



# Trabajo Practico de Especificacion

## Especificacion y WP(Weakness Precondition)

2 de septiembre de 2024

Algoritmos y Estructuras de Datos 1

pesutipolimardiano

| Integrante        | LU     | Correo electrónico      |
|-------------------|--------|-------------------------|
| Nievas, Martin    | 453/24 | tinnivas@gmail.com      |
| Bercovich, Maximo | 002/01 | email2@dominio.com      |
| Apellido, Nicolas | 003/01 | email3@dominio.com      |
| Pomsztein, Andy   | 624/24 | pomszteinandy@gmail.com |



**Facultad de Ciencias Exactas y Naturales**  
Universidad de Buenos Aires

Ciudad Universitaria - (Pabellón I/Planta Baja)

Intendente Güiraldes 2610 - C1428EGA

Ciudad Autónoma de Buenos Aires - Rep. Argentina

Tel/Fax: (+54 +11) 4576-3300

<http://www.exactas.uba.ar>

# 1. Enunciados

## 1.1. Especificacion

**1. grandesCiudades** A partir de una lista de ciudades, devuelve aquellas que tienen m´as de 50.000 habitantes.  
**proc grandesCiudades** (in ciudades: seq⟨Ciudad⟩) : seq⟨Ciudad⟩

**2. sumaDeHabitantes:** Por cuestiones de planificacion urbana, las ciudades registran sus habitantes mayores de edad por un lado y menores de edad por el otro. Dadas dos listas de ciudades del mismo largo con los mismos nombres, una con sus habitantes mayores y otra con sus habitantes menores, este procedimiento debe devolver una lista de ciudades con la cantidad total de sus habitantes.

**proc sumaDeHabitantes** (in menoresDeCiudades: seq⟨Ciudad⟩, in mayoresDeCiudades: seq⟨Ciudad⟩) : seq⟨Ciudad⟩

**3. hayCamino:** Un mapa de ciudades esta conformada por ciudades y caminos que unen a algunas de ellas.

A partir de este mapa, podemos definir las distancias entre ciudades como una matriz donde cada celda  $i, j$  representa la distancia entre la ciudad  $i$  y la ciudad  $j$  (Fig. 2). Una distancia de 0 equivale a no haber camino entre  $i$  y  $j$ . Notar que la distancia de una ciudad hacia s´ı misma es cero y la distancia entre  $A$  y  $B$  es la misma que entre  $B$  y  $A$ .

**proc hayCamino** o (in distancias: seq⟨seq⟨Z⟩⟩, in desde: Z, in hasta: Z) : Bool

**4. cantidadCaminosNSaltos:** Dentro del contexto de redes informaticas, nos interesa contar la cantidad de “saltos” que realizan los paquetes de datos, donde un salto se define como pasar por un nodo. Asi como definimos la matriz de distancias, podemos definir la matriz de conexi´on entre nodos, donde cada celda  $i, j$  tiene un 1 si hay un unico camino a un salto de distancia entre el nodo  $i$  y el nodo  $j$ , y un 0 en caso contrario. En este caso, se trata de una matriz de conexi´on de orden 1, ya que indica cu´ales pares de nodos poseen 1 camino entre ellos a 1 salto de distancia. Dada la matriz de conexi´on de orden 1, este procedimineto debe obtener aquella de orden  $n$  que indica cuantos caminos de  $n$  saltos hay entre los distintos nodos. Notar que la multiplicaci´on de una matriz de conexi´on de orden 1 consigo misma nos da la matriz de conexi´on de orden 2, y asi sucesivamente.<sup>4</sup>

**proc cantidadNSaltos** s (inout conexi´on: seq⟨seq⟨Z⟩⟩, in n: Z)

**5. caminoMinimo:** caminoMinimo: Dada una matriz de distancias, una ciudad de origen y una ciudad de destino, este procedimiento debe devolver la lista de ciudades que conforman el camino mas corto entre ambas. En caso de no existir un camino, se debe devolver una lista vacia.

**proc caminoMinimo** (in origen: Z, in destino: Z, in distancias: seq⟨seq⟨Z⟩⟩) : seq⟨Z⟩

## 1.2. WP (Weakest Precondition)

## 2. Resolucion de Ejercicios

**Ejercicio 1:**

**pred sonCiudadesValidas** (in ciudades: seq⟨Ciudad⟩) {

$((\forall i : \mathbb{Z}) (0 \leq i < |ciudades| \longrightarrow_L ciudades[i].habitantes > 0))$

}

**pred sonTodasCiudadesGrandes** (in ciudades: seq⟨Ciudad⟩) {

$((\forall i : \mathbb{Z}) (0 \leq i < |ciudades| \longrightarrow_L ciudades[i].habitantes > 50000))$

}

**aux cantidadCiudadesGrandes** (in ciudades : seq⟨Ciudad⟩) :  $\mathbb{Z} = \sum_{i=0}^{|s|-1} (\text{if } s[i].habitantes > 50000 \text{ then } 1 \text{ else } 0 \text{ fi})$ ;

**proc grandesCiudades** (in ciudades : seq⟨Ciudad⟩) : seq⟨Ciudad⟩

**requiere** {sonCiudadesValidas(ciudades)}

**asegura**  $\{|res| = cantidadCiudadesGrandes(ciudades) \wedge sonTodasCiudadesGrandes(ciudades) \wedge (\forall c : Ciudad) (c \in res \longrightarrow_L c \in ciudades)\}$