Algorithm and Data Structures Recap

Georg Hasebe — 22/02/2024

Algorithms and Data Structures Recap

Math notation and terminology

- $\mathbb{N} = \{1,2,3,\ldots\}$, $\mathbb{N}_0 = \{0,1,2,3,\ldots\}$, $[n] = \{1,2,3,\ldots,n\}$
- $\ln$ logarithm to the base e, $\log$ logarithm to the base 2
- $|X|$ cardinality of X (number of elements in X)
- $2^X = \{Y \mid Y \subseteq X\}$ (power set of X)
- $\binom{X}{k} = \{Y \mid Y \subseteq X, |Y| = k\}$ (the set of subsets of X of cardinality k)

Examples

$$[3] = \{1,2,3\}$$

$$X = \{1,2\}$$

$$|X| = 2$$

$$2^X = \{\emptyset, \{1\}, \{2\}, \{1,2\}\}$$

$$A = \{1,2,3,4\}$$

$$\binom{A}{2} = \{\{1,2\}, \{2,3\}, \{3,4\}, \{1,3\}, \{1,4\}, \{2,4\}\}$$

(*) The notation $\binom{A}{k}$ also denotes the binomial coefficient, where A is not a set but an integer.

Algorithms and Data Structures Recap

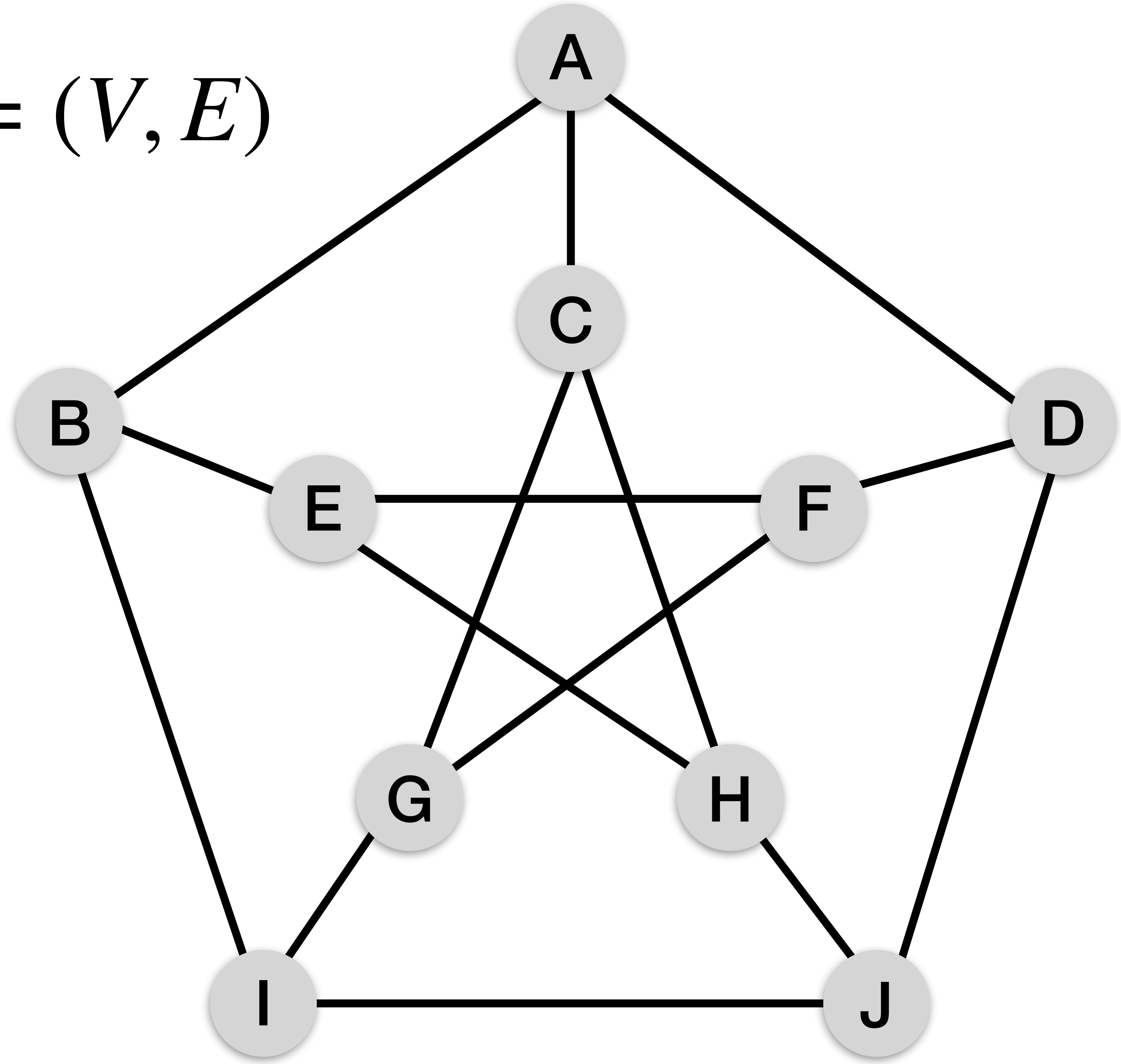
Graph notation and terminology

- $G = (V, E)$, $|V|$, $|E|$…
- $N_G(v) := \{u \in V \mid \{u, v\} \in E\}$ (Nachbarschaft von v in G)
- $G[X], \quad X \subseteq V$ (induced subgraph with vertex set X and edges of E that have both endpoints in X)
- deg(v), deg$_{\text{in}}(v)$, deg$_{\text{out}}(v)$, deg$^-(v)$, deg$^+(v)$ (number of edges: in/- for “eingehend”, out/+ for “ausgehend”)

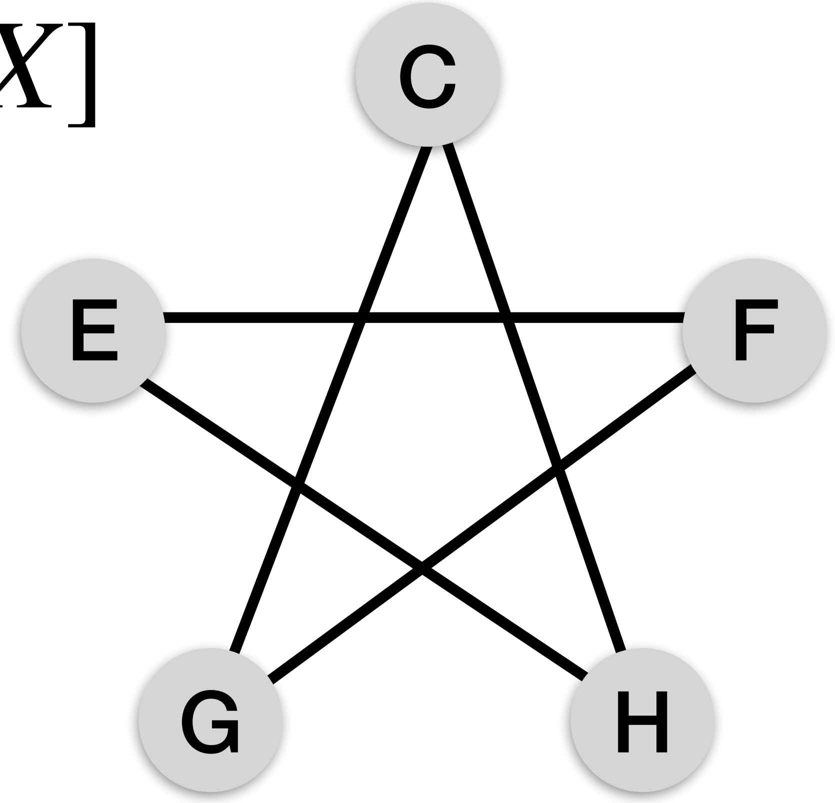
(*) We often shorten $N_G(v)$ to $N(v)$ when it’s clear what graph G is meant.

Examples

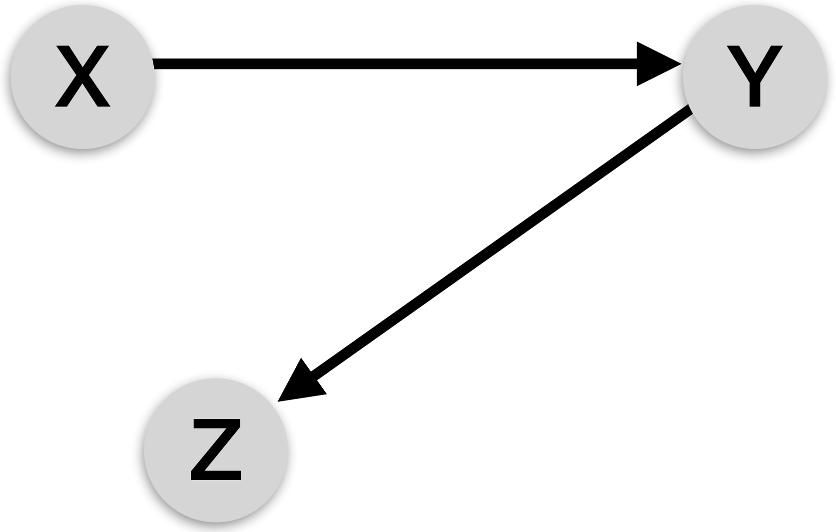
$G = (V, E)$



$G[X]$



$X = \{C, E, F, G, H\}$



$N_G(C) = \{A, G, H\}$

$\text{deg}(F) = 2$

$\text{deg}^+(X) = \text{deg}_{\text{out}}(X) = 1$

$\text{deg}^-(X) = \text{deg}_{\text{in}}(X) = 0$

Algorithms and Data Structures Recap

Notation and terminology

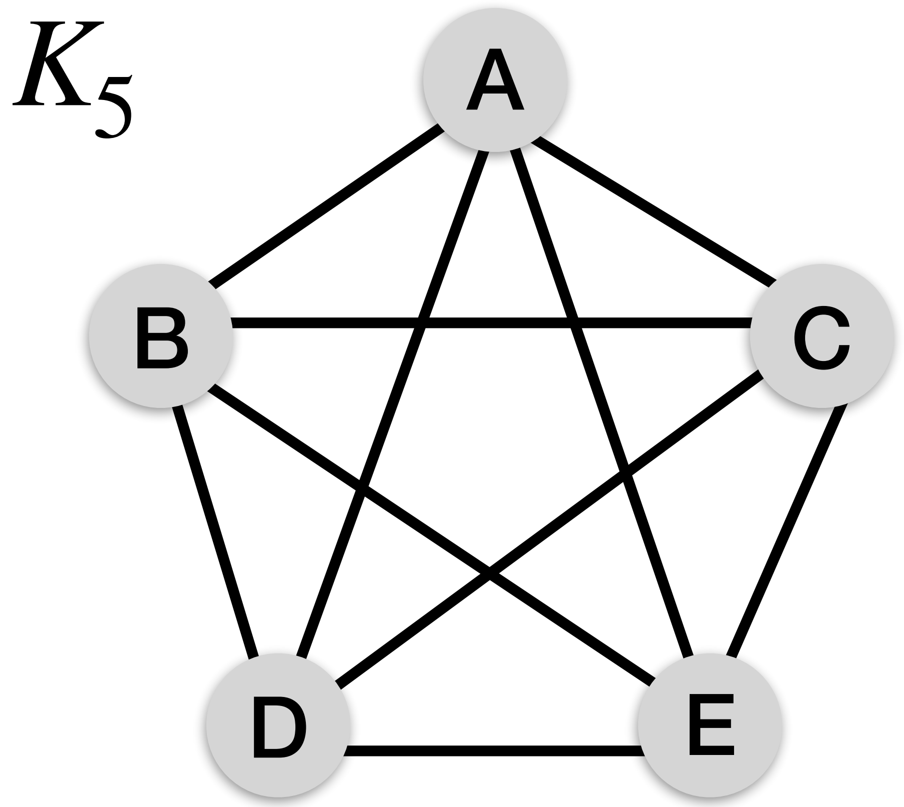
- A sequence of vertices $< v_1, v_2, \ldots, v_k >$, such that v_i and v_{i+1} are connected by an edge
  - Is a walk (Weg).
  - Is a closed walk (geschlossener Weg), if $v_1 = v_k$.
  - Is a path (Pfad), if all vertices are distinct.
  - Is a cycle (Kreis), if all vertices are distinct and $v_1 = v_k$.
  - Has length $k - 1$.

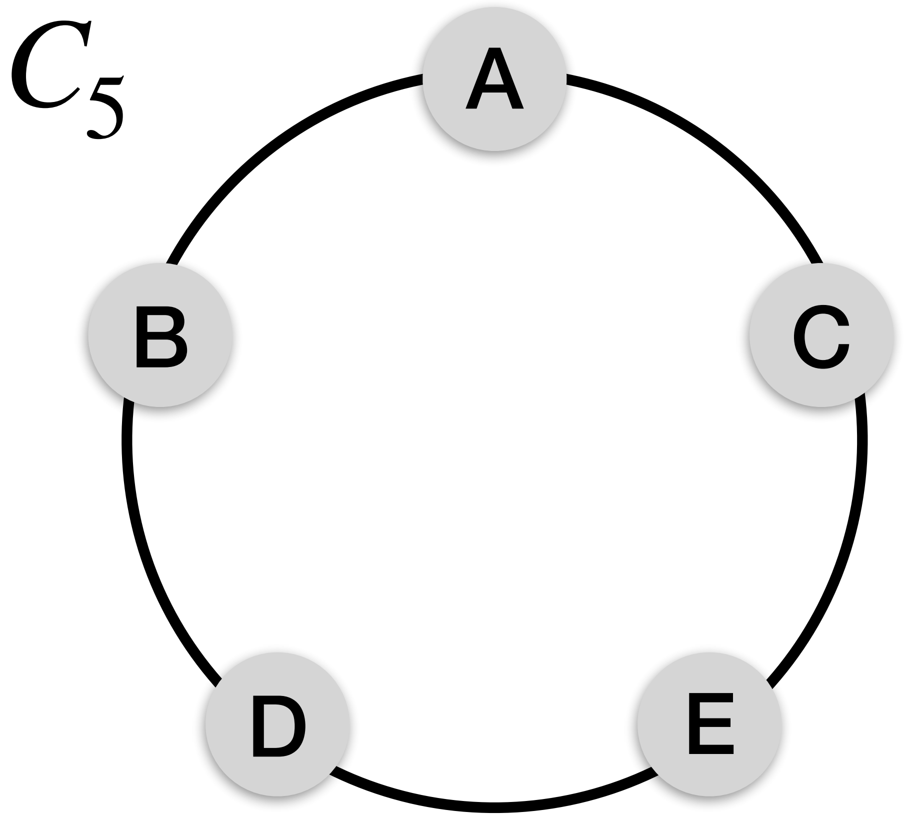
Algorithms and Data Structures Recap

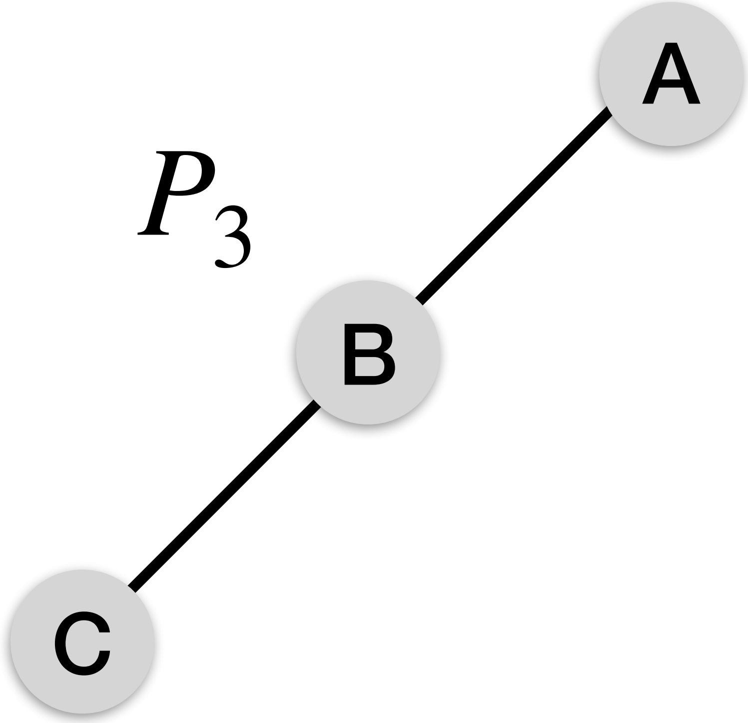
Important graphs

- K_n denotes the *complete graph* on n vertices, where every pair of different vertices is connected by an edge.
- C_n denotes the *cycle graph* on n vertices.
- P_n denotes the *path graph* on n vertices.

Examples







Algorithms and Data Structures Recap

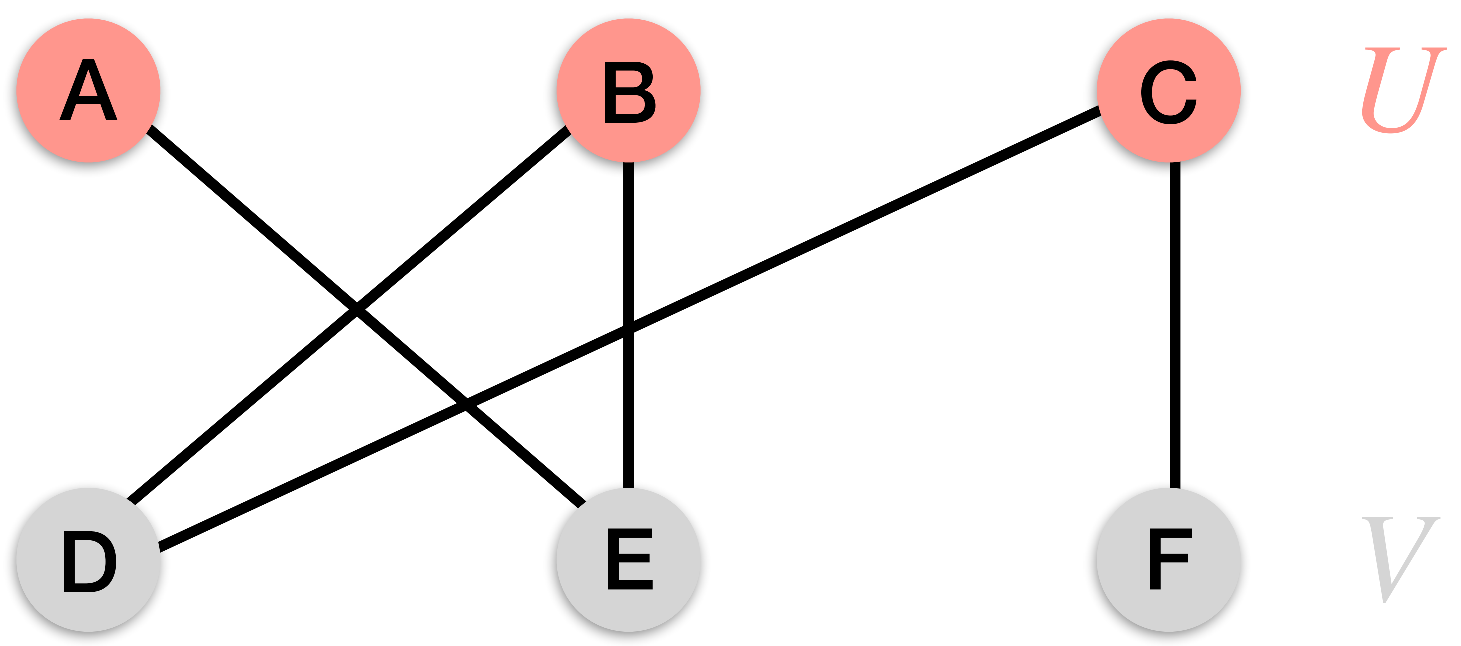
Bipartite graphs

- A *bipartite graph* is a graph whose vertices can be divided into two disjoint sets U and V such that every edge connects a vertex in U to one in V.
- We usually write $G = (U \uplus V, E)$ for a bipartite graph G.

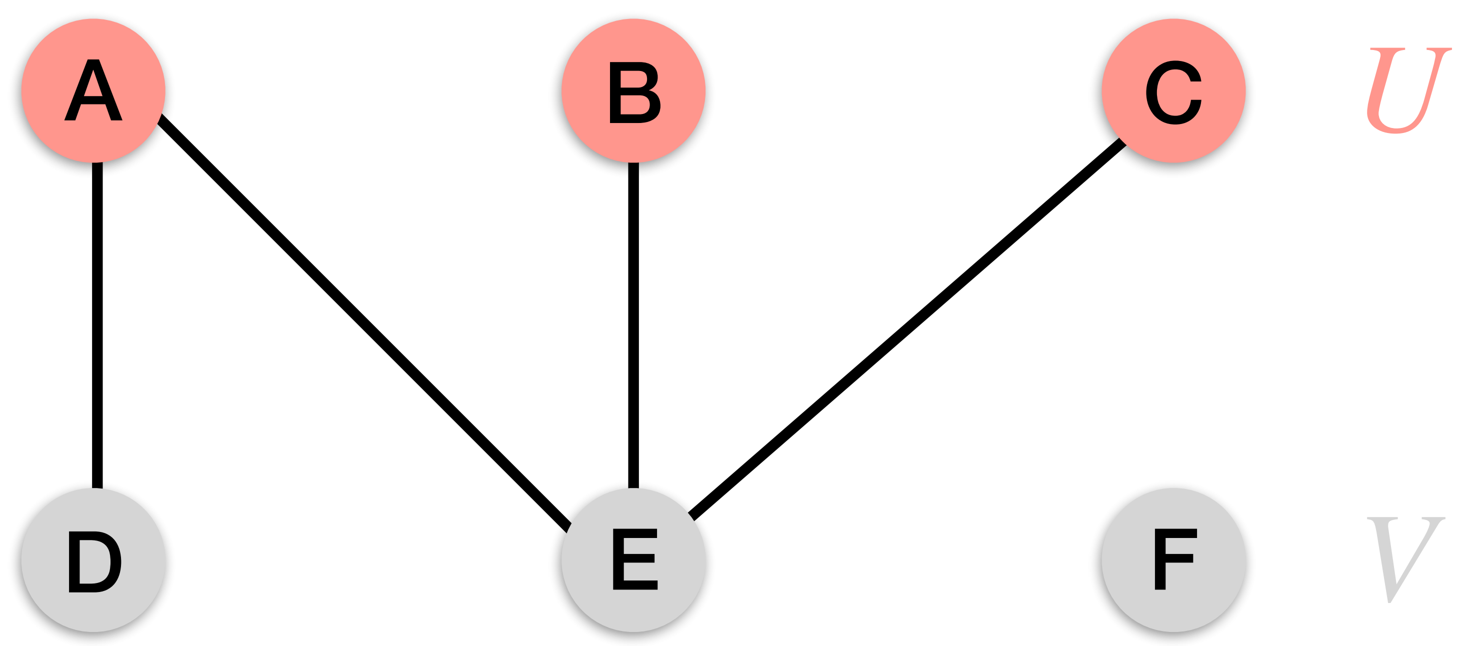
(*) disjoint simply means that $U \cap V = \emptyset$
(**) the symbol $\uplus$ is often used to denote a disjoint union of sets, i.e. $U \cup V$ such that $U \cap V = \emptyset$

Example

$G = (U \uplus V, E)$

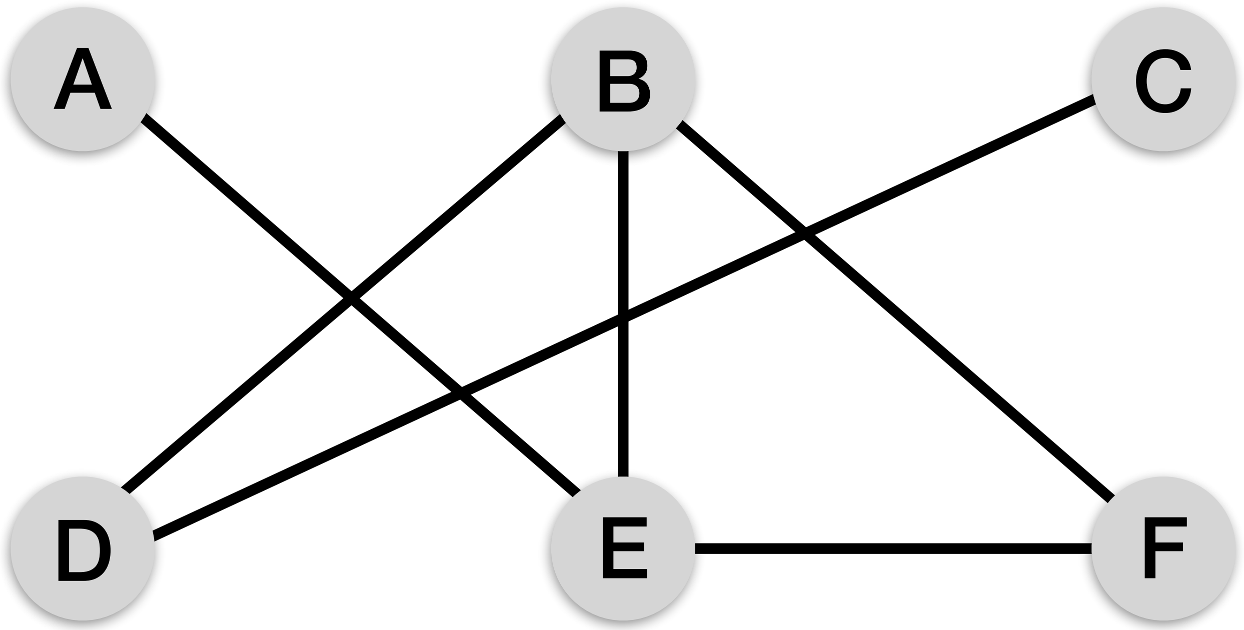


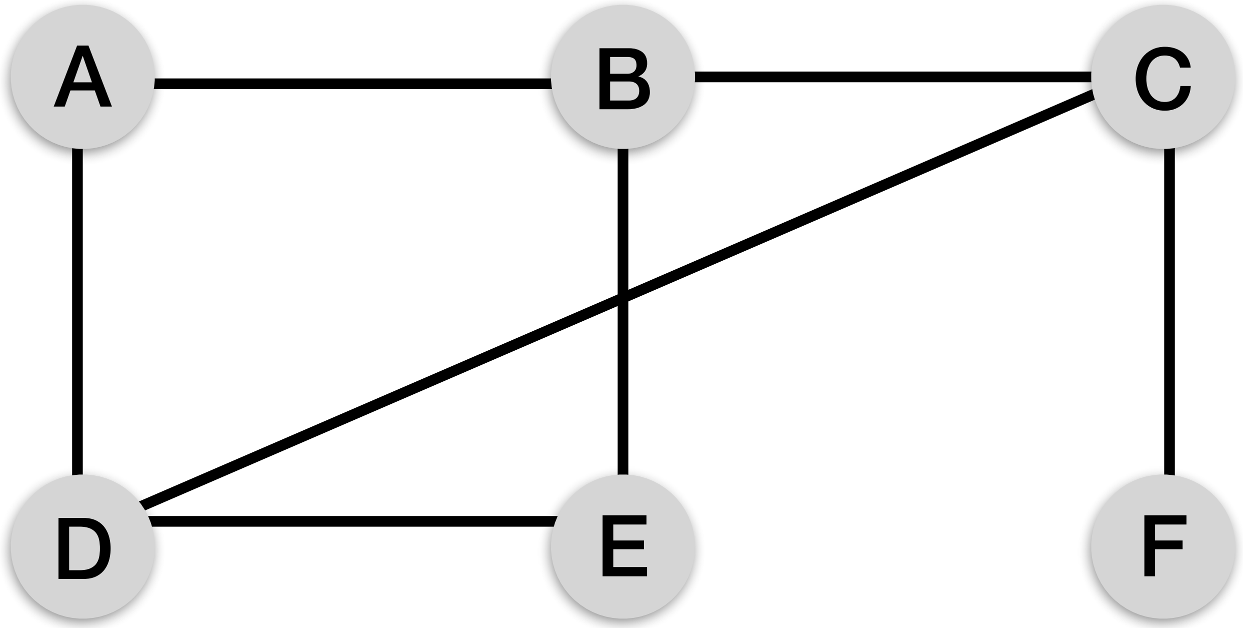
$\tilde{G} = (U \uplus V, \tilde{E})$

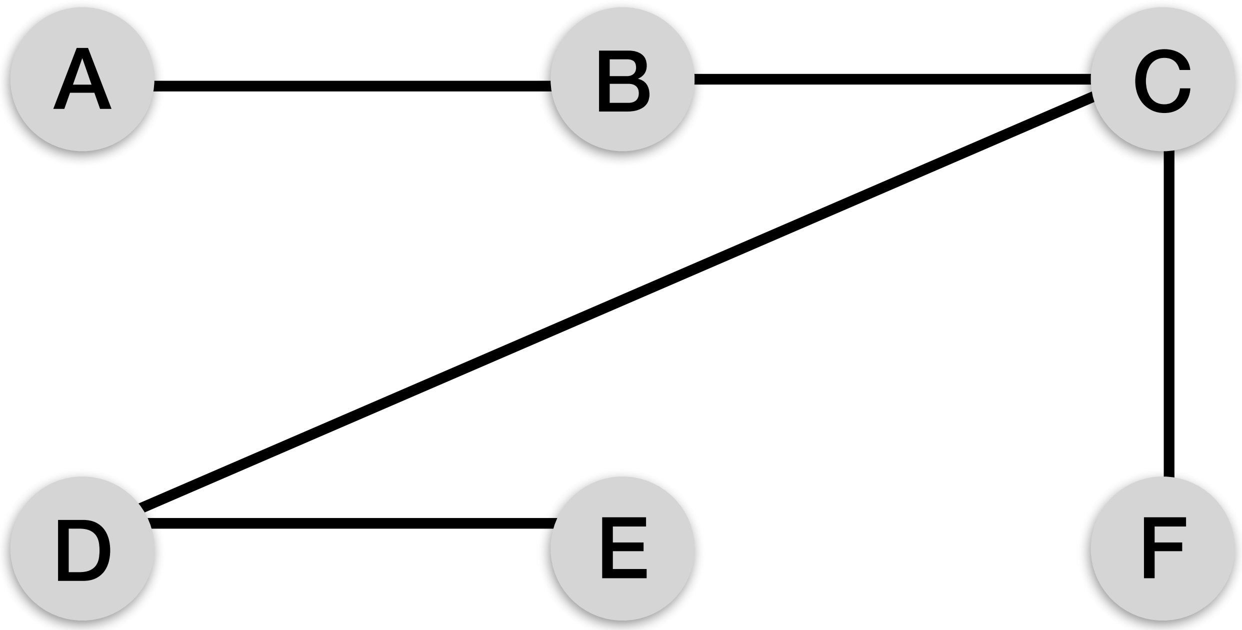


(*) A *bipartite graph* must not be connected, hence $\tilde{G}$ is also *bipartite*.

Bipartite?

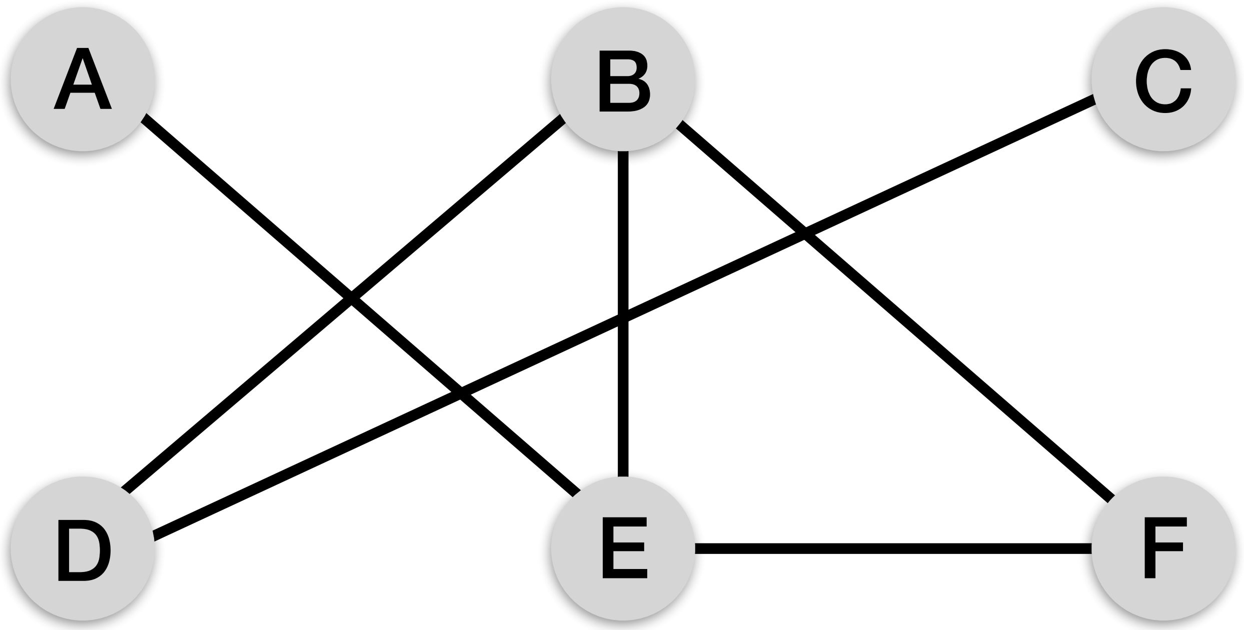






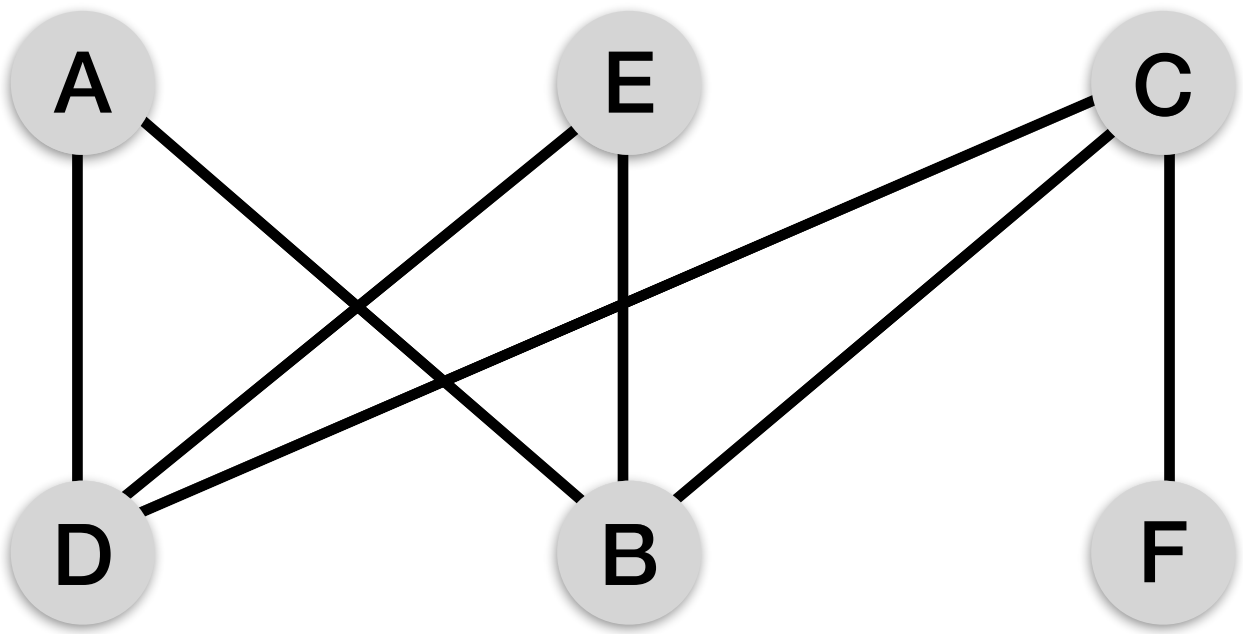
Bipartite?

(1)



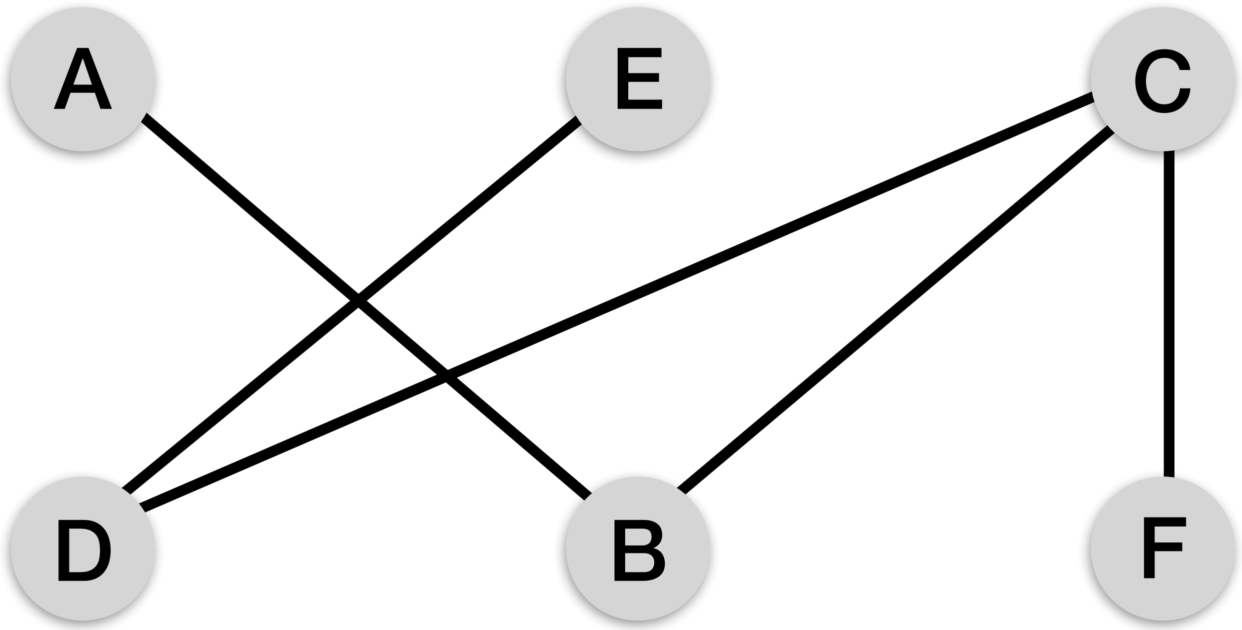
not bipartite

(2)



bipartite

(3)



bipartite

(*) The left graph contains an odd-length cycle $< B, E, F >$. A bipartite graph cannot contain any odd-length cycles!
(**) Proof left as an exercise (or click here).

Algorithms and Data Structures Recap

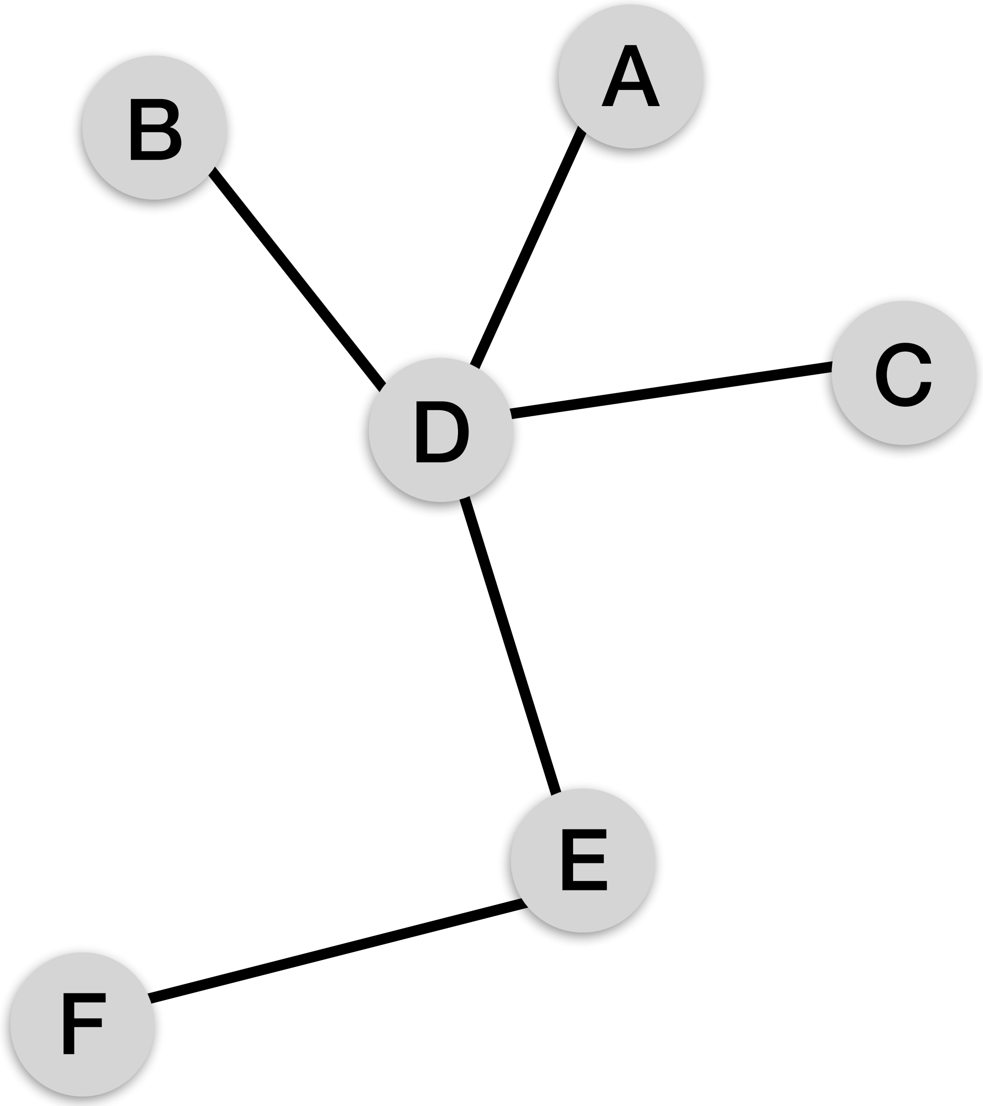
Trees

A *tree* is an undirected graph G that satisfies any of the following equivalent conditions:

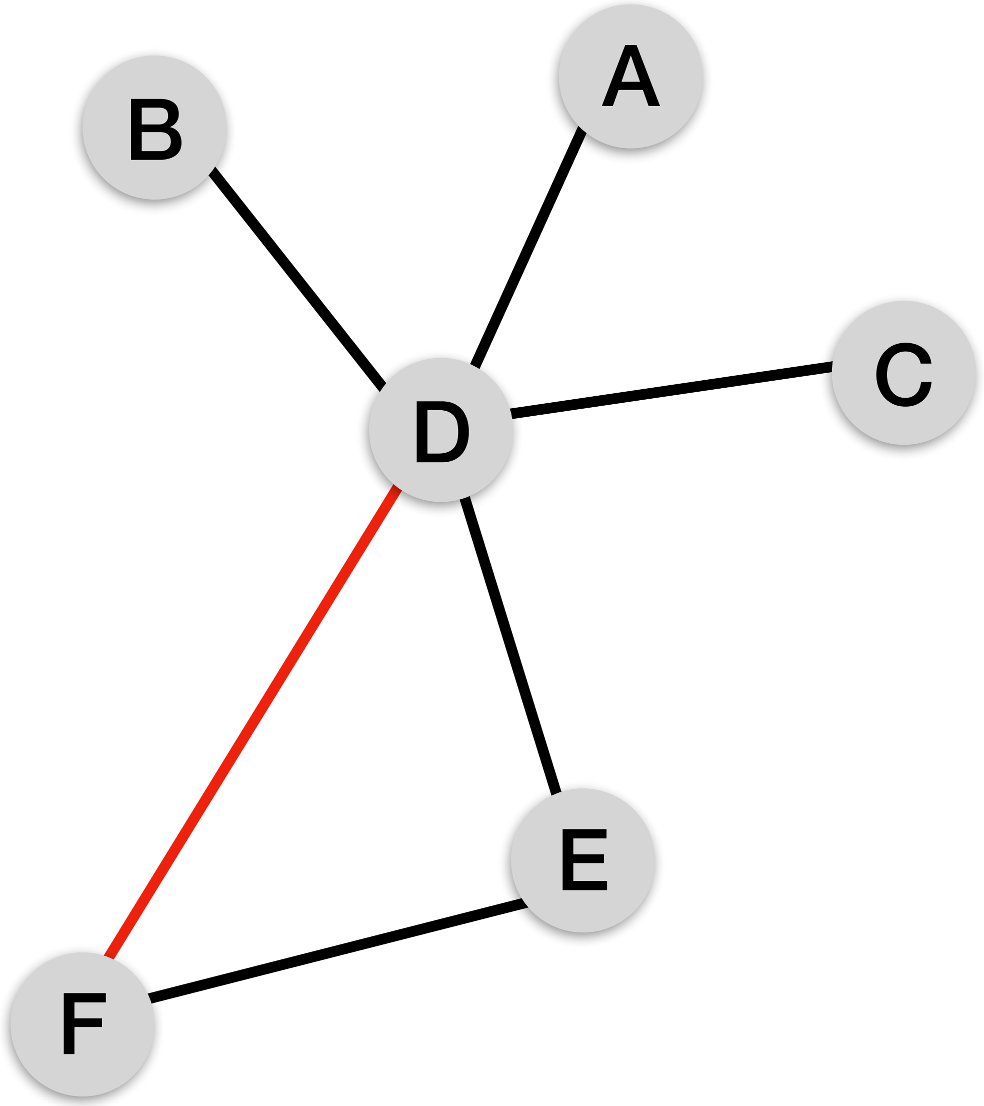
- G is connected and acyclic.
- G is acyclic and a simple cycle is formed if any edge is added.
- G is connected and has $n-1$ vertices.
- G is connected, but would be disconnected if single edge is removed.
- Any two vertices in G can be connected by a unique path.

(*) for the third bullet point we assume that G has a finite amount of vertices.

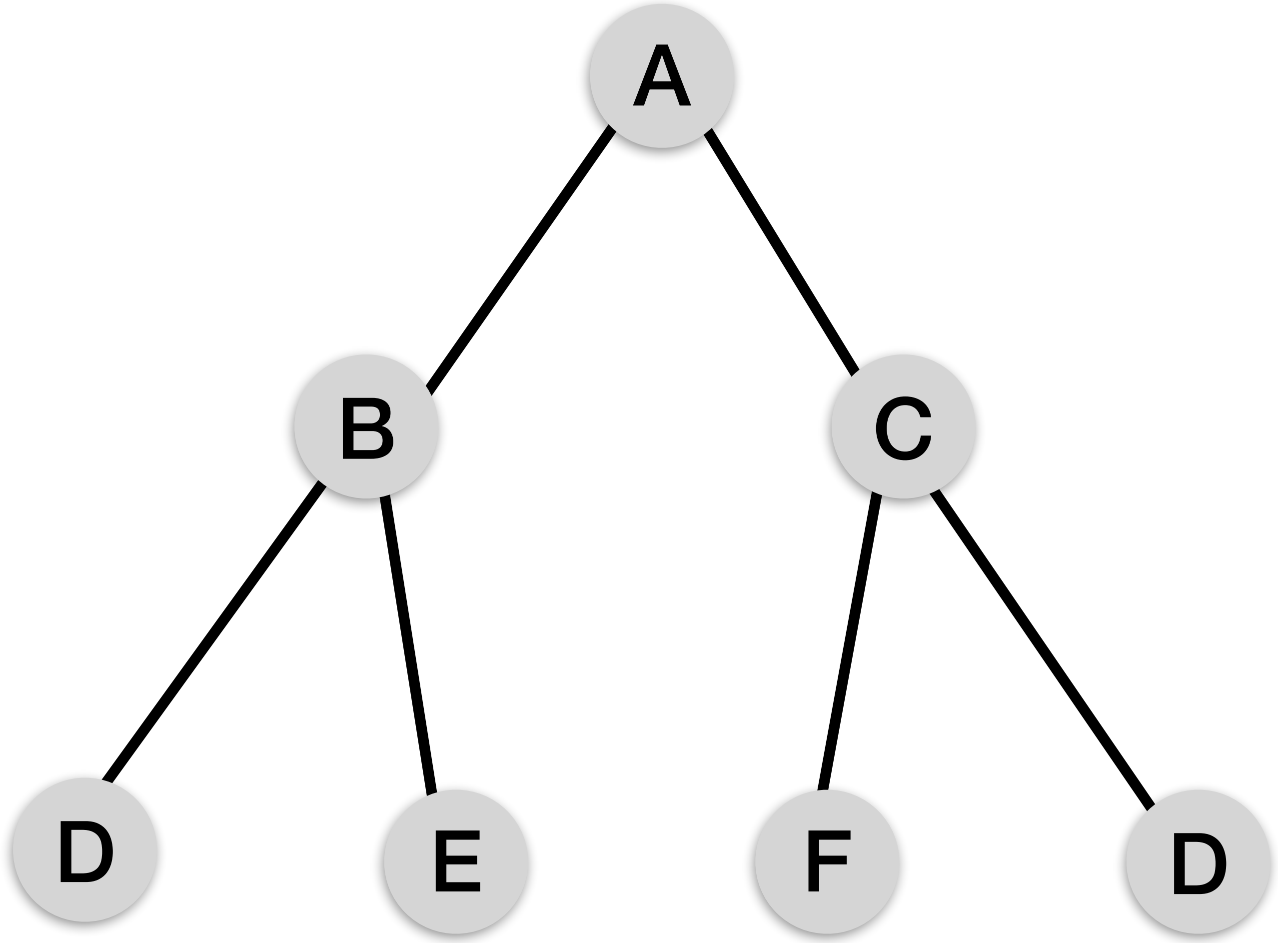
Examples I



$T_1 = (V, E)$



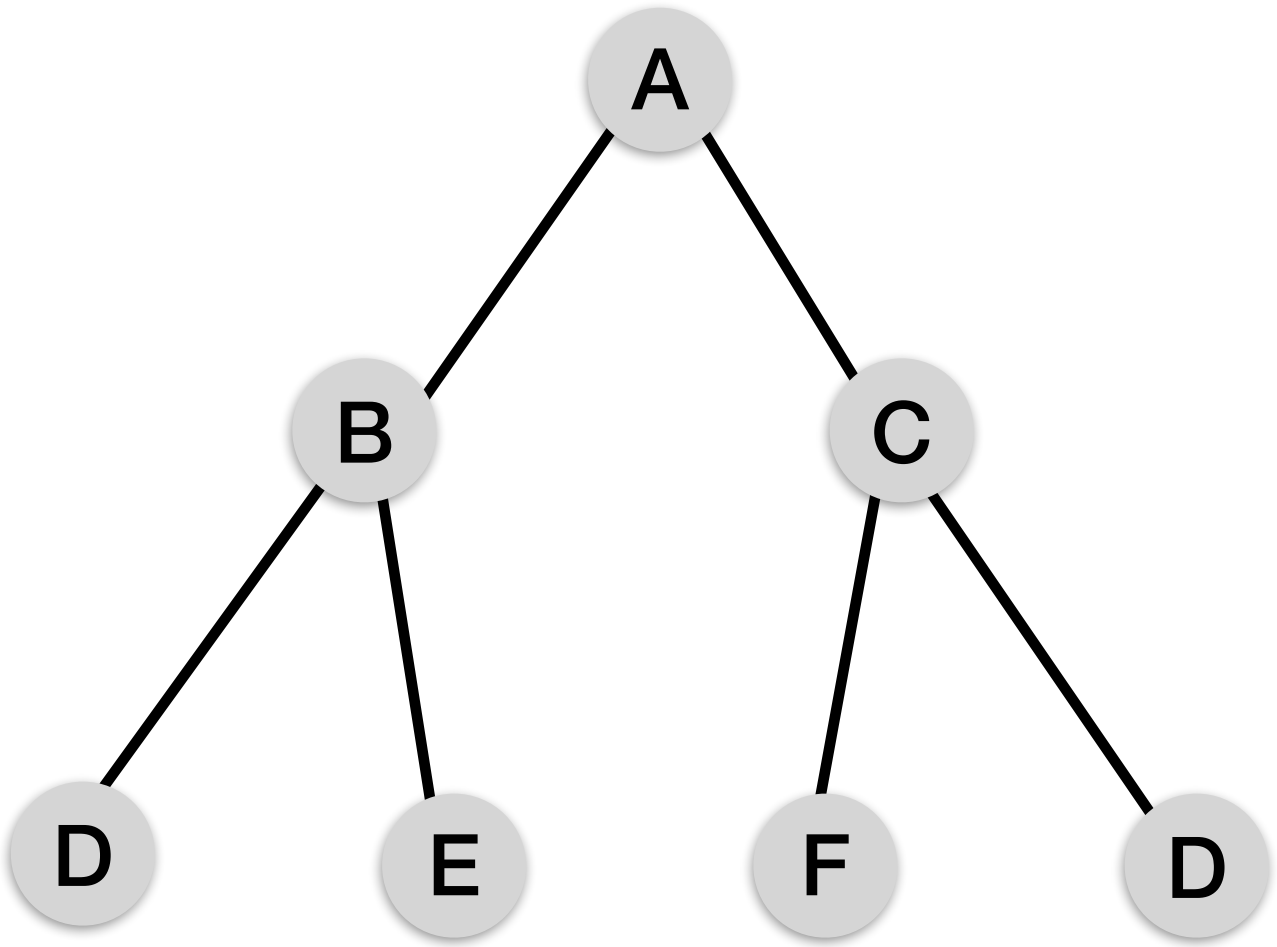
$G = (V, E)$



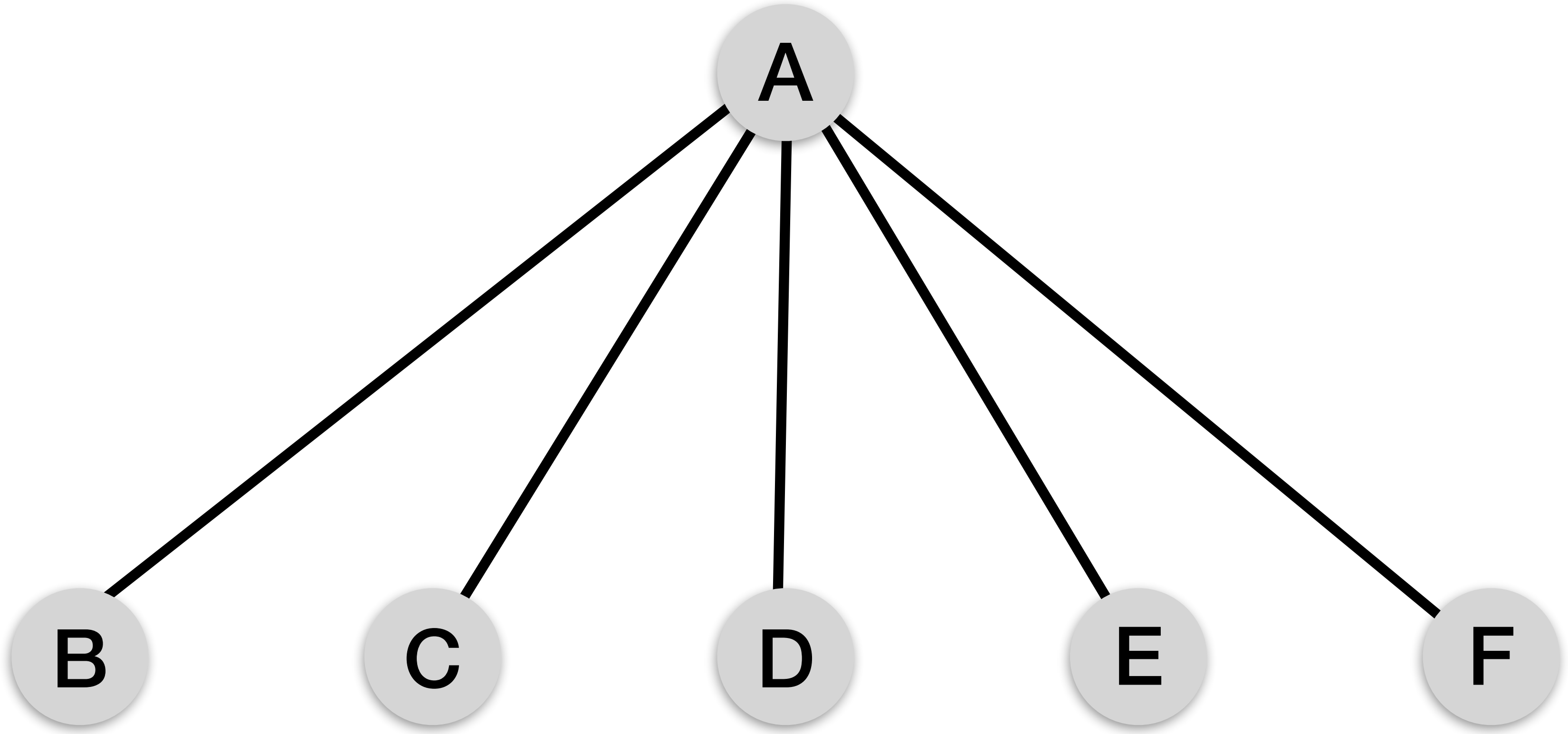
$T_2 = (V, E)$

(*) T_1 and T_2 are trees, T_2 is a *binary tree* since any node has at most two children.
(**) G is not a tree since (1) it has a cycle (2) it has $n = 6$ edges (instead of $n - 1 = 5$) (3) it would not be disconnected if the red edge were to be removed (4) D and F are connected by two paths.

Examples II



$T_1 = (V, E)$



$T_2 = (V, E)$

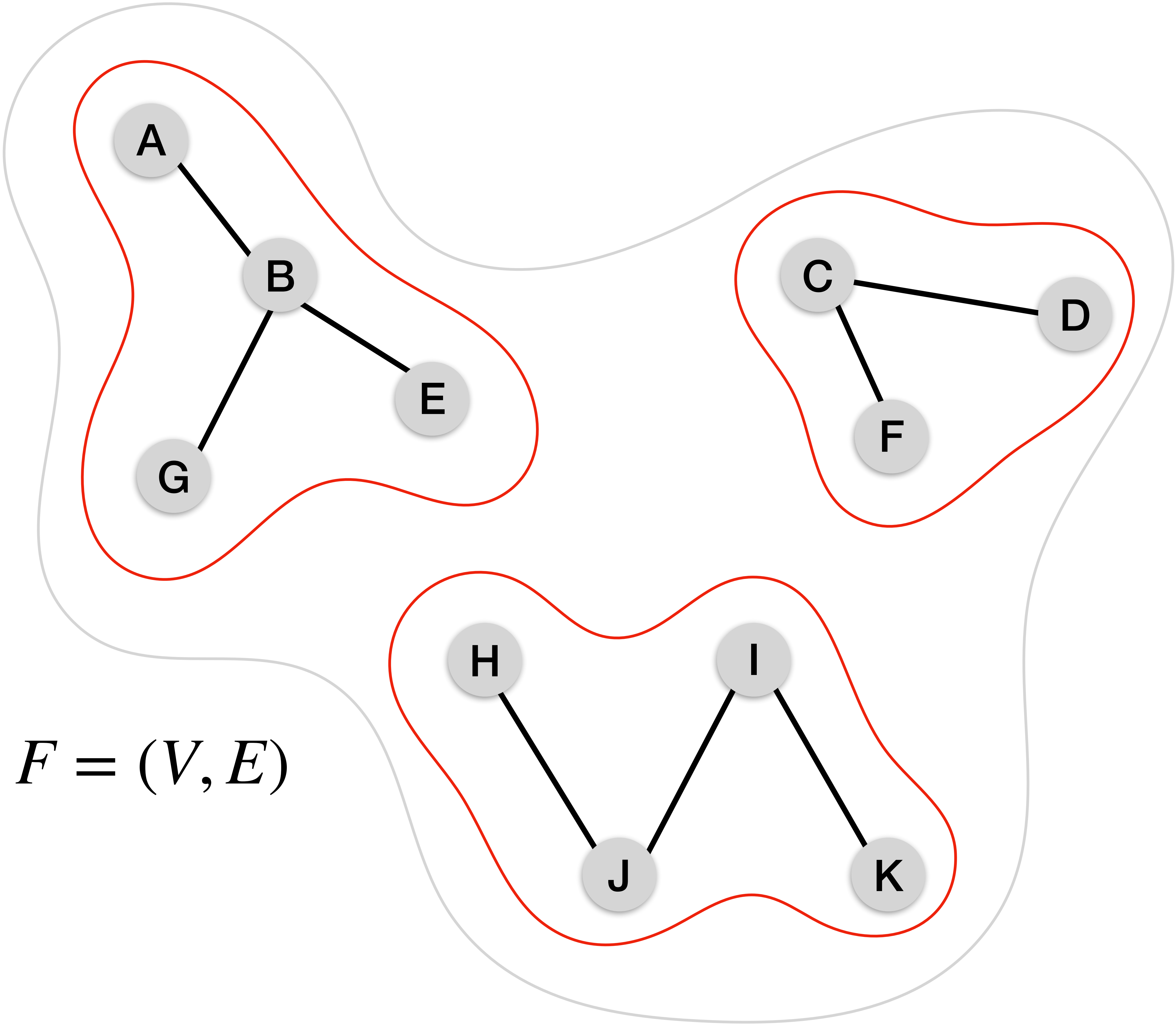
(*) Students sometimes think of only binary trees (e.g. T_1) when reasoning about trees. However, T_2 is a tree as well!

Algorithms and Data Structures Recap

Trees, connected components and forests

- A *leaf* is a vertex in a tree of degree one.
- A *connected component* of an undirected graph is a connected subgraph that is not part of any larger connected subgraph. The connected components of any graph partition its vertices into disjoint sets.
- A *forest* is an undirected and acyclic graph whose connected components are trees. In other words, it consists of a disjoint union of trees.

Examples



$$F = (V, E)$$

(*) F is a *forest*. The three *connected components* of F are in red blobs, each of the graphs contained in one of the red blobs is a tree.

(**) The subgraph on the vertices $\{C, D, F\}$ is a tree, and both F and D is a *leaf*.

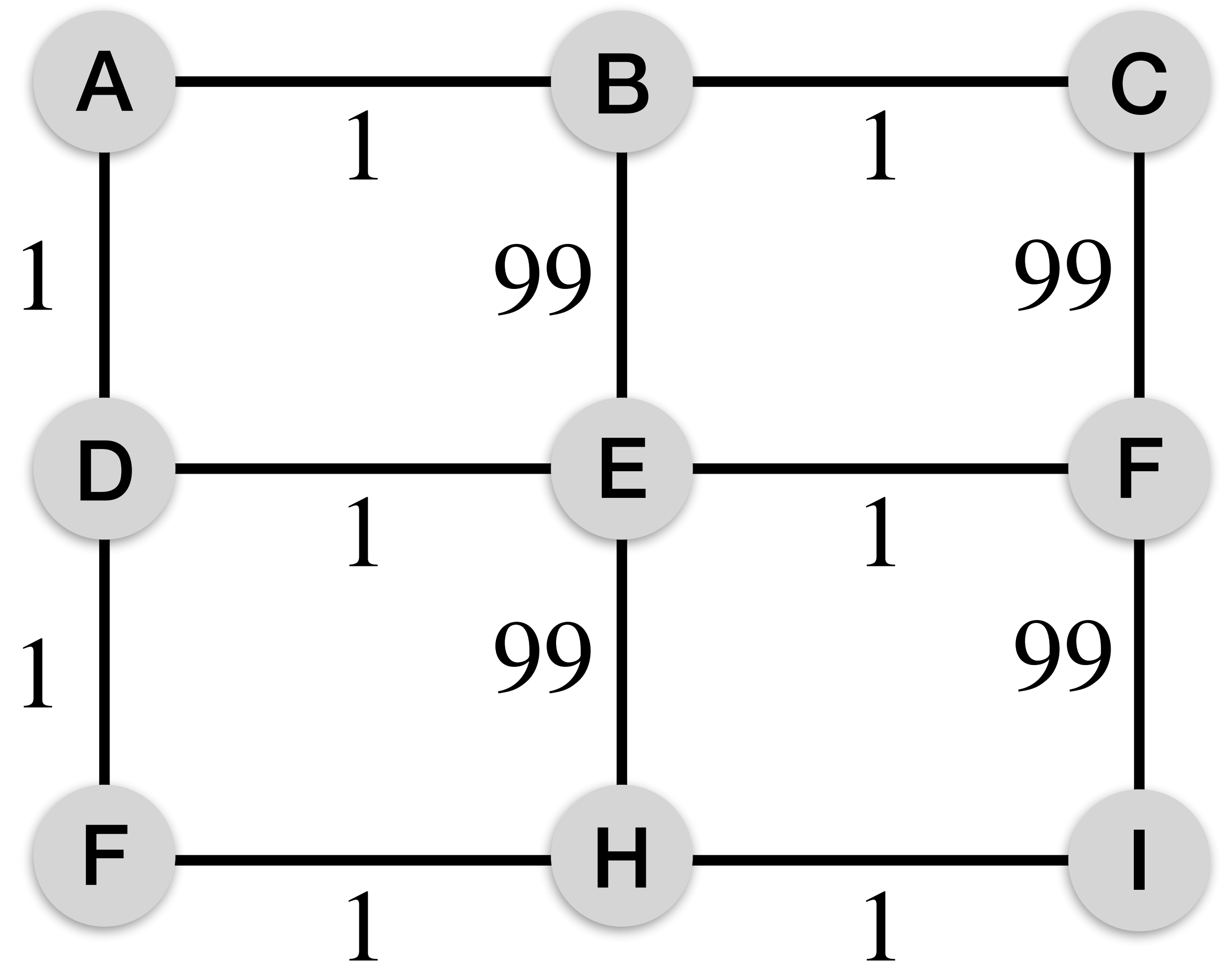
Algorithms and Data Structures Recap

Spanning trees

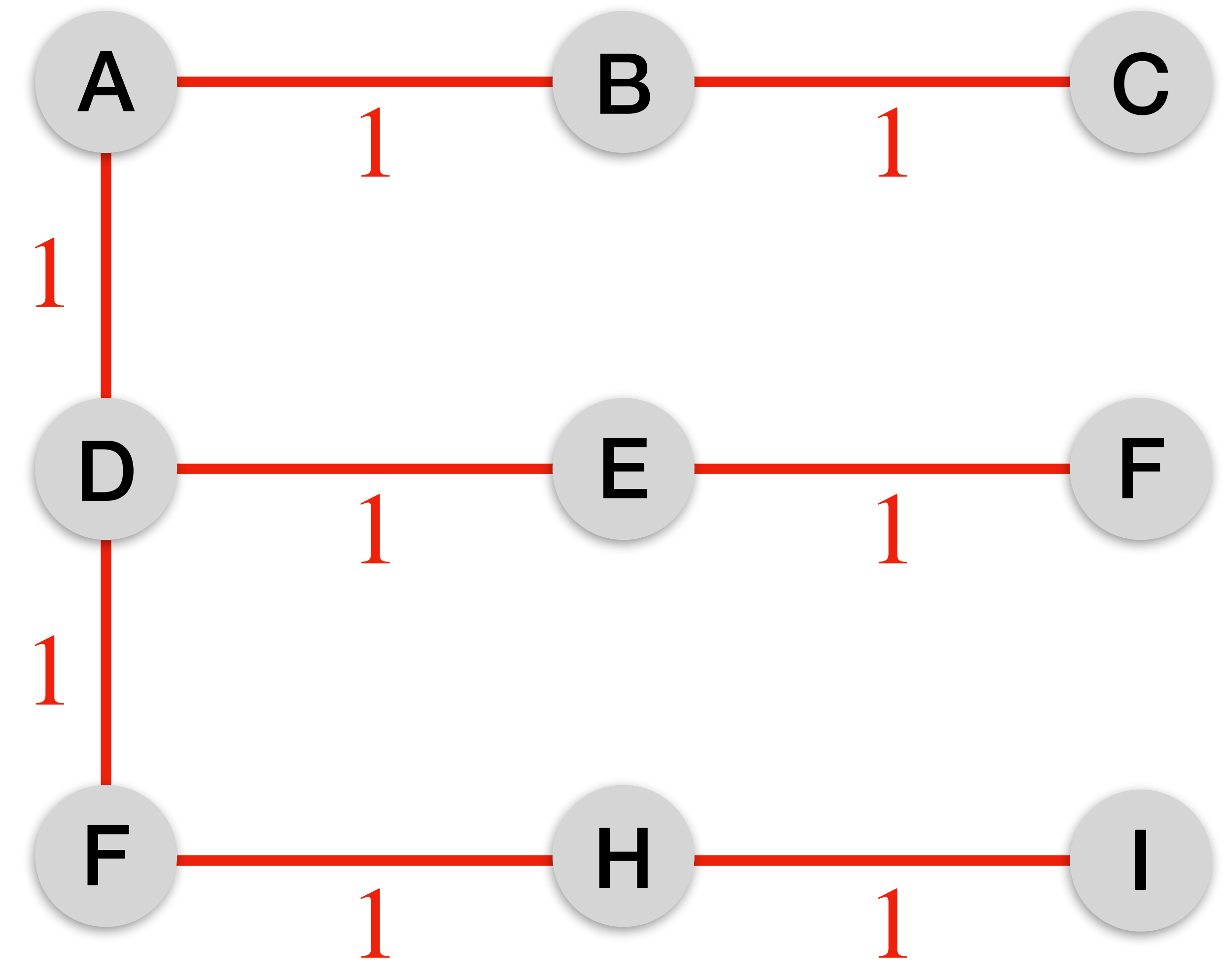
- A *spanning tree* T of an undirected graph G is a subgraph that is a tree which includes all the vertices of G.
- A *minimum spanning tree* (MST) is a spanning tree with the minimum possible sum of edge weights.

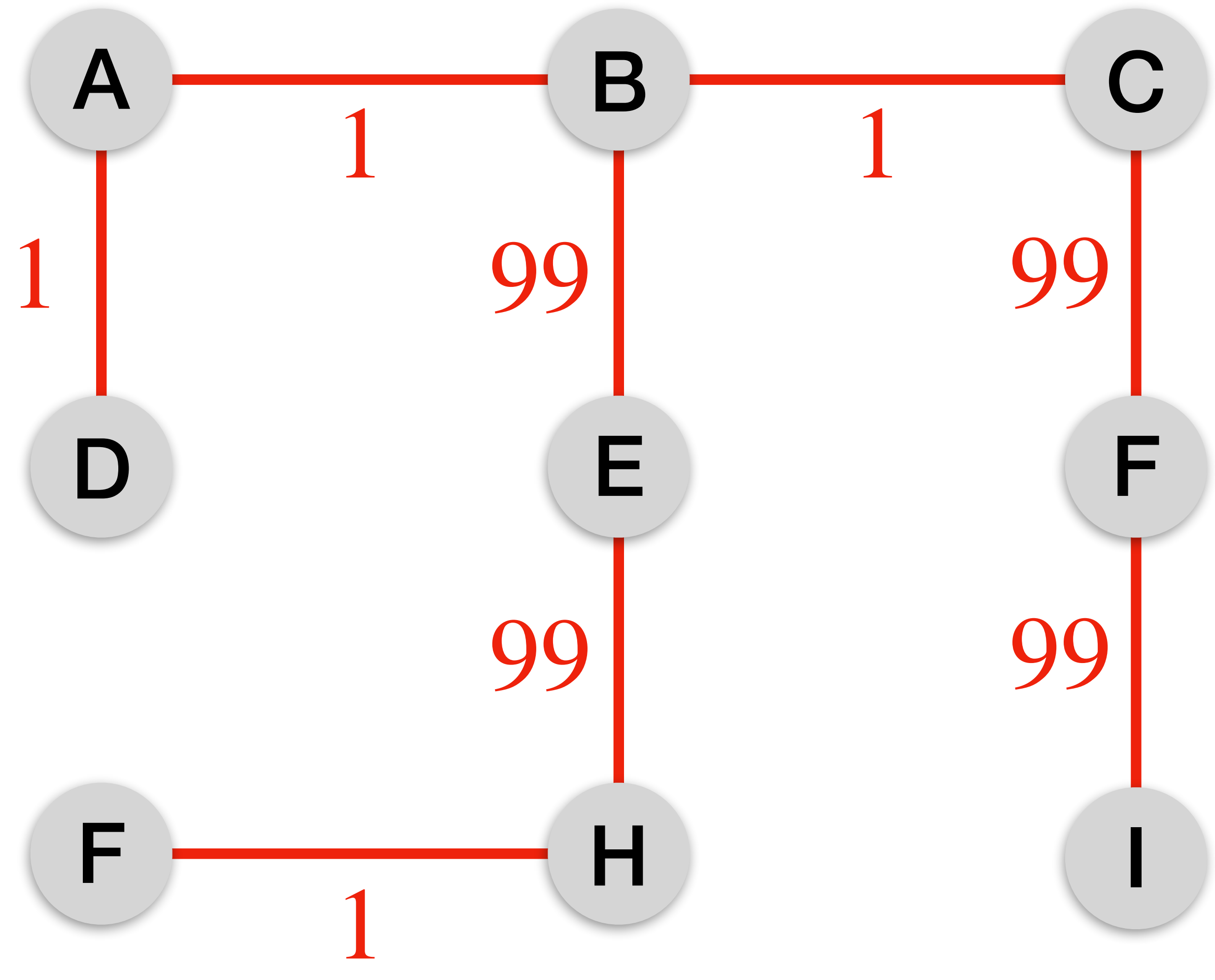
Examples

$G = (V, E)$



$T_1 = (V, E_2)$





$T_2 = (V, E_2)$

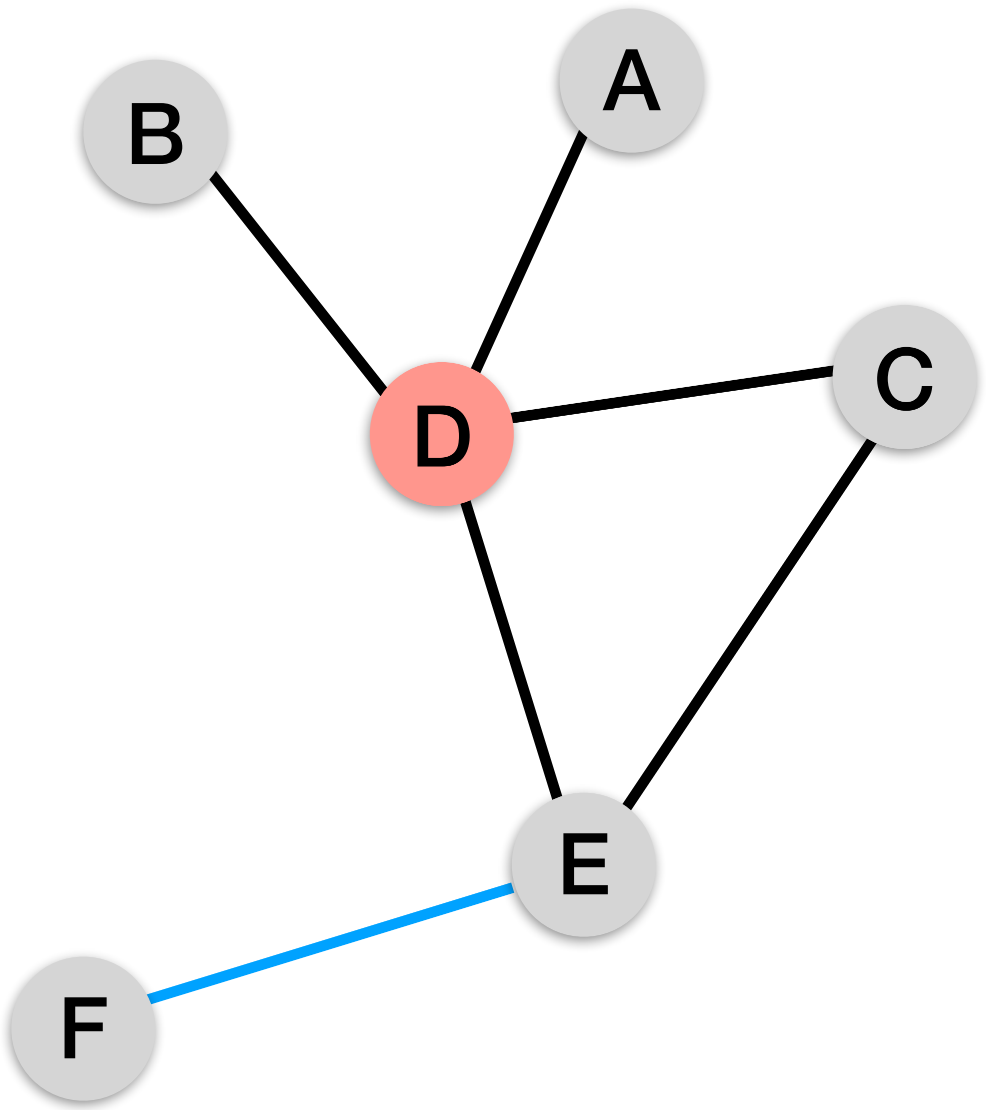
(*) Both T_1 and T_2 are *spanning trees* of G.
(**) T_1 is the unique *minimum spanning tree* of G of total weight 8.
(***) G is the so called *grid graph*.

Algorithms and Data Structures Recap

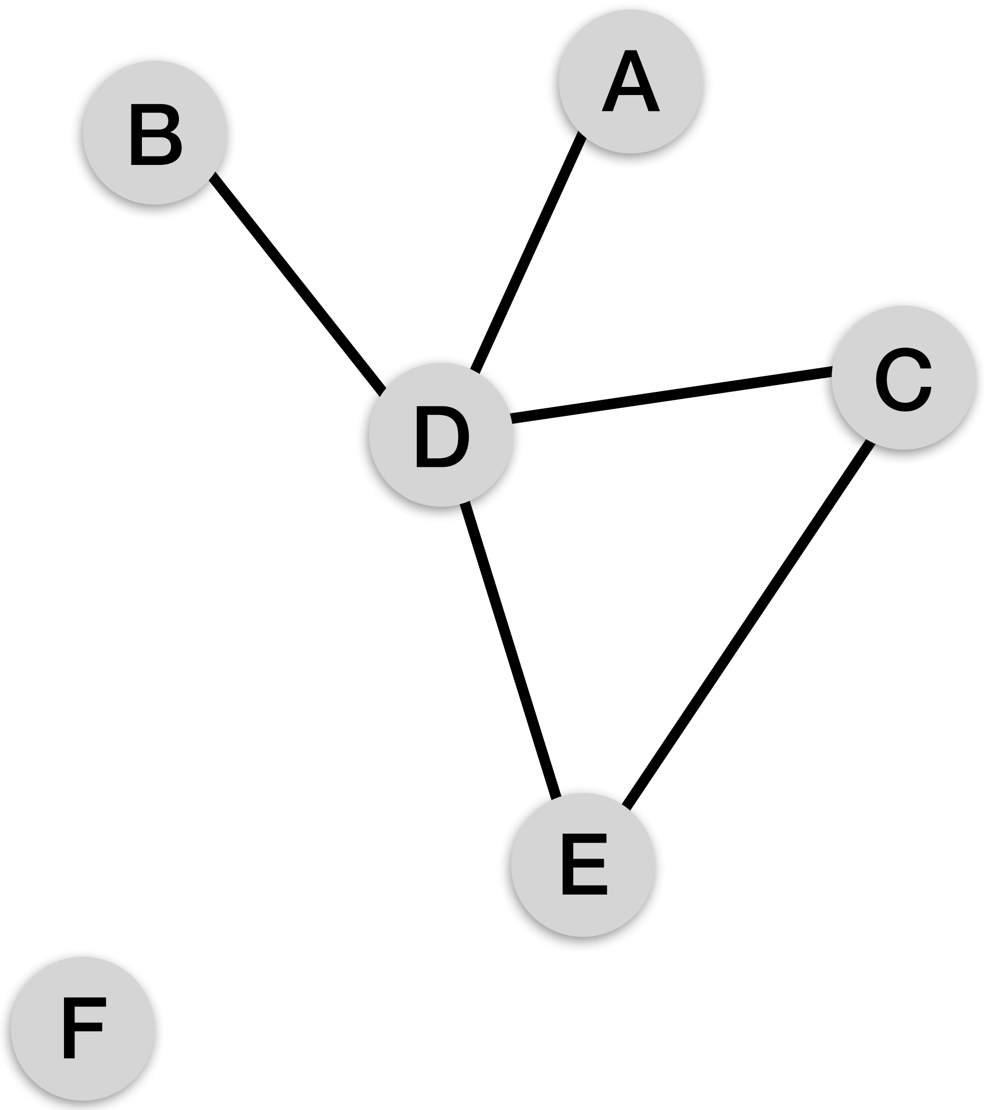
Cut vertices and bridges

- A *cut vertex* is any vertex whose removal increases the number of connected components.
- A *bridge* is any edge whose removal increases the number of connected components.

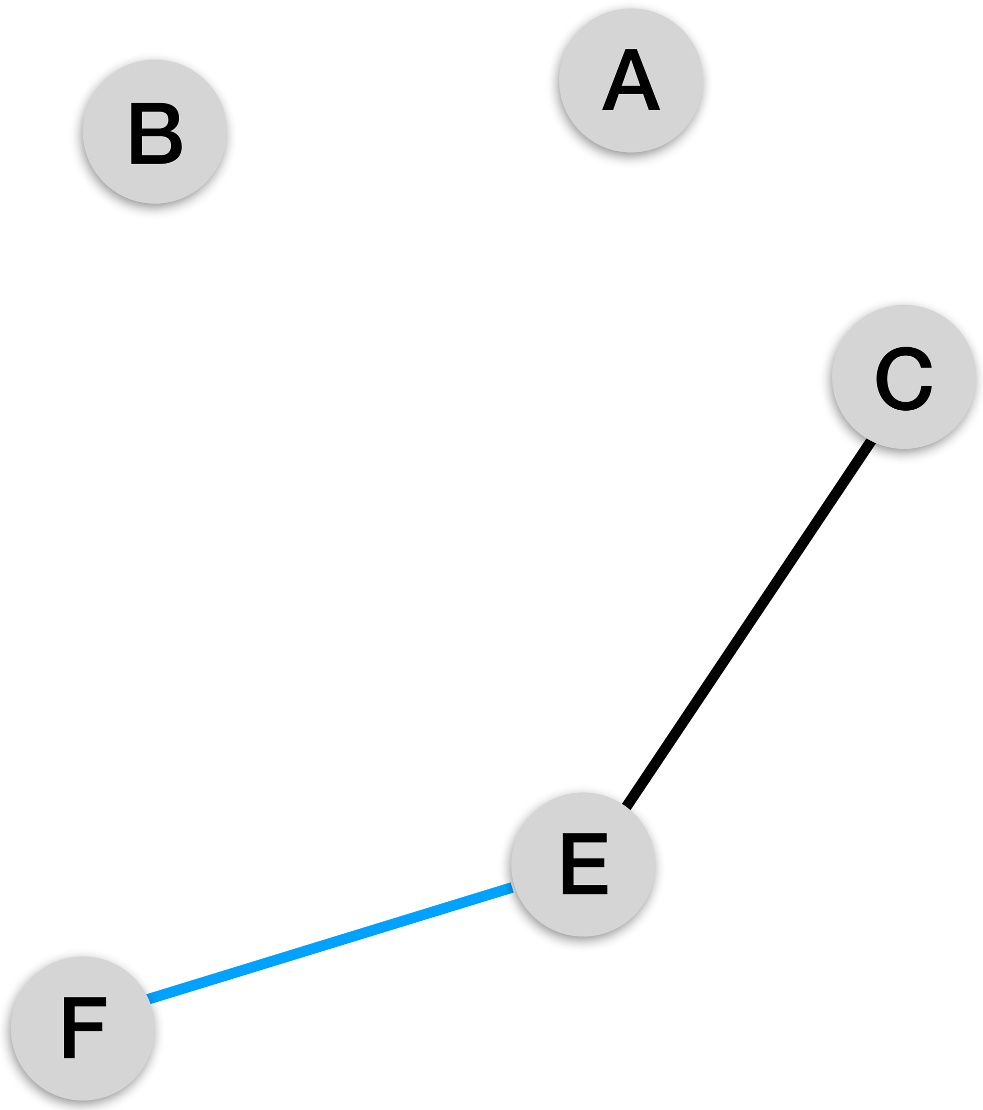
Examples



$G = (V, E)$



$G_1 = (V, E_1)$



$G_2 = (V_2, E_2)$

(*) The blue edge is a *bridge*. It's removal increases the number of connected components in G (which is originally one) to two (see graph G_1).
(**) The reddish vertex is a *cut vertex*. It's removal increases the number of connected components in G (which is originally one) to three (see in graph G_2).
(***) Note there are multiple cut vertices and bridges in G. The vertex C is *not* a cut vertex and the edge $\{E, C\}$ is *not* a bridge.

AnD Recap

BFS-Visit-Iterative(G, v)

$Q \leftarrow \emptyset$
Markiere v als aktiv
Enqueue(Q, v)
while $Q \neq \emptyset$ **do**
  $w \leftarrow$ Dequeue(Q)
  Markiere w als besucht
  for each $(w, x) \in E$ **do**
    if x nicht aktiv und x noch nicht besucht **then**
      Markiere x als aktiv
      Enqueue(Q, x)

Queue

Stack

DFS-Visit-Iterative(G, v)

$S \leftarrow \emptyset$
Push(S, v)
while $S \neq \emptyset$ **do**
  $w \leftarrow$ Pop(S)
  if w noch nicht besucht **then**
    Markiere w als besucht
    for each $(w, x) \in E$ in reverse order **do**
      if x noch nicht besucht **then**
        Push(S, x)

Solving graph exercises?

- Questions that I often received in my AnD exercise classes are
  - How do I solve graph exercises?
  - How rigorous/long should my proof be?
- Of course there is not a single “one size fits all” solution, but what I always tell them is, the more you know about graphs (e.g. properties of a graph, useful lemmas, equivalences…), the better! Let’s go through an example.

For any graph $G = (V, E)$ we have: the number of vertices with odd degree is even.

For any graph $G = (V, E)$ we have: the number of vertices with odd degree is even.

Theorem 1.2. For every graph $G = (V, E)$ we have $\sum_{v \in V} \deg(v) = 2 \cdot |E|$.

(*) knowing Theorem 1.2. things get a lot easier!

For any graph $G = (V, E)$ we have: the number of vertices with odd degree is even.

Theorem 1.2. For every graph $G = (V, E)$ we have $\sum_{v \in V} \deg(v) = 2 \cdot |E|$.

Proof. Let V_e and V_o be the vertex set of the vertices with even and odd degree.

We have $\sum_{v \in V} \deg(v) = \sum_{v \in V_e} \deg(v) + \sum_{v \in V_o} \deg(v)$.

The sum of all even degrees is even. The sum of k odd numbers is even if and only if k is even. Using Theorem 1.2. we conclude that $|V_o|$ is even.

(*) the first gray box is corollary 1.3. in the script.

Why proofs?

Why bother showing us proofs?

Why bother showing us proofs in more detail than the amount of detail used in the lecture?

Similar to the reasoning from before: **To equip you with as many ideas and concepts as possible so that you have access to them at the exam.**

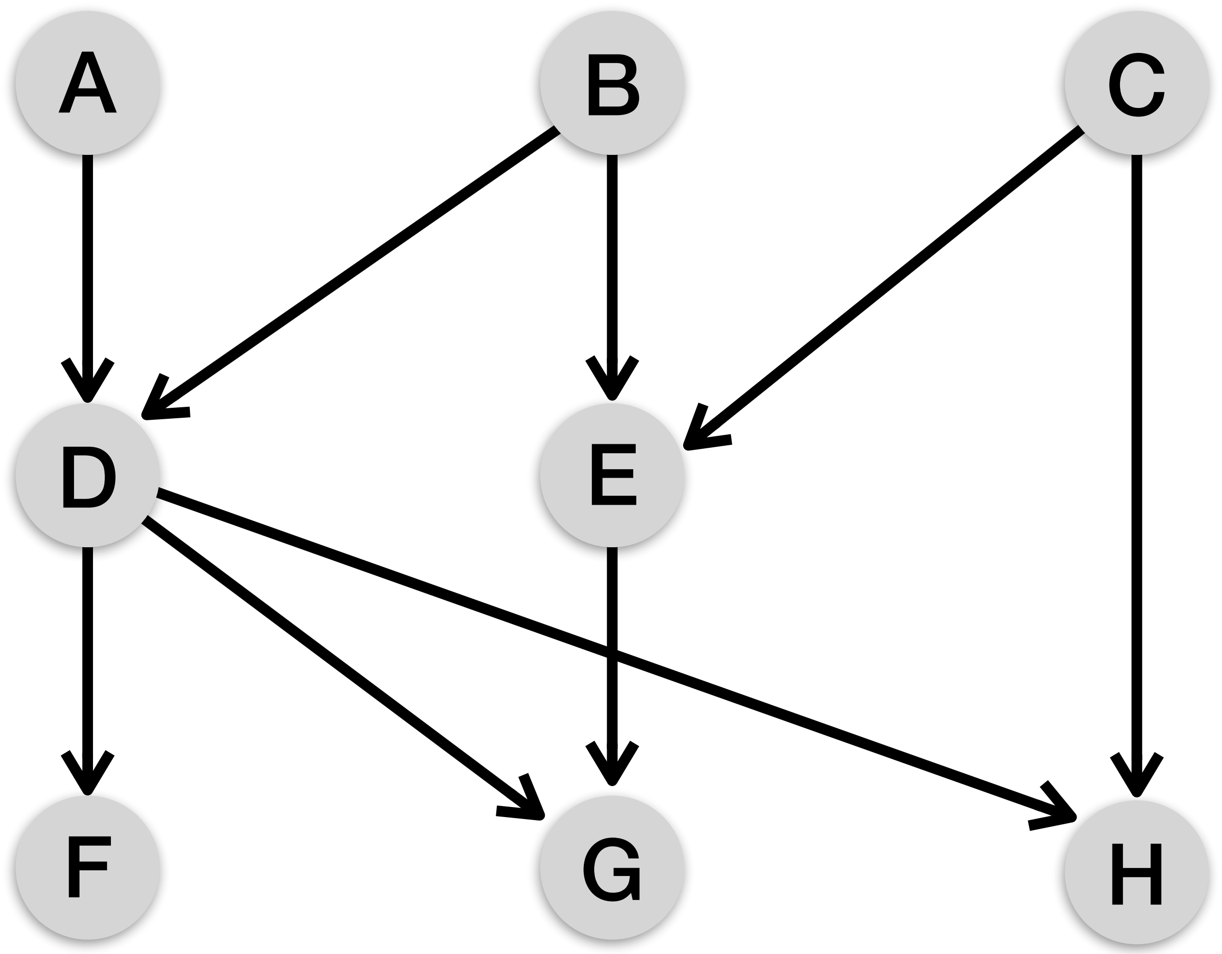
Algorithms and Data Structures Recap

Topological sort/ordering

- A *topological ordering* of a directed graph is a linear ordering of its vertices such that for every directed edge (u, v) from vertex u to vertex v, u comes before v in the ordering.
- A *directed acyclic graph* (DAG) is a directed graph without directed cycles.
- A topological ordering is only possible if and only if the graph is a DAG.

Example

$$G = (V, E)$$



$$A, B, C, D, E, F, G, H$$

(*) note that G has many topological orderings. Another one is C, A, B, E, D, F, G, H.