



รายงาน

เรื่อง โปรแกรม GRAPH STRUCTURE BY 2 DIMENSION ARRAY

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 2 DIMENSION ARRAY

จัดทำโดย

นายพงษ์พันธุ์ เลาหวงศ์

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 2 DIMENSION ARRAY รวมถึงอธิบายหลักการทำงานของโปรแกรม GRAPH STRUCTURE BY 2 DIMENSION ARRAY และอธิบายผลลัพธ์การใช้งานโปรแกรม GRAPH STRUCTURE BY 2 DIMENSION ARRAY

ผู้จัดทำรายงานหวังว่า รายงานฉบับนี้จะเป็นประโยชน์กับผู้สนใจ หรือนักศึกษาทุกท่านที่กำลังหาศึกษาในหัวข้อของโปรแกรม GRAPH STRUCTURE BY 2 DIMENSION ARRAY หากมีข้อเสนอแนะหรือข้อผิดพลาดประการใด ผู้จัดทำขอน้อมรับไว้ และขออภัยมา ณ ที่นี้

ผู้จัดทำ

นายพงษ์พันธุ์ เลาวพงศ์

วันที่ 05/09/2567

สารบัญ

	หน้า
คำนำ	ก
สารบัญ	๗
โค้ดของโปรแกรม GRAPH STRUCTURE BY 2 DIMENSION ARRAY พร้อมคำอธิบาย	1
หลักการทำงานของโปรแกรม GRAPH STRUCTURE BY 2 DIMENSION ARRAY	4
ผลลัพธ์การใช้งานโปรแกรม GRAPH STRUCTURE BY 2 DIMENSION ARRAY	10
บรรณานุกรม	12

โค้ดของโปรแกรม GRAPH STRUCTURE BY 2 DIMENSION ARRAY พร้อมคำอธิบาย

```
#include <stdio.h> // ใช้ printf

#include <conio.h> // ใช้ getch

#define MaxNode 6 // กำหนดจำนวนโหนดสูงสุดของกราฟ

int graph[MaxNode][MaxNode] = {

    {0,1,1,1,0,0},

    {1,0,1,0,1,0},

    {1,1,0,0,0,0},

    {1,0,0,0,1,1},

    {0,1,0,1,0,0},

    {0,0,0,1,0,0}

}; // ประกาศอาร์เรย์และเก็บข้อมูลกราฟ

char NodeName[MaxNode] = {'A','B','C','D','E','F'}; // เก็บชื่อโหนด

void DispArray2D() // แสดงค่าของอาร์เรย์ 2 มิติ

{

    int i,j; // i=แถว, j=คอลัมน์

    printf(" ");

    for (j=0;j<=MaxNode;j++) // แสดงชื่อคอลัมน์ของอาร์เรย์

        printf("%c ",NodeName[j]);

    printf("\n"); // ขึ้นบรรทัดใหม่

    for (i=0;i<MaxNode;i++) // ลูปแถว

    {
```

โค้ดของโปรแกรม GRAPH STRUCTURE BY 2 DIMENSION ARRAY พร้อมคำอธิบาย (ต่อ)

```

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

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

        printf("%d ",graph[i][j]); // แสดงค่าของการเชื่อมต่อ

    printf("\n");

}

}

void DispSetOfVertex() // แสดงชุดของจุดยอด

{

    int i;

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

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

    {

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

        if(i != MaxNode-1)

            printf(",");

    }

    printf("}\n");

}

void DispSetOfEdge() // แสดงชุดของขอบ (Edge)

{

    int i,j;

```

โค้ดของโปรแกรม GRAPH STRUCTURE BY 2 DIMENSION ARRAY พร้อมคำอธิบาย (ต่อ)

```

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

for (i=0;i<MaxNode;i++) // ลูปแถว

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

    {

        if(graph[i][j]==1)

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

    }

    printf("\n");

}

int main()

{

    printf("GRAPH (ADJACENCY MATRIX REPRESENTATION METHOD)\n");

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

    DispArray2D();

    DispSetOfVertex();

    DispSetOfEdge();

    getch();

    return(0);

} // จบ Main

```

หลักการการทำงานของโปรแกรม GRAPH STRUCTURE BY 2 DIMENSION ARRAY

การเขียนโปรแกรมเพื่อจัดการกับโครงสร้างข้อมูลแบบกราฟ (Graph) สามารถทำได้หลายวิธี โดยวิธีที่นิยมวิธีหนึ่งคือการใช้ Adjacency Matrix หรือ อาร์เรย์ 2 มิติ เพื่อแทนโครงสร้างของกราฟ ซึ่งโปรแกรม GRAPH STRUCTURE BY 2 DIMENSION ARRAY ถูกออกแบบมาให้ทำงานกับกราฟที่มี 6 โหนด โดยข้อมูลของเส้นเชื่อมระหว่างโหนดถูกจัดเก็บในอาร์เรย์ 2 มิติ `graph[MaxNode][MaxNode]` พร้อมกับฟังก์ชันที่ทำงานร่วมกันเพื่อแสดงผลข้อมูลของกราฟ ได้แก่ การแสดงโครงสร้างของกราฟในรูปแบบ Adjacency Matrix การแสดงเซตของจุดยอด (Set of Vertex) และการแสดงเซตของเส้นเชื่อม (Set of Edge)

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. การกำหนดจำนวนโหนดสูงสุดของกราฟ

```
#define MaxNode 6 // กำหนดจำนวนโหนดสูงสุดของกราฟ
```

บรรทัดนี้ใช้การกำหนดนิยาม (`#define`) เพื่อกำหนด ค่าคงที่ ที่ชื่อว่า `MaxNode` ซึ่งมีค่าเป็น 6 หมายความว่า กราฟนี้จะประกอบด้วยจำนวนโหนดสูงสุด 6 โหนด โหนดเหล่านี้ถูกเก็บในอาร์เรย์ 2 มิติและจะใช้ในโปรแกรมในส่วนต่าง ๆ ที่ต้องการทราบจำนวนโหนดสูงสุด

3. การประกาศและกำหนดค่าให้กับอาร์เรย์ 2 มิติสำหรับการเก็บข้อมูลกราฟ

```
int graph[MaxNode][MaxNode] = {
    {0,1,1,1,0,0},
    {1,0,1,0,1,0},
    {1,1,0,0,0,0},
    {1,0,0,0,1,1},
    {0,1,0,1,0,0},
    {0,0,0,1,0,0}
}; // ประกาศอาร์เรย์และเก็บข้อมูลกราฟ
```

ในส่วนนี้มีการประกาศอาร์เรย์ 2 มิติที่ชื่อว่า graph ขนาด 6x6 เพื่อใช้เก็บข้อมูลโครงสร้างของกราฟ โดย:

- ค่า 0 หมายถึงไม่มีเส้นเชื่อมระหว่างโหนดสองโหนดนั้น
- ค่า 1 หมายถึงมีเส้นเชื่อมระหว่างโหนดสองโหนดนั้น

ข้อมูลที่เก็บในอาร์เรย์นี้เรียกว่า Adjacency Matrix หรือ เมทริกซ์ความเชื่อมโยง ที่ใช้แทนกราฟ โดยในตัวอย่างนี้มีการกำหนดค่าตามลำดับ ซึ่งแต่ละบรรทัดจะอธิบายการเชื่อมต่อของโหนดหนึ่งไปยังโหนดอื่น ๆ เช่น

- โหนด A (แถวแรก) มีการเชื่อมต่อกับโหนด B, C, และ D (ค่า 1 ในคอลัมน์ที่ 2, 3, และ 4)
- โหนด B (แถวที่สอง) มีการเชื่อมต่อกับโหนด A, C, และ E
- โหนดอื่น ๆ ก็เช่นกัน โดยค่าของแต่ละบรรทัดแสดงความสัมพันธ์การเชื่อมต่อระหว่างโหนด

4. การเก็บชื่อโหนด

```
char NodeName[MaxNode] = {'A','B','C','D','E','F'}; // เก็บชื่อโหนด
```

โปรแกรมกำหนดอาร์เรย์ NodeName[] ที่ใช้เก็บชื่อของโหนดในกราฟ ซึ่งประกอบด้วยโหนด A, B, C, D, E, และ F โดยอาร์เรย์นี้ใช้เพื่อแสดงชื่อของโหนดในระหว่างการแสดงผล เช่น ในการแสดงผลตารางของกราฟ (Adjacency Matrix) หรือการแสดงเซตของจุดยอด (Vertices) และเส้นเชื่อม (Edges)

5. ฟังก์ชัน DispArray2D การแสดงผลโครงสร้างกราฟในรูปแบบอาร์เรย์ 2 มิติ

```
void DispArray2D() // แสดงค่าของอาร์เรย์ 2 มิติ
{
    int i,j; // i=แถว, j=คอลัมน์

    printf(" ");

    for (j=0;j<=MaxNode;j++) // แสดงชื่อคอลัมน์ของอาร์เรย์

        printf("%c ",NodeName[j]);

    printf("\n"); // ขึ้นบรรทัดใหม่

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

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

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

            printf("%d ",graph[i][j]); // แสดงค่าของการเชื่อมต่อ

        printf("\n");

    }

}
```

ฟังก์ชันนี้ทำหน้าที่แสดงโครงสร้างกราฟในรูปแบบ Adjacency Matrix โดยใช้ลูปสองชั้นในการเดินทางผ่านข้อมูลในอาร์เรย์ 2 มิติ และแสดงผลในรูปแบบตาราง เพื่อให้ผู้ใช้เข้าใจการเชื่อมต่อระหว่างโหนดได้ง่ายขึ้น

รายละเอียดการทำงาน

- ฟังก์ชันเริ่มต้นด้วยการประกาศตัวแปร i และ j ซึ่งจะใช้เป็นตัวนับในลูปสำหรับเดินทางในอาร์เรย์
- แถวบนสุดของตารางจะแสดงชื่อโหนด (A, B, C, D, E, F) โดยใช้ลูปที่วิ่งจาก 0 ถึง MaxNode และใช้ชื่อโหนดจากอาร์เรย์ NodeName[]

- จากนั้นลูปชั้นนอกจะเริ่มที่แต่ละแถว ซึ่งแต่ละแถวจะเริ่มต้นด้วยการแสดงชื่อโหนด (เช่น A, B, C) ตามด้วยค่าของแต่ละคอลัมน์ที่สอดคล้องกับเส้นเชื่อม
- ในลูปชั้นใน จะวิ่งผ่านแต่ละคอลัมน์และแสดงค่าที่อยู่ในอาร์เรย์ `graph[i][j]` ซึ่งค่าเหล่านี้จะเป็น 1 หากมีเส้นเชื่อมระหว่างโหนด และเป็น 0 หากไม่มี

6. ฟังก์ชัน `DispSetOfVertex` การแสดงเซตของจุดยอด (Set of Vertex)

```
void DispSetOfVertex() // แสดงชุดของจุดยอด
{
    int i;

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

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

        if(i != MaxNode-1)

            printf(",");
    }

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

ฟังก์ชันนี้ทำหน้าที่แสดงเซตของจุดยอดในกราฟ โดยจะลูปผ่านชื่อโหนดทั้งหมดและแสดงผลในรูปแบบ {A, B, C, D, E, F} ซึ่งช่วยให้ผู้ใช้เห็นว่ากราฟมีโหนดใดบ้างที่อยู่ในโครงสร้าง

รายละเอียดการทำงาน

- ฟังก์ชันเริ่มต้นด้วยการประกาศตัวแปร `i` ที่จะใช้เป็นตัวนับในลูป
- โปรแกรมจะแสดงข้อความ "Set of Vertex = {" เพื่อเริ่มต้นการแสดงผล
- จากนั้นลูปจะวิ่งจาก 0 ถึง `MaxNode` โดยในแต่ละรอบลูปจะดึงค่าในอาร์เรย์ `NodeName[i]` และแสดงชื่อโหนดนั้นออกมา

- ถ้าโหนดนั้นไม่ใช่ตัวสุดท้าย (ตรวจสอบด้วยเงื่อนไข $\text{if}(i \neq \text{MaxNode}-1)$), โปรแกรมจะแสดงเครื่องหมายจุลภาค (,) หลังชื่อโหนด
- เมื่อถึงโหนดสุดท้าย โปรแกรมจะแสดงเครื่องหมายปิด } เพื่อปิดเซต

9. ฟังก์ชัน DispSetOfEdge การแสดงเซตของเส้นเชื่อม (Set of Edge)

```
void DispSetOfEdge() // แสดงชุดของขอบ (Edge)
{
    int i,j;

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

    for (i=0;i<MaxNode;i++) // ลูปแถว
        for (j=0;j<MaxNode;j++) // ลูปคอลัมน์
        {
            if(graph[i][j]==1)
                printf("(%c,%c)",NodeName[i],NodeName[j]); // แสดงแต่ละขอบ (Edge)
        }

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

ฟังก์ชันนี้ทำหน้าที่แสดงเซตของเส้นเชื่อม (Edges) ระหว่างโหนดต่าง ๆ โดยจะลูปผ่านข้อมูลในอาร์เรย์ 2 มิติ `graph[][]` และแสดงผลคู่ของโหนดที่มีเส้นเชื่อมกัน

รายละเอียดการทำงาน

- ฟังก์ชันเริ่มต้นด้วยการประกาศตัวแปร `i` และ `j` ที่จะใช้เป็นตัวนับในลูปสองชั้น
- โปรแกรมจะแสดงข้อความ "Set of Edge = {" เพื่อเริ่มต้นการแสดงผลของเส้นเชื่อม
- ลูปชั้นนอกจะวิ่งจาก 0 ถึง `MaxNode` สำหรับการเข้าถึงแถวแต่ละแถว และลูปชั้นในจะวิ่งจาก 0 ถึง `MaxNode` สำหรับคอลัมน์

- ในแต่ละรอบของลูปชั้นใน โปรแกรมจะตรวจสอบว่าค่าของ `graph[i][j]` เท่ากับ 1 หรือไม่ ถ้าใช่ หมายความว่าโหนดที่ตำแหน่ง `i` และ `j` มีเส้นเชื่อมกัน ดังนั้นโปรแกรมจะแสดงคู่ของโหนดในรูปแบบ (โหนดต้นทาง, โหนดปลายทาง)
- เมื่อแสดงผลเสร็จสิ้น โปรแกรมจะแสดงเครื่องหมายปิด } เพื่อปิดเซตของเส้นเชื่อม

10. ฟังก์ชัน main ฟังก์ชันหลักของโปรแกรม

```
int main()
{
    printf("GRAPH (ADJACENCY MATRIX REPRESENTATION METHOD)\n");

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

    DispArray2D();

    DispSetOfVertex();

    DispSetOfEdge();

    getch();

    return(0);

} // จบ Main
```

ฟังก์ชัน `main()` ทำหน้าที่เป็นจุดเริ่มต้นของโปรแกรม โดยเรียกใช้ฟังก์ชันต่าง ๆ เพื่อแสดงผลลัพธ์ที่ต้องการ

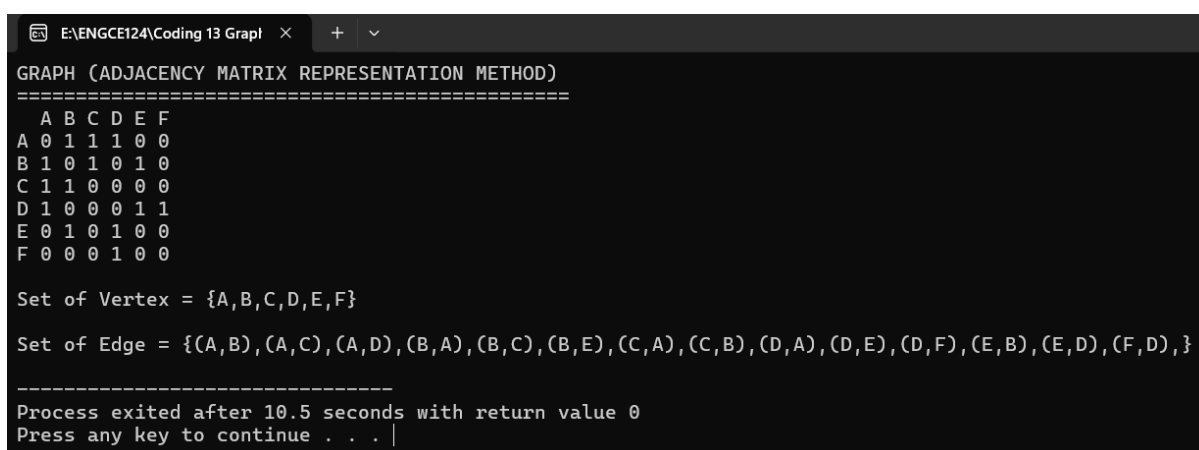
รายละเอียดการทำงาน

- ฟังก์ชันเริ่มต้นด้วยการแสดงข้อความหัวข้อ "GRAPH (ADJACENCY MATRIX REPRESENTATION METHOD)" เพื่อบอกว่าโปรแกรมนี้เกี่ยวกับการแสดงกราฟด้วยวิธี Adjacency Matrix
- จากนั้นเรียกใช้ฟังก์ชัน `DispArray2D()` เพื่อแสดงตารางของโครงสร้างกราฟ
- ต่อด้วยการเรียกใช้ฟังก์ชัน `DispSetOfVertex()` เพื่อแสดงเซตของจุดยอด
- สุดท้ายเรียกใช้ฟังก์ชัน `DispSetOfEdge()` เพื่อแสดงเซตของเส้นเชื่อม
- โปรแกรมจะหยุดรอการกดปุ่มใด ๆ ด้วยฟังก์ชัน `getch()` ก่อนจะสิ้นสุดการทำงาน

ผลลัพธ์การใช้งานโปรแกรม GRAPH STRUCTURE BY 2 DIMENSION ARRAY

โปรแกรมตัวอย่างนี้เป็นโปรแกรมที่ออกแบบมาเพื่อแสดงข้อมูลของ กราฟ (Graph) ในรูปแบบ Adjacency Matrix ซึ่งเป็นหนึ่งในวิธีที่นิยมใช้ในการแทนโครงสร้างของกราฟ โปรแกรมจะทำการแสดงข้อมูลของกราฟในรูปแบบตารางการเชื่อมต่อ และแสดงเซตของจุดยอด (Vertices) รวมถึงเส้นเชื่อม (Edges) โดยเมื่อรันโปรแกรม ผู้ใช้งานจะได้รับผลลัพธ์ที่แสดง 3 ส่วนหลัก ได้แก่

- แสดงโครงสร้างกราฟในรูปแบบตาราง (Adjacency Matrix)
- แสดงเซตของจุดยอด (Set of Vertex)
- แสดงเซตของเส้นเชื่อม (Set of Edge)



```

E:\ENGCE124\Coding 13 Graph >
GRAPH (ADJACENCY MATRIX REPRESENTATION METHOD)
=====
  A B C D E F
A 0 1 1 1 0 0
B 1 0 1 0 1 0
C 1 1 0 0 0 0
D 1 0 0 0 1 1
E 0 1 0 1 0 0
F 0 0 0 1 0 0

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

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

-----
Process exited after 10.5 seconds with return value 0
Press any key to continue . . .
  
```

1. แสดงโครงสร้างกราฟในรูปแบบตาราง (Adjacency Matrix)

หลังจากผู้ใช้งานโปรแกรม ผลลัพธ์แรกที่แสดงออกมาก็คือ โครงสร้างของกราฟในรูปแบบของตาราง หรือที่เรียกว่า Adjacency Matrix ตารางนี้จะแสดงความสัมพันธ์ระหว่างโหนด (Node) ต่าง ๆ ว่าโหนดใดมีการเชื่อมต่อกับโหนดใดบ้าง โดยตารางนี้มีขนาด 6x6 ซึ่งสอดคล้องกับจำนวนโหนดในกราฟที่โปรแกรมกำหนดไว้ รูปแบบของตารางจะแสดงชื่อของโหนดในแถวและคอลัมน์ พร้อมกับค่า 0 หรือ 1 ในแต่ละเซลล์ของตาราง

- ค่า 1 หมายถึง มีการเชื่อมต่อระหว่างโหนดสองโหนดนั้น
- ค่า 0 หมายถึง ไม่มีการเชื่อมต่อระหว่างโหนดสองโหนดนั้น
- ในตารางนี้ เห็นว่าโหนด A เชื่อมต่อกับโหนด B, C, และ D เพราะค่าในเซลล์ที่เกี่ยวข้องมีค่าเป็น 1
- โหนด B เชื่อมต่อกับโหนด A, C, และ E
- โหนดอื่น ๆ ก็สามารถอ่านความเชื่อมโยงได้เช่นเดียวกัน

```

E:\ENGCE124\Coding 13 Grap
GRAPH (ADJACENCY MATRIX REPRESENTATION METHOD)
=====
  A B C D E F
A 0 1 1 1 0 0
B 1 0 1 0 1 0
C 1 1 0 0 0 0
D 1 0 0 0 1 1
E 0 1 0 1 0 0
F 0 0 0 1 0 0

```

2. แสดงเซตของจุดยอด (Set of Vertex)

หลังจากแสดงตารางการเชื่อมต่อของโหนดแล้ว โปรแกรมจะทำการแสดงเซตของจุดยอด หรือ Set of Vertex ซึ่งแสดงชื่อของโหนดทั้งหมดที่มีอยู่ในกราฟ โดยจะแสดงเป็นลำดับของตัวอักษรในรูปแบบเซต

- เซตนี้แสดงให้เห็นว่าในกราฟประกอบด้วยโหนด A, B, C, D, E, และ F ซึ่งเป็นโหนดทั้งหมดที่ใช้ในการคำนวณและแสดงผล

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

3. แสดงเซตของเส้นเชื่อม (Set of Edge)

สุดท้าย โปรแกรมจะทำการแสดง Set of Edge ซึ่งคือเซตของคู่โหนดที่มีการเชื่อมต่อกันในกราฟ ข้อมูลนี้แสดงความสัมพันธ์ระหว่างโหนดสองโหนดในรูปแบบของคู่ (โหนดต้นทาง, โหนดปลายทาง) โดยโปรแกรมจะลูปผ่านอาร์เรย์ 2 มิติ graph[] เพื่อตรวจสอบว่ามีการเชื่อมต่อระหว่างโหนดใดบ้าง

- คู่ของโหนดที่แสดงในเซตนี้คือโหนดที่มีการเชื่อมต่อกัน เช่น (A,B) หมายความว่าโหนด A และ B มีการเชื่อมต่อกัน
- สำหรับกราฟในตัวอย่างนี้ ซึ่งเป็น กราฟแบบไม่กำกับทิศทาง (Undirected Graph) การเชื่อมต่อระหว่างโหนดจะแสดงทั้งสองทิศทาง เช่น ถ้าแสดง (A,B) ก็แสดง (B,A) ด้วย ซึ่งหมายถึงการเชื่อมต่อระหว่างสองโหนดนั้นเป็นไปได้ทั้งสองทาง

```

Set of Edge = {(A,B),(A,C),(A,D),(B,A),(B,C),(B,E),(C,A),(C,B),(D,A),(D,E),(D,F),(E,B),(E,D),(F,D),}
-----
Process exited after 10.5 seconds with return value 0
Press any key to continue . . .

```

บรรณานุกรม

ChatGPT. (-). Exploring Graph Structures through 2D Array Representation. สืบค้น 5 กันยายน 2567,
จาก <https://chatgpt.com/>