Algoritmos y Programación

Práctica 4.1: Fuerza Bruta

N-Queens mediante Fuerza Bruta

- Tres VPLs:
  1) Dado un número codificado en una determinada Base de numeración calcular el siguiente número.
  2) Utilizando yield escribir un iterador para resolver problemas mediante fuerza bruta.
  3) Utilizando nuestro iterador calcular mediante fuerza bruta todas las soluciones para colocar N reinas en un tablero de N x N.

VPL 1: Siguiente número

- 1357 (base 10) → 1358
- 0101 (base 2) → 0110
- 02333 (base 4) → 03000
- 011111111 (base 2) → 100000000
- 111111111 (base 2) → 000000000

Base	Sistema
2	Sistema binario
3	Sistema ternario
5	Sistema quinario
8	Sistema octal
10	Sistema decimal
12	Sistema duodecimal
16	Sistema hexadecimal
20	Sistema vigesimal
60	Sistema sexagesimal
64	Base64

https://es.wikipedia.org/wiki/Base_(aritm%C3%A9tica)

VPL 1: Formato del fichero de entrada

- La primera línea es un descriptor con dos valores: cuantos números hay que procesar y la Base numérica de los números.
- El resto de las líneas son los números.

Ejemplo

7 números en base 2 → 7 2

Los 7 números

00
01
10
11
011
101
0111

Salida de nuestro programa:

[0, 0] - [0, 1]
[0, 1] - [1, 0]
[1, 0] - [1, 1]
[1, 1] - [0, 0]
[0, 1, 1] - [1, 0, 0]
[1, 0, 1] - [1, 1, 0]
[0, 1, 1, 1] - [1, 0, 0, 0]

La salida contiene dos listas separadas por un guión: una con los dígitos del número de entrada y otra con los dígitos del siguiente número.

VPL 1

main.py

```python
from solve import *

first_line = input().split()
num_values = int(first_line[0])
base       = int(first_line[1])

for j in range(num_values):
    data = input()

    # Convertimos la string en la lista que contiene
    # el número de entrada.
    digits = []
    for digit in data:
        digits.append(int(digit))

    # Mostramos la lista con el número de entrada y
    # la lista con el siguiente número.
    print(digits, '- ', end="")
    print(next_number(digits, base))

```

solve.py

```python
def next_number(digits, base):
    """
    :param digits: list containing all the digits of a number
                   in the given base
    :param base: numeric base of the number
    :return: list representing the next value of the number

     Example: digits = [0, 1, 0, 1]   number 5
                 base = 2

              returns [0, 1, 1, 0]    number 6
    """

    next_digits = digits.copy()

    # Añade tu código aqui
    # ...

    return next_digits

```

VPL 2: Iterador para fuerza bruta

2. Utilizando yield escribir un iterador para resolver problemas mediante fuerza bruta.

Formato del fichero de entrada

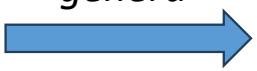
Una única línea con dos números: número de dígitos y base del número.

Primer ejemplo: 4 2

4 dígitos

Base 2

genera



[0, 0, 0, 0]
[0, 0, 0, 1]
[0, 0, 1, 0]
[0, 0, 1, 1]
[0, 1, 0, 0]
[0, 1, 0, 1]
[0, 1, 1, 0]
[0, 1, 1, 1]
[1, 0, 0, 0]
[1, 0, 0, 1]
[1, 0, 1, 0]
[1, 0, 1, 1]
[1, 1, 0, 0]
[1, 1, 0, 1]
[1, 1, 1, 0]
[1, 1, 1, 1]

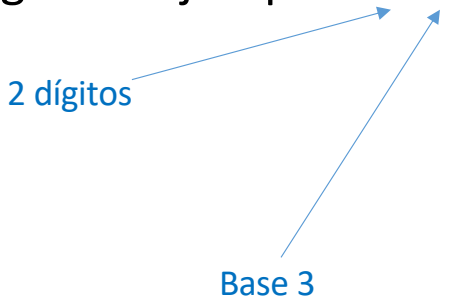
VPL 2: Iterador para fuerza bruta

2. Utilizando yield escribir un iterador para resolver problemas mediante fuerza bruta.

Formato del fichero de entrada

Una única línea con dos números: número de dígitos y base del número.

Segundo ejemplo: 2 3



genera

[0, 0]
[0, 1]
[0, 2]
[1, 0]
[1, 1]
[1, 2]
[2, 0]
[2, 1]
[2, 2]

VPL 2

solve.py

```python
# 1. Copia aqui tu solución del primer ejercicio de esta semana

def next_number(digits, base):
    # ...
    return digits

# -----------------------------------------------------------

class My_Iterator:

    def __init__(self, num_digits, base):
        # 2.1 Añade código aqui
        # ...

    def next(self):
        # 2.2 Añade código aqui
        # ...


        # Cuando no quedan valores simplemente retornamos
        return
```

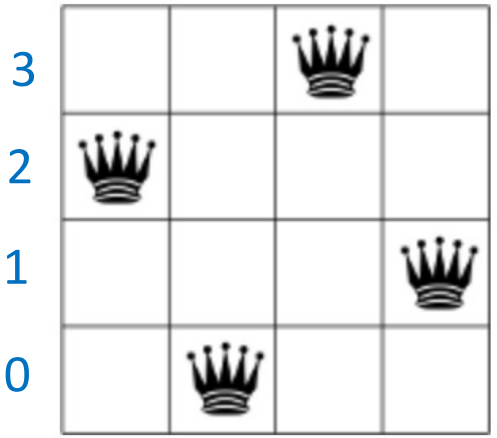
main.py

```python
from solve import *

first_line   = input().split()
num_digits = int(first_line[0])
base       = int(first_line[1])

obj = My_Iterator(num_digits, base)
for c in obj.next():
    print(c)
```

VPL 3: N-Queens (fuerza bruta)

3) Utilizando nuestro iterador resolver mediante fuerza bruta el problema de colocar N reinas en un tablero de N x N

Ejemplo: 4 reinas en un tablero de 4x4



Debemos colocar las N reinas de forma que ninguna esté en la misma fila, columna o diagonal que otra reina.

Codificamos cada solución como una lista que indica la fila donde debemos colocar cada reina. Esta solución sería: [2,0,3,1]

VPL 3: Formato del fichero de entrada

- El fichero sólo tiene una línea con el valor de N

Ejemplo

4

4 reinas en un tablero de 4 x 4

Salida de nuestro programa:

[1, 3, 0, 2]
[2, 0, 3, 1]

VPL 3

my_iterator.py

```python
# Copia aqui tu solución de los dos primeros ejercicios
# de esta semana

def next_number(digits, base):
    # ...
    return digits

# ---------------------------------------------------------

class My_Iterator:

    def __init__(self, num_digits, base):
        # 1. Añade código aqui
        # ...

    def next(self):
        # 2. Añade código aqui
        # ...

        # Cuando no quedan valores simplemente retornamos
        return

```

main.py

```python
from solve import *

first_line = input().split()
num_queens = int(first_line[0])

solutions_list = solve(num_queens)

for solution in solutions_list:
    print(solution)
```

solve.py

```python
from my_iterator import *

def solve(num_queens):
    """
    Using your brute force iterator compute all the
    solutions to place the given number of queens in
    a square board.

    :param num_queens: number of queens to place in the board
    :return: list of lists containing all the solutions

    For example, if num_queens = 4 there are two solutions,
    and it returns:
       solutions_list = [ [1, 3, 0, 2], [2, 0, 3, 1] ]

    """

    solutions_list = []

    # solve it here!


    return solutions_list

```