

Fundamentos de análisis y diseño de algoritmos

Computación iterativa

Algoritmo iterativo

Correctitud de un algoritmo iterativo

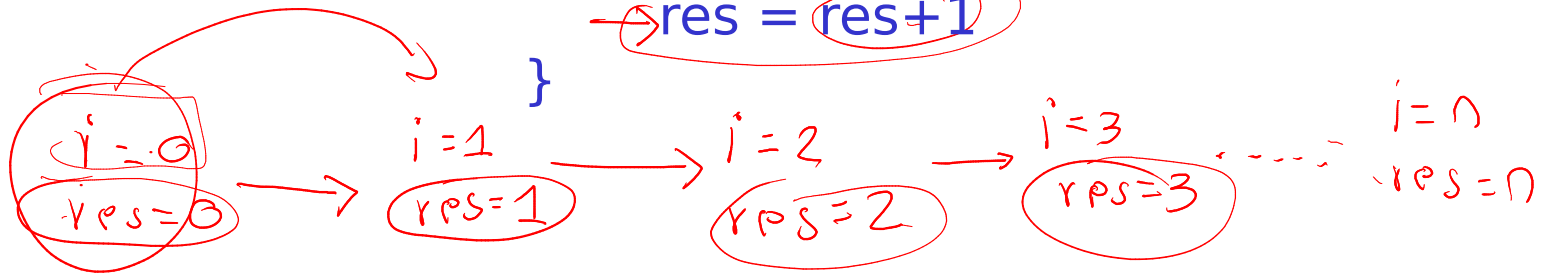
Invariantes de ciclo

res = 0

for i = 0, i <= n, i++){

res = res + 1

}



$res = i$

$(i, res) \rightarrow (i+1, res+1)$

Estado inicial $(0, 0)$
Estado final (n, n)

Computación iterativa

Una **computación iterativa** se caracteriza por comenzar en un estado inicial S_0 y transformar ese estado en un conjunto de estados intermedios hasta llegar a un estado final S_j

$$S_0 \rightarrow S_1 \rightarrow S_2 \rightarrow \dots \rightarrow S_j$$

Todo estado se debe caracterizar por cumplir una condición, llamada **invariante**.

Computación iterativa

¿Qué es una especificación?

Una **especificación** se define como la descripción de los siguientes parámetros:

Entrada: indica las precondiciones

Salida: indica las poscondiciones

Idea iterativa: muestra cómo deberían cambiar los estados, comenzando desde el inicial hasta llegar al final

Estados: especifica la forma de cada estado en forma de tupla, además, se muestra cuál es el invariante de estado

Estado inicial: muestra los valores que forman el estado inicial

Estado final: muestra los valores que forman el estado final

Transformación de estados: de manera formal especifica cómo se realizan, en términos generales, los cambios de un estado al siguiente

Computación iterativa

¿Qué es demostrar correctitud de un algoritmo?

Un algoritmo es correcto *con respecto a una especificación*

Será correcto si para cada entrada que cumple las precondiciones, el algoritmo termina cumpliendo la poscondición

Además, para el caso específico de algoritmos iterativos, se cuenta con un método formal de probar la correctitud

Computación iterativa

Especificación para el cálculo de factorial

Entrada: $N \geq 0$ $N \in \mathbb{Z}^+$ $0! = 1$

Salida: resultado = $N!$

Idea: Iteración (índice, resultado)

$(0, 1) \rightarrow (1, 1) \rightarrow (2, 2) \rightarrow (3, 6) \rightarrow \dots \rightarrow (N, N!)$

Estados: Tupla de la forma (índice, resultado) tal que resultado = índice! (Invariante)

Estado inicial: índice = 0, resultado = 1

Estado final: índice = N , resultado = $N!$

Transformación de estados:

$(\text{índice}, \text{resultado}) \rightarrow (\text{índice} + 1, \text{resultado} * (\text{índice} + 1))$

$n \in \mathbb{Z}^+$

$S!$
 \downarrow
 $\text{factorial}(n)$
 $n \in \{N \cup 0\}$

$(8, 8!)$

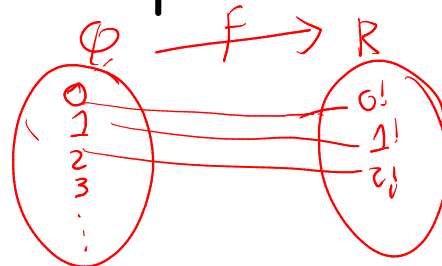
$(3, 6) \rightarrow (4, 24)$

Corrección

Un **especificación** es la definición de un problema en términos de su **precondición** Q y **poscondición** R

Un algoritmo A es **correcto con respecto a una especificación** si para cada conjunto de valores que cumplen Q , los valores de salida cumplen R

Se denota como $\{Q\} A \{R\}$. "A es correcto con respecto a la precondición Q y a la poscondición R "



Computación iterativa

Algoritmo para el cálculo de factorial

Factorial(int N){

int indice=0;

int resultado=1;

while !(indice==N){

indice=indice +1;

resultado= resultado * indice;

}

System.out.println(resultado);

}

Invariante

$$R = I!$$

Condición inicial

I R

$$(0, 1) \rightarrow (1, 1) \rightarrow (2, 2)$$

$$\rightarrow (3, 6) \rightarrow \dots \rightarrow (N, N!)$$

Transformación

$$(I, R) \rightarrow (I+1, R \times (I+1))$$

Estado Final

Computación iterativa

Algoritmo para el cálculo de factorial

```
Factorial(int N){  
    int indice=0;  
    int resultado=1;  
    while !(indice==N){  
        indice=indice +1;  
        resultado= resultado * indice;  
    }  
    System.out.println(resultado);  
}
```

*¿Es correcto el
algoritmo con respecto
a la especificación?*

$$a = \left(a + \frac{x}{a} \right) / 2$$

$$|a^2 - x| \leq \delta$$

$$x = 10$$

$$\delta = 0.01$$

$$a = 1$$

$$|a^2 - x| \leq 0.01$$

$$\delta \leq 0.01$$

$$a = \left(1 + \frac{10}{1} \right) / 2 = \frac{11}{2} = 5.5$$

$$\approx 2.8 \times 10^1 \quad |a^2 - x| \leq 0.01 \quad 1 \leq 0.01$$

$$a = \left(5.5 + \frac{10}{5.5} \right) / 2 = 3.659 \quad |3.659^2 - 10| \leq 0.01 \quad \times$$

$$a = \left(3.659 + \frac{10}{3.659} \right) / 2 = 3.196 \quad |3.196^2 - 10| \leq 0.01 \quad 0.21 \quad \times$$

$$a = \left(3.196 + \frac{10}{3.196} \right) / 2 = 3.162 \quad |3.162^2 - 10| \leq 0.01 \quad 0.002 \quad \checkmark$$

9.998

Computación iterativa

Especificación para el cálculo de raíz de X

Entrada: $X \geq 0 \wedge X \in \mathbb{R} \wedge \delta > 0$ $\wedge \delta \in \mathbb{R}$

Salida: a tal que $|a^2 - X| \leq \delta$

Idea: Dado X , inicie la aproximación de a con el valor 1.0 y mejorela utilizando el cambio de a por $(a + X/a)/2$

$(1, 1.0) \rightarrow (2, (1.0 + x/1.0)/2) \rightarrow \dots \rightarrow (N, a)$ $\approx \sqrt{x}$ $|a^2 - x| \leq \delta$
Iteración

Estados: Tupla de la forma (índice, aproximación) tal que $a > 0$
(Invariante)


Estados inicial: $a = 1.0$

Estado final: a tal que $|a^2 - X| \leq \delta$

Transformación de estados: $(\text{índice}, a) \rightarrow (\text{índice} + 1, (a + X/a)/2)$

Computación iterativa

Algoritmo para el cálculo de raíz de X

```
raizIterativa(double X, double delta){  
    double a=1.0;  
    while ( !(Math.abs(a*a-X)<=delta) ){  
        a = (a + X/a)/2.0;  
    }  
    System.out.println(a);  
}
```

Computación iterativa

Identifique en los algoritmos `Factorial(int N)` y `raizIterativa(double X, double delta)` los estados inicial y final, así como la transformación dada en la especificación

¿Cómo se manejan las condiciones de entrada en el algoritmo?

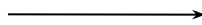
Computación iterativa

Algoritmo para el cálculo de factorial

```
Factorial(int N){
```

```
    int indice=0;
```

```
    int resultado=1;
```



Condiciones iniciales

```
    while !(indice==N){
```

```
        indice=indice +1;
```

```
        resultado= resultado * indice;
```

```
    }
```

```
    System.out.println(resultado);
```

```
}
```

Computación iterativa

Algoritmo para el cálculo de factorial

```
Factorial(int N){
```

```
    int indice=0;
```

```
    int resultado=1;
```

```
    while !(indice==N){
```

```
        indice=indice +1;
```

```
        resultado= resultado * indice;
```

```
    }
```

```
    System.out.println(resultado);
```

```
}
```

Transformación de estados:

$(\text{índice}, \text{resultado}) \rightarrow (\text{índice} + 1, \text{resultado} * (\text{índice} + 1))$



Computación iterativa

Algoritmo para el cálculo de raíz de X

```
raizIterativa(double X, double delta){
```

```
    double a=1.0;
```

—————→ *Condición inicial*

```
    while ( !(Math.abs(a*a-X)<=delta) ){
```

```
        a = (a + X/a)/2.0;
```

```
    }
```


```
    System.out.println(a);
```

```
}
```


Computación iterativa

Algoritmo para el cálculo de raíz de X

```
raizIterativa(double X, double delta){  
    double a=1.0;  
    while ( !(Math.abs(a*a-X)<=delta) ){  
        a = (a + X/a)/2.0;  
    }  
    System.out.println(a);  
}
```

Transformación de estados:
 $(\text{índice}, a) \rightarrow (\text{índice}+1, (a+X/2)/2)$

contador

Computación iterativa

El esquema de un algoritmo iterativo es el siguiente:

$S_0 = \text{Estado inicial}$

$S \leftarrow S_0$

$S_{i+1} \leftarrow S_i$

while ! isFinal(S) do

$S \leftarrow \text{Transform}(S)$

Computación iterativa

Cómo probar que un algoritmo iterativo A es correcto con respecto a una especificación (precondición Q , poscondición R)

1. **Inicialización:** Pruebe que el estado inicial S_0 cumple el invariante
2. **Invarianza:** Prueba que la transformación conserva el invariante
3. **Éxito:** Si S es un estado final \wedge se cumple el invariante $P \rightarrow R$
4. **Terminación:** A termina

Computación iterativa

Computa (int A, int B) { ^{Entradas}

int res=0, i=1;

while (i<=B) {

i=i+1; ^{B+1}

res=res + A;

System.out.println(res);

}

$S = (i, res)$

$S_0 = (1, 0)$ ←

$(i, res) \rightarrow (i+1, res+A)$

$(1, 0) \rightarrow (2, A) \rightarrow (3, 2A) \rightarrow (4, 3A) \rightarrow$
 _{S_0}

$\rightarrow (5, 4A) \rightarrow \dots$

$res = (i-1)A$

Invariante

Valor
Constante

$\rightarrow (B+1, BA)$

Qué calcula Computa(2,3)?

Computación iterativa

Probar correctitud

1. **Inicialización:** *Pruebe que el estado inicial S_0 cumple el invariante*

El estado inicial es $(1,0)$, Se verifica que se cumpla el invariante, se tiene que $i=1$ ✓

Computación iterativa

Probar correctitud

1. **Inicialización:** *Pruebe que el estado inicial S_0 cumple el invariante*

El estado inicial es $(1,0)$, Se verifica que se cumpla el invariante, se tiene que $i=1$.

$$\text{res} = (i-1) \cdot A$$
$$\text{res} = 0$$

$$(1, 0) \checkmark$$

Computación iterativa

2. **Invarianza:** Prueba que la transformación conserva el invariante

¿Puedo calcular un estado cualquiera con la invariante?
o ¿Lo puedo estimar?

Se considera que antes de entrar el ciclo, $i=k$ y se prueba.

Si $i=k$,

(k, res)

$$\text{res} = (k-1)A$$

$$\text{res} = (5-1)A$$

$$\text{res} = 4A$$

$(1, 0) \rightarrow (2, A) \rightarrow (3, 2A) \rightarrow (4, 3A) \rightarrow (5, 4A)$

$$\text{res} = (3-1)A$$

$$2A$$

Computación iterativa

Computa (int A, int B){

int res=0, i=1;

while (i<=B){

i=i+1;

res=res + A;

}

System.out.println(res);

}

$B \geq 0$ X

$A \in \mathbb{Z}$

Computación iterativa

2. Invarianza: Prueba que la transformación conserva el invariante

$$\begin{aligned} i &= i + 1 \\ res &= res + A \end{aligned}$$

$$(i, (i-1)A) \longrightarrow (i+1, \underbrace{(i-1)A + A}) = (i+1, iA)$$

$$iA - A + A = iA$$

$$res = (i-1)A$$

$$\begin{aligned} res &= ((i+1)-1)A \\ res &= iA \end{aligned}$$

Usar la transformación de estados y comprobar si se sigue cumpliendo la invariante de ciclo en el siguiente estado.

Computación iterativa

3. Éxito: *Invariante $P \wedge S$ es un estado final $\rightarrow R$*

$(B+1, AB) \checkmark$

Computación iterativa

3. Éxito: *Invariante $P \wedge S$ es un estado final $\rightarrow R$*

$$(B+1, AB) = Si$$

$$(i, (i-1)A)$$

$$(B+1, BA)$$

Computación iterativa

4. Terminación: A termina

En cada iteración i aumenta, por lo que en algún momento finito tendrá que alcanzar el valor de B y el algoritmo terminará

$B+1$

$i = 0, 1, 2, 3, 4, \dots, B, B+1$

Computación iterativa

¿Qué calcula `Computa3(4)`?

Expresa la forma de los estados

Muestre la idea iterativa que presenta el algoritmo

Pruebe la correctitud

Indique la precondition y poscondición

Debe calcular la invariante para el ciclo interno y el ciclo externo

```
Computa3 (int N){
```

```
int A, B, i, j;
```

```
  A=0;
```

```
  i=1;
```

```
  while (i<=N){
```

```
    B=1;
```

```
    j=1;
```

```
    while (j<=3){
```

```
      B=B*i;
```

```
      j++;
```

```
    A=A+B;
```

```
    i++;
```

```
  }
```

```
  System.out.println("Resultado=" + A);
```

```
}
```

↙ Estado = (^{contador} i, ^{resultado} A)

$$S_0 = (1, 0)$$

$$(1, 0) \rightarrow (2, 1) \rightarrow (3, 9) \rightarrow (4, 36)$$

$$\rightarrow (5, 100)$$

$$(i, A) \rightarrow (i+1, i^3 + A)$$

Diagram showing the transition from state (i, A) to $(i+1, i^3 + A)$. The variable i is incremented by 1, and the value i^3 is added to A . The increment of i is labeled with a '1' and the addition of i^3 to A is also labeled with a '1'.

$$B = i^3$$

$$(1, 0) \rightarrow (2, 0 + 1^3) \rightarrow (3, 0 + 1^3 + 2^3) \rightarrow (4, 0 + 1^3 + 2^3 + 3^3)$$

$$(i-1)^3 + A$$
$$= 3^3 + 0A?$$

$$\sim (1, 0) \rightarrow (2, 0^3 + 1^3) \rightarrow (3, 0 + 1^3 + 2^3) \rightarrow (4, 0 + 1^3 + 2^3 + 3^3)$$

$$0^n = 0 \quad n \in \mathbb{Z}^+$$

$$\sum_{k=1}^2 (k-1)^3 = 0^3 + 1^3$$

$$(6, 0^3 + 1^3 + 2^3 + 3^3 + 4^3 + \dots + 7^3)$$

$$\sum_{k=1}^4 (k-1)^3$$

mayor el índice

$$\left(F, \sum_{k=1}^F (k-1)^3 \right)$$

Invariante

$$S_0 = (1, 0)$$

$$\sum_{k=1}^1 (k-1)^3$$

$$j = N+1$$

$$(N+1, 0^3 + 1^3 + 2^3 + \dots + N^3)$$

$$A = \sum_{k=1}^{N+1} (k-1)^3$$

$$(k-1)^3 = k^3 - 3k^2 + 3k - 1$$


```
Algoritmo(int n){
```

```
    i = 0
```

```
    s = 3
```

```
    while(i<=n){
```

```
        j = 0
```

```
        p = 4
```

```
        while(j <= 2n){
```

```
            p += 2
```

```
            j+=1
```

```
        }
```

```
        s+=2p
```

```
        i+=2
```

```
    }
```

```
}
```

Computación iterativa

¿Qué calcula `Algoritmo(6)`?

Expresa la forma de los estados

Muestre la idea iterativa que presenta el algoritmo

Pruebe la correctitud

Indique la precondición y poscondición

```

BS (int A[], int N){
    int i, j, aux;
    for ( i=1; i < N ; i++)
        for ( j=N; j > i ; j--){
            if ( A[j] < A[i] ){
                aux=A[j];
                A[j]=A[i];
                A[j-1]=aux;
            }
        }
}

```

Invariante Externo

El subarreglo $A[i..n]$ tiene su menor elemento en la posición $A[i]$

Invariante Interno:

El subarreglo $A[i..j]$ cumple que $A[i] < A[j]$

A partir del procedimiento indicado a continuación, que tiene como entrada un arreglo A indexado de la forma $[1..n]$. Indicar:

1. Invariante de ciclo para el iterador interno (línea 3)
2. Invariante de ciclo para el iterador externo (línea 2)
3. ¿Qué calcula?

Referencias

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. Introduction to Algorithms, Third Edition (3rd ed.). The MIT Press. Pages 18-20

Gracias

Próximo tema:

Notación de crecimiento de funciones