PARCIAL 1 ESTRUCUTRA DE DATOS NO LINEALES

JUAN JOSÉ CARO VANEAS

1. ¿Cómo modelaste el problema como un grafo? Explica qué elementos representan los nodos y las aristas.

El problema se modela como un **grafo dirigido y ponderado** en un espacio tridimensional (coordenadas X, Y, Z), ya que los drones operan en una ciudad con altura.

- **Nodos**: Representan los puntos de navegación en la ciudad, como intersecciones, puntos de entrega, puntos de recarga y el inicio/destino del dron. Cada nodo tiene coordenadas (X, Y, Z) que indican su posición espacial.
- **Aristas**: Representan las conexiones posibles entre nodos (rutas que el dron puede tomar). Estas aristas tienen pesos asociados que reflejan el costo de recorrerlas, considerando distancia, consumo de batería y condiciones del viento. No hay aristas entre nodos si una zona de interferencia bloquea el camino directo.

2. ¿Cómo se determinan los pesos de las conexiones entre los nodos? ¿Qué factores consideraste?

Los pesos de las aristas se calculan como una combinación de varios factores:

- **Distancia**: La distancia euclidiana entre dos nodos en 3D:

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

- **Consumo de batería**: Proporcional a la distancia y ajustado por la resistencia del viento (más viento en contra implica mayor consumo).
- **Condiciones del viento**: Se añade un factor de penalización si el viento está en contra (por ejemplo, un multiplicador > 1), o se reduce si es a favor (multiplicador < 1).
- **Obstáculos**: Si una arista cruza una zona de interferencia, se elimina o se le asigna un peso infinito para evitarla.

El peso final de una arista podría definirse como:
Peso=Distancia×(1+Factor_Viento)×Consumo Batería por unidad

3. ¿Cómo afectó la presencia de zonas de interferencia en el cálculo de la ruta?

Las zonas de interferencia se tratan como áreas restringidas:

- Durante la construcción del grafo, se verifica si una arista cruza una zona de interferencia (usando geometría computacional, como intersección de líneas con volúmenes 3D).
- Si una arista cruza una zona restringida, se elimina del grafo o se le asigna un peso infinito, forzando al algoritmo de Dijkstra a buscar rutas alternativas.
- Esto asegura que el dron nunca vuele a través de estas zonas, ajustando la ruta óptima para rodearlas.

4. ¿Cómo implementaste el algoritmo de Dijkstra en tu solución? Explica el flujo del algoritmo paso a paso.

El algoritmo de Dijkstra encuentra la ruta de menor costo desde el nodo inicial hasta el destino, considerando los pesos de las aristas. El flujo es:

1. **Inicialización**:
    - Crear un conjunto de distancias desde el nodo inicial a todos los demás, inicialmente infinito excepto para el nodo inicial (distancia 0).
    - Usar una cola de prioridad para seleccionar el nodo con menor distancia acumulada.
    - Marcar todos los nodos como no visitados.
2. **Bucle principal**:
    - Seleccionar el nodo no visitado con la menor distancia acumulada.
    - Actualizar las distancias a sus nodos vecinos si se encuentra un camino más corto a través del nodo actual.
    - Marcar el nodo como visitado.
3. **Terminación**:
    - Repetir hasta que el nodo destino sea visitado o no queden nodos alcanzables.

4. **Reconstrucción de la ruta**:
    - Usar un diccionario de predecesores para rastrear la ruta desde el destino hasta el inicio.

5. ¿Cómo aseguraste que el dron busque puntos de recarga si es necesario?

Para incluir puntos de recarga:

- Se añade una restricción de batería: cada arista consume batería proporcional a su peso.
- Si la batería restante al llegar a un nodo es menor a un umbral crítico (por ejemplo, 10%), se fuerza al algoritmo a redirigir el dron al punto de recarga más cercano antes de continuar.
- Esto se logra modificando el grafo dinámicamente: al detectar batería baja, se recalcula la ruta desde el nodo actual al punto de recarga más cercano y luego al destino, sumando los costos.

6. ¿Qué estrategias utilizaste para optimizar la ejecución del algoritmo en mapas grandes?

- **Cola de prioridad**: Usar una cola de prioridad (heap) para seleccionar el nodo con menor distancia en tiempo O(log V) en lugar de buscar linealmente (O(V))
- **Poda de nodos**: Eliminar nodos y aristas inaccesibles (por zonas de interferencia) antes de ejecutar Dijkstra.
- **Discretización del espacio**: Reducir el número de nodos usando una cuadrícula 3D en lugar de modelar cada punto posible en la ciudad.
- **Cacheo**: Almacenar rutas precalculadas entre puntos de recarga para reutilizarlas si el dron necesita recargar varias veces.

7. ¿Cómo modificarías el algoritmo para que tome en cuenta el viento y otras condiciones climáticas en tiempo real?

- **Pesos dinámicos**: Actualizar los pesos de las aristas en tiempo real según datos climáticos (por ejemplo, APIs del clima).
- **Recálculo periódico**: Ejecutar Dijkstra nuevamente cada cierto tiempo o tras detectar cambios significativos en las condiciones (por ejemplo, viento fuerte repentino).

- **Peso ajustado**: Incluir un factor climático en la fórmula del peso:
  $Peso = Distancia \times (1 + k \cdot Velocidad\_Viento)$
  donde k, depende de la dirección del viento relativa al trayecto.

8. Si se agregaran múltiples drones operando simultáneamente, ¿cómo modificarías tu solución para evitar colisiones?

- **Grafo dinámico**: Añadir un sistema de reservas temporales para nodos y aristas, evitando que dos drones ocupen el mismo espacio al mismo tiempo.
- **Coordinación**: Usar un algoritmo centralizado que asigne rutas a cada dron, ajustando los pesos para penalizar rutas ya ocupadas por otros drones.
- **Tiempo como dimensión**: Convertir el grafo en un grafo espacio-temporal, donde los nodos incluyen una coordenada de tiempo, permitiendo planificar rutas en 4D (X, Y, Z, T).

9. ¿Qué mejoras podrías hacer para que la ruta se recalculase dinámicamente si un obstáculo nuevo aparece en el trayecto?

- **Detección de obstáculos**: Integrar sensores o actualizaciones en tiempo real que notifiquen nuevos obstáculos.
- **Recálculo parcial**: En lugar de rehacer todo el grafo, identificar las aristas afectadas por el nuevo obstáculo, eliminarlas o ajustar sus pesos, y ejecutar Dijkstra desde el nodo actual del dron.
- **Algoritmo adaptativo**: Usar una variante como D* Lite, que es más eficiente para recálculos dinámicos, en lugar de Dijkstra puro.

Datos del Grafo Propuesto

- Nodos:
    1. A(0,0,0): Inicio
    2. B(2,0,0): Interseccion
    3. C(2,2,0): Interseccion
    4. D(0,2,0): Interseccion
    5. E(1,1,1): Punto de recarga
    6. F(3,1,1): Interseccion

7. G(2,3,1): Interseccion
8. H(0,3,1): Destino

1. **A a B**:
    - Coordenadas: A(0,0,0) a B(2,0,0)
    - Distancia: Raiz cuadrada de [(2-0)^2 + (0-0)^2 + (0-0)^2] = Raiz cuadrada de 4 = 2
    - Viento: X aumenta (de 0 a 2), viento en contra, Factor_Viento = 0.2
    - Peso: 2 x (1 + 0.2) = 2 x 1.2 = 2.4
    - No cruza zona de interferencia: Incluida
2. **A a E**:
    - Coordenadas: A(0,0,0) a E(1,1,1)
    - Distancia: Raiz cuadrada de [(1-0)^2 + (1-0)^2 + (1-0)^2] = Raiz cuadrada de 3 = 1.732
    - Viento: X, Y, Z cambian (no alineado), Factor_Viento = 0
    - Peso: 1.732 x (1 + 0) = 1.732
    - No cruza zona de interferencia: Incluida
3. **A a D**:
    - Coordenadas: A(0,0,0) a D(0,2,0)
    - Distancia: Raiz cuadrada de [(0-0)^2 + (2-0)^2 + (0-0)^2] = Raiz cuadrada de 4 = 2
    - Viento: Y cambia, Factor_Viento = 0
    - Peso: 2 x (1 + 0) = 2
    - No cruza zona de interferencia: Incluida
4. **B a F**:
    - Coordenadas: B(2,0,0) a F(3,1,1)
    - Distancia: Raiz cuadrada de [(3-2)^2 + (1-0)^2 + (1-0)^2] = Raiz cuadrada de 3 = 1.732

- Viento: X, Y, Z cambian, Factor_Viento = 0
- Peso: 1.732 x (1 + 0) = 1.732
- No cruza zona de interferencia: Incluida

5. **C a G**:
   - Coordenadas: C(2,2,0) a G(2,3,1)
   - Distancia: Raiz cuadrada de [(2-2)^2 + (3-2)^2 + (1-0)^2] = Raiz cuadrada de 2 = 1.414
   - Viento: Y, Z cambian, Factor_Viento = 0
   - Peso: 1.414 x (1 + 0) = 1.414
   - No cruza zona de interferencia: Incluida
   - Nota: En el grafo final, C no se conecta a nada porque la zona de interferencia lo aisla, pero lo dejamos aqui para referencia.
6. **D a E**:
   - Coordenadas: D(0,2,0) a E(1,1,1)
   - Distancia: Raiz cuadrada de [(1-0)^2 + (1-2)^2 + (1-0)^2] = Raiz cuadrada de 3 = 1.732
   - Viento: X, Y, Z cambian, Factor_Viento = 0
   - Peso: 1.732 x (1 + 0) = 1.732
   - No cruza zona de interferencia: Incluida
7. **D a H**:
   - Coordenadas: D(0,2,0) a H(0,3,1)
   - Distancia: Raiz cuadrada de [(0-0)^2 + (3-2)^2 + (1-0)^2] = Raiz cuadrada de 2 = 1.414
   - Viento: Y, Z cambian, Factor_Viento = 0
   - Peso: 1.414 x (1 + 0) = 1.414
   - No cruza zona de interferencia: Incluida
8. **E a F**:

- Coordenadas: E(1,1,1) a F(3,1,1)
- Distancia: Raiz cuadrada de [(3-1)^2 + (1-1)^2 + (1-1)^2] = Raiz cuadrada de 4 = 2
- Viento: X aumenta (de 1 a 3), viento en contra, Factor_Viento = 0.2
- Peso: 2 x (1 + 0.2) = 2 x 1.2 = 2.4
- No cruza zona de interferencia: Incluida

9. **F a G**:
    - Coordenadas: F(3,1,1) a G(2,3,1)
    - Distancia: Raiz cuadrada de [(2-3)^2 + (3-1)^2 + (1-1)^2] = Raiz cuadrada de 5 = 2.236
    - Viento: X disminuye (de 3 a 2), viento a favor, Factor_Viento = -0.1
    - Peso: 2.236 x (1 - 0.1) = 2.236 x 0.9 = 2.012
    - No cruza zona de interferencia: Incluida
10. **G a H**:
    - Coordenadas: G(2,3,1) a H(0,3,1)
    - Distancia: Raiz cuadrada de [(0-2)^2 + (3-3)^2 + (1-1)^2] = Raiz cuadrada de 4 = 2
    - Viento: X disminuye (de 2 a 0), viento a favor, Factor_Viento = -0.1
    - Peso: 2 x (1 - 0.1) = 2 x 0.9 = 1.8
    - No cruza zona de interferencia: Incluida