INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

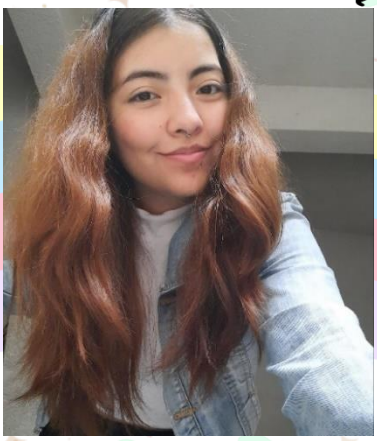
ANÁLISIS DE ALGORITMOS

EJERCICIOS 03:

SIMULACIÓN PRODUCTO2MAYORES

ALUMNA: MÉNDEZ CASTAÑEDA AURORA

GRUPO: 3CM13



PROFESOR: FRANCO MARTÍNEZ EDGARDO ADRIÁN

Ejercicios 03: "Simulación Producto2Mayores"

Para el algoritmo analizado por casos en clase y video lección "Producto2Mayores", realice la simulación de su mejor, peor y caso medio; realizando las modificaciones y adaptaciones necesarias para verificar los tres casos en n=2500 y n=5000 considerando al menos 10,000 iteraciones del algoritmo con cada n y diferente distribución de los números.

- Para el mejor caso basta con tener un archivo de números que coloque en los dos primeros números a los dos mayores.
- Para el peor caso basta con tener un archivo ordenado ascendentemente para cada n.
- Para el caso medio se deberán de hacer al menos 10,000 iteraciones para cada n generando arreglos aleatorios en cada iteración y comprobar el número de operaciones básicas promedios totales para enfrentarlas al modelo del caso medio.

```
1  func Producto2Mayores(A, n)
2  if (A[1] > A[2])
3  mayor1 = A[1];
4  mayor2 = A[2];
5  else
6  mayor1 = A[2];
7  mayor2 = A[1];
8  i = 3;
9
10 while (i <= n)
11 if (A[i] > mayor1)
12 mayor2 = mayor1;
13 mayor1 = A[i];
14 else if (A[i] > mayor2)
15 mayor2 = A[i];
16 i = i + 1;
17
18 return = mayor1 * mayor2;
```

El problema del algoritmo de producto mayores es que dado un arreglo de valores, encontrar el producto de los dos números mayores.

El tamaño de problema es n, el cual representa el número de elementos en el arreglo.

Las operaciones básicas para este algoritmo serán las comparaciones con los elementos del arreglo y las asignaciones a los elementos mayores.

Análisis Temporal

Mejor Caso: este ocurre cuando los dos primeros números del arreglo son los números mayores, es decir, el arreglo se encuentre ordenado descendentemente.

Función temporal: $f_t(n) = 2n - 1$

Para la simulación de este caso haremos uso de un archivo (numerosOrdenadosDescendentemente.txt) que contiene 5000 números ordenados descendentemente para poder llenar nuestro arreglo A. Probaremos con dos diferentes n realizando 10000 iteraciones.

$f_t(n)$		
n	**Teórico**	**Empírico**
2500	4999	4999.664
5000	9999	9999.832

Peor caso: para el peor caso tenemos dos instancias diferentes, una de ellas es que el arreglo este ordenado ascendentemente y la otra instancia es que uno de los dos números mayores se encuentre al principio del arreglo y el otro al último de este. Sin embargo, para estas dos diferentes instancias se obtiene la misma función temporal.

Función temporal: $f_t(n) = 3n - 3$

Para la simulación tomaremos la instancia de que el arreglo se encuentre ordenado ascendentemente, es por eso por lo que usaremos un archivo de 5000 números con dicha característica (numerosOrdenadosAscendentemente.txt) y lo probaremos para dos n diferentes realizando 10000 iteraciones.

$f_t(n)$		
n	**Teórico**	**Empírico**
2500	7497	7496.224
5000	14997	14997.480

Caso medio: para el caso medio consideraremos que las instancias del mejor y peor caso son equiprobables, es decir que cada caso tiene la misma probabilidad de ocurrencia, entonces en promedio se harán:

- Caso: arreglo ordenado descendentemente $= \frac{1}{3}(2n - 1)$
- Caso: arreglo ordenado ascendentemente $= \frac{1}{3}(3n - 3)$
- Caso: uno de los dos números mayores se encuentre al principio del arreglo y el otro al último de este $= \frac{1}{3}(3n - 3)$

Por lo tanto, nuestro caso medio sería:

$$f_t(n) = \frac{1}{3}\big(2(3n - 3) + (2n - 1)\big) = \frac{1}{3}(8n - 7)$$

Función temporal: $f_t(n) = \frac{1}{3}(8n - 7)$

Para la simulación en lugar de hacer uso de archivos para llenar nuestro arreglo, tendremos que crear uno con números aleatorios para cada iteración.

$f_t(n)$		
n	**Teórico**	**Empírico**
2500	6664.3	5006.304
5000	13331	10007.432

La diferencia de resultados podría deberse a que los arreglos obtenidos aleatoriamente no abarcaron los tres casos de forma igual.

Código C

```c
#include<stdio.h>
#include<stdlib.h>
#include<time.h>

#define N_VECES 10000
/*DEFINICIÓN DE FUNCIONES*/
int siExisteN_en_A(int *A, int n, int num);
void generarArreglo(int *A,int n);

int main(int argc, char* argv[]){
    int n,i=0,*A,k,cont=0,mayor1,mayor2;
    float promedio;

    if (argc!=2) { //Si no se introducen exactamente 2 argumentos (Cadena de ejecución y cadena=n)
        printf("\nIndique el tamanio del algoritmo");
        exit(1);
    } else { //Tomar el segundo argumento como tamaño del arreglo
        n=atoi(argv[1]);
    }

    A = malloc(sizeof(int)*n); //asignamos el tamaño del arreglo de acuerdo al valor n

    if(A == NULL) //en caso de que no se haya llenado correctamente y el arreglo no tenga elementos
        exit(1);

    /*for(i=0;i<n;i++) //llenamos el arreglo
        scanf("%d", &A[i]);*/


    for(k=0; k<N_VECES; k++){ // realizamos N_VECES iteraciones el algoritmos

        //mandamos a llamar la función para crear un arreglo de números aleatorios para caso medio
        generarArreglo(A,n);
        /*for(int l=0; l<n; l++)
            printf("\n%d", A[l]);*/

        cont=0; //variable auxiliar para contabilizar las operaciones básicas

        if(A[0] > A[1]){
            cont++; // if true
            mayor1 = A[0]; cont++; //asignacion mayor1
            mayor2 = A[1]; cont++; //asignacion mayor2
        } else {
            cont++; //if false
            mayor1 = A[0]; cont++;//asignnacion mayor1
            mayor2 = A[1]; cont++;//asignacion mayor2
        }

        i = 2;
        while (i < n){
            if(A[i] > mayor1){
                cont++; //if1 true
```

```
                mayor2 = mayor1; cont++; //asignacion mayor2
                mayor1 = A[i]; cont++; //asignacion mayor1
            } else if(A[i] > mayor2){
                cont++; //if2 true
                mayor2 = A[i]; cont++;//asignacion mayor2
            } else {
                cont +=2; //en caso de que ninguno de los if's de arriba se cumpla
            }
            i++;
        }
        promedio = promedio + (float)cont; //se va sumando las operaciones totales de cada iteración
    }
    promedio = promedio/N_VECES;//se obtiene el promedio de las operaciones totales de cada iteración
    printf("\nPromedio = %f operaciones",promedio);
    printf("\nTotal de operaciones: %d",cont);

}

/*Esta función tiene como propósito crear un arreglo de n números aleatorios
Recibe como parámetros el arreglo A y el tamaño del arreglo n.*/
void generarArreglo(int *A,int n){
    int i, j, num;
    srand(time(NULL));

    for (i=0; i<n; i++){
        while(siExisteN_en_A(A, n, num = rand()%100000));
        A[i] = num;
    }
}

/*Esta función nos ayuda a verificar si se repite algún número que será insertado
al arreglo, y si es el caso, busca otro para ingresarlo y que no haya números repetidos
en el arreglo.

Recibe como parámteros el arreglo A, el tamaño del arreglo y el número que será o
no insertado al arreglo.*/
int siExisteN_en_A(int *A, int n, int num){
    int i, r=0;

    for(i=0; i<n && !r; i++){
        if(A[i] == num)
            r=1;
    }
    return r;
}
```

Las instrucciones de compilación y ejecución son las siguientes:

→ `gcc producto2mayores.c -o producto2mayores`

→ `producto2mayores.exe 2500`

Funcionamiento:

```
C:\Users\jorge\Desktop\ESCOM\5to Semestre\Algoritmos\Programas\Ejercicios03>producto2mayores.exe 2500
32767 32744
Promedio = 5006.423828 operaciones
Total de operaciones: 5006
```