Sri SAI RAM ENGINEERING COLLEGE
INSTITUTE OF TECHNOLOGY
West Tambaram, Chennai - 44

Sairam INSTITUTIONS

SAIRAM DIGITAL RESOURCES

YEAR II SEM III

UNIT NO 4

NON LINEAR DATA STRUCTURES –GRAPHS

CS8391

DATA STRUCTURES
(Common to CSE & IT)

Introduction Definition - Representation of Graph

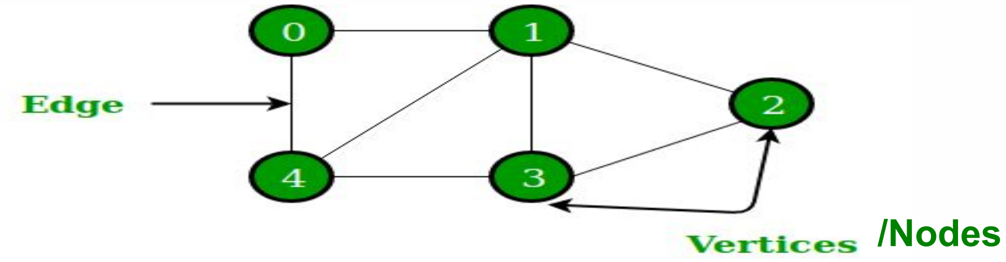
COMPUTER SCIENCE & ENGINEERING

4 QUALITY EDUCATION

GRAPHS – Introduction

INTRODUCTION

- **Graph : Mathematics** – Graph, defined as a pictorial representation or a diagram that represents data or values in an organized manner. The points on the graph often represent the relationship between two or more things. Visual representations help us to understand data quickly.
- **Graph : Data structure** - A Graph is a non-linear data structure consisting of nodes and edges. The nodes are sometimes also referred to as vertices and the edges are lines or arcs that connect any two nodes in the graph. More formally a Graph can be defined as, A Graph consists of a finite set of **vertices(or nodes)** and set of **Edges** which connect a pair of nodes. In the below Graph, the set of vertices V = {0,1,2,3,4} and the set of edges E = {01, 12, 23, 34, 04, 14, 13}.



GRAPHS – Introduction

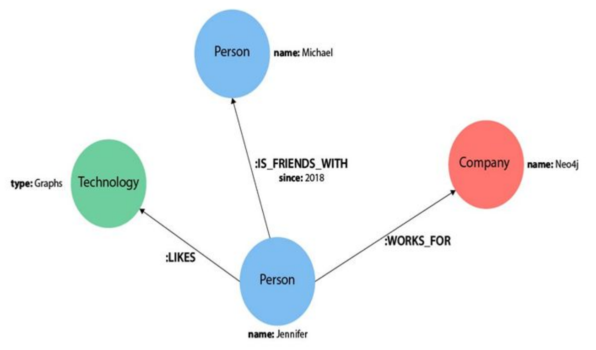
- Graphs are used to solve many real-life problems.
- Graphs are used to represent networks.
- The networks may include
  paths in a city or
  telephone network or
  circuit network.
- Graphs are also used in social networks like LinkedIn, Facebook.
  For example, in Facebook,
  each person is represented with a vertex(or node)
  each node is a structure and contains information like person id, name, gender, locale etc.

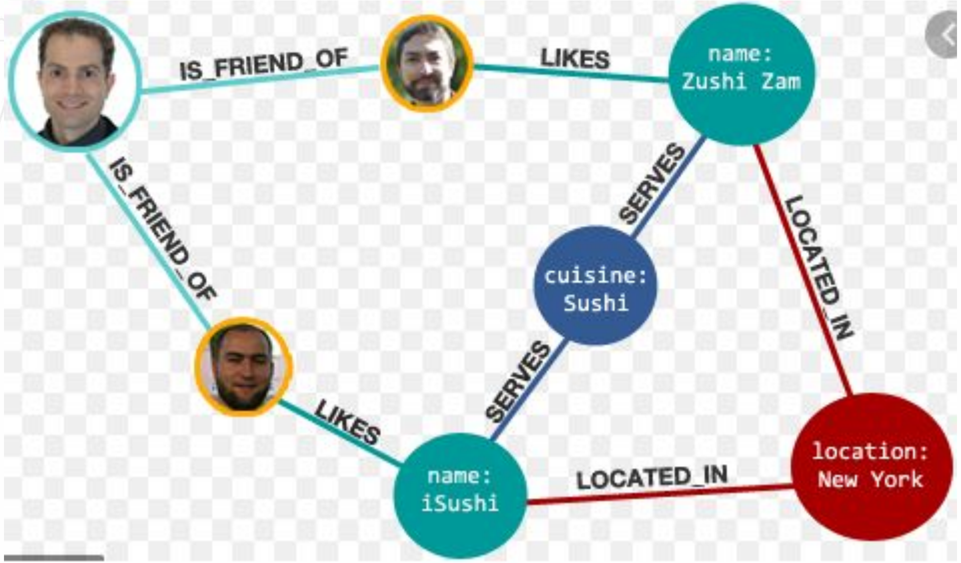
GRAPHS – Introduction

For example, in Facebook,

each person is represented with a vertex(or node)

each node is a structure and contains information like person id, name, gender, locale etc.

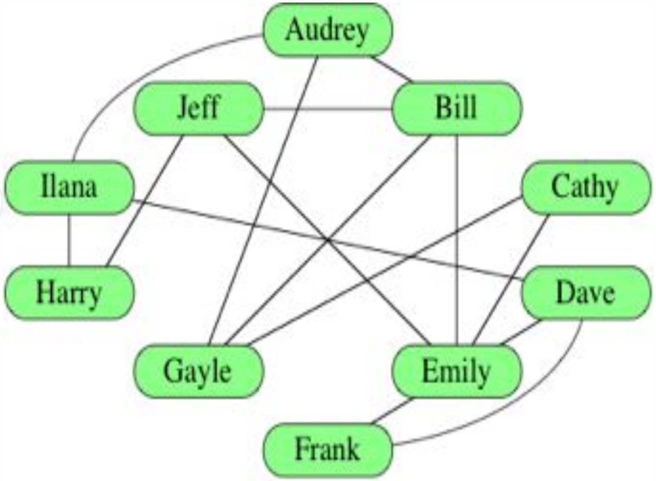




GRAPHS – Introduction

Social Network as Graph (Undirected Graph)

- **Vertices** – Name of the person
- **Edges** – Connects the person who know each other (relationship both ways)
- The edges are denoted by connecting vertices *u* and *v* by the pair (u,v)
- In an undirected edge (u,v) is the same as (v,u)



Graph – representing social network

GRAPHS – Introduction

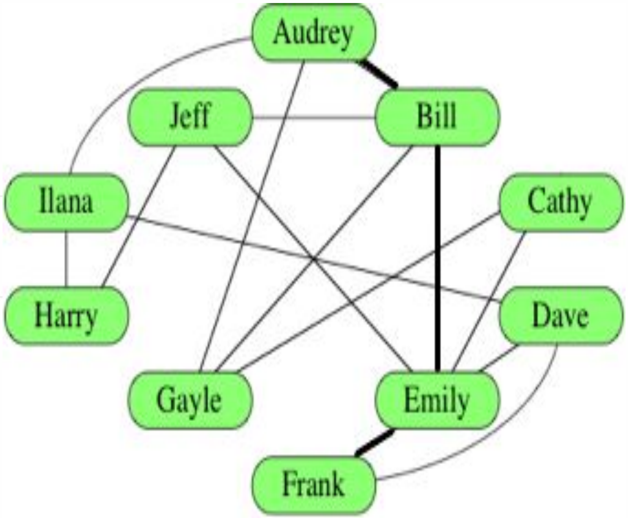
Social Network as Graph (Directed Graph- Relation both ways)

- Audrey and Frank do not know each other.

Suppose that Frank wanted to be introduced to Audrey. How could he get an introduction?

Frank knows Emily, who knows Bill, who knows Audrey. i.e. there is a **path** of three edges between Frank and Audrey. In fact, that is the most direct way for Frank to meet Audrey; there is no path between them with fewer than three edges.

The path between two vertices with the fewest edges is a **shortest path**.



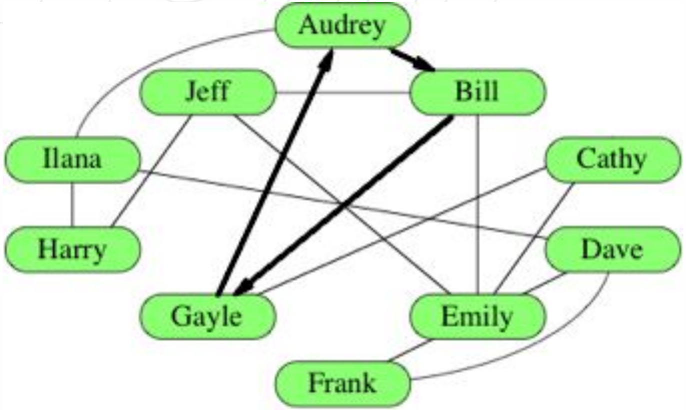
Graph – representing social network

GRAPHS – Introduction

Social Network as Graph (Cycle)

When a path goes from a particular vertex back to itself, that's a cycle. The social network contains many cycles; one of them goes from Audrey to Bill to Emily to Jeff to Harry to Ilana and back to Audrey. There's a shorter cycle containing Audrey, shown below: Audrey to Bill to Gayle and back to Audrey.

What other cycles can you find?

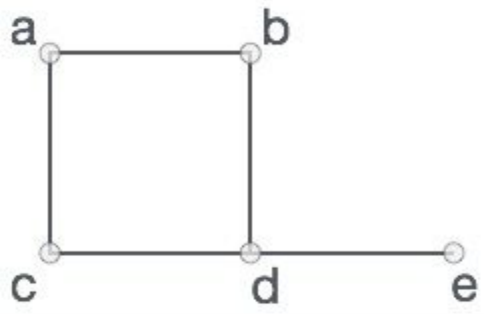


GRAPHS – Definition

- A graph is a pictorial representation of a set of objects where some pairs of objects are connected by links. The interconnected objects are represented by points termed as **vertices**, and the links that connect the vertices are called **edges**.
- Formally, a graph is a pair of sets **(V, E)**, where **V** is the set of vertices and **E** is the set of edges, connecting the pairs of vertices. Take a look at the following graph −
- In the graph,

  V = {a, b, c, d, e}

  E = {ab, ac, bd, cd, de}



GRAPH

GRAPHS – Terminologies

Graph Data Structure – Terminologies with definitions

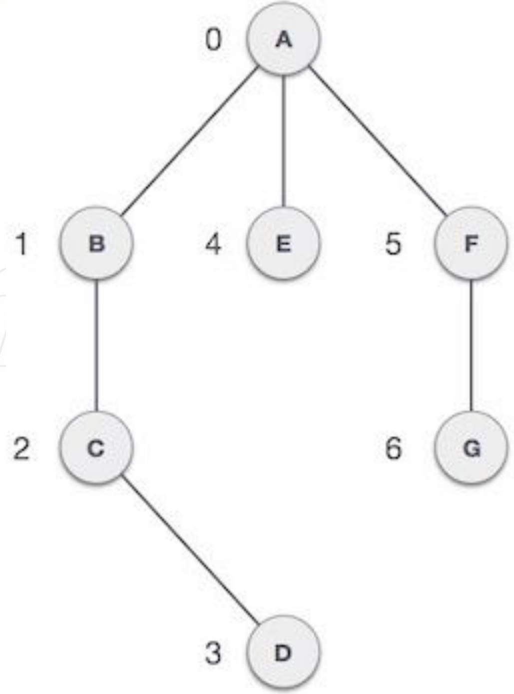
- Mathematical graphs can be represented in data structure. Graph can be represented as
  an array of vertices and
  a two-dimensional array of edges.
- **Terminologies** – **VERTEX, EDGE,ADJACENCY,PATH**

Vertex − Each node of the graph is represented as a vertex. In the following example, the labeled circle represents vertices. Thus, A to G are vertices. We can represent them using an array as shown in the following image. Here A can be identified by index 0. B can be identified using index 1 and so on.

GRAPHS – Terminologies

- **Edge** − Edge represents a path between two vertices or a line between two vertices. In the following example, the lines from A to B, B to C, and so on represents edges. A two-dimensional array is used to represent an array as shown in the following image.

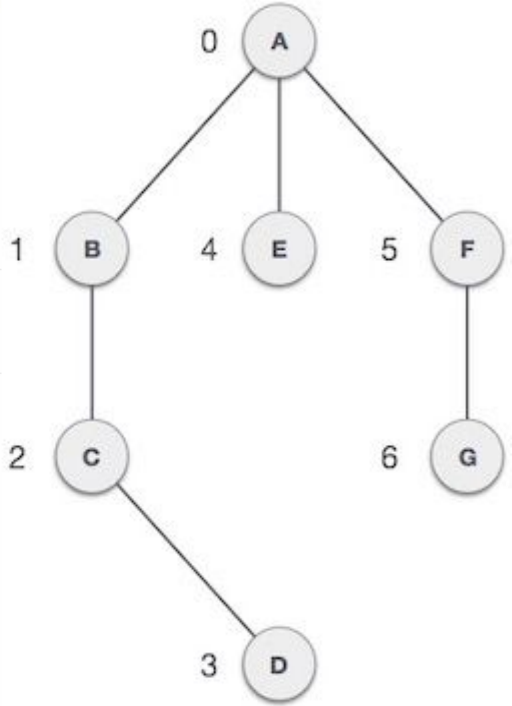
Here AB can be represented as 1 at row 0, column 1, BC as 1 at row 1, column 2 and so on, keeping other combinations as 0.



GRAPHS – Terminologies

- Adjacency − Two node or vertices are adjacent if they are connected to each other through an edge. In the following example, B is adjacent to A, C is adjacent to B, and so on.

- Path − Path represents a sequence of edges between the two vertices. In the following example, ABCD represents a path from A to D.



GRAPHS – Terminologies

Degree of Vertex

It is the number of vertices adjacent to a vertex V.

Notation − deg(V).

In a simple graph with n number of vertices, the degree of any vertices is −

$\deg(v) \leq n - 1 \ \forall \ v \in G$

- A vertex can form an edge with all other vertices except by itself. So the degree of a vertex will be up to the **number of vertices in the graph minus 1**. This 1 is for the self-vertex as it cannot form a loop by itself. If there is a loop at any of the vertices, then it is not a Simple Graph.

Degree of vertex can be considered under two cases of graphs −

- Undirected Graph
- Directed Graph

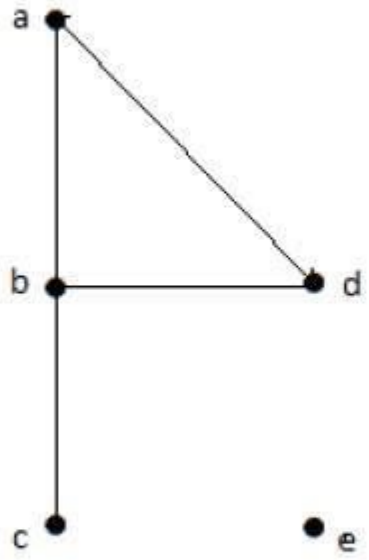
GRAPHS – Terminologies

Degree of Vertex in an Undirected Graph

An undirected graph has no directed edges. Consider the following examples.

Example 1

Take a look at the following graph −



In the Undirected Graph,

- deg(a) = 2, as there are 2 edges meeting at vertex ‘a’.
- deg(b) = 3, as there are 3 edges meeting at vertex ‘b’.
- deg(c) = 1, as there is 1 edge formed at vertex ‘c’
- So ‘c’ is a **pendent vertex**.
- deg(d) = 2, as there are 2 edges meeting at vertex ‘d’.
- deg(e) = 0, as there are 0 edges formed at vertex ‘e’.
- So ‘e’ is an **isolated vertex**.

GRAPHS – Terminologies

Degree of Vertex in a Directed Graph

In a directed graph, each vertex has an **indegree** and an **outdegree**.

- Indegree of a Graph

  Indegree of vertex V is the number of edges which are coming into the vertex V.

  Notation − deg−(V).

- Outdegree of a Graph

  Outdegree of vertex V is the number of edges which are going out from the vertex V.

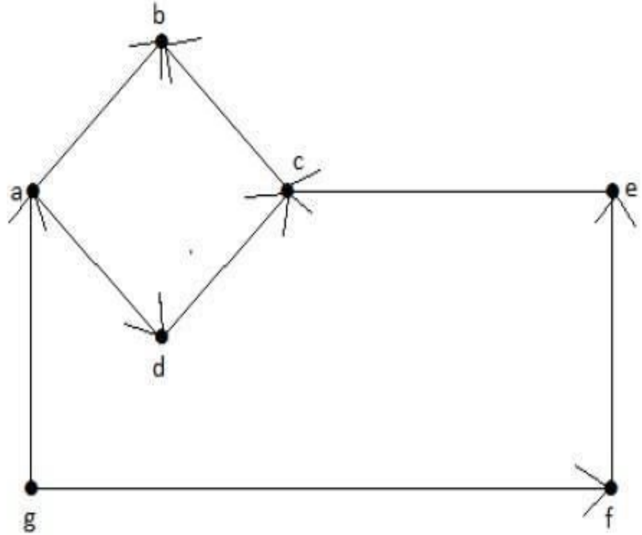
  Notation − deg+(V).

GRAPHS – Terminologies

Example 1

Take a look at the following directed graph. Vertex ‘a’ has two edges, ‘ad’ and ‘ab’, which are going outwards. Hence its outdegree is 2. Similarly, there is an edge ‘ga’, coming towards vertex ‘a’. Hence the indegree of ‘a’ is 1. The indegree and outdegree of other vertices are shown in the following table −

Vertex	In Degree	Out Degree
a	1	2
b	2	0
c	2	1
d	1	1
e	1	1
f	1	1
g	0	2



GRAPHS – operations

Basic Operations - Following are basic primary operations of a Graph −

- **Add Vertex** − Adds a vertex to the graph.
- **Add Edge** − Adds an edge between the two vertices of the graph.
- **Display Vertex** − Displays a vertex of the graph.

REPRESENTING GRAPHS

Representing a graph largely depends on the operations, intended to support.

For example the following operations can be done on the graph $G = (V, E)$:

(1) Map over the vertices $v \in V$.

(2) Map over the edges $(u, v) \in E$.

(3) Map over the neighbors of a vertex $v \in V$, or in a directed graph the in-neighbors or out-neighbors.

(4) Return the degree of a vertex $v \in V$.

(5) Determine if the edge (u, v) is in E.

(6) Insert or delete vertices.

(7) Insert or delete edges.

REPRESENTING GRAPHS

Traditionally, there are four standard representations, all of which assume that vertices are numbered from 1, 2, . . . , n (or 0, 1, . . . , n − 1).

Adjacency matrix. An n × n matrix of binary values in which location (i, j) is 1 if (i, j) ∈ E and 0 otherwise. Note that for an undirected graph the matrix is symmetric and 0 along the diagonal. For directed graphs the 1s can be in arbitrary positions.

The main problem with adjacency matrices is their space demand of Θ(n 2). Graphs are often sparse, with far fewer edges than Θ(n 2).

Example : Using an adjacency matrix, the following graph is represented as follows.

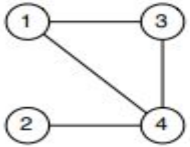


Figure An undirected graph.

$$\begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

REPRESENTING GRAPHS

Adjacency list. An array A of length n where each entry A[i] contains a pointer to a linked list of all the out-neighbors of vertex i. In an undirected graph with edge {u, v} the edge will appear in the adjacency list for both u and v.

Example : Using an adjacency matrix, the following graph is represented as follows.

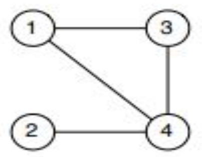
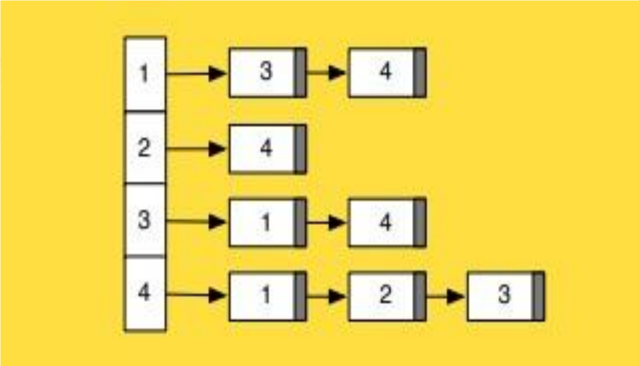


Figure An undirected graph.



REPRESENTING GRAPHS

Adjacency array. Similar to an adjacency list, an adjacency array keeps the neighbors of all vertices, one after another, in an array adj; and separately, keeps an array of indices that tell us where in the adj array to look for the neighbors of each vertex.

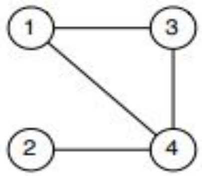
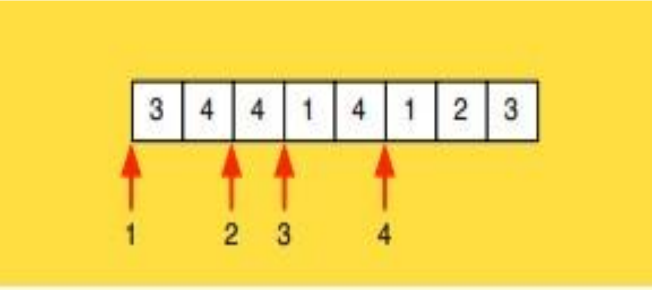


Figure An undirected graph.



REPRESENTING GRAPHS

Edge list. A list of pairs (i, j) $\in$ E.

Edge Sets. The simplest representation of a graph is based on its definition as a set of vertices V and a set of directed edges A $\subseteq$ V × V . If we use the set ADT, the keys for the edge set are simply pairs of vertices. The representation is similar to the edge list representation, but it abstracts away from the particular data structure used for the set—the set could be implemented as a list, an array, a tree, or a hash table.

Although edge sets are efficient for finding, inserting, or deleting an edge, they are not efficient if we want to identify the neighbors of a vertex v.

REPRESENTING GRAPHS

Adjacency Tables. To more efficiently access neighbors, use adjacency tables, which are a generalization of adjacency lists and adjacency arrays. The adjacency table representation is a table that maps every vertex to the set of its (out) neighbors. This is simply an **edge-set table.**

Adjacency Sequences. A special case of adjacency tables are adjacency sequences. Recall that a sequence is a table with a domain taken from {0, . . . , n − 1}. However the cost model sequences allow for faster random access, requiring only O(1) work to access the i th element rather than O(log n). This allows, for example, accessing a vertex at less cost. This is traded off for the fact that certain operations, such as sub selecting vertices, is more difficult. Because of the reduced cost of access, we will sometimes use a sequence of sequence of integers ((int seq) seq) to represent a graph.

APPLICATIONS OF GRAPHS

Since they are powerful abstractions, graphs can be very important in modeling data. In fact, many problems can be reduced to known graph problems. Here we outline just some of the many applications of graphs.

1. **Social network graphs:** to tweet or not to tweet. Graphs that represent who knows whom, who communicates with whom, who influences whom or other relationships in social structures. An example is the twitter graph of who follows whom. These can be used to determine how information flows, how topics become hot, how communities develop, or even who might be a good match for who, or is that whom.
2. **Transportation networks.** In road networks vertices are intersections and edges are the road segments between them, and for public transportation networks vertices are stops and edges are the links between them. Such networks are used by many map programs such as Google maps, Bing maps and now Apple IOS 6 maps (well perhaps without the public transport) to find the best routes between locations. They are also used for studying traffic patterns, traffic light timings, and many aspects of transportation.

APPLICATIONS OF GRAPHS

3. **Utility graphs.** The power grid, the Internet, and the water network are all examples of graphs where vertices represent connection points, and edges the wires or pipes between them.

Analyzing properties of these graphs is very important in understanding the reliability of such Utilities under failure or attack, or in minimizing the costs to build infrastructure that matches required demands.

4. Document link graphs. The best known example is the link graph of the web, where each web page is a vertex, and each hyperlink a directed edge. Link graphs are used, for example, to analyze relevance of web pages, the best sources of information, and good link sites.

5.Protein-protein interactions graphs. Vertices represent proteins and edges represent interactionsbetweenthemthatcarryoutsomebiologicalfunctioninthecell. Thesegraphs can be used, for example, to study molecular pathways—chains of molecular interactions inacellularprocess. Humanshaveover120Kproteinswithmillionsofinteractionsamong them.

APPLICATIONS OF GRAPHS

6. Network packet traffic graphs. Vertices are IP (Internet protocol) addresses and edges are the packets that flow between them. Such graphs are used for analyzing network security, studying the spread of worms, and tracking criminal or non-criminal activity.

7. Scene graphs. In graphics and computer games scene graphs represent the logical or spacial relationships between objects in a scene. Such graphs are very important in the computer games industry.

8. Finite element meshes. In engineering many simulations of physical systems, such as the flow of air over a car or airplane wing, the spread of earthquakes through the ground, or the structural vibrations of a building, involve partitioning space into discrete elements. The elements along with the connections between adjacent elements forms a graph that is called a finite element mesh.

9. Robot planning. Vertices represent states the robot can be in and the edges the possible transitions between the states. This requires approximating continuous motion as a sequence of discrete steps. Such graph plans are used, for example, in planning paths for autonomous vehicles.

APPLICATIONS OF GRAPHS

10. Neural networks. Vertices represent neurons and edges the synapses between them. Neural networks are used to understand how our brain works and how connections change when we learn. The human brain has about 1011 neurons and close to 1015 synapses.

11. Graphs in quantum field theory. Vertices represent states of a quantum system and the edges the transitions between them. The graphs can be used to analyze path integrals and summing these up generates a quantum amplitude (yes, I have no idea what that means).

12. Semantic networks. Vertices represent words or concepts and edges represent the relationships among the words or concepts. These have been used in various models of how humans organize their knowledge, and how machines might simulate such an organization.

13. Graphs in epidemiology. Vertices represent individuals and directed edges the transfer of an infectious disease from one individual to another. Analyzing such graphs has become an important component in understanding and controlling the spread of diseases.

APPLICATIONS OF GRAPHS

14. Graphs in compilers. Graphs are used extensively in compilers. They can be used for type inference, for so called data flow analysis, register allocation and many other purposes. They are also used in specialized compilers, such as query optimization in database languages.

15. Constraint graphs. Graphs are often used to represent constraints among items. For example the GSM network for cell phones consists of a collection of overlapping cells. Any pair of cells that overlap must operate at different frequencies. These constraints can be modeled as a graph where the cells are vertices and edges are placed between cells that overlap.

16. Dependence graphs. Graphs can be used to represent dependences or precedences among items. Such graphs are often used in large projects in laying out what components rely on other components and used to minimize the total time or cost to completion while abiding by the dependences.

QUERIES?

THANK YOU