



DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Descripción del tp

Subtítulo del tp

13 de septiembre de 2024

Materia de la carrera

Grupo 42

Integrante	LU	Correo electrónico
Krivonosoff, Thiago	310/24	thiagokribas@gmail.com
Pelli, Agustin	002/01	email2@dominio.com
Miguel, Facundo	003/01	email3@dominio.com
Montenegro, Ulises	477/24	ulinicolasmonte@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. Punto 1

1.1. Ejercicio 1

```
proc grandesCiudades (in ciudades : seq⟨Ciudad⟩) : seq⟨Ciudad⟩
  requiere {noRepetidos(ciudades) ∧ noHabitantesNegativos(ciudades)}
  asegura {|res| ≤ |ciudades|}
  asegura {(∀elem : Ciudad) ((elem.habitantes > 50000 ∧ elem ∈ ciudades) ↔ elem ∈ res)}

pred noRepetidos (in ciudades: seq⟨Ciudad⟩) {
  (∀i : ℤ) (0 ≤ i < |ciudades|) →L (∀j : ℤ) (0 ≤ j < |ciudades|) →L (i ≠ j → ciudades[i].nombre ≠
  ciudades[j].nombre)
}

pred noHabitantesNegativos (in ciudades: seq⟨Ciudad⟩) {
  (∀i : ℤ) (0 ≤ i < |ciudades|) →L ciudades[i].habitantes >= 0
}
```

1.2. Ejercicio 2

```
proc sumaDeHabitantes (in menoresDeCiudades: seq⟨Ciudad⟩, in mayoresDeCiudades: seq⟨Ciudad⟩) : seq⟨Ciudad⟩
  requiere {noRepetidos(menoresDeCiudades) ∧ noRepetidos(mayoresDeCiudades)}
  requiere {noHabitantesNegativos(menoresDeCiudades) ∧ noHabitantesNegativos(mayoresDeCiudades)}
  requiere {mismosElementos(menoresDeCiudades, mayoresDeCiudades)}
  requiere {|menoresDeCiudades| = |mayoresDeCiudades|}
  asegura {|res| = |mayoresDeCiudades|}
  asegura {(∀elem : Ciudad) ((elem ∈ res) →
  (∀i : ℤ) (0 ≤ i < |res|) →L (∃j : ℤ) (0 ≤ j < |res|) ∧L
  mayoresDeCiudades[i].nombre = menoresDeCiudades[j].nombre →
  res[i].nombre = mayoresDeCiudades[i].nombre ∧
  res[i].habitantes = mayoresDeCiudades[i].habitantes + menoresDeCiudades[j].habitantes)}

pred mismosElementos (in s1: seq⟨Ciudad⟩, in s2: seq⟨Ciudad⟩) {
  (∀i : ℤ) (0 ≤ i < |s1|) →L (∃j : ℤ) (0 ≤ j < |s1|) ∧L s1[i].nombre = s2[j].nombre
}
```

1.3. Ejercicio 3

```
proc hayCamino (in distancia: seq⟨seq⟨ℤ⟩⟩, in desde: ℤ, in hasta: ℤ) : Bool
  requiere {esCuadrada(distancia)}
  requiere {0 ≤ desde < |distancia|}
  requiere {0 ≤ hasta < |distancia|}
  requiere {filaIgualColumna(distancia)}
  requiere {matrizTodosPositivos(distancia)}
  asegura {res = True ↔ (∃sec : seq⟨ℤ⟩) (|sec| > 1) ∧L secuenciaEsCamino(distancia, sec, desde, hasta)}

pred secuenciaEsCamino (in distancia: seq⟨seq⟨ℤ⟩⟩, in sec: seq⟨ℤ⟩, in desde: ℤ, in hasta: ℤ) {
  sec[0] = desde ∧ sec[|sec| - 1] = hasta ∧ (∀i : ℤ) (0 ≤ i < |sec|) →L
  0 ≤ sec[i] < |distancia| ∧ todosConexionAnterior(sec, distancia)
}

pred todosConexionAnterior (in sec: seq⟨ℤ⟩, in mat: seq⟨seq⟨ℤ⟩⟩) {
  (∀i : ℤ) (1 ≤ i < |sec|) →L mat[sec[i]][sec[i - 1]] ≠ 0
}

pred esCuadrada (in mat: seq⟨seq⟨ℤ⟩⟩) {
  (∀i : ℤ) (0 ≤ i < |mat|) →L |mat| = |mat[i]|
}

pred filaIgualColumna (in mat: seq⟨seq⟨ℤ⟩⟩) {
  (∀i : ℤ) (0 ≤ i < |mat|) ∧ (∀j : ℤ) (0 ≤ j < |mat|) →L mat[i][j] = mat[j][i]
}

pred matrizTodosPositivos (in mat: seq⟨seq⟨ℤ⟩⟩) {
  (∀i : ℤ) (0 ≤ i < |mat|) ∧ (∀j : ℤ) (0 ≤ j < |mat|) →L mat[i][j] >= 0
}
```

1.4. Ejercicio 4

```

proc cantidadCaminosNSaltos (inout conexion: seq⟨seq⟨Z⟩⟩, in n: Z)
  requiere {1 ≤ n}
  requiere {esCuadrada(conexion)}
  requiere {filaIgualColumna(conexion)}
  requiere {(∀i : Z) (0 ≤ i < |conexion|) →L (∀j : Z) (0 ≤ j < |conexion|) →L conexion[i][j] ∈ [0, 1]}
  requiere {conexion = C0}
  asegura {(∃sec : seq⟨seq⟨Z⟩⟩) (|sec| = n) ∧ sec[0] = C0 ∧
    (∀i : Z) (1 ≤ i < |sec|) →L sec[i] = multiplicarMatrices(C0, sec[i - 1]) →L conexion = sec[|sec| - 1]}

aux inversa (in mat: seq⟨seq⟨Z⟩⟩) : seq⟨seq⟨Z⟩⟩ =
  (∀i : Z) (0 ≤ i < |mat|) →L (∀j : Z) (0 ≤ j < |mat|) res[j][i] = mat[i][j];
aux multiplicarMatrices (in mat1: seq⟨seq⟨Z⟩⟩, in mat2: seq⟨seq⟨Z⟩⟩) : seq⟨seq⟨Z⟩⟩ =
  (∀i : Z) (0 ≤ i < |mat1|) →L (∀j : Z) (0 ≤ j < |mat1[i]|) →L
  res[i][j] = productoEscalar(mat1[i], inversa(mat2)[j]);
aux productoEscalar (∈ fila : seq⟨Z⟩, ∈ col : seq⟨Z⟩) : Z =
  (∀i : Z) (0 ≤ i < |fila|) →L (∀j : Z) (0 ≤ j < |col|) →L res = ∑i=0|fila|-1 fila[i] * fila[j];

```

1.5. Ejercicio 5

```

proc caminoMinimo (in origen: Z, in destino: Z, in distancias: seq⟨seq⟨Z⟩⟩) : seq⟨Z⟩
  requiere {esCuadrada(distancias)}
  requiere {0 ≤ destino, origen < |distancias|}
  requiere {filaIgualColumna(distancias)}
  requiere {matrizTodosPositivos(distancias)}
  asegura {(∃s1 : seq⟨Z⟩) (|s1| > 1) ∧ res = s1 ↔ (∀s2 : seq⟨Z⟩) (|s2| > 1) →L
    (secuenciaEsCamino(distancias, s1, origen, destino) ∧ secuenciaEsCamino(distancias, s2, origen, destino)) →L
    (longitudCamino(s1) ≤ longitudCamino(s2, distancias))}

aux longitudCamino (in sec: Z, in distancias: Z) : Z = ∑i=1|sec|-1 mat[sec[i]][sec[i - 1]];

```

2. Punto 2

2.1. Ejercicio 1

Demostramos que la implementación es correcta con correcta con respecto a la especificación dada mediante teorema de invariante y teorema de terminación.

Por teorema del invariante primero debemos demostrar los siguientes puntos:

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

- $P_c \rightarrow I$
- $\{I \wedge B\} S \{I\}$
- $I \wedge \neg B \rightarrow Q_c$

Primer paso Probamos primero la implicación de la precondition del ciclo hacia el invariante:

$$P_c \rightarrow I$$

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

$$0 \leq 0 \leq |\text{ciudades}| \wedge \sum_{j=0}^{0-1} \text{ciudades}[j].\text{habitantes} = 0$$

$$\text{True} \wedge \text{True}$$

$$\text{True}$$

Segundo paso Ahora probamos que vale la siguiente tripla de Hoare:

$$\{I \wedge B\} S \{I\}$$

Hacemos uso del axioma.

$$wp(S, I) \cong wp(S_1, wp(S_2, I))$$

$$wp(S_2, I) \cong def(S_2) \wedge_L I_i^{i:=i+1}$$

$$wp(S_2, I) \cong True \wedge 0 \leq i + 1 \leq |ciudades| \wedge_L \sum_{j=0}^i ciudades[j].habitantes = res$$

Terminamos de definir el wp de S2:

$$wp(S_2, I) \cong 0 \leq i + 1 \leq |ciudades| \wedge_L \sum_{j=0}^i ciudades[j].habitantes = res$$

Ahora definimos el wp de S1:

$$wp(S_1, wp(S_2, I)) \cong wp(res = res + ciudades[i].habitantes, 0 \leq i + 1 \leq |ciudades| \wedge_L \sum_{j=0}^i ciudades[j].habitantes = res)$$

$$wp(S_1, wp(S_2, I)) \cong def(res = res + ciudades[i].habitantes) \wedge_L I_{res}^{res+ciudades[i].habitantes}$$

$$wp(S_1, wp(S_2, I)) \cong True \wedge_L 0 \leq i + 1 \leq |ciudades| \wedge_L \sum_{j=0}^i ciudades[j].habitantes = res + ciudades[i].habitantes$$

$$wp(S_1, wp(S_2, I)) \cong 0 \leq i + 1 \leq |ciudades| \wedge_L \sum_{j=0}^i ciudades[j].habitantes = res + ciudades[i].habitantes$$

Ahora queda definido el wp de S:

$$wp(S, I) \cong 0 \leq i < |ciudades| \wedge_L \sum_{j=0}^{i-1} ciudades[j].habitantes = res$$

Ahora veo la implicacion del invariante y la guarda hacia wp(S,I):

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

$$0 \leq i < |ciudades| \wedge 0 \leq i \leq |ciudades| \wedge_L \sum_{j=0}^{i-1} ciudades[j].habitantes = res \longrightarrow (sigueabajo)$$

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

Se cancelan los terminos y queda:

$$0 \leq i \leq |ciudades| \longrightarrow True$$

$$True$$

Tercer paso Ahora probamos que vale la siguiente implicación: $I \wedge \neg B \longrightarrow Q_c$

$$0 \leq i \leq |ciudades| \wedge_L \sum_{j=0}^{i-1} ciudades[j].habitantes = res \wedge i \geq |ciudades| (sigueabajo)$$

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

Se cancelan las sumatorias y la igualdad

$$0 \leq i \leq |ciudades| \longrightarrow True$$

$True$

Ahora por teorema de terminación ahora debemos demostrar que la ejecución del ciclo siempre termina, nuestra función variante:

$$F_v = |\mathbf{ciudades}| - i - 1$$

- $\{I \wedge B \wedge F_v = v_0\} S \{F_v < v_0\}$
- $I \wedge F_v \leq 0 \longrightarrow \neg B$

Primer paso Probamos la primera implicación:

$$\{I \wedge B \wedge F_v = v_0\} S \{F_v < v_0\}$$

$$\{0 \leq i \leq |\mathbf{ciudades}| \wedge_L \sum_{j=0}^{i-1} \mathbf{ciudades}[j].habitantes = res \wedge 0 \leq i \leq |\mathbf{ciudades}| \wedge F_v = |\mathbf{ciudades}| - i - 1\} \longrightarrow (\text{sigueabajo})$$

$$wp(S, F_v < v_0) =$$

$$wp(S1, wp(S2, F_v < v_0)) =$$

$$wp(S1, wp(i := i + 1, F_v < v_0)) =$$

$$wp(S1, True \wedge F_v = |\mathbf{ciudades}| - i - 2 < v_0) =$$

$$wp(res := res + \mathbf{ciudades}[i].habitantes, True \wedge F_v = |\mathbf{ciudades}| - i - 2 < v_0) =$$

$$True \wedge F_v = |\mathbf{ciudades}| - i - 1 < v_0 =$$

$$F_v = |\mathbf{ciudades}| - i - 1 < v_0$$

Finalmente la implicación nos queda de la forma

$$\{0 \leq i \leq |\mathbf{ciudades}| \wedge_L \sum_{j=0}^{i-1} \mathbf{ciudades}[j].habitantes = res \wedge 0 \leq i \leq |\mathbf{ciudades}| \wedge F_v = |\mathbf{ciudades}| - i - 1\} \longrightarrow (\text{sigueabajo})$$

$$|\mathbf{ciudades}| - i - 1 < |\mathbf{ciudades}| - i$$

Se cancelan los terminos y queda

$$\{0 \leq i \leq |\mathbf{ciudades}| \wedge_L \sum_{j=0}^{i-1} \mathbf{ciudades}[j].habitantes = res \wedge_L F_v = |\mathbf{ciudades}| - i - 1\} \longrightarrow True$$

$True$

Segundo paso Probamos la segunda implicación:

$$I \wedge F_v \leq 0 \longrightarrow \neg B$$

$$0 \leq i \leq |\mathbf{ciudades}| \wedge_L res = \sum_{j=0}^{i-1} \mathbf{ciudades}[j].habitantes \wedge_L |\mathbf{ciudades}| - i - 1 \leq 0 \longrightarrow i \geq |\mathbf{ciudades}|$$

$$0 \leq i \leq |\mathbf{ciudades}| \wedge |\mathbf{ciudades}| \leq i + 1 \longrightarrow i \geq |\mathbf{ciudades}|$$

$$i = |\mathbf{ciudades}| \longrightarrow i \geq |\mathbf{ciudades}|$$

$True$

2.2. Ejercicio 2

Teniendo en cuenta la correctitud del programa demostrada en el 2.1, sabemos que el valor devuelto por el programa, siempre que se cumpla la precondition, tendrá la pinta de:

$$\text{res} = \text{sumatoria}$$

En el procedimiento, se especifica que el parámetro de entrada es de tipo IN, por lo tanto en la precondition y la postcondición, el parámetro *ciudades* mantendrá las mismas características, entre ellas que sus elementos son todos mayores o iguales a 0. Por lo tanto, al tomar un elemento de *ciudades*, obligatoriamente será menor o igual a la sumatoria del total de elementos de *ciudades*, es decir:

$$ciudades[i].habitantes \leq \text{sumatoria} \text{ con algún } i \ 0 \leq i < |ciudades|$$

Ahora bien, según la postcondición sabemos que $\text{res} =$ la sumatoria de todos sus elementos, entre los cuales sabemos que al menos alguno de ellos es mayor a 50.000, por lo tanto:

$$50,000 < ciudades[k].habitantes \quad \text{con algún } k \quad 0 \leq k < |ciudades|$$

Podemos afirmar entonces que:

$$50,000 < ciudades[k].habitantes \leq \text{sumatoria} \quad \text{con algún } k \quad 0 \leq k < |ciudades|$$

Y por transitividad concluimos que:

$$50,000 < \text{sumatoria}$$