ใบงานที่ 14

เรื่อง โครงสร้างข้อมูลกราฟ

เสนอ

อาจารย์ ปิยพล ยืนยงสถาวร

จัดทำโดย

นาย สารินทร์ อินต๊ะรักษา รหัส 65543206082-1

ใบงานนี้เป็นส่วนหนึ่งของรายวิชา โครงสร้างข้อมูลและขั้นตอนวิธี

หลักสูตรวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา

ประจำภาคที่ 1 ปีการศึกษา 2566

คำสั่ง/คำชี้แจง

- สร้างโค้ดโปรแกรมตามตัวอย่างในเอกสารประกอบการสอนสัปดาห์ที่7
- แสดงโค้ดโปรแกรมเป็นส่วนๆ พร้อมอธิบาย
- แสดงผลการรันโปรแกรม พร้อมอธิบายการทำงาน
- สรุปผลการทดลอง

ลำดับขั้นการทดลอง

```
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},
}; //Declare array and keep data of graph
struct Node //Declare structure of every node
{
 char info;
 struct Node *next;
};
struct Node *Start[MaxNode], *p; //Declare pointer uses
Node *Allocate() //Allocate 1 node from storage pool
{
 struct Node *temp;
 temp=(Node*)malloc(sizeof(Node)); //Allocate node by size declare
 return(temp);
}
```

- สร้างตัวแปร NodeName แบบอาร์เรย์ที่มีขนาด 4 ช่องและกำหนดให้มีสมาชิก 4 ตัวคือ 'A', 'B', 'C', 'D'
- ประกาศตัวแปร graph แบบอาร์เรย์ 2 มิติ โดยกำหนดขนาดของอาร์เรย์โดยใช้ MaxNode โดยใช้ 1 แทนการเชื่อมต่อระหว่างโหนดและ 0 แทนการไม่ได้เชื่อมต่อ
- ประกาศโครงสร้างข้อมูล Node ซึ่งมีสมาชิก 2 ส่วน คือ info ที่เก็บข้อมูลชนิด char และ next ที่เป็นตัวแปรเก็บ pointer ไปยังโหนดถัดไป
- ประกาศตัวแปร Start แบบอาร์เรย์ขนาด MaxNode ซึ่งจะใช้เก็บ pointer ไปยังโหนดแรกของแต่ละลิงค์ลิสต์.
- ประกาศฟังก์ชัน Allocate ที่ใช้สำหรับการจองหน่วยความจำเพื่อสร้างโหนดใหม่ โดยจะคืนตัวแปร pointer ไปยังโหนดใหม่ที่สร้าง.

```
void CreateHead() //Create Head Node
{
 int i;
 for (i=0;i<MaxNode;i++) //Count by Maximum of Node
 {
 p=Allocate();
 p->info=NodeName[i]; //Let INFO = Node Name
 p->next=NULL; //Let NEXT = NULL
 Start[i]=p; //Let Start of each node = Address of first Node
 }
}
void TransferToAdjacent() //Transfer array to Adjacency list of graph
{
 int i,j;
 struct Node *Rear; //Counter and point at last pointer finally
 for (i=0;i<MaxNode;i++) //row loop
 {
 Rear=Start[i]; //pointer Rear Start hear
 for(j=0;j<MaxNode;j++) //column loop
 {
 if (graph[i][j]==1) //if PATH?
 {
 p=Allocate(); //get new Node
 p->info=NodeName[j]; //Let info = NodeName[j]
 p->next=NULL; //Let NEST = NULL
 Rear->next=p; //Next of Rear point to New Node
 Rear=p; // Skip Rear pointer to Next Node
 }
 }
 }
}
```

- CreateHead: ฟังก์ชันนี้ใช้สร้าง (head node) สำหรับแต่ละโหนดในกราฟ โดยวนลูปผ่านโหนดทุกตัว และสร้างโหนดใหม่ด้วยฟังก์ชัน Allocate และกำหนดค่า info ให้กับโหนดใหม่เท่ากับชื่อของโหนดจาก NodeName และกำหนด next เป็น NULL แล้วเก็บ pointer ไปยังโหนดแรกของแต่ละลิงค์ลิสต์ในตัวแปร Start.
- TransferToAdjacent: ฟังก์ชันนี้ใช้สร้างลิงค์ลิสต์ adjacency list โดยวนลูปผ่านเมทริกซ์ graph และตรวจสอบว่ามีการเชื่อมต่อระหว่างโหนดหรือไม่ (โดยใช้ค่า 1 ใน graph แสดงถึงการเชื่อมต่อ) หากมีการเชื่อมต่อ ฟังก์ชันจะสร้างโหนดใหม่ด้วย Allocate และกำหนดค่า info เท่ากับชื่อของโหนดปลายทางจาก NodeName และกำหนด next เป็น NULL จากนั้นเพิ่มโหนดใหม่นี้ในลิงค์ลิสต์ adjacency list โดยการกำหนด Rear->next เป็นโหนดใหม่และเลื่อน Rear ไปยังโหนดใหม่เพื่อให้เตรียมสร้างโหนดถัดไปในกรณีที่มีการเชื่อมต่อเพิ่มเติม.

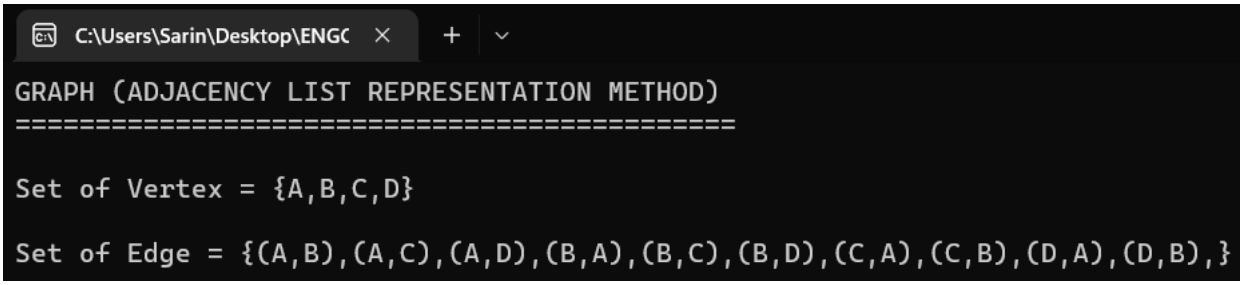
```c
void DispSetOfVertex() //Display set of Vertex
{
 int i;
 printf("\nSet of Vertex = {");
 for (i=0;i<MaxNode;i++) //Count only Start Node
 {
 printf("%c",Start[i]->info); //Display each node name
 if(i != MaxNode-1)
 printf(",");
 }
 printf("}\n");
}
void DispSetOfEdge() //Display set of Edge
{
 int i;
 struct Node *Temp;
 printf("\nSet of Edge = {");
 for (i=0;i<MaxNode;i++) //row loop
 {
 Temp=Start[i]; //Let Temp pointer Start hear
 Temp=Temp->next; //Skip Temp pointer to Next Node
 while (Temp != NULL) //Point at Node 2nd
 {
 printf("(%c,%c),",Start[i]->info,Temp->info); //Show each Edge
 Temp=Temp->next; //Skip Temp pointer to Next Node
 }
 }
 printf("}\n");
}
```

- DispSetOfVertex: ฟังก์ชันนี้ใช้สำหรับแสดงผลเซตของโหนดในกราฟ โดยวนลูปผ่านตัวแปร Start ที่เก็บ pointer ไปยังโหนดแรกของแต่ละลิงค์ลิสต์และแสดงชื่อของโหนด
- DispSetOfEdge: ฟังก์ชันนี้ใช้สำหรับแสดงผลเซตของเส้นเชื่อม (edges) ในกราฟ โดยวนลูปผ่านโหนดแรกของแต่ละลิงค์ลิสต์และนำ Temp ไปยังโหนดถัดไปในลิงค์ลิสต์ โดยใช้ Temp=Temp->next ในแต่ละรอบของลูป และแสดงชื่อของโหนดแรกและโหนดปลายทางที่เชื่อมกันในรูปแบบของเซต

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

- แสดงข้อความ "GRAPH (ADJACENCY LIST REPRESENTATION METHOD)"
- เรียกใช้ฟังก์ชัน CreateHead เพื่อสร้างโหนดหัวสำหรับแต่ละโหนดในกราฟ.
- เรียกใช้ฟังก์ชัน TransferToAdjacent เพื่อแปลงเมทริกซ์ graph เป็น adjacency list
- เรียกใช้ฟังก์ชัน DispSetOfVertex เพื่อแสดงผลเซตของโหนดในกราฟ.
- เรียกใช้ฟังก์ชัน DispSetOfEdge เพื่อแสดงผลเซตของเส้นเชื่อมในกราฟ.
- รอผู้ใช้กดปุ่มใดๆบนคีย์บอร์ดเพื่อจบการทำงาน โดยใช้ getch()
- ส่งค่า 0 ออกจากฟังก์ชัน main เพื่อบอกว่าโปรแกรมทำงานเสร็จสิ้นและไม่มีข้อผิดพลาด.

ผลลัพธ์การทดลอง



สรุปผลการทดลอง

การทดลองนี้ช่วยในการเรียนรู้และเข้าใจวิธีการสร้างและแสดงข้อมูลของกราฟแบบ adjacency list โดยใช้ภาษา C และการใช้โครงสร้างข้อมูลแบบโหนดและลิงค์ลิสต์ (linked list) ในการเก็บข้อมูลของโหนดและการเชื่อมต่อของกราฟ

สื่อ / เอกสารอ้างอิง

ไฟล์ประกอบการสอนสัปดาห์ที่ 8 ของ อาจารย์ ปิยพล ยืนยงสถาวร เรื่อง : โครงสร้างข้อมูลกราฟ