Proyecto 3 Diseño y Análisis de Algoritmos **Documento Solución Propuesta**
Juan Felipe Hortúa - 202320918
Juan Felipe Ochoa - 202320053

Descripción del Problema

El problema consiste en encontrar el mínimo número de líneas (horizontales y verticales) que cubren todos los focos de infección en el campo de Don Jaime. Este problema se puede clasificar como Minimum Vertex Cover, un problema NP-Complete, lo que significa que no existe un algoritmo completamente eficiente que garantice encontrar la solución óptima en todos los casos dentro de un tiempo polinomial.

Modelado del Problema

Podemos pensar en este problema de la siguiente manera:

- Cada coordenada X única representa una posible línea vertical (drone)
- Cada coordenada Y única representa una posible línea horizontal (artrópodo)
- Cada foco (x,y) debe ser cubierto por al menos una línea: la línea vertical X o la línea horizontal Y

El objetivo es seleccionar el conjunto mínimo de líneas que cubra todos los focos.

Estrategia de solución

Algoritmo 1: Aproximación por Aristas

Este algoritmo garantiza encontrar una solución que sea, como máximo, 2-aproximado, es decir el doble del óptimo.
El algoritmo funciona de la siguiente manera:

1. Ordena los focos para procesarlos sistemáticamente
2. Para cada foco sin cubrir:
    - Agrega ambas líneas que lo cubren (la horizontal Y y la vertical X)
    - Marca como cubiertos todos los focos que estas líneas alcancen
3. Continuar hasta que todos los focos estén cubiertos

Ventaja: tiene garantía teórica de factor 2-aproximado, del cual se habló anteriormente.

Algoritmo 2: Selección Greedy

Este algoritmo selecciona iterativamente la mejor opción disponible:

1. Mientras haya focos sin cubrir:
    - Evalúa todas las líneas horizontales posibles y cuenta cuántos focos cubre cada una
    - Evalúa todas las líneas verticales posibles y cuenta cuántos focos cubre cada una
    - Marca esos focos como cubiertos
2. Repite hasta cubrir todos los focos

Ventaja: En la práctica suele dar mejores resultados que el algoritmo de aproximación, ya que el de aproximación de todas maneras no puede ser completamente eficiente.

Eliminación de Líneas Redundantes (Optimización)

Después de aplicar cualquiera de los dos algoritmos, se ejecuta una fase de optimización:

1. Para cada línea horizontal, verifica si todos sus focos ya están cubiertos por líneas verticales. De ser así, elimina esa línea horizontal
2. Para cada línea vertical, verifica que todos sus focos ya están cubiertos por líneas horizontales. De ser así elimina esa línea vertical
3. Repite hasta que no se puedan eliminar más líneas

Selección Final

La solución ejecuta ambos algoritmos, optimiza ambos resultados, y selecciona el que use menos líneas totales (h + v mínimo).

Se eligió este enfoque híbrido entre aproximado y greedy, porque combina garantías teóricas con desempeño práctico. El algoritmo de aproximación garantiza un factor 2, mientras que el greedy suele dar mejores resultados en casos reales. También la eliminación de redundancias mejora cualquier solución inicial. Finalmente, al ejecutar ambos y elegir el mejor, nos aseguramos de obtener buenos resultados en diferentes tipos de casos.

Diagrama Explicación Algoritmo:

ENTRADA: Focos (2,2), (2,4), (4,2), (4,4)

Visualización:

```
  X:  1  2  3  4  5
 Y:
 5
 4        •     •     ← Focos en Y=4
 3
 2        •     •     ← Focos en Y=2
 1
```

ALGORITMO DE APROXIMACIÓN:

Paso 1: Procesa (2,2) → Agrega líneas X=2 y Y=2

Cubre: (2,2), (2,4), (4,2)

Paso 2: Procesa (4,4) → Agrega líneas X=4 y Y=4

Cubre: (4,4), (4,2) [ya cubierto]

Resultado: 2 verticales (X=2, X=4) + 2 horizontales (Y=2, Y=4)

OPTIMIZACIÓN:

- ¿Y=2 es necesaria? Sí, porque cubre (2,2) y (4,2)
- ¿Y=4 es necesaria? Sí, porque cubre (2,4) y (4,4)
- ¿X=2 es necesaria? No, sus focos están cubiertos por Y=2 y Y=4
- ¿X=4 es necesaria? No, sus focos están cubiertos por Y=2 y Y=4

Resultado optimizado: 2 horizontales (Y=2, Y=4)

ALGORITMO GREEDY:

Paso 1: Todas las líneas cubren 2 focos

Elige X=2 → Cubre (2,2), (2,4)

Paso 2: X=4 cubre 2 focos, Y=2 cubre 1, Y=4 cubre 1

Elige X=4 → Cubre (4,2), (4,4)

Resultado: 2 verticales (X=2, X=4)

SELECCIÓN FINAL:

- Aproximación optimizada: 2 líneas horizontales
- Greedy: 2 líneas verticales
- Ambas usan 2 líneas → Elegir cualquiera (el código elige greedy)

SALIDA: 0 horizontales, 2 verticales

```
0
2 2 2 2 4 4 2 4 4
```

Análisis de Complejidades

Complejidad Temporal

Aproximación por Aristas:

- Ordenar focos: O(n log n)
- Procesar cada foco: O(n)
  Por cada foco, revisar todos los focos para marcar cubiertos
- Total: $O(n^2)$

Selección Greedy:

- Mientras haya focos sin cubrir (peor caso n iteraciones)
  Evaluar líneas horizontales: $O(k_y)$ donde $k_y \leq n$. Por cada línea, contar focos cubiertos: O(n)
  Evaluar líneas verticales: $O(k_x)$ donde $k_x \leq n$. Por cada línea, contar focos cubiertos: O(n)
- Total: $O(n^2 * k)$ donde $k \leq n$, por lo tanto $O(n^3)$ en el peor caso

Optimización:

- Mientras haya cambios (peor caso k iteraciones donde $k \leq 2n$):
  Revisar cada línea: O(k)
  Verificar cobertura: O(n)
- Total: $O(k^2 * n) = O(n^3)$ en el peor caso

Construcción de líneas finales:

- Ordenar líneas O(k log k) donde $k \leq 2n$
- Encontrar min/max por línea: O(n * k)
- Total: $O(n * k) = O(n^2)$ en el peor caso

Complejidad Temporal Total: $O(n^2) + O(n^3) + O(n^3) = O(n^3)$

Complejidad Espacial

Estructura de datos principales:

- Lista de focos: O(n)
- Conjunto de líneas (lineas_x, lineas_y): O(k) donde $k \leq 2n$
- Conjunto de focos cubiertos: O(n)
- Conjunto de temporales en greedy: O(n)

Complejidad Espacial Total: $O(n) + O(k) + O(n) + O(n) = O(n)$

Conclusiones

La solución implementada combina eficientemente dos enfoques complementarios. Un algoritmo con garantía teórica, por lo que nos referimos a que es de aproximación 2-factor. Un algoritmo con un buen desempeño práctico, es decir el algoritmo greedy. Finalmente una fase de optimización que mejora ambos resultados. Esta estrategia combinada permite obtener soluciones de alta calidad para el problema de cobertura mínima, balanceando la complejidad $O(n^3)$ con calidad de los resultados obtenidos. Finalmente el código es eficiente para los tamaños

del problema especificados ($n \leq 1000$) y produce salidas que cumplen con el requisito de minimizar el número total de líneas utilizadas.