02_Transitivite

November 8, 2021

1 Transitivité

Marc Lorenzi

3 novembre 2021

[1]:
```
import matplotlib.pyplot as plt
import math
```

Nous allons dans ce notebook discuter du concept de *transitivité*. Nous supposons que le lecteur a lu le premier notebook sur les *ensembles héréditairement finis*. Rappelons que ces ensembles sont les éléments de l'ensemble $V = \bigcup_{n\in\mathbb{N}} V_n$ où $V_0 = \emptyset$ et pour tout $n \in \mathbb{N}$, $V_{n+1} = \mathcal{P}(V_n)$.

Le module `hf0` contient les fonctions du premier notebook.

[2]:
```
from hf01 import *
```

1.1 1. Introduction

1.1.1 1.1 Notion d'ensemble transitif

Définition. Soit A un ensemble. A est *transitif* si pour tout $x \in A$, $x \subseteq A$.

Dit autrement, A est transitif si pour tout $x \in A$, pour tout $y \in x$, $y \in A$.

La fonction `est_transitif` prend en paramètre un ensemble $A \in V$. Elle renvoie `True` si A est transitif et `False` sinon.

[3]:
```
def est_transitif(A):
    for x in A:
        for y in x:
            if y not in A: return False
    return True
```

[4]:
```
est_transitif(neumann(20))
```

[4]: True

[5]:
```
est_transitif(tau(20))
```

[5]:
```
True
```

[6]:
```
est_transitif(psi(123456))
```

[6]:
```
False
```

Parmi les 100 premiers ensembles de V, lesquels sont transitifs ?

[7]:
```
for A in HF([], 100):
    if est_transitif(A): print(phi(A), tostring(A))
```

```
0 0
1 1
3 2
7  3
11 3
15 {0,1, 2,2}
23  4
31 {0,1, 2,2, 3}
39 {0,1, 2,{0, 2}}
47 {0,1, 2,2,{0, 2}}
55 {0,1, 2, 3,{0, 2}}
63 {0,1, 2,2, 3,{0, 2}}
71 {0,1, 2,{1, 2}}
79 {0,1, 2,2,{1, 2}}
87 {0,1, 2, 3,{1, 2}}
95 {0,1, 2,2, 3,{1, 2}}
```

1.1.2 1.2 Quelques ensembles transitifs

Proposition. Pour tout $n \in \mathbb{N}$, V_n est transitif.

Démonstration. Nous avons déjà vu au paragraphe 1.1 du premier notebook que pour tout $A \in V_n$, $A \subseteq V_n$, d'où le résultat. □

Proposition. V est transitif.

Démonstration. Soit $A \in V$. Il existe $n \in \mathbb{N}$ tel que $A \in V_n$. Comme V_n est transitif, $A \subseteq V_n$. Or, $V_n \subseteq V$, donc $A \subseteq V$. □

Proposition. Les entiers de von Neumann sont transitifs.

Démonstration. Soit $n \in \mathbb{N}$. On a

$$\overline{n} = \{\overline{0}, \ldots, \overline{n-1}\}$$

Tout élément de $\overline{n}$ est de la forme $x = \overline{k}$ où $k \leq n-1$. On a alors

$$x = \{\overline{0}, \ldots, \overline{k-1}\} \subseteq \overline{n} \ \square$$

Proposition. Pour tout $n \in \mathbb{N}$, τ_n est transitif.

Démonstration. Identique à celle faite pour les entiers de von Neumann. □

1.1.3 1.3 Compter les ensembles transitifs

Parmi les n premiers ensembles héréditairement finis, combien sont transitifs ?

La fonction `stats_trans` prend en paramètre un entier N. Elle renvoie une liste de taille n_{max}. Le $k$ième élément de cette liste est le nombre d'ensembles transitifs A tels que $\varphi(A) < 2^k$.

[8]:
```
def stats_trans(nmax):
    stats = [1]
    A = []
    for k in range(nmax - 1):
        s = 0
        for n in range(2 ** k, 2 ** (k + 1)):
            A = succ(A)
            if est_transitif(A): s += 1
        stats.append(stats[-1] + s)
    return stats
```

Pour l'évaluation de la cellule suivante, soyons patients, cela prend environ une minute …

[9]:
```
stats25 = stats_trans(25)
print(stats25)
```

```
[1, 2, 3, 4, 6, 8, 12, 20, 36, 53, 87, 155, 291, 547, 1059, 2083, 4131, 6187,
10299, 18523, 34971, 67867, 133659, 265243, 528411]
```

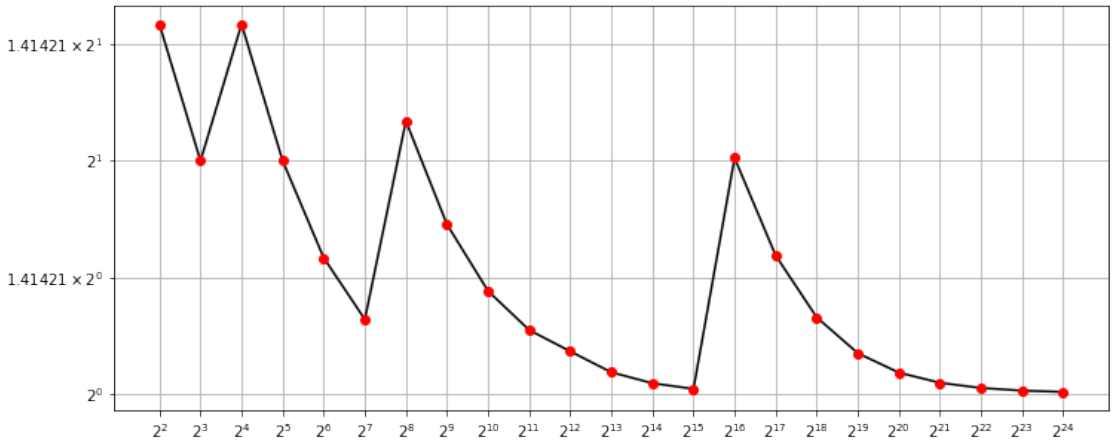
Soit T_n le nombre d'ensembles transitifs parmi les n premiers ensembles de V. posons

$$h(n) = \frac{n}{2^{\lfloor \lg \lg n \rfloor + 1}}$$

où $\lg n$ est le logarithme de n en base 2. Traçons tout à fait au hasard $T_n \,/\, h(n)$ en fonction de n.

[10]:
```
def lg(n): return math.log(n, 2)
def h(n): return n / (2 * 2 ** math.floor(lg(lg(n))))
```

[11]:
```
plt.rcParams['figure.figsize'] = (12, 5)
ns = [2 ** k for k in range(2, 25)]
sts = [stats25[k] / h(2 ** k) for k in range(2, 25)]
plt.loglog(ns, sts, 'k', base=2)
plt.loglog(ns, sts, 'or', base=2)
plt.xticks(ns)
plt.yticks([2 ** (k / 2) for k in range(0, 4)])
plt.grid()
```



On obtient quelque chose de fort intéressant. Une étude complémentaire semble nécessaire.

Proposition. Soit $n \geq 1$. Soit $\ell = \lfloor \lg n \rfloor$. Il existe au moins $2^{n-\ell-1}$ ensembles transitifs A tels que $\varphi(A) < 2^n$.

Démonstration. Soit $B = \{\psi(0), \ldots, \psi(\ell)\}$. Soit $A = B \cup A'$ où $A' \subseteq \{\psi(\ell+1), \ldots, \psi(n-1)\}$. On a alors $A \subseteq \{\psi(0), \ldots, \psi(n-1)\}$ et donc

$$\varphi(A) = \sum_{x \in A} 2^{\varphi(x)} \leq \sum_{k=0}^{n-1} 2^k = 2^n - 1 < 2^n$$

Il existe $2^{n-\ell-1}$ choix possibles pour A', et donc aussi pour A. Montrons maintenant que A est transitif. Soit $x \in A$. On a $2^{\varphi(x)} \leq \varphi(A) < 2^n$ et donc $\varphi(x) < n$. Soit $y \in x$. On a $2^{\varphi(y)} \leq \varphi(x)$ et donc $\varphi(y) \leq \lg \varphi(x) < \lg n$, d'où $\varphi(y) \leq \ell$. Ainsi, $y \in \psi(l+1)\!\downarrow \subseteq A$, d'où $y \in A$. □

Dans le cas où n est une puissance de 2, le résultat précédent peut être amélioré (en fait, doublé).

Proposition. Soit $n \geq 0$. Il existe au moins $2^{2^n - n}$ ensembles transitifs A tels que $\varphi(A) < 2^{2^n}$.

Démonstration. Soit $B = \{\psi(0), \ldots, \psi(n-1)\}$. Soit $A = B \cup A'$ où $A' \subseteq \{\psi(n), \ldots, \psi(2^n - 1)\}$. On a alors $A \subseteq \{\psi(0), \ldots, \psi(2^n - 1)\}$ et donc

$$\varphi(A) = \sum_{x \in A} 2^{\varphi(x)} \leq \sum_{k=0}^{2^n-1} 2^k = 2^{2^n} - 1 < 2^{2^n}$$

Il existe $2^{2^n - n}$ choix possibles pour A', et donc aussi pour A. Montrons maintenant que A est transitif. Soit $x \in A$. On a $2^{\varphi(x)} \leq \varphi(A) < 2^{2^n}$ et donc $\varphi(x) < 2^n$. Soit $y \in x$. On a $2^{\varphi(y)} \leq \varphi(x) < 2^n$ et donc $\varphi(y) < n$. Ainsi, $y \in B \subseteq A$, d'où $y \in A$. □

Au vu de ces résultats, si nous reprenons la courbe vue un peu plus haut, nous constatons effectivement que

- Les ordonnées des points sont toutes au moins égales à 1.
- L'ordonnée des points d'abscisses 2^{2^2}, 2^{2^3} et 2^{2^4} est supérieure ou égale à 2.

Remarque. Soit $n \geq 2$. Combien y a-t-il d'ensembles transitifs dans V_n ? Pour tout $A \in V$, on a $A \in V_n$ si et seulement si $\varphi(A) < 2^{\uparrow n}$. Or,

$$2^{\uparrow n} = 2^{2^{2^{\uparrow n-2}}} = 2^{2^m}$$

où $m = 2^{\uparrow n-2}$. Il y a donc dans V_n au moins 2^{2^m-m} ensembles transitifs. En simplifiant,

$$2^{2^m-m} = \frac{2^{2^m}}{2^m} = \frac{2^{\uparrow n}}{2^{\uparrow n-1}}$$

1.1.4 1.4 Transitivité, rang et cardinal

Proposition. Soit $n \in \mathbb{N}$. Soit $A \in V$ un ensemble transitif de rang n. Alors, pour tout $k \in [\![0, n-1]\!]$, A possède un élément de rang k.

Démonstration. Faisons une récurrence « descendante » sur k.

- Comme A est de rang n, A possède un élément de rang $n-1$.
- Soit $k \in [\![1, n-1]\!]$. Supposons que A possède un élément x de rang k. Comme A est transitif, $x \subseteq A$. Or, x étant de rang k, x possède un élément y de rang $k-1$. Comme $x \subseteq A$, $y \in A$. A possède donc un élément de rang $k-1$. □

Corollaire. Soit A un ensemble transitif. On a $|A| \geq \text{rg } A$.

Démonstration. Soit $n = \text{rg } A$. Par la proposition précente, pour tout $k \in [0, n-1]$, A possède un élément x_k de rang k. Les x_k ayant des rangs distincts, ils sont distincts. A possède donc au moins n éléments. □

Une question se pose : pour un entier n donné, combien y a-t-il d'ensembles de rang n et de cardinal n ?

Proposition. Pour tout $n \in \mathbb{N}$, il existe $2^{\frac{1}{2}(n-1)(n-2)}$ ensembles transitifs de rang n et de cardinal n.

Démonstration. Faisons une récurrence sur n. Quelques détails sont laissés au lecteur.

- Pour $n = 0$, c'est immédiat.
- Soit $n \in \mathbb{N}$. Supposons la propriété vérifiée pour n. Soit A un ensemble transitif de rang $n+1$ et cardinal $n+1$. Pour tout $k \in [0, n]$, A possède un élément de rang k, donc, puisque $|A| = n+1$, A possède exactement un élément de rang k. Soit x l'unique élément de A de rang n. Soit $A' = A \setminus \{x\}$. L'ensemble A' est transitif (exercice), de rang n et de cardinal n. Il y a donc $2^{\frac{1}{2}(n-1)(n-2)}$ ensembles A' possibles. Que vaut x ? Par la transitivité de A, $x \subseteq A$. De plus, $x \notin x$, donc $x \subseteq A'$. Comme rg $x = n$, x doit contenir l'unique élément de A' de rang $n-1$. Cette condition est de plus suffisante (exercice). Il reste à choisir parmi les $n-1$ autres éléments de A' ceux qui appartiennent à x, ce qui donne 2^{n-1} possibilités pour x. Le nombre total de possibilités pour l'ensemble A est donc

$$2^{\frac{1}{2}(n-1)(n-2)} 2^{n-1} = 2^{\frac{1}{2}(n-1)(n-2)+(n-1)} = 2^{\frac{1}{2}(n-1)n} \quad \square$$

Faisons une petite simulation en testant sur les 1 million premiers ensembles héréditairement finis. Nous devrions trouver :

- 1 ensemble de rang 0.
- 1 ensemble de rang 1.
- 1 ensemble de rang 2.
- 2 ensembles de rang 3.
- 8 ensembles de rang 4.
- Quelques ensembles de rang 5 parmi les 64 ensembles transitifs de rang 5 et de cardinal 5.

[12]:
```
for A in HF([], 10 ** 6):
    if est_transitif(A) and rang(A) == len(A):
        print('%8d%5d %s' % (phi(A), rang(A), tostring(A)))
```

```
       0    0 0
       1    1 1
       3    2 2
       7    3 3
      11    3 3
      23    4 4
      39    4 {0,1, 2,{0, 2}}
      71    4 {0,1, 2,{1, 2}}
     135    4 {0,1, 2, 3}
     267    4 {0,1,2,{2}}
     523    4 {0,1,2,{0,2}}
    1035    4 {0,1,2,<0,1>}
    2059    4 4
   65559    5 5
  131095    5 {0,1, 2, 3,{0, 3}}
  262167    5 {0,1, 2, 3,{1, 3}}
  524311    5 {0,1, 2, 3,{0,1, 3}}
```

Nous obtenons en prime 4 ensembles de rang 5 sur les 64 ensembles transitifs de rang 5 et de cardinal 5. Inutile d'espérer obtenir ainsi tous les ensembles transitifs de rang 5 et de cardinal 5 …

[13]:
```
print(phi(neumann(5)))
```

```
66185228434044942951864067458396061614989522267577311297802947435570493724401440
54926786849079892677363449438396804714392395685714020540640274053608744608383105
20368482324399959044049927980075147183260434105703798308704637800852606194444172
05199197123751210704970352727833755425876102776028267313405809429548880554782040
76527756282836288423832546544852034830757494334599030994164266692672337972959818
58347350547325004154098838683614231599137708122187727119017722495531534022877597
89517121744336755350465901655205184917370974202405586941211065395540765567663193
29717336725423031361224418294199950040238819545005308038554 7
```

Pouvons nous *calculer* les ensembles de rang n et de cardinal n ? Oui, il suffit de reprendre les idées de la démonstration de la proposition ci-dessus. La fonction `transitifsnn` prend en paramètre un entier n. Elle renvoie la liste des ensembles transitifs de rang n et de cardinal n.

[14]:
```
def transitifsnn(n):
    if n == 0: return [[]]
    elif n == 1: return [[[]]]
    else:
        T1 = transitifsnn(n - 1)
        T = []
        for A1 in T1:
            t = A1[-1]
            BS = parties(A1[:-1])
            for B in BS:
                T.append(A1 + [reunion(B, [t])])
        return T
```

Voici les 64 ensembles transitifs de rang 5 et de cardinal 5. On affiche leur image par φ si elle est inférieure à 10^{20}.

[15]:
```
for A in transitifsnn(5):
    if phi(A) < 10 ** 20:
        print('%20s %s' % (phi(A), tostring(A)))
    else:
        print('%20s %s' % (' ', tostring(A)))
```

```
               65559 5
              131095 {0,1, 2, 3,{0, 3}}
              262167 {0,1, 2, 3,{1, 3}}
              524311 {0,1, 2, 3,{0,1, 3}}
             1048599 {0,1, 2, 3,{ 2, 3}}
             2097175 {0,1, 2, 3,{0, 2, 3}}
             4194327 {0,1, 2, 3,{1, 2, 3}}
             8388631 {0,1, 2, 3, 4}
          4294967335 {0,1, 2,{0, 2},{{0, 2}}}
          8589934631 {0,1, 2,{0, 2},{0,{0, 2}}}
         17179869223 {0,1, 2,{0, 2},<0, 2>}
         34359738407 {0,1, 2,{0, 2},{0,1,{0, 2}}}
         68719476775 {0,1, 2,{0, 2},{ 2,{0, 2}}}
        137438953511 {0,1, 2,{0, 2},{0, 2,{0, 2}}}
        274877906983 {0,1, 2,{0, 2},{1, 2,{0, 2}}}
        549755813927 {0,1, 2,{0, 2},{0,1, 2,{0, 2}}}
1844674407370955168 7 {0,1, 2,{1, 2},{{1, 2}}}
3689348814741910330 3 {0,1, 2,{1, 2},{0,{1, 2}}}
7378697629483820653 5 {0,1, 2,{1, 2},{1,{1, 2}}}
                      {0,1, 2,{1, 2},{0,1,{1, 2}}}
                      {0,1, 2,{1, 2},<1, 2>}
                      {0,1, 2,{1, 2},{0, 2,{1, 2}}}
                      {0,1, 2,{1, 2},{1, 2,{1, 2}}}
                      {0,1, 2,{1, 2},{0,1, 2,{1, 2}}}
                      {0,1, 2, 3,{ 3}}
                      {0,1, 2, 3,{0, 3}}
```

```
{0,1, 2, 3,{1, 3}}
{0,1, 2, 3,{0,1, 3}}
{0,1, 2, 3,{ 2, 3}}
{0,1, 2, 3,{0, 2, 3}}
{0,1, 2, 3,{1, 2, 3}}
{0,1, 2, 3,{0,1, 2, 3}}
{0,1,2,{2},<2,2>}
{0,1,2,{2},{0,{2}}}
{0,1,2,{2},{1,{2}}}
{0,1,2,{2},{0,1,{2}}}
{0,1,2,{2},{2,{2}}}
{0,1,2,{2},{0,2,{2}}}
{0,1,2,{2},{1,2,{2}}}
{0,1,2,{2},{0,1,2,{2}}}
{0,1,2,{0,2},{{0,2}}}
{0,1,2,{0,2},{0,{0,2}}}
{0,1,2,{0,2},<0,2>}
{0,1,2,{0,2},{0,1,{0,2}}}
{0,1,2,{0,2},{2,{0,2}}}
{0,1,2,{0,2},{0,2,{0,2}}}
{0,1,2,{0,2},{1,2,{0,2}}}
{0,1,2,{0,2},{0,1,2,{0,2}}}
{0,1,2,<0,1>,{<0,1>}}
{0,1,2,<0,1>,{0,<0,1>}}
{0,1,2,<0,1>,{1,<0,1>}}
{0,1,2,<0,1>,{0,1,<0,1>}}
{0,1,2,<0,1>,{2,<0,1>}}
{0,1,2,<0,1>,{0,2,<0,1>}}
{0,1,2,<0,1>,{1,2,<0,1>}}
{0,1,2,<0,1>,{0,1,2,<0,1>}}
{0,1,2,3,{3}}
{0,1,2,3,{0,3}}
{0,1,2,3,{1,3}}
{0,1,2,3,{0,1,3}}
{0,1,2,3,{2,3}}
{0,1,2,3,{0,2,3}}
{0,1,2,3,{1,2,3}}
5
```

Le lecteur courageux listera les 1024 ensembles transitifs de rang 6 et de cardinal 6. Nous nous contenterons d'afficher leur nombre …

[16]:
```
len(transitifsnn(6))
```

[16]: 1024

Au delà du courage, il y a la témérité.

[17]:
```
len(transitifsnn(7))
```

[17]: 32768

Et enfin l'inconscience …

[18]:
```
print(2 ** ((8 - 1) * (8 - 2) // 2))
```

2097152

Exercice. Certains auront remarqué que nos listes d'ensembles transitifs de rang n et de cardinal n commencent par τ_n et finissent par $\overline{n}$. Prouver le résultat suivant :

Proposition. Soit $n \in \mathbb{N}$. Soit A un ensemble transitif de rang n et de cardinal n. Alors,

$$\tau_n \leq A \leq \overline{n}$$

1.2 2. Bitransitivité

La transitivité n'est pas héréditaire : les éléments d'un ensemble transitif n'ont aucune raison d'être eux-mêmes transitifs. Ceci suggère une définition.

Définition. Soit $A \in V$. L'ensemble A est *bitransitif* si A est transitif et tous ses éléments sont aussi transitifs.

Par exemple, les entiers de von Neumann sont bitransitifs. Quel est l'intérêt de la bitransitivité ? Nous avons déjà vu que la relation $\in$ est irréflexive et asymétrique sur V. Il ne lui manque que la transitivité pour être une relation d'ordre strict.

Proposition. Soit $A \in V$ un ensemble transitif. La relation $\in$ est transitive sur A si et seulement si A est bitransitif.

Démonstration. Supposons A bitransitif. Soient $x, y, z \in A$. Supposons $x \in y$ et $y \in z$. Comme z est transitif, on a $x \in z$. La relation $\in$ est donc transitive. Inversement, supposons que la relation $\in$ est transitive sur A. Soit $z \in A$. Soit $y \in z$. Soit $x \in y$. Comme A est transitif, $z \subseteq A$ et donc $y \in A$. En répétant l'argument, $x \in A$. Par la transitivité de la relation $\in$ sur A, on a $x \in z$. Ainsi, pour tout $y \in z$, $y \subseteq z$ et donc z est transitif. □

[19]:
```
def est_bitransitif(A):
    if not est_transitif(A): return False
    for x in A:
        if not est_transitif(x): return False
    return True
```

Voici, parmi les 10000 premiers ensembles de V, ceux qui sont bitransitifs. On regarde à chaque fois si ce sont des entiers de von Neumann.

[20]:
```
print(' (A) rg A |A| VN?  A')
for A in HF([], 10000):
    n = rang(A)
```

```
    if est_bitransitif(A): print('%4d%4d%4d%6s %s' % (phi(A), rang(A), len(A), A␣
↪== neumann(n), tostring(A)))
```

```
(A) rg A |A| VN?  A
  0   0   0  True 0
  1   1   1  True 1
  3   2   2  True 2
 11   3   3  True 3
2059   4   4  True 4
```

Nous n'obtenons que des entiers de von Neumann. Est-ce normal ? Oui.

Proposition. Soit $n \in \mathbb{N}$. Soit A un ensemble bitransitif de cardinal n. Alors $A = \overline{n}$.

Démonstration. Montrons-le par récurrence sur n.

- C'est évident si $n = 0$.
- Soit $n \in \mathbb{N}^*$. Supposons que pour tout $k < n$, le seul ensemble bitransitif de cardinal k est $\overline{k}$. Soit A un ensemble bitransitif de cardinal n. Soit x un élément de A. Comme A est bitransitif, x l'est aussi. Par la transitivité de A, $x \subseteq A$ et donc $|x| \leq |A|$. De plus, $x \notin x$ alors que $x \in A$, donc $|x| < |A|$. Par l'hypothèse de récurrence, il existe $k \in \mathbb{N}$ tel que $x = \overline{k}$. De plus, $k = |x| < n$. Tous les éléments de A sont ainsi des entiers de von Neumann : il existe $k_1, \ldots, k_n \in [0, n-1]$ distincts deux à deux tels que

$$A = \{\overline{k_1}, \ldots, \overline{k_n}\}$$

Remarquons que

$$\overline{n} = \{\overline{0}, \ldots, \overline{n-1}\}$$

On a $A \subseteq \overline{n}$ et ces deux ensembles ont le même cardinal, n. Ainsi, $A = \overline{n}$. □

Remarquons que le numéro d'ordre de l'entier de von Neumann suivant, $\overline{5}$, est bien trop grand pour être atteint par une énumération. Par contre, rien ne nous empêche de demander à Python si $\overline{5}$ ou $\overline{100}$ est bitransitif.

[21]:
```
est_bitransitif(neumann(100))
```

[21]:
```
True
```

1.3 3. Opérations sur les ensembles transitifs

1.3.1 3.1 Réunion

Proposition. Soient A et B deux ensembles transitifs. Alors $A \cup B$ est transitif.

Démonstration. Soit $x \in A \cup B$. Si $x \in A$ alors, par transitivité de A, $x \subseteq A \subseteq A \cup B$. De même si $x \in B$.

Corollaire. Soit $n \in \mathbb{N}$. Soient $A_1, \ldots, A_n \in V$ des ensembles transitifs. Alors $\bigcup_{k=1}^{n} A_k \in V$ et est transitif.

Démonstration. Nous avons vu dans le premier notebook que la réunion de deux éléments de V est encore dans V. Le résultat s'en suit par la proposition précédente et une récurrence sur n.

1.3.2 3.2 Intersection

Proposition. Soient A et B deux ensembles transitifs. Alors $A \cap B$ est transitif.

Démonstration. Soit $x \in A \cap B$. On a $x \in A$ donc, par transitivité de A, $x \subseteq A$. De même, $x \subseteq B$, donc $x \subseteq A \cap B$.

Corollaire. Soit $n \in \mathbb{N}$. Soient $A_1, \ldots, A_n \in V$ des ensembles transitifs. Alors $\bigcap_{k=1}^{n} A_k \in V$ et est transitif.

Démonstration. Nous avons vu dans le premier notebook que l'intersection de deux éléments de V est encore dans V. Le résultat s'en suit par la proposition précédente et une récurrence sur n.

1.3.3 3.3 Parties

Proposition. Soit A un ensemble transitif. Alors, $\mathcal{P}(A)$ est transitif.

Démonstration. Soit $X \in \mathcal{P}(A)$. Soit $Y \in X$. Puisque $X \subseteq A$, on a $Y \in A$. Par la transitivité de A, $Y \subseteq A$ et donc $Y \in \mathcal{P}(A)$.

Corollaire. Soit $A \in V$ un ensemble transitif. Alors $\mathcal{P}(A) \in V$ et $\mathcal{P}(A)$ est transitif.

1.3.4 3.4 Produit cartésien

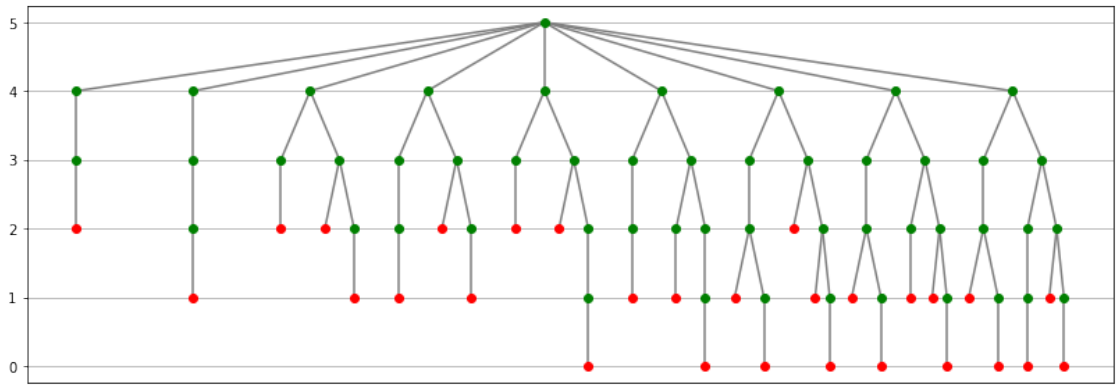
Un produit cartésien n'est *jamais* transitif. En effet, un ensemble transitif continent $\emptyset$, et $\emptyset$ n'est pas un couple.

[22]:
```
A = produit(neumann(3), tau(3))
print(est_transitif(A))
print(tostring(A))
```

```
False
{ 2, 3,<0,1>,<1,0>,<0, 2>,<1, 2>,<2,0>,<2,1>,<2, 2>}
```

Par exemple, $\{1, 2\} \in A$ alors que $1 \notin A$ (et 2 non plus).

[23]:
```
tracer_arbre(A)
```



1.4 4. Clôture transitive

1.4.1 4.1 C'est quoi ?

Notons $\mathcal{T}$ l'ensemble des ensembles transitifs héréditairement finis.

Soit $A \in V$. Soit $n = \text{rg } A$. On a alors $A \in V_{n+1}$ et $V_{n+1} \in \mathcal{T}$. Considérons

$$\overline{A} = \bigcap_{X \in \mathcal{T}, A \subseteq X} X$$

Proposition. $\overline{A}$ est, au sens de l'inclusion, le plus petit ensemble de $\mathcal{T}$ contenant A.

Démonstratoin. $\overline{A}$ est l'intersection d'un ensemble non vide d'ensemble transitifs. C'est donc un ensemble transitif. De plus, clairement, $A \subseteq \overline{A}$. Enfin, tout ensemble transitif qui contient A apparaît dans l'intersection qui définit $\overline{A}$, et contient donc $\overline{A}$.

Définition. $\overline{A}$ est la *clôture transitive* de A.

1.4.2 4.2 Calcul de la clôture transitive

Définition. Pour tout ensemble $A \in V$, *l'union* de A est l'ensemble

$$\bigcup A = \bigcup_{x \in A} x$$

Proposition. Soit $A \in V$. Alors $\bigcup A \in V$. Si $\text{rg } A \geq 1$, alors $\text{rg } \bigcup A = \text{rg } A - 1$.

Démonstration. Pour tout $x \in A$, $x \in V$. $\bigcup A$, union finie d'ensembles héréditairement finis, est donc un élément de V. De plus,

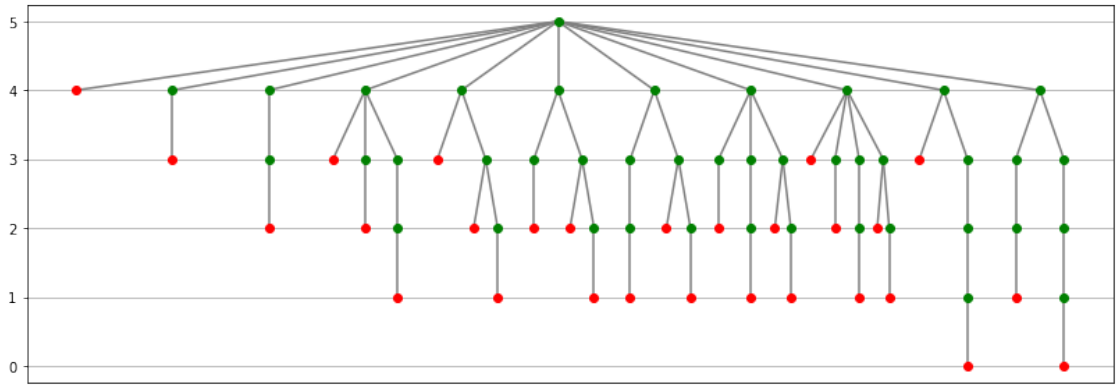
$$\text{rg } \bigcup A = \text{rg } \bigcup_{x \in A} x = \max\{\text{rg } x : x \in A\} = \text{rg } A - 1 \ \square$$

La fonction `union_ensemble` prend en paramètre un ensemble $A \in V$. Elle renvoie $\bigcup A$.

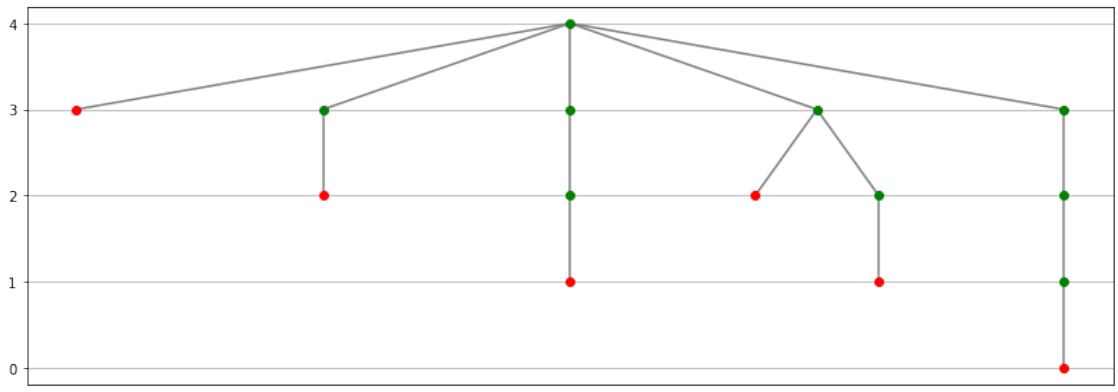
[24]:
```
def union_ensemble(A):
    B = []
    for x in A:
        B = reunion(B, x)
    return B
```

Faisons un petit test. Prenons un ensemble $A \in V$. Calculons $\bigcup A$, $\bigcup\bigcup A$, $\bigcup\bigcup\bigcup A$, etc.

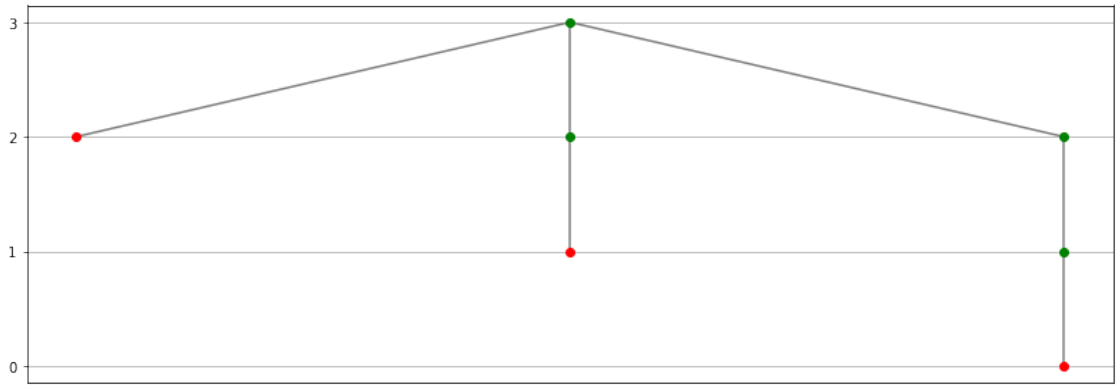
[25]:
```
A = psi(1234567)
tracer_arbre(A)
```



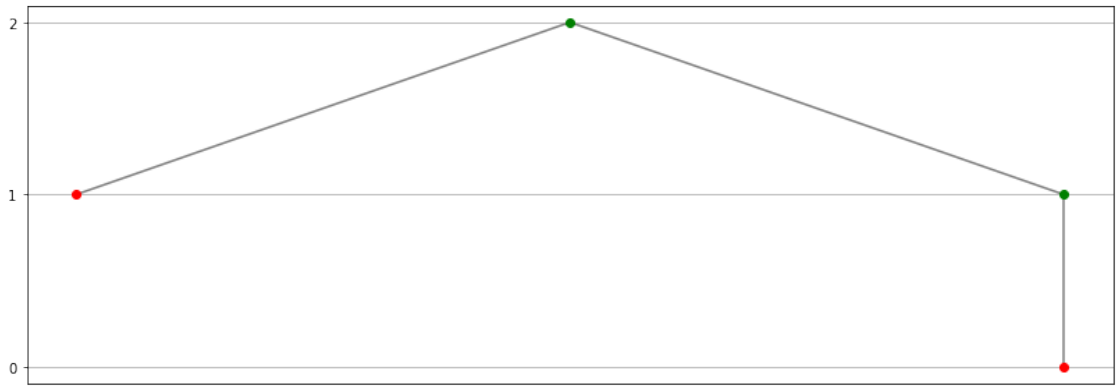
[26]:
```
A = union_ensemble(A)
tracer_arbre(A)
```



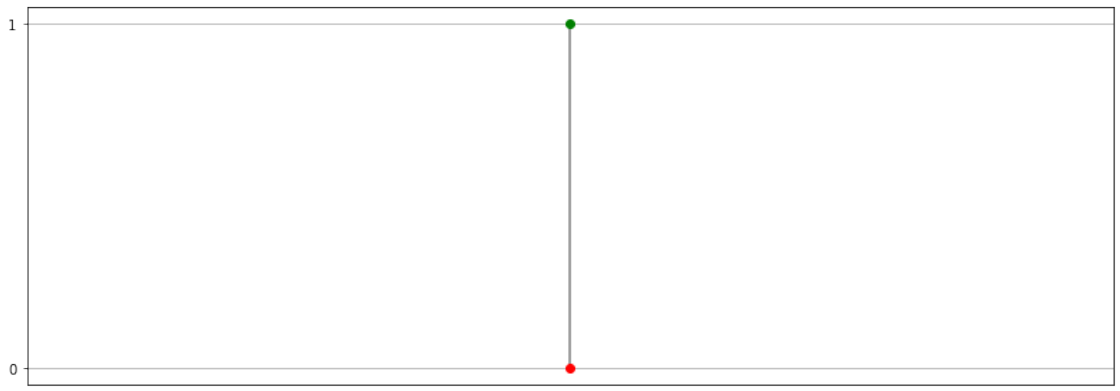
[27]:
```
A = union_ensemble(A)
tracer_arbre(A)
```



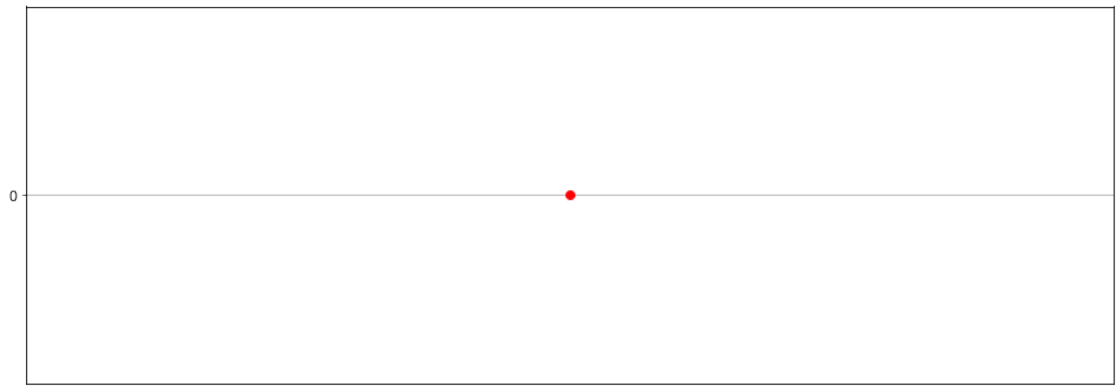
[28]:
```
A = union_ensemble(A)
tracer_arbre(A)
```



[29]:
```
A = union_ensemble(A)
tracer_arbre(A)
```



[30]:
```
A = union_ensemble(A)
tracer_arbre(A)
```



On a compris … Soit $A \in V$. Définissons par récurrence sur n une suite $(A_n)_{n\in\mathbb{N}}$ d'ensembles en posant $A_0 = A$ puis, pour tout $n \in \mathbb{N}$,

$$A_{n+1} = \bigcup A_n$$

Proposition. Soit r le rang de A. On a pour tout $k \in [0, r]$, rg $A_k = r - k$.

Démonstration. En effet, rg $A_{k+1} =$ rg $A_k - 1$, d'où le résultat par une récurrence immédiate. □

En conséquence, rg $A_r = 0$ et donc, facilement, pour tout $k \geq r$, $A_k = \emptyset$. Posons

$$C = \bigcup_{k=0}^{r-1} A_k = \bigcup_{k\in\mathbb{N}} A_k$$

Proposition. On a $C = \overline{A}$.

Démonstration. Soit $x \in C$. Il existe $k \in [\![0, r-1]\!]$ tel que $x \in A_k$. Soit $y \in x$. Alors $y \in A_{k+1}$, donc $y \in C$. Ainsi, C est transitif.

Soit T un ensemble transitif contenant A. Montrons par récurrence sur k que pour tout $k \in \mathbb{N}$, $A_k \subseteq T$.

- C'est vrai par hypothèse pour $k = 0$.
- Soit $k \in \mathbb{N}$. Supposons que $A_k \subseteq T$. Soit $y \in A_{k+1} = \bigcup A_k$. Il existe donc $x \in A_k$ tel que $y \in x$. Par l'hypothèse de récurrence, $x \in T$. Comme T est transitif et $y \in x$, on a aussi $y \in T$. Ainsi, $A_{k+1} \subseteq T$. □

Tous les A_k sont inclus dans T donc leur réunion C l'est aussi. C est donc le plus petit ensemble transitif contenant A. □

La fonction `cloture` prend en paramètre un ensemble $A \in V$. Elle renvoie $\overline{A}$. Plutôt que d'effectuer une boucle `for`, elle effectue une boucle `while` pour calculer les ensembles A_k, et sort de cette boucle dès qu'un tel ensemble est vide. Notre théorie montre que cette boucle s'exécute au plus rg $A - 1$ fois.

[31]:
```
def cloture(A):
    C = []
    E = A
    while E != []:
        C = reunion(C, E)
        E = union_ensemble(E)
    return C
```

La clôture transitive d'un ensemble A, même si A a un petit cardinal, peut être un énorme ensemble. Par exemple, si $A = \{V_n\}$ (et est donc de cardinal 1), $\bigcup A = V_n$ est de cardinal $2^{\uparrow n}$. Cela dit, si nous sommes capables de passer un ensemble A en paramètre à la fonction `cloture`, c'est que Python a pu calculer A. La clôture transitive de A est l'ensemble des éléments, des éléments des éléments, etc. de A, et tous ces objets sont déjà stockés dans la représentation de A. Une estimation très grossière est que si A est stocké sur n bits, sa clôture transitive ne demandera pas plus de n^2 bits pour être stockée.

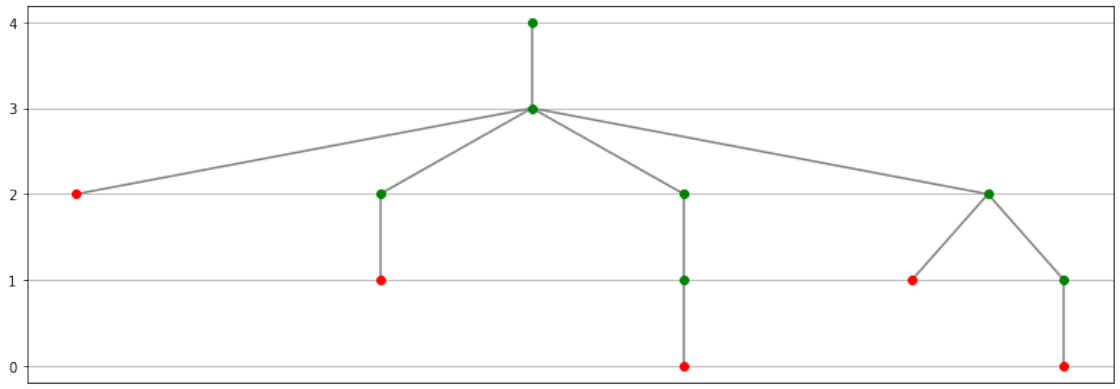
[32]:
```
A = psi(123456)
print(tostring(A))
print(len(A), est_transitif(A))
C = cloture(A)
print(tostring(C))
print(len(C), est_transitif(C))
```

```
{{1, 2},{0,2},{0, 2,2},{1, 2,2},{0,1, 2,2}, 4}
6 False
{0,1, 2,2, 3,{1, 2},{0,2},{0, 2,2},{1, 2,2},{0,1, 2,2}, 4}
11 True
```

Prenons un exemple graphiquement parlant, avec $A = \{V_3\}$ …

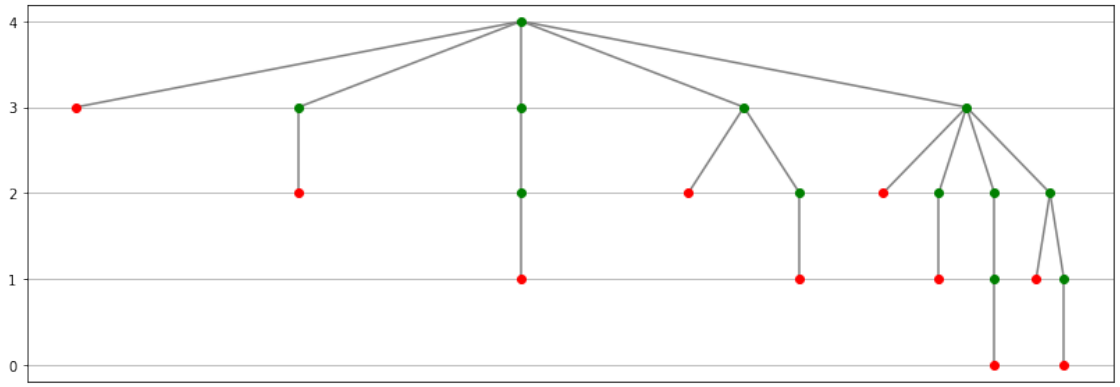
[33]:
```
A = [psi(ppow(2, 4) - 1)]
print(tostring(A))
tracer_arbre(A)
```

```
{{0,1, 2,2}}
```



[34]:
```
A = [psi(ppow(2, 4) - 1)]
print(tostring(cloture(A)))
tracer_arbre(cloture(A))
```

```
{0,1, 2,2,{0,1, 2,2}}
```



1.5 5. Le graphe des éléments de V

1.5.1 5.1 Graphes

Nous considérerons dans ce qui va suivre uniquement des graphes *dirigés* (ou orientés). Un *graphe dirigé* est un couple $G = (S, A)$ où

- S est un ensemble fini, l'ensemble des *sommets* de G.
- $A \subseteq S \times S$ est l'ensemble des *arêtes* de G.

Si $a = (x, y)$ est une arête de G, x est *l'origine* de a et y est *l'extrémité* de a. On note éventuellement $x \longrightarrow y$.

Pour tout $x \in S$, un *successeur* de x est un élément y de S tel que $(x, y) \in A$. Un *prédécesseur* de x est un élément y de S tel que $(y, x) \in A$. On note

$$x^+ = \{y \in S : (x, y) \in A\}$$

et

$$x^- = \{y \in S : (y, x) \in A\}$$

On appelle *source* du graphe G tout $x \in S$ tel que $x^- = \emptyset$ et *puits* de G tout $x \in S$ tel que $x^+ = \emptyset$. Un graphe peut avoir 0, une ou plusieurs sources (ref: puits).

Le graphe G est *acyclique* s'il n'existe pas d'entier $n \in \mathbb{N}$ et de sommets de G $x_0, \ldots, x_n$ distincts tels que

$$x_0 \longrightarrow x_1 \longrightarrow \ldots \longrightarrow x_n \longrightarrow x_0$$

Un graphe acyclique sera appelé de façon abrégée un DAG (Directed Acyclic Graph).

1.5.2 5.2 Le graphe d'un ensemble héréditairement fini

Tout ensemble $A \in V$ peut être représenté par un graphe $\mathcal{G}(A)$. Les sommets de ce graphe sont A et les éléments de $\overline{A}$. Pour tous sommets x et y de $\mathcal{G}(A)$, il y a une arête d'origine x et d'extrémité y si et seulement si $y \in x$.

Remarquons que comme il n'existe pas, dans V, d'ensembles $x_1, \ldots, x_n$ tels que $x_1 \in \ldots \in x_n \in x_1$, le graphe $\mathcal{G}(A)$ est acyclique. C'est donc un DAG.

Proposition. Soit $A \in V$.

- $\mathcal{G}(A)$ possède une unique source qui est A.
- $\mathcal{G}(A)$ possède un unique puits qui est $\emptyset$.

Démonstration.

- Clairement, A est une source de $\mathcal{G}(A)$ puisque $A \notin A$ et pour tout $x \in \overline{A}$, $A \notin x$. De plus, pour tout $x \in \overline{A}$, $x \in A$ ou alors il existe $y \in \overline{A}$ tel que $x \in y$. Ainsi, x n'est pas une source de $\mathcal{G}(A)$.
- Soit x un sommet de $\mathcal{G}(A)$. x est un puits si et seulement si x n'a pas d'élément. □

Le graphe $\mathcal{G}(A)$ possède une dernière propriété remarquable. Soient x et y deux sommets du graphe. Supposons que $x^+ = y^+$. Cela signifie que x et y ont les mêmes éléments. On en déduit que $x = y$.

Définition. Le DAG G est *extensionnel* si pour tous sommets x, y de G, $x^+ = y^+ \implies x = y$.

Nous venons donc de montrer :

Proposition. Soit $A \in V$. Le graphe $\mathcal{G}(A)$ est un DAG extensionnel ayant une unique source et un unique puits.

La fonction **graphe** prend en paramètre un ensemble $A \in V$. Elle renvoie la liste des arêtes du graphe de A.

[35]:
```
def graphe(A):
    C = cloture(A)
    aretes = []
    for x in C:
        for y in x:
            if (x, y) not in aretes:
                aretes.append((x, y))
    for y in A:
        aretes.append((A, y))
    return aretes
```

Prenons par exemple $\mathcal{G}(\overline{2} \times \tau_3)$. On n'affiche que les 15 premiers caractères de la représentation des sommets.

[36]:
```
G = graphe(produit(neumann(2), tau(3)))
print('Nombre d\'arêtes : ', len(G))
for x, y in G:
    if len(tostring(x)) > 20:
        print('%-20s %-20s' % (tostring(x)[:15] + '...', tostring(y)))
    else:
        print('%-20s %-20s' % (tostring(x), tostring(y)))
```

```
Nombre d'arêtes :  23
1                     0
 2                    1
2                     0
2                     1
 3                    2
{0, 2}                0
{0, 2}                2
{1, 2}                1
{1, 2}                2
<0,1>                 1
<0,1>                 2
<1,0>                  2
<1,0>                 2
<0, 2>                1
<0, 2>                {0, 2}
<1, 2>                2
<1, 2>                {1, 2}
{ 2, 3,<0,1>,<1…    2
{ 2, 3,<0,1>,<1…    3
{ 2, 3,<0,1>,<1…    <0,1>
{ 2, 3,<0,1>,<1…    <1,0>
{ 2, 3,<0,1>,<1…    <0, 2>
```

```
{ 2, 3,<0,1>,<1…   <1, 2>
```

À vous d'essayer.

[37]:
```
G = graphe(psi(123456))
print('Nombre d\'arêtes : ', len(G))
for x, y in G:
    if len(tostring(x)) > 35:
        print('%-35s %-30s' % (tostring(x)[:30] + '...', tostring(y)))
    else:
        print('%-35s %-30s' % (tostring(x), tostring(y)))
```

```
Nombre d'arêtes :  26
1                                   0
 2                                  1
2                                   0
2                                   1
 3                                  2
{1, 2}                              1
{1, 2}                              2
{0,2}                               0
{0,2}                               2
{0, 2,2}                            0
{0, 2,2}                            2
{0, 2,2}                            2
{1, 2,2}                            1
{1, 2,2}                            2
{1, 2,2}                            2
{0,1, 2,2}                          0
{0,1, 2,2}                          1
{0,1, 2,2}                          2
{0,1, 2,2}                          2
 4                                  3
{{1, 2},{0,2},{0, 2,2},{1, 2,2…   {1, 2}
{{1, 2},{0,2},{0, 2,2},{1, 2,2…   {0,2}
{{1, 2},{0,2},{0, 2,2},{1, 2,2…   {0, 2,2}
{{1, 2},{0,2},{0, 2,2},{1, 2,2…   {1, 2,2}
{{1, 2},{0,2},{0, 2,2},{1, 2,2…   {0,1, 2,2}
{{1, 2},{0,2},{0, 2,2},{1, 2,2…    4
```

1.5.3 5.3 La fonction de tracé

La fonction `niveaux` prend en paramètre un ensemble $A \in V$. Soit $n = \text{rg } A$. La fonction renvoie une liste `niv` de longueur $n+1$ telle que pour $0 \leq k \leq n-1$, `niv[k]` est la liste des éléments de $\overline{A}$ de rang k, et `niv[n]` est le singleton $[A]$.

[38]:
```
def pos(x, s):
    n = len(s)
    for k in range(n):
        if x == s[k]: return k
```

[39]:
```
def niveaux(A):
    C = cloture(A)
    n = rang(A)
    niv = n * [None]
    for k in range(n): niv[k] = []
    for x in C:
        niv[rang(x)] = reunion(niv[rang(x)], [x])
    niv.append([A])
    return niv
```

[40]:
```
A = psi(1234567)
niv = niveaux(A)
n = rang(A)
for k in range(n + 1): print(k, [tostring(x) for x in niv[k]])
```

```
0 ['0']
1 ['1']
2 [' 2', '2']
3 [' 3', ' 3', '{0,2}', '<0,1>', '<1,0>', '{1, 2,2}', '{0,1, 2,2}']
4 ['{0, 3}', '{ 2, 3}']
5 ['{0,1, 2, 3,{0,2},<0,1>,<1,0>,{1, 2,2},{0,1, 2,2},{0, 3},{ 2, 3}}']
```

Nous ne commentons pas la fonction `tracer_graphe` ci-dessous. Elle prend en paramètre un ensemble $A \in V$ et trace le DAG $\mathcal{G}(A)$.

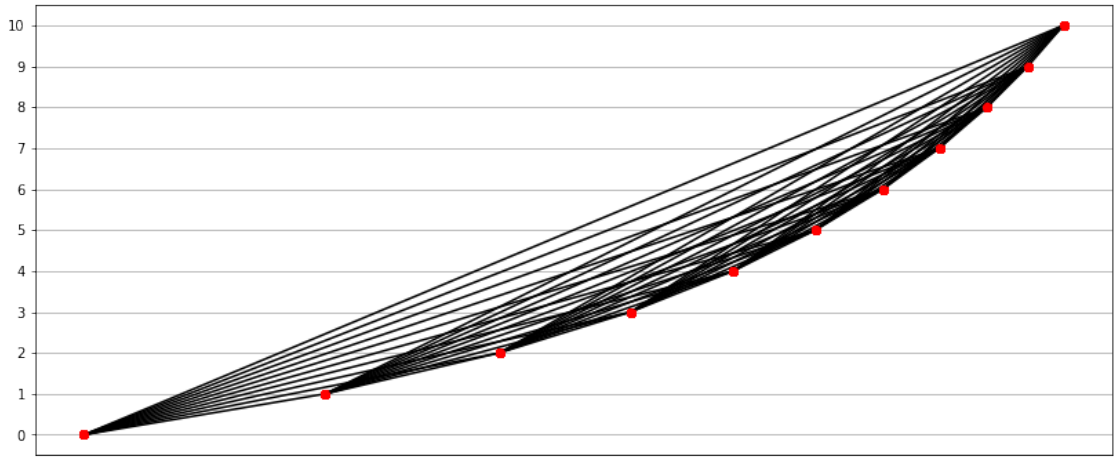
[41]:
```
def g(i, k, l, d):
    return math.atan(k + d) + (i + 1) / (l + 1)
```

[42]:
```
def tracer_graphe(A, d=5):
    plt.rcParams['figure.figsize'] = (14, 6)
    niv = niveaux(A)
    n = len(niv)
    for k in range(n):
        lk = len(niv[k])
        for i in range(lk):
            for y in niv[k][i]:
                k1 = rang(y)
                lk1 = len(niv[k1])
                j = pos(y, niv[k1])
                plt.plot([g(i, k, lk, d), g(j, k1, lk1, d)], [k, k1], 'k')
                plt.plot([g(i, k, lk, d), g(j, k1, lk1, d)], [k, k1], 'or')
    plt.xticks([])
    plt.yticks(range(n))
```

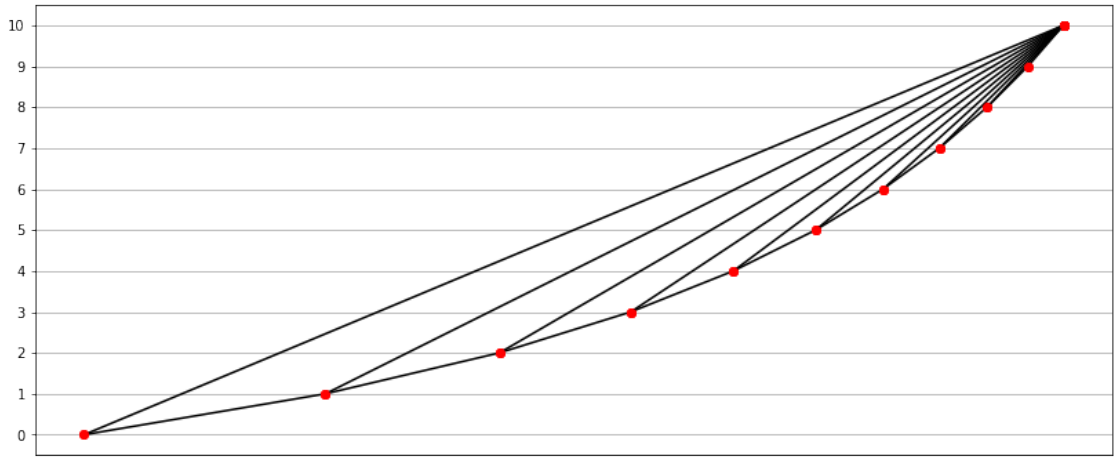
```
    plt.grid()
```

1.5.4 5.4 Exemples

[43]:
```
A = neumann(10)
tracer_graphe(A)
```

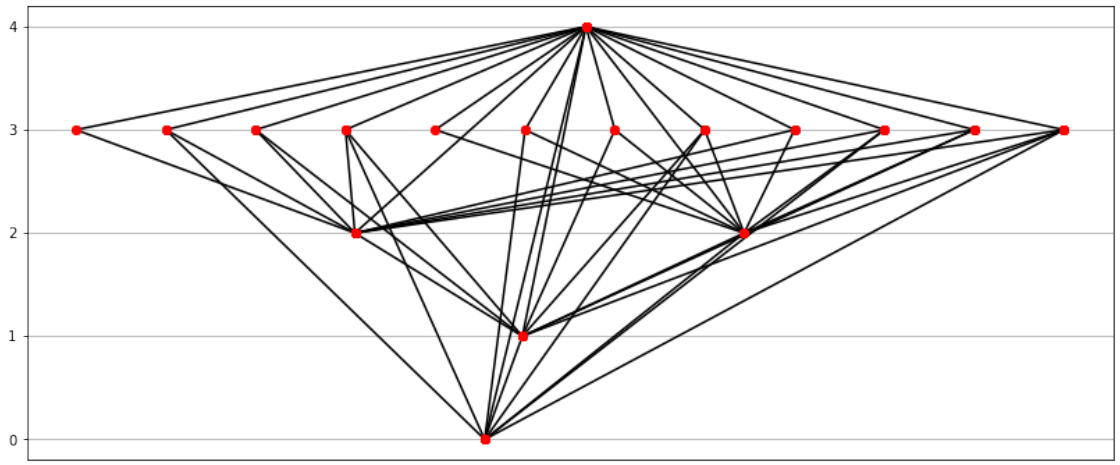


[44]:
```
A = tau(10)
tracer_graphe(A)
```



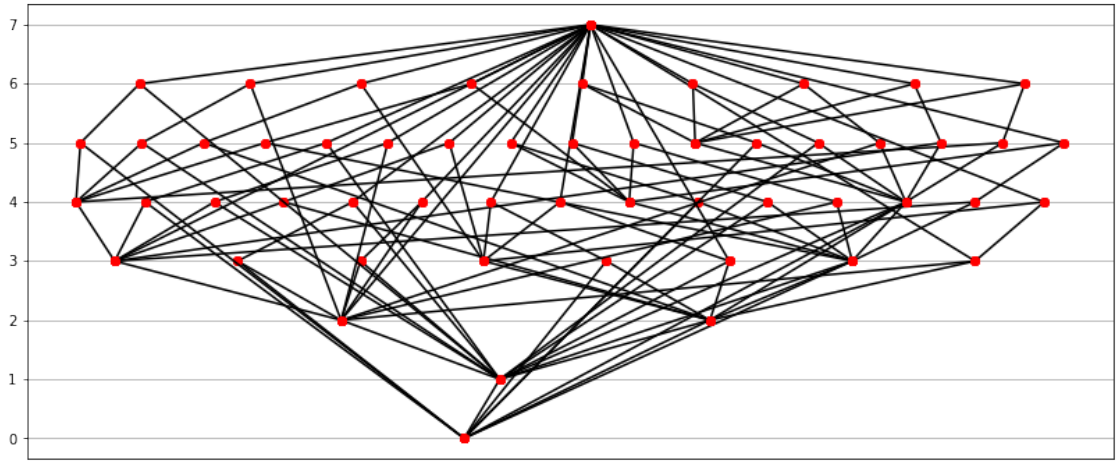
Voici V_4.

[45]:
```
A = psi(ppow(2, 5) - 1)
tracer_graphe(A)
```



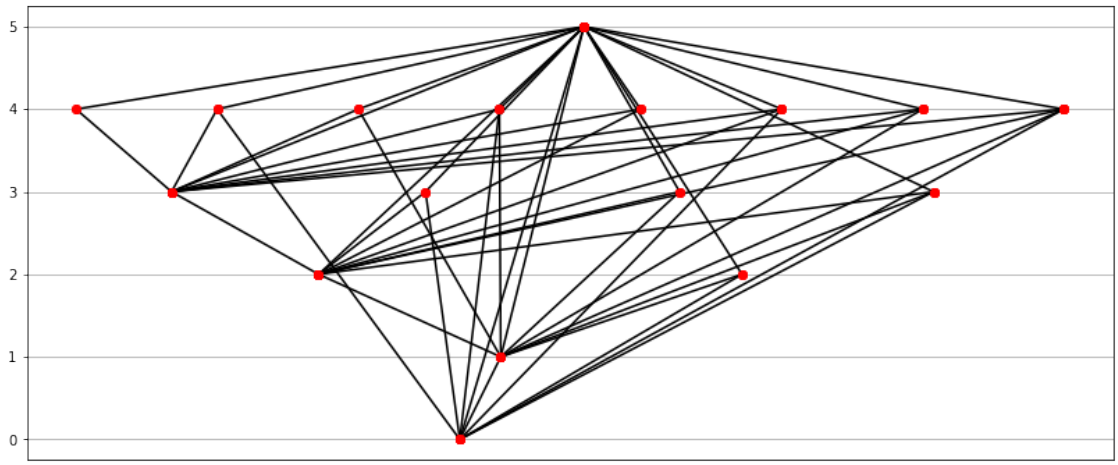
Voici $\overline{5} \times \tau_5$.

[46]:
```
A = produit(neumann(5), tau(5))
tracer_graphe(A)
```



Et pour finir, $\mathcal{P}(\tau_4)$.

[47]:
```
A = parties(tau(4))
tracer_graphe(A)
```



1.6 6. Isomorphismes

1.6.1 6.1 Isomorphismes de graphes

Soient $G = (S, A)$ et $G' = (S', A')$ deux graphes. On appelle *isomorphisme* de G sur G' toute bijection $f : S \longrightarrow S'$ telle que pour tout $(x, y) \in S \times S$, $(x, y) \in A \iff (f(x), f(y)) \in A'$.

Deux graphes isomorphes sont donc le « même » graphe, « à renommage près » des sommets. Lorsqu'il existe un isomorphisme de G sur G', nous dirons que G et G' sont *isomorphes* et nous noterons $G \simeq G'$.

1.6.2 6.2 Le théorème d'isomorphisme.

Nous avons déjà vu que pour tout $A \in V$, $\mathcal{G}(A)$ est un DAG extensionnel ayant une unique source et un unique puits. Nous laissons au lecteur le soin de vérifier que, étant donnés deux graphes G et G' isomorphes,

- Si G est un DAG, alors G' est un DAG.
- Si G est extensionnel, alors G' est extensionnel.
- Les sources (resp : puits) de G s'envoient par l'isomorphisme sur les sources (resp : puits) de G'. En conséquence, si G a une unique source (resp : puits) alors G' aussi.

En conséquence, si $A \in V$ et G est un graphe isomorphe à $\mathcal{G}(A)$, alors G est un graphe extensionnel ayant une unique source et un unique puits.

Définition. Un *V-graphe* est un DAG extensionnel ayant une unique source et un unique puits.

Il est tentant de se poser la question de la réciproque. La réponse est affirmative.

Théorème. Soit G un V-graphe. Il existe un unique ensemble $A \in V$ tel que $G \simeq \mathcal{G}(A)$.

Nous allons montrer ce résultat en plusieurs étapes.

1.6.3 6.3 Graphe induit

Définition. Soit $G = (S, A)$ un graphe. Soit $S' \subseteq S$. Le graphe induit par G sur S' est $G \restriction S' = (S', A')$ où $A' = A \cap (S' \times S')$.

Remarquons qu'un graphe induit d'un DAG est encore un DAG. De même, un graphe induit d'un DAG extensionnel est encore un DAG extensionnel. Si $A \in V$ et $x \in A$, le graphe induit par $\mathcal{G}(A)$ sur $\{x\} \cup \overline{x}$ est $\mathcal{G}(x)$.

1.6.4 6.4 Quelques résultats préliminaires

Définition. Soit $G = (S, A)$ un graphe. Soient $x, y \in S$. Un *chemin* d'origine x et d'extrémité y est une suite d'arêtes $\gamma = ((x_0, x_1), \ldots, (x_{n-1}, x_n))$ où $n \in \mathbb{N}$, $x_0 = x$ et $x_n = y$. Nous noterons

$$\gamma = (x\ x_1\ x_2 \ldots\ x_{n-1}\ y)$$

La *longueur* du chemin γ est $\ell(\gamma) = n$. Le cas $n = 0$ nous dit que pour tout $x \in S$ il existe un (unique) chemin de longueur 0 d'origine x et d'extrémité x.

Si $\gamma = (x \ldots y)$ est un chemin d'origine x et d'extrémité y, et $\gamma' = (y \ldots z)$ est un chemin d'origine y et d'extrémité z, la *concaténation* de γ et γ', $\gamma \oplus \gamma' = (x \ldots y \ldots z)$ est un chemin d'origine x et d'extrémité z.

Remarquons que si G est un DAG, alors les x_k sont nécessairement tous distincts, car sinon, G possèderait un cycle. Les longeurs des chemins dans un DAG $G = (S, A)$ sont donc majorées par $|S| - 1$.

Proposition. Tout DAG possède au moins une source et au moins un puits.

Démonstration. Soit G un DAG. Soit γ un chemin de longueur maximale dans G. L'origine de γ est une source, son extrémité est un puits, sinon on pourrait augmenter γ en un chemin de longueur strictement plus grande.

Lemme 1. Soit G un DAG. Pour tout sommet x de G, il existe un chemin d'origine x et d'extrémité un puits de G.

Démonstration. Il suffit ici aussi de considérer un chemin d'origine x et de longueur maximale. L'extrémité de ce chemin est un puits. □

Lemme 2. Soit G un DAG. Pour tout sommet x de G, il existe un chemin d'origine une source de G et d'extrémité x.

Démonstration. Il suffit de considérer un chemin d'extrémité x et de longueur maximale. L'origine de ce chemin est une source. □

Le résultat de ce lemme suggère une définition.

Définition. Soit G un DAG ayant une unique source s et un unique puits t. Le *rang* de G, rg G, est la longueur du plus *long* chemin de s à t.

Cette définition n'est évidemment pas innocente : si $A \in V$, on a alors rg $\mathcal{G}(A) =$ rg A.

1.6.5 6.5 La preuve du théorème

Preuve du théorème - Existence. Montrons l'existence par récurrence forte sur le rang du graphe G. Dans ce qui suit, nous n'entrons pas dans tous les détails de la preuve.

- Si G est de rang 0, $G = (\{s\}, \emptyset)$ a un unique sommet et pas d'arête. Clairement, $G \simeq \mathcal{G}(\emptyset)$.
- Soit $n \in \mathbb{N}^*$. Supposons que pour tout V-graphe de rang strictement inférieur à n, il existe $A \in V$ tel que $G \simeq \mathcal{G}(A)$. Soit G un V-graphe de source s et de rang n. Soient $x_1, \ldots, x_n$ les successeurs de s. Pour tout $k \in [\![1, n]\!]$, le sous-graphe G_k de G de source x_k est un V-graphe de rang strictement inférieur à n. Par l'hypothèse de récurrence, il existe $A_k \in V$ tel que $G_k \simeq \mathcal{G}(A_k)$. Par l'extensionalité de G, les A_k sont distincts. Soit $A = \{A_1, \ldots, A_n\}$. On a alors $G \simeq \mathcal{G}(A)$.

Preuve du théorème - Unicité. Montrons par récurrence forte sur $n = \max(\text{rg } A, \text{rg } A')$ que pour tous $A, A' \in V$, si $\mathcal{G}(A) \simeq \mathcal{G}(A')$, alors $A = A'$. ici encore, nous ne donnerons pas tous les détails.

- Si $n = 0$ c'est clair puisqu'alors $A = A' = \emptyset$.
- Soit $n \in \mathbb{N}^*$. Supposons la propriété vraie pour tous les entiers strictement inférieurs à n. Soient $A, A' \in V$ tels que $\max(\text{rg } A, \text{rg } A') = n$. Supposons que $\mathcal{G}(A) \simeq \mathcal{G}(A')$. Soit f un isomorphisme de $\mathcal{G}(A)$ sur $\mathcal{G}(A')$. rappelons que les sommets de $\mathcal{G}(A)$ sont A et les éléments de $\overline{A}$, et de même pour $\mathcal{G}(A')$. Un isomorphisme envoyant la source du premier graphe sur la source de l'autre, on a $f(A) = A'$.

Soient $x_1, \ldots, x_n$ les éléments de A. Ce sont les successeurs de la source A dans le graphe $\mathcal{G}(A)$. Ils sont donc envoyés par f sur les successeurs de A' dans $\mathcal{G}(A')$, qui sont les éléments de A'. Notons $x'_k = f(x_k)$ pour $k = 1, \ldots, n$. On a donc $A = \{x_1, \ldots, x_n\}$ et $A' = \{x'_1, \ldots, x'_n\}$.

Pour $k = 1, \ldots, n$, le sous-graphe de $\mathcal{G}(A)$ de source x_k est isomorphe, via f, au sous-graphe de $\mathcal{G}(A')$ de source x'_k. Ces deux graphes sont respectivement $\mathcal{G}(x_k)$ et $\mathcal{G}(x'_k)$. Comme $\max(\text{rg } x_k, \text{rg } x'_k) < n$, l'hypothèse de récurrence affirme que $x'_k = x_k$. On en déduit que A et A' ont les mêmes éléments, donc que $A = A'$. Remarquons qu'en prime, $\mathcal{G}(A) = \mathcal{G}(A')$, et que l'isomorphisme f est l'identité de A. □