Estructuras de datos
Clase teórica 1

UNIVERSIDAD NACIONAL DE COLOMBIA SEDE MEDELLÍN

Contenido

- Eficiencia algorítmica
- Notación "Big Oh"

Material elaborado por: Julián Moreno

Facultad de Minas, Departamento de Ciencias de la Computación y la Decisión

¿Qué es un algoritmo?

Un algoritmo es un conjunto ordenado de instrucciones bien definidas, no-ambiguas y finitas que permite resolver un determinado problema computacional.

Un corolario de esta definición es que un determinado problema computacional puede ser resuelto por diversos (infinitos) algoritmos.

Esta característica es la motivación de este curso: diseñar buenos (si no los mejores) algoritmos, usando las estructuras de datos adecuadas, para resolver problemas computacionales.

¿Cómo se mide un algoritmo?

Una manera objetiva de determinar que tan "bueno" es un algoritmo es por medio del número de operaciones básicas que este debe realizar para resolver un problema cuya entrada tiene un tamaño *n*. Es decir, calcular un *f(n)*

Ejemplo: Diseñar un algoritmo para determinar la suma de los números enteros positivos hasta el *x* ($1 \leq x \leq 1000$)

¿Cómo se mide un algoritmo?

Alternativa 1:

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        int x, i;
        long r;
        r = 0;
        Scanner entrada = new Scanner(System.in);
        x = entrada.nextInt();
        for (i=1; i<=x; i++){
            r += i;
        }
        System.out.println(r);
    }

}
```

Cantidad de operaciones básicas: f = 7+3x

¿Cómo se mide un algoritmo?

Alternativa 2:

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        int x;
        long r;
        Scanner entrada = new Scanner(System.in);
        x = entrada.nextInt();
        r = x*(x+1)/2;
        System.out.println(r);
    }

}
```

Cantidad de operaciones básicas: f = 6

Notación "Big Oh"

En resumen consiste en:

- Considerar el peor escenario
- Realizar un análisis asintótico, es decir, enfocarse en valores grandes de n
- Lo anterior implica no prestar atención a términos constantes o de orden menor

En el ejemplo anterior, tendríamos que:

Alternativa 1: $f(n) = 7+3n$, por tanto $O(n)$

Alternativa 2: $f(n) = 6$, por tanto $O(1)$

Utilidad de la notación “Big Oh”

Aunque esta notación no es exacta, si es un buen descriptor del comportamiento de un algoritmo a medida que el tamaño de la entrada crece.

Podemos decir que un algoritmo es mejor (más eficiente) que otro si su “*O*” es menor. En otras palabras si su tiempo de ejecución (determinado por la cantidad de operaciones básicas), considerando el peor escenario, crece más lentamente a medida que se aumenta el tamaño de la entrada.

Comparación de eficiencias

Habiendo comprendido el concepto de la notación Big O, ¿cuál es el orden de complejidad o eficiencia del algoritmo de ordenamiento *bubbleSort*?, ¿por qué?

```
static void bubbleSort(float A[]){
    int i, j;
    float aux;
    for (i = A.length-1; i > 0; i--){
        for (j = 0; j < i; j++){
            if (A[j] > A[j+1]){
                aux = A[j];
                A[j] = A[j+1];
                A[j+1]= aux;
            }
        }
    }
}
```

f(n) = 3 + 12n(n-1)/2

Por tanto **O(n^2)**

Comparación de eficiencias

¿Será que esto es lo mejor que se puede hacer para ordenar? Es decir, ¿no hay un algoritmo con menor ‘O’? ¿Qué tal selectSort o insertSort?

Resulta que sí, y uno de los más conocidos es el mergeSort, cuya eficiencia es *O(n.log(n))*

Comparación de eficiencias

Supongamos que estamos trabajando en un computador con procesador de un solo núcleo a 3,2Ghz lo que nos da un aproximado de 1,000,000,000 operaciones por segundo. En la siguiente tabla vamos a relacionar el tamaño de un problema determinado con lo que demorarían en resolverlo una serie de algoritmos con eficiencias diferentes.

	log(n)	n	n.log(n)	n^2	n^3	2^n
10	= 3 nanosegs	10 nanosegs	= 33 nanosegs	100 nanosegs	1 microseg	= 1 microseg
100	= 7 nanosegs	100 nanosegs	= 664 nanosegs	10 microsegs	1 milliseg	= 4E+10 milenios
1000	= 10 nanosegs	1 microseg	= 10 microsegs	1 milliseg	1 seg	
10.000	= 13 nanosegs	10 microsegs	= 133 microsegs	100 milisegs	17 minutos	
100.000	= 17 nanosegs	100 microsegs	= 2 milisegs	10 segs	= 12 dias	
1'000.000	= 20 nanosegs	1 milliseg	= 20 milisegs	= 17 minutos	= 32 años	
1E+9 (mil millones)	= 30 nanosegs	1 seg	= 30 segs	= 32 años		
1E+12 (un billón)	= 40 nanosegs	= 17 minutos	= 11 horas			
1E+15 (mil billones)	= 50 nanosegs	= 12 dias	= 1 año y medio			
1E+18 (un trillón)	= 60 nanosegs	= 31 años	= 19 siglos			
1E+21 (mil trillones)	= 70 nanosegs	= 317 siglos				
1E+24 (un cuatrillón)	= 80 nanosegs					
1E+27 (mil cuatrillones)	= 90 nanosegs					