Manual คู่มือการใช้งาน

1 ขั้นตอนแรกให้สร้างไฟล์สำหรับ main ขึ้นมาเพื่อทำการเรียกใช้งาน class Kruskals_Graph (หรือใช้ไฟล์ main ที่มีให้ในไฟล์ rar)

```
1   public class runGraph {
        Run | Debug
2       public static void main(String[] args) {
3
4       }
5   }
```

2 วิธีการเรียกใช้งานตัว class Kruskals_Graph ให้สร้างObject จาก Kruskals_Graph

```
1   public class Main {
2
        Run | Debug
3       public static void main(String[] args) {
4
5           Kruskals_Graph graph = new Kruskals_Graph();
6
7       }
8
9   }
10
```

ไม่จำเป็นต้องเขียนรับค่าหรือแสดงผล เนื่องจากใน Kruskals_Graph มี คำสั่งแสดงผล และรับค่าให้อยู่แล้ว

วิธีการทำงานของKruskals_Graph.java

class Kruskals_Graph ประกอบไปด้วย attribute constructorและ method ดังนี้

Attribute

```
private HashMap<String,ParentNode> node = new HashMap<>();
private ArrayList<BuildGraph> path = new ArrayList<>();
private ParentNode A = new ParentNode();
private ParentNode B = new ParentNode();
```

```
7  public class Kruskals_Graph extends BuildGraph {
8
9      private HashMap<String,ParentNode> node = new HashMap<>();
10     private ArrayList<BuildGraph> path = new ArrayList<>();
11     private ParentNode A = new ParentNode();
12     private ParentNode B = new ParentNode();
```

- node ใช้สำหรับเก็บ ค่า HashMap เป็น ชื่อ node ตามด้วย ParentNode
- path ใช้สำหรับเก็บ ข้อมูลของเส้นและ node เช่น node ที่ต่อกันด้วย weight
-A ประกาศค่าสำหรับการใช้งานเรียกข้อมูลใน class ParentNode
-B ประกาศค่าสำหรับการใช้งานเรียกข้อมูลใน class ParentNode

Constructor

```
14     public Kruskals_Graph(){
15
16         Scanner sc = new Scanner(System.in);
17         int numLine = sc.nextInt();
18         for(int i = 0; i < numLine; i++){
19
20             String firstName = sc.next();
21             String secondName = sc.next();
22             int weight = sc.nextInt();
23
24             node.put(firstName, new ParentNode(firstName, rank: 0));
25             node.put(secondName, new ParentNode(secondName, rank: 0));
26
27             path.add(new BuildGraph(weight, firstName, secondName));
28             // set path
29
30         }
31
32         sc.close();
33
34         sortByCompare(); // sort by weight
35         setGraphParent(); // check circuit
36         graphToGrid();
37
38     }
```

รับค่าเข้ามาเป็น numLine คือจำนวนเส้น จากนั้นวนรอบการทำงานตามจำนวนเส้น ในแต่ละรอบจะทำการรับค่าชื่อ node ที่เชื่อมต่อเส้นเข้าด้วยกัน และรับค่า weigth นำข้อมูลมาใส่เข้าไปในpath ให้เป็นการเก็บค่าด้วย object BuildGraph และ เก็บค่าใน node โดยให้ key เป็น node ทั้งสองตัว ส่วน value จะเก็บข้อมูลของ Parent Node

Method

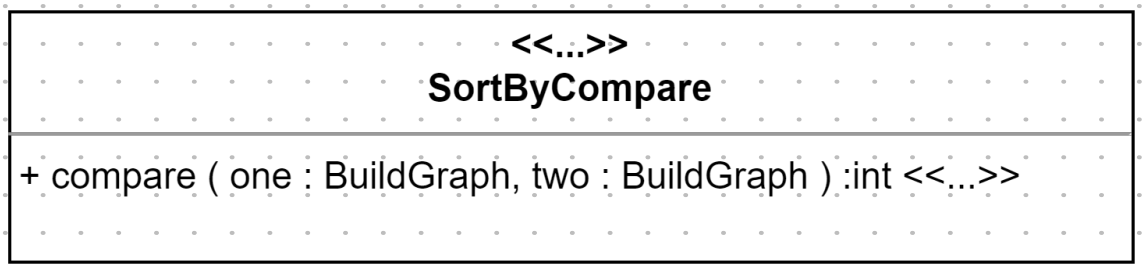
- public void setGraphParent()
  Method setGraphParent ใช้สำหรับตรวจสอบสถานะของเส้นที่ว่าถ้าหากมีการเชื่อมต่อของ node ตาม path นั้นๆ จะเกิด simple circuit หรือไม่

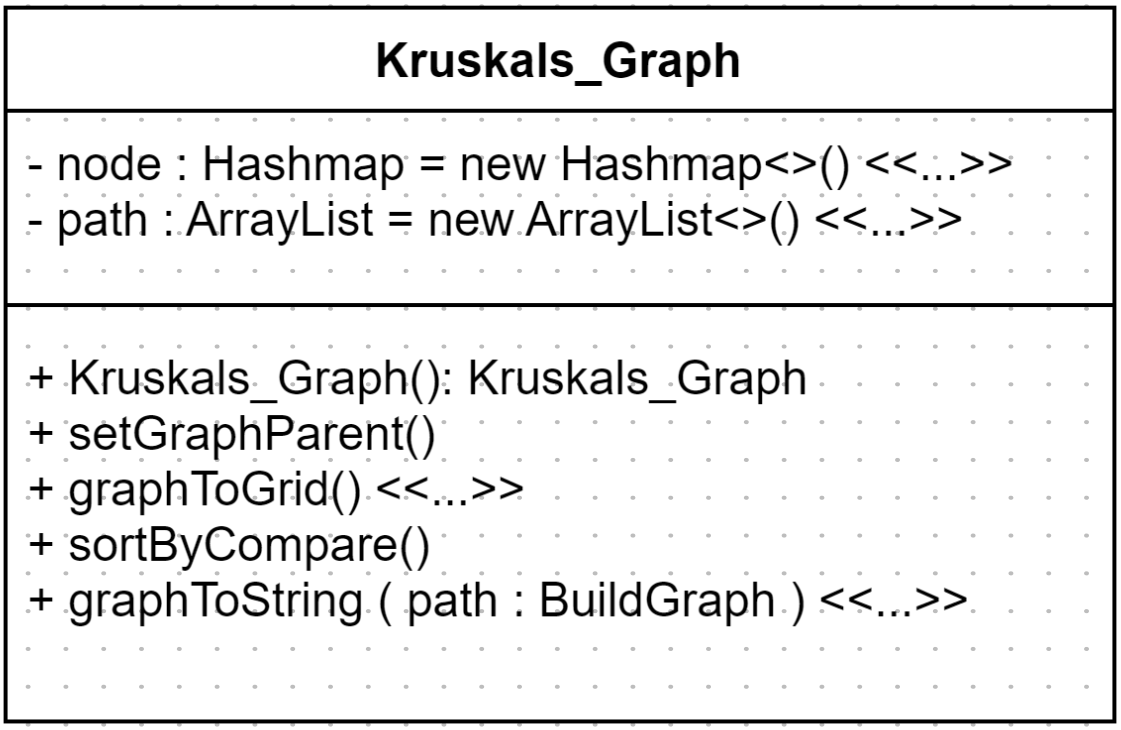
- public void graphToGrid()
  Method graphToGrid ใช้สำหรับแสดงผลสถานะของแต่ละเส้นว่ามีการ Add หรือไม่โดยแสดงผลเป็นตาราง

- public void graphToString(BuildGraph path)
  Method graphToString ใช้สำหรับแสดงผลในรูปของ weigth node1 node2 (node ที่ เชื่อมต่อกัน)

- public void sortByCompare()
  Method sortByCompare ใช้สำหรับการเรียงข้อมูลจากมากไปน้อยด้วยข้อมูลของ weigth ผ่าน class SortByCompare ที่ implements Comparator<BuildGraph> โดยใช้คำสั่ง Collections.sort

```
107     public void sortByCompare(){
108         Collections.sort(path, new SortByCompare());
109     }
110 
111 }
112 
113 class SortByCompare implements Comparator<BuildGraph> {
114 
115     @Override
116     public int compare(BuildGraph one, BuildGraph two) {
117         return one.getWeight() - two.getWeight();
118     }
```





วิธีการทำงานของBuildGraph.java

class BuildGraph ประกอบไปด้วย attribute constructorและ method ดังนี้

Attribute

private int weight;

private String start, end, status;

- weigth ใช้สำหรับบอก ค่าน้ำหนักของเส้น node
- start ใช้สำหรับบอก ชื่อ node แรกที่เชื่อมต่อกับ node ที่สองด้วยน้ำหนัก = weigth
- end ใช้สำหรับบอก ชื่อ node ที่เชื่อมต่อกับ node ที่แรกด้วยน้ำหนัก = weigth
- status ใช้สำหรับบอกสถานะว่า path นี้ได้มีการเพิ่มเข้าไปใน เส้นทางหรือไม่

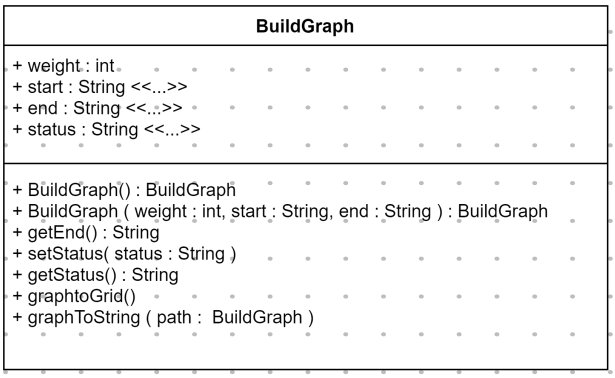
```
1   public class BuildGraph {
2
3       private int weight;
4       private String start, end, status;
```

Constructor

```
6       public BuildGraph() {
7       }
8
9       public BuildGraph(int weight, String start, String end) {
10
11          this.weight = weight;
12          this.start = start;
13          this.end = end;
14
15      }
```

ใช้กำหนดค่าที่รับเข้ามา เป็น weigth start และ end

Method



- public int getWeight()
  Method getWeight ใช้สำหรับเข้าถึงค่าของ weigth ใน path

- public String getStart()
  Method getStart ใช้สำหรับเข้าถึงค่าของ node แรก ใน path

- public String getEnd()
  Method getEnd ใช้สำหรับเข้าถึงค่าของ node สอง ที่เชื่อมต่อกับ nodeแรก

- public void setStatus(String status)
  Mathod setStatus ใช้สำหรับเรียกเพื่อตั้งค่าให้ กับ statusของ path

- public String getStatus()
  Method getStatus ใช้สำหรับเข้าถึง status ของ path

- public void graphToGrid()
  Method graphToGrid ใช้สำหรับแสดงผลสถานะของแต่ละเส้นว่ามีการ Add หรือไม่ โดยแสดงผลเป็นตาราง

- public void graphToString(BuildGraph path)
  Method graphToString ใช้สำหรับแสดงผลในรูปของ weigth node1 node2 (node ที่ เชื่อมต่อกัน)

วิธีการทำงานของParentNode.java

class ParentNode ประกอบไปด้วย attribute constructorและ method ดังนี้

Attribute

```java
public class ParentNode {

    private String parent = "";
    private String node;
    private int rank = 0;
    private ArrayList<String> child = new ArrayList<>();
```

private String parent = "";

private String node;

private int rank = 0;

private ArrayList<String> child = new ArrayList<>();

- path ใช้สำหรับบอกชื่อของ node ตั้งต้น
- node ใช้สำหรับบอกชื่อ node นั้นๆ
- rank ใช้สำหรับนับจำนวน path ที่มีการเชื่อมต่อ
- child ใช้สำหรับเก็บข้อมูลชื่อของ node ทั้งหมดที่เชื่อมต่อยู่กับ node นั้น

Constructor

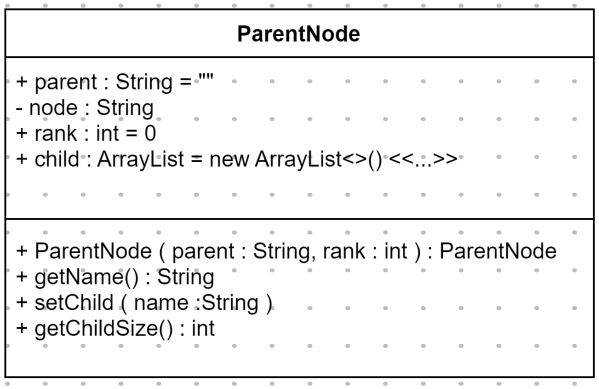
```java
    public ParentNode() {}

    public ParentNode(String parent, int rank) {

        this.parent = parent;
        this.rank = rank;
        this.node = parent;

    }
```

ใช้กำหนดค่าที่รับเข้ามา เป็น parent rank และ node



Method

- public String getName()
  Method getName ใช้สำหรับเข้าถึงชื่อของ node นั้นๆ

- public int getRank()
  Method getRank ใช้สำหรับเข้าถึงค่าของ rank ใน parent นั้น

- public void setRank(int rank)
  Method setRank ใช้สำหรับรับค่า rank เข้ามา update ใน parent

- publicvoid setParent(String parent)
  Mathod setParent ใช้สำหรับรับชื่อ parent ตัวใหม่เข้ามาแทนที่ parent ที่เคยอยู่

- public String getParent()
  Method getParent ใช้สำหรับเข้าถึง parent ปัจจุบันของ node นั้น

- public void setChild(String name)
  Method setChild ใช้สำหรับรับค่าชื่อของ node ที่เชื่ออยู่กับ node นั้น add เข้ามาเก็บใน Arraylist child

- public ArrayList<String> getChild()
  Method getChild ใช้สำหรับเข้าถึง arraylist ที่เก็บข้อมูล node ที่เชื่อต่อกับ node ปัจจุบัน

- public int getChildSize()
  Method getChildSize ใช้สำหรับเข้าถึง จำนวน node ที่เชื่ออยู่กับ node ปัจจุบัน

หลักการทำงานของ class Kruskals_Graph

```java
public class Kruskals_Graph extends BuildGraph {

    private HashMap<String,ParentNode> node = new HashMap<>();
    private ArrayList<BuildGraph> path = new ArrayList<>();
    private ParentNode A = new ParentNode();
    private ParentNode B = new ParentNode();

    public Kruskals_Graph(){

        Scanner sc = new Scanner(System.in);
        int numLine = sc.nextInt();
        for(int i = 0; i < numLine; i++){

            String firstName = sc.next();
            String secondName = sc.next();
            int weight = sc.nextInt();

            node.put(firstName, new ParentNode(firstName, rank: 0));
            node.put(secondName, new ParentNode(secondName, rank: 0));

            path.add(new BuildGraph(weight, firstName, secondName));
            // set path

        }

        sc.close();

        sortByCompare(); // sort by weight
        setGraphParent(); // check circuit
        graphToGrid();

    }
```

ทำการรับค่าเข้ามาเป็น 2 node ที่เชื่อมต่อกัน และเก็บค่าเข้าใน node และ path

```java
    public void sortByCompare(){
        Collections.sort(path, new SortByCompare());
    }

}

class SortByCompare implements Comparator<BuildGraph> {

    @Override
    public int compare(BuildGraph one, BuildGraph two) {
        return one.getWeight() - two.getWeight();
    }
```

หลังจากนั้นทำการเรียก method sortByCompare(); เพื่อเรียงข้อมูลจากน้อยไปมาก โดยเรียงจาก weigth

หลักการทำงานของ method setGraphParent

```
40      public void setGraphParent(){
41
42          for(BuildGraph connect : path) {
43
44              A = node.get(connect.getStart());
45              B = node.get(connect.getEnd());
46
47              if( !( A .getParent().equals( B .getParent())) ){
48
49                  if( node.get( A . getParent() ) .getRank() >=  node.get( B . getParent() ) .getRank()){
50                      B .setParent( A .getParent() );
51                      A .setRank( node.get( A . getParent() ).getRank() + 1 );
52
53                      if( B .getChildSize() == 0) A .setChild( node.get(connect.getEnd()) .getName() );
54                      else{
55                          for(String child : B .getChild()){
56                              A.setChild(child);
57                              node.get(child).setParent(A.getParent());
58                          }
59                      }
60
61                  }else{
62
63                      A .setParent( B .getParent() );
64                      B .setRank( node.get( B . getParent() ).getRank() + 1 );
65
66                      if( A .getChildSize() == 0) B .setChild( node.get(connect.getStart()) .getName() );
67                      else{
68                          for(String child : A .getChild()){
69                              B.setChild(child);
70                              node.get(child).setParent( B.getParent() );
71                          }
72                      }
73
74                  } connect.setStatus(status: "Added");
75
76              }else connect.setStatus(status: "Not Added");
77          }
78
79      }
```

หลังจากทำการเรียงข้อมูลจากน้อยไปมากด้วย weigth จะทำการตรวจสอบสถานะของแต่ละ path ด้วย method setGraphParent เริ่มจากการววนรอบการทำงานด้วยข้อมูล path ทั้งหมดโดยแทนข้อมูล path ในแต่ละรอบด้วย ตัวแปรที่ชื่อว่า connect จากนั้นใช้แนวคิดของเซ็ทว่า ถ้า root ของ graph ที่ node ที่จะทำการเพิ่ม path เป็นคนละ root กัน ซึ่งหมายความว่ากราฟเดิมของทั้งสองจุดที่กำลังจะเพิ่ม path เข้าไปอยู่คนละเซ็ทกันสามารถเพิ่ม path และไม่ทำให้เกิด circuit ได้ แต่ถ้าหาก path ที่กำลังจะถูกเพิ่ม มาจากเซ็ทเดียวกันจะทำให้เกิดวงจรหรือ simple circuit โดยเริ่มจากให้ทุก node ที่มีเริ่มที่มี parent เป็น ตัวเองจากนั้นทำการตรวจสอบจาก path ที่รับเข้ามาว่ามากจากเซ็ทเดียวกันหรือไม่ด้วย parent ถ้า parent คนละตัวกันจะทำการเปลี่ยนค่า parent ของ path รวมไปถึง root ของ path ทั้งหมดให้เป็น parent หรือ เซ็ทเดียวกัน โดยที่จะนับ rank ของเซ็ท rank จะถูกเพิ่มตามจำนวนครั้งที่มีการแอด root เข้าไปใน parent นั้นๆจากนั้นจะทำการตรวจสอบ ในกรณีที่มีpath เชื่อมต่อกันโดยให้ เซ็ทที่ขนาดใหญ่กว่าเป็น parent ของเซ็ทที่มีขนาดเล็กกว่า โดยขนาดของเซ้ทจะนับจาก rank ของ parent นั้นๆ เมื่อทำการตรวจสอบ หากเกิดวงจรจะตั้งค่าสถานะให้เป็น "Not Added" หากไม่จะตั้งค่าสถานะเป็น "Added"

Input / Output

Input

- บรรทัดแรก รับค่าเข้ามาเป็น จำนวนเส้นของ path (int)
- บรรทัด n (ตามค่าที่รับมาจากบรรทัดแรก) ในแต่ละบรรทัดประกอบไปด้วย node 1(String) node 2(String) weigth(int)

Output

แสดงผลเป็น ตารางสถานะของ path "Added" หรือ "Not Added"

ตัวอย่างการแสดงผล

ตัวอย่าง Input

```
11
Minneapolis Chicago 355
Louisville Cincinnati 83
Chicago Milwaukee 74
St.Louis Louisville 242
Louisville Milwaukee 348
Louisville Nashville 151
Chicago Louisville 269
Minneapolis Nashville 695
Louisville Detroit 306
St.Louis Chicago 262
Cincinnati Detroit 230
```

ตัวอย่าง Output

idx	Edge	weight	status
1	Chicago - Milwaukee	74	Added
2	Louisville - Cincinnati	83	Added
3	Louisville - Nashville	151	Added
4	Cincinnati - Detroit	230	Added
5	St.Louis - Louisville	242	Added
6	St.Louis - Chicago	262	Added
7	Chicago - Louisville	269	Not Added
8	Louisville - Detroit	306	Not Added
9	Louisville - Milwaukee	348	Not Added
10	Minneapolis - Chicago	355	Added
11	Minneapolis - Nashville	695	Not Added

ตัวอย่าง Input

```
6
ChiangMai Lampang 110
ChiangMai Kalasin 200
Kalasin Uttraradit 304
Uttraradit Nan 102
Nan Lampang 150
Lampang Uttraradit 48
```

ตัวอย่าง Output

idx	Edge	weight	status
1	Lampang - Uttraradit	48	Added
2	Uttraradit - Nan	102	Added
3	ChiangMai - Lampang	110	Added
4	Nan - Lampang	150	Not Added
5	ChiangMai - Kalasin	200	Added
6	Kalasin - Uttraradit	304	Not Added

ตัวอย่าง Input

```
9
AbuDhabi Sharjah 139
Sharjah Fujarah 103
Fujarah Ajman 49
Kalba Ajman 100
Kalba Dubai 299
Dubai AbuDhabi 349
AbuDhabi Fujarah 495
Kalba Sharjah 108
Dubai Ajman 98
```

ตัวอย่าง Output

idx	Edge	weight	status
1	Fujarah - Ajman	49	Added
2	Dubai - Ajman	98	Added
3	Kalba - Ajman	100	Added
4	Sharjah - Fujarah	103	Added
5	Kalba - Sharjah	108	Not Added
6	AbuDhabi - Sharjah	139	Added
7	Kalba - Dubai	299	Not Added
8	Dubai - AbuDhabi	349	Not Added
9	AbuDhabi - Fujarah	495	Not Added

ตัวอย่าง Input

```
7
Hokkaido Tohoku 48
Tohoku Kanto 100
Kanto Chubu 102
Chubu Kyushu 108
Kyushu Hokkaido 109
Hokkaido Kanto 46
Kyushu Tohoku 72
```

ตัวอย่าง Output

idx	Edge	weight	status
1	Hokkaido - Kanto	46	Added
2	Hokkaido - Tohoku	48	Added
3	Kyushu - Tohoku	72	Added
4	Tohoku - Kanto	100	Not Added
5	Kanto - Chubu	102	Added
6	Chubu - Kyushu	108	Not Added
7	Kyushu - Hokkaido	109	Not Added

สมาชิกกลุ่ม

64050162	พลอยชมพู	ตุลสุข
64050229	วิศว์	ศิริวัฒน์
64050231	วุฒิชัย	ปัดไธสง