```
function [tour, valor] = insercion_mas_cercana(A)
// Ejecuta el algoritmo de la Inserción más cercana sobre la instancia TSP dada
// Entrada:
//  A = matríz de distancias de la instancia TSP
// Salida:
//  tour = vector con los vértices a ser recoridos por el tour
//  valor = valor del tour generado

[nodos, zzz] = size(A); // en "nodos" se guarda la cantidad de vértices de la instancia

vertices_permitidos = 1:nodos; // al principio todos los vertices son permitidos

// Buscamos la menor distancia entre 2 vértices, y agregamos ambos al tour
// (basta con buscar sobre los pares (u, v) tales que u < v)
minima_dist = %inf;
v1 = -1;
v2 = -1;
for u=1:(nodos-1)
    for v=(u+1):nodos
        if (minima_dist > A(u,v)) then
            minima_dist = A(u,v);
            v1 = u;
            v2 = v;
        end
    end
end
tour = [v1 v2];

// Prohibimos el uso de los vértices v1 y v2 en el futuro
vertices_permitidos(find(vertices_permitidos==v1)) = [];
vertices_permitidos(find(vertices_permitidos==v2)) = [];

for longitud=2:(nodos-1)
    // Por cada vertice permitido elegimos el vertice w mas cercano del tour
    wvector = -ones(1,nodos); // wvector va a tener el w correspondiente
                              // a cada v; al comienzo el vector vale -1
    for v=vertices_permitidos
        minima_dist = %inf;
        minimo_w = -1;
        for w=tour
            if (minima_dist > A(v,w)) then
                minima_dist = A(v,w);
                minimo_w = w;
            end
        end
        wvector(v) = minimo_w;
    end

    // Elegimos el próximo vértice "v" cuyo "w" es el MAS CERCANO
    minima_dist = %inf;
    minimo_v = -1;
    for v=vertices_permitidos
        if (minima_dist > A(v,wvector(v))) then
            minima_dist = A(v,wvector(v));
            minimo_v = v;
        end
    end

    // Ahora elegimos donde insertarlo (entre dos vértices del tour que más mejore el tour)
    minima_suma = %inf;
    minimo_j = -1;
    for j=1:longitud
        // En "siguiente" colocamos el siguiente a "j", es decir j+1
        // (excepto para el último vértice del tour, en cuyo caso el siguiente es el primero)
        siguiente = j+1;
        if (j == longitud) then siguiente = 1; end
        // Calculamos la suma de distancias entre el v hallado, y los vertices del tour en "j" y
"siguiente", menos la distancia entre "j" y "siguiente"
        suma = A(tour(j),minimo_v) + A(minimo_v,tour(siguiente)) - A(tour(j),tour(siguiente));
        if (suma < minima_suma) then
            minima_suma = suma;
            minimo_j = j;
```

```
        end
    end

    // Insertamos el (minimo) "v" en la posición (mínimo) "j+1", en el tour; luego prohibimos dicho
vértice.
    tour = [tour(1:minimo_j) minimo_v tour(minimo_j+1:$)];
    vertices_permitidos(find(vertices_permitidos==minimo_v)) = [];
end

// Calcula el valor del tour
valor = 0;
for j=1:nodos
    // En "siguiente" colocamos el siguiente a "j", es decir j+1
    siguiente = j+1;
    if (j == nodos) then siguiente = 1; end
    // Añadimos el costo de ir de "j" al siguiente
    valor = valor + A(tour(j),tour(siguiente));
end

endfunction
```