UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE CIENCIAS
DEPARTAMENTO DE MATEMÁTICA Y CIENCIA DE LA COMPUTACIÓN
LICENCIATURA EN CIENCIA DE LA COMPUTACIÓN
Profesor(es): Nicolas Thériault
Segundo Semestre de 2019

Programación Avanzada – Laboratorio 1

1. Objetivos

Objetivo principal: Implementar algoritmos básicos de aritmética de polinomios: sumas, restas y multiplicación de dos polinomios.

Objetivos segundarios:

- Implementar algoritmos clásicos de sumas, restas y multiplicación de polinomios de forma estable.
- Analizar y aplicar un algoritmo del tipo dividir-y-conquistar, especificamente el algoritmo de Karatsuba para la multiplicación de polinomios.
- Obtener una implementación eficaz de la multiplicación de polinomios aplicable para polinomios de cualquier grados, optimizada para la multiplicación de dos polinomios del mismo grado.

El laboratorio se trabajará en grupos de 2 o 3 alumnos, entregando un resultado (informe y programas) por grupo.

2. Programación

La programación debe ser en C (no en C++), utilizando a lo más las librerías estándares siguientes:

- stdio.h
- stdlib.h
- math.h
- time.h

Además, para evitar cualquier problemas de compatibilidad, se recomienda utilizar solamente la función *clock()* para la medición de tiempo.

Para la entrega final, se trabajará con polinomios con coeficientes de tipo “double” (64 bites). Como la arimética con puntos flotantes conlleva un riesgo de errores de redondeo, se recomienda desarrollar y testear los algoritmos utilizando polinomios con coeficientes de tipo “long” donde no hay problemas de redondeo.

La estructura de datos utilizada debe permitir la creación de polinomios de cualquier grado (utilizando hasta 1 GB de RAM para el trabajo).

Crear funciones para:

1. Recibir un polinomio entrado manualmente [opcional];
2. Generación de polinomios aleatorios (de grado dado);
3. Escribir/imprimir un polinomio;
4. Crear una copia de un polinomio dado;
5. Sumar dos polinomios;
6. Restar Polinomios (calcular la diferencia entre dos polinomios);
7. Multiplicar polinomios con la formula clásica, guardando la tabla completa de productos de coeficientes;
8. Multiplicar polinomios con la formula clásica, reduciendo la utilización de memoria via una estrategia reducir-y-conquistar en una de las entradas;
9. Comparar los resultados de dos técnicas de multiplicación (comprobando que la diferencia de los resultados es 0).
10. Seleccionar/copiar parte de un polinomio (desde un grado mínimo hasta un grado máximo);
11. Multiplicar un polinomio por una potencia de x, o sumar dos polinomios multilplicando una de las entradas por una potencia de x [se puede programar solamente una de las dos funciones o programar ambas];
12. Multiplicar polinomios con 1 paso de inducción para el algoritmo dividir-y-conquistar clásico (con 4 multiplicaciones de tamaño $d/2$);
13. Multiplicar polinomios con 1 paso de multiplicación de Karatsuba (dividir-y-conquistar);
14. Multiplicación de polinomios de cualquier grado, optimizada para la multiplicación de dos polinomios del mismo grado.

Además, se debe utilizar una de las funciones de multiplicación clásica (sin inducción) y la función de multiplicación con un paso de multiplicación de Karatsuba para determinar el grado mínimo a partir de cual es conveniente utilizar la estrategia dividir-y-conquistar de Karatsuba, lo que se utilizará en la función de multiplicación optimizada.

Generación aleatoria de polinomios: para la versión final, los coeficientes deberían ser eligidos como enteros una distribución cerca de uniforme (por ejemplo básada en la función *rand()*) con valores entre -2^{62} y 2^{62}, y luego convertidos a double. No olvidarse de iniciar el generador aleatorio en la versión final.

3. Se solicita

1. Programar los diferentes algoritmos en C, asegurando que hacen un uso apropiado de memoría (por ejemplo que se liberan corectamente todos los espacios de memoria utilizados al terminar las funciones) y liberando la memoria **antes** de cerrar las funciones.

2. La selección de los distintos algoritmos/funciones se debe hacer a través de un menu (selección de opciones) que incluye el cierre del programa.

3. El programa final debe permitir verificar que todos los algoritmos de multiplicación devuelven el mismo resultado (i.e. tienen diferencias iguales a 0) y evaluar los tiempos de trabajo para el algoritmo de multiplicación final.

4. Determinar los límites práctico de las diferentes funciones de multiplicación, en particular los grados máximos de entrada que las funciones pueden manipular, y si posible justificar la razón, por ejemplo si hay límites debidas a la generación de las estructuras de dados utilizadas (arreglos, etc), o por el tiempo de trabajo.

5. Si no está indicado en la programación (con el indicador *const*), documentar cuales parámetros de las funciones no pueden ser modificados para las funciones. Además, documentar cuando algunos parámetros pueden ser repetidos en el mismo llamado de la función.

6. Analizar experimentalmente los tiempos de cálculo obtenidos, comparando:

   - La multipicación directa (clásica) con el algoritmo utilizando Karatsuba (para grados pequeño y medianos, típicamente hasta grado 2^{12}).
   - El crecimiento a grande escala del algoritmo utilizando Karatsuba con su crecimiento teórico (idealmente hasta grado 2^{24} o más).

   Observaciones: Si es conveniente, utilizar escalas logarítmicas para las gráficas. El tiempo puede ser como proporción de una operación especifica (valor base) en vez de segundos. Para tiempos menores a 1 segundo, se recomienda utilizar el promedio de tiempo sobre varios cálculos similares.

7. El informe debería detallar:

   - La estructura de datos utilizada para los polinomios (con las razones generales de porque se eligió).
   - El formato utilizado en los polinomios (puede tener coeficiente principal 0 o no, etc).
   - Las estrategias utilizadas en la funciones 7, 8, 12, 13 y 14 (multiplicación de polinomios sin producir una tabla de todos los productos de coeficientes).
   - Como se hicieron las mediciones de tiempo (como se seleccionaron de los grados medidos, sobre cuantos cálculos se evaluó, en que computador/procesador, etc).
   - Conclusiones obtenida de los análisis de tiempos.

4. Evaluación

La nota del laboratorio se calculará según la ponderación siguiente:

- Algoritmos [**25 %**]:
  El informe detalla las eleciones que se utilizaron durante la implementación y la determinación del grado mínimo para aplicar la multiplicación de Karatsuba.

- Análisis [**15 %**]:
  El análisis de los tiempos experimentales es correcto y completo, y justifica las conclusiones obtenidas.

- Informe [**10 %**]:
  El informe está escrito en lenguaje apropiado, sin faltas de ortografía o gramática..

- Implementación [**20 %**]
  El programa está escrito de forma que puede ser leído y/o re-utilizado fácilmente por otros programadores: la redacción es limpia (con espacios y divisiones claras) y bien documentada, las sub-funciones y las variables tienen nombres naturales (que indican a que sirven) o van acompañadas de comentarios aclarando a que sirven.
  Todas las versiones (algoritmos) de la multiplicación dan el resultado correcto.

- Eficiencia [**30 %**]
  Los tiempos de calculo de la multiplicación iterativa completa comparados con una implementación "básica" hecha por el profesor que satisface lo solicitado. Rangos de notas:

  - 1: no incluye una multiplicación iterativa completa.
  - 1.1–2.4: más de 20 veces más lenta
  - 2.5–3.9: 5 a 20 veces más lenta
  - 4–5.4: 2 a 5 veces más lenta
  - 5.5–6.9: a lo más 2 veces más lenta
  - 7: tiempos parecidos o mejores