Introduction to Graph Algorithms

Data Structures and Algorithms (094224)

Yuval Emek

Winter 2022/23

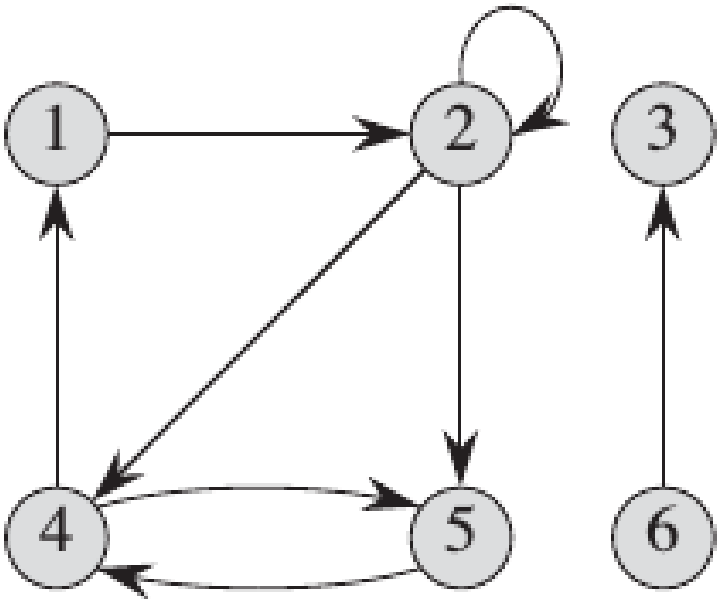
The role of graphs

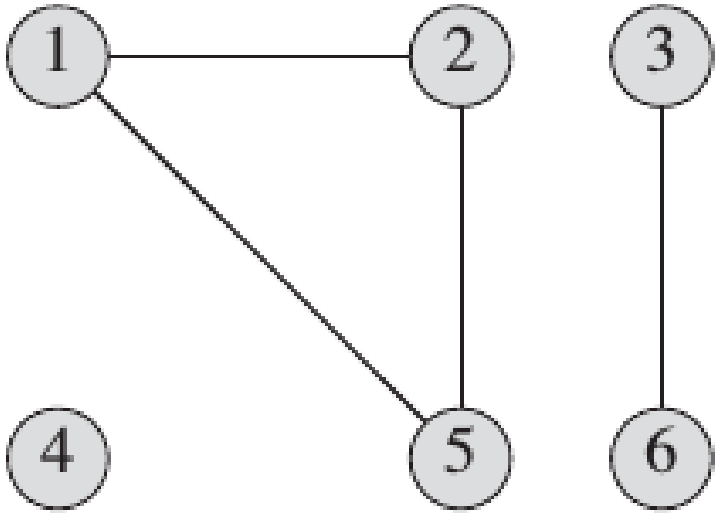
- A fundamental combinatorial structure
- A mathematical model for communication networks, street and road maps, power grids, VLSI circuits, biological cellular networks
- Abstraction for pairwise relations between the components of a system
- Fundamental to discrete algorithms
  - An abundance of algorithms on (and data structures for) graphs
  - Provide useful insights for many other algorithms and data structures

Directed and Undirected graphs

- *Graph* ⟨גרף⟩ $G = (V, E)$
  - V = a set of *vertices* ⟨צמתים⟩ (or *nodes* ⟨קודקודים⟩)
  - E = a set of vertex pairs referred to as *edges* ⟨קשתות⟩ (or *arcs* ⟨צלעות⟩)
  - Unless stated otherwise, our graphs are *finite* ⟨סופיים⟩: V, E finite sets
- *Directed* ⟨מכוון⟩ graph (or *digraph*):
  - The edges are ordered pairs
  - $(u, v) \neq (v, u)$
  - $E \subseteq V \times V$ is a (binary) relation over V
  - Edge (v, v) is possible — called a *self-loop* ⟨לולאה עצמית⟩
    - The directed graph is called *simple* ⟨פשוט⟩ if it doesn't have self-loops
- *Undirected* ⟨לא מכוון⟩ graph:
  - The edges are unordered pairs
  - $(u, v) = (v, u)$
  - Similar to symmetric relation, but strictly speaking, not a relation at all
  - Unless stated otherwise, no self-loops
- (Un)directed graph: u, v are the *endpoints* ⟨קצוות⟩ of edge (u, v)

Pictorial representation of graphs





Incidence, adjacency, and degrees

- $e = (u, v) \in E$ in a directed graph:
  - e is *incident from* or *leaves* ⟨יוצאת⟩ u and *incident to* or *enters* ⟨נכנסת⟩ v
  - v is *adjacent* ⟨סמוך⟩ to u
- $e = (u, v) \in E$ in an undirected graph:
  - e is *incident on* ⟨נוגעת⟩ u and v
  - u and v are *adjacent* ⟨סמוכים⟩ to each other
  - u and v are *neighbors* ⟨שכנים⟩
- Directed graph:
  - *out-degree* ⟨דרגת יציאה⟩ of v = #edges incident from v
  - *in-degree* ⟨דרגת כניסה⟩ of v = #edges incident to v
  - *degree* ⟨דרגה⟩ of v = in-degree of v + out-degree of v
- Undirected graph:
  - *degree* ⟨דרגה⟩ of v = #edges incident on v
- Questions:
  - In directed graphs, what is $\sum_{v \in V} \deg_{\text{in}}(v)$? $\sum_{v \in V} \deg_{\text{out}}(v)$?
  - In undirected graphs, what is $\sum_{v \in V} \deg(v)$?

תותשקה רפסמ

Paths

- The vertex sequence $P = \langle v_0, v_1, \ldots, v_k \rangle$ is a *path* ⟨מסלול⟩ in $G = (V, E)$ if $(v_{i-1}, v_i) \in E$ for every $1 \leq i \leq k$
  - The path leads from v_0 to v_k
- If such a path P exists, then v_k is *reachable* ⟨נגיש⟩ from v_0
- The *length* ⟨אורך⟩ of path P is k
- We say that path P *contains* ⟨מכיל⟩ vertices $v_0, v_1, \ldots, v_k$ and edges $(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$
- Path P is *simple* ⟨פשוט⟩ if $v_0, v_1, \ldots, v_k$ are distinct
- A *subpath* ⟨תת־מסלול⟩ of path P is a contiguous subsequence of the vertex sequence $\langle v_0, v_1, \ldots, v_k \rangle$
  - A path by its own right

From a path to a simple path

Lemma

If $G = (V, E)$ admits a path from u to v, then $G = (V, E)$ admits a simple path from u to v.

Proof. Let $P = \langle x_0, x_1, \ldots, x_k \rangle$ be a shortest path from $u = x_0$ to $v = x_k$

- Assume by contradiction that P is not simple
  - $x_i = x_j$ for some $0 \leq i < j \leq k$
- Let $P' = \langle x_0, x_1, \ldots, x_i = x_j, x_{j+1}, \ldots, x_k \rangle$
- P' is also a path from u to v
- $\text{length}(P') < \text{length}(P)$
- Contradiction! $\Longrightarrow$ assumption must be wrong ■

Cycles

- Path $C = \langle v_0, v_1, \ldots, v_k \rangle$ in a directed graph forms a *cycle* $\langle$**מעגל**$\rangle$ if
  - $v_0 = v_k$; and
  - $k \geq 1$
- Path $C = \langle v_0, v_1, \ldots, v_k \rangle$ in an undirected graph forms a *cycle* $\langle$**מעגל**$\rangle$ if
  - $v_0 = v_k$;
  - $k \geq 3$; and
  - $v_{i-1} \neq v_{i+1}$ for every $0 < i < k$
- Directed graph: cycle contains at least 1 edge
  Undirected graph: cycle contains at least 3 edges
- Cycle C is *simple* $\langle$**פשוט**$\rangle$ if $v_1, \ldots, v_k$ are distinct
- Paths $\langle v_0, v_1, \ldots, v_{k-1}, v_0 \rangle$ and $\langle v'_0, v'_1, \ldots, v'_{k-1}, v'_0 \rangle$ form the same cycle if $\exists j \in \mathbb{Z}$ such that $v'_i = v_{i+j \pmod k}$ for $i = 0, 1, \ldots, k-1$
- An (un)directed graph with no cycles is called *acyclic* $\langle$**חסר מעגלים**$\rangle$

Subgraphs

- (Un)directed graph $G = (V, E)$
- $G' = (V', E')$ is a *subgraph* ⟨תת־גרף⟩ of G if $V' \subseteq V$ and $E' \subseteq E$
  - A graph by its own right
- The subgraph of G *induced* ⟨מושרה⟩ by $V' \subseteq V$ is $G' = (V', E')$, where $E' = \{(u, v) \in E \mid u, v \in V'\}$
- A subgraph $G' = (V', E')$ of G is *spanning* ⟨פורש⟩ if $V' = V$

Connectivity

- Undirected graph $G = (V, E)$
  - Called *connected* ⟨קשיר⟩ if v is reachable from u for every $u, v \in V$
  - The equivalence classes of the "is reachable from" relation on the vertices are called *connected components* ⟨רכיבים קשירים⟩
- Directed graph $G = (V, E)$
  - Called *strongly connected* ⟨קשיר היטב⟩ if v is reachable from u for every $u, v \in V$
  - The equivalence classes of the "are mutually reachable" relation on the vertices are called *strongly connected components* ⟨רכיבים קשירים היטב⟩

Number of edges in connected graphs

Lemma

If undirected graph $G = (V, E)$ is connected, then $|E| \geq |V| - 1$.

Proof. By induction on $|V|$.

- Base: holds (with equation) for $|V| = 1$
- Hypothesis: holds for any graph with n vertices
- Step: consider a connected graph $G = (V, E)$ with $|V| = n + 1$
  - Assume by contradiction that $|E| < |V| - 1$
  - Sum of degrees $< 2|V| - 2 \Longrightarrow$ average degree < 2
  - Pigeonhole principle: $\exists v \in V$ of degree $\deg(v) < 2$
  - $\deg(v) = 1$ (can't be 0 as G is connected)
  - $G' =$ the subgraph of G induced by $V \setminus \{v\}$

The proof continues

- Inductive step — cont.
  - G' is connected
    - $\deg(v) = 1$ means that v can't be an interior vertex in any simple path
    - Removing it from G doesn't affect reachability between other vertices
  - G' has $|V| - 1$ vertices and $|E| - 1$ edges
  - Inductive hypothesis: $|E| - 1 \geq |V| - 2 \implies |E| \geq |V| - 1$
  - Contradiction! $\implies$ assumption must be wrong ■

(Un)directed versions

- The *directed version* $\langle$**גרסה מכוונת**$\rangle$ of an undirected graph $G = (V, E)$ is the directed graph $G' = (V', E')$ such that
  - $V' = V$
  - $E' = \{(u, v) \mid (u, v) \in E\}$
  - Observe: $|E'| = 2|E|$
- The *undirected version* $\langle$**גרסה לא מכוונת**$\rangle$ of a directed graph $G = (V, E)$ is the undirected graph $G' = (V', E')$ such that
  - $V' = V$
  - $E' = \{(u, v) \mid (u, v) \in E, u \neq v\}$
  - Observe: $|E'| \leq |E|$

Important graph classes

Undirected graphs:

- A *complete* ⟨מלא⟩ graph $G = (V, E)$:
  - $E = \{(u, v) \mid u, v \in V, u \neq v\}$
- A *bipartite* ⟨דו צדדי⟩ graph $G = (V, E)$:
  - V can be partitioned into $V = V_1 \cup V_2$, $V_1 \cap V_2 = \emptyset$, such that $E \subseteq V_1 \times V_2$
- An acyclic undirected graph is called a *forest* ⟨יער⟩
- A connected forest is called a *tree* ⟨עץ⟩

Directed graphs:

- A directed acyclic graph is abbreviated (and called) *DAG*

1. Graph theory — basic definitions
   - Trees
2. Graph representation
3. Eulerian cycles

Characterizing trees

Theorem

The following statements are equivalent for undirected graph $G = (V, E)$:

1. *G is a tree*
2. *Any two vertices in G are connected by a unique simple path*
3. *G is connected, but if any edge is removed, it disconnects*
4. *G is connected and $|E| = |V| - 1$*
5. *G is acyclic and $|E| = |V| - 1$*
6. *G is acyclic, but if any edge is added, a cycle is formed*

Proof: by showing that $(1) \Rightarrow (2) \Rightarrow (3) \Rightarrow (4) \Rightarrow (5) \Rightarrow (6) \Rightarrow (1)$

Tree characterization proof (1) $\Rightarrow$ (2)

- By (1): G is connected
  - $\Longrightarrow$ any two vertices in V admit ≥ 1 simple paths
- Left to show: inequality is not strict
- Assume by contradiction: G admits simple (u, v)-paths $p_1 \neq p_2$
- Let x be the vertex at which p_1 and p_2 first diverge
  - Exists because u appears in both p_1 and p_2 and $p_1 \neq p_2$
- Let y be the vertex following x at which p_1 and p_2 reconverge
  - Exists because v appears in both p_1 and p_2
- Let p_i' be the (x, y)-subpath of p_i for $i = 1, 2$
- p_1' and p_2' share no vertices other than their endpoints
- $\Longrightarrow$ path obtained by concatenating p_1' and reverse of p_2' is a cycle
- But G is a tree due to (1), hence cycle free $(\rightarrow\leftarrow)$

Tree characterization proof (2) $\Rightarrow$ (3)

- By (2): any two vertices in V are connected by (unique) simple path
  - $\Longrightarrow$ G is connected
- Left to show: removal of edge $e = (u, v) \in E$ disconnects G
- (u, v) is a path between u and v in G
  - $\Longrightarrow$ must be the unique simple (u, v)-path due to (2)
- Thus, if edge e is removed, then no simple (u, v)-path in G
  - $\Longrightarrow$ no (u, v)-path in G
- Therefore, removal of e disconnects G

Tree characterization proof (3) $\Rightarrow$ (4)

- By (3): G is connected
  - $\Longrightarrow |E| \geq |V| - 1$
- Prove that $|E| \leq |V| - 1$ by induction on $|V|$
- Base: connected graph with $n = 1$ or $n = 2$ vertices has $n - 1$ edges
- Let $|V| = n \geq 3$
- Assume: all graphs satisfying (3) with $< n$ vertices also satisfy $|E| \leq |V| - 1$
- Construct G' by removing an arbitrary edge $e \in E$ from G
  - Exists because $|E| \geq n - 1 \geq 2$
- G' has $k \geq 2$ connected components due to (3)
  - (Actually, $k = 2$)
- Each connected component $G_i = (V_i, E_i)$ of G' satisfies (3)
  - Otherwise, G does not satisfy (3)
- Inductive hypothesis: $|E_i| \leq |V_i| - 1$
- Thus, $|E| - 1 = \sum_{i=1}^{k} |E_i| \leq \sum_{i=1}^{k} (|V_i| - 1) = n - k$
  - $\Longrightarrow |E| \leq n - k + 1 \leq n - 1$

Tree characterization proof (4) $\Rightarrow$ (5)

- By (4): $|E| = |V| - 1$
- Left to show: G is acyclic
- Assume by contradiction: G contains a cycle $\langle v_0, v_1, \ldots, v_{k-1}, v_0 \rangle$
  - W.l.o.g.: cycle is simple
- Let $G_k = (V_k, E_k)$ be the subgraph of G consisting of the cycle
  - $|V_k| = |E_k| = k$
- If $k < |V|$, then $\exists v_{k+1} \in V - V_k$ adjacent to some $v_i \in V_k$
  - G is connected by (4)
- Let $G_{k+1} = (V_{k+1}, E_{k+1})$ be the graph obtained from G_k by adding vertex v_{k+1} and edge (v_i, v_{k+1})
  - $|V_{k+1}| = |E_{k+1}| = k + 1$
- If $k + 1 < |V|$, then...
- ...obtain graph $G_n = (V_n, E_n)$
  - $V_n = V$
  - $|V_n| = |E_n| = n$
- $E_n \subseteq E \Longrightarrow |E| \geq |E_n| = |V|$ $(\rightarrow\leftarrow)$

Tree characterization proof (5) $\Rightarrow$ (6)

- By (5): G is acyclic
- Left to show: if any edge is added to G, then a cycle is formed
- Let k be #connected components of G
- Each connected component is a tree
  - Connected and acyclic
- Already proved (1) $\Rightarrow$ (5), hence #edges in all connected components (together) is $|V| - k$
- By (5): $k = 1$, thus G is a tree
  - Connected and acyclic
- Already proved (1) $\Rightarrow$ (2), hence any two vertices in V are connected by a unique simple path
- $\Longrightarrow$ adding any edge to G closes a cycle

Tree characterization proof (6) $\Rightarrow$ (1)

- By (6): G is acyclic
- Left to show: G is connected
- Assume by contradiction: G has no path connecting vertices $u, v \in V$
- By (6): adding edge $e = (u, v)$ to E closes a cycle C
- Removing edge e from C yields (u, v)-path P
- By definition, P is included in G $_{(\rightarrow\leftarrow)}$ $\blacksquare$

Rooted trees

- A *rooted tree* ⟨עץ מושרש⟩ is a tree with one designated vertex
  - Called the *root* ⟨שורש⟩
- Rooted tree $T = (V, E)$ with root r
- Any vertex on the unique simple path from v to r is called an *ancestor* ⟨אב קדמון⟩ of v
  - v is always an ancestor of v
- If u is an ancestor of v, then v is a *descendant* ⟨צאצא⟩ of u
- The *subtree rooted at v* ⟨תת־עץ מושרש⟩ is the tree induced by the descendants of v rooted at v
- If u is the next vertex after v on the unique path from v to r, then u is the *parent* ⟨אב⟩ of v and v is a *child* ⟨בן⟩ of u
  - r is the only vertex in T with no parent
- A vertex with no children is a *leaf* ⟨עלה⟩
- A non-leaf is an *internal vertex* ⟨צומת פנימי⟩
- If two vertices share a parent, then they are *siblings* ⟨אחים⟩

Rooted trees — cont.

- The *degree* ⟨דרגה⟩ of v in T = #children it has
  - Depends on whether we treat T as a graph or a rooted tree
- The *depth* ⟨עומק⟩ of v is the length of the unique path from v to r
  - What is the depth of r?
- The *height* ⟨גובה⟩ of T is the the maximum depth of any of its leaves
- The *height* ⟨גובה⟩ of v is the height of the subtree rooted at v
- A *level* ⟨רמה⟩ of T consists of all vertices of the same depth
- The rooted tree is *ordered* ⟨מוסדר⟩ if the children of each internal vertex are ordered

1. Graph theory — basic definitions
   - Trees
2. Graph representation
3. Eulerian cycles

Representing graphs in a computer

- How do we represent (un)directed graph $G = (V, E)$ in a computer?
  - Prerequisite for graph algorithms
- Conventions:
  - $n = |V|$
    - Inside asymptotic notation, sometimes V
  - $m = |E|$
    - Inside asymptotic notation, sometimes E
  - In pseudocode, we write $G.V$ and $G.E$ (attributes of G)
- Convenient to assume that the vertices are numbered $1, \ldots, n$
  - Arbitrary order
- Sometimes, vertices/edges have additional attributes (should be stored as well)
  - Common: edge *weights* ⟨משקלים⟩ $w : E \to \mathbb{R}$
    - *Weighted graph* ⟨גרף ממושקל⟩

Adjacency matrix

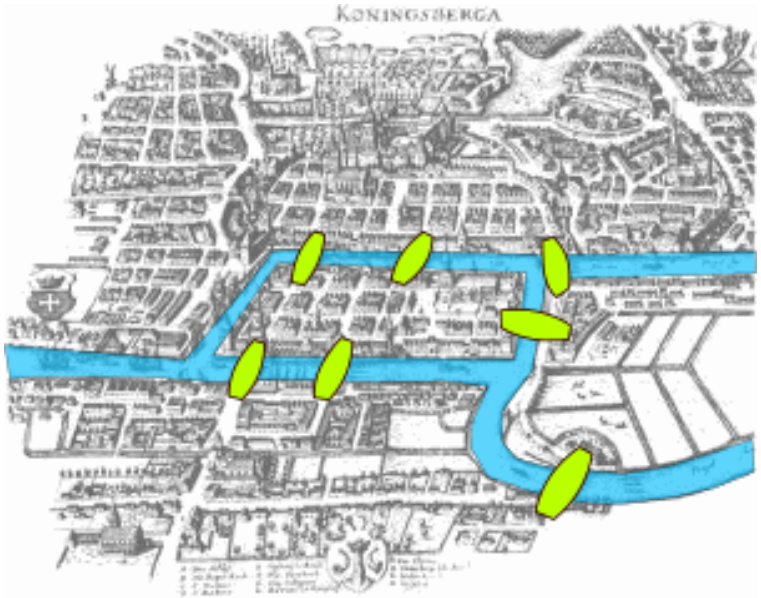
- Matrix $A \in \{0,1\}^{n \times n}$
  - $A(i,j) = \begin{cases} 1, & (i,j) \in E \\ 0, & \text{otherwise} \end{cases}$
  - Called *adjacency matrix* ⟨**מטריצת שכנויות**⟩
- Advantage: can answer the query *"does* $(i,j) \in E$*?"* in $\Theta(1)$ time
- If the graph is weighted, then $A \in \mathbb{R}^{n \times n}$
  - Depending on the context, $(i,j) \notin E$ represented by $A(i,j) = 0$, $A(i,j) = +\infty$, or $A(i,j) = -\infty$
- Entry $A(i,j)$ can include a pointer to an object with additional attributes of edge (i,j)
- How much memory does the adjacency matrix representation require?

Adjacency lists

- Array *Adj* of n lists
- For each $u \in V$, $Adj[u]$ is an *adjacency list* ⟨**רשימת שכנויות**⟩ that contains an entry for each vertex $v \in V$ such that $(u, v) \in E$
  - The entry of v includes all attributes of edge (u, v)
  - Typically implemented with a (doubly) linked list
- Directed graph: edge (u, v) represented only in $Adj[u]$
  Undirected graph: edge (u, v) represented in $Adj[u]$ and $Adj[v]$
  - In the directed case, can include a list for the entering edges too if necessary
- Advantage: can enumerate the edges incident on (resp., from) vertex v in $\Theta(\deg(v))$ (resp., $\Theta(\deg_{\text{out}}(v))$ time
  - How much time does it take in adjacency matrix representation?
- How much memory does the adjacency lists representation require?

1. Graph theory — basic definitions
   - Trees
2. Graph representation
3. Eulerian cycles

The seven bridges of Königsberg



- The city of Königsberg (Prusia) included 7 bridges
- The problem: device a walk through the city that would cross every bridge exactly once
- Resolved negatively by Leonhard Euler in 1736
  - Considered as one of the origins of modern graph theory

The Eulerian cycle problem

- Undirected graph $G = (V, E)$
- A cycle C in G that goes through every edge in E exactly once is called an *Eulerian cycle* ⟨מעגל אוילרי⟩
- The Eulerian cycle problem:
  - Input: undirected graph $G = (V, E)$
    Output: An Eulerian cycle in G or *"no Eulerian cycle"*
- An undirected graph that admits an Eulerian cycle is called an *Eulerian graph* ⟨גרף אוילרי⟩
  - If all degrees are positive, then G is connected

Characterizing Eulerian graphs

Theorem

A connected undirected graph is Eulerian iff the degree of every vertex is even.

- Connected undirected graph $G = (V, E)$
- For every cycle C in G and vertex $v \in V$,

$$\#\text{times } C \text{ enters } v \quad = \quad \#\text{times } C \text{ leaves } v$$

- If C is Eulerian, then we can classify (all) edges incident on v as entering or leaving w.r.t. C
  - $\#\text{entering edges} \quad = \quad \#\text{leaving edges}$
- Impossible if $\deg(v)$ is odd
- Eulerian cycle $\Longrightarrow$ all degrees are even

Even degrees suffice

- Connected undirected graph $G = (V, E)$, all degrees are even
- Construct an Eulerian cycle C as follows:
  1. Pick an arbitrary start vertex $v \in V$
  2. Follow an arbitrary path P from v, erasing the edges from G upon traversing them, until getting back to v
     - Cannot get stuck at any vertex $\neq v$ because of the even degrees!
     - P forms a cycle
     - All degrees remain even!
  3. Merge P into C
  4. If C contains a vertex that includes incident edges that have not been removed from G, then call this vertex v and goto line 2
     - Finish after all edges are erased from G because G is connected
- All degrees are even $\Longrightarrow$ Eulerian cycle ■
- The proof is *constructive*
- Not a proper pseudocode
  - Missing data structure details
  - Missing details in line 3