초등교육연구 제14집 제2호(2010. 2.) pp. 113~129
제주대학교 초등교육연구소

한국정보올림피아드 경시부문 교재 개발 연구

- 초등부를 중심으로 -

김 종 훈 (교육대학 컴퓨터교육전공 교수)
김 병 수 (제주동초등학교 교사)

목 차

요 약

한국정보올림피아드 경시부문 초등부 전국대회를 준비하는 학생들을 위해 본 교재를 개발하고자 한다. 본 교재는 J.S.Bruner의 표상이론에 맞게 알고리즘의 진행 절차를 시각화하여 이해가 쉽고 빠르게 되도록 구성하였으며 이를 실제 코드로 작성하여 자신의 논리를 프로그래밍 언어로 표현할 수 있게 하였다. 또한 기출문제, 확인문제 등을 통하여 배운 알고리즘을 확실하게 익혔는지 확인한다. 이러한 교재 개발이 경시대회를 준비하는 초등학생들에게 관련 도서가 전무하여 기회를 놓쳐버리는 일 등이나 사교육비를 들여 학원에만 의존해야 하는 최근의 경향에 대한 대안으로 사용되기 바란다.

ABSTRACT

Kim, Jong Hoon
Kin, Byeong Su

The purpose of this study is the development of textbook for the elementary students' final of Korea Olympiad in Informatics. This textbook is designed with the pictures of algorithm's process for easy understanding by J.S.Bruner's EIS theory. It makes learners can express their own logics in computer programming languages. and questions can confirm if learners are understanding the algorithm or not. This study may give a lot of chances to many elementary students who cannot prepare the competitions because of the money and find any books about it.

Ⅰ. 서 론

가) 연구의 필요성 및 목적

교육정보화 및 컴퓨터 교육의 목표는 학습자로 하여금 디지털 정보 지식의 생산자이자 소비자가 되게 하는 것이지 단순한 정보 소비자를 만들자는 것이 아니다. 학생을 정보 생산자가 되게 하려면 학생에게 컴퓨터 관련 기능 기술과 함께 그 기능 기술이 있게 한 컴퓨터 논리, 디지털 논리 더 나아가 디지털 문화논리까지도 함께 가르쳐야 한다. 디지털 논리를 습득할 때 학생은 컴퓨터식 사고를 할 수 있게 되고, 컴퓨터식 사고를 할 수 있는 상태에서 습득한 기능 기술일 때 그 기능 기술은 디지털 정보 지식, 더 나아가 디지털 매체에 관련한 프로그램을 개발할 수 있는 능력이 된다. 컴퓨터식 사고가 뒷받침되지 않는 컴퓨터 관련 기능 기술은 정보 소비능력 밖에는 되지 못하는 것이다[1].

우리나라에서 초등학생이 컴퓨터식 사고 능력을 개발하기 위한 교육과정은 현재 정보 영재 교육에서 찾아볼 수 있다.

정보 영재 육성을 위한 교육기관의 노력과 많은 관심에도 불구하고, 적합한 교재가 없어 독학으로 공부하기에 많은 무리가 있는 분야가 정보 영재 분야이다. 전 세계적으로 정보 영재에 대한 중요성을 인식하고 있으며 국내에서도 과학 영재를 육성하기 위해 최근 교육기관이 설립되고 있다. 정보 영재 교육은 주로 Basic, C, C++, Java를 가르치는 언어중심의 이벤트성 프로그래밍 교육이나 홈페이지 만들기 등과 같은 툴 위주의 수업으로 이루어지고 있다. 실제 영재 교육은 툴을 잘 다루는 학습에 초점을 맞추기보다는 주어진 문제를 해결하기 위한 창의력을 키울 수 있는 프로그래밍 로직 교육이 지속적으로 이루어져야 한다[2].

본 연구에서는 이러한 컴퓨터식 사고 능력을 개발하기 위해서 한국정보올림피아드 경시대회 초등부 전국대회 수준의 로직 문제를 해결해나가기 위한 교재를 개발하는 것을 목적으로 한다.

나) 연구의 방법

J.S.Bruner는 EIS 이론에서 어떤 연령의 아동이든지 간에 그 연령에 맞는 언어로 표현되기만 하면, 어떤 과제이든지 학습이 가능하다고 전제하고 있다. 그의 이론에 따르면 표상은 활동적 표상, 영상적(심상적) 표상, 상징적 표상의 3단계로 발달한다고 한다. 따라서 지식의 전달이 이러한 표상 발달 순서에 맞추어 적절히 제공되기만 하면 어느 연령에서든지 그것이 지도될 수 있다는 것이다[3].

따라서 본 연구에서는 컴퓨터 알고리즘을 표, 도식, 그림을 이용하여 영상적(심상적) 표상의 단계로 쉽게 설명할 수 있도록 하고 작은 영역에서 큰 영역으로, 낮은 수준에서 높은 수준으로의 단계적 절차를 가져 하나의 상징적 표상인 프로그래밍 언어를 이용하여 알고리즘을 구현할 수 있는 체계적인 교재 개발을 하는데에 그 목적이 있다.

다) 이론적 배경

(1) 한국정보올림피아드

한국정보올림피아드(KOI, Korea Olympiad in Informatics)는 행정안전부에서 주최하고 한국정보문화진흥원에서 주관하는 대회이다. 국내 최고의 IT영재들이 참가하여

실력을 겨루는 대회로써 국내 최고의 권위를 가진 대회이며 지역(시도)에서 선발된 학생들이 주어진 문제해결 능력을 겨루는 경시대회와 학생이 스스로 개발한 소프트웨어의 작품성을 평가하는 공모대회로 진행이 된다. 경시부분 우수 입상자에게는 국제정보올림피아드(IOI, International Olympiad in Informatics) 참가 후보자격을 부여하고 있으며 국내 주요대학 자체 입학전형에 의거, 정보화 특기생으로 선발되는 기회가 주어지고 있다[4].

경시대회의 주요 내용은 〈표 1〉과 같다[2].

〈표 1〉 경시대회 소개

- 개최일시
  - 매년 7월 초 ~ 7월 중순
- 출제방향
  - 수학적 지식 및 논리적 사고능력을 필요로 하는 알고리즘과 그 구현을 경시하는 문제출제 (IOI 출제경향)
- 경시내용
  - 각 분야별 수준에 맞는 문제해결 및 프로그램 작성 능력 평가
- 사용언어
  - Visual Basic, Visual C++

(2) 경시대회 출제 주요 알고리즘

(가) 시간복잡도 : 문제의 크기에 따라 걸리는 상대적인 시간 개념

- O-notation(big-Oh notation) : 함수가 들어가 최대에 비례하는 시간이 걸린다는 것을 의미

(나) 정렬 : 임의의 순서로 구성되어 있는 다수의 자료를 일정한 순서로 재배열함을 의미[5]

- 선택정렬, 버블정렬, 삽입정렬, 쉘정렬, 퀵정렬, 힙정렬, 병합정렬, 기수정렬

(다) 그래프 알고리즘

- 너비 우선 탐색, 깊이 우선 탐색, 최단 경로 찾기

(라) 욕심쟁이 알고리즘 : Greedy 알고리즘이라 하며 여러 선택에 있어서 앞으로의 선택을 고려하지 않고 각각의 선택마다 가장 최선의 선택을 하여 가장 최선의 선택을 하여 최적의 해를 찾기를 기대하는 알고리즘[6]

(마) 백트래킹 : 검색 공간의 모든 가능한 배치 방법에 대해 어떤 작업을 반복하기 위한 조직정직 방법[8]

(바) 분할정복 : 작은 사례에 대해서 해답을 바로 구할 수 있으면 바로 구하고 그렇지 않으면 다시 문제를 2개 이상으로 분할하는 알고리즘[7]

(사) 동적계획법 : 작은 부분 문제들의 해를 먼저 구하여 저장하고 더 큰 문제의 해를 구할 때 작은 문제의 해를 반복 계산하지 않고 저장된 결과를 사용하는 알고리즘[6]

Ⅱ. 본 론

1. 교재의 순서와 내용

본 연구에서 개발하고자 하는 교재는 한국정보올림피아드 경시대회 초등부 전국대회를 위한 것이므로 프로그래밍 언어에 대한 기초 사용법은 제외한다.

본 교재의 주요내용인 다양한 알고리즘에 대한 설명은 표, 도식, 그림을 이용하여 절차를 확인할 수 있게 하고 이를 숙지한 후 Visual C++ 프로그램에서 C언어로 코딩하는 과정으로 전개된다.

교재의 순서와 주요 내용은 〈표 2〉와 같다.

〈표 2〉 교재의 순서와 주요내용

구분	영 역	주 제	내 용
1	소 개	알고리즘	시간복잡도
2	정 렬	정렬의 종류와 특징	선택 / 버블 / 삽입 / 쉘 / 퀵 / 힙 / 병합
3	트 리	트리의 종류와 순회	이진 / 수식 / 최소비용신장 / 이진탐색트리, 힙, 허프만 코드
4	그 래 프	그래프의 종류와 탐색	너비 우선 / 깊이 우선 탐색, 최단경로 알고리즘
5	알고리즘	다양한 알고리즘	욕심쟁이 알고리즘
6			백트래킹
7			분할정복
8			동적계획법

2. 교재 개발의 예

⊙ 그래프의 탐색

그래프에서 모든 정점을 방문하는 것을 그래프의 탐색이라 하는데, 깊이 우선 탐색(DFS, depth first search)과 너비 우선 탐색(BFS, breadth first search)이 있다.

■ 깊이 우선 탐색(DFS)

깊이 우선 탐색은 주로 스택을 이용하여 구현한다. 직접 스택을 만들어 사용하기도 하며 재귀호출을 이용하여 시스템 스택을 이용하기도 한다. 흔히 깊이 우선 탐색을 백트래킹과 혼용하여 사용하기도 한다.

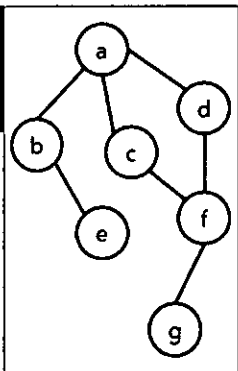
〈표 3〉은 깊이 우선 탐색의 원리를 순서대로 설명한 것이다[7].

〈표 3〉 DFS 알고리즘

① 시작 정점을 스택에 넣는다.
② 스택에서 정점 하나를 꺼내어 출력하고, 방문 정점을 확인한다.
③ 방문한 정점에서 직접 연결된 정점들 중 방문하지 않은 정점들을 스택에 차례로 넣는다. (여기에서는 인접한 정점이 복수 개일 경우에는 알파벳 내림차순으로 스택에 저장)
④ 모든 정점을 방문할 때까지 ②~③번 과정을 반복한다.

1단계

시작정점을 스택에 넣는다.



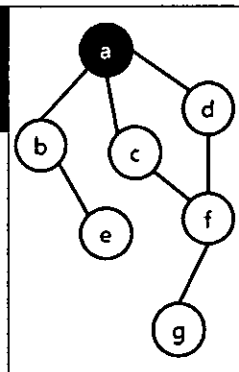
스택: a

방문정점 확인배열	a	b	c	d	e	f	g
	0	0	0	0	0	0	0

화면출력:

2단계

스택에서 자료(a)를 꺼내어 출력, a와 연결된 d, c, b를 스택에 입력



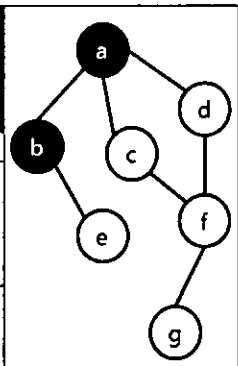
스택: b, c, a

방문정점 확인배열	a	b	c	d	e	f	g
	1	0	0	0	0	0	0

화면출력: a

3단계

스택에서 자료(b)를 꺼내어 출력, b와 인접한 e를 스택에 입력, 이미 방문한 a는 입력하지 않음



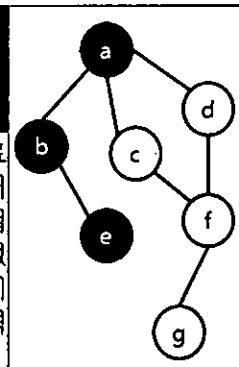
스택: e, c, d

방문정점 확인배열	a	b	c	d	e	f	g
	1	1	0	0	0	0	0

화면출력: b

4단계

스택에서 자료(e)를 꺼내어 출력, e와 인접한 정점이 없으므로 스택에 아무것도 입력하지 않음



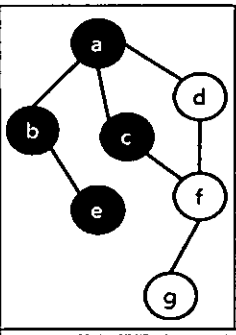
스택: c, d

방문정점 확인배열	a	b	c	d	e	f	g
	1	1	0	0	1	0	0

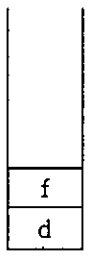
화면출력: e

6단계

스택에서 자료(c)를 꺼내어 출력, c와 인접한 f를 스택에 입력, 이미 방문한 a는 입력하지 않음



스택

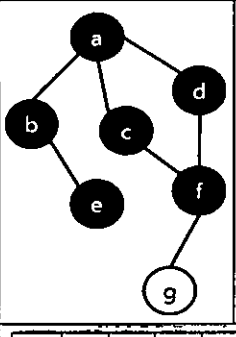


방문정점	a	b	c	d	e	f	g
확인배열	1	1	1	0	1	0	0

화면출력: c

7단계

스택에서 자료(d)를 꺼내어 출력, d와 인접한 a, f는 이미 방문했기 때문에 입력하지 않음



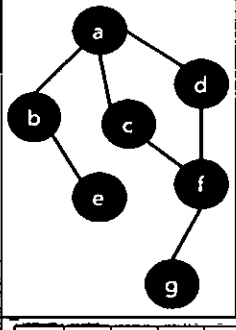
스택

방문정점	a	b	c	d	e	f	g
확인배열	1	1	1	1	1	1	0

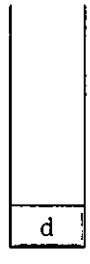
화면출력: d

8단계

스택에서 자료(g)를 꺼내어 출력, g와 인접한 f는 이미 방문했기 때문에 입력하지 않음



스택

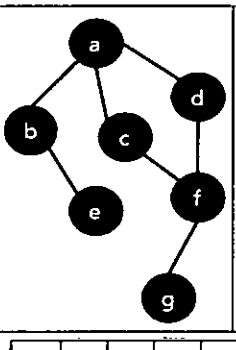


방문정점	a	b	c	d	e	f	g
확인배열	1	1	1	1	1	1	1

화면출력: g

9단계

스택에서 자료(d)를 꺼내어 이미 방문한 정점임을 확인하고 출력하지 않음



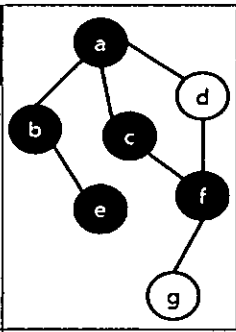
스택

방문정점	a	b	c	d	e	f	g
확인배열	1	1	1	1	1	1	1

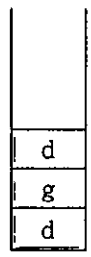
화면출력:

7단계

스택에서 자료(f)를 꺼내어 출력, f와 인접한 g, d를 스택에 입력, 이미 방문한 c는 입력하지 않음



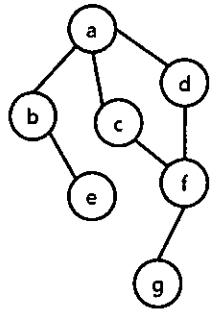
스택



방문정점	a	b	c	d	e	f	g
확인배열	1	1	1	0	1	1	0

화면출력: f

이러한 과정으로 깊이 우선 탐색의 진행 과정은 다음과 같다.

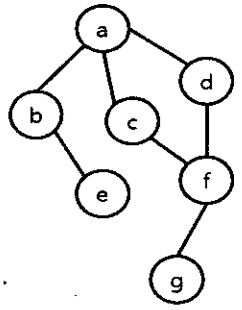


a - b - e - c - f - d - g

▶ 깊이 우선 탐색(DFS)의 구현

이를 C언어로 코딩하기 위해서는 먼저 그래프를 〈표 4〉와 같은 인접행렬로 만들어야 한다. 이 때 그래프의 정점간 연결이 될 경우에만 '1'로 표시한다.

〈표 4〉 인접행렬



	a	b	c	d	e	f	g
a	0	1	1	1	0	0	0
b	1	0	0	0	1	0	0
c	1	0	0	0	0	1	0
d	1	0	0	0	0	1	0
e	0	1	0	0	0	0	0
f	0	0	1	1	0	0	1
g	0	0	0	0	0	1	0

이를 2차원 배열로 나타내면 〈표 5〉와 같다.

〈표 5〉 그래프 G[v][i]

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

이를 바탕으로 〈표 6〉과 같은 소스를 만들 수 있다. 이 소스는 재귀호출을 이용하였다. 재귀호출은 스택 구조를 이용하게 되므로 굳이 스택을 구현할 필요는 없다.

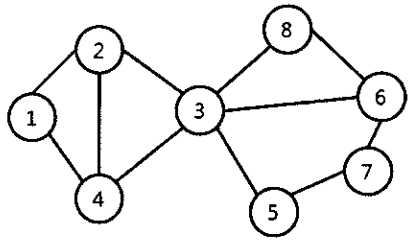
〈표 6〉 깊이 우선 탐색(DFS) 소스

```
#include <stdio.h>
#define MAX 8  // 정점은 7개, 0번째 배열을 합하여 MAX를 8로 지정

int G[MAX][MAX]={0};  // 인접행렬이 입력될 2차원 배열

// 정점 방문 확인 배열 : Check[MAX]
```

```
// 정점의 알파벳이 정수(integer)형으로 Check에 저장하여 알파벳도 출력할 수 있게 함
// 정점을 방문후에는 해당 배열 위치에 1을 저장
int Check[MAX]={0,'a','b','c','d','e','f','g'};

void DFS(int v){
      int i;

       if(Check[v]==1) return;  // 이미 방문하였으면 리턴

       printf("%c\t",Check[v]);  // 방문한 배열의 알파벳 출력

       Check[v]=1;  // 방문한 정점의 알파벳 출력 후 1을 저장

       // 방문한 정점과 연결된 정점 중 현재 방문하지 않은 정점을 찾아 재귀호출
      for(i=1;i<MAX;i++) if(G[v][i] && Check[i]!=1) DFS(i);
}

void main(){
      int i,j;

      // 사용자가 인접행렬 입력(올림피아드에서는 파일 입출력 사용함)
      for(i=1;i<MAX;i++)
           for(j=1;j<MAX;j++)
             scanf("%d",&G[i][j]);

      // 첫 방문 정점을 1(즉, Check[1]='a', 정점 a)를 방문하면서 DFS 함수 호출
      DFS(1);
}
```

▶ 확인문제

다음의 그래프에서 깊이 우선 탐색(DFS)으로 탐색한 결과가 아닌 것은?



① 1-2-3-8-6-7-5-4 ② 1-4-3-2-6-8-7-5 ③ 3-6-8-7-5-2-4-1

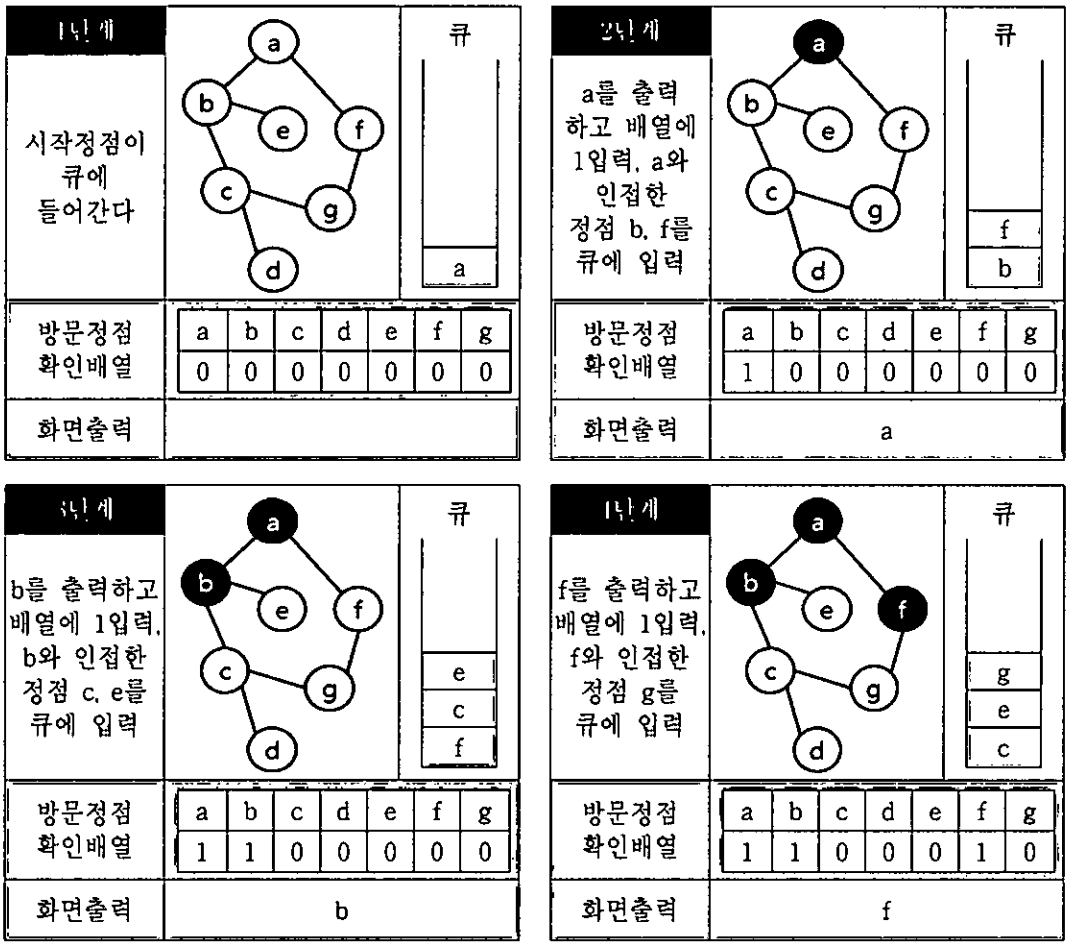
④ 2-1-3-6-8-7-5-4 ⑤ 6-8-3-5-7-2-1-4

너비 우선 탐색(BFS)

너비 우선 탐색은 깊이 우선 탐색과는 달리 큐를 이용한다. 너비 우선 탐색은 현 정점에서 제일 가까운 정점들을 먼저 탐색하고 점점 더 멀리 탐색하되 같은 기회를 모든 경우의 수에 분배해 나가며 가장 빠른 탐색을 할 수 있는 것에 초점을 맞춘다. 〈표 7〉은 너비 우선 탐색의 원리를 순서대로 설명한 것이다[7].

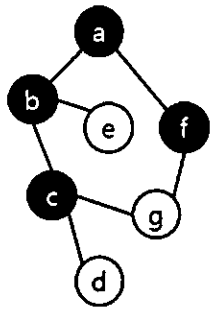
〈표 7〉 BFS 알고리즘

① 시작 정점을 큐에 넣는다.
② 큐에서 정점 하나를 꺼내서 출력하고, 방문 정점을 체크한다.
③ 출력한 정점에 직접 연결된 정점들 중 방문 체크되지 않은 정점들만 큐에 차례로 넣는다(여기서는 오름차순으로 넣는 것으로 정의).
④ 모든 정점이 방문체크 되지 않았으면 ②번 과정으로 간다.



5단계

c를 출력하고 배열에 1입력, c와 인접한 정점 d, g를 큐에 입력



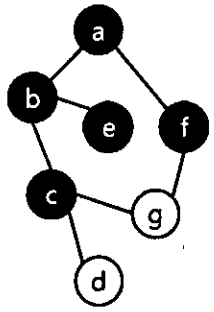
큐: g, d, g, e

방문정점	a	b	c	d	e	f	g
확인배열	1	1	1	0	0	1	0

화면출력: c

6단계

e를 출력하고 배열에 1입력, e와 인접한 정점 b는 이미 방문하여 큐에 입력하지 않음



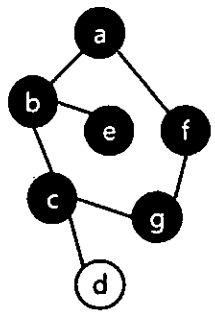
큐: g, d, g

방문정점	a	b	c	d	e	f	g
확인배열	1	1	1	0	1	1	0

화면출력: e

7단계

g를 출력하고 배열에 1입력, g와 인접한 정점 c, f는 이미 방문하여 큐에 입력하지 않음



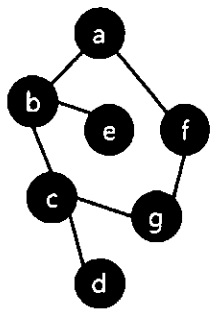
큐: g, d

방문정점	a	b	c	d	e	f	g
확인배열	1	1	1	0	1	1	1

화면출력: g

8단계

d를 출력하고 배열에 1입력, d와 인접한 정점 c, g는 이미 방문하여 큐에 입력하지 않음



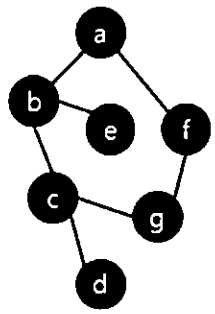
큐: g

방문정점	a	b	c	d	e	f	g
확인배열	1	1	1	1	1	1	1

화면출력: d

9단계

g를 큐에서 꺼냈지만 이미 방문했으므로 출력하지 않음

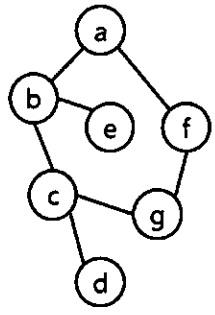


큐: (비어 있음)

방문정점	a	b	c	d	e	f	g
확인배열	1	1	1	1	1	1	1

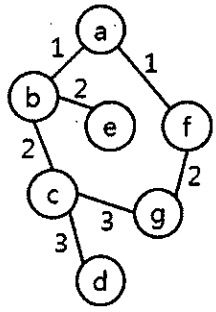
화면출력:

이러한 과정으로 깊이 우선 탐색의 진행 과정은 다음과 같다.



a - b - f - c - e - g - d

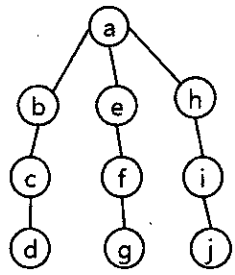
이러한 너비 우선 탐색은 〈그림 1〉로 알고리즘을 쉽게 이해할 수도 있다.



〈그림 1〉 BFS 탐색

첫 방문 정점인 a를 기준으로 다른 정점간의 거리를 1이라고 할 때 〈그림1〉과 같이 정점의 거리를 계산할 수 있다. 너비 우선 탐색은 깊이 우선 탐색과는 달리 같은 거리에 있는 정점을 순서대로 방문한다고 이해하면 쉽다.

〈그림 2〉와 같은 그래프에서는 깊이 우선 탐색(DFS)과 너비 우선 탐색(BFS)의 차이를 확실히 구별할 수 있을 것이다.



[DFS 탐색 순서] [BFS 탐색 순서]

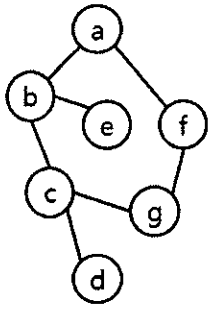
a b c d e f g h i j a b e h c f i d g j

〈그림 2〉 DFS와 BFS 탐색의 비교

▶ 너비 우선 탐색(DFS)의 구현

위의 알고리즘을 구현하기 위해서는 먼저 그래프를 〈표 8〉와 같은 인접행렬로 만들어야 한다. 이 때 그래프의 정점간 연결이 될 경우에만 '1'로 표시한다.

〈표 8〉 인접행렬



	a	b	c	d	e	f	g
a	0	1	0	0	0	1	0
b	1	0	1	0	1	0	0
c	0	1	0	1	0	0	1
d	0	0	1	0	0	0	0
e	0	1	0	0	0	0	0
f	1	0	0	0	0	0	1
g	0	0	1	0	0	1	0

이를 2차원 배열로 나타내면 〈표 9〉와 같다.

〈표 9〉 그래프 G[v][i]

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

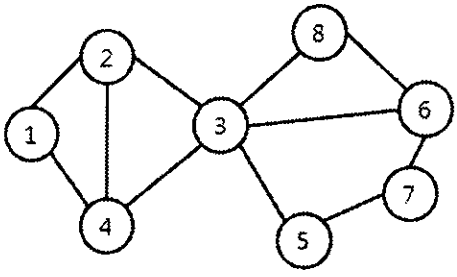
〈표 10〉 너비 우선 탐색(BFS) 소스

```
#include <stdio.h>
#define MAX 50 // 최대 정점이 50개까지 가능하도록 함
int G[MAX][MAX]={0}; // 인접행렬
int Check[MAX]={0}; // 정점 방문 확인 배열
int Que[(MAX+1)*MAX/2]={0}; // 큐로 사용될 배열, 최악의 경우(완전 그래프) 대비
int rows; // 몇줄 몇행인지 변수로 받음
int rear=-1,front=-1; // 큐의 입력, 출력 위치를 저장하는 변수

// 큐에 데이터 삽입시
void quePush(int vertexNo){
      front++;  // 함수가 호출될 때마다 출력 위치를 저장하는 front 변수가 1씩 증가
      Que[front]=vertexNo;  // 큐에 현재의 정점 번호 저장(예, a=1, b=2, c=3)
}
```

```
// 큐에서 데이터 출력시
int quePop(){
      int vertexNo;
      if(rear==front){  // rear와 front가 같은 경우는 큐에 자료가 없는 경우
            return 0;
      }else{
            rear++;
            vertexNo=Que[rear];
            if(Check[vertexNo]!=1){  //큐에서 꺼낸 정점이 미방문 정점일 경우
                  printf("%c",Check[vertexNo]);  // 방문 정점의 알파벳 출력
                  Check[vertexNo]=1;  //방문한 배열 위치에 1 저장
            }
            return vertexNo;  // 큐에서 꺼내진 정점을 반환
      }
}

void BFS(int vertexNo){
      int j;
      if(front==-1) quePush(vertexNo);  // 최초 함수 호출시 입력된 정점을 큐에 삽입
      while(vertexNo=quePop()){  // 큐에서 정점을 하나가 꺼내질 경우
            for(j=1;j<=rows;j++)
                  // 꺼내진 정점과 연결된 정점 중 미방문 정점을 큐에 삽입
                  if(G[vertexNo][j] && Check[j]!=1) quePush(j);
      }
}

void main(){

      int i,j;
      scanf("%d",&rows);

      for(i=1;i<=rows;i++)
            for(j=1;j<=rows;j++) scanf("%d",&G[i][j]);

      for(i=1;i<=rows;i++) Check[i]='a'+i-1;
        BFS(1);
}
```

▶ 확인문제

다음의 그래프에서 너비 우선 탐색(BFS)으로 탐색한 결과가 아닌 것은?



① 1-2-4-3-5-8-6-7 ② 1-4-2-3-8-5-6-7 ③ 3-2-4-5-8-1-7-6

④ 2-1-3-4-5-6-8-7 ⑤ 6-8-7-3-5-2-4-1

Ⅲ. 결 론

시중에 정보올림피아드를 준비하는 초등학생들이 구입하여 볼 수 있는 도서는 많지 않다. 특히 전국 본선대회를 겨냥한 도서는 더더욱 찾아보기 힘들다. 그래서 많은 학생들은 학원에서 경시대회를 준비할 수밖에 없으며 상당 부분을 학원 강사에게 의존하게 된다. 이런 이유에서 이 교재 개발로 인해 많은 초등학생들에게 더 많은 기회가 돌아갈 수 있으리라고 생각된다.

정보올림피아드 경시대회를 준비하다보면 알고리즘을 눈으로 보고 머리로만 이해하고 넘어가는 초등학생들이 많다. 하지만 실제 스스로 그림을 그려가며 코딩을 해보지 않으면 자신의 알고리즘이 되지 않는다.

따라서 본 교재에는 눈으로 쉽게 이해할 수 있는 알고리즘의 진행 절차뿐만이 아니라 스스로 코딩할 수 있고 배운 알고리즘을 어떠한 문제에 적용하여 사용할 수 있는지를 코드를 작성해가며 익힐 수 있도록 해야 할 것이다.

정보올림피아드 경시대회의 문제들은 단순히 어떠한 알고리즘을 외우고 있는지를 묻지 않는다. 논리적으로 사고하고 그것을 프로그래밍 언어로 표현할 수 있는지를 묻는 고차원적인 질문이다. 따라서 이러한 경시대회를 준비하는 교재를 개발하기 위해서는 단순한 따라하기 식의 나열보다는 그 원리를 설명하는 부분이 강조되어야 할

것이다.

마지막으로 본 교재가 한국정보올림피아드 경시대회를 준비하는 초등학생에게 실제 본선 경시대회의 성적 면에서 어느 정도의 효과를 주는 지에 대한 연구가 계속되어야 할 것이다.

《참 고 문 헌》

[1] 목영해, 디지털 문화와 교육, 문음사, 2001.

[2] 하성욱 외, 쉽게 배우는 실전 알고리즘 & 정보올림피아드 도전하기, 영진닷컴, 2002.

[3] 조한혁 외, 인터넷 상의 조작 도구를 이용한 수학교육 프로그램 개발(E-수학교육 논문집), 한국수학교육학회, 2002 .

[4] 한국정보화진흥원, http://www.nia.or.kr

[5] 김순근 외, 속전속결 알고리즘 입문, 영진닷컴, 2005.

[6] 이영호 외, 윌드와 함께하는 정보올림피아드 알고리즘 이론편, yeswoa 닷컴, 2004.

[7] 경상남도교육청, 정보올림피아드 및 프로그래밍 교육 교재(경남교육2008-015), 경상남도교육청, 2008.

[8] 스티븐 스키에나, 미구엘 레비야, Programming Challenges : 알고리즘 트레이닝 북, 한빛미디어, 2004.