3. Queremos encontrar la suma de los elementos de un multiconjunto de números naturales. Cada suma se realiza exactamente entre dos números x e y y tiene costo $x + y$.

   Por ejemplo, si queremos encontrar la suma de $\{1, 2, 5\}$ tenemos 3 opciones:

   - $1 + 2$ (con costo 3) y luego $3 + 5$ (con costo 8), resultando en un costo total de 11;
   - $1 + 5$ (con costo 6) y luego $6 + 2$ (con costo 8), resultando en un costo total de 14;
   - $2 + 5$ (con costo 7) y luego $7 + 1$ (con costo 8), resultando en un costo total de 15.

Queremos encontrar la forma de sumar que tenga costo mínimo, por lo que en nuestro ejemplo la mejor forma sería la primera.

$a)$ Explicitar una estrategia golosa para resolver el problema.

S es nuestro multiconjunto

```
SumaGolosa(S):
  S' = min-heap(S)
  total = 0

  while(|S'| > 1):
    x = S'.extract()
    y = S'.extract()

    S'.insert(x+y)
    total+= x+y

  return total
```

Invariante: En el k-ésimo paso, nuestra solución G toma los elementos minimos $x, y \in S$, hacemos:

$$S - \{x, y\} \wedge_L S \cup (x + y) \quad \text{y} \quad G_k = x + y + \sum_{i<k} G_i$$

$b)$ Demostrar que la estrategia propuesta resuelve el problema.

Invariante: en la k-ésima iteración, $x, y \in S_k$ son reemplazados por $x + y$ y la suma acumulada es:

$$G_k = x + y + \sum_{i<k} G_i$$

Por inducción fuerte sobre la cantidad de combinaciones parciales:

Caso base $k = 1$

Tenemos solo 2 elementos, por ende son los mínimos, y la suma acumulada es $x + y$. Trivialmente $G = O$

Paso recursivo $k - 1 \Rightarrow k$

H.I: Suponemos que existe solución óptima $O' = \{g_1, ..., g_{k-1}, o_k, ..., o_n\}$

Queremos ver que si intercambiamos o_k por g_k en O', sigue siendo óptima.

Recordemos:

$$G_k = x + y + \sum_{i<k} G_i, \text{con } x, y \text{ los minimos de } S_k$$

y por **H.I.** G es tan buena como O' hasta el $k-1$-ésimo paso, por lo tanto en el paso k tenemos:

$$x + y + \sum_{i<k} G_i \leq v + w + \sum_{i<k} G_i$$

donde O' toma $v, w \in S_k^O$ y G toma $x, y \in S_k^G$

Esto se reduce a $x + y \leq v + w$, entonces G_k es al menos tan buena como $O'_k \implies G$ es óptima.

---

c) Implementar esta estrategia en un algoritmo iterativo. **Nota:** el mejor algoritmo simple que conocemos tiene complejidad $\mathcal{O}(n \log n)$ y utiliza una estructura de datos que implementa una secuencia ordenada.

```java
public static int sg(List<Integer> S) {
    PriorityQueue<Integer> heap = new PriorityQueue<>(S);
    int x,y;
    int total = 0;

    while(heap.size() > 1) {
        x = heap.poll();
        y = heap.poll();

        heap.offer(x+y);
        total += x+y;
    }

    return total;
}
```