Graph Theory:
Basic Notions

Vincenzo Bonifaci

February 24, 2017

1 Graphs

Definition 1.1. A *directed graph* $G(V, E)$, or *digraph*, is given by a nonempty set of *nodes* V and a set of *arcs* (or *edges*) $E \subseteq V \times V$.

Notice that this allows *loops* (arcs (i, i) with $i \in V$) and that $(i, j) \neq (j, i)$. The maximum number of arcs of a directed graph is n^2 where $n = |V|$.

Definition 1.2. An (undirected) *graph* $G(V, E)$ is given by a nonempty set of *vertices* (or nodes) V and a set of *edges* $E \subseteq \binom{V}{2}$ where $\binom{V}{2} = \{\{i, j\} : i \in V, j \in V, i \neq j\}$.

Notice that this does not allow *loops* and that $\{i, j\} = \{j, i\}$. The maximum number of edges of a graph is $|\binom{V}{2}| = \binom{n}{2} = n(n-1)/2$ where $n = |V|$.

If $e = \{a, b\} \in E$ then a and b are *adjacent* or *neighbors* in G. The edge e is *incident* to a and b; a and b are the *endpoints* of e. Two different edges e,e' are *incident* if they share an endpoint.

For a directed graph, if $e = (a, b) \in E$ then a is the *tail* of e and b is the *head* of e.

Example 1.3. The *complete graph* (*clique*) K_n on n vertices has $|V| = n$ and $E = \binom{V}{2}$. It has $n(n-1)/2$ edges. The graph K_3 is also called a *triangle*.

Definition 1.4. A graph is *bipartite* if there exist $V_1, V_2 \subseteq V$ such that $V_1 \cup V_2 = V$, $V_1 \cap V_2 = \emptyset$, and no edge in E has both endpoints in V_1 or both endpoints in V_2.

Example 1.5. The *complete bipartite graph* (*bipartite clique*) $K_{p,q}$ has $V = V_1 \cup V_2$, $V_1 \cap V_2 = \emptyset$, $|V_1| = p$, $|V_2| = q$, $E = \{\{i, j\} : i \in V_1, j \in V_2\}$. It has pq edges.

The *degree* of a vertex v in G, denoted $\deg_G(v)$ or simply $\deg(v)$, is the number of edges incident to v. For a directed graph we have two quantities, the *out-degree* $\deg^+(v)$ and *in-degree* $\deg^-(v)$, which count the arcs leaving v and entering v, respectively.

Lemma 1.1 (Handshaking lemma). $\sum_{v \in V} \deg(v) = 2|E|$.

Proof. In the sum, every edge is counted exactly twice. $\square$

Very often, the letters n and m are used to denote the number of nodes and number of arcs of G, respectively. Any reasonable description of the graph should not use more than $O(n+m)$ words of memory (assuming that each node identifier fits in a single word, that is, the number of bits in a word is more than $\log_2 n$).

Many interesting graphs are *sparse*, that is, they satisfy $m \ll n^2$. So it may be possible to represent a sparse graph with much fewer than $O(n^2)$ words.

2 Subgraphs

By "removing" nodes and/or edges from a graph we obtain a *subgraph*. If we only remove nodes (and edges incident to them, but nothing else) we obtain an *induced subgraph*. If we only remove edges, we obtain a *spanning subgraph*. The formal definitions follow.

Definition 2.1. Let $G(V, E)$ be a graph. For $V' \subseteq V$, let $E|V' = \{\{a, b\} \in E : a \in V', b \in V'\}$.

- A graph of the form $G'(V', E|V')$ is an *induced subgraph* of G (it is induced by V').
- A graph of the form $G'(V', E')$ with $V' \subseteq V$, $E' \subseteq E|V'$ is a *subgraph* of G.
- A graph of the form $G'(V, E')$ with $E' \subseteq E$ is a *spanning subgraph* of G.

3 Walks, paths and cycles

A *walk* on graph $G(V, E)$ is a sequence $x_1, e_1, x_2, \ldots, x_{p-1}, e_{p-1}, x_p$ (for some $p \geq 1$) with $x_i \in V$ for all $1 \leq i \leq p$, $e_i = \{x_i, x_{i+1}\} \in E$ for all $1 \leq i \leq p-1$.

- The *endpoints* of the walk are x_1 and x_p.
- The *length* of the walk is $p-1$.
- The walk is *closed* if $x_p = x_1$.

A *path* is a walk for which the e_i are distinct and the x_i are distinct.

A *cycle* is a closed walk for which the e_i are distinct and all the x_i are distinct except for $x_p = x_1$.

The definitions can be easily extended to directed graphs by requiring that the direction of the walk is consistent with the direction of the arcs traversed by it.

A graph is *acyclic* if it contains no cycle; otherwise it is *cyclic*.

Whether a directed graph is acyclic or not can be determined in time $O(n+m)$, by running a complete *postorder* depth-first visit of the graph and marking nodes with the "time" they are visited. Say that node u gets the index t_u. If there is an arc (u, v) with $t_u < t_v$, then the graph has a cycle. Otherwise, the timestamps give a *topological order* of the nodes: a mapping $t : V \to \{1, \ldots, n\}$ such that, if $t_u < t_v$, then there is no arc from u to v. Topological orders are certificates of acyclicity and can be used to speed up algorithms for directed acyclic graphs.

Exercise 3.1. Show that if G has a topological order, then it is acyclic.

4 Connectivity and distances

Node $u \in V$ is connected to node $v \in V$ (equivalently, v is *reachable* from u) if there is a path from u to v.

A (directed) graph is *(strongly) connected* if for every $u, v \in V$, u is connected to v.

Note that in any *connected* graph, $m \geq n-1$ (why?) and so, for example, $O(n+m)$ can be shortened to $O(m)$ for connected graphs.

Whether a graph is connected or not can be determined in time $O(n+m)$, by performing a visit of the graph from an arbitrary node: if not all nodes are visited, the graph must be disconnected. For directed graphs, determining strong connectivity also costs $O(n+m)$: from an arbitrary starting node, we perform both a "forward" visit (on the original graph) and a "backward" visit of the "reverse" graph, obtained by replacing each arc (u, v) by (v, u). The original graph is strongly connected if and only if all the nodes are touched by both visits.

A *connected component* of G is an induced subgraph that is connected and *maximal*, that is, not properly contained in any connected induced subgraph of G.

The *out-component* of node u in a digraph is the set of nodes reachable from u (including u). Similarly, the *in-component* of node v is the set of nodes from which v can be reached (including v).

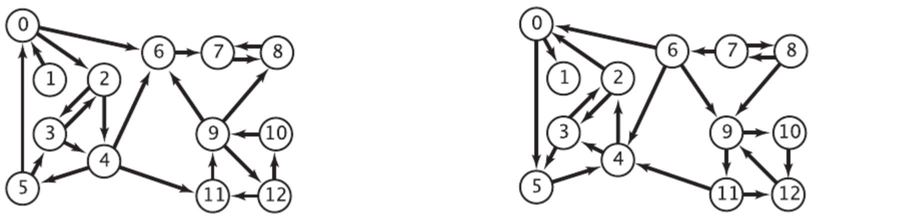
The strongly connected component of node u is the intersection of the out-component of u and the in-component of u (why?).

The connected components of an undirected graph can be determined in linear time by an exhaustive visit of the graph. For directed graphs, there is also a linear time algorithm to determine the strongly connected components, but it is slightly more complicated and requires two distinct visits (Kosaraju's algorithm).

Kosaraju's algorithm for finding strong components in digraphs

1. Given G, construct its reverse graph G^R (each arc (u, v) in G becomes (v, u) in G^R).
2. Construct the node ordering σ given by a *reverse post-order* depth-first search of G^R.
3. Use the ordering σ to perform a complete depth-first search of G. Each set of nodes visited in an outer DFS call is a strongly connected component of G.

Example 4.1. Consider the graph in the right part of the following figure.



The reversed graph is depicted on the left. The post-order sequence of nodes, after being reversed, is 1 0 2 3 4 11 9 12 10 6 7 8 5. The third step of the algorithm yields the strongly connected components: $\{1\}$, $\{0, 5, 4, 3, 2\}$, $\{11, 12, 9, 10\}$, $\{6\}$, $\{7, 8\}$.

A *tree* is an undirected graph that is connected and acyclic. A tree on n nodes has $n-1$ edges.

A *forest* is an undirected graph that is acyclic. Each connected component of a forest is a tree.

If u is connected to v, a *shortest path from u to v* is a path from u to v of minimum length.

The distance $d(u, v)$ from u to v is the length of a shortest path from u to v.

The distance from u to all other nodes of the graph can be computed in linear time with a single breadth-first search from u.

The *diameter* of a graph is

$$D = \max_{u,v \in V \colon u \text{ is connected to } v} d(u, v).$$

Determining exactly the diameter of a graph can be costly. After ensuring that the graph is connected, we can, by using n breadth-first searches (one from every node) determine all distances between pairs of nodes (and therefore, the diameter) in $O(mn)$ time. However, if we only need a rough estimate of the diameter, we can run a breadth-first search visit from an arbitrary node u: if the highest distance from u to any other node is B, then $B \leq D \leq 2B$. Therefore, approximating the diameter within a factor of two costs $O(n + m)$ time.

Exercise 4.1. Prove that in the special case when G is a tree, the diameter can be computed in linear time.

Exercise 4.2 (Open research problem!)**.** Find an efficient algorithm that computes a value B such that $B \leq D \leq 1.1B$. The algorithm should be asymptotically faster than $O(mn)$.