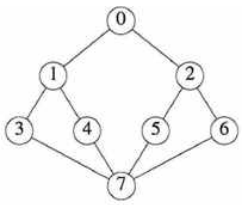
자료구조응용

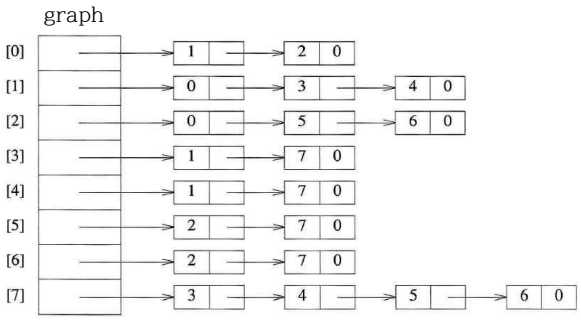
16. Graph: DFS, BFS

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

(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: 파일로부터 구성된 인접리스트의 노드 순서가 그림(graph)과 동일하지 않을 수 있음

(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 스택**의 사용

※ dfs(0), dfs(1), ..., dfs(n)를 각각 출력함

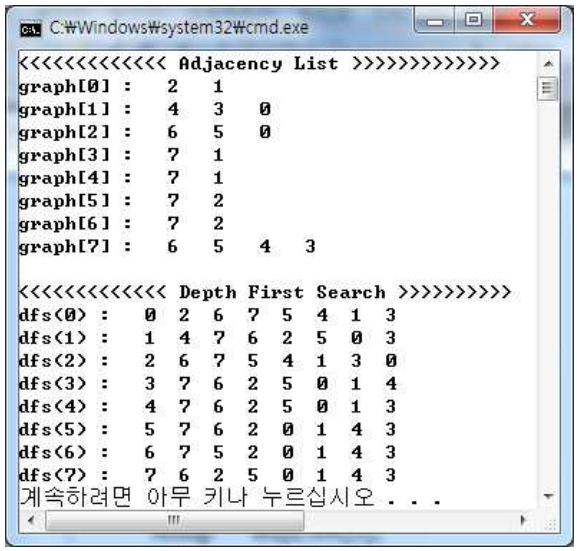
(3) 구현 세부사항(참조)

```
#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



2. 위 1번 문제에 대해 dfs 대신 bfs의 결과를 출력하는 프로그램을 작성하라.

(1) 실행순서

①, ② - 1번과 동일

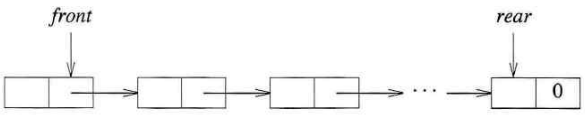
③ bfs의 결과 출력

※ Program 6.2 및 linked queue를 사용함

※ bfs(0), bfs(1), ..., bfs(n)를 각각 출력함

(2) 구현 세부사항(참조)

① Linked Queue



Linked queue

```
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

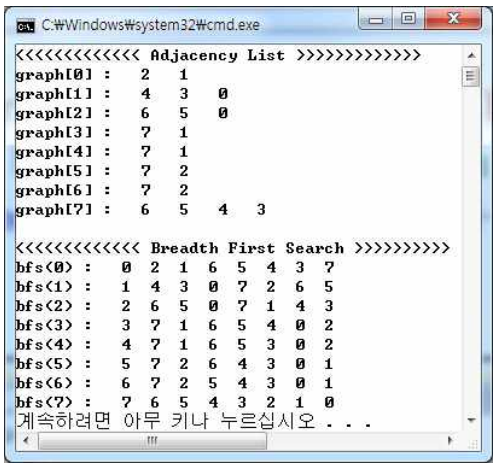
※ Program 4.7~4.8은 다중 큐에 대한 함수이므로 단일 큐에 대한 함수로 수정 (즉, i와 관련된 부분을 삭제). element를 int로, data를 vertex로 수정

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("%5d",v);
  visited[v] = TRUE;
  addq(v);
  while (front) {
    v = deleteq();
    for (w = graph[v]; w; w = w→link)
      if (!visited[w→vertex]) {
        printf("%5d", w→vertex);
        addq(w→vertex);
        visited[w→vertex] = TRUE;
      }
    }
}
```

Program 6.2: Breadth first search of a graph

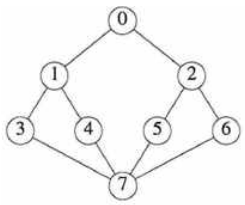
(3) 실행 예

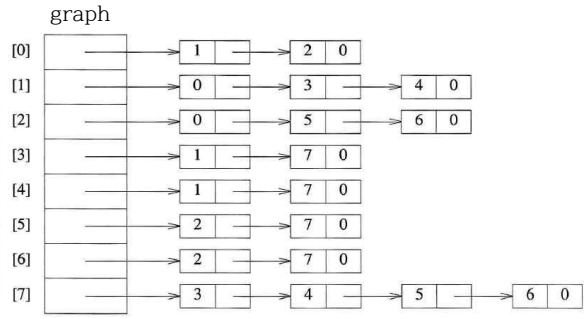


3. 입력된 무방향그래프의 connected component를 출력하는 프로그램을 작성하라.

(1) 입력파일(input.txt) 및 자료구조

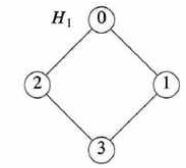
```
8 10
0 1
0 2
1 3
1 4
2 5
2 6
3 7
4 7
5 7
6 7
```

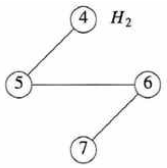


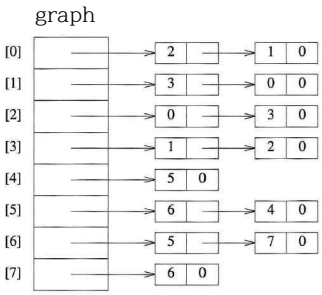


G1

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







G2

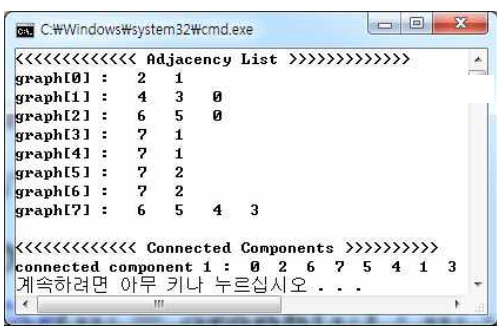
(2) 구현 세부사항

```
void connected(void)
{/* determine the connected components of a graph */
int i;
for (i = 0; i < n; i++)
   if(!visited[i]) {
      dfs(i);
      printf("\n");
   }
}
```

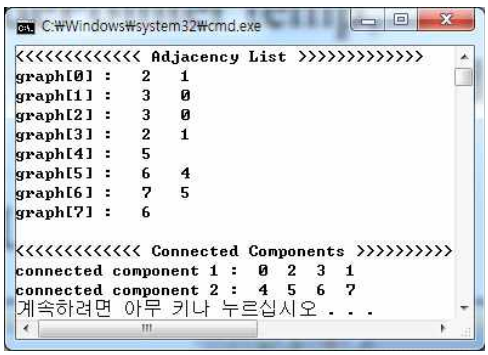
Program 6.3: Connected components

(3) 실행 예

G1



G2



■ 제출 형식

- 공학인증 시스템(ABEEK)에 과제를 올릴 때 제목:
- 1차 제출: 학번 이름 DS-16(1), 2차 제출: 학번 이름 DS-16(2)
- 솔루션 이름 : DS-16
- 프로젝트 이름 : 1, 2, 3
- 실행화면을 캡쳐하여 한글파일에 추가 후 솔루션 폴더에 포함.
- 한글 파일명 : 학번_이름_실습결과.hwp
- 솔루션 폴더를 압축하여 게시판에 제출할 것.
- 압축 파일 명: 학번_이름_DS-16.zip

- 제출은 2회걸쳐 가능(수정 시간 기준으로 처리)
- 제출 종료기한 :

1차 종료시 까지 제출 (100%)
2차 종료시 까지제출 (80%)