Respuestas

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

- **Nodos**: Cada celda del tablero (coordenadas (x, y)) representa un nodo en el grafo.
- **Aristas**: Las conexiones entre nodos representan los movimientos válidos del dron (arriba, abajo, izquierda, derecha). Cada arista tiene un peso que representa el costo de moverse de un nodo a otro.

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

- Los pesos de las aristas se determinan aleatoriamente dentro de un rango definido $peso_aristas = (1, 5)$Esto simula la variabilidad en el costo de moverse entre nodos, que podría depender de factores como la distancia, el terreno o el consumo de energía.

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

- Las zonas de interferencia se modelan como obstáculos. Estos nodos se excluyen del grafo, lo que significa que el dron no puede pasar por ellos. Esto limita las posibles rutas y obliga al algoritmo a encontrar caminos alternativos.

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

- Se crea una cola de prioridad con el nodo inicial y su distancia (0).
- Se inicializan dos diccionarios: dist (distancias mínimas) y previos (nodos anteriores).
    2. **Bucle principal**:

- Se extrae el nodo con la distancia mínima de la cola.
- Si es el nodo meta, se termina el bucle.
- Para cada vecino, se calcula la nueva distancia. Si es menor que la registrada, se actualiza la distancia y se añade a la cola.
    3. **Reconstrucción del camino**:

- Se sigue la cadena de nodos en previos desde la meta hasta el inicio.

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

- En el código proporcionado, no se implementa explícitamente la búsqueda de puntos de recarga. Sin embargo, podrían añadirse como nodos especiales con un peso negativo o prioridad alta, y modificar el algoritmo para redirigir al dron hacia ellos cuando la batería esté baja.

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

- Se utiliza una cola de prioridad (heap) para gestionar eficientemente los nodos por visitar.
- Se evita recalcular distancias para nodos ya visitados.
- Se limita el grafo a nodos alcanzables, excluyendo obstáculos.

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

- Podrían añadirse factores de peso dinámicos a las aristas, ajustando los costos en función de las condiciones climáticas actuales (por ejemplo, aumentar el peso si hay viento en contra).
- Se requeriría una actualización en tiempo real de los pesos durante la ejecución del algoritmo.

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

- Se podría implementar un sistema de reserva de nodos, donde cada dron "reserva" los nodos que planea usar en su ruta.
- También podría usarse un algoritmo de planificación centralizado que asigne rutas evitando superposiciones.

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

- Implementar un sistema de detección de obstáculos en tiempo real.
- Recalcular la ruta usando el grafo actualizado, excluyendo el nuevo obstáculo.
- Usar un algoritmo incremental como D Lite para optimizar el recálculo.

    9.1. ***¿Cómo compararías Dijkstra con A para resolver este problema? ¿Cuál sería más eficiente en este caso?***

- **Dijkstra**: Explora todos los nodos de manera uniforme, garantizando la ruta más corta, pero siendo menos eficiente en mapas grandes.
- **A**: Usa una heurística para guiar la búsqueda hacia la meta, siendo más eficiente en términos de tiempo y memoria.
- **Conclusión**: A sería más eficiente en este caso, especialmente en mapas grandes, ya que reduce el número de nodos explorados