UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

Reporte de Proyecto

Teoría Aditiva de los Números y graficación con Haskell

ASIGNATURA

Programación Declarativa 2023-2

ALUMNO

Rosas Franco Diego Angel - 318165330

PROFESOR

Manuel Soto Romero

AYUDANTE

Juan Pablo Yamamoto Zazueta

7 de junio de 2023

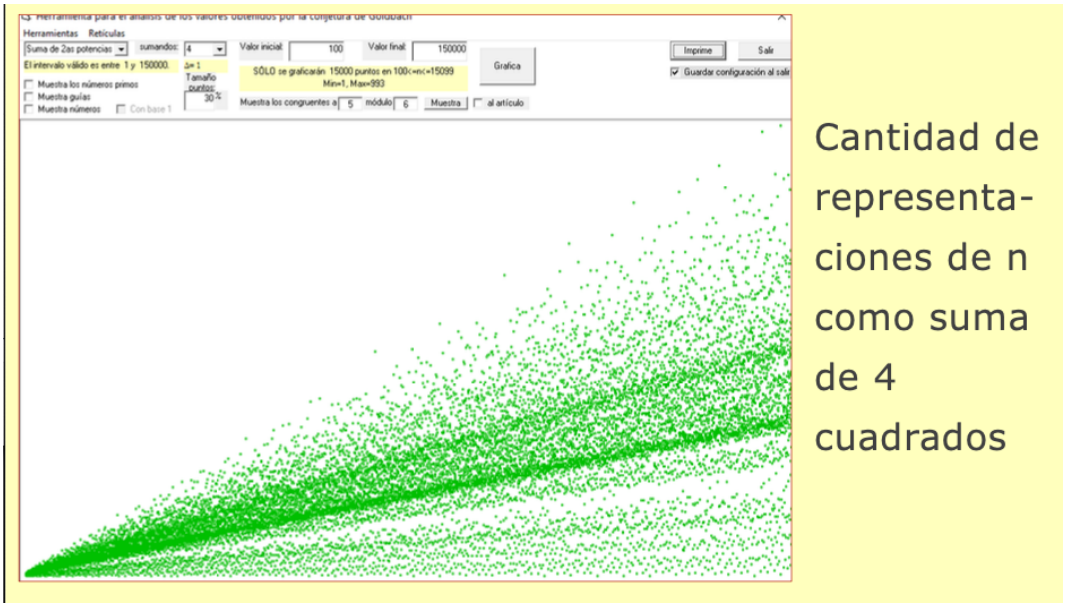
1. Preliminares

1.1. Definición del problema

El problema a resolver partió primero de un problema matemático de la Teoría Aditiva de los Números (TAN), el encontrar la cantidad de representaciones de un número n con suma de 4 cuadrados.
Se buscaría dar una implementación en Haskell de este problema y mostrar una gráfica que mostrara las representaciones para las n de un rango dado.
Además, se buscó sentar una base para la posibilidad de no quedarse solamente con la suma de 4 cuadrados, también poder de tener sumas de 3 o 4 números poligonales junto con una opción gráfica de los resultados.

1.1.1. Antecedentes y revisión bibliográfica del problema

En un inicio este problema me lo propuso una colega matemática mientras le platicaba sobre el proyecto de la materia. En su caso, el tema (TAN) fue abordado en un Seminario en la Facultad de Ciencias por el profesor J. César Guevara Bravo, y en una de las diapositivas de su presentación usada en el seminario venía el problema que se propuso para este proyecto.
En estas diapositivas se veía que se había apoyado un software de graficación para mostrar el comportamiento de los números bajo la propiedad de la suma de cuatro cuadrados.



Sin embargo, parece que el profesor perdió el software usado y el al no tener conocimientos de programación perdió la posibilidad de hacer este tipo de cálculos por compuadora.
Por ello, investigué si había algun software en línea que se le pareciera o alguna implementación hecha en algún lenguaje de programación, y encontré una hecha justamente en Haskell y que igualmente daba una propuesta gráfica, sin embargo, se limitaba al problema de la suma de 4 cuadrados y además contaba con una gráfica de líneas para presentar los resultados, sin embargo, en este tipo de gráfica no se podían apreciar de una forma gráfica comportamientos interesantes de los números, por lo que, si que valía la pena realizar este problema en Haskell y además sumarle la opción de poder manejar los otros números poligonales.

1.2. ¿Por qué Haskell?

Hubo dos principales razones para elegir Haskell:

- La primera porque la intención era usar el paradigma funcional completamente y Haskell al ser puramente funcional nos facilitaba esto.
- Y la segunda tiene que ver con el hecho de que la idea me la propusiera mi colega matemática, y es que cuándo le pregunté porque el profesor no había programado de nuevo el software, me respondió que para

el profesor y en su opinión de ella para varios matemáticos el aprender a programar puede ser algo confuso en un inicio. Y es que Haskell tiene la característica de que al ser declarativo y funcional, el programar en el se parece mucho al dar una definición como tal, lo cuál no debería ser algo ajeno para los matemáticos y de hecho puede ser un motivante para que lo consideren a usar en vez de Python que suele ser uno de los comúnes a elegir entre ellos.

2. Implementación

2.1. Uso de Stack

Primeramente, para un manejo sencillo de bibliotecas y mantenimiento futuro del proyecto y futura documentación que se pudiera generar del mismo a partir de los comentarios hechos en el código (con el formato para documentación con Haddock), se decidió usar Stack.

2.1.1. Archivos principales

Al usar Stack, tendríamos dos archivos, el archivo `Main.hs` que contendría la ejecución o el uso de las funciones relacionadas a TAN, y el archivo `TAN.hs` que justamente contendría todas las funciones relacionadas con la Teoría Aditiva de Números y por lo tanto todo lo relacionado a la parte teorica de nuestro proyecto.

2.2. Archivo TAN.hs

El archivo que nos interesa principalmente es `TAN.hs` pues todo el proyecto se aborda en el.

2.2.1. Estructura de TAN.hs

En el podremos encontrar comentadas las siguientes secciones:

- FUNCIONES AUXILIARES: Donde encontraríamos funciones que si bien no están relacionadas directamente con la Teoría Aditiva de los Números nos ayudarían en funciones siguientes.
- FUNCIONES GENERAR NUMEROS POLIGONALES: En esta sección encontraríamos funciones para poder obtener el número poligonal de un número n dado. Esto conla finalidad de que a futuro estas pudieran ser usadas en las funciones siguientes.
- FUNCIONES TAN: Aqui ahora si encontraríamos las sumas que nos interesan (TAN), en específico la función que resuelve el principal problema propuesto para el proyecto, la de la suma de cuatro cuadrados.
- FUNCIONES GRÁFICAS: Ahora, tocaría definir funciones relacionadas a la parte gráfica, en seguida se aborda un poco más a detalle esto.
- FUNCIONES GRÁFICAS TAN: Y finalmente aqui encontramos gráficas de las funciones de la sección FUNCIONES TAN aplicadas a un determinado rango para así ver el comportamiento de los números.

2.3. Sección FUNCIONES GRÁFICAS y Gloss

Para la representación gráfica se utilizó la biblioteca Gloss, que se define como una biblioteca para dibujar gráficos vectoriales de forma sencilla con el uso de OpenGL de fondo. A continuación se describe como se uso y se logró la creación del gráfico.

2.3.1. Creación del gráfico

- Primero se debían generar los puntos que se le daría a la gráfica, para ello, se genero una lista de tuplas, donde en una tupla (x,y), x sería n y y el resultado de aplicarle la función a n, además se daría un rango a recorrer para n.
- Una vez teneindo los puntos toca generar el gráfico, para ello se definió una "Pintura" que estaba basada en muchas otras, en el código esto es un constructor llamado `pictures`. ¿Y que subpinturas contendría esta general que representaría la gráfica? Serían puntos, pues nuestra gráfica sería una de puntos.

- Entonces, para generar un punto se usó la función `circleSolid`, y para darle su lugar correspondiente en el gráfico se usó `translate` que es un constructor que pone en la pintura general la pintura actual (el circulo) en las coordenadas dadas. Y esas coordenadas se obtienen justamente de los puntos que generamos antes.

- Ya habíendo generado la gráfica con `pictures`, procederíamos a imprimir este gráfico en una ventana del sistema operativo, esto lo haríamos con el tipo `display`. Usaríamos el constructor `InWindow` de este tipo que nos permitiría indicarle el titulo de la ventana, su tamaño y su posición. Ya habíendo creado este tipo, tocaría indicarle el color que tendría de fondo y la pintura (nuestro gráfico) que contendría.

3. Retos del desarrollo

Dejando de lado los problemas que encontré y que abordaré en la siguiente sección, uno de los principales retos fue el conseguir un algoritmo eficiente para poder encontrar la combinación de los números que su suma diera n. Ya que en un inicio eso tenía complejidad exponencial, pero afortunadamente esta pudo ser disminuida con unos pequeños cambios.

4. Cosas a mejorar

Inicialmente, en la propuesta yo quería solamente resolver el problema de la suma de los 4 cuadrados y dar su gráfica similar a la del software del profesor de mi colega, pero en el transcurso del proyecto intente agregar otras sumas como: suma de tres primos o suma de tres cuadrados. Sin embargo, comencé a tener conflictos de tipos cuando intenté hacer sumas de otros poligonales, por lo que, al final por cuestiones de tiempo no pude lograr completar del todo este extra que le quería dar al proyecto.

Además, quería dar una interfaz gráfica para que tuviera una mejor experiencia de usuario la aplicación, pero me encontré con problemas de hilos de ejecución pues la idea era usar otra biblioteca que se encargara de esa interfaz de usuario y que Gloss se encargara de las gráficas como quedaron al final en el proyecto, pero al momento de cerrar una de las ventanas que generabamos con Gloss esto detenía el ciclo del proyecto totalmente.

Lo que intenté fue dar un menú simple en terminal que permitiera elegir entre las opciones de gráficos que ya tenía, ya que, si lograba cerrar estas ventanas y que siguiera el menú sin cerrar todo podía pasar a hacer uso de otra biblioteca para la interfaz gráfica sin problema, pero en esa creación de menú fue cuando encontré el problema de hilos que no me dió tiempo de abordar para entender su funcionamiento y aprovecharlo correctamente.

Afortunadamente, creo que deje funciones útiles para que el programa pudiera seguir trabajandose en futuro y una vez resueltos los problemas que mencioné, pueda usarse correctamente y no quede tan solo como un proyecto final.

Ademas, estoy seguro que el algoritmo de las sumas se puede mejorar aún más, para así poder soportar números más grandes y que no tome tanto tiempo.

5. Referencias

- gloss: Painless 2D vector graphics, animations and simulations. (n.d) Recuperado de: https://hackage.haskell.org/package/gloss

- Guevara Bravo, J. C. (2021, 29 de abril). Seminario de Docencia FC título: Acercarnos a las formas aditivas de los números a través del microscopio.