



Trabajo práctico 1

Especificación y WP

10 de septiembre de 2024

Algoritmos y Estructuras de Datos - DC - UBA

Grupo AJMS

Integrante	LU	Correo electrónico
Ferechian, Matías	693/23	matifere@gmail.com
Nestmann, Sofía	366/23	sofianestmann@gmail.com
Mirasson, Javier	594/23	javierestebanmn@gmail.com
Ramirez, Ana	931/23	correodeanar@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. Especificación

1.1. grandesCiudades

```
proc grandesCiudades (in ciudades : seq⟨Ciudad⟩) : seq⟨Ciudad⟩{  
  requiere {true}  
  asegura { (∀i : ℤ) (  
    (0 ≤ i < |ciudades|) ∧L ((ciudades[i] ∈ res) →L (ciudades[i]1 > 50000))  
  ) }  
}
```

1.2. sumaDeHabitantes

```
proc sumaDeHabitantes (in menoresDeCiudades : seq⟨Ciudad⟩, in mayoresDeCiudades : seq⟨Ciudad⟩) : seq⟨Ciudad⟩{  
  requiere { (|menoresDeCiudades| = |mayoresDeCiudades|) ∧L ((∀i, j : ℤ≥0) (  
    0 ≤ i, j < |menoresDeCiudades| ∧ menoresDeCiudades[i]0 = mayoresDeCiudades[j]0  
  ) }  
  asegura { (∀n, m : ℤ) (  
    (0 ≤ n, m < |menoresDeCiudades|) ∧L ((menoresDeCiudades[n]0 = mayoresDeCiudades[m]0) ∧  
    (ciudades[n] ∈ res) → ((ciudades[n]1 = menoresDeCiudades[n]1 + mayoresDeCiudades[m]1) ∧  
    (ciudades[n]0 = menoresDeCiudades[n]0))) }  
}
```

1.3. hayCamino

```
proc hayCamino (in distancias : seq⟨seq⟨ℤ⟩⟩, in desde : ℤ, in hasta : ℤ) : Bool{  
  requiere { (∀i, j : ℤ) (  
    (0 ≤ i, j, desde, hasta < |distancias|) ∧L ((i = j) → (distancias[i][j] = 0)) ∧ (distancias[i][j] = distancias[j][i])  
  ) }  
  asegura { res = true ↔ (∃p : seq⟨ℤ⟩) (  
    (p[0] = desde) ∧ (p[|p| - 1] = hasta) ∧ (∀k : ℤ) (  
      (0 ≤ k < |p| - 1) ∧L (distancias[p[k]][p[k + 1]] > 0)  
    )  
  ) }  
}
```

1.4. cantidadCaminosNSaltos

Para la siguiente especificación tendremos en cuenta que: Dada la matriz de orden 1:

$$M_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Si queremos encontrar la matriz M_2 de orden 2, nos queda que:

$$M_2 = M_1 \times M_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 2 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

Luego M_2 contiene la cantidad de 2-saltos asociados a cada par i,j que se encuentre dentro de la matriz

```
proc cantidadCaminosNSaltos (inout conexión : seq⟨seq⟨ℤ⟩⟩, in n : ℤ) : {  
  requiere { n > 0 ∧ (∀i, j : ℤ) (  
    (0 ≤ i, j < |conexión|) ∧L ((i = j) → (conexión[i][j] = 0)) ∧ (conexión[i][j] = conexión[j][i]) ∧ ( 0 ≤ conexión[i][j] ≤ 1 )  
  ) }  
  asegura { (∃p : seq⟨seq⟨seq⟨ℤ⟩⟩⟩) (  
    (∀i, j, k : ℤ) (  
      (0 ≤ k < n - 1) ∧ (0 ≤ i, j < n) ∧L (p[k] = conexión) ∧L  
      (conexión[i][j] = multEntreMatrices(p[k], p[k + 1]))  
    )  
  ) }
```

```

) }
}
aux multEntreMatrices (mUno : seq⟨seq⟨ℤ⟩⟩, mDos : seq⟨seq⟨ℤ⟩⟩) : ℤ = (∀i, j : ℤ) (
  (0 ≤ i, j < |mUno|) ∧L ∑k=1|mUno| mUno[i][k] × mDos[k][j]
);

```

1.5. caminoMinimo

```

proc caminoMinimo (in origen : ℤ, in destino : ℤ, in distancias : seq⟨seq⟨ℤ⟩⟩) : seq⟨ℤ⟩ {
  requiere { true }
  asegura { true }
}

```

2. Demostraciones de correctitud