GRAPHS

MSCI 240: Algorithms & Data Structures

lecture summary

lots of definitions

more definitions

graph data structures
- how to represent graphs in code

Topic	Building Java Programs	Algorithms (Sedgewick)
classes, ADTs	chapter 8	1.2
arrays	chapter 7	
ArrayList<T>	chapter 10	1.3
Stack/Queue	chapter 14, (11)	1.3
LinkedList	chapter 16	1.3
Complexity	chapter 13	1.4
Searching		pp. 46-47
Sorting		chapter 2.1-2.3
Recursion	chapter 12	1.1 (p. 25)
Binary Trees	chapter 17	chapter 3.1-3.2
Dictionaries	chapter 18.1	chapter 3.4
Graphs	N/A (Wikipedia good)	chapter 4.1
Heaps/Priority Queues	chapter 18.2	chapter 2.4

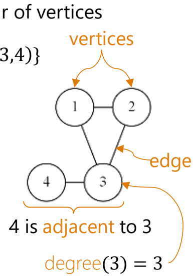
graph definitions

a graph is a set of vertices (a.k.a. nodes), V, and a set of edges, E, where each edge connects a pair of vertices

e.g., $V = \{1,2,3,4\}, E = \{(1,2),(1,3),(2,3),(3,4)\}$

adjacent: u is adjacent to $v \Leftrightarrow (u,v) \in E$

degree (of a vertex $v \in V$): the number of edges connected to a vertex

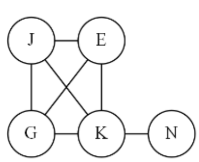


e.g. (social),

$V = \{\text{Jerry}, \text{Elaine}, \text{George}, \text{Kramer}, \text{Newman}\}$

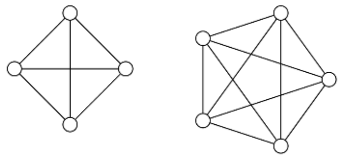
adjacent nodes?

degree of J,E,G,K,N?



complete graph: a graph where each vertex is connected to every other vertex

e.g., complete graphs of size 4 and 5



directed graph: each edge in E is an ordered pair

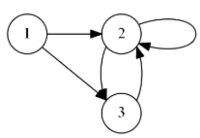
has a direction

in-degree (of $v \in V$): number of edges entering a vertex

in-degree of 1,2,3?

out-degree (of $v \in V$): number of edges leaving a vertex

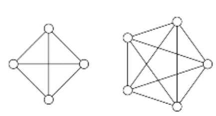
out-degree of 1,2,3?



how many edges are in a complete (undirected) graph?

let $n = |V|$ (number of vertices)

let $m = |E|$ (number of edges)



$$m = (n-1) + \cdots + 2 + 1 = \sum_{i=1}^{n-1} i = \frac{n \cdot (n-1)}{2} \in O(n^2)$$

alternatively, $m = \binom{n}{2} = \frac{n!}{2!\,(n-2)!} = \frac{n \cdot (n-1)}{2} \in O(n^2)$

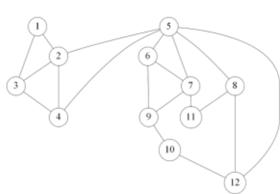
more definitions

dense graph
when m is in $\Theta(n^2)$ (e.g., almost all the edges)

sparse graph
when $m \ll n^2$ (e.g., only 1-2 edges per vertex, max, $\therefore m \in O(n)$)
($\ll$ means way less than)

path: sequence of vertices connected by edges
e.g., [1,2,5,7,11,8,12] is a path from 1 to 12

length of path: number of edges along the path
e.g., [1,2,5,7,11,8,12] has length 6



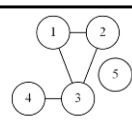
graph data structures

there are two primary ways to represent graphs:

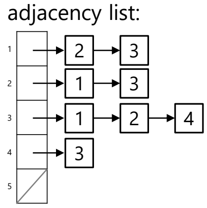
adjacency list
good for sparse graphs

adjacency matrix
good for dense graphs

$V = \{1,2,3,4,5\}$
$E = \{(1,2),(1,3),(2,3),(3,4)\}$



adjacency list:



adjacency matrix:

$$\begin{array}{c c} & \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

IntGraphList class (undirected)

```
public class IntGraphList {
    private HashMap<Integer, LinkedList<Integer>> adjacencyList;

    public IntGraphList() {
        adjacencyList = new HashMap<>();
    }

    // precondition: value has not been added before
    public void addVertex(int value) {
        adjacencyList.put(value, new LinkedList<>());
    }

    // precondition: first and second must already be added as vertices
    public void addEdge(int first, int second) {
        adjacencyList.get(first).add(second);
        adjacencyList.get(second).add(first);
    }
}
```


IntGraphMatrix class (undirected)

```
public class IntGraphMatrix {
    private boolean[][] adjMatrix;
    private int numVertices;

    public IntGraphMatrix(int numVertices) {
        adjMatrix = new boolean[numVertices][numVertices];
        this.numVertices = numVertices;
    }

    // no addVertex method (fixed number of vertices, numbered 1 to n)

    // precondition: 0 <= first,second < numVertices
    public void addEdge(int first, int second) {
        adjMatrix[first][second] = true;
        adjMatrix[second][first] = true;
    }
}
```


how much space does an adjacency list use?

how much space does an adjacency matrix use?

sparse vs. dense graphs?

graph summary

graphs have vertices and edges (pairs of vertices)
- definitions: adjacent, degree (in/out), complete, sparse, dense, path

graphs can be undirected or directed

graphs can be stored with an adjacency list (better for sparse) or an adjacency matrix (better for dense)

next:
- breadth-first search