UNIVERSIDAD NACIONAL GUILLERMO BROWN

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.**

UNIVERSIDAD NACIONAL GUILLERMO BROWN

Tema Principal: El grupo deberá definir un tema.

Por ejemplo: **Análisis de Datos Médicos**

El trabajo final se dividirá en dos partes: una parte teórica-práctica y una parte de codificación. El objetivo será analizar un conjunto de datos médicos reales o simulados utilizando las estructuras de datos vistas en clase para resolver problemas de organización, consulta y análisis de datos clínicos.

Requerimientos:

- **Lenguaje**: Python
- **Problema a Resolver (Ejemplo):** "Análisis y Gestión de Datos Médicos" (pueden ser datos de pacientes, diagnósticos, tratamientos, etc.).

Parte 1: Teoría y Realización Práctica

1. **Desarrollo de Clases e Interfaces (Unidad 1)**:
    - Define una clase Paciente que encapsule la información médica relevante (nombre, edad, historial de enfermedades, medicamentos, etc.).
    - Crea interfaces para la gestión de los datos médicos, permitiendo añadir, eliminar, y modificar información.
2. **Estructuras Recursivas (Unidad 2)**:
    - Implementa un algoritmo recursivo que permita buscar en el historial de tratamientos de un paciente la aparición de ciertas enfermedades o medicamentos clave.
3. **Árboles Binarios (Unidad 3)**:
    - Usa un árbol binario de búsqueda para organizar los pacientes de una clínica por su número de identificación médica (ID). Debes permitir inserciones, eliminaciones, y búsquedas eficientes.
4. **Árboles Generales (Unidad 4)**:
    - Modela el historial clínico de cada paciente como un árbol general donde cada nodo es un evento médico (consulta, diagnóstico, tratamiento) y las ramas conectan las visitas y tratamientos asociados.
5. **Cola de Prioridades y Heap Binaria (Unidad 5)**:
    - Utiliza una cola de prioridades para gestionar la urgencia de los pacientes. Los pacientes más críticos deben ser atendidos primero, basándote en la gravedad de su condición.
6. **Análisis de Algoritmos (Unidad 6)**:

- Analiza la eficiencia de los algoritmos utilizados (inserción en árbol binario, recorrido en árbol general, búsqueda en cola de prioridad) en términos de tiempo y espacio. Asegúrate de explicar la complejidad de cada uno.

Parte 2: Codificación y Algoritmos

1. **Grafos (Unidad 7)**:
   - Modela la red de hospitales y clínicas como un grafo. Los nodos representan hospitales y las aristas las conexiones entre ellos (distancias o tiempos de transferencia de pacientes).
2. **Recorridos DFS y BFS (Unidad 8)**:
   - Implementa un algoritmo DFS y BFS para encontrar el camino más corto entre hospitales para transferir un paciente en caso de emergencia.
3. **Ordenamiento Topológico (Unidad 9)**:
   - Utiliza ordenamiento topológico para modelar la secuencia de pasos necesarios para diagnosticar una enfermedad en base a los síntomas. Algunos pasos requieren la realización de pruebas antes de avanzar a la siguiente fase del diagnóstico.
4. **Problemas NP y Camino Mínimo (Unidad 10)**:
   - Analiza el problema del camino mínimo entre varios hospitales utilizando el algoritmo de Dijkstra para encontrar la mejor ruta para una ambulancia que necesita trasladar a un paciente crítico.

Entregables:

1. **Informe Teórico-Práctico**: Explicación de las decisiones de diseño, análisis de complejidad y justificación de las estructuras de datos utilizadas para resolver los problemas.
2. **Código en Python**: Implementación de las soluciones propuestas. Se evaluará la claridad, eficiencia y adherencia a buenas prácticas de programación.
3. **Conclusión**: Reflexión sobre cómo el uso de estructuras de datos optimizadas impacta el análisis y procesamiento eficiente de grandes volúmenes de datos médicos.

Fecha de entrega:

- A convenir según el calendario académico.

Este trabajo final permitirá integrar todos los conceptos de la materia en un contexto práctico y relevante como el análisis y gestión de datos médicos.

UNIVERSIDAD NACIONAL GUILLERMO BROWN

Otro Ejemplo

Trabajo Final para la Cátedra Estructuras de Datos en Python

Tema Principal: Desarrollo de un Juego de Dragon Ball

Este trabajo final también se dividirá en una parte teórica-práctica y una parte de codificación. El objetivo será desarrollar un sistema de juego basado en el universo de **Dragon Ball**, donde se gestionen personajes, combates, niveles de poder, y habilidades, utilizando las estructuras de datos vistas en clase.

Requerimientos:

- **Lenguaje**: Python
- **Problema a Resolver**: Creación y gestión de personajes, batallas y evolución de poderes en un juego basado en Dragon Ball.

Parte 1: Teoría y Realización Práctica

1. **Desarrollo de Clases e Interfaces (Unidad 1)**:
    - Define una clase Personaje que encapsule atributos como nombre, nivel de poder, habilidades, raza (Saiyajin, Namekuseijin, Terrícola, etc.).
    - Crea interfaces que permitan gestionar combates, subir de nivel y adquirir nuevas habilidades.
2. **Estructuras Recursivas (Unidad 2)**:
    - Implementa un algoritmo recursivo que calcule la evolución de poder de un personaje tras cada combate, teniendo en cuenta multiplicadores como las transformaciones (Super Saiyajin, Kaioken, etc.).
3. **Árboles Binarios (Unidad 3)**:
    - Utiliza un árbol binario para organizar los personajes según su nivel de poder, permitiendo buscar rápidamente los personajes más fuertes para los enfrentamientos.
4. **Árboles Generales (Unidad 4)**:
    - Modela el árbol de habilidades de un personaje como un árbol general, donde cada nodo representa una técnica y las ramas las transformaciones o mejoras derivadas de cada habilidad (por ejemplo, Kamehameha -> Kamehameha x10).
5. **Cola de Prioridades y Heap Binaria (Unidad 5)**:
    - Implementa una cola de prioridades para gestionar los combates en un torneo. Los personajes con el mayor nivel de poder tendrán prioridad en los enfrentamientos.
6. **Análisis de Algoritmos (Unidad 6)**:

- Analiza la eficiencia de los algoritmos utilizados para las batallas, la evolución de poder y la organización de los personajes. Discute la complejidad en términos de tiempo y espacio para cada estructura de datos.

Parte 2: Codificación y Algoritmos

1. **Grafos (Unidad 7)**:
   - Crea un grafo que represente el universo de Dragon Ball, donde los nodos son planetas (Tierra, Namek, Vegeta) y las aristas son rutas espaciales entre ellos. Los personajes deben poder viajar entre planetas para entrenar o luchar.
2. **Recorridos DFS y BFS (Unidad 8)**:
   - Implementa un algoritmo DFS y BFS para encontrar el camino más rápido entre planetas, buscando a personajes que han desaparecido (como cuando Goku viaja por el espacio para entrenar).
3. **Ordenamiento Topológico (Unidad 9)**:
   - Usa el ordenamiento topológico para planificar las etapas de entrenamiento de un personaje. Algunas habilidades requieren dominar otras técnicas antes de ser desbloqueadas, siguiendo una jerarquía.
4. **Problemas NP y Camino Mínimo (Unidad 10)**:
   - Aplica el problema del camino mínimo utilizando Dijkstra para encontrar la mejor ruta en el mapa del universo de Dragon Ball para recolectar las Esferas del Dragón lo más rápido posible.

Entregables:

1. **Informe Teórico-Práctico**: Explicación de las decisiones de diseño, análisis de complejidad y justificación de las estructuras de datos utilizadas en la creación del juego.
2. **Código en Python**: Implementación de las mecánicas del juego, incluyendo la gestión de personajes, batallas, y desplazamientos entre planetas.
3. **Conclusión**: Reflexión sobre cómo las estructuras de datos eficaces contribuyen a la creación de un juego funcional y entretenido en un universo como el de Dragon Ball.

Fecha de entrega:

- A definir según el calendario académico.

Este trabajo final permitirá integrar los conceptos de la materia en un contexto creativo y lúdico, explorando cómo las estructuras de datos apoyan el desarrollo de un juego basado en Dragon Ball.

UNIVERSIDAD NACIONAL GUILLERMO BROWN

Otro Ejemplo

Trabajo Final para la Cátedra: Estructuras de Datos en Python

Tema Principal: Manejo de una Plataforma de Streaming

El trabajo final se dividirá en una parte teórica-práctica y otra de codificación. El objetivo será diseñar y desarrollar la gestión de contenido, usuarios y recomendaciones dentro de una plataforma de streaming utilizando las estructuras de datos vistas en clase.

Requerimientos:

- **Lenguaje**: Python
- **Problema a Resolver**: Gestión de usuarios, contenido y recomendaciones en una plataforma de streaming de videos y series.

Parte 1: Teoría y Realización Práctica

1. **Desarrollo de Clases e Interfaces (Unidad 1)**:
    - Define una clase Usuario que encapsule la información básica de un usuario (nombre, edad, preferencias, historial de visualización).
    - Define otra clase Contenido para gestionar películas, series, episodios, duración, género, y su popularidad. Crea interfaces que permitan añadir contenido nuevo, gestionar usuarios, y su interacción con la plataforma.
2. **Estructuras Recursivas (Unidad 2)**:
    - Implementa un algoritmo recursivo que permita a los usuarios buscar y clasificar el contenido según sus preferencias y visualizaciones previas, generando recomendaciones personalizadas.
3. **Árboles Binarios (Unidad 3)**:
    - Utiliza un árbol binario de búsqueda para organizar el catálogo de contenido de la plataforma según popularidad (número de visualizaciones). El objetivo es permitir búsquedas rápidas de contenido más popular.
4. **Árboles Generales (Unidad 4)**:
    - Modela el catálogo de series como un árbol general, donde cada nodo representa una serie y sus ramas representan las temporadas y episodios. Los usuarios podrán recorrer este árbol para navegar entre episodios y temporadas.
5. **Cola de Prioridades y Heap Binaria (Unidad 5)**:
    - Implementa una cola de prioridades para gestionar las listas de reproducción o “watchlist” de los usuarios, priorizando el contenido más popular o reciente para recomendar.
6. **Análisis de Algoritmos (Unidad 6)**:

- Analiza la eficiencia de los algoritmos implementados, desde la organización del catálogo hasta la generación de recomendaciones. Evalúa la complejidad temporal y espacial de cada operación.

Parte 2: Codificación y Algoritmos

1. **Grafos (Unidad 7)**:
   - Modela las relaciones entre el contenido mediante un grafo, donde los nodos son películas o series, y las aristas representan similitudes (por género, actores, o preferencia de usuarios). Utiliza este grafo para generar recomendaciones basadas en lo que otros usuarios con gustos similares han visto.
2. **Recorridos DFS y BFS (Unidad 8)**:
   - Implementa un recorrido DFS y BFS sobre el grafo de contenido para encontrar películas o series similares a las que un usuario ha visto. El DFS podría explorar a fondo un tipo de contenido específico, mientras que el BFS explorará de manera más amplia.
3. **Ordenamiento Topológico (Unidad 9)**:
   - Usa ordenamiento topológico para gestionar la carga y visualización del contenido según restricciones de disponibilidad (por ejemplo, lanzamientos secuenciales de episodios o temporadas).
4. **Problemas NP y Camino Mínimo (Unidad 10)**:
   - Implementa el algoritmo de Dijkstra para encontrar la mejor secuencia de reproducción de contenido que maximice la satisfacción del usuario, teniendo en cuenta restricciones como tiempo disponible y preferencias.

Entregables:

1. **Informe Teórico-Práctico**: Explicación de las decisiones de diseño, análisis de complejidad, y justificación de las estructuras de datos utilizadas para optimizar la plataforma de streaming.
2. **Código en Python**: Implementación de las funcionalidades clave del sistema, incluyendo gestión de usuarios, catálogo, y generación de recomendaciones.
3. **Conclusión**: Reflexión sobre cómo el uso de estructuras de datos mejora la eficiencia y personalización de la experiencia en una plataforma de streaming.

Fecha de entrega:

- A convenir según el calendario académico.

Este trabajo final permitirá aplicar los conceptos de la materia a un entorno actual y práctico como lo es la gestión de una plataforma de streaming, integrando algoritmos y estructuras de datos para optimizar la experiencia del usuario.