Pràctica 3 Grafs. Propietats

En aquesta sessió de pràctiques presentarem les funcions de **SageMath** que ens permetran treballar amb les principals propietats de Grafs introduïdes en els primers temes de MATD.

1. **Complementari d'un graf**

El complementari d'un graf $G = (V(G), E(G))$ de ordre n, és un graf CG que té per vèrtexs, els mateixos vèrtexs que G i per arestes $E(CG) = E(K_n) - E(G)$.

Troba en el guió de la pràctica 2, la instrucció de **SageMath** que permetrà obtenir el graf complementari del graf G.

Utilitza-la per a obtenir el complementari del graf estrella **graphs.StarGraph(5)** i representa'ls gràficament.

In [ ]:

```

```

In [ ]:

```
#Representem-los gràficament
# G1 és el graf estrella i CG1 el seu complementari
k = []
k.append(plot(G1))
k.append(plot(CG1))
G=graphics_array(k)
G.show()
```

Troba el graf complementari del graf de Petersen i representa'ls gràficament.

In [ ]:

```

```

1. **Grafs isomorfs**

Dos grafs $G_1 = (V(G_1), E(G_1)$ i $G_2 = (V(G_2), E(G_2))$ són isomorfs si existeix una aplicació entre els seus conjunts de vèrtexs que conserva les adjacències.

Comprova si el graf cicle d'ordre 5 i el graf estrella d'ordre 5 són isomorfs.

In [ ]:

```

```

1. **Graf autocomplementari**

Direm que un graf G és autocomplementari, si és isomorf al seu graf complementari.

La instrucció **G.is_self_complementary()** ens permet comprovar-ho. Cerca un graf que sigui autocomplementari i comprova que efectivament ho és utilitzant la instrucció anterior.

In [ ]:

I també trobant el seu complementari i comprovant que els dos grafs són isomorfs.

In [ ]:

1. **Graf bipartit**

Un graf $G = (V(G), E(G))$ es diu que és bipartit si el conjunt de vèrtexs del graf es pot descompondre com una unió disjunta de conjunts

$$V(G) = V(G_1) \cup V(G_2) \text{ i } V(G_1) \cap V(G_2) = \emptyset,$$

de manera que no hi ha arestes en el graf, que tinguin els seus vèrtexs extrems en el mateix conjunt de la bipartició.

La funció **G.is_bipartite()** permet comprovar si el graf G és bipartit. Vegem com funciona amb un exemple.

In [ ]:
```
G=graphs.GridGraph([3,4])
G.is_bipartite()
```

In [ ]:
```
plot(G)
```

Cerca un exemple de graf que sigui bipartit i un altre exemple de graf que no hi sigui.

In [ ]:

1. **Graf regular**

Un graf G es diu que és regular, si tots els seus vèrtexs tenen el mateix grau.

Per a comprovar-ho, podrem utilitzar la funció **G.is_regular()** de **SageMath**.

Cerca un graf que sigui regular i comprova que efectivament ho és utilitzant la instrucció anterior.

In [ ]:

A continuació calcula la seqüència de graus dels seus vèrtexs.

In [ ]:

Sigui H un graf simple definit pel conjunt de vèrtexs

$$V = \{1, 2, \dots, 20\}$$

i conjunt d'arestes

$$E = \{(i, j) : \ \mathrm{mcd}(i, j) > 1\}.$$

Defineix el graf H i representa'l

In [ ]:

I calcula la seqüència de graus dels seus vèrtexs.

In [ ]:

1. **Subgraf**

L'objectiu d'aquesta secció de la pràctica és treballar amb el concepte de subgraf. Comencem recordant la seva definició.

Donat un graf $G = (V(G), E(G))$, un **subgraf** $H = (V(H), E(H))$ és un graf que compleix

$$V(H) \subset V(G) \quad \text{i} \quad E(H) \subset E(G)$$

Representem tots els subgrafs (no isomorfs) del graf complet K_4, utilitzant la instrucció de **SageMath**.

$$graphs(vertices, property = lambda\ x : True, augment =' \ edges', size = None)$$

In [ ]:
```
L = graphs(4,augment='vertices')
graphs_list.show_graphs(L)
```

Genera tots els subgrafs no isomorfs del graf complet K_5

In [ ]:

A continuació representarem tots els subgrafs no isomorfs de grandària 4 del graf complet K_5

In [ ]:
```
L = graphs(5, size=4)
graphs_list.show_graphs(L)
```

Genera tots els subgrafs no isomorfs del graf complet K_5 que tinguin mida 6

In [ ]:
```

```

Ara veurem quants dels subgrafs no isomorfs del graf complet K_4 són bipartits

In [ ]:
```
#property defineix la propietat de ser bipartit
property = lambda G: G.is_bipartite()
# Incloem en una llista només els que tinguin la propietat
L=list(graphs(4, property))
#comptem els elements de la llista
len(L)
```

I els representem gràficament

In [ ]:
```
graphs_list.show_graphs(L)
```

Ara, determina quants subgrafs no isomorfs del graf complet K_5 són bipartits i després representa'ls

In [ ]:
```

```

Els conceptes amb els que treballaràs a partir d'aquest input estan inclosos en el guió de la pràctica 3 de MATD.

1. **Acoloriments d'un graf**

Calcula el número cromàtic i el polinomi cromàtic del graf de Petersen.

In [ ]:
```

```

Calcula el nombre d'acoloriments diferents que permet el graf de Petersen.

In [ ]:
```

```

Llegeix en la pàgina 6 del guió de la pràctica, el procés complet per a trobar un quadrat llatí (QL) d'ordre n i a continuació mira d'identificar els diferents passos en la següent rutina (pàgina 7, guió de la pràctica)

In [1]:
```
from sage.graphs.graph_coloring import vertex_coloring
```

In [2]:
```
def QL(nn):
    # generem el grid graf
    G=graphs.GridGraph([nn,nn])
    # afegim les branques necessaries
    for i in range(nn):
        for j in range(nn):
            G.add_edges([((i,j),(i,k)) for k in range(j+1,nn)])
            G.add_edges([((i,j),(k,j)) for k in range(i+1,nn)])
    # generem l'acoloriment de G
    # (retorna una k-particio del vertexs)
    vc=vertex_coloring(G,value_only=False)
    # ql es la llista que contindra el QL
    # es una llista de llistes (les files)
    ql =[]
    # els seguents llacos fan els
    #passos 3b i 3c
    for i in range(nn):
        laux =[]
        for j in range(nn):
            for k in range(nn):
                if ((i,j) in vc[k]):
                    laux.append(k)
                    break
        ql.append(laux)
    return G.plot(vertex_colors=vc,vertex_labels=False),ql
```

A continuació prova la rutina per trobar un QL d'ordre 7 i mostra-ho en format de taula.

In [ ]:

In [ ]:

Llegeix en la pàgina 9 del guió de la pràctica, el procés complet per a trobar un sudoku d'ordre n i a continuació mira d'identificar els diferents passos en la següent rutina (pàgina 9 del guió de la pràctica)

In [ ]:
```
def SK():
    nn=9
    # generem el grid graf
    G=graphs.GridGraph([nn,nn])
    # afegim les branques necessaries
    for i in range(nn):
        for j in range(nn):
            G.add_edges([((i,j),(i,k)) for k in range(j+1,nn)])
            G.add_edges([((i,j),(k,j)) for k in range(i+1,nn)])
    # afegim les condicions addicionals
    for i in [0 ,3 ,6]:
        for j in [0 ,3 ,6]:
            G.add_edges([((i,j),(i+1,j+1)),((i,j),(i+1,j+2)),((i,j)
,(i+2,j+1)),((i,j),(i+2,j+2))])
            G.add_edges([((i,j+1),(i+1,j)),((i,j+1),(i+1,j+2)),((i,
j+1),(i+2,j)),((i,j+1),(i+2,j+2))])
            G.add_edges([((i,j+2),(i+1,j)),((i,j+2),(i+1,j+1)),((i,
j+2),(i+2,j)),((i,j+2),(i+2,j+1))])
            G.add_edges([((i+1,j),(i+2,j+1)),((i+1,j),(i+2,j+2))])
            G.add_edges([((i+1,j+1),(i+2,j)),((i+1,j+1),(i+2,j+2))]
)
            G.add_edges([((i+1,j+2),(i+2,j)),((i+1,j+2),(i+2,j+1))]
)
    # generem l'acoloriment de G
    # (retorna una k-particio del vertexs)
    vc=vertex_coloring(G,value_only=False)
    # ql es la llista que contindra el SK
    # es una llista de llistes (les files)
    sk =[]
    # els seguents llacos fan els
    # passos 3b i 3c que tambe funcionen # pel SKs
    for i in range(nn):
        laux =[]
        for j in range(nn):
            for k in range(nn):
                if ((i,j) in vc[k]):
                    laux.append(k+1)
                    break
        sk.append(laux)
    return sk
```

I a continuació genera el sudoku.

In [ ]:
```
```