UNIVERSIDAD ESTATAL A DISTANCIA
VICERRECTORÍA ACADÉMICA
DIRECCIÓN DE EXTENSIÓN UNIVERSITARIA

TÉCNICO UNIVERSITARIO
EN COMPUTACIÓN E INFORMÁTICA

GUÍA DE ESTUDIO

ANÁLISIS Y DISEÑO DE ALGORITMOS I

CÓDIGO 50289

Elaborada por
Kay Guillén Díaz

2009

Edición académica
Virginia Ramírez Cascante

Encargado de cátedra
Grettel Mena Araya

Revisión filológica
Jacqueline Murillo Fernández

Esta guía de estudio ha sido confeccionada para ser utilizada en el curso Análisis y Diseño de Algoritmos I, código 50289, del programa Técnico Universitario en Computación e Informática que imparte la UNED.

CONTENIDO

PRESENTACIÓN **5**

DESCRIPCIÓN DEL CURSO **6**

Objetivo general **6**

Requisitos y valor en créditos **6**

Consejos de estudio **6**

Material de apoyo **7**

GUÍA DE LECTURAS **10**

Tutoría presencial I **11**

Instrucciones 11
TEMA 1: Análisis de algoritmos 11
Objetivo 11
Recomendaciones y sugerencias 12
Ejercicios 14
Glosario 14
TEMA 2: Bases de programación en JAVA 15
Objetivo 15
Contenidos 15
Recomendaciones y sugerencias 15
Ejercicios 16
Glosario 17

Tutoría presencial II **18**

Instrucciones 18
TEMA 3: Recursividad 18
Objetivo 18
Contenidos 18
Recomendaciones y sugerencias 19
Ejercicios 21
Glosario 21
TEMA 4: Reglas prácticas para el cálculo de la eficiencia 21
Objetivo 21
Contenidos 21
Recomendaciones y sugerencias 22
Ejercicios 23
Glosario 23

Tutoría presencial III **24**

Instrucciones 24

TEMA 5: Análisis de complejidad de algoritmos 24
Objetivo 24
Contenidos 24
Recomendaciones y sugerencias 25
Ejercicios 25
Tutoría presencial IV 26
Instrucciones 26
TEMA 6: Consideraciones sobre las técnicas de diseño de algoritmos 26
Objetivo 26
Contenidos 26
Recomendaciones y sugerencias 27
Ejercicios 27
Glosario 28
TEMA 7: Programación dinámica 28
Objetivo 28
Contenidos 28
Recomendaciones y sugerencias 29
Ejercicios 29
Glosario 30
TEMA 8: Algoritmos de vuelta atrás 30
Objetivo 30
Contenidos 30
Recomendaciones y sugerencias 30
Ejercicios 31
Glosario 31
REFERENCIAS 32

PRESENTACIÓN

El análisis es una actividad fundamental en el mundo de la informática, ya que, es una herramienta básica en la solución de problemas.

En este proceso se deben tomar en cuenta diferentes variables de orden económico que se encuentran diluidas en varios aspectos. Uno de ellos es el periodo de uso del procesador del equipo, que va a influir directamente en el tiempo en que se obtendrá la respuesta requerida y, por consiguiente, en la cantidad total de la actividad en la que está involucrada el uso del sistema computacional; y el espacio de memoria que se utilizará en la resolución del problema que se trata.

Desde esta perspectiva, el análisis y diseño de algoritmos constituye en el desarrollo de sistemas computacionales, un elemento fundamental en la formación de programadores y, por lo tanto, un requisito indispensable en la adquisición de destrezas.

El curso de Análisis y diseño de algoritmos I, que se ofrece como parte del primer bloque del programa de Técnico Universitario en Computación e Informática, de la Universidad Estatal a Distancia, tiene como fin introducir al estudiante en el área de la construcción de algoritmos eficaces, que promuevan la optimización del uso de los recursos disponibles.

DESCRIPCIÓN DEL CURSO

Objetivo general

Resolver problemas a través del lenguaje de programación orientado a objetos, según la estructura de control, el tipo de datos, la gestión de memoria y los mecanismos de abstracción.

Requisitos y valor en créditos

El curso Análisis y Diseño de Algoritmos, forma parte del plan de estudios de Técnico Universitario en Computación e Informática, impartido por la Universidad Estatal a Distancia, y pertenece al Bloque I, nivel técnico, por lo que no posee requisitos y no tiene correquisitos.

Consejos de estudio

Este curso es de naturaleza teórico-práctica, por lo que se recomienda para el y la estudiante:

- Ajustarse al programa de estudio propuesto y mantener el orden de los temas, ya que estos son acumulativos.
- Programar el tiempo de estudio a partir de las siguientes consideraciones:
    - Asistencia a las tutorías (1,5 horas) y a los laboratorios (1,5 horas), 3 horas por semana, 12 horas por mes.
    - Trabajo en la plataforma: 7 horas por semana, 28 horas por mes.
    - Realización de las tareas y los proyectos del curso: 10 horas por mes.

- Leer los materiales didácticos antes de asistir a las tutorías, puesto que estas únicamente se dedicarán a la aclaración de dudas, realización de ejercicios y prácticas.
- Revisar constantemente la plataforma, ya que en esta se colocarán propuestas de lecturas.
- Participar en los foros programados para ampliar aprendizaje e intercambiar conocimiento de formas dinámicas.
- Desarrollar los ejercicios y prácticas propuestas para cada tema para alcanzar su adecuada comprensión, la aprehensión del conocimiento y el desarrollo de las habilidades requeridas en el área técnica.

Material de apoyo

La informática es un área en constante evolución debido a la gran diversidad de investigación que se incrementa día a día. En particular, los expertos y las expertas siempre están a la búsqueda de desarrollar algoritmos eficientes, con el fin de minimizar los tiempos de respuesta y el requerimiento de los recursos computacionales.

Por esta razón se recomienda el estudio constante de información actualizada. De igual manera, se le insta a participar de este proceso y llevar a cabo sus propias investigaciones.

Los libros que se utilizarán como material obligatorio de estudio, son:

- Brassard, G.; Bratley, P. (1997). *Fundamentos de Algoritmia.* España: Prentice Hall.
- García, L. y otros. (2001). *Construcción lógica de programas. Teoría y problemas resueltos.* España: Editorial Rama.
- Ramírez, F. (2006). *Lógica de programación. Algoritmos y su implementación en VB.NET, C#, C++ y JAVA.* México: Aprenda Practicando Ediciones.

Otras fuentes posibles de consulta son:

- Arnow, D.; Weis. D. (2000). *Introducción a la programación en JAVA. Un enfoque Orientado a Objetos.* México: Addison-Wesley.
- Camacho, D.; Valls, J.M.; García, J.; Molina, J.M. y Buenos, E. (2003). *Programación y algoritmos y ejercicios resueltos en JAVA.* 2.ª edición. Madrid: Pearson Educación, S.A.
- Martí, N.; Segura, C.; Verdejo, J.A. (2006). *Especificación, derivación y análisis de algoritmos. Ejercicios resueltos.* Madrid: Pearson Educación, S.A.
- Hernández, R.; R. Lázaro, J. C. y Dormido, S. R. (2001). *Estructuras de datos y algoritmos.* 3.ª edición. Madrid: Pearson Educación, S.A.
- Laza, R. (2008). *Metodología y tecnología de programación.* España: Prentice Hall
- Baase, S. y Gelder, A. (2002). *Algoritmos computacionales. Introducción al análisis y diseño.* 3.ª edición. México: Pearson Educación, S.A.

En el cuadro 1 se presentan algunos enlaces digitales que puede consultar para complementar su aprendizaje.

CUADRO 1: **ENLACES DIGITALES**

ENLACE	**DESCRIPCIÓN DEL SITIO**
http://www.algoritmia.net/	Tiene foros, y secciones con información referente a algoritmos, estructuras de datos y soporte en técnicas de programación.
http://www.desarrolloweb.com/manuales/67	Manual para el desarrollo de las habilidades analíticas y creadoras de los programadores y las programadoras. Explica las bases de la programación y la creación de algoritmos.
http://www.idg.es/pcworldtech	Revista de computación *PC World* en línea.
http://www.pc-actual.com	Revista de computación *PC-Actual* en línea
http://www.mkm-pi.com	Sitio con enlaces a diferentes revistas de computación en línea y noticias del ámbito computacional.
http://www.lawebdelprogramador.com/cursos/enlace.php?idp=4683&id=5&texto=Algoritmia	Sitio con acceso al *Manual de análisis y diseño de algoritmos.*

GUÍA DE LECTURAS

En el cuadro 2 se desglosan los temas según las lecturas recomendadas.

CUADRO 2: **DESGLOSE DE TEMAS**

TEMAS	LIBRO	CAPÍTULO	SESIÓN	TUTORÍA
TEMA 1: Análisis de algoritmos	Brassard, G.; Bratley, P. (1997). *Fundamentos de Algoritmia.* Madrid: Prentice Hall. Martí, N.; Segura, C.; Verdejo, J. A. (2006). *Especificación, derivación y análisis de algoritmos. Ejercicios resueltos*. Madrid: Pearson Educación, S.A.	1 1	1 1	1 1
TEMA 2: Bases de programación en JAVA	Ramírez, F. (2006). *Lógica de programación. Algoritmos y su implementación en VB.NET, C#, C++ y JAVA.* México: Aprenda practicando ediciones.	13	2 y 3	1
TEMA 3: Recursividad	Martí, N.; Segura, C.; Verdejo, J.A. (2006). *Especificación, derivación y análisis de algoritmos. Ejercicios resueltos.* Madrid: Pearson Educación, S.A.	26	4	2
TEMA 4: Reglas prácticas para el cálculo de la eficiencia	Brassard, G.; Bratley, P. (1997). *Fundamentos de Algoritmia.* Madrid: Prentice Hall.	2	5 y 6	2
TEMA 5: Análisis de complejidad de algoritmos	Brassard, G.; Bratley, P. (1997). *Fundamentos de Algoritmia.* Madrid: Prentice Hall.	3 y 4	7 y 8	3
TEMA 6: Consideraciones sobre las técnicas de diseño de algoritmos	Brassard, G.; Bratley, P. (1997). *Fundamentos de Algoritmia.* Madrid: Prentice Hall.	7	9 y 10	4
TEMA 7: Programación dinámica	Brassard, G.; Bratley, P. (1997). *Fundamentos de Algoritmia.* Madrid: Prentice Hall.	8	11	4
TEMA 8: Algoritmos de vuelta atrás	Brassard, G.; Bratley, P. (1997). *Fundamentos de Algoritmia.* Madrid: Prentice Hall.	9	12	4

Tutoría presencial I

Esta tutoría comprende los siguientes temas de estudio:

- Tema 1: Análisis de algoritmos
- Tema 2: Bases de programación en JAVA

Instrucciones

Para la comprensión y asimilación de cada uno de los temas, el y la estudiante debe:

- Realizar las lecturas del libro de texto que se indican para cada tema.
- Realizar las lecturas complementarias al tema que se recomiendan en la plataforma multimedial.
- Realizar los ejercicios que se indican para cada tema en el libro de ejercicios resueltos.
- Llevar un registro de las dudas que surjan para ser evacuadas en la tutoría.

TEMA 1: Análisis de algoritmos

Objetivo

Reconocer el uso de los algoritmos y sus principales técnicas demostrativas.

Contenidos

Los contenidos de este tema se encuentran en los libros de Brassard y Bratley (1997) y Martí, N.; Segura, C. y Verdejo, J. (2006), como aparecen en el cuadro 3.

CUADRO 3: **CONTENIDOS TEMA 1**

TEMA	LIBRO	PÁGINA
Especificación de algoritmos	Brassard y Bratley (1997)	2
Especificar e implementar algoritmos	Martí, N.; Segura, C. y Verdejo, J. (2006)	1
Cuantificadores para escribir expresiones algorítmicas	Brassard y Bratley (1997)	11-13
Asociación de valores a variables	Martí, N.; Segura, C. y Verdejo, J. (2006)	1; 22-24
La precondición y la poscondición en lógica de primer orden	Martí, N.; Segura, C. y Verdejo, J. (2006)	2, 21
Formulas atómicas y el predicado	Martí, N.; Segura, C. y Verdejo, J. (2006)	2

Recomendaciones y sugerencias

Un algoritmo es una serie de pasos que siguen una secuencia lógica y que permiten resolver un problema.

Ejemplo: ¿Qué debe hacer una persona por las mañanas para ir al trabajo?

1. Levantarse
2. Quitarse la ropa de dormir
3. Bañarse
4. Vestirse
5. Desayunar

Cada una de estas actividades se puede subdividir, a su vez, en una serie de pasos:

1. Levantarse
    Despertarse cuando suena el despertador
    Quitarse las cobijas
    Sentarse en la cama
    Poner los pies en el suelo
    Levantarse
2. Quitarse la ropa de dormir

    Desabotonarse la ropa
    Quitarse el pantalón de dormir
    Quitarse la camisa de dormir
3. Bañarse
    Tomar un paño
    Caminar hacia el baño
    Ingresar al baño
    Abrir la ducha
    Enjabonarse
    Quitarse el jabón
    Cerrar la ducha
    Tomar el paño
    Secarse
4. Vestirse
5. Desayunar

Cada una de estas actividades se puede desglosar de forma tan detallada como se desee, de manera que se asegure llegar al resultado esperado.

> Desarrolle el algoritmo para las actividades que faltan: Vestirse y desayunar.

Para seleccionar el algoritmo óptimo al resolver el problema que se plantea, se establecen las propiedades matemáticas de estos y se utiliza alguna técnica de demostración. En particular se mencionan dos: la técnica de demostración por

contradicción o prueba indirecta y la prueba de demostración por inducción matemática. Para profundizar sobre estos temas consúltese el libro de Brassard y Bratley (1997), de la página 15 a la 35.

Ejercicios

Revise, analice y estudie los ejercicios resueltos del libro de Martí, Segura y Verdejo (2006), sección 1.2, de la página 3 a la página 18.

Desarrolle los ejercicios propuestos del libro de Martí, Segura y Verdejo (2006), sección 1.3, de la página 18 a la página 19.

Glosario

algoritmo. Serie de pasos que siguen una secuencia lógica, y que permiten resolver un problema.

cuantificador. Operaciones que introducen variables. Son de dos tipos: cuantificador universal "◎" y el cuantificador existencial "◎" (Louden, 2004).

expresión algorítmica. Aquella que contiene una especificación formal de algoritmos basada en lógica de predicados de pre y poscondición (Louden, 2004).

fórmula atómica. Es aquella que no puede ser subdividida en otra fórmula.

lógica de primer orden. Manera de expresar de manera formal los enunciados lógicos (Louden, 2004).

poscondición. Predicado que incluye como variables los parámetros de entrada y salida de un algoritmo (Louden, 2004).

predicado. Nombres de la funciones que son verdaderas o falsas, como las funciones *booleanas* (Louden, 2004).

precondición. Predicado que incluye como variables los parámetros de entrada de un algoritmo (Louden, 2004).

variable. Representa una cantidad aún no especificada. Siempre tienen un tipo asociado que señala el valor que puede tomar (Louden, 2004).

TEMA 2: Bases de programación en **JAVA**

Objetivo

Diseñar algoritmos a través del lenguaje de programación JAVA.

Contenidos

Los contenidos de este tema se encuentran en el libro de Ramírez (2006), como se muestran en el cuadro 4.

CUADRO 4: **CONTENIDOS TEMA 2**

TEMA	PÁGINA
Generalidades del lenguaje *JAVA*	371
Estructura básica de programas	143; 371-374
Solicitud de datos por pantalla	375
Condicionales y *bucles* en *JAVA*	144; 377-381
Compilación y ejecución de programas	381

Recomendaciones y sugerencias

JAVA es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a inicios de los años noventa. En su sintaxis hace uso de muchas de las instrucciones de C y C++, y a diferencia de este, es interpretado, esto es, cada línea del

programa se traduce al código máquina del sistema operativo donde está siendo ejecutado en el momento de la corrida del programa.

En el cuadro 5 se presentan las instrucciones básicas de programación.

CUADRO 5: **INSTRUCCIONES DE PROGRAMACIÓN BÁSICAS**

INSTRUCCIÓN	DESCRIPCIÓN	EJEMPLO
IF	Ejecuta una acción si la condición es verdadera, u otra acción si la condición es falsa.	*IF* es martes, ir a clases de natación /* Va a clases de natación si el día es martes *Else* ir a trabajar /* Sino, va a trabajar
	También se pueden hacer instrucciones *IF* de forma anidada.	*IF*es sábado o domingo, me puedo levantar a las 8:00 a.m. *Else IF* es martes, levantarse a las 6:00 a.m. e ir a clases de natación *Else* levantarse a las 6:00 a.m. e ir a trabajar
While	Es una estructura de repetición que realiza una acción mientras se cumple una condición, que es verificada antes de iniciar la acción.	*While* no esté lloviendo, puedo ir a jugar fútbol al aire libre
For	Es una estructura de repetición que realiza una acción mientras se cumple una condición.	*For* contador que va desde 1 hasta 10 Aguantar la respiración Incrementar el contador en 1
Do… while	Es una estructura de repetición que realiza una acción al menos una vez, ya que la condición se verifica luego de que se ha llevado a cabo la acción.	*Do* Comer helado Incrementar el contador en 1 *While* el contador sea menor a 10
Switch	Selecciona una instrucción a ejecutar, con base en el valor de entrada	*Switch Opcion* *Case* 1 Instrucción a ejecutar si la variable *Opcion* = 1 *Break*; *Case* 2 Instrucción a ejecutar si la variable *Opcion* = 1 *Break* *Case* 3 Instrucción a ejecutar si la variable *Opcion* = 1 *Break*; *Default:* Instrucción a ejecutar si la variable *Opcion* es diferente de 1, 2 ó 3

Ejercicios

Para el estudio de los ejercicios que se sugieren en este apartado, debe instalar en el computador, el lenguaje de programación JAVA.

Estudie, analice y comprenda los ejercicios resueltos de las secciones 13.01 a la 13.05, ubicadas en las páginas de la 382 a la 393. Además, desarrolle los ejercicios de las secciones 13.06 a la 13.07, ubicadas en las páginas de la 384 a la 395 del libro de Ramírez (2006).

Glosario

lenguaje de programación. Es una estructura simbólica que permite el desarrollo de un código fuente; posee estructura, sintaxis y semántica, y haciendo uso de palabras reservadas, expresiones y símbolos especiales (Ramírez, 2006).

condicional. Expresión lógica que determina la ejecución de un bloque de código si es verdadera. Puede ser simple o compuesto (Ramírez, 2006).

***bucles*.** Estructura de control en la que una expresión lógica determina la ejecución de un bloque de código. Puede ser de "comparación al inicio" o de "comparación al final" (Ramírez, 2006).

compilación. Traducción del código fuente a lenguaje de máquina.

Tutoría presencial II

Esta tutoría comprende los siguientes temas de estudio:

- Tema 3: Recursividad
- Tema 4: Reglas prácticas para el cálculo de la eficiencia

Instrucciones

Para la comprensión y asimilación de cada uno de los temas debe:

- Realizar las lecturas del libro de texto que se indican para cada tema.
- Realizar las lecturas complementarias al tema que se recomiendan en la plataforma multimedial.
- Realizar los ejercicios que se indican para cada tema, del libro de ejercicios resueltos.
- Llevar un registro de las dudas que surjan para ser evacuadas en la tutoría.

TEMA 3: Recursividad

Objetivo

Interpretar el proceso de recursividad en los algoritmos computacionales para resolver problemas.

Contenidos

Los contenidos de este tema se encuentran en el libro de Martí, N.; Segura, C. y Verdejo, J. (2006), como se muestra en el cuadro 6.

CUADRO 6: **CONTENIDOS TEMA 3**

TEMA	PÁGINA
Características de la recursividad	26
Razonamiento recursivo	26-27
Algoritmos recursivos lineales	27
Diseño de algoritmos recursivos	27, 143
Análisis por casos y composición	143

Recomendaciones y sugerencias

En programación se entiende por recursividad un programa que se llama a sí mismo.

El ejemplo más claro de algo recursivo es la función factorial que se define como

$$F(n) = n * F(n-1), \text{ con } n > 0$$

El algoritmo de esta función es

función *Factorial (n)*

Si *n=0* **entonces devolver** 1

Sino devolver *n* Factorial (n-1)*

Todo algoritmo recursivo consta de dos partes:

1. Un caso trivial, no recursivo: **Si** *n=0* **entonces devolver** 1
2. Un caso complejo que se define en términos de uno más simple: **Sino devolver** *n * Factorial (n-1)*

Es muy importante asegurarse de definir el caso trivial, de lo contrario se puede crear un *bucle* o ciclo infinito.

Analice el caso de la función factorial. Como se mencionó, es definida por

$$F(n) = n * F(n-1), \text{ con } n > 0$$

Suponga $n = 4$, el caso complejo. Un caso más sencillo que 4! es 3!, y más sencillo que 3! es 2!, y más sencillo que 2! es 1!, y más sencillo es 0! Que, por definición, es 1, este es el caso trivial.

Entonces,

$$F(4) = 4 * F(3) = 4 * 3 * F(2) = 4 * 3 * 2 * F(1) = 4 * 3 * 2 * 1 * F(0) = 4 * 3 * 2 * 1 * 1 = 24.$$

En el cuadro 7 se muestra un ejemplo de la función factorial.

CUADRO 7: **FUNCIÓN FACTORIAL**

función *Factorial (4)* **Si** 4=0 **entonces devolver** 1 **Sino devolver** *4 * Factorial (4-1)*	Llamada recursiva *Factorial(3)*
función *Factorial (3)* **Si** 3=0 **entonces devolver** 1 **Sino devolver** *3 * Factorial (3-1)*	Llamada recursiva *Factorial(2)*
función *Factorial (2)* **Si** 2=0 **entonces devolver** 1 **Sino devolver** *2 * Factorial (2-1)*	Llamada recursiva *Factorial(1)*
función *Factorial (1)* **Si** 1=0 **entonces devolver** 1 **Sino devolver** 1 * *Factorial (1-1)*	Llamada recursiva *Factorial(0)*
función *Factorial (0)* **Si** 0=0 **entonces devolver** 1 **Sino devolver** *0 * Factorial (0-1)*	Devuelve 1

Ejercicios

Estudie, analice y comprenda los ejercicios resueltos de las secciones 13.01 a la 13.07, ubicadas en las páginas de la 382 a la 395, del libro de Martí, N.; Segura, C. y Verdejo, J. (2006).

Glosario

recursividad. Un programa que se llama a sí mismo.

algoritmos recursivos lineales. Un algoritmo es recursivo lineal si su ejecución genera un única llamada recursiva (Martí, N.; Segura, C. y Verdejo, J., 2006, p. 27).

algoritmos recursivos múltiples. Un algoritmo es recursivo múltiple si una misma invocación genera más de una llamada recursiva (Martí, N.; Segura, C. y Verdejo, J., 2006, p. 27).

TEMA 4: Reglas prácticas para el cálculo de la eficiencia

Objetivo

Reconocer el uso de los algoritmos y sus principales técnicas demostrativas.

Contenidos

Los contenidos de este tema se encuentran en el libro de Brassard y Bratley (1997) como se muestra en el cuadro 8.

CUADRO 8: **CONTENIDOS TEMA 4**

TEMA	PÁGINA
Enfoques para la selección de algoritmos eficientes (empírico, a priori, híbrido)	67-70
Análisis de caso peor y caso medio	70-73
La operación elemental	73-76
Aceleración de algoritmos	76-78

Recomendaciones y sugerencias

Un algoritmo se dice que es eficaz cuando funciona correctamente para todos lo casos del problema que debe resolver. Se demuestra que es incorrecto cuando se encuentra un contraejemplo para el que no sirve, pero antes de decidir que no es eficaz, se debe asegurar que está dentro del dominio de definición del mismo.

> El dominio de definición es el conjunto de soluciones posibles definidas para un algoritmo; por ejemplo, una solución puede ser diseñada para sumar únicamente números positivos.

Se dice que un algoritmo es eficiente cuando no solo hace lo que se espera, sino que además lo ejecuta de la mejor forma posible, con el uso óptimo de los recursos disponibles (memoria y tiempo que utiliza en la resolución del problema).

Hay tres enfoques para analizar la eficiencia de los algoritmos:

- **Empírico:** se programan las técnicas, se prueban en distintos escenarios y se comparan entre sí.
- **Teórico:** se determina matemáticamente la cantidad de recursos necesarios.

- **Híbrido:** consta de dos pasos; primero se determina teóricamente la eficacia, y luego, de forma empírica, se definen los parámetros numéricos específicos para un programa en una computadora en particular.

Ejercicios

Estudie, analice y comprenda los ejemplos que se presentan en la sección 2.7 del libro de Brassard y Bratley (1997), páginas de la 78 a la 85. Además, los ejercicios de la sección 2.9 números 2.2 al 2.21, del mismo libro, páginas de la 86 a la 89.

Glosario

dominio de definición. Conjunto de soluciones posibles definidas para un algoritmo.

operación elemental. Operación que se ejecuta en un tiempo constante, independientemente de los parámetros de entrada.

caso peor, caso medio. Tiempo que dura ejecutándose un algoritmo en el peor de los casos (tiempo que más dura) y en el caso medio.

Tutoría presencial III

Esta tutoría comprende el siguiente tema de estudio:

- Tema 5: Análisis de complejidad de algoritmos

Instrucciones

Para la comprensión y asimilación de cada uno de los temas, el y la estudiante debe:

- Realizar las lecturas del libro de texto que se indican para cada tema.
- Realizar las lecturas complementarias al tema que se recomiendan en la plataforma multimedial.
- Resolver los ejercicios que se indican para cada tema, del libro de ejercicios resueltos.
- Llevar un registro de las dudas que surjan para ser evacuadas en la tutoría.

TEMA 5: Análisis de complejidad de algoritmos

Objetivo

Identificar la complejidad en la eficacia de los algoritmos interactivos y recursivos con base al consumo de recursos.

Contenidos

Los contenidos de este tema se encuentran en los libros de Martí, N.; Segura, C. y Verdejo, J. (2006) y Brassard y Bratley (1997) como se muestra en el cuadro 9.

CUADRO 9: **CONTENIDOS TEMA 5**

TEMA		PÁGINA
Los costes reales de los algoritmos	Martí, N.; Segura, C. y Verdejo, J. (2006)	61
Funciones de costes	Martí, N.; Segura, C. y Verdejo, J. (2006)	61
El teorema del límite	Martí, N.; Segura, C. y Verdejo, J. (2006)	62
Instrucciones con sus costes	Martí, N.; Segura, C. y Verdejo, J. (2006)	62-63
Complejidad en algoritmos recursivos, la resolución en la recurrencia	Brassard y Bratley (1997)	132-160

Recomendaciones y sugerencias

Al momento de analizar cuál algoritmo es el más adecuado para un problema determinado, se debe considerar no solo que sea correcto (efectivo), sino también que sea eficiente, para lo que debe evaluarse los recursos computacionales que utiliza: tiempo y memoria. A la cantidad de memoria se le llama coste en espacio, y al tiempo coste en tiempo, y se busca que ambos sean tan bajos como sea posible y que el resultado sea correcto.

Esto significa que ante dos algoritmos correctos, se elige el que tenga menor coste. Para definir el coste de un algoritmo, se debe conocer el coste de cada una de las operaciones que lo componen. Este se calcula en función del tamaño de las entradas que tenga, para que sea independiente del computador en que se ejecuta, y del lenguaje de programación y el compilador.

Ejercicios

Estudie, analice y comprenda los ejercicios resueltos del 3.1 al 3.27, en las páginas 63 a la 88, del libro de Martí, N.; Segura, C. y Verdejo, J. (2006).

Desarrolle los ejercicios propuestos del libro de Brassard y Bratley (1997), ejercicios 4.30 al 4.43, de las páginas 163 a la 165.

Tutoría presencial IV

Esta tutoría comprende los siguientes temas de estudio:

- Tema 6: Consideraciones sobre las técnicas de diseño de algoritmos
- Tema 7: Programación dinámica
- Tema 8: Algoritmos de vuelta atrás

Instrucciones

Para la comprensión y asimilación de cada uno de los temas, el y la estudiante debe:

- Realizar las lecturas complementarias al tema que se recomiendan en la plataforma multimedial.
- Realizar los ejercicios que se indican para cada tema, del libro de ejercicios resueltos.
- Llevar un registro de las dudas que surjan para ser evacuadas en la tutoría.

TEMA 6: Consideraciones sobre las técnicas de diseño de algoritmos

Objetivo

Analizar la técnica "divide y vencerás" en el diseño de algoritmos eficientes para la solución de problemas tanto reales como ficticios.

Contenidos

Los contenidos de este tema se encuentran en el libro de Brassard y Bratley (1997) como se muestra en el cuadro 10.

CUADRO 10: **CONTENIDOS TEMA 6**

TEMA	PÁGINA
El caso general de los algoritmos divide y vencerás	247-255
Las tres condiciones del algoritmo divide y vencerás	252
La búsqueda binaria	255-257
Enfoques para la ordenación	257-266

Recomendaciones y sugerencias

La técnica "divide y vencerás" busca tomar un problema difícil y dividirlo en tantas partes como sea necesario, de forma que la solución de cada una sea trivial y la suma de las soluciones brinde la respuesta final.

Para que el uso de esta técnica sea eficiente, se requiere que el problema se pueda dividir en partes más sencillas, de similar tamaño, del mismo tipo, cada una con su propia solución, y que al final todas se puedan combinar para obtener la respuesta del problema original.

El diseño de este tipo de algoritmos puede pensarse de forma recursiva o lineal. En este último caso, se debe elegir la estructura de datos más apropiada para almacenar los resultados de los subproblemas.

Ejercicios

Revise, analice y comprenda los ejercicios y algoritmos desarrollados a lo largo del capítulo, en las secciones 7.3, 7.4, 7.5 y 7.6, del libro de Brassard y Bratley (1997).

Desarrolle los ejercicios propuestos del libro de Brassard y Bratley (1997), ejercicios 7.2 al 7.16, de las páginas 282 a la 284, tome como base los ejercicios y algoritmos previamente estudiados en este mismo tema.

Glosario

divide y vencerás. Técnica que consiste en dividir un problema grande en problemas más pequeños

búsqueda binaria. Técnica que consiste en buscar un dato en una lista de datos, dividiendo cada vez la lista de búsqueda a la mitad.

estructura de datos. Forma de organizar un conjunto de datos para facilitar su manipulación.

TEMA 7: Programación dinámica

Objetivo

Aplicar la programación dinámica para optimizar el rendimiento computacional.

Contenidos

Los contenidos de este tema se encuentran en el libro de Brassard y Bratley (1997) como se muestran en el cuadro 11.

CUADRO 11: **CONTENIDOS TEMA 7**

TEMA	PÁGINA
El manejo de tablas de resultados	291-298
El principio de optimalidad	298-309
Enfoques que aplican recursión	309-311
Las funciones con memoria	311-312

Recomendaciones y sugerencias

La programación dinámica, al igual que el algoritmo de divide y vencerás, busca dividir un problema grande en pequeños casos. La diferencia es que en vez de hacer refinamientos progresivos, se busca solucionar los casos más sencillos y, con la solución de estos, obtener la de los casos más grandes hasta obtener la respuesta final. Otra característica de la programación lineal es que los casos pueden ser superpuestos, por ejemplo, la resolución de la sucesión de *Fibonacci*, definida como

$$fib(n) = n, \text{ si } n < 2$$
$$fib(n) = fib(n-1) + fib(n-2), \text{ si } n \geq 2$$

En este caso, si se desea calcular *fib(5),* se tendrá:

$$fib(5) = fib(4) + fib(3)$$
$$fib(5) = \left(fib(3) + fib(2)\right) + \left(fib(2) + fib(1)\right)$$
$$fib(5) = \left(fib(2) + \left(fib(1)\right) + \left(fib(1) + fib(0)\right)\right) + \left(fib(1) + fib(0) + 1\right)\right)$$

Como se observa, *fib(2)* se calcula tres veces. En valores mayores de *n*, se calculará aún más veces. Esto se evita si se utiliza una tabla de resultados de los diferentes casos y se reutilizan cuando sea necesario, en vez de volver a calcularlos.

Ejercicios

Revise, analice y comprenda los ejercicios y algoritmos desarrollados a lo largo del capítulo, en las secciones 8.1, 8.2, 8.4, 8.5, 8.6, 8.7 y 8.8 del libro de Brassard y Bratley (1997).

Desarrolle los ejercicios propuestos del libro de Brassard y Bratley (1997), ejercicios del 8.2 al 8.9 y del 8.26 al 8.31, de las páginas 312 a la 315. Tome como base los ejercicios y algoritmos previamente estudiados en este mismo tema.

Glosario

tablas de resultados. Estructura de datos que permite guardar los resultados de los cálculos de los casos más pequeños en la resolución de un problema.

funciones con memoria. Funciones que utilizan tablas de resultados, de forma que guardan memoria de lo valores que han sido previamente calculados.

TEMA 8: Algoritmos de vuelta atrás

Objetivo

Establecer el uso de algoritmos de vuelta atrás para la solución de problemas.

Contenidos

Los contenidos de este tema se encuentran en el libro de Brassard y Bratley (1997) según se muestra en el cuadro 12.

Cuadro 12: **CONTENIDOS TEMA 8**

TEMA	PÁGINA
Analizando las posibilidades para buscar una solución	319 - 342
Funcionamiento del algoritmo vuelta atrás	342
Algoritmos vuelta atrás recursivos	343-348

Recomendaciones y sugerencias

Pueden existir muchas formas de solucionar un mismo problema programático; pero, como ya se ha mencionado, se busca elegir aquel método que utilice la menor cantidad de recursos computacionales.

Una de las características para que la programación dinámica sea mejor que la recursiva, es evitar calcular casos anteriores a través de la memorización.

Ahora bien, si el problema por resolver es muy grande, puede pasar que la cantidad de memoria requerida sea muy grande, lo cual está fuera de lo deseable. Es entonces cuando se puede pensar en los algoritmos de vuelta atrás, cuya característica es que sólo memoriza aquellas soluciones que conforme avanza el cálculo de los casos, van demostrando ser los mejores y borra los demás.

Ejercicios

Revise, analice y comprenda los ejercicios y algoritmos desarrollados a lo largo del capítulo, en las secciones 9.1, 9.2, 9.3, 9.4, 9.5 y 9.6 del libro de Brassard y Bratley (1997).

Desarrolle los ejercicios propuestos del libro de Brassard y Bratley (1997), ejercicios del 9.40 al 9.49, de las páginas 361 y 362. Tome como base los ejercicios y algoritmos previamente estudiados en este mismo tema

Glosario

vuelta atrás. Técnica de programación que soluciona un problema de atrás hacia adelante, compara los resultados previamente memorizados contra los últimos calculados y desecha el de mayor coste.

REFERENCIAS

Brassard, G.; Bratley, P. (1997). *Fundamentos de Algoritmia.* España: Prentice Hall.

García, L. y otros. (2001). *Construcción lógica de programas. Teoría y problemas resueltos.* España: Editorial Rama.

Louden, K. (2004). *Lenguajes de programación: Principios y práctica.* 2.ª Edición. México: Thomson Learning

Ramírez, F. (2006). *Lógica de programación. Algoritmos y su implementación en VB.NET, C#, C++ y JAVA.* México: Aprenda Practicando Ediciones.