TEC | Tecnológico de Costa Rica

Tecnológico de Costa Rica

PRIMER PROYECTO, TALLER DE PROGRAMACIÓN

NUMERADORES Y DENOMINADORES PARA UN COCIENTE DADO

Estudiante:
Luis Daniel Barboza Alfaro

12 de abril, 2024

RESUMEN EJECUTIVO

El proyecto que se presenta a continuación, se basa en desarrollar un programa que sea eficiente para encontrar todas las posibles combinaciones de números de cinco dígitos, que al dividirse entre sí, den como resultado un número C previamente especificado.

También se debe especificar un límite R que indique la cantidad máxima de veces que un dígito puede ser repetido en la combinación (se incluye tanto numerador y denominador), para entender mejor esta función, supongamos que tenemos que C=2 y R=4, esto permitiría la combinación 44446/22223=2, ya que no hay dígitos que se repitan más de 4 veces, por el contrario no sería válido 44444/22222=2 ya que se excede el número de repeticiones permitidas.

El código adicionalmente recibe otra entrada de un entero T el cual nos indica el número de casos de prueba a resolver, donde $1 \leq T \leq 100$. Cada caso de prueba consiste en dos enteros, C (resultado de la división), y R (veces que se puede repetir un número), donde $1 \leq C \leq 20$ y $1 \leq R \leq 5$. Para la salida, debe ser una línea por cada combinación de números que cumpla las condiciones previamente establecidas, ordenada de menor a mayor en los numeradores. Por cada caso de prueba se deben imprimir las combinaciones, una por línea, además al acabar cada caso, se debe dejar una línea en blanco. Si hubiera algún caso en el que ninguna solución dé el resultado ingresado no se imprime nada.

Para la solución, se tuvo que investigar primero sobre divisibilidad, listas, comandos, entre otras cosas, esto para obtener primero la forma matemática más eficiente para llegar al resultado esperado,ya que se cuenta con un límite de quince segundos para que el algoritmo sea ejecutado en su totalidad.

Una vez investigado, se implementa un algoritmo que genera todas las posibles combinaciones de números de cinco dígitos, utilizando iteración hasta que a supere 99 999 el cual es el número máximo que se puede representar respetando las condiciones dadas, esto lo realiza usando el concepto de divisibilidad encontrado.

Además, el código realiza una verificación para las restricciones que establece R, en cada combinación se cuenta el número de veces que se repite cada dígito, luego se compara con R, y si la cantidad excede dicha condición, la combinación es descartada. Las combinaciones que sí cumplen con las restricciones, son consideradas válidas, y serán impresas.

Uno de los principales problemas con este código fue la condición del tiempo, este fue el factor que provocó una investigación más a fondo sobre divisibilidad. Otro problema fue al imprimir los resultados, ya que al ser demasiadas combinaciones generadas, imprimir por medio de $print$ en el terminal (en este caso de Visual Studio Code) se demoraba mucho tiempo, ya que requería permisos y realizar otras funciones antes de poder imprimir, lo que hacía que no se cumpliera la ejecución del código en el plazo establecido, porque aunque el algoritmo hicera la búsqueda rápidamente, cuando se imprimían los resultados, consumía los segundos restantes y hasta más.

Luego de buscar una solución adecuada y eficiente para los problemas encontrados, se implementó el algoritmo, realizando varias pruebas, para poder evaluar su eficiencia y buen funcionamiento. Los resultados de las pruebas mostraron que el algoritmo fue capaz de encontrar todas las combinaciones válidas de números en un tiempo razonable, para corroborar esto, se creó una gráfica que nos muestra el tiempo de ejecución para una cantidad de casos T. Además, se realizaron pruebas para verificar que todas las combinaciones generadas cumplieran con las restricciones establecidas por R.

Con estos resultados, podemos respaldar la eficiencia y utilidad del código en la busqueda de combinaciones, tal y como se solicitaba.

INTRODUCCIÓN

El presente documento expone el proceso para solucionar un problema planteado, se muestra

desde cuál fue la solución, hasta los resultados de las pruebas del código y conclusiones obtenidas, el problema consiste en la búsqueda de combinaciones de números que cumplan con restricciones de repetición de dígitos. El proyecto se enfocó en el desarrollo de un código eficiente que pueda generar todas las posibles combinaciones de números de cinco dígitos, considerando las restricciones de repetición establecidas por el usuario. En el proyecto se emplean varias funciones matemáticas para lograr mejorar su eficiencia y tiempo de ejecución.

MARCO TEÓRICO

Este proyecto fue desarrollado usando python3 como lenguaje de programación para desarrollar el programa, se utilizó Visual Studio Code, ya que tiene una interfaz buena, fácil de entender, ademas de su eficiencia y agilidad en la programación, y se usó GitHub para subir la carpeta con los archivos del proyecto.

A continuación, se hará una breve reseña sobre antecedentes e historia de python3, Visual Studio Code y github.

GitHub:

Fue creada por Chris Wanstrath , PJ Hyett , Tom Preston-Werner y Scott Chacon en Febrero de 2008 en San Francisco, Estados Unidos y hasta la fecha sigue estando ubicada en el mismo lugar. Es una plataforma en la Nube que sirve como almacenamiento de código fuente y archivos de un proyecto de Software, Web, Aplicación Móvil, etc.[1].

Visual Studio Code:

La primera versión beta de Visual Studio Code se publicó en noviembre de 2015 y la primera versión estable, Visual Studio Code 1.0, se publicó en abril de 2016. Se publica una nueva versión a principios de cada mes (salvo en enero). Su desarrollador es Microsoft.[2].

python3:

La primera versión de Python fue lanzada en 1991, por Guido van Rossum quien era un programador holandés, en el año 2000, se lanza Python 2.0, que empezó a ser desarrollado por la comunidad, esta versión tenía nuevas características como completa recolección de basura y completo soporte a Unicode. En el 2008 se lanza Python 3.0, una versión mejor, y era incompatible con las anteriores. Python tiene facilidad para la programación orientada a objetos, imperativa y funcional, además contiene implícitas estructuras de datos, que permiten hacer tareas complejas en pocas líneas [3].

La biblioteca usada fue time, ya que en este comando nos ayudaba a medir el tiempo de ejecución del programa.

DESCRIPCIÓN DE LA SOLUCIÓN

El código inicia con la función $limite$, la cual tiene la función de verificar que en cada combinación no se supere el límite de R, que es la variable que guarda la cantidad máxima de repeticiones permitidas a cada dígito. Además, para combinaciones de cuatro dígitos, añade un cero a la derecha ya que debe ser tomado en cuenta.
En la función $generar$, se generan todas las combinaciones válidas, empleando la regla de divisibilidad:

$$a/b = c \implies b = ac$$

En esta función se buscan las combinaciones que den como resultado C, que es el resultado esperado, así aplicando esta fórmula se irán multiplicando todos los números a por C, ayudándonos a ir obteniendo todos los resultados posibles, luego se hace la verificacion para el numerador y denominador, par ver que se cumpla la condición R antes mencionada. Este proceso se repetirá para la cantidad T de casos solicitados, generando una cadena de texto que se mostrará con las combinaciones posibles para los casos.

Como se mencionó en el resumen ejecutivo, la principal complicación fue el tiempo, en la busqueda de combinaciones se solucionó al implementar divisibilidad, y en el caso del $print$, lo que se hace es simplemente usar direccionales en la terminal de Linux para abrir primeramente los casos de prueba (input), luego se ejecutará y las combinaciones serán almacenadas en un archivo de texto que se abrirá o creará directamente desde la teminal, este sería el $output$, gracias a esto, el tiempo de ejecución reduce considerablemente, haciendo que el código se ejecute antes de los 15 segundos, por lo que la complicación fue solucionada, en la Figura 1 podemos ver lo rápido que ejecuta inclusive 100 casos.

CONCLUSIONES

En este proyecto se logró analizar y comprender más a fondo el proceso de desarrollo de algoritmos y su aplicación en la resolución de problemas numéricos complejos.

Se pudo desarrollar un programa que fuera eficaz en la generación de combinaciones que sean válidas cumpliendo las restricciones de repetición. Además de usar técnicas para hacerlo óptimo.

Tras la realización de varias pruebas, se afirma que el código creado es capaz de encontrar las combinaciones esperadas en tiempo razonable, además de que las combinaciones generadas cumplen con las restricciones. Los resultados son correctos, se generan la cantidad de casos de prueba indicados, separados por un renglón en blanco, además de mostrar cada una de las combinaciones de cinco dígitos posibles, que den como resultado un número C, respetando el límite de repetición para los dígitos, por lo que podemos garantizar la fiabilidad del algoritmo.

APRENDIZAJES

Durante la realización de este proyecto tuve varios aprendizajes, empezando por buscar más sobre el backtraking [**back**], logré mejorar el uso de listas [4], la importancia de hacer óptimo un código, y tratar de que se ejecute en el menor tiempo posible, aprendí sobre la historia y antecedentes de pyhton, además de aprender a implementar la función “time”, para el control del tiempo de ejecución de los programas [5].

BIBLIOGRAFÍA

[1] Anónimo. *Que es GitHub, Historia y otros Detalles*. 2019. URL: https://blog.nubecolectiva.com/que-es-github-historia-y-otros-detalles/.

[2] Anónimo. *Visual Studio Code*. 2024. URL: https://www.mclibre.org/consultar/informatica/lecciones/vsc.html.

[3] Challenger-Pérez, Ivet and Díaz-Ricardo, Yanet and Becerra-García, Roberto Antonio. *El lenguaje de programación Python*. 2014. URL: https://www.redalyc.org/articulo.oa?id=181531232001.

[4] Juan José Lozano Gómez. *Listas en Python: El tipo list y operaciones más comunes*. $s.f.$. URL: https://https://j2logo.com/python/tutorial/tipo-list-python/.

[5] Julio César Echeverri Marulanda. *Medir Tiempo de Ejecución en Python*. 2015. URL: https://https://julioecheverri.wordpress.com/2015/12/25/medir-tiempo-de-ejecucion-en-python/.

ANEXO

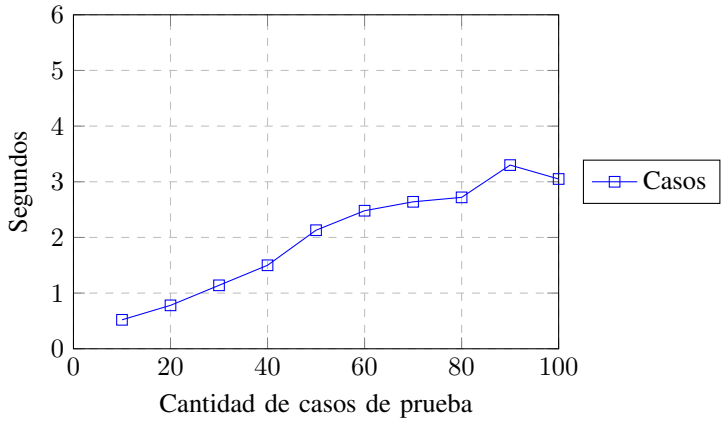

Figura 1. Gráfica con diferente cantidad de casos de prueba