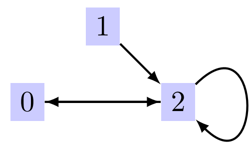
Mathématiques pratiques
Relations binaires

Lors de ce travail pratique, on va explorer l'ensemble des relations binaires sur un ensemble à n éléments, disons pour fixer les idées $E_n = [\![0, n-1]\!]$. On étudiera également une relation aléatoire.

Exercice 1 : génération de relations

Se donner une relation binaire sur E_n, c'est se donner une partie du produit cartésien $E_n \times E_n$. Par exemple, la partie $\{(0,2),(1,1),(1,2),(2,0)\}$ de E_3 est le graphe d'une relation binaire dont une représentation sagittale serait :



On peut la coder sur python via une liste de tuples :

```
[(0,2), (1,1), (1,2), (2,0)]
```

1. En vous inspirant du TP sur les *parties* d'un ensemble, écrire une fonction `all_relations(n)` qui renvoie une liste de toutes les parties de E_n. Par exemple, `all_relations(2)` devrait renvoyer :

```
[[(0, 0), (0, 1), (1, 0), (1, 1)],
 [(0, 0), (0, 1), (1, 0)],
 [(0, 0), (0, 1), (1, 1)],
 [(0, 0), (0, 1)],
 [(0, 0), (1, 0), (1, 1)],
 [(0, 0), (1, 0)],
 [(0, 0), (1, 1)],
 [(0, 0)],
 [(0, 1), (1, 0), (1, 1)],
 [(0, 1), (1, 0)],
 [(0, 1), (1, 1)],
 [(0, 1)],
 [(1, 0), (1, 1)],
 [(1, 0)],
 [(1, 1)],
 []]
```

   que l'on interprète comme les graphes des 16 relations binaires possibles.

2. Générer toutes les relations binaires sur E_3. Combien de relations obtenez-vous ? Combien de relations binaires distinctes sur E_n pourrait-on générer ?

Exercice 2 : matrices d'adjacences

Soit une relation binaire sur E_n codée par une liste de tuples `rel`. La fonction `mat_adj(n,rel)` ci-dessous crée la matrice d'adjacence associée.

```python
def mat_adj(n, rel):

    A = matrix(n) # what are we doing?
    for x in range(n):
        for y in range(n): # what is the purpose of this loop?
            if (x,y) in rel:
                A[x,y] = 1 # what are we doing?
    return A
```

1. Recopier le code et le commenter. Comprenez la sortie pour `rel = [(1,2), (4,3)]` et `n = 6`.
2. Soient G_A et G_B les graphes de deux relations binaires $\mathcal{R}_A$ et $\mathcal{R}_B$ sur E_n. On note `A` et `B` les matrices d'adjacence associées.
   (a) Implémenter la fonction `oplus(A,B)` qui calcule la matrice d'adjacence de la relation $\mathcal{R}$ dont le graphe est $G_A \cup G_B$. Il s'agit du $\oplus$ du cours. Tester par exemple :

   ```python
   # two relations on E10
   relA = [(0,0), (1,2), (2,4), (3,6), (4,8)] # n RA m \iff m = 2n
   relB = [(0,0), (1,3), (2,6), (3,9)] # n RB m \iff m = 3n
   A = mat_adj(10, relA)
   B = mat_adj(10, relB)
   print(oplus(A,B))
   ```

   (b) Implémenter la fonction `otimes(A,B)` qui calcule la matrice d'adjacence de la relation produit $\mathcal{R}_A\mathcal{R}_B$. C'est le $\otimes$ du cours. Tester par exemple :

   ```python
   # two relations on E10
   relA = [(0,0), (1,2), (2,4), (3,6), (4,8)] # n RA m \iff m = 2n
   relB = [(0,0), (1,3), (2,6), (3,9)] # n RB m \iff m = 3n
   A = mat_adj(10, relA)
   B = mat_adj(10, relB)
   print(otimes(A,B)) # two bits at (0,0) and (1,6)
   ```

   (c) Implémenter la fonction `is_contained(A,B)` retournant `True` si le graphe associé à la matrice `A` est contenu dans le graphe associée à la matrice `B`.

   ```python
   # two relations on E10
   relA = [(0,0), (1,2), (2,4), (3,6), (4,8)]
   relB = [(0,0), (1,2), (4,8)]
   A = mat_adj(10, relA)
   B = mat_adj(10, relB)
   print(is_contained(A,B))
   print(is_contained(B,A))
   ```

3. Avec des considérations sur les matrices d'adjacence, dénombrer à la main puis vérifier à l'ordinateur le nombre de relations binaires sur E_n réflexives. Même questions avec symétriques, antisymétriques et transitives.

Exercice 3 : relation binaire aléatoire

On commencera par fixer la graine pour la génération *pseudo-aléatoire* de nombres.

```python
set_random_seed(0)
```

1. Générer une relation binaire aléatoire $\mathcal{Q}$ sur E_{40} de la façon suivante : pour chaque paire $(x, y) \in E_{40}^2$, la probabilité que $x\mathcal{Q}y$ est $\frac{1}{9}$.
2. La relation générée est-elle réflexive ? Sinon, la rendre réflexive et obtenir ainsi une relation $\mathcal{R}$.

3. Cette nouvelle relation est-elle symétrique ? Sinon, la rendre symétrique et obtenir ainsi une relation $\mathcal{S}$. (Vérifier qu'elle est toujours réflexive...)
4. Cette nouvelle relation est-elle transitive ? Sinon, la rendre transitive et obtenir ainsi une relation $\mathcal{T}$. (Vérifier qu'elle est restée réflexive et symétrique...)
5. La relation $\mathcal{T}$ devrait maintenant être une relation d'équivalence sur E_{40}. Si on note `T` la matrice d'adjacence associée, commenter la visualisation fournie par :

```python
import matplotlib.pyplot as plt # may require a pip install of matplotlib
plt.imshow(T, cmap='binary')
```

6. Faire varier la probabilité p que $x\mathcal{Q}y$ dans la relation initiale pour voir l'incidence que cela a sur le résultat. En particulier : quel est le seuil de probabilité p_n qu'il faut choisir pour être quasi-certain que la relation générée aléatoirement puis rendue équivalente soit complète [1] ?

Exercice 4 : relation d'équivalence

Soient $\mathcal{R}$ et $\mathcal{S}$ deux relations binaires sur E_n. On dit que $\mathcal{R}$ et $\mathcal{S}$ sont *transitivement équivalentes* si et seulement si elles ont la même fermeture transitive. Formellement :

$$\boxed{\mathcal{R} \sim \mathcal{S} \iff \forall \mathcal{T} \text{ transitive, } \mathcal{R} \subset \mathcal{T} \iff \mathcal{S} \subset \mathcal{T}}$$

1. Se convaincre à la main que $\sim$ est une relation d'équivalence sur l'ensemble des relations binaires sur E_n.
2. Décrire informatiquement les classes d'équivalences de $\sim$ dans le cas de $E_3 = \{0, 1, 2\}$. Combien de relations sont transitivement équivalentes à la relation d'équivalence complète *i.e.* celle où tout en relation avec tout ?

1. Tout est en relation avec tout