Trabajo Final para la Cátedra: Estructuras de Datos en Python

Pautas para el Trabajo:

• Los trabajos se desarrollarán en grupos de 2 o 3 personas.
• La conformación del grupo, se deberá respetar hasta el final.
• La fecha de entrega del trabajo es la definida por el Docente. Grupo que no presenta en esa fecha, se le bajarán puntos.

Sobre la Nota Final:

• La nota final del trabajo se evaluará teniendo en cuenta varios aspectos clave. En primer lugar, se considerará la entrega en tiempo y forma. Esto implica que el trabajo debe ser entregado dentro del plazo estipulado y en el formato adecuado, con todos los archivos y documentación solicitados. Cualquier retraso será penalizado de acuerdo con las reglas establecidas previamente.
• Otro criterio importante es la resolución del problema. Se evaluará la correctitud de la solución implementada, valorando el uso adecuado de las estructuras de datos y algoritmos enseñados en clase. Además, se tendrá en cuenta la eficiencia de la implementación, prestando especial atención a la complejidad computacional y el manejo de casos excepcionales o borde.
• La documentación complementaria también tendrá un peso significativo en la evaluación. El material adicional presentado, como presentaciones en PowerPoint o videos explicativos, será analizado en términos de su calidad y claridad. Se espera que estos documentos expongan de manera coherente el problema y las soluciones adoptadas, utilizando gráficos o tablas para respaldar la explicación cuando sea necesario. La documentación debe además justificar las decisiones de diseño y demostrar cómo la solución cumple con los requisitos planteados.
• Por último, la defensa individual de cada estudiante jugará un rol crucial en la nota final. Durante esta etapa, se evaluará la capacidad de cada uno para defender y justificar su solución, mostrando un conocimiento profundo del código implementado y de los conceptos aplicados. También se valorará la habilidad para responder preguntas técnicas relacionadas con la solución, así como la claridad y seguridad en la exposición oral.
• Asimismo, se tendrá en cuenta el uso de buenas prácticas, como la correcta documentación del código, el cumplimiento de convenciones de programación y el manejo de versiones, si corresponde.
• Cualquier duda, consulta deberá coordinarse con los Docentes de la Materia.

> **Importante:**
>
> - Cada entrega parcial será evaluada bajo estos mismos criterios. La calidad y entrega de los materiales adicionales (presentaciones, infografías, videos, síntesis) es obligatoria en cada etapa, no solo en la entrega final.
> - La defensa oral del trabajo será únicamente en la última entrega.
> - Todo el material (código, documentación y adicionales) debe ser subido a un repositorio de GitHub, cuyo enlace se entregará en cada entrega.

Trabajo Práctico Integrador – Personajes de Ciencia Ficción y Estructuras de Datos en Python

Hilo Conductor: "Universo de Personajes de Ciencia Ficción"

Modalidad y Entregas

- Trabajo grupal (2 o 3 integrantes)
- El trabajo se desarrolla en **4 entregas parciales** (una cada tres semanas)
- Cada entrega debe incluir: desarrollo teórico, implementación en Python, documentación y material adicional (presentación, video, infografía, etc.)
- Defensa oral individual y grupal al finalizar

Objetivo General

Integrar y aplicar los conceptos de estructuras de datos y análisis de algoritmos en el contexto de la gestión, análisis y simulación de personajes de ciencia ficción, desarrollando tanto la fundamentación teórica como la implementación práctica.

Entrega 1: Modelado y Encapsulamiento (Unidades 1 y 2)

Teoría

- Explicar el concepto de encapsulamiento y su importancia en la programación orientada a objetos.
- Justificar la elección de atributos y métodos para modelar personajes de ciencia ficción.
- Analizar ventajas y desventajas de distintas representaciones de listas y pilas para gestionar inventarios y poderes.

Práctica

- Definir la clase `Personaje` con atributos: nombre, especie, planeta, habilidades, inventario, nivel de poder, etc.
- Implementar interfaces para agregar/eliminar habilidades y objetos del inventario.
- Implementar una estructura recursiva para calcular el poder total de un personaje considerando transformaciones o fusiones.

Adicional obligatorio

- Presentación (diapositivas o video corto) explicando el diseño de la clase y la lógica de encapsulamiento.

Entrega 2: Árboles y Jerarquías (Unidades 3 y 4)

Teoría

- Explicar la diferencia entre árboles binarios y árboles generales. Ejemplificar con jerarquías de personajes y evolución de habilidades.
- Analizar la eficiencia de los recorridos y búsquedas en cada tipo de árbol.

Práctica

- Implementar un árbol binario para organizar personajes según nivel de poder.
- Modelar el árbol de habilidades de un personaje como un árbol general (cada nodo es una habilidad, ramas son mejoras o transformaciones).
- Implementar recorridos (preorden, inorden, postorden) y búsquedas en ambos árboles.

Adicional obligatorio

- Infografía o póster digital mostrando la jerarquía de personajes y el árbol de habilidades de al menos un personaje.

Entrega 3: Prioridades, Grafos y Algoritmos (Unidades 5, 6, 7, 8)

Teoría

- Explicar el concepto de cola de prioridades y heap binaria. Justificar su uso en la gestión de misiones o combates.
- Describir la representación de grafos y su aplicación en universos de ciencia ficción (planetas, alianzas, rutas espaciales).
- Analizar la complejidad de los algoritmos de recorrido (DFS, BFS) y su aplicabilidad en la búsqueda de aliados o enemigos.

Práctica

- Implementar una cola de prioridades para gestionar misiones o combates (prioridad por nivel de amenaza o recompensa).
- Modelar el universo como un grafo: nodos (planetas, bases, naves), aristas (rutas, alianzas, conflictos).
- Implementar DFS y BFS para encontrar rutas o alianzas entre personajes o planetas.

Adicional obligatorio

- Video explicativo (máx. 5 minutos) mostrando el funcionamiento de la cola de prioridades y los recorridos en el grafo.

Entrega 4: Ordenamiento, Problemas NP y Síntesis Final (Unidades 9 y 10)

Teoría

- Explicar el ordenamiento topológico y su utilidad en la planificación de entrenamientos, misiones o evolución de personajes.

- Analizar el problema del camino mínimo y su relevancia en la optimización de recursos o desplazamientos en el universo.
- Reflexionar sobre la eficiencia de las estructuras y algoritmos implementados a lo largo del trabajo.

Práctica

- Implementar ordenamiento topológico para planificar la secuencia de entrenamiento o evolución de habilidades.
- Aplicar el algoritmo de Dijkstra para encontrar el camino óptimo entre dos planetas o bases.
- Realizar un análisis de eficiencia de las principales operaciones implementadas.

Adicional obligatorio

- Documento de síntesis (PDF) que integre: fundamentación teórica, decisiones de diseño, análisis de eficiencia, capturas de pantalla del código y resultados, y reflexión grupal sobre el aprendizaje.

Evaluación

- Cumplimiento y calidad de cada entrega parcial
- Profundidad y claridad en la fundamentación teórica
- Correctitud, eficiencia y documentación del código
- Calidad de los materiales adicionales
- Defensa oral individual y grupal

Recomendaciones

- Utilizar bibliografía y recursos audiovisuales sugeridos en el programa de la materia
- Citar fuentes y justificar todas las decisiones de diseño
- Mantener un registro de versiones y cambios en el código
- Consultar dudas con el equipo docente con anticipación

Este trabajo práctico busca integrar teoría y práctica, promoviendo la creatividad y el análisis crítico en el contexto de la ciencia ficción y la informática.