UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
ESCUELA DE COMPUTACIÓN
ALGORITMOS Y ESTRUCTURAS DE DATOS

PRÁCTICA DE COMPLEJIDAD EN TIEMPO

1. Explique el concepto de dominancia de funciones (gráfica y formalmente). Explique cuál es su fin, es decir, ¿para qué es utilizada?
2. ¿Qué entiende usted por el término complejidad en tiempo?
3. Demuestre formalmente las siguientes propiedades del concepto de dominancia de funciones:
   - Si $f(n) = O(g(n))$ y $g(n) = O(h(n))$ entonces $f(n) = O(h(n))$
   - $f(n) = O(f(n))$
   - Si $f(n) = O(g(n))$ y $g(n) = O(f(n))$ entonces $f(n) = g(n)$
4. Si dos algoritmos tienen el mismo orden teórico en tiempo de ejecución, entonces ¿ambos tardarán el mismo tiempo en ejecutarse en la práctica? Suponga que ambos se ejecutan en plataformas totalmente iguales. Explique su respuesta
5. Discuta la veracidad de la siguiente afirmación: $n^2 = O(n\log_2 n)$. En caso de ser cierto, demuéstrelo formalmente. En caso de ser falso, indique dos formas distintas que permitan concluir que dicho razonamiento es erróneo.
6. Para las siguientes f(n), halle una cota g(n), y demuestre mediante el concepto de dominancia de funciones que f(n) = O(g(n))
   a) $f(n) = 8n^3 – 576n^2 + 832n – 248$
   b) $f(n) = \ln(an) + b$
   c) $f(n) = n^2 – \log_2 n + 7$
   d) $f(n) = an^2 + bn + c$
   e) $f(n) = n + n^2 + n^3$.
   f) $f(n) = a2^{bn} + c$
   g) $f(n) = 3^x + \log_2(n + 1)$
7. ¿Es cierto que el siguiente código:

```
if Cond then
   X = X + 1;
else
   X = X – 1;
end
```

   es de O(1)? (justifique en detalle su respuesta)
8. ¿Bajo qué condiciones un algoritmo es de O(1)?
9. Supongamos que el tiempo de ejecución de un módulo A es una constante M. Calcular el orden de magnitud de la función complejidad T(n) que mide el tiempo de ejecución de los siguientes algoritmos, donde n es el tamaño de la entrada (N en los algoritmos), y demuestre formalmente mediante el concepto de dominancia de funciones cual es su complejidad.
   a) Algoritmo 1:

```
Integer I = 1
do
   Integer J = 1
   do
      Integer K = 1
      do
         Modulo A;
         K = K + 1;
      while K ! = J;
      J = J + 1;
   while J ! = I;
   I = I + 1;
while I ! = N ;
```

   b) Algoritmo 2:

```
J = N ;
while J > 1 do
        A= A*2;
        J = J/2;
end
```

10. Analice el tiempo de ejecución T(n) de los siguientes fragmentos de programa y calcule su orden de complejidad:

```
Integer sum=0;
for  i = 0 to n do
  for  j = 0 to n*n do
       sum = sum + 1;
  end
end
```

```
Integer i = 1;
Integer x = 0;
while (i<=n)  do
     x = x + 3;
     i = i * 2;
end
```

```
Integer  x=0;
for  i = 0 to n do
  for  j = 0 to i*i do
    for  k = 0 to j do
         x = x + 2;
    end
  end
end
```

```
Integer sum=0;
for  i = 0 to n do
  for  j = 0 to i do
       sum = sum + 1;
  end
end
```

```
Integer i = 1;
Integer x = 0;
do
    Integer  j = 1;
    while ( j <= n)  do
      x = x + 1;
      j = j * 2;
    end;
   i = i + 1;
while (i!=n);
```

11. Suponga que cuenta con un arreglo desordenado de enteros. Se desea implementar una operación selectora, que dado un parámetro *k*, retorna el entero correspondiente a la posición *k* del arreglo si este estuviera ordenado (obviamente, sin ordenarlo). Defina un algoritmo que sea eficiente y determinístico, es decir, dado cierto *k* y cualquier arreglo, el tiempo de búsqueda sea el mismo. Calcule formalmente la complejidad en tiempo del programa.
12. Dado el siguiente algoritmo:

```
void  Alfa ( )
        Integer  N,X,Y,Z,M;
        X=0; Y=0; Z=0;
        Read (N);
        M = 2^N;
        for I=1 to M do
                Z = (4*I) + 8 + Z;
                Y = Y + (2 * I);
                X =  X + 1;
      end
end
```

a) Calcule la función T(n) del algoritmo e indique su orden de complejidad en tiempo
b) Reescriba este algoritmo para que su orden de complejidad sea de O(1). Recuerde que el nuevo algoritmo debe tener los mismos resultados que el original.

13. Para el siguiente algoritmo indique su complejidad en tiempo. Explique detalladamente que ocurre.

```
for i = 1 to n do
  Integer j,k1,k2;
    for j = 1 to n-1 do
       i = i + 1;
       j = j + 1;
    end

    for k1 = i to n do
        for k2 = k1 to n do
            k1 = k1 +1;
            k2 = k2 + 1;
        end
    end
end
```

14. Calcule la complejidad del siguiente algoritmo:

```
Integer j;
j = 1;
while j < N do
    j =  j * b;
end
```

Donde b es una constante, dado que:

a) N también es un valor constante
b) N es la entrada de datos del programa

15. Calcule la complejidad en tiempo para los siguientes fragmentos de código:

```
Integer  i=1;
Integer  x=0;
while ( i <= n ) do
   x = x + 1;
   i = i + 3;
end
```

```
Integer  i=1;
Integer  x=0;
while ( i <= n ) do
   x = x + 3;
   i = i + 3;
end
```

```
Integer  x = 0;
for i =1 to n do
   for j =1 to n do
      for k =1 to n do
         x = x + 2;
      end
   end
end
```

```
Integer  x=0;
for i =1 to n do
   for j =1 to i do
      for k =1 to j do
         x =  x +2;
      end
   end
end
```

```
for i =1 to n do
   Integer  j = 1;
   while( j <= i ) do
      j = j *2;
      x = x + 2;
   end
end
```

16. Carly quiere llegar a la casa de Sam, para ello Carly interpreta las calles de la ciudad como una matriz, en dicha matriz no existen obstáculos, pero no hay movimientos diagonales, tampoco existen los movimientos arriba e izquierda, solo abajo y derecha. Construya dos algoritmos que dada la posición de Carly y la posición de la casa de Sam, nos digan que si es posible llegar a casa de Sam y si es posible nos digan el mínimo número de pasos para llegar a dicha casa. Realice la solución obvia empleando backtracking de $O(2^n)$, y realice la solución O(1). Ejecute el algoritmo con una entrada relativamente grande y tome el tiempo. De esta manera se dará cuenta de cuan importante es realizar código eficiente.

GDAYED/2014