Universidad de San Buenaventura

Facultad ingeniería de sistemas

UNIVERSIDAD DE
SAN BUENAVENTURA
CALI

Parcial corte 1

Análisis de algoritmos

Presenta:

Juan Felipe Hurtado Villani

Cristian Apraez

Samuel Martínez

Profesor

Carlos Andrés Delgado

1. ANALISIS DE ALGORITMOS:

Punto 1:
Algoritmo 1:

C#	ALGORITMO	OPERACIÓN
C0	def algoritmo1(n):	
C1	i=2*n+1	1
C2	res=0	1
C3	while i>0:	(2n+1)+1
C4	j=i*i	(2n+1)+1
C5	res+=j	(2n+1)+1
C6	i-=1	(2n+1)+1
C7	return res	1

Complejidad:
8n+11
Forma de estado:
(i, res)
Estado inicial:
(2n+1, 0)
Transformaciones:
n → 1
(3,0) → (2,9) → (1,13)
(i, res) → (i-1, res + i*i)
Estado final:
$(0,\sum_{i=1}^{2n+1} i^2)$
Invariante:
$res = \sum_{i=1}^{2n+1} i^2$

Algoritmo 2:

C#	ALGORITMO	OPERACIÓN
C0	def algortimo2(n):	
C1	i = 0	1
C2	j = 0	1
C3	res=0	1
C4	while i < 3*n:	3n+1
C5	j = 2*1	3n
C6	res -= j	3n
C7	1+ = 1	3n
C8	return res	1

Complejidad:
12n+5
Forma de estado:
(i, res)
Estado inicial:
(0, 0)
Transformaciones:
(i, res) → (i+1, res -2 *1)
Estado final:
i = 3n+1
Invariante:
res = $\sum_{i=1}^{3n+1} 2 * 1$

Algoritmo 3:

C#	ALGORITMO	OPERACIÓN
C0	def algoritmo3(n)	
C1	i = 4*n+2	1
C2	j = 0	1
C3	res = 0	1
C4	while i > 0:	4n+3→ 4n+1
C5	while j <= 3*n + 4:	3n+6
C6	res+=4	3n+5
C7	j+=1	3n+5
C8	i-=1	4n+2
C9	return res	1

Complejidad:
17n+25
17n+23
Forma de estado:
(i, res)
Estado inicial:
(4n+2, 0)
Transformaciones:
(i, res) → (i-1, res 4+1)
Estado final:
$(0, \sum_{i=1}^{4n+2} i + 1)$
Invariante:
res = $\sum_{i=1}^{4n+2} i + 1$
2. DISEÑO DE ALGORITMOS:
Sabemos que,
$(k + 1)^3 = k^3 + 3 * k^2 + 3 * k + 1$
Podemos escribir la identidad anterior para k de 1 hasta n:
$2^3 = 1^3 + 3 * 1^2 + 3 * 1 + 1$ .... (1)
$3^3 = 2^3 + 3 * 2^2 + 3 * 2 + 1$ .... (2)
$4^3 = 3^3 + 3 * 3^2 + 3 * 3 + 1$ .... (3)
$5^3 = 4^3 + 3 * 4^2 + 3 * 4 + 1$ .... (4)
...
$n^3 = (n - 1)^3 + 3 * (n - 1)^2 + 3 * (n - 1) + 1$ .... (n - 1)
$(n + 1)^3 = n^3 + 3 * n^2 + 3 * n + 1$ .... (n)

Ponemos la ecuación (n - 1) en la ecuación n,
$(n + 1)^3 = (n - 1)^3 + 3 * (n - 1)^2 + 3 * (n - 1) + 1 + 3 * n^2 + 3 * n + 1$
$= (n - 1)^3 + 3 * (n^2 + (n - 1)^2) + 3 * (n + (n - 1)) + 1 + 1$

Al poner toda la ecuación, obtenemos
$(n + 1)^3 = 1^3 + 3 * \Sigma k^2 + 3 * \Sigma k + \Sigma 1$
$n^3 + 3 * n^2 + 3 * n + 1 = 1 + 3 * \Sigma k^2 + 3 * (n * (n + 1)) /2 + n$
$n^3 + 3 * n^2 + 3 * n = 3 * \Sigma k^2 + 3 * (n * (n + 1)) /2 + n$
$n^3 + 3 * n^2 + 2 * n - 3 * (n * (n + 1)) /2 = 3 * \Sigma k^2$
$n * (n^2 + 3 * n + 2) - 3 * (n * (n + 1)) /2 = 3 * \Sigma k^2$
$n * (n + 1) * (n + 2) - 3 * (n * (n + 1)) /2 = 3 * \Sigma k^2$
$n * (n + 1) * (n + 2 - 3/2) = 3 * \Sigma k^2$
$n * (n + 1) * (2 * n + 1) /2 = 3 * \Sigma k^2$
$n * (n + 1) * (2 * n + 1) /6 = \Sigma k^2$

```python
def sumaCuadrados(n) :
    #  Iterar i desde 1
    #  y n buscando
    #  el cuadrado de i
    #  y añadiendolo a la suma
    sum = 0
    for i in range(1, n + 1) :
        sum = sum + (i * i)
    return sum


n = 4
print(sumaCuadrados(n))
```

Forma de estado:

(n, sum)

Estado inicial:

(4, 0)

Transformaciones:

(n, sum) → (n+1, sum + i*i)

Estado final:

$(4, \sum_{n=4}^{sum+i*i} i * i)$

Invariante:

sum = $\sum_{n=4}^{sum+i*i} i * i$