Basic Graph Terms

GRAPH:

› A simple graph G consists of **a non-empty finite set V of elements called vertices/nodes** and a **finite set E of distinct unordered pairs of distinct elements of V called edges**. V and E are called the **vertex set and edge set respectively**. G is written as a two-tuple **G=(V,E)**

› If the edges don't have a direction (indicated with an arrow), graph G is an **undirected graph**. Else, it is a **directed graph or digraph**

› If G has multiple edges between a pair of nodes, it is called a **multigraph if it is undirected or a multidigraph if it is directed**

DEFINITIONS FOR UNDIRECTED GRAPHS:

› In an undirected graph, a **proper edge joins one vertex to another** and a **self loop joins a vertex to itself**.

› **|V| is called the order of the graph** and **|E| is called the size of the graph**. G is **finite if both order and size are finite**, else it is **infinite**

› Edge e ∈ E (represented by vertex pair (u,v) where u,v ∈ V) in an undirected graph can also be represented by (v,u).

› Vertices u and v are **adjacent to each other if there's an edge e = (u,v) ∈ E**. e is then **incident upon u and v**, and u and v are also called **neighbors of each other**

› Degree of vertex v in an undirected graph is given by **deg(v) = # of proper edges + 2 * # of self-loops**

› A **multiedge is a set of multiple edges between the same vertex pair**.

› A **loopless graph** is a **graph without self-loops**. A graph **without self-loops and multiedges** is called a **simple graph**

DEFINITIONS FOR DIRECTED GRAPHS OR DIGRAPHS:

Basic Graph Terms (cont)

› A directed edge is called an **arc**

› If an arc **starts at vertex u and ends at vertex v** , then **u and v are called the head and tail of the arc respectively**. Vertex v is the **successor of vertex u**, and vertex u is the **predecessor of vertex v.**

› The **set of arcs which end at vertex v** are called **in-edges or in-arcs**. Similarly, the **set of arcs which start from vertex v** are called **out-edges or out-arcs**.

› The **out-degree of a vertex** is **the number of edges starting from it**. Similarly, the **in-degree of a vertex** is the **number of edges ending at it**. Each **self-loop at vertex v** contributes **a count of 1 to both the in-degree and out-degree**. The **degree of a vertex in a digraph** is given by **deg(v) = in-degree of v + out-degree of v**

› A directed graph is **strongly connected if there's a path from a to b, ∀ a, b ∈ V**

› A directed graph is **weakly connected if there's a path between every two vertices in the underlying undirected graph**

Planar and Bipartite Graphs

› Graph G is **planar** if it **can be drawn in the plane with edges intersecting ONLY at vertices of G**.

› Such a drawing is called **embedding of G in the plane**.

› Graph G is **bipartite if V = V_1 U V_2 with V' = V_1 . V_2 = null set and every edge of G takes the form (a,b) with a ∈ V_1 and b ∈ V_2**

› If **each vertex in V_1 is joined with every vertex in V_2, then we get a complete bipartite graph, denoted by $K_{m,n}$ where $|V_1|$ = m and $|V_2|$=n**

Planar and Bipartite Graphs (cont)

› If **G=(V,E) is a loop-free, undirected graph and E != null set**, an **elementary subdivision of G results when edge e = (u,w) is removed from G and then edges (u,v) and (v,w) are added to G - e, where v ∉ V**

› **Loop free, undirected graphs G_1 = (V_1, E_1) and G_2 = (V_2, E_2)** are called **homeomorphic if they are isomorphic OR they both can be obtained from the same loop free, undirected graph H by a sequence of elementary subdivisions.**

› **Kuratowski's Theorem: Graph G is nonplanar iff it contains a subgraph homeomorphic to either K_5 or $K_{3,3}$**

› **Euler's theorem: Let G=(V,E) be a connected planar graph or multigraph, with |V| = v and |E| = e. If r is the # of regions in the plane determined by a planar embedding/depiction of G, and one of these regions (called the infinite region) has infinite area, then v - e + r =2**

› **For each region R in a planar embedding of a (planar) graph or multigraph, the degree of R, denoted deg (R), is the # of edges traversed in a (shortest) closed walk about (the edges in) the boundary of R.**

› **Corollary: Let G = (V;E) be a connected planar graph or multigraph with |V| = v and |E| = e > 2, and r regions. Then 3r <= 2e and e <=(3v - 6)**

Subgraphs, Complements and Graph Isomorphism

› The **subgraph G_S of graph G=(V,E) is given by G_S = (V_S, E_S) where ∅ (null set) != V_S , and V_S and E_S are subsets of V and E respectively**

› NOTE: **if (u,v) ∈ E_S, then u,v ∈ V_S**

› A **spanning subgraph of G, where G can be directed or undirected** is a **subgraph composed of all vertices of G**, ie: **V_S = V**

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com

Subgraphs, Complements and Graph Isomorphism (cont)

› If G = (V,E) is a directed/undirected graph, and if U is not a null set and is a subset of V, then the induced subgraph on U is a graph whose vertex set is U, and whose edge set contains every edge in E having endpoints in U.

› Deleting an edge e in graph G (which can be directed/undirected) creates a subgraph denoted by G - e , which contains all vertices of G and all edges except e.

› Deleting a vertex v in graph G(which can be directed/undirected) creates a subgraph denoted by G - v, which contains all vertices of G except for v, and all edges in G except those incident on vertex v

› If V is a set of n vertices, then the complete graph on V, denoted by K_n, is a loop-free, undirected graph where ∀ a, b ∈ V AND a != b, there exists edge (a,b)

› If G is a loopless, undirected graph with n vertices, then the complement of G (denoted by G with a bar on top) is the subgraph of K_n containing all vertices in G and all edges not in G

› If $G = K_n$, then its complement has all the n vertices and no edges. Such a graph is called null graph.

› For two undirected graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, a function $f: V_1 \to V_2$ is called graph isomorphism if a) f is one-to-one and onto and b) ∀ a,b ∈ V_1, (a,b) ∈ E_1 iff (f(a), f(b)) ∈ E_2. The two graphs are called isomorphic graphs if such a function exists.

Walks, Paths and Circuits

› A walk in graph G is a sequence of nodes $v_1, v_2, \dots, v_n$ where n>=2 and (v_i, v_{i+1}) ∈ E for i = 1, 2, ... , n-1.

› Repetition of vertices and edges in a walk is permitted

› If G isn't weighted, then the length of the walk is n-1. Else, the length is the sum of the weights of all edges in the walk (provided the weights are non-negative)

› A trail is a type of walk in which all edges are different. Repetition of vertices is permitted while repetition of edges are not.

› A path is a trail in which all the vertices are different (except the starting and ending vertices may be the same). There is no repetition of edges and vertices.

› If the starting vertex is the same as the ending vertex, the walk, trail or path is open, else it is closed.

› G is connected if each vertex pair is joined by a path, else it is disconnected

› Vertex v is reachable from vertex u if there's a path connecting the two.

› A circuit or cycle is a walk with the same starting and ending vertices, with no repeating vertices in between.

› Hamiltonian Path: a path containing all of G's vertices

› Hamiltonian Cycle: a cycle containing all of G's vertices

THEOREMS FOR HAMILTON CYCLES AND PATHS:

› Theorem: Let G = (V,E) be a loop-free undirected graph, where |V| = n >= 2. If deg (x) + deg (y) >= n - 1 for all x, y ∈ V , x != y, then G has a Hamilton path.

Walks, Paths and Circuits (cont)

› Corollary: Let G = (V,E) be a loop-free undirected graph, where |V| = n >= 2. If deg (v) >= (n - 1)/2 for all v ∈ V , then G has a Hamilton path.

› Ore's theorem: Let G = (V,E) be a loop-free undirected graph, where |V| = n >= 3. If deg (x) + deg (y) >= n for all nonadjacent x, y ∈ V , then G contains a Hamilton cycle.

› Corollary, ie: Dirac's theorem: Let G = (V,E) be a loop-free undirected graph, where |V|= n >= 3. If deg (v) >= n/2 for all v ∈ V , then G contains a Hamilton cycle.

› Theorem: Every simple graph G = (V,E) with |V| = n >= 3 and at least $(n^2 - 3n + 6)/2$ edges has a Hamilton cycle.

Graph Representation

› Adjacency Matrix: It is an n x n matrix where n = |V|

› Entry $A_{i,j}$ in adjacency matrix A for an undirected graph G, where 1<=i,j<=n is equal to the # of edges between vertices v_i and v_j if i=j OR 2 * # of self loops if i != j

› The row-sum or column-sum in adjacency matrix A is equal to the degree of the corresponding vertex

› If G is directed, then $A_{i,j}$ = # of arcs from v_i to v_j

› The row sum in adjacency matrix A for a digraph is equal to out-degree of the corresponding vertex and the column sum is equal to the in-degree

Euler Trails and Circuits

› Theorem: Let G = (V,E) be an undirected graph or multigraph. Then Sum over v∈V (deg(v)) = 2 * |E|

› Corollary: For any undirected graph or multigraph, the # of vertices with odd degree must be even.

Sponsored by ApolloPad.com
Everyone has a novel in them. Finish Yours!
https://apollopad.com

Euler Trails and Circuits (cont)

› **If G = (V,E) is an undirected graph/multigraph, with at least 2 vertices and no vertices are isolated**, then:

● **G has a Euler circuit if there's a circuit in G traversing every edge of the graph exactly once**

● **If there's an open trail from a to b in G, and the trail traverses each edge in G exactly once, then the trail is called a Euler trail.**

› **Theorem:** Let **G = (V,E) be an undirected graph/multigraph with at least 2 vertices and no vertices are isolated.** Then **G has an Euler circuit iff every vertex in G has an even degree**.

› **Corollary:** If **G=(V,E) is an undirected graph/multigraph with at least 2 vertices and no isolated vertices, we can construct an Euler trail iff G is connected and has EXACTLY 2 vertices of odd degree**.

› **Theorem:** Let **G = (V,E) be an directed graph/multidigraph with at least 2 vertices and no vertices are isolated.** Then **G has a directed Euler circuit iff out-degree of v = in-degree of v, ∀ v ∈ V**.

› An **undirected graph/multigraph where each vertex has the same degree is called a regular graph.**. A **k-regular graph is an undirected graph/multigraph in which all vertices have the same degree k**.

Trees

› **Tree: a connected graph without any cycle**

› **TREE OBSERVATIONS:**

● If the **removal of an edge from connected graph G** makes it **disconnected**, then **G is a tree**

● There's **a unique path between every vertex pair in a tree**

● **If |V|>2 adding a new edge creates a cycle in a tree**

● **In a tree, |E| = |V| - 1**

● **Trees are bipartite**

Trees (cont)

● **Every tree can be colored using 2 colors**

› An ensemble of trees is called a **forest**

› A tree T=(V',E') is a **spanning tree of graph G=(V,E) if V'=V and E' is a subset of E**

› If |V|=n, then there are $n^{(n-2)}$ **spanning trees on V for complete graph** K_n.

› **If** $K_{m,n}$ **is a complete bipartite graph, then the total # of spanning trees is** $m^{(n-1)} * n^{(m-1)}$.

› **TYPES OF TREES:**

● **Path graph/linear graph:** Has n vertices in a line, where vertices i and i+1 are connected by an edge, where i = 1,2,...,n

● **Star tree:** Has n>=2 vertices (where just one vertex is the internal vertex), and n-1 leaves.

› **ROOTED TREE:**

● Has one node specified as the **root node**

● All other nodes in the tree are the **descendants of the root node**

● Each edge of this tree divides the set of vertices which are attached to either side of this edge into two groups. The **set of nodes away from the root node are called the descendant set of nodes** with respect to this edge. Similarly, the **set of nodes towards the root node are called the ancestral set of nodes with respect to this edge**.

● **Child of vertex v** is a vertex **whose immediate ancestor is v**. A **vertex with children is called an internal vertex**.

● **Depth of a vertex**: # of edges in the unique path from the root to the vertex.

● **Ordered tree**: Rooted tree where **children or branches of every internal vertex are linearly ordered**. ie: if the vertex has k children, then there is a 1st child, 2nd child and so on, all the way up to the kth child.

Trees (cont)

● **Binary tree**: An ordered, rooted tree where **each internal vertex has at most 2 children**

● **m-ary tree:** An ordered, rooted tree where **each internal vertex has at most m children, with m = 2,3,...**

● **Subtree:** A tree **rooted at a specific node which contains all its descendant nodes and the corresponding connecting edges.**

● **Height of a rooted tree: Maximum # of edges in the unique path from root to any vertex.**

● **Balanced tree:** It's an **m-ary tree of height b** where **all leaves have height b or b-1, where b=2,3,...**

● **d-regular tree: Each internal vertex has exactly d children**. In a **complete d-regular tree, all leaves have the same depth**

› **MATRIX TREE THEOREM:**

● Let A be the adjacency matrix of graph G.

● Obtain matrix D, where the diagonal elements are the sums of the columns of A and the rest of the elements are 0

● Obtain matrix M where M = D - A.

● Find an (i,j) cofactor of M, determined by: $(-1)^{(i+j)}$ * determinant (M'), where M' = resulting matrix on removing row i and column j from M. **This cofactor gives the number of spanning trees for graph G.**

● Suppose adjacency matrix A was represented with the edge names (in lowercase letters) in place of 1's for connected edges, and 0's for non-connected edges. D would then have its diagonal entries be the sums of the columns, and 0's for the remaining elements. **If we take the (i,j) cofactor of matrix M, where M = D - A, then the result gives all possible configurations for spanning trees of graph G**

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com

Minimum Spanning Trees (MST)

› For a **weighted graph G**, the **minimum spanning tree is a spanning tree with the smallest possible sum of edge weights**

› For a weighted graph G, **MSTs are not unique**

› Algorithms to determine the MST of a graph G are **greedy algorithms**

› **PRIM'S ALGORITHM:**

● Pick a random vertex as the starting vertex of the MST

● Follow steps 3 to 5 till there are vertices not included in the MST (called fringe vertices)

● Find edges connecting any tree vertex with the fringe vertices

● Find the minimum along these edges

● Add the chosen edge to the MST as long as it doesn't form a cycle

● By the end, you'll obtain the MST

› **KRUSKAL'S ALGORITHM:**

● Sort edges in increasing order of weight

● Pick the first edge in the sorted order

● Iterate through the sorted order from the second edge onwards till the end, and see if you can add that edge without creating a cycle

● By the end, the MST will be obtained.

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com