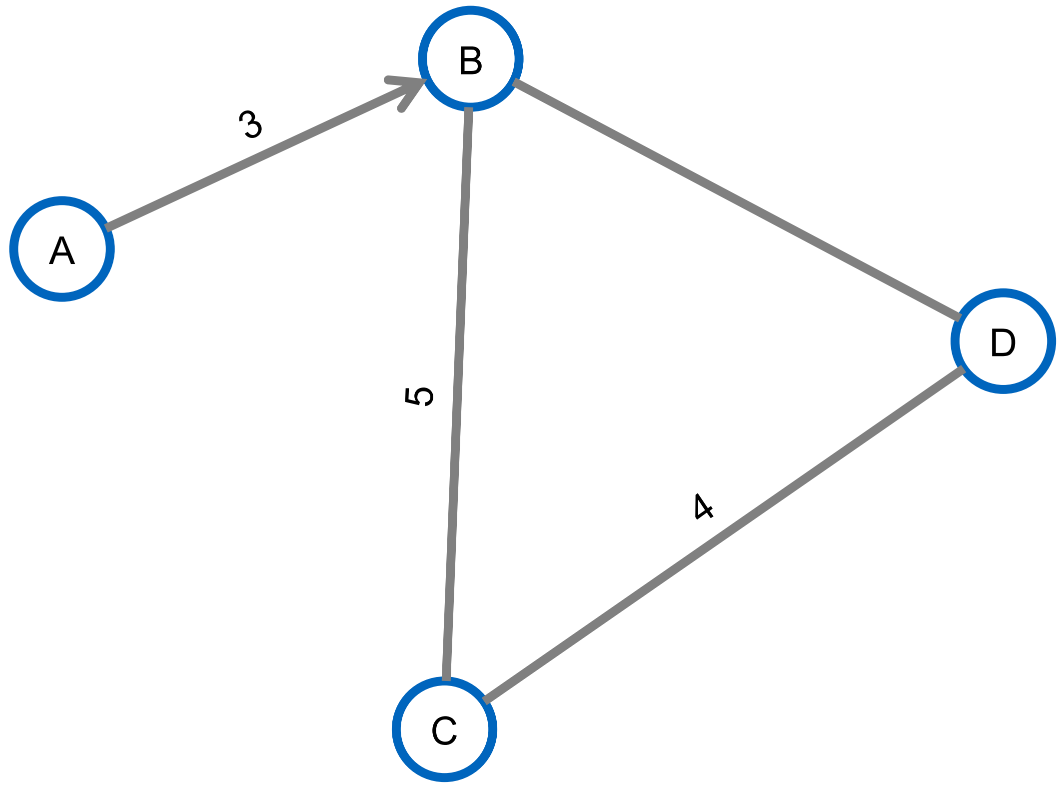
Graphs – Basic Elements

Weighted Edge

3

Unweighted Edge

Edge weights are interpretable as **cost** (e.g. time, distance …)



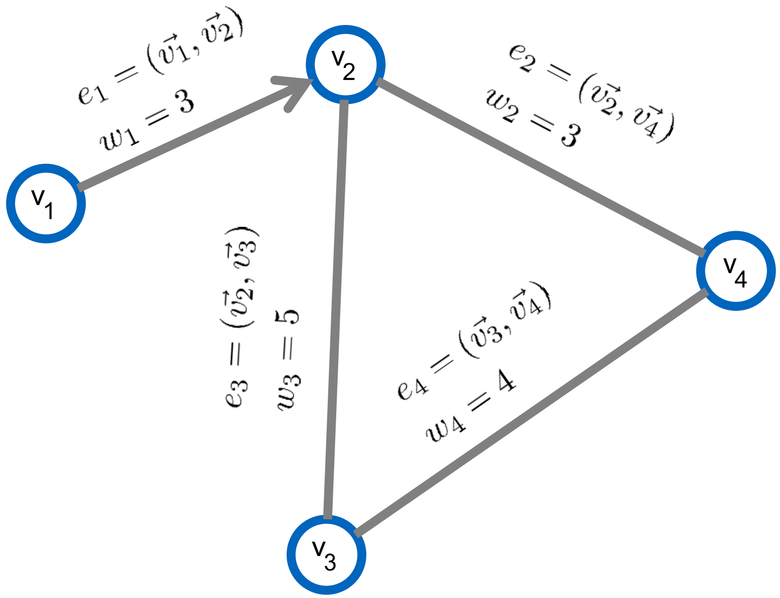
Introduction – Modelling Streets with Graphs

Graphs – Formal Description

$G:$	$Graph$
$V:$	$Nodes$
$\vec{v}, \vec{u}:$	$Node$
$E:$	$Edges$
$e:$	$Edge$
$W:$	$Weights$
$w:$	$Weight$
$d(\vec{u}, \vec{v}):$	$Distance$

$$G = (V, E)$$

$$e = (\vec{u}, \vec{v});\ \vec{u}, \vec{v} \in V$$



$$E = \{e_1, e_2, e_3\}$$

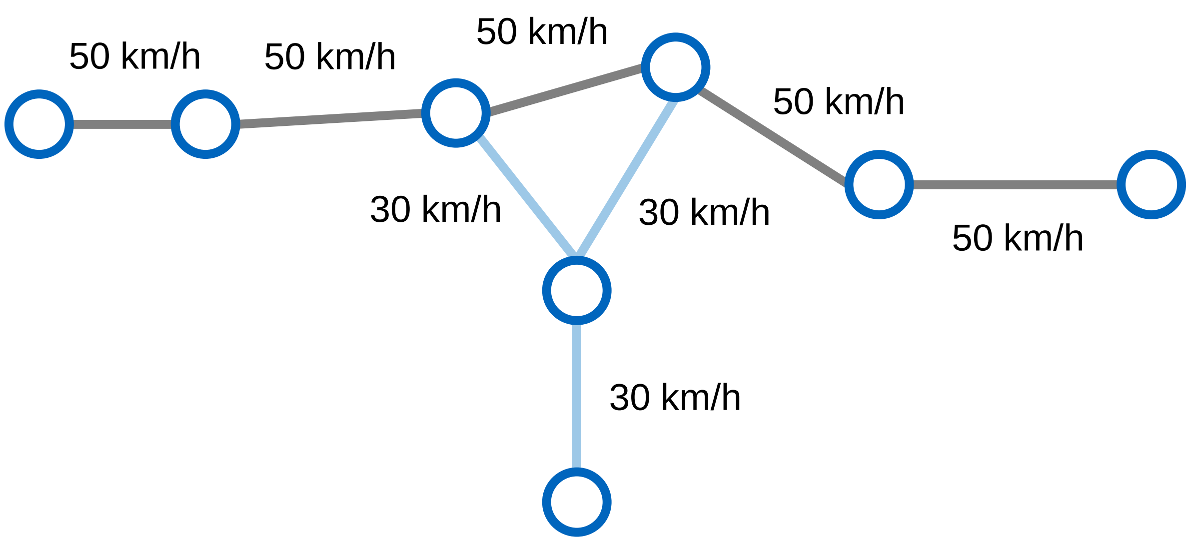
$$V = \{\vec{v_1}, \vec{v_2}, \vec{v_3}, \vec{v_4}\}$$

$$W = \{w_1, w_2, w_3, w_4\}$$

Introduction – Modelling Streets with Graphs
From OSM to Routable Graphs

Initial Graph

OSM Data provides **geometry** and **speed limits**

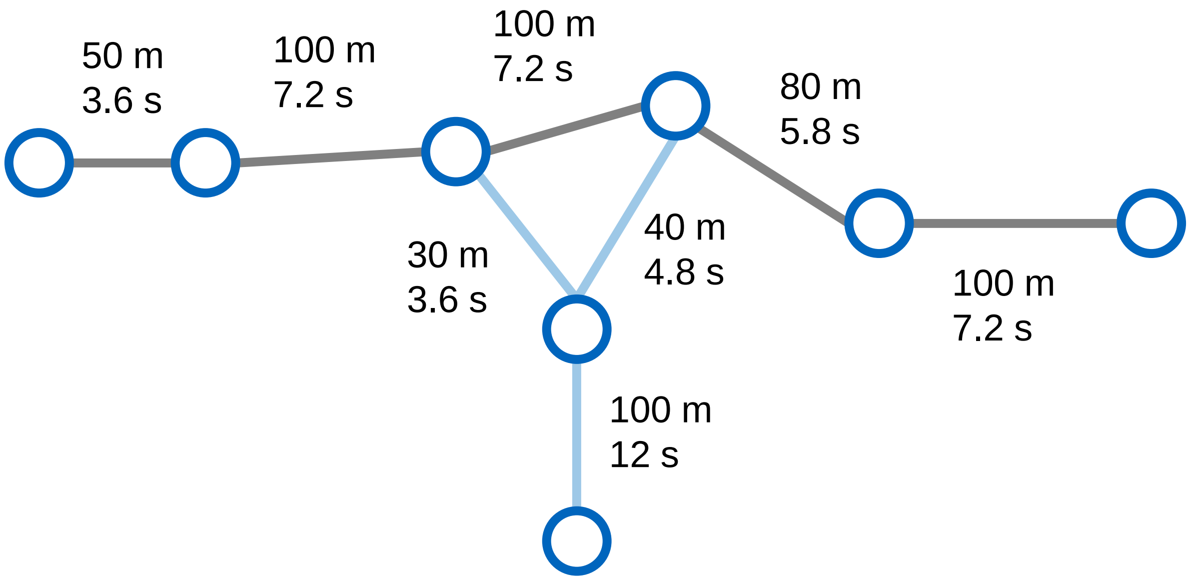


Introduction – Modelling Streets with Graphs

From OSM to Routable Graphs

Augmented Graph

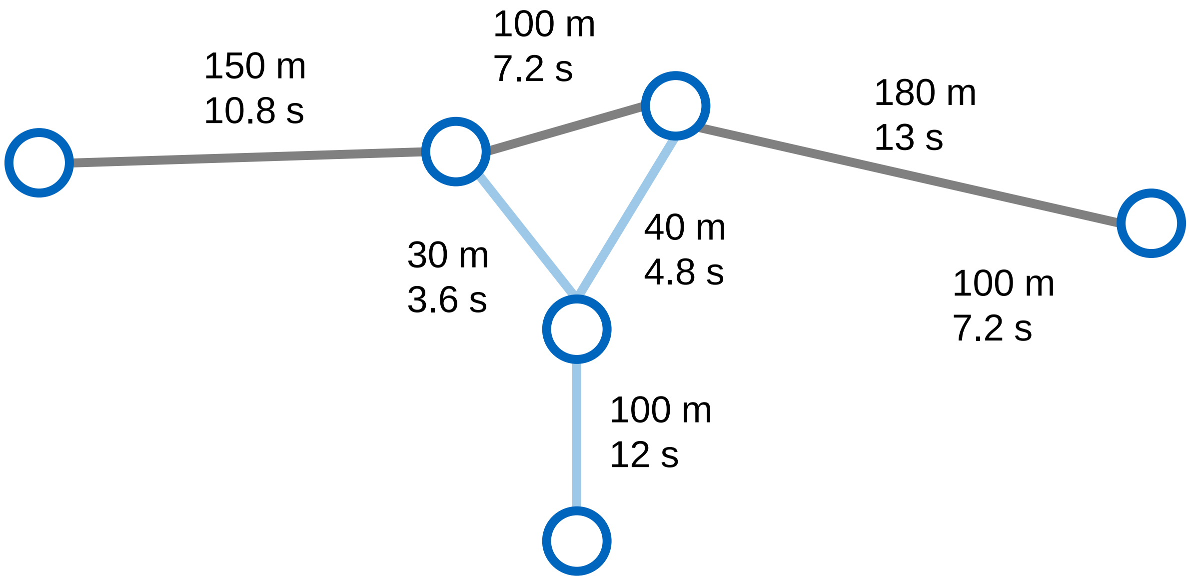
Edge weights like **distances** and **timespans** are calculated



Introduction – Modelling Streets with Graphs
From OSM to Routable Graphs

Compressed Graph

Graph compression **reduces complexity**:
Reduction of computational effort

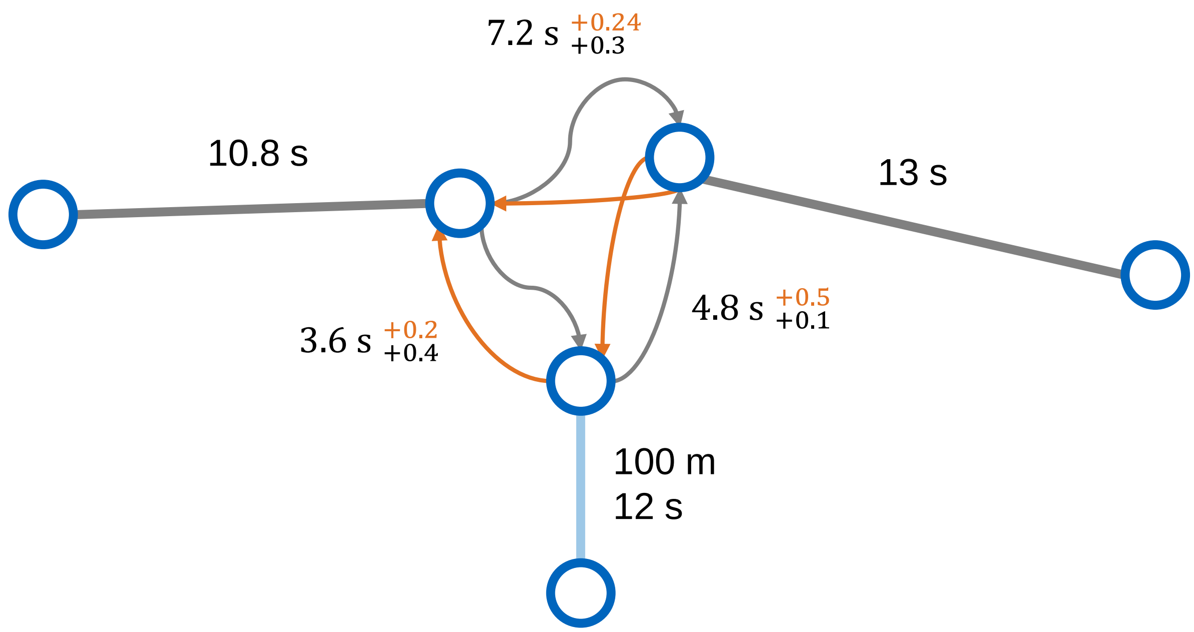


Introduction – Modelling Streets with Graphs

From OSM to Routable Graphs

Compressed Graph with turn restrictions and turn penalties

Turn restrictions are derived from OSM tags, **turn penalties** may be introduced for time optimal routing



Introduction – Modelling Streets with Graphs

Lecture Assumptions

Assumption	Description
$w_i \geq 0 \; \forall w_i \in W$	Only **non-negative edge weights**
$e = (\vec{v}, \vec{u}) = (\vec{u}, \vec{v}) \; \forall \; \vec{u}, \vec{v} \in V, e \in E$ $w = w((\vec{v}, \vec{u})) = w((\vec{u}, \vec{v}))$	**Edges** are **undirected**
$\vec{v} = \begin{pmatrix} v_x \\ v_y \end{pmatrix}, \vec{v} \in V$	Nodes are given by their coordinates in a **two dimensional space**
$w(\vec{v}, \vec{u}) = d(\vec{v}, \vec{u}) = \lVert\vec{v} - \vec{u}\rVert_2$ $\lVert\vec{v} - \vec{u}\rVert_2 = \sqrt{(v_x - u_x)^2 + (v_y - u_y)^2}$	**Edge weights** are defined by a distance function. The **euclidean distance** is used

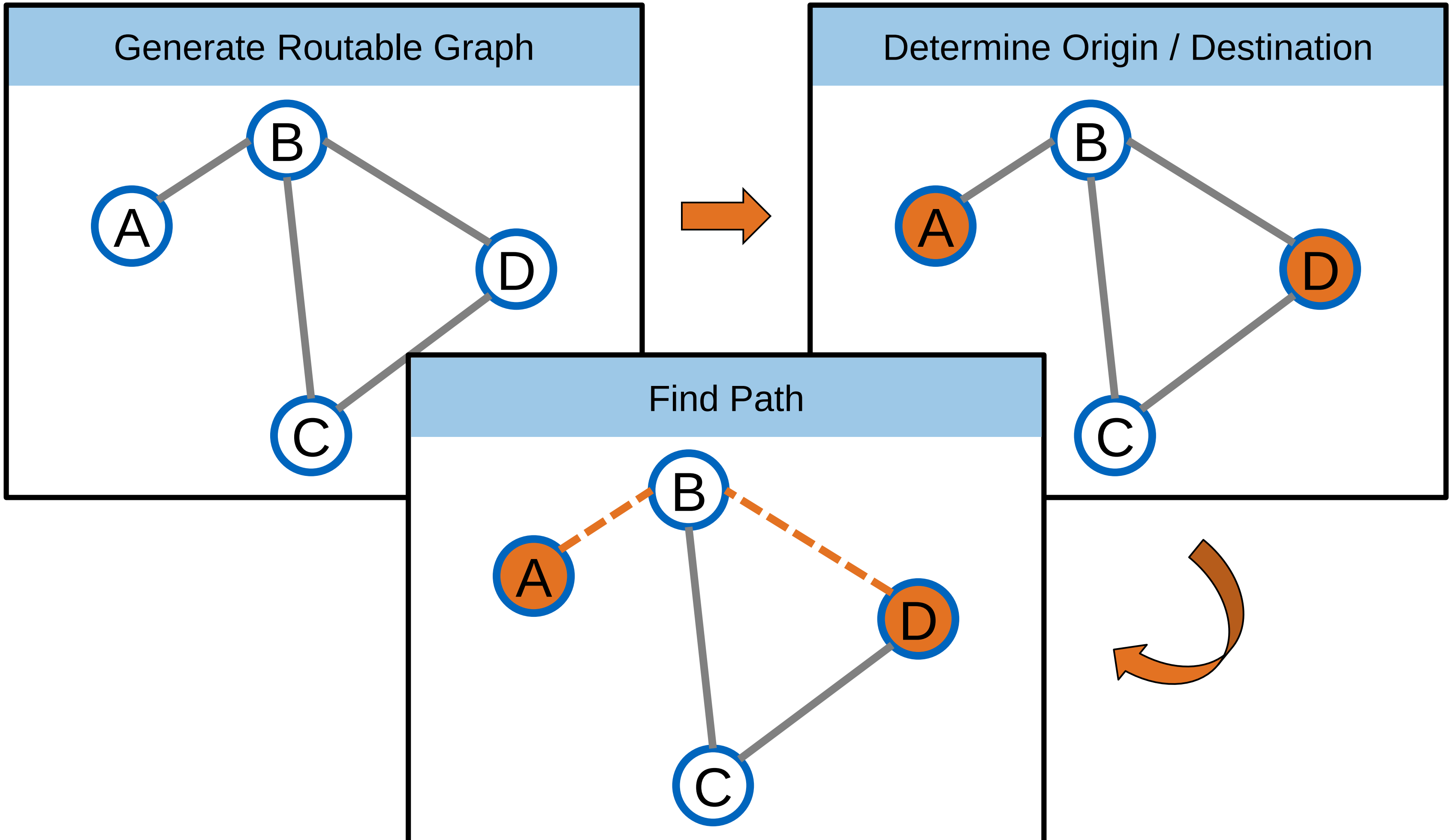
Introduction – Navigation

What is needed for car navigation?

HMI	User Interaction
Navigation Engine	Routing Positioning Guidance
Data	Map Traffic POI

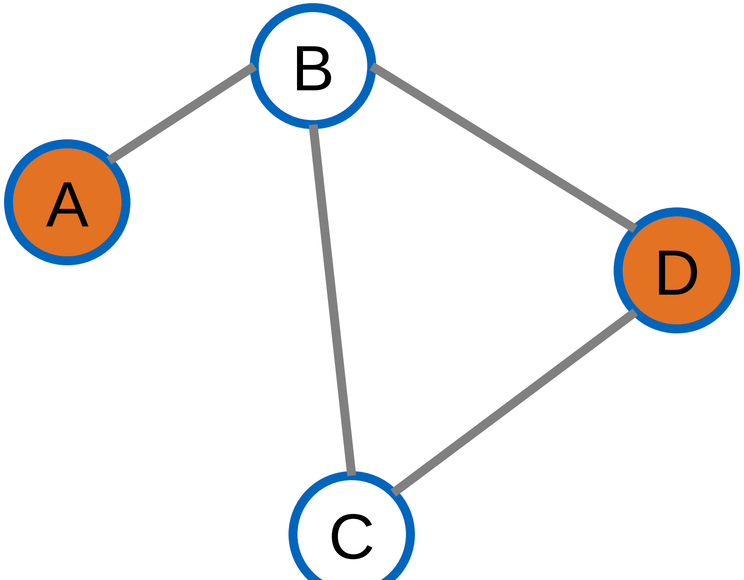
Algorithms

Overview Routing

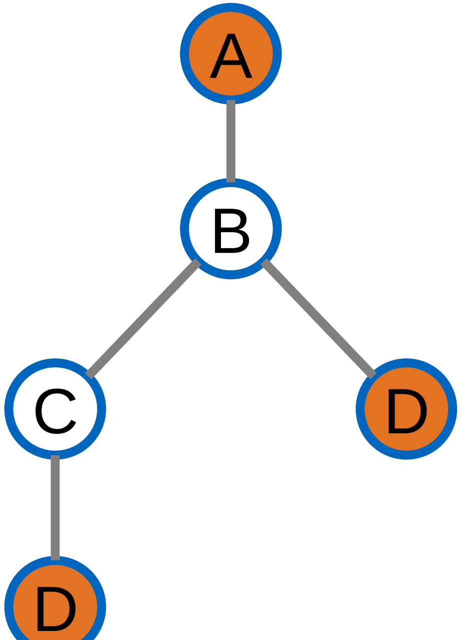


Algorithms
Search Trees

Graph

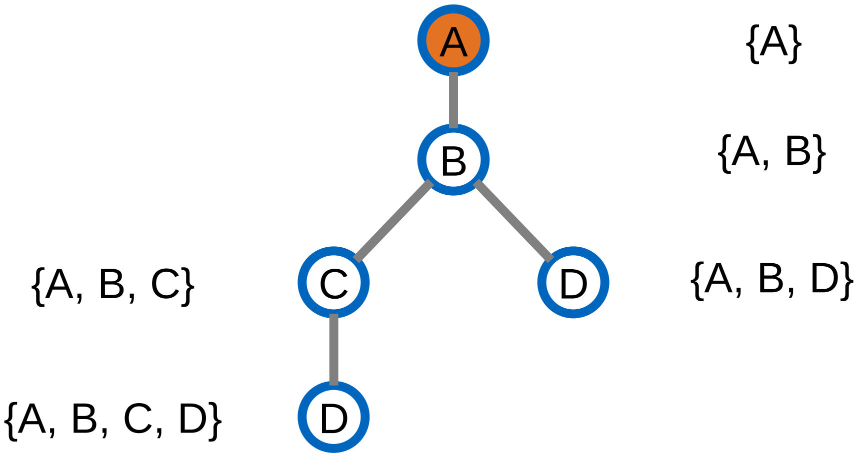


Search Tree



Algorithms

Search Trees



A search tree can be made from a graph

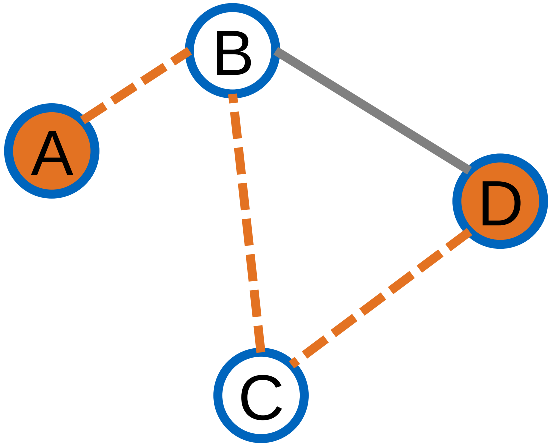
Each child node denotes a path that is a one-step extension of the path denoted by its parent

Converting graphs into search trees:
Tracing out all possible paths until no further extension is possible

Algorithms
Pathfinding: Problem Formulations

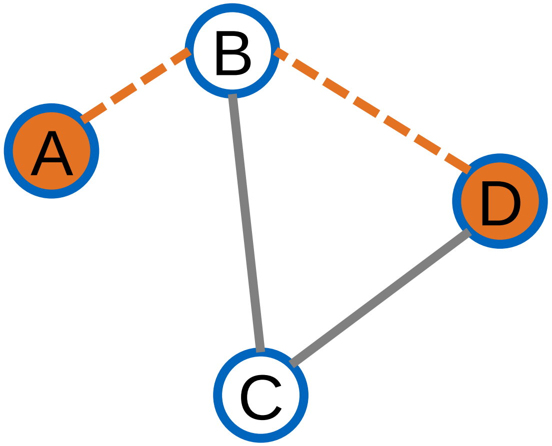
A : Find a Path

- Depth First
- Breadth First
- Best First



B : Find an optimal Path

- Dijkstra
- A*



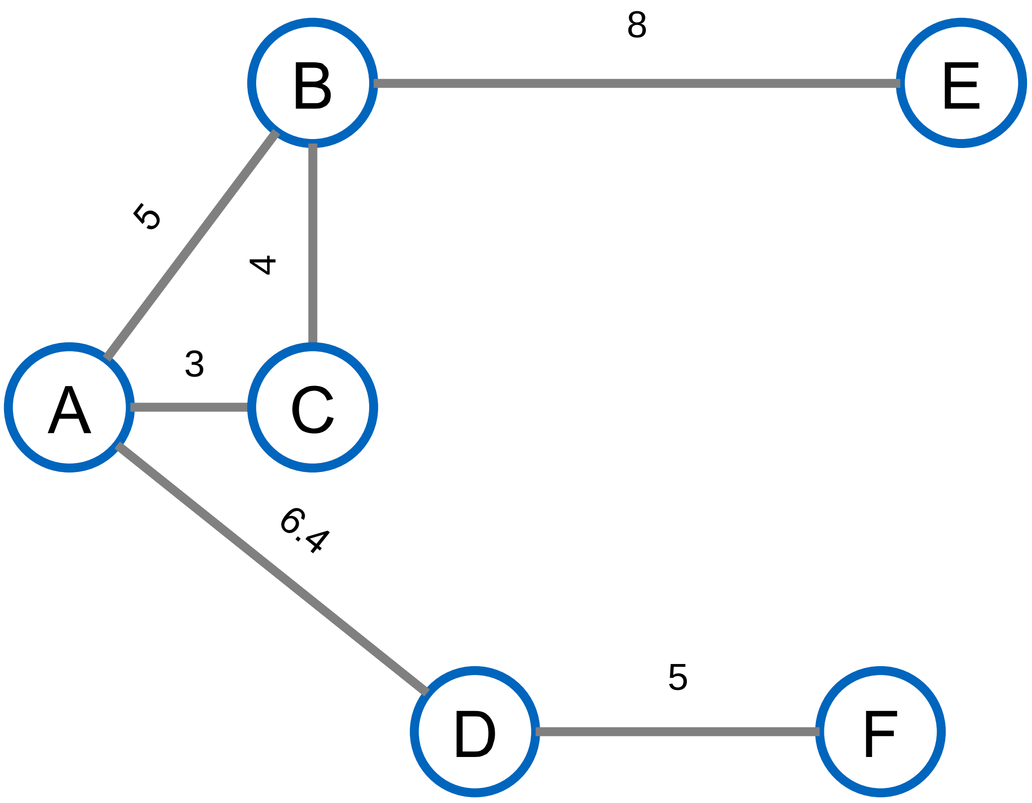
Algorithms
Completeness

A search algorithm is complete if it is guaranteed to find an existing solution in a finite amount of time.

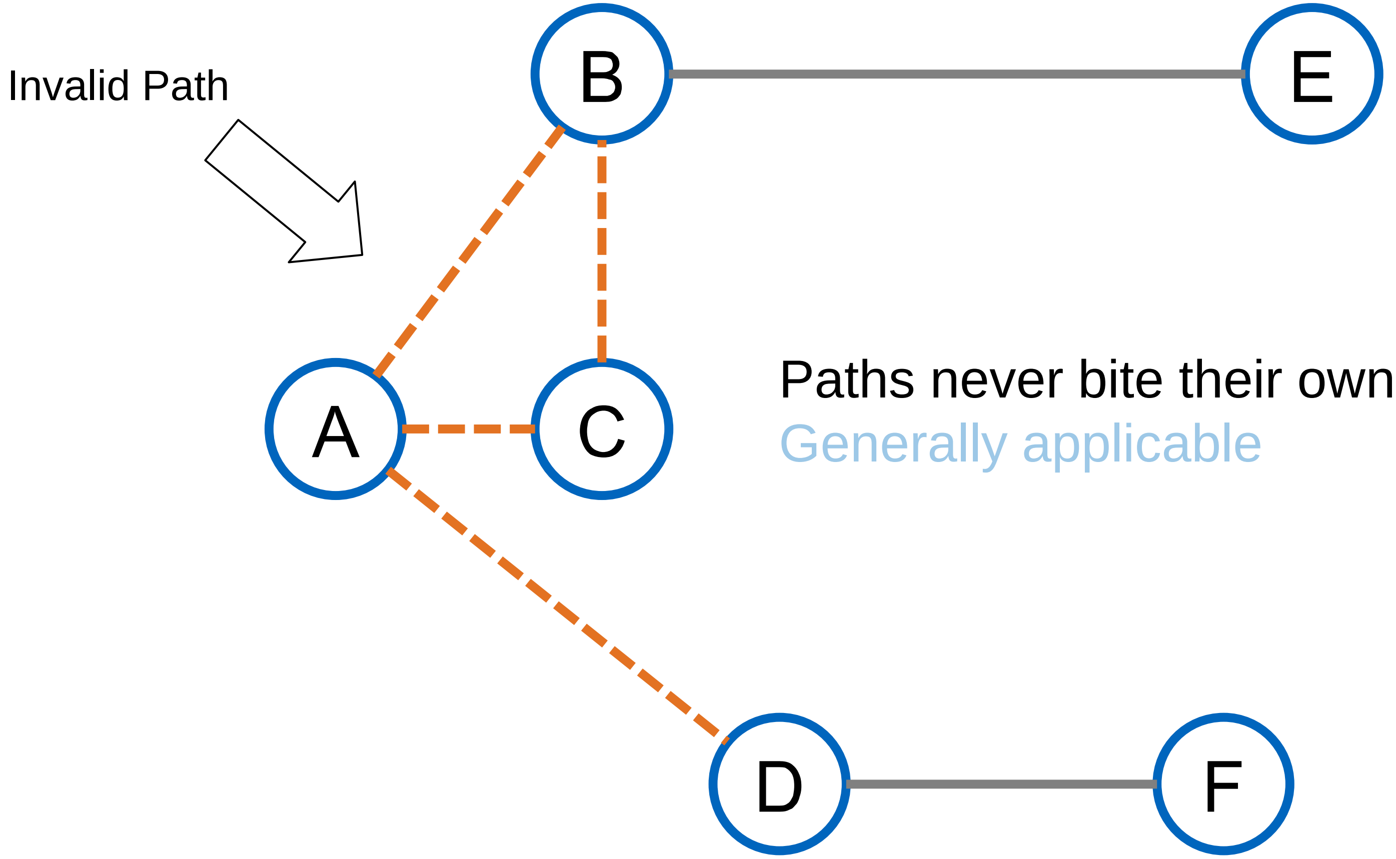
(Almost) all algorithms in this lecture

Algorithms

Pathfinding Example Graph



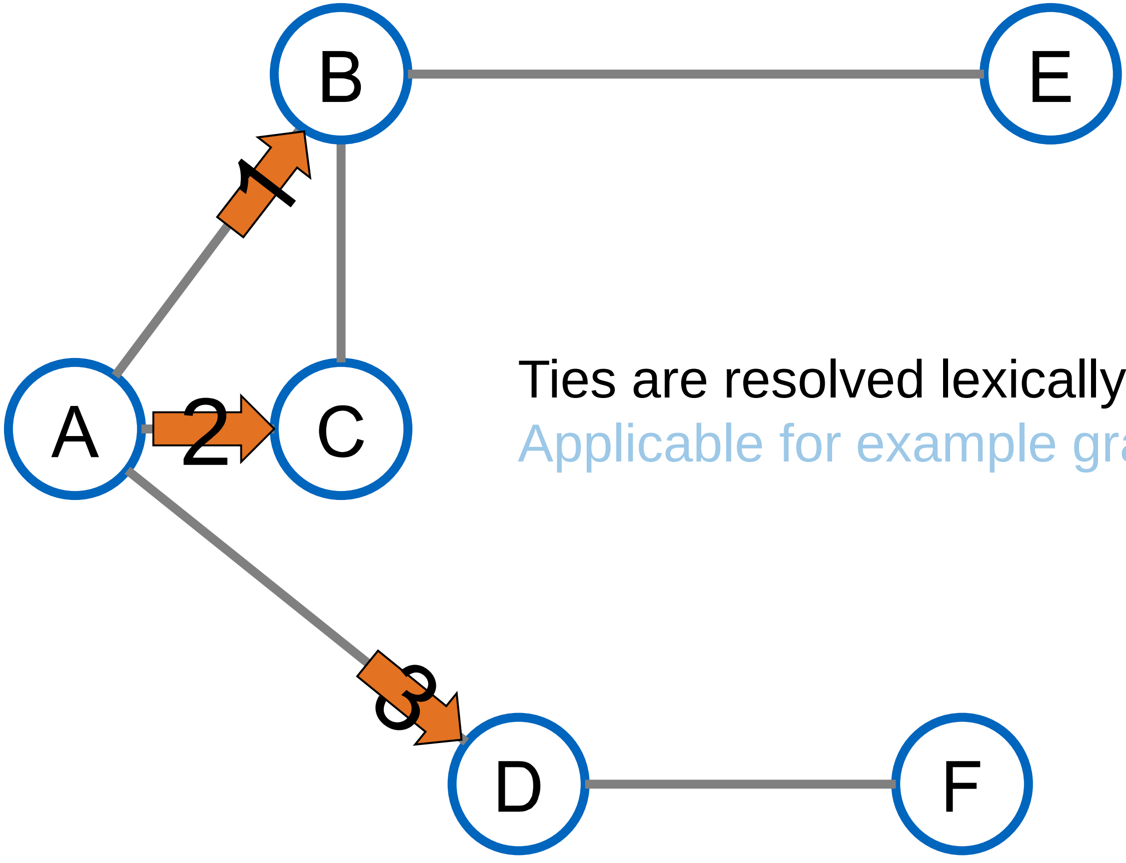
Algorithms
Ground Rules Example Graph



Paths never bite their own tail

Generally applicable

Algorithms
Ground Rules Example Graph



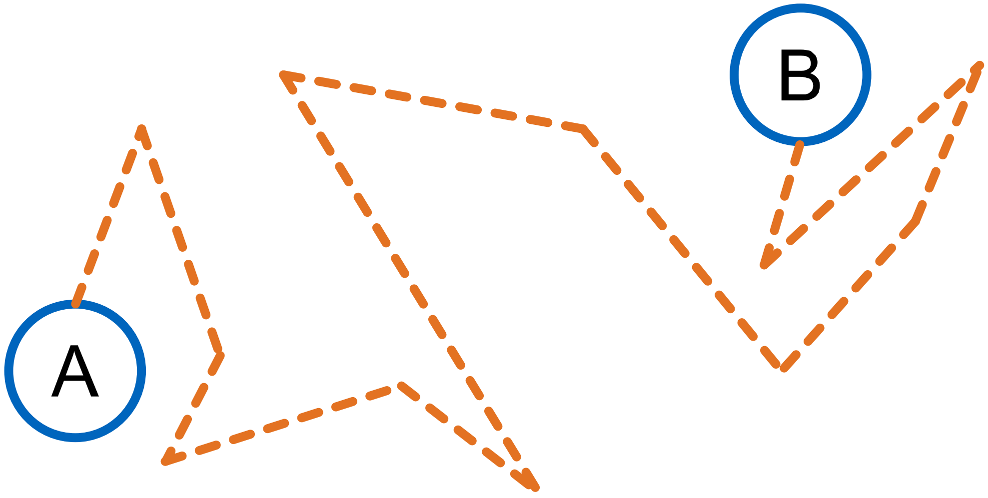
Ties are resolved lexically

Applicable for example graph

Algorithms – British Museum
Description

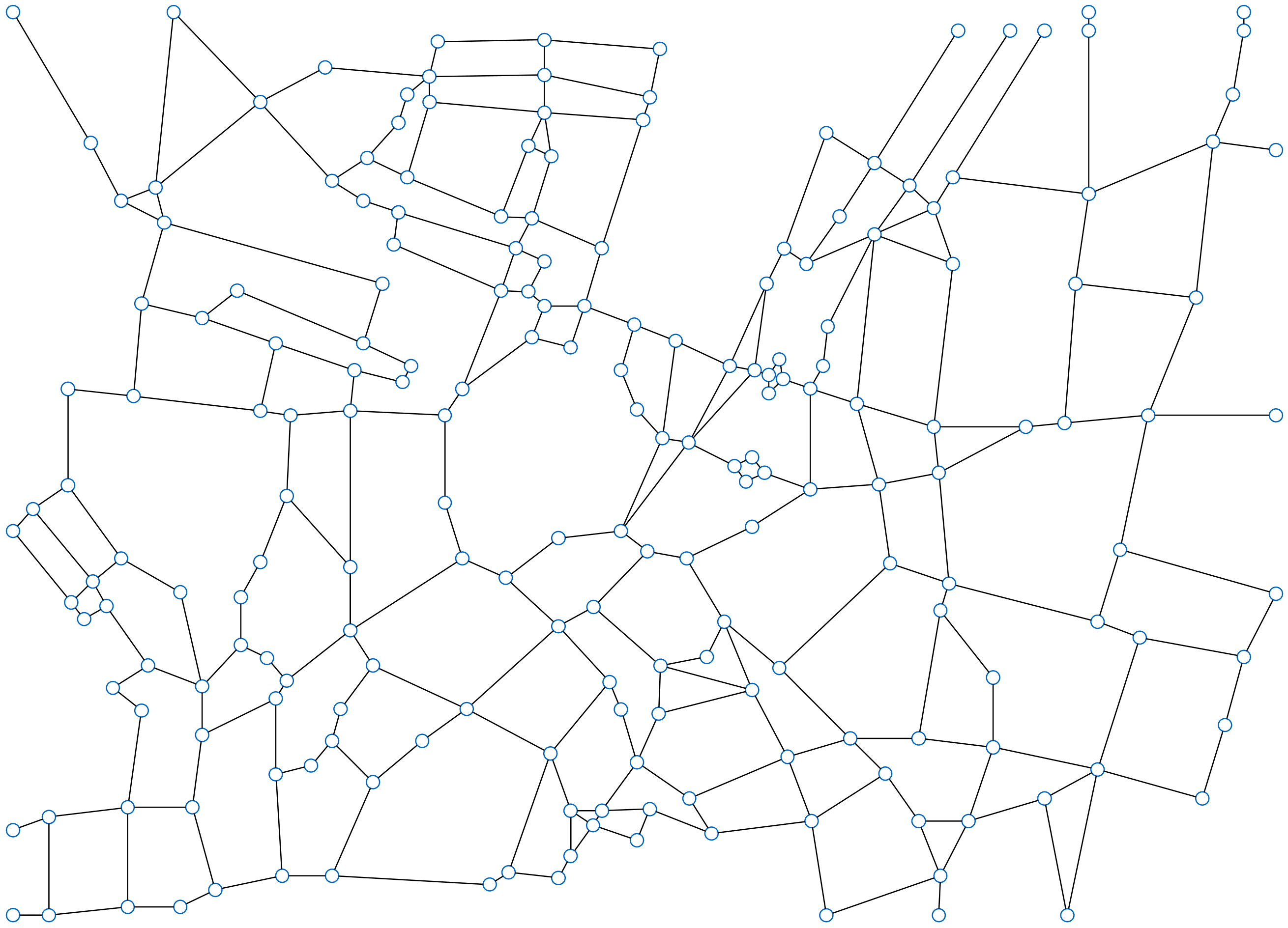
Algorithm:

- Randomly append Nodes to path while not stuck.
- If destination was reached: Success
- If stuck, start over



Algorithms

Munich Demo Graph

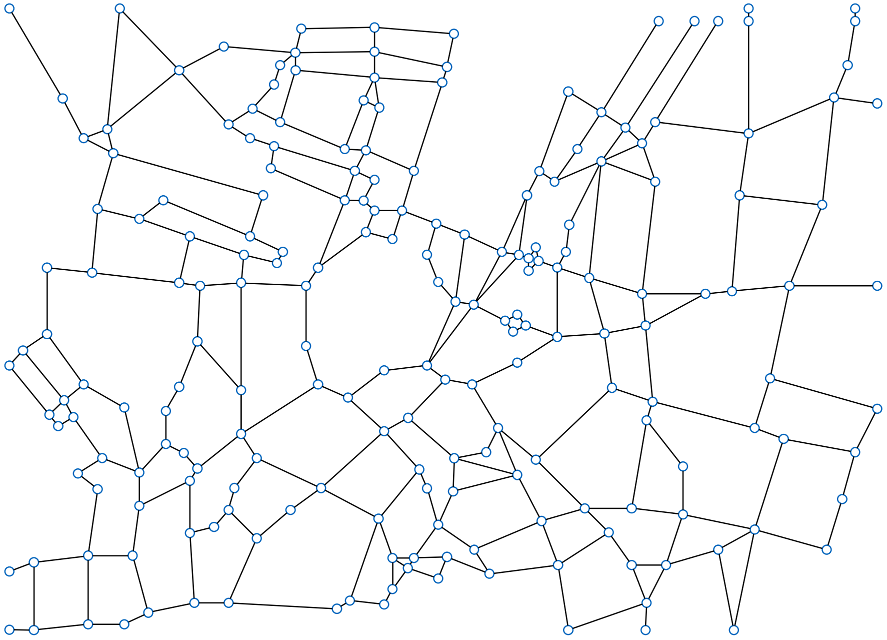


Algorithms

Munich Demo Graph

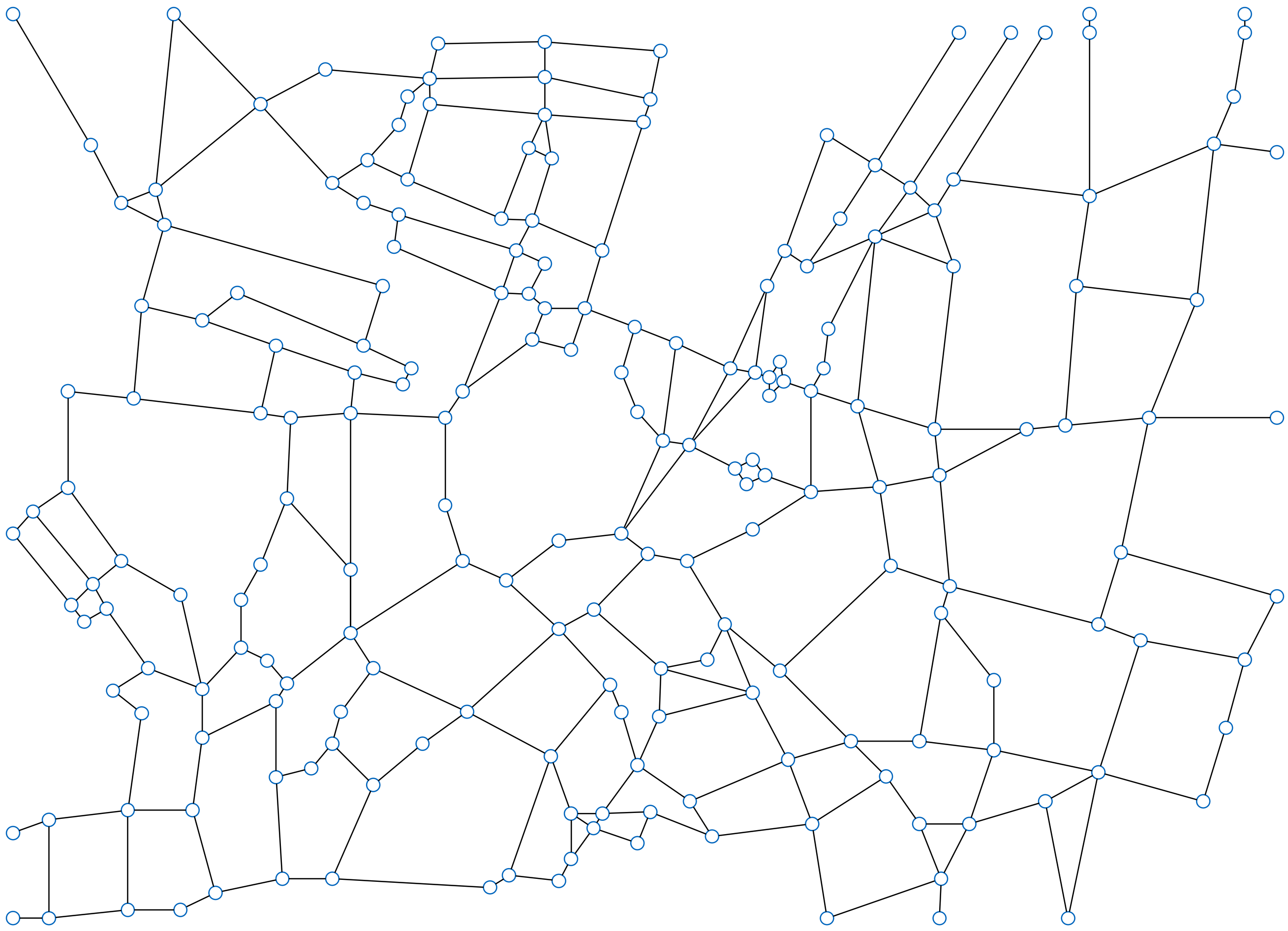
Properties:

1. Bidirectional Edges
2. Static Weights
3. Weights = Distances
4. Main Roads
5. No Turn Restrictions
6. Basic Road Geometry



Algorithms – British Museum

Munich Demo



Algorithms – Depth First
Description

Algorithm:
From „Artificial Intelligence“ by Patrick H. Winston

- Form a one-element queue consisting of a zero-length path that contains only the root node

---

- Until the first path in the queue terminates at the goal node or the queue is empty,
    - Remove the first path from the queue; create new paths by extending the first path to all the neighbors of the terminal node
    - Reject all new paths with loops
    - Add the new paths, if any, to the front of the queue

- If the goal node is found, announce success; otherwise announce failure

Algorithms – Depth First
Description

Algorithm:
From „Artificial Intelligence“ by Patrick H. Winston

- Form a one-element queue consisting of a zero-length path that contains only the root node

---

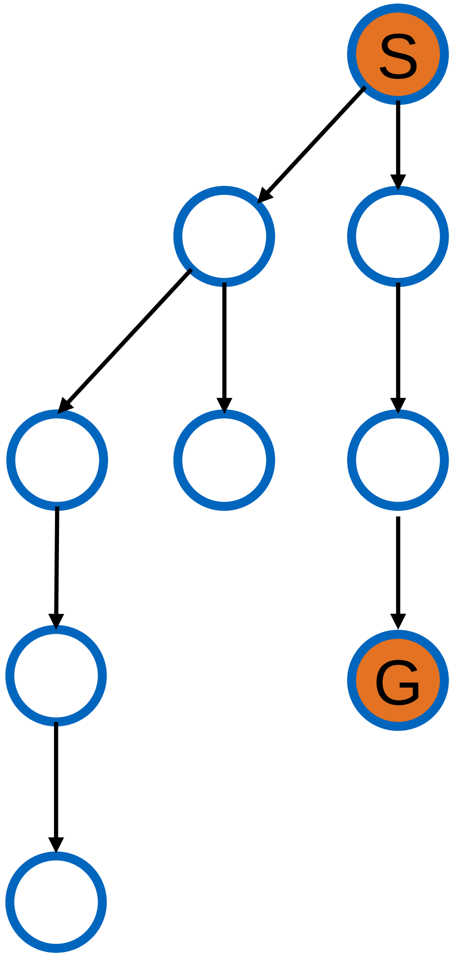
- Until the first path in the queu… …he queue is empty,
    - Remove … …ue, create new paths by extending the first path t… …ghbors of the terminal node
    - Reject all new paths with loops
    - Add the new paths, if any, to the front of the queue

- If the goal node is found, announce success; otherwise announce failure

Diving into the Search Tree

Algorithms – Depth First

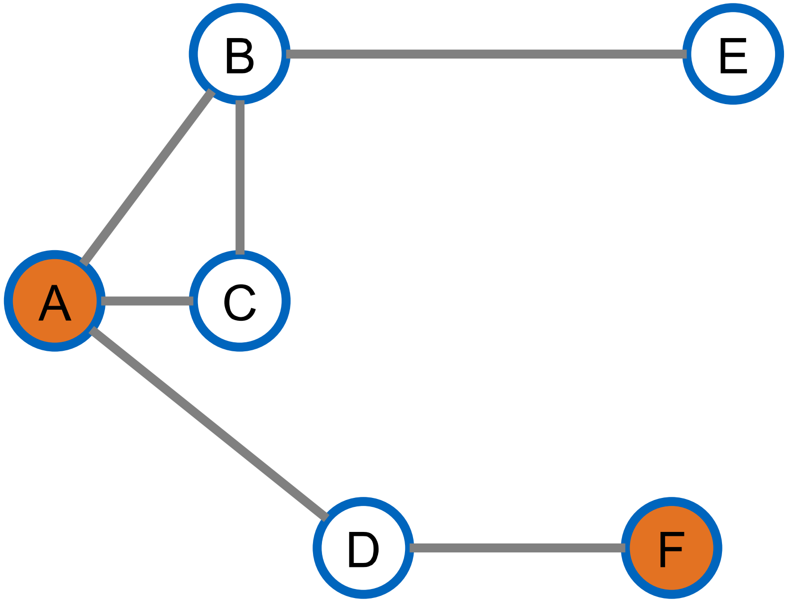
Characteristic Search Tree



narrow and deep

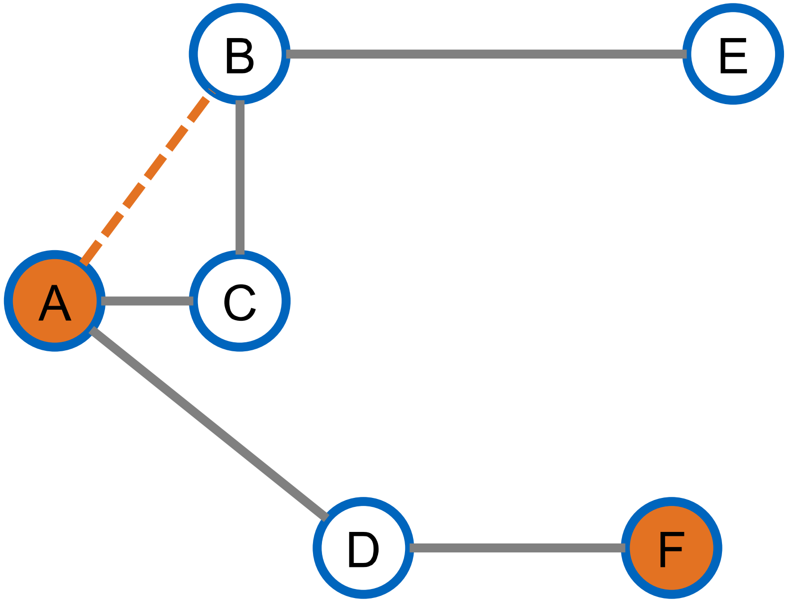
Algorithms – Depth First

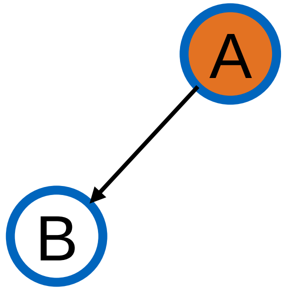
Pathfinding Example Graph



Algorithms – Depth First

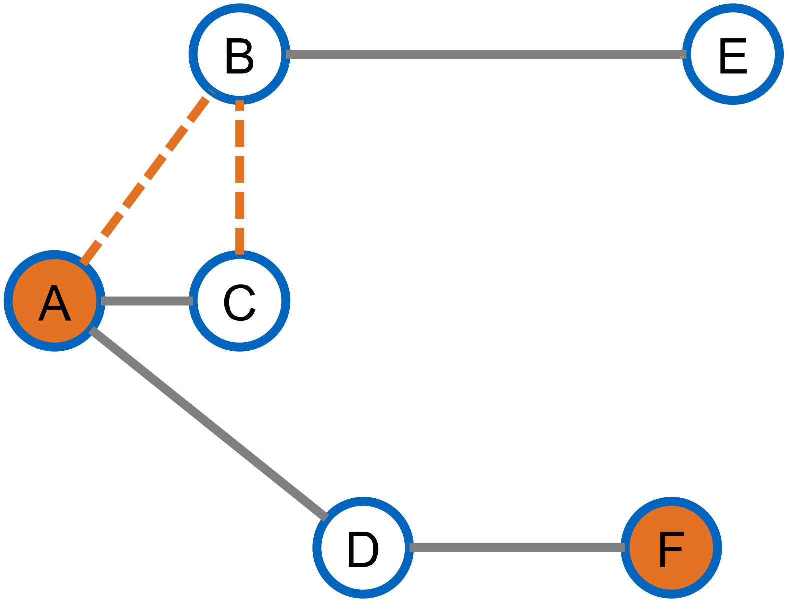
Pathfinding Example Graph

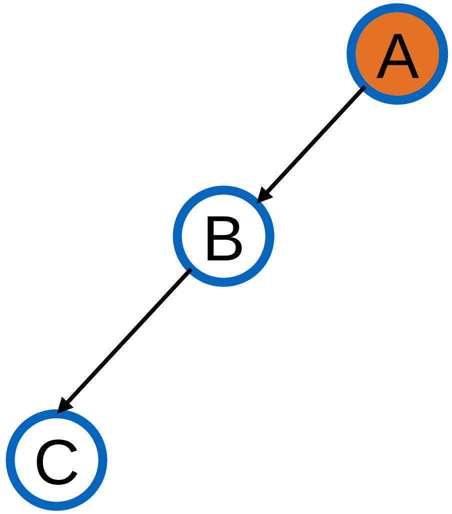




Algorithms – Depth First

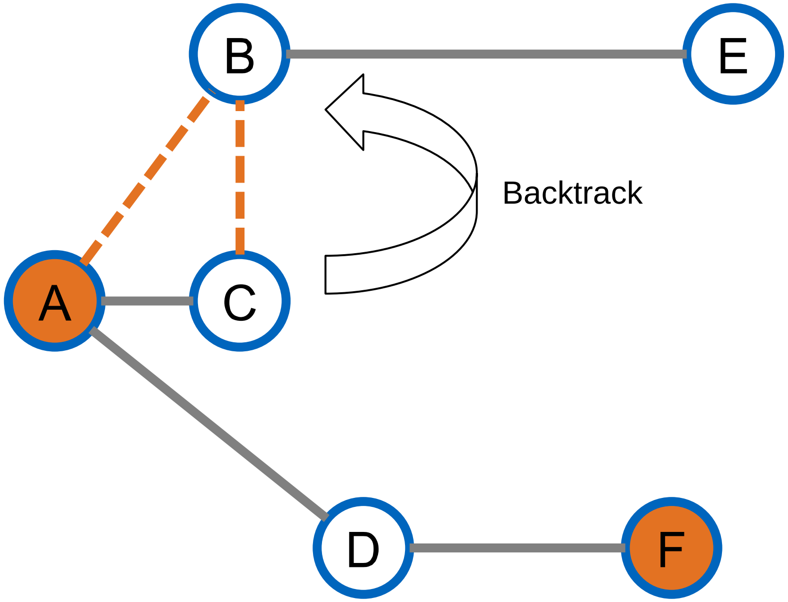
Pathfinding Example Graph

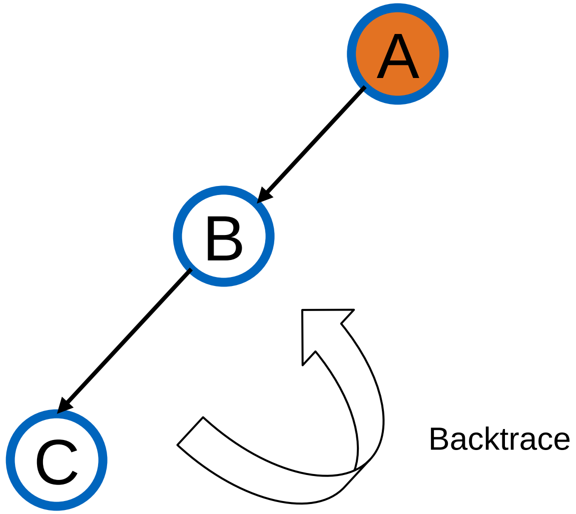




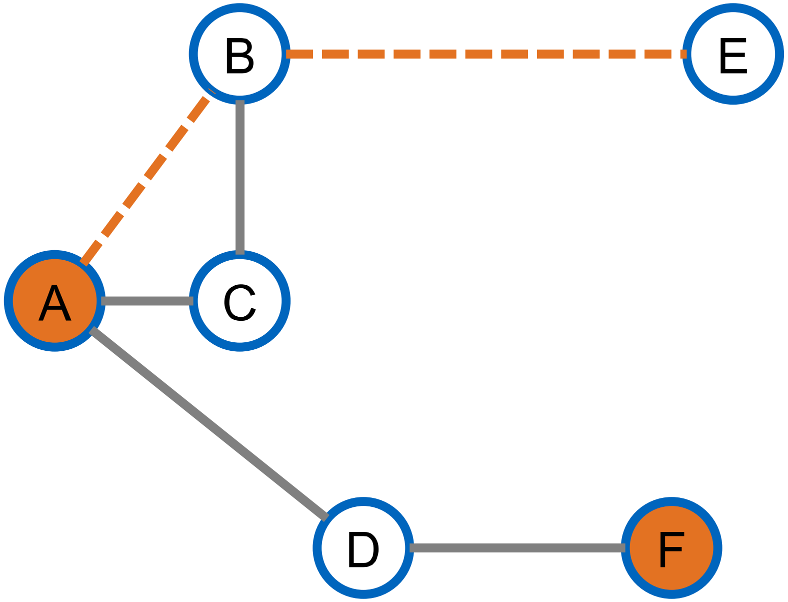
Algorithms – Depth First

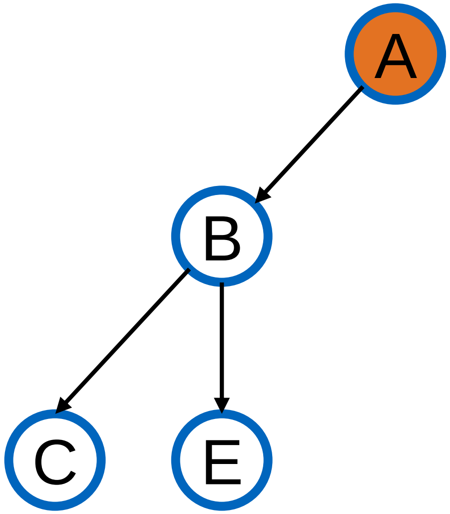
Pathfinding Example Graph





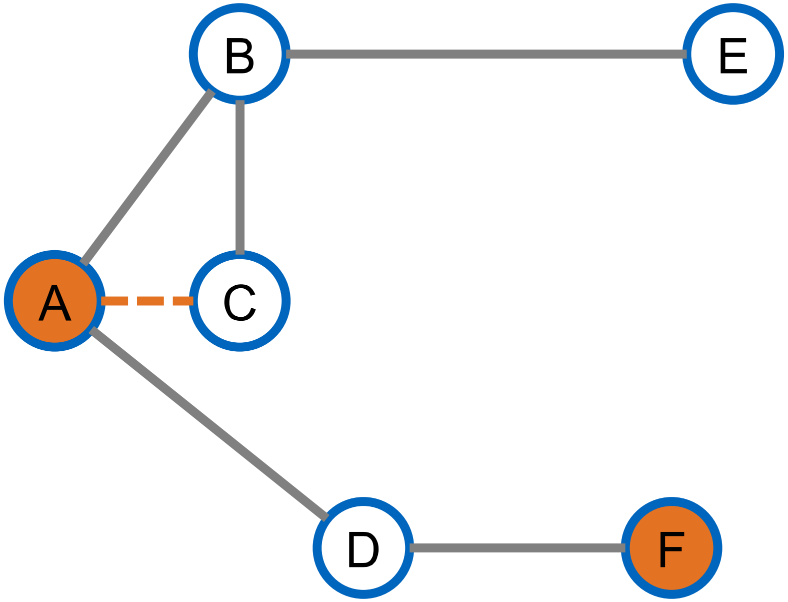
Algorithms – Depth First
Pathfinding Example Graph

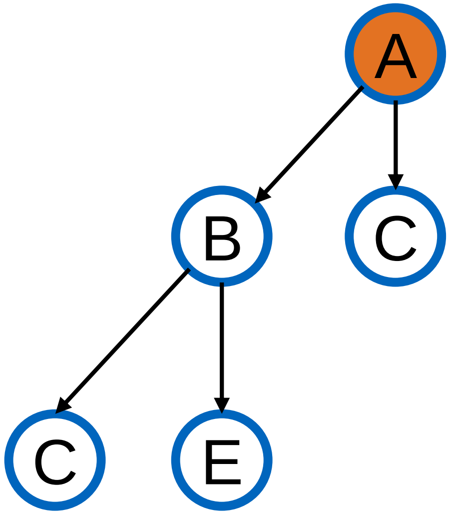




Algorithms – Depth First

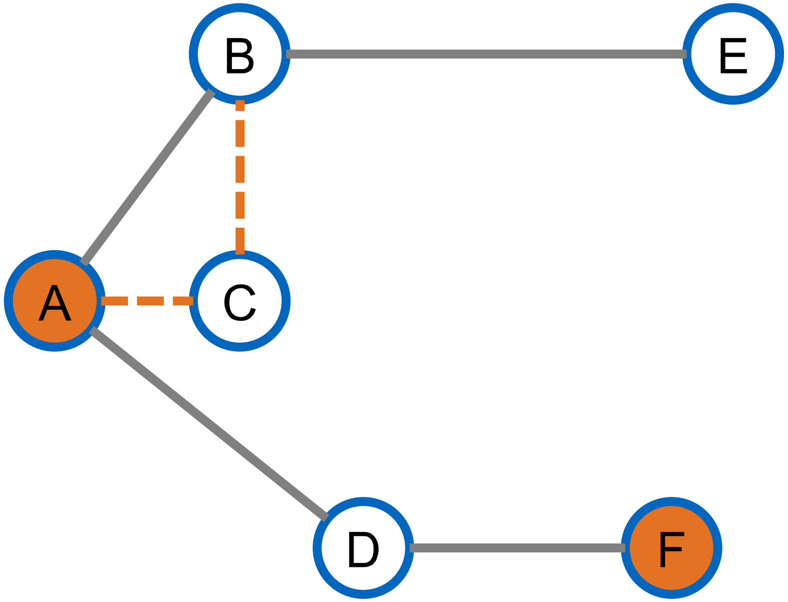
Pathfinding Example Graph

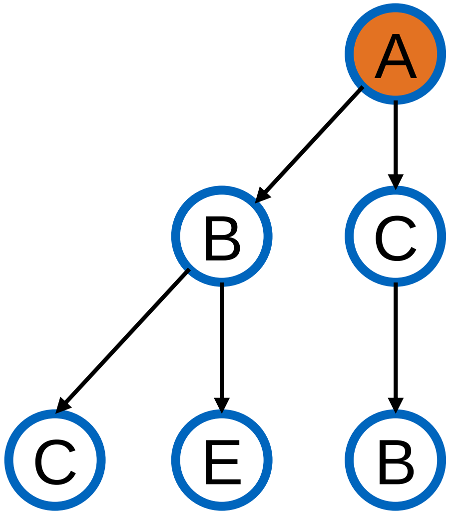




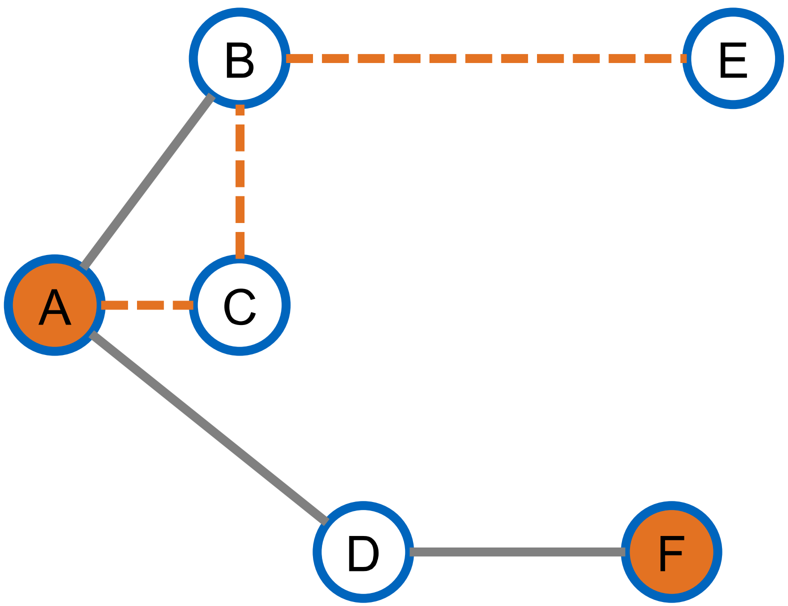
Algorithms – Depth First

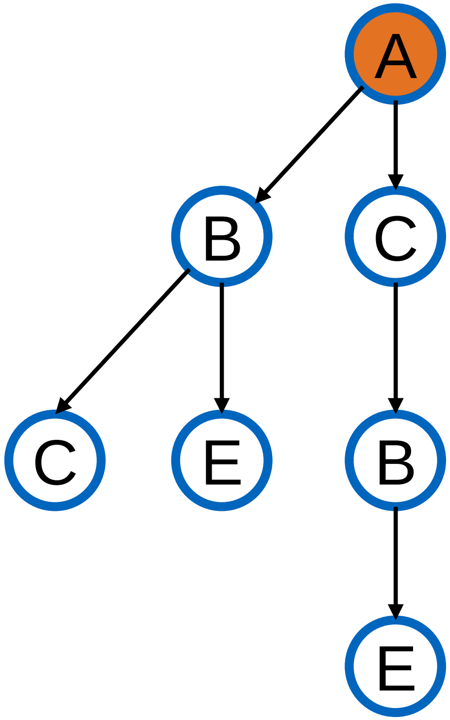
Pathfinding Example Graph





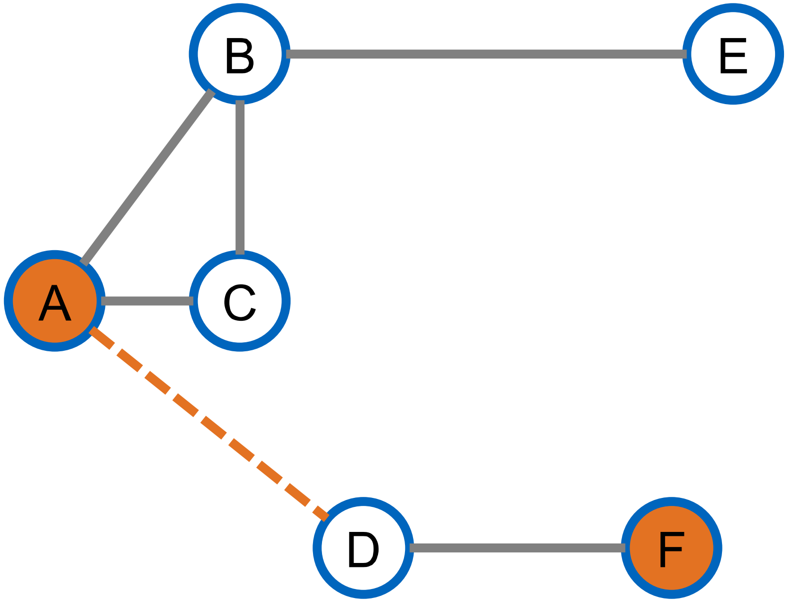
Algorithms – Depth First
Pathfinding Example Graph

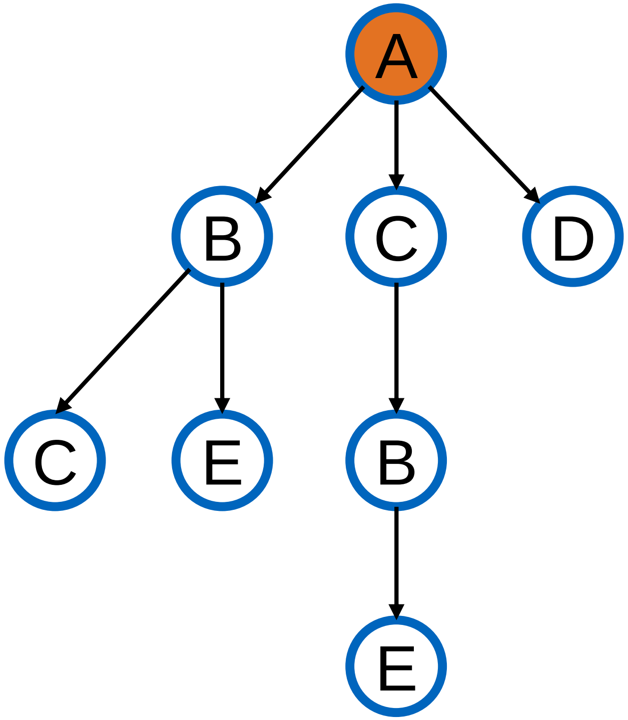




Algorithms – Depth First

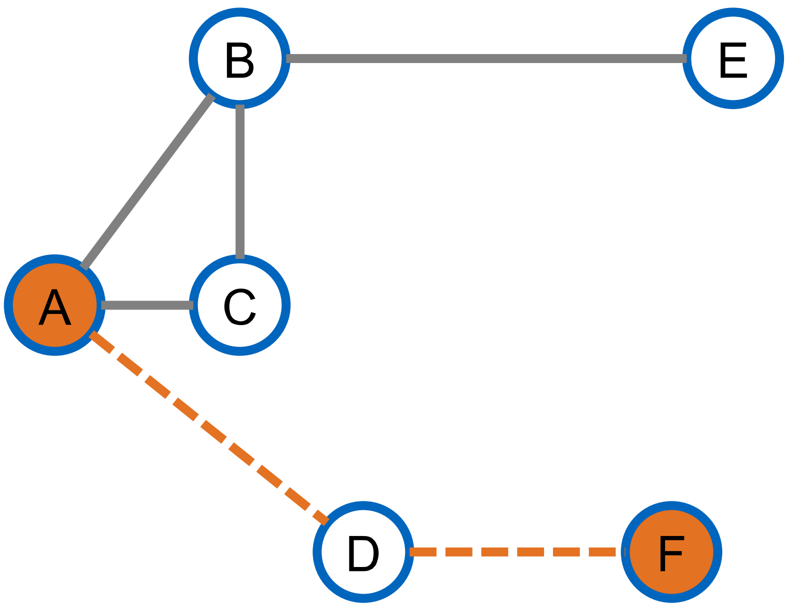
Pathfinding Example Graph

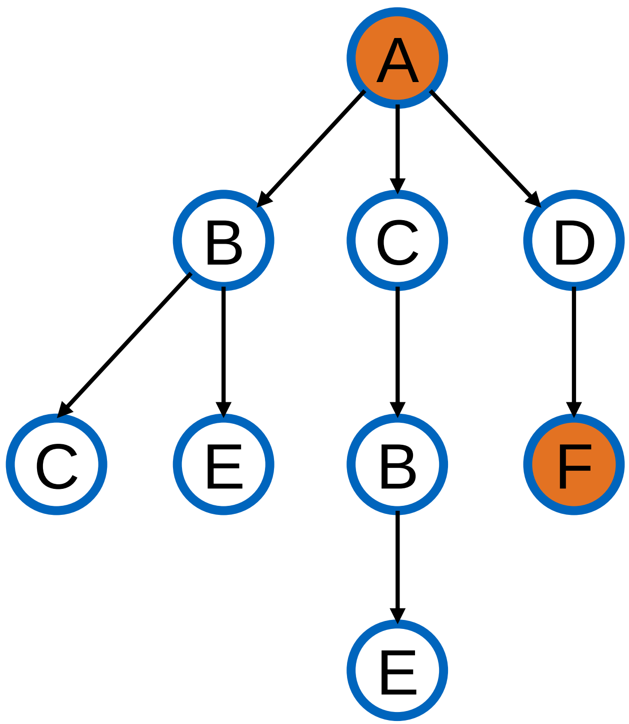




Algorithms – Depth First

Pathfinding Example Graph





Algorithms – Breadth First
Description

Algorithm:
From „Artificial Intelligence“ by Patrick H. Winston

- Form a one-element queue consisting of a zero-length path that contains only the root node

---

- Until the first path in the queue terminates at the goal node or the queue is empty,
    - Remove the first path from the queue; create new paths by extending the first path to all the neighbors of the terminal node
    - Reject all new paths with loops
    - Add the new paths, if any, to the back of the queue

- If the goal node is found, announce success; otherwise announce failure

Algorithms – Breadth First
Description

Algorithm:
From „Artificial Intelligence“ by Patrick H. Winston

- Form a one-element queue consisting of a zero-length path that contains only the root node

---

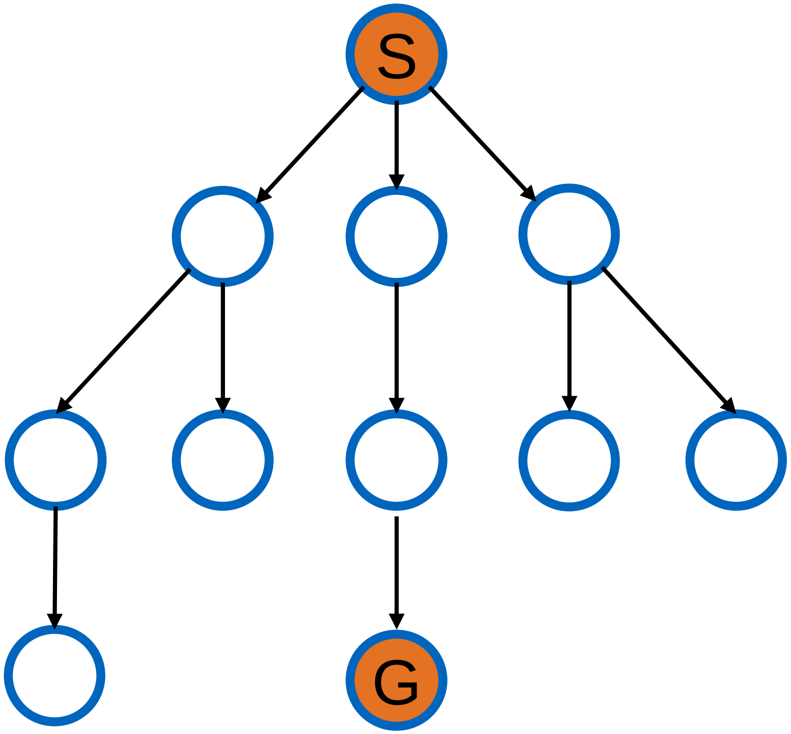
- Until the first path in the qu[illegible]he queue is empty,
  - Remove [illegible]ue, create new paths by extending the first path t[illegible]ghbors of the terminal node
  - Reject all new paths with loops
  - Add the new paths, if any, to the back of the queue

- If the goal node is found, announce success; otherwise announce failure

Scanning the Search Tree
Level by Level

Algorithms – Breadth First

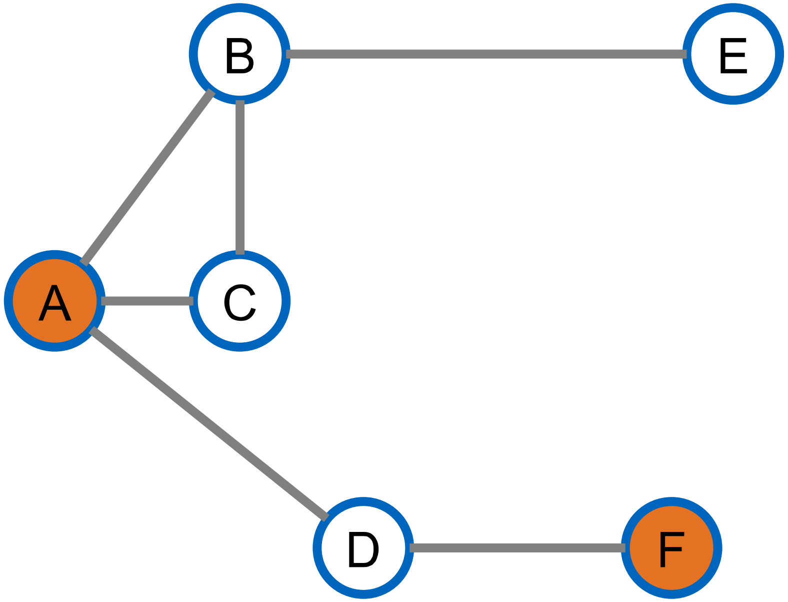
Characteristic Search Tree



wide and shallow

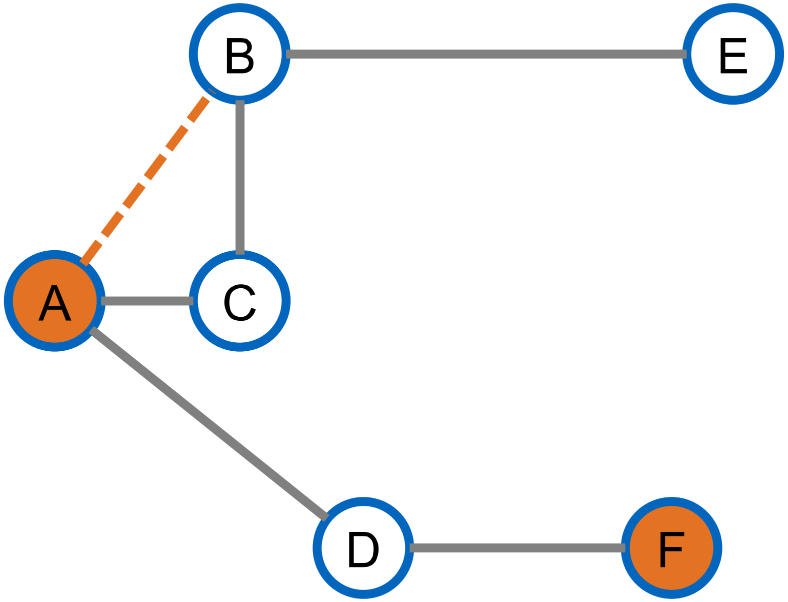
Algorithms – Breadth First

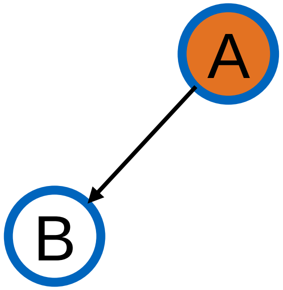
Pathfinding Example Graph



Algorithms – Breadth First

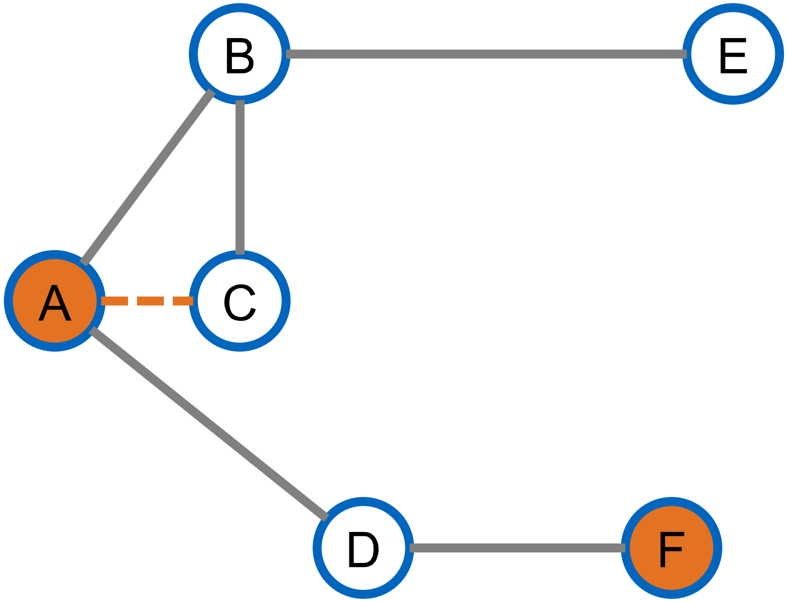
Pathfinding Example Graph

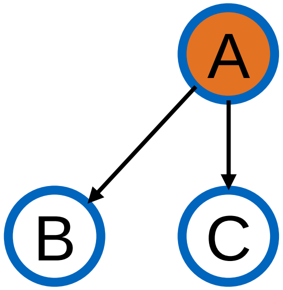




Algorithms – Breadth First

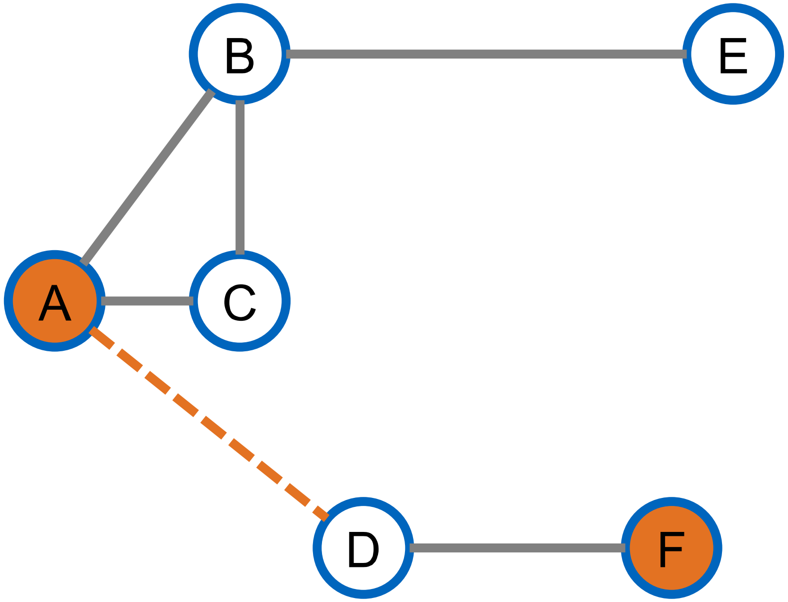
Pathfinding Example Graph

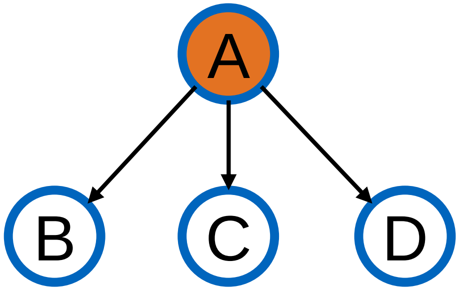




Algorithms – Breadth First

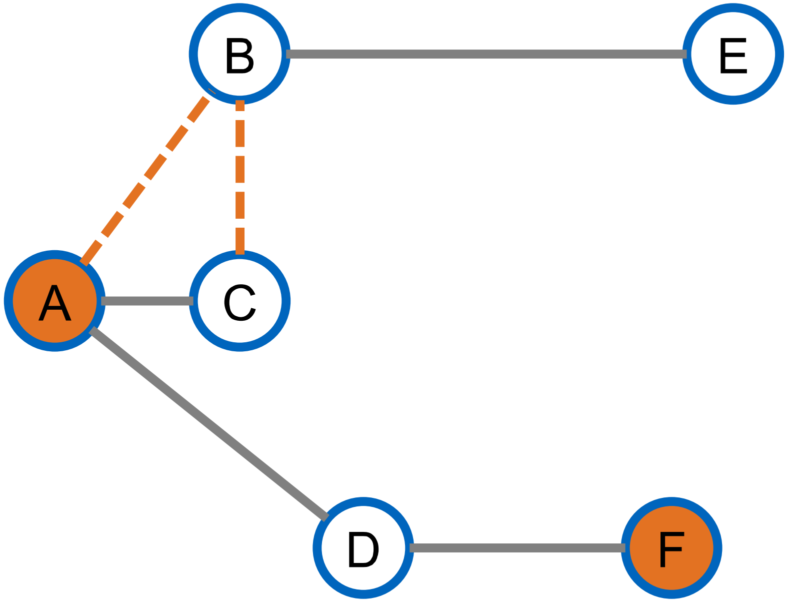
Pathfinding Example Graph

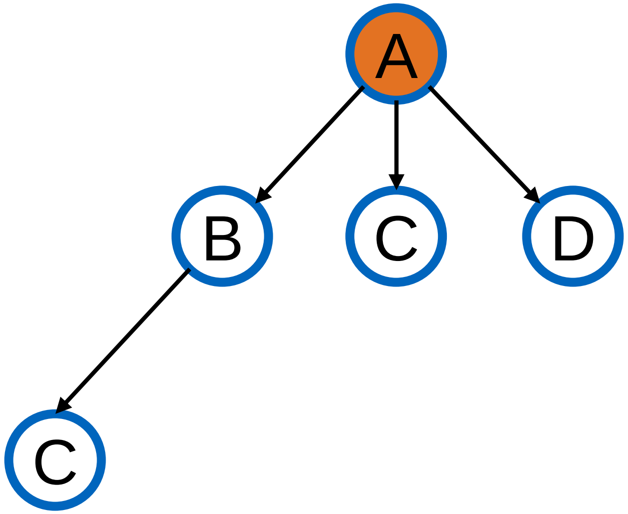




Algorithms – Breadth First

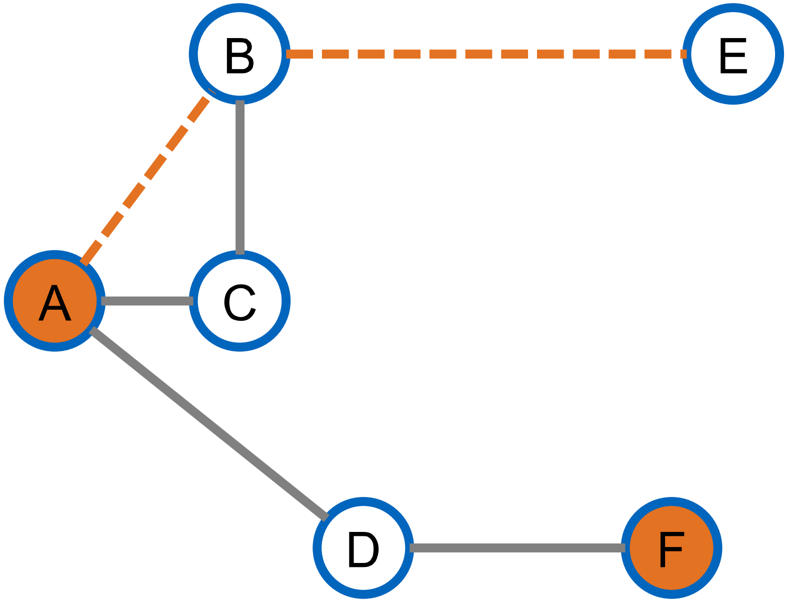
Pathfinding Example Graph

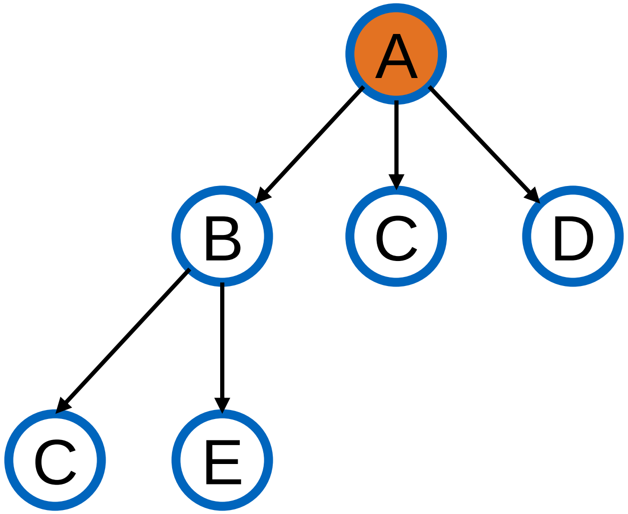




Algorithms – Breadth First

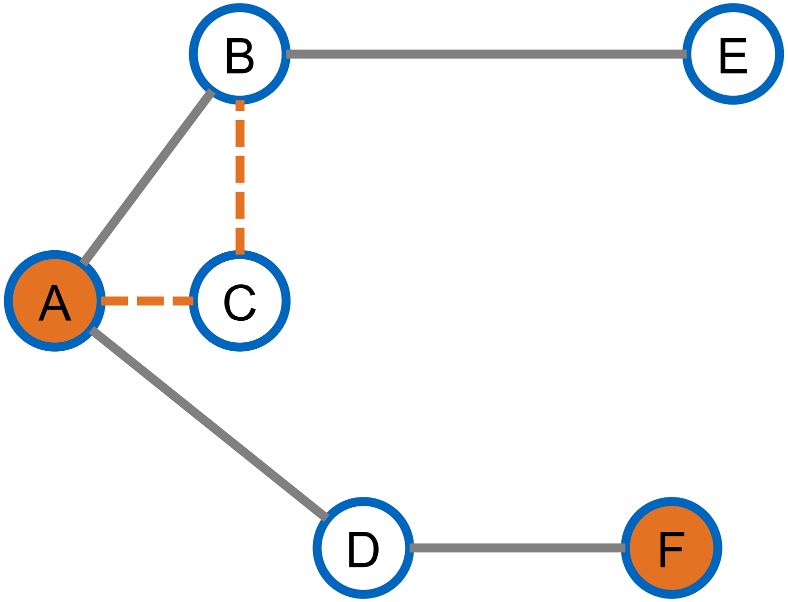
Pathfinding Example Graph

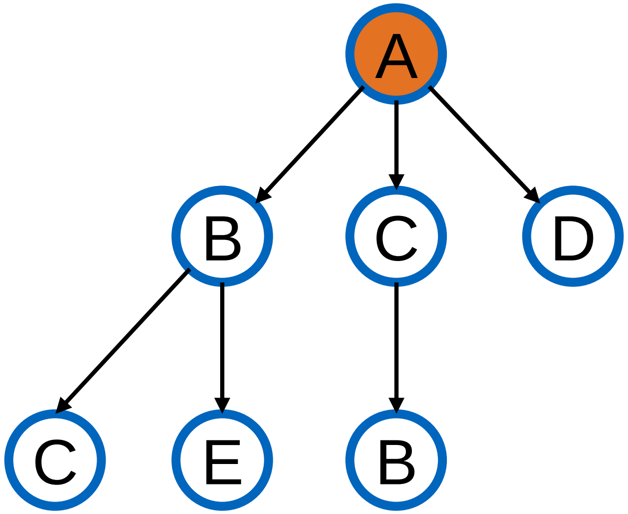




Algorithms – Breadth First

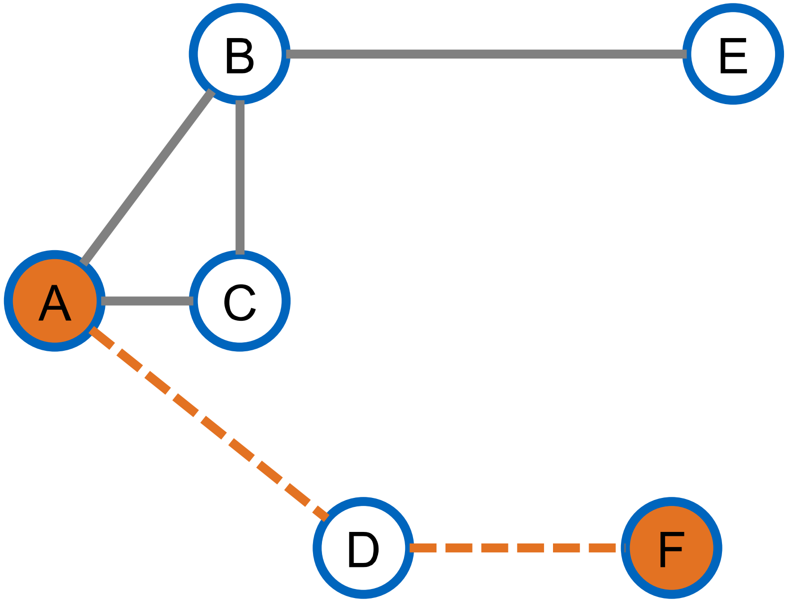
Pathfinding Example Graph

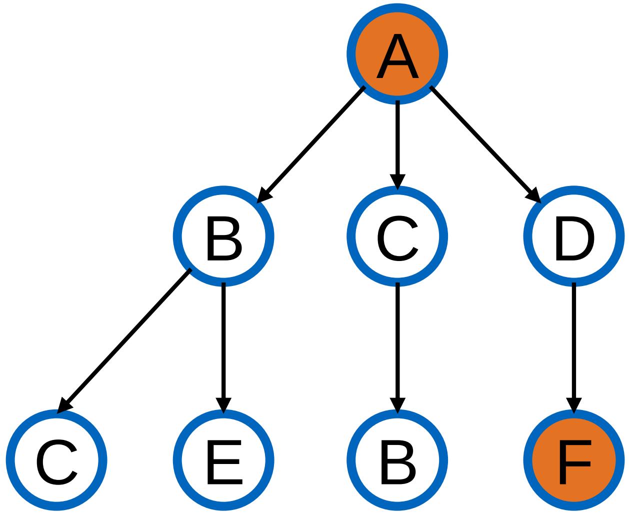




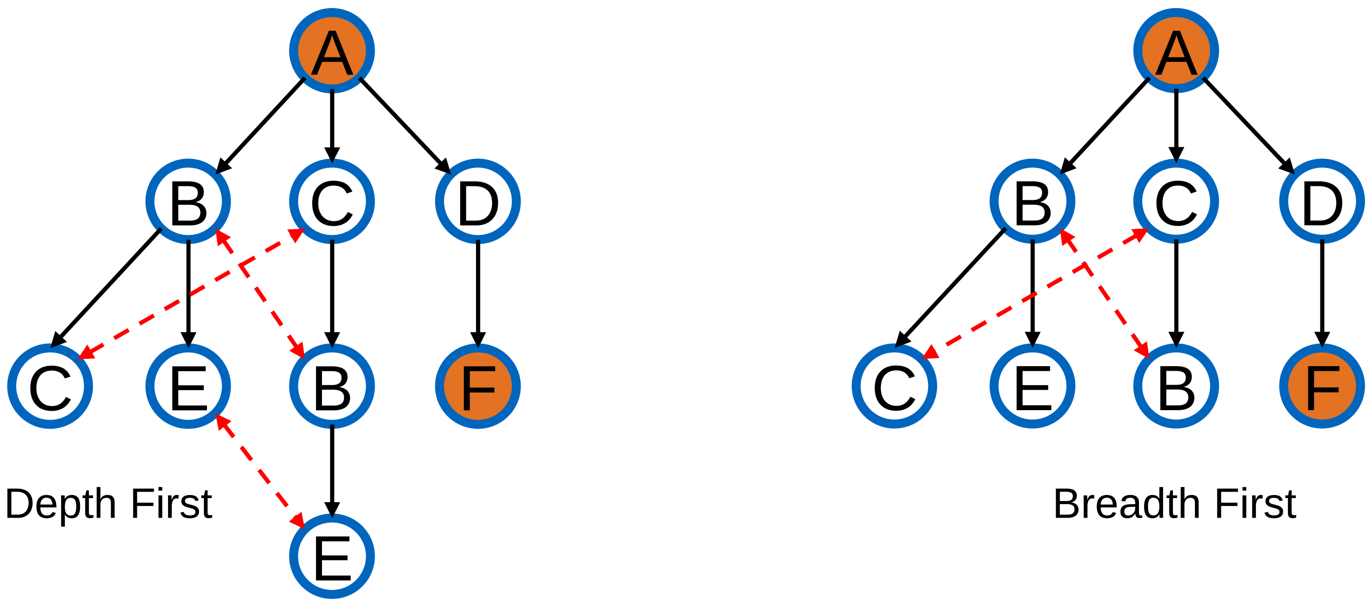
Algorithms – Breadth First

Pathfinding Example Graph





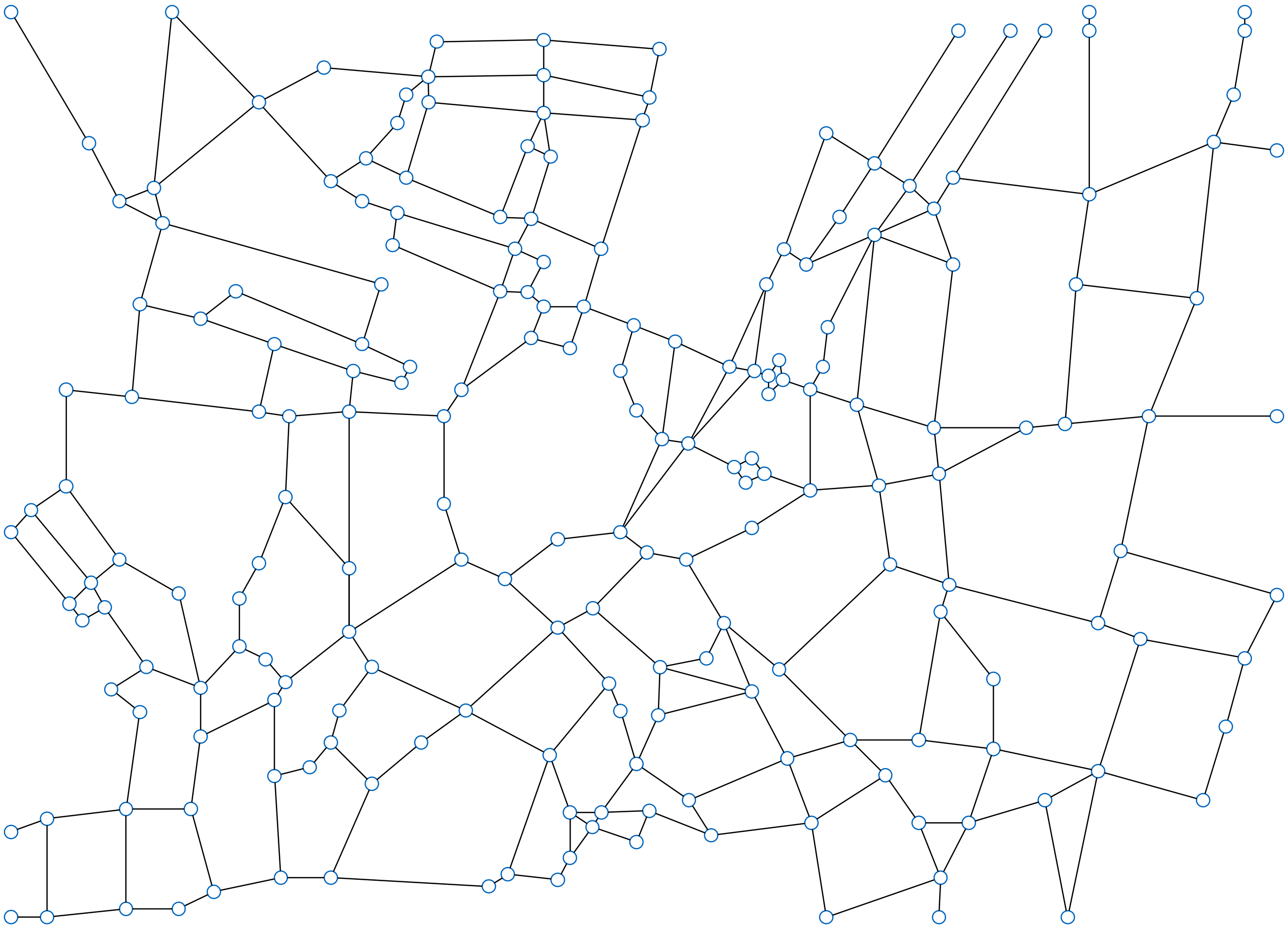
Algorithms – Enhancements
Shortcoming of Basic Algorithms - Revisitation



Prevent Revisitation

Algorithms – Breadth First Enhanced

Munich Demo



Algorithms – Enhancements
Informedness

A search algorithm is informed if it employs additional information about the problem that has the potential of guiding the search toward its goal.

Algorithms – Best First
Description

Algorithm:

- Form a one-element queue consisting of a zero-length path that contains only the root node

---

- Until the first path in the queue terminates at the goal node or the queue is empty,
    - Remove the first path from the queue; create new paths by extending the first path to all the unvisited neighbors of the terminal node
    - Reject all new paths with loops
    - Add the new paths, if any, to the queue
    - Sort all paths by an heuristic function evaluated at their terminal nodes

---

- If the goal node is found, announce success; otherwise announce failure

![TUM logo omitted]

Algorithms – Best First

Description

Algorithm:

- Form a one-element queue consisting of a zero-length path that contains only the root node

---

- Until the first path in the queue terminates … the queue is empty,
    - Remove … new paths by extending the first path … of the terminal node
    - Reject all new paths with loops
    - Add the new paths, if any, to the queue
    - Sort all paths by an heuristic function evaluated at their terminal nodes

---

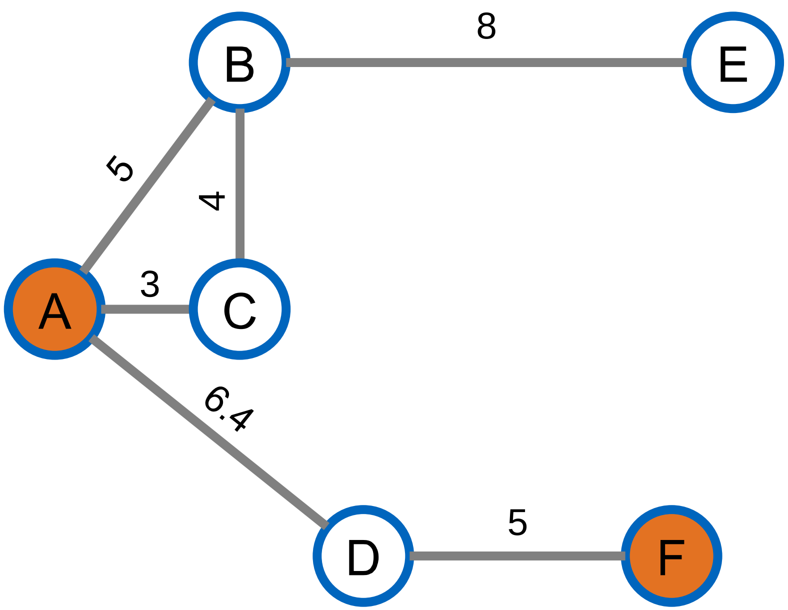
- If the goal node is found, announce success; otherwise announce failure

Always Take the Seemingly Best Step

Algorithms – Best First

Pathfinding Example Graph

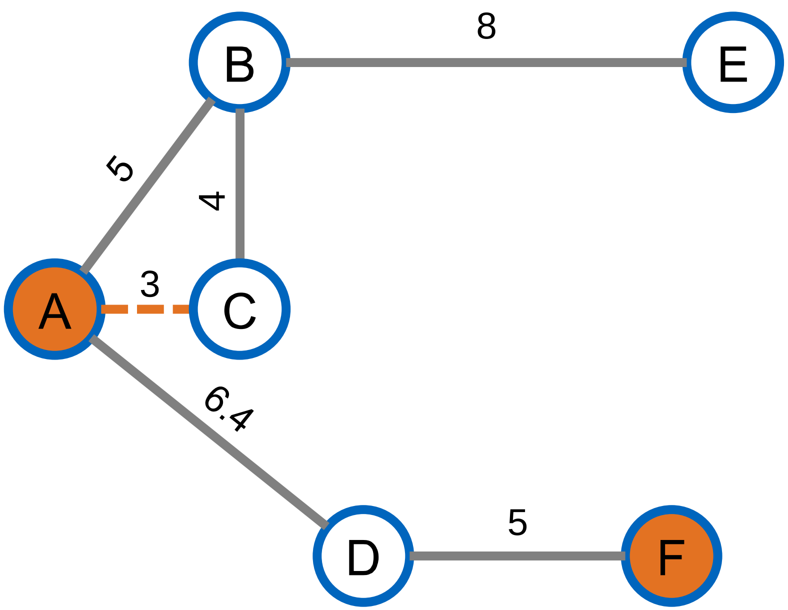
Example: Shortest Step Heuristic

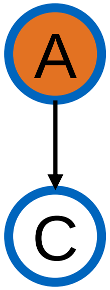


Algorithms – Best First

Pathfinding Example Graph

Example: Shortest Step Heuristic

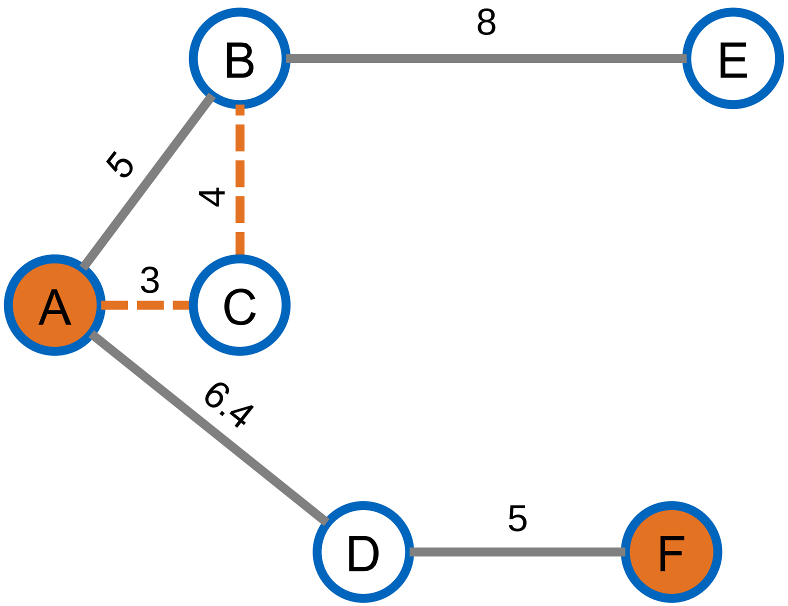


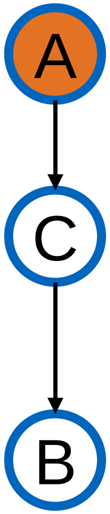


Algorithms – Best First

Pathfinding Example Graph

Example: Shortest Step Heuristic

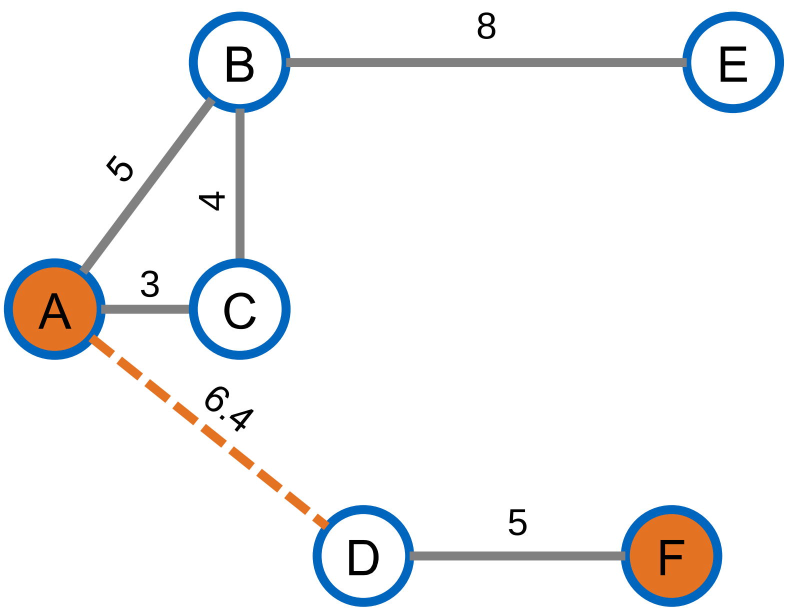


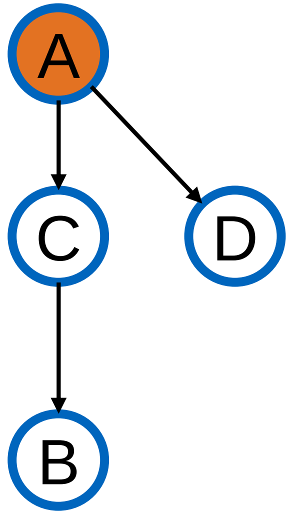


Algorithms – Best First

Pathfinding Example Graph

Example: Shortest Step Heuristic

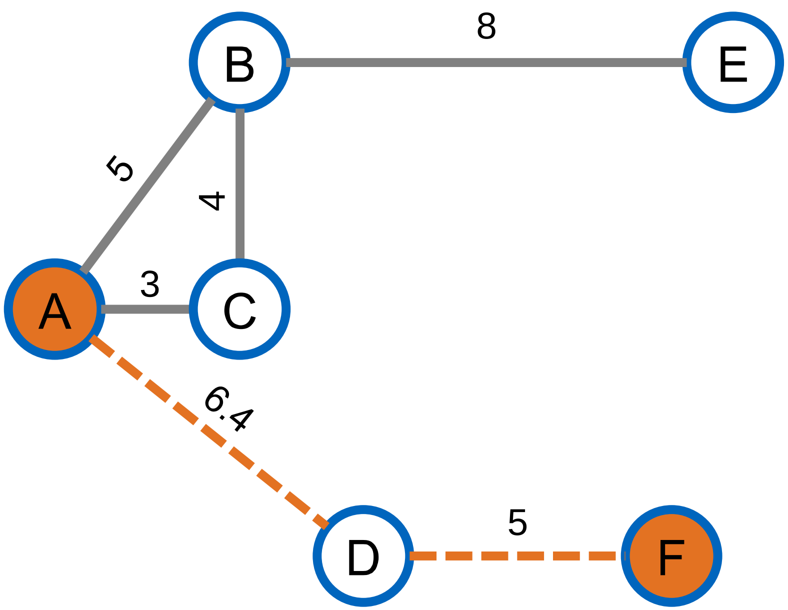


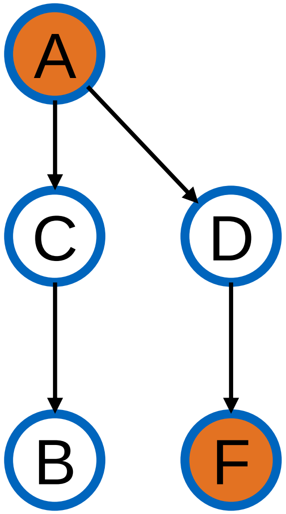


Algorithms – Best First

Pathfinding Example Graph

Example: Shortest Step Heuristic

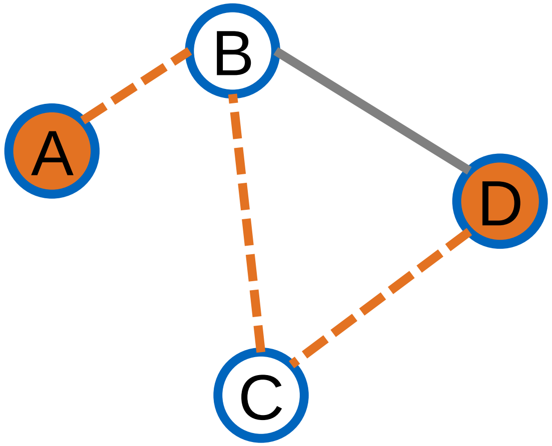




Pathfinding: Problem Formulations

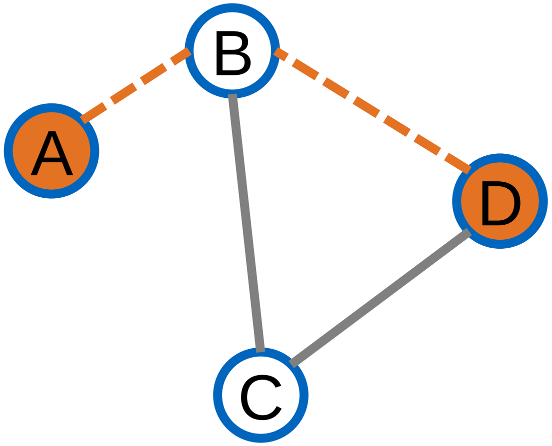
A : Find a Path

- Depth First
- Breadth First
- Best First



B : Find an optimal Path

- Dijkstra
- A*



Algorithms – Dijkstra
Description

Algorithm:

- Form a one-element queue consisting of a zero-length path that contains only the root node

Core Idea: Next Slide

- If the goal node is found, announce success; otherwise announce failure

Algorithms – Dijkstra
Description

Algorithm – Core Idea:

- Until the first path in the queue terminates at the goal node or the queue is empty,
  - Remove the first path from the queue; create new paths by extending the first path to all the neighbors of the terminal node
  - Reject all new paths with loops
  - If two or more paths reach a common node, delete all those paths except the one that reaches the common node with the minimum cost.
  - Add the new paths, if any, to the queue
  - Sort all paths by their accumulated costs

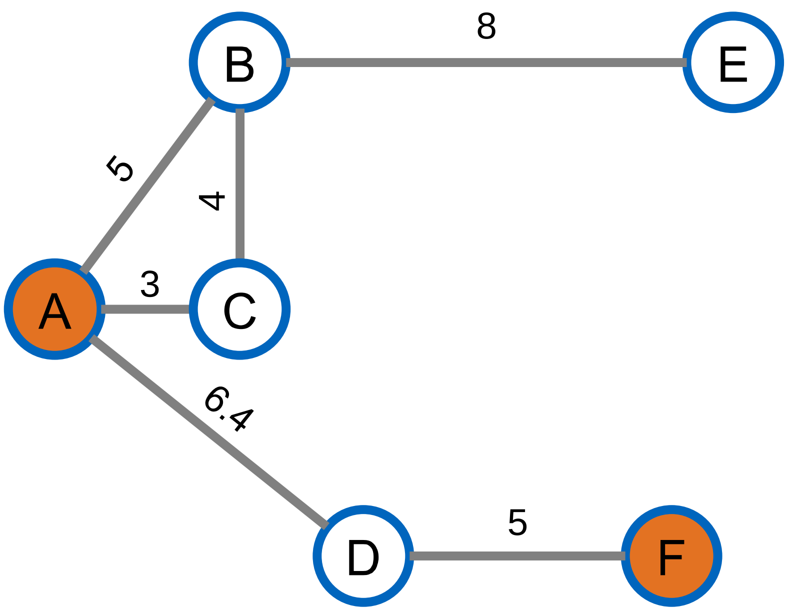
Algorithms – Dijkstra
Description

Algorithm – Core Idea:

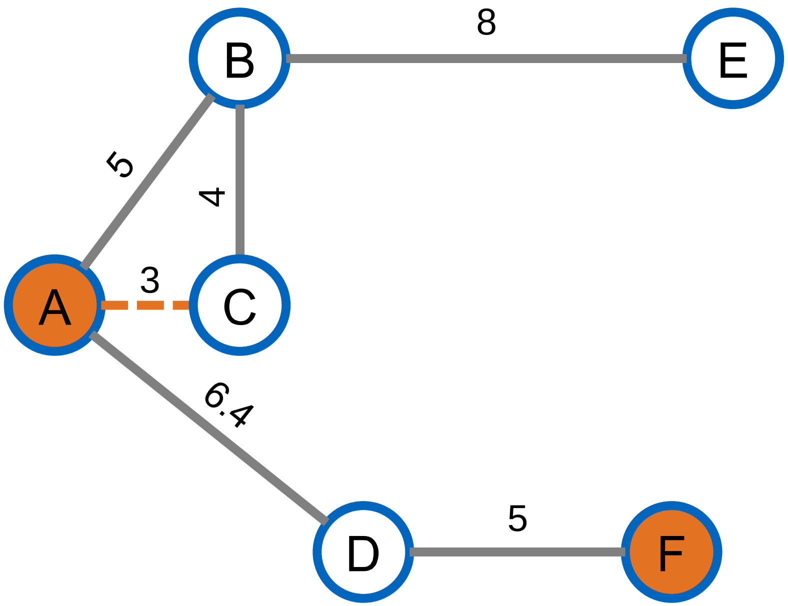
- Until the first path in the queue terminates at the goal node or the queue is empty,
  - Remove the first path from the queu… …y extending the first path to all the neigh…
  - Reject a…
  - If two or m… …each a common node, delete all those paths except the one that reaches the common node with the minimum cost.
  - Add the new paths, if any, to the queue
  - Sort all paths by their accumulated costs

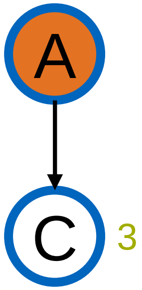
Explore all Shortest Paths until Goal is Reached

Algorithms – Dijkstra
Pathfinding Example Graph

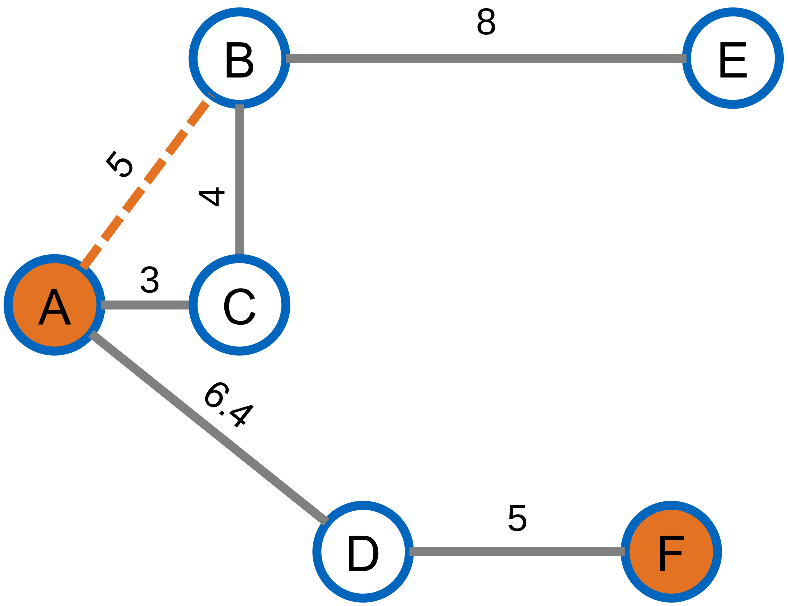


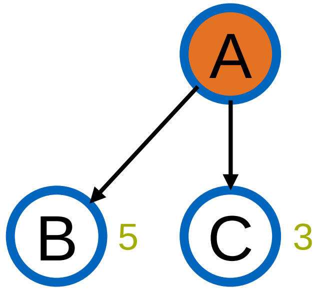
Algorithms – Dijkstra
Pathfinding Example Graph





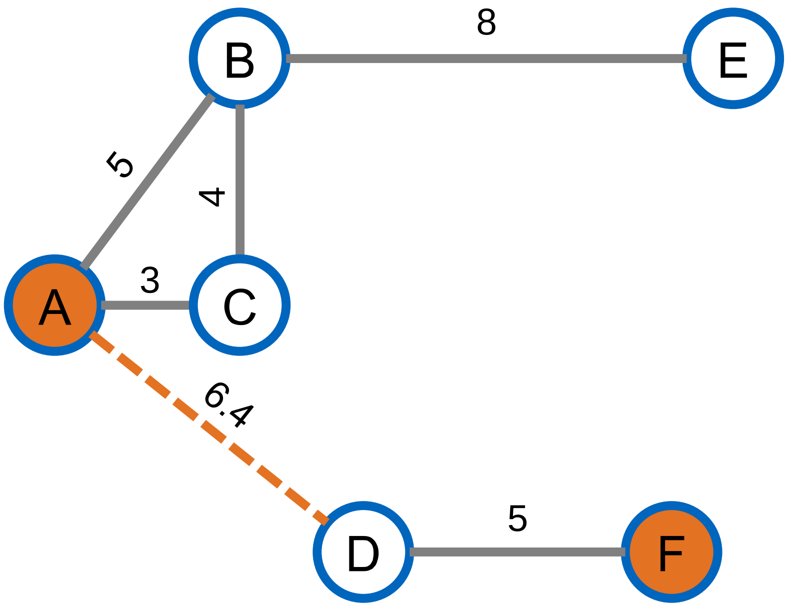
Algorithms – Dijkstra
Pathfinding Example Graph

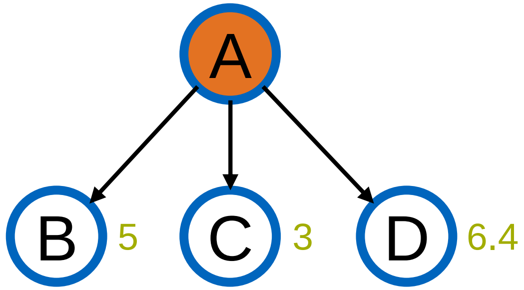




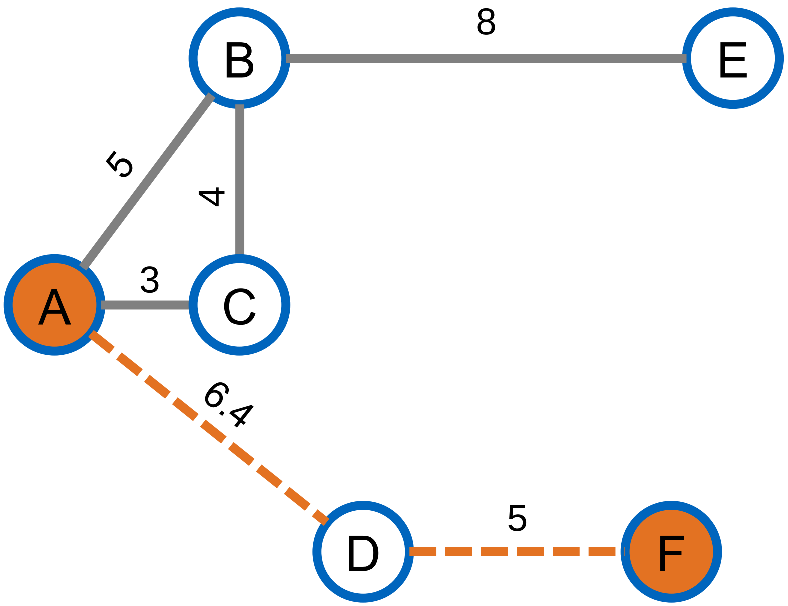
Algorithms – Dijkstra

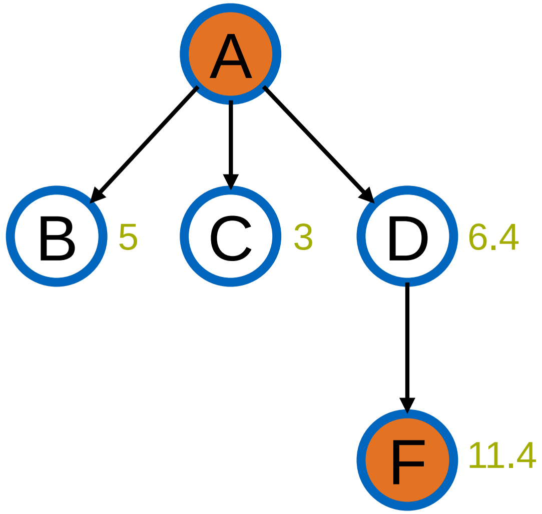
Pathfinding Example Graph





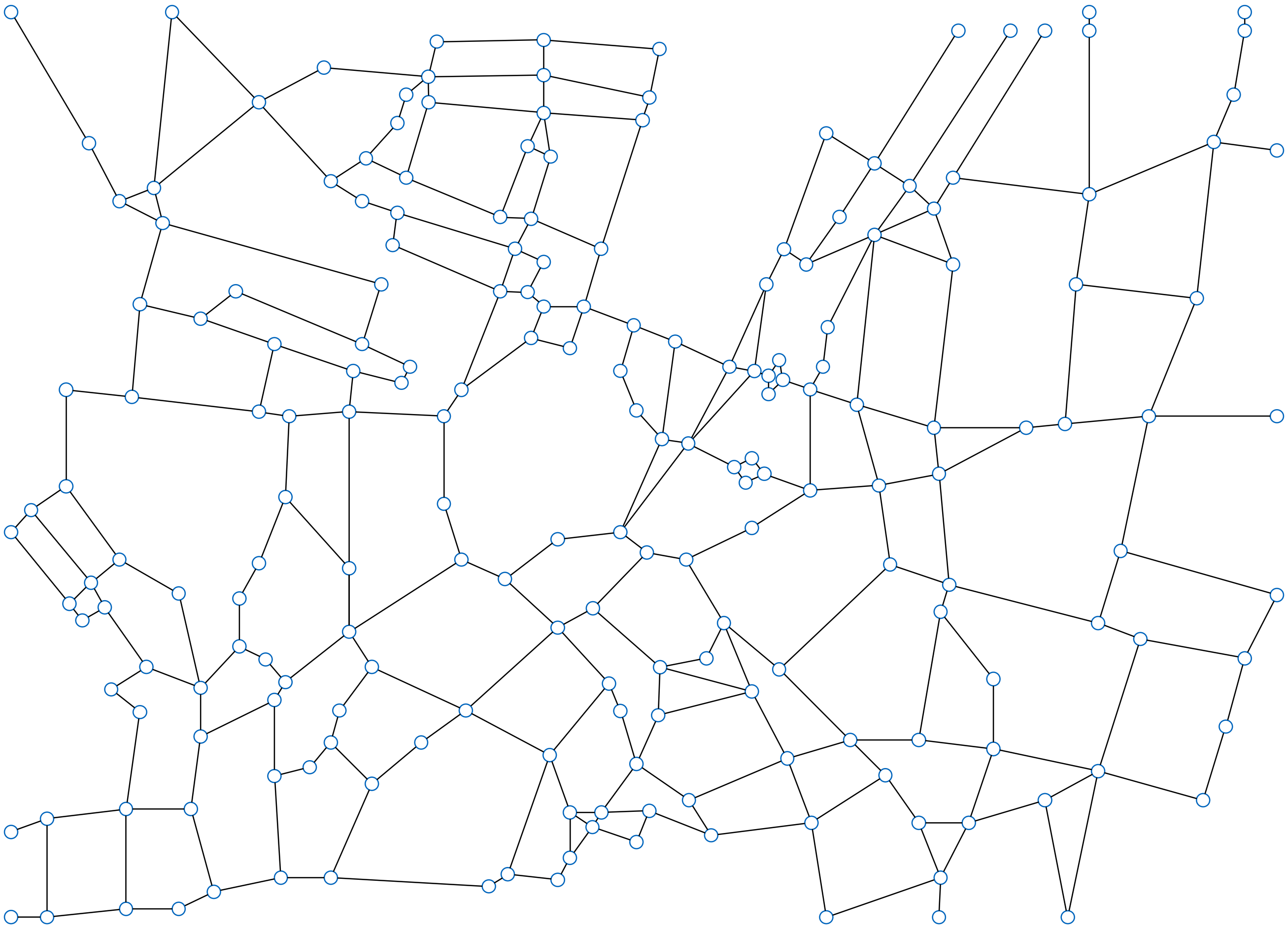
Algorithms – Dijkstra
Pathfinding Example Graph





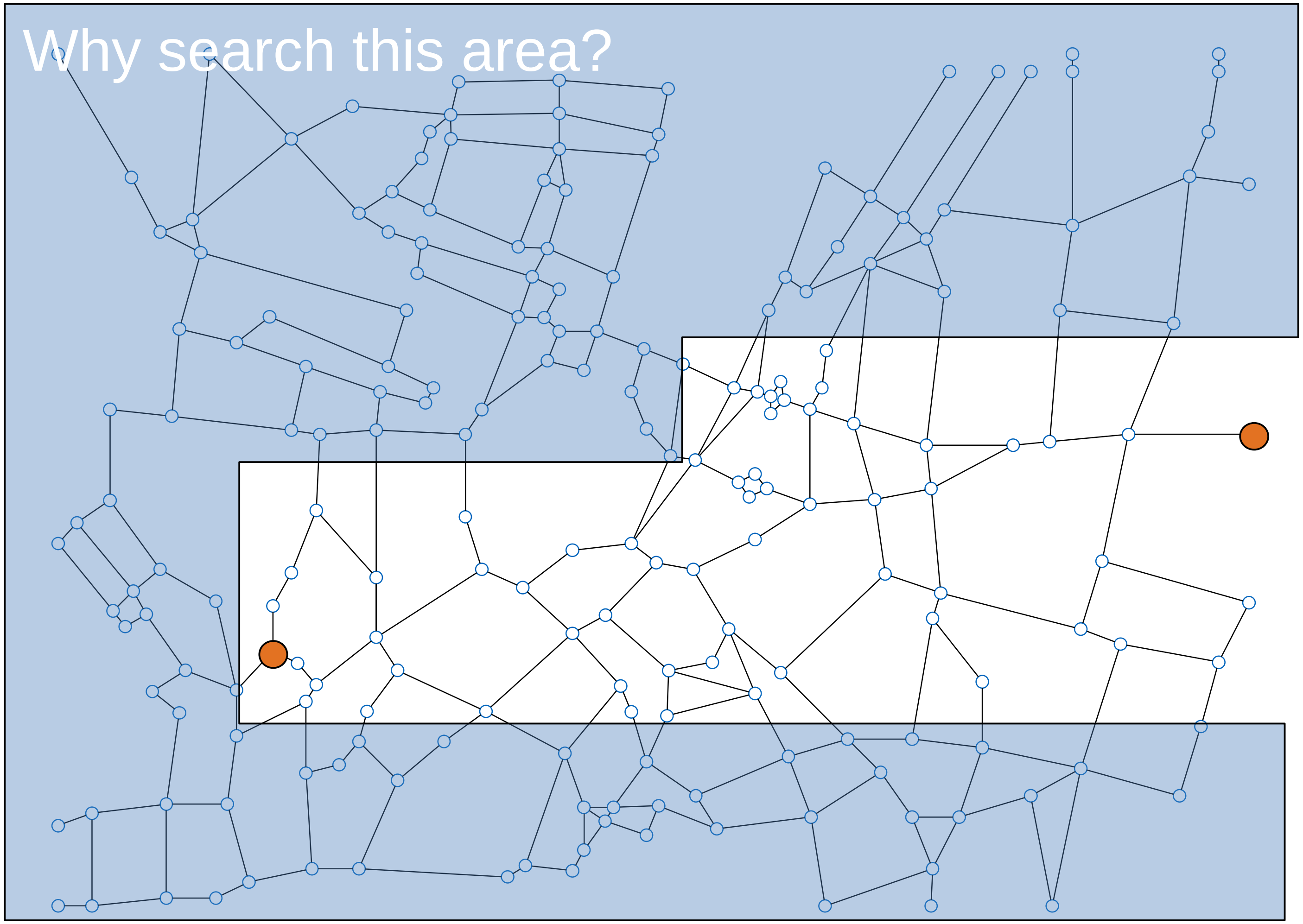
Algorithms – Dijkstra

Munich Demo



Algorithms

A* Approach



Algorithms – A*

Description

Algorithm:

From „Artificial Intelligence“ by Patrick H. Winston

- Form a one-element queue consisting of a zero-length path that contains only the root node

Core Idea: Next Slide

- If the goal node is found, announce success; otherwise announce failure

Algorithms – A*
Description

Algorithm – Core Idea:
From „Artificial Intelligence“ by Patrick H. Winston

- Until the first path in the queue terminates at the goal node or the queue is empty,
    - Remove the first path from the queue; create new paths by extending the first path to all the neighbors of the terminal code.
    - Reject all new paths with loops.
    - If two or more paths reach a common node, delete all those paths except the one that reaches the common node with the minimum cost.
    - Sort the entire queue by the sum of the path length and a lower-bound estimate of the cost remaining, with least-cost paths in front.

Algorithms – A*
Description

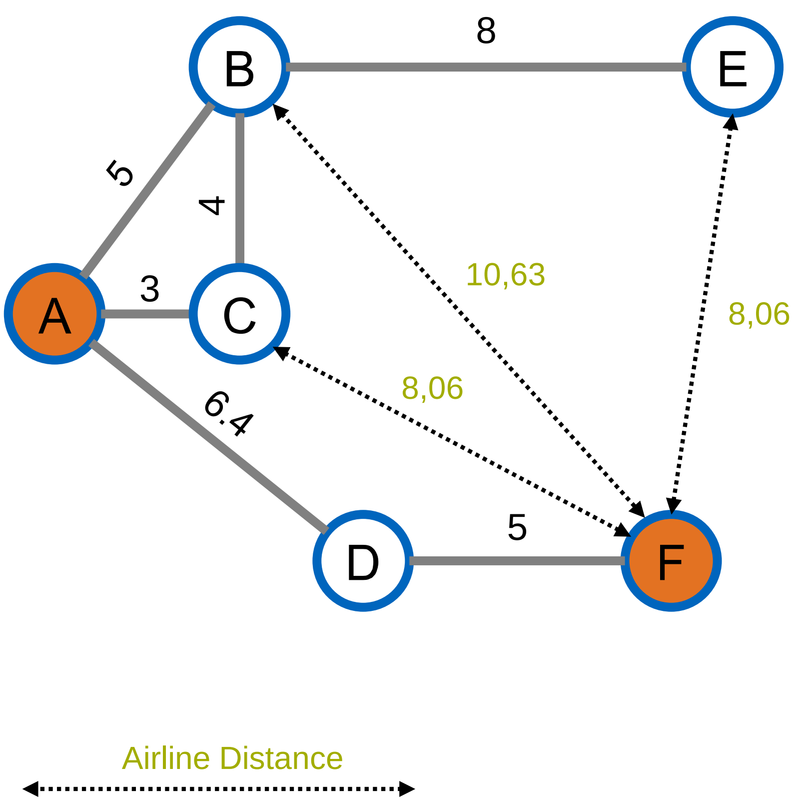
Algorithm – Core Idea:
From „Artificial Intelligence“ by Patrick H. Winston

- Until the first path in the queue terminates at the goal node or the queue is empty,
  - Remove the first path fr… …y extending the first path t…
  - Reject al…
  - If two or more paths reach a common node, delete all those paths except the one that reaches the common node with the minimum cost.
  - Sort the entire queue by the sum of the path length and a lower-bound estimate of the cost remaining, with least-cost paths in front.

Compare Lower Bound Estimate: Airline Distance

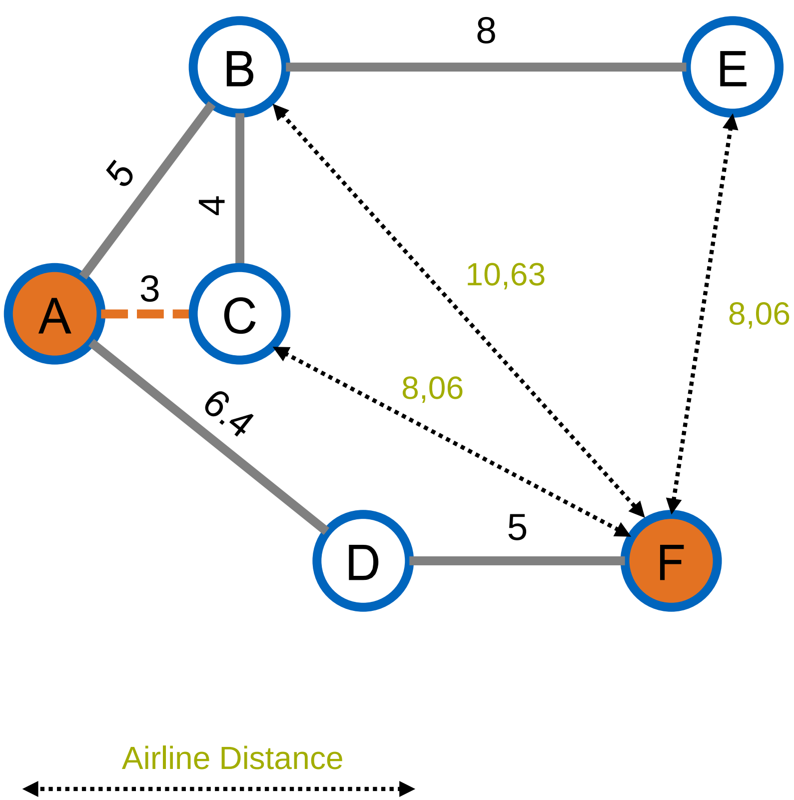
Algorithms – A*

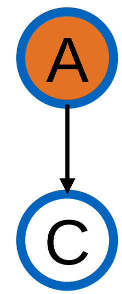
Pathfinding Example Graph



Algorithms – A*

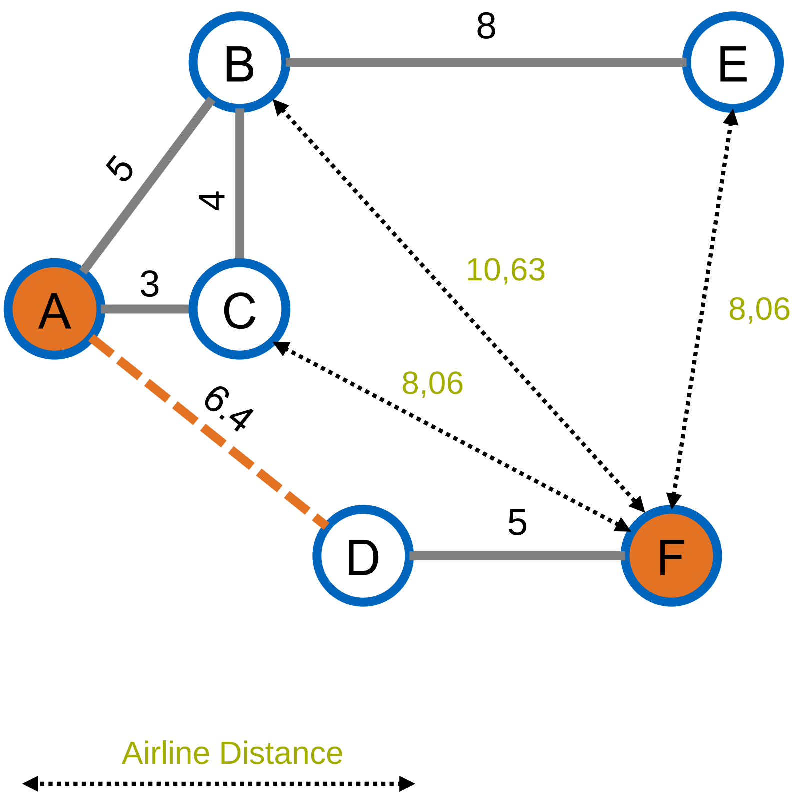
Pathfinding Example Graph

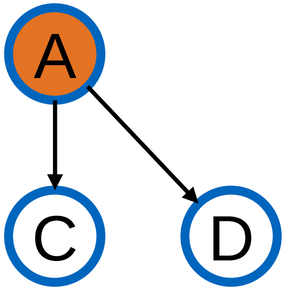




Algorithms – A*

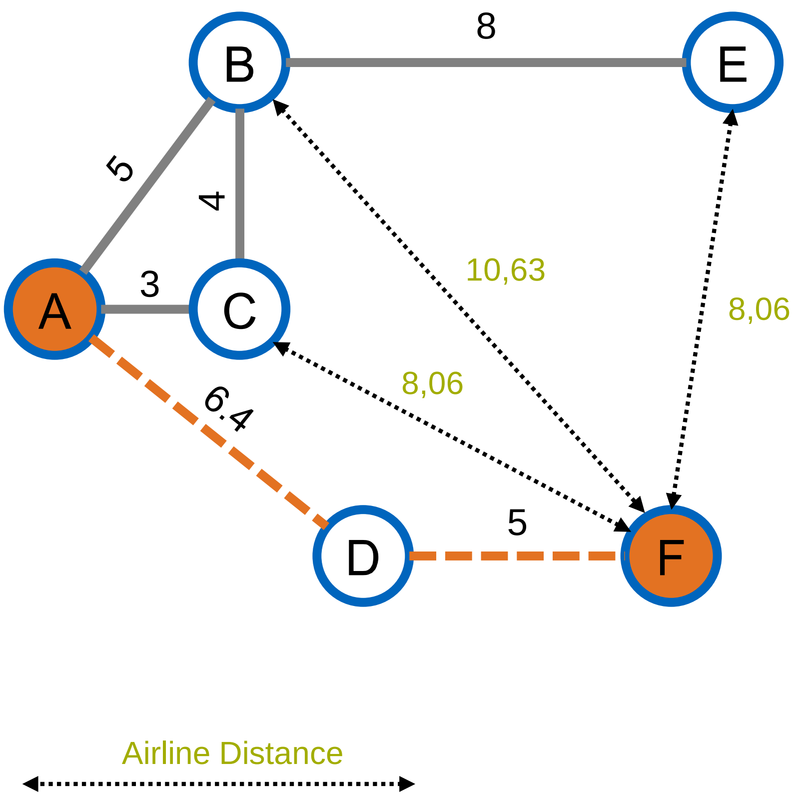
Pathfinding Example Graph

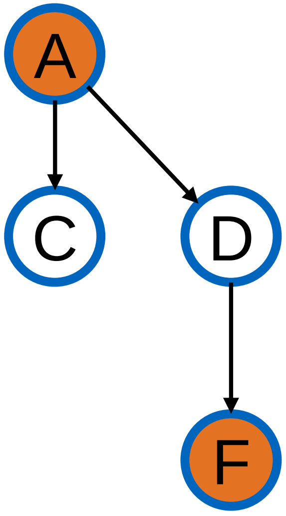




Algorithms – A*

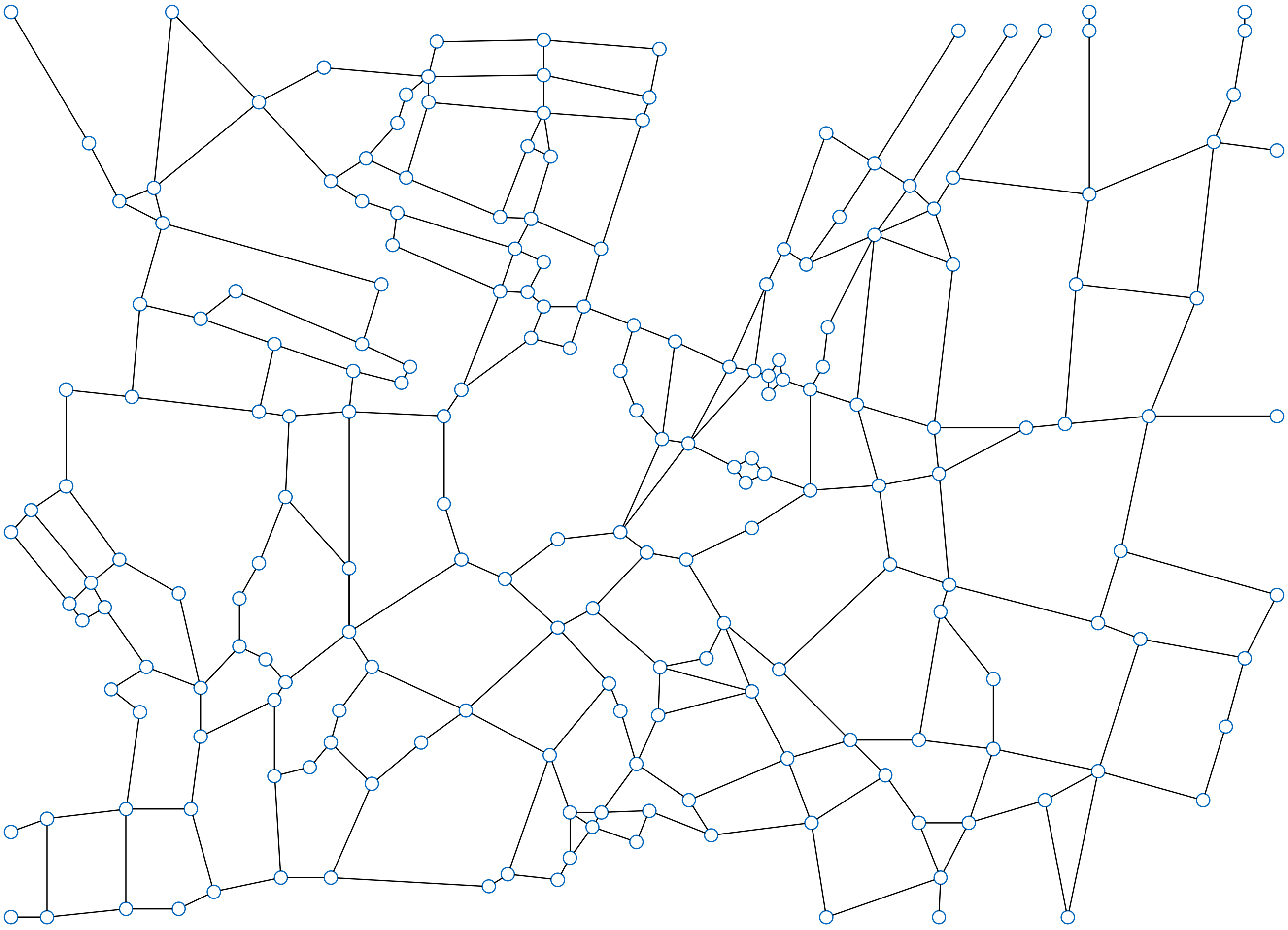
Pathfinding Example Graph





Algorithms – A*

Munich Demo



Algorithms
Attributes

Informed
Best First, Dijkstra, A*

Uninformed
Breadth First, Depth First

Complete
Breadth First, Depth First, Best First, Dijkstra, A*

Optimal
Dijkstra, A*

Application
Real life conditions

HMI	User Interaction
Navigation Engine	<u>Routing</u> Positioning Guidance
Data	<u>Map</u> Traffic POI

Application
Real life conditions

Data	<u>Map</u> Traffic POI

Number of Nodes:
~ 4 billion worldwide

Number of Ways:
~ 400 million worldwide

Vast Amounts of Data

Application

Real life conditions

Data	<u>Map</u> Traffic POI

Static Weights:
distance
cost
scenaric value
CO_2 emissions

Different Types of Static Weights

Application
Real life conditions

Data	Map Traffic POI

Dynamic Weights:
congestion
construction
ferry schedules
tolls
mountain passes

Dynamic Weights

Application
Real life conditions

Navigation Engine	<u>Routing</u> Positioning Guidance

Performance:
small RAM
dynamic rerouting
short query time

Performance

Application
Real life conditions

Navigation Engine	Routing Positioning Guidance

Positioning:
noise
signal loss
initial positioning

Inaccurate Positioning

Summary
What we learned today

Highway geometry, spatial information and traffic rules can be represented by a digital data model

Mathematical graphs can be used to model physical street geometry and information relevant for routing

Different routing algorithms are used to calculate a path through given mathematical graphs

Real life applications bring additional challenges not covered by the theoretical approaches covered in this lecture

Vocabulary and ideas