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
Primer Semestre de 2020

Departamento de Matemática y Ciencia de la Computación – USACH – 2020

Programación Avanzada – Trabajo 1

1. Objetivos

Objetivo principal: Implementar algoritmos para resolver sistemas lineales de gran tamaño.

Objetivos secundarios:

- Implementar algoritmos clásicos de eliminación gausiana (triangular superior o completa).
- Resolver sistemas lineales a través del cálculo de la matriz inversa.
- Comparar los diferentes algoritmos del punto de vista de la complejidad.
- Obtener una implementación eficaz aplicables a sistemas lineales grandes.
- Utilizar experimentos y el estudio de complejidad para extrapolar el comportamiento a grande escala del programa obtenido.

Se trabajará en grupos de 2 o 3 estudiantes, entregando las respuestas (con desarrollo) de la parte teórica en formato pdf o jpg, y el programa (parte práctica) en un archivo .c.

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.

El álgebra lineal se trabajará módulo $p = 2^{31} - 1$ (aritmética exacta) para evitar temas de redondeo. Las funciones de aritmética básica (suma, resta, productos y inverso modular) están disponibles en el archivo "formato_lab1.c" (disponible en la página web del curso).

Por razones técnicas (incompatibilidad del generador aleatorio sencillo *rand* con el módulo p utilizado), se debe utilizar el generador aleatorio *arc4random* al momento de producir elementos al azar.

Observación: trabajando módulo $p = 2^{31} - 1$, la probabilidad que una matriz cuadrada $n \times n$ con entradas aleatorias no sea invertible es $\approx 2^{-30}$.

3. Requerimientos

El programa deberá ser capaz de trabajar con sistemas lineales de gran tamaño:

- Debe poder trabajar sistemas con un mínimo de 10^6 entradas (por ejemplo, con 1000 filas y 1000 columnas).
- Puede utilizar las funciones *malloc* y/o *calloc* para asignar arreglos grandes, pero estos no deberán ser de más de 10^5 espacios de memoria consecutivos.

Como mínimo, crear funciones para:

1. Generar un sistema lineal al azar del tamaño entregado por el usuario / la usuaria;
2. Leer archivos conteniendo un sistema lineal, donde la primera entrada es el número n de filas (int), la segunda entrada es el número m de columnas (int), luego de cuales las nm entradas de la matriz (aumentada) se entregan por filas, separados por un espacio o un cambio de linea;
3. Resolver el sistema lineal por eliminación Gaussiana completa;
4. Resolver el sistema por eliminación triangular superior y substitución reversa;
5. Calcular el inverso de una matriz cuadrada y utilizar este para resolver el sistema asociado.

Se podrán definir otras funciones (funciones auxiliares, etc) si deseado.
Además se recomienda tener funciones para escribir el sistema y/o su solución cuando el tamaño del sistema es pequeño (hasta 6×6).

El programa debe utilizar un sistema de menú para el usurio / la usuaria, y debe liberar la memoria antes de cerrar el proceso. El menú debe permitir medir los tiempos de trabajos para sistemas grandes (sin mostrar el resultado del sistema). El trabajo para generar un sistema aleatorio o leer el sistema desde un archivo no se debe incluir en este tiempo.

4. Estudio teórico

Contestar las siguientes preguntas. Se evaluará el desarrollo (justificación) de sus respuestas.

1. [**10 puntos**] Describir la(s) estructura(s) de datos utilizada(s) y justificarla(s).
2. [**30 puntos**] Estudiar la complejidad de los algoritmos programados:
   Indicación: Dado que la probabilidad que una entrada sea 0 "por accidente" es de $1/p$, pueden suponer que el algoritmo no encuentra ninguna entrada 0 a menos que sea producido directamente por la eliminación Gaussiana (eliminación en la columno del pivote).

a) Determinar la complejidad de la eliminación Gaussiana completa lo más exactamente posible (en término de las distintas operaciones mód p).

b) Determinar la complejidad de la eliminación triangular superior lo más exactamente posible, incluyendo el costo de la substitución reversa.

c) Lo más exactamente posible, determinar la complejidad del calculo de la matriz inversa con su algoritmo, y la complejidad de resolver el sistema por multiplicación con la matriz inversa.

d) Asintoticamente, ¿Cómo se comparan las trés técnicas? y ¿Cuál recomendarían para resolver un (único) sistema?.

e) ¿Cuantás veces se debe tener sistemas con la misma matriz para que el calculo de la matriz inversa se vuelva ventajoso?

f) Para cada una de las tres técnicas, obtener su forma asintotica de tipo $\Theta(n^d)$, dando valores de las constantes c_1, c_2 y N_0 asociadas.

3. [**20 puntos**] Utilizando experimentos elegidos cuidadosamente, justificar las respuestas a lo siguiente:

   a) ¿Cuál es el tamaño n del sistema cuadrado ($n \times n$) lo más grande que podrían resolver utilizando el RAM de su computador?

   b) ¿Cuánto tiempo tomaría resolver este sistema?

   c) ¿Cómo eligieron los ejemplos utilizados?

4. [**10 puntos extra**] Pregunta abierta:

   a) Si tenían que resolver un sistema cuadrado con n tres veces más grande que el tamaño máximo en la pregunta anterior, ¿Qué tipo de ajustes podrían hacer en el algoritmo para disminuir el efecto de trabajar con la memoria swap?

   Observación: Si quieren una indicación del efecto de utilizar la memoria swap, pueden correr una versión corta de la eliminación Gaussiana (con los primeros 10 pivotes) para valores de n menores y mayores que el límite. (Se recomienda re-iniciar el computador después del experimento).

5. Evaluación de la programación

La nota del laboratorio se calculará según la ponderación siguiente:

- Redacción [**10 %**]
  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.

- Ejecución [**15 %**]
  El programa compila sin advertencias. El ejecutable devuelve resultados correctos.
  Si el compilador no produce un ejecutable (por errores de compilación), la nota será 1.

- Eficiencia [**75 %**]
  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 para la eficiencia:

  - 1: no produce un ejecutable o no devuelve resultados correctos
  - 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