CMSC 351 - Introduction to Algorithms
Spring 2012
Lecture 13

Instructor: MohammadTaghi Hajiaghayi
Scribe: Rajesh Chitnis

1 Introduction

In this lecture we give a brief introduction to Graph Theory.

2 Preliminaries

A graph G with n vertices (nodes) and m edges (arcs) consists of a vertex set $V(G) = \{v_1, v_2, \ldots, v_n\}$ and an edge set $E(G) = \{e_1, e_2, \ldots, e_m\}$, where each edge is an unordered pair of vertices. We write uv or $\{u, v\}$ for an edge between u and v. If $uv \in E(G)$, then u and v are adjacent. The vertices contained in an edge are called its endpoints.

We can visualize a graph on paper by assigning a point to each vertex and drawing a curve for each edge between the points representing its endpoint. The vertices can be any objects like cities in a country, points in the plane, etc. and edges are showing relation between the two objects. The graphs can model lots of problems. Graphs can also model people and their friendship relations. In this course we will denote a graph by $G = (V, E)$.

A directed graph or digraph is a graph whose edges are ordered pairs. The edges are represented as (u, v) which means that there is an edge from u to v. Sometimes we consider more general models that allow repeated edges or edges with both endpoints same (loops). Such graphs are called multi-graphs. In this course we usually consider only simple graphs. Note that simple graphs have at most $\binom{n}{2} = \theta(n^2)$ edges.

Let $G = (V, E)$ be an undirected graph. An induced subgraph of G is a graph $H = (V', E')$ such that $V' \subseteq V$ and E' includes all edges in E both of whose incident vertices are in V'. If we just have $E' \subseteq E$ then it is called as subgraph of G.

A degree of a vertex v is represented by $d_v(G)$ or $deg_v(G)$ and is the number of vertices adjacent to that vertex.

Theorem 1 *Let* $G = (V, E)$ *be an undirected graph. Then* $\sum_{v \in V} deg_v(G) = 2|E|$

Proof: The equation follows by double counting. ■

3 Representation of Graphs for Algorithms

There are two ways to represent graphs: as a collection of adjacency lists or as an adjacency matrix. Adjacency list is preferred for sparse graphs, i.e., graphs having small number of edges like $O(n)$. Adjacency matrix is preferred for dense graphs, i.e., graphs having large number of edges like $\Omega(n^2)$.

The adjacency-list representation of a graph $G = (V, E)$ consists of an array Adj of $|V|$ lists, one for each vertex of V. For each $u \in V$, the adjacency list $Adj[u]$ contains pointers to all vertices v such that there is an edge $uv \in E$ (the elements of the list can be in any arbitrary order or in a sorted order). Irrespective of whether the graph is directed or undirected, the adjacency-list representation has the memory amount $O(|V| + |E|)$. The disadvantage is that determining if an edge uv is present in the graph or not can take $O(\deg_v(G))$ time.

For the adjacency matrix representation of a graph $G = (V, E)$, we assume that the vertices are numbered $1, 2, \ldots, |V|$ in some arbitrary manner. The adjacency matrix is a $|V| \times |V|$ array $A = (a_{ij})$ such that

$$a_{ij} = \begin{cases} 1 & \text{if } ij \in E \\ 0 & \text{otherwise} \end{cases}$$

The adjacency matrix of a graph requires $\theta(|V|^2)$ space independent of the number of edges in the graph. However checking if $uv \in E$ is in $\theta(1)$.

Our graphs can be weighted also (for which each edge has an associated weight) and can be easily represented by both methods described above. A **complete graph** or **clique** is a simple graph in which every pair of vertices forms an edge and thus $|E| = \binom{n}{2}$. An independent set in a graph is a vertex subset $S \subseteq V(G)$ such that the induced subgraph $G[S]$ has no edges. The **complement** of a simple graph, written as $\overline{G}$, is a graph with same vertex set as G such that u, v are adjacent in G if and only if u, v are not adjacent in $\overline{G}$.

Example: Job Assignments and Bipartite Graphs:
Suppose we have m jobs and n people and each person can do some of the jobs. Can we make assignments to fill the jobs? We model the available assignments by a graph having a vertex for each job and each person, putting job j adjacent to person p if p can perform job j. If each person can do only one job, then the assignment problem is the matching problem.

A graph is **bipartite** if its vertex set can be partitioned into at most two independent sets V_1, V_2 (like the person-job graph above). Then we represent it as $G(V_1, V_2, E)$. A **complete bipartite** graph is a bipartite graph in which the edge set consists of all pairs having a vertex from V_1 and V_2. For bipartite graphs, we can have a simple adjacency matrix as follows: We have a matrix (array) of size $|V_1| \times |V_2|$, namely $A = (a_{ij})$ such that

$$a_{ij} = \begin{cases} 1 & \text{if } ij \in E \\ 0 & \text{otherwise} \end{cases}$$

Note that this array is not symmetric anymore.

A **walk** of length k is a sequence $v_0, v_1, \ldots, v_k$ of vertices such that $e_i = \{v_{i-1}, v_i\} \in E$. A **path** is a walk with no repeated vertex. A **cycle** is a walk such that the first and the last vertex are the same, i.e., $v_0 = v_k$. A cycle is **simple** if there is no repeated vertex.

Theorem 2 *If there is a walk between u and v in G, then there is a path between them as well.*

An undirected graph is **connected** if every pair of vertices is connected by a path. The **connected components** of a graph are the equivalence classes of vertices under the "connected" relation.

A **tree** is a connected undirected graph with no cycles. A **DAG** (directed acyclic graph) is a directed graph with no cycles. The following three statements are equivalent:

1. G is a tree.
2. G is connected and $|E| = |V| - 1$.
3. G has no cycle and $|E| = |V| - 1$.

Theorem 3 *If G is a tree then $|E| = |V| - 1$.*

Proof: First note that there is a vertex which is a leaf (a vertex of degree one) since there is no cycle. Then proof follows by induction on $|V|$. ■

A **rooted tree** is a tree in which one of the vertices is distinguished from the others, called the root. We often call vertices of a rooted tree (in general for directed graphs) as **nodes**. If the last edge on the path from the root r of a tree T to a node x is (y, x), then y is the **parent** of x and x is the **child** of y. A node with no children is called a **leaf**. A node which is not a leaf is called as an **internal** node. Consider a node x in a rooted tree T with root r. Any node y on the unique path from r to x is called as an **ancestor** of x. If y is an ancestor of x, then x is a **descendant** of y.

Rooted trees are often used to show hierarchial (data) structures. Because of this we can make the tree directed towards the leaves or at least put the root on top of the tree. Then the root is connected to other nodes, which are at the level one of the hierarchy; they in turn are connected to other nodes at level two and so on. The number of children of a node x in a rooted tree is called the **out-degree** of x. The length of the path from the root r to a node x is called as the **depth** of x in T. The largest depth of any node in T is called as the **height** of T.

Rooted trees of out-degree 2 (every vertex has out-degree at most 2) are called **binary trees**. In this case we identify children by **left** (for first) and **right** (for second). The common representation of trees is the linked-list representation in which each node keeps pointers (as an array for binary trees; otherwise linked lists) to its children and also a pointer to its parent. See Figures 2 and 3 for examples of the tree given in Figure 1. We will see another implicit representation for binary trees in the next lecture.

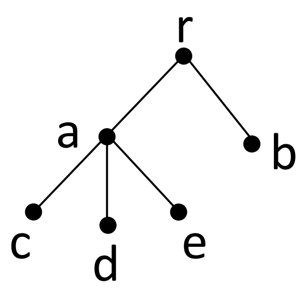

Figure 1: Tree T

References

[1] Udi Manber, *Introduction to Algorithms - A Creative Approach*

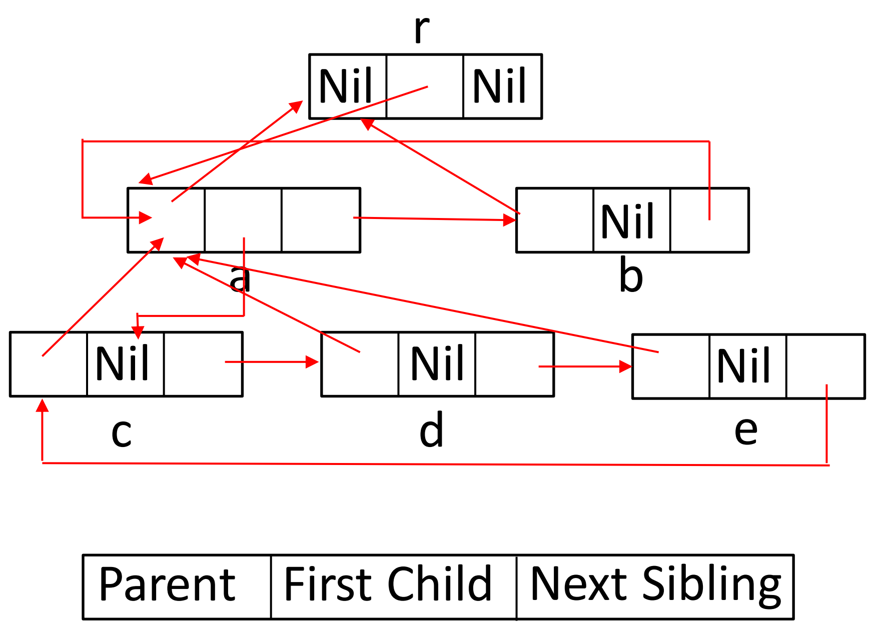

Figure 2: Representation of tree T when we do not know how many maximum children any vertex can have.

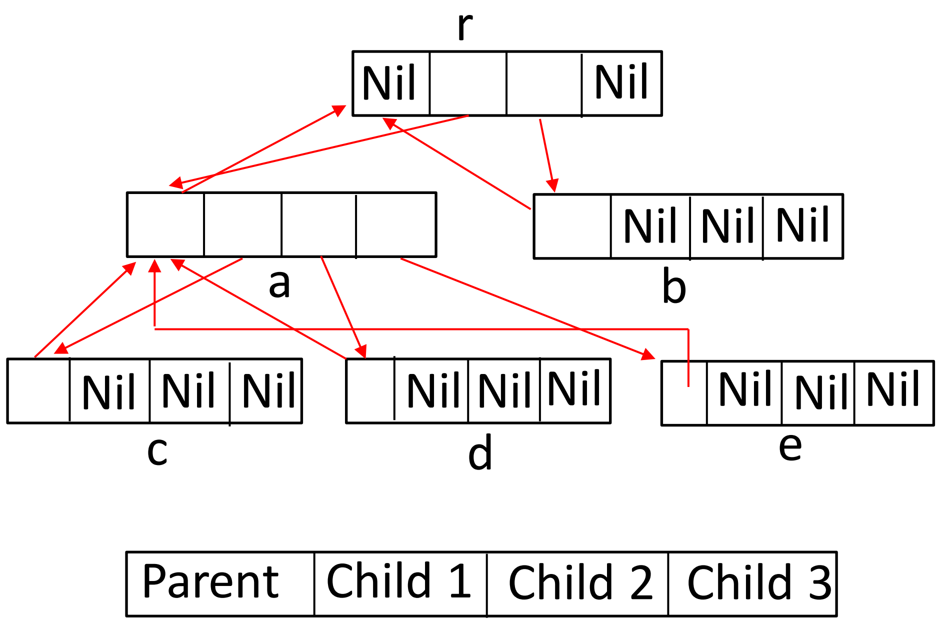

Figure 3: Representation of tree T when we know that any vertex can have at most three (or any fixed number) children.