



CS1103

Programación Orientada a Objetos 2

Unidad 3 - Semana 5 - Analisis Asintótico - Ejercicios

Carlos Arias

Marvin Abisrror

Rubén Rivas

Objetivos

Al finalizar la sesión, los alumnos podrán analizar algoritmos a través de notaciones matemáticas y realizar comparaciones que ayuden a encontrar la solución de problemas de forma optima.



Ejercicio #1

Cuál es el tiempo de ejecución de fun()

```
int fun(int n) {  
    int count = 0;  
    for (int i = 0; i < n; i++) {  
        for (int j = i; j > 0; j--) {  
            count += 1;  
        }  
    }  
    return count;  
}
```

Ejercicio #2

Cuál es el tiempo de ejecución de fun2()

```
int fun2(int n) {  
    int i, j;  
    for (i = 1; i <= n; i++) {  
        for (j = 1; j < log(i); j++) {  
            cout<<"abc"<<endl;  
        }  
    }  
    cout<<endl;  
}
```

Ejercicio #3

Sea $peor(n)$ y $promedio(n)$ respectivamente, el caso peor y promedio en el tiempo de ejecución de un algoritmo dado un tamaño de entrada n . Cual de las siguientes opciones es verdadero?

A) $promedio(n) = \Omega(peor(n))$

B) $promedio(n) = \Theta(peor(n))$

C) $promedio(n) = O(peor(n))$

D) $promedio(n) = o(peor(n))$

Ejercicio #4

Cual de las siguientes opciones es $O(n^2)$?

A) $7^{20} * n$

B) $17^{15} * n^n + 144$

C) n^3/\sqrt{n}

D) $n^{1.98}$

Ejercicio #5

Cual es el tiempo de ejecución del siguiente algoritmo?

```
void recursive(int n) {  
    if (n <= 1)  
        return;  
    cout<<"utec ";  
    recursive(n/2);  
    recursive(n/2);  
}
```


Ejercicio #6

Cual es el tiempo de ejecución del siguiente algoritmo?

```
void loop(int n, int k) {  
    for (int i = 0; i < n; i++) {  
        for (int j = 1; j < n; j=j*k) {  
            cout<<"utec "<<endl;  
        }  
    }  
    cout<<endl;  
}  
  
int main() {  
    int n = 10, k = 2;  
    loop(n, k);  
  
    return 0;  
}
```

Ejercicio #7

Cuál de las siguientes opciones provee una secuencia creciente en relación al orden de crecimiento de las funciones f_1 , f_2 , f_3 , y f_4 ?

$$f_1 = n^{\log n}$$

$$f_2 = n \log n$$

$$f_3 = n^{\frac{3}{2}}$$

$$f_4 = 2^n$$

$$A) f_2 < f_1 < f_4 < f_3$$

$$B) f_1 < f_2 < f_3 < f_4$$

$$C) f_2 < f_3 < f_1 < f_4$$

$$D) f_4 < f_3 < f_1 < f_2$$

Ejercicio #8

Elaborar una función **get_anagrams** con 2 parámetros, el primero para recibir una palabra y el segundo el nombre de un archivo y que devuelva un vector con todas las palabras que sean anagrama del primer parámetro contenidas en el archivo, cada palabra estará escrita en una línea del archivo.

```
vector<string> get_anagrams(string word, string file_name);
```

Determinar el tiempo de ejecución y evaluar el mejor y peor escenario.



Ejercicio #9

Elaborar una clase functor **suma_cuatro** que genere un arreglo aleatorio de enteros de tamaño **n**, y que retorne el arreglo generado y un set de arreglos con todas las posibles combinaciones de ítems que genere una suma igual a un valor dado. La clase debe sobrecargar el operador parámetros.

```
pair<int*, set<int*>> operator()(int value);
```

Determinar el tiempo de ejecución y evaluar el mejor y peor escenario.

Ejercicio #10

Elaborar una clase functor **par_lejano** que genere un arreglo aleatorio de **doubles** de tamaño **n**, y que retorne un **pair** con los 2 valores que tengan la mayor diferencia entre 2 valores del arreglo (en valor absoluto) , el programa debía ser lineal en el peor de los casos. La clase debe sobrecargar el operador parámetros.

```
pair<int, int> operator()();
```

Determinar el tiempo de ejecución y evaluar el mejor y peor escenario.

Ejercicio #11

Elaborar una función **par_cercano** que genere un arreglo aleatorio de **doubles** de tamaño **n**, y que retorne un **pair** con los 2 valores que tengan la menor diferencia (en valor absoluto) entre 2 valores del arreglo, el programa debía ser linealitmica en el peor de los casos. La clase debe sobrecargar el operador parámetros.

```
pair<int, int> operator()();
```

Determinar el tiempo de ejecución y evaluar el mejor y peor escenario.

Ejercicio #12

Elaborar una clase functor **minimo_local** que genere un arreglo aleatorio de **int** de tamaño **n** con distintos valores que encuentre el valor mínimo-local, que es aquel índice i de modo que $a[i-1] < a[i] < a[i+1]$. El programa debería usar aproximadamente $2\log N$ comparaciones en el peor de los casos.

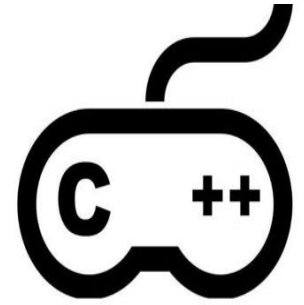
Respuesta: Examinar la mitad $a[n/2]$ y sus lados $a[n/2 - 1]$ y $a[n/2 + 1]$ si $a[n/2]$ es el mínimo local detener y devolver el valor, sino buscar usando el mismo criterio en la mitad del lado cuyo valor sea el menor.

Explorando lo aprendido

- Analisis Asintotico
- Big O - Upper Bound
- Omega Ω - Lower Bound
- Theta Θ - Exact Bound
- Complejidad espacial



Bibliografía:



- **B. Stroustrup The c++ Programming Language**
4t edition
- **C++ Primer, Fifth Edition; 2013;** Stanley B. Lippman, Josée Lajoie, Barbara E. Moo