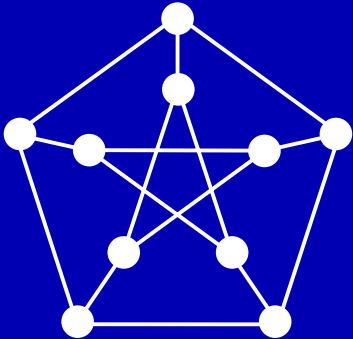


# Gianluca Della Vedova

- Large-Scale Graph Algorithms
- Ufficio U14-2041
- <https://www.unimib.it/gianluca-della-vedova>
- [gianluca.dellavedova@unimib.it](mailto:gianluca.dellavedova@unimib.it)
- <https://github.com/gdv/large-scale-graph-algorithms>
- Everything at <https://elearning.unimib.it/>

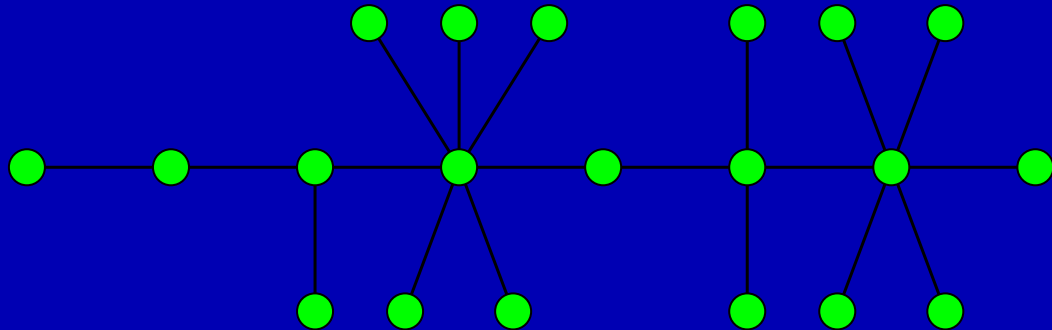
# Example



## Notation

- number of vertices:  $n$
- number of edges/arcs:  $m$

# Better representation



## Almost a path

- Compact representation

# Breadth-first visit

**Data:** graph  $G$ , vertex  $\text{root}$

$Q \leftarrow$  a queue;

label  $\text{root}$  as explored;

$Q.\text{enqueue}(\text{root})$ ;

**while**  $Q \neq \emptyset$  **do**

$v \leftarrow Q.\text{dequeue}()$ ;

**foreach** *edge*  $(v, w)$  **do**

**if**  $w$  is not labeled as explored **then**

            label  $w$  as explored;

$Q.\text{enqueue}(w)$

# Depth-first visit

```
Data: graph  $G$ , vertex root  
 $S \leftarrow$  a stack;  
 $S.\text{push}(\text{root});$   
while  $S \neq \emptyset$  do  
   $v \leftarrow S.\text{pop}();$   
  if  $v$  is not labeled as explored then  
    label  $v$  as explored;  
    foreach edge  $(v, w)$  do  
       $S.\text{push}(w)$ 
```

# Dijkstra's algorithm

**Data:** graph  $G$ , vertex source

$Q \leftarrow$  a queue;

**foreach** *vertex*  $v$  **do**

$\text{dist}[v] \leftarrow \infty$ ;

$Q.\text{enqueue}(v)$

$\text{dist}[\text{source}] \leftarrow 0$ ;

**while**  $Q \neq \emptyset$  **do**

$u \leftarrow$  vertex in  $Q$  minimizing  $\text{dist}[u]$ ;

$Q.\text{deque}(u)$ ;

**foreach** *neighbor*  $v$  of  $u$  *still in*  $Q$  **do**

$\text{alt} \leftarrow \text{dist}[u] + \text{Graph.Edges}(u, v)$ ;

**if**  $\text{alt} < \text{dist}[v]$  **then**

$\text{dist}[v] \leftarrow \text{alt}$ ;

$\text{prev}[v] \leftarrow u$ ;

**return**  $\text{dist}[], \text{prev}[]$ ;

# Dijkstra's algorithm — priority queue

**Data:** graph  $G$ , vertex source

$Q \leftarrow$  a priority queue;

**foreach** *vertex*  $v$  **do**

$\text{dist}[v] \leftarrow \infty$ ;

$Q.\text{add\_with\_priority}(v, \text{dist}[v])$

$\text{dist}[\text{source}] \leftarrow 0$ ;

**while**  $Q \neq \emptyset$  **do**

$u \leftarrow Q.\text{extract\_min}$ ;

**foreach** *neighbor*  $v$  of  $u$  still in  $Q$  **do**

$\text{alt} \leftarrow \text{dist}[u] + \text{Graph.Edges}(u, v)$ ;

**if**  $\text{alt} < \text{dist}[v]$  **then**

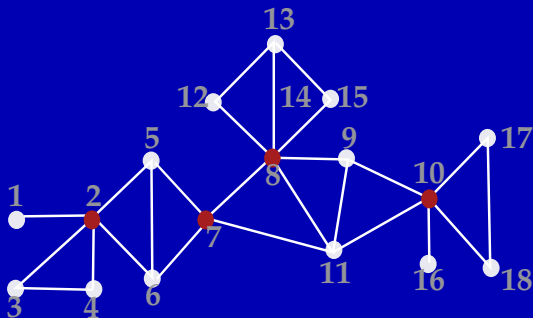
$\text{dist}[v] \leftarrow \text{alt}$ ;

$\text{prev}[v] \leftarrow u$ ;

$Q.\text{decrease\_priority}(v, \text{alt})$

**return**  $\text{dist}[], \text{prev}[]$ ;

# Biconnected components





# Find articulation points

**Data:** connected graph  $G$ , vertex root

$S \leftarrow$  a stack;

$S.\text{push}((\text{root}, \text{nil}))$ ;

$d \leftarrow -1$ ;

**while**  $S \neq \emptyset$  **do**

$d \leftarrow d + 1$ ;

$(v, p) \leftarrow S.\text{peek}()$ ;

**if**  $v$  is not explored **then**

        label  $v$  as explored;  $\text{parent}(v) \leftarrow p$ ;  $\text{depth}[v] \leftarrow d$ ;

$\text{lowpoint}[v] = \text{depth}[v]$ ;

**foreach** *edge*  $(v, w)$  **do**

$S.\text{push}((w, v))$ ;

**else**

$\text{lowpoint}[v] =$

$\min \{ \min_{w \in N(v), w \neq p} \{ \text{depth}[w] \}, \min_{w \in N(v), \text{parent}(w)=v} \{ \text{lowpoint}[w] \} \}$

$d \leftarrow d - 1$ ;

$v \leftarrow S.\text{pop}()$ ;

# Find bridges

**Data:** connected graph  $G$ , vertex root

$S \leftarrow$  a stack;

$S.\text{push}((\text{root}, \text{nil}))$ ;

$x \leftarrow -1$ ;

**while**  $S \neq \emptyset$  **do**

$(v, p) \leftarrow S.\text{peek}()$ ;

**if**  $v$  is not explored **then**

$x \leftarrow x + 1$ ;

        label  $v$  as explored;  $P(v) \leftarrow p$ ;  $\text{num}[v] \leftarrow x$ ;

**foreach** edge  $(v, w)$  **do**

$S.\text{push}((w, v))$ ;

**else**

$\text{nd}[v] = 1 + \sum_{w \in N(v), P(w)=v} \text{nd}[w]$ ;

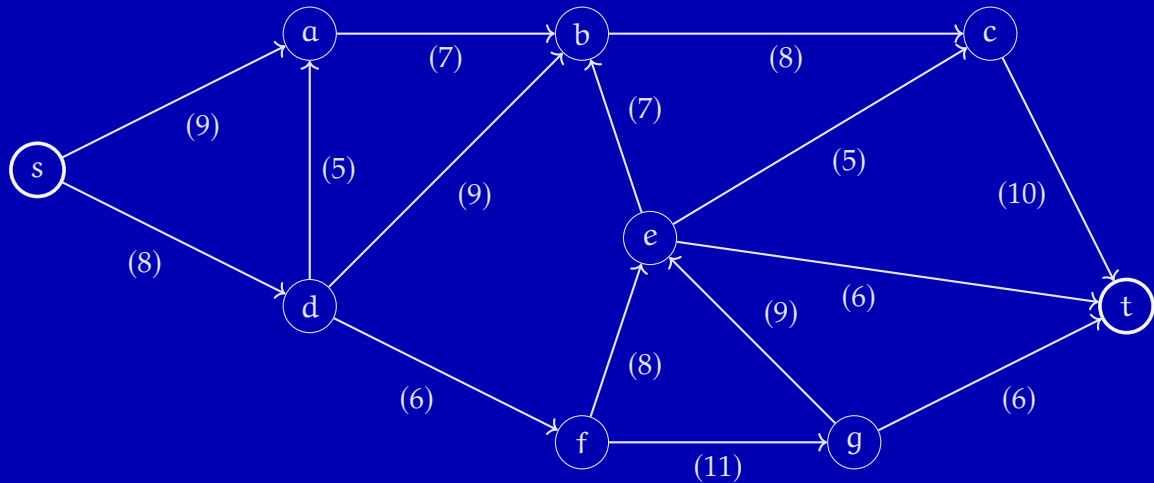
$l[v] =$

$\min \{ \text{num}[v], \min_{w \in N(v), w \neq p, P(w) \neq v} \{ \text{num}[w] \}, \min_{w \in \text{children}(v)} \{ l[w] \} \}$ ;

$h[v] =$

$\max \{ \text{num}[v], \max_{w \in N(v), w \neq p, P(w) \neq v} \{ \text{num}[w] \}, \max_{w \in \text{children}(v)} \{ h[w] \} \}$ ;

# Max flow



# Max flow – Min cut theorem

Let  $f$  be a flow of a graph  $G = (V, E)$ . Then the following three conditions are equivalent:

- 1  $f$  is a maximum flow
- 2 the residual graph has no augmenting path
- 3 there is a cut  $(S, T)$  of  $G$  such that  $c(S, T) = |f|$

# Figures

- David Eppstein, Public Domain,  
<https://commons.wikimedia.org/w/index.php?curid=10261635>

# License

This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License (<https://creativecommons.org/licenses/by-sa/4.0/>).

