Graph Theory and Optimization
Introduction on Graphs

Nicolas Nisse

Université Côte d’Azur, Inria, CNRS, I3S, France

October 2018
Outline

1. Vertex/Edge
2. Neighbor/Degree
3. Path/Cycle
4. Trees
5. SubGraph
A graph $G = (V, E)$

Vertices: $V = V(G)$ is a finite set

Edges: $E = V(E) \subseteq \{\{u, v\} \mid u, v \in V\}$ is a binary relation on $V$

Example: $G = (V, E)$ with $V = \{a, b, c, d, e, f, g, h, i, j\}$ and $E = \{\{a, b\}, \{a, c\}, \{a, f\}, \{b, g\}, \{b, h\}, \{c, f\}, \{c, d\}, \{d, g\}, \{d, e\}, \{e, j\}, \{f, g\}, \{f, i\}, \{g, h\}, \{h, i\}, \{h, j\}, \{i, j\}\}$. 
A graph $G = (V, E)$

**Vertices:** $V = V(G)$ is a finite set

**Edges:** $E = V(E) \subseteq \{\{u, v\} \mid u, v \in V\}$ is a binary relation on $V$

Example: $G = (V, E)$ with $V = \{a, b, c, d, e, f, g, h, i, j\}$ and $E = \{\{a, b\}, \{a, c\}, \{a, f\}, \{b, g\}, \{b, h\}, \{c, f\}, \{c, d\}, \{d, g\}, \{d, e\}, \{e, j\}, \{f, g\}, \{f, i\}, \{g, h\}, \{h, i\}, \{h, j\}, \{i, j\}\}$.

**Exercise:** What is the maximum number of edges of a graph with $n$ vertices?
Outline

1. Vertex/Edge
2. Neighbor/Degree
3. Path/Cycle
4. Trees
5. SubGraph
Graph: terminology and notations (Neighbor/Degree)

- Two vertices $x \in V$ and $y \in V$ are adjacent or neighbors if $\{x, y\} \in E$.
  - i.e. there is an edge $\{x, y\}$.
- $N(x)$: set of neighbors of $x \in V$.
  - ex: $N(g) = \{b, d, f, h\} \subseteq V$.
- Degree of $x \in V$: number of neighbors of $x$.
  - i.e., $\deg(x) = |N(x)|$.

Exercise: Prove that, for any graph $G = (V, E)$, $\sum_{x \in V} \deg(x) = 2|E|$.
Graph: terminology and notations (Neighbor/Degree)

- two vertices $x \in V$ and $y \in V$ are adjacent or neighbors if $\{x, y\} \in E$ i.e. there is an edge $\{x, y\}$

- $N(x)$: set of neighbors of $x \in V$

- degree of $x \in V$: number of neighbors of $x$ i.e., $\deg(x) = |N(x)|$

**Exercise:** Prove that, for any graph $G = (V, E)$, $\sum_{x \in V} \deg(x) = 2|E|$
Outline

1. Vertex/Edge
2. Neighbor/Degree
3. Path/Cycle
4. Trees
5. SubGraph
**Path:** sequence \((v_1, \cdots, v_\ell)\) of distinct vertices such that consecutive vertices are adjacent, i.e., \(\{v_i, v_{i+1}\} \in E\) for any \(1 \leq i < \ell\)

**ex:** \(P_1 = (a, b, g, h, i)\)
Graph: terminology and notations (Path/Cycle)

- **Path**: sequence \((v_1, \cdots, v_\ell)\) of distinct vertices such that consecutive vertices are adjacent, i.e., \(\{v_i, v_{i+1}\} \in E\) for any \(1 \leq i < \ell\)

  **ex**: \(P_1 = (a, b, g, h, i)\), \(P_2 = (a, c, f, g, b, h, i, j, e)\)
Graph: terminology and notations (Path/Cycle)

- **Path**: sequence \((v_1, \cdots, v_\ell)\) of distinct vertices such that consecutive vertices are adjacent, i.e., \(\{v_i, v_{i+1}\} \in E\) for any \(1 \leq i < \ell\)

- **Cycle**: path \((v_1, \cdots, v_\ell)\) such that \(\ell \geq 3\) and \(\{v_1, v_\ell\} \in E\)

**ex:** \(C_1 = (d, e, j, i, h, g)\), \(C_2 = (a, c, f)\)
Graph: terminology and notations (Path/Cycle)

- **Path**: sequence \((v_1, \cdots, v_\ell)\) of distinct vertices such that consecutive vertices are adjacent, i.e., \(\{v_i, v_{i+1}\} \in E\) for any \(1 \leq i < \ell\)

- **Cycle**: path \((v_1, \cdots, v_\ell)\) such that \(\ell \geq 3\) and \(\{v_1, v_\ell\} \in E\)

- **\(G = (V, E)\)** is connected if, for every two vertices \(x \in V\) and \(y \in V\), there exists a path from \(x\) to \(y\).
**Path**: sequence \((v_1, \cdots, v_\ell)\) of distinct vertices such that consecutive vertices are adjacent, i.e., \(\{v_i, v_{i+1}\} \in E\) for any \(1 \leq i < \ell\)

**Cycle**: path \((v_1, \cdots, v_\ell)\) such that \(\ell \geq 3\) and \(\{v_1, v_\ell\} \in E\)

**G** = \((V, E)\) is **connected** if, for every two vertices \(x \in V\) and \(y \in V\), there exists a path from \(x\) to \(y\).

**Exercise**: Prove that if \(|E| < |V| - 1\) then \(G = (V, E)\) is NOT connected
Outline

1. Vertex/Edge
2. Neighbor/Degree
3. Path/Cycle
4. Trees
5. SubGraph
Graph: terminology and notations (Tree)

- **Tree**: connected graph without cycles
- **Leaf**: vertex of degree 1 in a tree

Trees are important because:

- "simple" structure + "minimum" structure ensuring connectivity

Theorem:

\[
T = (V, E)
\text{ is a tree } \iff T \text{ connected and } |V| = |E| + 1
\]
Graph: terminology and notations (Tree)

- **Tree**: connected graph without cycles
- **Leaf**: vertex of degree 1 in a tree

Trees are important because:

“simple" structure + “minimum" structure ensuring connectivity

**Theorem:**

\[ T = (V, E) \text{ is a tree} \iff T \text{ connected and } |V| = |E| + 1 \]
Graph: terminology and notations (Tree)

**Theorem:** \( T = (V, E) \) is a tree \( \iff \) \( T \) connected and \( |V| = |E| + 1 \)

\( \iff \) By contradiction:

- if \( T \) not a tree, then \( \exists \) a cycle \((v_1, \cdots, v_\ell)\)
- Let \( T' \) be obtained from \( T \) by removing edge \( \{v_1, v_\ell\} \)
- \( T' \) is connected \( \quad \) "technical" part, to be proved
- \( |E(T')| = |E| - 1 = |V| - 2 = |V(T')| - 2 \)
- so \( |E'| < |V'| - 1 \) and \( T' \) is not connected by previous Exercise

A contradiction
**Theorem:** $T = (V, E)$ is a tree $\iff T$ connected and $|V| = |E| + 1$

$\Rightarrow$ Induction on $|V|$  

- Let $P = (v_1, \cdots, v_\ell)$ be a longest path in $T$ ($\ell$ max., in particular $\ell \geq 2$)
- $v_1$ is a leaf. By contradiction:
  - assume $\deg(v_1) > 1$, and $x \in N(v_1) \setminus \{v_2\}$
  - $x \notin V(P)$ otherwise there is a cycle in $T$
  - then, $(x, v_1, \cdots, v_\ell)$ path longer than $P$, a contradiction

- then $S = T \setminus \{v_1\}$ is a tree  
  "technical" part, to be proved

- $|V(S)| < |V|$ so, by induction $|V(S)| = |E(S)| + 1$

- $|V| = |V(S)| + 1$ and $|E| = |E(S)| + 1$, so $|V| = |E| - 1$
Outline

1. Vertex/Edge
2. Neighbor/Degree
3. Path/Cycle
4. Trees
5. SubGraph
Subgraph of $G = (V, E)$: any graph $H = (V', E')$ with $V' \subseteq V$ and $E' \subseteq \{\{x, y\} \in E \mid x, y \in V'\}$ obtained from $G$ by removing some vertices and edges.
Graph: terminology and notations (subgraph)

- **Subgraph** of $G = (V, E)$: any graph $H = (V', E')$ with $V' \subseteq V$ and $E' \subseteq \{\{x, y\} \in E \mid x, y \in V'\}$

  obtained from $G$ by removing some vertices and edges
Subgraph of $G = (V, E)$: any graph $H = (V', E')$ with
$V' \subseteq V$ and $E' \subseteq \{ \{x, y\} \in E \mid x, y \in V' \}$

*obtained from $G$ by removing some vertices and edges*
Graph: terminology and notations (subgraph)

- **Subgraph** of $G = (V, E)$: any graph $H = (V', E')$ with $V' \subseteq V$ and $E' \subseteq \{ \{x, y\} \in E \mid x, y \in V' \}$

  obtained from $G$ by removing some vertices and edges

$$
\begin{align*}
& a \\
& b \\
& c \\
& g \\
& h \\
& f \\
& e \\
& k
\end{align*}
$$
Graph: terminology and notations (subgraph)

- **Subgraph** of $G = (V, E)$: any graph $H = (V', E')$ with $V' \subseteq V$ and $E' \subseteq \{ \{x, y\} \in E \mid x, y \in V' \}$

  obtained from $G$ by removing some vertices and edges
Graph: terminology and notations (subgraph)

- **Subgraph**: $H = (V', E')$ with $V' \subseteq V$ and $E' \subseteq \{\{x, y\} \in E \mid x, y \in V'\}$

- **Spanning subgraph of $G$**: subgraph $H = (V', E')$ where $V' = V$
  
  obtained from $G$ by removing only some edges
Graph: terminology and notations (subgraph)

- **Subgraph**: $H = (V', E')$ with $V' \subseteq V$ and $E' \subseteq \{\{x, y\} \in E \mid x, y \in V'\}$
- **Spanning** subgraph of $G$: subgraph $H = (V', E')$ where $V' = V$
  
  obtained from $G$ by removing only some edges
- **Spanning tree** of $G$: spanning subgraph $H = (V, E')$ with $H$ a tree
Graph: terminology and notations (subgraph)

- **Subgraph**: $H = (V', E')$ with $V' \subseteq V$ and $E' \subseteq \{\{x, y\} \in E \mid x, y \in V'\}$
- **Spanning subgraph of $G$**: subgraph $H = (V', E')$ where $V' = V$
  obtained from $G$ by removing only some edges
- **Spanning tree of $G$**: spanning subgraph $H = (V, E')$ with $H$ a tree

**Exercise**: A graph $G$ is connected if and only if $G$ has a spanning tree
Summary: To be remembered

All definitions will be important in next lectures
Please remember:

- graph, vertex/vertices, edge
- neighbor, degree
- path, cycle
- connected graph
- tree
- subgraph, spanning subgraph