```
function vector = nExpand(n,x)
    for i=1:n
        vector(i) = x
    end
    vector = vector'
endfunction

function val = minsum(vector) //dado los vecinos de un vertice (se guarda en un vector de 1xn filas)
    val = %inf
    [one,n] = size(vector)
    while vector <> [] //mientras el vector no sea vacio
        primero = vector(1)
        vector(1) = [] //elimino el primer elemento del vector
        for i=1:n-1
            suma = primero + vector(i)
            if suma <= val then
                val = suma
            end
        end
        n = n-1
    end
endfunction

function CotaInf = uno_arbol(A, v1)
// Ejecuta el algoritmo del 1-arbol óptimo sobre la instancia TSP dada
// Entrada:
//  A = matríz de distancias de la instancia TSP
//  v1 = vértice inicial
// Salida:
//  cotainf = valor de la cota inferior

[nodos, zzz] = size(A); // en "nodos" se guarda la cantidad de vértices de la instancia
vector = A(v1,:) //sea vector la fila de los vecinos de v1
vector(v1) = [] //elimino a v1 del vector, ya que figura infinito
a = minsum(vector) // calculo min {c_e + c_f, donde e y f son aristas incidentes en v1}
A(v1,:) = []
A(:,v1) = []
[G,costos] = genera_grafo_costos(A) // A ahora es la matriz de G-{v1}
[F,b] = kruskal(G,costos) // calculo el costo del arbol recubridor minimo de G
CotaInf = a + b //la cota inferior sera a + b
endfunction
```