Ing. de Sistemas e Informática
Estructuras de Datos y Algoritmos
Jorge Mario Londoño P.

Ejercicios de repaso 2

Análisis de algoritmos

1. Obtener aproximaciones *tilde* para las siguientes expresiones. Indicar el orden de crecimiento de cada una.

a) $N+1/N^2$
b) $1+1/N$
c) $(1+1/N)(1+2/N)$
d) $2N^3-15N^2+N$
e) $lg(2N)/lg(N)$
f) $lg(N^2+1)/lg(N)$
g) $N^{100}/2^N$

2. Determinar la función T(N) que describe el tiempo requerido por los siguientes algoritmos en función de los tiempos requeridos por las operaciones elementales.

a. Obtener el máximo de un vector

```
int max=datos[0];
for(int i=1; i<datos.length; i++)
     if (datos[i]>max) max=datos[i];
return max;
```

b. Contar elementos repetidos en una lista

```
int count = 0;
for (Node x = first; x != null; x = x.next)
    for (Node y = x.next; y != null; y = y.next)
         if (x.item.equals(y.item))
            count++;
return count;
```

c. Chequear si una matriz es diagonal (Asumir matriz cuadrada NxN)

```
static boolean esMatrizDiagonal(double[][] a) {
        if (a.length!=a[0].length) return false;
        boolean esDiagonal = true;
        for(int i=0; i<a.length; i++)
            for(int j=0; j<a[0].length; j++)
                if (i!=j && a[i][j]!=0)
                    esDiagonal = false;
        return esDiagonal;
```

```
}
```

d. Multiplicar dos matrices (Considerar matrices cuadradas NxN. También se puede considerar el caso más general utilizando tres variables independientes: Primera matriz MxN y segunda matriz NxP).

```
public double[][] product(double[][] a, double[][] b) {
      double[][] c = new double[a.length][b[0].length];
      for(int i=0; i<a.length; i++)
            for(int j=0; j<b[0].length; j++) {
                  for(int k=0; k<a[0].length; k++) {
                        c[i][j] += a[i][k]*b[k][j];
                  }
            }
      return c;
}
```

3. Determinar el orden de crecimiento (en función de N) de los siguientes fragmentos de código. Seleccionar el modelo de costo representativo, estimar la frecuencia del modelo de costo y su orden de crecimiento.

a.

```
int sum=0;
for(int n=N; n>0; n/=2)
     for(int i=0; i<n; i++)
          sum++;
```

b.

```
int sum=0;
for(int i=1; i<N; i*=2)
     for(int j=0; j<i; j++)
          sum++;
```

c.

```
int sum=0;
for(int i=1; i<N; i*=2)
     for(int j=0; j<N; j++)
          sum++;
```

Estimación de espacio

4. Se declara un arreglo de objetos Fecha:

```
Fecha[] listaCumpleannos = new Fecha[N];
class Fecha {
      int anno;
      byte mes;
      byte dia;
}
```

a. Estimar el espacio mínimo requerido por el arreglo.

b. Estimar el espacio máximo requerido por el arreglo y todas las instancias de Fecha.

5. La siguiente es una implementación de un árbol ternario:

```
class Ternario<T> {
      Node raíz;
      class Node {
            T item;
            Node izquierdo;
            Node centro;
            Node derecho;
      }
}
```

Asumiendo que el programa utiliza un `Ternario` de N items, y que los objetos de tipo T miden K bytes:

a) Estimar el espacio requerido por un objeto `Node` (1 punto)

b) Estimar el espacio total requerido por la instancia de `Ternario`. (1 punto)

Respuestas seleccionadas

Análisis de Algoritmos

1
a) ~N
b) ~1
c) ~1
d) ~$2N^3$
e) ~1
f) ~2
g) la función tiende a 0 para N muy grande

3
a. Asumir $N=2^b$.
Ciclo externo se repite para n= 2^b, 2^{b-1}, ..., 2^0, un total de b+1 veces.
Ciclo interno se repite para i=0,...,n-1, para cada valor de n.
Frecuencia de la instrucción sum++ es
$2^b+2^{b-1}+ ...+ 2^0 = (2^{b+1}-1)/(2-1)$
b. El ciclo externo itera para los valores i=2^0,2^1,...,2^b, tales que $2^b<N$. El ciclo interno itera con j=0,...,i-1 para cada valor de la i.
c. (lgN+1)N