

Ejercicio 1: Invariante de ciclo while (Insertion-Sort)

Salvador García González 119718

Proponemos la siguiente invariante para el ciclo while (lineas 5-8). Al iniciar la iteración i -ésima del ciclo while, el subarreglo $A[i, \dots, j-1]$ contiene elementos del arreglo original, tales que $A[i] \leq \dots \leq A[j-1]$

Inicialización

Al iniciar la primera ejecución del while, al haberse ejecutado las líneas 2-4, tenemos que $i=j-1=0$ y el subarreglo $A[i, \dots, j-1]$ consiste en un solo elemento $A[0]$, el cual está ordenado. Por lo tanto, el invariante se cumple.

Mantenimiento

Las siguientes iteraciones del while, los elementos $A[j-1], A[j-2], \dots$ se mueven una posición a la derecha siempre y cuando $A[i] > \text{llave}$. De esta forma, tenemos que $A[i] \leq A[i+1] \leq \dots \leq A[j-1]$ y el invariante es cierto.

Terminación

El while termina cuando $i < 0$ o $A[i] \leq \text{llave}$. Al finalizar el ciclo, por la condición de mantenimiento tenemos que $A[i, \dots, j-1]$ está ordenado, si i llega a -1 , los elementos del subarreglo ya estarán ordenados y si $A[i] \leq \text{llave}$, significa que ya no tenemos que mover el elemento y el subarreglo estará ordenado.

Ejercicio 2: Bubble Sort

1) Implementa el método swap con una complejidad de tiempo $\Theta(1)$

```
función swap(A[j], A[j-1])
```

```
    tmp = A[j-1]
```

C_1

$$T(n) = C_1 + C_2 + C_3 = C$$

```
    A[j-1] = A[j]
```

C_2

entonces el método tiene

```
    A[j] = tmp
```

C_3

complejidad $\Theta(1)$

```
end function
```

2) Prueba la correctez del programa Bubble Sort usando invariantes de ciclo

El algoritmo consta de dos ciclos for, por lo que se probará correctez para cada uno de ellos.

Invariante del for interno:

Al inicio de la i -ésima iteración, el valor $A[j]$ es el mínimo valor en el subarreglo $A[j, A.length]$

Inicialización

Al inicio de la primera iteración $j = A.length - 1$. El subarreglo $A[j, A.length - 1]$ contiene un único valor $A[j]$ y el valor de $A[j]$ es el mínimo en el subarreglo $A[j, A.length]$

Mantenimiento

El if statement de la linea 4 compara los elementos $A[j]$ y $A[j-1]$, intercambiando $A[j]$ y $A[j-1]$. Si $A[j]$ es menor que $A[j-1]$ y dejándolos en el mismo lugar en el caso contrario. Dada la condición inicial que el

valor en $A[j]$ es el valor mínimo en el subarreglo $A[j, A.length-1]$ esto significa que el valor en $A[j-1]$ ahora será el valor más pequeño en el subarreglo $A[j-1, A.length]$. Esto también implica que cada valor en el subarreglo $A[j, A.length-1]$ es mayor que el valor $A[j-1]$

Terminación

Este for termina cuando $j=i+1$ y dada la propiedad de mantenimiento el valor $A[i]$ que es $A[j-1]$ será el valor más pequeño en el subarreglo $A[i, A.length] = (A[j-1, A.length])$

Invariante For externo

Al inicio de la i -ésima iteración, el subarreglo $A[0, \dots, i-1]$ estará ordenado

Inicialización

Antes de la primera iteración, el arreglo $A[0, \dots, i-1]$ estará vacío. Por definición, el arreglo vacío está ordenado

Mantenimiento

Dadas las condiciones del for interno, al final de cada iteración, el valor $A[i]$ será el más pequeño en el subarreglo $A[i, A.length]$. Dado que los valores del subarreglo $A[i, j-1]$ fueron ordenados y fueron menores que el valor $A[i]$, los valores en el subarreglo $A[i, i]$ están ordenados.

Terminación

El for termina cuando i es igual a $A.length-2$. Dada la condición de mantenimiento, los valores en $A[i, A.length-2]$ están ordenados y son menores que el valor en $A[A.length-1]$. Entonces, los valores en $A[i, A.length-1]$ están ordenados

Ejercicio 3 : Evaluar un polinomio

$$p(x) = a_0 + a_1 x + \dots + a_{n-1} x^{n-1} + a_n x^n$$

Entrada: $n+1$ coeficientes $A = \langle a_0, \dots, a_n \rangle$ de polinomio $p(x)$ grado n y numero z

Salida: Resultado de evaluar $p(x)$ en el número z , $\sum_{i=0}^n a_i z^i$

d) Podemos reescribir la evaluación como: $A^T z$ con $z = [z^0, z^1, z^2, \dots, z^n]$

1 - Evaluar ingenuo Pseudocódigo:

```
1 function evaluar-ingenuo[A, z]
2     salida=0
3     for i=0 to A.length-1 do
4         expvalor = 1
5         for j=0 to i-1 do
6             expvalor = (expvalor)(z)
7         end for
8         salida=salida + A[i] (expvalor)
9     end for
10    return salida
11 end function
```

c_1 constante
 c_2 n
 c_3 constante
 c_4 $\sum_{i=0}^{i-1}$
 c_5 $\sum_{i=0}^n$
 c_6
 c_7 constante
 c_8 -
 c_9 -
 c_{10} -

2 - Evaluar ingenuo correctez:

Correctez:

- i) el algoritmo termina
- ii) es invariante de ciclo

i) Suponga un input finito: $A \cdot \text{length} < a \in \mathbb{N}$ y un escalar $z < b \in \mathbb{N}$

- Entonces el for de las líneas 3-9 se ejecuta desde $i=0$ a $i = A \cdot \text{length} - 1$ por lo que es finito y termina
- El segundo ciclo for de las líneas 5-7 se ejecuta desde $i=0$ a $i=j-1$, por lo tanto es finito y termina

ii) Invariante de ciclos : Se tiene que demostrar que es invariante de ciclo para el for de 3-9 y para el for 5-7

Ciclo 3-9 (for):

Tenemos $A^T z$ como la evaluación definida en a,
entonces el invariante de ciclo es:

Para la i -ésima iteración, se habrá evaluado hasta $A[0, \dots, i-1]^T z[0, \dots, i-1]$

Inicialización : Es verdadero antes de la primera iteración

Para inicializar, la variable salida se asigna un 0, por otra parte $i=0$. En la inicialización $\text{salida}=0$ ya que no se ha evaluado parte alguna del polinomio

Mantenimiento: Si es verdadero antes de la iteración de ciclo, entonces permanece verdadero al final de la iteración

Sup. $\text{salida} = A[0, \dots, i-1]^T z[0, \dots, i-1]$ antes de la iteración i , entonces queremos demostrar que con la iteración i salida sera-

$A[0, \dots, i]^T z[0, \dots, i]$.

Tenemos que $A[0, \dots, i]^T z[0, \dots, i] = \underline{A[0, \dots, i-1]^T z[0, \dots, i-1]} + \underline{A[i]^T z[i]}$
 por otra parte $A[i]^T z[i] = A[i] z^i$ por definición.

Por hipótesis, el valor de salida antes de la iteración es salida:

salida = $A[0, \dots, i-1]^T z[0, \dots, i-1]$ entonces basta por demostrar que durante la iteración i , se le añade el valor $A[i]^T z[i]$.

La invariantes de ciclo for de las líneas 5-7 nos garantizan que se evalúa $z[i] = z^i$, entonces por la linea 8, se le añade $A[i]^T z[i]$ al valor de salida.

Terminación:

El ciclo termina cuando $i = A.length = n$ entonces tenemos que salida = $A[0, \dots, n-1]^T z[0, \dots, n-1]$, lo que se quería demostrar al concluir el ciclo. La linea 10 da como return el valor de salida.

Tiempo de ejecución

$$T(n) = C_1 + nC_2 + C_3 + C_4 (\sum_{j=0}^{n-1} 1) + C_5 (\sum_{j=0}^{n-1} j) + C_7 n$$

↳ es desarrollar la suma de una serie que tiene un tiempo n^2

$$\Rightarrow T(n) = an^2 + bn + \text{cons.}$$

Por lo tanto podemos decir que tiene una cota asintótica $\Theta(n^2)$

Ejercicio 4: Regla de Horner

- 1) implementar evaluar-RH, el cual recibe A y z pero utilizando la regla de Horner.

$$p(x) = a_0 + a_1 x + \dots + a_{n-1} x^{n-1} + a_n x^n$$

Entrada: n+1 coeficientes A(a_0, \dots, a_n) de polinomio $p(x)$ grado n y numero z

Salida: Resultado de evaluar $p(x)$ en el numero z, $\sum_{i=0}^n a_i z^i$

```
1 function evaluar-ingenuo[A, z] -  
2     salida=0 C2 1  
3     for A.length-1 downto i=0 C3 n  
4         salida = salida*z + A[i] C4 n  
5     end for -  
6     return salida -  
7 end function -
```

2 - Evaluar ingenuo correctez:

Correctez:

- i) el algoritmo termina
- ii) es invariante de ciclo

i) Supongamos input finito: A.length < a a ∈ N y un escalar z < b b ∈ N

- Entonces el for de las líneas 3-5 se ejecuta desde i=0 a i=A.length-1 por lo que es finito y termina

Invariante de ciclo, ciclo for:

ii) Invariante de ciclo : Se tiene que demostrar que es invariante de ciclo para el for de 3-9 y para el for 5-7

Para la iésima iteración, se habrá evaluado parcialmente tal que siga algo por el estilo:

$$\begin{array}{l} \boxed{3x^3 + 2x^2 + x + 3} \\ | (3), (3z+2), (3z^2+2z+1) | \\ \hline \end{array} \quad \begin{array}{lll} i=3(n-1) & 3 & A_{n-1} x^0 \\ i=2(n-2) & 3x+2 & A_{n-1} x^1 + A_{n-2} \\ i=1(n-3) & 3x^2+2x+1 & A_{n-1} x^2 + A_{n-2} x + A_{n-3} \\ \\ i=0 & 3x^3+2x^2+x+3 & A_{n-1} x^3 + A_{n-2} x^2 + A_{n-3} x + A_{n-4} \end{array}$$

Inicialización

Al inicializar el ciclo se tiene $i = A.length - 1$ entonces $\sum_{j=n-1}^i A_j x^{n-j-1}$
por lo tanto cumple

Mantenimiento

Si el invariante cumple para el ciclo anterior, entonces en la siguiente iteración, todos los coeficientes en el ciclo anterior se les suma 1 al exponente (por el $n-j-1$). Además se añade el término actual

Terminación

Al terminar (cuando $i=0$) entonces $\sum_{j=n-1}^0 A_j x^{n-j-1}$ tendrá n elementos. cada uno con un exponente que va desde 0 hasta ($n-1$)

tiempo de ejecución:

$$T(n) = C_2(1) + C_3(n) + C_4(n) \Rightarrow T(n) = \alpha n + \text{cons}$$

$$\Rightarrow T(n) = \alpha n + \text{cons}$$

Por lo tanto podemos decir que tiene una cota asintótica $\Theta(n)$

El segundo (regla de Horner) es mejor ya que su cota asintótica es $O(n)$ en lugar de $O(n^2)$.