Eficiencia Algorítmica. Prueba de Software

Objetivos:

Que el alumno:

- Se familiarice con los conceptos de *Eficiencia de Algoritmos, Pruebas de Algoritmos.*
- Lleve a la práctica la elaboración de algoritmos más pequeños y más rápidos.
- Observe y practique el uso de los recursos (tiempo, memoria, etc.) en forma eficiente.
- Implemente soluciones aplicando estrategias algorítmicas.
- Se familiarice con la construcción de casos de pruebas de algoritmos.
- Conozca la importancia de probar el algoritmo con técnicas y estrategias para dotar de calidad al producto software.
- Continúe en el desarrollo de implementar las soluciones de problemas con un enfoque estructurado.

Metodología

- Lectura de la conceptualización de eficiencia algorítmica, prueba de algoritmos, técnicas de optimización y estrategias algorítmicas.
- El alumno deberá resolver individualmente los ejercicios propuestos
- Se podrá realizar trabajos en grupos para consolidar conceptos, comprensión de lo solicitado y alternativasde solución.
- El alumno deberá codificar las soluciones que proponga de cada uno de los ejercicios propuestos en lasclases prácticas de laboratorio.
- Interactuar en el aula virtual de la asignatura.

Duración

De acuerdo a la planificación de la asignatura, se deberán utilizar para la resolución de los ejercicios de esta serie, no más de dos (2) clases prácticas.

Ejercicios propuestos sobre práctica de Eficiencia Algorítmica y Prueba de Software

1. Calcule el grado de complejidad de los siguientes algoritmos.

```
Function sum(n: integer):integer;
var
j, Suma : integer;
Begin
{1}        Suma := 0;
{2}        for  j := 1 to n do
{3}           if suma<> 0 then
{4}                    Suma := Suma + j;
else
{5}                 Suma := Suma *j;
End;
```

2. Calcule el grado de complejidad de los siguientes algoritmos.

a.
```
void Calculo (double a, double b, double c) {
   double resultado;
   resultado = a + b + b*c + (a+b-c)/(a+b) + 4.0;
   cout << resultado << endl;
}
```

b.
```
float Suma (float arreglo[ ], int cantidad) {
      float suma= 0;
      for (int i = 0; i < cantidad; i++)
            suma += arreglo [i];
      return suma;
   }
```

3. Grafique lo siguientes ordenes de complejidad

2^n
n^3
n^2
n
log(n)
n(log(n)

4. Escriba de una manera más óptima cada uno de los siguientes trozos de algoritmos.

a) for (i = 1; i <= 2000; i++)
Y := Y + ((X*X*X) – i);

b) y:= 1/(2*x*t-1) + 1/(2*x*t-2) + 1/(2*x*t-3) + 1/(2*x*t-4)

5. Dado los siguientes códigos de algoritmo determine cual es el mas costoso y justifique su respuesta.

(1) Desde i = 1 hasta 10000
X = i(2)
Fin-desde

(2) Desde i = 1 hasta 10000
x = i * i
fin-desde

6. La siguiente tabla muestra el tiempo que tardaría un computador en realizar f(n) operaciones, para distintas funciones “f” y distintos valores de “n”. Determine cuantas operaciones por segundo realiza el ordenador.

f(n)	n	10	20	30
N		0.00001s	0.00002s	0.00003s
n log(n)		0.00003s	0.00008s	0.00014s
n(2)		0.0001s.	0.0004s	0.0009s
2(n)		0.001s	1.04s	17 min

(s-segundo; m-minuto)

7. Analizamos una búsqueda binaria en el siguiente vector ordenado.

1	2	3	4	5	6	7

Si buscamos el valor "7" cuantas comparaciones debemos efectuar?.

8. Determine la complejidad de los siguientes bucles con contador explicito

Bucle	for (int i= 0; i < K; i++)
	algo_de_O(1);
		`for (int i= 0; i < N; i++)`
		`  for (int j= 0; j < N; j++)`
		`  algo_de_O(1);`
			`for (int i= 0; i < N; i++)`
			`  for (int j= 0; j < i; j++)`
			`  algo_de_O(1);`

Complejidad			

9. Construya el grafo de flujos del siguiente código (prueba del camino básico):

```
{
  s1;
  while (c1) {
    if (c2) s2 else s3;
    s4;
  }
  if (c3) s5 else s6;
}
```

10. Dado el código en C para encontrar el máximo común divisor (Algoritmo de Euclides). Dibuje el grafo y determine los posibles caminos independientes.

```
A0    void euclid(int m, int n)
      {
        // Asumimos que ambos m y n son mayores que cero
        //Forzamos que m >= n  por eficiencia
        //Retorna el mcd

        int r;
A1      if (n > m) {
A2            r = m;
A3            m = n;
A4            n = r;
        }
A5      r = m % n;          /* m modulo n */
A6      while (r != 0) {
A7            m = n;
A8            n = r;
A9            r = m % n;    /* m modulo n */
        }
A10     return n;
      }
```

11. Dado el siguiente código:

```
Abrir archivos;
Leer archivo ventas, al final indicar no más registros;
Limpiar línea de impresión;
WHILE (haya registros ventas) DO
  Total nacional = 0;
  Total extranjero = 0;
    WHILE (haya reg. ventas) y (mismo producto)
      IF (nacional) THEN
        Sumar venta nacional a total nacional
      ELSE
        Sumar venta extranjero a total extranjero
      ENDIF;
      Leer archivo ventas, al final indicar no más registros;
    ENDWHILE;
    Escribir línea de listado;
    Limpiar área de impresión;
ENDWHILE;
Cerrar archivos.
```

Se pide:

a) Dibuje el grafo
b) Complejidad ciclomatica
c) Caminos independientes

Ejercicios Complementarios de Optimización, Eficiencia Algorítmica y Prueba

1. Dado un conjunto de tríos de valores A, B, C (mayores que cero) determinar, de entre los que forman triángulo, los distintos tipos (escaleno, isósceles, equilátero) y también cuál de ellos es recto. Informar de acuerdo al diseño de salida. La solución del problema deberá ser óptima (menos instrucciones, menos memoria, etc.).

LADO 1	LADO 2	LADO 3	TIPO	RECTO
--	--	--	Escaleno	SI
--	--	--	No triángulo	
--	--	--	Equilátero	NO
--	--	--	Escaleno	SI

2. Dada el resultado de una encuesta sobre la audiencia de programas de televisión (entrada: Ve el programa “A”: sí o no; ve el programa “B”: sí o no; ve el programa “C”: sí o no), se desea saber lo siguiente:

a) Cuántos ven solamente el programa A
b) Cuántos ven los tres programas
c) Cuántos no ven ningún programa
d) Cuántos NO ven A, pero si algún otro.

La solución del problema deberá ser óptima (el alumno deberá demostrar la optimización realizada).

3. Implemente la solución para calcular el mínimo, máximo y el medio de tres números al menos de dos maneras diferentes. Luego compare ambas soluciones (análisis del algoritmo) a partir de la unidad de trabajo que realiza cada una de ellas. Finalmente verifique que incidencia tiene los datos en ambas soluciones.

4. Proponga tres estructuras de módulos (ej. expresiones algebraicas) donde se verifique la importancia de la NO repetición de cálculos innecesarios.