CIS 575. Introduction to Algorithm Analysis
Material for January 31, 2024

Graphs: Basic Concepts

The topic of this note is treated in great detail in *Cormen*'s Appendix B.4.

1 Graphs

A **graph** consists of **nodes**, also called **vertices**, connected by **edges**. Formally, a graph G is a pair (V, E) where V is the vertices and E the edges. There are two kinds of graphs:

Directed Graphs where each edge has a direction; it goes from its **source** to its **target** (which may equal the source in that we allow *self-loops*)

Undirected Graphs where each edge does not have a direction; it is between two *different* nodes (we do *not* allow self-loops).

An edge between nodes u and w in an undirected graph can be considered an edge from u to w and also an edge from w to u. For both kinds of graph, if an edge goes from u to w then u and w are referred to as the **end points** of that edge. An edge may have extra data associated with it, for example a number (which could denote the distance between its end points).

Observe that we are *not* modeling *hypergraphs* (which would allow an edge to connect arbitrarily many nodes): in an undirected graph, each edge thus has exactly two end points; in a directed graph, each edge has two end points except for a self-loop which has only one.

Also, we are *not* modeling *multigraphs* in that we do not allow "parallel edges": for a directed graph, two different edges cannot have the same source *and* the same target (even if they have different extra data); for an undirected graph, two different edges cannot have the same two end points.

Figure 1 depicts an example undirected graph (left), and an example directed graph (right).

A **path** in a graph is a sequence of nodes $u_0, u_1, \ldots u_{n-1}, u_n$ $(n \geq 0)$ where for each $i \in 1..n$ there is an edge from u_{i-1} to u_i; such a path is from u_0 to u_n and if $n > 0$ we say the path is *non-empty*. For an undirected graph, if there is a path from u to w then there also is a path from w to u.

For the undirected graph in Figure 1(left), there is a path A, B, D from A to D whereas the node sequence A, C, B is not a valid path (since there is no edge from A to C). For the directed graph in Figure 1(right), there is a path Y, Z, W from Y to W but also a path Y, Z, X, Y, Z, W from Y to W.

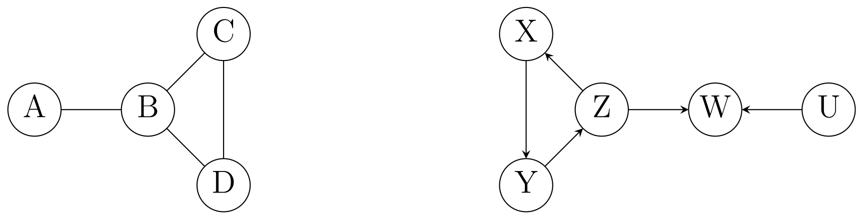

Figure 1: An undirected graph (left), and a directed graph (right).

1.1 Size Measures

Until now in this course, we have expressed the running time (or space use) of an algorithm in terms of a single entity n, usually the number of elements in the array given as input. But for an algorithm that as input takes a graph (V, E), we shall often express its running time in terms of *two* entities: **n**, the number of nodes (which is $|V|$), and **a**, the number of edges (which is $|E|$).

It turns out that many graph algorithms run in time $\Theta(n + a)$. If we know that $a \in O(n)$ (in which case the graph will be "sparse"), this can be rewritten as $\Theta(n)$; if we know that $a \in \Theta(n^2)$ (in which case the graph will be "dense"), this can be rewritten as $\Theta(n^2)$. But without further assumptions on the graph, we can*not* further simplify $\Theta(n + a)$. On the other hand, we can always simplify $\Theta(n^2 + a)$ to $\Theta(n^2)$ since $a \leq n^2$ holds for all graphs (as we are not allowing multigraphs).

For the purpose of analyzing an algorithm on a graph (V, E), it is often convenient for each $i \in V$ to write a_i for the number of edges from node i; in that case $\sum_{i \in V} a_i = a$ for a directed graph but $\sum_{i \in V} a_i = 2a$ for an undirected graph.

1.2 Cycles

For a **directed** graph, a **cycle** is

> a non-empty path from a node to itself.

For the graph in Figure 1(right), the path X, Y, Z, X is a cycle.

For an **undirected** graph, the definition is somewhat more involved as we would certainly not want any edge between u and w to count as a cycle even though u, w, u is a non-empty path. Rather, we define a **cycle** as

> a non-empty path from a node to itself where no edge occurs more than once.

For the graph in Figure 1(left), the path C, D, B, C is a cycle. Observe that a cycle in an undirected graph must contain at least 3 different nodes.

1.3 Connectivity

For an **undirected** graph, we say that nodes u and w are **connected** if there is a path from u to w (which is equivalent to saying that there is a path from w to u). Given a graph $G = (V, E)$, with $V' \subseteq V$ we say that V' is connected if u and w are connected for all

$u, w \in V'$; we say that G is connected if V is connected. Thus the graph in Figure 1(left) is connected.

For a **directed** graphs, there are several notions of being "connected". The most common is **strongly connected**:

> u, w are strongly connected if there is a path from u to w, and also a path from w to u.

Given a graph $G = (V, E)$, with $V' \subseteq V$ we say that V' is strongly connected if u and w are strongly connected for all $u, w \in V'$; we say that G is strongly connected if V is strongly connected.

For other notions, we define:

- u and w are **unilaterally connected** if there exists a path from u to w *or* a path from w to u
- u and w are **weakly connected** if u and w are connected in the corresponding *un*directed graph (where there is an edge between two nodes iff the directed graph has an edge from one to the other).

Clearly, "strongly connected" implies "unilaterally connected" which implies "weakly connected". For the graph in Figure 1 (right), X and Z are strongly connected, X and W are unilaterally but not strongly connected, and X and U are weakly but not unilaterally connected.