Notes on Graph Theory

Aidan Backus

May 1, 2017

These are my notes on graph theory, based on CS61B, Data Structures, taught by Josh Hug, and Math 55, Discrete Math, taught by Vera Serganova.

1 Definitions

A graph is a pair of sets V of **vertices** and E of **edges**, which are ordered pairs in V^2. An edge of the form (a, b) is called an edge from a to b.

The **degree** of a vertex is the number of edges connected to that vertex.

Theorem 1.1 (Handshaking Lemma)**.** *The sum of all degrees of vertices in a graph is even.*

Proof. Each edge contributes twice to the sum: once for its start point and the other for its end. So the sum always increases by 2 for each edge. □

A **loop** is an edge from a vertex v to itself, i.e. it is of the form (v, v); this counts twice in the counting of degree. A graph is called **symmetric** or **undirected** if for all edges (a, b) there exists an edge (b, a). A graph has **multiedges** if there are multiple edges from a vertex a to b. If a graph has no loops, is symmetric and has no multiedges, the graph is called a **simple graph**.

A **component** of a simple graph is an equivalence class of the vertices, such that the vertices are only connected to each other.

Graphs which are not undirected are called **directed** or **digraphs**.

A **walk** is a means of getting from one vertex to the next by taking various edges. A walk which contains each vertex only once is called a **path** and a walk which begins and ends on the same vertex is called a **cycle**.

If there exists a path between any two vertices in the graph, the graph is called **connected**.

A bijection $\phi : V_{G_1} \to V_{G_2}$ is called an **isomorphism** if each vertex in the domain has edges mapping to the corresponding vertices as in the codomain. If one such ϕ exists then G_1 and G_2 are called **isomorphic**.

Two vertices are **adjacent** if there is an edge from one to the other.

If the vertices are labeled by $\{1, 2, \ldots, n\}$, the **adjacency matrix** of a finite graph is the matrix in $\mathbb{R}^{n \times n}$ whose ijth entry is the number of edges from i to j.

Theorem 1.2. *The adjacency matrix of an undirected graph is symmetric. Moreover, its eigenvalues are all real and nonnegative.*

Proof. Let A be one such adjacency matrix. If there is an entry in A_{ij} then there must be the same entry in A_{ji} because the graph is symmetric. By the spectral theorem, its eigenvalues are real. Its eigenvalues cannot be negative because there cannot be negative entries in this matrix (since there cannot be a negative number of edges between two vertices). □

A vertex is called **incident** to an edge if the edge starts or ends on the vertex.

If the vertices are labeled by $\{1, 2, \ldots, n\}$, edges labeled by $\{1, 2, \ldots, m\}$, the **incidence matrix** of a finite graph is the matrix in $\mathbb{R}^{n\times m}$ such that if M is the adjacency matrix, then M_{ij} is one if i is incident to j or zero otherwise.

The **null graph** of order n, written N_n, is the graph (unique up to isomorphism) with n vertices and 0 edges.

The **complete graph** of order n, written K_n, is the graph (unique up to isomorphism) with n vertices and edges connecting all vertices.

The **cycle graph** of order n, written C_n, is the graph with n vertices, and a edge connecting each vertex in exactly one cycle.

The **wheel graph** of order n, written W_n, is the cycle graph of order $n-1$, with another vertex which every other vertex is adjacent to.

The **path graph** of order n, written P_n, is the cycle graph of order $n+1$ but a vertex removed.

A **regular graph** of order n is one such that every vertex is of degree n.

A **bipartite graph** is one where there exists a partition of V into two equivalence classes, such that if A is an equivalence class and $a \in A$, then a is not adjacent to any $b \in A$.

2 Paths

Theorem 2.1. *If G is a bipartite graph, then each cycle in G has even length.*

Proof. Suppose that G had an odd-length cycle, $v_0, v_1, \ldots, v_n, v_0$, n even, and let A and B be the equivalence classes of V. Without loss of generality assume $v_0 \in A$. Then $v_1 \in B$, $v_2 \in A$, and so on, so that $v_i \in A$ if and only if i is even. But n is even, so $v_n \in A$, which implies $v_0 \in A$, a contradiction. □

Theorem 2.2. *If G is a simple graph with n vertices, m edges, and k components, then*

$$n - k \leq m \leq \frac{(n-k)(n-k+1)}{2}.$$

Proof. Prove $m \geq n - k$ by induction on m. If G is null, then $m = 0$, but $n = k$, so that this is true. Otherwise, suppose that G contains k components and m_0 edges, and that m_0 is chosen so that within a component, each vertex has precisely one path to another vertex. Then if we remove an edge, we end up with a graph with $k+1$ components and $m_0 - 1$ edges. The induction hypothesis implies that $m_0 - 1 \geq n - (k+1)$. So this bound holds.

To prove $m \leq \frac{(n-k)(n-k+1)}{2}$, suppose that $C_1, C_2 \subseteq G$ are complete components of G with v_1, v_2 edges respectively, such that $v_1 \geq v_2 \geq 1$. Treat C_1, C_2 as subgraphs, and add a vertex

to C_1, removing it from C_2, and complete this new C_1'. Then the total number of edges in C_1' and C_2 increases by $v_1 - v_2 + 1 \geq 0$ so that the inequality is an equality precisely when a graph is complete. □

Suppose G is bipartite with equivalence classes V_1 and V_2. A **complete matching** from V_1 to V_2 is a bijection $f : V_1 \to X \subseteq V_2$ such that $f(v_1) = v_2$ if and only if there exists an edge from v_1 to v_2.

Theorem 2.3 (Hall's marriage theorem)**.** *Suppose G is bipartite with equivalence classes V_1 and V_2, and if $A \subseteq V_1$ then $\phi(A) \subseteq V_2$ is the set of vertices adjacent to vertices in A. Then there exists a complete matching from V_1 to V_2 if and only if $|A| \leq |\phi(A)|$ for all $A \in 2^{V_1}$.*

Proof. Suppose that there is a complete matching but $|A| > |\phi(A)|$. Then the bijection f is not onto, a contradiction.

Now suppose that $m = |V_1|$.

Consider a set $A \subseteq V_1$ where $|A| < m$. If $|\phi(A)| \geq |A|$ then we can map these vertices to their counterparts, and recursively reduce the problem to one of the form $V_1 \setminus A \to V_2 \setminus \phi(A)$. The hypothesis still holds because for each a we are removing from a set $B \subseteq V_1$, we are removing a $f(a)$ from $\phi(B) \subseteq V_2$.

By induction we reduce the problem to one where $m = 1$. In this case the only nontrivial subset of V_1 is itself. Clearly if $m \leq |\phi(V_1)|$ then we can simply map V_1 to a member of $\phi(V_1)$ and be done. □

Suppose G is a connected graph.

A **cutset** of G is a minimal set of edges such that removing a single edge splits G into components. The **edge connectivity** $\lambda(G)$ is the cardinal number of the smallest cutset of G.

A **separating set** of G is a minimal set of vertices such that removing a single vertex splits G into components. A separating set of cardinal number 1 is called a **cut vertex**. The **vertex connectivity** $\kappa(G)$ is the cardinal number of the smallest separating set of G.

A **Eulerian cycle** is one that contains every edge of G precisely once.

Theorem 2.4 (Seven Bridges of Konigsberg)**.** *G contains an Eulerian cycle if and only if the degree of every vertex in G is even.*

Lemma 2.4.1. *Suppose H is a graph and every vertex of H has degree at least 2. Then H contains a cycle.*

Proof of the lemma. If H is not simple, then it contains a loop or multiedge, and so it contains a cycle.

Otherwise, H is simple, and contains a walk through the vertices v_j, where v_0 is any vertex in H, and v_i is any vertex adjacent to v_{i-1} except v_{i-2}, which necessarily exists because the degree of v_{i-1} is at least 2. But H contains finitely many vertices, so eventually we will find a cycle. □

Proof of the Seven Bridges of Konigsberg. Suppose G has an Eulerian cycle P. All vertices appear in P, because P contains every edge and G is connected. Each time P passes through

a vertex, its degree increases by 2: once for the way in, once for the way out. So the vertex has an even degree.

Now suppose that G may not have a Eulerian cycle but does have vertices of even degree. G is connected, so that the degree of each vertex is at least 2. So G contains a cycle C_0. If C_0 is not Eulerian, remove the edges of C_0 from G to create a graph G_0, possibly disconnected. Because C passes through every vertex twice, removing C_0 from G will create a graph whose vertices still have even degree.

Suppose then, recursively, that each component of G_i has an Eulerian cycle. (Here we are defining $G = G_0$.) Then G_{i-1} must have an Eulerian cycle, which will go through G_i and C_{i-1}: follow C_{i-1} until we find a non-isolated vertex of G_{i-1}. Then traverse the Eulerian cycle of G_{i-1}, before returning of C_{i-1}. Since C_{i-1} is a cycle, we will eventually reach the original point of C_{i-1}. This path is an Eulerian cycle in the component of G_i.

If, at any point, we find a null graph G_n, then each component contains precisely one vertex of G_n, which of course has a trivial Eulerian cycle. □

Corollary 2.4.1. *G has an Eulerian cycle if and only if its edges can be partitioned into disjoint cycles.*

Proof. If G has an Eulerian cycle, then it has a cycle containing all of its edges, so we can impose a trivial partition on it.

Now suppose G may not have a Eulerian cycle but whose edges are partitioned into equivalence classes of disjoint cycles. Then every vertex must be part of a cycle, because G is connected, so G meets the hypothesis of the Seven Bridges of Konigsberg. □

Theorem 2.5 (A* search algorithm)**.** *Suppose G is a weighted, directed graph with vertices a and b. If we have a heuristic $h : V \to [0, \infty)$, estimating the distance from any vertex v to b, we can compute the shortest path from a to b as follows:*

1. *Initialize an empty set S, the so-called closed set.*
2. *Initialize an empty function $f : V \to V$, the backtracking function.*
3. *Initialize an empty function $g : V \to [0, \infty)$, the backtracking cost.*
4. *Initialize an empty function $j : V \to [0, \infty)$, the estimated cost to the end goal through that function.*
5. *Initialize an empty priority queue PQ, the so-called open set, ordered by j.*
6. *Set $f(a) = a$, $g(a) = 0$, $j(a) = h(a)$, and insert a into PQ.*
7. *While PQ is not empty:*
   (a) *Pop v from PQ.*
   (b) *If $v = b$, then:*
      i. *Create a queue Q, the back-trace.*
      ii. *While $v \neq a$:*

A. *Push v onto Q.*
B. *Set $v = f(v)$.*

iii. *Return Q.*

(c) *Insert v into S.*

(d) *For each vertex w adjacent to v and not in S, with distance $d(v, w)$:*

i. *Define $t = g(w) + d(v, w)$.*
ii. *If w is not in PQ, push it onto PQ.*
iii. *Otherwise:*
A. *Set $g(w) = min(t, g(w))$.*
B. *Sink/swim w through PQ.*
C. *Set $j(w) = g(w) + h(w)$.*
D. *If w sunk or swam, set $f(w) = v$.*

8. *Conclude that such path exists.*

Proof. The neighbor loop will compute which paths have the shortest distance, and the priority queue will choose the shortest one. □

If we want a path to every vertex rather than just b, then we can simply return f and define $h(v) = 0$ for all h. This special case of A* is known as **Dijsktra's algorithm**.

3 Trees

A graph without cycles is called a **forest**; each component of a forest is called a **tree**.

Theorem 3.1. *Suppose T is a graph with n vertices, then each of the following statements is equivalent:*

1. *T is a tree.*
2. *T is a forest with $n - 1$ edges.*
3. *T is connected and has $n - 1$ edges.*
4. *T is connected and each edge is cut.*
5. *There exists exactly one path between any two vertices of T.*
6. *T contains no cycles but adding an edge creates a cycle.*

Proof. Suppose T is a tree. Then T is a forest, and a forest must have fewer than n edges, otherwise there would be a cycle. But T must have maximally many edges because it is connected. So T has $n - 1$ edges.

Suppose T is a forest with $n - 1$ edges. Then T has maximally many edges without adding any cycles, so it is connected.

Suppose T is connected with $n-1$ edges. Then removing an edge will disconnect the graph because it's impossible to have a connected graph with $n-2$ edges. So each edge is cut.

Suppose T is connected with entirely cut edges, so that there exists a path between any two points in T. But there can only be one such path, otherwise one could remove an edge in the second path without disconnecting T, and that edge would not be cut.

Suppose there exists exactly one path between any two points in T. Then T cannot contain a cycle, because that would provide a second path; but adding an edge would link to a vertex that there already exists a path to from the current vertex, so that there would exist a cycle.

Suppose that T contains no cycles, but adding an edge would create one. Then T is a forest, and it is impossible to add an edge and maintain this property. The only way one could add an edge to a forest is if two of its vertices did not have a path between them. So T is connected, therefore a tree. □

Suppose G is a connected graph. Then we can remove edges from its cycles until a tree called a **spanning tree** is left over. The minimal number of edges removed to create a spanning tree is called the **cycle rank** $\gamma(G)$. If the edges are weighted, a spanning tree such that the sum of the edges is minimized is a **minimum spanning tree**.

Theorem 3.2 (Prim's algorithm)**.** *If G is a weighted, connected, undirected graph, we can compute a minimum spanning tree as follows:*

1. *Initialize vertices in G with a cost ∞.*
2. *Initialize an empty tree T and priority queue PQ.*
3. *Insert all vertices into PQ.*
4. *While not all vertices of G are also vertices of T:*
   (a) *Pop a vertex V_i from PQ.*
   (b) *Add V_i to T, and if there exists an associated edge E_i, add E_i to T.*
   (c) *For each edge E_{ij} adjacent to V_i, if V_j is in PQ and the weight of E_i is less than the cost of V_j:*
      i. *Set the cost of V_j to the weight of E_i.*
      ii. *Set the edge E_j associated with V_j to E_{ij}.*
5. *Return T.*

Proof. Prim's algorithm generates a tree T because precisely one edge is mapped to each vertex. T is spanning because G is connected and all edges are listed in the priority queue; it is minimum because Prim's algorithm is a special case of Dijkstra's algorithm. □

To traverse a tree, we start at a point V_0 and then can use **breadth first search**, which considers all points adjacent to V_i before considering points adjacent to those vertices; or **depth first search**, which considers the points adjacent to V_i before the "siblings" of V_i.