**INFORME DESAFIO 2**

Juan José Quiceno Pabón

Programa de Ingeniería en Telecomunicaciones

Universidad de Antioquia

Facultad de Ingeniería

Augusto Salazar

Aníbal Guerra

Medellín, Colombia

2024

**ANÁLISIS INICIAL**

En las primeras observaciones respecto al desafío planteado, puedo identificar un problema que no es particularmente complejo en términos de lógica aplicada. Sin embargo, la verdadera dificultad radica en la elección de los algoritmos que implementaré, ya que, a diferencia del primer desafío propuesto, creo que el tiempo de codificación se reducirá significativamente, pero vendrá acompañado de una mayor inversión de tiempo en el análisis y diseño de la solución.

En resumen, se solicita desarrollar una red metro, la cual tendrá diferentes líneas interrelacionadas por medio de estaciones de transferencia en cada una. Estas líneas tendrán n números de estaciones al gusto del usuario con un x tiempo de recorrido entre cada una de ellas, para que así, a la hora de simularlo se puedan hacer los respectivos cálculos teniendo en cuenta la eliminación o el añadido de algunas estaciones durante la ejecución del programa.

Como ideas iniciales me surge una estructura de datos muy sencilla para una red metro con una sola línea, un arreglo con n espacios donde se almacenaran cada una de las estaciones, pero en el caso de una red con múltiples líneas la metodología cambia, ya que ahora tengo que agregar un arreglo extra que contenga esas nuevas estaciones, y la idea es que en el caso de una nueva línea, en el espacio de la estación de transferencia se guarde ese arreglo, de esta forma no hay que manejar arreglos independientes y desconectados del resto, teniendo siempre como referencia la primera estación que representara la transferencia entre líneas. Ya en cuanto al tiempo de recorrido entre estaciones, tengo varias ideas, pero no muy claras aún, por lo que dejare eso a una etapa posterior del desarrollo.

Ya para la parte de simulación, en vista de que el calculo del recorrido solo puede hacerse en 1 línea especifica, simplifica mas las cosas, ya que es aplicar una suma entre los elementos que acompañan las estaciones dentro de los arreglos.

**Otras posibles implementaciones:**

* Darle a la estructura de datos una forma bidimensional, donde haya n arreglos que almacenen cada una de las estaciones junto con su tiempo para llegar a la siguiente estación.
* En el caso de otra línea es lo mismo, solo que la primera estación será la referenciada como estación de transferencia.

**ETAPAS DEL DESARROLLO**

1. **Etapa Inicial del Desarrollo**: Durante esta fase inicial, nos enfocaremos exclusivamente en el análisis del problema y en la construcción del informe basado en las conclusiones obtenidas. La creación del algoritmo consistirá en la inclusión de ideas claras y concisas que nos ayudarán a definir el siguiente paso en el proceso de desarrollo. Cabe mencionar que el progreso se reflejará en el repositorio “Desafio\_2” en GitHub a medida que avancemos día a día.
2. **Etapa de Codificación**: En esta etapa, daremos prioridad a establecer las bases del programa. Comenzaremos por implementar los métodos y funciones más sencillos, para luego ir construyendo progresivamente el algoritmo final. Evitaremos abordar directamente los métodos o funciones más complejas del programa, como, por ejemplo, el cálculo del tiempo de recorrido entre estaciones, sin antes tener una base de datos bien estructurada.
3. **Etapa de Pruebas**: Durante este período, supervisaremos el comportamiento del programa durante su ejecución y las diferentes respuestas que proporciona ante diversos escenarios, como excepciones o datos de entrada incorrectos. Esta supervisión permitirá reforzar aún más el algoritmo, gracias a los problemas que surjan durante la ejecución.

**¿CÓMO FUNCIONA EL PROGRAMA?**

**AVANCES**

* **Día 1 – 25 de abril**:
* Se comienza con el proceso de análisis del problema y desarrollo del informe.
* **Día 2** – **26 de abril:**
* Se define la metodología y el orden que se seguirá durante el desarrollo del problema en el informe.
* **Día 2** – **26 de abril:**
* Se creo una clase **Metro**, que de momento será mas que suficiente para el desarrollo del programa.
* Se realizan diferentes pruebas para determinar cuál será la mejor forma de administrar una base de datos con 1 o varias líneas de metro.
* Se realizaron pruebas en Online GDB con un arreglo unidimensional que almacena 5 estaciones, allí se colocaron a prueba 2 métodos, uno que agregaba una nueva estación y otro que las eliminaba.
* **PROBLEMAS**: Al realizar pruebas con arreglos unidimensionales, no imaginé como aplicar esos algoritmos a un arreglo que contenga varias líneas, ni de como modificar el arreglo principal para agregar esa línea.
* Surge una idea para manejar el arreglo principal, esta consta en crear sub-arreglos en cada método que maneje la línea que se especifique.
* Surge una idea de tener un método que me permita restablecer distintas variables en base a la línea de metro que se desee modificar.
* **Día 3** – **28 de abril:**
* Se comenzó a codificar con las ideas que surgieron durante los últimos días, resultando en un gran avance en la parte de manejo de la estructura de datos principal.
* Se definieron los atributos que conforman la clase **Metro**.
* Se decidió que el arreglo principal será tratado como un arreglo de arreglos, siendo fiel al desafío propuesto con una red que contiene diferentes líneas.
* Se logro desarrollar diferentes métodos que permiten añadir estaciones y líneas, consultar la información general del sistema y de líneas en específico.
* Para llevar un orden durante la ejecución, se aplicaron diferentes menús mediante switch para manejar el flujo del programa a gusto del usuario.
* **PROBLEMA**: Todavía no se tiene muy claro cómo se alinearán los datos correspondientes al tiempo con la estructura de datos diseñada.