UAB
Universitat Autònoma de Barcelona

Estructuras de Datos

Código: 104347
Créditos ECTS: 6

2025/2026

Titulación	Tipo	Curso
Ingeniería de Datos	OB	2

Contacto

Nombre: Gema Sanchez Albaladejo

Correo electrónico: gemma.sanchez@uab.cat

Equipo docente

Gema Sanchez Albaladejo

Daniel Soto Alvarez

Idiomas de los grupos

Puede consultar esta información al final del documento.

Prerrequisitos

La asignatura no tiene ningún prerrequisito oficial. De todas formas, se asume que el alumnado ha cursado las asignaturas previas de Fundamentos de programación y Programación Avanzada, así como Grafos, Topología y Geometría Discreta. Por lo tanto, sabe utilizar las estructuras básicas y avanzadas de la programación, Orientación a objetos y el concepto de grafo con los diferentes métodos de recorrido sobre ellos.

Objetivos y contextualización

Esta asignatura forma parte de la materia Representación de los Datos y debe verse como la continuación lógica de la asignatura Programación Avanzada y la continuación práctica de la asignatura de matemática Discreta. El objetivo básico es profundizar en las estructuras de datos básicos introducidas en Fundamentos de programación junto con las nociones de programación orientada a objetos introducidas en Programación Avanzada y ampliar con otras estructuras de datos más complejas así como algoritmos eficientes para recorre la foto. Se introducirá el concepto de algoritmo recursivo con algoritmos recursivos simples y más complejos como los relacionados con recorridos de árboles y grafos

De esta forma, los objetivos formativos que se proponen para la asignatura son los siguientes:

- Ser capaz de analizar un problema complejo, diseñar una solución óptima, implementarla, calcular su coste y probarla.
- Entender y saber utilizar estructuras de datos complejas como árboles, grafos etc. y utilizarlas correctamente y de una manera eficiente para resolver problemas algorítmicos complejas.

- Dotar al alumno de la capacidad de diseño de algoritmos para la resolución de problemas complejos, viendo algoritmos complejos de recorrido y búsqueda en estructuras de datos complejas. Además de analizar la complejidad temporal y espacial de ellos a fin de elegir la solución que más se adapte a las necesidades de cada momento.
- Introducir el concepto de recursividad y su aplicación al recorrido de estructuras complejas recursivas, además de ser capaz de analizar la complejidad de algoritmos recursivos.
- Programar en un lenguaje de programación real y ser capaz de depurar los propios programas.
- Desarrollar los programas siguiendo unas normas de estilo tendentes ac onseguir programas de calidad.

Competencias

- Buscar, seleccionar y gestionar de manera responsable la información y el conocimiento.
- Desarrollar un pensamiento y un razonamiento crítico y saber comunicarlo de manera efectiva, tanto en las lenguas propias como en inglés.
- Diseñar soluciones algorítmicas eficientes para problemas computacionales, implementarlas en forma de desarrollo de software robustos, estructurados y fáciles de mantener, y verificar su validez.
- Trabajar cooperativamente, en entornos complejos o inciertos y con recursos limitados, en un contexto multidisciplinar, asumiendo y respetando el rol de los diferentes miembros del equipo.

Resultados de aprendizaje

1. Buscar, seleccionar y gestionar de manera responsable la información y el conocimiento.
2. Desarrollar programas con un buen estilo de programación y bien documentados y saber depurarlos, testearlos y corregirlos.
3. Desarrollar un pensamiento y un razonamiento crítico y saber comunicarlo de manera efectiva, tanto en las lenguas propias como en inglés.
4. Seleccionar y aplicar la combinación de estructuras de datos y estrategias de resolución más apropiada para resolver de manera eficiente un problema informático.
5. Trabajar cooperativamente, en entornos complejos o inciertos y con recursos limitados, en un contexto multidisciplinar, asumiendo y respetando el rol de los diferentes miembros del equipo.

Contenido

0. Introducción
Objetivos y presentación de la asignatura. Repaso de Programación avanzada.

1. Estructuras de datos linieals. Listas, colas, pilas.

Representación y manipulación de estructuras de datos dinámicas: repaso de listas, introducción a las pilas y colas.

2. Estructuras de datos no lineales. hash

Técnicas de "Hashing". Listas hash. Funciones hash. Como se implementan en python. Diccionarios.

3. Recursividad y Algoritmos de ordenación

Introducción a los algoritmos recursivos. Método de la burbuja, QuickSort, mergesort. Recursividad. Cálculo complejidad.

4. Estructuras de datos no lineales. grafos
Representaciones y recorridos. BFS, DFS, Resolución de problemas con grafos.

5. Estructuras de datos no lineales. Árboles.

Definición y representación de un árbol. Recorridos de árboles es. Binary Heaps.

Actividades formativas y Metodología

Título	Horas	ECTS	Resultados de aprendizaje
Tipo: Dirigidas			
Clases presenciales	48	1,92	1, 2, 3, 4, 5
Trabajo autónomo	93	3,72	1, 2, 3, 4, 5

La metodología docente de la asignatura parte del principio que dice que *"programar es la única forma de aprender a programar"* y, por tanto, estará centrada principalmente en el trabajo práctico del alumnado. Las sesiones presenciales de clase se organizarán para introducir los contenidos teóricos de la asignatura, desde una perspectiva muy práctica a partir de ejemplos y de ejercicios y problemas de programación que se deberán resolver a cla sse directamente con el ordenador. El objetivo principal de la asignatura es que el alumnado sepa resolver un problema dado, de manera eficiente, utilizando estructuras de datos complejas, si es necesario. Por esta razón el aprendizaje se centrará en acompañar al alumnado en su tarea de resolución de problemas a partir de unos conceptos teóricos estudiados previamente de manera autónoma. Se utilizará principalmente el lenguaje de programación Python.

Por otra parte, se realizará un proyecto de programación que se tendrá que ir desarrollando de forma principalmente autónoma durante todo el curso (con seguimiento y control por parte del profesor en sesiones puntuales) y que supondrá integrar de forma práctica casi todos los conceptos y herramientas de programación introducidos en las sesiones presenciales en la resolución de un problema real complejo. Además, se propondrá un conjunto de ejercicios que deberán resolverse de forma individual a lo largo del curso (algunos de los cuales se resolverán y discutirán en las sesiones presenciales) que deben servir para comprender, integrar y aplicar los conceptos desarrollados en las sesiones presenciales. En las actividades del curso (sesiones presenciales, problemas y prácticas) se utilizará principalmente el lenguaje de programación Python.

A nivel presencial, las sesiones de clase se organizarán en cuatro horas semanales y se harán en un aula con ordenadores o electrificadas para facilitar el trabajo práctico del alumnado. Es necesario que el alumno lleve su propio portátil a clase si las aulas no disponen de uno. En las sesiones presenciales se irán introduciendo los conceptos que se detallan en el temario de la asignatura. en algunos casos, se podrán poner a disposición del alumnado vídeos explicativos u otro material complementario que el alumnado deberá visionar antes de la sesión de clase. Las sesiones de clase tendrán un enfoque bastante práctico con ejemplos y ejercicios que se plantearán a los alumnos para facilitar la comprensión y aprendizaje de los conceptos explicados. Estos ejercicios se realizarán y discutirán durante la sesión y servirán para ir introduciendo los contenidos de la asignatura y ver su aplicación práctica.

El alumnado deberá completar las clases presenciales con el trabajo personal autónomo en la realización de los ejercicios que se vayan proponiendo y que deben servir para acabar de entender los contenidos de la asignatura. Hay que tener presente que el temario de la asignatura tiene una continuidad lógica a lo largo del curso, por lo que para poder seguir correctamente una clase necesario haber asimilado lo explicado en las sesiones anteriores. Algunos de estos ejercicios se entregarán de forma individual como parte de la evaluación de la asignatura.

Además, el alumnado deberá hacer en grupos de 2 un proyecto de programación que se desarrollará de forma autónoma durantetodo el curso fuerade las sesiones presenciales. El proyecto de programación

permitirá abordar un problema de programación de cierta complejidad que integre la mayoría de los conceptos explicados durante el curso. durante el curso, se dedicarán algunas sesiones presenciales el control, seguimiento y evaluación del trabajo realizado por el alumnado en el proyecto de programación.

La gestión de la docencia de la asignatura se realizará a través de Caronte (http://caronte.uab.cat/), y de campus virtual (https://cv.uab.cat/) .

Nota: se reservarán 15 minutos de una clase dentro del calendario establecido por el centro o por la titulación para que el alumnado rellene las encuestas de evaluación de la actuación del profesorado y de evaluación de la asignatura o módulo.

Evaluación

Actividades de evaluación continuada

Título	Peso	Horas	ECTS	Resultados de aprendizaje
Examen de recuperación	50% Nota final	3	0,12	3, 4
Primer Parcial	25% Nota final	2	0,08	3, 4
Problemas	20% Nota final	0	0	1, 2, 3, 4
Proyecto	30% Nota final	2	0,08	1, 2, 3, 4, 5
Segundo parcial	25% Nota final	2	0,08	3, 4

La evaluación de la asignatura tendrá en cuenta tres tipos de actividades de evaluación: entrega de problemas, evaluación individual y proyecto de programación. la nota final de la asignatura se obtiene combinando la evaluación de estas 3 actividades de la siguiente manera:

Nota Final = (0.2 * Evaluación Problemas) + (0.3 * Proyecto) + (0.5 * Evaluación Individual)

- Entrega de problemas: en este apartado se incluye la entrega de los ejercicios que se propongan a lo largo del curso y otras actividades que se realicen en las sesiones de problemas.

Se deberá conseguir una nota mínima de 5 en esta actividad para poder aprobar la asignatura.
Los ejercicios que se entreguen fuera de plazo o que tengan una evaluación de suspendido se podrán recuperar y volver a entregar en cualquier momento del curso antes de la fecha del examen final de la asignatura, con una reducción sobre la nota del 20%. Los problemas estarán ponderados según el peso del tema al conjunto de la asignatura, y el número de problemas que tengan que entregar por cada tema.

- Evaluación individual: en este apartado se incluye el resultado de las pruebas individuales que se harán a lo largo del curso. Habrá dos pruebas parciales que se realizarán durante el período lectivo del curso en horarios de clase y una prueba final durante el periodo oficial de exámenes. Esta prueba final será de recuperación y sólo tendrán que hacer los estudiantes que no hayan superado alguno de los dos parciales. Si se ha superado uno de los dos parciales, pero el otro no, en esta prueba sólo se ha de recuperar la parte de la asignatura correspondiente al parcial que no se haya superado.

Se deberá conseguir una nota mínima de 4 en cada uno de los dos parciales y una nota promedio mínima de 5 para poder aprobar la asignatura.

La nota final será la media de los dos parciales: Evaluación Individual = (0.5 * Parcial1) + (0.5 * Parcial2)

- Proyecto: incluye todo el trabajo del proyecto de programación. Incluye la evaluación de las dos entregas del proyecto (una entrega parcial a mitad de curso y la entrega final) y la evaluación del seguimiento del proyecto que se hará a las sesiones presenciales que corresponda. La nota final se calculará de la forma siguiente:

Proyecto = (0.2 * Evaluación seguimiento proyecto) + (0.3 * Entrega Parcial 1) + (0.5 * Entrega Final)

- Se deberá conseguir una nota mínima de 4 en la evaluación del seguimiento del proyecto y una nota mínima de 5 en el entrega final del proyecto para poder aprobar el proyecto.
- Se deberá conseguir una nota mínima de 5 en el proyecto para poder aprobar la asignatura.
- La nota de la entrega final del proyecto se podrá recuperar si la nota del proyecto es> = 3 y la nota dela evaluación individual es> = 5.

No evaluable: Un alumno se considerará no evaluable (NA) si no hace como mínimo el 50% de las entregas de ejercicios y no hace ninguna de las pruebas de evaluación: parcial 1, parcial 2, prueba final de recuperación, entrega final de la práctica.

suspendidos: Si el cálculo de la nota final es igual o superior a 5 pero no se llega al mínimo exigido en alguna de las actividades de evaluación, la nota final será suspendido y se pondrá un 4.5 en la nota del expediente del alumno.

Convalidaciones: No se convalidarán proyectos de años anteriores.

MH: Se darán tantas matrículas como puedan dentro de la normativa de la universidad, empezando por las notas más altas y siempre y cuando la nota mínima sea un 9.

revisiones: Para cada actividad de evaluación, se indicará un lugar, fecha y hora de revisión en la que el estudiante podrá revisar la actividad con el profesor. En este contexto, se podrán hacer reclamaciones sobre la nota de la actividad, que serán evaluadas por el profesorado responsable de la asignatura. Si el estudiante no se presenta en esta revisión, no se revisará posteriormente esta actividad.

Nota importante sobre copias y plagios:
Sin perjuicio de otras medidas disciplinarias que se estimen oportunas, y de acuerdo con la normativa académica vigente, las irregularidades cometidas por un estudiante que puedan conducir a una variación de la calificación se calificarán con un cero (0). las actividades de evaluación calificadas de esta forma y por este procedimientono serán recuperables. Si es necesario superar cualquiera de estas actividades de evaluación para aprobar la asignatura, esta asignatura quedará suspendida directamente, sin oportunidad de recuperarla en el mismo curso. Estas irregularidades incluyen, entre otros:

- la copia total o parcial de una práctica, informe, o cualquier otra actividad de evaluación;
- dejar copiar;
- presentar un trabajo de grupo no hecho íntegramente por los miembros del grupo;
- presentar como propios materiales elaborados por un tercero, aunque sean traducciones o adaptaciones, y en general trabajos con elementos no originales y exclusivos del estudiante;
- tener dispositivos de comunicación (como teléfonos móviles, smart watches, etc.) accesibles durante las pruebas de evaluación teórico-prácticas individuales (exámenes).

En estos casos, la nota numérica del expediente será el valor menor entre 3.0 y la media ponderada de las notas (y por tanto no será posible el aprobado por compensación).
En la evaluación de las entregas de problemas y prácticas se utilizarán herramientas de detección de copia del código del programa.

Nota sobre la planificación de las actividades de evaluación:
Las fechas de evaluación continua y entrega de trabajos se publicarán en el principio de curso y pueden estar sujetos a cambios de programación por motivos de adaptación a posibles incidencias. Siempre se informará a Caronte sobre estos cambios ya que se entiende que esta es la plataforma habitual de intercambio de información entre profesores y estudiantes.

Evaluación única:

Esta asignatura no preveé el sistema de evaluación única.

Uso de la IA:

Uso restringido: Para esta asignatura, se permite el uso de tecnologías de Inteligencia Artificial (IA) exclusivamente en labores de soporte, como la búsqueda bibliográfica o de información, la corrección de textos o las traducciones. El estudiante tendrá que identificar claramente qué partes han sido generadas con esta tecnología, especificar las herramientas utilizadas e incluir una reflexión crítica sobre cómo éstas han influido en el proceso y el resultado final de la actividad. La no transparencia del uso de la IA en esta actividad evaluable se considerará falta de honestidad académica y puede acarrear una penalización parcial o total en la nota de la actividad, o sanciones mayores en casos de gravedad.

Bibliografía

Problem solving with algorithms and data Structures using Python. Bradley N. Miller and David L. Ranum. Franklin, Beedle and associates, 2005.

Python Programming: an introduction to computer science. John Zelle. Franklin,Beedle and associates, 2004.

Data Structures and Algorithms in python. Michael T. Googrich, Roberto Tamassia, Michael H. Goldwasser. Ed. Wiley. 2013.

https://www.geeksforgeeks.org/python-programming-language/?ref=ghm

Software

Spyder (Anaconda)

Grupos e idiomas de la asignatura

La información proporcionada es provisional hasta el 30 de noviembre de 2025. A partir de esta fecha, podrá consultar el idioma de cada grupo a través de este enlace. Para acceder a la información, será necesario introducir el CÓDIGO de la asignatura

Nombre	Grupo	Idioma	Semestre	Turno
(PAUL) Prácticas de aula	81	Catalán	primer cuatrimestre	manaña-mixto
(PAUL) Prácticas de aula	82	Catalán	primer cuatrimestre	manaña-mixto