DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Especificación y wp

En búsqueda del camino

12 de octubre de 2024 — Algoritmos y estructuras de datos

Grupo datapath

Integrante	LU	Correo electrónico
Andreone, Joaquin	122/24	jandreone06@gmail.com
Comerci, Lucas	818/24	lukicomerci@gmail.com
Luis, Theo	130/23	theoluis44@gmail.com
Zea, Marcos	405/09	zea.marcos@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. Especificaciones

proc grandesCiudades (in ciudades : $seq\langle Ciudad\rangle$) : $seq\langle Ciudad\rangle$
requiere $\{sinRepetidos(ciudades) \land cantidadHabitantesValida(ciudades)\}$
asegura $\{|res| \leq |ciudades| \land sinRepetidos(res)\}$
asegura $\{(\forall i : \mathbb{Z})\ (0 \leq i < |ciudades| \land_L ciudades[i]_1 \geq 50000 \longrightarrow_L ciudades[i] \in res)\}$
asegura $\{(\forall i : \mathbb{Z})\ (0 \leq i < |res| \longrightarrow_L res[i] \in ciudades)\}$

proc sumaDeHabitantes (in menoresDeCiudades: $seq\langle Ciudad\rangle$, in mayoresDeCiudades: $seq\langle Ciudad\rangle$) : $seq\langle Ciudad\rangle$
requiere $\{sinRepetidos(menoresDeCiudades) \land sinRepetidos(mayoresDeCiudades) \land$
$mismasCiudades(menoresDeCiudades, mayoresDeCiudades) \land$
$cantidadHabitantesValida(menoresDeCiudades) \land cantidadHabitantesValida(mayoresDeCiudades)\}$
asegura $\{|res| = |menoresDeCiudades|\}$
asegura $\{(\forall i, j : \mathbb{Z})\ (0 \leq i, j < |menoresDeCiudades| \land_L$
$menoresDeCiudades[i]_0 = mayoresDeCiudades[j]_0 \longrightarrow_L (\exists k : \mathbb{Z})\ (0 \leq k < |res| \land$
$(res[k]_0 = menoresDeCiudades[k]_0 \land res[k]_1 = menoresDeCiudades[i]_1 + mayoresDeCiudades[j]_1)))\}$

proc hayCamino (in distancias : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$, in desde: $\mathbb{Z}$, in hasta: $\mathbb{Z}$) : Bool
requiere $\{0 \leq desde, hasta < |distancias| \land esSimetrica(distancias) \land tieneDiagonalNula(distancias)\}$
asegura $\{res = \text{true} \leftrightarrow (\exists camino : seq\langle \mathbb{Z}\rangle)\ (esCamino(distancias, desde, hasta, camino))\}$

proc cantidadCaminosNSaltos (inout conexion : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$, in n: $\mathbb{Z}$)
requiere $\{esMatrizDeConexion(conexion) \land n \geq 1 \land conexion = Conexion_0\}$
asegura $\{mismasDimensiones(conexion, Conexion_0)\}$
asegura $\{(\exists potenciasDeConexion : seq\langle seq\langle seq\langle \mathbb{Z}\rangle\rangle\rangle)\ (|potenciasDeConexion| = n \land$
$potenciasDeConexion[0] = Conexion_0 \land todasMismasDimensiones(potenciasDeConexion, Conexion_0) \land$
$sonTodasPotenciasDe(potenciasDeConexion, Conexion_0) \land conexion = potenciasDeConexion[n-1])\}$

proc caminoMinimo (in origen : $\mathbb{Z}$, in destino: $\mathbb{Z}$, in distancias: $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) : $seq\langle \mathbb{Z}\rangle$
requiere $\{0 \leq origen < |distancias| \land 0 \leq destino < |distancias| \land esSimetrica(distancias) \land$
$tieneDiagonalNula(distancias)\}$
asegura $\{\neg(\exists\ camino : seq\langle \mathbb{Z}\rangle)\ (esCamino(distancias, origen, destino, camino)) \longrightarrow res =<>\}$
asegura $\{(\exists\ camino : seq\langle \mathbb{Z}\rangle)\ (esCamino(distancias, origen, destino, camino) \land$
$(\forall\ otroCamino : seq\langle \mathbb{Z}\rangle)\ (esCamino(distancias, origen, destino, otroCamino) \longrightarrow$
$longitudDeCamino(camino, distancias) \leq longitudDeCamino(otroCamino, distancias))) \longrightarrow res = camino\}$

1.1. Predicados y funciones auxiliares globales

pred sinRepetidos (ciudades: $seq\langle Ciudad\rangle$) {
$(\forall i : \mathbb{Z})\ (0 \leq i < |ciudades| \longrightarrow_L (\forall j : \mathbb{Z})\ (0 \leq j < |ciudades| \land_L ciudades[i]_0 = ciudades[j]_0 \longrightarrow_L j = i))$
}
pred cantidadHabitantesValida (in ciudades: $seq\langle Ciudad\rangle$) {
$(\forall i : \mathbb{Z})\ (0 \leq i < |ciudades| \longrightarrow_L ciudades[i]_1 \geq 0)$
}
pred esSimetrica (matriz:$seq\langle seq\langle \mathbb{Z}\rangle\rangle$) {
$(\forall i : \mathbb{Z})\ (0 \leq i < |matriz| \longrightarrow_L |matriz[i]| = |matriz|) \land (\forall i, j : \mathbb{Z})\ (0 \leq i, j < |matriz| \land_L matriz[i][j] = matriz[j][i])$
}
pred tieneDiagonalNula (matriz: $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) {
$(\forall i, j : \mathbb{Z})\ (0 \leq i, j < |matriz| \land i = j \longrightarrow_L matriz[i][i] = 0)$
}
pred mismasCiudades (in ciudades: $seq\langle Ciudad\rangle$, in otrasCiudades: $seq\langle Ciudad\rangle$) {
$(\forall i : \mathbb{Z})\ (0 \leq i < |ciudades| \longrightarrow_L (\exists j : \mathbb{Z})\ (0 \leq j < |otrasCiudades| \land_L ciudades[i]_0 = otrasCiudades[j]_0))$
}
pred esCamino (distancias: $seq\langle seq\langle \mathbb{Z}\rangle\rangle$, desde: $\mathbb{Z}$, hasta: $\mathbb{Z}$, camino: $seq\langle \mathbb{Z}\rangle$) {
$camino \neq <> \land_L desde = camino[0] \land hasta = camino[|camino|-1] \land desde \neq hasta \land (\forall i : \mathbb{Z})\ (0 \leq i < |camino| \longrightarrow_L$
$0 \leq camino[i] < |distancias|) \land (\forall i : \mathbb{Z})\ (0 \leq i < |camino| - 1 \longrightarrow_L distancias[camino[i]][camino[i+1]] \neq 0)$
}
pred esMatrizDeConexion (matriz: $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) {
$esSimetrica(matriz) \land tieneDiagonalNula(matriz) \land (\forall i, j : \mathbb{Z})\ (0 \leq i, j < |matriz| \land i \neq j \longrightarrow_L$
$(matriz[i][j] = 1 \lor matriz[i][j] = 0))$
}
pred esProductoDe (matrizA: $seq\langle seq\langle \mathbb{Z}\rangle\rangle$, matrizB: $seq\langle seq\langle \mathbb{Z}\rangle\rangle$, matriz: $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) {
$mismasDimensiones(matrizA, matrizB) \land mismasDimensiones(matrizA, matriz) \land$
$(\forall i, j : \mathbb{Z})\ (0 \leq i, j < |matriz| \longrightarrow_L matriz[i][j] = \sum_{k=0}^{|A|-1} matrizA[i][k] \cdot matrizB[k][j])$

}
pred mismasDimensiones (matrizA: $seq\langle seq\langle \mathbb{Z}\rangle\rangle$, matrizB: $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) {
$|matrizA| = |matrizB| \land (\forall i : \mathbb{Z})\ (0 \leq i < |matrizA| \longrightarrow_L |matrizA[i]| = |matrizB[i]|)$
}
pred todasMismasDimensiones (matrices: $seq\langle seq\langle seq\langle \mathbb{Z}\rangle\rangle\rangle$, matriz: $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) {
$(\forall i : \mathbb{Z})\ (0 \leq i < |matrices| \longrightarrow_L mismasDimensiones(matrices[i], matriz))$
}
pred sonTodasPotenciasDe (matrices: $seq\langle seq\langle seq\langle \mathbb{Z}\rangle\rangle\rangle$, matriz: $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) {
$(\forall i : \mathbb{Z})\ (1 \leq i < |matrices| \longrightarrow esProductoDe(matriz, matrices[i-1], matrices[i]))$
}
aux longitudDeCamino (camino: $seq\langle \mathbb{Z}\rangle$, distancias: $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) : $\mathbb{Z}$ =
$\sum_{i=0}^{|camino|-2} distancias[camino[i]][camino[i+1]]$;

2. Demostraciones de correctitud

La implementación es correcta con respecto a la especificación

```
1 | S1: res := 0;
2 | S2: i := 0;
3 |     while (i < ciudades.length) do
4 | S3:     res := res + ciudades[i].habitantes
5 | S4:     i = i + 1
6 |     endwhile
```

Como se ve en la implementacion de la especificacion, el programa consta solo de un ciclo while. Se tienen que cumplir estas condiciones:

1. $P \Longrightarrow wp(S1; S2; P_c)$
2. $\{P_c\}$**while** B **do** S_c **endwhile**$\{Q_c\}$
3. $Q_c \Longrightarrow Q$

donde $S_c = S3; S4$. Para demostar el punto 2, debemos usar el "Teorema del Invariante" y el "Teorema de terminación de un ciclo". Ambos se cumplen bajo estas condiciones:

2.1 $P_c \Longrightarrow I$

2.2 $\{I \land B\}S_c\{I\}$

2.3 $I \land \neg B \Longrightarrow Q_c$

2.4 $\{I \land B \land v_0 = fv\}S_c\{fv < v_0\}$

2.5 $I \land fv \leq 0 \Longrightarrow \neg B$

Basado en la especificacion y la implementacion propongo:

- $P_c \equiv res = 0 \land i = 0$
- $B \equiv i < |ciudades|$
- $S_3 \equiv res := res + ciudades[i]_1$
- $S_4 \equiv i := i + 1$
- $Q_c \equiv res = \sum_{j=0}^{|ciudades|-1} ciudades[j]_1$
- $I \equiv 0 \leq i \leq |ciudades| \land res = \sum_{j=0}^{i-1} ciudades[j]_1$
- $fv = |ciudades| - i$

1. $P \Longrightarrow wp(S1; S2; P_c)$

Calculo la $wp(S1; S2; P_c)$:

$$\begin{aligned} wp(S1; S2; P_c) &\equiv wp(S1, wp(S2, P_c)) \equiv wp(S1, wp(i := 0, res = 0 \wedge i = 0)) \equiv wp(S2, res = 0) \\ &\equiv wp(res := 0, res = 0) \\ wp(S1; S2; P_c) &\equiv True \end{aligned}$$

Verifico que $P \Longrightarrow wp(S1; S2; P_c)$, o sea que $P \Longrightarrow True$. Pero esto último es trivial, pues es una tautología.

2.1. $P_c \Longrightarrow I$

Reemplazo:

$$res = 0 \wedge i = 0 \Longrightarrow 0 \leq i \leq |ciudades| \wedge res = \sum_{j=0}^{i-1} ciudades[j]_1$$

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

$$res = \sum_{j=0}^{i-1} ciudades[j]_1 \equiv 0 = \sum_{j=0}^{0-1} ciudades[j]_1 = 0 \equiv True$$

Por lo tanto, se cumple $P_c \Longrightarrow I$

2.2. $\{I \wedge B\} S_c \{I\}$

Para que la tripla de Hoare sea valida, la precondición debe implicar la "Weakest precondition"del código y la postcondición. Es decir:

$$\{I \wedge B\} S_c \{I\} \Longleftrightarrow ((I \wedge B) \Longrightarrow_L wp(S_c, I))$$

$$wp(S_c, I) \equiv wp(S_3, wp(S_4, I)) \equiv wp(S_3, wp(i := i + 1, 0 \leq i \leq |ciudades| \wedge res = \sum_{j=0}^{i-1} ciudades[j]_1))$$

$$\equiv wp(S_3, def(i+1) \wedge_L 0 \leq i + 1 \leq |ciudades| \wedge res = \sum_{j=0}^{i+1-1} ciudades[j]_1)$$

$$\equiv wp(res := res + ciudades[i]_1, 0 \leq i + 1 \leq |ciudades| \wedge res = \sum_{j=0}^{i} ciudades[j]_1)$$

$$\equiv def(res) \wedge_L def(ciudades[i]_1) \wedge 0 \leq i + 1 \leq |ciudades| \wedge res + ciudades[i]_1 = \sum_{j=0}^{i} ciudades[j]_1$$

$$\equiv 0 \leq i < |ciudades| \wedge_L 0 \leq i + 1 \leq |ciudades| \wedge res + ciudades[i]_1 = \sum_{j=0}^{i} ciudades[j]_1$$

Combino los intervalos de i y resto en ambas partes de la igualdad $ciudades[i]_1$ para transformar el rango de la sumatoria

$$\equiv 0 \leq i < |ciudades| \wedge_L res = \sum_{j=0}^{i-1} ciudades[j]_1$$

Como $I \wedge B \equiv wp(S_c, I)$, por lo tanto $(I \wedge B) \Longrightarrow wp(S_c, I)$ que era lo que necesitaba para probar que la tripla de Hoare sea válida.

2.3. $I \wedge \neg B \Longrightarrow Q_c$

Reemplazo:

$$I \wedge \neg B \equiv 0 \leq i \leq |ciudades| \wedge res = \sum_{j=0}^{i-1} ciudades[j]_1 \wedge i \geq |ciudades|$$

Si $i \geq |ciudades| \wedge i \leq |ciudades|$ entonces $i = |ciudades|$

$$i = |ciudades| \wedge res = \sum_{j=0}^{i-1} ciudades[j]_1 \Longrightarrow res = \sum_{j=0}^{|ciudades|-1} ciudades[j]_1 \equiv Q_c$$

2.4. $\{I \wedge B \wedge v_0 = fv\} S_c \{fv < v_0\}$

Para que la tripla de Hoare sea válida, la precondición debe implicar la "Weakest precondition"del programa y la post-condicion. Es decir:

$$\{I \wedge B \wedge v_0 = fv\} S_c \{fv < v_0\} \iff ((I \wedge B \wedge v_0 = fv) \implies wp(S_c, fv < v_0))$$

Calculo wp:

$$\begin{aligned} wp(S_c, fv < v_0) &\equiv wp(S_3, wp(S_4, fv < v_0)) \equiv wp(S_3, wp(i := i + 1, |ciudades| - i < v_0)) \\ &\equiv wp(res := res + ciudades[i]_1, def(i+1) \wedge_L |ciudades| - (i+1) < v_0) \\ &\equiv def(ciudades[i]) \wedge_L |ciudades| - i - 1 < v_0 \\ &\equiv 0 \leq i < |ciudades| \wedge_L |ciudades| - i - 1 < v_0 \end{aligned}$$

Y desarrollo la precondición:

$$I \wedge B \wedge v_0 = fv \equiv 0 \leq i \leq |ciudades| \wedge res = \sum_{j=0}^{i-1} ciudades[j]_1 \wedge i < |ciudades| \wedge |ciudades| - i = v_0$$

Ignoro lo que corresponde a res porque no lo necesito para la demostración, y combino los rangos de i:

$$0 \leq i < |ciudades| \wedge |ciudades| - i = v_0$$

Entonces:

$$\{I \wedge B \wedge v_0 = fv\} \implies wp(S_c, fv < v_0) \iff$$
$$0 \leq i < |ciudades| \wedge |ciudades| - i = v_0 \implies 0 \leq i < |ciudades| \wedge_L |ciudades| - i - 1 < v_0 \iff$$
$$|ciudades| - i - 1 < |ciudades| - i \iff -1 < 0 \equiv True$$

Por lo tanto, la tripla de Hoare es válida.

2.5. $I \wedge fv \leq 0 \implies \neg B$

Desarrollo:

$$I \wedge fv \leq 0 \equiv 0 \leq i \leq |ciudades| \wedge res = \sum_{j=0}^{i-1} ciudades[j]_1 \wedge |ciudades| - i \leq 0 \implies \neg B$$

Ignoro lo que corresponde a res porque no lo necesito para la demostración:

$$0 \leq i \leq |ciudades| \wedge |ciudades| - i \leq 0 \implies i \geq |ciudades| \equiv$$
$$0 \leq i \leq |ciudades| \wedge |ciudades| \leq i \implies i \geq |ciudades| \equiv$$
$$i = |ciudades| \implies i \geq |ciudades| \equiv True$$

3. $Q_c \implies Q$

$$Q_c \equiv res = \sum_{j=0}^{|ciudades|-1} ciudades[j]_1 \implies res = \sum_{i=0}^{|ciudades|-1} ciudades[i]_1 \equiv Q$$

Completamente trivial, se cumple.

El valor devuelto es mayor a 50.000

Como en mi especificación el valor de entrada $ciudades = seq\langle Ciudad\rangle$ es un in, esto me asegura que la lista $ciudades$ no se modifica durante el programa, con lo cual hace que P sea valido siempre.

Calculo res a partir de la expresión en Q. Tomo $ciudades = seq\langle Ciudad\rangle$ y como mi P es valido, se tiene que existe un j, donde $0 \leq j < |ciudades|$, tal que $ciudades[j]_1 > 50000$. Basta ver que

$$res = \sum_{i=0}^{|ciudades|-1} ciudades[i]_1 = ciudades[j]_1 + \sum_{i=0 \wedge i \neq j}^{|ciudades|-1} ciudades[i]_1 > 50000 + \sum_{i=0 \wedge i \neq j}^{|ciudades|-1} ciudades[i]_1 > 50000$$

$$res > 50000$$

Por lo tanto, el valor devuelto es mayor a 50000.