

"강의자료실"에서  
**14.AIGraph.c** 다운로드  
받아서 수정하기.

# Chapter 14. 그래프 (Graph)

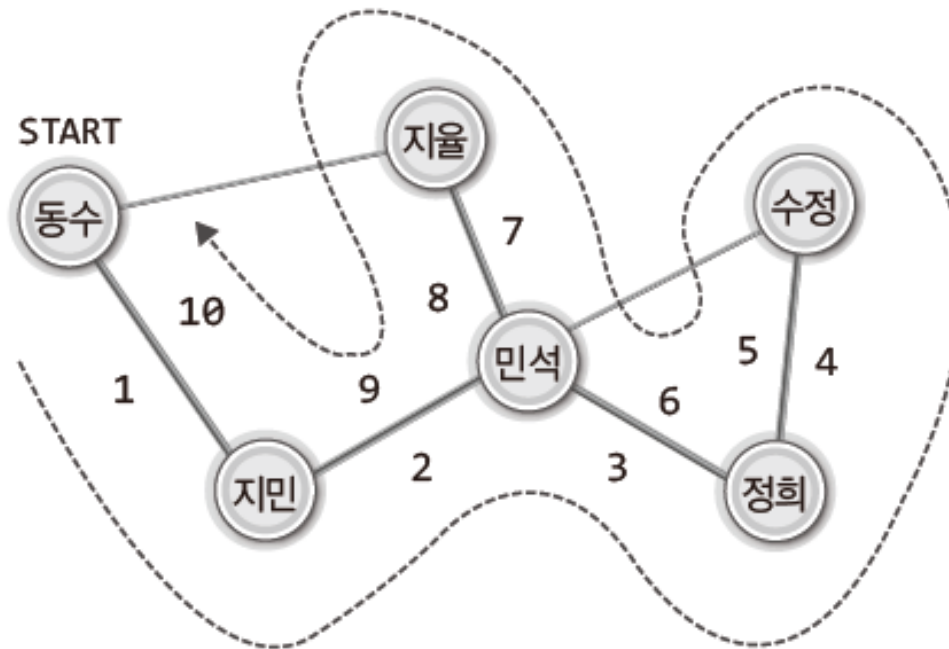
12주차  
DFS

# 그래프의 탐색

- ◆ 모든 정점을 1회씩 방문하기
  - 복잡한 작업 -> 별도의 알고리즘이 필요.
- ◆ 깊이우선탐색 (DFS) (Depth First Search)
- ◆ 너비우선탐색 (BFS) (Breadth First Search)

# 깊이 우선 탐색

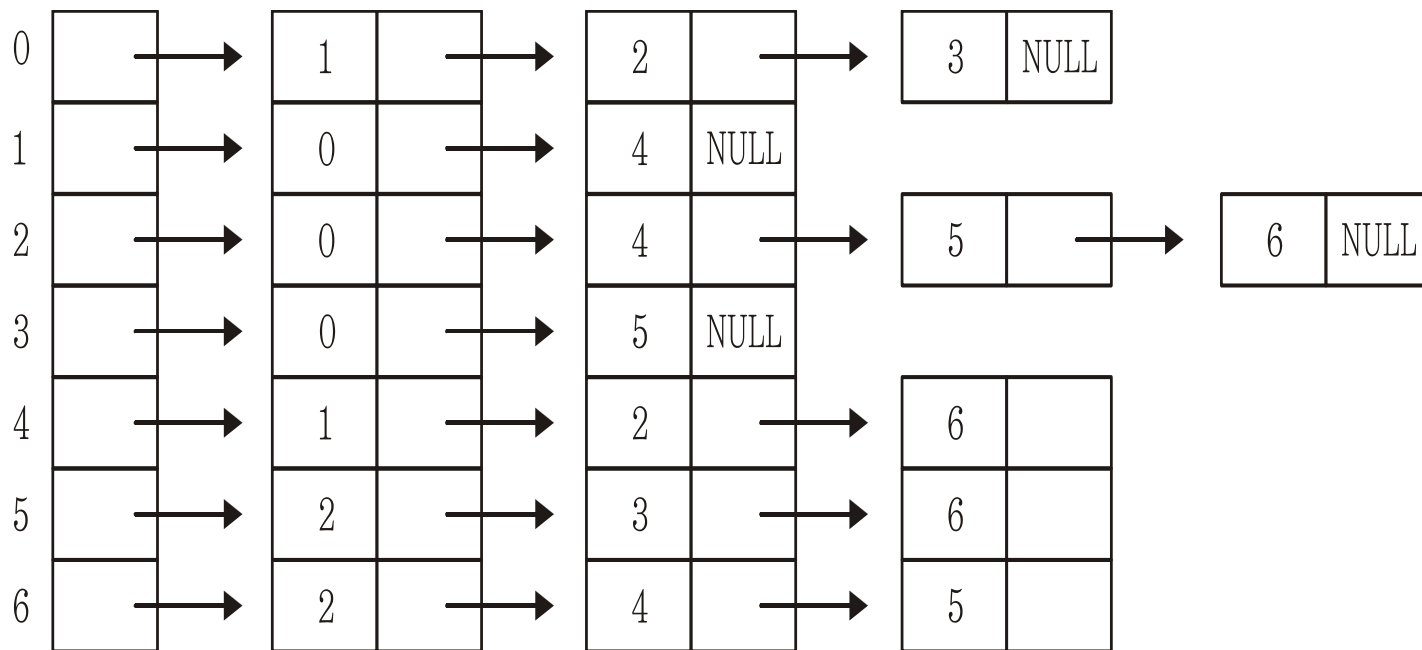
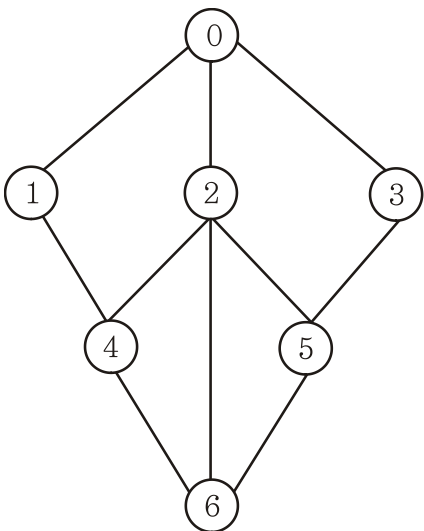
- ◆ 한 사람에게만 연락을 한다.
  - 현재 위치가 연락한 사람에게로 옮겨 간다.
- ◆ 연락할 사람이 없으면, 자신에게 연락한 사람에게 이를 알린다.
  - 자신에게 연락 준 사람한테로 현재 위치가 옮겨(되돌아) 간다.
- ◆ 처음 연락을 시작한 사람의 위치에서 연락은 끝이 난다



간선의 번호는  
지나가는 순서.

# 깊이 우선 탐색의 예

◆ 깊이우선탐색 결과 : **0 1 4 2 5 3 6**



# 깊이 우선 탐색의 구현 모델

## ◆ 운행 방법

- 시작 정점의 **한 방향으로**, 갈 수 있는 경로가 있는 곳까 **지 깊이 탐색**
- 더 이상 갈 곳이 없게 되면 가장 마지막에 만났던 갈림길 연결선이 있는 정점으로 되돌아가기 (**스택을 사용**)
- 다른 방향의 연결선으로 탐색을 계속 (모든 정점을 방문 할 때까지)

## ◆ 특징

- “트리의 **전위 운행**”, “미로 탐색” 과 유사

# 깊이 우선 탐색의 구현 모델

## ◆ 깊이 우선 탐색의 수행 순서

- (1) 시작 정점  $v$ 를 결정하여 현재 정점으로 삼기.
- (2) (방문하지 않았다면) 현재 정점을 방문.
- (3) 현재 정점에 인접한 정점 중에서
  - ❖ 방문하지 않은 정점  $w$ 가 있으면,  
현재 정점을 스택에 push하고  $w$ 를 현재 정점으로 삼기
  - 방문하지 않은 정점이 없으면,  
스택을 pop하여 현재 정점으로 삼기
  - ❖ (2)부터 다시 수행.
- (4) 스택이 공백이 될 때까지 수행.

## ◆ 각 정점에 대해, 방문여부 저장을 위한 변수(배열) 사용.

0 으로 초기화.

```
pg->visitInfo = (int*)malloc(sizeof(int) * pg->numV);
memset(pg->visitInfo, 0, sizeof(int) * pg->numV);
```

정점  $visitV$ 를 방문했으면 1로 바꾸기.

```
pg->visitInfo[visitV] = 1;
```

# 깊이 우선 탐색의 구현 (스택 추가)

```
#include <stdio.h>

#define TRUE 1
#define FALSE 0

#define STACK_LEN 100

typedef int Data;

typedef struct _arrayStack
{
    Data stackArr[STACK_LEN];
    int topIndex;
} ArrayStack;

typedef ArrayStack Stack;
```

“강의자료실”에서  
**14.AIGraph.c** 다운로드  
받아서 수정하기.

# 깊이 우선 탐색의 구현 (스택 함수 추가)

```
void StackInit(Stack* pstack)
{
    pstack->topIndex = -1;
}
int SIsEmpty(Stack* pstack)
{
    if (pstack->topIndex == -1)
        return TRUE;
    else
        return FALSE;
}
void SPush(Stack* pstack, Data data)
{
    pstack->topIndex += 1;
    pstack->stackArr[pstack->topIndex] = data;
}
```



# 깊이 우선 탐색의 구현 (스택 함수 추가)

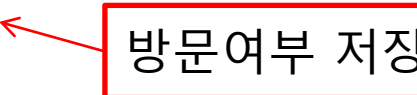
```
Data SPop(Stack* pstack) {
    int rIdx;
    if (SIsEmpty(pstack)) {
        printf("Stack Memory Error!");
        exit(-1);
    }
    rIdx = pstack->topIndex;
    pstack->topIndex -= 1;
    return pstack->stackArr[rIdx];
}

Data SPeek(Stack* pstack) {
    if (SIsEmpty(pstack)) {
        printf("Stack Memory Error!");
        exit(-1);
    }
    return pstack->stackArr[pstack->topIndex];
}
```

# 깊이 우선 탐색의 구현 (DFS 선언)

```
// 정점의 이름들을 상수화
enum { A, B, C, D, E, F, G, H, I, J };

typedef struct _ual
{
    int numV;    // 정점의 수
    int numE;    // 간선의 수
    List* adjList; // 간선의 정보
    int* visitInfo;
} ALGraph;
```



방문여부 저장

# 깊이 우선 탐색의 구현 (기존 함수 내용 수정)

```
void GraphInit(ALGraph* pg, int nv) {
    int i;
    pg->adjList = (List*)malloc(sizeof(List) * nv);
    pg->numV = nv;
    pg->numE = 0;    // 초기의 간선 수는 0개
    for (i = 0; i < nv; i++) {
        ListInit(&(pg->adjList[i]));
        SetSortRule(&(pg->adjList[i]), WhoIsPrecede);
    }
    pg->visitInfo = (int*)malloc(sizeof(int) * pg->numV);
    memset(pg->visitInfo, 0, sizeof(int) * pg->numV);
}

void GraphDestroy(ALGraph* pg) {
    if (pg->adjList != NULL)
        free(pg->adjList);

    if (pg->visitInfo != NULL)
        free(pg->visitInfo);
}
```

방문여부 등록을 위한  
공간의 할당 및 초기화

# 깊이 우선 탐색의 구현 (방문 처리 함수 추가)

```
int VisitVertex(ALGraph* pg, int visitV)
{
    if (pg->visitInfo[visitV] == 0)
    {
        pg->visitInfo[visitV] = 1;
        printf("%c ", visitV + 65); // 방문 정점 출력
        return TRUE;
    }

    return FALSE;
}
```

아직 미 방문인지 확인 후 방문 표시.

방문 처리

# 깊이 우선 탐색의 구현 (함수 추가)

```
// Depth First Search: 정점의 정보 출력
void DFSshowGraphVertex(ALGraph* pg, int startV)
{
    Stack stack;
    int visitV = startV;
    int nextV;

    // DFS를 위한 스택의 초기화
    StackInit(&stack);

    VisitVertex(pg, visitV);    // 시작 정점 방문

    .....

    // 탐색 정보 초기화
    memset(pg->visitInfo, 0, sizeof(int) * pg->numV);
}
```

# 깊이 우선 탐색의 구현 (함수 추가)

555-559p

```
while (LFirst(&(pg->adjList[visitV]), &nextV) == TRUE) {
    int visitFlag = FALSE;
    do {
        if (VisitVertex(pg, nextV) == TRUE) {
            SPush(&stack, visitV);
            visitV = nextV;
            visitFlag = TRUE;
            break;
        }
    } while (LNext(&(pg->adjList[visitV]), &nextV));

    if (visitFlag == FALSE) {
        if (SIsEmpty(&stack) == TRUE) // 스택이 비면 DFS종료
            break;
        else
            visitV = SPop(&stack);
    }
}
```

# 깊이 우선 탐색의 구현 (main 함수 수정)

```
int main(void) {
    ALGraph graph;
    GraphInit(&graph, 7);           // A, B, C, D, E, F, G의 정점 생성

    AddEdge(&graph, A, B); AddEdge(&graph, A, D);
    AddEdge(&graph, B, C); AddEdge(&graph, D, C);
    AddEdge(&graph, D, E); AddEdge(&graph, E, F);
    AddEdge(&graph, E, G);
    ShowGraphEdgeInfo(&graph);

    DFSShowGraphVertex(&graph, A); printf("\n");
    DFSShowGraphVertex(&graph, C); printf("\n");
    DFSShowGraphVertex(&graph, E); printf("\n");
    DFSShowGraphVertex(&graph, G); printf("\n");

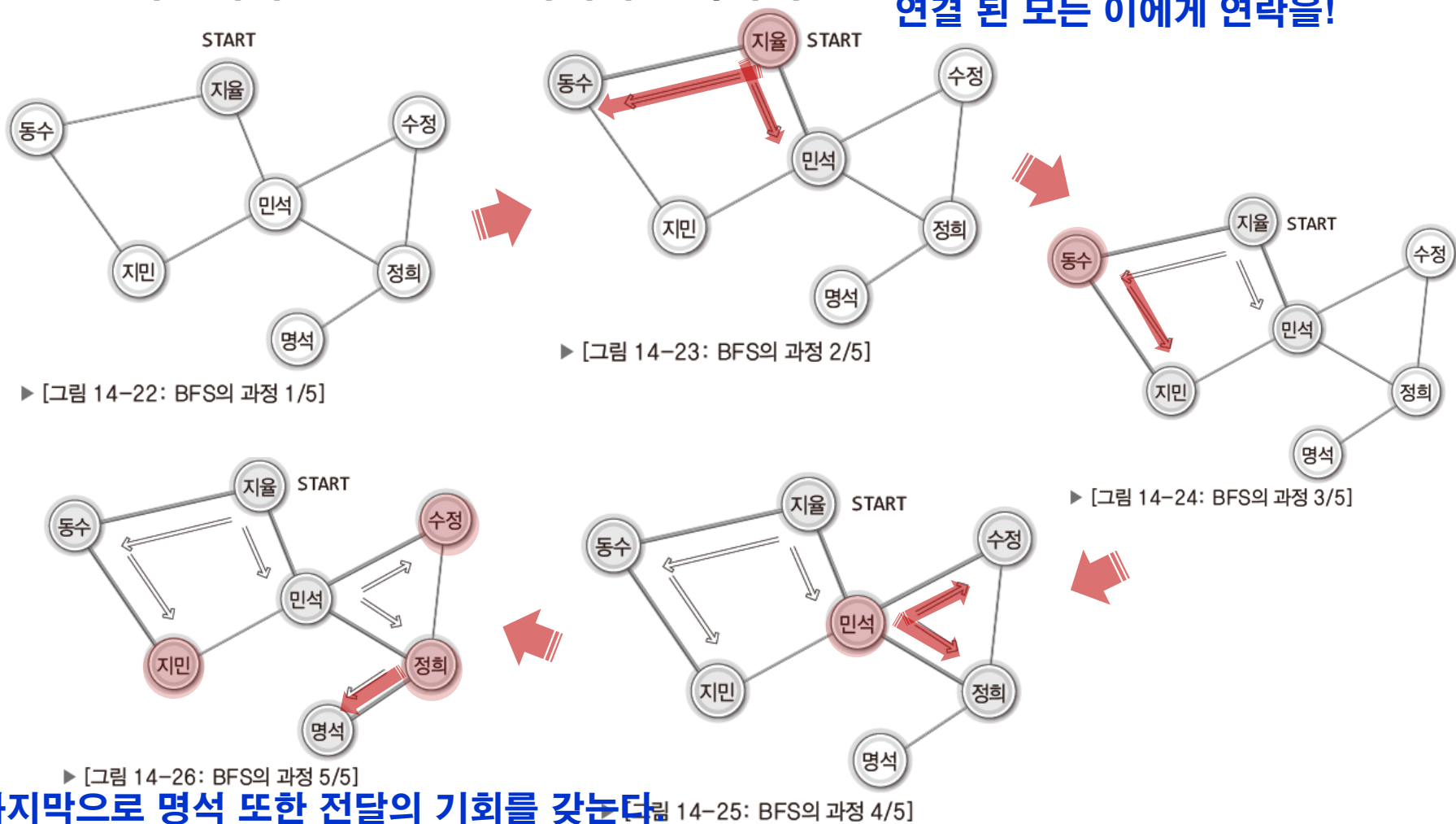
    GraphDestroy(&graph);
    return 0;
}
```

## [실행결과]

A와 연결된	정점:	B D
B와 연결된	정점:	A C
C와 연결된	정점:	B D
D와 연결된	정점:	A C E
E와 연결된	정점:	D F G
F와 연결된	정점:	E
G와 연결된	정점:	E
A B C D E F G		
C B A D E F G		
E D A B C F G		
G E D A B C F		

# 너비우선탐색

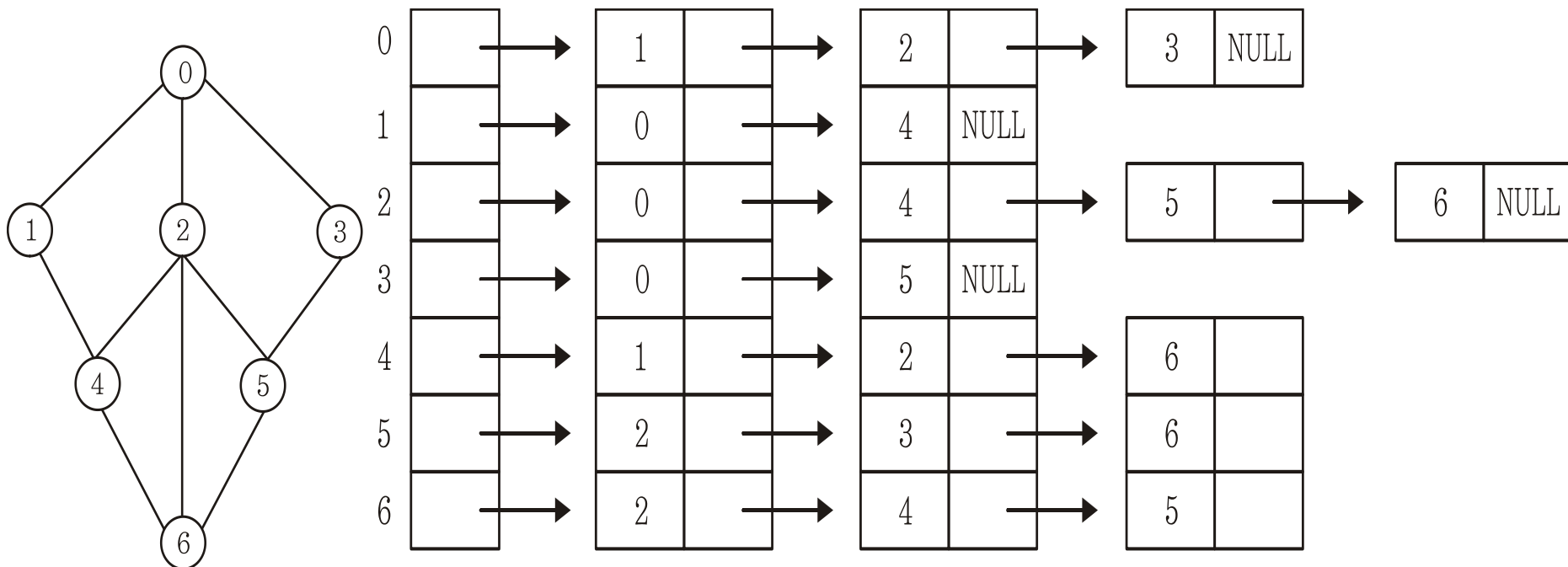
- ◆ 자신에게 연결된 모든 이에게 연락하기
- ◆ 연락 받은 이들은 다시 차례로, 자신에게 연결된 모든 이에게 연락하기.





# 너비 우선 탐색의 예

◆ 너비 우선 탐색 결과: **0 1 2 3 4 5 6**



# 너비 우선 탐색의 구현 모델 (breadth first search : BFS)

## ◆ 운행 방법

- 큐를 사용하여 시작 정점에서 가까운 정점부터 먼저 방문 -> 멀리 있는 정점들은 나중에 방문하는 순회방법

## ◆ 너비 우선 탐색의 수행 순서

- (1) 시작 정점을 {방문하고 큐에 삽입}.
- (2) 큐에서 정점 한 개를 꺼내어, 그 정점의 인접한 정점들 중에서 방문하지 않은 정점을 차례로 {방문하면서 큐에 삽입}.
- (3) 큐가 공백이 될 때까지 (2)를 반복.

# 너비 우선 탐색의 구현

## (14.AIGraphDFS.c 에 코드 추가하여 구현: 큐 선언)

```
////////////////////////////////////  
// 큐 선언 (배열 선언 아래에 위치!)  
  
#define QUE_LEN      100  
  
typedef struct _cQueue  
{  
    int front;  
    int rear;  
    Data queArr[QUE_LEN];  
} CQueue;  
  
typedef CQueue Queue;
```

# 너비 우선 탐색의 구현

(14.AIGraphDFS.c 에 코드 추가하여 구현: 큐 구현 1/3)

```
void QueueInit(Queue* pq) {
    pq->front = 0;
    pq->rear = 0;
}

int QIsEmpty(Queue* pq) {
    if (pq->front == pq->rear)
        return TRUE;
    else
        return FALSE;
}

int NextPosIdx(int pos) {
    if (pos == QUE_LEN - 1)
        return 0;
    else
        return pos + 1;
}
```

# 너비 우선 탐색의 구현

## (14.AIGraphDFS.c 에 코드 추가하여 구현: 큐 구현 2/3)

```
void Enqueue(Queue* pq, Data data) {
    if (NextPosIdx(pq->rear) == pq->front)    {
        printf("Queue Memory Error!");
        exit(-1);
    }

    pq->rear = NextPosIdx(pq->rear);
    pq->queArr[pq->rear] = data;
}

Data Dequeue(Queue* pq) {
    if (QIsEmpty(pq))    {
        printf("Queue Memory Error!");
        exit(-1);
    }

    pq->front = NextPosIdx(pq->front);
    return pq->queArr[pq->front];
}
```

# 너비 우선 탐색의 구현

(14.AIGraphDFS.c 에 코드 추가하여 구현: 큐 구현 3/3)

```
Data QPeek(Queue* pq) {
    if (QIsEmpty(pq)) {
        printf("Queue Memory Error!");
        exit(-1);
    }

    return pq->queArr[NextPosIdx(pq->front)];
}
```

# 너비 우선 탐색의 구현

(14.ALGraphDFS.c 에 코드 추가하여 구현: BFSShowGraphVertex  
1/2)

```
void BFSShowGraphVertex(ALGraph* pg, int startV) {
    Queue queue;
    int visitV = startV;
    int nextV;

    QueueInit(&queue); // DFS를 위한 큐의 초기화
    VisitVertex(pg, visitV); // 시작 정점 탐색

    while (LFirst(&(pg->adjList[visitV]), &nextV) == TRUE) {
        if (VisitVertex(pg, nextV) == TRUE)
            Enqueue(&queue, nextV);

        while (LNext(&(pg->adjList[visitV]), &nextV) == TRUE) {
            if (VisitVertex(pg, nextV) == TRUE)
                Enqueue(&queue, nextV);
        }

        if (QIsEmpty(&queue) == TRUE) // 큐가 비면 BFS 종료
            break;
        else
            visitV = Dequeue(&queue);
    }

    // 탐색 정보 초기화
    memset(pg->visitInfo, 0, sizeof(int) * pg->numV);
}
```

# 너비 우선 탐색의 구현

## (14.ALGraphDFS.c 에 코드 추가하여 구현: BFShowGraphVertex 2/2)

```
void BFShowGraphVertex(ALGraph* pg, int startV) {
    Queue queue;
    int visitV = startV;
    int nextV;
    QueueInit(&queue); // DFS를 위한 큐의 초기화
    VisitVertex(pg, visitV); // 시작 정점 탐색

    while (LFirst(&(pg->adjList[visitV]), &nextV) == TRUE) {
        if (VisitVertex(pg, nextV) == TRUE)
            Enqueue(&queue, nextV);

        while (LNext(&(pg->adjList[visitV]), &nextV) == TRUE) {
            if (VisitVertex(pg, nextV) == TRUE)
                Enqueue(&queue, nextV);
        }

        if (QIsEmpty(&queue) == TRUE) // 큐가 비면 BFS 종료
            break;
        else
            visitV = Dequeue(&queue);
    }
    memset(pg->visitInfo, 0, sizeof(int) * pg->numV); // 탐색 정보 초기화
}
```



# 너비 우선 탐색의 구현

(14.AIGraphDFS.c 에 코드 추가하여 구현: main 함수)

```
main(void) {
    ALGraph graph;
    GraphInit(&graph, 7);      // A, B, C, D, E, F, G의 정점 생성

    AddEdge(&graph, A, B);
    AddEdge(&graph, A, D);
    AddEdge(&graph, B, C);
    AddEdge(&graph, D, C);
    AddEdge(&graph, D, E);
    AddEdge(&graph, E, F);
    AddEdge(&graph, E, G);

    ShowGraphEdgeInfo(&graph);

    DFSShowGraphVertex(&graph, A); printf("\n");
    DFSShowGraphVertex(&graph, C); printf("\n");
    DFSShowGraphVertex(&graph, E); printf("\n");
    DFSShowGraphVertex(&graph, G); printf("\n");

    BFSShowGraphVertex(&graph, A); printf("\n");
    BFSShowGraphVertex(&graph, C); printf("\n");
    BFSShowGraphVertex(&graph, E); printf("\n");
    BFSShowGraphVertex(&graph, G); printf("\n");

    GraphDestroy(&graph);
    return 0;
}
```

## [실행결과]

```
A와 연결된 정점: B D
B와 연결된 정점: A C
C와 연결된 정점: B D
D와 연결된 정점: A C E
E와 연결된 정점: D F G
F와 연결된 정점: E
G와 연결된 정점: E

A B C D E F G
C B A D E F G
E D A B C F G
G E D A B C F
A B D C E F G
C B D A E F G
E D F G A C B
G E D F A C B
```