Facultad de Ingenieria, Industria y Tecnologia
Estructura de Datos

EXAMEN PARCIAL PRACTICO

Algoritmos Iterativos vs. Recursivos

Analisis de Complejidad Temporal y Notacion Big-O

Catedratico:	Ing. Brandon Chitay
Asignatura:	Estructura de Datos
Facultad:	Ingenieria, Industria y Tecnologia
Tipo de evaluacion:	Examen Parcial Practico
Valor total:	100 puntos
Modalidad:	Individual
Lenguaje:	Java (version 11 o superior)
Tiempo de entrega:	1 semana a partir de la fecha de asignacion

Objetivo General

El estudiante implementara en Java un conjunto de algoritmos en sus versiones iterativa y recursiva, medira sus tiempos de ejecucion con distintos volumenes de datos, registrara los resultados en Excel y generara graficas de tendencia que le permitan identificar y justificar la notacion O grande (Big-O) de cada algoritmo en ambas versiones.

Competencias a Evaluar

- Implementar algoritmos iterativos y recursivos correctamente en Java.
- Aplicar la medicion de tiempo de ejecucion con System.nanoTime().
- Analizar graficas de rendimiento generadas en Excel.
- Identificar la complejidad algoritmica (Big-O) a partir de datos experimentales.
- Documentar y presentar resultados de forma clara y estructurada.

Algoritmos a Implementar

Implementa cada uno de los siguientes 4 algoritmos en sus versiones ITERATIVA y RECURSIVA (8 implementaciones en total). La columna 'Big-O esperado' es una guia: tu trabajo es CONFIRMAR esa notacion experimentalmente con las graficas.

#	Algoritmo	Descripcion	Iterativo	Recursivo
A1	**Factorial**	Calcula n!. Caso base: 0! = 1. Limita n a 20 maximo.	O(n)	O(n)
A2	**Serie de Fibonacci**	Calcula el n-esimo termino. Limita n a 30 en la version recursiva.	O(n)	O(2^n)
A3	**Busqueda Lineal**	Busca un valor en un arreglo. Retorna el indice o -1 si no existe.	O(n)	O(n)
A4	**Ordenamiento Burbuja**	Ordena un arreglo de n enteros de menor a mayor usando el metodo burbuja.	O(n^2)	O(n^2)

Por que Fibonacci recursivo es O(2^n)? Porque cada llamada genera DOS llamadas adicionales, y el arbol de recursion crece de forma exponencial. Esto lo observaras claramente en la grafica.

Requisitos Tecnicos

3.1 Estructura del Proyecto Java

Organiza tu proyecto de la siguiente manera:

```
ProyectoED_[TuNombre]/
  src/
    algorithms/
      Factorial.java
      Fibonacci.java
      BusquedaLineal.java
      OrdenamientoBurbuja.java
    benchmark/
      Medidor.java        // clase utilitaria para medir tiempos
      Main.java           // punto de entrada: ejecuta todas las pruebas
    utils/
      GeneradorDatos.java // genera arreglos de distintos tamanos
  resultados/
    tiempos.csv          // exportacion para abrir en Excel
```

3.2 Medicion de Tiempo

Usa este patron exacto en Java para medir cada algoritmo:

```
long inicio     = System.nanoTime();
// --- llamada al algoritmo ---
long fin        = System.nanoTime();
long durNanos   = fin - inicio;
double durMs    = durNanos / 1_000_000.0;
```

Reglas:

- Ejecuta cada prueba 5 veces y registra el PROMEDIO para reducir el efecto del garbage collector.
- Usa System.nanoTime(), NO System.currentTimeMillis() (mayor precision).
- No incluyas la inicializacion de datos dentro del bloque medido; solo mide el algoritmo puro.
- Para Fibonacci y Factorial limita n a 30 en recursivo para no esperar tiempos exagerados.

3.3 Tamanos de Datos

Prueba cada algoritmo con los siguientes valores de n:

n = 5	n = 10	n = 15	n = 20	n = 25	n = 30
Muy pequeno	Pequeno	Pequeno-med	Mediano	Grande	Muy grande

Para Busqueda Lineal y Burbuja usa tamanos mayores: n = 100, 500, 1000, 5000, 10000. Esto hace que las diferencias sean mas visibles en las graficas.

Analisis en Excel

4.1 Estructura del Archivo Excel

Crea un archivo tiempos_rendimiento.xlsx con 4 hojas:

- Hoja 1 - Datos Crudos: tabla con columnas Algoritmo | Version | n | Tiempo_ms.
- Hoja 2 - Comparativa: una tabla por algoritmo mostrando iterativo vs recursivo lado a lado.
- Hoja 3 - Graficas: todas las graficas de tendencia.
- Hoja 4 - Conclusiones: tabla de Big-O experimental vs teorico.

4.2 Graficas Requeridas (una por algoritmo)

Para cada uno de los 4 algoritmos genera una grafica de dispersion (XY) con lineas y marcadores:

- Eje X: tamano de entrada (n).
- Eje Y: tiempo de ejecucion en milisegundos (ms).
- Dos series: version iterativa y recursiva, con colores distintos y leyenda.
- Agrega una linea de tendencia (trendline) a cada serie.
- Incluye titulo descriptivo y etiquetas de ejes.

4.3 Tabla de Conclusiones en Excel

Algoritmo	Version	Big-O medido	Big-O teorico	Coincide?
A1 - Factorial	Iterativo	(escribe tu resultado)	O(n)	___
	Recursivo	(escribe tu resultado)	O(n)	___
A2 - Fibonacci	Iterativo	(escribe tu resultado)	O(n)	___
	Recursivo	(escribe tu resultado)	O(2^n)	___
A3 - Busqueda Lineal	Iterativo	(escribe tu resultado)	O(n)	___
	Recursivo	(escribe tu resultado)	O(n)	___
A4 - Burbuja	Iterativo	(escribe tu resultado)	O(n^2)	___
	Recursivo	(escribe tu resultado)	O(n^2)	___

Entregables

Son 3 entregables. TODOS son obligatorios para obtener la nota completa.

Entregable 1 — Repositorio de Codigo en GitHub

- Crea un repositorio PUBLICO con el nombre: ED_Parcial_[TuNombre].
- Incluye todo el codigo fuente Java organizado segun la estructura de la seccion 3.1.
- Agrega un README.md con: tu nombre, descripcion del proyecto, instrucciones para compilar y ejecutar, y enlace al video.
- Realiza al menos 3 commits distintos durante el desarrollo.
- El codigo debe compilar y ejecutar correctamente al clonar el repositorio.

Entregable 2 — PDF de Documentacion Tecnica

Estructura obligatoria del PDF (en este orden):

1. Portada: nombre, carne, asignatura, catedratico y fecha.
2. Introduccion: breve explicacion de que es la notacion Big-O y para que sirve.
3. Descripcion de algoritmos: explica con tus propias palabras como funciona cada uno.
4. Fragmentos de codigo: las partes mas relevantes (no copies TODO el codigo).
5. Tabla de tiempos medidos: resultados por algoritmo y tamano de datos.
6. Graficas: capturas de las graficas de Excel con sus lineas de tendencia.
7. Analisis y conclusiones: comparacion iterativo vs recursivo y justificacion de Big-O.
8. Referencias: libros, sitios web y documentacion consultada.

Entregable 3 — Video de Demostracion

Graba un video de 5 a 10 minutos donde muestres:

- Tu entorno de desarrollo (IDE, Java instalado, version).
- Compilacion y ejecucion del proyecto (sin errores).
- Explicacion verbal de al menos 2 algoritmos (1 iterativo y 1 recursivo).
- Recorrido por el archivo Excel: graficas y tabla de conclusiones.
- Sube el video a YouTube (puede ser Privado, pero con enlace compartible).
- Incluye el enlace en tu README.md y en la portada del PDF.

Rubrica de Evaluacion

Criterio	Pts	Descripcion
1. Implementacion de algoritmos	40	8 algoritmos x 5 pts. Compila, es correcto y produce resultados validos.
A1 Factorial iterativo	**5**	Implementacion correcta usando ciclo.
A1 Factorial recursivo	**5**	Caso base correcto y llamada recursiva bien definida.
A2 Fibonacci iterativo	**5**	Calcula el n-esimo termino con ciclo, sin repetir calculos.
A2 Fibonacci recursivo	**5**	Calcula el n-esimo termino recursivamente (sin memoizacion).
A3 Busqueda Lineal iterativa	**5**	Recorre el arreglo secuencialmente y retorna el indice correcto.
A3 Busqueda Lineal recursiva	**5**	Busca recursivamente reduciendo el subproblema en cada llamada.
A4 Burbuja iterativo	**5**	Ordena correctamente con ciclos anidados.
A4 Burbuja recursivo	**5**	Version recursiva del burbuja que ordena correctamente.
2. Medicion de tiempos	**15**	Usa System.nanoTime(), promedia 5 ejecuciones y no incluye inicializacion dentro de la medicion.
3. Excel y graficas	**20**	Archivo con 4 hojas, 4 graficas con lineas de tendencia, ejes etiquetados y tabla de conclusiones Big-O.
4. PDF de documentacion	**15**	Documento claro con todas las secciones, graficas incluidas y analisis propio justificado.
5. Video de demostracion	**5**	Video en YouTube (5-10 min) que cubre todos los puntos solicitados.
6. Repositorio GitHub	**5**	Repo publico, nombre correcto, estructura organizada, README con enlace al video, minimo 3 commits.
TOTAL	**100 pts**	

Restricciones y Politica de Integridad Academica

- El trabajo es INDIVIDUAL. Queda prohibido compartir codigo con otros estudiantes.
- Puedes consultar libros, tutoriales y documentacion oficial; debes CITAR todas tus fuentes.
- Usa inteligencia artificial y agrega un ANEXO con todos los prompts que utilizaste para generar tus soluciones.
- Entregas con codigo identico o muy similar recibiran nota de 0 en automatico.
- El video debe mostrar TU pantalla y TU voz explicando el trabajo.

Preguntas Guia para la Documentacion

Responde brevemente estas preguntas en tu seccion de Analisis y Conclusiones del PDF:

- ¿Por que la version recursiva de Fibonacci es drasticamente mas lenta que la iterativa a medida que n crece?
- ¿Que significa que un algoritmo sea O(n^2)? ¿Como se refleja en la grafica del Burbuja?
- Si n se duplica, ¿cuanto aumenta el tiempo de ejecucion para O(n), O(n^2) y O(2^n)?
- ¿Cual es el costo de usar recursion en Java en terminos de memoria (stack)?
- ¿En que situaciones elegirías la version recursiva sobre la iterativa, y viceversa?

Firmas

Ing. Brandon Chitay | Catedratico Estructura de Datos

Exito en tu examen. Recuerda: el codigo que escribes hoy es la base de todo lo que construiras manana.

Universidad Da Vinci de Guatemala • Facultad de Ingenieria, Industria y Tecnologia