Boîte de snake
De combien de manières différentes peut-on ranger un serpent dans une boîte carré de coté c de façon à ce que le serpent occupe toute la place ?

Gabriel Doriath–Döhler* & Théo Rudkwiecz*

July 9, 2018

Serpent

1 Introducion

Un pacours est une liste ordonnée de positions. Soit L un parcours. c est la taille de la grille.

$$L \text{ est valide} \iff \begin{cases} \forall (x,y) \in [\![1;c]\!]^2 ; (x,y) \in L \quad \text{(Tout les points sont atteints)} \\ \forall p \in L ; p \in [\![1;c]\!]^2 \quad \text{(Rien ne dépasse du carré)} \\ \forall (p,q) \in L^2 ; p \neq q \quad \text{(Aucun point est atteint deux fois)} \\ \forall i \in [\![1;c[\![/ L_i = (x,y) ; [L_{i+1} = (x \pm 1, y)] \vee [L_{i+1} = (x, y \pm 1)] \\ \text{(Tout les déplacements sont justes)} \end{cases}$$

On cherche $max(card(\Omega))$

$$\Omega = \bigcup L/L \ est\ valide$$

On peut aussi dire que l'on cherche l à dénombrer l'ensemble des chaînes hamiltoniennes orientées dans un graphe carré de taille c.

graphe(5_5).png

Figure 1: Représentation du graphe carré pour $c = 5$

1.0.1 Définition

- Une case de coordonnés (x,y) est dite paire si et seulement si $x+y$ est paire et impaire si et seulement si $x+y$ est impaire.
- Un chemin, contrairement à un parcours ne prend pas en compte les sens dans le quel le serpent est mis. Ainsi: $((0,0);(1,0)) \Leftrightarrow ((1,0);(0,0))$.

2 Propriété

1.

$$2 \mid c \Leftrightarrow (x+y)\ de\ L_0 \equiv 1 + (x+y)\ de\ L_{-1}[2]$$
$$2 \nmid c \Leftrightarrow (x+y)\ de\ L_0 \equiv (x+y)\ de\ L_{-1}[2]$$

2.

$$2 \nmid c \Leftrightarrow \neg \exists L / L\ est\ valide \wedge 2 \nmid (x+y)\ de\ L_0$$

graphe(5_5)_marquer.png

Figure 2: Représentation du graphe carré pour $c = 5$, avec en rouge les cases d'où aucun parcours part

3 Démonstation

1. Soit (x, y) les coordonnés du point de départ. On fait $c^2 - 1$ déplacements. Donc la somme des coordonnés du point d'arrivé est : $c^2 - 1 + (x+y)$. D'où :

$$\begin{cases} 2 \mid c \Leftrightarrow (x+y)\ de\ L_0 \equiv 1 + (x+y)\ de\ L_{-1}[2] \\ 2 \nmid c \Leftrightarrow (x+y)\ de\ L_0 \equiv (x+y)\ de\ L_{-1}[2] \end{cases}$$

2. Lemme (évident) :

$$2 \nmid c \Leftrightarrow \begin{cases} \text{nombre de cases paires} = \frac{c^2+1}{2} \\ \text{nombre de cases impaires} = \frac{c^2-1}{2} \end{cases}$$

Par l'absurbe : On suppose qu'il existe un trajet partant d'une case impaire allant vers une case impaire. Entre ces deux points il devra parcourir $\frac{c^2+1}{2}$ case paires et $\frac{c^2-1}{2} - 2$ case impaires. Or le trajet est forcément constitué d'une alternance de cases paires et impaires. Ce trajet n'existe donc pas. ■

4 Algorithme et résultat

4.0.1 Résutats

c	Nb parcours	Temps
1	1	0
2	8	0
3	40	0
4	552	0.019
5	6 648	2.088
6	458 696	603.924
7	27 070 560	?
8	6 046 626 568	?
9	1 490 832 682 992	?
10	1 460 089 659 025 264	?
11	1 573 342 970 540 617 696	?
12	6 905 329 711 608 694 708 440	?
13	33 304 011 435 341 069 362 631 160	?
14	663 618 176 813 467 308 855 850 585 056	?
15	14 527 222 735 920 532 980 525 200 234 503 048	?

Figure 3: Résultats de l'algorithme et de l'OEIS[1]

4.0.2 Visualisation des parcours possibles pour $c = 3$

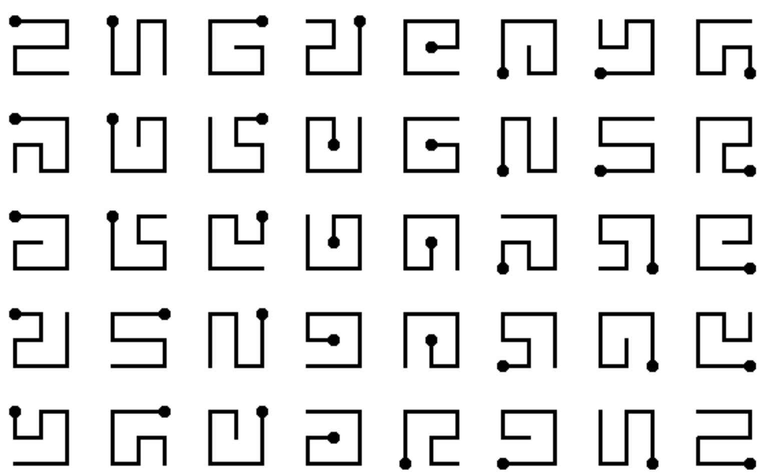

Figure 4: Représentation de Ω pour $c = 3$

[1]https://oeis.org/The On-Line Encyclopedia of Integer Sequences[?]

4.0.3 Algorithme en

```python
def point_generator(c):
    """List du quarts du graphe"""
    pts = []
    for x in range(1, c // 2 + 1):
        for y in range(1, c // 2 + 1):
            pts.append(x * 10 + y)
    if c % 2:
        for y in range(1, c // 2 + 1):
            pts.append((c // 2 + 1)* 10 + y)
    return pts

def refeur(c):
    """Ensemble des points du graphe de taille c"""
    r = set()
    for x in range(1, c + 1):
        for y in range(1, c + 1):
            r.add(x * 10 + y)
    return r

def walkeur(pts, c, ref, pred):
    """Visite tt les trajets depuis un point"""
    if len(pts) == c ** 2:
        return [pts]
    else:
        result = []
        for d in 1, 10, -10, -1:
            if d != pred * - 1 and pts[-1] + d in ref and pts[-1] + d not in pts:
                result.extend(
                    walkeur(pts + [pts[-1] + d], c, ref, d)
                )
        return result

cote = int(input('Cote: '))
nb_parcours = 0
nb_mid = 0
r = refeur(cote)
for p in point_generator(cote):
    nb_parcours += len(walkeur([p], cote, r, 0))
if cote % 2:
    nb_mid += len(walkeur([(cote // 2 + 1) * 11], cote, r, 0))
print(f"Nb parcours: {nb_parcours * 4 + nb_mid}")
```

Figure 5: Algorithme de dénombrement des parcours en