รายงาน

เรื่อง โปรแกรม GRAPH STRUCTURE BY ADJACENCY 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 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 LIST รวมถึงอธิบายหลักการทำงานของโปรแกรม GRAPH STRUCTURE BY ADJACENCY LIST และอธิบายผลลัพธ์การใช้งานโปรแกรม GRAPH STRUCTURE BY ADJACENCY LIST

ผู้จัดทำรายงานหวังว่า รายงานฉบับนี้จะเป็นประโยชน์กับผู้ที่สนใจ หรือนักศึกษาทุกท่าน ที่กำลังหาศึกษาในหัวข้อของโปรแกรม GRAPH STRUCTURE BY ADJACENCY LIST หากมีข้อแนะนำหรือข้อผิดพลาดประการใด ผู้จัดทำขอน้อมรับไว้ และขออภัยมา ณ ที่นี้

ผู้จัดทำ

นายพงษ์พันธุ์ เลาวพงศ์

วันที่ 05/09/2567

สารบัญ

	หน้า
คำนำ	ก
สารบัญ	ข
โค้ดของโปรแกรม GRAPH STRUCTURE BY ADJACENCY LIST พร้อมคำอธิบาย	1
หลักการทำงานของโปรแกรม GRAPH STRUCTURE BY ADJACENCY LIST	6
ผลลัพธ์การใช้งานโปรแกรม GRAPH STRUCTURE BY ADJACENCY LIST	13
บรรณานุกรม	15

โค้ดของโปรแกรม GRAPH STRUCTURE BY ADJACENCY LIST พร้อมคำอธิบาย

```
#include <stdio.h> //ใช้ printf

#include <conio.h> //ใช้ getch

#include <stdlib.h> //ใช้ malloc

#define MaxNode 4 //กำหนดจำนวนโหนดสูงสุดของกราฟ

char NodeName[4] = {'A','B','C','D'};

int graph[MaxNode][MaxNode] = {

    {0,1,1,1},

    {1,0,1,1},

    {1,1,0,0},

    {1,1,0,0},

}; //ประกาศอาร์เรย์และเก็บข้อมูลของกราฟ

struct Node //ประกาศโครงสร้างของแต่ละโหนด

{

    char info;

    struct Node *next;

}; struct Node *Start[MaxNode], *p; //ประกาศพอยน์เตอร์ที่ใช้

Node *Allocate() //จัดสรรโหนด 1 โหนดจากหน่วยความจำ

{

    struct Node *temp;

    temp=(Node*)malloc(sizeof(Node)); //จัดสรรโหนดตามขนาดที่ประกาศ

    return(temp);
```

โค้ดของโปรแกรม GRAPH STRUCTURE BY ADJACENCY LIST พร้อมคำอธิบาย (ต่อ)

```
}
void CreateHead() //สร้างโหนดหัว
{
    int i;
    for (i=0;i<MaxNode;i++) //นับจำนวนโหนดตามจำนวนสูงสุด
    {
        p=Allocate();
        p->info=NodeName[i]; //กำหนดข้อมูลให้เท่ากับชื่อโหนด
        p->next=NULL; //กำหนดค่า NEXT ให้เป็น NULL
        Start[i]=p; //กำหนด Start ของแต่ละโหนดให้เท่ากับตำแหน่งของโหนดแรก
    }
}
void TransferToAdjacent() //ถ่ายโอนข้อมูลอาร์เรย์ไปยังลิสต์ของกราฟ
{
    int i,j;
    struct Node *Rear; //ตัวนับและพอยน์เตอร์ที่ชี้ไปยังโหนดสุดท้าย
    for (i=0;i<MaxNode;i++) //วนซ้ำแถว
    {
        Rear=Start[i]; //พอยน์เตอร์ Rear เริ่มต้นที่นี่
        for(j=0;j<MaxNode;j++) //วนซ้ำคอลัมน์
        {
```

โค้ดของโปรแกรม GRAPH STRUCTURE BY ADJACENCY LIST พร้อมคำอธิบาย (ต่อ)

```
            if (graph[i][j]==1) //ถ้ามีเส้นทาง
            {
                p=Allocate(); //สร้างโหนดใหม่
                p->info=NodeName[j]; //กำหนดข้อมูลให้เท่ากับ NodeName[j]
                p->next=NULL; //กำหนด NEXT ให้เป็น NULL
                Rear->next=p; //ชี้ Rear ไปยังโหนดใหม่
                Rear=p; //เลื่อนพอยน์เตอร์ Rear ไปยังโหนดถัดไป
            }
        }
    }
}
void DispSetOfVertex() //แสดงเซตของเวอร์เท็กซ์
{
    int i;
    printf("\nSet of Vertex = {");
    for (i=0;i<MaxNode;i++) //นับเฉพาะโหนดเริ่มต้น
    {
        printf("%c",Start[i]->info); //แสดงชื่อของแต่ละโหนด
        if(i != MaxNode-1)
            printf(",");
    }
```

โค้ดของโปรแกรม GRAPH STRUCTURE BY ADJACENCY LIST พร้อมคำอธิบาย (ต่อ)

```
    printf("}\n");
}
void DispSetOfEdge() //แสดงเซตของขอบ (Edge)
{
    int i;
    struct Node *Temp;
    printf("\nSet of Edge = {");
    for (i=0;i<MaxNode;i++) //วนซ้ำแถว
    {
        Temp=Start[i]; //ให้พอยน์เตอร์ Temp เริ่มต้นที่นี่
        Temp=Temp->next; //เลื่อนพอยน์เตอร์ Temp ไปยังโหนดถัดไป
        while (Temp != NULL) //ชี้ไปยังโหนดที่ 2
        {
            printf("(%c,%c),",Start[i]->info,Temp->info); //แสดงแต่ละขอบ
            Temp=Temp->next; //เลื่อนพอยน์เตอร์ Temp ไปยังโหนดถัดไป
        }
    }
    printf("}\n");
}
int main()
{
```

โค้ดของโปรแกรม GRAPH STRUCTURE BY ADJACENCY LIST พร้อมคำอธิบาย (ต่อ)

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

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

    CreateHead();

    TransferToAdjacent();

    DispSetOfVertex();

    DispSetOfEdge();

    getch();

    return(0);

} //สิ้นสุดฟังก์ชัน main
```

หลักการทำงานของโปรแกรม GRAPH STRUCTURE BY ADJACENCY LIST

โปรแกรม GRAPH STRUCTURE BY ADJACENCY LIST เป็นการสร้างโครงสร้างกราฟแบบ Adjacency List โดยสามารถแสดงผลของกราฟเป็นชุดของจุดยอด (Vertex) และชุดของเส้นเชื่อม (Edge) โดยโปรแกรมนี้รองรับทั้งกราฟแบบมีทิศทาง (Directed Graph) และแบบไม่มีทิศทาง (Undirected Graph)

1. การประกาศใช้งานไลบรารี

```
#include <stdio.h> //ใช้ printf

#include <conio.h> //ใช้ getch

#include <stdlib.h> //ใช้ malloc
```

ในส่วนนี้โปรแกรมจะทำการ include ไลบรารี่มาตรฐานสามตัว ได้แก่

- #include <stdio.h> : รวมไลบรารี <stdio.h> ซึ่งมีฟังก์ชันมาตรฐานสำหรับการรับและแสดงผลข้อมูล เช่น printf() ที่ใช้ในการพิมพ์ข้อความไปยังหน้าจอ
- #include <conio.h> : รวมไลบรารี <conio.h> ซึ่งมีฟังก์ชันสำหรับการจัดการการป้อนข้อมูลจากคีย์บอร์ด เช่น getch() ที่ใช้ในการรอให้ผู้ใช้กดปุ่ม
- #include <stdlib.h> : รวมไลบรารี <stdlib.h> ซึ่งมีฟังก์ชันสำหรับการจัดการหน่วยความจำและการแปลงข้อมูล เช่น malloc() ที่ใช้ในการจัดสรรหน่วยความจำแบบไดนามิก

2. การกำหนดจำนวนโหนดสูงสุดของกราฟ

```
#define MaxNode 4 //กำหนดจำนวนโหนดสูงสุดของกราฟ
```

บรรทัดนี้ใช้การกำหนดนิยาม (#define) เพื่อกำหนดค่าคงที่ MaxNode เป็น 4 ซึ่งหมายถึงจำนวนสูงสุดของโหนด (nodes) ที่สามารถใช้ในกราฟได้ ในกรณีนี้มี 4 โหนด ซึ่งหมายถึงกราฟนี้จะมีจุดยอด 4 จุด โหนดเหล่านี้ถูกเก็บในอาร์เรย์ 2 มิติและจะใช้ในโปรแกรมในส่วนต่าง ๆ ที่ต้องการทราบจำนวนโหนดสูงสุด

3. การประกาศและกำหนดค่าให้กับอาร์เรย์ 2 มิติสำหรับการเก็บข้อมูลกราฟ

```
char NodeName[4] = {'A','B','C','D'};

int graph[MaxNode][MaxNode] = {

    {0,1,1,1},

    {1,0,1,1},

    {1,1,0,0},

    {1,1,0,0},

}; //ประกาศอาร์เรย์และเก็บข้อมูลของกราฟ
```

ในส่วนนี้จะประกาศอาร์เรย์ NodeName ขนาด 4 ของชนิดข้อมูล char เพื่อเก็บชื่อของจุดยอดในกราฟ ในที่นี้จุดยอดจะถูกระบุด้วยตัวอักษร 'A', 'B', 'C', และ 'D' จากนั้นจะประกาศอาร์เรย์สองมิติ graph ขนาด MaxNode x MaxNode (ในกรณีนี้คือ 4 x 4) เพื่อเก็บข้อมูลการเชื่อมต่อของกราฟ โดยค่าในอาร์เรย์จะเป็น 0 หรือ 1 ซึ่งบ่งชี้ว่าแต่ละจุดยอดมีการเชื่อมต่อกันหรือไม่ (1 หมายถึงมีการเชื่อมต่อ, 0 หมายถึงไม่มีการเชื่อมต่อ)

4. การประกาศโครงสร้างข้อมูลของโหนด และการประกาศพอยน์เตอร์ที่ใช้

```
struct Node //ประกาศโครงสร้างของแต่ละโหนด

{

    char info;

    struct Node *next;

}; struct Node *Start[MaxNode], *p; //ประกาศพอยน์เตอร์ที่ใช้
```

ในส่วนนี้จะประกาศโครงสร้างข้อมูล Node ซึ่งจะใช้ในการเก็บข้อมูลของแต่ละโหนดในลิสต์เชื่อมโยง (Linked List) ของกราฟ โดยมีตัวแปร info ใช้เก็บข้อมูลที่เกี่ยวข้องกับโหนด เช่น ชื่อของจุดยอด และตัวแปร next เป็นพอยน์เตอร์ที่ชี้ไปยังโหนดถัดไปในลิสต์เชื่อมโยง รวมถึงมีการประกาศอาร์เรย์ของพอยน์เตอร์ Start ขนาด MaxNode (ในที่นี้คือ 4) ซึ่งจะใช้เก็บที่อยู่ของโหนดเริ่มต้น (Head Node) ของลิสต์เชื่อมโยงสำหรับแต่ละจุดยอดในกราฟ รวมถึงประกาศพอยน์เตอร์ p ซึ่งจะใช้เป็นตัวช่วยในการสร้างและจัดการโหนดใหม่ในลิสต์เชื่อมโยง

5. ฟังก์ชัน Allocate สำหรับจัดสรรหน่วยความจำ

```
Node *Allocate() //จัดสรรโหนด 1 โหนดจากหน่วยความจำ
{
    struct Node *temp;
    temp=(Node*)malloc(sizeof(Node)); //จัดสรรโหนดตามขนาดที่ประกาศ
    return(temp);
}
```

ในส่วนของฟังก์ชัน Allocate() ทำหน้าที่จัดสรรหน่วยความจำสำหรับโหนดใหม่ โดยใช้คำสั่ง malloc ซึ่งเป็นฟังก์ชันใน C สำหรับจัดสรรหน่วยความจำแบบไดนามิก เมื่อโปรแกรมต้องการสร้างโหนดใหม่ ฟังก์ชันนี้จะถูกเรียกเพื่อจัดสรรหน่วยความจำสำหรับโหนดใหม่ และคืนค่าพอยน์เตอร์ไปยังโหนดนั้น

6. ฟังก์ชัน CreateHead สำหรับการสร้างโหนดเริ่มต้น

```
void CreateHead() //สร้างโหนดหัว
{
    int i;
    for (i=0;i<MaxNode;i++) //นับจำนวนโหนดตามจำนวนสูงสุด
    {
        p=Allocate();
        p->info=NodeName[i]; //กำหนดข้อมูลให้เท่ากับชื่อโหนด
        p->next=NULL; //กำหนดค่า NEXT ให้เป็น NULL
        Start[i]=p; //กำหนด Start ของแต่ละโหนดให้เท่ากับตำแหน่งของโหนดแรก
    }
}
```

ในส่วนของฟังก์ชัน CreateHead ทำหน้าที่สร้างโหนดเริ่มต้น (Head Node) ของกราฟ โดยวนลูปตามจำนวนสูงสุดของโหนดที่กำหนดไว้ในโปรแกรม (MaxNode) สำหรับแต่ละโหนด โดยมีขั้นตอนการสร้างคือ

- เรียกใช้ฟังก์ชัน Allocate() เพื่อสร้างโหนดใหม่
- กำหนดให้ตัวแปร info ในโหนดนั้นเป็นชื่อของจุดยอด เช่น 'A', 'B', 'C', 'D'
- กำหนดค่า next เป็น NULL เพื่อแสดงว่าโหนดนี้เป็นโหนดสุดท้ายของลิสต์
- จากนั้นเก็บโหนดที่สร้างไว้ในอาร์เรย์ Start[i] ซึ่งจะชี้ไปยังโหนดแรกของแต่ละจุดยอด

7. ฟังก์ชัน TransferToAdjacent สำหรับถ่ายโอนข้อมูลจากอาร์เรย์

```
void TransferToAdjacent() //ถ่ายโอนข้อมูลอาร์เรย์ไปยังลิสต์ของกราฟ
{
    int i,j;
    struct Node *Rear; //ตัวนับและพอยน์เตอร์ที่ชี้ไปยังโหนดสุดท้าย
    for (i=0;i<MaxNode;i++) //วนซ้ำแถว
    {
        Rear=Start[i]; //พอยน์เตอร์ Rear เริ่มต้นที่นี่
        for(j=0;j<MaxNode;j++) //วนซ้ำคอลัมน์
        {
            if (graph[i][j]==1) //ถ้ามีเส้นทาง
            {
                p=Allocate(); //สร้างโหนดใหม่
                p->info=NodeName[j]; //กำหนดข้อมูลให้เท่ากับ NodeName[j]
                p->next=NULL; //กำหนด NEXT ให้เป็น NULL
                Rear->next=p; //ชี้ Rear ไปยังโหนดใหม่
                Rear=p; //เลื่อนพอยน์เตอร์ Rear ไปยังโหนดถัดไป
```

7. ฟังก์ชัน TransferToAdjacent สำหรับถ่ายโอนข้อมูลจากอาร์เรย์ (ต่อ)

```
            }
        }
    }
}
```

ในส่วนของฟังก์ชัน TransferToAdjacent ทำหน้าที่ถ่ายโอนข้อมูลจากอาร์เรย์ graph ซึ่งเป็นอาร์เรย์แสดงความเชื่อมโยงระหว่างจุดยอด ไปยัง Adjacency List โดยจะวนลูปผ่านทุกจุดยอด (แถว) และตรวจสอบทุกจุดยอดอื่น ๆ (คอลัมน์) หากพบว่าในอาร์เรย์ graph[i][j] มีค่าเป็น 1 (แสดงว่ามีเส้นทางเชื่อมระหว่างจุดยอด i และ j) จะสร้างโหนดใหม่แล้วเพิ่มเข้าไปในลิสต์ของจุดยอด i

8. ฟังก์ชัน DispSetOfVertex สำหรับแสดงชุดของจุดยอดของกราฟ

```
void DispSetOfVertex() //แสดงเซตของเวอร์เท็กซ์
{
    int i;
    printf("\nSet of Vertex = {");
    for (i=0;i<MaxNode;i++) //นับเฉพาะโหนดเริ่มต้น
    {
        printf("%c",Start[i]->info); //แสดงชื่อของแต่ละโหนด
        if(i != MaxNode-1)
            printf(",");
    }
    printf("}\n");
}
```

ในส่วนของฟังก์ชัน DispSetOfVertex ทำหน้าที่แสดงชุดของจุดยอด (Vertex) ที่อยู่ในกราฟ โดยจะแสดงชื่อของแต่ละจุดยอดที่เก็บไว้ในตัวแปร info ของโหนดเริ่มต้น (Head Node) ซึ่งเก็บอยู่ในอาร์เรย์ Start[] โปรแกรมจะแสดงผลในรูปแบบ {A,B,C,D} ตามลำดับที่สร้างไว้

9. ฟังก์ชัน DispSetOfEdge สำหรับแสดงชุดของเส้นเชื่อมของกราฟ

```
void DispSetOfEdge() //แสดงเซตของขอบ (Edge)
{
    int i;
    struct Node *Temp;
    printf("\nSet of Edge = {");
    for (i=0;i<MaxNode;i++) //วนซ้ำแถว
    {
        Temp=Start[i]; //ให้พอยน์เตอร์ Temp เริ่มต้นที่นี่
        Temp=Temp->next; //เลื่อนพอยน์เตอร์ Temp ไปยังโหนดถัดไป
        while (Temp != NULL) //ชี้ไปยังโหนดที่ 2
        {
            printf("(%c,%c),",Start[i]->info,Temp->info); //แสดงแต่ละขอบ
            Temp=Temp->next; //เลื่อนพอยน์เตอร์ Temp ไปยังโหนดถัดไป
        }
    }
    printf("}\n");
}
```

ในส่วนของฟังก์ชัน DispSetOfEdge ทำหน้าที่แสดงชุดของเส้นเชื่อม (Edge) ที่เชื่อมต่อระหว่างจุดยอดในกราฟ โดยจะวนลูปผ่านจุดยอดทุกจุดในอาร์เรย์ Start[] และเริ่มต้นจากโหนดถัดไปของจุดยอดแต่ละจุด

(เนื่องจากโหนดแรกคือ Head Node) จากนั้นจะแสดงคู่จุดยอดที่เชื่อมต่อกันในรูปแบบ (A,B) โปรแกรมจะแสดงทุกคู่ของจุดยอดที่มีเส้นเชื่อมอยู่

10. ฟังก์ชัน main ฟังก์ชันหลักของโปรแกรม

```
int main()
{
    printf("GRAPH (ADJACENCY LIST REPRESENTATION METHOD)\n");
    printf("=============================================\n");
    CreateHead();
    TransferToAdjacent();
    DispSetOfVertex();
    DispSetOfEdge();
    getch();
    return(0);
} //สิ้นสุดฟังก์ชัน main
```

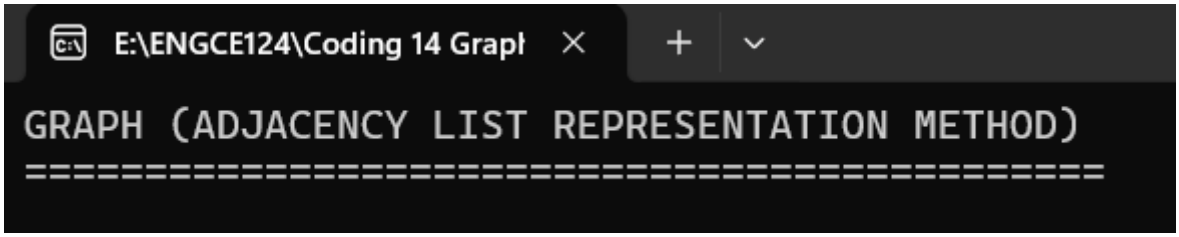
ในส่วนของฟังก์ชัน main เป็นฟังก์ชันหลักของโปรแกรม โดยเริ่มต้นจากการพิมพ์หัวเรื่องโปรแกรม จากนั้นเรียกใช้ฟังก์ชันตามลำดับ

- CreateHead() เพื่อสร้างโหนดเริ่มต้น
- TransferToAdjacent() เพื่อถ่ายโอนข้อมูลจากอาร์เรย์ไปยังลิสต์
- DispSetOfVertex() เพื่อแสดงชุดของจุดยอด
- DispSetOfEdge() เพื่อแสดงชุดของเส้นเชื่อม ฟังก์ชัน getch() ใช้ในการรอให้ผู้ใช้กดปุ่มก่อนจบโปรแกรม

ผลลัพธ์การใช้งานโปรแกรม GRAPH STRUCTURE BY ADJACENCY LIST

โปรแกรมนี้ถูกออกแบบมาเพื่อสร้างและจัดการโครงสร้างกราฟโดยใช้วิธีการ Adjacency List ซึ่งจะช่วยให้สามารถแสดงชุดของจุดยอด (Vertex) และชุดของเส้นเชื่อม (Edge) ในกราฟได้อย่างชัดเจน โดยมีขั้นตอนการใช้งานโปรแกรม

1. การเริ่มต้นโปรแกรม เมื่อโปรแกรมเริ่มทำงาน จะมีการพิมพ์ข้อความหัวเรื่องลงในหน้าจอคอนโซลเพื่อแจ้งให้ผู้ใช้ทราบว่าโปรแกรมกำลังทำงานและแสดงประเภทของการแสดงผลกราฟที่ใช้



2. การสร้างโหนดเริ่มต้น (Head Nodes) ฟังก์ชัน CreateHead() จะถูกเรียกใช้เพื่อสร้างโหนดเริ่มต้นสำหรับกราฟ โดยการสร้างโหนดใหม่สำหรับแต่ละจุดยอดในกราฟ ข้อมูลที่เกี่ยวข้องกับแต่ละโหนดคือชื่อของจุดยอด (เช่น 'A', 'B', 'C', 'D') และการเชื่อมโยงระหว่างโหนดจะถูกตั้งค่าเป็น NULL เพื่อเริ่มต้น

3. การถ่ายโอนข้อมูลไปยัง Adjacency List ฟังก์ชัน TransferToAdjacent() จะถูกเรียกใช้เพื่อถ่ายโอนข้อมูลจากอาร์เรย์ graph ไปยังลิสต์เชื่อมโยง (Adjacency List) ของกราฟ

- โปรแกรมจะวนลูปผ่านแต่ละจุดยอด (แถวของอาร์เรย์ graph) และตรวจสอบการเชื่อมโยงกับจุดยอดอื่น ๆ (คอลัมน์ของอาร์เรย์ graph)
- หากพบการเชื่อมโยง (ค่าในอาร์เรย์เป็น 1) โปรแกรมจะสร้างโหนดใหม่สำหรับจุดยอดที่เชื่อมโยงและเพิ่มโหนดเหล่านั้นไปยังลิสต์เชื่อมโยงของจุดยอดที่กำลังพิจารณา

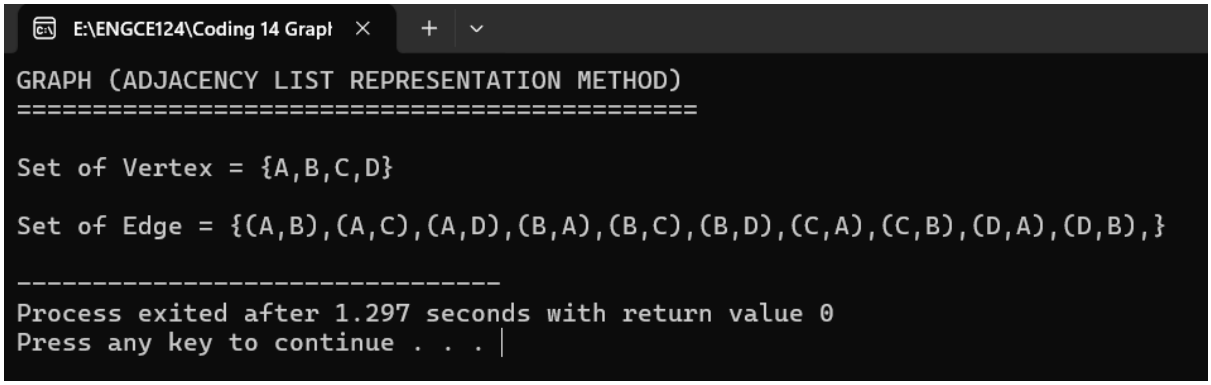
4. การแสดงผลชุดของจุดยอด (Vertex) ฟังก์ชัน DispSetOfVertex() จะถูกเรียกใช้เพื่อแสดงชุดของจุดยอดในกราฟ

- โปรแกรมจะพิมพ์ข้อความที่ระบุว่าเป็นชุดของจุดยอด
- จะทำการวนลูปผ่านอาร์เรย์ Start ที่เก็บที่อยู่ของโหนดเริ่มต้น (Head Node) และพิมพ์ชื่อของแต่ละจุดยอด
- ผลลัพธ์จะถูกแสดงในรูปแบบ {A, B, C, D} ซึ่งแสดงชุดของจุดยอดทั้งหมดในกราฟ

5. การแสดงผลชุดของเส้นเชื่อม (Edge) ฟังก์ชัน DispSetOfEdge() จะถูกเรียกใช้เพื่อแสดงชุดของเส้นเชื่อมในกราฟ

- โปรแกรมจะพิมพ์ข้อความที่ระบุว่าเป็นชุดของเส้นเชื่อม
- จะทำการวนลูปผ่านอาร์เรย์ Start และสำหรับแต่ละจุดยอด จะเดินผ่านลิสต์เชื่อมโยงของจุดยอดนั้น ๆ
- โปรแกรมจะพิมพ์คู่ของจุดยอดที่เชื่อมโยงกันในรูปแบบ (A,B), (A,C), และอื่น ๆ
- ผลลัพธ์จะถูกแสดงในรูปแบบ {(A,B), (A,C), (A,D), (B,C), (B,D)} ซึ่งแสดงชุดของเส้นเชื่อมทั้งหมดในกราฟ

6. การสิ้นสุดโปรแกรม โปรแกรมจะรอให้ผู้ใช้กดปุ่ม (ด้วย getch()) ก่อนที่จะสิ้นสุดการทำงาน หลังจากที่ผู้ใช้กดปุ่ม โปรแกรมจะกลับไปที่ระบบปฏิบัติการพร้อมกับรหัสการสิ้นสุด 0 (return 0)



รายละเอียดของผลลัพธ์โดยสรุป

- Set of Vertex: แสดงชุดของจุดยอดทั้งหมดในกราฟ ซึ่งในกรณีนี้คือ {A, B, C, D} การแสดงผลนี้บ่งบอกว่ากราฟประกอบด้วย 4 จุดยอด
- Set of Edge: แสดงชุดของเส้นเชื่อมทั้งหมดที่เชื่อมต่อระหว่างจุดยอดในกราฟ เช่น (A,B), (A,C), (A,D), (B,C), และ (B,D) การแสดงผลนี้บ่งบอกถึงการเชื่อมโยงที่มีอยู่ในกราฟ

บรรณานุกรม

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