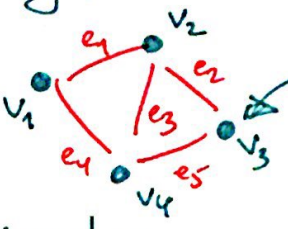
TNSL20, 05.09.2017, L2

Chapter 4: Shortest Paths

4.1 Graphs

We start with important data structure: graphs (there even exists an entire research area „graph theory")

Nodes ≙ vertices
Links ≙ edges



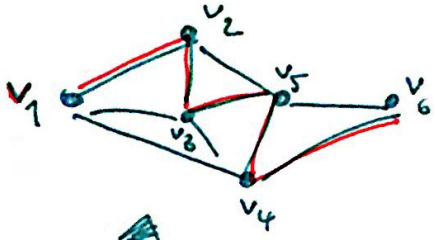
one vertex

$e_1 = \{v_1, v_2\}$

(set, as direction doesn't matter)

edge connects two vertices v and w
⇒ • v and w are adjacent
• e incident to v and w

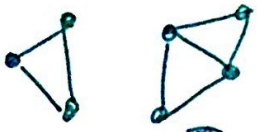
Path:



path from v_1 to v_6

connected (the entire graph)

one



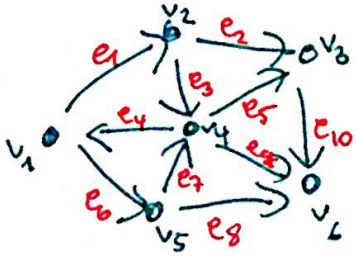
← two connected components)

not connected (the entire graph)

We write $G = (V, E)$ in general undirected.

graph G — (set of) vertices — (set of) edges

What's directed?

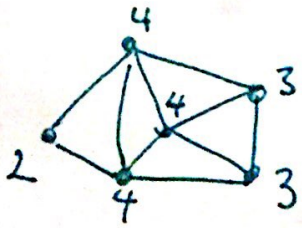


G is a digraph

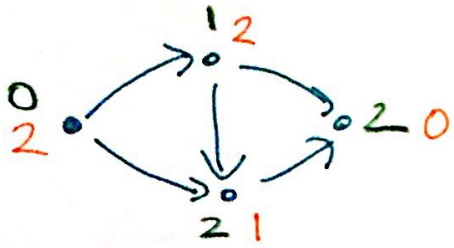
$e_1 = (v_1, v_2)$

$e_2 = (v_2, v_3)$

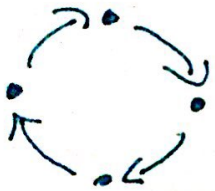
degree: # edges incident to node



directed: indegree - those that „arrive" at vertex v
outdegree - —//— „leave" vertex v



cycle:

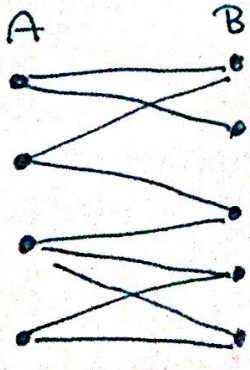


graphs with cycles: cyclic
—//— without —//— : acyclic

Loops: 

DAG: directed acyclic graph

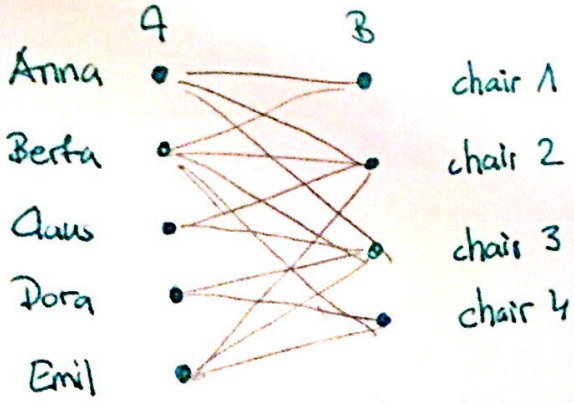
bipartite graph: two node sets A and B
- edges only from vertex in A to vertex in B
- no edge from —//— —//— —— A
- —//— —— //— —— B —//— — B



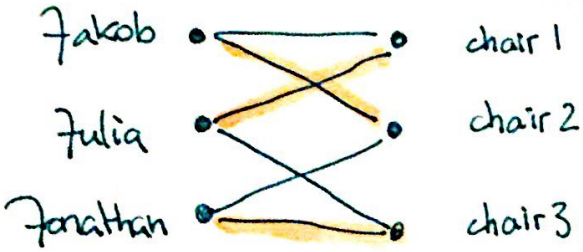
matching:

Example: A - kids in a class
B - chairs → " "

we want to match A to B, i.e., kids to chairs



$5 = |A| > |B| = 4$ ⇒ not all kids can get a chair
⇒ we want to match as many as possible
⇒ maximum matching



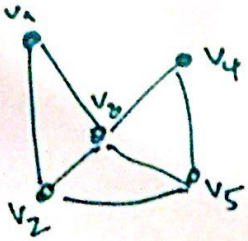
if we can match all:
perfect matching

graphs: - many edges: dense
- few edges: sparse

(dense close to n^2)
(close to n)

how much possible?
each node connected to all others

complete graph
$\frac{n(n-1)}{2}$ edges



adjacency matrix

	v_1	v_2	v_3	v_4	v_5
v_1	0	1	1	0	0
v_2	1	0	1	0	1
v_3	1	1	0	1	1
v_4	0	0	1	0	1
v_5	0	1	1	1	0

adjacency list:

v_1: v_2, v_3
v_2: v_1, v_3, v_5
v_3: v_1, v_2, v_4, v_5
v_4: v_3, v_5
v_5: v_2, v_3, v_4

How that looks like as data structure: later

4.2 Shortest Paths (from a Single Source)

Example: optimal line-breaking / word wrapping

Shortest Path Problem:

from a starting point (=:s) shortest path to any other point
Let weight function be $c: E(G) \to \mathbb{R}$

Formal problem?

Input: Digraph $G=(V,E)$
weight function $c: E(G) \to \mathbb{R}$
two vertices s,t

Output: An s-t-path of minimum total weight.

Procedure from page 173 (= our algorithm)

1. Distance to $s=0$ / Length of a shortest path to $s=0$ / $\ell(s)=0$
   Distance to all other vertices $=\infty$ / $\ell(v)=\infty$ for all other vertices
   $\forall v \in V \setminus \{s\}$
   $R := \emptyset$
2. Find vertex v with lowest distance of those not yet considered, i.e. $v \in V \setminus R$
   // we start with $v=s$
3. $R := R \cup \{v\}$ "union" ≙ adding to a set
4. FOR all vertices w adjacent to v:

FOR $\forall w \in V(G) \setminus R$ with $(v,w) \in E(G)$ DO
IF $(\ell(w) > \ell(v) + c(\{v,w\})$ THEN
$\ell(w) = \ell(v) + c(\{v,w\})$
$p(w) = v$

compare: (a) $\ell(w)$
(b) $\ell(v) + c(\{v,w\})$
IF (b) is smaller ($\ell(w) > \ell(v) + c(\{v,w\})$)
$\ell(w) = \ell(v) + c(\{v,w\})$
predecessor $(w) = v$

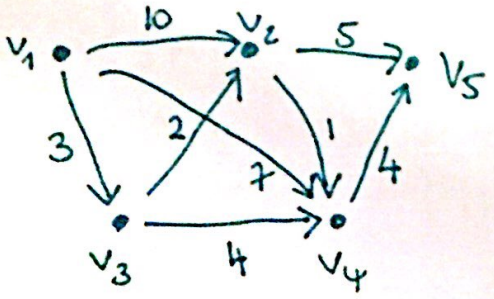
5. IF $R \neq V(G)$ GOTO (2)

new 2.
Find vertex $v \in V(G) \setminus R$ with $\ell(v) = \min_{w \in V(G) \setminus R} \ell(w)$
"without"

"We finish when there are no more nodes to examine"
→ need to keep track what we examined:
Let R be the set of vertices we already examined
→ we start with $R = \emptyset$
→ when $R = V$ we are done

Examples:

- example from Figure 7.5, 7.6
-



from $v_1 = s$

1. $\ell(v_1) = 0$, $\ell(v) = \infty \; \forall v \in V \setminus \{v_1\}$, $R = \emptyset$
2. $v = s = v_1$
3. $R = \{v_1\}$
4. v_2: $\ell(v_2) = \infty > \ell(v_1) + c(\{v_1, v_2\}) = 0 + 10$
   $\rightarrow \ell(v_2) = 10$, predecessor $(v_2) = v_1$
   v_3: $\ell(v_3) = \infty > \ell(v_1) + c(\{v_1, v_3\}) = 0 + 3$
   $\rightarrow \ell(v_3) = 3$, predecessor $(v_3) = v_1$
5. $\{v_1\} \neq \{v_1, \dots, v_5\} \rightarrow$ GOTO 2
2. choose from $V \setminus R = \{v_2, \dots, v_5\}$: $v = v_3$
3. $R = \{v_1, v_3\}$
4. v_2: $\ell(v_2) = 10 > \ell(v_3) + c(\{v_3, v_2\}) = 3 + 2 = 5$
   $\rightarrow \ell(v_2) = 5$, predecessor $(v_2) = v_3$
   v_4: $\ell(v_4) = \infty > \ell(v_3) + c(\{v_3, v_2\}) = 3 + 4 = 7$
   $\rightarrow \ell(v_4) = 7$, predecessor $(v_4) = v_3$
5. $\{v_1, v_3\} \neq \{v_1, \dots, v_5\} \rightarrow$ GOTO 2
2. choose from $\{v_2, v_4, v_5\}$: $v = v_2$
3. $R = \{v_1, \dots, v_3\}$
4. v_5: $\ell(v_5) = \infty > \ell(v_2) + c(\{v_2, v_5\}) = 5 + 5 = 10$
   $\rightarrow \ell(v_5) = 10$, predecessor $(v_5) = v_2$
   v_4: $\ell(v_4) = 7 > \ell(v_2) + c(\{v_2, v_4\}) = 5 + 1 = 6$
   $\rightarrow \ell(v_4) = 6$, predecessor $(v_4) = v_2$
5. $\{v_1, v_2, v_3\} \neq V \rightarrow$ GOTO 2
2. choose from $\{v_4, v_5\}$: $v = v_4$

3. $R = \{v_1, \dots, v_4\}$
4. v_5: $\ell(v_5) = 10 = \ell(v_4) + c(\{v_4, v_5\}) = 6 + 4 = 10$
   → nothing
5. $\{v_1, \dots, v_4\} \neq V$ → GOTO 2
2. choose from $\{v_5\}$: $v = v_5$
3. $R = \{v_1, \dots, v_5\}$
4. /
5. $R = V$
   ↳ DONE

Some formalizations

↳ Algorithm 4.2: Dijkstra