ALGORÍTMICA Y MODELOS DE COMPUTACIÓN

Practica 1 - Algoritmos Exhaustivos y Divide y Vencerás

Nombre: Saúl Rodríguez Naranjo

ETSI
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA
uhu.es

ÍNDICE

1. Estudio Teórico del Pseudocódigo

1.1 Algoritmo Exhaustivo

El pseudocódigo del algoritmo exhaustivo es el siguiente:

```
funcion exhaustivo(T[]: Punto, i: Entero, d: Entero) return solucion

	Trio solucion := Trio(T[0], T[1], T[2]) --> 1 Asignacion + 1 de llamada a Constructor + 3 accesos a estructuras indexadas = 5 OE
	Trio aux


	Para a1 := i Hasta d Hacer --> 1, 1, 2 (1 Salto)
		Para a2 := a1 + 1 Hasta d Hacer --> 2, 1, 2 (1 Salto)
			Para a3 := a2 + 1 Hasta d Hacer --> 2, 1, 2 (1 Salto)

			aux := Trio(T[a1], T[a2], T[a3]) --> 1 Asignacion + 1 de llamada a Constructor + 3 accesos a estructuras indexadas = 5 OE

			Si aux.distancia < solucion.distancia --> 1 Comparacion

				solucion := aux --> 1 Asignacion

			fSi

			fPara
		fPara
	fPara

	return solucion

ffuncion
```

$$T_{Exhaustivo} = 5 + T_{Bucle(1)}$$

$$T_{Bucle(1)} = T_{Inicialización(1)} + T_{Condición(1)} + T_{Salto} + \sum_{a1=i=0}^{d-1} T_{Ciclo(1)}$$

$$T_{Ciclo(1)} = T_{Bucle(2)} + T_{Incremento} + T_{Salto} + T_{Condición}$$

$$T_{Bucle(2)} = T_{Inicialización(2)} + T_{Condición(2)} + T_{Salto} + \sum_{a2=a1+1=1}^{d-1} T_{Ciclo(2)}$$

$$T_{Ciclo(2)} = T_{Bucle(3)} + T_{Incremento} + T_{Salto} + T_{Condición}$$

$$T_{Bucle(3)} = T_{Inicialización(3)} + T_{Condición(3)} + T_{Salto} + \sum_{a3=a2+1=2}^{d-1} T_{Ciclo(3)}$$

1. Estudio Teórico del Pseudocódigo

$$T_{Ciclo(3)} = T_{Cuerpo} + T_{Incremento} + T_{Salto} + T_{Condición} = 7 + 4 = 11$$

$$T_{Bucle(3)} = 2 + 1 + 1 + \sum_{a3=a2+1=2}^{d-1} 11 = 4 + [11 \cdot (d-2)] = 11d - 18$$

$$T_{Ciclo(2)} = 11d - 18 + 2 + 1 + 1 = 11d - 14$$

$$T_{Bucle(2)} = 2 + 1 + 1 + \sum_{a2=a1+1=1}^{d-1} 11d - 14 = 4 + [(d-1)(11d-14)]$$
$$= 11d^2 - 25d + 18$$

$$T_{Ciclo(1)} = 11d^2 - 25d + 18 + 2 + 1 + 1 = 11d^2 - 25d + 22$$

$$\boldsymbol{T_{Bucle(1)} = 1 + 1 + 1 + \sum_{a1=i=0}^{d-1} (11d^2 - 25d + 22)}$$
$$\boldsymbol{= 3 + [d \cdot (11d^2 - 25d + 22)]}$$
$$\boldsymbol{= 11d^3 - 25d^2 + 22d + 8 \in O(n^3)}$$

Debido a que los bucles siempre se realizan, ya que algoritmo no es capaz de distinguir cuando ha encontrado el trio solución, su Caso Medio = Caso Peor = Caso Mejor.

1. Estudio Teórico del Pseudocódigo

1.1 Algoritmo Divide y Vencerás

El Pseudocódigo del Divide y Vencerás es el siguiente:

```
funcion Divide_y_Venceras(Puntos[]: Punto) return solucion

    Si Puntos.longitud <= 6
       return exhaustivo(Puntos[], 0, Puntos.longitud - 1)
    Si no Entonces

       mitad := Puntos.longitud / 2
       mitad_izq := Punto[mitad]
       mitad_der := Punto[Puntos.longitud - mitad];

       Para i := 0 Hasta mitad Hacer
          mitad_izq[i] := Punto(Puntos[i])
       fPara

       Para i := 0 Hasta (Puntos.longitud - mitad) Hacer
          mitad_der[i] = new Punto(Puntos[i + mitad])
       fPara

       menor_izq := Divide_y_Venceras(mitad_izq[])
       menor_der := Divide_y_Venceras(mitad_der[])

       Si menor_izq.distancia < menor_der.distancia
          solucion := menor_izq
       Si no Entonces
          solucion := menor_der
       fSi

       dmin = solucion.distancia

       Para a := mitad Hasta 0 Hacer
          Si Puntos[mitad + 1].coordenadaX() - Puntos[a].coordenadaX() > dmin
             pararBucle
          fSi
       fPara

       Para b := mitad + 1 Hasta (Puntos.longitud - 1) Hacer
          Si Puntos[b].coordenadaX() - Puntos[mitad].coordenadaX() > dmin
             pararBucle
```

I. Estudio Teórico del Pseudocódigo

```
        fsi
    fPara

    a := a + 1
    b := b - 1

    Para c := a Hasta mitad Hacer
        Para d := mitad + 1 Hasta b Hacer
            Para e := d + 1 Hasta b Hacer

                aux := Trio(Puntos[c], Puntos[d], Puntos[e])

                Si aux.distancia < solucion.distancia
                    solucion := aux
                fSi

            fPara
        fPara
    fPara

    Para c := b (Hasta mitad + 1) Hacer
        Para d := a Hasta mitad Hacer
            Para e := d + 1 Hasta mitad Hacer

                aux := Trio(Puntos[c], Puntos[d], Puntos[e])

                Si aux.distancia < solucion.distancia
                    solucion := aux
                fSi

            fPara
        fPara
    fPara

    return solucion

  fSi

ffuncion
```

1. Estudio Teórico del Pseudocódigo

Si resolvemos la complejidad por reducción por división nos queda lo siguiente:

$$T(n) = \begin{cases} O(n^3) & n \leq 6 \\ a \cdot T\left(\frac{n}{2}\right) + f(n) & n > 6 \end{cases}$$

- Siendo el caso base O(n3) debido a que usamos la función exhaustiva para evaluarlo.
- En cuanto a f(n) es el coste de preparación de las llamadas y de combinación de los resultados. Como evaluamos con 3 bluces for el caso intermedio, es decir, los extremos del array, al final esta complejidad es O(n3).
- El atributo a, son las veces que llamamos a nuestra función recursiva.

Teniendo en cuenta lo anterior, finalmente el sistema quedaría de la siguiente forma:

$$\boldsymbol{T(n) = \begin{cases} O(n^3) & n \leq 6 \\ O(n^3 \log n) & n > 6 \end{cases}}$$

Al igual que ocurre con el algoritmo exhaustivo, en el algoritmo divide y vencerás, como usamos de caso base al algoritmo exhaustivo, Case Medio = Caso Peor = Caso Mejor.

1. Estudio Teórico del Pseudocódigo

1.3 Ejecuciones Representativas

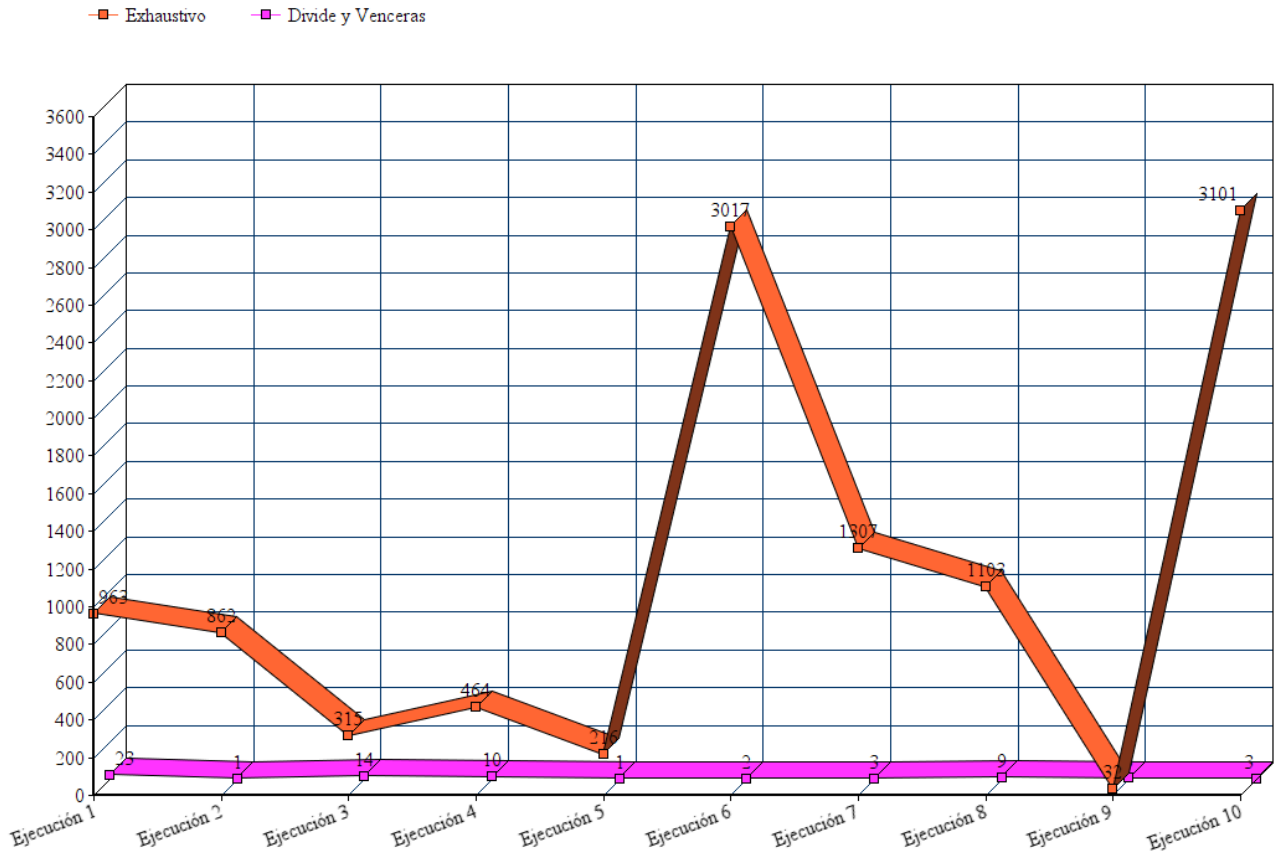
Se nos pide realizar 10 ejecuciones para los distintos algoritmos, como no existen casos para ellos, se realizarán 10 ejecuciones con tamaño aleatorio (entre 70 y 700 elementos) y datos aleatorios para los distintos algoritmos.

Algoritmo Exhaustivo	Tiempo de Ejecución	Algoritmo Divide y Vencerás	Tiempo de Ejecución
Ejecución 1	963 ms	Ejecución 1	23 ms
Ejecución 2	862 ms	Ejecución 2	1 ms
Ejecución 3	315 ms	Ejecución 3	14 ms
Ejecución 4	464 ms	Ejecución 4	10 ms
Ejecución 5	216 ms	Ejecución 5	1 ms
Ejecución 6	3017 ms	Ejecución 6	2 ms
Ejecución 7	1307 ms	Ejecución 7	3 ms
Ejecución 8	1103 ms	Ejecución 8	9 ms
Ejecución 9	32 ms	Ejecución 9	6 ms
Ejecución 10	3101 ms	Ejecución 10	3 ms

Como podemos observar, el algoritmo divide y vencerás es bastante superior.

2. Resolución Gráfica

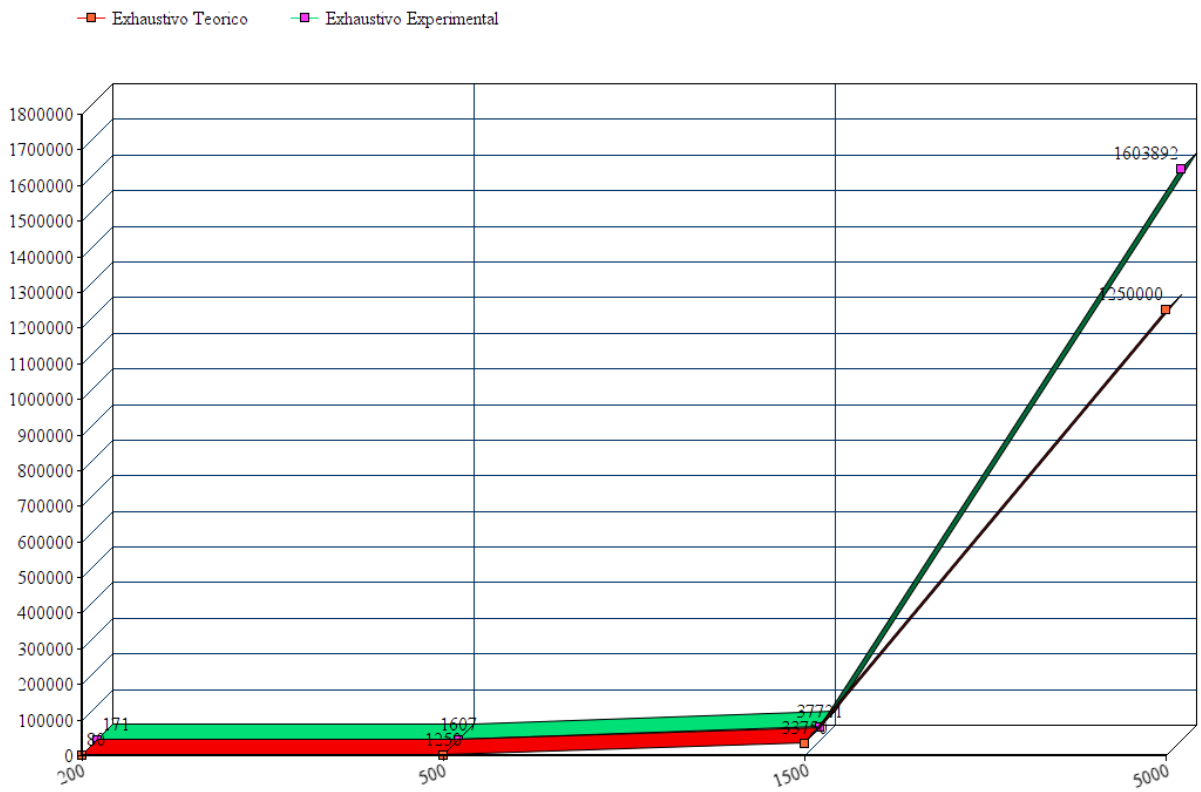
Debido a que no existen casos límites, para la representación gráfica utilizaremos los datos anteriormente obtenidos.



3. Comparación de los Resultados Teóricos con los Experimentales

3.1 Algoritmo Exhaustivo

Algoritmo Exhaustivo(Teórico)	Tiempo de Ejecución	Algoritmo Exhaustivo (Experimental)	Tiempo de Ejecución
200	80 ms	200	171 ms
500	1250 ms	500	1607 ms
1500	33750 ms	1500	37721 ms
5000	1250000 ms	5000	1603892 ms



3. Comparación de los Resultados Teóricos con los Experimentales

3.2 Algoritmo Divide y Vencerás

Algoritmo Divide y Vencerás (Teórico)	Tiempo de Ejecución	Algoritmo Divide y Vencerás (Experimental)	Tiempo de Ejecución
200	1.6 ms	200	2 ms
500	6.3 ms	500	8 ms
1500	10.7 ms	1500	14 ms
5000	46.2 ms	5000	38 ms