Algorithm 1 Algoritmo de Kadane

Require: $A : Array \quad n : largo\ de\ A \quad xa, xb : int\ pointer$
Ensure: *Suma máxima del subarreglo entre xa y xb*

```
sumaLocal ← A[0], sumaTotal ← A[0];
i ← 0, aux ← 0, xa ← 0, xb ← 0;
for i < n do
    if A[i] > sumaLocal + A[i] then
        sumaLocal ← A[i];
        aux ← i;
    else
        sumaLocal ← sumaLocal + A[i];
    end if
    #Si existe una suma mayor a la anterior, se actualiza junto con las nuevas
    posiciones del arreglo.
    if sumaTotal < sumaLocal then
        sumaTotal ← sumaLocal;
        xa ← aux;
        xb ← i;
    end if
    i ← i + 1;
end for
return sumaTotal;
```

Algorithm 2 Algoritmo de Kadane para matrices

Require: $M : Matrix\ N * N \quad n : largo\ de\ M \quad xa, xb, ya, yb : int\ pointer.$
Ensure: *Suma máxima del subrectangulo entre xa y xb, ya e yb.*

```
sumaMax ← int_min, sumAux;
xaAux ← 0, xbAux ← 0;
temp ← Array;
i ← 0;
for i < n do
    temp ← new Array[n];
    j ← i;
    for j < n do
        k ← 0;
        for k < n do                    # Se acumulan los valores entre i y j
            temp[k] ← temp[k] + M[k][j];
            k ← k + 1;
        end for
        sumAux ← kadaneArray(temp, n, &xaAux, &xbAux);
        if sumAux > sumMax then         # Si existe una suma mayor.
            sumMax ← sumAux;            # Se actualiza la suma
            xa ← xaAux, xb ← xbAux;     # Se act. las filas de la suma.
            ya ← i, yb ← j;             # Se act. las columnas.
        end if
        j ← j + 1;
    end for
    i ← i + 1;
end for
```

Algorithm 3 Algoritmo de Kadane para cubos

Require: $C : Cube\ N * N * N \quad n : largo\, de\, C \quad xa, xb, ya, yb, za, zb : int\, pointer.$
Ensure: *Suma máxima del subcubo entre xa y xb, ya e y za y zb.*

```
sumMax ← int_min, sumAux
xaAux, xbAux, yaAux, ybAux : variables auxiliares para limites
temp ← Matriz[n][n];
i ← 0;
for i < n do
    temp ← new Matriz[n][n];
    j ← 0
    for j < n do
        temp[k][l] ← C[k][l][j], j = 0...n − 1, k = 0...n − 1;
        sumAux = kadaneMatrix(temp, n, &xaAux, &xbAux, &yaAux, &ybAux);
        if sumMax < sumAux then      # Si existe una suma mayor.
            sumMax ← sumAux;         # Se actualiza la suma
            xa ← xaAux, xb ← xbAux;  # Se act. las filas de la suma.
            ya ← yaAux, yb ← ybAux;  # Se act. las columnas.
            za ← i, zb ← j;          # Se act. el alto del subcubo.
        end if
        j ← j + 1;
    end for
    i ← i + 1;
end for
```