Laura Natalia Rojas
Cristian Álvarez
Ana María Durán

DOCUMENTO TÉCNICO HASH V DISYUNTOS

https://github.com

Especificación:

Se solita implementar las diferentes versiones de tabla hash, donde no se utilice una estructura de datos no indexada como colección en profundidad, es posible implementar bien sea una linked list, un árbol BST o una cola de prioridad. Por otra parte, es solicitado extender el prototipo funcional presentado en el laboratorio anterior pero esta vez añadiendo un reporte/característica usando Tablas de hash y la implementación vista de conjuntos disyuntos.

Adjuntar casos de prueba en el informe.

Entrada:

Pedido

Salida:

Resumen del pedido

Diseño:

Estrategia

Quisimos plantear un escenario en el cual se conectarán muchas relaciones de forma no dirigida, con el objetivo de ver maneras óptimas de llevar pedidos de un lugar a otro, por ende, escogimos una empresa de mensajería, escogimos dos ciudades: Bogotá y Medellín, ahora se va a plantear una implementación con hash.

Algoritmo y Documentación

```python
from sys import stdin
from random import randint

class HashTable:
    def __init__(self, size):
        self.elements = [[] for x in range(size)]


    def getElements(self):
        return self.elements

    def printElements(self):
        for index in range(len(self.elements)):
            print(index, ': ', self.elements[index])

    def hash(self, key):
        return hash(key) % len(self.elements)

    def assign(self, index, element):
        self.elements[index].append(element)

    def insert(self, key, value):
        index = self.hash(key)
        print('Inserting', value, 'with key', key, 'on index', index)
        self.assign(index, (key,value))
```

```python
def search(self, key):
    index = self.hash(key)
    for e in self.elements[index]:
        if e[0] == key:
            return e[1]
    return None

def update(self, key, value2):
    index = self.hash(key)
    for e in self.elements[index]:
        if e[0] == key:
            e[1] = value2

def delete(self, key):
    index = self.hash(key)
    element = self.search(key)
    if element:
        self.elements[index].remove(element)
```

Casos de Prueba

Entrada	Justificación	Salida
0 1 Lara 14/02/2023 0	Probar formato de archivo de pedido	[('Lara', '14/02/2023')]
0 1 Lara 14/02/2023 Hora Lara 0	Probar búsqueda de pedido por hora a partir del nombre	[('Lara', '14/02/2023')] 14/02/2023
1 Lara 14/02/2023 Hora Lara resumen 0	Probar resumen de los pedidos	[('Lara', '14/02/2023')] 14/02/2023 0 : [('Lara', '14/02/2023')]

Fuentes

HashLab13.py

Laura Natalia Rojas
Cristian Álvarez
Ana María Durán

CONJUNTOS DISYUNTOS

Entrada:

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

Salida:

Resumen del pedido

Diseño:

Estrategia

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

Algoritmo y Documentación

Laura Natalia Rojas
Cristian Álvarez
Ana María Durán


```python
from sys import stdin


class DisjointSets:

    def __init__(self, A):
        self.sets = [set([x]) for x in A]

    def getSets(self):
        return self.sets

    def findSet(self, x):
        for si in self.sets:
            if x in si:
                return si
        return None

    def makeSet(self, x):
        if self.findSet(x) is None:
            self.sets.append(set([x]))
            return self.sets[len(self.sets)-1]
        return self.findSet(x)

    def union(self, x, y):
        s1 = self.findSet(x)
        s2 = self.findSet(y)

        if s1 is None:
            s1 = self.makeSet(x)
        if s2 is None:
            s2 = self.makeSet(y)
        if s1 != s2:
            s3 = s1.union(s2)
            self.sets.remove(s1)
            self.sets.remove(s2)
            self.sets.append(s3)
```

```python
    def connectedComponents(self, Arcs):
        for e in Arcs:
            self.union(e[0], e[1])
            print('After processing arc', e, self.sets)
        result = []
        for si in self.sets:
            if len(si) > 1:
                result.append(si)
        return result
```

Casos de Prueba

Entrada	Justificación	Salida
0 anuel,barbara camilo andres,juan fran,fred esteban 1	Probar pedidos cuando se ingresan 1 y 2 ordenes de clientes	Sus pedidos son: [{'anuel,barbara'}, {'camilo'}, {'andres,juan'}, {'fran,fred'}, {'esteban'}]

pepe 2 camilo esteban 3 0		
0 anuel,barbara camilo andres,juan fran,fred esteban 1 pepe 2 0	Probar pedidos agregando una orden	Sus pedidos son: [{'anuel,barbara'}, {'andres,juan'}, {'fran,fred'}, {'pepe'}, {'camilo', 'esteban'}]

Fuentes

DisjointSets13.py