Informe Laboratorio Conjuntos Disyuntos y Tabla Hash

(Montones Binarios grafos)

ELABORADO POR: JUAN CAMILO ROJAS CATRO Y FRANCISCO JAVIER ROJAS

PRESENTADO A: SESBASTIAN CAMILO MARTINEZ REYES

ESCUELA COLOMBIANA DE INGENIERÍA JULIO GARAVITO

AYED

PROGRAMA DE INGENIERÍA DE SISTEMAS

BOGOTÁ D. C.

5 DE NOVIEMBRE DE 2020

**Ruta Fácil**

**1. Contexto:**

A veces las tiendas tienen que hacer varios domicilios y esto puede llegar a ser problemático pues se pueden complicar haciendo las rutas, se confunden o no saben que ruta tomar, omitiendo clientes, perdiendo tiempo en la entrega gastando combustible y de cierta forma empeorando el servicio, para resolver este problema simplemente utilizaran el programa que tiene como objetivo crear la ruta más corta siempre y cuando los lugares de entregan tengan vías de acceso, también tiene una funcionalidad secundaria donde una persona dando las conexiones existentes en una mapa obtiene el mejor lugar para colocar la centrar de abastos, todo esto ingresando su ubicación (tienda) y las ubicaciones de los clientes para la primera función y solo ingresando las ubicaciones para la segunda función.

Ahora se incluye en el proyecto una nueva función para el administrador de un negocio la cual consiste en organizar a sus trabajadores con mejor o peor desempeño, esto con fines flexibles, el administrador puede dar advertencia o sacar a los peores, puede priorizar las rutas importantes a los mejores o dar bonos, en pocas palabras el administrador puede mejorar su negocio a partir del sistema de puntos- rendimiento, los puntos son calculados por su parte aconsejablemente en relación con su puntuación por parte de los clientes, la cantidad de pedidos realizados diariamente y la rapidez de los pedidos.

Con intención mejorar la organización de los repartidores añadimos una nueva función en la cual el gerente de un local puede asignar ciertos pedidos representados por conjuntos a un repartidor, si se le asignan otros pedidos a un repartidor puede juntar los pedidos, y puede saber si ya asigno un pedido por su nombre.

Además para organizar mejor aquellos pedidos que se realizan con anticipación creamos

**2. Algoritmo:**

* **Estrategia:**

Grafos:

A medida que se leen los datos de la entrada se guardan en una matriz de adyacencia, luego por el recorrido BFS se encuentra la ruta más corta desde la tienda hasta pasar por todos los clientes.

Montones:

Se van leyendo los datos de los trabajadores, por fecha nombre y puntuación, estos datos se almacenan como colas de prioridad tanto maximal como minimal.

Conjuntos Disyuntos:

Se leen cada conjunto y se guardan, dependiendo de cada acción escogida se pueden unir los conjuntos, buscarlos y añadir nuevos.

HashTable:

Se leen todos los datos llave valor y luego con el tamaño indicado se crea la HashTable luego se hace la operación escogida.

* **Estructuras de datos:**

Se utilizarán diccionarios, listas unidimensionales y listas bidimensionales.

* **Especificaciones:**

Grafos:

El grafo es mixto, se toma como punto de partida la tienda, no existe prioridad entre los clientes y para simplificar el problema 0 es la tienda y los n clientes son números de 1 hasta n.

Montos:

Se tiene que ingresar la fecha el nombre y la puntuación como se especificada (AAAA-MM-DD Nombre Puntuación)

Conjuntos Disyuntos:

Para crear un conjunto que represente mas de un pedido se debe ingresar como “nombre1,nombre2”, el resto de acciones son intuitivas y se especifican.

HashTable:

Cada pedido (llave-valor) se debe hacer en una línea de la entrada como se especifica.

* **Entrada:**

Grafos:

La entrada consiste en primero un numero único por línea que decide la funcionalidad (encontrar la ruta o encontrar de domicilios o encontrar la ruta fácil hasta la central), luego líneas consecutivas donde cada línea tendrá dos números, los cuales pueden ser clientes y/o la tienda, cada línea representa una conexión por vías entre estos. Por último, dependiendo de lo funcionalidad escogida le preguntará por

Montos:

Una vez se llega a la opción de insertar domiciliario, las siguientes líneas existentes deben ser la fecha, el nombre y la puntuación de los domiciliarios que se deseen ingresar.

Conjuntos Disyuntos:

Primero se inicia el programa con la entrada especificada, luego se ingresa en una sola línea todos los juntos del conjunto disyunto, si un conjunto tiene mas de 1 elemento se especifica como se debe ingresar.

HashTable:

La entrada consiste en primero el n tamaño de la HashTable, luego las siguientes n líneas son datos llave valor que se van ingresar en la HasTable.

* **Salida:**

Grafos:

La salida dependerá de la funcionalidad escogida, eso debe tenerlo en cuenta el usuario, sin embargo, en ambos casos la salida es la ruta más corta.

Montos:

La salida debe ser la cola de prioridad escogida, ya sea maximal o minimal.

Conjuntos:

Conjuntos Disyuntos:

La salida es el resultado del conjunto disyunto, después de su creación o en su defecto después de realizar cualquiera de las especificaciones ofrecidas.

HashTable:

Depende de la opción escogida, nos devuelve la hora de la entrega especificada, o todas las entregas posibles.

**3. Casos de prueba:**

* **Entrada:**

Grafos:

1

4 5

0 2

2 1

2 3

1 2

3 0

2

4 5

0 2

2 1

2 3

1 2

3 0

Montones:

2

2

1

2020-11-12 Carlos 0

2020-05-12 Santiago 43

2020-11-12 Tatiana 50

2018-11-12 Federico 100

2019-11-12 Jhon 130

Conjuntos Disyuntos:

0

anuel,barbara camilo andres,juan fran,fred esteban

1

pepe

2

camilo esteban

3

0

HashTable:

0

6

Juan 20/11/20-20:00

Pipe 21/11/20-13:00

Camilo 21/11/20-15:30

Fran 22/11/20-11:00

Paco 22/11/20-10:46

Diego 24/11/20-16:10

1

2

0

* **Salida:**

Grafos:

[0, 1, 2, 1, 2] ['tienda', 'Cliente2', 'Cliente1', 'Cliente3', 'Cliente2', 'Cliente1', 'Cliente3']

las centrales de abastos pueden ser [0]

Montones:

Next element ... ('2020-11-12', 'Carlos', 0)

Next element ... ('2020-11-06', 'Ernesto', 10)

Next element ... ('2020-11-06', 'Andres', 24)

Next element ... ('2020-05-12', 'Santiago', 43)

Next element ... ('2020-11-12', 'Tatiana', 50)

Next element ... ('2020-11-06', 'Vanessa', 65)

Next element ... ('2020-11-06', 'Juan', 87)

Next element ... ('2020-11-06', 'Melissa', 90)

Next element ... ('2018-11-12', 'Federico', 100)

Next element ... ('2020-11-06', 'Javier', 120)

Next element ... ('2019-11-12', 'Jhon', 130)

Conjuntos Disyuntos:

Sus pedidos son: [{'anuel,barbara'}, {'camilo'}, {'andres,juan'}, {'fran,fred'}, {'esteban'}]

Sus pedidos son: [{'anuel,barbara'}, {'andres,juan'}, {'fran,fred'}, {'pepe'}, {'camilo', 'esteban'}]

HashTable:

20/11/20-20:00

0 : []

1 : [('Juan', '20/11/20-20:00'), ('Diego', '24/11/20-16:10')]

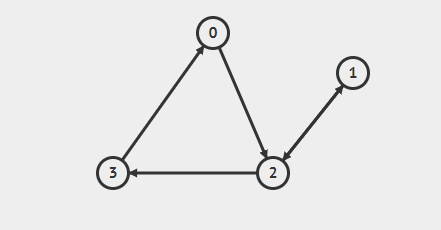
2 : []

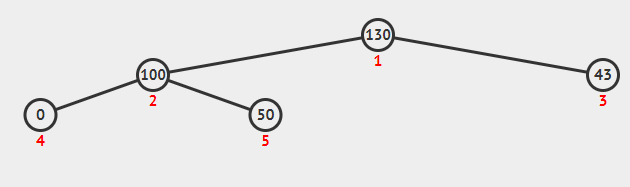
3 : []

4 : [('Camilo', '21/11/20-15:30')]

5 : [('Pipe', '21/11/20-13:00'), ('Fran', '22/11/20-11:00'), ('Paco', '22/11/20-10:46')]

* **Grafo:**





**¡Para una mejor compresión sobre la entrada y la salida si se abre el programa adjuntado Disjointsets.py se explica mediante prints cómo funciona el programa para conjuntos disyuntos y el archivo prueba.txt para HashTable!**

**4. Análisis:**

* **Temporal:**

Tanto el recorrido BFS como la función principal tiene una complejidad de O(n^2)