Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 2do. Semestre 2024

PROYECTO 2 – Simulación de ensamblaje de máquinas

201701010 – Bryant Herrera Rubio

Resumen

El proyecto presentado desarrolla un sistema de simulación para una línea de ensamblaje automatizada, utilizando estructuras de datos enlazadas para gestionar máquinas, productos y acciones a lo largo del tiempo. La máquina, con varias líneas de producción y componentes, sigue instrucciones detalladas para ensamblar productos en diferentes etapas. Cada línea de ensamblaje se mueve y realiza acciones de ensamblaje en secuencias definidas por instrucciones específicas, registrando el tiempo total de cada proceso.

El código está dividido en módulos que representan elementos clave como productos, posiciones, acciones y segundos, todos gestionados mediante nodos en listas enlazadas. Se utiliza una estructura que permite almacenar y actualizar posiciones de ensamblaje, junto con una simulación del proceso de ensamblaje, calculando el tiempo total requerido.

Palabras clave

- API
- Framework
- Linked Lists
- Simulation
- Assembly Line

Abstract

The presented project develops a simulation system for an automated assembly line, utilizing linked data structures to manage machines, products, and actions over time. The machine, with multiple production lines and components, follows detailed instructions to assemble products in various stages. Each assembly line moves and performs assembly actions in sequences defined by specific instructions, recording the total time of each process.

The code is divided into modules representing key elements such as products, positions, actions, and seconds, all managed through nodes in linked lists. A structure is used to store and update assembly positions, along with a simulation of the assembly process, calculating the total time required.

Keywords

- *API*
- *Framework*
- *Linked Lists*
- *Simulation*
- *Assembly Line*

Introducción

En la actualidad, la simulación de procesos industriales es una herramienta fundamental para optimizar la producción y reducir costos. Este proyecto tiene como objetivo desarrollar un simulador de líneas de ensamblaje, centrado en la representación de máquinas, productos y las acciones necesarias para su elaboración. Utilizando estructuras de datos como listas enlazadas, se modelan los movimientos y tiempos de ensamblaje para distintos productos, proporcionando una representación precisa del proceso productivo. A través del uso de una API y frameworks de simulación, se busca analizar y mejorar la eficiencia de las líneas de producción, evaluando factores como el tiempo total de ensamblaje y el uso de componentes en cada máquina.

Desarrollo del tema

Para este proyecto, se utilizó Python como el lenguaje base, dado que es ampliamente reconocido por su simplicidad y flexibilidad, lo que facilita el manejo de estructuras de datos como listas enlazadas y permite una implementación clara de simulaciones. Además, se empleó el framework Flask para el desarrollo de la API, dado que ofrece un enfoque minimalista para la creación de aplicaciones web. Flask resulta muy ventajoso para este tipo de proyectos debido a su ligereza y su facilidad para crear rutas y manejar solicitudes HTTP.

Ventajas y desventajas de Flask

Una de las principales ventajas de Flask es su sencillez. Permite desarrollar aplicaciones web de manera rápida, manteniendo un código claro y conciso, lo cual fue crucial para la implementación de la simulación de la línea de ensamblaje. Además, su gran extensibilidad mediante paquetes adicionales lo convierte en una opción muy flexible. En el caso de este proyecto, se utilizó junto con Graphviz para generar diagramas y gráficos de los procesos de ensamblaje, lo que permitió visualizar el flujo de producción y evaluar posibles mejoras.

Sin embargo, al tratarse de un framework minimalista, Flask no cuenta con funcionalidades preconfiguradas, como las que ofrecen frameworks más robustos como Django. Esto puede ser una desventaja si el proyecto requiere un alto nivel de escalabilidad o gestión más avanzada de bases de datos, usuarios, o seguridad, ya que estas características requieren de más configuración manual en Flask.

APIs utilizadas

El proyecto utilizó varias APIs internas para gestionar el flujo de datos entre el backend y el frontend. Una de las principales APIs es la encargada de cargar y procesar archivos XML, donde se detallan las características de las máquinas y productos a ensamblar. Esta API extrae la información del archivo y la estructura en listas enlazadas, lo que permite manejar múltiples productos y máquinas de manera eficiente.

Otra API relevante es la que calcula el tiempo total de ensamblaje. A través de esta API, el usuario selecciona un producto y el sistema recorre cada una de las instrucciones de ensamblaje del producto, calculando los tiempos involucrados en mover el brazo de ensamblaje y ensamblar cada componente. Esta API devuelve tanto el tiempo total como un gráfico que ilustra el proceso de ensamblaje paso a paso.

El uso de estas APIs facilita la comunicación entre el frontend y el backend, ya que proporcionan endpoints claros para cada función del sistema. Esto asegura que los usuarios puedan interactuar de manera fluida con el simulador, y que los datos se procesen en el servidor sin exponer la lógica interna del sistema.

Diseño HTML y Frontend

En cuanto al diseño del frontend, se utilizó HTML, CSS y JavaScript para crear una interfaz sencilla e intuitiva. A través de la interfaz, el usuario puede cargar archivos XML, seleccionar máquinas y productos, y visualizar los resultados del proceso de ensamblaje. El uso de JavaScript permite que la interacción sea dinámica, actualizando la información en la página sin necesidad de recargarla completamente, lo que mejora la experiencia del usuario.

Además, el sistema permite generar gráficos interactivos del proceso de ensamblaje, mostrando visualmente cómo se mueve el brazo de ensamblaje de una posición a otra y cómo se ensamblan los distintos componentes. Estos gráficos fueron generados con la ayuda de Graphviz, integrado con Flask, lo que permite generar imágenes de manera dinámica en el backend y mostrarlas en el frontend.

Backend

El backend es el núcleo del sistema, donde se procesa toda la lógica de la simulación. Se gestionan las listas enlazadas que representan las posiciones del brazo de ensamblaje, las acciones en cada segundo del proceso, y los productos que son ensamblados por cada máquina. Cada uno de estos elementos está encapsulado en su propia clase, como Lista_Productos, Lista_Posicion y Lista_Maquinas, lo que facilita la manipulación de los datos de manera modular y clara.

Además, se implementó una API para simular el ensamblaje. Esta API permite ejecutar el proceso paso a paso, moviendo el brazo de ensamblaje y ensamblando los componentes uno a la vez, mientras calcula el tiempo total requerido para completar el producto. Los resultados se devuelven tanto en texto como en forma de gráficos.

Funciones importantes del proyecto

1. **cargar_archivo(ruta)**: Esta función carga un archivo XML que contiene la descripción de las máquinas y productos, y lo procesa para generar listas enlazadas que representan las máquinas, productos y sus instrucciones de ensamblaje.
2. **calcular_tiempo_ensamblaje(maquina):** Calcula el tiempo total de ensamblaje de un producto específico dentro de una máquina. A través de este cálculo, se evalúan los tiempos de movimiento del brazo y los tiempos de ensamblaje de cada componente.
3. **generar_grafico(producto, maquina, instrucciones, ruta_salida):** Esta función genera un gráfico en formato PNG que visualiza el proceso de ensamblaje, conectando cada paso del ensamblaje con flechas que representan el movimiento y ensamblaje de componentes.
4. **simular_ensamblaje(maquinas):** Ejecuta la simulación del ensamblaje de múltiples máquinas al mismo tiempo. Calcula el tiempo que tardan en ensamblar productos y mueve los brazos de ensamblaje en función de las instrucciones de cada producto.
5. **Lista_Posicion y Lista_Productos:** Son estructuras de datos esenciales que gestionan las posiciones del brazo de ensamblaje y la lista de productos respectivamente. Facilitan la manipulación eficiente de las instrucciones de ensamblaje.

Uso de listas enlazadas y su comparación con las estructuras de datos nativas de Python

En este proyecto, se utilizó extensivamente una estructura de datos particular: las listas enlazadas. Estas listas permiten una mayor flexibilidad en la manipulación de los datos, especialmente cuando se trata de simulaciones dinámicas como una línea de ensamblaje. En una lista enlazada, cada elemento (o nodo) contiene un valor y una referencia al siguiente nodo de la lista, lo que permite una inserción y

eliminación eficiente de elementos en cualquier posición de la lista sin necesidad de reorganizar los elementos.

Ventajas de las listas enlazadas

Inserción y eliminación rápida: A diferencia de las listas nativas de Python, en las que insertar o eliminar elementos en una posición intermedia puede ser costoso, en una lista enlazada solo es necesario ajustar los punteros de los nodos antes y después de la posición afectada. Esto hace que las listas enlazadas sean más eficientes para este tipo de operaciones, especialmente cuando se trata de manejar grandes volúmenes de datos o cambios frecuentes en la estructura.

Uso de memoria dinámica: Las listas enlazadas solo reservan memoria para los nodos que están en uso, lo que puede ser una ventaja en situaciones donde la cantidad de datos varía constantemente. Las listas nativas de Python, por otro lado, reservan bloques de memoria contiguos, lo que puede provocar sobrecarga de memoria si se requiere redimensionar el bloque.

Acceso secuencial: Aunque las listas enlazadas permiten una inserción y eliminación eficiente, su acceso a los elementos es secuencial. Esto significa que, para acceder a un nodo en una posición específica, es necesario recorrer todos los nodos anteriores, lo que resulta en un acceso más lento en comparación con las listas de Python.

Comparación con listas nativas, diccionarios y tuplas

Listas nativas: Las listas nativas de Python son muy flexibles, permitiendo el acceso directo a cualquier elemento en una posición dada en tiempo constante (O(1)). Sin embargo, cuando se trata de inserciones o eliminaciones frecuentes en posiciones intermedias, las listas enlazadas son más eficientes, ya que no requieren un desplazamiento de elementos. Además, las listas nativas tienden a ser menos eficientes en el manejo de datos dinámicos y no ofrecen la misma flexibilidad en cuanto a la manipulación directa de los enlaces entre elementos.

Diccionarios: Los diccionarios en Python ofrecen un acceso extremadamente rápido a los valores a través de claves, generalmente en tiempo O(1), lo que es útil para operaciones donde el acceso rápido a los datos es prioritario. Sin embargo, no son tan útiles para mantener un orden específico de los elementos o para realizar operaciones que dependan de un orden secuencial, como ocurre en este proyecto con los tiempos de ensamblaje y las posiciones del brazo. Las listas enlazadas son más adecuadas para estructuras donde el orden de los elementos y su manipulación secuencial son clave, como en la simulación de líneas de ensamblaje.

Tuplas: Las tuplas son estructuras de datos inmutables en Python, lo que significa que no pueden ser modificadas después de su creación. Son ideales para almacenar datos que no cambiarán a lo largo del programa. Sin embargo, para un proyecto como este, donde la simulación requiere constantes actualizaciones de las posiciones de los brazos de ensamblaje, las tuplas no son adecuadas. Las listas enlazadas, al ser mutables, ofrecen la flexibilidad necesaria para manejar cambios en el tiempo y las posiciones de los brazos de ensamblaje.

Aplicación de las listas enlazadas en este proyecto
En este proyecto, las listas enlazadas se utilizaron para manejar de forma dinámica las acciones por segundo, las posiciones del brazo y las máquinas ensambladoras. Las listas enlazadas permiten:

Agregar nuevos segundos al flujo de ensamblaje sin necesidad de reorganizar la estructura de datos completa.
Mantener un control flexible sobre las posiciones de los brazos, lo que facilita la simulación del movimiento a lo largo de los componentes.
Insertar o actualizar productos y máquinas en cualquier punto del proceso sin tener que

reorganizar toda la lista, lo que es esencial para una simulación en tiempo real donde las máquinas pueden añadir nuevos productos en cualquier momento.
Este uso específico de listas enlazadas, en conjunto con estructuras nativas como diccionarios y listas, permitió gestionar de manera eficiente el flujo de trabajo dentro del simulador, proporcionando un equilibrio adecuado entre la facilidad de acceso y la flexibilidad en la modificación de los datos.

Conclusiones

En resumen, este proyecto es un excelente ejemplo de cómo la simulación de procesos industriales puede ser manejada de manera eficiente mediante el uso de frameworks ligeros como Flask, junto con APIs internas bien definidas. Aunque Flask tiene algunas limitaciones en términos de escalabilidad, su simplicidad y flexibilidad fueron clave para el éxito de esta simulación de líneas de ensamblaje. La integración con herramientas de visualización como Graphviz y el uso de estructuras de datos adecuadas permitió crear un simulador que no solo optimiza el proceso de ensamblaje, sino que también ofrece una interfaz clara y funcional para los usuarios.

Referencias bibliográficas

1. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms (3rd ed.). MIT Press.

2. Goodrich, M. T., & Tamassia, R. (2014). Data Structures and Algorithms in Python. Wiley.

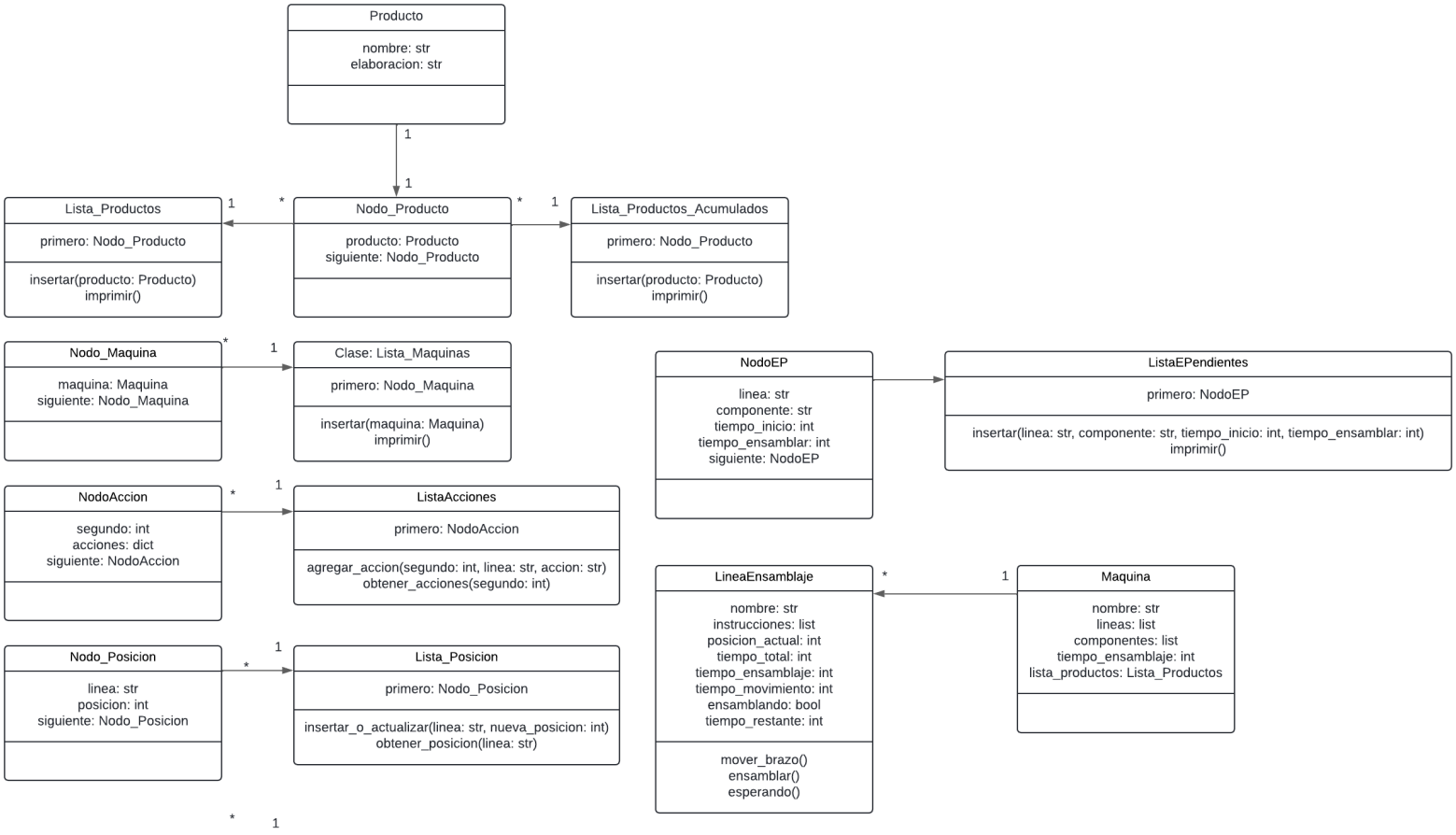
3. Cheng, Y., & Liu, Z. (2017). "Design and Implementation of a Simulation System Based on Python". International Journal of Software Engineering & Applications (IJSEA), 8(1), 1-12..

4. Van Rossum, G., & Drake, F. L. (2009). Python 3 Reference Manual. CreateSpace Independent Publishing Platform.

Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 2do. Semestre 2024

Anexos

Diagrama de clases:



Diagramas de actividades: