INFORME DE LABORATORIO 1

Autores: *Angie Paola Jaramillo Ortega, Juan Manuel Rivera Florez*

Laboratorio de Electrónica Digital 3
Departamento de Ingeniería Electrónica y de Telecomunicaciones
Universidad de Antioquia

Resumen

Este informe presenta el desarrollo e implementación de un algoritmo en lenguaje C para calcular permutaciones gráciles, un problema combinatorio con aplicaciones en teoría de grafos y telecomunicaciones. La solución emplea un enfoque basado en backtracking con poda temprana para optimizar la búsqueda, reduciendo el número de caminos explorados en comparación con una búsqueda exhaustiva. Se evalúa el desempeño del algoritmo en función del tiempo de ejecución y el tamaño del conjunto de entrada, demostrando su efectividad hasta ciertos valores de N. Sin embargo, se observa que para valores elevados de N, el tiempo de ejecución se incrementa exponencialmente, lo que plantea desafíos computacionales significativos.

Palabras clave: permutaciones gráciles, lenguaje C, algoritmos combinatorios, complejidad factorial.

Introducción

Las permutaciones gráciles representan un problema combinatorio de gran interés en el ámbito de las matemáticas discretas y la computación. Estas estructuras consisten en arreglos ordenados de números enteros donde el valor absoluto de las diferencias entre elementos consecutivos cumplen dos condiciones fundamentales: deben ser todas distintas entre sí y deben cubrir exactamente el conjunto de valores 1, 2, ..., N-1 para un arreglo de tamaño N. Esta particularidad genera problemas relevantes en el diseño de redes de comunicación y la teoría de grafos, donde encuentran aplicaciones prácticas en la optimización de recursos y la asignación eficiente de frecuencias.

El desarrollo de un algoritmo eficiente en lenguaje C para calcular estas permutaciones enfrenta desafíos computacionales significativos debido a su naturaleza factorial, lo que exige estrategias de optimización robustas. El objetivo principal radica en manejar valores de N en el rango de 2 a 50, incorporando un sistema de control de tiempo que permita interrumpir el cálculo al superar un límite predefinido. Para abordar esta complejidad, se plantea como solución una implementación de backtracking con poda temprana que permita superar las limitaciones relacionadas con el crecimiento exponencial de la búsqueda. La metodología empleada combina el diseño de un algoritmo recursivo con mecanismos de poda eficientes y mediciones de tiempo de ejecución.

Marco teórico

Permutación

Una permutación es un reordenamiento de los elementos de un conjunto en una secuencia particular. Para un conjunto de N elementos distintos, existen N! (N factorial) posibles permutaciones, donde $N! = N(N-1)...1$. Este concepto fundamental de combinatoria subyace al estudio de las permutaciones gráciles.

Permutaciones gráciles

Una permutación $(a_1, a_2, ..., a_N)$ se denomina grácil si cumple dos condiciones [1]:

- Las diferencias $|a_{i+1} - a_i|$ son únicas para todo i $\in$ 1, 2, ..., N-1
- Las diferencias forman exactamente el conjunto 1, 2, ..., N-1

Backtracking con poda

El backtracking es una técnica de búsqueda y exploración de soluciones en problemas combinatorios. Consiste en probar todas las posibles opciones de manera recursiva y, si una opción no lleva a una solución válida, se retrocede (backtrack) para intentar otra alternativa.[2]

El backtracking con poda mejora este proceso descartando caminos innecesarios antes de explorarlos completamente, usando condiciones o para reducir el número de posibilidades a evaluar. Esto optimiza el tiempo de ejecución y evita cálculos inútiles.

Implementación

La función main es el punto de entrada del programa. Recibe dos argumentos de línea de comandos: N (el tamaño del conjunto de números a permutar) y M (el tiempo máximo de ejecución en minutos). Valida que N esté en el rango $1 < N < 50$ y que M sea positivo. Luego, inicializa los arreglos necesarios para rastrear las permutaciones, los números usados y las diferencias absolutas entre números consecutivos. Llama a la función recursiva encontrarPermutaciones para generar todas las permutaciones "gráciles"posibles dentro del tiempo límite. Finalmente, imprime el número total de permutaciones gráciles encontradas y el tiempo total de ejecución.

La función encontrarPermutaciones implementa un algoritmo recursivo de backtracking para generar todas las permutaciones "gráciles"de un conjunto de números del 1 al N. Una permutación es considerada "grácil"si las diferencias absolutas entre números consecutivos son únicas. La función comienza verificando si el tiempo transcurrido desde el inicio de la ejecución ha superado el límite especificado (M minutos). Si el tiempo límite se excede, la función termina inmediatamente para evitar continuar con cálculos innecesarios.

En cada nivel de recursión, la función intenta colocar un número en la posición actual del arreglo arregloNumeros. Para cada número candidato, verifica que no haya sido usado previamente y que, si no es la primera posición, la diferencia absoluta con el número anterior no haya sido utilizada antes. Si el número cumple con estas condiciones, se coloca en la posición actual, se marcan las restricciones correspondientes (el número como usado y la diferencia como utilizada), y se llama recursivamente a la función para llenar la siguiente posición.

Cuando se completa una permutación válida (es decir, cuando todas las posiciones están llenas), se incrementa el contador de permutaciones válidas (totalPermutaciones). Después de cada llamada recursiva, la función deshace los cambios realizados (desmarcando el número y la diferencia) para explorar otras posibles combinaciones. Este proceso continúa hasta que se hayan explorado todas las permutaciones posibles o se alcance el tiempo límite.

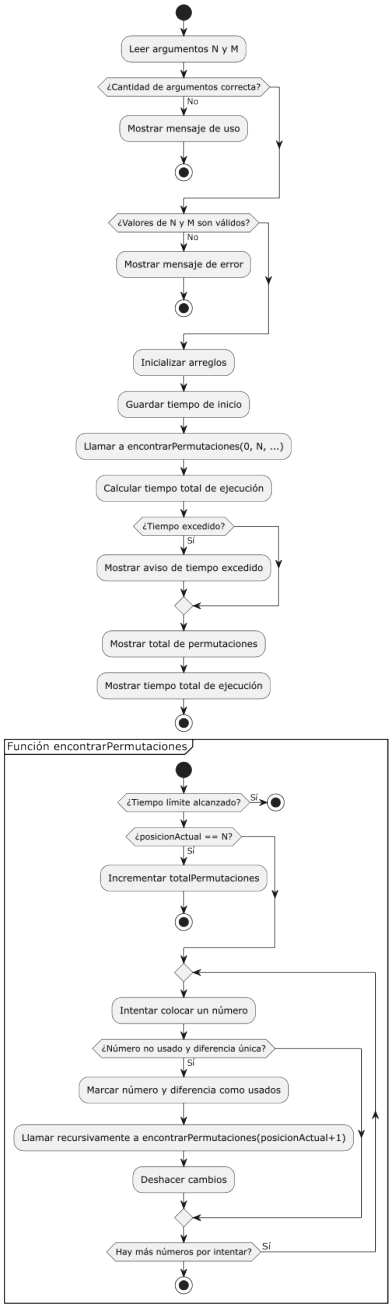

Figura **0-1**: Diagrama de flujo.

Tiempos de ejecución

Cuadro **0-1**: Tiempos de ejecución

N	Tiempo de ejecución (seg)
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0.01
11	0.079
12	0.359
13	2.104
14	13.285
15	108.812
16	681.004

Figura **0-2**: Gráfica de los tiempos de ejecución en función de N

Los tiempos mostrados en la gráfica representan valores promedios obtenidos a partir de múltiples ejecuciones del programa en un entorno específico. Sin embargo, en la práctica, el tiempo real de ejecución puede diferir debido a diversos factores. Entre ellos, las especificaciones del hardware juegan un papel fundamental. Por lo tanto, los valores mostrados en la gráfica deben interpretarse como referencias generales en lugar de tiempos absolutos.

Conclusiones

El modelo propuesto, basado en un algoritmo de backtracking, cumple con el objetivo de generar permutaciones gráciles de un conjunto de números del 1 al N. Los resultados observados muestran que el algoritmo es capaz de encontrar todas las permutaciones válidas dentro de las restricciones impuestas (diferencias únicas entre números consecutivos). Esto confirman la viabilidad del enfoque propuesto, destacando al mismo tiempo los retos computacionales asociados con este tipo de problemas combinatorios.

Para valores elevados de N, el tiempo de ejecución aumenta considerablemente, lo que puede generar que se supere el límite especificado de tiempo. A pesar de estas limitaciones, el trabajo aporta una comprensión clara de los desafíos asociados a problemas combinatorios y valida el enfoque adoptado para abordar esta problemática.

Bibliografía

[1] Departamento de Ingeniería Electrónica y Telecomunicaciones. Práctica 1: Permutaciones gráciles. Technical report, Universidad de Antioquia, 2025.

[2] B. y Poda. Backtracking y Poda. `https://docs.google.com/presentation/d/1hsQm00G8TxYrmCjDXB1uMM1PaczZ46HqWCLZAx3OXqU/htmlpresent`, 2020. [Accessed: 30-Mar-2025].