**Ejercicios: Análisis Teórico de Algoritmos**

**Problema 1.** Escribir un método, llamado sumPair0, que acepta un array de enteros, vector, como parámetro y devuelve el número de pares (i, j), como vector [i] + vector [j]

= 0. Calcular la función del tiempo de ejecución T (n).

public static int sumPar(int[] v){

int count = 0;

int n=v.length;

for(int i = 0; i <n; i++) {

for (int j = i+1; j <n; j++){

if (v[j]+v[i]==0)

count++;

}

}

return count;

}

Calcular T(n)

n=v.length

El peor caso es que siempre se ejecute count++

|  |  |  |
| --- | --- | --- |
|  | Numero de Operaciones | Porqué |
| int j=i+1 | 3 | Declaración, asignación, suma |
| j<n | 1\*(n-1)+1=n | Para j=1 hasta n-1 la condición es verdadera, se hace una vez mas como falsa j=n |
| j++ | 2\*(n-1)=2n-2 | Se hace desde j=1 hasta n-1 |
| v[i]+v[j]==0 | 4\*(n-1)=4n-4 | Se considera 4, por dos accesos al vector, la suma y la comparación, se repite (n-1) veces |
| count++ | 2\*(n-1)=2n-2 | 2 por count=count+1, y se repite n-1 veces |
|  | 9n-5 |  |

public static int sumPar(int[] v){

int count = 0;

int n=v.length;

for(int i = 0; i <v.length; i++) {

for (int j = 1+i; j <v.length; j++){

if (v[j]+v[i]==0)

dobles++;

}

}

return count;

}

|  |  |  |
| --- | --- | --- |
|  | Numero de Operaciones | Porque? |
| int count=0; | 2 | Declaración+asignación |
| int n=v.length; | 3 | Declaración+asignación+Acceso a vector |
| int i=0; | 2 | Declaración+asignación |
| i<n | 1\*n+1=n+1 | Para i=0 hasta n-1, la condición es verdadera, pero se ejecuta una vez más cuando i=n |
| i++ | 2\*n = 2n | La instrucción equivale a 2 operaciones, y se hace desde i=0 hasta n-1 |
| Bucle interno | (9n-5)\*n = 9n2-5n | El bucle interno es ejecutado desde i=0 hasta n-1(n veces) |
| return count | 1 |  |
|  | 9n2-2n+9 |  |

El T(n) es 9n2-2n+9 <= c n2

El BigO es O(n2)

F(n) = n2

9n2-2n+9 <= **c** n2  para n>=n0

**Escogemos c=9**

9n2-2n+9 <= 9 n2

9 <= 9 n2- 9n2+2n

9 <= 2n

4.5 <= n, entonces un valor válido sería, **n0=5**

**Problema 2.** Escribir un método, llamado sumTriple0, que acepte un array de enteros, vector, como parámetro y devuelva el número de triples (i, j, k), como vector [i] + vector [j] + vector [k] = 0 tal que i<j<k. Calcular la función del tiempo de ejecución T (n).

public static int sumTriple(int[] v) {

int count = 0;

int n=v.length;

for (int i = 0; i<n; i++) {

for (int j = i+1; j<n; j++) {

for (int k = j+1; k<n; k++) {

if (v[i] + v[j] + v[k] == 0) {count++;}

}

}

}

return count;

}

|  |  |  |
| --- | --- | --- |
|  | Número de Operaciones | Porqué? |
| int k=j+1 | 3 | Declaración, asignación, suma |
| k<n | 1\*(n-2)+1=n-1 | Desde k=2 hasta n-1 la condición es verdadera, se hace una vez más como falsa k=n |
| k++ | 2\*(n-2)=2n-4 | La instrucción equivale a 2 operaciones y se hace desde k=2 hasta n-1 |
| v[i]+v[j] +v[k]==0 | 6\*(n-2)=6n-12 | Se considera 6, por 3 accesos al vector, las 2 sumas y la comparación, se repite (n-2) veces |
| count++ | 2\*(n-2)=2n-4 | 2 por count=count+1, y se repite n-2 veces |
| Bucle interno | 11n-18 |  |
|  |  |  |
|  |  |  |
|  |  |  |
| Bucle Intermedio |  |  |
| int j = i+1 | 3 | Declaración, asignación, suma |
| j<n | 1\*(n-1)+1 =n | Para j=1 hasta n-1 la condición es verdadera, se hace una vez más como falsa j=n |
| j++ | 2\*(n-1)=2n-2 | Equivale a 2 operaciones y se hace desde j=1 hasta n-1 |
| Instrucción interna | (11n-18)\*(n-1)=11n2-29n+18 | Se hace desde j=1 hasta n-1 |
|  | 11n2-26n+19 |  |
| Bucle externo y algoritmo completo |  |  |
| int count=0; | 2 | Declaración+asignación |
| int n=v.length; | 3 | Declaración+asignación+Acceso a vector |
| int i = 0 | 2 | Declaración+asignación |
| i<n | n+1 | Para i=0 hasta n-1, la condición es verdadera, pero se ejecuta una vez más cuando i=n |
| i++ | 2\*n | La instrucción equivale a 2 operaciones, y se hace desde i=0 hasta n-1 |
| Instruccion interna | (11n2-26n+19)\*n=11n3-26n2+19n | La instrucción interna es ejecutada desde i=0 hasta n-1 (n veces) |
| return count | 1 |  |
|  | 11n3-26n2+22n+9 |  |

T(n) = 11n3-26n2+22n+9, el BigO O(n3)

F(n)= n3

11n3-26n2+22n+9 <= **c**n3, para n>=n0

**Escogemos c=11**

11n3-26n2+22n+9 <= 11 n3

26n2-22n-9 >= 0

(n-a)(n-b)>=0, podemos escoger n0 = 2

🡨----- -30.16 ------------ 1.15 ----------🡪