Proyecto Barrio Sostenible 🌱

Este proyecto busca optimizar la distribución de recursos energéticos en un barrio mediante el uso de grafos y algoritmos genéticos. El objetivo es minimizar las pérdidas energéticas y garantizar una cobertura eficiente del consumo eléctrico en las casas.

Se usa un algoritmo genético para:

1. Ubicar generadores eléctricos de manera óptima.
2. Distribuir paneles solares para maximizar la generación y cubrir las necesidades energéticas.

BarrioSustentable.txt

Contiene las clases que estructuran los datos de los nodos y las aristas del grafo.
Clases principales:
NodeData: Representa un nodo en el grafo (casas)

Atributos:
label: Etiqueta identificadora del nodo.
Pe_inst: Lista de potencias generadas instaladas.
Con_inst: Lista de consumos instalados (por hora).
SOC: Lista del estado de carga (SOC - State of Charge).
SR: Lista de estados booleanos (por ejemplo, estado activo/inactivo).
X, Y: Coordenadas espaciales del nodo.
Techo_m2: Área disponible en el techo para instalación de paneles solares.

EdgeData: Representa una arista, es decir, la conexión entre dos nodos en el grafo.

Atributos:
Pe: Potencia entrante.
Pc: Potencia consumida.
Z: Impedancia de la conexión.
D: Distancia entre los nodos.
R: Resistencia de la conexión.
C: Condición booleana asociada (por ejemplo, conexión activa o válida).

Algoritmo.py

Implementa un algoritmo genético para optimizar la ubicación de un generador eléctrico y la distribución de paneles solares en un barrio. El objetivo es minimizar la pérdida de energía debido a la distancia y garantizar una cobertura eficiente del consumo eléctrico de las casas.
Funciones:
-calcular_perdida_energia: Evalúa las pérdidas energéticas considerando la distancia entre el generador y las casas.
-inicializar_poblacion: Crea una población inicial de soluciones factibles con generadores y paneles solares distribuidos.
-verificar_restricciones: Garantiza que las soluciones cumplan con las restricciones físicas y de consumo.
-evaluar_poblacion: Calcula la aptitud de cada solución basada en la pérdida de energía.

-seleccionar_padres: Selecciona las mejores soluciones como padres para la próxima generación.
-cruzar y mutar: Generan nuevas soluciones combinando y alterando las soluciones existentes.

Proceso:
1. Genera soluciones iniciales basadas en restricciones del barrio.
2. Itera sobre generaciones para mejorar las soluciones mediante selección, cruce y mutación.
3. Identifica la mejor solución con la menor pérdida energética y una distribución óptima de paneles.

Resultados:
- Ubicación óptima del generador eléctrico.
- Distribución eficiente de paneles solares entre las casas.
- Visualización gráfica de la configuración final.

Aristas.txt y Aristas2.txt
Tiene datos para hacer pruebas de las conexiones entre los nodos (casas).
Casa.txt
Datos para pruebas de las casas.

ConsumoTotal.py
Cálculo de Consumo Total eléctrico total de cada casa en función de los electrodomésticos instalados y su uso. Para determinar la necesidad de energía de cada nodo en el grafo y optimizar la distribución de paneles solares.
calcular_consumo_total(casa, consumo_vector):
- Calcula el consumo total de una casa multiplicando el consumo por electrodoméstico (en kWh) por su patrón de uso.
- Lista de prueba de consumos promedio por electrodoméstico:
  [1048, 200, 5, 90, 113, 1225, 640, 2000, 100, 22, 1125, 800, 950].

Proporción de producción solar:
- Un panel solar produce un promedio de 2 kWh diarios.
- El modelo asume que hay 100 paneles solares disponibles para distribuir entre las casas, cada uno con dimensiones de 1x1.6 metros.

LeerTXT
Construye un grafo con nodos que representan casas y aristas que modelan conexiones energéticas a aprtir de datos de archivos externos.
read_nodes(file_path):
- Lee los datos de nodos desde un archivo especificado (`Casas.txt` o `Casa2.txt`).
read_edges(file_path):
- Lee las aristas desde un archivo especificado (`aristas.txt` o `Aristas2.txt`).
- Cada línea del archivo debe contener:
Se agregan los datos al grafo.

Limitaciones y Futuras Mejora
Actualmente, el algoritmo optimiza la ubicación de una única estación y la distribución de paneles solares, pero presenta las siguientes limitaciones:

-Constancia horaria: No considero las variaciones en el consumo y producción a lo largo del día. Las pérdidas se calculan como si fueran iguales en todas las horas.
-Estaciones múltiples:Solo calcula la optimización para una estación. No considera la proximidad a múltiples estaciones para reducir aún más las pérdidas energéticas.
-Restricciones espaciales: La ubicación de la estación puede asignarse en cualquier punto, incluso zonas no aptas como calles, plazas o áreas restringidas.

Próximos pasos
Incorporar un modelo de variación horaria para reflejar patrones de consumo y generación a lo largo del día.
Implementar soporte para múltiples estaciones y calcular las pérdidas en función de la estación más cercana.
Añadir restricciones espaciales basadas en un mapa o conjunto de zonas permitidas para la ubicación de estaciones.