Carla Pérez Gavilán Del Castillo, A01023033
17 de abril de 2020
Prof. Vicente Cubells
Análisis y Diseño de Algoritmos

PROBLEMA 4:

Cálculo de mediana de dos vectores del mismo tamaño ordenados de forma ascendente

Sean X e Y dos vectores de tamaño n, ordenados de forma no decreciente. Se necesita implementar un algoritmo para calcular la mediana de los 2n elementos que contienen X e Y. Recuerde que la mediana de un vector de k elementos es aquel elemento que ocupa la posición (k+1)÷2 una vez el vector está ordenado de forma creciente. Dicho de otra forma, la mediada es aquel elemento que, una vez ordenado el vector, deja la mitad de los elementos a cada uno de sus lados. Como en este caso k = 2n (y por tanto par) se busca el elemento en posición n de la unión ordenada de X e Y

TÉCNICA: Divide y Vencerás

Cálculo de Complejidad

NOTA: Se calculará la complejidad de la función que calcula la mediana únicamente, las demás operaciones de impresión o acceso a una posición en el arreglo se consideran despreciables

FUNCIÓN:
template <class T>
T Media<T>::twoArrayMedian(T arr_a[], T arr_b[], int n)

No.	Instrucción	Operaciones Elementales	Ciclo	Veces	Total Simplificado
1	if(n<=0){	2	1	1	**2**

Carla Pérez Gavilán Del Castillo, A01023033
17 de abril de 2020
Prof. Vicente Cubells
Análisis y Diseño de Algoritmos

2	return -1;	0	0	0	0
3	if(n==1){	1	1	1	**1**
4	return (arr_a[0]+arr_b[0])/2;	4	1	1	**4**
5	if(n==2){	1	1	1	**1**
6	return (std::max(arr_a[0],arr_b[0])+std::min(arr_a[1], arr_b[1]))/2;	5 **	1	1	**5**
7	int median_a = getMedian(arr_a, n);	**Ver cálculo de complejidad de la función abajo			**9**
8	int median_b = getMedian(arr_b, n);				**9**
9	if(median_a==median_b){	1	1	1	**1**
10	return median_b;	1	1	1	**1**
11	if(median_a>median_b){	1	1	1	1

Carla Pérez Gavilán Del Castillo, A01023033
17 de abril de 2020
Prof. Vicente Cubells
Análisis y Diseño de Algoritmos

12	if(n%2==0){	2	2	2	2
13	return twoArrayMedian(arr_a + n / 2 - 1, arr_b, n - n / 2 + 1);	cada vez que llamamos la función de forma recursiva dividimos el arreglo en dos por lo tanto consideramos **ver demostración abajo			$Log_2(n)$
14	return twoArrayMedian(arr_b + n / 2, arr_a, n - n / 2);				$Log_2(n)$
COMPLEJIDAD TOTAL		Peor de los casos			$T(n) = 24$
		Mejor de los casos			$T(n) = 36\ Log_2(n)$

**Nota: consideramos las funciones max y min de la std despreciables.

FUNCIÓN AUXILIAR:

```
template <class T>
T Media<T>::getMedian(T sort_vect[], int n)
```

	Instrucción	Operaciones Elementales	Veces	Ciclos	Total Simplificado
1	if(n%2==0){	2	2	2	2

Carla Pérez Gavilán Del Castillo, A01023033
17 de abril de 2020
Prof. Vicente Cubells
Análisis y Diseño de Algoritmos

2	return (sort_vect[n/2]+sort_vect[n/2+1])/2;	7	7	7	7
3	return sort_vect[n/2];	2	2	2	2
Peor Caso		9	Mejor caso		4

DEMOSTRACIÓN DE LLAMADA RECURSIVA

** para poder comprender el comportamiento general de la recursividad nos fijamos en la variable n de forma recursiva.

$$T(a, b, 0) = -1$$
$$T(a, b, 1) = a[0]+b[0]/2$$
$$T(a,b, 2) = \max(a[0],b[0])+\min(a[1],b[1]))/2$$
$$T(a, b, 3) = T(a,b,1) = a[0]+b[0]/2$$
$$T(a, b, 4) = T(a,b,3) = a[0]+b[0]/2$$

Por lo tanto si consideramos lo siguiente:

$n - n/2 = n(1 - 1/2) = n(1/2)$

considerando que el más uno por ser constante es despreciable, veremos que la complejidad de la función es logarítmica

CÁLCULO DE COTA ASINTÓTICA

$$\lim_{n\to\infty} \frac{36 \cdot Log_2(n)}{Log_2(n)} = 36$$

Como el resultado es una constante sabemos que pertenencen al mismo orden de complejidad:

$O(n) = Log_2(n)$