Representación del coloreado de grafos. Teorema de los cuatro colores.

Trabajo final de la asignatura Programación Declarativa.
Cuarto curso. Primer cuatrimestre. Curso 2019/2020.

José Manuel Cuevas Muñoz

Grado en Ingeniería Informática. Escuela politécnica superior de Córdoba.

10 de diciembre de 2019

Tabla de contenidos

Introducción histórica

Teorema

Dado cualquier mapa geográfico con regiones continuas, este puede ser coloreado con cuatro colores diferentes, de forma que no queden regiones adyacentes con el mismo color.

Introducción histórica

Teorema

Fue propuesto por primera vez por Francis Guthrie cuando intentaba pintar el mapa de Inglaterra con el mínimo número de colores posibles

Introducción histórica

Demostración

No se demostró hasta 1989 por los matemáticos alemanes Kenneth Appel y Wolfgang utilizando las conjeturas de Haken Heinrich Heesch y con la ayuda de Dorothea Blostein y un ordenador. Las conjeturas son las siguientes:

- Un mapa no se puede satisfacer si al menos en una de sus partes no se puede colorear con cinco colores.
- Si un mapa no es coloreable con cuatro colores entonces uno de los submapas no es coloreable.

Finalmente, el problema se redujo a 1482 submapas, todos con solución.

Introducción histórica

Demostración

No se demostró hasta 1989 por los matemáticos alemanes Kenneth Appel y Wolfgang utilizando las conjeturas de Haken Heinrich Heesch y con la ayuda de Dorothea Blostein y un ordenador. Las conjeturas son las siguientes:

- Un mapa no se puede satisfacer si al menos en una de sus partes no se puede colorear con cinco colores.
- Si un mapa no es coloreable con cuatro colores entonces uno de los submapas no es coloreable.

Finalmente, el problema se redujo a 1482 submapas, todos con solución.

Introducción histórica

Demostración

No se demostró hasta 1989 por los matemáticos alemanes Kenneth Appel y Wolfgang utilizando las conjeturas de Haken Heinrich Heesch y con la ayuda de Dorothea Blostein y un ordenador. Las conjeturas son las siguientes:

- Un mapa no se puede satisfacer si al menos en una de sus partes no se puede colorear con cinco colores.
- Si un mapa no es coloreable con cuatro colores entonces uno de los submapas no es coloreable.

Finalmente, el problema se redujo a 1482 submapas, todos con solución.

Introducción histórica

Demostración

No se demostró hasta 1989 por los matemáticos alemanes Kenneth Appel y Wolfgang utilizando las conjeturas de Haken Heinrich Heesch y con la ayuda de Dorothea Blostein y un ordenador. Las conjeturas son las siguientes:

- Un mapa no se puede satisfacer si al menos en una de sus partes no se puede colorear con cinco colores.
- Si un mapa no es coloreable con cuatro colores entonces uno de los submapas no es coloreable.

Finalmente, el problema se redujo a 1482 submapas, todos con solución.

Algoritmos de coloración

Algoritmo de coloración

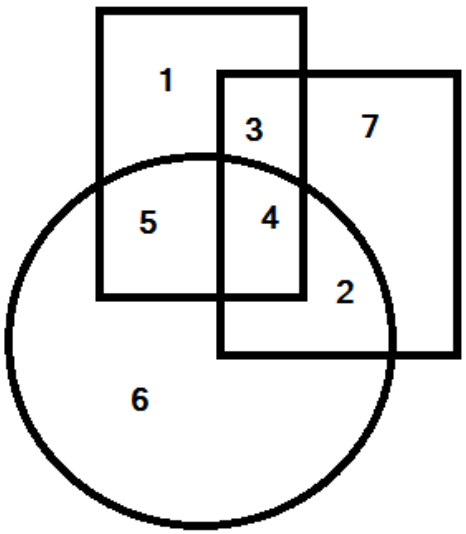
- El problema de colorear un grafo es $NP - Completo$, siendo normalmente de orden $O(2^n n)$ encontrar la solución idónea.
- La única forma de que siempre dé el menor número de colores es utilizando algoritmos de fuerza bruta.

Algoritmos de coloración

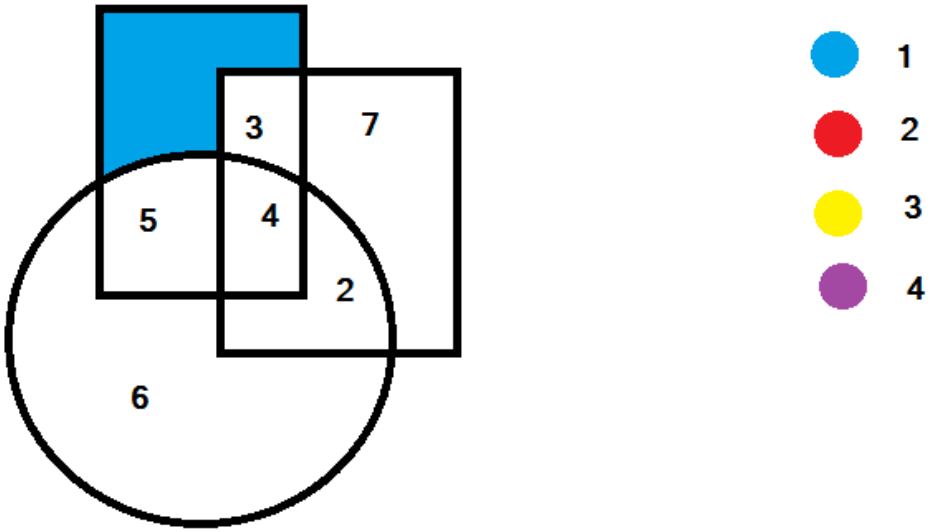
Algoritmo Voraz de coloración

- Este algoritmo de orden $O(n)$ se basa en colorear las áreas en el orden dado con el primer color disponible.
- Su actuación depende del orden en que se miren las áreas a colorear. Encontrar el mejor orden es NP-complejo.
- No siempre da la mejor solución pero tiende a dar una cercana y formada por cuatro colores.

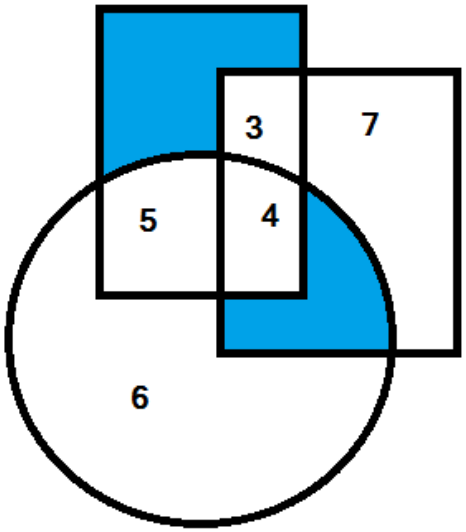
Ejemplo Algoritmo Voraz



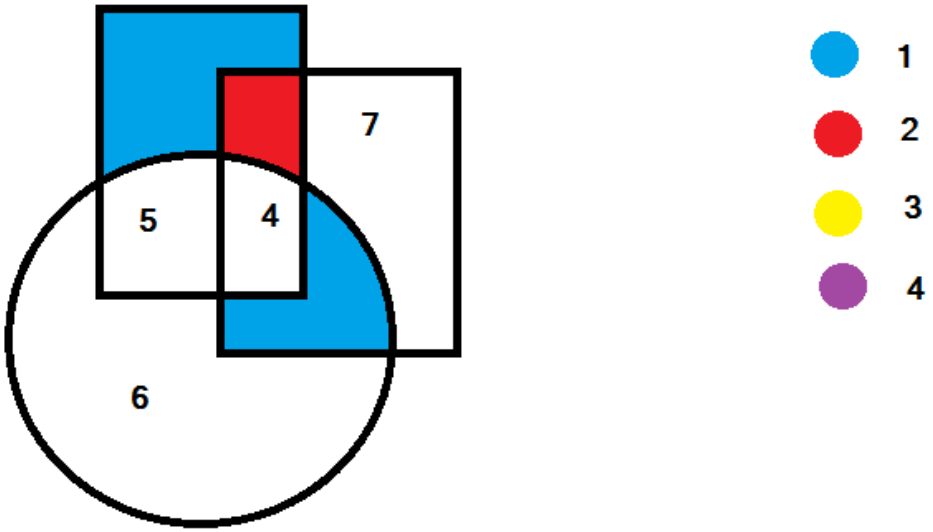
Ejemplo Algoritmo Voraz



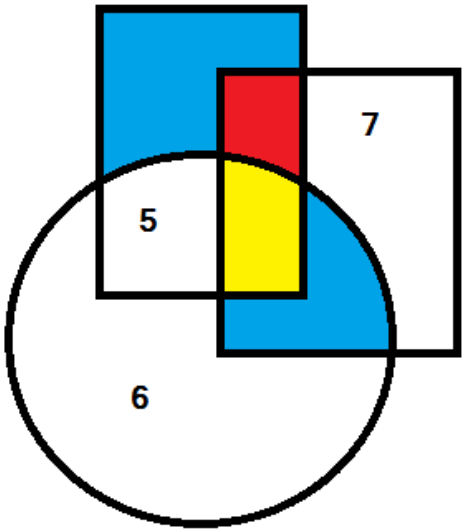
Ejemplo Algoritmo Voraz



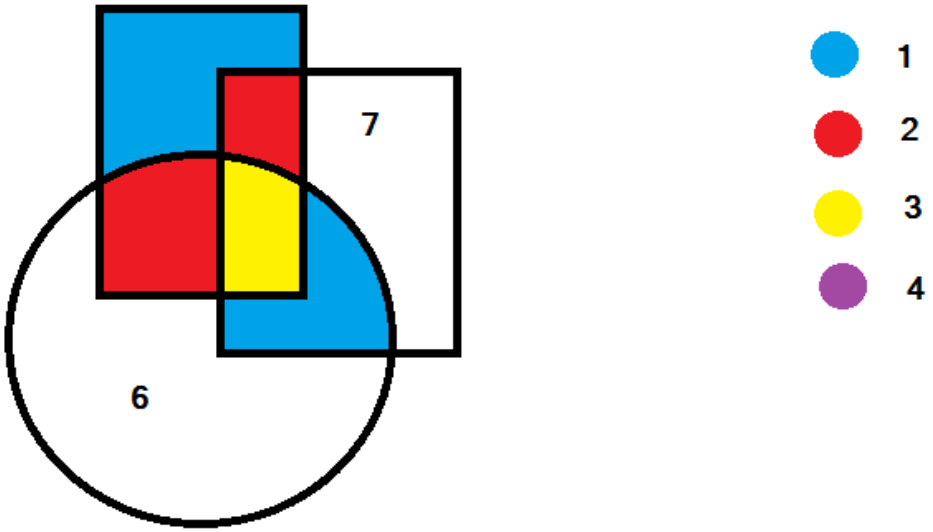
Ejemplo Algoritmo Voraz



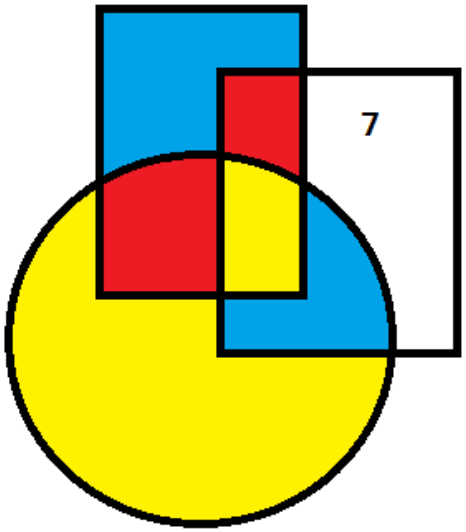
Ejemplo Algoritmo Voraz



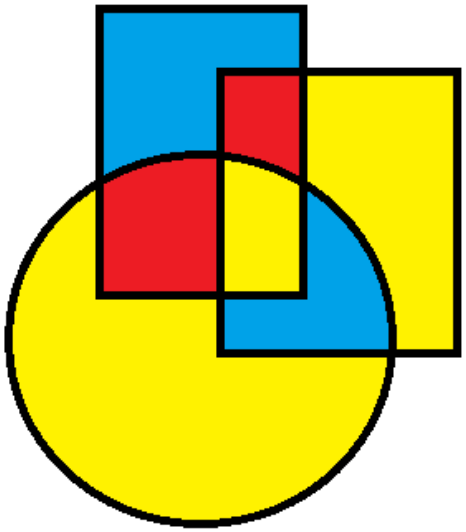
Ejemplo Algoritmo Voraz



Ejemplo Algoritmo Voraz

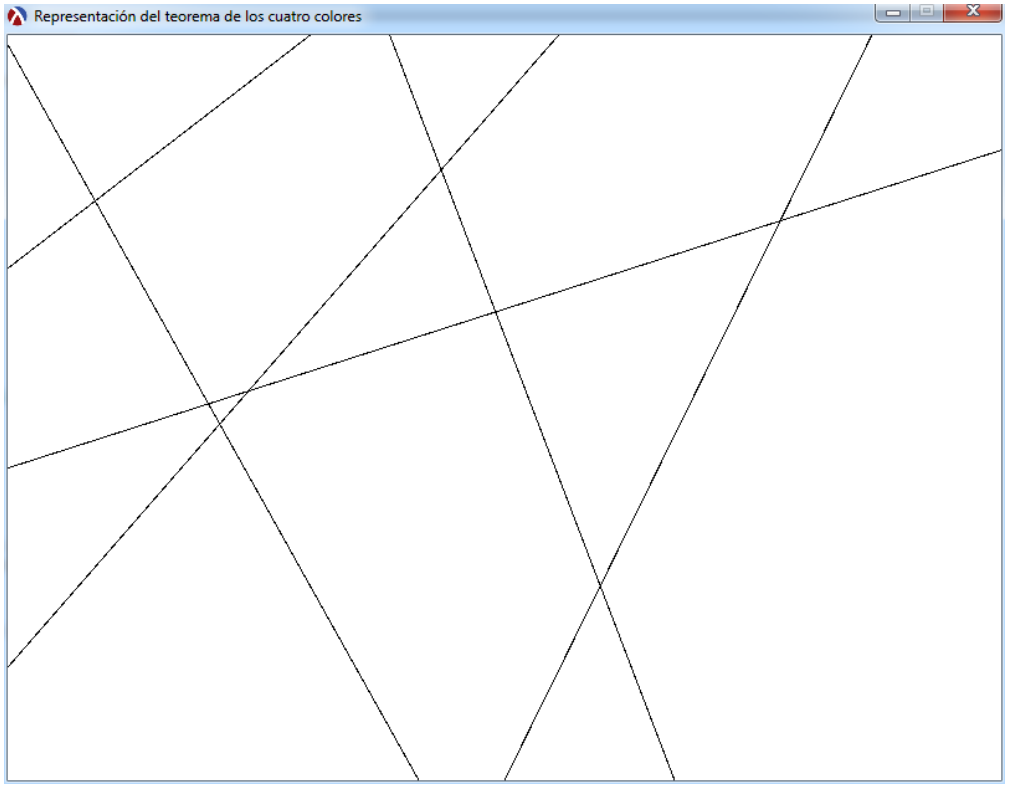


Ejemplo Algoritmo Voraz



Implementación

En nuestro caso el usuario definirá las áreas mediante rectas definidas en el espacio de la pantalla.



Implementación

Implementación

Podríamos decir que el programa se divide en dos partes:

- Parte geométrica: Sacar las distintas áreas que conforman la figura
- Parte de coloreado: Colorear las áreas de forma que dos áreas adyacentes no sean del mismo color.

Tipo abstracto de dato Recta

Recta

Tiene dos atributos:

- Índice: Índice de la recta del 1 al n. Las rectas del marco se definen por x0,xF,y0 e yF.
- Parámetros: Lista con los parámetros a, b y c de la recta siendo esta: ay+bx+c=0.

Tipo abstracto de dato Recta

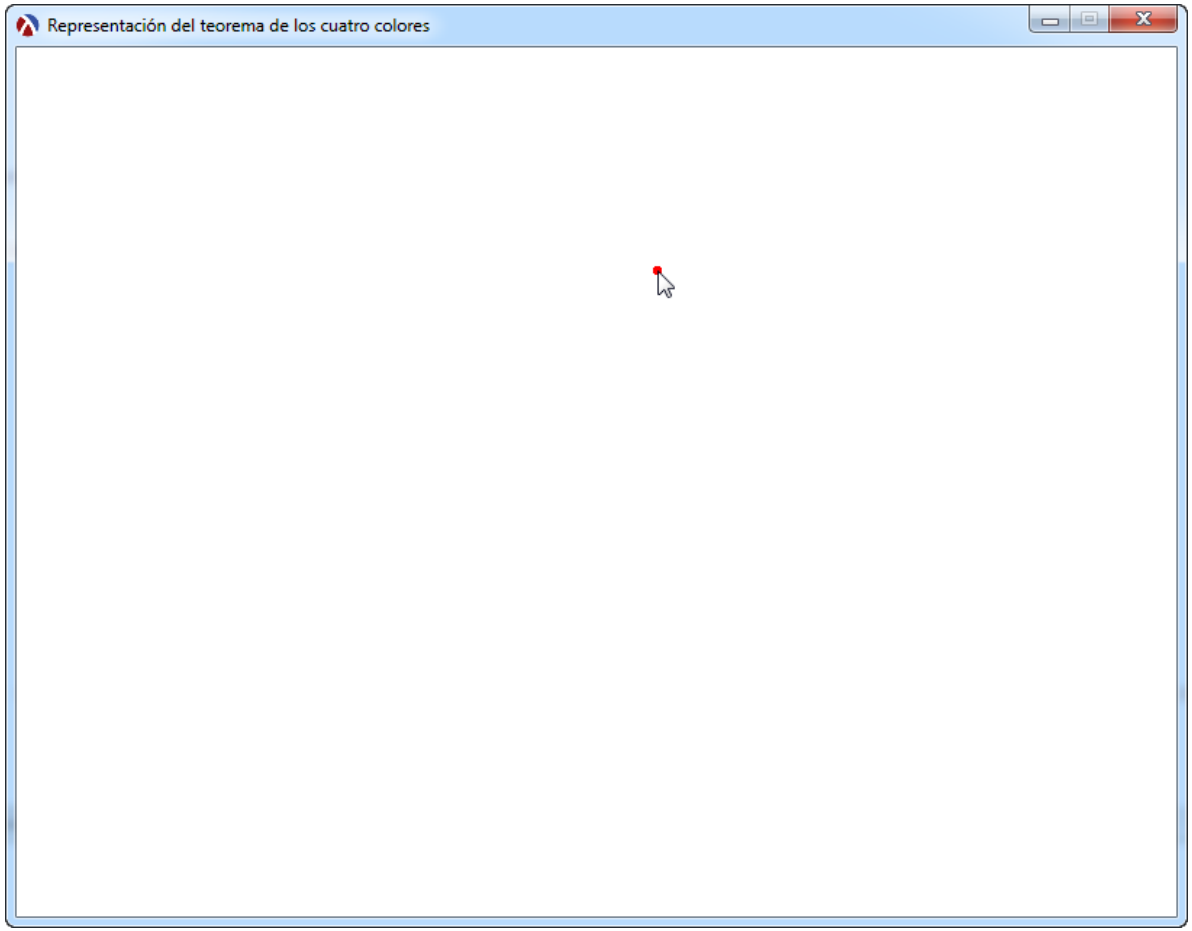
Recta

Para definir cada recta utilizamos dos puntos. De modo que el usuario introduce dos puntos clickeando en la pantalla y se calculan los parámetros de la recta.
Las funciones utilizadas para esto son:

- detectar-puntos-linea: Detecta los dos puntos que definirán una linea
- rectadospuntos: Calcula los parámetros de la recta que para por dos puntos
- dibujar-linea: Dibuja la linea delimitada por dos puntos

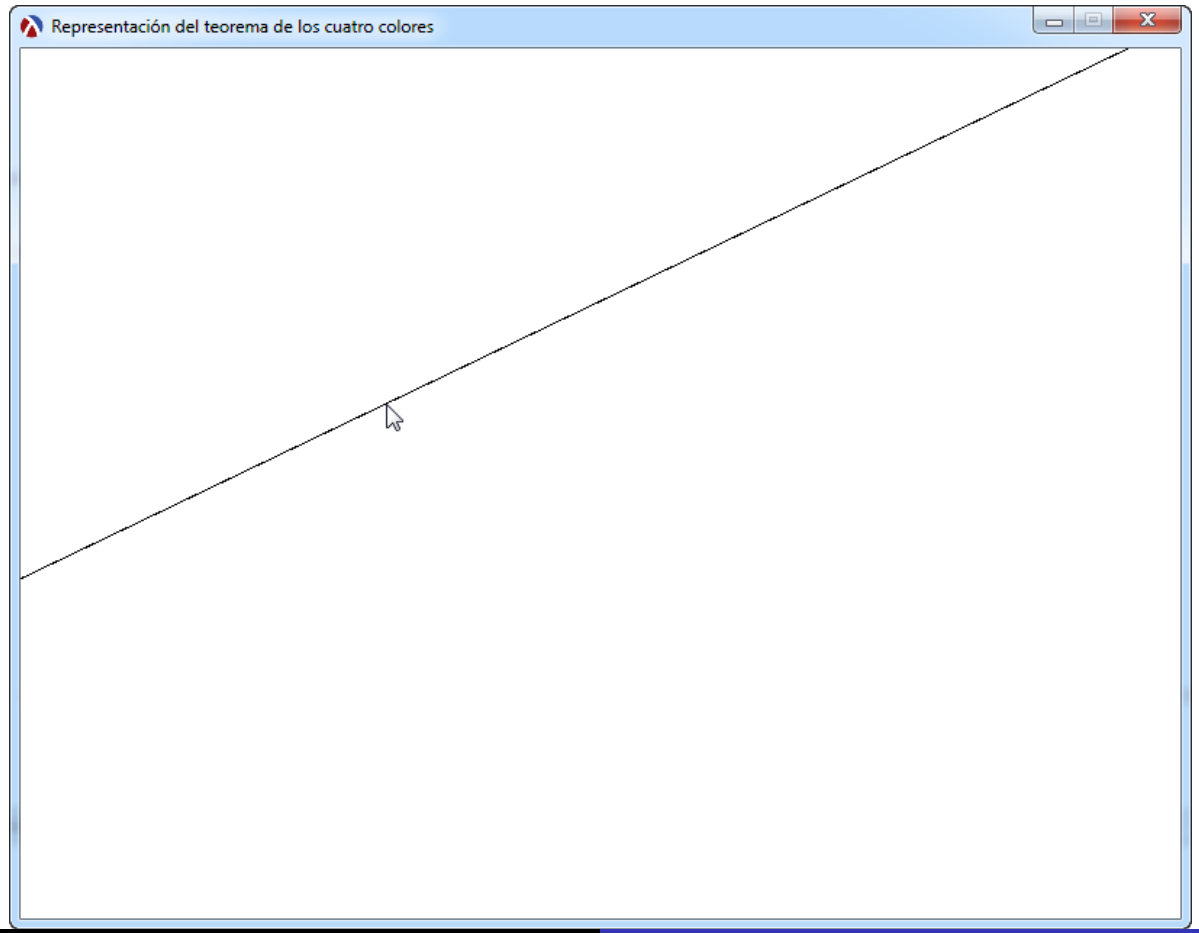
Ejemplo de definición de recta

Primero introducimos uno de los puntos con el ratón.



Ejemplo de definición de recta

Tras introducir el segundo punto, se dibuja la recta y se guarda esta en el Tipo abstracto de dato Recta.



Tipo abstracto de dato Punto de corte

Punto de corte

Representa el punto de corte entre dos rectas. Tiene dos atributos:

- Rectas: Lista con el índice de las rectas que se cortan en ese punto.
- Punto: Lista con las x e y donde esas dos rectas se cortan.

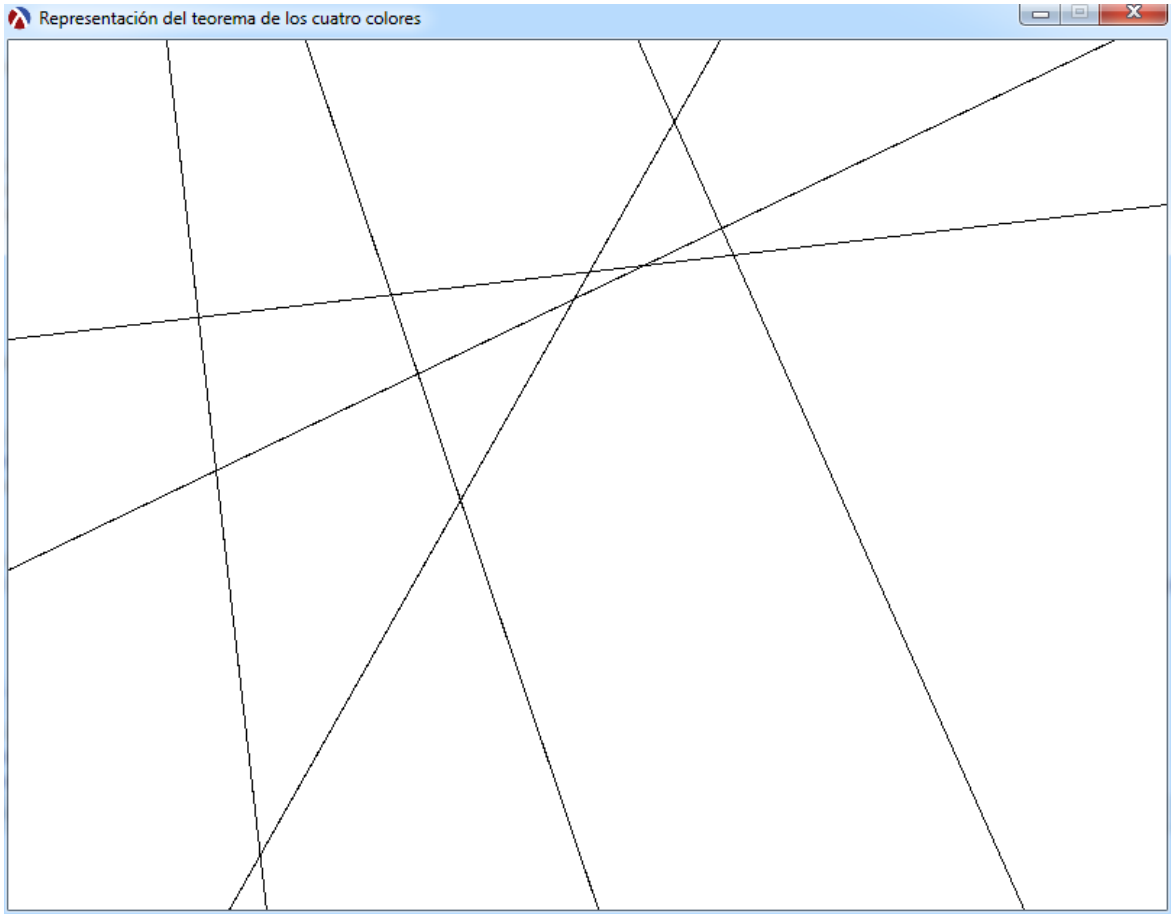
Tipo abstracto de dato Punto de corte

Punto de corte

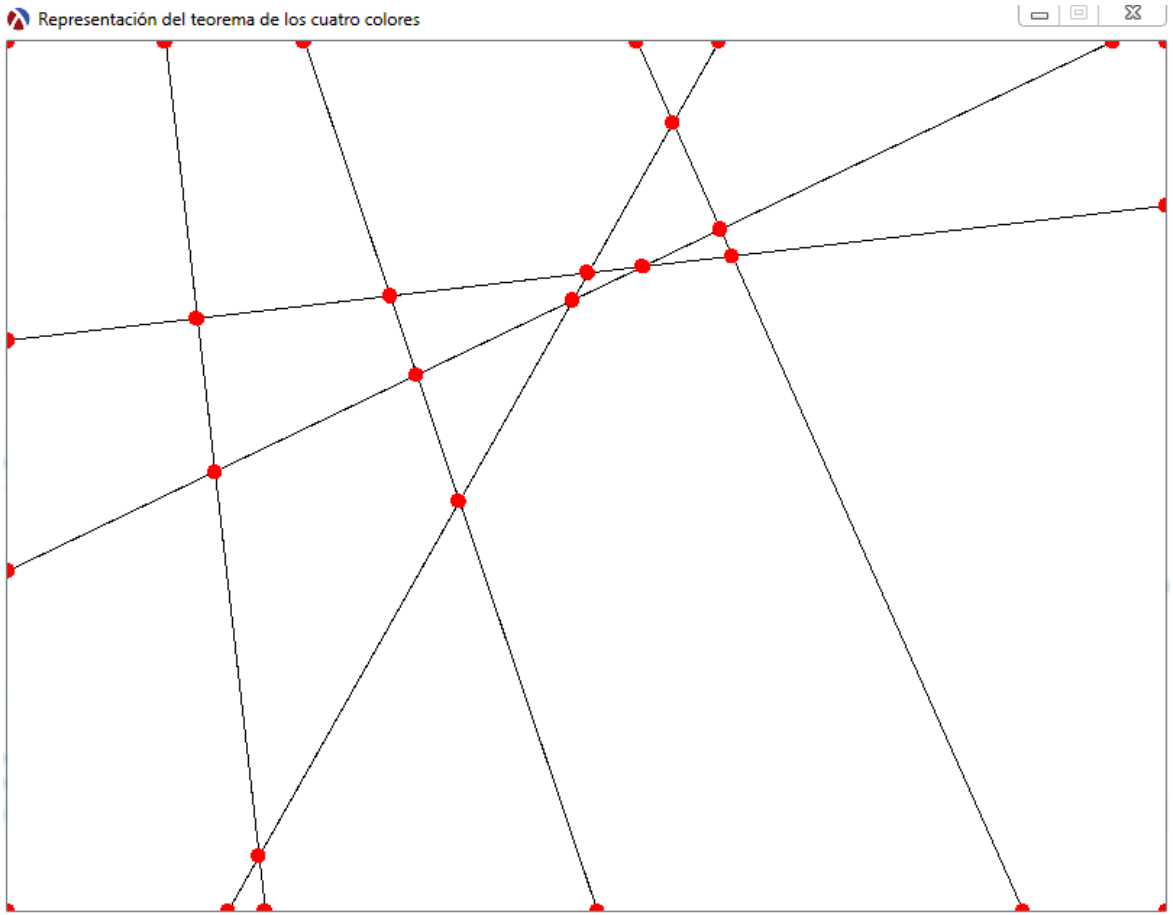
Para calcular los puntos de corte, calculamos la x de dos rectas si la y fuera igual y tras esto, calculamos el valor de la y en la recta con el x conseguido.
Las funciones utilizadas son:

- valorf: Calcula el valor y de una recta en el punto x.
- cruce: Calcula el punto de cruce si dos rectas no son paralelas.
- detectarPuntosdeCruce/detectarPuntosdeCruceMarco

Ejemplo de definición de los puntos de cruce



Ejemplo de definición de los puntos de cruce



Tipo abstracto de dato Par

Par

Representa el segmento definido entre dos puntos de corte y la recta. Tiene tres atributos:

- Recta: TAD recta la cual define ese punto de corte.
- Punto_inicial: TAD punto de cruce con el primer punto del segmento.
- Punto_final: TAD punto de cruce con el segundo punto del segmento.

Tipo abstracto de dato Par

Par

Para calcular los pares, vamos recta por recta consiguiendo los puntos de cruce de forma ordenada y cada dos puntos de cruce contiguos forman un par.
Las funciones utilizadas son:

- getPuntos: Función que consigue los puntos de cruce que pasan por una recta de forma ordenada.
- pares: Función que consigue todos los pares que pasan por una recta.
- getPares/getParesEjes: Consiguen todos los pares.

Tipo abstracto de dato Área

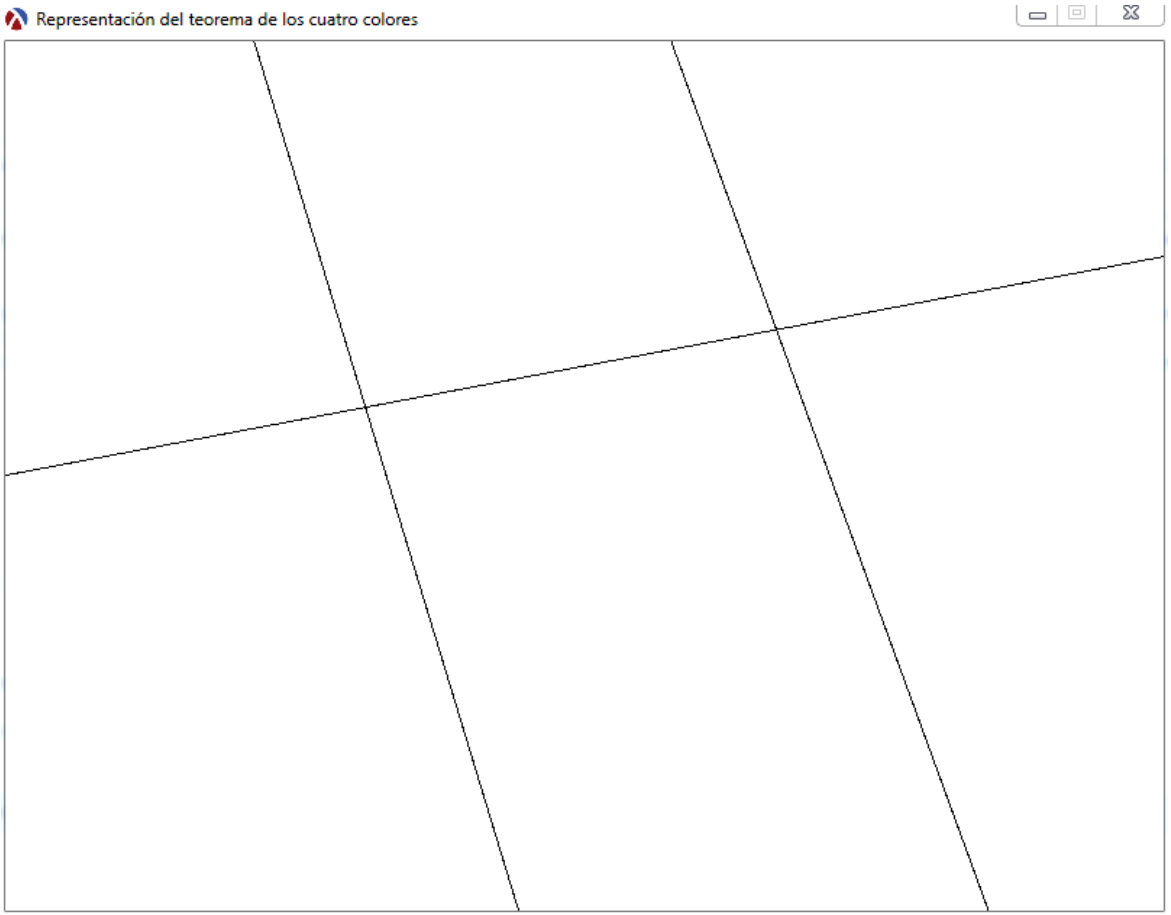
Area

Representa una de las áreas en las que se dividen la figura.
Tiene cuatro atributos:

- Índice: Índice de 1 a n de ese área
- Pares: Lista del TAD par que definen los pares que delimitan un área.
- Áreas adyacentes: Lista con el índice de las áreas que son adyacentes a esta.
- Color: Color del área. 0 si no tiene color.

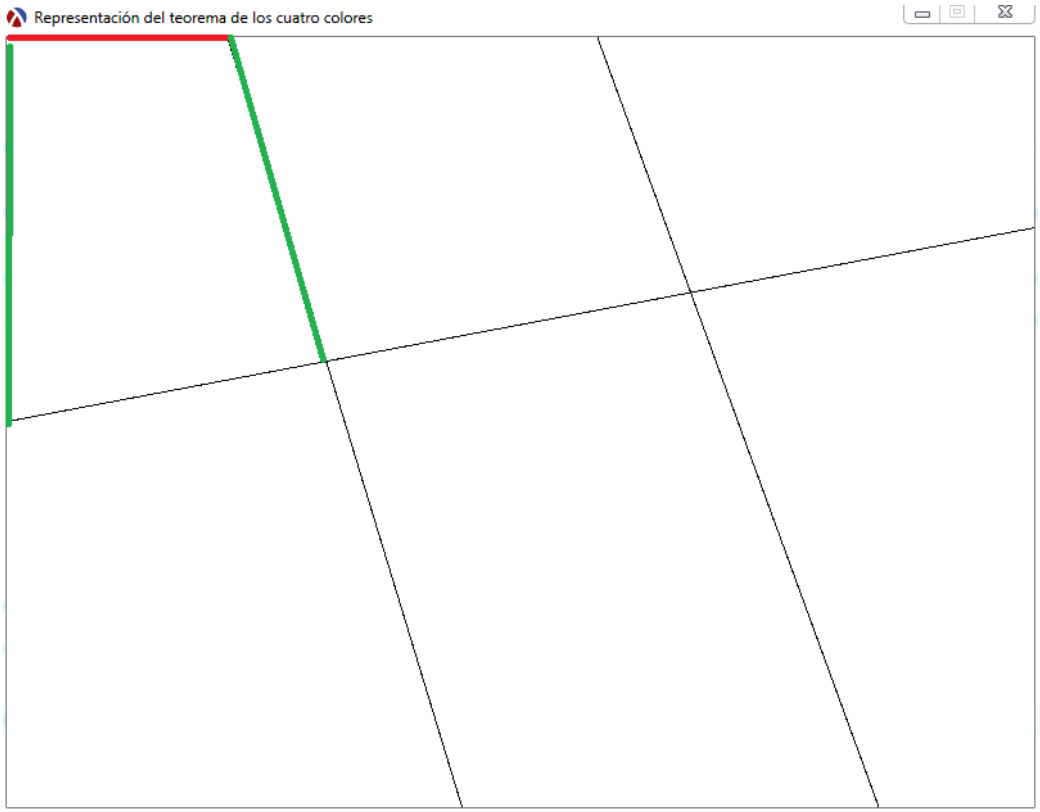
Tipo abstracto de dato Área

Para sacar las áreas solo necesitamos todos los pares de la figura.



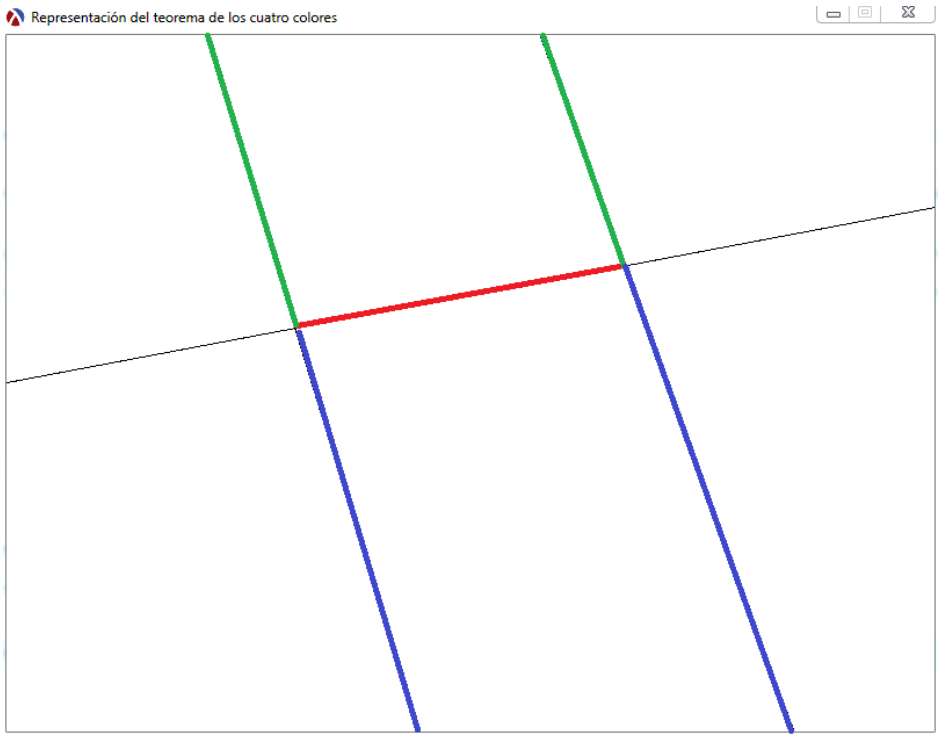
Tipo abstracto de dato Área

El primer paso sería sacar los pares que al menos tengan un punto en común y cuya recta sea distinta y guardarlos en un conjunto. En la figura se muestra esto en el caso del par rojo.



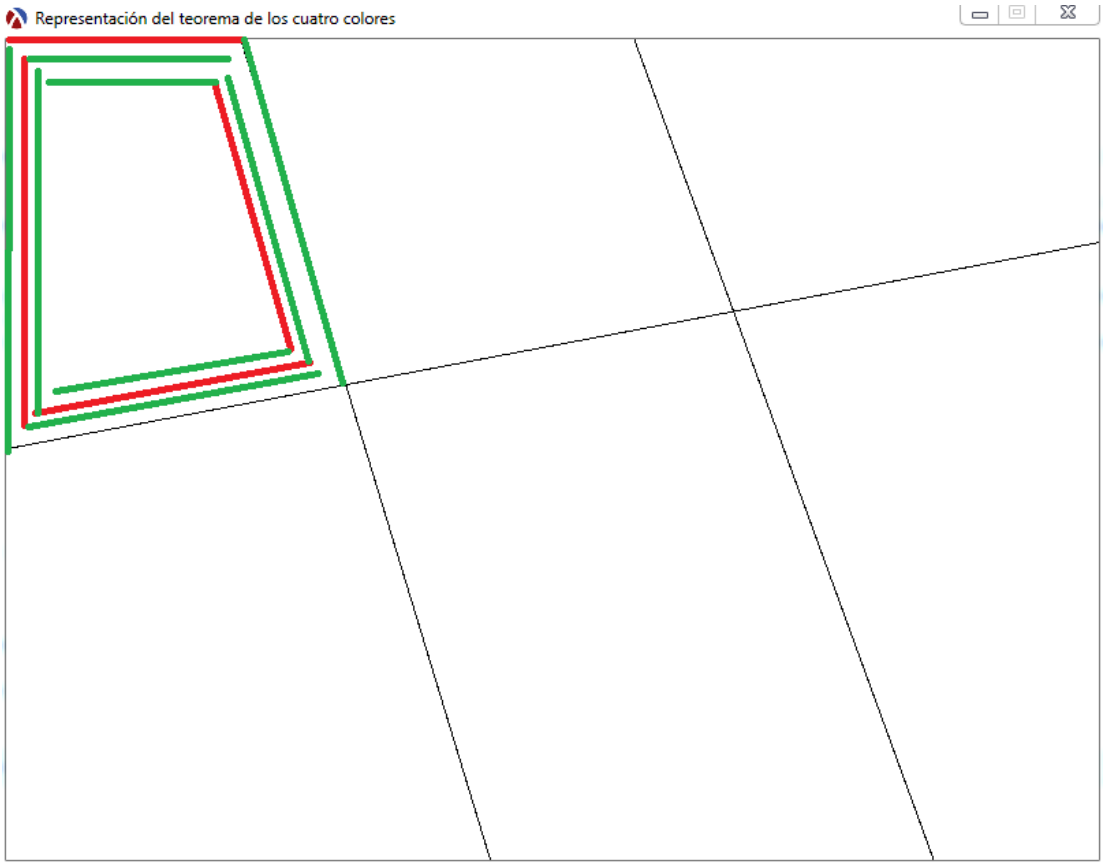
Tipo abstracto de dato Área

Si hay mas de dos pares que cumplan esta condición, en vez de guardarlo todo en un conjunto, guardamos en un conjunto los pares que están a la izquierda de la recta y los que están a la derecha de la recta.



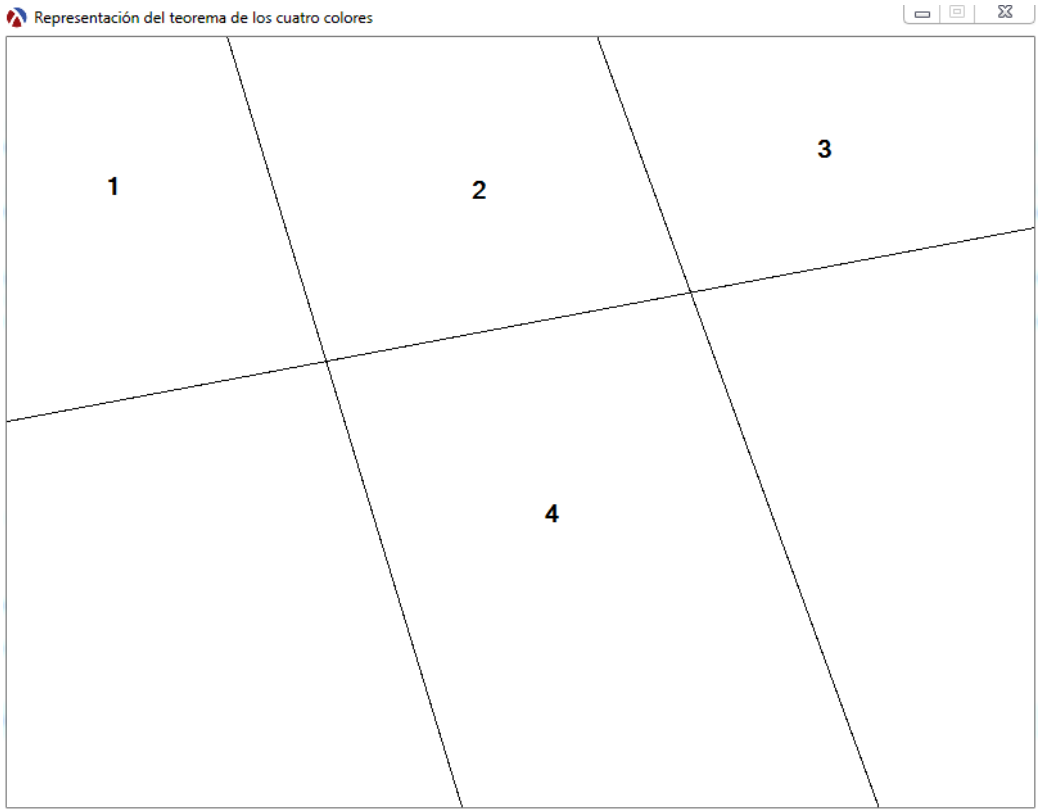
Tipo abstracto de dato Área

Tras esto, vamos uniendo los conjuntos que tengan dos o más pares en común hasta definir los pares que conforman el área.



Tipo abstracto de dato Área

Por último calculamos las áreas adyacentes como las áreas que comparten un par. Por ejemplo, las áreas adyacentes a 2 son 1,3 y 4.



Algoritmo Voraz de coloreado

Algoritmo Voraz de coloreado

Con las áreas que teníamos anteriormente podemos realizar el algoritmo voraz de coloreado. La idea sería:

1. Por cada área que conforma la figura
   1. Vemos los colores que tienen sus áreas adyacentes.
   2. Cogemos el mínimo color de la lista de colores que no tenga un área adyacente.
   3. Pintamos de ese color el área y ponemos ese color al TAD área.

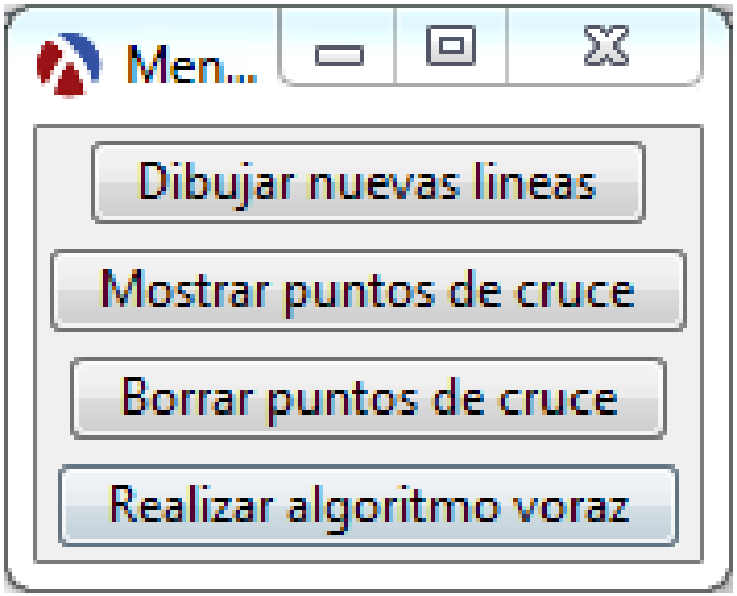
Muestra de los resultados

Resultados

La idea detrás de esto es comprobar como tras dibujar una serie de rectas, las areas definidas entre esta serie de rectas se coloreen de manera que no haya dos areas adyacentes con el mismo color y ver como moyoritariamente el algoritmo converge a una solución de menos de cuatro colores.

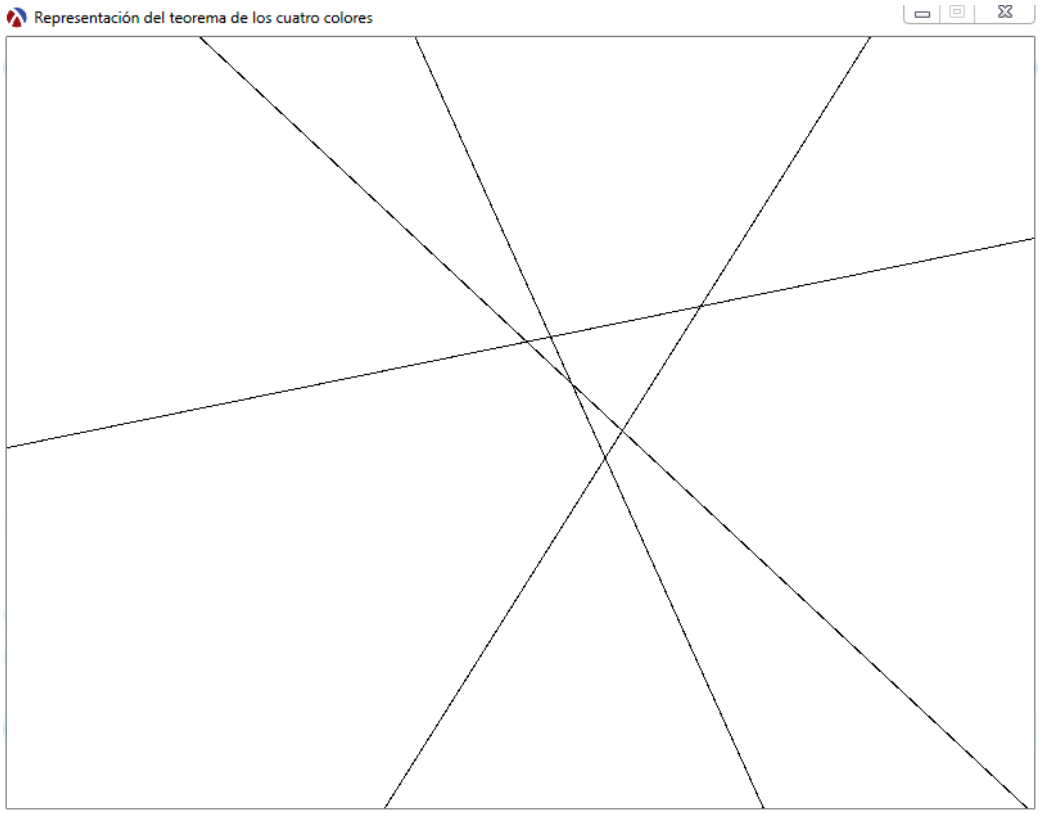
Muestra de los resultados

Para poder realizar esto cree un menú donde hay varias opciones donde cada opción realiza algo sobre la figura.

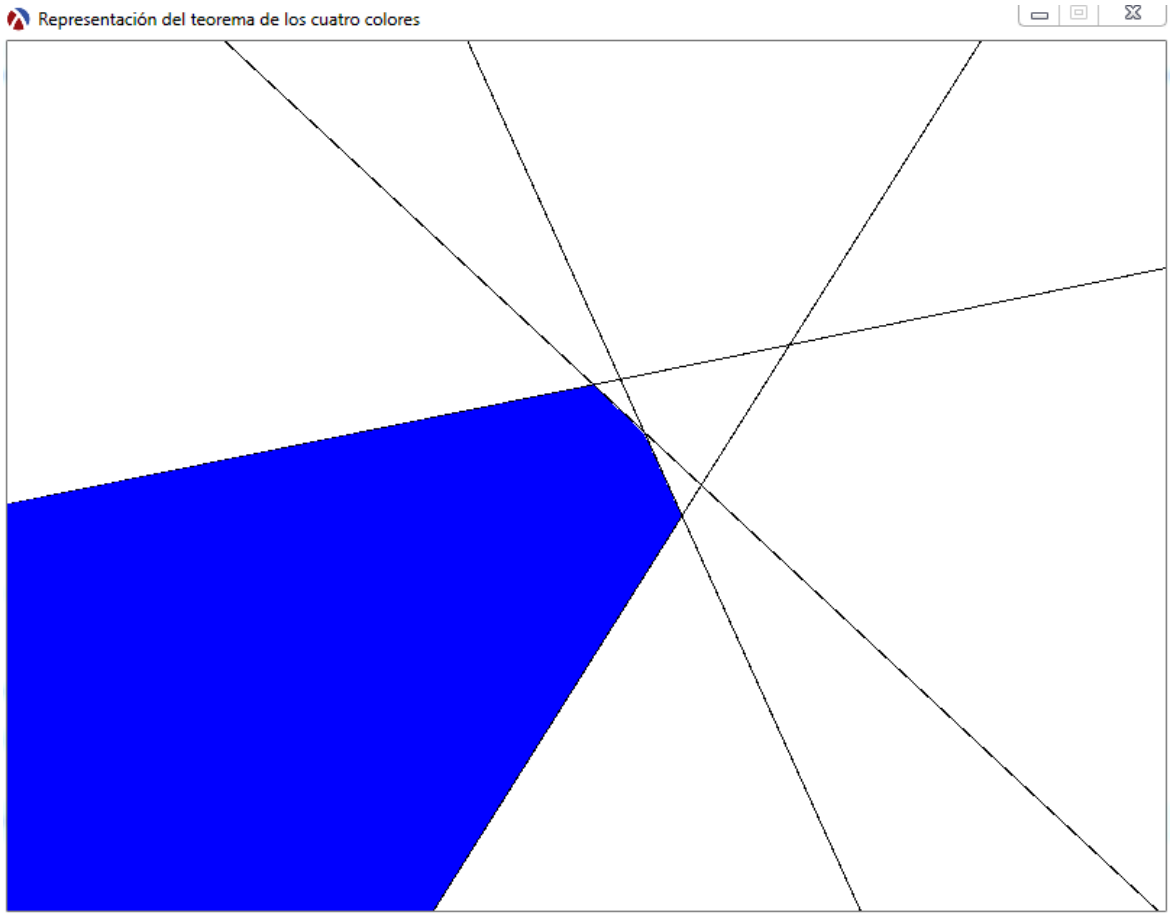


Muestra de los resultados

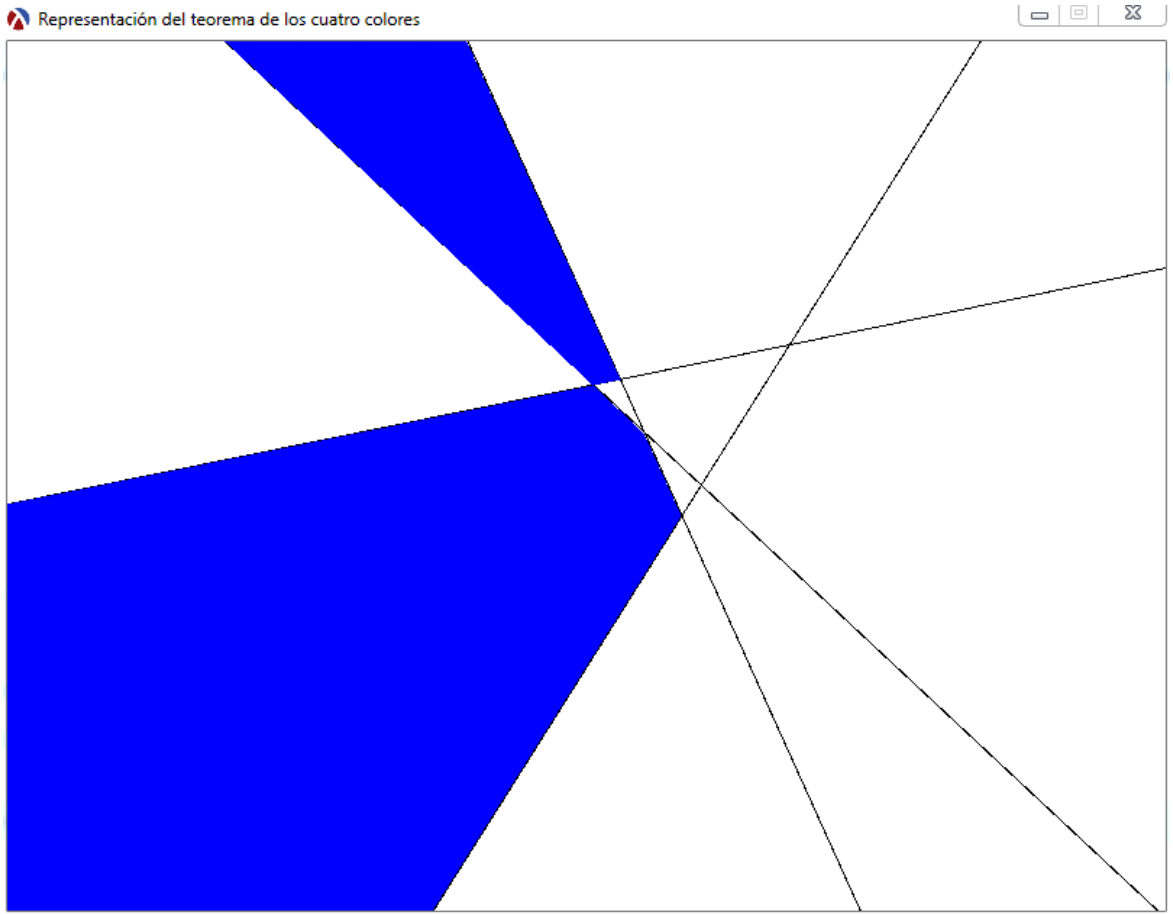
La idea sería que tras dibujar una serie de lineas, como las que se muestra el algoritmo siga paso a paso.



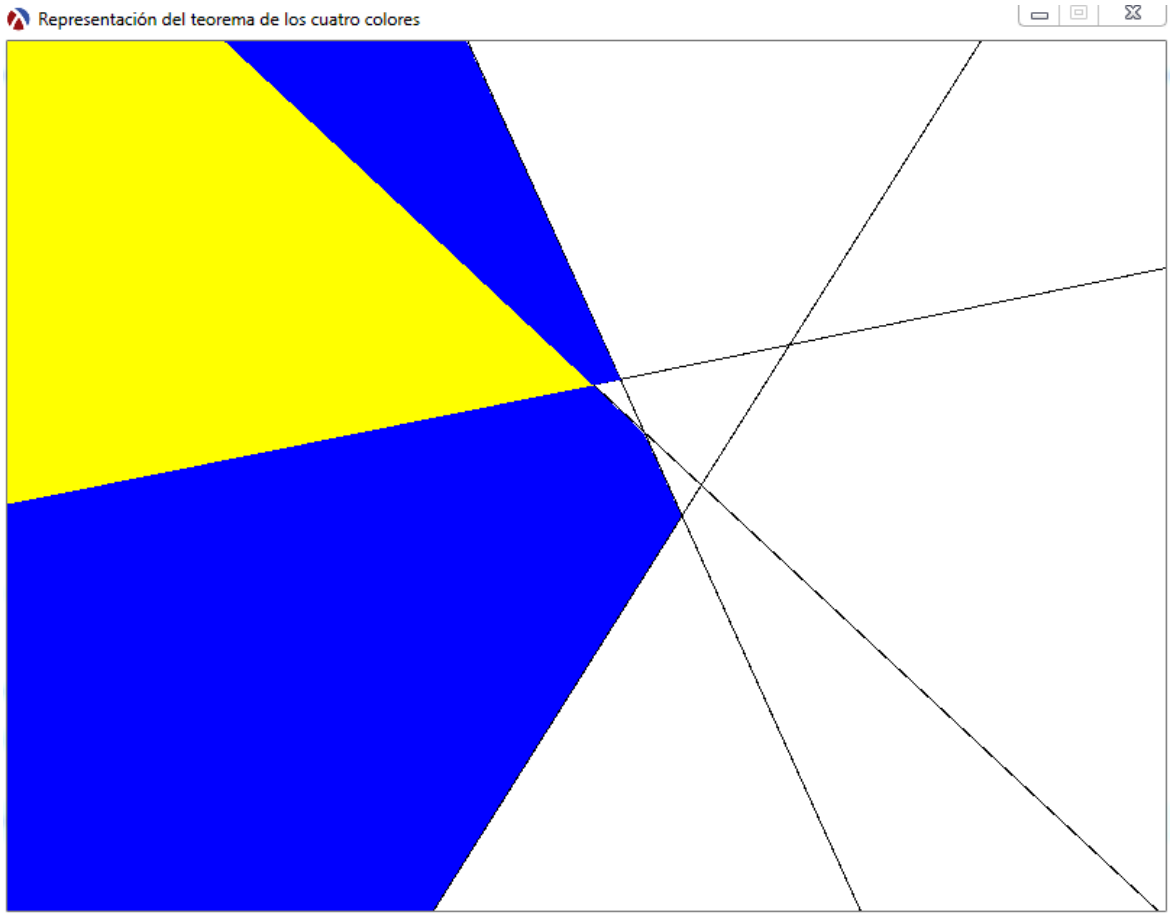
Muestra de los resultados



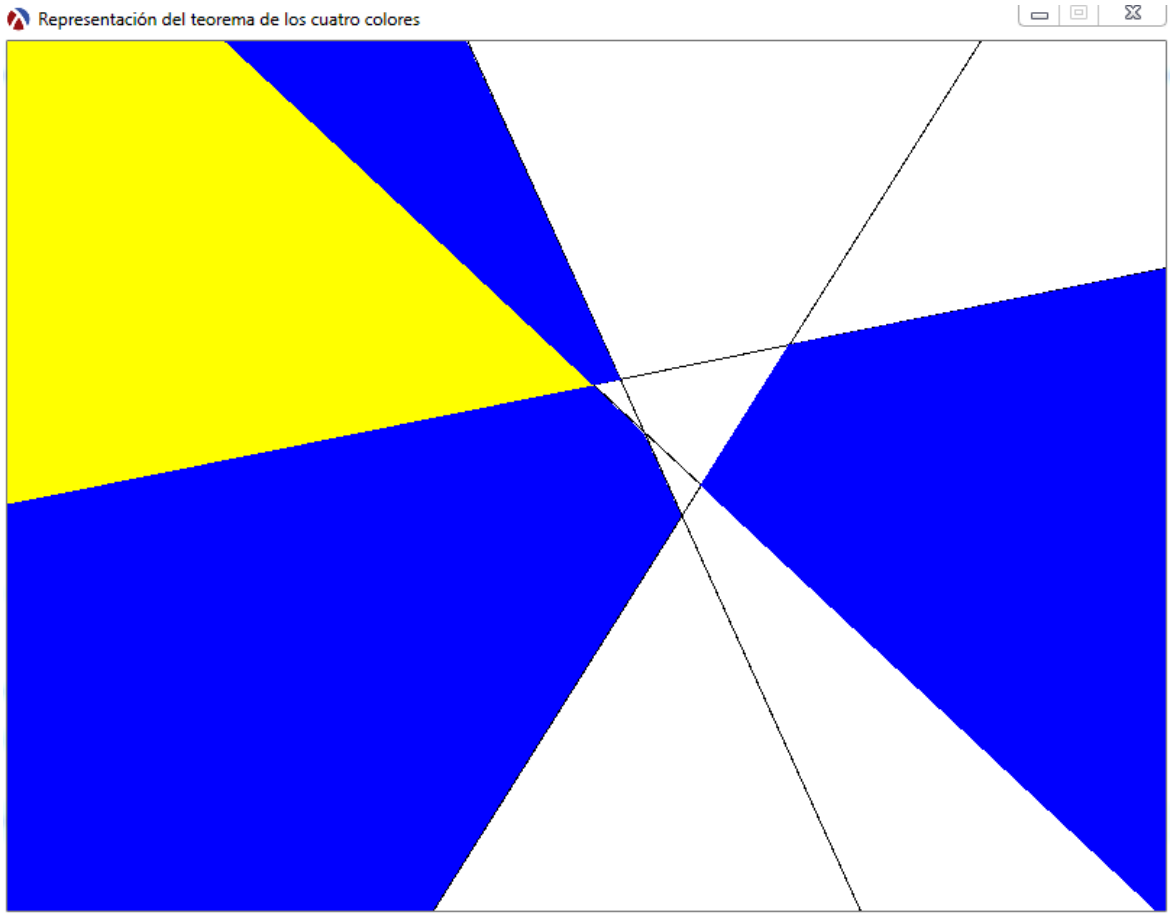
Muestra de los resultados



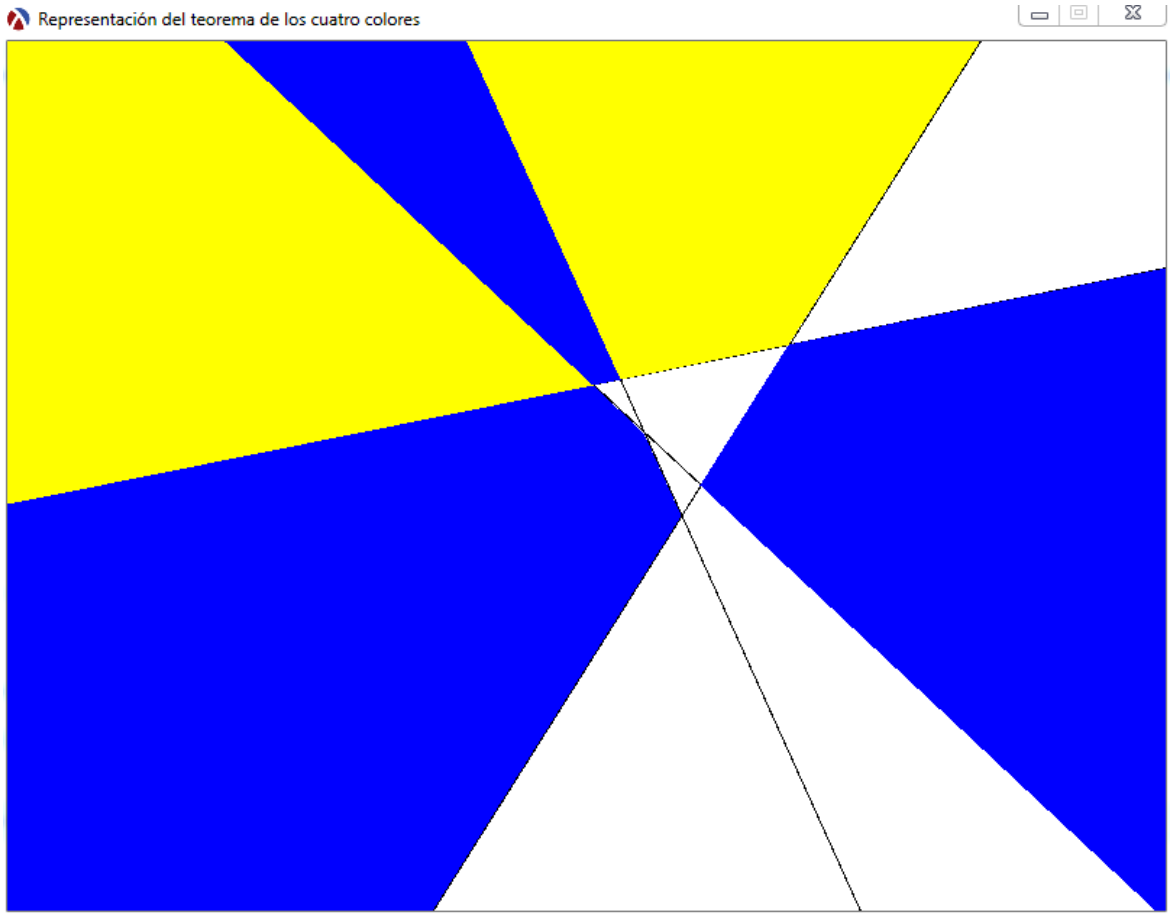
Muestra de los resultados



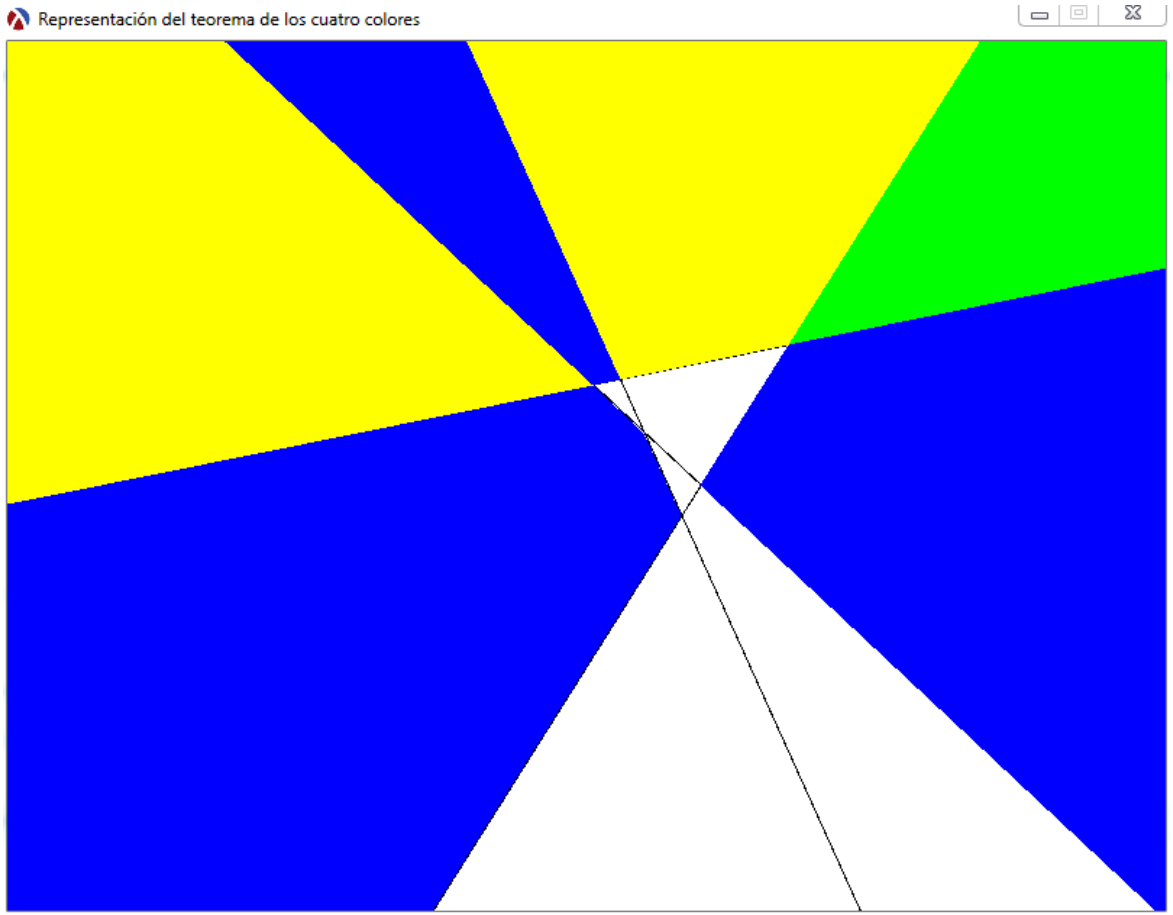
Muestra de los resultados



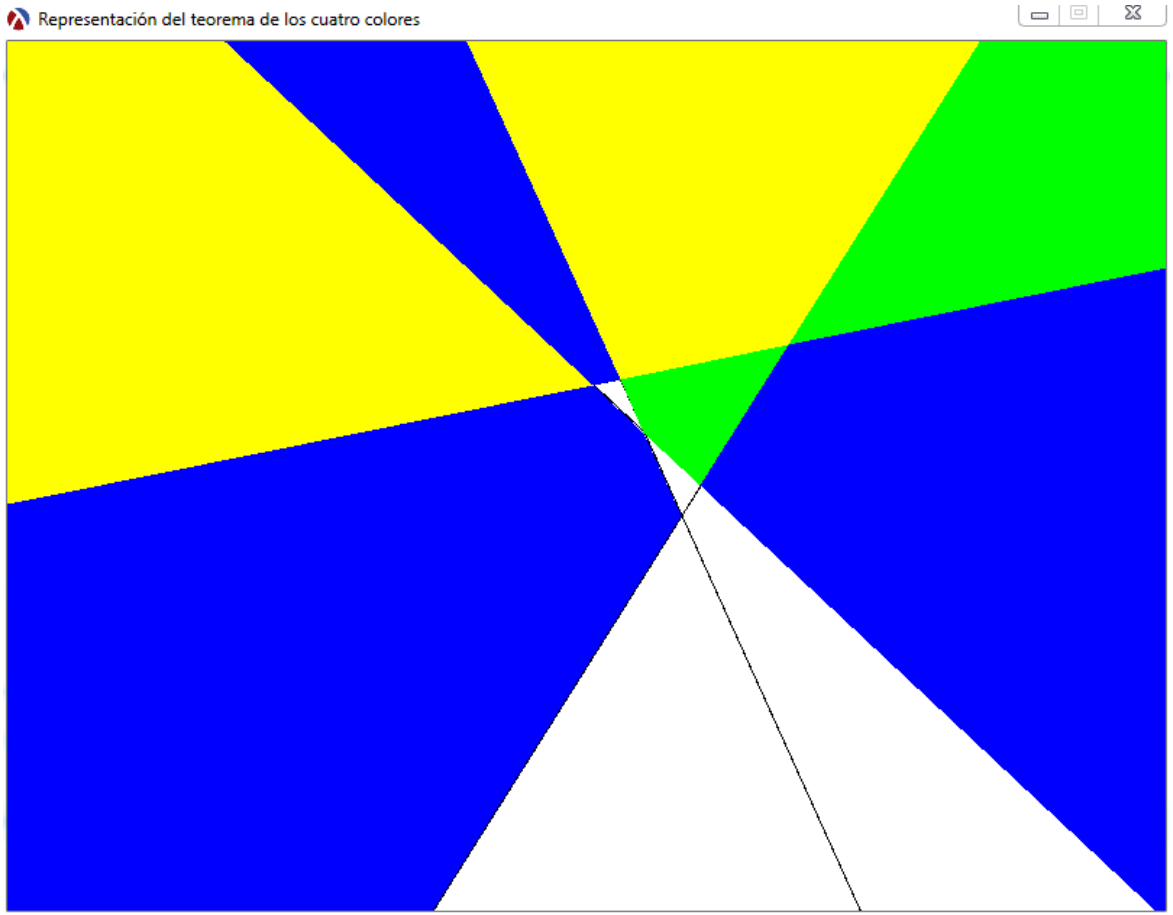
Muestra de los resultados



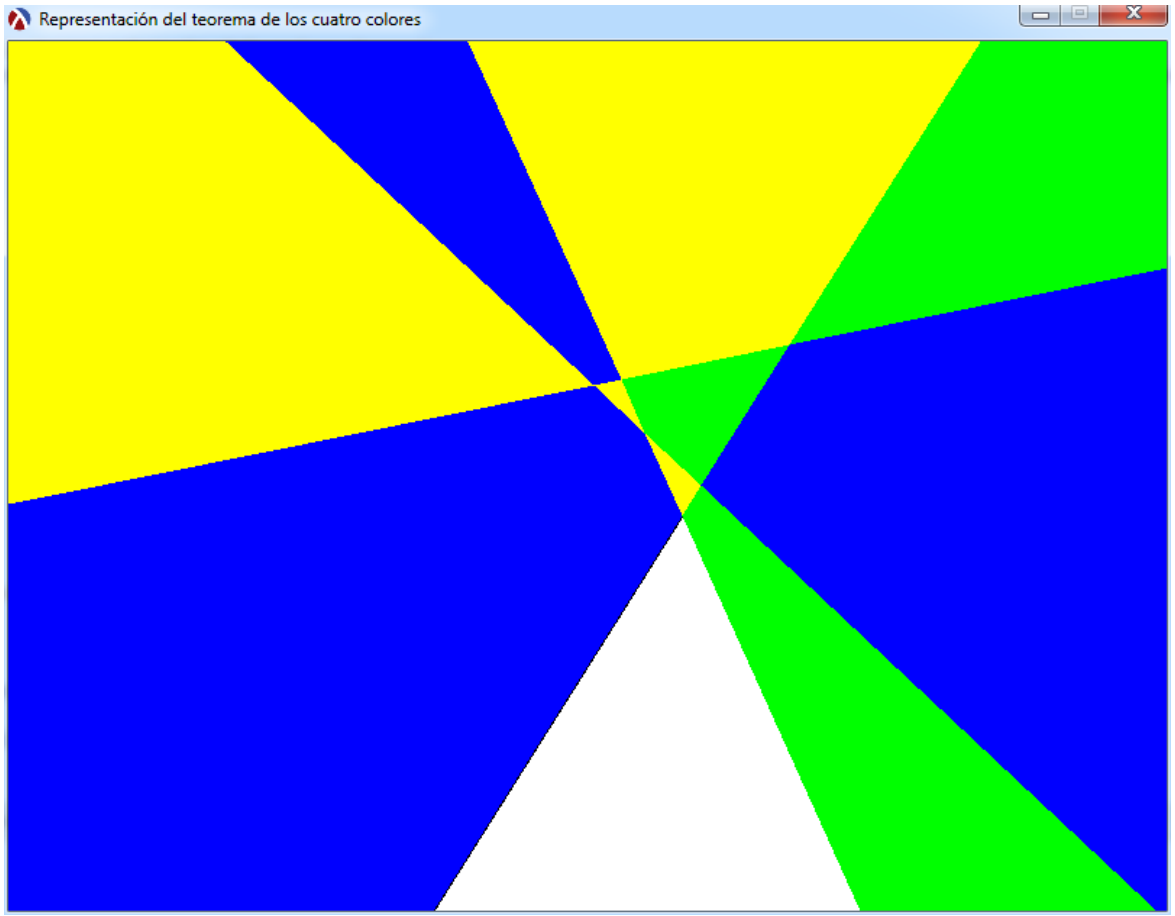
Muestra de los resultados



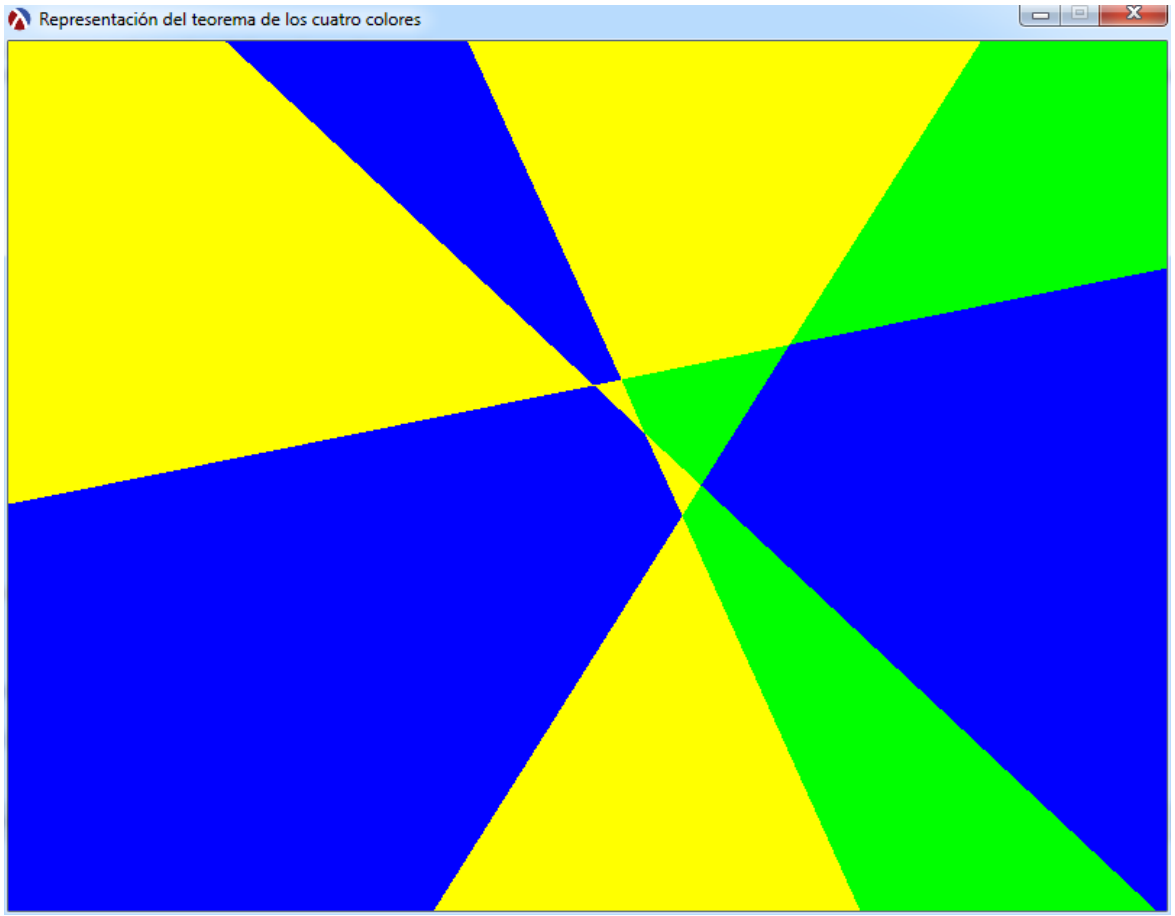
Muestra de los resultados



Muestra de los resultados



Muestra de los resultados



Conclusiones

Conclusiones

- Lo primero que he comprobado es que realmente no he conseguido ninguna configuración para la que consiga hacerlo en más de cuatro áreas.
- En parte el tratamiento con listas ha facilitado mucho mi trabajo.

Conclusiones

Conclusiones

- Lo primero que he comprobado es que realmente no he conseguido ninguna configuración para la que consiga hacerlo en más de cuatro áreas.
- En parte el tratamiento con listas ha facilitado mucho mi trabajo.

Futuras Mejoras

Futuras Mejoras

- Sería ideal que las rectas no tengan que ser infinitas, si no usar segmentos definidos desde un punto hasta otro punto. Esto daría lugar a mapas más complejos.
- Debería implementar el algoritmo de fuerza bruta u otro tipo de algoritmo que dé la solución idónea.

Futuras Mejoras

Futuras Mejoras

- Sería ideal que las rectas no tengan que ser infinitas, si no usar segmentos definidos desde un punto hasta otro punto. Esto daría lugar a mapas más complejos.
- Debería implementar el algoritmo de fuerza bruta u otro tipo de algoritmo que dé la solución idónea.

Bibliografía

- Every Planar Map is Four Colorable. Kenneth Appel and Wolfgang Haken. 1989
- PLT Miscellaneous Libraries: Reference Manual. 2002
- https://docs.racket-lang.org/gui/index.html
- Algoritmos greedy sobre grafos. Universidad de Granada.