Chapter 2

Mathematical Preliminaries

2.1 Sets

Definition 2.1 (Set Partition)**.** *Given a set S, a partitioning of S is a set T of subsets of S such that each element of S is in exactly one subset $t \in T$. We refer to each element of T as a* **partition** *and the set T as a* **partitioning** *of S.*

2.2 Relations and Functions

Definition 2.2 (Cartesian Product)**.** *Consider two sets A and B. The* **Cartesian product** $A \times B$ *is the set of all ordered pairs (a, b) where $a \in A$ and $b \in B$. In set notation this is*

$$\{(a, b) : a \in A, b \in B\}\,.$$

Definition 2.3 (Relation)**.** *A* **(binary) relation from a set A to set B** *is a subset of the Cartesian product of A and B. For a relation $R \subseteq A \times B$, the set $\{a : (a, b) \in R\}$ is referred to as the* **domain** *of R, and the set $\{b : (a, b) \in R\}$ is referred to as the* **range** *of R.*

Definition 2.4 (Function or Mapping)**.** *A* **mapping from A to B** *is a relation $R \subset A \times B$ such that $|R| = |domain(R)|$, i.e., for every a in the domain of R there is only one b such that $(a, b) \in R$. A mapping is also called a* **function**.

Example 2.5. *The Cartesian product of $A = \{0, 1, 2, 3\}$ and $B = \{a, b\}$ is the set of all pairings:*

$$A \times B = \{(0, a), (0, b), (1, a), (1, b), (2, a), (2, b), (3, a), (3, b)\} \ .$$

The set:

$$X = \{(0, a), (0, b), (1, b), (3, a)\}$$

is a relation from A to B since $X \subset A \times B$, but not a mapping (function) since a is repeated.
On the other hand,

$$Y = \{(0, a), (1, b), (3, a)\}$$

is both a relation and a function from A to B since each element only appears once on the left.
The domain of Y is $\{0, 1, 3\}$ and the range is $\{a, b\}$. It is, however, not a sequence since there is a gap in the domain.

2.3 Trees

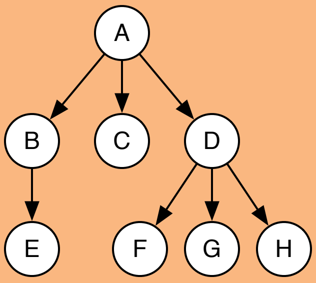
A rooted tree is a tree with a distinguished root node that can be used to access all other nodes (Definition **??**). An example of a rooted tree along with the associated terminology is given in Example 2.7, and an example of a binary search tree is given in Example **??**.

Definition 2.6 (Rooted Tree)**.** *A* rooted tree *is a directed graph such that*

1. *One of the vertices is the* root *and it has no in edges.*
2. *All other vertices have one in-edge.*
3. *There is a path from the root to all other vertices.*

Terminology. When talking about rooted trees, by convention we use the term *node* instead of vertex. A node is a *leaf* if it has no out edges, and an *internal node* otherwise. For each directed edge (u, v), u is the *parent* of v, and v is a *child* of u. For each path from u to v (including the empty path with $u = v$), u is an *ancestor* of v, and v is a *descendant* of u. For a vertex v, its *depth* is the length of the path from the root to v and its *height* is the longest path from v to any leaf. The *height of a tree* is the height of its root. For any node v in a tree, the *subtree rooted at* v is the rooted tree defined by taking the induced subgraph of all vertices reachable from v (i.e. the vertices and the directed edges between them), and making v the root. As with graphs, an *ordered rooted tree* is a rooted tree in which the out edges (children) of each node are ordered.

Example 2.7. *An example of a rooted tree:*



root	A
leaves	E, C, F, G, *and* H
internal nodes	A, B, *and* D
children of A	B, C *and* D
parent of E	B
descendants of A	*all nodes, including* A *itself*
ancestors of F	F, D *and* A
depth of F	*2*
height of B	*1*
height of the tree	*2*
subtree rooted at D	*the rooted tree consisting of* D, F, G *and* H

2.4 Graph Theory

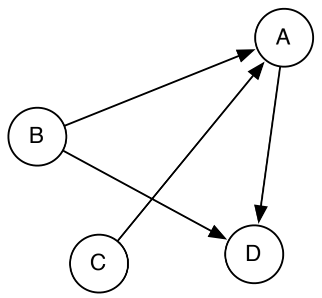
2.4.1 Basic Definitions

Definition 2.8. *A* directed graph *or (*digraph*) is a pair* $G = (V, A)$ *where*

- V *is a set of* vertices *(or nodes), and*
- $A \subseteq V \times V$ *is a set of* directed edges *(or arcs).*

In a digraph, each arc is an ordered pair $e = (u, v)$. A digraph can have *self loops* (u, u).

Exercise 2.9. *Identify the vertices and the arcs in the example digraph shown in Figure 12.1.*



An example of a directed graph on 4 vertices.

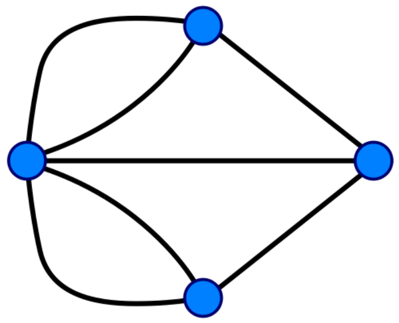

An undirected graph on 4 vertices[1]

Figure 2.1: Example Graphs.

Definition 2.10 (Undirected graph)**.** *An* undirected graph *is a pair $G = (V, E)$ where*

- *V is a set of* vertices *(or nodes), and*
- *$E \subseteq \binom{V}{2}$ is a set of edges.*

In an undirected graph, each edge is an unordered pair $e = \{u, v\}$ (or equivalently $\{v, u\}$). By this definition an undirected graph cannot have self loops since $\{v, v\} = \{v\} \notin \binom{V}{2}$.

While directed graphs represent asymmetric relationships (my web page points to yours, but yours does not necessarily point back), undirected graphs represent symmetric relationships.

Directed graphs are in some sense more general than undirected graphs since we can easily represent an undirected graph by a directed graph by placing an arc in each direction. Indeed, this is often the way we represent undirected graphs in data structures.

Graphs come with a lot of terminology, but fortunately most of it is intuitive once we understand the concept. At this point, we will just talk about graphs that do not have any data associated with edges, such as weights. In Chapter 16 we will talk about weighted graphs, where the weights on edges can represent a distance, a capacity or the strength of the connection.

Neighbors. A vertex u is a ***neighbor*** of (or equivalently ***adjacent*** to) a vertex v in a graph $G = (V, E)$ if there is an edge $\{u, v\} \in E$. For a directed graph a vertex u is an ***in-neighbor*** of a vertex v if $(u, v) \in E$ and an ***out-neighbor*** if $(v, u) \in E$. We also say two edges or arcs are neighbors if they share a vertex.

Neighborhood. For an undirected graph $G = (V, E)$, the ***neighborhood*** $N_G(v)$ of a vertex $v \in V$ is its set of all neighbors of v, i.e., $N_G(v) = \{u \mid \{u, v\} \in E\}$. For a directed graph we use $N_G^+(v)$ to indicate the set of out-neighbors and $N_G^-(v)$ to indicate the set of in-neighbors of v. If we use $N_G(v)$ for a directed graph, we mean the out neighbors. The neighborhood of a set of vertices $U \subseteq V$ is the union of their neighborhoods, e.g. $N_G(U) = \bigcup_{u \in U} N_G(y)$, or $N_G^+(U) = \bigcup_{u \in U} N_G^+(u)$.

Incidence. We say an edge is ***incident*** on a vertex if the vertex is one of its endpoints. Similarly we say a vertex is incident on an edge if it is one of the endpoints of the edge.

Degree. The ***degree*** $d_G(v)$ of a vertex $v \in V$ in a graph $G = (V, E)$ is the size of the neighborhood $|N_G(v)|$. For directed graphs we use ***in-degree*** $d_G^-(v) = |N_G^-(v)|$ and ***out-degree*** $d_G^+(v) = |N_G^+(v)|$. We will drop the subscript G when it is clear from the context which graph we're talking about.

Paths. A ***path*** in a graph is a sequence of adjacent vertices. More formally for a graph $G = (V, E)$, $\mathit{Paths}(G) = \{P \in V^+ \mid 1 \le i < |P|, (P_i, P_{i+1}) \in E\}$ is the set of all paths in G, where V^+ indicates all positive length sequences of vertices (allowing for repeats). The length of a path is one less than the number of vertices in the path—i.e., it is the number of edges in the path. A path in a finite graph can have infinite length. A ***simple path*** is a path with no repeated vertices. Please see the remark below, however.

Remark 2.11. *Some authors use the terms walk for path, and path for simple path. Even in this book when it is clear from the context we will sometimes drop the "simple" from simple path.*

Reachability and connectivity. A vertex v is ***reachable*** from a vertex u in G if there is a path starting at v and ending at u in G. We use $R_G(v)$ to indicate the set of all vertices reachable from v in G. An undirected graph is ***connected*** if all vertices are reachable from all other vertices. A directed graph is ***strongly connected*** if all vertices are reachable from all other vertices.

Cycles. In a directed graph a ***cycle*** is a path that starts and ends at the same vertex. A cycle can have length one (i.e. a ***self loop***). A ***simple cycle*** is a cycle that has no repeated vertices other than the start and end vertices being the same. In an undirected graph a (simple) ***cycle*** is a path that starts and ends at the same vertex, has no repeated vertices other than the first and last, and has length at least three. In this course we will exclusively talk about simple cycles and hence, as with paths, we will often drop simple.

Exercise 2.12. *Why is important in a undirected graph to require that a cycle has length at least three? Why is important that we do not allow repeated vertices?*

Trees and forests. An undirected graph with no cycles is a ***forest*** and if it is connected it is called a ***tree***. A directed graph is a forest (or tree) if when all edges are converted to undirected edges it is undirected forest (or tree). A ***rooted tree*** is a tree with one vertex designated as the root. For a directed graph the edges are typically all directed toward the root or away from the root.

Directed acyclic graphs. A directed graph with no cycles is a ***directed acyclic graph*** (DAG).

Distance. The ***distance*** $\delta_G(u, v)$ from a vertex u to a vertex v in a graph G is the shortest path (minimum number of edges) from u to v. It is also referred to as the ***shortest path length*** from u to v.

Diameter. The ***diameter*** of a graph is the maximum shortest path length over all pairs of vertices: $\mathsf{diam}(G) = \max\{\delta_G(u, v) : u, v \in V\}$.

Multigraphs. Sometimes graphs allow multiple edges between the same pair of vertices, called ***multi-edges***. Graphs with multi-edges are called ***multi-graphs***. We will allow multi-edges in a couple algorithms just for convenience.

Sparse and dense graphs. By convention we will use the following definitions:

$$\begin{aligned} n &= |V| \\ m &= |E| \end{aligned}$$

Note that a directed graph can have at most n^2 edges (including self loops) and an undirected graph at most $n(n - 1)/2$. We informally say that a graph is *sparse* if $m \ll n^2$ and *dense* otherwise. In most applications graphs are very sparse, often with only a handful of neighbors per vertex when averaged across vertices, although some vertices could have high degree. Therefore, the emphasis in the design of graph algorithms, at least for this book, is typically on algorithms that work well for sparse graphs.

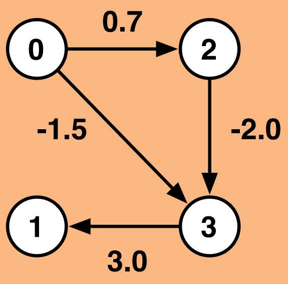
2.4.2 Weighted Graphs and Their Representation

Many applications of graphs require associating weights or other values with the edges of a graph. Such graphs can be defined as follows.

Definition 2.13 (Weighted and Edge-Labeled Graphs)**.** *An* edge-labeled graph *or a* weighted graph *is a triple $G = (E, V, w)$ where $w : E \to L$ is a function mapping edges or directed edges to their labels (weights) , and L is the set of possible labels (weights).*

In a graph, if the data associated with the edges are real numbers, we often use the term "weight" to refer to the edge labels, and use the term "weighted graph" to refer to the graph. In the general case, we use the terms "edge label" and edge-labeled graph. Weights or other values on edges could represent many things, such as a distance, or a capacity, or the strength of a relationship.

Example 2.14. *An example directed weighted graph.*



As it may be expected, basic terminology on graphs defined above straightforwardly extend to weighted graphs.

2.4.3 Subgraphs

When working with graphs, we sometimes wist to refer to parts of a graph. To this end, we can use the notion of a subgraph, which refers to a graph contained in a larger graph. A subgraph can be defined as any subsets of edges and vertices that together constitute a well defined graph.

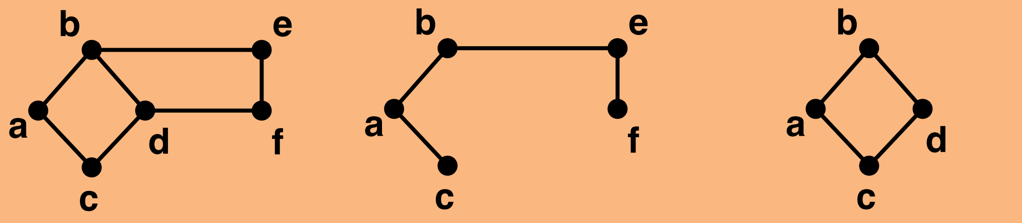
Definition 2.15 (Subgraph)**.** *Let $G = (V, E)$ and $H = (V', E')$ be two graphs. H is a subgraph of if $V' \subseteq V$ and $E' \subseteq E$.*

Note that since H is a graph, the vertices defining each edge are in the vertex set V', i.e., for an undirected graph $E' \subseteq \binom{V'}{2}$). There are many possible subgraphs of a graph.

An important class of subgraphs are ***vertex-induced subgraphs***, which are maximal subgraphs defined by a set of vertices. A vertex-induced subgraph is maximal in the sense that it contains all the edges that it can possibly contain. In general when an object is said to be a *maximal* "X", it means that nothing more can be added to the object without violating the property "X".

Definition 2.16 (Vertex-Induced Subgraph)**.** *The subgraph of $G = (V, E)$ induced by $V' \subseteq V$ is the graph $H = (V', E')$ where $E' = \{\{u, v\} \in E \mid u \in V', v \in V'\}$.*

Example 2.17. *Some vertex induced subgraphs:*



Original graph *Induced by* $\{a, b, c, e, f\}$ *Induced by* $\{a, b, c, d\}$

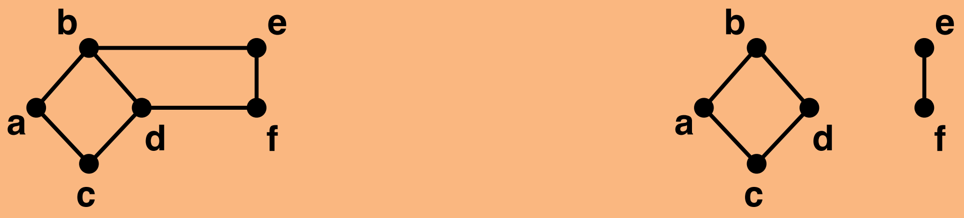
Although we will not use it in this course , it is also possible to define an induced subgraph in terms of a set of edges by including in the graph all the vertices incident on the edges.

Definition 2.18 (Edge-Induced Subgraph). *The subgraph of* $G = (V, E)$ *induced by* $E' \subseteq E$ *is a graph* $H = (V', E')$ *where* $V' = \cup_{e \in E} e$.

2.4.4 Connectivity and Connected Components

Recall that in a graph (either directed or undirected) a vertex v is reachable from a vertex u if there is a path from u to v. Also recall that an undirected graph is connected if all vertices are reachable from all other vertices.

Example 2.19. *Two example graphs shown. The first in connected; the second is not.*



Graph 1 *Graph 2*

An important subgraph of an undirected graph is a ***connected component*** of a graph.

Definition 2.20 (Connected Component or Component). *Let* $G = (V, E)$ *be a graph. A subgraph* H *of* G *is a connected component of* G *if it is a maximal connected subgraph of* G.

Here, “maximal” means we cannot add any more vertices and edges from G to H without disconnecting H. In the graphs shown in Example 2.19, the first graph has one connected component (hence it is connected); the second has two connected components.

Using vertex-induced subgraphs, we can specify a connected component of a graph by simply specifying the vertices in the component.

Example 2.21. *Connected components of our second example graph Example 2.19 can be specified as* $\{a, b, c, d\}$ *and* $\{e, f\}$.