UBA – Facultad de Ciencias Exactas y Naturales – Departamento de Computación

Algoritmos y Estructura de Datos I

Primer cuatrimestre de 2018

Versión: 4 de junio de 2018

TPI - "Funciones sobre señales: Implementación"

BUGS arreglados en la versión:

- bug en auxiliar `distancia` del ejercicio distanciaAcordeón.
- bug en post-condición de `ctrlf`.
- bug en auxiliar: `mean`

Fecha de entrega: Lunes 11 de Junio (hasta las 17hs)

1. Ejercicios

1. Implementar las funciones definidas en el archivo *auxiliares.cpp* (*definidas en auxiliares.h*).
2. Escribir tests para la función distanciaAcordeón. Ver sección **Testing**.
3. Implementar las funciones especificadas en la sección **Especificación** en el archivo *solucion.cpp*. Utilizar los tests provistos por la cátedra para verificar sus soluciones (los cuales **No pueden modificar**).
4. Completar (agregando) los tests necesarios para cubrir todas las líneas de los archivos soluciones.cpp y auxiliares.cpp. Se recomienda utilizar **lcov** para dicha tarea.

2. Testing

Para el siguiente ejercicio se pide escribir tests que cubran distintas particiones del dominio para la siguiente especificación. Los test deben ser útiles para encontrar errores en la implementación del algoritmo que la catedra posee. Los tests deben ser correctos con respecto a la especificación y debe cubrir las particiones del dominio que hayan considerado.
Opcional: Implementar el algoritmo.

`proc distanciaAcordeon` (`in` s: *señal*, `in` q: *señal*, `out` distancia: $\mathbb{R}$, `out` asignaciones: $seq\langle \mathbb{Z} \times \mathbb{Z} \rangle$) {

`Pre` $\{|s| > 0 \land |q| > 0\}$

`Post` $\{sonAsignacionesValidas?(s, q, asignaciones) \land$
$distancia = distancia(s, q, asignaciones) \land$
$((\forall asig : seq\langle \mathbb{Z} \times \mathbb{Z} \rangle) sonAsignacionesValidas?(s, q, asig) \longrightarrow_L distancia(s, q, asig) \geq distancia)\}$

`pred sonAsignacionesValidas?` (s: *señal*, q: *señal*, asignaciones: $seq\langle \mathbb{Z} \times \mathbb{Z} \rangle$) {
$asignacionesEnRango(s, q, asignaciones) \land$
$todosLosPuntosEstanAsignados?(s, q, asignaciones) \land$
$\neg asignacionesCruzadas(asignaciones)\}$

`pred asignacionesEnRango` (s: *señal*, q: *señal*, asignaciones: $seq\langle \mathbb{Z} \times \mathbb{Z} \rangle$) {
$(\forall t : seq\langle \mathbb{Z} \times \mathbb{Z} \rangle) t \in asignaciones \longrightarrow_L 0 \leq t_0 < |s| \land 0 \leq t_1 < |q|\}$

`pred asignacionesCruzadas` (asignaciones: $seq\langle \mathbb{Z} \times \mathbb{Z} \rangle$) {
$(\forall a : \mathbb{Z} \times \mathbb{Z}) a \in asignaciones \longrightarrow_L$
$\neg(\exists a1 : \mathbb{Z} \times \mathbb{Z}) a1 \in asignaciones \land_L a1_0 > a_0 \land a1_1 < a_1\}$

`pred todosLosPuntosEstanAsignados?` (s: *señal*, q: *señal*, asignaciones: $seq\langle \mathbb{Z} \times \mathbb{Z} \rangle$) {
$((\forall i : seq\langle \mathbb{Z} \times \mathbb{Z} \rangle) 0 \leq i < |s| \longrightarrow_L (\exists a : seq\langle \mathbb{Z} \rangle) a \in asignaciones \land_L i = a_0) \land$
$(\forall i : seq\langle \mathbb{Z} \times \mathbb{Z} \rangle) 0 \leq i < |q| \longrightarrow_L (\exists a : seq\langle \mathbb{Z} \rangle) a \in asignaciones \land_L i = a_1))\}$

`fun distancia` (s: *señal*, q: *señal*, asignaciones: $seq\langle \mathbb{Z} \times \mathbb{Z} \rangle$) : $\mathbb{R}$ =
$\sqrt{\sum_{i=0}^{|asignaciones|-1} (s[asignaciones[i]_0] - q[asignaciones[i]_1])^2}$;

}

3. Especificación

Implementar algoritmos que cumplan las siguientes especificaciones

`proc valida` (in s: *señal*, out result: Bool) {

Pre $\{True\}$

Post $\{result = true \leftrightarrow valida(s)\}$

}

`proc máximo` (in s: *señal*, out max: $\mathbb{Z}$, out latencia: $\mathbb{Z}$) {

Pre $\{valida(s) \wedge_L existeUnicoMaximo(s)\}$

Post $\{enRango(latencia, s) \wedge_L (s[latencia] = max \wedge esMaximo(max, s)\}$

`pred esMaximo` (i: $\mathbb{Z}$, s: *señal*) $\{(\forall j : \mathbb{Z})\ (enRango(j, s) \wedge j \neq i) \longrightarrow_L s[j] < s[i]\}$

`pred existeUnicoMaximo` (s : *señal*) $\{(\exists i : \mathbb{Z} \wedge enRango(i, s) \wedge_L esMaximo(i, s))\}$

}

`proc media` (in s: *señal*, in epsilon: $\mathbb{R}$, out res: $\mathbb{R}$) {

Pre $\{valida(s)\}$

Post $\{mean(s) == res\}$

}

`proc ctrlf` (in s: *señal*, in e: $\mathbb{Z}$, out res: $seq\langle\mathbb{Z}\rangle$) {

Pre $\{valida(s)\}$

Post $\{(\forall i : \mathbb{Z})(0 \leq i < |s| \longrightarrow_L i \in res \iff s[i] = e) \wedge todosDistintos(res)\}$

}

`proc medianaEntera` (in s: *señal*, out res: $\mathbb{Z}$, out latencia : $\mathbb{Z}$) {

Pre $\{valida(s)\}$

Post {
$(\exists s' : \text{señal})\ esPermutacion(s, s') \wedge ordenada(s') \wedge enRango(latencia, s) \wedge_L$
$\wedge\ s[latencia] = s'[div(|s'|, 2) - paridad(s)]$
$\wedge\ s[latencia] = res \wedge ((\forall j : \mathbb{Z})0 \leq j < latencia \longrightarrow_L s[j] \neq s[latencia]))\}$

`fun paridad` (s : $seq\langle\mathbb{Z}\rangle$) : $\mathbb{Z}$ = (if $par(|s|)$ then 1 else 0 fi) ;

}

`proc histograma` (in s: *señal*, in bins: $\mathbb{Z}$, out cuentas: $seq\langle\mathbb{Z}\rangle$, out limites: $seq\langle\mathbb{R}\rangle$) {

Pre $\{bins > 0 \wedge alMenos2ValoresDistintos(s)\}$

Post $\{bins = |cuentas| \wedge |limites| = |cuentas| + 1 \wedge_L$
$((\forall i : \mathbb{Z})\quad enRango(i, cuentas) \longrightarrow_L$
$(cuentas[i] = cuentaBin(s, bin(bins, s, i))$
$\wedge\ limites[i] == bin(bins, s, i)_1 \wedge limites[i + 1] == bin(bins, s, i)_2)\}$

`fun bin` (bins: $\mathbb{Z}$, s: *señal*, i: $\mathbb{Z}$) : $\mathbb{Z} \times \mathbb{Z}$ = $(min(s) + (rango(s)/bins) * i), (min(s) + (rango(s)/bins) * (i + 1))$;

`fun rango` (s : *señal*) : $\mathbb{Z}$ = $max(s) - min(s)$;

`fun cuentaBin` (s : *señal*, bin : $\mathbb{Z}\times\mathbb{Z}$) : $\mathbb{Z}$ = $(\sum_{j=0}^{|s|-1}$ if $bin_1 \leq s[j] < bin_2$ then 1 else 0 fi)+if $bin_2 = max(s)$ then 1 else 0 fi ;

`pred alMenos2ValoresDistintos` (s : *señal*) $\{(\exists i : \mathbb{Z})(\exists j : \mathbb{Z})enRango(i, s) \wedge enRango(j, s) \wedge_L s[i] \neq s[j]\}$

}

`proc slidingWindows` (in s: *señal*, in longitudes: $seq\langle\mathbb{Z}\rangle$, out promedios:$seq\langle\mathbb{R}\rangle$, out ventanas: $seq\langle\mathbb{Z} \times \mathbb{Z}\rangle$) {

Pre $\{|s| > 0 \wedge todosDistintos(longitudes) \wedge longitudesPositivas(longitudes)\}$

Post $\{ventanasValidas(s, longitudes, ventanas) \wedge promediosValidos(s, longitudes, promedios, ventanas)\}$

pred `ventanasValidas` (s: *señal*, longitudes: $seq\langle\mathbb{Z}\rangle$, ventanas: $seq\langle\mathbb{Z}\times\mathbb{Z}\rangle$) {
$|ventanas| = (\sum_{i=0}^{|longitudes|-1} \lceil |s|/longitudes[i] \rceil \wedge (\forall t:\mathbb{Z})t \in longitudes \longrightarrow_L$
$(\exists vs: seq\langle\mathbb{Z}\times\mathbb{Z}\rangle)|vs| = \lceil |s|/t\rceil \wedge incluido(vs, ventanas) \wedge intervalosCorrectos(vs, t, s)\}$

pred `intervalosCorrectos` (vs : $seq\langle\mathbb{Z}\times\mathbb{Z}\rangle$, t : $\mathbb{Z}$, s : *señal*) {
$((\forall l:\mathbb{Z})0 \leq l < \lceil |s|/t\rceil \longrightarrow_L (\exists inter \in \mathbb{Z}\times\mathbb{Z})inter \in vs \wedge (inter_0 = l * t) \wedge (inter_1 = (l+1)*t-1))\}$

pred `promediosValidos` (s: *señal*, longitud: $\mathbb{Z}$, promedios: $seq\langle\mathbb{R}\rangle$, ventanas: $seq\langle\mathbb{Z}\times\mathbb{Z}\rangle$) {
$|ventanas| = |promedios| \wedge_L$
$(\forall i:\mathbb{Z})0 \leq i < |promedios| \longrightarrow_L promedio(s, ventanas[i]) = promedios[i])\}$

fun `promedio` (s : *señal*, ventana: $\mathbb{Z}\times\mathbb{Z}$) : $\mathbb{R} = \frac{(\sum_{j=ventana_0}^{ventana_1} \textsf{ if } enRango(i,s) \textsf{ then } s[i] \textsf{ else } s[|s|-1] \textsf{ fi})}{(ventana_1 - ventana_0)+1}$;

pred `longitudesPositivas` (s: $seq\langle\mathbb{Z}\rangle$) $\{(\forall x: seq\langle\mathbb{Z}\rangle)x \in s \longrightarrow_L x > 0\}$

pred `incluido` (xs : $seq\langle\mathbb{Z}\times\mathbb{Z}\rangle$, ys : $seq\langle\mathbb{Z}\times\mathbb{Z}\rangle$) $\{(\forall x:\mathbb{Z}\times\mathbb{Z})x \in xs \longrightarrow_L x \in ys\}$

}

proc `distanciaEuclideana` (in p: *señal*, in q: *señal*, out distancia: $\mathbb{R}$) {

Pre $\{valida(s) \wedge valida(q) \wedge |p| = |q|\}$

Post $\{distancia = distanciaEuclideana(p, s)\}$

fun `distanciaEuclideana` (in p: *señal*, in q: *señal*) : $\mathbb{R}$ =
$\sqrt{\sum_{i=0}^{|p|-1}(p[i]-q[i])^2}$;

}

proc `completar` (inout s: *señal*, in huecos: $seq\langle\mathbb{Z}\rangle$) {

Pre $\{s_0 = s \wedge huecosEnRango(huecos, s_0) \wedge_L hayCerosEnPosicionesDeHuecos(s_0, huecos)\}$

Post $\{(\exists noHuecos: seq\langle\mathbb{Z}\rangle)esSecuenciaDeValores(huecos, noHuecos) \wedge$
$losValoresEnHuecosCambian(s, huecos, noHuecos) \wedge losValoresEnNoHuecosNoCambian(s, s_0, noHuecos)\}$

pred `huecosEnRango` (huecos: $seq\langle\mathbb{Z}\rangle$, s: $seq\langle\mathbb{Z}\rangle$) $\{(\forall e:\mathbb{Z})e \in huecos \longrightarrow_L 0 \leq e < |s|\}$

pred `hayCerosEnPosicionesDeHuecos` (s: $seq\langle\mathbb{Z}\rangle$, huecos: $seq\langle\mathbb{Z}\rangle$) $\{(\forall e:\mathbb{Z})e \in huecos \longrightarrow_L s[e] = 0\}$

pred `losValoresEnHuecosCambian` (s: *señal*, huecos: $seq\langle\mathbb{Z}\rangle$, noHuecos: $seq\langle\mathbb{Z}\rangle$) {
$(\forall hueco:\mathbb{Z})hueco \in huecos \longrightarrow_L (|noHuecos| = 0 \wedge s[hueco] = 0) \vee$
$(|noHuecos| > 0 \wedge$
$(\neg hayValoresMasChicos(noHuecos, hueco) \wedge_L (\exists e:\mathbb{Z})esElMayorMasCercano(e, noHuecos, hueco) \wedge s[hueco] =$
$s[e]) \vee$
$(\neg hayValoresMasGrandes(noHuecos, hueco) \wedge_L (\exists e:\mathbb{Z})esElMenorMasCercano(e, noHuecos, hueco) \wedge s[hueco] =$
$s[e]) \vee$
$(hayValoresMasChicos(noHuecos, hueco) \wedge hayValoresMasGrandes(noHuecos, hueco) \wedge_L$
$(\exists e:\mathbb{Z})esElMayorMasCercano(e, noHuecos, hueco) \wedge (\exists e_1:\mathbb{Z})esElMenorMasCercano(e_1, noHuecos, hueco) \wedge$

$s[hueco] = round((e + e_1)/2)\}$

pred `hayValoresMasChicos` (s: $seq\langle\mathbb{Z}\rangle$, e: $\mathbb{Z}$) $\{(\exists e_1:\mathbb{Z})e_1 \in s \wedge e_1 < e\}$

pred `hayValoresMasGrandes` (s: $seq\langle\mathbb{Z}\rangle$, e: $\mathbb{Z}$) $\{(\exists e_1:\mathbb{Z})e_1 \in s \wedge e_1 > e\}$

pred `esElMenorMasCercano` (e: $\mathbb{Z}$, s: $seq\langle\mathbb{Z}\rangle$, e1: $\mathbb{Z}$) $\{(\forall e_2:\mathbb{Z})e_2 \in s \wedge e_2 \neq e \longrightarrow_L (e_2 < e \vee e_1 < e_2) \wedge e < e_1\}$

pred `esElMayorMasCercano` (e: $\mathbb{Z}$, s: $seq\langle\mathbb{Z}\rangle$, e1: $\mathbb{Z}$) $\{(\forall e_2:\mathbb{Z})e_2 \in s \wedge e_2 \neq e \longrightarrow_L (e_2 < e_1 \vee e < e_2) \wedge e_1 < e\}$

pred `esSecuenciaDeValores` (huecos: $seq\langle\mathbb{Z}\rangle$, s: $seq\langle\mathbb{Z}\rangle$, noHuecos: $seq\langle\mathbb{Z}\rangle$) $\{valoresEnRango(noHuecos, s) \wedge$
$|huecos| + |noHuecos| = |s| \wedge$
$(\forall elem:\mathbb{Z})elem \in huecos \longleftrightarrow elem \notin noHuecos\}$

pred `valoresEnRango` (indices: $seq\langle\mathbb{Z}\rangle$, s: $\mathbb{Z}$) $\{(\forall e:\mathbb{Z})e \in indices \longrightarrow_L 0 \leq e < |s|\}$

pred `losValoresEnNoHuecosNoCambian` (s: $seq\langle\mathbb{Z}\rangle$, s_0: $seq\langle\mathbb{Z}\rangle$, noHuecos: $seq\langle\mathbb{Z}\rangle$) $\{(\forall pos:\mathbb{Z})pos \in noHuecos \longrightarrow_L$
$s[pos] = s_0[pos]\}$

}

`proc sacarOutliers` (inout s: *señal*, out borrados: $seq\langle\mathbb{Z}\rangle$) {

`Pre` $\{s_0 = s\}$

`Post` $\{valoresEnRango(borrados, s) \land |s_0| = |s| \land_L$
$(\forall i : \mathbb{Z})0 \leq i < |s_0| \longrightarrow_L$
$(esOutlier(s_0[i], s_0) \land s[i] = 0 \land i \in borrados) \lor$
$(\neg esOutlier(s_0[i], s_0) \land s[i] = s_0[i] \land i \notin borrados)\}$

`pred esOutlier` (v : $\mathbb{Z}$, s: $seq\langle\mathbb{Z}\rangle$) $\{(\exists s' : seq\langle\mathbb{Z}\rangle) esPermutacion(s, s') \land ordenada(s') \land_L v > s'[\lfloor |s'| * 0.95 \rfloor]\}$

}

`pred valida` (s: *señal*) $\{|s| > 0 \land_L (max(s) \leq 2^{15} - 1 \land min(s) \geq -2^{15})\}$

`pred enRango` (i: $\mathbb{Z}$, s: *señal*) $\{0 \leq i < |s|\}$

`fun mean` (s: *señal*) : $\mathbb{R} = \sum_{i=0}^{|s|-1} s[i]/|s|$;

`pred par` (n : $\mathbb{Z}$) $\{n \mod 2 = 0\}$

`pred ordenada` (s : $seq\langle\mathbb{Z}\rangle$) $\{(\forall i, j : \mathbb{Z})0 \leq i < j < |s| \longrightarrow_L s[i] \leq s[j]\}$

`pred esPermutacion` (s : $seq\langle\mathbb{Z}\rangle$, s': $seq\langle\mathbb{Z}\rangle$) $\{|s| = |s'| \land_L (\forall i, j : \mathbb{Z}) enRango(i, s) \land enRango(j, s) \longrightarrow_L$
$cantidadDeApariciones(s[i], s) = cantidadDeApariciones(s[i], s')\}$

`fun cantidadDeApariciones` (e : $\mathbb{Z}$, s : $seq\langle\mathbb{Z}\rangle$) : $\mathbb{Z} = \sum_{i=0}^{|s|-1}$ if $s[i] == e$ then 1 else 0 fi ;

`pred todosDistintos` (s: $seq\langle\mathbb{Z}\rangle$) $\{(\forall i : seq\langle\mathbb{Z}\rangle)(\forall j : seq\langle\mathbb{Z}\rangle)0 \leq j < |s| \land 0 \leq i < |s| \land i \neq j \longrightarrow_L s[i] \neq s[j]\}$

Términos y condiciones

El trabajo práctico se realiza de manera grupal con grupos de exactamente 2 personas. Para aprobar el trabajo se necesita:

- Que todos los ejercicios estén resueltos.
- Que las soluciones sean correctas.
- Que todos los tests provistos por la cátedra funcionen.
- Que las soluciones sean prolijas: evitar repetir implementaciones innecesariamente y usar adecuadamente funciones auxiliares.

Pautas de Entrega

Se debe enviar un e-mail conteniendo informe a la dirección algo1-tt-doc@dc.uba.ar. Dicho mail debe cumplir con el siguiente formato:

- El título debe ser `[ALGO1;TPI]` seguido inmediatamente del nombre del grupo.
- En el cuerpo del email deberán indicar: Nombre, apellido, libreta universitaria de cada integrante.
- Debe enviarse el archivo comprimido (.zip) que contenga los archivos `soluciones.cpp` y `auxiliares.cpp` completos. Además, enviar la carpeta de tests.

Importante: se admitirá un único envío, sin excepción alguna. Por favor planifiquen el trabajo para llegar a tiempo con la entrega.

Fecha de entrega: Lunes 11 de Junio (hasta las 17hs)