65050389 ธรรมสรณ์ ประเสริฐศรี

Assignment 5 -> All-Pair Minimum Bottleneck Paths

```
Enter the number of vertices: 5
```

```java
Scanner scanner = new Scanner(System.in);

        int n;
        while (true) {
            try {
                System.out.print("Enter the number of vertices: ");
                n = Integer.parseInt(scanner.nextLine());
                break;
            } catch (NumberFormatException e) {
                System.out.println("Error: " + e.getMessage());
            }
        }
```

(1) ให้ผู้ใช้กรอกจำนวน vertices ที่ต้องการ ต้องกรอกเป็น Integer ถ้าหากผู้ใช้ไม่ได้กรอกเป็น Integer โปรแกรมจะแจ้งเตือน และให้ผู้ใช้ป้อนใหม่

```
Enter weight of edge(v1, v2): 2
Enter weight of edge(v1, v3): 5
Enter weight of edge(v1, v4): 3
Enter weight of edge(v1, v5): 15
Enter weight of edge(v2, v3): 2
Enter weight of edge(v2, v4): 3
Enter weight of edge(v2, v5): 20
Enter weight of edge(v3, v4): 10
Enter weight of edge(v3, v5):
Enter weight of edge(v4, v5): 20
```

```java
int[][] adjMatrix = new int[n][n];

for (int i = 0; i < n; i++) {
    for (int j = i + 1; j < n; j++) {
        System.out.print("Enter weight of edge(v" + (i + 1) + ", v" + (j + 1) + "): ");
        String w = scanner.nextLine();

        while (true) {
            if (w.isEmpty()) {
                adjMatrix[i][j] = Integer.MAX_VALUE;
                adjMatrix[j][i] = Integer.MAX_VALUE;
                break;
            }

            try {
                int weight = Integer.parseInt(w);
                if (weight > 0) {
                    adjMatrix[i][j] = weight;
                    adjMatrix[j][i] = weight;
                    break;
                }
            } catch (NumberFormatException e) {
                System.out.println(x:"Error: Invalid input.");
            }

            System.out.println(x:"Error: Invalid input.");
            System.out.print("Enter weight of edge(v" + (i + 1) + ", v" + (j + 1) + "): ");
            w = scanner.nextLine();
        }
    }
}
scanner.close();
return adjMatrix;
```

(2) สร้าง adjMatrix ขนาด n x n และให้ผู้ใช้กรอกค่า weight ของแต่ละเส้นเชื่อมในกราฟ โดยจะมีการตรวจสอบความถูกต้องในการป้อนด้วย คือ ผู้ใช้ต้องป้อนเฉพาะตัวเลข > 0 และสามารถป้อนข้อมูลว่างได้ เผื่อมีกรณีที่ไม่มีเส้นเชื่อม ถ้าหากกรอกผิด โปรแกรมจะเตือนและให้ป้อนใหม่ ซึ่งกระบวนการนี้จะใช้ for loop ในการรับค่า

Assignment 5 -> All-Pair Minimum Bottleneck Paths

```java
public static int[][] minBPs(int[][] cost) {
    int n = cost.length;

    for (int k = 0; k < n; k++) {
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                int newCost = Math.max(cost[k][i], cost[k][j]);
                if (newCost < cost[i][j]) {
                    cost[i][j] = newCost;
                }
            }
        }
    }

    return cost;
}
```

(3) สร้าง matrix cost เพื่อเปรียบเทียบค่า weight ระหว่างจุด i และ j กับค่าที่สามารถทำให้เส้นเชื่อมที่ผ่านจุด k มีค่าน้ำหนักน้อยที่สุด หากพบว่าค่าใหม่น้อยกว่าค่าเดิม โปรแกรมจะอัพเดตค่าน้ำหนักใน matrix cost ด้วยค่าใหม่นั้น ๆ และส่ง matrix cost ที่อัพเดตแล้ว ไปใช้งานต่อไป

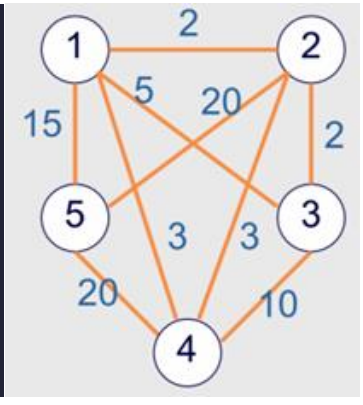
```java
public static void printMatrix(int[][] matrix) {
    for (int[] row : matrix) {
        for (int cell : row) {
            System.out.print(cell == Integer.MAX_VALUE ? "inf" : cell + " ");
        }
        System.out.println();
    }
    System.out.println();
}
```

(4) แสดงผลลัพธ์ matrix โดยถ้าหากค่าใน matrix นั้นเป็น Integer.MAX_VALUE (หมายถึงไม่มีเส้นเชื่อมระหว่างจุด) จะทำการใส่ inf ไว้ในค่านั้นแทน

```java
Run | Debug
public static void main(String[] args) {
    int[][] adjMatrix = inputGraph();

    System.out.println();
    System.out.println(x:"Adjacency Matrix:");
    printMatrix(adjMatrix);

    int[][] cost = minBPs(adjMatrix);
    System.out.println(x:"Minimum Bottleneck Paths:");
    printMatrix(cost);
}
```



(5) Main Program เรียกใช้ method ต่าง ๆ เพื่อรับค่า และหลังจากนั้นจะทำการแสดงผลลัพธ์ในรูปแบบของ Adjacency Matrix และ Minimum Bottleneck Paths

```
Enter the number of vertices: 5
Enter weight of edge(v1, v2): 2
Enter weight of edge(v1, v3): 5
Enter weight of edge(v1, v4): 3
Enter weight of edge(v1, v5): 15
Enter weight of edge(v2, v3): 2
Enter weight of edge(v2, v4): 3
Enter weight of edge(v2, v5): 20
Enter weight of edge(v3, v4): 10
Enter weight of edge(v3, v5):
Enter weight of edge(v4, v5): 20

Adjacency Matrix:
0 2 5 3 15
2 0 2 3 20
5 2 0 10 inf
3 3 10 0 20
15 20 inf20 0

Minimum Bottleneck Paths:
0 2 2 3 15
2 0 2 3 15
2 2 0 3 15
3 3 3 0 15
15 15 15 15 0
```