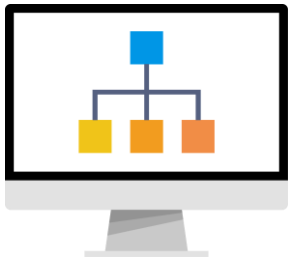


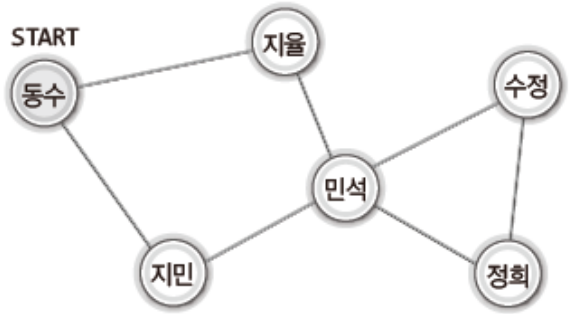
Data Structure & Algorithm
자료구조 및 알고리즘

24. 그래프 (Graph, Part 2)

덕성여자대학교
DUKSUNG WOMEN'S UNIVERSITY

그래프의 탐색

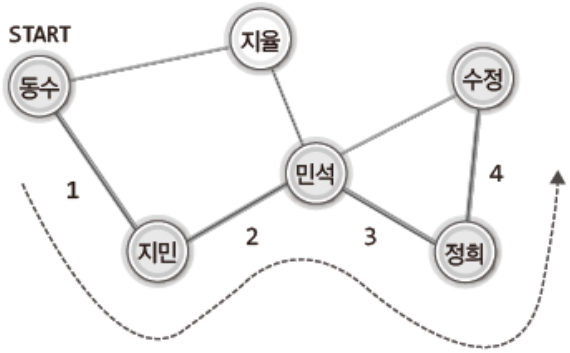
깊이 우선 탐색: Depth First Search



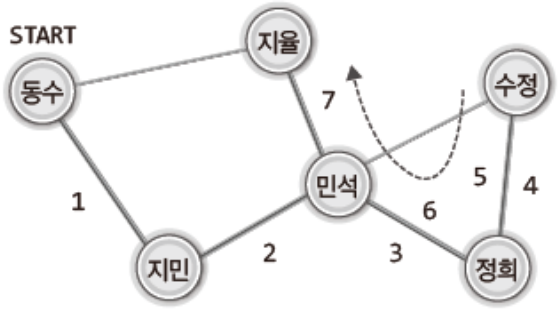
▶ [그림 14-16: DFS의 과정 1/4]

동수로부터 비상 연락망 가동!

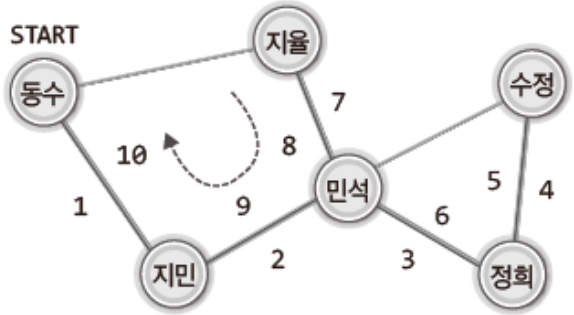
한 사람에게만
연락을 취해 나간다!



▶ [그림 14-17: DFS의 과정 2/4]



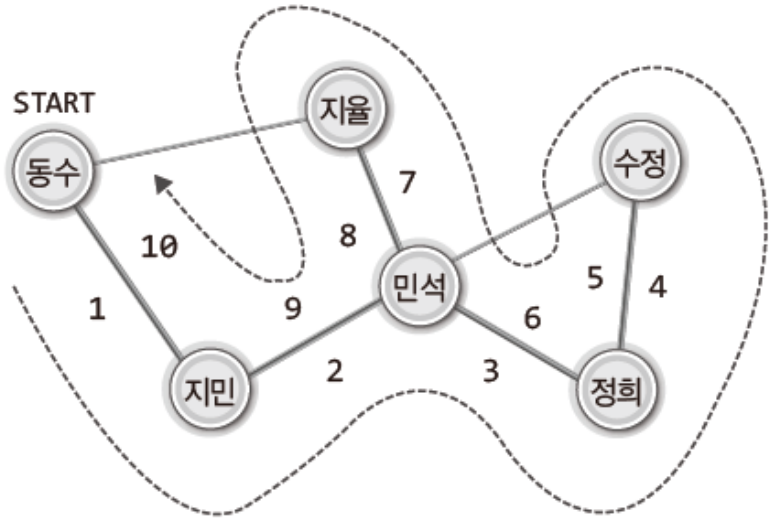
▶ [그림 14-18: DFS의 과정 3/4]



▶ [그림 14-20: DFS의 과정 4/4]

시작 점으로 되돌아 오면 연락 끝!

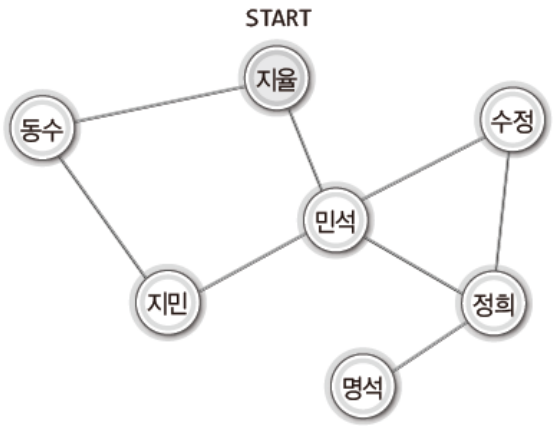
깊이 우선 탐색: 정리



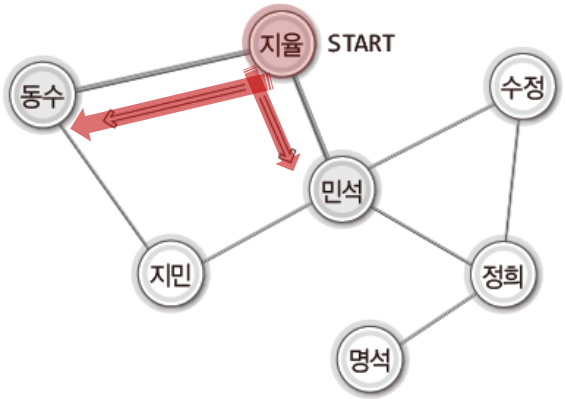
깊이 우선 탐색 과정의 핵심 세 가지

- 한 사람에게만 연락을 한다.
- 연락할 사람이 없으면, 자신에게 연락한 사람에게 이를 알린다.
- 처음 연락을 시작한 사람의 위치에서 연락은 끝이 난다

너비 우선 탐색: Breadth First Search

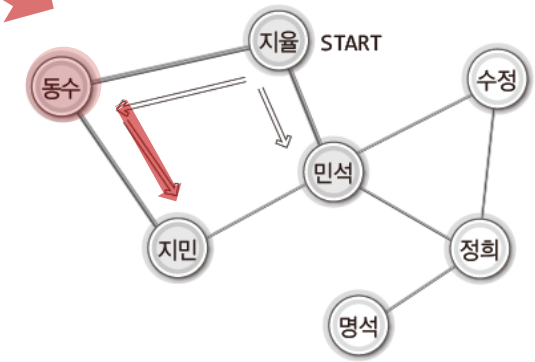


▶ [그림 14-22: BFS의 과정 1/5]

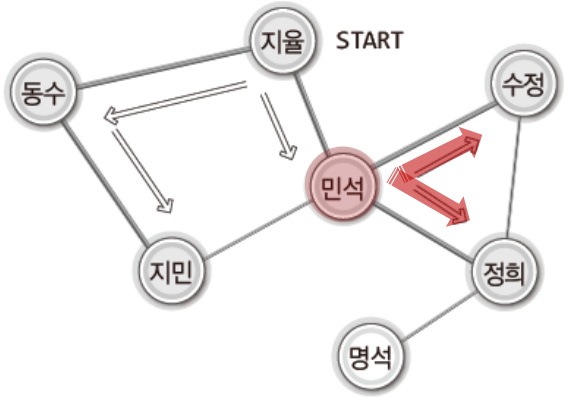


▶ [그림 14-23: BFS의 과정 2/5]

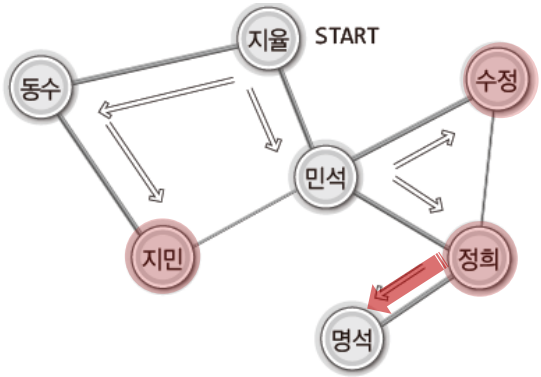
연결 된 모든 이에게 연락을!



▶ [그림 14-24: BFS의 과정 3/5]



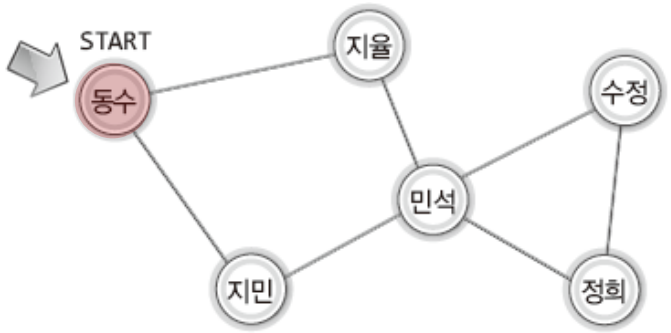
▶ [그림 14-25: BFS의 과정 4/5]



▶ [그림 14-26: BFS의 과정 5/5]

마지막으로 명석 또한 전달의 기회를 갖는다.

깊이 우선 탐색의 구현 모델: 과정 1~



방문 정보의 기록을 목적으로!

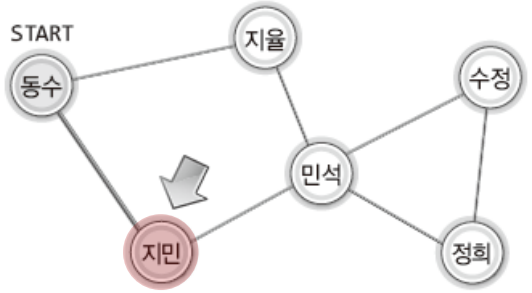
동수	지민	민석	정희	수정	지율

▶ [그림 14-28: DFS의 구현 1/7]

1) 내가 아는 사람 중에 연락을 받지 못한 사람이 있다?
   1) 나를 스택에 push
   2) 연락을 받지 못한 사람 중 한 사람에게 연락을 맡긴다.
2) 그게 아니면, 나를 스택에서 pop

경로 정보의 추적을 목적으로!

스택 !



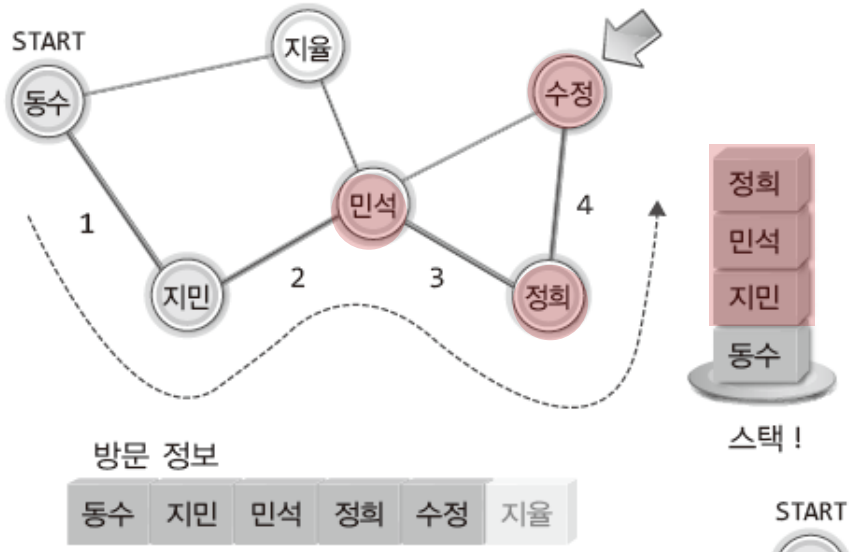
방문 정보

동수	지민	민석	정희	수정	지율

스택 !

▶ [그림 14-29: DFS의 구현 2/7]

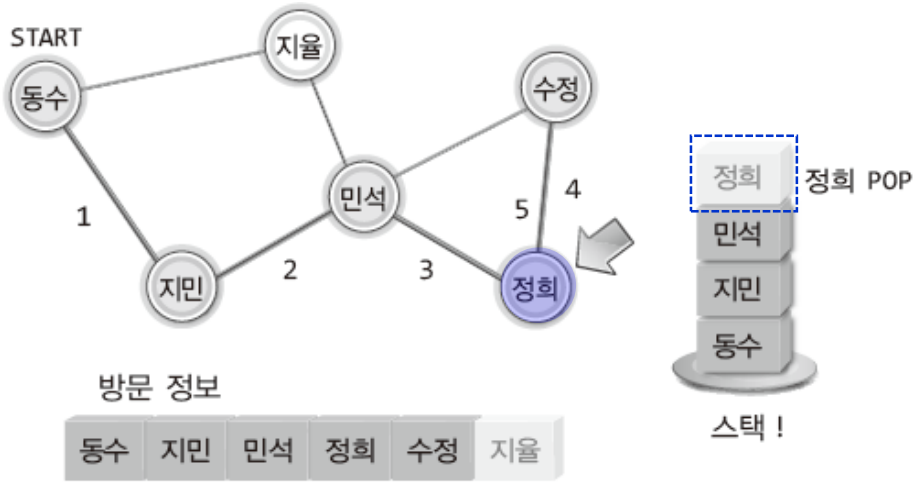
깊이 우선 탐색의 구현 모델: 과정 3~



▶ [그림 14-30: DFS의 구현 3/7]

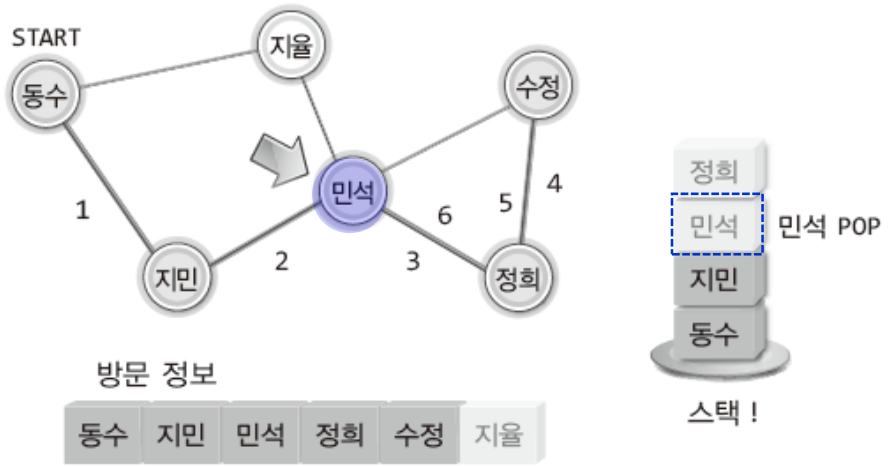
수정은 자신에게 연락한 사람의 정보를

스택에서 얻는다!



▶ [그림 14-31: DFS의 구현 4/7]

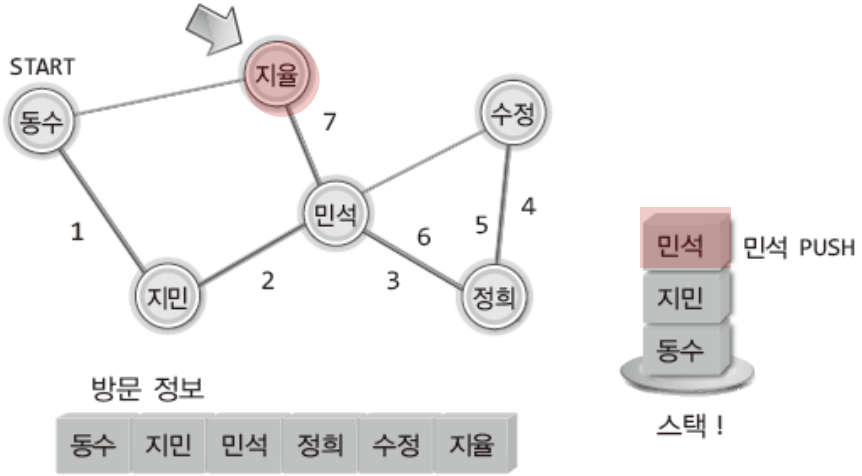
깊이 우선 탐색의 구현 모델: 과정 5~



▶ [그림 14-32: DFS의 구현 5/7]

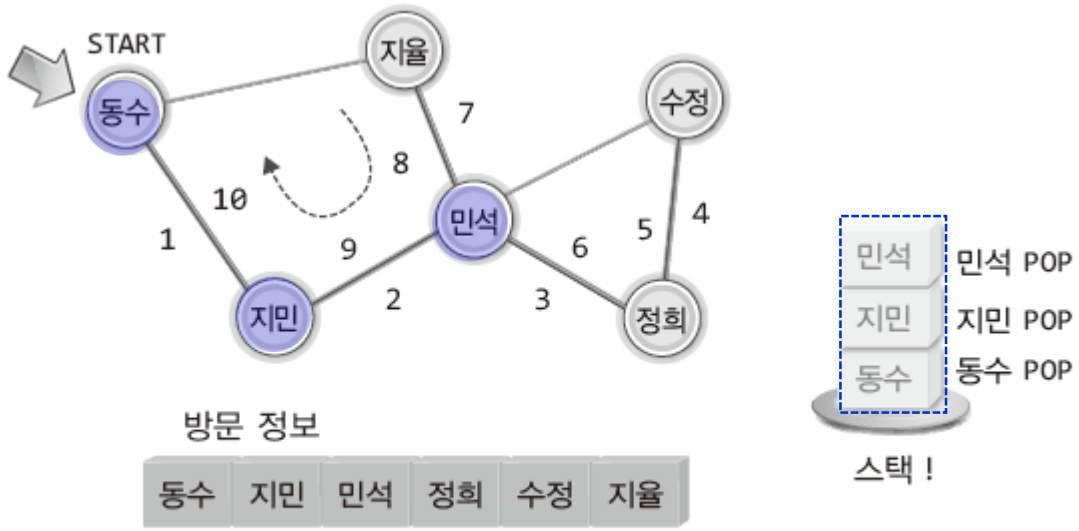
민석은 이전에 방문이 이뤄졌지만, 이와 상관 없이

민석을 떠날때 민석의 정보는 스택에 저장된다.



▶ [그림 14-33: DFS의 구현 6/7]

깊이 우선 탐색의 구현 모델: 과정 7



▶ [그림 14-34: DFS의 구현 7/7]

스택에 저장된 정보를 마지막까지 꺼내어 역으로 그 경로를 추적하다 보면 시작 위치로 이동이 가능하다!

깊이 우선 탐색의 실제 구현: 파일의 구성

ALGraph.h
ALGraph.c

→ DFShowGraphVertex 함수의 선언 및 정의 추가하여!

ALGraphDFS.h
ALGraphDFS.c

깊이 우선 탐색의 실제 구현

void DFShowGraphVertex(ALGraph * pg, int startV);

- 그래프의 모든 정점 정보를 출력하는 함수
- DFS를 기반으로 정의가 된 함수

구현 결과를 반영한 파일의 구성

- ALGraphDFS.h, ALGraphDFS.c 그래프 관련
- ArrayBaseStack.h, ArrayBaseStack.c 스택 관련(Chapter 06에서 구현)
- DLinkedList.h, DLinkedList.c 연결 리스트 관련(Chapter 04에서 구현)
- DFSMain.c

깊이 우선 탐색의 실제 구현: ALGraphDFS.h

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

정점의 이름을 결정하는 방법

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

탐색과정에서 탐색이 진행된 정점 정보를 담기 위한 멤버 추가!

```
// 그래프의 초기화
void GraphInit(ALGraph * pg, int nv);

// 그래프의 리소스 해제
void GraphDestroy(ALGraph * pg);

// 간선의 추가
void AddEdge(ALGraph * pg, int fromV, int toV);

// 간선의 정보 출력
void ShowGraphEdgeInfo(ALGraph * pg);

// 정점의 정보 출력: Depth First Search 기반
void DFShowGraphVertex(ALGraph * pg, int startV);
```

멤버 visitInfo 관련 추가 코드

```
void GraphInit(ALGraph * pg, int nv)
{
    . . . .
    // 정점의 수를 길이로 하여 배열을 할당
    pg->visitInfo = (int *)malloc(sizeof(int) * pg->numV);

    // 배열의 모든 요소를 0으로 초기화!
    memset(pg->visitInfo, 0, sizeof(int) * pg->numV);
}
```

멤버 visitInfo 관련 추가 코드

```
void GraphDestroy(ALGraph * pg)
{
    . . . .
    // 할당된 배열의 소멸!
    if(pg->visitInfo != NULL)
        free(pg->visitInfo);
}
```

깊이 우선 탐색의 실제 구현: Helper Func

방문한 정점의 정보를 기록 및 출력

```
int VisitVertex(ALGraph * pg, int visitV)
{
    if(pg->visitInfo[visitV] == 0)          // visitV에 처음 방문일 때 '참'인 if문
    {
        pg->visitInfo[visitV] = 1;          // visitV에 방문한 것으로 기록
        printf("%c ", visitV + 65);         // 방문한 정점의 이름을 출력
        return TRUE;            // 방문 성공!
    }
    return FALSE;       // 방문 실패!
}
```

이미 방문한 정점이라면 FALSE가 반환된다!

DFShowGraphVertex 함수의 구현에 필요한, DFShowGraphVertex 함수 내에서 호출이 되는 함수로써 방문한 정점의 정보를 그래프의 멤버 visitInfo가 가리키는 배열에 등록하는 기능을 제공한다.

깊이 우선 탐색의 실제 구현: DFShow~ 함수의 정의

```
void DFShowGraphVertex(ALGraph * pg, int startV)
{
    . . . . 초기화 영역 . . . .
    while(LFirst(&(pg->adjList[visitV]), &nextV) == TRUE)
    {        연결된 정점의 정보를 얻어서!
        int visitFlag = FALSE;
        if(VisitVertex(pg, nextV) == TRUE) {
            . . . . 방문을 시도했는데 방문에 성공하면
        } else {
            . . . . 방문을 시도했는데 방문한적 있는 곳이라면
        }

        if(visitFlag == FALSE) {  연결된 정점과의 방문이 모두 완료되었다면,
            if(SIsEmpty(&stack) == TRUE)
                break;  스택이 비면! 종료!
            else
                visitV = SPop(&stack);
        }                  되돌아 가기 위한 POP 연산!
    }
    . . . . 마무리 영역 . . . .
}
```

```
Stack stack;
int visitV = startV;
int nextV;
StackInit(&stack);
VisitVertex(pg, visitV);  시작 정점 방문!
SPush(&stack, visitV);
                시작 정점 떠나면서
                스택으로!
```

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

깊이 우선 탐색의 실제 구현: DFShow~ 함수의 정의

```
void DFShowGraphVertex(ALGraph * pg, int startV)
{
    . . . . 초기화 영역 . . . .
    while(LFirst(&(pg->adjList[visitV]), &nextV) == TRUE)
    {
        int visitFlag = FALSE;
        if(VisitVertex(pg, nextV) == TRUE) {
            . . . . 방문을 시도했는데 방문에 성공하면
        } else {
            . . . . 방문을 시도했는데 방문한적 있는 곳이라면
        }

        if(visitFlag == FALSE) {
            if(SIsEmpty(&stack) == TRUE)
                break;
            else
                visitV = SPop(&stack);
        }
    }
    . . . . 마무리 영역 . . . .
}
```

방문한 정점을 떠나야 하니 해당 정보 스택으로!

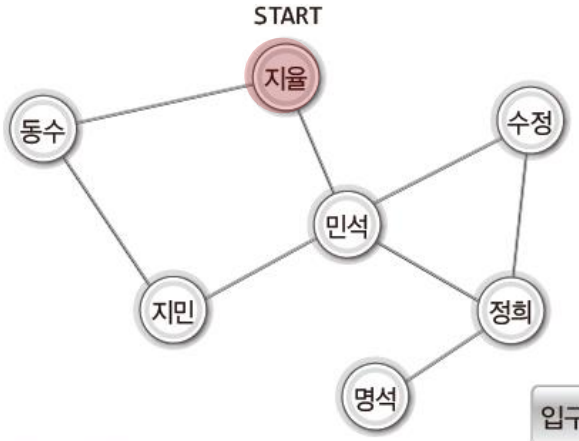
```
SPush(&stack, visitV);
visitV = nextV;
visitFlag = TRUE;
```

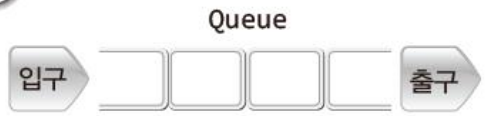
연결된 다른 정점을 찾아서 방문을 시도하는 일련의 과정!

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

너비 우선 탐색의 구현 모델: 과정 1~

1) 내가 아는 사람 중에 연락을 받지 못한 사람이 있다?
   1) 모두 연락한 후 queue에 enqueue
2) dequeue하여 나온 사람이 다음차례!

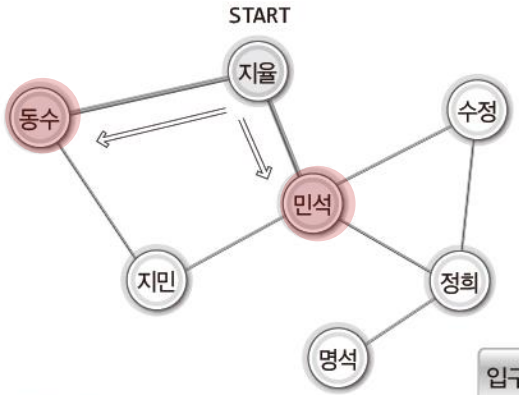


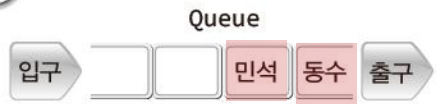


방문 정보

동수	지민	민석	정희	수정	지율	명석

▶ [그림 14-35: BFS의 구현 1/5]





동수 ,민석 enqueue

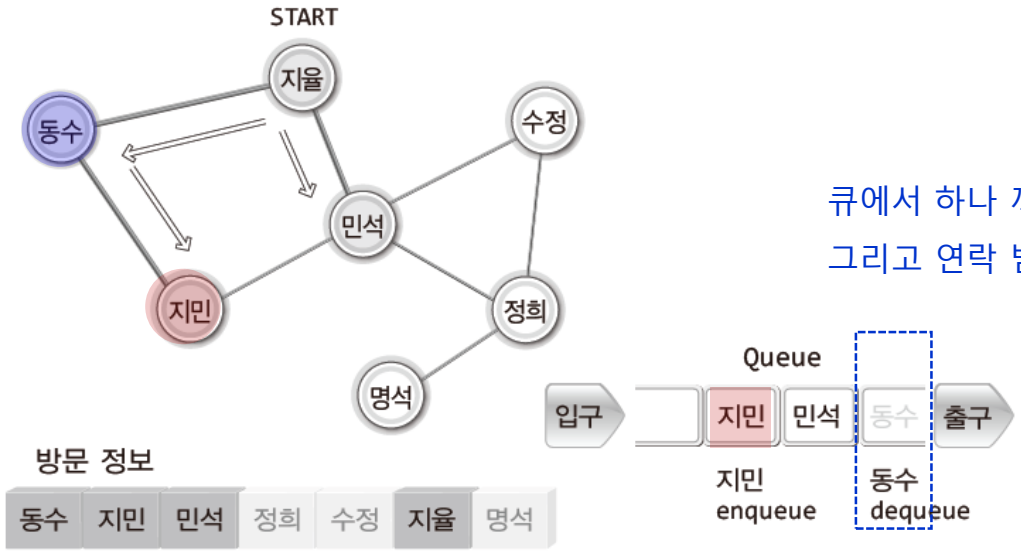
동수와 민석은 연락을 받기만 했을 뿐 연락을 취하지는 않은 대상! 이러한 대상의 정보를 큐에 저장!

방문 정보

동수	지민	민석	정희	수정	지율	명석

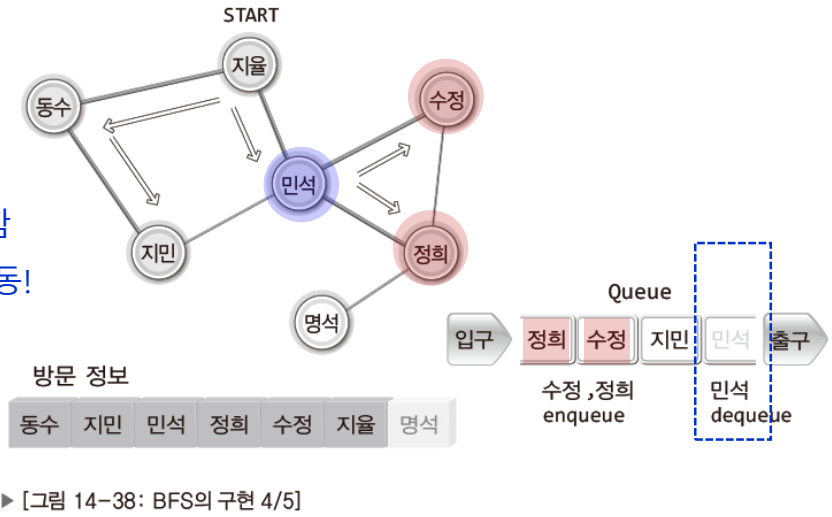
▶ [그림 14-36: BFS의 구현 2/5]

너비 우선 탐색의 구현 모델: 과정 3~



큐에서 하나 꺼낸 동수를 중심으로 연락을 취함
그리고 연락 받은 지민 정보는 큐로 이동!

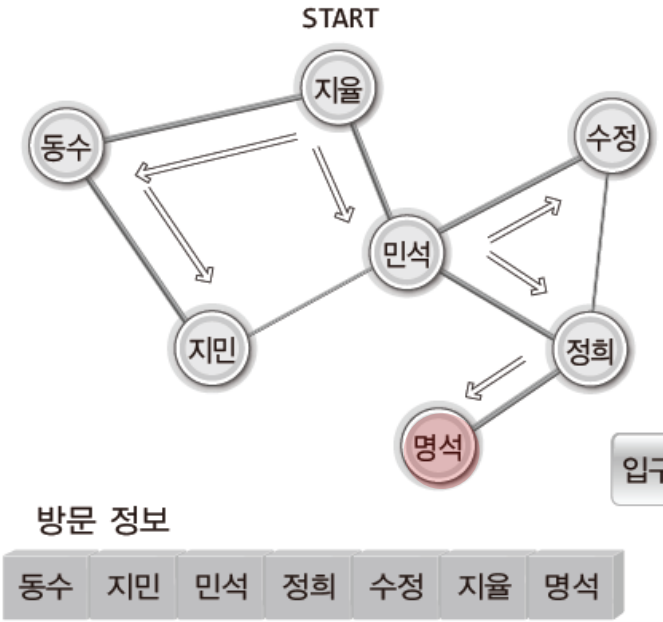
▶ [그림 14-37: BFS의 구현 3/5]



큐에서 하나 꺼낸 민석을 중심으로 연락을 취함
그리고 연락 받은 수정과 정희 정보는 큐로 이동!

▶ [그림 14-38: BFS의 구현 4/5]

너비 우선 탐색의 구현 모델: 과정 5

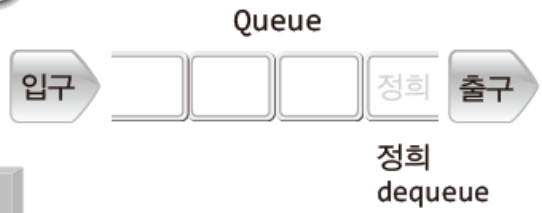


▶ [그림 14-39: BFS의 구현 5/5]

큐에서 꺼내어 방문을 진행하고,

방문한 정점의 정보를 다시 큐에 넣고.. 하는 일련의 과정을 반복!

언제까지? 큐가 완전히 빌 때까지!



명석의 정보도 큐에 들어갔다가 나온다!

위의 그림에서는 명석의 정보가 마지막에 큐에 들어간다.

그리고 빠져나오면서 종료하게 된다!

너비 우선 탐색의 실제 구현

ALGraph.h
ALGraph.c

→ BFShowGraphVertex 함수의 선언 및 정의 추가하여!

ALGraphBFS.h
ALGraphBFS.c

너비 우선 탐색의 실제 구현

void BFShowGraphVertex(ALGraph * pg, int startV);

- 그래프의 모든 정점 정보를 출력하는 함수
- BFS를 기반으로 정의가 된 함수

구현 결과를 반영한 파일의 구성

- ALGraphBFS.h, ALGraphBFS.c — 그래프 관련
- CircularQueue.h, CircularQueue.c — 큐 관련(Chapter 07에서 구현)
- DLinkedList.h, DLinkedList.c — 연결 리스트 관련(Chapter 04에서 구현)
- BFSain.c

너비 우선 탐색의 실제 구현: ALGraphBFS.h

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

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

// 그래프의 초기화
void GraphInit(ALGraph * pg, int nv);

// 그래프의 리소스 해제
void GraphDestroy(ALGraph * pg);

// 간선의 추가
void AddEdge(ALGraph * pg, int fromV, int toV);

// 그래프의 간선 정보 출력
void ShowGraphEdgeInfo(ALGraph * pg);

// BFS 기반 그래프의 정점 정보 출력
void BFShowGraphVertex(ALGraph * pg, int startV);
```

ALGraphDFS.h와 동일하다!

너비 우선 탐색의 실제 구현

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

    QueueInit(&queue);
    VisitVertex(pg, visitV);
        시작점 방문!

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

        while(LNext(&(pg->adjList[visitV]), &nextV) == TRUE)
        {       계속해서 visitV에 연결된 정점 정보 얻음
            if(VisitVertex(pg, nextV) == TRUE)
            Enqueue(&queue, nextV);
        }

        if(QIsEmpty(&queue) == TRUE)
            break;    큐가 비면 탈출 조건이 성립!
        else
            visitV = Dequeue(&queue);
    }

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

코드의 전체적인 느낌이 DFShowGraphVertex와 유사하다. 그리고 그 함수보다 간결하다!

시간 복잡도 분석

- DFS든 BFS든 모든 정점을 한번씩 방문해야 한다.
  - O(V)

- 인접 리스트를 썼을 때
  - (정점 1에 연결된 간선 수) = (정점 1의 차수)
  - (정점 1의 차수) + (2의 차수) + … = 2E = O(E)

- 인접 행렬을 썼을 때
  - (정점 1에서 V개의 이웃 확인) + (정점 2에서 V개의 이웃 확인) … = V * V = O(V^2)

요약

	깊이 우선 탐색 (DFS)	너비 우선 탐색 (BFS)
방법	“파고들기”	“넓혀가기”
자료구조	스택	큐
인접리스트 시간복잡도	$O(V+E)$	$O(V+E)$
인접행렬 시간복잡도	$O(V^2)$	$O(V^2)$

트리도 그래프이므로, 트리를 탐색하는 방법은

- 전위 순회 (깊이 우선 탐색, preorder)
- 중위 순회 (inorder)
- 후위 순회 (postorder)
- 너비 우선 탐색 (level order)

출석 인정을 위한 보고서 제출

- 다음 질문에 대한 답을 PDF로 포털에 제출

1) DFS를 이용하여 연락을 했을 때 X가 연락을 받았다면 시작 사람으로부터 X까지 연락이 전해진 경로가 최단 연락 횟수를 가지는 것이 보장되는가? BFS로 했을 때는?

2) DFS 또는 BFS를 이용하여 오른쪽 그림과 같은 미로에서 S에서 F로 가는 길을 찾을 수 있을까?
   1) 미로를 어떻게 그래프로 변환해야 할까?
   2) 미로의 크기가 N * M 이라면 시간복잡도는 얼마일까?