Data structure and algorithm in Python

Graph

Xiaoping Zhang

School of Mathematics and Statistics, Wuhan University

Table of contents

Graphs

Graphs

Example : Graphs

A graph is a way of representing relationships that exist between pairs of objects. That is, a graph is a set of objects, called vertices, together with a collection of pairwise connections between them, called edges.

- A graph G is simply a set V of vertices and a collection E of pairs of vertices from V, called edges.
- A graph is a way of representing connections or relationships between pairs of objects from some set V.

Graphs

Edges in a graph are either directed or undirected.

- An edge (u, v) is said to be directed from u to v if the pair (u, v) is ordered, with u preceding v.
- An edge (u, v) is said to be undirected if the pair (u, v) is not ordered.

Undirected edges are sometimes denoted with set notation, as $\{u, v\}$, but for simplicity we use the pair notation (u, v), noting that in the undirected case (u, v) is the same as (v, u).

Graphs

Example : undirected, directed and mixed graph

- If all the edges in a graph are undirected, then we say the graph is an undirected graph.
- Likewise, a directed graph, also called a digraph, is a graph whose edges are all directed.
- A graph that has both directed and undirected edges is often called a mixed graph.

An undirected or mixed graph can be converted into a directed graph by replacing every undirected edge (u, v) by the pair of directed edges (u, v) and (v, u).

Graphs

- The two vertices joined by an edge are called the end vertices (or endpoints) of the edge.
- If an edge is directed, its first endpoint is its origin and the other is the destination of the edge.
- Two vertices u and v are said to be adjacent if there is an edge whose end vertices are u and v.

Graphs

- An edge is said to be incident to a vertex if the vertex is one of the edge' s endpoints.
- The outgoing edges of a vertex are the directed edges whose origin is that vertex.
- The incoming edges of a vertex are the directed edges whose destination is that vertex.

Graphs

- The degree of a vertex v, denoted $deg(v)$, is the number of incident edges of v.
- The in-degree and out-degree of a vertex v are the number of the incoming and outgoing edges of v, and are denoted $indeg(v)$ and $outdeg(v)$, respectively.

Graphs

- The definition of a graph refers to the group of edges as a collection, not a set, thus allowing two undirected edges to have the same end vertices, and for two directed edges to have the same origin and the same destination. Such edges are called parallel edges or multiple edges.
- Another special type of edge is one that connects a vertex to itself. Namely, we say that an edge (undirected or directed) is a self-loop if its two endpoints coincide.

Graphs

- With few exceptions, graphs do not have parallel edges or self-loops. Such graphs are said to be simple.
- Thus, we can usually say that the edges of a simple graph are a set of vertex pairs (and not just a collection).

Throughout this chapter, we assume that a graph is simple unless otherwise specified.

Graphs

- A path is a sequence of alternating vertices and edges that starts at a vertex and ends at a vertex such that each edge is incident to its predecessor and successor vertex.
- A cycle is a path that starts and ends at the same vertex, and that includes at least one edge.

1. A path is simple if each vertex in the path is distinct.
2. A cycle is simple if each vertex in the cycle is distinct, except for the first and last one.
3. A directed path is a path such that all edges are directed and are traversed along their direction.
4. A directed cycle is similarly defined.

Graphs

- A directed graph is acyclic if it has no directed cycles.
- If a graph is simple, we may omit the edges when describing path P or cycle C, as these are well defined, in which case P is a list of adjacent vertices and C is a cycle of adjacent vertices.

Graphs

- Given vertices u and v of a (directed) graph G, we say that u reaches v, and that v is reachable from u, if G has a (directed) path from u to v.
- In an undirected graph, the notion of reachability is symmetric, that is to say, u reaches v if an only if v reaches u.
- However, in a directed graph, it is possible that u reaches v but v does not reach u, because a directed path must be traversed according to the respective directions of the edges.

- A graph is connected if, for any two vertices, there is a path between them.
- A directed graph $\vec{G}$ is strongly connected if for any two vertices u and v of $\vec{G}$, u reaches v and v reaches u.

- A subgraph of a graph G is a graph H whose vertices and edges are subsets of the vertices and edges of G, respectively.
- A spanning subgraph of G is a subgraph of G that contains all the vertices of the graph G.
- If a graph G is not connected, its maximal connected subgraphs are called the connected components of G.
- A forest is a graph without cycles.
- A tree is a connected forest, that is, a connected graph without cycles.
- A spanning tree of a graph is a spanning subgraph that is a tree.

Graphs

Proposition

If G is a graph with m edges and vertex set $\mathcal{V}$, then

$$\sum_{v \in \mathcal{V}} deg(v) = 2m.$$

Graphs

Proposition

If G is a directed graph with m edges and vertex set $\mathcal{V}$, then

$$\sum_{v \in \mathcal{V}} indeg(v) = \sum_{v \in \mathcal{V}} outdeg(v) = m.$$

Proposition

Let G be a simple graph with n vertices and m edges.

- If G is undirected, then

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

- If G is directed, then

$$m \leq n(n-1).$$

Proposition

Let G be a undirected graph with n vertices and m edges.

- If G is connected, then

$$m \geq n - 1.$$

- If G is a tree, then

$$m = n - 1.$$

- If G is a forest, then

$$m \leq n - 1.$$

Graphs

The Graph ADT

The Graph ADT

Since a graph is a collection of vertices and edges, we model the abstraction as a com- bination of three data types:

- Vertex
- Edge
- Graph

The Graph ADT

A Veretex is lightweight object that stores an arbitrary element provided by the user, supporting the method:

- `element()`: Retrieve the stored element.

The Graph ADT

An Edge stores an associated object, supporting the following methods:

- `element()`: Retrieve the stored element.
- `endpoints()`: Return a tuple (u, v) such that vertex u is the origin of the edge and vertex v is the destination; for an undirected graph, the orientation is arbitrary.
- `opposite(v)`: Assuming vertex v is one endpoint of the edge (either origin or destination), return the other endpoint.

The Graph ADT

The primary abstraction for a graph is the Graph ADT. We presume that a graph can be either undirected or directed, with the designation declared upon construction; recall that a mixed graph can be represented as a directed graph, modeling edge $\{u, v\}$ as a pair of directed edges (u, v) and (v, u).

The Graph ADT

The Graph ADT includes the following methods

- `vertex_count()`: Return the number of vertices.
- `vertices()`: Return an iteration of all vertices.
- `edge_count()`: Return the number of edges.
- `edges()`: Return an iteration of all edges.
- `get_edge(u,v)`: Return the edge from vertex u to v, if one exists; otherwise return None.
- `degree(v, out=True)`: For an undirected graph, return the number of edges incident to vertex v. For a directed graph, return the number of outgoing (resp. incoming) edges incident to vertex v, as designated by the optional parameter.

The Graph ADT

- `incident_edges(v, out=True)`: Return an iteration of all edges incident to vertex v. In the case of a directed graph, report outgoing edges by default; report incoming edges if the optional parameter is set to False.
- `insert_vertex(x=None)`: Create and return a new Vertex storing element x.
- `insert_edge(u, v, x=None)`: Create and return a new Edge from vertex u to vertex v, storing element x (None by default).
- `remove_vertex(v)`: Remove vertex v and all its incident edges from the graph.
- `remove_edge(e)`: Remove edge e from the graph.