Algorithmique 2
Matthieu Boyer
Planche A

Khôlle MP2I
02/12

N-Reines

On cherche à place N reines sur un échiquier de taille N par N de sorte que deux reines différentes ne s'attaquent. Renvoyez toutes les solutions du problème. On pourra par exemple stocker pour chaque colonne/ligne/diagonale un booléen qui indique si elle contient une reine. . On pourra aussi procéder par récurrence en s'arrêtant et revenant en arrière.

Solution: N-Reines

Inspiré de `https://leetcode.com/problems/n-queens/solutions/5077109/bitmanipulation-c-100/`.

De Manhattan à Brooklyn

La distance de Manhattan entre deux points est définie par :

$$d\left((a,b),(c,d)\right) = |c-a| + |d-b|$$

Étant donné une liste de points, donnez le minimum des distances maximales possibles en retirant un point.

Solution: De Manhattan à Brooklyn

```python
import numpy as np
def get_max_i_j(points):
        plus_max_i, plus_min_i, minus_max_i, minus_min_i = -1,-1,-1,-1
        x_plus_y_max , x_plus_y_min = float('-inf'), float('inf')
        x_minus_y_max, x_minus_y_min = float('-inf'), float('inf')
        for i,point in enumerate(points):
        x = point[0]
        y = point[1]
        if x + y > x_plus_y_max:
                x_plus_y_max = x + y
                plus_max_i = i
        if x + y < x_plus_y_min:
                x_plus_y_min = x + y
                plus_min_i = i
        if x - y > x_minus_y_max:
                x_minus_y_max = x - y
                minus_max_i = i
        if x - y < x_minus_y_min:
                x_minus_y_min = x - y
                minus_min_i = i
        if x_plus_y_max - x_plus_y_min > x_minus_y_max - x_minus_y_min:
                return x_plus_y_max - x_plus_y_min, plus_max_i, plus_min_i
        else:
                return x_minus_y_max - x_minus_y_min, minus_max_i, minus_min_i

def minimumDistance(self, points0: List[List[int]]) -> int:
        a,i,j = get_max_i_j(points0)
        bi, _, _ = get_max_i_j(points0[0:i]+points0[i+1:])
        bj, _, _ = get_max_i_j(points0[0:j]+points0[j+1:])
        return min(bi,bj)
```

Algorithmique 2
Matthieu Boyer
Planche C

Khôlle MP2I
02/12

Jardins Suspendus

Un fermier a une parcelle de terre rectangulaire de m lignes et n colonnes, divisées en cellules. Une cellule est soit fertile (représentée par un 1) soit stérile (représentée par un 0). Une parcelle pyramidale est définie comme un ensemble de cellules tel que :

- Le nombre de cellules dans la parcelle est strictement plus grand que 1.
- Toutes les cellules de la parcelle sont fertiles.
- Le sommet de la pyramide est la cellule la plus haute de la pyramide. La hauteur d'une pyramide est le nombre de lignes atteintes. Si (r, c) est le sommet de la pyramide et h est sa hauteur, alors pour tout $i \in [\![r, r + h - 1]\!]$, et $j \in [\![c - (i - r), c + (i - r)]\!]$, la cellule (i, j) est dans la pyramide.

On définit de même une pyramide inverse : son sommet est sa cellule la plus basse et elle doit contenir toutes les cases i, j pour $i \in [\![r - h + 1, r]\!]$ et $j \in [\![c - (i - r), c + (i - r)]\!]$. Étant donné une matrice binaire $m \times n$ représentant la parcelle de terre, donnez le nombre total de parcelles pyramidales et de parcelles pyramidales inverses qu'on peut trouver. Deux parcelles peuvent se recouvrir, mais pas être incluses dans une autre.

Solution: Jardins Suspendus

On fait de la programmation dynamique : `dp[i][j]` représente le nombre de lignes de la plus grande pyramide ayant pour sommet i, j.

```
def countPyramids(grid):
        m, n, dp, counter = len(grid), len(grid[0]), copy.deepcopy(grid), 0
        for i in range(m - 2, -1, -1):
                for j in range(1, n - 1):
                        if dp[i][j] > 0 and dp[i + 1][j] > 0:
                                dp[i][j] = min(dp[i + 1][j - 1], dp[i + 1][j + 1] + 1)
                                counter += dp[i][j] - 1

        dp = grid
        for i in range(1, m):
                for j in range(1, n - 1):
                        if dp[i][j] > 0 and dp[i + 1][j] > 0:
                                dp[i][j] = min(dp[i - 1][j - 1], dp[i - 1][j + 1] + 1)
                                counter += dp[i][j] - 1
        return counter
```