If you send me e-mail, it would help a lot if you include the course name in your subject header and an informative title.

The two classes I am teaching are both algorithms classes and it will help me to know which one you are asking me about.

For example:
CSC 482: Clockwise BFS

Change the title on subsequent messages- it will help ensure I do not accidentally miss any of your questions.

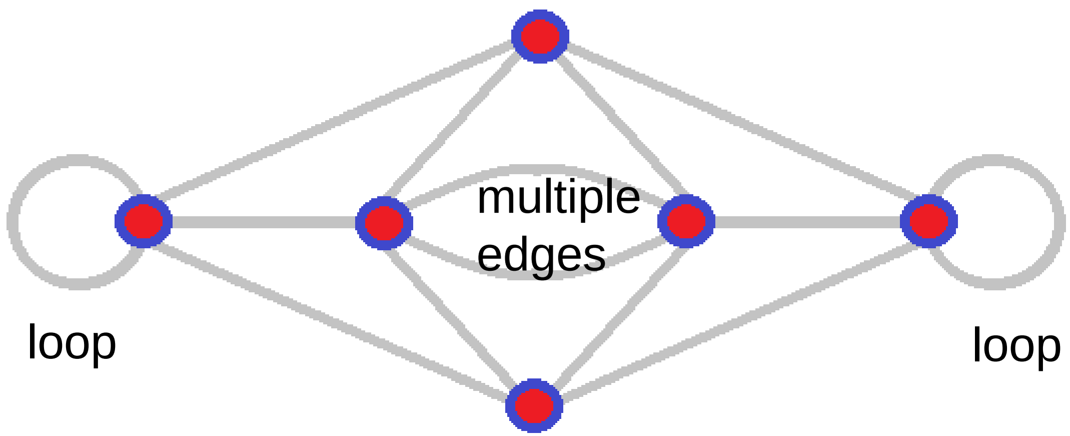
Graphs

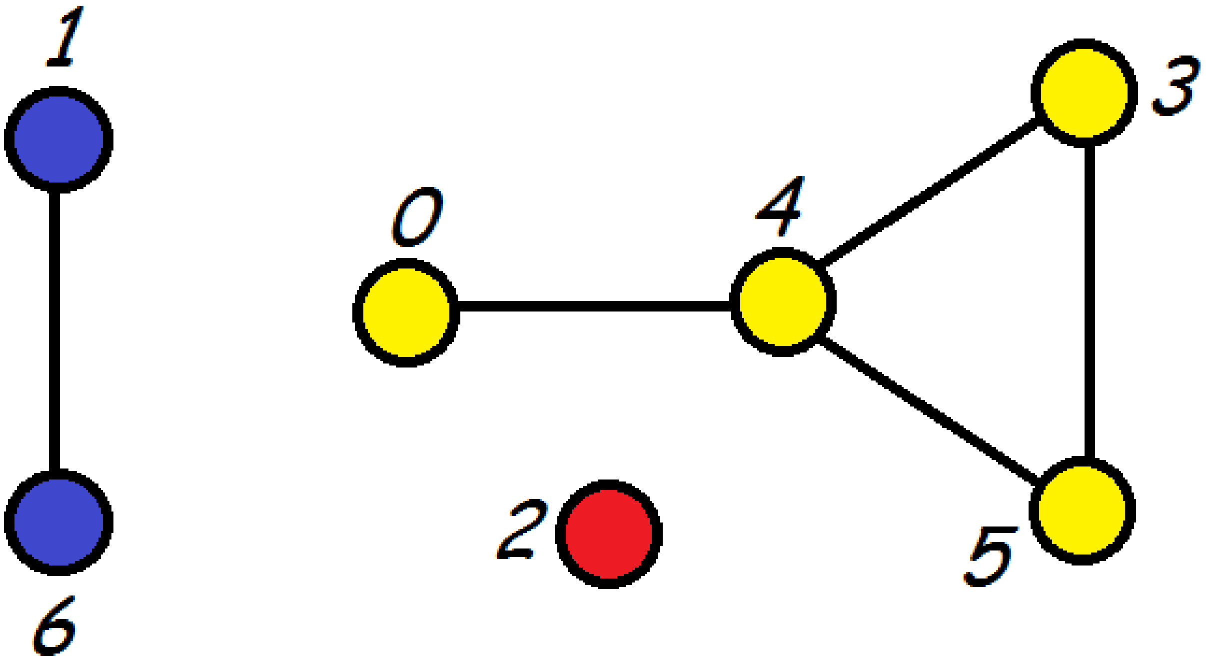
An undirected graph G consists of a set V of vertices and a set E of edges where each edge in E is associated with an unordered pair of vertices from V.

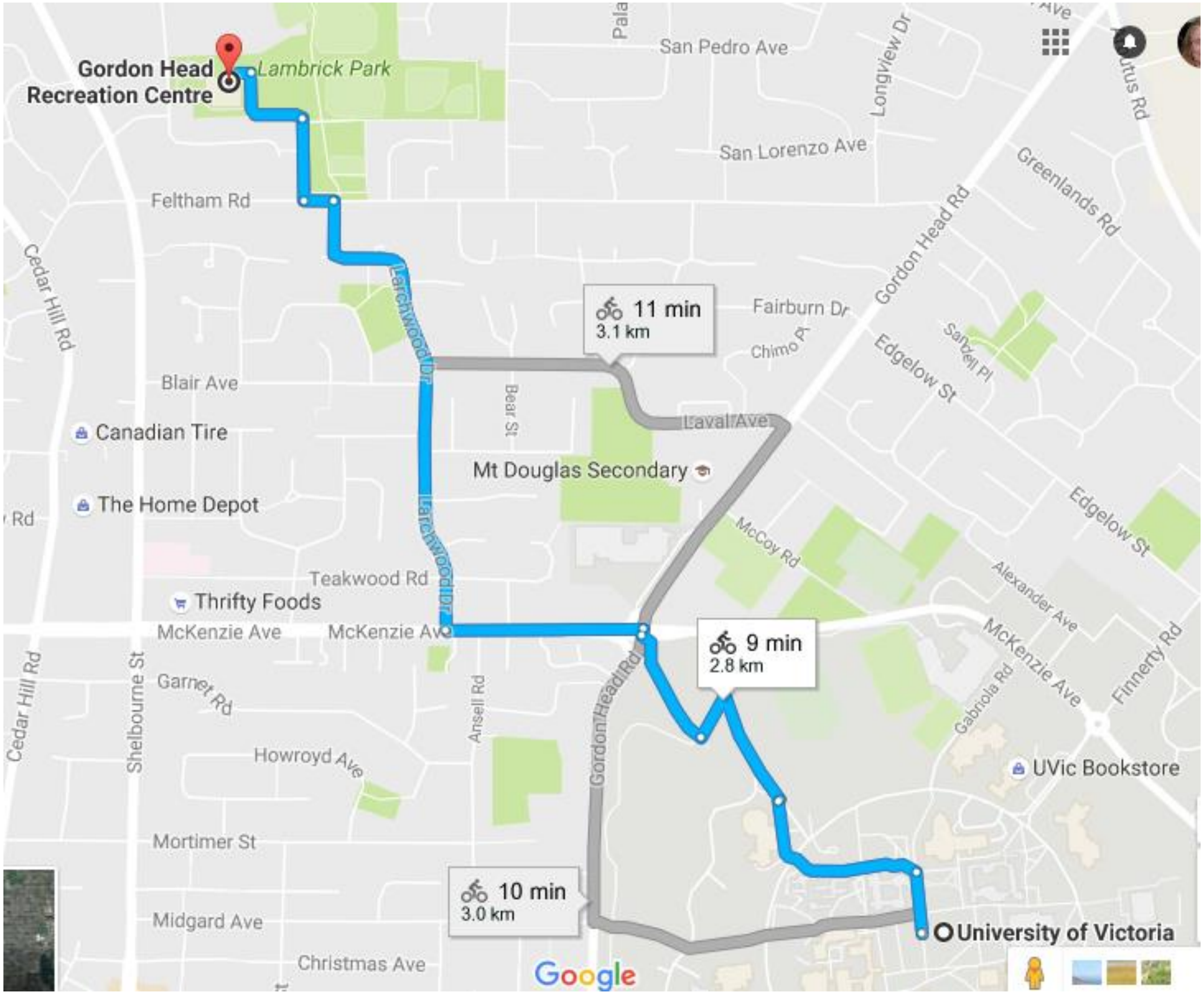
The degree of a vertex v is the number of edges incident to v.

If (u, v) is in E then u and v are adjacent.

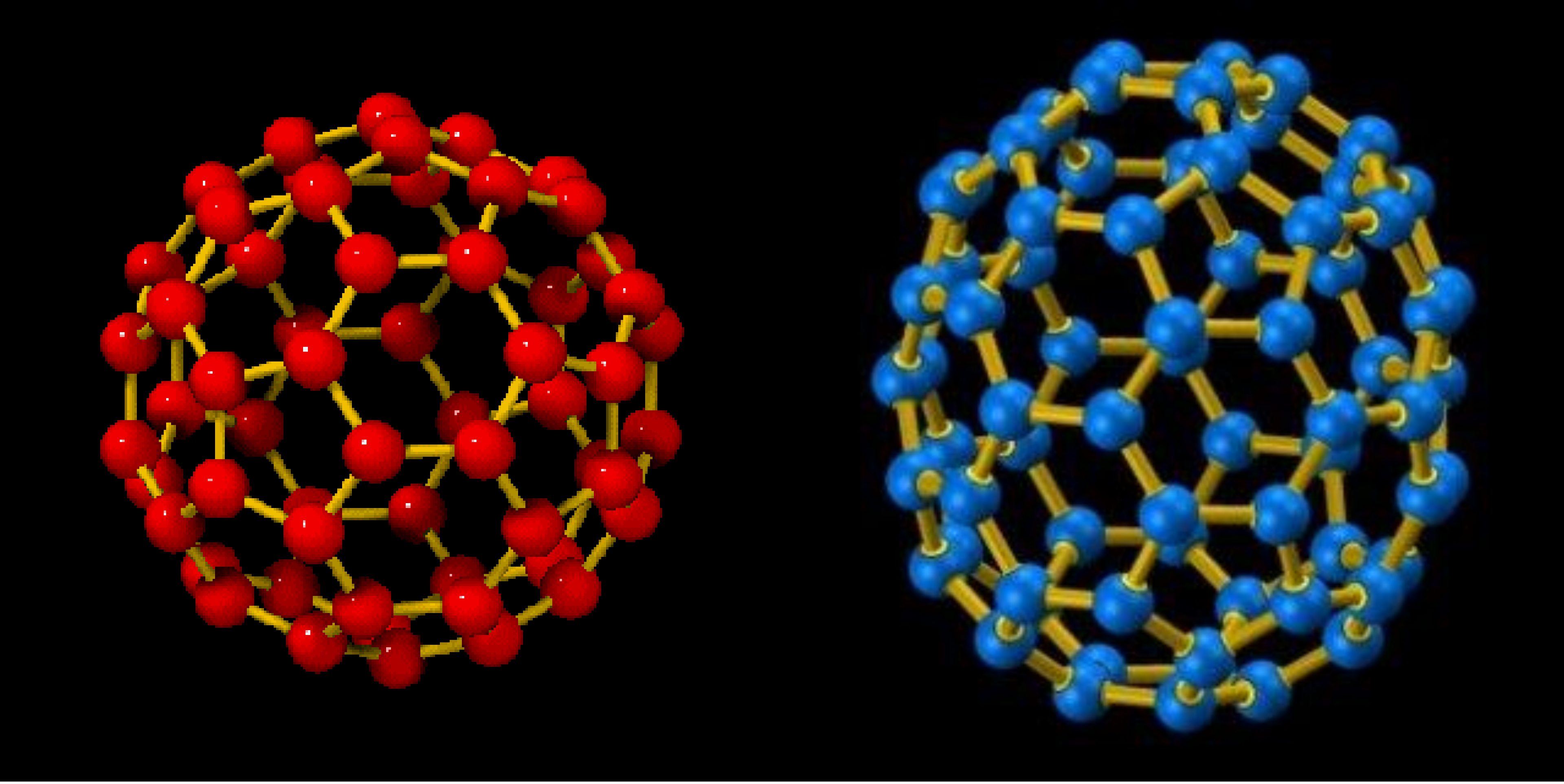
A simple graph has no loops or multiple edges.



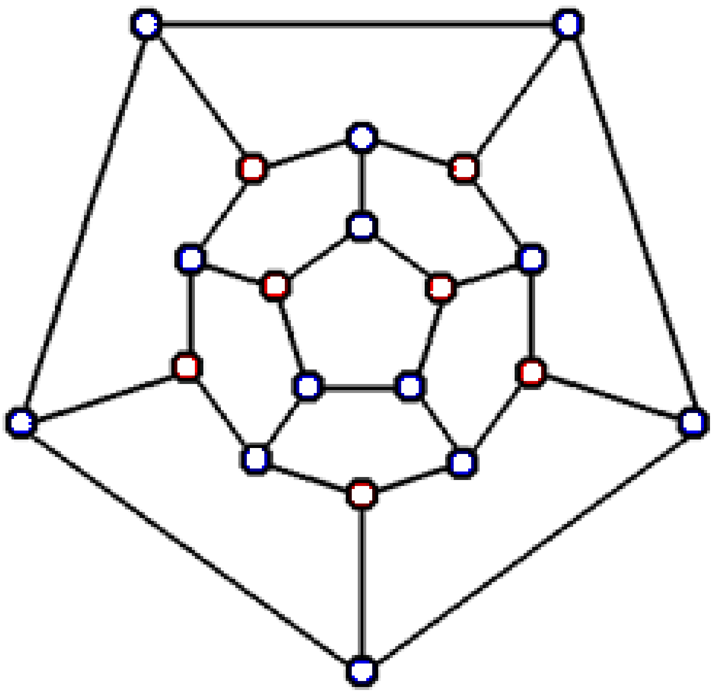

1
0
4
3
2
5
6

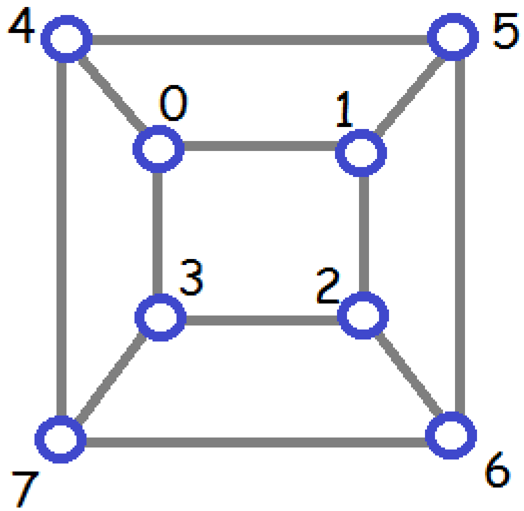


Road Network

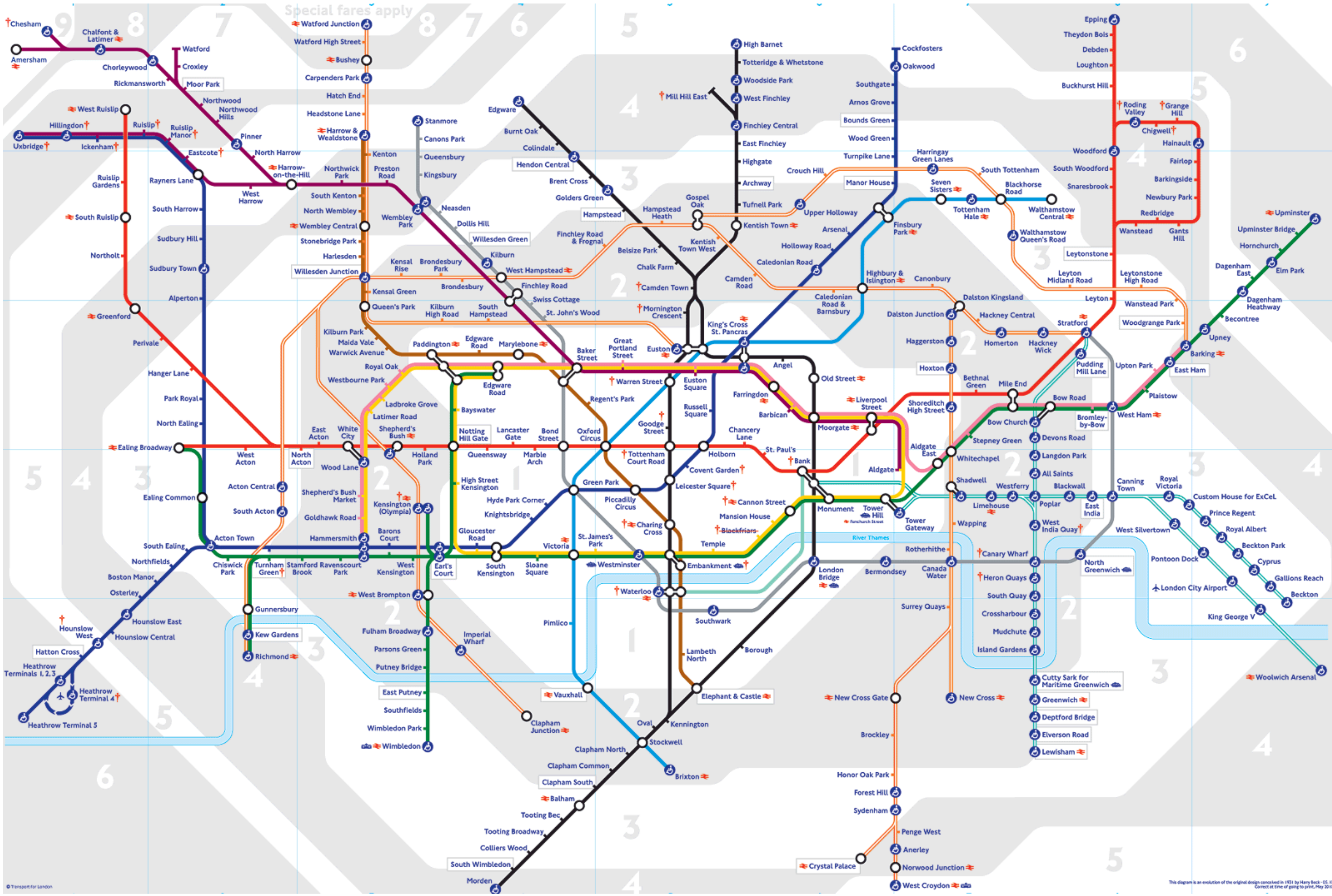


Graphs representing chemical molecules





Polyhedra



Tube map from London, England.

Computer networks

Social networks
Facebook: vertices are people,
add an edge between pairs of people are friends

Water networks

Electrical networks

Wireless networks- add an edge if two sensors are close enough to transmit to each other

Data Structures for Graphs

How can graphs be stored in the computer?

How does this affect the time complexity of algorithms for graphs?

A cycle of a graph is an alternating sequence of vertices and edges of the form v_0, (v_0, v_1), v_1, (v_1, v_2), v_2, (v_2, v_3), … ,v_{k-1}, (v_{k-1}, v_k), v_k where except for $v_0 = v_k$ the vertices are distinct.

Exercise: define path, define connected.

A tree is a connected graph with no cycles.

A subgraph H of a graph G is a graph with $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$.

H is spanning if $V(H) = V(G)$.

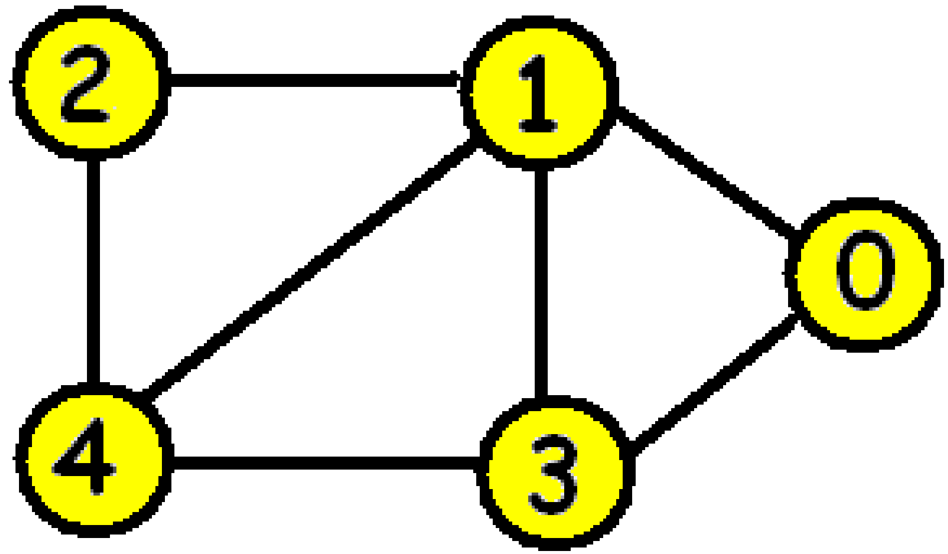
Spanning tree- spanning subgraph which is a tree.

Strange Algorithms

Input: a graph G
Question: does G have a spanning tree?

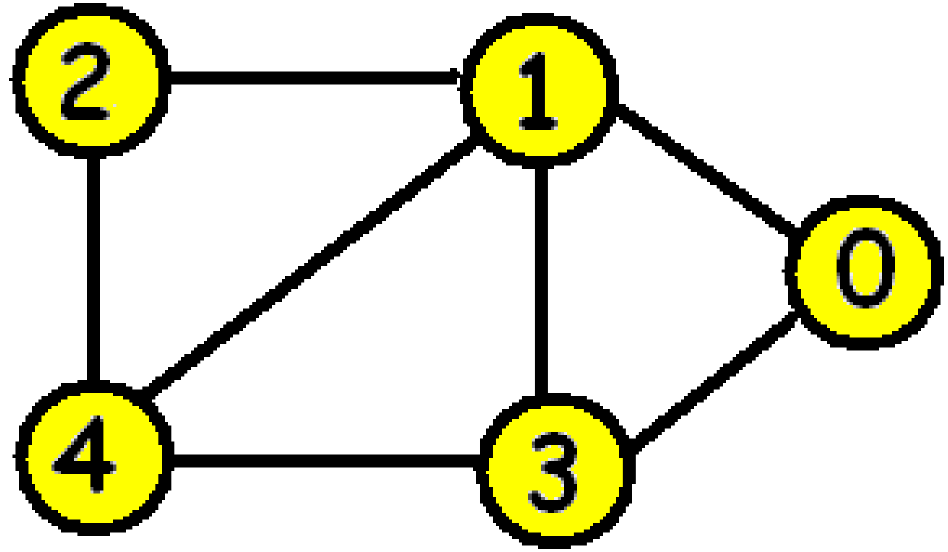
This can be answered by computing a determinant of a matrix and checking to see if it is zero or not.

For lower bound arguments, it is essential to not make too many assumptions about how an algorithm can solve a problem.

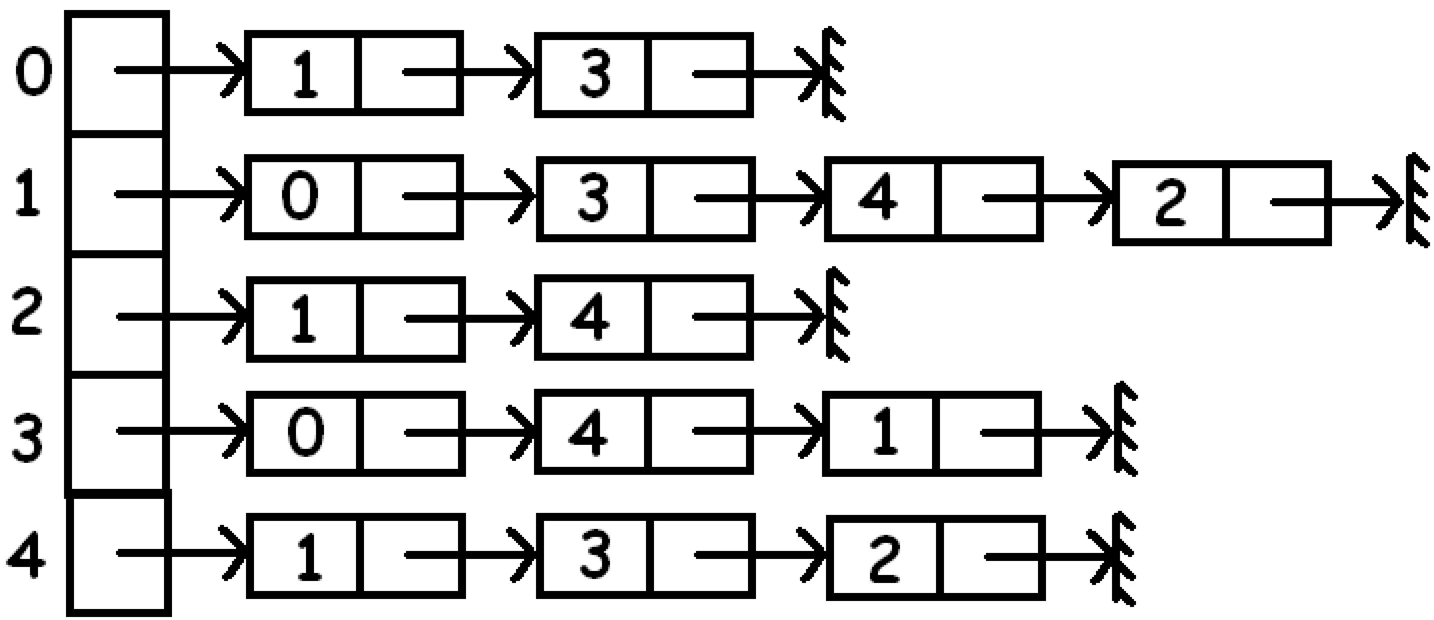


Adjacency matrix:

	0	1	2	3	4
0	0	1	0	1	0
1	1	0	1	1	1
2	0	1	0	0	1
3	1	1	0	0	1
4	0	1	1	1	0



Adjacency list:



Adjacency lists:

Lists can be stored:

1. sorted,
2. in arbitrary order,
3. in some other specific order- for example a rotation system has the neighbours of each vertex listed in clockwise order in some planar embedding of a graph (a picture drawn on the plane with no edges crossing).

Data structures for graphs:

n= number of vertices

m= number of edges

Adjacency matrix: Space $\Theta(n^2)$

Adjacency list: Space $\Theta(n + m)$

How long does it take to do these operations:

1. Insert an edge?
2. Delete an edge?
3. Determine if an edge is present?
4. Traverse all the edges of a graph?

Data structures for graphs:

n= number of vertices

m= number of edges

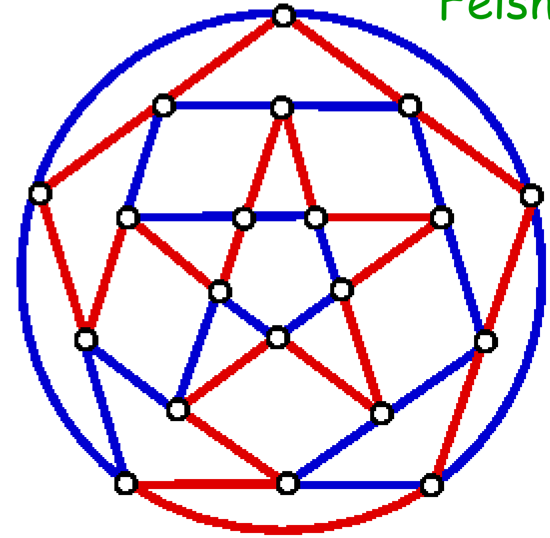
Adjacency matrix: Space $\Theta(n^2)$

Adjacency list: Space $\Theta(n + m)$

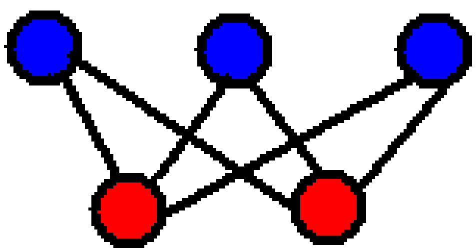
How long does it take to do these operations:

1. Insert an edge?
2. Delete an edge?
3. Determine if an edge is present?
4. Traverse all the edges of a graph?

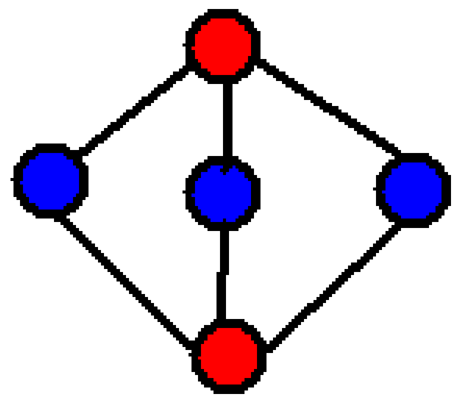
Stefan Felsner



A graph G is planar if it can be drawn on the plane with no crossing edges.



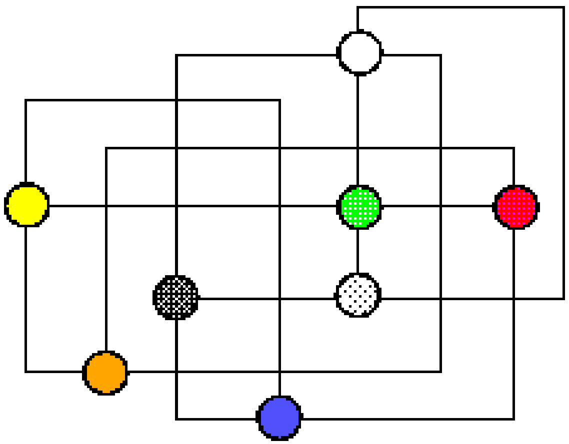
planar graph G



planar embedding of G

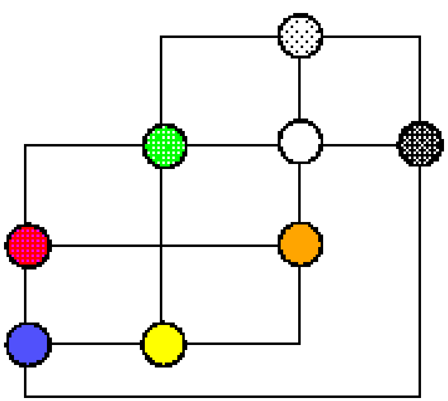
Used as a starting point to find nice pictures of non-planar graphs:

Map 4-colouring:

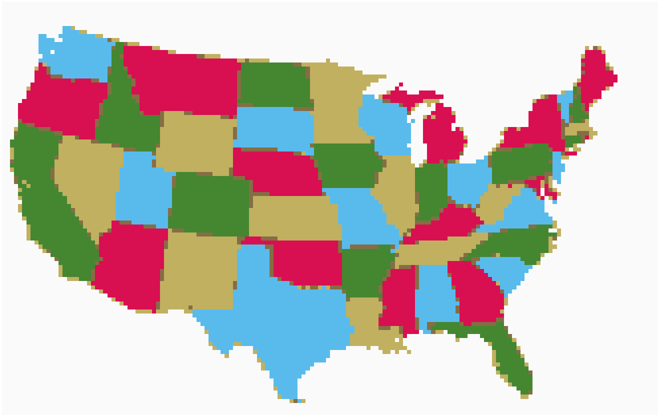


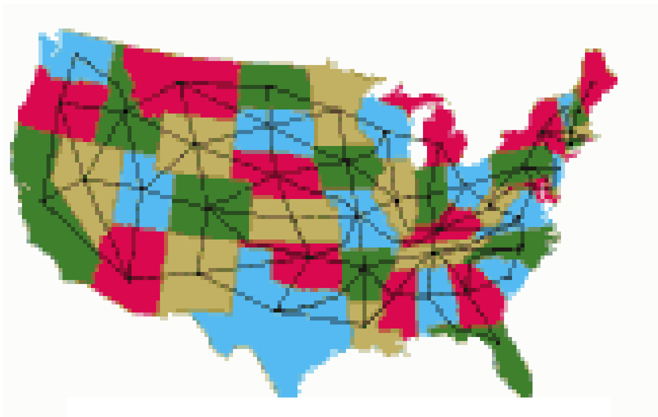
A graph showing normal relationships with lots of crossings in it.

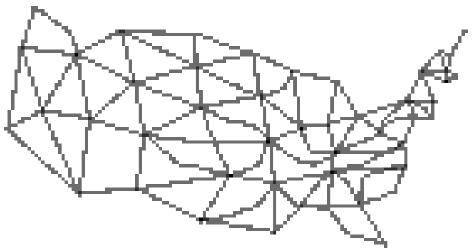
Codeguru



The same graph optimized to show only one crossing in it. The relations are maintained as it is.



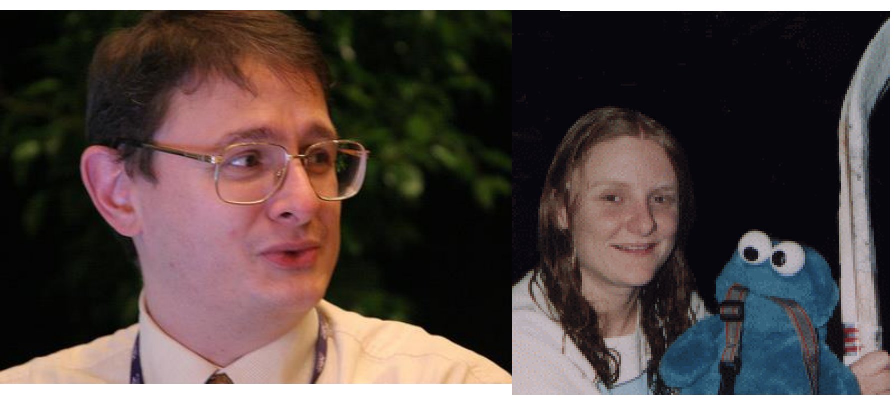




Linear time algorithms for embedding:

Hopcroft & Tarjan, '74

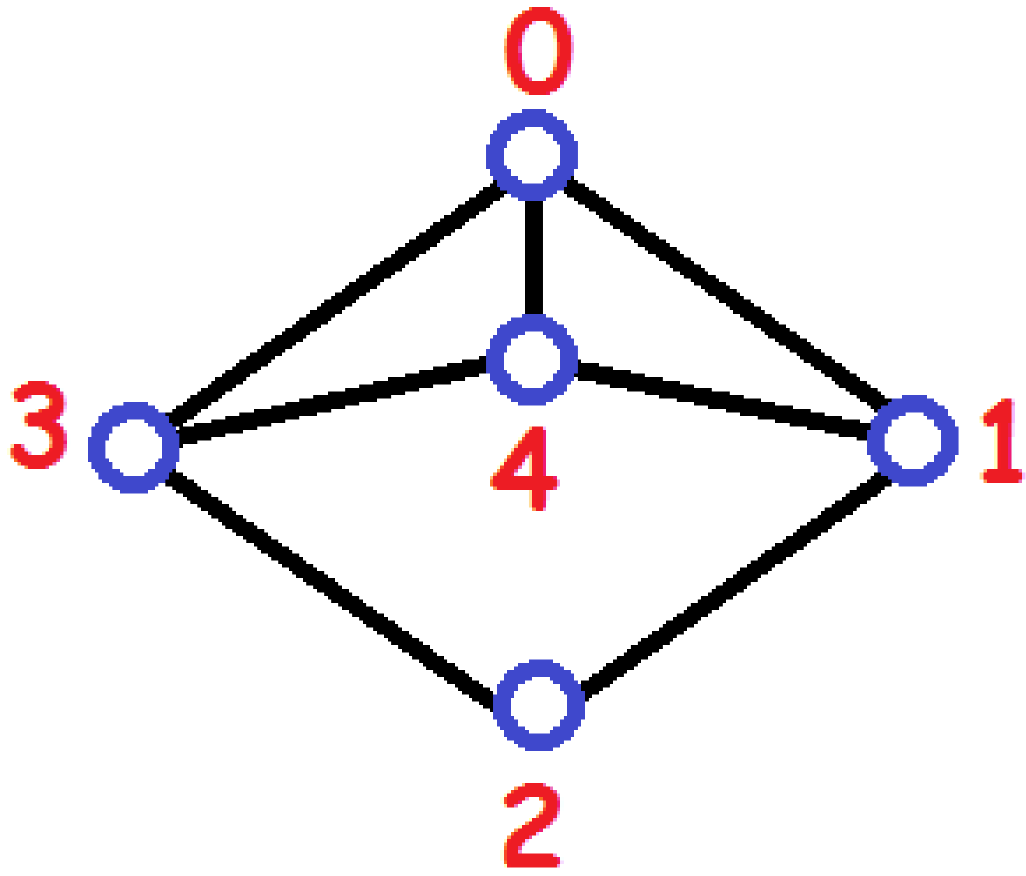
Booth and Lueker, '76



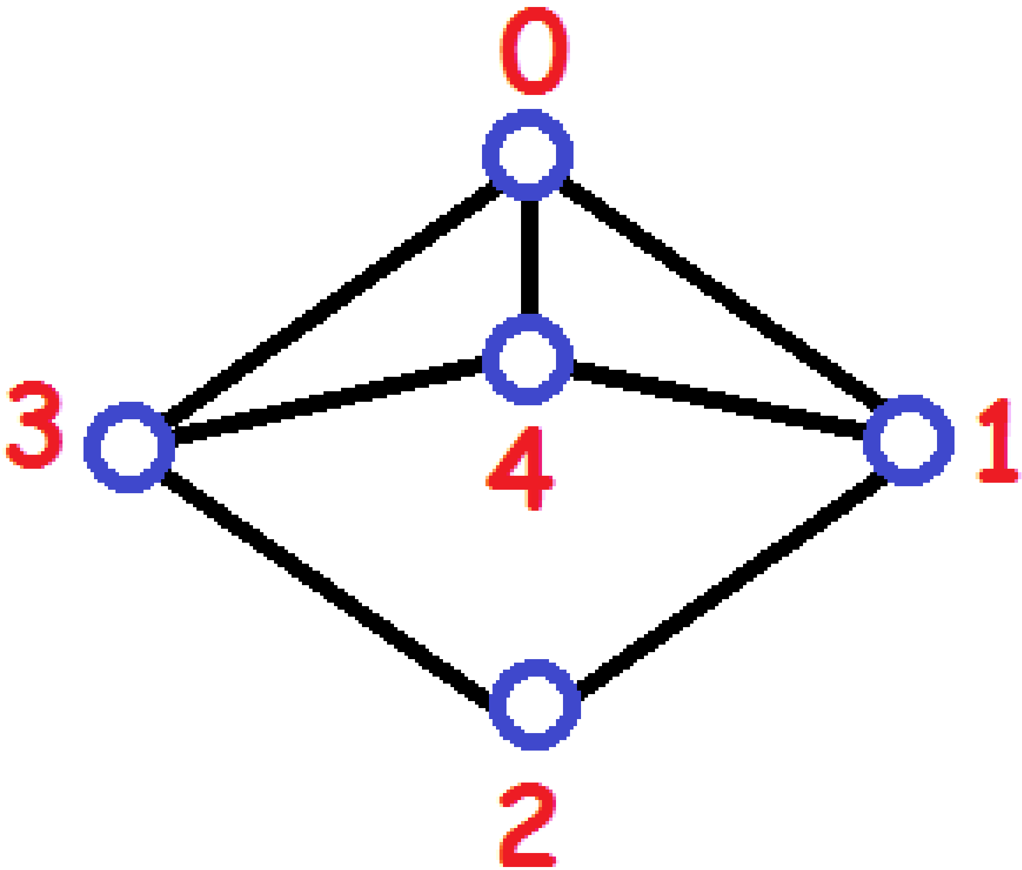
Boyer & Myrvold, '01

OPEN: Find a really simple O(n) or maybe O(n log n) algorithm.

What is the rotation system for this graph?



What is the rotation system for this graph?



0: 1 4 3

1: 0 2 4

2: 1 3

3: 0 4 2

4: 0 1 3

To walk the faces from a rotation system:

Treat each edge as two arcs:
(u,v) → (u, v) and (v,u).

Mark all arcs as not visited.

For each unvisited arc (u,v) do:
walk the face with (u, v).

To walk the face with (u,v):

Arcs traversed are marked as visited.

The next arc to choose after an arc (u,v) is the arc (v,w) such that w is the vertex in the list of neighbours of v that comes after u in cyclic order.

Continue traversing arcs until returning to arc (u,v).

Walk the faces of this planar embedding of a graph:

0: 1 4 3

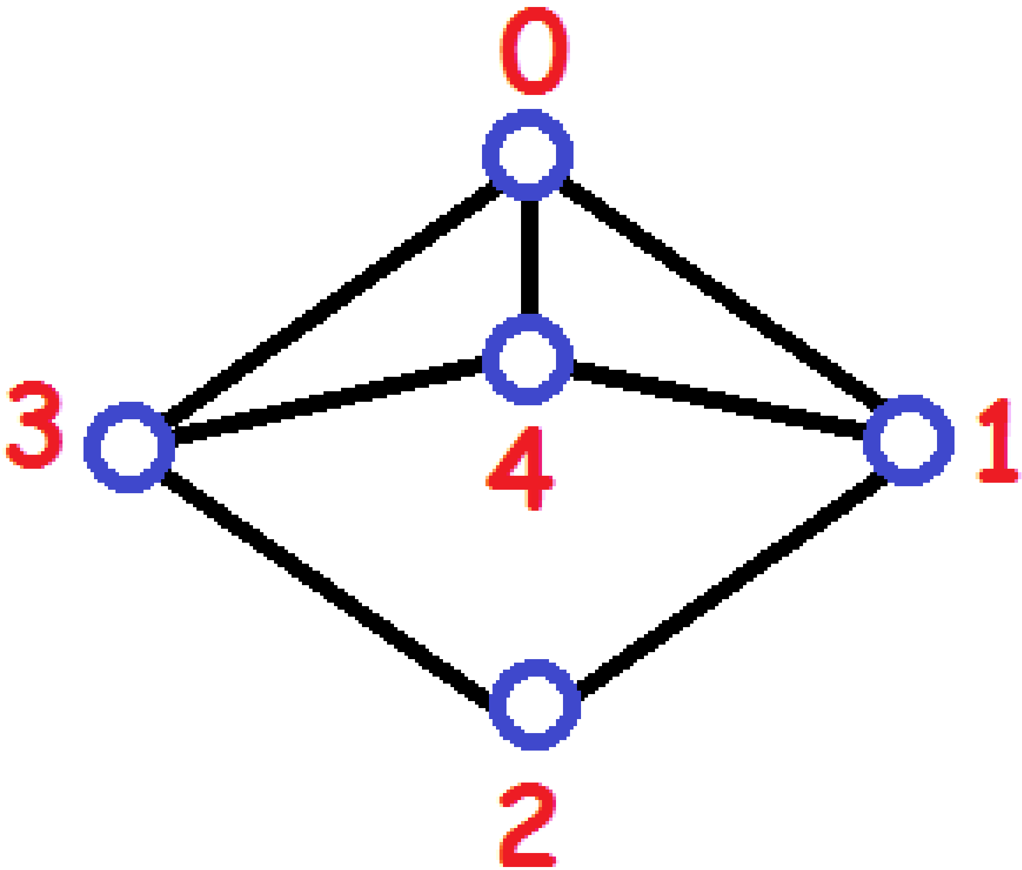
1: 0 2 4

2: 1 3

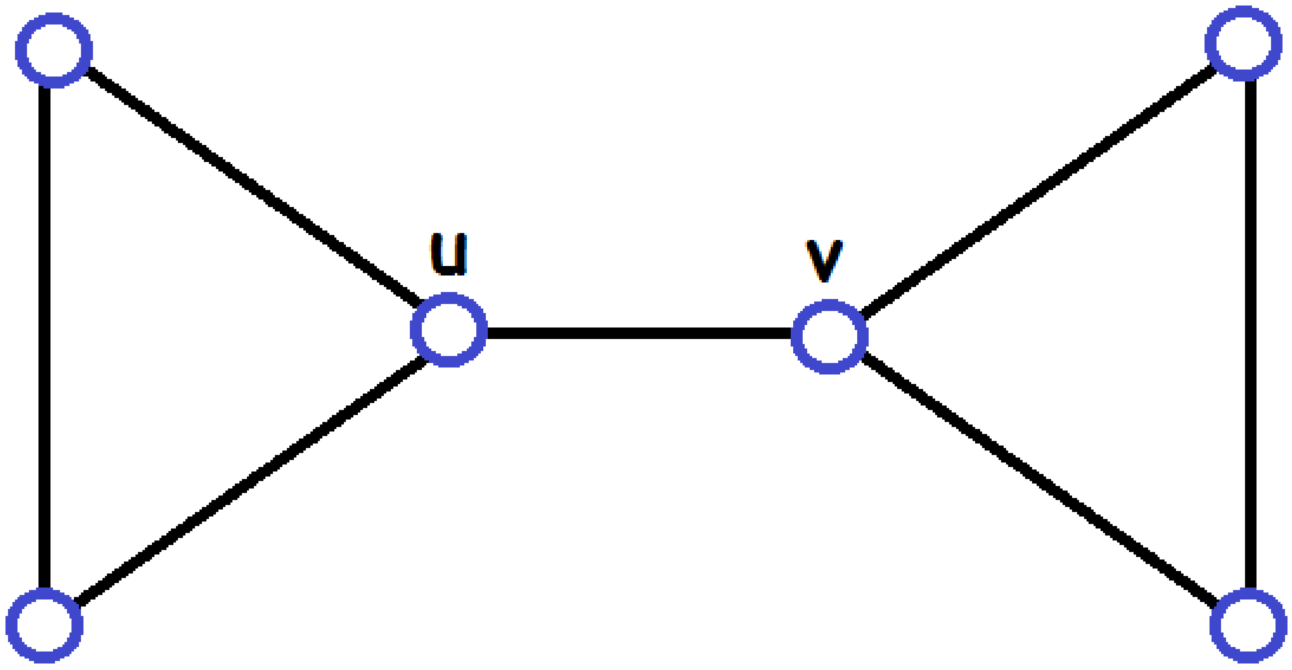
3: 0 4 2

4: 0 1 3

This graph has 4 faces.



0: 1 4 3
1: 0 2 4
2: 1 3
3: 0 4 2
4: 0 1 3



Important: Do not stop until seeing the starting ARC again. It's possible to have both (u,v) and (v,u) on the same face.

G connected on an orientable surface:

genus $g = (2 - n + m - f)/2$

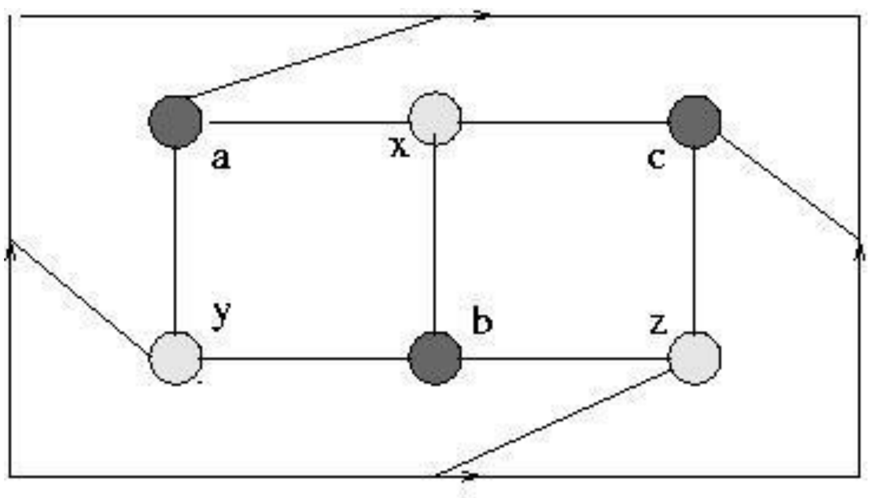
0 plane

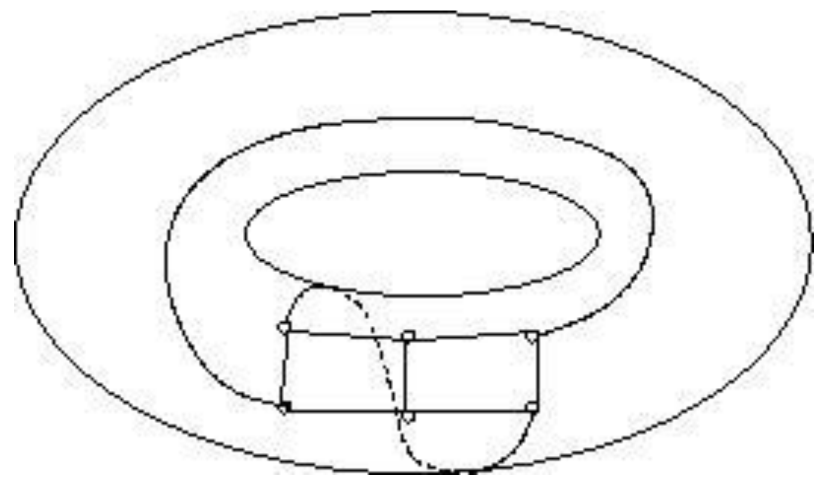
1 torus

2

Greg McShane

Torus Embedding





Embedding:

Linear time: Juvan, Marincek & Mohar, '94

$O(n^3)$: Juvan & Mohar, preprint, implementation is buggy

Faces can have repeated vertices and this makes embedding hard: