4. Dada una matriz D de $n \times n$ números naturales, queremos encontrar una permutación π[1] de $\{1, \ldots, n\}$ que minimice $D_{\pi(n)\pi(1)} + \sum_{i=1}^{n-1} D_{\pi(i)\pi(i+1)}$. Por ejemplo, si

$$D = \begin{pmatrix} 0 & 1 & 10 & 10 \\ 10 & 0 & 3 & 15 \\ 21 & 17 & 0 & 2 \\ 3 & 22 & 30 & 0 \end{pmatrix},$$

entonces $\pi(i) = i$ es una solución optima.

a) Diseñar un algoritmo de *backtracking* para resolver el problema, indicando claramente cómo se codifica una solución candidata, cuáles soluciones son válidas y qué valor tienen, qué es una solución parcial y cómo se extiende cada solución parcial.

b) Calcular la complejidad temporal y espacial del mismo.

c) Proponer una poda por optimalidad y mostrar que es correcta.

[1] Una permutacion de un conjunto finito X es simplemente una función biyectiva de X en X.

a)

```
n, D dados por enunciado.
minimo = inf
perm_minimal = {}

f bt(pi, usados, suma):
  si |pi| es n:
    total = suma + D[pi[n]][pi[1]]
    si suma < minimo:
      minimo = suma
      perm_minimal = pi[n][1]

  para cada i en 1...k:
    si i no está en usados
      si pi es vacio:
        sol = 0
      sino:
        sol = suma + D[pi[k][i]]

      bt(pi+{i}, usados+{i}, sol)

      pi = pi sin el ultimo elemento
      usados = usados-{i}

bt({},{},0)
solucion = perm_minimal
```

Solución candidata: Algún π

Solución válida: $|\pi| = n \wedge D_{\pi(n)\pi(1)} + \sum_{i=1}^{n-1} D_{\pi(i)\pi(i+1)}$

Solución parcial: Una primera construcción de π, $|\pi| < n$

b)

Un conjunto de n elementos tiene $n!$ permutaciones, y hacemos n operaciones por paso, luego la complejidad temporal es $O(n! \cdot n)$

usados es $O(n)$, al igual que pi. Luego la complejidad espacial es el tamaño de D, $O(n^2)$

c)

Una poda podría ser dejar de recorrer ramas que superan el minimo actual.