Algoritmia(notas del profesor)

Estos apuntes son notas de clase realizadas por el profesor.

El lector ha de complementarlas con las sesiones impartidas de manera presencial

Dra. MPuerto Paule Ruiz

paule@uniovi.es

Potencia en base 2 (diferentes versiones)

- Versión 0
  - Caso trivial: exponente==0 devuelve 1
  - Caso general: return (2 * recursivePow(exponente-1));
- Versión 1
  - Caso trivial: exponente==0 devuelve 1
  - Caso general: return (recursivePow1(exponente-1) + recursivePow1 (exponente-1));
- Versión 2
  - Caso trivial: exponente==0 devuelve 1
  - Caso general:

    if (exponent % 2 == 0) return (recursivePow2(exponent/2) * recursivePow2 (exponent/2));
    else return (recursivePow2(exponent/2) * recursivePow2 (exponent/2) * 2);
- Versión 3 (memoria auxiliar)
  - Caso trivial: exponente==0 devuelve 1
  - Caso general:

```
long result = recursivePow3 (exponent/2);
if (exponent % 2 == 0) return (result * result);
else                   return (result * result * 2);
```

Pruebas unitarias simples que comprueben el buen funcionamiento de los algoritmos

Bucles

- Pasamos por parámetro el número de iteraciones
  - Un bucle for
  - Dos bucles for anidados
  - Tres bucles for anidados
  - Bucle logarítmico

Mediciones de tiempos de los algoritmos (TestBench)

- Cronómetro-

```
public static void test(int times, int startN, int endN,String className,String methodName){

for(int workLoad = startN ; workLoad < endN ; workLoad++){
  long startTime = System.currentTimeMillis();
  for(int time = 0 ; time < times ; time++)
       testAlgorithm(className, methodName, workLoad);
  long finalTime = System.currentTimeMillis();
System.out.println("Carga"+ workLoad + ", "+" Tiempo: "+ ((finalTime - startTime) / times));

}
```

Llamada test (Clase Main)

TestBench.test(5, 1, 5, "ed.unidad1.Algoritmos", “linear");
TestBench.test(5, 1, 5, "ed.unidad1.Algoritmos", "quadratic");
TestBench.test(5, 1, 5, "ed.unidad1.Algoritmos", "cubic");
TestBench.test(5, 1, 5, "ed.unidad1.Algoritmos", "logarithmic");
TestBench.test(5, 1, 5, "ed.unidad1.Algoritmos", "logarithmic");
TestBench.test(5, 1, 5, "ed.unidad1.Algoritmos", "recursivePow");
TestBench.test(5, 1, 5, "ed.unidad1.Algoritmos", "recursivePow1");
TestBench.test(5, 1, 5, "ed.unidad1.Algoritmos", "recursivePow2");
TestBench.test(5, 1, 5, "ed.unidad1.Algoritmos", "recursivePow3");

Tiempos de ejecución →0

Método doNothing

- doNothing: dormimos el hilo de ejecución en el método 250 ms
  - Conseguimos simular que los algoritmos “realizan un trabajo”
- Incluir retardo en los métodos

Tiempos de ejecución más grandes

Análisis de las complejidades (teórico)

- linear→ o(n)
- quadratic→o(n^2)
- cubic→o(n^3)
- logarithmic→ o($\log_2 n$)
- recursivePow→ o(n) (n llamadas recursivas)

Escribir tiempos de ejecución en un fichero CSV

- Objetivo: Obtener gráficas que demuestren los tiempos de ejecución teóricos
- Completar método test incluyendo la escritura en un fichero CVS (separador “;”)

public static void test(String output, int times, int startN, int endN,String className,String methodName) throws IOException

TestBench.test("linear.csv", 5, 1, 10, "ed.unidad1.Algoritmos", "linear");