Para hacer funcionar el programa debe correrse con npm run dev (carpeta raíz)

1.Capas de la aplicación:

Front-end:

Carpeta pages: se guardan las paginas de navegación del front end, en este caso solo está la pagina index que tiene el componente drag and drop

Backend:

Carpeta server: guarda el servidor node

Carpeta routes: guarda la url con la cual se desea activar un controlador

Carpeta controllers: carpeta de controladores en la cual Tolls.js se activa primero para generar la lista de adyacencia y junctions.js genera las rutas posibles del problema.

Carpeta model: lastimosamente lee los datos harcodeados de disco ya que la entrada de archivos del componente drag and drop no logra enviar al servidor el archivo .txt guardado en el drag and drop.

Archivo entrada.txt en la carpeta raíz: guarda todos los posibles casos como los define el problema(funciona si se pone mas de un caso)

La solución del problema solo se logra mostrar en el servidor, no en el front al usuario

2. responsabilidades de cada clase:

Front:

Accept(ruta:’./components/dreagNdrop/dragNdrop.js’): componente para capturar archivos

Pages: (ruta:’.pages/index.js’): pagina principal para el usuario localhost:3000/index.js

(solo muestra la respuesta en el servidor node)

backend

server: (ruta:’./server/index.js’ ): carpeta que genera el servidor node, también al cual se conecta con el front end, se le dan los correspondientes endpoints para que funcione.

Routes:tolls.js (ruta :’./server/routes/tolls. ’):indica el controlador que va a utilizer el endpoint creado para un proceso

Controller: tolls.js (‘ruta:’server/controller/tolls.js’):

createTolls: se comporta como función main para empezar a crear las listas de adyacencia

makeTolls: recorre todos los datos capturados para empezar a guardarlos en un objeto con una llave dependiendo de donde sale y genera los valores de los peajes de y hacia x

createKey: genera una llave para el diccionario según el valor x de la ruta que se esta recorriendo

Controller: junctions.js(ruta :’./server/ controller / junctions.js ’)

Genera las rutas que se pueden recorrer hasta el numero maximo de intersecciones que puede haber, ej:

2->1->3

Model: readTolls.js(ruta :’./server/ model / readTolls.js ’)

Originalmente se tenia pensado que el archivo .txt capturado del drag and drop del front end fuera leído en la carpeta model para poderlo manipular de lado del servidor.

Lo que hace actualmente en el código es leer el archivo harcodeado en la carpeta raíz llamado entrada.txt para hacer el procedimiento que se pide en el ejercicio

3. pruebas unitarias: No

Preguntas:

Principio de responsabilidad única?

Este principio consiste en que cada clase debe tener una sola responsabilidad en la funcionalidad del software que se desarrolla , y esta responsabilidad debe estar encapsulada en una clase.

Se deben separar las responsabilidades para el desarrollo del problema

Que es código limpio?

El código limpio revela de forma clara la intención de su funcionalidad, debe dar una idea de lo que hace antes de explorarlo para entenderlo, desde su nombre, hasta las variables que recibe etcétera.

De esta manera el código, sin importar el lenguaje expresa claramente sus propósitos.