SumaGolosa

15. 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.
    b) Demostrar que la estrategia propuesta resuelve el problema.
    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.

(a) ¿Cómo armo mi solución? Voy tomando elementos de X y los sumo

Estrategia greedy: Si no empecé tomo el mínimo i de X. Luego, el mínimo j de X\{i}. Los sumo y me guardo el costo y el valor que sumaron. Luego, saco i y j de X y me aseguro de no tomarlos de nuevo. Si no era el primer elemento tomo el mínimo que no haya agarrado de X y se lo sumo a la variable donde llevo acumulando la suma parcial y ese resultado se lo sumo al costo

Veamos que la estrategia es correcta, es decir, que llego a una solución óptima

Para eso vamos a probar $p(i)$: $\forall i \leq n$ (donde n = # de mi multiconjunto X) mi solución golosa $v^* \leq S^*$ donde S^* es una solución candidata.

Caso base: $i = 1$ como $v^* = \min\{X\} \rightarrow v^* \leq x \quad \forall x \in X \rightarrow$ como $S^* = x$, $v^* \leq S^*$

Caso inductivo $p(i) \rightarrow p(i+1) \quad \forall i \in \mathbb{N}$

Si $i \geq n \rightarrow$ trivial $p(i) \rightarrow p(i+1)$ es verdadero siempre

Si $i<n$, veamos que vale

Sean v_j^* y s_j^* el valor de ambas soluciones después de sumar el j-ésimo elemento de X

$$\rightarrow v_{i+1}^* = v_i^* + \min\{X \setminus \underbrace{\text{los elementos de } X \text{ que ya sumé}}_{(*)}\} \overset{HI}{\leq} s_i^* + \min\{X \setminus *\}$$

$$\leq s_i^* + x \quad \forall x \,/\, x \in X \setminus * = s_{i+1}^*$$

© SG (vector X) {

sort (X); ordena X → $\Theta(n \log n)$ con $n = |X|$

int costo = 0;

int valorAcum = X[0];

for (int j = 1; j < n; j++) { $\Theta(n)$

valorAcum += X[j]

costo += valorAcum

}

return costo

Complejidad temporal: $\Theta(n \log n)$

" espacial: $\Theta(1)$

Pensamos mal el algoritmo y la estrategia: el enunciado estaba antiguo

↳ Es muy parecido al algo de Huffman

Sea {3,3,4,5,10}

SG hace 3+3=6 y lo reinserta {6,4,5,10}
hace 4+5=9 y lo reinserta {6,9,10}
hace 6+9=15 y lo reinserta {10,15}.

La estrategia golosa va a ser: tomar los 2 elementos ⊕ chicos de X, sumarlos y reinsertarlo al conjunto. Así hasta que $X=\emptyset$

Puedo aprovechar los heaps

```
SG( vector X){
  heap= minheapify(X) Θ(n) → algo de Floyd
  Costo = 0
  While (heap ≠ ∅){  Θ(n) → min heap va a tener n elementos
     min1= min_heap O(1)
     eliminar(min1); Θ(log n)
     min2= min_heap O(1)
     eliminar (min2)
     Costo+= min1+min2
     añadir (min1+min2) Θ(log n)
  }
  return costo
```