Programación Avanzada Trabajo Práctico N° 2 Complejidad Computacional - Programación dinámica

DIIT - Universidad Nacional de La Matanza

1. Introducción

A través del presente trabajo se espera que los alumnos codifiquen, evalúen y comparen distintos algoritmos que resuelven, mediante diferentes técnicas, el mismo problema.

2. Objetivos

2.1. Polinomios

Dise˜nar un programa para evaluar un polinomio P(x) de grado n.

```
public class Polinomio {
    private int grado ;
    private double [] coeficientes;

     /*La posición 0 del arreglo de coeficientes contiene el coeficiente de grado n y la
            posición n contiene al término independiente.*/

    public Polinomio {

       double evaluarMSucesivas (double x) { ... }

       double evaluarRecursiva (double x) { ... }

       double evaluarRecursivaPar (double x) { ... }

       double evaluarProgDinamica (double x) { ... }

       double evaluarMejorada (double x) { ... }

       double evaluarPow(double x) { ... }

       double evaluarHorner( double x) { ... }

       }

   }
```

1. Escribir evaluarMSucesivas utilizando cálculo de potencia por multiplicaciones sucesivas

2. Escribir evaluarRecursiva utilizando el siguiente cálculo de potencia recursiva:

a) Sin considerar si el exponente es par o impar:

$$potencia(x,n)=x* potencia(x,n-1)$$

b) Considerando si el exponente es par o impar:

Si n es par:

$$potencia(x,n)=potencia(x*x,n/2)$$

Si n es impar

$$potencia(x,n)=x* potencia(x,n-1)$$

3. Escribir evaluarProgDinamica, almacenando las potencias de X ya calculadas.

4. Escribir evaluarMejorada, con un algoritmo de igual complejidad computacional que el anterior, pero que ejecute en un tiempo menor.

5. Escribir evaluarPow, valiéndose del método Math.pow(x,n) provisto por el lenguaje Java. Se debe incluir dentro de alguno de los métodos anteriores donde se considere que es apropiado. Investigue la CC de Math.pow.
6. Escribir evaluarHorner, aplicando el algoritmo de Horner de análisis numérico. (Investigar)

2.2. Binomio de Newton

Dado un binomio de la forma $(ax + b)^n$, conocido como elinomio de Newton, se desea:
1. Obtener el coeficiente del término k del desarrollo de dicho binomio. Tener en cuenta que el coeficiente k (el que corresponde al término de x^k) se calcula haciendo $\binom{n}{k} a^k b^{n-k}$.
2. Desarrolle también, un método que permita obtener todos los coeficientes del polinomio, o sea, el desarrollo completo del binomio, y aplique los métodos vistos en el punto anterior para evaluarlo en algún valor de x.
Desarrolle la class BinomioDeNewton, que permita cumplir con las consignas anteriores.
Nota: Investigar distintas técnicas para obtener los coeficientes del desarrollo del binomio. Estas serán recursivas y no recursivas. Usar técnicas de programación dinámica con y sin memorización. En el cálculo de las potencias se podrán reutilizar los métodos ya implementados.

3. Análisis de complejidad computacional

Indique la función de complejidad computacional asociada a cada uno de los métodos implementados.

4. Gráficos y tablas de rendimiento comparativo

Compare el tiempo de ejecución de todos los métodos implementados en los puntos 2.1 y 2.2
Genere todos los casos que considere necesarios para realizar el análisis.

5. Conclusiones

A partir de los análisis comparativos extraiga conclusiones.