Este final de Algoritmos y Estructuras de Datos I fue tomado el 09-12-21.
Lo resolví para practicar para la siguiente fecha.
Saludos, Fede Arienti (fa.arienti@gmail.com)

---

1. Demostrar que el siguiente ciclo es correcto usando directamente el Axioma 5 (es decir, sin usar el teorema de corrección de ciclos).

$$\{x \ > \ 0\}$$
$$\text{while } (x > 0) \ \{ \ x := x - 2; \ \}$$
$$\{x = 0 \ \vee \ x = -1\}$$

Sean:

- P = $\{x \ > \ 0\}$
- Q = $\{x = 0 \ \vee \ x = -1\}$
- B = $x > 0$
- S = $\{ \ x := x - 2; \}$

Por el axioma 5, sabemos que el ciclo es correcto respecto a su especificación si:

$$P \implies wp(S, \ Q) = (\exists \ i : \mathbb{Z})(i \geq 0 \ \wedge_L \ H_i(Q)) = \bigvee_{i=0}^{\infty} H_i(Q)$$

donde $H_k(Q)$ es el conjunto de estados a partir de los cuales el ciclo termina en exactamente k iteraciones. Se define recursivamente $\forall k \geq 0$ por:

- $H_0(Q) = def(B) \ \wedge_L \ \neg B \ \wedge \ Q$
- $H_{k+1}(Q) = def(B) \ \wedge_L \ B \ \wedge \ wp(S, \ H_k(Q))$

Podemos notar que:

$$\begin{aligned} H_0(Q) &\equiv def(B) \ \wedge_L \ \neg B \ \wedge \ Q \\ &\equiv x \leq 0 \ \wedge \ (x = 0 \ \vee \ x = -1) \\ &\equiv x = 0 \ \vee \ x = -1 \end{aligned}$$

y calcular:

$$\begin{aligned} H_1(Q) &\equiv def(B) \ \wedge_L \ B \ \wedge \ wp(x := x - 2, \ H_0(Q)) \\ &\equiv x > 0 \ \wedge \ (x - 2 = 0 \ \vee \ x - 2 = -1) \\ &\equiv x = 2 \ \vee \ x = 1 \end{aligned}$$

$$\begin{aligned} H_2(Q) &\equiv def(B) \ \wedge_L \ B \ \wedge \ wp(x := x - 2, \ H_1(Q)) \\ &\equiv x > 0 \ \wedge \ (x - 2 = 2 \ \vee \ x - 2 = 1) \\ &\equiv x = 4 \ \vee \ x = 3 \end{aligned}$$

$$\begin{aligned} H_3(Q) &\equiv def(B) \ \wedge_L \ B \ \wedge \ wp(x := x - 2, \ H_2(Q)) \\ &\equiv x > 0 \ \wedge \ (x - 2 = 4 \ \vee \ x - 2 = 3) \\ &\equiv x = 6 \ \vee \ x = 5 \end{aligned}$$

en general, parece cumplirse:

$$H_k(Q) \equiv x = 2k \ \vee \ x = 2k - 1$$

Lo podemos demostrar por inducción.

Sea el caso base : $H_0(Q) \equiv x = 0 \ \vee \ x = -1$.

Quiero probar que $\forall k \geq 0$:

$$H_k(Q) \equiv x = 2k \ \vee \ x = 2k - 1 \implies H_{k+1}(Q) \equiv x = 2k + 2 \ \vee \ x = 2k + 1$$

Esto es cierto, ya que:

$$\begin{aligned} H_{k+1}(Q) &\equiv def(B) \ \wedge_L \ B \ \wedge \ wp(x := x - 2, \ H_k(Q)) \\ &_{ax,1, \ h.i.} \equiv x > 0 \ \wedge \ x - 2 = 2k \ \vee \ x - 2 = 2k - 1 \\ &\equiv x > 0 \ \wedge \ x = 2k + 2 \ \vee \ x = 2k + 1 \\ &_{k \geq 0} \equiv x = 2k + 2 \ \vee \ x = 2k + 1 \end{aligned}$$

Dada esta observación, podemos asegurar que:

$$\begin{aligned} wp(S, \ Q) &\equiv \bigvee_{i=0}^{\infty} H_i(Q) \\ &\equiv (\forall k : \mathbb{Z})(0 \leq k \longrightarrow_L x = 2k \ \vee \ x = 2k - 1) \\ &\equiv x \geq -1 \end{aligned}$$

y entonces, el programa es correcto respecto a su especificación, ya que:

$$\begin{aligned} P &\implies x \geq -1 \\ &\iff x > 0 \implies x \geq -1 \\ &\iff true \end{aligned}$$

---

2. Consideremos el siguiente problema, que surge en el contexto de algunos algoritmos de ordenamiento de secuencias. Recibimos como entrada una secuencia s y un entero i tal que $0 \leq$ i $<$ s, y suponemos que el número S[i] aparece solamente una vez en la secuencia. Llamamos x al valor s[i] en la secuencia recibida como parametro. El problema consiste en permutar los elementos de s de modo tal que:

   a) en primer lugar aparezcan los elementos menores que x.

   b) a continuación esté ubicado el valor x.

   c) y finalmente aparezcan los elementos mayores que x.

   Por ejemplo, si s = $<$ 3, 5, 7, 4, 2, 8 $>$ y además i = 3 (notar que x = s[i] = 4 aparece solamente una vez en s), entonces un resultado posible podria ser modificar la secuencia s para que sea s = $<$ 3, 2, 4, 5, 8, 7 $>$. No se solicita que los elementos mantengan su orden relativo, solamente que cumplan las condiciones a), b) y c) expresadas más arriba. Por ejemplo, otro resultado posible puede ser s = $<$ 2, 3, 4, 7, 8, 5 $>$.

   a) Dar una especificación para este problema.

b) Dar un algoritmo para este problema que solamente modifique la secuencia intercambiando elementos de la misma y que tenga complejidad computacional lineal.

c) Demostrar que este algoritmo es correcto para la especificación propuesta usando las técnicas formales vistas en la materia.

d) ¿Por qué el algoritmo propuesto tiene complejidad computacional lineal?

e) Escriba el control flow graph del algoritmo propuesto.

---

a) `proc permutarPorPivote` (inout s: $seq\langle\mathbb{Z}\rangle$, in i : $\mathbb{Z}$) {

`Pre` $\{s = s_0 \wedge |s| > 0 \wedge 0 \leq i < |s| \wedge_L \#apariciones(s,\ s[i]) = 1\}$

`Post` $\{mismos(s, s_0) \wedge_L$
$(\exists k : \mathbb{Z})(0 \leq k < |s| \wedge s[k] = s_0[i] \wedge_L ($
$(\forall n : \mathbb{Z})(0 \leq n < k \longrightarrow_L s[n] < s[k])\ \wedge$
$(\forall m : \mathbb{Z})(k < m < |s| \longrightarrow_L s[m] > s[k])$
$))\}$

}

`pred mismos` (s1 : $seq\langle\mathbb{Z}\rangle$, s2 : $seq\langle\mathbb{Z}\rangle$) {
$|s1| = |s2| \wedge_L (\forall i : \mathbb{Z})(0 \leq i < |s1| \longrightarrow_L \#apariciones(s1,\ s1[i]) = \#apariciones(s2,\ s1[i]))$
}

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

b) un algoritmo posible para resolver el problema es:

$\{Pre\}$
$swap(s,\ i,\ 0);$
$k := 0;$
$j := 1;$
$while\ (j < |s|)\ do$
$if\ (s[j] < s[k])\ then$
$swap(s,\ k+1,\ j);$
$swap(s,\ k,\ k+1);$
$k := k + 1;$
$j := j + 1;$
$else$
$j := j + 1;$
fi
od
$\{Post\}$

donde $swap(s : seq\langle\mathbb{Z}\rangle,\ i,\ j : \mathbb{Z})$ es un reemplazo sintáctico para:

$\{P_{swap} : 0 \leq i,\ j < |s| \wedge s = s_0\}$
$x := s[i]$
$s[i] := s[j]$
$s[j] := x$
$\{Q_{swap} : s[i] = s_0[j] \wedge s[j] = s_0[i]\}$

c) este algoritmo es correcto respecto a la especificación si:

$$Pre \implies wp(Bl;\ C,\ Post)^*$$

*donde Bl corresponde al bloque de las tres primeras líneas del programa, y C al ciclo.

Dado que aparece en varias ocaciones, encontremos primero $wp(swap(s,\ i,\ j),\ Q_{swap})$:

$$\begin{aligned} wp(swap(s,\ i,\ j),\ Q_{swap}) &\equiv wp(x := s[i];\ s[i] := s[j];\ s[j] := x,\ Q_{swap}) \\ {}_{ax\ 3} &\equiv wp(x := s[i],\ wp(s[i] := s[j],\ wp(s[j] := x,\ Q_{swap}))) \end{aligned}$$

Para simplificar las cuentas (aunque no lo parezca), podemos observar que el bloque es equivalente, haciendo un cambio de notación, a:

$\{P_{swap} : 0 \leq i,\ j < |s| \land s = s_0\}$
$x := s[i];$
$s := (s;\ i : s[j]);$
$s := (s;\ j : x);$
$\{Q_{swap} : s = (s_0;\ i : s_0[j],\ j : s_0[i])\}$

tal que:

$$\begin{aligned} wp(s[j] := x,\ Q_{swap}) &\equiv wp(s := (s;\ j : x),\ Q_{swap})) \\ {}_{ax\ 1} &\equiv 0 \leq i,\ j < |s| \land_L s_1^* = (s_0;\ i : s_0[j],\ j : s_0[i]) \end{aligned} \tag{1}$$

*$s_1 = (s;\ j : x)$

$$\begin{aligned} wp(s[i] := s[j],\ (1.)) &\equiv wp(s := (s;\ i : s[j]), (4.)) \\ {}_{ax\ 1,\ simplif} &\equiv 0 \leq i,\ j < |s| \land_L s_2^* = (s_0;\ i : s_0[j],\ j : s_0[i]) \end{aligned} \tag{2}$$

*$s_2 = (s;\ j : x,\ i : s[j])$

$$\begin{aligned} wp(x := s[i],\ (2.)) & \\ {}_{ax\ 1,\ simplif} &\equiv 0 \leq i,\ j < |s| \land_L s_3^* = (s_0;\ i : s_0[j],\ j : s_0[i]) \\ &\iff 0 \leq i,\ j < |s| \land s = s_0 \end{aligned} \tag{3}$$

*$s_3 := (s;\ j : s[i],\ i : s[j])$

Entonces, por equivalencia, $P_{swap} \implies wp(swap(s,\ i,\ j),\ Q_{swap})$ y swap es correcto respecto a su especificación.

Volviendo al ejercicio, podemos demostrar que el algorítmo propuesto es correcto respecto a la especificación si lo reducimos solo a la demostración del ciclo y aplicamos el teorema de la corrección de ciclo.

Los candidatos a utilizar son:

- $Q_C \equiv mismos(s, s_0) \land_L$
  $(\exists k : \mathbb{Z})(0 \leq k < |s| \land s[k] = s_0[i] \land_L ($

$$(\forall n : \mathbb{Z})(0 \leq n < k \longrightarrow_L s[n] < s[k]) \ \wedge$$
$$(\forall m : \mathbb{Z})(k < m < |s| \longrightarrow_L s[m] > s[k])$$
$$))$$

- $P_C \equiv k = 0 \ \wedge \ j = 1 \ \wedge \ |s_0| > 0 \ \wedge_L \ s = (s_0;\ i : s_0[0],\ 0 : s_0[i])^* \ \wedge \ \#apariciones(s_0,\ s_0[i]) = 1$
- $B \equiv j < |s|$
- $I \equiv mismos(s,\ s_0) \ \wedge \ 0 \leq k < j \leq |s| \ \wedge_L \ ($
  $s[k] = s_0[i] \ \wedge \ \#apariciones(s,\ s[k]) = 1 \ \wedge \ ($
  $(\forall n : \mathbb{Z})(0 \leq n < k \longrightarrow_L s[n] < s[k]) \ \wedge$
  $(\forall m : \mathbb{Z})(k < m < j \longrightarrow_L s[m] > s[k])$
  $))$
- $f_v = |s| - j$

*por post-condición de $swap(s,\ i,\ 0)$

Entonces:

1) $P_C \implies I$, dado que:
   - $s = (s_0;\ i : s_0[0],\ 0 : s_0[i]) \implies mismos(s,\ s_0)^*$

     *por permutacion
   - $k = 0 \ \wedge \ j = 1 \ \wedge \ |s_0| > 0 \implies 0 \leq k < j \leq |s|$
   - $k = 0 \ \wedge \ s = (s_0;\ i : s_0[0],\ 0 : s_0[i]) \implies s[k] = s_0[i]^*$

     *$s[k] = s[0] = s_0[i]$
   - $\#apariciones(s_0,\ s_0[i]) = 1 \ \wedge \ s = (s_0;\ i : s_0[0],\ 0 : s_0[i]) \ \wedge \ k = 0 \implies \#apariciones(s,\ s[k]) = 1$
   - $k = 0 \ \wedge \ j = 1 \ \wedge \ |s_0| > 0 \implies ($
     $(\forall n : \mathbb{Z})(0 \leq n < k \longrightarrow_L s[n] < s[k]) \ \wedge$
     $(\forall m : \mathbb{Z})(k < m < j \longrightarrow_L s[m] > s[k])$
     $)^*$

     *por vacuidad

2) $I \ \wedge \ \neg B \implies Q_C$, dado que:
   - $mismos(s,\ s_0) \implies mismos(s,\ s_0)$
   - $s[k] = s_0[i] \implies s[k] = s_0[i]^*$
   - $0 \leq k < j \leq |s| \ \wedge \ j \geq |s| \implies 0 \leq k < |s|^* \ \wedge \ j = |s|$
   - $j = |s| \implies ($
     $(\forall n : \mathbb{Z})(0 \leq n < k \longrightarrow_L s[n] < s[k]) \ \wedge$
     $(\forall m : \mathbb{Z})(k < m < |s| \longrightarrow_L s[m] > s[k])^*$
     $)$

   *es decir: el invariante implica que existe un k que satistaface lo pedido en cada caso.

3) $\{\ I \ \wedge \ B\ \}\ S_C^*\ \{\ I\ \}$ dado que $I \ \wedge \ B \implies wp(S_C,\ I)$

   *el cuerpo del ciclo

   Demostración:
   Sea

$$wp(S_C,\ I) \equiv \ wp(\text{if } B_{if} \text{ then } S1_C \text{ else } S2_C \text{ fi},\ I)$$
$$_{ax\ 4} \equiv def(B_{if}) \ \wedge_L \ ((B_{if} \ \wedge \ wp(S1_C,\ I)) \ \vee \ (\neg B_{if} \ \wedge \ wp(S2_C,\ I)))$$

para B_{if} la guarda del condicional, $S1_C$ el bloque *then* del algoritmo y $S2_C$ el bloque *else*.

Por un lado, tenemos:

$$\begin{aligned} wp(S1_C,\ I) &\equiv\ wp(swap(s,\ k+1,\ j);\ swap(s,\ k,\ k+1);\ k := k+1;\ j := j+1,\ I) \\ {}_{ax\ 3} &\equiv\ wp(swap(s,\ k+1,\ j),\ wp(swap(s,\ k,\ k+1),\ wp(k := k+1,\ wp(j := j+1,\ I)))) \end{aligned}$$

tal que, por el axioma 1 y asumiendo $\forall x : T$, def(x) es verdadero:

$$\begin{aligned} wp(j := j+1,\ I) \equiv mismos(s,\ s_0)\ \wedge\ 0 \leq k < j+1 \leq |s|\ \wedge_L\ (\\ s[k] = s_0[i]\ \wedge\ \#apariciones(s,\ s[k]) = 1\ \wedge\ (\\ (\forall n : \mathbb{Z})(0 \leq n < k \longrightarrow_L s[n] < s[k]) \wedge \\ (\forall m : \mathbb{Z})(k < m < j+1 \longrightarrow_L s[m] > s[k]) \\ ) \end{aligned} \tag{4}$$

$$\begin{aligned} wp(k := k+1,\ (4.)) \equiv mismos(s,\ s_0)\ \wedge\ 0 \leq k+1 < j+1 \leq |s|\ \wedge_L (\\ s[k+1] = s_0[i]\ \wedge\ \#apariciones(s,\ s[k+1]) = 1\ \wedge\ (\\ (\forall n : \mathbb{Z})(0 \leq n < k+1 \longrightarrow_L s[n] < s[k+1]) \wedge \\ (\forall m : \mathbb{Z})(k+1 < m < j+1 \longrightarrow_L s[m] > s[k+1]) \\ ) \end{aligned} \tag{5}$$

$$\begin{aligned} wp(swap(s,\ k,\ k+1),\ (5.)) \equiv mismos(s_1^*,\ s_0)\ \wedge\ 0 \leq k+1 < j+1 \leq |s_1|\ \wedge_L (\\ s_1[k+1] = s_0[i]\ \wedge\ \#apariciones(s_1,\ s_1[k+1]) = 1\ \wedge\ (\\ (\forall n : \mathbb{Z})(0 \leq n < k+1 \longrightarrow_L s_1[n] < s_1[k+1]) \wedge \\ (\forall m : \mathbb{Z})(k+1 < m < j+1 \longrightarrow_L s_1[m] > s_1[k+1]) \\ ) \end{aligned} \tag{6}$$

$^*s_1 := (s;\ k : s[k+1],\ k+1 : s[k])$

$$\begin{aligned} wp(swap(s,\ j,\ k+1),\ (6.)) \equiv mismos(s_2^*,\ s_0)\ \wedge\ 0 \leq k+1 < j+1 \leq |s|\ \wedge_L (\\ s_2[k+1] = s_0[i]\ \wedge\ \#apariciones(s_2,\ s_2[k+1]) = 1\ \wedge\ (\\ (\forall n : \mathbb{Z})(0 \leq n < k+1 \longrightarrow_L s_2[n] < s_2[k+1]) \wedge \\ (\forall m : \mathbb{Z})(k+1 < m < j+1 \longrightarrow_L s_2[m] > s_2[k+1]) \\ ) \end{aligned} \tag{7}$$

$^*s_2 := (s;\ k : s[j],\ k+1 : s[k],\ j : s[k+1])$

Entonces, si $B_{if} = s[j] < s[k]$ es verdadero, $I \wedge B \implies wp(S1_C,\ I)$. Ya que:

- $mismos(s,\ s_0) \implies mismos(s_2,\ s_0)^*\ \wedge\ |s_2| = |s|$

  $^*s2$ *es una permutacion de* s

- $0 \leq k < j \leq |s| \ \wedge \ j < |s| \implies 0 \leq k+1 < j+1 \leq |s| = -1 \leq k < j < |s|^*$

  **por subconjunto*

- $s[k] = s_0[i] \implies s_2[k+1]^* = s_0[i]$

  $^*s_2[k\ +\ 1] = s[k]$

- $\#apariciones(s,\ s[k]) = 1 \implies \#apariciones(s_2,\ s_2[k+1]) = 1^*$

  $^*s_2[k\ +\ 1] = s[k]$ *y* s_2 *es permutacion de* s

- $(\forall n : \mathbb{Z})(0 \leq n < k \longrightarrow_L s[n] < s[k]) \ \wedge \ s[j] < s[k]$
  $\implies (\forall n : \mathbb{Z})(0 \leq n < k+1 \ \longrightarrow_L \ s_2[n] < s_2[k+1])$
  $\equiv (\forall n : \mathbb{Z})(0 \leq n < k \ \longrightarrow_L \ s[n] < s[k]) \ \wedge \ s_2[k] < s_2[k+1]^*$

  $^*s_2[k] = s[j] \wedge s_2[k\ +\ 1] = s[k]$

- $(\forall m : \mathbb{Z})(k < m < j \longrightarrow_L s[m] > s[k]) \ \wedge \ 0 \leq k < j < |s|$
  $\implies (\forall m : \mathbb{Z})(k+1 < m < j+1 \ \longrightarrow_L \ s_2[m] > s_2[k+1])$
  $\equiv (\forall m : \mathbb{Z})(k+1 < m < j \ \longrightarrow_L \ s[m] > s[k])^* \ \wedge \ s_2[j] > s_2[k+1]$
  $\equiv (\forall m : \mathbb{Z})(k+1 < m < j \ \longrightarrow_L \ s[m] > s[k]) \ \wedge \ s[k+1] > s[k]^*$
  $\equiv (\forall m : \mathbb{Z})(k < m < j \ \longrightarrow_L \ s[m] > s[k])$

  **por subconjunto*

  **entran en el rango del* $\forall$ *del lado izquierdo*

Por el otro lado, tenemos:

$$
\begin{aligned}
wp(S2_C,\ I) \equiv \ & wp(j := j+1,\ I) \\
_{ax\ 1} \equiv \ & mismos(s,\ s_0) \ \wedge \ 0 \leq k < j+1 \leq |s| \ \wedge_L \ (\\
& s[k] = s_0[i] \ \wedge \ \#apariciones(s,\ s[k]) = 1 \ \wedge \ (\\
& (\forall n : \mathbb{Z})(0 \leq n < k \longrightarrow_L s[n] < s[k]) \ \wedge \\
& (\forall m : \mathbb{Z})(k < m < j+1 \longrightarrow_L s[m] > s[k]) \\
&))
\end{aligned}
$$

Entonces, si $B_{if} = s[j] < s[k]$ es false, $I \wedge B \implies wp(S2_C,\ I)$. Ya que:

- $0 < j \leq |s| \ \wedge \ j < |s| \implies 0 < j+1 \leq |s|$
- $(\forall m : \mathbb{Z})(k < m < j \longrightarrow_L s[m] > s[k]) \ \wedge \ s[j] \geq s[k] \ \wedge \ \#apariciones(s,\ s[k]) = 1$
  $\implies (\forall m : \mathbb{Z})(k < m < j+1 \longrightarrow_L s[m] > s[k])$
  $\equiv (\forall m : \mathbb{Z})(k < m < j \longrightarrow_L s[m] > s[k]) \wedge s[j] > s[k]$
- el resto de los predicados son verdaderos por igualdad.

Nos queda por demostrar que $I \ \wedge \ B \implies def(B_{if})$. Esto es inmediato, ya que:

$$0 < j \leq |s| \ \wedge \ j < |s| \implies def(B_{if}) = 0 \leq i,\ j < |s|$$

□

4) $\{I \ \wedge \ B \ \wedge \ f_v = f_0\}\ S_C\ \{f_v < f_0\}$, dado que:

- para cualquier valor de verdad de B_{if}, se ejecuta la sentencia '$j := j+1$;', tal que (por ax. 1):

  $0 \leq k < j \leq |s| \ \wedge \ j < |s| \ \wedge \ f_v = f_0 \implies |s| - (j+1) < f_0^*$

  $^*|s| \ -\ j \ -\ 1 < |s| \ -\ j$

5) $I \ \wedge \ f_v \leq 0 \implies \neg B$, dado que:

$$0 \leq k < j \leq |s| \ \wedge \ |s| - j \leq 0 \equiv |s| = j \implies j \geq |s|$$

Es decir, a partir de P_C, podemos afirmar que el ciclo termina en una cantidad finita de pasos y en un estado que satisface Q_C.

Como $Pre \implies wp(swap(s,\ i,\ 0);\ k := 0;\ j := 1,\ P_C)$ por equivalencia:

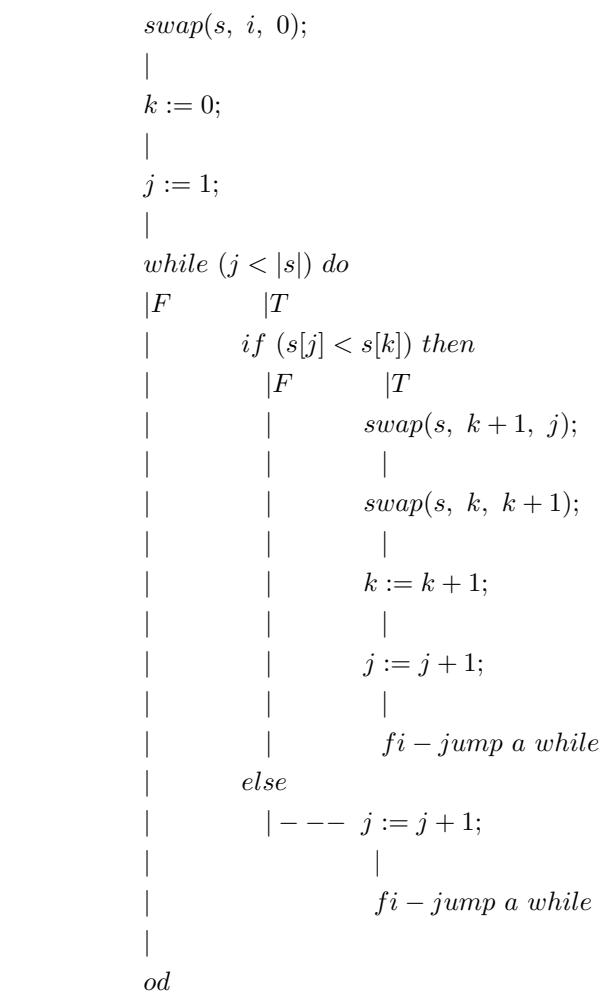
$$\begin{aligned} wp(swap(s,\ i,\ 0);\ k := 0;\ j := 1, P_C) &\equiv^{*} wp(swap(s,\ i,\ 0),\ wp(k := 0,\ wp(j := 1,\ P_C))) \\ &\equiv 0 = 0 \ \wedge \ 1 = 1 \ \wedge \ |s_0| > 0 \ \wedge \ 0 \leq i < |s| \ \wedge \\ &\qquad s = s_0 \ \wedge_L \ \#apariciones(s,\ s[i]) = 1 \\ &\equiv |s_0| > 0 \ \wedge \ 0 \leq i < |s| \ \wedge \\ &\qquad s = s_0 \ \wedge_L \ \#apariciones(s,\ s[i]) = 1 \end{aligned}$$

$^{*} ax\ 3$

Entonces, el algorítmo es correcto respecto a su especificación. □

d) El algorítmo propuesto tiene complejidad lineal porque itera $|s| - 1$ veces, como define la guarda y la inicialización de j. En este sentido, es lineal con respecto al tamaño de la lista.

e) cfg:



3. Enunciar la propiedad de monotonía de la precondición más débil. ¿Por qué es importante esta propiedad en el contexto de las demostraciones de corrección vistas en la materia?

Dado dos predicados A y B, y una sentencia S, la propiedad de monotonía establece que:

$$\{A \implies B\} \implies \{wp(S,\ A) \implies wp(S,\ B)\} \tag{8}$$

Esta propiedad es importante ya que a partir de la misma se puede derivar que:

Dados los predicados A, B y C, y las sentencias $S1$ y $S2$, si:

$$A \implies wp(S1,\ B)$$
$$B \implies wp(S2,\ C)$$

Entonces:

$$A \implies wp(S1;\ S2,\ C) \tag{9}$$

Demostración:

$$\begin{aligned} A &\implies wp(S1,\ B) \\ _{monotonia} &\implies wp(S1,\ wp(S2,\ C)) \\ _{ax3} &\implies wp(S1;\ S2,\ C) \end{aligned}$$

---

4. Una matriz cuadrada de enteros es triangular inferior si los elementos por encima de la diagonal son todos cero. Se dice triangular superior si los elementos por debajo de la diagonal son todos cero. Por ejemplo, la matriz de la izquierda es triangular inferior, y la matriz de la derecha es triangular superior. Una matriz se dice triangular si es triangular inferior o triangular superior.

1	0	0	0	0
3	1	0	0	0
2	0	4	0	0
1	1	4	1	0
5	6	7	1	1

1	6	6	6	0
0	2	6	0	5
0	0	3	0	4
0	0	0	5	2
0	0	0	0	2

a) Especificar el problema de determinar si una matriz de enteros es triangular.

b) Proponer un algoritmo para resolver el problema especificado en el punto anterior,

c) ¿Cuál es la complejidad computacional del algoritmo propuesto? (atención, esta pregunta tiene trampa!)

---

a) `proc esTriangular` (in m : $seq\langle seq\langle \mathbb{Z} \rangle\rangle$, out res : Bool) {

`Pre` $\{esCuadrada(m)\}$

Post $\{res = true \iff (esTriangularInf(m) \ \vee \ esTriangularSup(m))\}$

}

pred esCuadrada (m : $seq\langle seq\langle \mathbb{Z} \rangle\rangle$) {

$|m| > 1^* \wedge (\forall i : \mathbb{Z})(0 \leq i < |m| \longrightarrow_L |m[i]| = |m|))$

}

*se interpreta que una matriz cuadrada no puede ser menor a 2x2

pred esTriangularInf (m : $seq\langle seq\langle \mathbb{Z} \rangle\rangle$) {

$(\forall i : \mathbb{Z})(0 \leq i < |m| \longrightarrow_L$

$(\forall j : \mathbb{Z})(0 \leq j < i \longrightarrow_L \ m[i][j] = 0)$

)

}

pred esTriangularSup (m : $seq\langle seq\langle \mathbb{Z} \rangle\rangle$) {

$(\forall i : \mathbb{Z})(0 \leq i < |m| \longrightarrow_L$

$(\forall j : \mathbb{Z})(i < j < |m| \longrightarrow_L \ m[i][j] = 0)$

)

}

b) un algoritmo posible es:

```
bool esTriangular (vector<vector<int>> m) {
    /* pre : esCuadrada(m) */
    bool resInf = true;
    bool resSup = true;
    for(int i = 0; i < m.size(); ++i) {
        for (int j = 0; j < i && resInf; ++j) {
            resInf &= m[i][j] == 0;
        }
        for (int j = i + 1; j < m.size() && resSup; ++j) {
            resSup &= m[i][j] == 0;
        }
    }
    return resInf || resSup;
}
```

c) podemos notar que, considerando una matriz $N \times N$:

- el ciclo externo se repite N veces.
- los ciclos internos se repiten, en el peor caso, $i + N - (i + 1) = N - 1$ veces $(0 \leq i < N)$.

Entonces, en términos de N, la complejidad de peor caso es del orden de $O(N * (N - 1)) = O(N^2)$.