uc3m | Universidad Carlos III de Madrid

Autor: Profesores EDA

Estructura de Datos y Algoritmos
Ejercicios Tema 3: Análisis 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).

Solución:

```
public static int sumPair0(int[] v) {
        int result=0;
        for (int i=0; i<v.length;i++) {
                for (int j=i+1;j<v.length;j++) {
                        if (v[i]+v[j]==0)
                                result=result+1;
                }
        }
        return result;
}
```

Tenemos que contar las operaciones primitivas

Vamos a definir n = v.length. Comenzamos con el bucle interno

```
for (int j=i+1;j<v.length;j++) {
        if (v[i]+v[j]==0)
                result=result+1;
}
```

Cuando analizamos un algoritmo, siempre consideramos el peor de los casos. El peor caso para este ciclo es cuando i = 0 (porque el ciclo se ejecuta más veces)

```
for (int j=i+1;j<v.length;j++) {
        if (v[i]+v[j]==0)
                result=result+1;
}
```

	#operations	Why?
int j=i+1	2	declaration+asignation

j<v.length	(n-1)+1	For j=1 to n-1, the condition is evaluated as true (n-1) times, and one more time as false for j=n
j++	n-1	The operation is executed for j=1 to n-1, that is, n-1 times
v[i]+v[j]==0	3*(n-1)	2 (Index two elements in the array) + 1 (evaluate the expression).
result=result+1;	1*(n-1)	1 (assignation)
	6n-3	

Nota: El tiempo de ejecución para una estructura If / Else: el tiempo de ejecución de la evaluación de la condición más el máximo de tiempos de ejecución de S1 (instrucciones para If) y S2 (instrucciones para Else).

Ya hemos calculado el tiempo de ejecución para el ciclo interno $T_{inner}(n)= 6n-3$

Ahora, vamos a calcular el tiempo de ejecución para todo el algoritmo:

```
int result=0;
for (int i=0; i<v.length;i++) {
        for (int j=i+1;j<v.length;j++) {
                if (v[i]+v[j]==0)
                        result=result+1;
        }
}
return result;
```

	#operations	**Why?**
int result=0;	2	declaration+asignation
int i=0	2	declaration+asignation
i<v.length	n+1	For i=0 to n-1, the condition is evaluated as true n times, and one more time as false when i=n
i++	N	The operation is executed for i=0 to n-1, that is, n times.
Inner loop	n*Tinner(n)= $n*(6n-2)=6n^2-2n$	The inner loop is executed n times (from i=0 to n-1). The running time of the inner loop is 7n-4.
Return result	1	Return
	$6n^2-n+6$	

Por lo tanto, el tiempo de ejecución del algoritmo es $T(n)= 6n^2-n+6$

el orden de complejidad $O(x^2)$ cuadrática

Problem 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 result=0;
        for (int i=0; i<v.length;i++) {
                for (int j=i+1;j<v.length;j++) {
                        for (int k=j+1;k<v.length;k++)
                        if (v[i]+v[j]+v[k]==0)
                                result=result+1;
                }
        }
        return result;
```

SOLUCION

Tenemos que contar las operaciones primitivas

Vamos a definir n = v.length. Comenzamos con el bucle interno. Cuando analizamos un algoritmo, siempre consideramos el peor de los casos. El peor caso para este ciclo es cuando i = 0 (porque el ciclo se ejecuta más veces)

```
for (int k=j+1;k<v.length;k++)
        if (v[i]+v[j]+v[k]==0)
        result=result+1;
}
```

	#operations	**Why?**
int k=j+1	2	declaration+asignation
k<v.length	(n-2)+1	For j=1 to n-1, the condition is evaluated as true (n-2) times, and one more time as false for j=n
k++	n-2	The operation is executed for j=1 to n-1, that is, n-2 times
v[i]+v[j]+v[k]==0	4*(n-2)	3 (Index three elements in the array) + 1 (evaluate the expression).
result=result+1;	1*(n-2)	1 (assignation)
	7n-11	

Ya hemos calculado el tiempo de ejecución para el ciclo interno $T_{inner}(n)= 7n-11$

Ahora, vamos a calcular el tiempo de ejecución para el ciclo intermedio que contiene el interno:

```
for (int j=i+1;j<v.length;j++) {
        for (int k=j+1;k<v.length;k++)
                if (v[i]+v[j]+v[k]==0)
                result=result+1;
}
```

	#operations	**Why?**
int j=i+1	2	declaration+asignation
j<v.length	(n-1)+1	For i=0 to n-1, the condition is evaluated as true n times, and one more time as false when i=n
j++	n-1	The operation is executed for i=0 to n-1, that is, n times.
Inner loop	n-1*Tinner(n)= (n-1)*(7n-11)=$6n^2$-16n+10	The inner loop is executed n times (from i=0 to n-1). The running time of the inner loop is 7n-11.
	$7n^2$-16n+12	

Ya hemos calculado el tiempo de ejecución para el ciclo intermedio $T_{inner}(n)= 7n^2-16n+12$

Ahora, vamos a calcular el tiempo de ejecución para todo el algoritmo:

```
public static int sumTriple(int[] v) {
        int result=0;
        for (int i=0; i<v.length;i++) {
                for (int j=i+1;j<v.length;j++) {
                        for (int k=j+1;k<v.length;k++)
                        if (v[i]+v[j]+v[k]==0)
                                result=result+1;
                }
        }
        return result;
```

	#operations	**Why?**
int result=0;	2	declaration+asignation
int i=0	2	declaration+asignation
i<v.length	n+1	For i=0 to n-1, the condition is evaluated as true n times, and one more time as false when i=n
i++	N	The operation is executed for i=0 to n-1, that is, n times.
Inner loop	n*Tinner(n)= $n*(7n^2-16n+12)=6n^3-3n$	The inner loop is executed n times (from i=0 to n-1). The running time of the inner loop is $7n^2-16n+12$.
Return result	1	Return
	$7n^3-16^2+4n+6$	

Por lo tanto, el tiempo de ejecución del algoritmo es T(n)= $7n^3-16^2+4n+6$

el orden de complejidad $O(x^3)$ cúbico