Graphs & Algorithmic

Peter Levinsky IT Roskilde

20.03.2025

Different Graphs

Two categories

- Trees
- Graphs / Net

Each category can be **with** or **without** weight

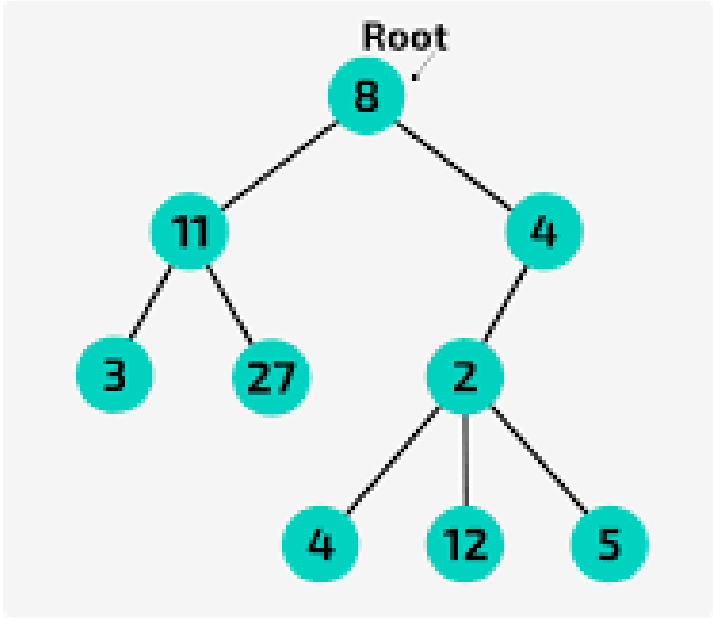
Trees

Special characteristic for Trees (actually a special version of a graph)

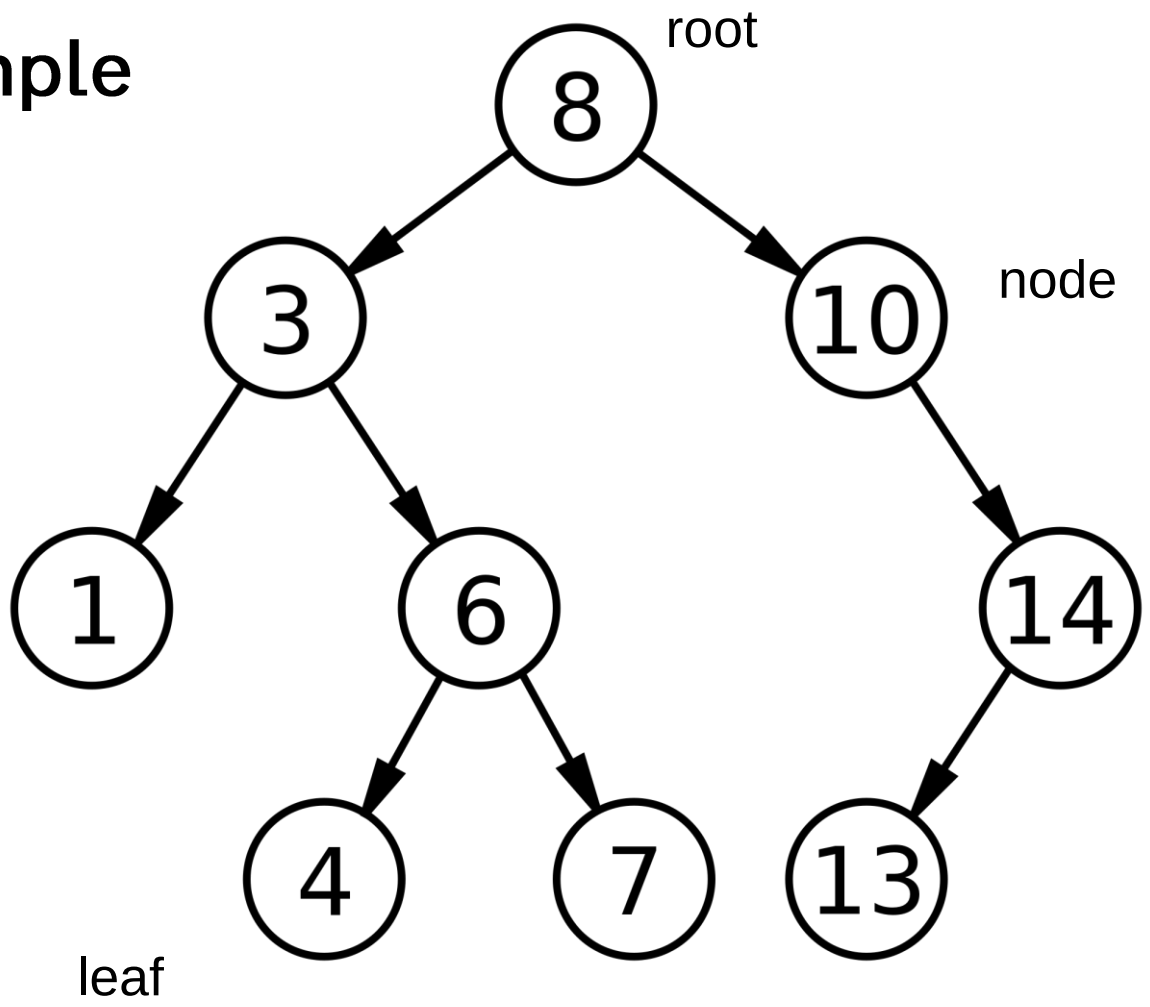
- Only one root
- Unique path between all nodes)

Different types of trees (with or without weight)

- Unsorted tree
- Binary sorted tree
  - Unbalanced
  - Balanced
- 2-3 trees
- heaps (special kind of sorted tree)



Tree – An example

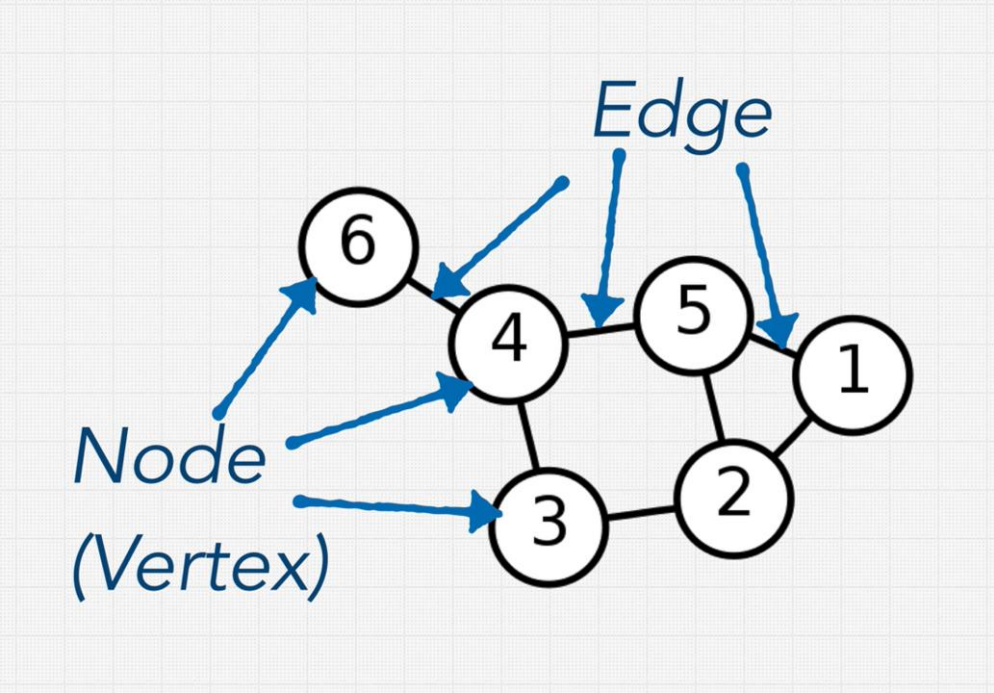


Graphs

Special characteristic for Graphs

- A network of nodes
  - There may be a start node and possibly an end node.
  - Between two nodes there is an edge, which can have a weight.

- Several paths between nodes



Graphs – An example

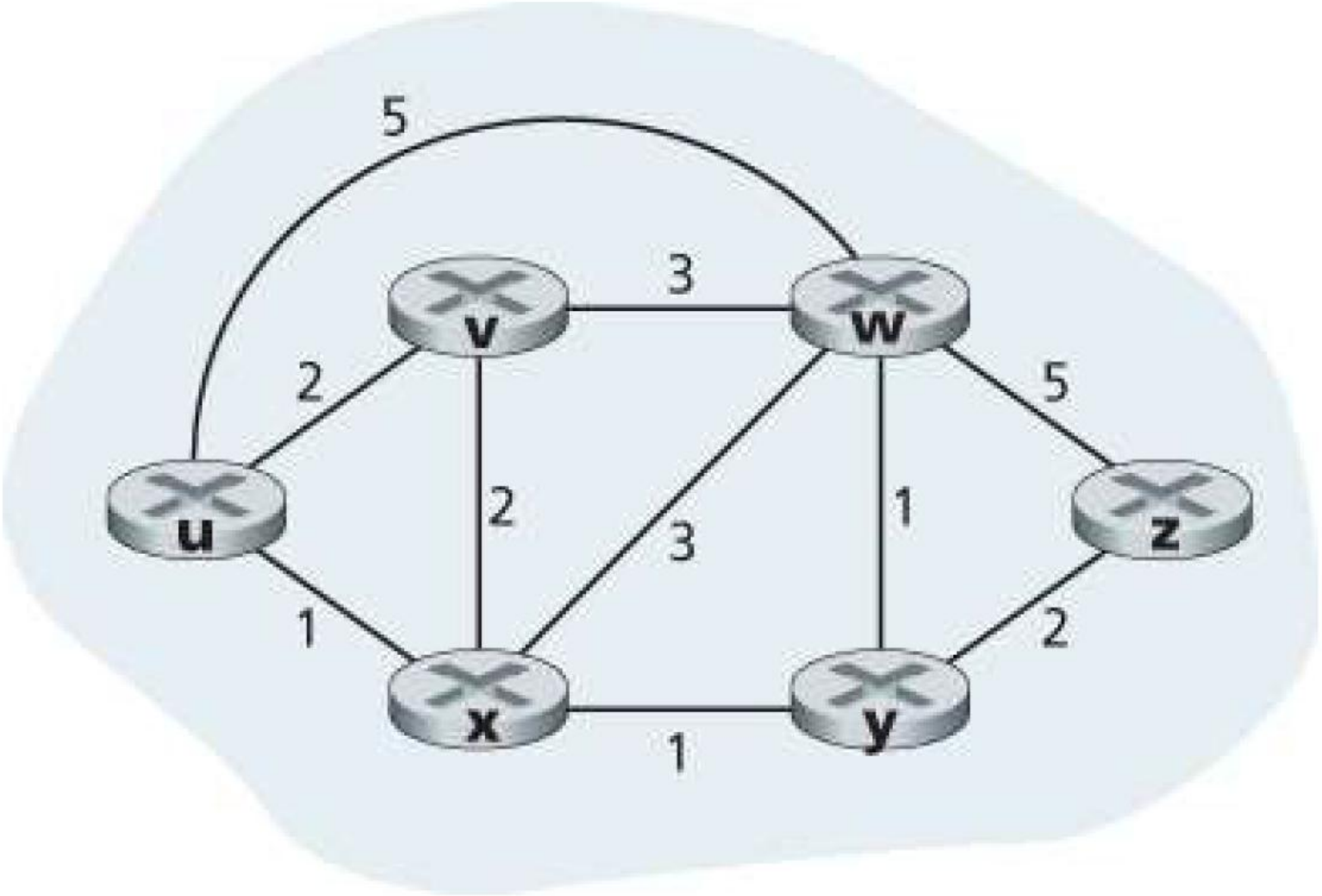


Figure 5.3 Abstract graph model of a computer network

Graphs and algorithms

Different algoritms

- **Dijkstra's algorithm** (shortest path to all nodes)

  Animation: https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm#/media/File:Dijkstras_progress_animation.gif

- **A* algorithm** (shortest path between two nodes)

  Animation:
  https://en.wikipedia.org/wiki/A*_search_algorithm#/media/File:Astar_progress_animation.gif

- Spanning tree (refactor a graph into a tree (min or max))

- Maximum Flow algorithm
  (how much can pass between two nodes)

Dijkstra’s algorithm

The base algorithm in Open Shortest Path First (OSPF) or Link State

Three steps:

1. Collect information of all nodes and edges

2. Chose a node where to start (first part of the solution)

3. Chose the edges with lowest weight to the next node

    1. This node is appended to the solution-net
    2. A set of new edges is calculated seen from the solution-net
    3. Continue 1&2 until all nodes is part of the solution-net

Dijkstra's algorithm

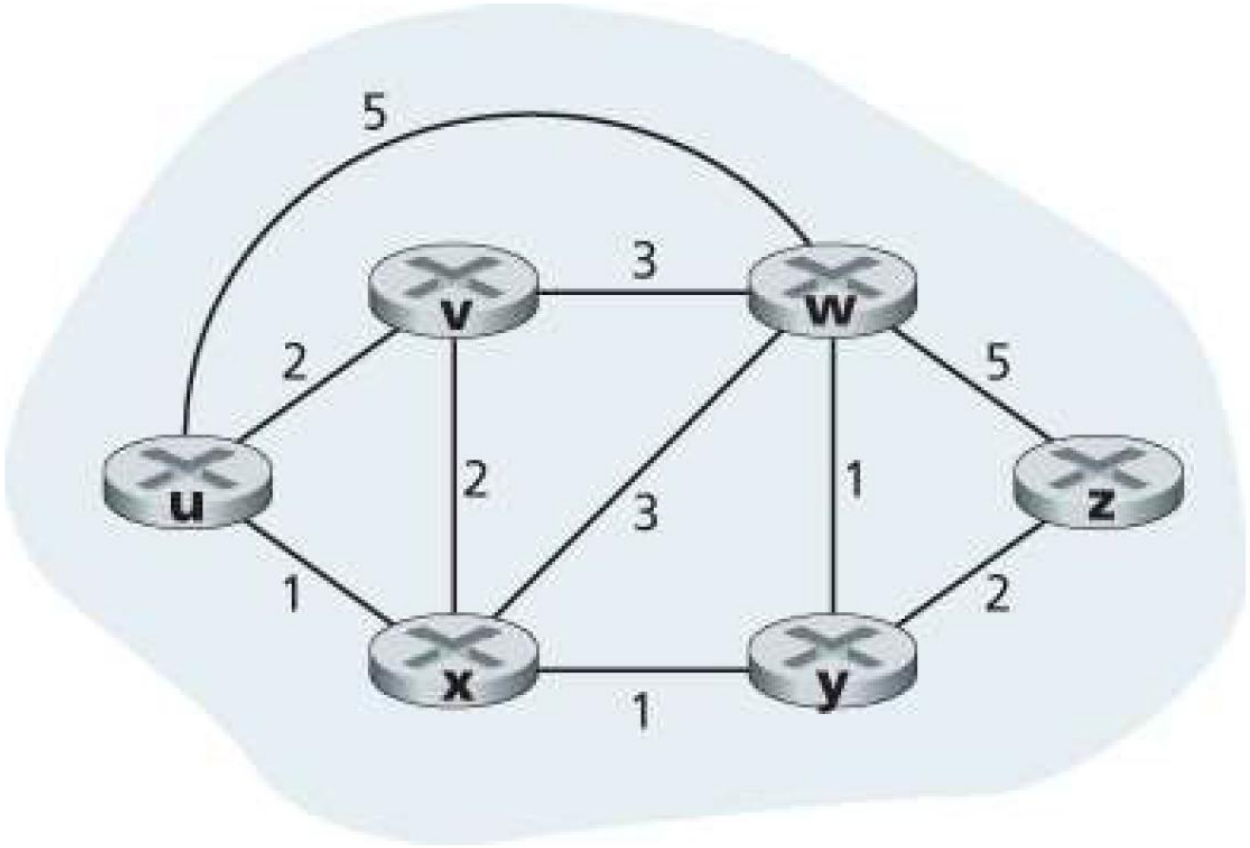
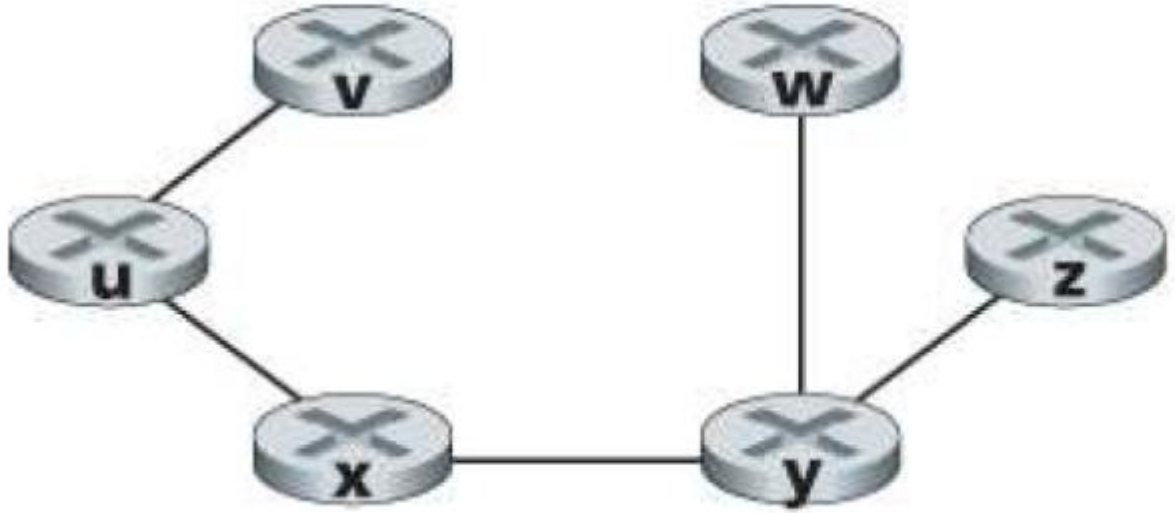


Figure 5.3 Abstract graph model of a computer network

Dijkstra's algorithm



Destination	Link
v	(u, v)
w	(u, x)
x	(u, x)
y	(u, x)
z	(u, x)

Figure 5.4 Least cost path and forwarding table for node u

A* algorithm

Used in games to find a way between two given points

1. Continuously collects information about all nodes and edges

2. Chose the start node and the end node

3. Chooses the ‘smallest’ distance of
   The distance so far + ‘estimated’ (heuristic) path to the end

    1. The node is appended to the solustion-net
    2. From the solution-net, the smallest estimated distance to the end node is selected i.e.
        1. Appended to the solution-net
        2. Remember the path until here

A* algorithm - Heuristic

For the A* algorithm to be effective, the heuristics must be good

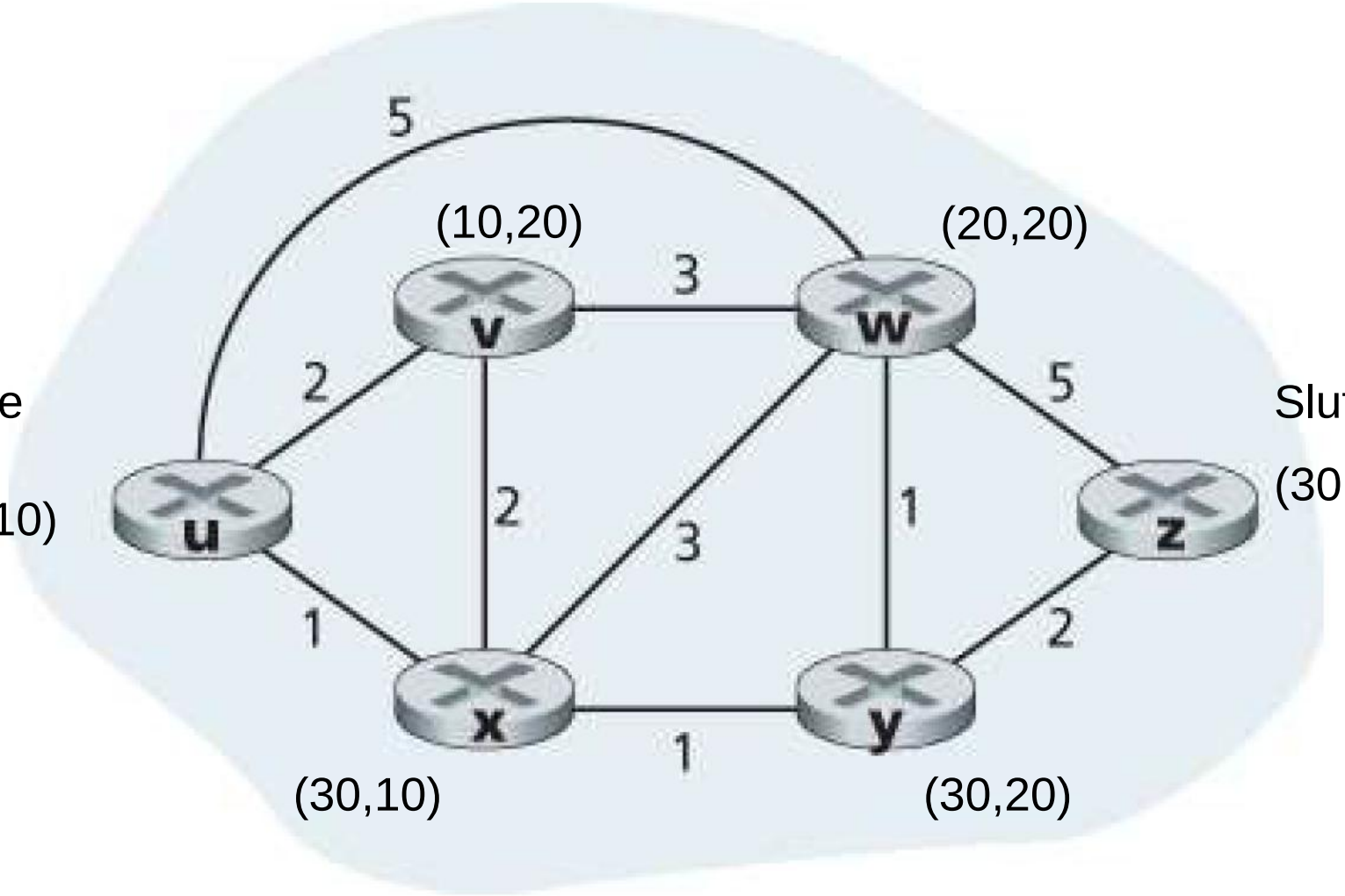
- Very good in games with x,y coordinates (x,y,z),
  - The Heuristic will be the distance between two points e.g.
    - Absolut
    - Pythagyras
    - …

- Less good in computer network,
  - There is no immediate distance between two routers
  - => difficult to define a good heuristic
    in other words if the heuristic consider the distance to be one between two nodes
    the result will resemble the Dijkstra's algorithm

A* algorithm - example

Start Node

(10,10)



Slut Node

(30,30)

Figure 5.3 Abstract graph model of a computer network

That’s it

Opgave: Algoritmer