CS5070 Mathematical Structures for Computer Science - Notes 6

José Garrido

Department of Computer Science
College of Computing and Software Engineering
Kennesaw State University

jgarrido@kennesaw.edu

August 14, 2021

Graphs (Chapter 4)

- Graphs are constructed of a collection of dots known as **vertices** and lines known as **edges**
- Two vertices connected by an edge is said to be **adjacent** vertices
- A graph is an abstraction of a real situation
- A graph is an ordered pair $G = (V, E)$ of two sets :
  - V is the set of vertices
  - E is the set of edges. Set E is a set of 2-element subsets of V
- Example of a graph:
  $V = \{a, b, c, d, e\}$
  $E = \{\{a, b\}, \{a, c\}, \{a, d\}, \{a, e\}, \{b, c\}, \{d, e\}\}$
- See Example 4.1.2 page 235
- Two (or more) graphs are equal if the set of vertices and the set of edges are the same

Isomorphic Graphs

- Two or more graphs are **isomorphic** if the are not be equal but may be the same except for the names of the vertices
- An **isomorphism** between two graphs G_1 and G_2 is a bijection $f : V_1 \Longrightarrow V_2$ between vertices of the graphs such that $\{a, b\}$ is an edge in G_1 if and only if $\{f(a), f(b)\}$ is an edge in G_2
- Two graphs are **isomorphic** if there is an isomorphism between them, and denoted $G_1 \cong G_2$
- An isomorphism is a function that renames the vertices
- See Example 4.1.3 on page 236
- A collection of isomorphic graphs is often known as an **isomorphism class**

Subgraphs

- Graph $G' = (V', E')$ is a **subgraph** of $G = (V, E)$, denoted by $G' \subseteq G$, provided $V' \subseteq V$ and $E' \subseteq E$
- $G' = (V', E')$ is an **induced subgraph** of $G = (V, E)$ provided $V' \subseteq V$ and every edge in E whose vertices are still in V' is also an edge in E'
- Every induced subgraph is also an ordinary subgraph, but not conversely
- See Example 4.1.4

More Complex Graphs

- When including two more edges connecting a pair of vertices, the graph is known as a **multigraph**
- A graph is **connected** if we can get from any vertex to any other vertex by following some path of edges
- A graph is **complete** if it has an edge between every pair of vertices, all possible edges
- There is only one complete graph with a given number of vertices. Denoted by K_n the complete graph on n vertices

Complete Graphs

- Each vertex in K_n is adjacent to $n-1$ other vertices. The **degree** of the vertex is the number of edges emanating from the vertex
- There are $n(n-1)/2$ edges in K_n or $\binom{n}{2}$ edges
- The sum of the degrees of all vertices will always be *twice* the number of edges, since each edge adds to the degree of two vertices. This implies that the sum of the degrees of all vertices in any graph must be even
- The last bullet is the **Handshake Lemma** (see page 240)

$$\sum_{v \in V} d(v) = 2e$$

To find the number of edges in a graph, we need the **degree sequence** for the graph. This is a list of every degree of every vertex in the graph, generally in non-increasing order. See Example 4.1.6 on page 240

Odd Degree

- Proposition 4.1.8. In any graph, the number of vertices with odd degree must be even
- A graph is **bipartite** if the vertices can be divided into two sets, A and B, with no two vertices in A adjacent and no two vertices in B adjacent
- If each vertex in A is adjacent to all the vertices in B, then the graph is a **complete bipartite graph**, known as $K_{m,n}$, where $|A| = m$ and $|B| = n$

Named Graphs

- K_n is a complete graph on n vertices
- $K_{m,n}$ is a complete bipartite graph with sets of m and n vertices
- C_n is the cycle on n vertices, just one big loop
- P_n is the path on $n+1$ vertices (n edges), just one long path
- See graph examples on page 241

Graph Theory Definitions

- **Graph**. A collection of **vertices**, some of which are connected by edges. It consists a pair of sets V and E where V is a set of vertices and E is a set of 2-element subsets of V.
- **Adjacent**. Two vertices are adjacent if they are connected by an edge. Two edges are adjacent if they share a vertex.
- **Bipartite Graph**. A graph for which it is possible to divide the vertices into two disjoint sets such that there are no edges between any two vertices in the same set.
- **Complete Bipartite Graph**. A bipartite graph for which every vertex in the first set is adjacent to every vertex in the second set.
- **Complete Graph**. A graph in which every pair of vertices is adjacent
- **Connected Graph**. There is a path from any vertex to any other vertex

More Definitions

- **Chromatic Number**. The minimum number of colors required in a proper vertex coloring of the graph
- **Cycle**. A path that starts and stops at the same vertex, but contains no other repeated vertices
- **Degree of a Vertex**. The number of edges incident to a vertex
- **Euler Path**. A walk that uses each edge exactly once
- **Euler Circuit** An Euler path that starts and stops at the same vertex
- **Multigraph**. A graph that can contain multiple edges between two vertices as well as single edge loops (an edge to itself)
- **Path**. A walk that doesn't repeat any vertices (or edges) except perhaps the first and last. If a path starts and ends at the same vertex, it is known as a **cycle**
- **Planar**. A graph that can be drawn (in the plane) without any edges crossing

More Definitions 2

- **Subgraph**. H is a subgraph of G if every vertex and edge of H is also a vertex or edge of G.
- **Induced Subgraph**. H is an induced subgraph of G if every vertex of H is a vertex of G and each pair of vertices in H are adjacent in H if and only if they are adjacent in G
- **Tree**. A connected graph with no cycles. If we remove the requirement that the graph is connected, the graph becomes a **forest**. The vertices in a tree with degree 1 are known as **leaves**
- **Vertex Coloring**. An assignment of colors to each of the vertices of a graph. A vertex coloring is **proper** if adjacent vertices are always colored differently
- **Walk**. A sequence of vertices such that consecutive vertices are adjacent. A walk in which no edge is repeated is known as a bf trail, and a trail in which no vertex is repeated is known as a **path**

Properties of Trees

- A **tree** is a connected graph containing no cycles. A **forest** is a graph containing no cycles
- **Proposition 4.2.1**. A graph T is a tree if and only if between every pair of distinct vertices of T there is a unique path
- **Corollary 4.2.2**. A graph F is a forest if and only if between any pair of vertices in F there is at most on path
- **Proposition 4.2.3**. Any tree with at least two vertices has at least two vertices of degree one
- **Proposition 4.2.4**. Let T be a tree with v vertices and e edges. Then $e = v - 1$

Rooted Trees

- One vertex of the tree is designated as the **root**, and every other vertex on the tree can be characterized by its position relative to the root
- From any vertex, we can travel back to the root in exactly one way. This also allows us to describe how distinct vertices in a rooted tree are related
- If two vertices are adjacent, then one of them is the **parent** of the other, which is known as the **child** of the parent
- The parent is the vertex closer to the root. The root of a tree is a parent, but is not the child of any vertex. All non-root vertices have *exactly one* parent

More on Rooted Trees

- A vertex v is a **descendant** of a vertex u provided u is a vertex on the path from v to the root
- In this case, vertex u is an **ancestor** of vertex v
- When there are pairs of vertices neither of which is a **descendant** of the other, they are known as **siblings**
- Vertices u and v are siblings provided they have the same parent. Note that these vertices are never adjacent
- See Example 4.2.5

Navigating a Tree

- To navigate through a tree is traversing a tree, starting at the root and visiting each vertex in some order
- When all visiting vertices in the same generation before any vertices of the next generation is known as **breath first search**
- **Depth first search** is visiting one child of the root, then visit a child of that vertex, then visit its child. When you get to a vertex with no children, retreat to its parent and see if the parent has any other children.
- The parent is the vertex closer to the root. The root of a tree is a parent, but is not the child of any vertex. All non-root vertices have *exactly one* parent

Spanning Trees

- If a connected graph is not a tree, then we can still use the traversal algorithms if we identify a subgraph that is a tree
- We can take a single edge of connected graph G, we need to visit all vertices, such a tree is known as a **spanning tree**. Every connected graph has a spanning tree
- **Spanning Tree**. Given a connected graph G, a spanning tree of G is a subgraph of G hat is a tree and includes all vertices of G

Spanning Trees Algorithm

- We can define an an algorithm for *finding* a spanning tree.
  - Start with a connected graph G
  - If there is no cycle,then G is already a spenning tree
  - If there is a cycle, let e be any edge in that cycle and consider the new graph $G_1 = G - e$. This graph is still connected since e belonged to a cycle
  - Repeat: if G_1 has no cycles, we are done.
  - If there is a cycle, define $G_2 = G_1 - e_1$, where e_1 is an edge in G_1
  - Continue until a tree is found.
- See Example 4.2.7
- In practice, these type of algorithms are applied to *weighted* graphs, which have values associated with each edge.
- If the goal is to find a spanning tree with the smallest possible combined weight, the tree is a **minimum spanning tree**

Planar Graphs

- When a connected graph can be drawn without any edges crossing, the graph is known as **planar**
- This planar graph divides the plane into regions known as **faces**
- For any (connected) planar graph with v vertices, e edges, and f faces:

$$v - e + f = 2$$

- Not all graphs are planar. If there are too many edges and too feww vertices, then some edges will need to intersect
- **Theorem 4.3.1**. K_5 is not planar. See proof in page 260
- **Theorem 4.3.2**. $K_{3,3}$ is not planar
- In general, if g is the size of the smallest cycle in a graph, then for any planar graph $g \cdot f \leq 2e$

Polyhedra

- A **polyhedron** is a geometric solid made up of flat polygonal faces joined at edges and vertices
- A **convex** polyhedron is a geometric solid in which any line segment connecting two points on the interior of the polyhedron must be entirely contained inside the polyhedron
- The cube is a **regular polyhedron** (also known as a **Platonic solid**) because each face is an identical regular polygon and each vertex joins an equal number of faces
- There are exactly four other regular polyhedra: the tetrahedron, octahedron, dodecahedron, and icosahedron with 4, 8, 12, and 20 faces respectively
- Every convex polyhedron can be projected onto the plane without edges crossing
- Since every polyhedron can be represented as a planar graph, Euler's formula for planar graphs holds for all convex polyhedra

Coloring

- If we place a vertex in the center of each region (of a map) and then connect two vertices if their states share a border, we get a graph
- Coloring regions on the map corresponds to coloring the vertices of the graph
- The graph cannot have vertices colored the same when these vertices are adjacent
- For a graph G, a coloring of the vertices is called a **vertex coloring**
- If the vertex coloring has the property that adjacent vertices are colored differently, then the coloring is known as **proper**
- The smallest number of colors needed to get a proper vertex coloring is called the **chromatic number** of the graph, $\chi(G)$
- See Example 4.4.1
- K_n has chromatic number n

More on Coloring

- **Theorem 4.4.2 The Four Color Theorem**. If G is a planar graph, then the chromatic number of G is less than or equal to 4. Thus any map can be properly colored with 4 or fewer colors
- Coloring regions on the map corresponds to coloring the vertices of the graph
- There are many applications of graph coloring. See Example 4.4.3 in page 270
- A **clique** in a graph is a set of vertices all of which are pairwise adjacent. In other words, a clique of size n is just a copy of the complete graph K_n
- The **clique number** of a graph is the largest n for which the graph contains a clique of size n

More on Coloring

- Any clique of size n cannot be colored with fewer than n colors, so this is a lower bound
- **Theorem 4.4.4** The chromatic number of a graph G is at least the clique number of G
- When the chromatic number of G is equal to the clique number, the graph is known as **perfect**
- Let $\Delta(G)$ be the largest degree of any vertex in graph G.
- **Theorem 4.4.5 Brooks' Theorem**. Any graph G satisfies $\chi(G) \leq \Delta(G)$, unless G is a complete graph or an odd cycle, in which case $\chi(G) = \Delta(G) + 1$

Coloring Edges

- Two edges are adjacent if they are incident to the same vertex
- The least number of colors required to properly color the edges of a graph G is called the **chromatic index** of G and written $\chi'(G)$
- See Example 4.4.6
- **Theorem 4.4.7 Vizing's Theorem**. For any graph G, the chromatic index $\chi'(G)$ is either $\Delta(G)$ or $\Delta(G)+1$

Euler Paths and Circuits

- A **walk** in a graph is a sequence of vertices such that every vertex in the sequence is adjacent to the vertices before and after it in the sequence
- An **Euler path**, in a graph or mutigraph, is a walk through the graph that uses every edge exactly once
- An **Euler circuit** is an Euler path that starts and stops at the same vertex
- One way to guarantee that a graph does not have an Euler circuit is to include a *spike*, which is a vertex of degree 1
- A graph has an Euler circuit if and only if the degree of every vertex is even
- A graph has an Euler path if and only if there are at most two vertices with odd degree

Hamilton Paths

- If there is a path that visits every vertex exactly once, the path is known as a **Hamilton path**
- A **Hamilton cycle** is a Hamilton path that starts and stops at the same vertex
- Given a bipartite graph, a **matching** is a subset of the edges for which every vertex belongs to exactly one of the edges
- **Matching Condition**. If a bipartite graph $G = \{A, B\}$ with two sets of vertices A and B, has a matching of A, then
  $$|N(S)| \geq |S|, \quad \text{for all } S \subseteq A$$
- Define $N(S)$ to be the set of all the **neighbors** of vertices in S. That is $N(S)$ contains all vertices (in B) that are adjacent to at least one of the vertices in S