



รายงาน

เรื่อง โปรแกรม NODE DIRECTORY METHOD

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 NODE DIRECTORY METHOD

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 NODE DIRECTORY METHOD รวมถึงอธิบายหลักการ ทำงานของโปรแกรม NODE DIRECTORY METHOD และอธิบายผลลัพธ์การใช้งานโปรแกรม NODE DIRECTORY METHOD

ผู้จัดทำรายงานหวังว่า รายงานฉบับนี้จะเป็นประโยชน์กับผู้สนใจ หรือนักศึกษาทุกท่านที่กำลังหา ศึกษาในหัวข้อของโปรแกรม NODE DIRECTORY METHOD หากมีข้อเสนอแนะหรือข้อผิดพลาดประการใด ผู้จัดทำขอน้อมรับไว้ และขออภัยมา ณ ที่นี้

ผู้จัดทำ

นายพงษ์พันธุ์ เลาวพงศ์

วันที่ 05/09/2567

สารบัญ

	หน้า
คำนำ	ก
สารบัญ	๗
โค้ดของโปรแกรม NODE DIRECTORY METHOD พร้อมคำอธิบาย	1
หลักการทำงานของโปรแกรม NODE DIRECTORY METHOD	5
ผลลัพธ์การใช้งานโปรแกรม NODE DIRECTORY METHOD	13
บรรณานุกรม	15

โค้ดของโปรแกรม NODE DIRECTORY METHOD พร้อมคำอธิบาย

```
#include <stdio.h> // ใช้ printf

#include <conio.h> // ใช้ getch

#define MaxNode 4 // กำหนดจำนวนโหนดสูงสุด

#define Block 4 // กำหนดขนาดบล็อกของแต่ละโหนด

#define MaxEdge 6 // กำหนดจำนวนเส้นเชื่อมสูงสุดของกราฟ

char Head[MaxNode][Block] = {

    {'A','-','1','1'},

    {'B','-','3','2'},

    {'C','-','2','5'},

    {'D','-','0',NULL},

}; // ประกาศอาเรย์และเก็บข้อมูลโหนดหลักของกราฟ

char Edge[MaxEdge][2] = {

    {'2','B'},

    {'5','A'},

    {'9','C'},

    {'7','D'},

    {'8','B'},

    {'6','D'},

}; // ประกาศอาเรย์และเก็บเส้นเชื่อมของกราฟ

void DispHead() // แสดงโหนดหลักในอาเรย์สองมิติ

{
```

โค้ดของโปรแกรม NODE DIRECTORY METHOD พร้อมคำอธิบาย (ต่อ)

```

int i,j; // i=แถว, j=คอลัมน์

printf("NODE...\n");

printf("No. Name Data Edge Pointer\n");

for (i=0;i<MaxNode;i++) // วนแถว
{
    printf("%d  ",i+1); // แสดงหมายเลขแถว

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

        printf("%c  ",Head[i][j]); // แสดงโหนด

    printf("\n");
}
}

void DispEdge() // แสดงเส้นเชื่อมในอาเรย์สองมิติ
{
    int i,j; // i=แถว, j=คอลัมน์

    printf("EDGE...\n");

    printf("No. Weight Node\n");

    for (i=0;i<MaxEdge;i++) // วนแถว
    {
        printf("%d  ",i+1); // แสดงหมายเลขแถว

        for (j=0;j<2;j++) // วนคอลัมน์

            printf("%c  ",Edge[i][j]); // แสดงเส้นเชื่อม
    }
}

```

โค้ดของโปรแกรม NODE DIRECTORY METHOD พร้อมคำอธิบาย (ต่อ)

```

    printf("\n");

}

}

void DispSetOfVertex() // แสดงเซตของโหนด (Vertex)
{
    int i;

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

    for (i=0;i<MaxNode;i++)
    {
        printf("%c",Head[i][0]); // แสดงชื่อของแต่ละโหนด

        if(i != MaxNode-1)

            printf(",");

    }

    printf("}\n");
}

void DispSetOfEdge() // แสดงเซตของเส้นเชื่อม (Edge)
{
    int i,j,AmtEdge,PT;

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

    for (i=0;i<MaxNode;i++) // วงแถว

    {

```

โค้ดของโปรแกรม NODE DIRECTORY METHOD พร้อมคำอธิบาย (ต่อ)

```

    AmtEdge=Head[i][2]-48; // แปลงตัวอักษรเป็นตัวเลขเพื่อหาจำนวนเส้นเชื่อม

    PT=Head[i][3]-48; // แปลงตัวอักษรเป็นตัวเลขเพื่อหาจุดเริ่มต้นของอาเรย์

    for (j=0;j<AmtEdge;j++) // วนตามจำนวนเส้นเชื่อม
    {
        printf("(%c%c)%c",Head[i][0],Edge[PT-1+j][1],Edge[PT-1+j][0]); // แสดงเส้นเชื่อมและ
        // นำหน้าของเส้นเชื่อม
    }

    printf("\n");
}

int main()
{
    printf("GRAPH NODE DIRECTORY REPRESENTATION METHOD\n");

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

    DispHead();

    DispEdge();

    DispSetOfVertex();

    DispSetOfEdge();

    getch();

    return(0);
} // จบโปรแกรมหลัก

```


หลักการการทำงานของโปรแกรม NODE DIRECTORY METHOD

โปรแกรม โปรแกรม NODE DIRECTORY METHOD มีหน้าที่ในการสร้างโครงสร้างกราฟโดยใช้วิธีการเก็บข้อมูลของโหนดและเส้นเชื่อมในรูปแบบอาร์เรย์สองมิติ (2D Array) จากนั้นจะแสดงข้อมูลของกราฟ ได้แก่ เซ็ตของโหนด (Vertex) และเซตของเส้นเชื่อม (Edge) รวมถึงน้ำหนักของแต่ละเส้นเชื่อม โปรแกรมสามารถนำไปใช้กับกราฟที่มีน้ำหนักได้ทั้งกราฟแบบมีทิศทาง (Directed Graph) และกราฟแบบไม่มีทิศทาง (Undirected Graph)

1. ส่วนในการประกาศในการไลบรารี

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

ในส่วนนี้โปรแกรมจะทำการ include ไลบรารีมาตรฐานสองตัว ได้แก่

- <stdio.h> : เป็นไลบรารีมาตรฐานที่ใช้สำหรับฟังก์ชันการทำงานเกี่ยวกับการรับ-ส่งข้อมูล เช่น ฟังก์ชัน printf() ที่ใช้ในการแสดงข้อความหรือข้อมูลบนหน้าจอ
- <conio.h> : ไลบรารีนี้ใช้สำหรับฟังก์ชันที่เกี่ยวกับการควบคุมหน้าจอและการรับข้อมูลจากคีย์บอร์ด เช่น ฟังก์ชัน getch() ซึ่งทำหน้าที่รับคีย์บอร์ดจากผู้ใช้โดยไม่แสดงผลบนหน้าจอ และรอให้ผู้ใช้กดปุ่มใด ๆ เพื่อดำเนินการต่อ

2. การกำหนดค่าคงที่

```
#define MaxNode 4 // กำหนดจำนวนโหนดสูงสุด
#define Block 4 // กำหนดขนาดบล็อกของแต่ละโหนด
#define MaxEdge 6 // กำหนดจำนวนเส้นเชื่อมสูงสุดของกราฟ
```

บรรทัดนี้ใช้การกำหนดนิยาม (#define) เพื่อกำหนดค่าคงที่ของจำนวนโหนดในกราฟ (MaxNode), ขนาดของบล็อกข้อมูลของแต่ละโหนด (Block), และจำนวนเส้นเชื่อมสูงสุดที่สามารถเก็บได้ในอาร์เรย์ (MaxEdge) ซึ่งตัวแปรโหนดเหล่านี้ถูกเก็บในอาร์เรย์ 2 มิติและจะใช้ในโปรแกรมในส่วนต่าง ๆ ที่ต้องการทราบดังกล่าวสูงสุด

3. การเก็บข้อมูลโหนดและเส้นเชื่อม

```
char Head[MaxNode][Block] = {

    {'A','-','1','1'},

    {'B','-','3','2'},

    {'C','-','2','5'},

    {'D','-','0',NULL},

}; // ประกาศอาเรย์และเก็บข้อมูลโหนดหลักของกราฟ

char Edge[MaxEdge][2] = {

    {'2','B'},

    {'5','A'},

    {'9','C'},

    {'7','D'},

    {'8','B'},

    {'6','D'},

}; // ประกาศอาเรย์และเก็บเส้นเชื่อมของกราฟ
```

ในส่วนนี้โปรแกรมใช้การเก็บข้อมูลของโหนดในอาเรย์สองมิติ Head และเก็บข้อมูลของเส้นเชื่อมในอาเรย์ Edge โดยแต่ละโหนดใน Head ประกอบด้วย

- ตัวอักษรแรกเป็นชื่อโหนด (เช่น A, B, C, D)
- ตัวอักษรถัดมาเป็นข้อมูลเสริม (-)
- ตัวเลขที่บอกจำนวนเส้นเชื่อม (1, 3, 2, 0)
- ตัวเลขที่บอกตำแหน่งเริ่มต้นของเส้นเชื่อมในอาเรย์ Edge

ส่วน Edge เก็บข้อมูลน้ำหนักของเส้นเชื่อมและโหนดที่เชื่อมต่อกัน (เช่น 2B หมายถึงเส้นเชื่อมจากโหนดอื่นมายังโหนด B ที่มีน้ำหนักเท่ากับ 2)

4. ฟังก์ชัน DispHead สำหรับแสดงข้อมูลของโหนด

```
void DispHead() // แสดงโหนดหลักในอาเรย์สองมิติ
{
    int i,j; // i=แถว, j=คอลัมน์

    printf("NODE...\n");

    printf("No. Name Data Edge Pointer\n");

    for (i=0;i<MaxNode;i++) // วนแถว
    {
        printf("%d ",i+1); // แสดงหมายเลขแถว

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

            printf("%c ",Head[i][j]); // แสดงโหนด

        printf("\n");
    }
}
```

ในส่วนของฟังก์ชัน DispHead มีหน้าที่ในการแสดงข้อมูลของโหนดทั้งหมดในอาเรย์ Head ซึ่งจัดเก็บข้อมูลโหนดในกราฟ เช่น ชื่อโหนด, ข้อมูลเกี่ยวกับจำนวนเส้นเชื่อม, และตำแหน่งการเชื่อมต่อ โดยการทำงานของฟังก์ชันนี้เริ่มต้นด้วยการแสดงหัวตาราง ซึ่งเป็นการพิมพ์ข้อความ "NODE..." ตามด้วยการกำหนดหัวชื่อของตารางที่จะแสดงรายละเอียดเกี่ยวกับโหนดในกราฟ ได้แก่ "No." (หมายเลขโหนด), "Name" (ชื่อโหนด), "Data" (ข้อมูลที่โหนดเก็บไว้), "Edge" (จำนวนเส้นเชื่อม), และ "Pointer" (ตำแหน่งของเส้นเชื่อมในอาเรย์ Edge) การแสดงหัวตารางนี้มีจุดประสงค์เพื่อให้ผู้ใช้งานสามารถมองเห็นและเข้าใจโครงสร้างของกราฟได้ง่ายขึ้น โดยเริ่มจากการเข้าใจแต่ละส่วนของโหนดก่อน ต่อจากนั้น ฟังก์ชันจะทำงานโดยใช้ลูป for เพื่อนำข้อมูลแต่ละแถวจากอาเรย์ Head มาแสดง แต่ละแถวจะแสดงข้อมูลของโหนดแต่ละตัวในกราฟ เช่น โหนดที่มีชื่อว่า A, B, C, และ D โดยข้อมูลที่ถูกแสดงออกมานั้นจะรวมถึงชื่อโหนดและจำนวนเส้นเชื่อมที่โหนดนั้นมี ซึ่งลูปนี้จะวนรอบเพื่อนำข้อมูลทุกโหนดมาแสดงผล เมื่อการแสดงผลเสร็จสิ้น ผลลัพธ์จะออกมาในรูปแบบตารางที่ประกอบด้วยข้อมูลทั้งหมดของโหนดแต่ละตัวในกราฟ ตารางนี้จะช่วยให้ผู้ใช้งานสามารถมองเห็นภาพรวม

ของโครงสร้างกราฟอย่างชัดเจน โดยเฉพาะส่วนของโหนดและเส้นเชื่อมที่เกี่ยวข้อง การแสดงผลเช่นนี้ไม่เพียงแต่ช่วยในการตรวจสอบข้อมูลของโหนดแต่ละตัว แต่ยังช่วยให้การทำงานกับกราฟเป็นเรื่องง่ายและมีประสิทธิภาพมากขึ้น

5. ฟังก์ชัน DispEdge สำหรับแสดงข้อมูลเส้นเชื่อมในอาเรย์

```
void DispEdge() // แสดงเส้นเชื่อมในอาเรย์สองมิติ
{
    int i,j; // i=แถว, j=คอลัมน์

    printf("EDGE...\n");

    printf("No. Weight Node\n");

    for (i=0;i<MaxEdge;i++) // วนแถว
    {
        printf("%d ",i+1); // แสดงหมายเลขแถว

        for (j=0;j<2;j++) // วนคอลัมน์

            printf("%c ",Edge[i][j]); // แสดงเส้นเชื่อม

        printf("\n");
    }
}
```

ในส่วนของ ฟังก์ชัน DispEdge มีหน้าที่แสดงข้อมูลเส้นเชื่อมในอาเรย์ Edge ซึ่งประกอบด้วยน้ำหนักของเส้นเชื่อมและโหนดที่เชื่อมต่อ การทำงานของฟังก์ชันนี้เริ่มต้นด้วยการแสดงข้อความ "EDGE..." ซึ่งเป็นการเตรียมการแสดงผลข้อมูลเส้นเชื่อมในกราฟ โดยหัวตารางที่จะแสดงต่อจากข้อความนี้จะประกอบไปด้วยข้อมูลสำคัญที่เกี่ยวข้องกับเส้นเชื่อม ได้แก่ "No." (หมายเลขของเส้นเชื่อม), "Weight" (น้ำหนักของเส้นเชื่อม), และ "Node" (ชื่อโหนดที่เส้นเชื่อมนั้นเชื่อมต่อ) การจัดโครงสร้างหัวตารางนี้ทำขึ้นเพื่อช่วยให้ผู้ใช้งานสามารถเข้าใจข้อมูลที่จะแสดงในลำดับถัดไปได้ง่าย ๆ และสามารถตรวจสอบความสัมพันธ์ระหว่างโหนดต่าง ๆ ได้อย่างชัดเจน ถัดจากการแสดงหัวตาราง ฟังก์ชันจะใช้ลูป for เพื่อวนลูปผ่านข้อมูลในอาเรย์ Edge ซึ่งเก็บข้อมูลเกี่ยวกับเส้นเชื่อมของกราฟ ในแต่ละรอบของลูปจะดึงข้อมูลเกี่ยวกับน้ำหนักของเส้นเชื่อมออกมา เช่น

ค่า 2, 5 หรือ 9 และชื่อของโหนดที่เส้นเชื่อมนั้นเชื่อมต่อกัน เช่น B, A หรือ C ฟังก์ชันจะทำงานโดยการแสดงรายละเอียดของเส้นเชื่อมแต่ละเส้นในกราฟออกมาให้ครบถ้วน ซึ่งทำให้เห็นได้อย่างชัดเจนว่าโหนดใดเชื่อมต่อกันและด้วยน้ำหนักเท่าใด ผลลัพธ์จากการทำงานของฟังก์ชันนี้คือการแสดงข้อมูลเส้นเชื่อมทั้งหมดในกราฟ โดยแสดงน้ำหนักของแต่ละเส้นเชื่อมพร้อมกับโหนดที่เกี่ยวข้อง ซึ่งช่วยให้ผู้ใช้งานสามารถเข้าใจโครงสร้างของกราฟในส่วนของเส้นเชื่อมได้อย่างละเอียด การแสดงผลในรูปแบบนี้ช่วยให้การวิเคราะห์กราฟเป็นเรื่องง่ายขึ้น เพราะสามารถเห็นได้ทั้งน้ำหนักและการเชื่อมต่อของแต่ละโหนดอย่างชัดเจนในรูปแบบที่เป็นระเบียบ

6. ฟังก์ชัน DispSetOfVertex สำหรับแสดงเซตของโหนด

```
void DispSetOfVertex() // แสดงเซตของโหนด (Vertex)
{
    int i;

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

    for (i=0;i<MaxNode;i++)
    {
        printf("%c",Head[i][0]); // แสดงชื่อของแต่ละโหนด

        if(i != MaxNode-1)
            printf(",");
    }

    printf("}\n");
}
```

ในส่วนของฟังก์ชัน DispSetOfVertex ทำหน้าที่แสดงเซตของโหนด (Vertex) ที่มีอยู่ในกราฟ โดยการทำงานของฟังก์ชันนี้เริ่มต้นด้วยการใช้ลูป for เพื่อวนรอบแต่ละแถวในอาร์เรย์ Head โดยที่แต่ละแถวเก็บข้อมูลเกี่ยวกับโหนดในกราฟ ในแต่ละรอบของลูป ฟังก์ชันจะดึงชื่อของโหนดออกมา เช่น A, B, C, และ D จากนั้นจะรวบรวมชื่อโหนดทั้งหมดที่ดึงออกมาเป็นเซตในรูปแบบ {A, B, C, D} การจัดการข้อมูลในรูปแบบเซตนี้มีข้อดีคือช่วยให้การแสดงผลเป็นระเบียบและเข้าใจง่าย เมื่อการดึงข้อมูลเสร็จสิ้น ฟังก์ชันจะทำการแสดงเซตของโหนดทั้งหมดที่มีอยู่ในกราฟ การแสดงผลในรูปแบบเซตนี้ทำให้ผู้ใช้งานสามารถมองเห็นโหนดทั้งหมดที่

ประกอบเป็นกราฟได้อย่างชัดเจนและรวดเร็ว ทำให้การตรวจสอบโครงสร้างของกราฟเป็นเรื่องง่ายและสะดวกขึ้น

7. ฟังก์ชัน DispSetOfEdge สำหรับแสดงเซตของเส้นเชื่อม

```
void DispSetOfEdge() // แสดงเซตของเส้นเชื่อม (Edge)
{
    int i,j,AmtEdge,PT;

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

    for (i=0;i<MaxNode;i++) // วนแถว
    {
        AmtEdge=Head[i][2]-48; // แปลงตัวอักษรเป็นตัวเลขเพื่อหาจำนวนเส้นเชื่อม
        PT=Head[i][3]-48; // แปลงตัวอักษรเป็นตัวเลขเพื่อหาจุดเริ่มต้นของอาเรย์
        for (j=0;j<AmtEdge;j++) // วนตามจำนวนเส้นเชื่อม
        {
            printf("(%c%c)%c,",Head[i][0],Edge[PT-1+j][1],Edge[PT-1+j][0]); // แสดงเส้นเชื่อมและ
            // น้ำหนักของเส้นเชื่อม
        }
    }

    printf("}\n");
}
```

ในส่วนของฟังก์ชัน DispSetOfEdge ทำหน้าที่แสดงเซตของเส้นเชื่อม (Edge) พร้อมน้ำหนักของเส้นเชื่อมแต่ละเส้นในกราฟ โดยการทำงานฟังก์ชันนี้เริ่มต้นด้วยการใช้ลูปเพื่อวนผ่านแต่ละโหนดในอาเรย์ Head โดยจะอ้างอิงข้อมูลจำนวนเส้นเชื่อม (AmtEdge) และตำแหน่งเริ่มต้น (PT) เพื่อนำข้อมูลเส้นเชื่อมจากอาเรย์ Edge มาใช้ ในแต่ละรอบของลูป ฟังก์ชันจะดึงข้อมูลเส้นเชื่อมที่เชื่อมโยงกับโหนดนั้น ๆ ออกมา สำหรับแต่ละโหนด ฟังก์ชันจะทำการแสดงเส้นเชื่อมของโหนดนั้นพร้อมกับน้ำหนักของแต่ละเส้นเชื่อมในรูปแบบ (Node1

Node2) Weight เช่น (A B) 2 ซึ่งแสดงให้เห็นว่าโหนด A เชื่อมต่อกับโหนด B ด้วยน้ำหนัก 2 การจัดรูปแบบนี้ช่วยให้ผู้ใช้งานสามารถเข้าใจความสัมพันธ์ระหว่างโหนดต่าง ๆ ได้อย่างชัดเจน ผลลัพธ์ที่ได้จากฟังก์ชันนี้คือการแสดงเซตของเส้นเชื่อมทั้งหมดในกราฟพร้อมกับน้ำหนักของแต่ละเส้นเชื่อม การแสดงผลในลักษณะนี้ช่วยให้ผู้ใช้งานสามารถเข้าใจโครงสร้างการเชื่อมต่อในกราฟได้อย่างครบถ้วนและชัดเจน ทำให้การวิเคราะห์และทำงานกับกราฟเป็นเรื่องง่ายขึ้น

8. ฟังก์ชัน main ฟังก์ชันหลักของโปรแกรม

```
int main()
{
    printf("GRAPH NODE DIRECTORY REPRESENTATION METHOD\n");

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

    DispHead();

    DispEdge();

    DispSetOfVertex();

    DispSetOfEdge();

    getch();

    return(0);

} // จบโปรแกรมหลัก
```

ในส่วนของฟังก์ชัน main เป็นฟังก์ชันหลักที่ทำหน้าที่ในการเริ่มต้นการทำงานของโปรแกรม โดยเรียกใช้ฟังก์ชันอื่น ๆ เพื่อแสดงผลข้อมูลของกราฟในรูปแบบต่าง ๆ ได้แก่ ข้อมูลของโหนด, ข้อมูลของเส้นเชื่อม, เซตของโหนด และเซตของเส้นเชื่อม โดยการทำงานแต่ละส่วนในฟังก์ชัน main ประกอบไปด้วย

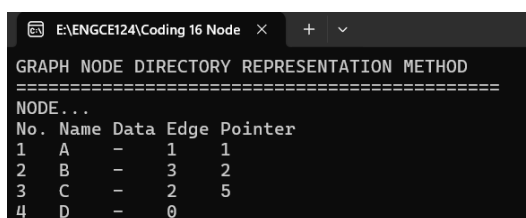
- แสดงข้อความหัวข้อ: ฟังก์ชันจะเริ่มต้นด้วยการแสดงหัวข้อ "GRAPH NODE DIRECTORY REPRESENTATION METHOD" และเส้นแบ่ง เพื่อให้ผู้ใช้ทราบว่าโปรแกรมนี้เกี่ยวข้องกับการแสดงผลโครงสร้างกราฟในรูปแบบของ Node Directory Method

- เรียกใช้ฟังก์ชัน `DispHead`: ฟังก์ชันนี้ทำการแสดงข้อมูลของโหนดทั้งหมดในกราฟ เช่น ชื่อโหนด, จำนวนเส้นเชื่อม และตำแหน่งของเส้นเชื่อมในอาร์เรย์เส้นเชื่อม (`Edge Array`) โดยแสดงในรูปแบบตารางที่เข้าใจง่าย
- เรียกใช้ฟังก์ชัน `DispEdge`: โปรแกรมแสดงข้อมูลเส้นเชื่อมทั้งหมดของกราฟ พร้อมทั้งระบุว่าเส้นเชื่อมนั้นมีน้ำหนักเท่าไร และเชื่อมต่อกับโหนดใดบ้าง
- เรียกใช้ฟังก์ชัน `DispSetOfVertex`: ฟังก์ชันนี้จะดึงชื่อโหนดจากอาร์เรย์ `Head` และแสดงเซตของโหนดทั้งหมดที่มีในกราฟ ตัวอย่างผลลัพธ์เช่น {A, B, C, D} ซึ่งทำให้เห็นว่ากราฟมีโหนดใดบ้าง
- เรียกใช้ฟังก์ชัน `DispSetOfEdge`: ฟังก์ชันนี้แสดงเซตของเส้นเชื่อมในกราฟ โดยแสดงคู่ของโหนดที่เชื่อมกันและน้ำหนักของเส้นเชื่อมในรูปแบบ (โหนด1 โหนด2) น้ำหนัก เช่น (A B) 2, (C D) 5, ... ซึ่งช่วยให้ผู้ใช้งานเห็นภาพการเชื่อมต่อระหว่างโหนดและความสัมพันธ์ระหว่างเส้นเชื่อมได้ชัดเจน
- ใช้คำสั่ง `getch()`: หลังจากแสดงผลทั้งหมด ฟังก์ชันใช้คำสั่ง `getch()` เพื่อรอการกดปุ่มจากผู้ใช้งานก่อนที่จะสิ้นสุดโปรแกรม ซึ่งเป็นเทคนิคที่ใช้เพื่อหยุดหน้าจอผลลัพธ์ไว้เพื่อให้ผู้ใช้งานได้เห็นข้อมูลก่อนที่โปรแกรมจะปิดตัว
- คำสั่ง `return(0)`: ฟังก์ชัน `main` ส่งค่ากลับเป็น 0 ซึ่งหมายถึงการสิ้นสุดการทำงานของโปรแกรมอย่างสมบูรณ์

ผลลัพธ์การใช้งานโปรแกรม NODE DIRECTORY METHOD

โปรแกรม NODE DIRECTORY METHOD ออกแบบมาเพื่อสร้างและแสดงกราฟโดยใช้วิธีการเก็บข้อมูลในรูปแบบ "Node Directory Method" ซึ่งช่วยให้ผู้ใช้งานสามารถเข้าใจโครงสร้างของกราฟได้อย่างชัดเจน โปรแกรมใช้การเก็บข้อมูลโหนดและเส้นเชื่อมในอาเรย์สองมิติ (2D Array) และแสดงผลในรูปแบบที่เข้าใจง่าย โดยเมื่อผู้รันโปรแกรม โปรแกรมจะเริ่มต้นโดยการแสดงข้อมูลของกราฟที่ประกอบด้วยโหนดและเส้นเชื่อมในรูปแบบต่าง ๆ ต่อไปนี้

1. การแสดงข้อมูลโหนด (DispHead): ฟังก์ชัน DispHead จะแสดงข้อมูลของโหนดทั้งหมดในกราฟ ซึ่งรวมถึงหมายเลขโหนด, ชื่อโหนด, ข้อมูลเสริม, จำนวนเส้นเชื่อมที่เชื่อมออกจากโหนด, และตำแหน่งเริ่มต้นในอาเรย์เส้นเชื่อม

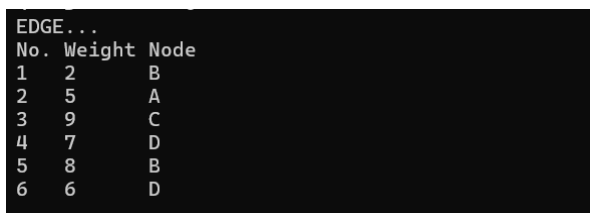


```

E:\ENGCE124\Coding 16 Node
GRAPH NODE DIRECTORY REPRESENTATION METHOD
=====
NODE...
No. Name Data Edge Pointer
1 A - 1 1
2 B - 3 2
3 C - 2 5
4 D - 0
  
```

ข้อมูลนี้แสดงว่าโหนด A มีเส้นเชื่อม 1 เส้นเริ่มที่ตำแหน่ง 1 ของอาเรย์เส้นเชื่อม, โหนด B มีเส้นเชื่อม 3 เส้นเริ่มที่ตำแหน่ง 2 เป็นต้น

2. การแสดงข้อมูลเส้นเชื่อม (DispEdge): ฟังก์ชัน DispEdge แสดงข้อมูลของเส้นเชื่อมในกราฟ ซึ่งรวมถึงหมายเลขเส้นเชื่อม, น้ำหนักของเส้นเชื่อม, และโหนดที่เชื่อมต่อ



```

EDGE...
No. Weight Node
1 2 B
2 5 A
3 9 C
4 7 D
5 8 B
6 6 D
  
```

ข้อมูลนี้แสดงให้เห็นว่าเส้นเชื่อมหมายเลข 1 มีน้ำหนัก 2 และเชื่อมกับโหนด B, เส้นเชื่อมหมายเลข 2 มีน้ำหนัก 5 และเชื่อมกับโหนด A เป็นต้น

3. การแสดงเซตของโหนด (DispSetOfVertex): ฟังก์ชัน DispSetOfVertex แสดงเซตของโหนดทั้งหมดที่มีอยู่ในกราฟในรูปแบบเซต

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

เซตนี้แสดงชื่อของโหนดทั้งหมดในกราฟ ทำให้เห็นว่าโหนดในกราฟคือ A, B, C, และ D

4. การแสดงเซตของเส้นเชื่อม (DispSetOfEdge): ฟังก์ชัน DispSetOfEdge แสดงเซตของเส้นเชื่อมในกราฟ พร้อมน้ำหนักของแต่ละเส้นเชื่อม

```
Set of Edge = {(AB)2, (BA)5, (BC)9, (BD)7, (CB)8, (CD)6, }
```

เซตนี้แสดงเส้นเชื่อมทั้งหมดในกราฟในรูปแบบ (Node1 Node2) Weight เช่น (A B)2 หมายถึงเส้นเชื่อมจาก โหนด A ไปยังโหนด B ที่มีน้ำหนัก 2

เมื่อโปรแกรมทำงานเสร็จสิ้น จะมีการแสดงผลลัพธ์ดังนี้

- ข้อมูลโหนด: การแสดงข้อมูลของแต่ละโหนดในกราฟในรูปแบบตารางซึ่งรวมถึงรายละเอียดต่าง ๆ ของโหนด
- ข้อมูลเส้นเชื่อม: การแสดงข้อมูลเส้นเชื่อมในกราฟในรูปแบบตารางพร้อมน้ำหนักของเส้นเชื่อมและโหนดที่เชื่อมต่อกัน
- เซตของโหนด: การแสดงเซตของโหนดทั้งหมดในกราฟ ซึ่งช่วยให้ผู้ใช้งานเห็นโหนดที่ประกอบเป็นกราฟทั้งหมด
- เซตของเส้นเชื่อม: การแสดงเซตของเส้นเชื่อมในกราฟพร้อมน้ำหนักของเส้นเชื่อมในรูปแบบที่เข้าใจง่าย

```
E:\ENGCE124\Coding 16 Node  X  +  ~
GRAPH NODE DIRECTORY REPRESENTATION METHOD
=====
NODE...
No. Name Data Edge Pointer
1  A  -    1    1
2  B  -    3    2
3  C  -    2    5
4  D  -    0
EDGE...
No. Weight Node
1  2    B
2  5    A
3  9    C
4  7    D
5  8    B
6  6    D

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

Set of Edge = {(AB)2, (BA)5, (BC)9, (BD)7, (CB)8, (CD)6, }

-----
Process exited after 2.669 seconds with return value 0
Press any key to continue . . . |
```

ในตอนท้ายของการทำงาน โปรแกรมจะรอให้ผู้ใช้งานกดปุ่มใด ๆ ผ่านคำสั่ง getch() ก่อนที่จะสิ้นสุดการทำงาน เพื่อให้ผู้ใช้งานมีโอกาสเห็นผลลัพธ์ทั้งหมดที่แสดงอยู่บนหน้าจอ การแสดงผลลัพธ์ที่เป็นระเบียบนี้ช่วยให้ผู้ใช้งานเข้าใจโครงสร้างของกราฟได้ง่ายขึ้น และสามารถใช้อ้างอิงข้อมูลนี้ในการวิเคราะห์หรือการดำเนินการเพิ่มเติมได้อย่างสะดวก

บรรณานุกรม

ChatGPT. (-). Graph Representation with Node Directory Method. สืบค้น 5 กันยายน 2567,
จาก <https://chatgpt.com/>