CECILIA AGUAYSOL

1

(a) $B \equiv d \,!= 0$ $\qquad S1 \equiv s[i] := s[i] / d$

$S2 \equiv skip$

$S \equiv$ if B then $S1$ else $S2$ endif

Para calcular la $wp(S, Q)$ vamos a asumir que todas las varial
están definidas.

Por el axioma 4 tenemos:

$wp(S,Q) \equiv def(B) \wedge_L ((B \wedge wp(S1, Q))$
$\vee (\neg B \wedge wp(S2, Q)))$

- $wp(S1, Q) \equiv wp(s[i] := s[i] / d, \text{todosPositivos}(s))$

$\equiv wp(setAt(s, i, s[i]/d), \text{todosPositivos}(s))$

$\equiv 0 \leq i < |s| \wedge_L \text{todosPositivos}(setAt(s, i, s[i]/d))$

def(s) ∧ def(i) ∧ def(s[i]/d) → AXIOMA 1

- $wp(S2, Q) \equiv wp(skip, \text{todosPositivos}(s)) \equiv \text{todosPositivos}(s$

AXIOMA 2

~~=> wp(S,Q) ≡~~

- $def(B) \equiv def(d \,!= 0) \equiv True$

$\Rightarrow wp(S,Q) \equiv ((d \,!= 0 \wedge 0 \leq i < |s| \wedge_L \text{todosPositivos}(setAt(s, i, s[i]/$
$\vee (d == 0 \wedge \text{todosPositivos}(s)))$

(b) Para ver que un programa es correcto respecto de la especifica
ción tenemos que demostrar $\{P\}\, S\, \{Q\}$

$\{P\}\, S\, \{Q\} \iff P \Rightarrow wp(S, Q)$

Supongamos que $d = 0$, por la $wp(S, Q)$ nos quedaría:

$P \Rightarrow \text{todosPrimos}(s)$

lo cual es cierto pues la Pre nos dice que $s = s_0$ entonces
$\text{todosPositivos}(s_0) = \text{todosPositivos}(s)$.

Por ende, en este caso, podemos afirmar que $P \Rightarrow wp(S, Q)$

Ahora debemos analizar el caso en el que $d \neq 0$:

Queremos ver que:

- $s = s_0 \wedge 0 \leq i < |s| \Rightarrow 0 \leq i < |s|$ ✓ Como $s = s_0$ podemos afirmar que es verdadero

- $s = s_0 \wedge todosPositivos(s_0) \Rightarrow todosPositivos(setAt(s, i, s[i]/d))$

Como setAt no modifica las longitudes de las secuencias, podemos usar que $|s| = |setAt(s, i, s[i]/d)|$. Entonces queremos ver que: ✓

$(\forall j : \mathbb{Z})(0 \leq j < |s| \rightarrow_L s[j] > 0) \Rightarrow (\forall j : \mathbb{Z})(0 \leq j < |s| \rightarrow_L setAt(s, i, s[i]/d)[j] > 0)$

Para analizar el setAt debemos ver el caso en el que $i = j$ $\quad i \neq j$

<u>CASO $i \neq j$</u>: $(\forall j : \mathbb{Z})(0 \leq j < |s| \rightarrow_L s[j] > 0)$

$\Rightarrow (\forall j : \mathbb{Z})(0 \leq j < |s| \rightarrow_L s[j] > 0)$ ✓ Como son iguales y $p \rightarrow p$, es correcto ✓

<u>CASO $i = j$</u>: $(\forall j : \mathbb{Z})(0 \leq j < |s| \rightarrow_L s[j] > 0)$ (a)

$\Rightarrow (\forall j : \mathbb{Z})(0 \leq j < |s| \rightarrow_L s[j]/d > 0)$ (b)

Como (a) forma parte de la Pre sabemos que todos los elementos de la secuencia son positivos. Entonces, para que (a) $\Rightarrow$ (b) ~~necesito que~~ necesariamente d debe ser positivo. Sin embargo $d \in \mathbb{R}$ así que habría casos en los que se cumple la Pre pero no la Post ✓

Por ejemplo: Sea $s = \langle 1, 2, 3 \rangle$ y $d = -2$ como $d \neq -2$, vamos a tener como resultado $s = \langle -1/2, -1, -3/2 \rangle$, lo cual no cumple la Post pues los elementos de s no son todos positivos.

En conclusión: el programa <u>no</u> es correcto respecto de la especificación. ✓

CECILIA AGUAYSOL

2

(a)

```
int res = 0;
int i = |S|;
while (i > 0){
    i--;
    res += S[i] * (i+1)
}
return res;
```

no es necesario

(b) $\{I \wedge B\}\ S\ \{I\}$ $\quad B \equiv i > 0$

$I \wedge B \equiv 0 < i \leq |S| \wedge res = \sum_{j=i}^{|S|-1} S[j] * (j+1)$

Sabiendo que $\{I \wedge B\} S \{I\} \iff \{I \wedge B\} \Rightarrow wp(S, I)$

$S1 \equiv i := i - 1$

$S2 \equiv res := res + S[i] * (i+1)$

$\Rightarrow wp(S, I) \equiv wp(S1, wp(S2, I))$

AXIOMA 3

- $wp(S2, I) \equiv wp(res := res + S[i] * (i+1),\ 0 \leq i \leq |S| \wedge res = \sum_{j=i}^{|S|-1} S[j] * (j+1))$

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

$\equiv 0 \leq i < |S| \wedge_L 0 \leq i \leq |S| \wedge res = \sum_{j=i+1}^{|S|-1} S[j] * (j+1)$

$\equiv 0 \leq i < |S| \wedge res = \sum_{j=i+1}^{|S|-1} S[j] * (j+1)$

- $wp(S1, wp(S2, I)) \equiv wp(i := i-1,\ 0 \leq i < |S| \wedge res = \sum_{j=i+1}^{|S|-1} S[j] * (j+1)$

$\equiv 0 \leq i - 1 < |S| \wedge res = \sum_{j=i}^{|S|-1} S[j] * (j+1)$

Qvq: • $0 \leq i \leq |S| \Rightarrow 0 \leq i-1 < |S|$

$\Rightarrow 1 \leq i \leq |S|$ ✓

• $\sum_{j=i}^{|S|-1} S[j] * (j+1) = res \Rightarrow res = \sum_{j=i}^{|S|-1} S[j] * (j+1)$ ✓

$\therefore \{I \wedge B\} \Rightarrow wp(S, I)$

(c) Para ver que el ciclo termina debemos probar que:

(1) $\{I \wedge B \quad v_0 = f_v\} \; S \; \{f_v < v_0\}$

(2) $I \wedge f_v \leq 0 \Rightarrow \neg B$

Propongo $f_v = i$ como función variante

2- ~~I~~ $0 \leq i \leq |S| \wedge res = \sum S[j] * (j+1) \wedge i \leq 0 \Rightarrow$

$i = 0 \Rightarrow i \geq 0$ ✓

1- $\{I \wedge B \wedge v_0 = i\} \Rightarrow wp(S, i < v_0)$

• $I \wedge B \wedge v_0 = i \equiv 0 < i \leq |S| \wedge res = \sum S[j] * (j+1) \wedge v_0 = i$

• $wp(S, i < v_0) \equiv wp(S1, wp(S2, i < v_0))$

$\equiv wp(S1, 0 \leq i < |S| \wedge_L i < v_0) \equiv wp(i := i-1, 0 \leq i < |S| \wedge_L i < v_0)$

$\equiv 0 \leq i-1 < |S| \wedge_L i-1 < v_0$

Qvq: • $0 < i \leq |S| \Rightarrow 1 \leq i \leq |S|$ ✓

• $v_0 = i \Rightarrow i - 1 < v_0$

$\Rightarrow i < v_0 + 1$

$\Rightarrow v_0 < v_0 + 1$ ✓

$\therefore$ El ciclo propuesto termina

CECILIA
AGUAYSOL

③ (a)

```
bool enRangoFila(int i, vector<vector<int>> a){
    int filas = a.size();
    bool res = false;
    if (i >= 0 && i < filas){
        res = true;
    }
    return res;
}
bool enRangoColumna(int i, vector<vector<int>> a){
    int columnas = a[0].size();
    bool res false;
    if (i >= 0 && i < columnas){
        res = true;
    }
    return res;
}
int multiplicarPosicion(vector<vector<int>> a, vector<vector(int>> b, int i,
                                                                  int j){
    int columnas = a[0].size() - 1;
    int k = 0, res = 0;
    while (k <= columnas){
        res += a[i][k] * b[k][j];
        k++;
    }
    return res;
}
```

```cpp
int maximoProducto (vector<vector<int>> a, vector<vector<int>> b){
    int res = 0;
    for (int i = 0; i < a.size(); i++) {
      for (int j = 0; j < b[0].size(); j++){
          if (multiplicarPosicion(a, b, i, j) >= res){
              res = multiplicarPosicion(a, b, i, j);
          }
      }
    }
    return res;
}
```

maximoProducto de complejidad $O(|a| * |b[0]| * |b[0]|)$

$\Rightarrow O(n^3)$ pues los tres ciclos dependen de tamaños de vectores

Falta b y c