Manual Técnico Proyecto 3 - Estructura De Datos

Graficar

Compatibilidad

-Windows 7,8,10

-Linux

Especificaciones

En esta sección se mencionará lo que debes conocer de la aplicación.

Lenguaje Utilizado

El lenguaje utilizado para la realización de este proyecto fue Java.

IDE y JDK utilizado

El IDE utilizado fue NetBeans en su versión 8.2, y el JDK utilizado fue 1.8.0

Estructuras Utilizadas

Lista Enlazada - Clientes

¿Está Vacía?

```
public boolean isEmpty(){
    return first==null;
}
```

Insertar

```
public void finalInsert(Client client){
    ClientListNode node= new ClientListNode(client);
    if(first==null){
    first=node;
    }else{
        ClientListNode pointer= first;
        while(pointer.next!=null){
            pointer=pointer.next;
            }
        pointer.next=node;
    }
    countClients++;
}
```

Tabla Hash - Mensajeros

Inicializar la tabla

```
public void initialize(){
    for(int i=0;i<size;i++){
        HashTableNode node= new HashTableNode(i,null);
        if(first==null){
            first=node;
        }else{
            HashTableNode pointer= first;
            while(pointer.next!=null){
                pointer=pointer.next;
            }
            pointer.next=node;
        }
    }
}
```

Calcular llave

```
public int calculateKey(BigInteger dpi){
    int key=(int) (dpi.longValue()% this.size);
    return key;
}
```

Encontrar siguiente número primo

```
public int findNextPrimeNumber(){
    int counter=this.size+1;
    while(! isPrimeNumber(counter)){
        counter++;
    }
    return counter;
}
```

Rehash

```
public void rehash(){
    int newSize=findNextPrimeNumber();
    int iterations=newSize-size;
    for(int i=0;i<iterations;i++){
        HashTableNode node= new HashTableNode(size+(i),null);
        System.out.println(node.key+"JJJJJJJJJJJ");
        HashTableNode pointer= first;
        while(pointer.next!=null){
            pointer=pointer.next;
        }
        pointer.next=node;
    }
    size=newSize;
    max=(int) Math.round(size*0.75);
    System.out.println("Nuevo tamaño---"+size);
    System.out.println("Nuevo max---"+max);
}
```

Verificar Libertad

```
public boolean isFree(int key){
    HashTableNode temp = first;
    do{
        if(temp.key==key){
            return temp.deliveryCourier == null;
        }
        temp = temp.next;
    }while(temp != null);
    //System.out.println("No se encontró");
    return false;
}
```

Solucionar Colisión

```java
public boolean solveCollision(BigInteger dpi, int collision){
    int key=((int) ((dpi.longValue()% 7)+1))*collision;
    System.out.println("Nueva llave "+key);
    collisionNewKey=key;
    return isFree(key);
}
```

Insertar

```java
public void insert(DeliveryCourier deliveryCourier){
    int key=calculateKey(deliveryCourier.getDpi());
    System.out.println("La llave a ingresar es: "+key);
    if(isFree(key)){
        HashTableNode temp = first;
        do{
            if(temp.key==key){
                temp.deliveryCourier=deliveryCourier;
                System.out.println("Asignamos a: "+temp.deliveryCourier.getName()+ "En llave "+key);
                //break;
            }
            temp = temp.next;
        }while(temp != null);
        filledKeys++;
        if(filledKeys==max){
            System.out.println("toca rehash");
            rehash();
        }
    }else{
        while(! solveCollision(deliveryCourier.getDpi(), collisions)){
            collisions++;
            if(collisionNewKey>size){
                firstFree(deliveryCourier);
                System.out.println("Asignamos a: "+deliveryCourier.getName());
                break;
            }
        }
```

```java
            collisions=1;
            HashTableNode temp = first;
            System.out.println("hhhhhhhh"+collisionNewKey);
            do{
                if(temp.key==collisionNewKey){
                    System.out.println("holaaaa");

                    temp.deliveryCourier=deliveryCourier;
                    System.out.println("Asignamos a: "+temp.deliveryCourier.getName());
                }
                temp = temp.next;
            }while(temp != null);
            filledKeys++;
            if(filledKeys==max){
                rehash();
            }

        }
    }
```

Generar Gráfico

```
public String generateGraph() throws IOException{
    String route="Reportes Texto/Tabla Hash.txt";
    String graph="Reportes Img/Tabla Hash.png";
    String tParam = "-Tpng";
    String tOParam = "-o";
    String pathString = "C:\\Program Files\\Graphviz\\bin\\dot.exe";
    String finalText="digraph HashTable{\n node[shape = record,fillcolor=\"bisque1\" color=\"black\" style=\"filled\" label=\
    HashTableNode temp = first;
    while(temp != null){
        if(temp.deliveryCourier!=null){
            finalText+="|"+temp.key+"| DPI: "+temp.deliveryCourier.getDpi()+"\\n"+temp.deliveryCourier.getName()+" "+temp.del
        }else{
            finalText+="|"+temp.key+"|"+temp.deliveryCourier+"|";
        }
        temp = temp.next;
    }
    finalText+="\"]Erwin;\n}";
    FileWriter fw = new FileWriter(route);
    BufferedWriter bw = new BufferedWriter(fw);
    bw.write(finalText);
    bw.close();
    String[] cmd = new String[5];
    cmd[0] = pathString;
    cmd[1] = tParam;
    cmd[2] = route;
    cmd[3] = tOParam;
    cmd[4] = graph;
    Runtime rt = Runtime.getRuntime();
    rt.exec( cmd );
    //System.out.println(finalText);
    return finalText;
}
```

Lista de adyacencia - Rutas

Graficar

```
public String generateAdjancecyList() throws IOException{
    String route="Reportes Texto/listaAdyacente.txt";
    String graph="Reportes Img/listaAdyacente.png";
    String tParam = "-Tpng";
    String tOParam = "-o";
    String pathString = "C:\\Program Files\\Graphviz\\bin\\dot.exe";

    String finalText="digraph G{\nnode [shape=box];\n";
    CitiesListNode temp = first;
    String rankSame="{rank=same; ";
    String conections="";
    String nodes="";
    while(temp != null){
        nodes+="N"+temp.hashCode()+"[label=\""+"Ciudad No. \n"+temp.city.getId()+"\"];\n";
        if(!temp.city.getRoutesList().isEmpty()){
            nodes+=temp.city.getRoutesList().collectLinkedList();
            conections+="N"+temp.hashCode()+ " -> ";
            conections+=temp.city.getRoutesList().collectConnections();
            //conections+="start"+ " -> "+"N"+temp.hashCode()+";\n";
        }
        if (temp.next!=null){
            conections+="N"+temp.hashCode()+ " -> "+"N"+temp.next.hashCode()+";\n";
            conections+="N"+temp.next.hashCode()+ " -> "+"N"+temp.hashCode()+";\n";
            rankSame+="N"+temp.hashCode()+",";
        }else{
            rankSame+="N"+temp.hashCode();
        }

        temp = temp.next;
```

```
    }
    rankSame+="};";


    finalText+=nodes+"\n";
    finalText+=conections+"\n";
    finalText+="start [shape=Mdiamond label=\"Lista De Adyacencia:\n "+""+"\"];";
    finalText+=rankSame;
    finalText+="}\n}";
    FileWriter fw = new FileWriter(route);
    BufferedWriter bw = new BufferedWriter(fw);
    bw.write(finalText);
    bw.close();

    String[] cmd = new String[5];
    cmd[0] = pathString;
    cmd[1] = tParam;
    cmd[2] = route;
    cmd[3] = tOParam;
    cmd[4] = graph;

    Runtime rt = Runtime.getRuntime();

    rt.exec( cmd );

    return finalText;
}
```

Grafo No Dirigido - Rutas

Graficar

```
public String generateUndirectedGraph() throws IOException{
    String route="Reportes Texto/Grafo.txt";
    String graph="Reportes Img/Grafo.png";
    String tParam = "-Tpng";
    String tOParam = "-o";
    String pathString = "C:\\Program Files\\Graphviz\\bin\\dot.exe";

    String finalText="digraph G{\nnode [shape=box];\n";
    CitiesListNode temp = first;
    String rankSame="{rank=same; ";
    String conections="";
    String nodes="";

    while(temp != null ){
        nodes+="N"+temp.city.getId()+"[label=\""+"Ciudad No. "+temp.city.getId()+"\"];\n";
        temp=temp.next;
    }
    temp = first;
    while(temp != null ){
        if(!temp.city.getRoutesList().isEmpty()){
            //nodes+=temp.city.getRoutesList().collectLinkedList();
            conections+=temp.city.getRoutesList().collectConnections2("N"+temp.city.getId());
            //conections+="start"+ " -> "+"N"+temp.hashCode()+";\n";
        }


        temp = temp.next;
```

```
        while(temp != null ){
            if(!temp.city.getRoutesList().isEmpty()){
                //nodes+=temp.city.getRoutesList().collectLinkedList();
                conections+=temp.city.getRoutesList().collectConnections2("N"+temp.city.getId());
                //conections+="start"+ " -> "+"N"+temp.hashCode()+";\n";
            }


            temp = temp.next;

        }
        rankSame+="};";


        finalText+=nodes+"\n";
        finalText+=conections+"\n";
        finalText+="start [shape=Mdiamond label=\"Grafo De Rutas:\n "+""+"\"];";
        finalText+=rankSame;
        finalText+="}\n}";
        FileWriter fw = new FileWriter(route);
        BufferedWriter bw = new BufferedWriter(fw);
        bw.write(finalText);
        bw.close();

        String[] cmd = new String[5];
        cmd[0] = pathString;
        cmd[1] = tParam;
        cmd[2] = route;
        cmd[3] = tOParam;
        cmd[4] = graph;
```