

Práctica 1

Fecha límite de entrega: sábado, 30 de septiembre, a las 23:59

Suma de la subsecuencia máxima: Dados n números enteros a_1, a_2, \dots, a_n , encontrar el valor máximo de $\sum_{k=i}^j a_k$ con $1 \leq i \leq j \leq n$ (por conveniencia, la suma de la subsecuencia máxima es 0 si todos los enteros son negativos).

Se proponen dos algoritmos para resolver este problema:

Algoritmo 1: $O(n^2)$

```

función sumaSubMax1 (v[1..n])
    (int v [], int n) los índices van desde 1 a n, en c los arrays van desde 0 hasta n-1
    sumaMax := 0 ;
    para i := 1 hasta n hacer
        estaSuma := 0 ;
        para j := i hasta n hacer
            estaSuma := estaSuma + v[j] ;
            si estaSuma > sumaMax entonces
                sumaMax := estaSuma
        fin si
    fin para
    fin para ;
    devolver sumaMax
fin función
  
```

Algoritmo 2: $O(n)$

```

función sumaSubMax2 (v[1..n])
    estaSuma := 0 ; sumaMax := 0 ;
    para j := 1 hasta n hacer
        estaSuma := estaSuma + v[j] ;
        si estaSuma > sumaMax entonces
            sumaMax := estaSuma
        sino si estaSuma < 0 entonces
            estaSuma := 0
    fin si
    fin para ;
    devolver sumaMax
fin función
  
```

Se pide:

1. Implemente en C los algoritmos propuestos (véase la figura 5).

2. Valide que los algoritmos funcionan correctamente. Chequee las siguientes secuencias:

intentar coger el valor mas grande

diseñar test para una para otro
void test 1() hacer pruebas con estas secuencias
void test 2() con vectores al azar
ambas funciones dan el mismo resultado

secuencia	resultado
-9, 2, -5, -4, 6	6
4, 0, 9, 2, 5	20
-2, -1, -9, -7, -1	0
9, -2, 1, -7, -8	9
15, -2, -5, -4, 16	20
7, -5, 6, 7, -7	15

cogo el ultimo 6
son todos positivos
son todos negativos
cojo solo el nueve
cogo todos me sigue dando positivo

Así mismo realice una segunda comprobación con vectores generados de forma aleatoria (figura 1) comprobando que ambos algoritmos devuelven el mismo resultado (figura 2).

ponerla al principio del main

```
void inicializar_semilla() {
    srand(time(NULL));
    /* se establece la semilla de una nueva serie de enteros pseudo-aleatorios */
}

void aleatorio(int v [], int n) {
    int i, m=2*n+1;
    for (i=0; i < n; i++)
        v[i] = (rand() % m) - n;
    /* se generan números pseudoaleatorio entre -n y +n */
}
```

la semilla se usa para trabajar cada vez sobre un numero distinto para calcular la siguiente secuencia

Figura 1: Inicialización de un vector con números pseudoaleatorios en el rango $[-n, \dots, +n]$

numeros aleatorios tiene que dar el mismo resultado

	sumaSubMax1	sumaSubMax2
[7 0 -3 3 -1 1 -5 5 -8]	7	7
[-8 8 1 -1 7 6 -5 -4 -7]	21	21
[-1 -7 -8 -6 7 -2 1 -2 -7]	7	7
[8 -3 8 9 -9 -5 -4 3 1]	22	22
[3 8 -4 5 6 -9 -4 -8 7]	18	18
[9 -6 -6 8 2 -7 -9 5 9]	14	14
[-5 -7 -6 -3 -8 -3 -5 -9 -3]	0	0
[-3 -7 -9 7 6 5 -7 -2 -2]	18	18
[-7 0 0 7 -1 3 -9 -5 -8]	9	9
[9 -5 3 8 -1 6 9 3 4]	36	36

Figura 2: Aplicadas sobre distintos vectores aleatorios, ambas funciones devuelven el mismo resultados.

- Para cada uno de los dos algoritmos, determine los tiempos de ejecución con vectores aleatorios de tamaño n igual a 500, 1000, 2000, 4000, 8000, 16000 y 32000. Use el código de la figura 3 para obtener la hora del sistema. Para generar los datos de prueba utilice el código de la figura 1 que genera vectores de números pseudoaleatorios en el rango $[-n, \dots, n]$.

```
#include <sys/time.h>
double microsegundos() { /* obtiene la hora del sistema en microsegundos */
    struct timeval t;
    if (gettimeofday(&t, NULL) < 0 )
        return 0.0;
    return (t.tv_usec + t.tv_sec * 1000000.0);
}
```

Figura 3: Obtención de la hora del sistema

4. Analice los resultados obtenidos realizando una comprobación empírica de la complejidad teórica (figura 4). Igualmente se realizará una comprobación empírica utilizando una cota subestimada y otra sobre-estimada para cada algoritmo.

```
$ ./p1
SumaSubMax 1
```

	n	t (n)	t (n) / n ^{1.8}	t (n) / n ²	t (n) / n ^{2.2}
(*)	500	341.666	0.004736	0.001367	0.000394
	1000	1200.000	0.004777	0.001200	0.000301
	2000	4819.000	0.005509	0.001205	0.000263
	4000	19178.000	0.006296	0.001199	0.000228
	8000	85178.000	0.008031	0.001331	0.000221
	16000	332606.000	0.009006	0.001299	0.000187
	32000	1270463.000	0.009879	0.001241	0.000156

Figura 4: Parte de la posible salida por pantalla del programa que mide los tiempos de ejecución del primer algoritmo.

5. Entregue los ficheros con el código C y el fichero .txt con el informe por medio de la tarea *Entrega Práctica 1* en la página de Algoritmos en <https://campusvirtual.udc.gal>. Se recuerda que el límite para completar la tarea es el sábado 30 de septiembre a las 23:59, y una vez subidos los archivos no se podrán cambiar. **Todos los compañeros que forman un equipo tienen que entregar el trabajo.**

```

#include <stdio.h>
int sumaSubMax1(int v[], int n) {
    int i, j;
    int estaSuma, sumaMax = 0;
    for (i = 0; i < n; i++) {
        estaSuma = 0;
        for (j = i; j < n; j++) {
            estaSuma += v[j];
            if (estaSuma > sumaMax) {
                sumaMax = estaSuma;
            }
        }
    }
    return sumaMax;
}
int sumaSubMax2(int v[], int n) {
    /* ... */
}
void listar_vector(int v[], int n){
    /* ... */
}
void test1() {
    /* ... */
}
void test2() {
    int i, a, b;
    int v[9];
    printf("test\n");
    printf("%33s%15s%15s\n", "", "sumaSubMax1", "sumaSubMax2");
    for (i=0; i<10; i++) {
        aleatorio(v, 9);
        listar_vector(v, 9);
        a = sumaSubMax1(v, 9);
        b = sumaSubMax2(v, 9);
        printf("%15d%15d\n", a, b);
    }
}
int main() {
    inicializar_semilla();
    test1();
    test2();
    return 0;
}

```

Figura 5: Código con la función `sumaSubMax1` y el segundo test