FORMATO DE ENTREGA DE PRÁCTICAS Y RÚBRICA GENERAL

(Adaptado para la asignatura de Algorítmica y Complejidad, Curso 2024-25)

Introducción al Documento

El presente documento tiene como finalidad establecer una guía estructurada y detallada para la entrega de prácticas en la asignatura Algorítmica y Complejidad, correspondiente al curso 2024-25. Este formato busca garantizar que los estudiantes presenten sus ejercicios de manera coherente, organizada y alineada con los objetivos de la asignatura. Asimismo, se incluye una rúbrica general que permite evaluar el desempeño de los estudiantes en distintos aspectos de sus entregas, fomentando el desarrollo de habilidades técnicas, analíticas y de comunicación.

La propuesta también incorpora normas específicas sobre el estilo de programación permitido, penalizaciones por retrasos o incumplimientos, y pautas para evitar el plagio académico. Además, se facilita un ejemplo de código estructurado para guiar a los estudiantes en la implementación modular y funcional, promoviendo las mejores prácticas en programación.

Índice

Nota Importante

Este documento tiene como propósito complementar las indicaciones proporcionadas en la guía docente de la asignatura Algorítmica y Complejidad. No pretende sustituir ni contradecir las disposiciones oficiales contenidas en dicha guía. En caso de discrepancia, prevalecerán siempre las directrices establecidas en la guía docente oficial.

1. Formato de entrega de prácticas

1.1. Estructura general del documento

Cada práctica se debe entregar en un **documento principal** (PDF) y un **código fuente** anexo (dentro de un fichero ZIP), siguiendo **estas secciones**:

1. **Portada**
    - Nombre de la asignatura: *Algorítmica y Complejidad*.
    - Práctica: PL1, PL2, PL3, etc. (según corresponda).
    - Título de la práctica: p. ej., “Práctica de Análisis de Complejidad” o “Dividir y Vencerás”.
    - Nombre(s) y apellidos completos del/de la estudiante.
    - Fecha de entrega.
2. **Índice**
    - Indicar la página o sección donde se encuentra cada parte del documento.
    - Si la práctica es extensa, se recomienda incluir una tabla de contenidos detallada.
3. **Objetivos de la práctica**
    - Explicar brevemente qué se pretende con la práctica.
    - Relacionar los ejercicios con la teoría impartida en la asignatura.
4. **Resolución de los ejercicios / problemas**

    Para cada ejercicio / problema:

    #### 4.1. Descripción del ejercicio / problema
    - Incluir un enunciado breve (o completo, si es requerido) de cada ejercicio que se ha resuelto.
    - En caso de que la práctica permita elegir ejercicios, especificar claramente cuáles se han elegido.

    #### 4.2. Descripción del algoritmo
    - **Enfoque conceptual:** describir la idea principal de cada algoritmo, su estrategia (recursividad, divide y vencerás, voraz, etc.) y su justificación.

- Evitar la Programación Orientada a Objetos (POO) a menos que el profesor lo autorice expresamente; la norma general de la asignatura es **no** utilizar POO.

4.3 . Análisis de complejidad

- Explicar el razonamiento que lleva al orden de complejidad (Big-O).
- Si hay recursividad, mostrar la ecuación de recurrencia y su resolución (o al menos la estimación asintótica).

4.4. Casos de prueba y resultados

- Describir los **casos de prueba** (inputs, outputs) y qué se pretendía comprobar con cada uno.
- Mostrar los resultados obtenidos, preferentemente con **tests unitarios** (p. ej., doctest en Python).
- Incluir capturas de la ejecución o pantallazos que muestren el correcto funcionamiento.

4.5. Cumplimiento de instrucciones específicas

- Se evaluará el grado de cumplimiento de las instrucciones particulares proporcionadas por el profesor en la práctica.
- En caso de que el estudiante adopte un enfoque personal o alternativo y/o criterio, deberá justificarlo adecuadamente en relación con los objetivos de la práctica, del tema y del curso.
- **Recomendación:** Revisar detalladamente las instrucciones y criterios específicos indicados para cada práctica antes de iniciar el desarrollo.

5. Conclusiones

Reflexión sobre lo aprendido, dificultades superadas y posibles extensiones.

6. Bibliografía y referencias

Citar libros, artículos, sitios web o cualquier fuente consultada. Seguir un estilo coherente de citación (APA o IEEE).

1.2. Código fuente

- El código fuente se incluirá en el **ZIP** con el formato de nombre:

  Apellidos.Nombre_PLX.zip

(por ejemplo, GomezGarcia.Maria_PL1.zip).

- Se deberá **evitar** la Programación Orientada a Objetos. Se recomienda un estilo **funcional o imperativo** con funciones y módulos independientes.
- Incluir **docstrings** y comentarios en cada función o procedimiento.
- Mantener un estilo de código limpio y coherente.
- No emplear librerías externas que resuelvan de forma directa el ejercicio, salvo autorización del profesor.

1.3. Entrega

- El modo de entrega será el que el profesor indique (Por lo general el Campus Virtual de la UAH).
- Debe subirse un único fichero .zip que contenga:
    1. El **documento principal** (informe) en PDF.
    2. Una **carpeta con el código fuente** (organizado por ejercicios o módulos).

2. Rúbrica general de evaluación

A continuación, se presenta una **rúbrica genérica** aplicable a la mayoría de las prácticas. Podrá ajustarse según las especificaciones de cada entrega, pero mantiene una estructura base:

Criterio	Descripción	Ponderación
1. Entrega y Formato (Estructura)	Se valora el cumplimiento estricto de las normas de entrega: - Portada con datos correctos - Informe en PDF - Código en carpeta separada - Nombre correcto del fichero ZIP	10%
2. Claridad en la Descripción de los Algoritmos	- Explicación clara de la estrategia y de la técnica de diseño. - Evitar meros "copiados" de pseudocódigo; se valora la comprensión del método.	10%
3. Análisis de Complejidad	- Justificación adecuada del orden de complejidad. - Uso de notación Big-O, Big-Theta, etc. - Explicación de la resolución de la recurrencia (si existe).	10%
4. Casos de Prueba y Resultados	- Selección de casos de prueba representativos. - Tests unitarios (si aplica). - Presentación clara de inputs/outputs y resultados intermedios.	20%

Criterio	Descripción	Ponderación
5. Calidad del Código (Implementación)	- Código funcional, sin POO. - Bien estructurado y comentado (docstrings y #comentarios). - Cumplimiento de la normativa (no usar librerías externas, salvo autorización).	20%
6. Cumplimiento de las instrucciones específicas de los ejercicios de la práctica	- Evaluación del cumplimiento de las instrucciones y/o criterios particulares dadas por el profesor para la práctica. - Si el estudiante adopta un criterio personal distinto, debe justificar su decisión en función de los objetivos de la práctica, del tema y de las competencias del curso. **Ejemplo de evaluación:** - Instrucciones cumplidas sin desviaciones: 100%. - Desviaciones justificadas: 70-90%. - Instrucciones ignoradas o no justificadas: 0-50%.	20%
7. Conclusiones y Presentación Final	- Reflexión personal, conclusiones de lo aprendido. - Estructura clara, ortografía y redacción adecuada.	10%

***Nota 1**: Las ponderaciones son orientativas y pueden modificarse según lo que indique el profesor. Sin embargo, esta distribución mantiene una coherencia general para las prácticas durante el curso.*

***Nota 2:** La idea es que en un futuro la entrega de ejercicios contemple un 50% o 60% de la nota y un control que abarque el resto. El control se realizaría en el tiempo que dure una práctica y consistiría en un par de ejercicios al estilo de los que el estudiante entrego en clase. También pudiera en un futuro aplicarse un control por práctica o cada 2 prácticas y reducir la ponderación de la entrega de ejercicios y su complejidad o rigurosidad al evaluar los ejercicios entregados.*

3. Normas específicas y observaciones importantes

1. **Prohibición de Programación Orientada a Objetos**
    - Según las normas de la asignatura, no se permite utilizar POO. La implementación debe ser lo más simple posible, preferiblemente en un estilo estructurado/funcional.
2. **Fechas de entrega**
    - Se establecerá una fecha límite para cada práctica.
    - Las entregas fuera de plazo se penalizarán de acuerdo con la normativa interna (por ejemplo, máximo 5 días de retraso con un 50% de penalización).
3. **Plagio o fraude académico**
    - Cualquier indicio de plagio o copia supondrá la calificación de 0 en la práctica.

- Se recomienda trabajar de manera individual (o en grupo, si está autorizado) y consultar dudas con el profesor para evitar problemas en este sentido.

4. **Lenguaje de programación**
   - Por defecto, se utilizará **Python 3.x**.
5. **Uso de librerías**
   - No se permite usar librerías que solucionen automáticamente la funcionalidad que el alumno debe implementar (p. ej., librerías de ordenación avanzada).
   - Están permitidas librerías básicas como sys, math o similares, salvo que se indique lo contrario.
6. **Formato del informe**
   - Se valorará la calidad del documento (PDF), la presentación de resultados y la coherencia de la estructura.
   - No se evaluarán favorablemente informes que se limiten a un "copiar y pegar" de código sin explicaciones.
   - Ver Formato de entrega de prácticas y rubrica general de evaluación.
7. **Evaluación continua**
   - La calificación de cada práctica formará parte de la nota final de la asignatura.
   - Un código funcional **no garantiza** la nota máxima si el informe (memoria) no cumple con los requisitos establecidos. Asimismo, entregar el informe sin que el código funcione correctamente no asegura el aprobado. La no entrega del informe, independientemente del estado del código, implica la suspensión automática de la práctica.