



Trabajo práctico 1

Especificación y WP

13 de octubre 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 {(|ciudades| > 0) ∧L noHayNombresRepetidos(ciudades)}
  asegura { (∀i : ℤ) (
    (0 ≤ i < |ciudades|) →L ((ciudades[i]1 > 50000) → (ciudades[i] ∈ res)) ∧L estaIncluida(res, ciudades)
  ) }
}
pred noHayNombresRepetidos (lista: seq⟨Ciudad⟩) {
  (∀i, j : ℤ) (
    (0 ≤ i, j < |lista|) →L (lista[i]0 ≠ lista[j]0)
  )
}
pred estaIncluida (s : seq⟨Ciudad⟩, t : seq⟨Ciudad⟩) {
  (∀k : Ciudad) (
    (k ∈ s) → (#apariciones(e, s) ≤ #apariciones(e, t))
  )
}
```

1.2. sumaDeHabitantes

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

1.3. hayCamino

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

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 ∧ matrizCuadradaSimetrica(conexión)) ∧L (∀i, j : ℤ) (
    (0 ≤ i, j < |conexión|) →L (0 ≤ conexión[i][j] ≤ 1))
} ∧ (C0 = conexión)}
  asegura {(∃p : seq⟨seq⟨seq⟨ℤ⟩⟩⟩) (
    (p[0] = C0) →L (∀k : ℤ) (
      ((0 < k < n) →L (multEntreMatrices(p[k - 1], p[k - 1], p[k]))) ∧L
      (multEntreMatrices(p[n - 1], p[n - 1], conexión))
    )
  )}
}

pred multEntreMatrices (mUno : seq⟨seq⟨ℤ⟩⟩, mDos : seq⟨seq⟨ℤ⟩⟩, resu : seq⟨seq⟨ℤ⟩⟩) {
  (∀i, j : ℤ) (
    (0 ≤ i, j < |resu|) →L (resu[i][j] = ∑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 {matrizCuadradaSimetrica(distancias) ∧ (0 ≤ origen, destino < |distancias|)}
  asegura {
    (existeCamino(distancias, origen, destino) ∧L
    (menorCamino(distancias, origen, destino) = res)) ∨
    ((¬existeCamino(distancias, origen, destino) ∨ (origen = destino)) →L res = [ ])}
  }
  aux todosLosCaminos (distancias : seq⟨seq⟨ℤ⟩⟩, origen : ℤ, destino : ℤ) : seq⟨seq⟨ℤ⟩⟩ = (∃!m : seq⟨seq⟨ℤ⟩⟩) (
    (|m| > 1) ∧L (∃p : seq⟨ℤ⟩) (
      ((|p| > 1) ∧L ((p[0] = origen) ∧ (p[|p| - 1] = destino) ∧ (∀k : ℤ) (
        (0 ≤ k < |p| - 1) →L (distancias[p[k]][p[k + 1]] > 0)
      ))) →L (p ∈ m)
    )
  ) ;
  aux menorCamino (distancias : seq⟨seq⟨ℤ⟩⟩, origen : ℤ, destino : ℤ) : seq⟨ℤ⟩ = (∃p : seq⟨ℤ⟩) (
    (∃i : ℤ) (
      (0 ≤ i < |todosLosCaminos(distancias, origen, destino)|) ∧L (∀j : ℤ) (
        ((0 ≤ j < |todosLosCaminos(distancias, origen, destino)|) →L
        ((∑n=1|todosLosCaminos(distancias, origen, destino)[i]|-1 distancias[n - 1][n]) ≤
        (∑n=1|todosLosCaminos(distancias, origen, destino)[j]|-1 distancias[n - 1][n])) →L
        (p = todosLosCaminos(distancias, origen, destino)[i])
      )
    )
  ) ;
  aux existeCamino (distancias : seq⟨seq⟨ℤ⟩⟩, desde : ℤ, hasta : ℤ) : Bool = true ↔ (∃p : seq⟨ℤ⟩) (
    (|p| > 1) ∧L ((p[0] = desde) ∧ (p[|p| - 1] = hasta) ∧ (∀k : ℤ) (
      (0 ≤ k < |p| - 1) →L (distancias[p[k]][p[k + 1]] > 0)
    ))
  ) ;
}

```

2. Demostraciones de correctitud

2.1. Demostración de implementación

Definimos como precondition y postcondition de nuestro programa:

$$P \equiv \{(\exists i : \mathbb{Z})(0 \leq i < |ciudades|) \wedge_L ciudades[i].habitantes > 50,000 \wedge (\forall j : \mathbb{Z}) (0 \leq j < |ciudades| \longrightarrow_L ciudades[j].habitantes \geq 0) \wedge (\forall i, j : \mathbb{Z}) (0 \leq i < j < |ciudades| \longrightarrow_L ciudades[j].nombre \neq ciudades[i].nombre)\}$$

$$Q \equiv \{\sum_{i=0}^{|ciudades|-1} ciudades[i].habitantes\}$$

Para demostrar la correctitud del programa probamos que:

$\{P\}S1;S2;S3\{Q\}$ es válida, con:

```
S1 ≡ res := 0
S2 ≡ i := 0
S3 ≡
while (i < ciudades.length) do
  res = res + ciudades[i].habitantes
  i = i + 1
end while
```

Se busca por monotonía demostrar que:

1. $P \longrightarrow wp\{S1; S2, P_c\}$
2. $P_c \longrightarrow wp\{S3, Q_c\}$

Siendo que Q_c coincide con el final del código, $Q_c \equiv Q$

Esto permite demostrar que $P \longrightarrow wp\{S1;S2;S3,Q\}$ es verdadera por lo cual la Tripla de Hoare es válida

Para demostrar $P_c \longrightarrow wp\{S3, Q_c\}$ se prueba la correctitud del ciclo mediante:

Teorema del Invariante y Teorema de Terminación

Teorema del Invariante:

Si existe un predicado I tal que:

1. $P_c \longrightarrow I$
2. $\{I \wedge B\}S\{I\}$
3. $I \wedge \neg B \longrightarrow Q_c$

Entonces el ciclo es parcialmente correcto respecto de la especificación

■ Definimos

$$P_c \equiv \{res = 0 \wedge i = 0\}$$

$$Q_c \equiv \{res = \sum_{i=0}^{|ciudades|-1} ciudades[i].habitantes\}$$

$$B \equiv \{i < |ciudades|\}$$

$$I \equiv \{0 \leq i \leq |ciudades| \wedge res = \sum_{j=0}^{i-1} ciudades[j].habitantes\}$$

Según el teorema del invariante

1. $P_c \longrightarrow I$

$$■ \quad 0 \leq i \leq |ciudades| \equiv 0 \leq 0 \leq |ciudades|$$

$$■ \quad res = 0 \wedge i = 0 \longrightarrow res = \sum_{j=0}^{i-1} ciudades[j].habitantes \equiv \sum_{j=0}^{0-1} ciudades[j].habitantes = 0$$

$P_c \longrightarrow I$ es válido

Ahora demuestro que:

$$2. \{I \wedge B\}S\{I\} \longleftrightarrow I \wedge B \longrightarrow wp(S, I)$$

$$\begin{aligned}
& \blacksquare wp(S, I) \equiv wp(\mathbf{S1}; \mathbf{S2}, I) \\
& \equiv wp(\mathbf{S1}, wp(\mathbf{S2}, I)) \\
& \equiv wp(\mathbf{S1}; i:=i+1, I) \\
& \equiv wp(res := res + ciudades[j].habitantes; i:=i+1, I) \\
& \equiv wp\left(res := res + ciudades[j].habitantes, wp\left(i := i, (0 \leq i \leq |ciudades|) \wedge res = \sum_{j=0}^{i-1} ciudades[j].habitantes\right)\right) \\
& \equiv wp\left(res := res + ciudades[j].habitantes, 0 \leq i+1 \leq |ciudades| \wedge \sum_{j=0}^{i+1-1} ciudades[j].habitantes\right) \\
& \equiv wp\left(res := res + ciudades[j].habitantes, -1 \leq i \leq |ciudades| - 1 \wedge \sum_{j=0}^i ciudades[j].habitantes\right) \\
& \equiv wp\left(res := res + ciudades[j].habitantes, i < |ciudades| \wedge \sum_{j=0}^{i-1} ciudades[j].habitantes + ciudades[i].habitantes\right) \\
& \equiv i < |ciudades| \wedge res + ciudades[j].habitantes = \sum_{j=0}^{i-1} ciudades[j].habitantes + ciudades[i].habitantes \\
& \equiv i < |ciudades| \wedge res = \sum_{j=0}^{i-1} ciudades[j].habitantes
\end{aligned}$$

$$\text{Ahora pruebo } I \wedge B \longrightarrow wp(S, I) \quad I \wedge B \equiv 0 \leq i \leq |ciudades| \wedge res = \sum_{j=0}^{i-1} ciudades[j].habitantes \wedge i < |ciudades|$$

$$\equiv i < |ciudades| \wedge \sum_{j=0}^{i-1} ciudades[j].habitantes$$

$$3. \{I \wedge \neg B\} \longrightarrow Qc$$

$$\begin{aligned}
& I \wedge \neg B \\
& \equiv 0 \leq i \leq |ciudades| \wedge res = \sum_{j=0}^{i-1} ciudades[j].habitantes \wedge \neg(i < |ciudades|) \\
& \equiv 0 \leq i \leq |ciudades| \wedge res = \sum_{j=0}^{i-1} ciudades[j].habitantes \wedge i \geq |ciudades| \\
& \equiv \sum_{j=0}^{|ciudades|-1} ciudades[j].habitantes
\end{aligned}$$

Por el Teorema del Invariante podemos concluir que el ciclo es parcialmente correcto respecto de la especificación
Teorema de Terminación

$$\begin{aligned}
& 4. \{I \wedge B \wedge fv = V_o\}S\{fv < V_o\} \\
& 5. (I \wedge fv \leq 0) \longrightarrow \neg B
\end{aligned} \tag{1}$$

Elijo como función variante:

$$fv = |ciudades| - i \tag{2}$$

■ Verifico 4.

$$\begin{aligned}
& \{I \wedge B \wedge fv = V_o\}S\{fv < V_o\} \longleftrightarrow \{I \wedge B \wedge fv = V_o\} \longrightarrow wp(S, fv < V_o) \\
& wp(S, fv < V_o) \equiv wp(res := res + ciudades[i].habitantes; i := i+1, |ciudades| - i < V_o) \\
& \equiv wp(res := res + ciudades[i].habitantes, wp(i := i+1; |ciudades| - i < V_o)) \\
& \equiv wp(res := res + ciudades[i].habitantes, |ciudades| - 1 - i < V_o)
\end{aligned} \tag{3}$$

Puedo ignorar la res ya que no es relevante en este caso

$$\begin{aligned}
& \equiv |ciudades| - 1 - i < |ciudades| - i \\
& -1 < 0
\end{aligned}$$

■ Verifico 5.

$$(I \wedge fv \leq 0) \longrightarrow \neg B$$

$$(I \wedge fv \leq 0) \equiv 0 \leq i \leq |ciudades| \wedge res = \sum_{j=0}^{i-1} ciudades[j].habitantes \wedge |ciudades| - i \leq 0$$

Ignoro res

$$\equiv i \leq |ciudades| \wedge |ciudades| \leq i \iff |ciudades| = i \longrightarrow \neg(i < |ciudades|)$$

$$\equiv |ciudades| = i \longrightarrow \neg(i < |ciudades|)$$

(4)

Queda verificado el Teorema de Terminación por lo que podemos concluir que el ciclo termina. Como ambos teoremas se cumplen, **el ciclo es correcto**.

Luego me falta ver que $P \longrightarrow wp(S1; S2, P_c)$

$$\begin{aligned} wp(S1; S2, P_c) &\equiv wp(res := 0; i := 0, P_c) \equiv wp(res := 0, def(i) \wedge_L P_{c_i}^0) \equiv wp(S2, res = 0 \wedge 0 = 0) \\ &\equiv wp(S2, true \wedge_L res = 0) \equiv def(res) \wedge_L P_{c_{res}}^0 \equiv 0 = 0 \equiv true \end{aligned}$$

Luego, como $P \longrightarrow true$ es tautología, siempre se cumple

Queda demostrado que **la tripla $\{P\}S1;S2;S3\{Q\}$ es válida**

2.2. Demostracion res > 50000

Se utiliza la misma definición del ejercicio anterior para P, B, fv . Se redefinen:

$$I \equiv \{0 \leq i \leq |ciudades| \wedge res = \sum_{j=0}^{i-1} ciudades[j].habitantes \wedge (\exists h : \mathbb{Z})(0 \leq h < |ciudades|) \wedge_L ciudades[h].habitantes >$$

$$50,000 \wedge (\forall j : \mathbb{Z}) (0 \leq j < |ciudades|) \longrightarrow_L (ciudades[j].habitantes \geq 0)\}$$

$$Q \equiv \{res = \sum_{i=0}^{|ciudades|-1} ciudades[i].habitantes \wedge res > 50000\}$$

Se procede de manera similar al ejercicio anterior probando por monotonía la validez del programa y la tripla de Hoare.

Resta chequear la validez del ciclo con el nuevo invariante y postcondicion.

Teorema del invariante

$$P_c \longrightarrow I$$

Como se probó en el punto anterior, las implicancias para $0 \leq i < |ciudades|$ y $res = \sum_{j=0}^{i-1} ciudades[j].habitantes$

se analiza lo agregado al invariante. Como lo agregado esta presente en P_c queda probado trivialmente que $P_c \longrightarrow I$

$$\{I \wedge B\}S\{I\} \iff I \wedge B \longrightarrow wp(S, I)$$

Lo agregado al invariante no afecta a la implicancia. Luego vale que: $I \wedge B \longrightarrow wp(S, I)$

$$\{I \wedge \neg B\} \longrightarrow Q_c$$

$$I \wedge \neg B \equiv 0 \leq i \leq |ciudades| \wedge res = \sum_{j=0}^{i-1} ciudades[j].habitantes \wedge (\exists i : \mathbb{Z})(0 \leq i < |ciudades|) \wedge_L ciudades[i].habitantes > 50,000$$

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

$$\equiv 0 \leq i \leq |ciudades| \wedge res = \sum_{j=0}^{i-1} ciudades[j].habitantes \wedge (\exists i : \mathbb{Z})(0 \leq i < |ciudades|) \wedge_L ciudades[i].habitantes > 50,000$$

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

$$\equiv i = |ciudades| \wedge res = \sum_{j=0}^{i-1} ciudades[j].habitantes \wedge (\exists i : \mathbb{Z})(0 \leq i < |ciudades|) \wedge_L ciudades[i].habitantes > 50,000$$

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

$$\equiv res = \sum_{j=0}^{|ciudades|-1} ciudades[j].habitantes \wedge (\exists i : \mathbb{Z})(0 \leq i < |ciudades|) \wedge_L ciudades[i].habitantes > 50,000$$

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

Sea k una posición en la sucesión $ciudades$ que cumple con la precondition de tener mas de 50000 habitantes

$$\sum_{j=0}^{k-2} ciudades[j].habitantes \text{ Para los elementos a la izquierda de la } k\text{-esima posición}$$

n para el numero de habitantes de la k-esima posición

$\sum_{j=k+1}^{i-1} ciudades[j].habitantes$ Para los elementos a la derecha de la k-esima posición

Luego

$$res = \sum_{j=0}^{|ciudades|-1} ciudades[j].habitantes \wedge (\exists i : \mathbb{Z})(0 \leq i < |ciudades|) \wedge_L ciudades[i].habitantes > 50,000$$

$$\wedge (\forall j : \mathbb{Z}) (0 \leq j < |ciudades| \longrightarrow_L ciudades[j].habitantes \geq 0) \equiv res = \sum_{j=0}^{|ciudades|-1} ciudades[j].habitantes \wedge$$

$$res > 50000$$

$$\{I \wedge \neg B\} \longrightarrow Q_c$$

Las implicancias del teorema de terminación no se ven afectadas por los cambios en el invariante y la postcondicion. Luego siguen

Habiendo analizado todos los cambios y siguiendo los razonamientos del ejercicio anterior,

queda demostrado que el resultado devuelto es mayor a 50000