Escritura del problema de solver de FlowFree

María Camila Aguirre Collante[1] Jessica Tatiana Naizaque Guevara[1]

[1]Departamento de Ingeniería de Sistemas, Pontificia Universidad Javeriana
Bogotá, Colombia
{aguirrec.mcamila, j.naizaque}@javeriana.edu.co

24 de noviembre de 2022

Resumen

En este documento se presenta la formalización del problema del solver del juego FlowFree, junto con la descripción del algoritmo a implementar para su solución. Además, se presenta un análisis experimental de algunos tableros propuestos. **Palabras clave:** flow, algoritmo, tablero, extremos, colores.

Índice

1. Introducción

Flow free es un juego presentado en un tablero con pares de puntos de colores que ocupan algunas posiciones del tablero. El objetivo del juego es conectar puntos del mismo color teniendo en cuenta que todo el tablero debe estar ocupado por algún color y cada par de colores debe estar conectado entre sí. El proyecto busca analizar el algoritmo que solucione los diferentes tableros presentados con el juego, implementar esta solución y probarla en diferentes casos. Por lo tanto, en el desarrollo del presente documento se podrá encontrar la formalización del problema (sección 2), análisis del problema (sección 3), algoritmo de solución (sección 4), análisis experimental (sección 5) y resultados obtenidos (sección 6).

2. Diseño y formalización del problema

Dado un tablero de juego con dimensiones $n \times n$, dado un conjunto de colores para representar en el tablero de juego y dado un conjunto de pares que hacen referencia a los extremos (inicio y fin) de cada uno de los colores; encontrar una solución del tablero, en la que se plasme la conexión entre todos los extremos de cada color ubicados en el tablero y se tengan todas las casillas del tablero con algún color en específico.

2.1. Diseño del problema del "FlowFree"

De acuerdo con lo anterior, el problema de FlowFree se define a partir de las siguientes entradas y salidas:

- Entradas:
  - una matriz $T : \mathbb{R}^{n\times n}$ donde n es el orden de la misma representando las dimensiones del tablero y $5 \leq n \leq 8$.
  - una secuencia C de colores representados con su inicial. Es decir, $c_i \in \mathbb{L} \mid \mathbb{L} = \langle R, B, Y, G, W, P, C\rangle$, para todo $1 \leq i \leq k \mid k = |C|$.
  - una secuencia P de pares de coordenadas para cada color del tablero. Definida como $P : \langle p_j \in \mathbb{E}\rangle$, para todo $1 \leq j \leq m \mid m = |P| = k$. A su vez, $\mathbb{E} : \langle e_0, e_1\rangle \mid e_0 \wedge e_1 \in T$ y, por eso, $e_0 \wedge e_1 < n$. Asimismo, $i = j$.
- Salidas:
  - una matriz $T' : \mathbb{R}^{n\times n}$. Donde, $T' : \langle t_x \in \mathbb{L}\rangle$, para todo $1 \leq x \leq n$, recordando que $\mathbb{L} = \langle R, B, Y, G, W, P, C\rangle$.
  - una secuencia A de caminos de coordenadas para cada color del tablero. Definida como $A : \langle a_y \in \mathbb{S}\rangle$, para todo $1 \leq y \leq z \mid z = |A| = k$. A su vez, $\mathbb{S} : \langle s_w \in T\rangle \mid s_w = \langle coor_0, coor_1\rangle \wedge [coor_0 \wedge coor_1] < n$. Asimismo, $y = j$.

3. Análisis del problema

El problema de FlowFree es de tipo decisión, esto se debe a que el algoritmo a proponer debe decidir qué caminos tomar para completar la conexión entre todos los extremos iniciales. Asimismo, el problema es de clase NP completo, esto se debe a que es un problema inicialmente de tipo de decisión, de tiempo superpolinomial, verificable en P y transformable en otros problemas NP.

4. Algoritmo de solución

Teniendo en cuenta el análisis del problema (sección 3), la solución propuesta en este proyecto será realizada por medio de un algoritmo de aproximación. La idea general de la solución es seguir el algoritmo de búsqueda A* con algunas modificaciones realizadas por las estudiantes. En este, se define como primera instancia una lista de pares relacionando el color con la distancia de Manhattan de los extremos correspondientes a este color. Esta lista de pares será organizada teniendo en cuenta el valor de la distancia de

Manhattan de menor a mayor, se recorrerá de manera que el algoritmo busque la solución o el camino más aceptable para cada par de puntos ingresados.

Ahora, para la búsqueda de cada camino, se tomará el extremo inicial y se revisarán sus casillas adyacentes, estas se almacenarán y a cada una se le asignará un costo que describe cuánto le cuesta a esa casilla llegar a la casilla final. De acuerdo con este costo, se organizarán las casillas adyacentes por medio de una cola de prioridad para llevar un orden en la revisión de estas casillas. El costo dependerá de la casilla que se esté revisando, por ejemplo, si la casilla no es un punto inicial pero ya tiene un color asignado, se le da x costo y si la casilla está libre, se le asigna y costo. Sin embargo, a estos costos, se les añadirá una heurística que se hallará mediante la distancia de Manhattan que se tenga desde la casilla a utilizar hasta la casilla final o extremo final del camino de dicho color.

Al finalizar la revisión de todos los colores con sus respectivos extremos, se verifica nuevamente que todos ya tengan una solución de llegada desde el punto inicial al punto final. Si existe algún color que no se haya solucionado correctamente o que haya sido interrumpido por la solución de otro, se realizará el algoritmo de solución para este color nuevamente.

Algoritmo 1 Calcular Costo

```
procedure COST(board, color, row, col)
    if board[row][col] ≠ 0 then
        g ← 10
    else
        g ← 1
    end if
    path ← camino actual del color
    if |path| > 0 then
        if board[row][col] ≠ 0 then
            g ← |path| + 10
        else
            g ← |path| + 1
        end if
        record ← historial de caminos del color
        if path ∈ record then
            g ← g + 7
        end if
        colorRow ← fila del extremo del color
        colorCol ← columna del extremo del color
        h ← ABS(row − colorRow) + ABS(col − colorCol)
    end if
    return g + h
end procedure
```

Algoritmo 2 Algoritmo aproximación **A***

```
procedure bestPath(board, colors, coordenates)
    let C[1...colors.length]
    for i ← 1 to colors.length do
        C.append(colors[i], distance)
    end for
    orderC ← SORT(C)
    let pritorityQueue una cola de ejecución
    while no todos los caminos se han completado do
        for p ← 1 to colors.length do
            adj ← obtener todos los adyacentes
            for actual ← 1 to adj.length do
                currentCost ← cost(board, C[p], coordenates[p][0], coordenates[p][1])
                priorityQueue.put((currentCost, adj[actual]))
            end for
            choosed ← priorityQueue.get()
            path ← camino actual del color
            path.append(choosed)
            board[choosed[0]][[choosed[1]] ← c[p]
        end for
    end while
end procedure
```

5. Análisis experimental

En esta sección se presentarán algunos los experimentos para confirmar la efectividad del algoritmo presentados en la sección 4. Gracias que se cuenta con una totalidad de 22 niveles, distribuidos en tableros con dimensiones de 5×5 hasta 9×9; es posible establecer los siguientes escenarios de pruebas:

1. **Niveles** 5×5
   - *a*) Se cuenta con 5 pares de colores, para los niveles $1, 2$ y 4
   - *b*) Se cuenta con 4 pares de colores, para los niveles 3 y 5
2. **Niveles** 6×6
   - *a*) Se cuenta con 6 pares de colores, para los niveles 1 y 4
   - *b*) Se cuenta con 5 pares de colores, para los niveles 3 y 5
   - *c*) Se cuenta con 4 pares de colores, para el nivel 2
3. **Niveles** 7×7
   - *a*) Se cuenta con 7 pares de colores, para los niveles $1, 2$ y 3
   - *b*) Se cuenta con 45 pares de colores, para el nivel 4
4. **Niveles** 8×8
   - *a*) Se cuenta con 7 pares de colores, para los niveles 1

6. Resultados y análisis de resultados

A partir de los experimentos especificados en la sección 5, es posible obtener los siguientes resultados:

1. **Niveles** 5×5

   a) En la figura 1 es posible evidenciar el tablero resuelto para el Nivel 1:

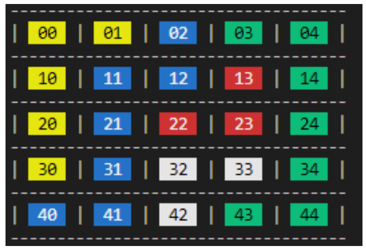

Figura 1: Resultado Nivel 1

   b) En la figura 2 es posible evidenciar el tablero resuelto para el Nivel 2:

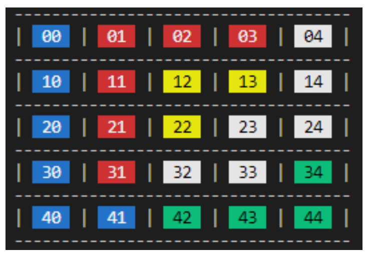

Figura 2: Resultado Nivel 2

   c) En la figura 3 es posible evidenciar el tablero resuelto para el Nivel 3:

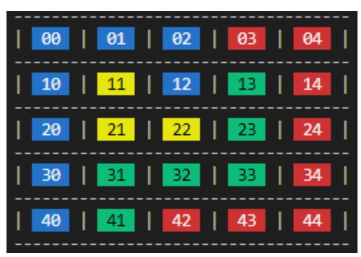

Figura 3: Resultado Nivel 3

   d) En la figura 4 es posible evidenciar el tablero resuelto para el Nivel 4:

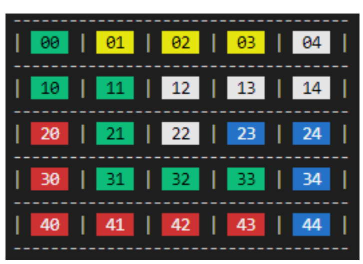

Figura 4: Resultado Nivel 4

e) En la figura 5 es posible evidenciar el tablero resuelto para el Nivel 5:

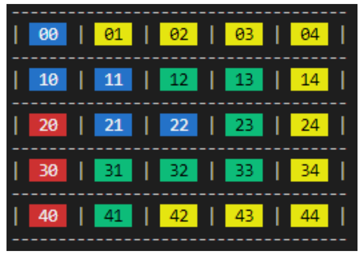

Figura 5: Resultado Nivel 5

Teniendo en cuenta lo anterior, es posible aseverar que el algoritmo funciona para tableros con esta dimensión.

2. **Niveles** 6×6

a) En la figura 6 es posible evidenciar el tablero resuelto para el Nivel 1:

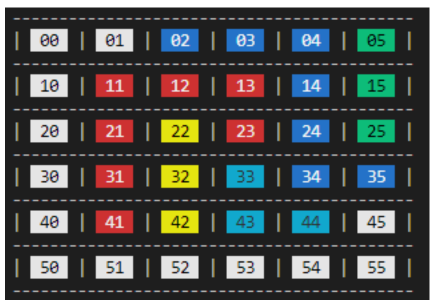

Figura 6: Resultado Nivel 1

b) En la figura 7 es posible evidenciar el tablero resuelto para el Nivel 2:

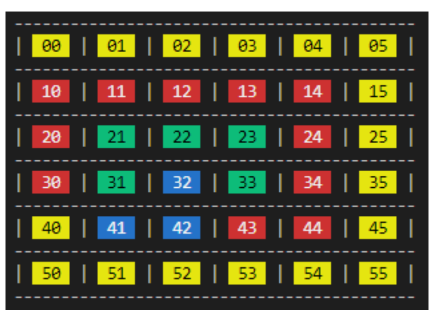

Figura 7: Resultado Nivel 2

c) En la figura 8 es posible evidenciar el tablero resuelto para el Nivel 3:

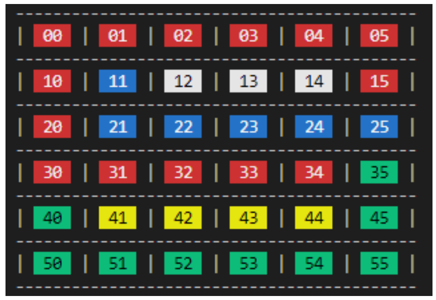

Figura 8: Resultado Nivel 3

d) En la figura 9 es posible evidenciar el tablero resuelto para el Nivel 4:

Figura 9: Resultado Nivel 4

e) En la figura 10 es posible evidenciar el tablero resuelto para el Nivel 5:

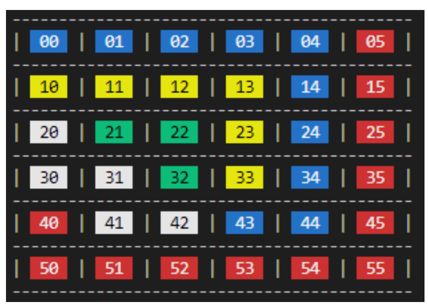

Figura 10: Resultado Nivel 5

Teniendo en cuenta lo anterior, es posible aseverar que el algoritmo funciona para tableros con esta dimensión.

3. **Niveles** 7×7

a) En la figura 11 es posible evidenciar el tablero resuelto para el Nivel 1:

Figura 11: Resultado Nivel 1

b) En la figura 12 es posible evidenciar el tablero resuelto para el Nivel 2:

Figura 12: Resultado Nivel 2

c) En la figura 13 es posible evidenciar el tablero resuelto para el Nivel 3:

Figura 13: Resultado Nivel 3

d) En la figura 14 es posible evidenciar el tablero resuelto para el Nivel 4:

Figura 14: Resultado Nivel 4

Teniendo en cuenta lo anterior, es posible aseverar que el algoritmo funciona para tableros con esta dimensión.

4. **Niveles** 8×8

   a) En la figura 15 es posible evidenciar el tablero del Nivel 1, en el cual se observa que cada color tiene la unión correspondiente a sus puntos iniciales. Sin embargo, al no tener la validación del llenado del tablero no se completa de manera correcta.

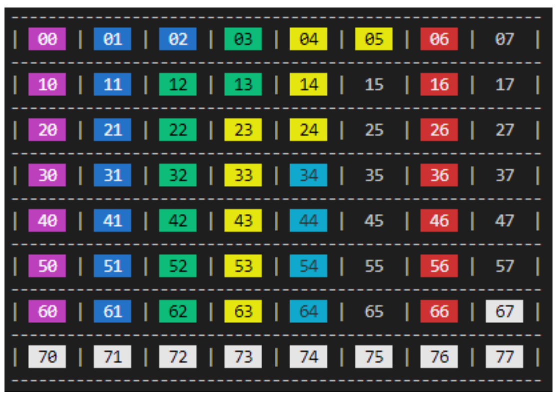

Figura 15: Resultado Nivel 1

Es importante resaltar que debido a la función de distancia de Manhattan en algunos casos de los niveles 8×8 no es posible obtener un tablero solucionado, adicionalmente, el no verificar si el tablero ha sido llenado en su totalidad se puede evidenciar que afecta el resultado. Debido