3. Dada una matriz simétrica M de $n \times n$ números naturales y un número k, queremos encontrar un subconjunto I de $\{1, \ldots, n\}$ con $|I| = k$ que maximice $\sum_{i,j \in I} M_{ij}$. Por ejemplo, si $k = 3$ y:

$$M = \begin{pmatrix} 0 & 10 & 10 & 1 \\ - & 0 & 5 & 2 \\ - & - & 0 & 1 \\ - & - & - & 0 \end{pmatrix},$$

entonces $I = \{1, 2, 3\}$ es una solución óptima.

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.

---

a)

```
I_maximo = {}
suma_maxima = -inf

f bt(pos, I):    //k entero, I conjunto

  //Caso base
  Si |I| = k
    sum_total = 0
    para i en [1 ... |I|]
      para j en [1 ... |I|]
        sum_total += M(I[i],I[j])

    si total > suma_maxima:
      suma_maxima = total
      I_maximo = I
    return

  //Paso recursivo
  para todo i en [pos ... n]:
    bt(i+1,I+{i})

//Se resuelve con:
bt(0,{})
//Luego, el resultado es I_maximo.
```

Solución candidata: I

Solución válida: $\sum_{i,j \in I} M_{ij}$, $|I| = k$

Solución parcial: I, $|I| < k$

b)

Cada nodo interno es $O(1)$ y hay $\sum_{i=0}^{k-1} \binom{n}{i} \leq 2^n$, esto es $O(2^n \cdot 1) \in O(2^n)$

Cada hoja (caso base) es $\sum_{i,j \in I} M_{ij} \in O(k^2)$ y al final tendremos $\binom{n}{k}$ hojas (toda combinación posible de tamaño k de n elementos), con $O(\binom{n}{k}) \in O(2^n)$, entonces la complejidad de las hojas es $O(2^n \cdot k^2)$

La complejidad final es: $O(2^n \cdot 1 + 2^n \cdot k^2) \in O(2^n \cdot k^2)$

c)

Una poda podría ser teniendo una solución completa (i.e. $|I| = k$) ver que la suma restante para una solución parcial I', $|I|' \leq k$ puede llegar a ser mayor a la suma de la solución máxima actual y dejar de recorrer la rama caso contrario.