동국대학교
dongguk university

실습: Week 11

Data Structures

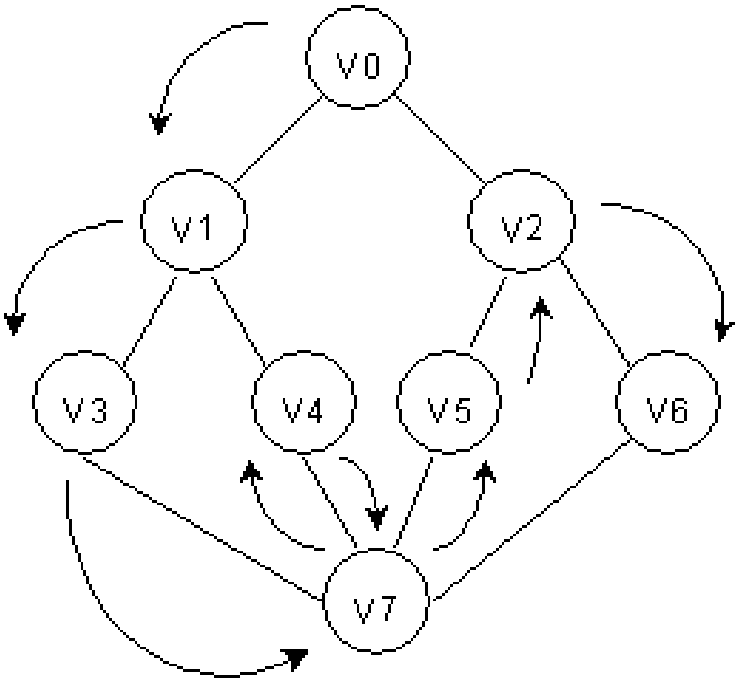
Contents

- Graph
  - Depth-First Search 구현
  - Breadth-First Search 구현
- 실습
  - 실습 11-1. DFS, BFS on adjacency list
  - 실습 11-2. DFS, BFS on adjacency matrix

Depth-First Search

- 그래프 탐색(graph search)
  - 그래프의 한 정점에서 시작하여 연결된 모든 정점들을 차례로 방문
- 깊이우선탐색(depth first search, DFS)
  - 정점 v에서 시작
  - 인접한 정점을 따라 계속 진행
  - 스택 이용 / 재귀함수 이용
- Time complexity
  - 인접리스트: $O(n+e)$
  - 인접행렬: $O(n^2)$
  - $e = |E|$, $n = |V|$, for $G = (V, E)$

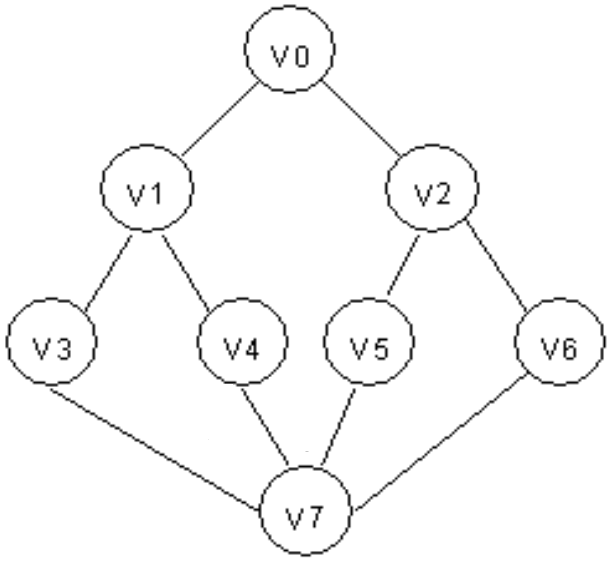
Depth-First Search

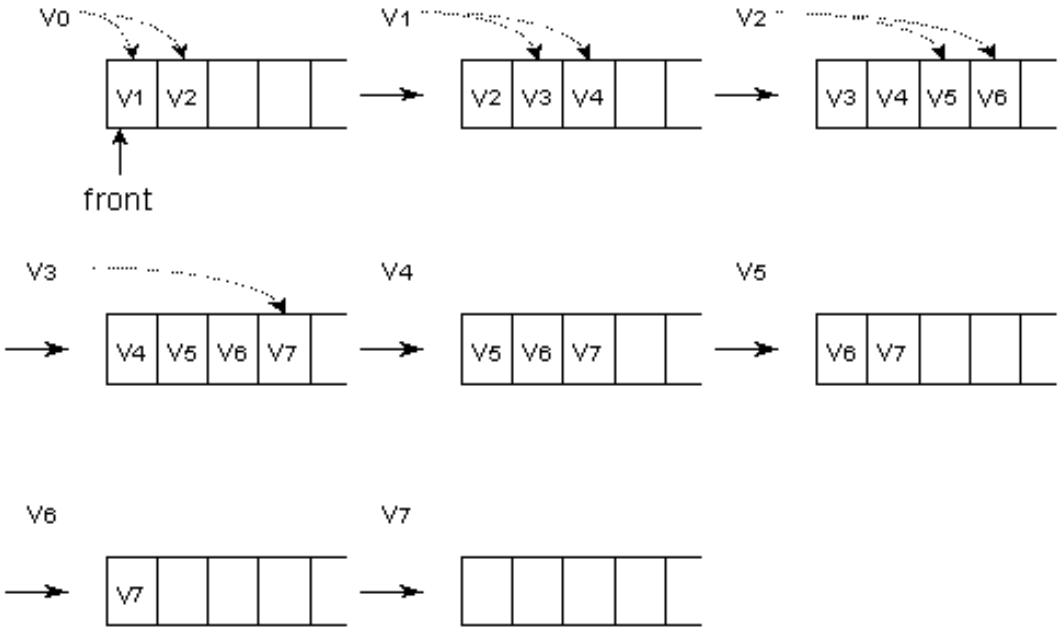


Breadth-First Search

- 너비우선탐색(breadth first search,BFS)
  - 정점 v에서 시작
  - V의 인접 리스트에 있는 모든 정점들을 방문
  - 큐 이용
- Time complexity
  - 인접리스트: $O(n+e)$
  - 인접행렬: $O(n^2)$

Breadth-First Search





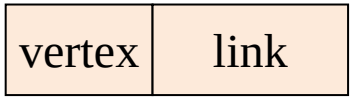
실습 11-1. DFS, BFS on adjacency list

- DFS와 BFS의 구현
  - Adjacency list로 표현된 그래프
- 명령어
  - D: DFS
  - B: BFS
  - Q: Quit

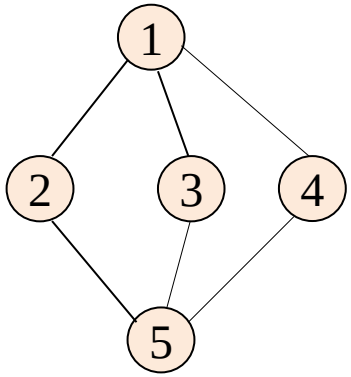
자료구조 및 함수

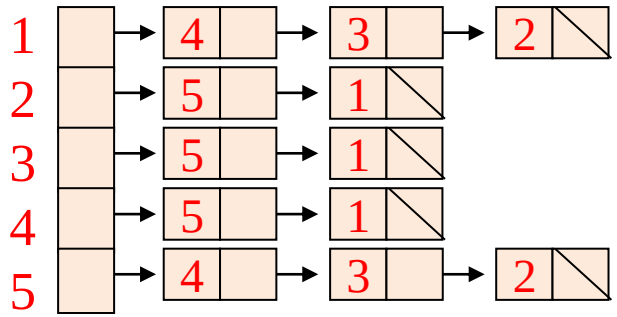
```
typedef struct node *node_pointer;
typedef struct node {
        int             vertex;
        node_pointer    link;
} node;

// Adjacency lists for a graph
node_pointer adj[MAX_VERTICES];
```



- void build_simple_graph();
  - 샘플 그래프의 생성





자료구조 및 함수

- void insert_edge(int v, int w);
  - 무방향 그래프에서 adjacency list에 edge (v, w) 를 삽입
- void dfs(int v);
  - vertex v에서 DFS 수행
- void bfs(int v);
  - vertex v에서 BFS 수행

자료구조 및 함수

- void addq(Element e);
  - 새 노드 생성
  - 큐의 맨 뒤에 삽입
- Element deleteq();
  - 큐의 맨 앞 노드 삭제
- boolean is_queue_empty();

실습 11-1. 실행 예

```
==========Command=========
D : DFS, B : BFS, Q : Quit
==========================

Command> b
 start vertex: 1
1  4  3  2  5

Command> b
 start vertex: 2
2  5  1  4  3

Command> b
 start vertex: 3
3  5  1  4  2
```

```
Command> d
 start vertex: 1
1  4  5  3  2

Command> d
 start vertex: 2
2  5  4  1  3

Command> d
 start vertex: 3
3  5  4  1  2
```

adj_list.h

```
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>
#include <stdlib.h>

#define    MAX_VERTICES 100
#define    boolean      int
#define    true         1
#define    false        0

// Adjacency list node
typedef struct node* node_pointer;
typedef struct node {
            int                    vertex;
            node_pointer           link;
} node;

// Adjacency lists for a graph
node_pointer adj[MAX_VERTICES];
int visited[MAX_VERTICES];
```

adj_list.h

```
void build_simple_graph();
void insert_edge(int v, int w);
void dfs(int v);
void bfs(int v);

typedef int Element;

// Global queue
typedef struct queue* queue_pointer;
typedef struct queue {
        Element     item;
        queue_pointer          link;
} queue;
queue_pointer front, rear;

void addq(Element e);
Element deleteq();
boolean is_queue_empty();
```

adj_list.c - main()함수

```
#include "adj_list.h"

void main() {
        char c;
        int i, v;
        front = rear = NULL;

        build_simple_graph();

        printf("===========Command========\n");
        printf("D : DFS, B : BFS, Q : Quit \n");
        printf("==========================\n");

        while (1) {
                printf("\nCommand> ");
                c = _getche();
                c = toupper(c);
```

adj_list.c - main()함수

```
                switch (c) {
                        case 'D':
                                printf("\n start vertex: ");
                                scanf("%d", &v);
                                for (i = 0; i < MAX_VERTICES; i++) visited[i] = false;
                                dfs(v);
                                printf("\n");
                                break;
                        case 'B':
                                printf("\n start vertex: ");
                                scanf("%d", &v);
                                for (i = 0; i < MAX_VERTICES; i++) visited[i] = false;
                                bfs(v);
                                printf("\n");
                                break;
                        case 'Q':
                                printf("\n");
                                exit(1);
                        default: break;

                }
        }
}
```

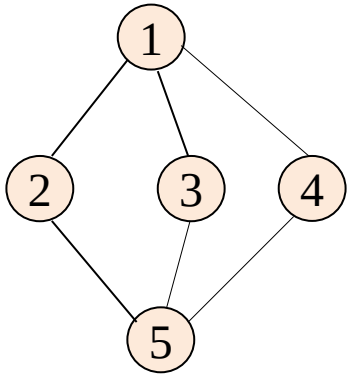
실습 11-2. DFS, BFS on adjacency matrix

- DFS와 BFS의 구현
  - Adjacency list로 표현된 그래프
- 명령어
  - D: DFS
  - B: BFS
  - Q: Quit

자료구조 및 함수

```
// Adjacency matrix for a graph
int adj[MAX_VERTICES][MAX_VERTICES];
```

- void build_simple_graph();
  - 샘플 그래프의 생성



	1	2	3	4	5
1	0	1	1	1	0
2	1	0	0	0	1
3	1	0	0	0	1
4	1	0	0	0	1
5	0	1	1	1	0

자료구조 및 함수

- void insert_edge(int v, int w);
  - 무방향 그래프에서 adjacency matrix에 edge (v, w) 를 삽입
- void dfs(int v);
  - vertex v에서 DFS 수행
- void bfs(int v);
  - vertex v에서 BFS 수행

자료구조 및 함수

- void addq(Element e);
  - 새 노드 생성
  - 큐의 맨 뒤에 삽입
- Element deleteq();
  - 큐의 맨 앞 노드 삭제
- boolean is_queue_empty();

실습 11-2. 실행 예

```
********* Command ********
D: DFS, B: BFS, Q: Quit
**************************

Command> d
 Start vertex: 1
1  2  5  3  4

Command> d
 Start vertex: 2
2  1  3  5  4

Command> d
 Start vertex: 3
3  1  2  5  4
```

```
Command> b
 Start vertex: 1
1  2  3  4  5

Command> b
 Start vertex: 2
2  1  5  3  4

Command> b
 Start vertex: 3
3  1  5  2  4
```

adj_matrix.h

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>

#define  MAX_VERTICES 100
#define  boolean   unsigned char
#define  true      1
#define  false     0

// Adjacency matrix for a graph
int adj[MAX_VERTICES][MAX_VERTICES];

int visited[MAX_VERTICES];

void build_simple_graph();
void insert_edge(int v, int w);

void dfs(int v);
void bfs(int v);
```

adj_matrix.h

```
typedef int Element;

// Global queue
typedef struct queue* queue_pointer;
typedef struct queue {
            Element             item;
            queue_pointer       link;
} queue;
queue_pointer front, rear;

void addq(Element e);
Element deleteq();
boolean is_queue_empty();
```

adj_matrix.c - main() 함수

```
#include "adj_matrix.h"

void main()
{
        char    c;
        int     i, v;
        front = rear = NULL;

        build_simple_graph();

        printf("********* Command ********\n");
        printf("D: DFS, B: BFS, Q: Quit \n");
        printf("**************************\n");

        while (1) {
                printf("\nCommand> ");
                c = _getche();
                c = toupper(c);
```

adj_matrix.c - main() 함수

```
                switch (c) {
                        case 'D':
                                printf("\n Start vertex: ");
                                scanf("%d", &v);
                                for (i = 0; i < MAX_VERTICES; i++) visited[i] = false;
                                dfs(v);
                                printf("\n");
                                break;
                        case 'B':
                                printf("\n Start vertex: ");
                                scanf("%d", &v);
                                for (i = 0; i < MAX_VERTICES; i++) visited[i] = false;
                                bfs(v);
                                printf("\n");
                                break;
                        case 'Q':
                                printf("\n");
                                exit(1);
                        default: break;
                }
        }
}
```