**การทำงานของโปรแกรม**

โดยจะให้ User เลือกสิ่งที่ต้องการจะทำ ได้แก่ Minimum Spanning Tree โดยใช้ Prim’s Algorithm และ Kruskal’s Algorithm, Shortest Path โดยใช้ Dijkstra’s Algorithm และ Automata Finite State Machine โดยจะกำหนด State รับค่า Input 0 กับ 1 โดย User จะ Input 1 2 3 4 ตามลำดับ

A black text on a white background

Description automatically generated

**การทำงานของ Minimum Spanning Tree Prim’s Algorithm**

จะใช้จุดใดๆเป็นจุดเริ่มต้น เลือกเส้นทางในกราฟเพื่อเชื่อมโดยไม่มีรอยต่อเส้นไหนที่ไม่จำเป็น และมี  
น้ำหนัก(ความยาว)ของเส้นทางที่น้อยที่สุด

จะ Loop While นำเส้นทางที่มี Weight น้อยที่สุดจาก PriorityQueue และตรวจสอบว่า Destination นี้เช็คไปแล้วหรือไม่ ถ้าไม่ จะเช็คแล้วนำไปเก็บใน Minimum Spanning Tree

**การทำงานของ Minimum Spanning Tree Kruskal’s Algorithm**

จะเรียงลำดับเส้นทางในกราฟตาม Weight ของเส้นทางจากน้อยไปมาก และจะสร้าง Union-Find Data Structure สำหรับการจัดการเชื่อมโยงในจุดของกราฟ ทำการ Loop ผ่านเส้นทางทั้งหมดที่ถูกเรียงลำดับไว้จะเลือกเส้นทางที่มีน้ำหนักน้อยที่สุดที่ทำให้ไม่เกิด Cycle ใน Minimum Spanning Tree โดยใช้ Union-Find Data Structure ตรวจสอบการเชื่อระหว่างจุดต้นทางและจุดปลายทาง เมื่อเส้นทางนั้นไม่เกิด Cycle จะเพิ่มเข้าไปใน Minimum Spanning Tree และทำการเชื่อมโยงจุดต้นทางและปลายด้วย Union-Find Data Structure

**การทำงาน Shortest Path Dijkstra’s Algorithm**

จะเป็นการหาผลรวมเส้นทางที่สั้นที่สุดโดยที่เริ่มจากจุดเริ่มต้นที่ผู้ใช้งานกรอกลงไปแล้วเรียงไปตามจุดแต่ละจุดผ่านเส้นที่ให้ผลรวมระยะทางที่สั้นที่สุดของจุดนั้นและไม่เกิดการวนซ้ำจุดเดิมขึ้นภายในกราฟ

**การทำงาน Automata Finite State Machine**

เป็น Finite State Machine ที่รับค่า 0 หรือ 1 โดยที่จะมี 3 State และ เมื่อลงท้ายด้วย 01 จะ Accept เสมอ โดย Input และ State จะเป็นดังตารางด้านล่าง

|  |  |  |
| --- | --- | --- |
| State\Input | 0 | 1 |
| (Start) |  |  |
|  |  |  |
| (Accept) |  |  |

โดยจะอ้างอิงจากรูปด้านล่าง

A diagram of a diagram

Description automatically generated