

자료구조응용

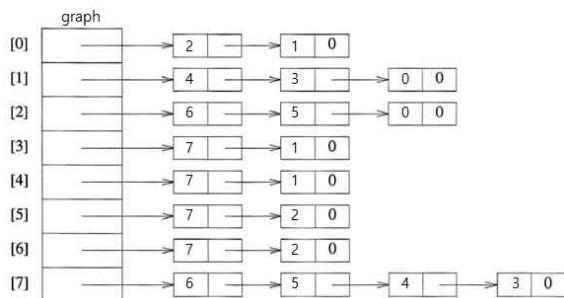
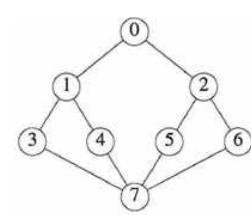
19. Graph: DFS, BFS (10점)

2025.11.17

1. 다음과 같이 무방향그래프(undirected graph) 데이터를 입력받아 인접리스트를 만들고 dfs 결과를 출력하는 프로그램을 작성하라.(5점)

(1) 입력파일(input.txt) 및 자료구조

```
8 10
0 1
0 2
1 3
1 4
2 5
2 6
3 7
4 7
5 7
6 7
```



- ※ 입력파일의 첫 줄은 정점(vertex) 수와 간선(edge)의 수를 나타냄
- ※ 그래프의 정점 인덱스는 0부터 시작됨
- ※ 주의: 교재의 adjacency list 그림을 본 문제의 데이터 입력 순서에 따라 수정한 그림

(2) 실행순서

- ① 정점(vertex)과 간선(edge)의 수를 입력받음
- ② 그래프를 구성하는 간선을 하나씩 입력받으면서 인접리스트를 구성함
 - ※ 같은 간선이 두 번 입력되지 않음을 가정함
 - ※ 항상 헤더 다음인 처음 노드로 입력되게 함
- ③ dfs의 결과 출력
 - ※ Program 6.1의 재귀함수호출을 이용함. 시스템 스택의 사용
 - ※ dfs(0), dfs(1), ..., dfs(n)를 각각 출력함

(3) 구현 세부사항

```
#define FALSE 0
#define TRUE 1
short int visited[MAX_VERTICES];

void dfs(int v)
{ /* depth first search of a graph beginning at v */
    nodePointer w;
    visited[v] = TRUE;
    printf("%5d", v);
    for (w = graph[v]; w; w = w->link)
        if (!visited[w->vertex])
            dfs(w->vertex);
}
```

Program 6.1: Depth first search

(4) 실행 예

```
<<<<<<< Adjacency List >>>>>>>
graph[0] : 2 1
graph[1] : 4 3 0
graph[2] : 6 5 0
graph[3] : 7 1
graph[4] : 7 1
graph[5] : 7 2
graph[6] : 7 2
graph[7] : 6 5 4 3

<<<<<<< Depth First Search >>>>>>
dfs<0> : 0 2 6 7 5 4 1 3
dfs<1> : 1 4 7 6 2 5 0 3
dfs<2> : 2 6 7 5 4 1 3 0
dfs<3> : 3 7 6 2 5 0 1 4
dfs<4> : 4 7 6 2 5 0 1 3
dfs<5> : 5 7 6 2 0 1 4 3
dfs<6> : 6 7 5 2 0 1 4 3
dfs<7> : 7 6 2 5 0 1 4 3
계속하려면 아무 키나 누르십시오 . . .
```

2. 위 1번 문제에 대해 dfs 대신 bfs의 결과를 출력하는 프로그램을 작성하라. (5점)

(1) 실행순서

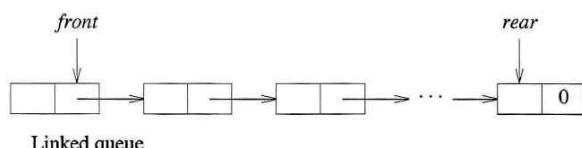
- ①, ② - 1번과 동일
- ③ bfs의 결과 출력
 - * Program 6.2 및 linked queue를 사용함
 - * bfs(0), bfs(1), ..., bfs(n)를 각각 출력함

(2) 구현 세부사항

① Linked Queue

The queue definition and the function prototypes used by bfs are:

```
typedef struct qNode *queuePointer;
typedef struct qNode {
    int vertex;
    queuePointer link;
} tQNode;
queuePointer front, rear;
void addq(int);
int deleteq();
```



```

void addq(int i, element item)
/* add item to the rear of queue i */
queuePointer temp;
MALLOC(temp, sizeof(*temp));
temp->data = item;
temp->link = NULL;
if (front[i])
    rear[i]->link = temp;
else
    front[i] = temp;
rear[i] = temp;
}

```

Program 4.7: Add to the rear of a linked queue

```

element deleteq(int i)
/* delete an element from queue i */
queuePointer temp = front[i];
element item;
if (!temp)
    return queueEmpty();
item = temp->data;
front[i] = temp->link;
free(temp);
return item;
}

```

Program 4.8: Delete from the front of a linked queue

* Program 4.7~4.8은 다중 큐에 대한 함수이므로 단일 큐에 대한 함수로 수정 (즉, i와 관련된 부분을 삭제). element를 int로, data를 vertex로 수정

② bfs 함수

```

void bfs(int v)
/* breadth first traversal of a graph, starting at v
   the global array visited is initialized to 0, the queue
   operations are similar to those described in
   Chapter 4, front and rear are global */
nodePointer w;
front = rear = NULL; /* initialize queue */
printf("%d", v);
visited[v] = TRUE;
addq(v);
while (front) {
    v = deleteq();
    for (w = graph[v]; w; w = w->link)
        if (!visited[w->vertex]) {
            printf("%d", w->vertex);
            addq(w->vertex);
            visited[w->vertex] = TRUE;
        }
}
}

```

Program 6.2: Breadth first search of a graph

(3) 실행 예

```
c:\> <<<<<<< Adjacency List >>>>>>>>
graph[0] : 2 1
graph[1] : 4 3 0
graph[2] : 6 5 0
graph[3] : 7 1
graph[4] : 7 1
graph[5] : 7 2
graph[6] : 7 2
graph[?] : 6 5 4 3

<<<<<<<< Breadth First Search >>>>>>>>
bfs<0> : 0 2 1 6 5 4 3 7
bfs<1> : 1 4 3 0 7 2 6 5
bfs<2> : 2 6 5 0 7 1 4 3
bfs<3> : 3 7 1 6 5 4 0 2
bfs<4> : 4 7 1 6 5 3 0 2
bfs<5> : 5 7 2 6 4 3 0 1
bfs<6> : 6 7 2 5 4 3 0 1
bfs<?> : 7 6 5 4 3 2 1 0
계속하려면 아무 키나 누르십시오 . . .
```

■ 제출 형식

- 솔루션 이름 : DS 19
- 프로젝트 이름 : 1, 2
- 제출 파일
 - ① 소스코드와 실행 결과가 보이도록 화면을 캡쳐한 보고서 파일("학번.pdf")
 - ※ 한글 [파일 → pdf로 저장하기...] 메뉴 사용
 - ② C 소스 파일을 하나의 디렉터리에 모아 압축한 파일 ("학번.zip")
 - ※ "학번.pdf"와 "학번.zip"을 하나로 압축하지 말고 별도 파일로 제출

■ 주의

- 소스 복사로는 실력향상을 기대할 수 없습니다!!!
- 1차 마감 : 수업일 자정
- 2차 마감 : 수업 익일 자정(만점의 60%, 반올림)
- 문항 별로 1차 2차 나눠서 제출할 수 없으며, 최종 제출 시간에 따라 1차, 2차로 구분함