



รายงาน

เรื่อง โปรแกรม GRAPH STRUCTURE BY ADJACENCY MULTI LIST

จัดทำโดย

นายพงษ์พันธุ์ เลาวพงศ์

รหัสนักศึกษา 66543206019-2

เสนอ

อาจารย์ปิยพล ยืนยงสถาวร

รายงานนี้เป็นส่วนหนึ่งของวิชา

ENGCE124

โครงสร้างข้อมูลและขั้นตอนวิธี

(Data Structures and Algorithms)

หลักสูตร วศ.บ.วิศวกรรมคอมพิวเตอร์

สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา ภาคพายัพ เชียงใหม่

ภาคเรียนที่ 1 ปีการศึกษา 2567

รายงาน

เรื่อง โปรแกรม GRAPH STRUCTURE BY ADJACENCY MULTI LIST

จัดทำโดย

นายพงษ์พันธุ์ เลาวพงศ์

รหัสนักศึกษา 66543206019-2

เสนอ

อาจารย์ปิยพล ยืนยงสถาวร

รายงานนี้เป็นส่วนหนึ่งของวิชา

ENGCE124

โครงสร้างข้อมูลและขั้นตอนวิธี

(Data Structures and Algorithms)

หลักสูตร วศ.บ.วิศวกรรมคอมพิวเตอร์

สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา ภาคพายัพ เชียงใหม่

ภาคเรียนที่ 1 ปีการศึกษา 2567

คำนำ

รายงานฉบับนี้เป็นส่วนหนึ่งของรายวิชา ENGCE124 โครงสร้างข้อมูลและขั้นตอนวิธี (Data Structures and Algorithms) หลักสูตร วศ.บ.วิศวกรรมคอมพิวเตอร์ สาขาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา ภาคพายัพ เชียงใหม่ ในระดับปริญญาตรีปีที่ 2 โดยมีจุดประสงค์ในการอธิบายโค้ดของโปรแกรม GRAPH STRUCTURE BY ADJACENCY MULTI LIST รวมถึงอธิบายหลักการทำงานของโปรแกรม GRAPH STRUCTURE BY ADJACENCY MULTI LIST และอธิบายผลลัพธ์การใช้งานโปรแกรม GRAPH STRUCTURE BY ADJACENCY MULTI LIST

ผู้จัดทำรายงานหวังว่า รายงานฉบับนี้จะเป็นประโยชน์กับผู้สนใจ หรือนักศึกษาทุกท่านที่กำลังหาศึกษาในหัวข้อของโปรแกรม GRAPH STRUCTURE BY ADJACENCY MULTI LIST หากมีข้อเสนอแนะหรือข้อผิดพลาดประการใด ผู้จัดทำขอน้อมรับไว้ และขออภัยมา ณ ที่นี้

ผู้จัดทำ

นายพงษ์พันธุ์ เลาวพงศ์

วันที่ 05/09/2567

สารบัญ

	หน้า
คำนำ	ก
สารบัญ	๗
โค้ดของโปรแกรม GRAPH STRUCTURE BY ADJACENCY MULTI LIST พร้อมคำอธิบาย	1
หลักการทำงานของโปรแกรม GRAPH STRUCTURE BY ADJACENCY MULTI LIST	4
ผลลัพธ์การใช้งานโปรแกรม GRAPH STRUCTURE BY ADJACENCY MULTI LIST	10
บรรณานุกรม	12

โค้ดของโปรแกรม GRAPH STRUCTURE BY ADJACENCY MULTI LIST พร้อมคำอธิบาย

```
#include <stdio.h> // ใช้ฟังก์ชัน printf

#include <conio.h> // ใช้ฟังก์ชัน getch

#define MaxEdge 8 // กำหนดจำนวนขอบสูงสุดของกราฟ

#define Block 5 // กำหนดจำนวนบล็อกของแต่ละโหนด

#define MaxNode 5 // กำหนดจำนวนโหนดสูงสุด

char NodeName[5] = {'A','B','C','D','E'};

char graph[MaxEdge][Block] = {

    {'0','A','B','2','5'},

    {'0','A','C','3','5'},

    {'0','A','D','4','6'},

    {'0','A','E','-','7'},

    {'0','B','C','6','7'},

    {'0','B','D','-','8'},

    {'0','C','E','-','8'},

    {'0','D','E','-','-'}

}; // ประกาศอาเรย์และเก็บข้อมูลของกราฟ

void DispArray2D(){ // แสดงค่าในอาเรย์ 2 มิติ

    int i,j; // i=แถว, j=คอลัมน์

    printf(" M V1 V2 E1 E2\n");

    for (i=0;i<MaxEdge;i++){ // ลูปแถว

        printf("%d ",i+1); // แสดงหมายเลขแถว
```

โค้ดของโปรแกรม GRAPH STRUCTURE BY ADJACENCY MULTI LIST พร้อมคำอธิบาย (ต่อ)

```

        for (j=0;j<Block;j++) // ลูปคอลัมน์

            printf("%c ",graph[i][j]); // แสดงค่าของแต่ละช่อง

        printf("\n");

    }

}

void DispSetOfVertex(){ // แสดงเซตของจุดยอด

    int i;

    printf("\nSet of Vertex = {");

    for (i=0;i<MaxNode;i++){

        printf("%c",NodeName[i]); // แสดงชื่อแต่ละโหนด

        if(i != MaxNode-1)

            printf(",");

    }

    printf("}\n");

}

void DispSetOfEdge(){ // แสดงเซตของขอบ

    int i,j;

    printf("\nSet of Edge = {");

    for (i=0;i<MaxEdge;i++) { // ลูปแถว

        printf("(%c,%c)",graph[i][1],graph[i][2]); // แสดงขอบ 1

        printf("(%c,%c)",graph[i][2],graph[i][1]); // แสดงขอบ 2
    }
}

```

โค้ดของโปรแกรม GRAPH STRUCTURE BY ADJACENCY MULTI LIST พร้อมคำอธิบาย (ต่อ)

```
}

printf("}\n");

}

int main(){

    printf("GRAPH ADJACENCY MULTI-LIST REPRESENTATION METHOD\n");

    printf("=====\n");

    DispArray2D();

    DispSetOfVertex();

    DispSetOfEdge();

    getch();

    return(0);

} // จบการทำงานของฟังก์ชัน main
```

หลักการการทำงานของโปรแกรม GRAPH STRUCTURE BY ADJACENCY MULTI LIST

โปรแกรม GRAPH STRUCTURE BY ADJACENCY MULTI LIST มีการสร้างและแสดงกราฟด้วยวิธี "Adjacency Multi List" ซึ่งเป็นวิธีการหนึ่งในการจัดเก็บข้อมูลของกราฟและทำให้สามารถเข้าถึงข้อมูลเกี่ยวกับจุดยอด (vertices) และขอบ (edges) ได้อย่างมีประสิทธิภาพ ในบทความนี้เราจะมาดูการทำงานของฟังก์ชันต่าง ๆ ที่ใช้ในการสร้างและแสดงผลกราฟอย่างละเอียด

1. การนำเข้าไลบรารี

```
#include <stdio.h> // ใช้ฟังก์ชัน printf
#include <conio.h> // ใช้ฟังก์ชัน getch
```

ในส่วนนี้จะใช้ #include เพื่อเรียกใช้งานไลบรารีที่จำเป็นสำหรับโปรแกรม

- #include <stdio.h> : ไลบรารีนี้ให้ฟังก์ชันพื้นฐานสำหรับการทำงานกับการป้อนและการแสดงผลข้อมูล เช่น printf ซึ่งใช้ในการพิมพ์ข้อความและข้อมูลไปยังหน้าจอ
- #include <conio.h> : ไลบรารีนี้ให้ฟังก์ชันที่เกี่ยวข้องกับการป้อนข้อมูลจากคีย์บอร์ดและการจัดการกับการแสดงผล เช่น getch ซึ่งใช้ในการรอการกดปุ่มจากผู้ใช้เพื่อให้โปรแกรมหยุดชั่วคราว

2. การกำหนดค่าคงที่ (Constants)ข

```
#define MaxEdge 8 // กำหนดจำนวนขอบสูงสุดของกราฟ
#define Block 5 // กำหนดจำนวนบล็อกของแต่ละโหนด
#define MaxNode 5 // กำหนดจำนวนโหนดสูงสุด
```

การใช้ #define คือการกำหนดค่าคงที่ที่ใช้ในโปรแกรม โดยจะมีรายละเอียดในการกำหนดค่า ดังนี้

- #define MaxEdge 8 : กำหนดจำนวนขอบสูงสุดที่กราฟสามารถมีได้เป็น 8 ขอบ ซึ่งหมายความว่าอาเรย์ graph จะเก็บข้อมูลขอบได้สูงสุด 8 ขอบ
- #define Block 5 : กำหนดจำนวนบล็อกที่แต่ละโหนดจะใช้ในการเก็บข้อมูลขอบและน้ำหนัก เป็น 5 บล็อก ซึ่งหมายความว่าแต่ละแถวในอาเรย์ graph จะมี 5 คอลัมน์
- #define MaxNode 5 : กำหนดจำนวนโหนดสูงสุดที่กราฟสามารถมีได้เป็น 5 โหนด ซึ่งหมายความว่าอาเรย์ NodeName จะเก็บชื่อของโหนดได้สูงสุด 5 ตัว

3. การประกาศอาเรย์สำหรับชื่อโหนด

```
char NodeName[5] = {'A','B','C','D','E'};
```

ในส่วนนี้จะประกาศอาเรย์ NodeName ที่เก็บชื่อของโหนดในกราฟ โดยเป็นอาเรย์ของชนิดข้อมูล char ขนาด 5 ตัว โดยกำหนดค่าเริ่มต้นของอาเรย์ให้มีชื่อโหนด 5 ตัว คือ 'A', 'B', 'C', 'D', และ 'E'

4. การประกาศอาเรย์สำหรับข้อมูลกราฟ

```
char graph[MaxEdge][Block] = {
    {'0','A','B','2','5'},
    {'0','A','C','3','5'},
    {'0','A','D','4','6'},
    {'0','A','E','-','7'},
    {'0','B','C','6','7'},
    {'0','B','D','-','8'},
    {'0','C','E','-','8'},
    {'0','D','E','-','-'},
}; // ประกาศอาเรย์และเก็บข้อมูลของกราฟ
```

ในส่วนนี้จะมีการประกาศอาเรย์ graph ซึ่งเป็นอาเรย์ 2 มิติ ขนาด MaxEdge x Block (8 x 5) สำหรับเก็บข้อมูลเกี่ยวกับขอบและน้ำหนักในกราฟ โดยกำหนดค่าเริ่มต้นของอาเรย์ ซึ่งแต่ละแถวแทนการเชื่อมโยงระหว่างจุดยอดสองจุดในกราฟ ข้อมูลในแต่ละแถวประกอบด้วย

- '0': รหัสหรือค่าตัวเลขที่ไม่ใช้
- 'A': จุดยอดต้นทาง
- 'B': จุดยอดปลายทาง
- '2': น้ำหนักของขอบระหว่าง 'A' และ 'B'
- '5': น้ำหนักที่สองสำหรับขอบเดียวกัน (ถ้ามี)

แต่ละแถวในอาเรย์ graph แสดงข้อมูลขอบที่เชื่อมต่อจุดยอดสองจุด และน้ำหนักที่เกี่ยวข้อง ซึ่งช่วยในการสร้างและแสดงกราฟในรูปแบบ "Adjacency Multi List"

5. ฟังก์ชัน DispArray2D สำหรับการแสดงข้อมูลของกราฟ

```
void DispArray2D(){ // แสดงค่าในอาเรย์ 2 มิติ

    int i,j; // i=แถว, j=คอลัมน์

    printf(" M V1 V2 E1 E2\n");

    for (i=0;i<MaxEdge;i++){ // ลูปแถว

        printf("%d ",i+1); // แสดงหมายเลขแถว

        for (j=0;j<Block;j++) // ลูปคอลัมน์

            printf("%c ",graph[i][j]); // แสดงค่าของแต่ละช่อง

        printf("\n");

    }

}
```

ในส่วนของฟังก์ชัน DispArray2D มีบทบาทสำคัญในการแสดงข้อมูลของกราฟในรูปแบบของอาเรย์ 2 มิติ โดยข้อมูลที่เก็บไว้ในอาเรย์นี้ประกอบด้วยรายละเอียดของขอบแต่ละขอบในกราฟ รวมถึงจุดยอดที่เชื่อมโยงกัน และน้ำหนักของขอบในรูปแบบของตาราง โดยการทำงานเริ่มต้นด้วยการกำหนดตัวแปร i และ j สำหรับการลูปผ่านแถวและคอลัมน์ของอาเรย์ จากนั้นพิมพ์หัวตารางเพื่อแสดงชื่อคอลัมน์ต่าง ๆ คือ "M", "V1", "V2", "E1", และ "E2" และใช้ลูป for เพื่อเดินทางผ่านแถวของอาเรย์ (แต่ละแถวแสดงถึงขอบหนึ่งของกราฟ) โดยพิมพ์หมายเลขของแถวที่ปัจจุบัน จากนั้นใช้ลูป for อีกตัวในการเดินทางผ่านคอลัมน์ของอาเรย์ เพื่อแสดงค่าของแต่ละช่องในตาราง และลงบรรทัดใหม่หลังจากแสดงข้อมูลของแต่ละแถวเสร็จสิ้น ซึ่งฟังก์ชันนี้ช่วยให้สามารถมองเห็นข้อมูลของกราฟในรูปแบบที่เป็นระเบียบและเข้าใจง่าย

6. ฟังก์ชัน DispSetOfVertex สำหรับแสดงเซตของจุดยอดของกราฟ

```
void DispSetOfVertex(){ // แสดงเซตของจุดยอด

    int i;

    printf("\nSet of Vertex = {");

    for (i=0;i<MaxNode;i++){

        printf("%c",NodeName[i]); // แสดงชื่อแต่ละโหนด

        if(i != MaxNode-1)

            printf(",");

    }

    printf("}\n");

}
```

ในส่วนของฟังก์ชัน DispSetOfVertex ทำหน้าที่ในการแสดงเซตของจุดยอด (vertex) ของกราฟ ซึ่งจุดยอดคือโหนดต่าง ๆ ที่ประกอบขึ้นเป็นกราฟ โดยการทำงาน เริ่มต้นด้วยการพิมพ์ข้อความเปิดของเซตของจุดยอดโดยใช้ลูป for เพื่อเดินทางผ่านอาร์เรย์ NodeName ซึ่งเก็บชื่อของจุดยอด เพื่อแสดงชื่อของแต่ละจุดยอด และเพิ่มเครื่องหมายจุลภาค (,) ระหว่างชื่อจุดยอดที่แสดง จากนั้นพิมพ์ปิดของเซตของจุดยอดเมื่อแสดงชื่อทั้งหมดเสร็จสิ้น ซึ่งการแสดงผลของจุดยอดช่วยให้เราสามารถมองเห็นว่าในกราฟนี้มีจุดยอดทั้งหมดกี่จุดและแต่ละจุดมีชื่อว่าอะไร

7. ฟังก์ชัน DispSetOfEdge สำหรับการแสดงเซตของขอบของกราฟ

```
void DispSetOfEdge(){ // แสดงเซตของขอบ

    int i,j;

    printf("\nSet of Edge = {");

    for (i=0;i<MaxEdge;i++) { // ลูปแถว
```

7. ฟังก์ชัน DispSetOfEdge สำหรับการแสดงเซตของขอบของกราฟ

```

        printf("(%c,%c)",graph[i][1],graph[i][2]); // แสดงขอบ 1

        printf("(%c,%c)",graph[i][2],graph[i][1]); // แสดงขอบ 2

    }

    printf("\n");

}

```

ในส่วนของฟังก์ชัน DispSetOfEdge ทำหน้าที่ในการแสดงเซตของขอบของกราฟ ขอบคือการเชื่อมโยงระหว่างจุดยอดสองจุดซึ่งมีความสำคัญในการแสดงความสัมพันธ์ในกราฟ โดยการทำงาน เริ่มต้นด้วยการพิมพ์ข้อความเปิดของเซตของขอบ โดยใช้ลูป for เพื่อเดินทางผ่านแถวของอาร์เรย์ graph เพื่อแสดงขอบที่เชื่อมต่อจากจุดยอด V1 ไปยัง V2 และ แสดงขอบที่เชื่อมต่อจากจุดยอด V2 ไปยัง V1 (เนื่องจากกราฟในที่นี่เป็นกราฟที่มีทิศทาง) จากนั้นจะพิมพ์ปิดของเซตของขอบเมื่อแสดงขอบทั้งหมดเสร็จสิ้น ซึ่งการแสดงเซตของขอบทำให้เราสามารถเห็นความสัมพันธ์ระหว่างจุดยอดต่าง ๆ ในกราฟได้ชัดเจน

8. ฟังก์ชัน main() ฟังก์ชันหลักของโปรแกรม

```

int main(){

    printf("GRAPH ADJACENCY MULTI-LIST REPRESENTATION METHOD\n");

    printf("=====\n");

    DispArray2D();

    DispSetOfVertex();

    DispSetOfEdge();

    getch();

    return(0);

} // จบการทำงานของฟังก์ชัน main

```

ในส่วนของ ฟังก์ชัน `main()` เป็นจุดเริ่มต้นของโปรแกรมที่ควบคุมการทำงานของฟังก์ชันทั้งหมดในโปรแกรม โดยการทำงานจะเริ่มต้นพิมพ์ข้อความหัวข้อของโปรแกรม จากนั้นจะเรียกใช้งานฟังก์ชันประกอบไปด้วย

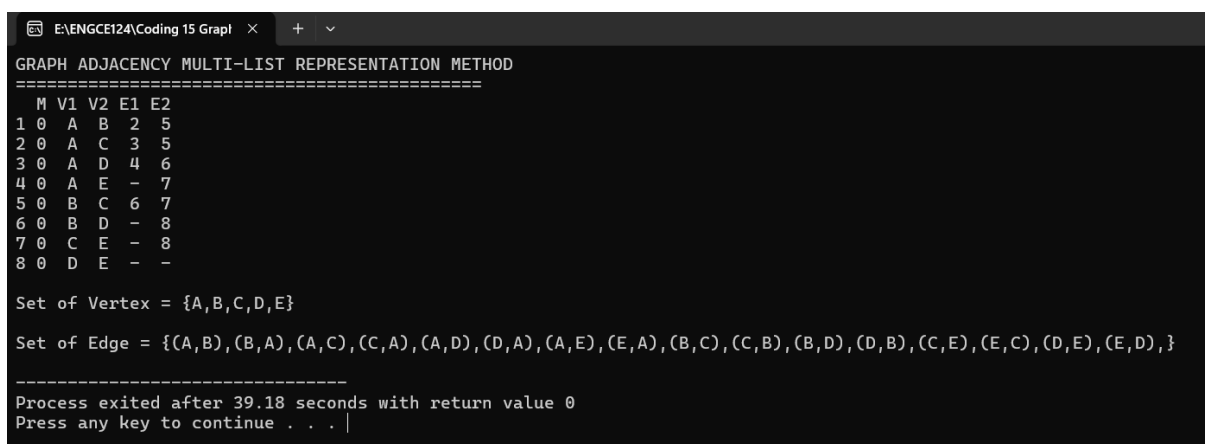
- ฟังก์ชัน `DispArray2D()` เพื่อแสดงข้อมูลของกราฟในรูปแบบของอาเรย์ 2 มิติ
- ฟังก์ชัน `DispSetOfVertex()` เพื่อแสดงเซตของจุดยอด
- ฟังก์ชัน `DispSetOfEdge()` เพื่อแสดงเซตของขอบ

ซึ่งเมื่อเรียกใช้งานฟังก์ชันเสร็จ จะทำการรอกการกดปุ่มจากผู้ใช้ (ใช้ `getch()`) เพื่อให้ผู้ใช้สามารถดูผลลัพธ์ ก่อนที่โปรแกรมจะจบการทำงาน โดยฟังก์ชัน `main()` ช่วยให้เราสามารถควบคุมการทำงานของโปรแกรมและแสดงผลข้อมูลกราฟได้อย่างครบถ้วน

ผลลัพธ์การใช้งานโปรแกรม GRAPH STRUCTURE BY ADJACENCY MULTI LIST

โปรแกรม GRAPH STRUCTURE BY ADJACENCY MULTI LIST สร้างขึ้นเพื่อแสดงโครงสร้างของกราฟ (Graph) โดยใช้วิธีการเก็บข้อมูลแบบ "Adjacency Multi List" ซึ่งมีการแสดงผลในรูปแบบต่าง ๆ ที่สำคัญ ได้แก่ ข้อมูลของกราฟในรูปแบบอาเรย์ 2 มิติ, เซตของจุดยอด (vertices), และเซตของขอบ (edges) โดยโปรแกรมนี้นี้มี 3 ฟังก์ชันหลักที่ทำหน้าที่แสดงผลข้อมูลกราฟ ได้แก่

- ฟังก์ชัน DispArray2D() : แสดงข้อมูลกราฟในรูปแบบอาเรย์ 2 มิติ
- ฟังก์ชัน DispSetOfVertex() : แสดงเซตของจุดยอด
- ฟังก์ชัน DispSetOfEdge() : แสดงเซตของขอบ



```

E:\ENGCE124\Coding 15 Graph >
GRAPH ADJACENCY MULTI-LIST REPRESENTATION METHOD
=====
M V1 V2 E1 E2
1 0 A B 2 5
2 0 A C 3 5
3 0 A D 4 6
4 0 A E - 7
5 0 B C 6 7
6 0 B D - 8
7 0 C E - 8
8 0 D E - -

Set of Vertex = {A,B,C,D,E}

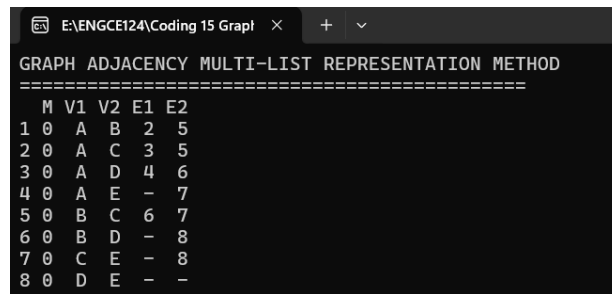
Set of Edge = {(A,B),(B,A),(A,C),(C,A),(A,D),(D,A),(A,E),(E,A),(B,C),(C,B),(B,D),(D,B),(C,E),(E,C),(D,E),(E,D),}

-----
Process exited after 39.18 seconds with return value 0
Press any key to continue . . .
  
```

1. การแสดงผลอาเรย์ 2 มิติของกราฟ [DispArray2D()]

เมื่อเรียกใช้ฟังก์ชัน DispArray2D() โปรแกรมจะแสดงข้อมูลที่เก็บอยู่ในอาเรย์ graph โดยมีการจัดรูปแบบเป็นตารางซึ่งมีข้อมูลสำคัญเกี่ยวกับขอบแต่ละขอบของกราฟ โดยมีโครงสร้างของการแสดงผล ประกอบไปด้วย

- M: หมายเลขของแถว (ขอบที่กราฟเชื่อมต่อ)
- V1: จุดยอดต้นทางของขอบ (Vertex 1)
- V2: จุดยอดปลายทางของขอบ (Vertex 2)
- E1: น้ำหนักหรือข้อมูลเสริมของขอบ
- E2: น้ำหนักหรือข้อมูลเสริมที่สอง (ถ้ามี)



GRAPH ADJACENCY MULTI-LIST REPRESENTATION METHOD

M	V1	V2	E1	E2
1	0	A	B	2
2	0	A	C	3
3	0	A	D	4
4	0	A	E	-
5	0	B	C	6
6	0	B	D	-
7	0	C	E	-
8	0	D	E	-

จากผลลัพธ์ข้างต้นแสดงให้เห็นถึง แถวแรก ($M = 1$) จะแสดงขอบระหว่างจุดยอด 'A' และ 'B' มีน้ำหนัก 2 และ 5 และแถวที่สอง ($M = 2$) จะแสดงขอบระหว่าง 'A' และ 'C' มีน้ำหนัก 3 และ 5 โดยแต่ละแถวแสดงถึงขอบระหว่างจุดยอดต้นทางและปลายทาง รวมถึงน้ำหนักของขอบที่ระบุ การแสดงผลนี้ช่วยให้ผู้ใช้สามารถเห็นโครงสร้างของกราฟทั้งหมดและข้อมูลขอบที่เชื่อมต่อจุดยอดต่าง ๆ

2. การแสดงเซตของจุดยอด [DispSetOfVertex()]

เมื่อเรียกใช้ฟังก์ชัน DispSetOfVertex() แสดงชื่อของจุดยอด (Vertices) ที่มีอยู่ในกราฟ ซึ่งเก็บอยู่ในอาร์เรย์ NodeName ฟังก์ชันนี้จะแสดงผลเป็นเซตของจุดยอดที่ถูกกำหนดไว้ในโปรแกรม

Set of Vertex = {A,B,C,D,E}

จากผลลัพธ์ข้างต้น จะแสดงชื่อของจุดยอดทั้งหมดในกราฟ ซึ่งมี 5 จุด ได้แก่ 'A', 'B', 'C', 'D', และ 'E' ซึ่งเซตนี้เป็นส่วนประกอบพื้นฐานของกราฟ โดยจุดยอดเหล่านี้จะถูกเชื่อมโยงกันด้วยขอบ การแสดงเซตของจุดยอดทำให้เราสามารถมองเห็นได้ว่ากราฟนี้มีจุดยอดทั้งหมดกี่จุด และมีชื่อของจุดยอดแต่ละตัวอย่างไร

3. การแสดงเซตของขอบ [DispSetOfEdge()]

เมื่อเรียกใช้ฟังก์ชัน DispSetOfEdge() ทำหน้าที่แสดงเซตของขอบ (Edges) ซึ่งแสดงความสัมพันธ์ระหว่างจุดยอดในกราฟ ขอบแต่ละอันถูกแสดงผลเป็นคู่ของจุดยอดที่เชื่อมต่อกัน ฟังก์ชันนี้จะแสดงทั้งขอบจากจุดยอดต้นทางไปยังจุดยอดปลายทาง และจากจุดยอดปลายทางกลับไปยังจุดยอดต้นทาง

Set of Edge = {(A,B),(B,A),(A,C),(C,A),(A,D),(D,A),(A,E),(E,A),(B,C),(C,B),(B,D),(D,B),(C,E),(E,C),(D,E),(E,D),}

จากผลลัพธ์ข้างต้นแสดงให้เห็นถึง ขอบแรกคือ (A,B) ซึ่งหมายถึงมีการเชื่อมต่อระหว่างจุดยอด 'A' และ 'B' และ ขอบที่สองคือ (B,A) ซึ่งแสดงว่ามีการเชื่อมโยงในทิศทางกลับกันจาก 'B' ไปยัง 'A' ฟังก์ชันนี้แสดงขอบทั้งหมดในกราฟในลักษณะที่ครอบคลุมทั้งการเชื่อมโยงแบบไปและกลับระหว่างจุดยอดแต่ละคู่ การแสดงเซตของขอบทำให้เราสามารถมองเห็นความสัมพันธ์ระหว่างจุดยอดได้ชัดเจนขึ้น ทั้งในทิศทางที่ไปและกลับของขอบแต่ละขอบ

บรรณานุกรม

ChatGPT. (-). Graph Structure Representation with Adjacency Multi List. สืบค้น 5 กันยายน 2567,
จาก <https://chatgpt.com/>