Chapter 2

Basic graph definitions

To quote Berge,

> It would be convenient to say that there are two theories and two kinds of graphs: directed and undirected. This is not true. *All graphs are directed,* but sometimes the direction need not be specified.

That is, for specific graph problems it is convenient to ignore the distinction between endpoints.

We define one combinatorial structure, a *graph*.[1] There are three ways to interpret this combinatorial structure, as an *undirected* graph, as a *directed* graph, and as a *bidirected* graph. Each kind of graph has its uses, and it is convenient to be able to view the underlying graph from these different perspectives.

In the traditional definition of graphs, vertices are in a sense primary, and edges are defined in terms of the vertices. We used this approach in defining rooted trees in Chapter 1. In defined graphs, we choose to make edges primary, and we will define vertices in terms of edges.

There are three reasons for choosing the edge-centric view:

- Self-loops and multiple edges, which occur often, are more simply handled by an edge-centric view.
- Contraction, a graph operation we discuss later, transforms a graph in a way that changes the identity of vertices but not of edges. The edge-centric view is more natural in this context, and simplifies the tracking of an edge as the graph undergoes contractions.
- The dual of an embedded graph is usefully viewed as a graph with the same edges, but where those edges form a different topology.

There is one seeming disadvantage: our definition of graphs does not permit the existence of isolated vertices, vertices with no incident edges. This disadvantage is mitigated by interpreting a subset of edges of a graph as a kind of subgraph.

[1] Our definition allows for self-loops and multiple edges, a structure traditionally called a *multigraph*.

2.1 Edge-centric definition of graphs

For any finite set E, a *graph on E* is a pair $G = (V, E)$ where V is a partition of the set $E \times \{1, -1\}$, called the *dart set* of G. That is, V is a collection of disjoint, nonempty, mutually exhaustive subsets of $E \times \{1, -1\}$. Each subset is a *vertex* of G. (The word *node* is synonymous with *vertex*). For any $e \in E$, the *darts of e* are the pairs $(e, +1)$ and $(e, -1)$, of which the *primary dart of e* is $(e, +1)$. For brevity, we can write $(e, +1)$ as e^+ and $(e, -1)$ as e^-.

For a graph $G = (V, E)$, we use $E(G)$ to denote E, the edge set of G, and we use $V(G)$ to denote V, the vertex set of G, and we use $D(G)$ to denote the dart set of of G.

Problem 2.1. *Write pseudocode to implement BFS given a graph represented as a partition of its dart set*

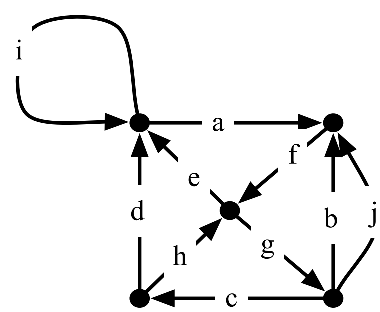

Figure 2.1: The vertex v is the subset of darts $\{(e, -1), (f, 1), (g, -1), (h, 1)\}$. An example of a walk is $(j, 1)$ $(a, -1)$ $(i, 1)$ $(i, -1)$ $(d, -1)$ $(d, 1)$.

rev Define the bijection rev on darts by $\text{rev}((e, \sigma)) = (e, -\sigma)$. For a dart d, $\text{rev}(d)$ is called the *reverse* of d, and is sometimes written as d^R.

endpoints, head and tail, self-loops, parallel edges The *head* of a dart (e, σ) is the block $v \in V$ such that v contains (e, σ). The *tail* of (e, σ) is the head of $(e, -\sigma)$.

Each element $e \in E$ has two *endpoints*, namely the head and tail of $(e, 1)$. If the endpoints are the same vertex, we call e a *self-loop*. In Figure 2.1, i is a self-loop. If two elements have the same endpoints, we say they are *parallel*, for example, b and j are parallel in Figure 2.1.

Edges and arcs We can interpret an element $e \in E$ as a directed *arc*, in which case we distinguish between its head and tail, which are, respectively, the head and tail of the primary dart $(e, +1)$. If we interpret e as an undirected *edge*, we do not distinguish between its endpoints. Thus use of the word *edge* or *arc* indicates whether we intend to interpret the element as undirected or directed. The edge or arc of a dart (e, σ) is defined to be e.

We will sometimes use the notation uv to refer to an edge or arc or dart whose endpoints are the vertices u and v (and where, if referring to an arc or dart, u is the tail and v is the head). This notation is formally problematic because the graph might have multiple edges with the same endpoints; we trust that the reader will not be confused.

Parallel arcs/edges and self-loops If two arcs have the same tail and the same head, we say they are *parallel arcs*. If two edges have the same pair of endpoints, we say they are *parallel edges*. If the endpoints of an edge/arc are the same, we say it is a *self-loop*. Our definition of graph permits parallel edges and self-loops.

Incidence, degree We say an edge/arc/dart is *incident* to a vertex v if v is one of the endpoints. The *degree* of a vertex v (written $\text{degree}(v)$) is the number of occurences of v as an endpoint of elements of E (counting multiplicity[2]). The outdegree of v (written $\text{outdegree}(v)$) is the number of arcs having v as a tail, and the indegree (written $\text{indegree}(v)$) is the number of arcs having v as a head.

Endpoint notation We sometimes write an arc as uv to indicate that its tail is u and its head is v, and we sometimes write an edge the same way to indicate that its endpoints are u and v. This notation has the potential to be ambiguous because of the possibility of parallel edges.

$V(G)$ **and** $E(G)$ For a graph $G = (V, E)$, we use $V(G)$ and $E(G)$ to denote V and E, respectively, and we use $n(G)$ and $m(G)$ to denote $|V(G)|$ and $|E(G)|$. We use $D(G)$ to denote the set of darts of G. We may leave the graph G unspecified if doing so introduces no ambiguity.

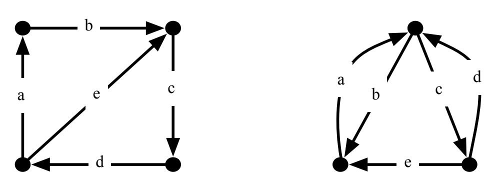

Figure 2.2: Two graphs corresponding to the edges $a, \ldots, e$.

[2]That is, a self-loop contributes two to the degree of a vertex.

2.2 Walks, paths, and cycles

Walks As illustrated in Figure 2.1, a non-empty sequence

$$d_1 \dots d_k$$

of darts is a *walk* if the head of d_i is the tail of d_{i+1} for every $1 \leq i \leq k$. To be more specific, it is a *x-to-y walk* if x is d_1 or the tail of d_1 and y is d_k or the head of d_k. We define d_i to be the *successor* in W of d_i to be d_{i+1} and we define *predecessor* of d_{i+1} to be d_i. We may designate a walk to be a *closed walk* if the tail of d_1 is the head of d_k, in which case we define the successor of d_k to be d_1 and the predecesor of d_1 to be d_k. We also refer to a closed walk as a *tour*.

Paths and cycles A walk is called a *path of darts* if the darts are distinct, a *cycle of darts* if in addition it is a closed walk. A path/cycle of darts is called a *path/cycle of arcs* if each dart is of the form $(e, +1)$. It is called a *path/cycle of edges* if no edge is represented twice.

Simple paths and cycles, internal vertices A cycle is *simple* if every vertex occurs at most once as the head of some d_i. A path is simple if it is not a cycle and every vertex occurs at most once as the head of some d_i. A vertex is said to belong to the path or cycle if the vertex is an endpoint of some d_i. The *internal vertices* of a path $d_1 \dots d_k$ are the heads of $d_1, \dots, d_{k-1}$. Two paths/cycles are dart-disjoint if they share no darts, and are vertex-disjoint if they share no vertices. Two paths are *internally vertex-disjoint* if they share no internal vertices.

Walks, paths, and cycles of arcs/edges A sequence $e_1, \dots, e_k$ of elements of E is a *directed walk* (or *diwalk*) if the sequence of corresponding darts $(e_1, 1), \dots, (e_k, 1)$ is a walk. It is a *directed path* (or *dipath*) if, in addition, $e_1, \dots, e_k$ are distinct. It is an undirected walk if there exist $i_1, \dots, i_k \in \{1, -1\}$ such that the sequence of darts $(e_1, i_1), \dots, (e_k, i_k)$ is a walk. It is an undirected path if in addition $e_1, \dots, e_k$ are distinct. The other definitions given for sequences of darts apply straightforwardly to paths consisting of elements of E.

Empty walks and paths In the above, we neglected to account for the possibility of an *empty* walk or path. Empty walks and paths are defined by a vertex in the graph; they contain no darts. We do not allow for the existence of empty cycles.

Lemma 2.2.1. *A u-to-v walk of darts contains a u-to-v path of darts as a subsequence.*

2.2.1 Connectedness

Given a graph $G = (V, E)$, for a vertex or dart x and a vertex or dart y, we say x and y are *connected* in G if there is a v_1-to-v_2 path of darts in G. Similarly, edges e_1 and e_2 are connected in G if there is a path of darts that starts with a dart of e_1 and ends with a dart of e_2.

More generally, given a subset E' of E, we say that v_1, v_2 are connected via E' in G if there is a v_1-to-v_2 path using only darts corresponding to edges of E'.

A subset of V is connected in a graph if every two vertices in the subset are connected. Connectedness is an equivalence relation on the vertex set. A *connected component* is an equivalence class of this equivalence relation. Equivalently, a connected component is a maximal connected vertex subset. Let $\kappa(G)$ denote the number of connected components of G.

The notion of connectivity can also be applied to edges; two edges are connected if there is a path containing both of them. Thus the phrase *connected component* could also refer to an equivalence class of this relation on edges, i.e. a maximal connected edge subset. The reader should be able to discern which meaning is intended.

2.2.2 Two-edge-connectivity and cut-edges

Edges e_1 and e_2 are *two-edge-connected* in G if G contains a cycle of edges containing both of them. The equivalence classes of this relation are called *two-edge-connected components*.

An edge e of G is a *cut-edge* if the two-edge-connected component containing e contains no other edges.

Lemma 2.2.2 (Cut-Edge Lemma)**.** *An edge e of G is a cut-edge iff every path between its endpoints uses e.*

2.2.3 Subgraphs and edge subgraphs

We will use the term *subgraph* in two ways:

1. According to the traditional definition, a *subgraph* of a graph $G = (V, E)$ is simply a graph $H = (V', E')$ such that $V' \subseteq V$ and $E' \subseteq E$.

2. Because we often want to relate features of a subgraph to the graph from which it came, we will often think of a subset E' of edges of G as a subgraph.

Ordinarily, the reader will be able to determine from context which meaning is intended (or it will not matter). When necessary, we will use the term *edge subgraph* to distinguish the latter meaning.

In a slight extension of this concept, in Chapter , when discussing the traveling salesman problem, we will work with an *edge multisubgraph* E', where E' is a *multiset* whose elements come from E.

One significant distinction between a graph and an edge subgraph is this: according to our definition, a graph G cannot contain a vertex with no incident edges, whereas ordinarily an edge subgraph E' is considered to include all the

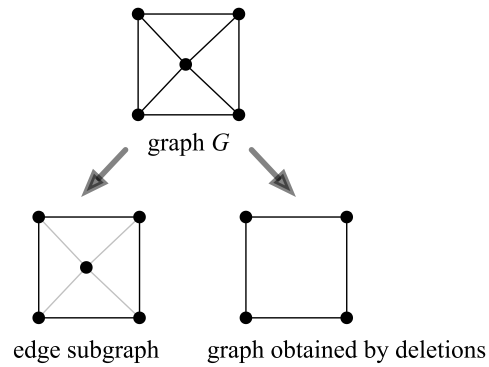

Figure 2.3: This figure illustrates the difference between an edge subgraph (shown on the bottom-left) and a traditional subgraph, a graph obtained by edge deletions (shown on the bottom-right). In the graph obtained by deletions, the center vertex does not exist since all its incident edges have been deleted. The edge subgraph does not formally include the grayed-out edges but still contains the center vertex. There are other advantages to the edge subgraph that we will discuss in the context of graph embeddings.

vertices of the original graph, and hence can include a vertex none of whose incident edges belong to E'.

The usual definitions (walk, path, cycle, connectedness, two-edge-connectivity) extend to an edge subgraph by restricting the darts comprising these structures to those darts corresponding to edges in E'. For example, two vertices x and y of G are connected in the edge subgraph E' if there is an x-to-y path of darts belonging to E'. As in graphs, a connected component of an edge subgraph of G is a maximal connected subset of $V(G)$. We define $\kappa_G(E')$ to be the number of connected components in this sense. For example, the edge subgraph on the bottom-left in Figure 2.3 has two connected components. (The graph on the bottom-right has only one.)

2.2.4 Deletion of edges and vertices

Deleting a set S of *edges* from G is the operation on a graph that results in the subgraph or edge subgraph of G consisting of the edges of G not in S. We denote this subgraph or edge subgraph by $G - S$.

The result of *deleting* a set V' of *vertices* from G is the graph (not the edge subgraph) obtained by deleting all the edges incident to the vertices in V'. This subgraph is denoted $G - V'$. Since isolated vertices (vertices with no incident edges) cannot exist according to our definition of graphs, deleted vertices cease to exist when deleted.

Deletion of multiple edges and/or vertices results in a graph or edge-subgraph that is independent of the order in which the deletions occured.

2.2.5 Contraction of edges

For a graph $G = (V, E)$ and an edge $uv \in E$, the *contraction of* e *in* G is an operation that produces the graph $G' = (V', E')$, where

- $E' = E - \{uv\}$, and
- the part of V containing u and the part of V containing v are merged (and uv is removed) to form a part V'.

The graph obtained from G by contracting e is denoted G/e.

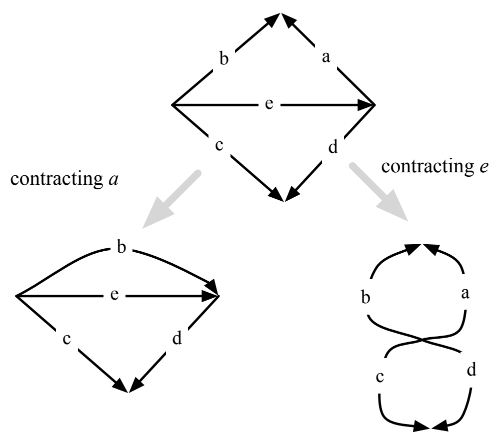

Figure 2.4: (a) A graph with edges $a, \ldots, e$. (b) The graph after the contraction of edge a.

Like deletions, the order of contractions of edges does not affect the result. For a set S of edges, the graph obtained by contracting the edges of S is denoted G/S.

2.2.6 Minors

A graph H is said to be a *minor* of a graph G if H can be obtained from G by edge contractions and edge deletions. The relation "is a minor of" is clearly reflexive, transitive, and antisymmetric.

Note that each vertex v of H corresponds to a *set* of vertices in G (the set merged to form v).