Problema #1: Scooter

Greedy

September 18, 2024

1 Problemática

1.1 ¿De qué trata el problema?

El campus de una universidad cuenta con n edificios, numerados desde 1 hasta n. En cada uno de estos edificios puede haber planeada una clase de matemáticas, o una clase de ciencia de la computación, o ninguna clase (nunca hay planeadas clases de ambas materias en un mismo edificio). Además de esto, en cada edificio hay a lo sumo un profesor, y cada profesor es experto de una de las materias, es decir, que hay profesores de matemática y de ciencia de la computación.

Como trabajador de University Express Inc., tu trabajo es transportar de manera rápida y amena a los profesores para que puedan impartir sus clases. Para esto se te ha otorgado nada más y nada menos que un scooter (una motorina vamos), en la que cabes tú y a lo sumo un pasajero.

Inicialmente serás la única persona en el scooter. Cuando llegues a uno de los edificios de la universidad puedes dejar o recoger a un profesor en dicho edificio. Para conseguir tu tarea se te ha permitido conducir a cada uno de los n edificios a lo sumo una vez, en el orden que desees (tambien puedes elegir en qué edificio empezar)

Al final de tu recorrido, en cada edificio donde haya planeada una clase de matemáticas debe haber un profesor experto en dicha materia, mientras que en cada edificio con una clase de ciencia de la computación planeada debe haber un profesor experto en esta materia. Planea un itinerario de viajes con el que puedan ser impartidas con éxito todas las clases planeadas.

1.2 Entrada

La entrada del problema constará de 3 elementos:

- Un número entero n ($1 \leq n \leq 2000$) que será la cantidad de edificios de la universidad
- Un string de longitud n conformado por los caracteres (`M`, `C`, `-`), el caracter en la posición i determina la materia de la clase que está programada en el edificio i: `M` representa una clase de matemáticas, `C` una de ciencia de la computación y `-` representa que no hay ninguna clase planeada en ese edificio.
- Un string de longitud n conformado otra vez por los caracteres (`M`, `C`, `-`), pero esta vez para definir los profesores que se encuentran inicialmente en cada edificio. Una `M` o `C` en la posición i representa que en

el edificio i hay un profesor de matemáticas o de ciencia de la computación respectivamente mientras una - representa que dicho edificio está vacío.

1.3 Salida

Se debe imprimir en una primera línea un entero l, que representa el número de operaciones que hace el itinerario realizado. Luego se deben imprimir l líneas donde se muestren cuáles fueron las operaciones realizadas. Dichas operaciones pueden ser las siguientes:

- `DRIVE x`: Ir al edificio con el número x ($1 \leq x \leq n$).
- `PICKUP`: Recoge al profesor que se encuentra en el edificio actual.
- `DROPOFF`: Deja al pasajero que se lleve en ese momento en el edificio actual.

Para considerar correcto un itinerario debe cumplir que no se hagan dos instrucciones `DRIVE` hacia el mismo edificio, para cada instrucción `PICKUP` debe haber un profesor en el edificio y ninguno en el scooter, para cada instrucción `DROPOFF` debe llevarse un pasajero en ese momento, no se puede volver a recoger a un profesor que se acaba de dejar en un edificio y por supuesto debe cumplirse la condición de solución del problema, es decir, en cada edificio debe haber un profesor de la materia que se requiera, ya sea porque fue transportado ahí o porque ya estaba en el edificio desde un principio.

2 Solución

2.1 Idea general de la solución

El problema plantea un desafío logístico, donde debemos mover a los profesores de manera eficiente entre los edificios. La solución se basa en un enfoque Greedy, que busca resolver el problema tomando decisiones localmente óptimas en cada paso. Tenemos las variables: n (cantidad de edificios), p(array que indica los profesores en cada edificio) y c(array que indica las clases programadas en cada edificio)

Primeramente vamos a definir como **Desajuste** a un estado del problema que incumple la condición de solución del problema, es decir, un desajuste ocurre cuando el profesor presente en un edificio no coincide con la clase que se imparte en ese mismo edificio, o cuando no hay profesor en un edificio que necesita uno para una clase específica. De manera un poco más formal podemos decir que un desajuste en un edificio i ($1 \leq i \leq n$) ocurre si y solo si alguna de las siguientes condiciones se cumple:

1. **Hay una clase pero no un profesor adecuado:**
    - Si en el edificio i se imparte una clase de matemáticas (`c[i] = M`) pero no hay un profesor en dicho edificio (`p[i] = -`) o hay un profesor de ciencia de la computación (`p[i] = C`)
    - Si en el edificio i se imparte una clase de ciencia de la computación (`c[i] = C`) pero no hay un profesor en dicho edificio (`p[i] = -`) o hay un profesor de matemáticas (`p[i] = M`)
2. **Hay un profesor pero no se imparte la clase correcta:**
    - Si en el edificio i hay un profesor de matemáticas (`p[i] = M`) pero no se imparte una clase en dicho edificio (`c[i] = -`) o se imparte una clase de ciencia de la computación (`c[i] = C`)
    - Si en el edificio i hay un profesor de ciencia de la computación (`p[i] = C`) pero no se imparte una clase en dicho edificio (`c[i] = -`) o se imparte una clase de matemáticas (`c[i] = M`)

Por tanto podemos decir que un problema con 0 desajustes está resuelto, ya que si el problema tiene 0 desajustes significa que en cada edificio donde se imparte una clase hay un profesor de dicha materia. Más adelante veremos que hay dos tipos de desajustes que son despreciables, es decir, no es necesario eliminarlos para resolver el problema. El objetivo del algoritmo es

eliminar todos estos desajustes necesarios, moviendo profesores entre edificios.

Para esto primeramente se agrupan los edificios en 6 categorias según el tipo de desajuste que presentan (nótese que en la definición de desajuste se plantean 6 condiciones que generan un desajuste), esto nos permite identificar claramente cuáles edificios tienen un problema de desajuste y qué tipo de profesores necesitamos mover.

El enfoque Greedy utilizado resuelve los desajustes uno a uno. Para esto se priorizan los edificios con un profesor que no coincide con la clase programada y los vacíos que requieren profesores. En cada paso, se mueve a un profesor del edificio equivocado a uno que necesita su especialidad. Esto garantiza que se resuelvan los desajustes de manera progresiva. Se lleva además una condición de balance para priorizar una materia sobre otra, es decir, si hay más edificios que requieren profesores de un tipo específico (por ejemplo, ciencias de la computación) que edificios con profesores disponibles de ese tipo, el algoritmo ajusta el balance moviendo primero a los profesores donde más se necesitan.

2.2 Explicación del código

2.3 Demostración formal