Pràctica Sessió 2 de MATD

L'objectiu d'aquesta sessió de pràctiques és familiaritzar-se amb la terminologia bàsica de Teoria de Grafs.

Grafs

A continuació treballarem amb diferents maneres de declarar grafs en **SageMath**. Veurem com podem definir un mateix graf, utilitzant diferents arguments en el procediment **Graph()**

Argument: Conjunt de vèrtexs i arestes

In [1]:
```
V={1,2,3,4}
E=[{1,2},{1,3},{2,3},{2,4},{3,4}]
G1=Graph([V,E],format='vertices_and_edges')
```

Representa gràficament el graf G1

In [ ]:
```

```

Argument: Associar a vèrtexs la seva llista d'adjacències

In [3]:
```
G2=Graph({1:[2,3],2:[3,4],3:[4]})
```

Representa el graf G2

In [ ]:
```

```

Argument: Llista d'arestes del graf

In [1]:
```
G3=Graph([{1,2},{1,3},{2,3},{2,4},{3,4}])
```

Representa el graf G3

In [ ]:

Troba el conjunt de vèrtexs i arestes del graf de Petersen.

In [ ]:
```
# conjunt de vèrtexs
```

In [ ]:
```
# conjunt d'arestes
```

Troba el conjunt de vèrtexs i arestes del graf de Möbius–Kantor.

In [ ]:

In [ ]:

Afegir vèrtexs i arestes

Crearem un nou graf, amb la instrucció **Graph()**. Es tracta d'un graf nul, sense vèrtexs, ni arestes.

In [13]:
```
G=Graph()
```

Podrem afegir vèrtexs al graf, d'un en un (**g.add_vertex**), o en conjunt (**g.add_vertices**)

In [16]:
```
G.add_vertex(0)
```

In [ ]:
```
G.add_vertices([1,2,3,4])
```

Representa el graf **G**

In [ ]:

I també podem afegir arestes al graf, d'una en una (**G.add_edge()**), o en conjunts (**G.add_edges()**)

In [18]:
```
G.add_edge((0,1));
G.add_edges([(0,2),(0,3),(0,4)])
```

Representa el graf **G** que hem creat en els passos anteriors.

In [ ]:

També podrem eliminar algun dels elements que hem afegit amb la instrucció **G.delete_()**

In [20]:
```
G.delete_edge((0,4));
G.delete_vertex(4)
```

Comprova amb un dibuix com han estat eliminats els elements

In [ ]:

Definiciones básicas

Utilitzant instruccions de Sagemath troba l'ordre (nombre de vèrtexs) i mida (nombre d'arestes) del graf de Hoffman-Singleton.

In [ ]:
```
# ordre
```

In [ ]:
```
# mida
```

Troba els vèrtexs adjacents al vèrtex "0" en el graf de Hofman-Singleton.

In [ ]:

Per a calcular el grau d'un vèrtex "v" d'un graf G en **SageMath** executarem la instrucció **G.degre(v)**

Troba el grau del vèrtex "0" del graf Hofman-Singleton.

In [ ]:

Troba la seqüència de graus del graf Moebius-Kantor utilitzant una instrucció de **SageMath** i contesta: El graf de Moebius-Kantor és regular?

In [ ]:

Matrius d'adjacència i incidència

Troba la matriu d'adjacència del graf de Petersen i comprova com és simètrica respecte a la seva diagonal principal. A més, el número d'uns de cada fila (columna) coincideix amb el grau del corresponent vèrtex del graf.

In [ ]:
```
# matriu d'adjacència
```

També podem utilitzar la matriu d'adjacència d'un graf com a argument del procediment **Graph()** per a crear un graf en **SageMath**. Vegem un exemple

In [ ]:
```
M=matrix([[0,1,1,0],[1,0,1,1],[1,1,0,1],[0,1,1,0]])
g=Graph(M)
g.show()
```

Una altra manera de representar grafs és mitjançant les matrius d'incidència. Mentre la matriu d'adjacència involucra les adjacències de vèrtexs, la matriu d'incidència involucra la incidència de vèrtexs i arestes. Troba la matriu d'incidència del graf de Petersen.

In [ ]:
```
# matriu d'incidència
```

Representació gràfica

Per a representar gràficament un graf podem utilitzar la intrucción **show()**, tal com hem fet al principi de la pràctica, però tenim altres alternatives.

In [ ]:
```
g.show()
```

Hi ha ocasions en què és millor representar el graf d'una manera determinada, per exemple, posant cada conjunt de vèrtexs dels grafs bipartits en una recta, o distribuint els vèrtexs de forma circular.

Els diferents algorismes de dissenys (layout) que es poden aplicar són "circular", "planar" o "spring", entre altres. Prova'ls per a representar el graf hipercub Q_3 (**graphs.CubeGraph(3)**).

In [ ]:
```
graphs.CubeGraph(3).show(layout="circular")
```

In [ ]:
```
# planar
```

In [ ]:
```
# spring
```

I també és possible indicar la posició dels vèrtexs utilitzant l'argument **pos**, per a indicar les coordenades de la posició exacta dels vèrtexs

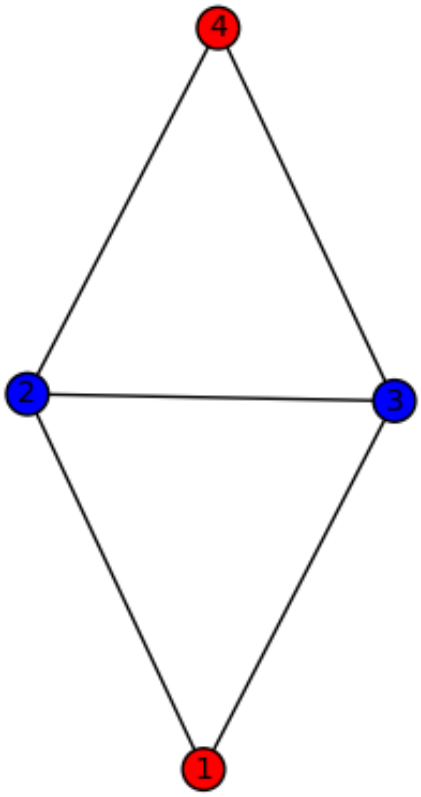
In [26]:
```
position={0:(1,0),1:(-1,0),2:(0,-1),3:(-1,-1)}
```

In [ ]:
```
g.show(pos=position)
```

Destacar elements del graf

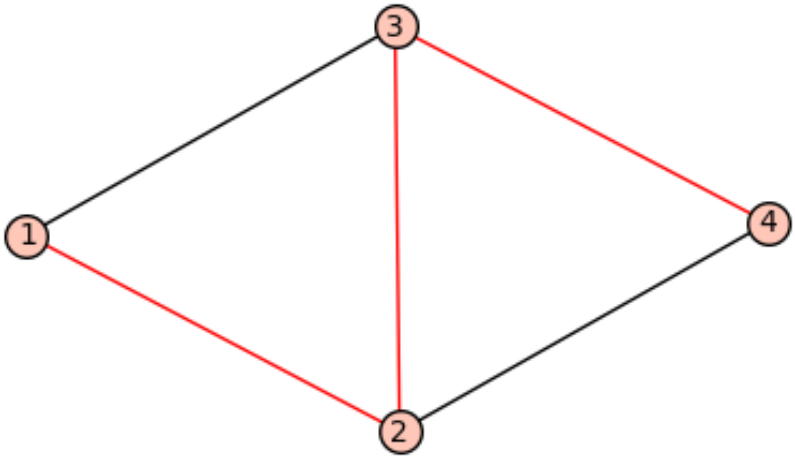
A continuació veurem en un exemple, com podem destacar en la representació gràfica d'un graf els seus vèrtexs

In [11]:
```
colors={"red":[1,4],"blue":[2,3]}
g=G1.graphplot(vertex_colors=colors)
g.show()
```



O les seves arestes

In [12]:
```
Ecolors={"red":[(1,2),(2,3),(3,4)],"black":[(1,3),(2,4)]}
g2=G1.graphplot(edge_colors=Ecolors)
g2.show()
```



Acoloreix els vèrtexs del graf Q_3 amb dos colors diferents, de manera que no hi hagi una aresta que tingui els seus dos extrems acolorits amb el mateix color.

In [ ]:

Crea un graf simple en el qual els vèrtexs siguin els números de l'1 al 10 i dos vèrtexs seran adjacents si un és múltiple de l'altre.

In [ ]: