Lab 10 - Graph Theory
Computer Science 1FC3

Author: Dai Tri Man Lê
ledt@mcmaster.ca

Abstract. Graphs, graphs' application in Computer Science and how to work with graphs in Maple.

1 Why Graphs?

Graphs are extremely important in Computer Science. There are several main reasons for that:

- Graphs can be used to represent many problems in computer science that are otherwise abstract. Finding a way to represent the solution to a problem as a graph can present new approaches to solving the problem or even lead directly to a solution derived from graph theory.
- Many problems computers are used to solve involve representing relationships between objects, places, or concepts. Because graphs can be either directed or undirected, they are a flexible method of showing connections. For instance, you can describe who knows whom in a room as a collection of vetices, each representing a person, and directed edges, each representing that one person knows another.

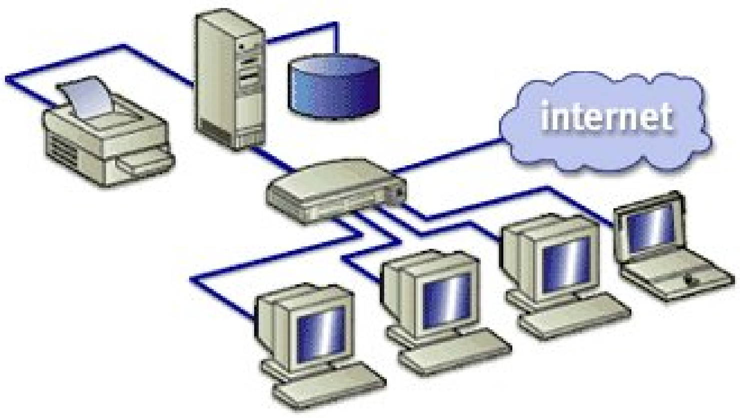
We shall now discuss different kinds of graphs and their applications in Computer Science.

2 Undirected Graphs

2.1 Simple Graph

Definition. A simple graph $G = (V, E)$ consists of V, a nonempty finite set of vertices and E, a set of unordered pairs of *distinct* elements of V called edges. **(End of Definition.)**

Simple graphs are often used to represent network models (models with links between elements) where the only information is the location of the connections. For example, the following diagram is a graph of a "real world" local area network (LAN).



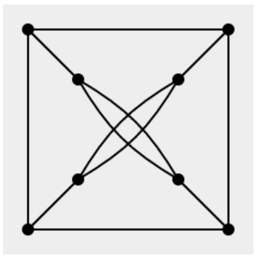
2.2 Multigraph and Pseudograph

Sometimes we want to model the situation when there more than one connection between some pairs of vertices, which is not allowed in simple graph. Hence, we need the concept of **multigraph**:

Definition. A multigraph $G = (V, E)$ consists of V, a nonempty finite set of vertices and E, a finite *multiset* set of unordered pairs of *distinct* elements of V called edges. Thus, multiple edges between two vertices are allowed in a multigraph.**(End of Definition.)**

Remark. a **multiset** (sometimes also called a bag) differs from a set in that each member has a multiplicity, which is a natural number indicating (loosely speaking) how many times it is a member, or perhaps how many memberships it has in the multiset. For example, in the multiset $\{a, a, b, b, b, c\}$, the multiplicities of the members a, b, and c are respectively 2, 3, and 1.**(End of Remark.)**

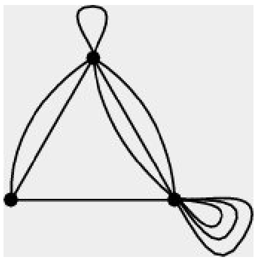
An example of a (nice!) multigraph is shown below:



Note that neither simple graphs nor multigraphs may have loops. All such graphs can only have edges between *distinct* pairs of vertices. In order to allow loops, we have need the notion of a **pseudograph** (multigraph with loops). A formal definition of pseudo graph is given below:

Definition. A pseudograph $G = (V, E)$ consists of V, a nonempty finite set of vertices and E, a finite *multiset* set of unordered pairs of elements of V called edges. Thus, multiple edges between two vertices and loops are allowed in a pseudo.**(End of Definition.)**

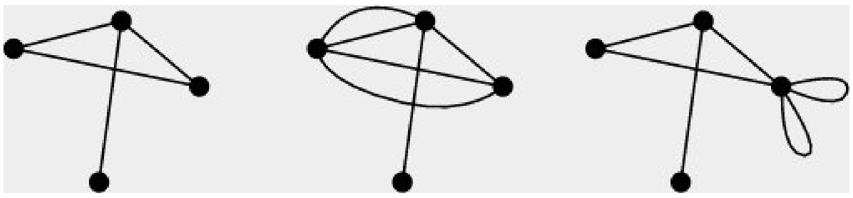
An example of a pseudo is shown below:



Remark. Graphs which are simple graphs, multigraphs and pseudographs are all called **undirected graph**, which means the direction of the edges does not matter. This is why each edge the previous definitions is precisely defined to be an *unordered pair* of vertices.
As we can see, as far as undirected graphs go, pseudographs are the most general type of graphs since they may contains loops and multiple edges. **(End of Remark.)**

2.3 Exercise 1:

1. Classify the following graphs:



2. Why in the definition of simple graph, when talking about the set of edges, we say "a set of unordered pairs" without mentioning the word "finite" as in the definition of multigraph and pseudograph? Is this a mistake?

2.4 Maple

Constructing and Visualizing Graphs

Maple has a large collection of commands related to graph theory, all found in the networks package. In order to access these commands we use the following command to include the network package:

```
with(networks);
```

Once we have loaded the package in this manner, we can use the newly defined Maple commands to create new graphs and then add or delete edges and vertices. To create a new graph called G1, we use the new command. This command to creates a new graph object and initializes all the underlying structures of the graph. The command only accepts one argument — the name of the new graph.

```
G1 := new():
```

This newly created graph does not yet have any vertices or edges. These must be added to the graph using such commands as addvertex, addedge. We continue to construct an example of a simple graph by adding vertices to represent cities and edges to represent the communications links of a computer network between the chosen cities. We use the addvertex command to add vertices and the addedge command to add edges and build up the graph.

For our example we have chosen a set of 4 places – T (Toronto), H (Hamilton), K (Kitchener), L (London). The following command adds this set of seven vertices to the newly created graph G1.

```
addvertex( T,H,K,L,G1);
```

The newly created vertex names are listed in response to the command.

To complete our graph representing a nation-wide computer network we next add the edges representing the connections between the various cities. Each connection is undirected. Information may flow in either direction. Undirected connections are defined by using a set of two vertices – the two cities joined by the connection. To add such edges to our graph we use the addedge command in a similar manner to the use of the addvertex command. This time the first argument to the procedure call is the set of vertex pairs representing the new edges. The last argument is the name of the graph to which the edges are to be added.

```
addedge([{T,H},{H,K},{K,L},{L,T},{H,L}],G1);
```

As each edge is created, a name is generated. The edge names are displayed as the value of this operation. The names are used so that, if necessary, more than one connection can occur between cities.

To discover the vertex pair associated with a particular edge in G1, use the ends command, as in

```
ends(e1,G1);
```

To find the set of all pairs corresponding to edges of a particular graph use the command

```
ends(G1);
```

To find the set of all edges in G1 use the command

```
edges(G1);
```

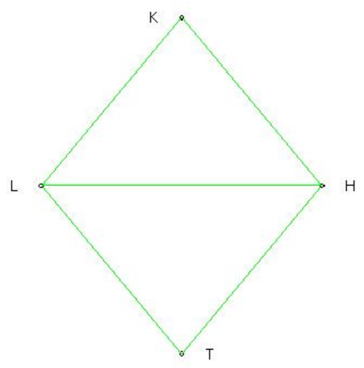
To find the set of all vertices in G1 use the command

```
vertices(G1);
```

To draw G1 use the command

```
draw(G1);
```

You should get a picture like this:



To delete edge(s) from a graph use the command

```
delete({e1},G1):
```

To delete a vertex or vetices from a Graph use the command

```
delete({T},G1):
```

Adjacency Matrices

Graphs can also be represented in the form of matrices which are called **adjacency matrices**. The major advantage of matrix representation is that the calculation of paths and cycles can easily be performed using well known operations of matrices. However, the disadvantage is that this form of representation takes away from the visual aspect of graphs. It would be difficult to illustrate in a matrix, properties that are easily illustrated graphically.

In Maple, we can find the adjacency matrix of a graph using command **adjacency**. For example, the following command will construct the adjacency matrix of G1 is our example:

```
adjacency(G1);
```

You should get a matrix like this:

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

This is a symmetric matrix! Do you know why?

2.5 Exercise 2:

1. Use the command addvertex, addedges to recover the original G1 graph. After that try to use this command:

   ```
   addedge([{T,H},{H,K},{K,L}],G1): adjacency(G1);
   ```

   Is this adjacency matrix different from the one from the original G1? Why or why not?

2. Use the following command to learn more about Maple's **networks** package:

   ```
   ?networks
   ```

3 Directed Graph and Directed Multigraph

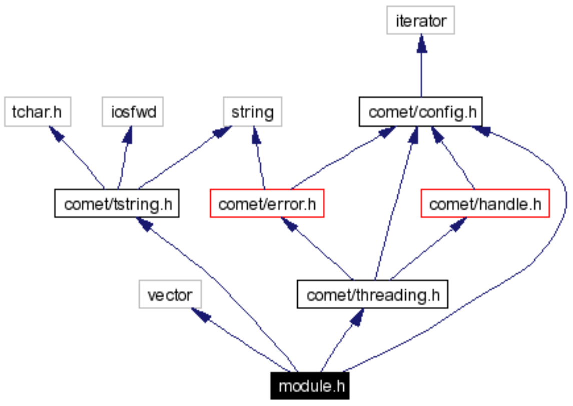
3.1 Directed Graph

Directed graphs are used when the direction of the connections is important. For example, if traffic in a computer network only flowed in particular directions, we might use a directed graph to model it. A formal definition of directed graph more formally below:

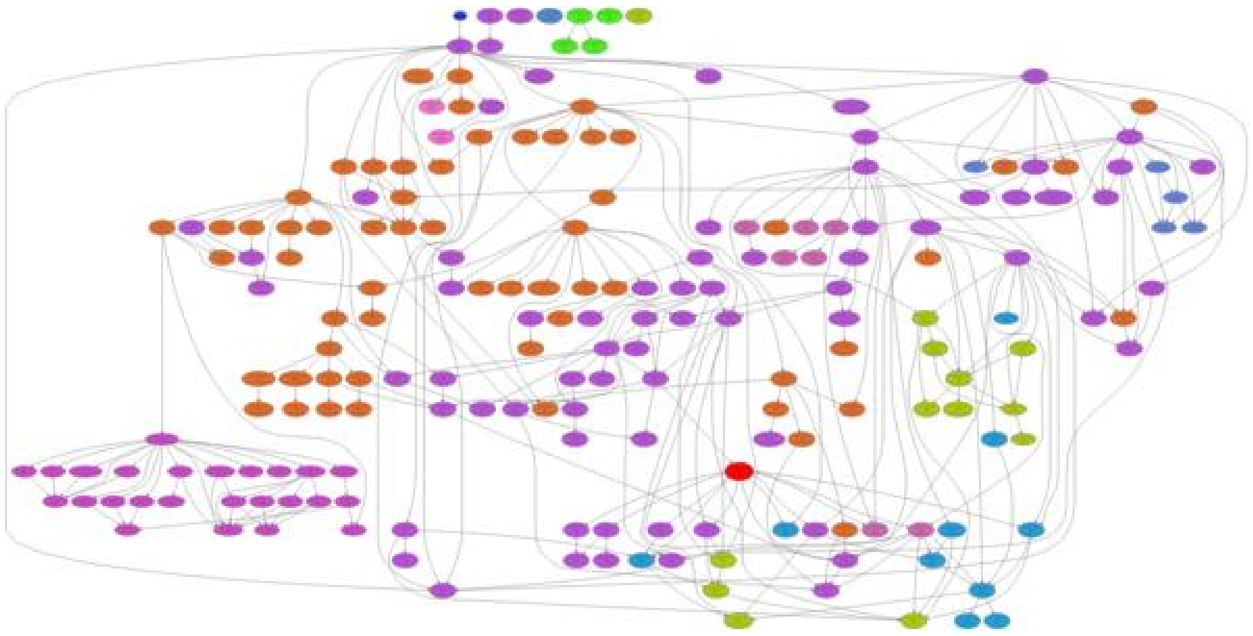
Definition. A directed graph $G = (V, E)$ consists of V, a nonempty finite set of vertices and E, a set of unordered pairs of elements of V called edges. **(End of Definition.)**

Remark. Question: How is this definition different from the definition of simple graph? **(End of Remark.)**

One interesting application (not mentioned in Rosen) of directed graph is to specify the hierarchical import relationships between modules. This kind of directed graph is called **module dependency graph**. Here is an example of a module dependency graph:



Here is another "small" and colorful one:

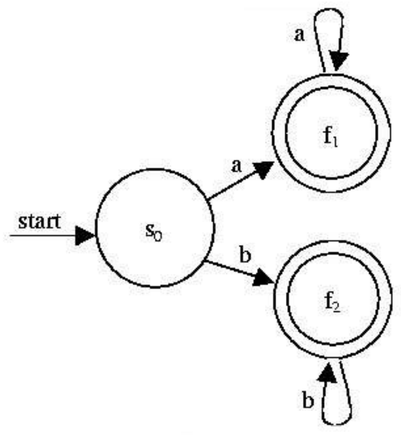


3.2 Directed Multigraph

Sometimes we also want to allow multiples edges between two vertices in directed graphs, so we need the notion of **directed multigraph**:

Definition. A directed multigraph $G = (V, E)$ consists of V, a nonempty finite set of vertices and E, a finite *multiset* set of ordered pairs of *distinct* elements of V called edges. Thus, multiple edges between two vertices are allowed in a multigraph.**(End of Definition.)**

An example application of directed multigraph in Computer Science is to graphically represent finite automata (finite state machine). Here is an example of an automaton:



This kind of diagram is called **state transition diagram** or simply **state diagram**.

3.3 Maple

The Maple commands for directed graph are exactly the same for undirected graph, except we represent edges as ordered pairs of two vertices rather than as sets of two vertices. For example, if we add one more alteration to our computer network, where we are limited to only unidirectional lines of transmission, we would need to create a directed graph in order to represent this new computer network. This can be easily accomplished by signifying an edge as an ordered list of 2 vertices instead of an (unordered) set of 2 vertices. As an example, consider the graph G2 which has the same vertex set as graph G1, but has only unidirectional information lines between cities. We would represent such a computer network as the following directed graph in Maple:

```
G2 := new():
addvertex( T,H,K,L,G2);
addedge([[T,H],[H,K],[K,L],[L,T],[H,L]],G2);
draw(G2);
adjacency(G2);
```

Notice the difference between adjacency matrices of G2 and G1.

3.4 Exercise 3:

1. Write a Maple program that takes input as a natural number n and constructs an n-complete graph K_n. Recall a complete graph is a simple graph with an edge between every pair of vertices.

2. Write a Maple program that takes input as a natural number n and constructs a complete n-partite graphs $K_{m_1,m_2,\ldots,m_n}$. That is, graphs where the vertex set is partitioned into m disjoint sets with $m_i, i = 1, 2 \ldots n$ vertices, respectively, and with every vertex in one of subsets of the partition connected to every vertex of the graph not in this subset by an edge.