https://doi.org/10.32431/kace.2019.22.3.005

프로그래밍 언어 학습 시스템에서 객관식 문제의 난이도 균등화 알고리즘에 대한 연구

김은정†

요 약

플립러닝 방식의 프로그래밍 언어 학습 시스템에서 사이버 강의에 대한 평가는 일반적으로 온라인에서 객관식 문제로 진행된다. 이때 출제되는 문제는 문제은행에서 랜덤하게 추출하여 학습자 개개인에게 주어진다. 이러한 평가 결과가 성적에 반영되기 위해서는 시험 문제의 형평성이 무엇보다 중요하다. 특히 프로그래밍 언어 과목에서는 문제의 유형에 따라 학습자가 생각하는 난이도가 서로 다를 수 있다. 본 논문에서는 객관식 문제의 유형을 2가지로 분류하여, 각 유형별로 난이도를 관리한다. 그리고 문제의 난이도와 유형을 함께 고려한 문제 출제 알고리즘을 제시하였다. 제시된 알고리즘은 프로그래밍 언어 과목의 특성을 고려할 때 기존의 출제 방식에 비해 보다 공정하고 효율적임을 실험을 통해 확인할 수 있었다.

주제어 : 사이버 강의, 플립 러닝, 자동 문제 출제 시스템

A Study on Difficulty Equalization Algorithm for Multiple Choice Problem in Programming Language Learning System

Eunjung Kim†

ABSTRACT

In programming language learning system of flip learning methods, the evaluation of cyber lectures generally proceeds from online to multiple choice questions. In this case, the questions are randomly extracted from the question bank and given to individual learners. In order for these evaluation results to be reflected in the grades, the equity of the examination question is more important than anything else. Especially in the programming language subject, the degree of difficulty that learners think can be different depending on the type of problem. In this paper, we classify the types of multiple-choice problems into two categories, and manage the difficulty level by each type. And we propose a question selection algorithm that considers both difficulty level and type of question. Considering the characteristics of the programming language, experimental results show that the proposed algorithm is more efficient and fair than the conventional method.

Keywords : Cyber Lecture, Flipped learning, Automatic Problem Selection Sytem

†정 회 원: 경성대학교 창의인재대학 교양학부 초빙교수
논문접수: 2019년 2월 15일, 심사완료: 2019년 5월 23일, 게재확정: 2019년 5월 27일

1. 서 론

4차 산업 혁명의 시대에 도래하여 소프트웨어의 중요성과 필요성이 커짐으로 인해 소프트웨어를 개발하는데 적절한 사고방식 즉 컴퓨팅 사고(computational thinking)에 대한 학습이 강조되고 있다. 이에 초중고에서 컴퓨팅 사고력 향상을 위한 프로그래밍 교육을 하기 시작했으며, 대학에서도 전교생에게 프로그래밍 교육을 하고 있는 추세이다. 컴퓨터 프로그램을 이용하여 어떤 문제를 해결하기 위해서는 주어진 문제를 정확히 이해하여 해결 방안을 세우고, 그 해결 방안을 알고리즘으로 기술하여 그것을 토대로 특정 언어의 문법으로 코딩하는 과정이 필요하다. 따라서 좋은 컴퓨터 프로그램을 만들기 위해서는 코딩은 마지막에 반드시 해야 할 필요조건이고, 그보다 더 중요한 것은 그 프로그램만이 가지고 있는 남다른 문제해결 방법이 있어야 한다. 이러한 컴퓨터로 해결할 수 있는 많은 일들의 문제 해결 방법을 찾는 과정 속에서 컴퓨팅 사고력을 향상 시킬 수 있을 거라고 기대하며 우리는 프로그래밍 교육을 하고 있다[1][2][3].

일반적인 컴퓨터 프로그래밍 교육은 특정 프로그래밍 언어를 바탕으로 오프라인 강의실에서 교수자 중심으로 이루어진다. 교수자가 프로그래밍 언어의 문법과 그 문법을 사용하여 해결할 수 있는 예제 프로그램을 설명하고, 학습자는 교수자의 설명에 따라 이해한다. 프로그래밍 학습의 효율을 향상시키기 위해서는 더 나아가 수업에서 배운 문법을 활용하여 학습자 스스로 새로운 문제를 해결하려는 노력과 연습이 필요하며, 해결하지 못한 문제는 교수자와 함께 방안을 찾아가는 과정을 학습하는 것이 중요하다. 하지만 제한된 시간과 공간에서 이루어지는 오프라인 수업에서는 현실적으로 교수자와 학습자가 이러한 과정을 함께 찾아가기에는 많은 어려움이 따른다.

프로그래밍 언어를 공부하는데 있어서 이러한 기존 오프라인 수업 방식의 문제점을 해결하기 위한 방안으로 활용되는 것이 플립 러닝(Flipped learning) 방식이다. 플립 러닝은 사이버 강의에서 강의 동영상으로 기본적인 문법과 그 문법을 사용한 예제 프로그램을 스스로 익히는 자기 주도적 학습을 수행한다. 그리고 오프라인 수업에서는 교수자와 함께 보다 확장된 새로운 문제를 해결할 수 있는 문제해결 역량 강화에 초점을 맞추어 수업을 진행한다[4][5][6]. 이러한 방식의 수업에서는 사이버 강의에 대한 평가를 온라인에서 객관식 문제로 진행하는 경우가 많으며, 이러한 평가 결과가 성적에 반영되기 위해서는 학습자 개개인에게 출제되는 평가 문제의 형평성이 무엇보다 중요하다고 할 수 있다. 일반적으로 사이버 학습 시스템에서 평가를 위해서는 문제 은행 방식을 많이 이용한다. 그리고 이러한 학습 시스템에서 효과적인 평가를 위해서 문제를 출제하는 방식과 문제의 난이도를 관리하는 방식에 대한 연구가 많이 있어왔다[7][8][9][10][11][12]. 사이버 학습 시스템에서 효율적인 평가를 위하여 주로 사용하는 문제 출제 방식은 교수자가 문제를 문제은행에 등록할 때 해당 문제의 난이도를 같이 입력하고, 이러한 난이도를 기반으로 문제가 자동으로 출제되도록 한다. 이는 학습자들의 학습 능력을 고려할 수 있고 보다 공정한 문제 출제가 가능하기 때문에 사이버 학습 시스템에서 많이 이용되고 있다. 하지만 프로그래밍 언어 과목의 학습 시스템에서는 객관식 문제의 유형에 따라 학습자가 느끼는 난이도가 서로 다를 수 있다. 따라서 기존의 출제 방식에 따라 문제의 난이도만을 고려하면 특정 유형의 문제가 편중되어 출제될 수 있고, 문제의 유형만을 고려하면 특정 난이도가 편중되어 출제될 수 있는 문제점이 있다. 이에 2가지를 모두 고려할 수 있는 새로운 문제 출제 알고리즘이 필요하다.

본 논문에서는 C 프로그래밍 언어 수업을 플립 러닝을 적용하여 수행할 때, 사이버 강의 평가를 위한 객관식 문제 출제 알고리즘에 초점을 맞추고자 한다. 일반적으로 온라인 평가는 채점도 온라인으로 수행되기 때문에 평가 문제는 사지선다형 또는 OX 문제로 구성되어져 있다. 프로그래밍 언어 과목의 특성상 문제의 유형은 크게 문법을 알고 있는지를 체크하는 문제와 프로그램 알고리즘을 분석하고 이해하여 수행 결과를 알아낼 수 있는지를 체크하는 문제로 나눌 수 있다. 이때 두 가지 유형에 대해서 교수자가 같은 난이도라고 판단한 문제일지라도 실제로 학습자들이 느끼는 난이도와는 차이가 있을 수 있다. 즉 올바른 문법을

판단하는 문제가 5개 출제된 경우와 알고리즘의 수행 결과를 판단하는 문제가 5개 출제된 경우는 제한된 문제 해결 시간 안에서 서로 공정하다고 단정할 수 없다. 이에 본 논문에서는 프로그래밍 학습 시스템에서 온라인 평가를 위한 객관식 문제를 자동 출제함에 있어서 예상 평균 점수에 맞는 난이도와 문제 유형을 함께 고려하는 문제 출제 알고리즘을 제시한다. 제시된 알고리즘은 프로그래밍 언어 과목의 특성을 고려할 때 난이도만을 고려하거나 문제의 유형만을 고려하는 출제 방식에 비해 보다 공정하고 효율적인 평가 문제를 출제할 수 있다.

2. 관련 연구

사이버 학습 시스템에서 평가를 위해 문제를 출제하는 방식에는 고정된 출제 방식이나 문제 은행을 이용한 무작위 출제 방식 또는 난이도에 기반한 자동 출제 방식이 있다.

먼저 고정된 출제나 문제 은행을 이용한 무작위 출제 방식에 대한 관련 연구는 다음과 같다.

[7]의 연구에서는 교수자가 문제에 대한 정보와 출제 기준을 정해진 형식에 맞추어 유형별로 서버에 저장해 둔다. 그리고 학습자가 응시할 때마다 요구하는 평가 형식으로 저장된 문제 정보를 불러와서 문제를 푼다. 공정하고 효과적인 평가를 위해서 각 학습자에게는 반드시 새로운 문제 유형이 출제된다. 이때 각 학습자에게 출제되는 서로 다른 유형의 문제는 유형별로 난이도가 서로 다를 수 있기 때문에 모든 학습자들에게 공정한 평가가 이루어진다고 단정할 수 없다.

[8]의 연구에서는 학습자들이 각 단원별 학습 후에 학습 진행 상태와 학업 성취도에 따라 수준별로 제시된 문제를 푼다. 여기서 수준별 문제는 학습자의 학습 진행 상태에서 랜덤하게 출제되는 문제를 풀고 다음 단계로 넘어가는 것이다. 이때 학습자별 문제의 수준에 대한 평가가 교수자의 임의 판단 기준에 따른 것이므로 객관적인 근거가 없고 학습자의 학습 진행 상태에 따른 각 단계별 수준에 대한 신뢰성이 떨어진다.

[9]의 연구에서 학습자가 스스로 필요한 문제를 출제하여 학습할 수 있으며, 평가를 위한 문제 출제 형식은 문제은행 방식을 이용한 동적 출제 방식을 사용한다. 학습자가 학습 결과를 즉각적으로 확인하여 직접 피드백 할 수 있으며, 모의고사를 통한 결과 분석에 따라 개인별 심화 및 보충 학습을 통해 완전 학습 목표에 도달할 수 있도록 하였다. 하지만 학습자 개인별로 동적 출제되는 문제의 난이도가 서로 다르기 때문에 완전 학습 목표에 도달했다는 의미에 신뢰성이 떨어진다.

다음으로 난이도에 기반 한 자동 출제 방식에 관한 관련 연구는 다음과 같다.

[10]의 연구에서는 학습자가 해당 교과에 대한 초기 문제풀이 과정에서 난이도별 출제 비율과 문제수를 직접 선택하도록 하고, 이후부터는 이전의 자료를 분석하여 각 단원별로 난이도를 고려하여 이를 반영한 문제를 자동으로 출제한다. 이는 누적된 분석 평가와 현재의 분석 평가를 통해 학습자의 수준에 맞는 문제를 자동 출제하므로 부족한 부분에 대하여 집중적 학습을 가능하게 한다.

[11]의 연구에서는 난이도에 따른 자동 문제 출제 시스템을 구현하였다. 여기서는 난이도를 5단계로 세분화하여서 처음 문제 은행에 문제를 출제할 때 해당 문제에 대한 난이도의 초기치를 출제자가 임의로 부여한다. 그리고 학습자의 평가 결과에 따라 난이도를 동적으로 재조정한다. 이러한 난이도를 기반으로 예상되는 평균 점수와 출제되는 문제수에 따라 정답률의 평균이 예상평균점수가 되도록 문제를 자동으로 출제한다. 본 논문에서의 난이도별 문제수 비율도 이 연구의 근거에 의하고 있다.

본 논문에서는 프로그래밍 언어 학습 시스템에서 평가를 온라인에서 객관식 문제로 진행하기 위한 공정한 문제 출제 알고리즘에 대하여 논하고자 한다. [7]의 연구를 기반으로 문제의 유형만을 고려하여 학습자 개개인에게 서로 다른 유형의 문제를 출제하면, 유형별 난이도가 서로 다를 수 있기에 공정한 문제 출제라 보기 어렵다. [11]의 연구를 기반으로 난이도만을 고려하여 해당 비율의 문제를 출제하면, 난이도의 객관성이 입증되었다 하더라도 특정 유형의 문제가 편중되어 출제될 수 있기에 문제 유형에 따른 균등한 출제는 이루어지지 않는다. 따라서 프로그래밍 언어 과목의 특성을 고려할 때 평가 문제의 난이도뿐만 아니라 평

가 문제의 유형도 함께 고려할 수 있는 새로운 문제 출제 알고리즘이 필요하다.

3. 평가 문제의 유형과 난이도에 따른 문제 출제 알고리즘

3.1 평가 문제의 유형

C 프로그래밍 언어의 객관식 평가 문제는 크게 문법적 에러 여부를 판단하는 문제와 프로그램 소스를 보고 흐름을 이해하여 실행 결과를 판단하는 문제로 이루어진다.

3.1.1 문법적 에러 여부를 판단하는 문제

문법적 에러 여부를 판단하는 문제는 크게 OX 문제와 사지선다형으로 나눌 수 있다. 2가지 유형에 대한 문제의 예는 다음과 같다.

<OX 문제>

- C 언어의 모든 명령문에는 마지막에 콜론(:)을 입력해야 한다. ---- (X)
- printf 함수는 출력할 내용을 “ ”로 묶어서 첫 번째 매개변수로 입력해야 하며 생략할 수 없다. ---- (○)

<사지선다형 문제>

- 다음 선언된 1차원 배열에 수행할 수 있는 명령어로 틀린 것은? (1)

```
int su[10] ;   int sum ;
```

① su = 10; ② su[0] = 20;
③ sum = su[1]+su[3]; ④ su[9] = su[5];

- 다음은 파일 입출력을 위한 명령 수행이다. 명령어에 대한 설명으로 틀린 것은 ? (2)

```
FILE  *fp = fopen(“성적.txt”, “ r ”);
```

① “성적.txt” 파일을 읽기용으로 열어서 파일 포인터 변수 fp에 연결하였다.
② “성적.txt” 파일이 존재하지 않을 경우에 새 파일을 생성한다.
③ “성적.txt” 파일에서 데이터를 입력받기 위해 fscanf 함수를 이용한다.
④ “성적.txt” 파일이 현재 실행 프로그램과 같은 폴더에 있어야 한다.

위의 예시된 문제들은 C 언어의 문법을 제대로 알고 있는 경우라면 직관적으로 답을 선택할 수 있다.

3.1.2 프로그램의 실행 결과를 판단하는 문제

프로그램의 소스를 보고 흐름을 이해하여 실행 결과를 판단하는 문제로 예시는 다음과 같다.

- 다음 프로그램의 실행 결과는? (2)

```
#include <stdio.h>
void main( )  {
  int a = 10 , b = 10 , c , d  ;
  c = a++ + 5 ;
  d = --b  ;
  printf("%d, %d, %d, %d ", a, b, c, d );
}
```

① 11, 9, 16, 9 ② 11, 9, 15, 9
③ 10, 10, 15, 10 ④ 10, 10, 16, 9

- 다음 프로그램의 실행 결과는? (1)

```
#include <stdio.h>
int  a , sum ;
void func( int a ) {
     a = a + 10;
     sum = sum + a ; }
void main( ) {
     int a = 10 ;
     func( a );
     a += 5 ;
     printf(" %d , %d ", a , sum );  }
```

① 15, 20 ② 15, 0 ③ 10, 10 ④ 10, 20

첫 번째 문제는 단항 연산자 (++ , --)의 선위형과 후위형 문법을 이해해야 하며 그에 따라 4개의 변수 값이 어떻게 변화하는지를 추적해야 답을 결정할 수 있다. 두 번째 문제는 지역변수와 전역변수에 대한 문법적 이해와 알고리즘의 흐름에 따라 변수 값의 흐름을 추적해야만 답을 결정할 수 있다. 이렇듯 예시에서 보이듯이 직관적으로 답을 찾을 수 있는 문법적 에러 여부를 판단하는 문제보다는 프로그램 소스를 보고 실행 결과의 답을 결정하는 문제가 더 많은 시간을 필요로 한다는

것을 예측할 수 있다.

3.2 데이터베이스 구조

온라인 학습 시스템에서 사용하는 전체 데이터베이스 중에서 본 논문에서는 C 언어의 객관식 문제 출제에 촛점을 맞추고자 하므로 시험 문제를 관리하는 [C언어문제관리] 테이블에 대한 구조만을 다루고자 한다. <표 1>에서 번호는 각 문제를 식별하기 위한 숫자이다. 단원은 문제가 속한 단원이 등록된다. 유형은 문법적 에러 여부를 판단하는 문제이면 'A'를 등록하고 프로그램 소스의 실행 결과를 판단하는 문제이면 'B'를 등록한다. 그리고 문제, 보기 4개, 정답을 입력한다. OX 문제의 경우에는 보기 4개는 비워두고 정답을 입력한다. 난이도는 최초 교수자의 판단에 의해 '상', '중' , '하' 로 구분하여 등록한다. 문제등록일은 최초 문제를 등록한 날짜를 입력한다.

<표 1> 'C언어문제관리' 테이블의 구조

필드명	데이터형식	비고
번호	int	각 문제를 식별하는 번호
단원	int	문제가 출제된 단원
유형	char	A(문법) / B(알고리즘)
문제	string	
보기1	string	
보기2	string	
보기3	string	
보기4	string	
정답	string	정답 표기
난이도	char	'상' or '중' or '하'
문제등록일	date	최초 문제 등록일

3.3 평가 문제 출제 알고리즘

평가를 위한 출제 알고리즘을 다음의 두 가지로 나누어 설명한다. 먼저 각 단원마다 수행되는 퀴즈 시험 유형의 단원별 평가 알고리즘과 중간 또는 기말시험과 같은 전체 단원에 대한 종합 평가 알고리즘으로 구분한다.

3.3.1 단원별 평가를 위한 출제 알고리즘

각 단원별 학습 후에 모든 학습자는 해당 단원에 속하는 문제를 대상으로 퀴즈 시험을 수행한다. 이때 학습자의 학습 능력에 따른 문제의 난이도와 문제의 유형을 함께 고려한 출제 알고리즘을 적용한다. 학습자들의 평균 점수에 대한 각 난이도별 출제 문제수 비율은 [11]의 연구에 근거하여 예상 평균점수에 따른 정답률별 문제수 비율에 따른다. [11]에서 난이도별 자동 문제 출제를 위하여 표준정규분포 방식에 의해 예상되는 평균점수에 따른 정답률별 문제수 비율을 구하였다. 예상되는 평균점수란 출제된 전체 문제들의 정답률의 평균이 예상평균점수가 되도록 출제한다는 의미이다. <표 2>는 [11]에서 구한 예상되는 평균점수에 따른 문제수 비율을 요약한 것이다. <표 2>에서 난이도 '상'은 아주 어려운 문제에 해당한다. '상중'은 비교적 어려운 문제로 분류하고, '중'은 어렵지도 쉽지도 않은 보통에 해당하는 문제로 분류한다. 그리고 '중하'는 비교적 쉬운 문제로 분류하고, '하'는 아주 쉬운 문제로 분류한다.

<표 2> 예상평균점수에 따른 정답률별 문제수비율(%)

난이도(정답률) / 예상평균점수	상	상중	중	중하	하
90점			7	24	69
80점		2	14	34	50
70점		7	24	42	27
60점	2	14	34	36	14
50점	7	24	38	24	7
40점	14	36	34	14	2
30점	27	42	24	7	
20점	50	34	14	2	
10점	69	24	7		

본 논문에서는 문제를 2가지 유형(문법 , 알고리즘)으로 나누어 각 유형별로 난이도를 관리한다. 유형별 난이도를 5개로 나누게 되면 총 10개의 난이도를 관리하게 된다. 단원별 평가와 같이 전체 시험 문제수가 10개 내외인 경우에는 난이도별 출제 문제수가 0~2개인 경우가 많으므로, 다시 이 개수를 유형별로 나누는데 어려움이 있다. 따라서 본 논문에서는 각 유형별로 난이도 '상', '중', '하' 3단계로 구분하고, '상'과 '상중'에 해당하는 문제를 난이도 '상' 문제에서 출제하고 '중하'와 '하'에 해당하는 문제를 난이도 '하' 문제에서 출제한다. 이때 서로 다른 유형에서 같은 난이도를 가진 문제에 대해 학습자들이 느끼는 어려움과 쉬움의 정

도가 서로 다르다는 것을 실험을 통해 확인할 수 있었다. 즉 같은 난이도 '상' 문제인 경우에 학습자들은 유형A(문법)보다는 유형B(알고리즘) 문제를 더 어려운 문제라고 생각하였다. 그리고 같은 난이도 '하'인 경우에 학습자들은 유형B(알고리즘)보다는 유형A(문법) 문제를 더 쉬운 문제라고 생각하였다. 여기에 대한 실험 결과는 4.1에서 설명한다. 이 실험을 바탕으로 난이도 '상' 문제에서는 유형B 문제를 유형A에 비해 더 어려운 문제로 간주하고, 난이도 '하' 문제에서는 유형A 문제를 유형B에 비해서 더 쉬운 문제로 간주하여 난이도별 문제수를 출제한다. <표 2>를 바탕으로 <표 3>은 '상'과 '상중'을 합치고, '중하'와 '하'를 합쳐서 총 10문제에 대해 예상평균점수 50점과 60점에 따른 난이도별 출제 문제수 비율을 계산한 결과이다.

<표 3> 난이도별 출제 문제수 비율(단원평가)

	예상평균점수	총 시험문제 (10문제)				
		상	상중	중	중하	하
출제비율(출제 문제수)	60점	2	14	34	36	14
		1.6 (2)		3.4(3)	5 (5)	
	50점	7	24	38	24	7
		3.1 (3)		3.8 (4)	3.1 (3)	

<표 4> 문제 유형별 출제 문제수(단원평가)

	예상평균점수	총 시험문제 (10문제)					
		상		중		하	
		A	B	A	B	A	B
출제비율(출제 문제수)	60점	87% (2)	13% (0)	33% (1)	67% (2)	28% (1)	72% (4)
	50점	77% (2)	23% (1)	50% (2)	50% (2)	23% (1)	77% (2)

<표 3>의 문제수 비율에 따른 유형별 출제 문제수는 <표 4>와 같다. 먼저, <표 3>에서 예상 평균점수 50점에 맞추어 출제할 경우에 전체 10문제 출제에 '상'/'상중' 3문제, '중' 4문제, '중하'/'하' 3문제가 출제된다. 여기서 '상'/'상중' 3문제를 살펴보자. 문제 3개중에서 실제로 '상' 과 '상중'이 차지하는 비율로 개수를 계산해 보면,

(상) 7 / 31 * 3 = 0.677 (1 개)

(상중) 24 / 31 * 3 = 2.32 (2 개)

이다. 그래서 난이도 '상' 문제는 유형B에서 1문제, 유형A에서 2문제를 출제한다. 난이도 '중'에 해당하는 문제수는 짝수인 경우에는 유형A와 유형B를 50%씩 출제하고, 홀수인 경우에는 균형을 맞추기 위해서 난이도 '상' 문제에서 개수가 적은 유형의 문제를 1개 더 출제한다. 난이도 '상'의 개수가 짝수이면, 난이도 '하' 문제에서 개수가 적은 유형의 문제를 1개 더 출제한다. 현재 출제 문제수가 4개이므로 각 유형에서 2개씩 출제한다. 그리고 '중하'/'하' 3문제 중에서 실제로 '중하'와 '하'가 차지하는 비율로 개수를 계산해 보면,

(중하) 24 / 31 * 3 = 2.32 (2 개)

(하) 7 / 31 * 3 = 0.677 (1 개)

이다. 그래서 난이도 '하' 문제는 유형B에서 2문제, 유형A에서 1문제를 출제한다.

다음으로 예상 평균점수 60점에 맞추어 출제할 경우를 살펴보자. 전체 10문제 출제에서 '상'/'상중' 2문제, '중' 3문제, '중하'/'하' 5문제가 출제된다. 여기서 '상'/'상중' 2문제의 비율을 계산하면,

(상) 2 / 16 * 2 = 0.25 (0 개)

(상중) 14 / 16 * 2 = 1.75 (2 개)

이다. 그래서 난이도 '상' 문제는 유형B에서 0문제, 유형A에서 2문제를 출제한다. 난이도 '중'은 문제수가 3개로 홀수이다. 이 경우에 난이도 '상'에서 유형B 문제가 출제수가 적기 때문에 '중' 문제에서 유형B를 1개 더 출제한다. 그리고 '중하'/'하' 5문제를 살펴보자. 문제 5개중에서 실제로 '중하' 와 '하'가 차지하는 비율로 개수를 계산해 보면,

(중하) 36 / 50 * 5 = 3.6 (4 개)

(하) 14 / 50 * 5 = 1.4 (1 개)

이다. 그래서 난이도 '하' 문제는 유형B에서 4문제, 유형A에서 1문제를 출제한다.

이에 대한 예상평균점수에 따른 자동 문제 출제 알고리즘은 [그림 1]과 같다.

3.3.2 종합 평가를 위한 출제 알고리즘

중간고사 또는 기말고사 유형의 여러 단원을 평가하기 위해서는 출제된 문제가 특정 단원에서 집중적으로 많이 출제되거나 또는 특정 단원이 출제되지 않는 경우가 없이 전반적으로 전체 학습 단원에서 골고루 출제되어야 한다. 이에 평가를 수행하는 전체 단원도 함께 고려되어야 한다.

먼저, 평가를 위한 30문제의 출제 문제수는 <표

```
function 문제출제(문제수, 예상평균점수)
 예상평균점수에 따른 난이도별출제개수 초기화
 난이도별출제개수(6,3)
       = '난이도' ,'유형' , '개수'를 초기화"
 for i = 1 to 6
          ' 출제난이도( 상A,상B,중A,중B,하A,하B)'
   출제난이도 = 난이도별출제개수(i,1)
   출제유형 = 난이도별출제개수(i,2)
   레코드셋 = select from C언어문제관리
              where 출제난이도 and 출제유형
   출제문제수 = 난이도별출제개수(i,3)
   count = 0
   do while(count < 출제문제수)
      난수(레코드번호)
          = int((총레코드개수-1 + 1) * rnd( ) + 1 )

      해당 레코드번호 문제를 사용자 시험DB에 출제
      count 를 1 증가
   loop
 end for
end function
```

[그림 1] 난이도&유형별 문제출제 알고리즘

2>를 바탕으로 계산한 <표 5>와 <표 6>에 따른다. 난이도와 유형에 따른 출제 문제수는 <표 3>과 <표 4>에서 설명한 계산 방법과 일치한다.

<표 5> 난이도별 출제 문제수 비율(종합평가)

	예상평균점수	총 시험문제 (30문제)				
		상	상중	중	중하	하
출제비율(출제문제수)	60점	2	14	34	36	14
		4.8 (5)		10.2(10)	15 (15)	
	50점	7	24	38	24	7
		9.3 (9)		11.4(12)	9.3 (9)	

<표 6> 문제 유형별 출제 문제수(종합평가)

	예상평균점수	총 시험문제 (30문제)					
		상		중		하	
		A	B	A	B	A	B
출제비율(출제문제수)	60점	87% (4)	13% (1)	50% (5)	50% (5)	28% (4)	72% (11)
	50점	77% (7)	23% (2)	50% (6)	50% (6)	23% (2)	77% (7)

다음으로 전체 단원 수를 고려하여 영역별 출제 문제수 비율을 정한다. 총 30문제를 출제함에 있어서 전체 단원 수에 대한 영역별 출제 문제수 비율은 <표 7>과 같다. <표 7>에서 총 단원수가 5장/6장일 경우에는 단원별로 6개/5개를 출제한다. 총 단원수가 7장이면 각 단원에서 4개씩 출제하고, 나머지 2개는 랜덤하게 임의 출제한다.

<표 7> 영역별 출제 문제수

총 단원수	출제 문제수 - 총시험문제(30문제)						
	1장	2장	3장	4장	5장	6장	7장
5장	6	6	6	6	6		
6장	5	5	5	5	5	5	
7장	4(+1)	4(+1)	4(+1)	4(+1)	4(+1)	4(+1)	4(+1)

이에 <표 5>, <표 6>, <표 7>를 기준으로 어느 하나의 단원에 치우치지 않도록 전체 영역에서 골고루 출제하는 자동 문제 출제 알고리즘은 [그림 2]와 같다. [그림 2]의 알고리즘에서 ①은 각 단원의 출제 개수를 초기화한다. ②는 단원 수만큼 배열을 잡아서 각 단원에서 출제된 문제의 개수를 체크한다. ③은 6개의 난이도/문제유형에 대해 반복 수행하며 문제를 출제한다. 이때 랜덤하게 선택한 레코드의 단원을 체크해서 현재 단원이 출제 가능한 경우이면 반복문을 빠져나가 사용자 시험DB에 문제를 등록한다. 그러나 현재 단원에서 출제 가능한 개수가 초과된 경우이면 계속해서 다음 레코드번호를 랜덤하게 추출한다.

```
function 문제출제(문제수, 예상평균점수, 단원수)
  예상평균점수에 따른 난이도별출제개수 초기화
  단원별 출제 개수 초기화                    ①
  난이도별출제개수(6,3) = '난이도','유형','개수'를 초기화"
  단원문제수(단원수) = {0}                    ②
  for i = 1 to 6  ' 출제난이도( 상A,상B,중A,중B,하A,하B)'
    출제난이도 = 난이도별출제개수(i,1)
    출제유형 = 난이도별출제개수(i,2)
    레코드셋 = select from C언어문제관리
               where 출제난이도 and 출제유형
    출제문제수 = 난이도별출제개수(i,3)
    count = 0
    do while(count < 출제문제수)
      do while(1)
        레코드번호=int((총레코드개수-1 + 1)*rnd() + 1)
        if (단원문제수(단원)<단원별출제개수 ) exit( ) ③
      loop
      해당 레코드번호 문제를 사용자 시험DB에 출제
      단원문제수(선택된 레코드의 단원)를 1 증가
      count 를 1 증가
    loop
  end for
end function
```

[그림 2] 영역별 문제출제 알고리즘

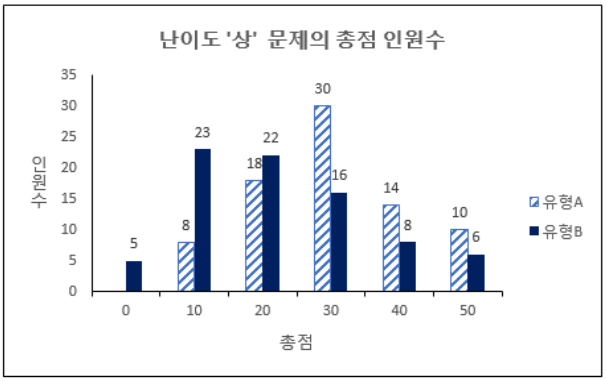
4. 실험 결과 및 분석

제시된 알고리즘의 효율성 분석을 위해 난이도만을 고려하는 방법과 문제의 유형만을 고려하는 방법 그리고 난이도와 문제 유형을 함께 고려하는 방법을 비교 분석하고자 한다. 먼저 난이도만을 고려하는 방법은 특정 유형의 문제가 편중되어 출제될 수 있으므로 프로그래밍 언어의 특성에 비추어 볼 때 효율적인 문제 출제 알고리즘이 될 수 없다. 따라서 본 논문에서 제안하는 알고리즘의 효율성 실험을 위해서 문제의 유형만을 고려하는 경우와 난이도와 유형을 함께 고려하는 경우에 한정하여 2가지 알고리즘을 실험하여 결과를 비교 분석하였다.

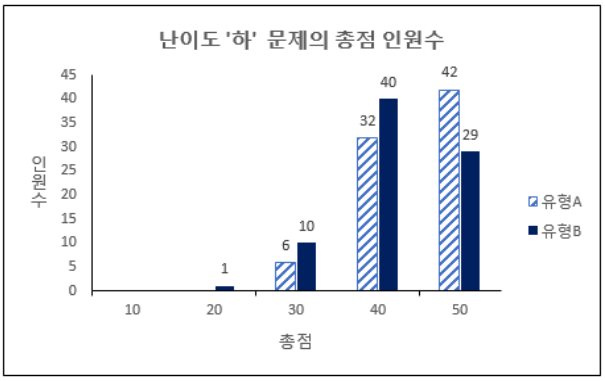
실험은 학부 1학년 학생들을 대상으로 운영되는 기초 컴퓨터 프로그래밍 수업에서 실행하였다. 현재 A대학에서 운영되는 기초 컴퓨터 프로그래밍 수업은 교양필수 과목으로 사이버 강의와 오프라인 강의가 함께 병행되어 운영되고 있다. 사이버 강의에서는 각 단원의 문법 및 기본적인 예제를 스스로 학습하는 과정이며, 오프라인 강의에서는 문제해결 역량 강화를 위하여 보다 확장된 예제를 같이 해결해 나가는데 초점을 맞추어 수업이 이루어진다. 사이버 강의에서는 각 단원별 수업이 끝나면 퀴즈 시험을 수행해야 하며 퀴즈 시험의 점수가 전체 성적에 포함되기 때문에 랜덤하게 출제되는 퀴즈 문제의 신뢰성과 객관성이 무엇보다 중요하다고 할 수 있다.

4.1 학습자들이 느끼는 유형별 난이도 분석

첫 번째 실험은 같은 난이도를 가진 유형A/유형B 문제에 대해서 학습자들이 느끼는 어려움과 쉬움의 정도를 실험하였다. 이를 위해 난이도 '상'에 해당하는 문제를 각 유형별 5개씩 10개, 난이도 '하'에 해당하는 문제를 각 유형별 5개씩 10개, 서로 다른 문제 4종류를 추출하였다. 추출된 문제를 오프라인 수업시간에 40명 인원인 2개의 분반에서 총 80명 학생들을 대상으로 난이도별로 구분하여 퀴즈 시험을 실시하고, 설문 조사도 함께 진행하였다. 한 분반에서 10명씩 같은 종류의 문제가 출제되었고, 학생들에게 퀴즈 시험의 용도를 설명하였다. 즉 퀴즈시험의 결과가 성적에 반영되지 않으며 중간/기말시험의 문제 출제를 위한 난이도 조사를 위한 것임을 명시하였다. 퀴즈 응시 시간은 제한을 두지 않았으며, 1문제당 10점을 부여하여 2가지 유형에 대해 50점 만점으로 총점을 계산하였다. 분석 결과는 [그림 3]과 [그림 4]와 같다.



[그림 3] 난이도 '상' 문제의 총점 분포 그래프

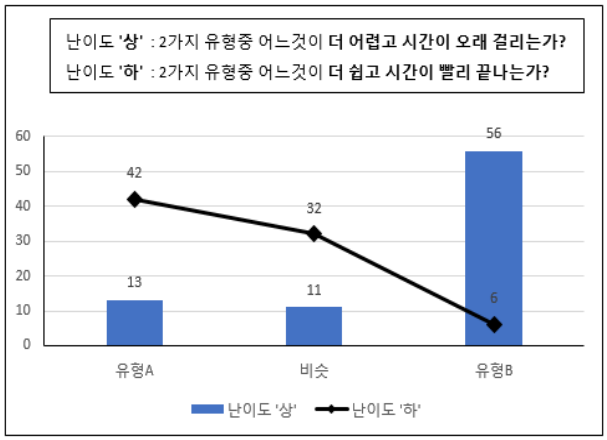


[그림 4] 난이도 '하' 문제의 총점 분포 그래프

[그림 3]은 유형A/유형B의 난이도 '상' 문제에 대한 퀴즈 응시 결과 80명 학생들의 총점 분포에 대한 그래프이다. 그림에서 보이듯이 유형A 문제는 총점 20점과 30점의 인원수가 가장 많았고, 유형B 문제는 총점 10점과 20점의 인원수가 가장 많았음을 확인할 수 있다. [그림 4]는 유형A/유형B의 난이도 '하' 문제에 대한 80명 학생들의 총점 분포 그래프이다. 난이도가 '하'인 문제에서는 유형A와 유형B 둘 다 대체적으로 40점과 50점의 인원수가 많음을 확인할 수 있다. 그러나 총점이 비슷해도 학생들이 느끼는 유형별 난이도는 서로 다를 수 있기에 설문을 하였다.

설문 내용은 난이도 '상'과 '하' 퀴즈 시험 마지

막 문제에서 유형A와 유형B 문제중에서 어느 유형이 더 어려운지와 쉬운지를 선택하도록 하였다. [그림 5]는 2가지 질문에 대하여 80명 학생이 선택한 응답 결과에 대한 그래프이다. 난이도 '상' 문제에서는 56명의 학생들이 유형B가 더 어렵다고 답하였고, 난이도 '하' 문제에서는 32명은 비슷하다고 했고 42명이 유형A가 더 쉽다고 답하였음을 알 수 있다. [그림 3], [그림 4]의 결과에서 많은 학생들의 총점이 같은 점수대임에도 학생들이 느끼는 실제 난이도는 많은 차이가 있었다. 이렇듯 유형A와 유형B에서 각각 어려운 '상' 문제가 출제되는 경우를 비교할 때, 제한된 응시 시간 안에 학습자가 느끼는 문제의 실제 난이도는 다를 수 있음을 예측할 수 있다. 따라서 문제의 유형만을 고려하여 학생 개개인에게 랜덤하게 문제를 출제하는 경우에는 난이도 측면에서 신뢰성과 객관성을 보장하기 어렵다.



[그림 5] 설문 조사 결과 그래프

4.2 문제 출제 알고리즘 비교 분석

다음으로 본 논문에서 제시하는 문제의 유형과 난이도를 함께 고려한 출제 알고리즘의 효율성을 실험하기 위하여 2가지 알고리즘을 적용하여 "연산자", "제어문" 단원에서 서로 다른 문제 4종류를 추출하였다. 추출된 문제를 오프라인 수업시간에 40명 인원인 2개 분반에서 80명 학생들을 대상으로 퀴즈 시험을 2주 동안 2회 실시하였다. 한 분반에서 10명씩 같은 종류의 문제가 출제되었고, 학생들에게 퀴즈 시험의 용도를 설명하였다. 즉 퀴즈 시험의 결과가 성적에 반영되지 않으며 개인별 학습 능력과 응시 시간을 분석하여 중간고사 시험 문제 출제에서 문제의 유형과 난이도를 조정하기 위한 것임을 명시하였다. 그리고 시험 결과 분석에서는 같은 총점을 받은 학생들도 하나의 문제에 대해 실제 느끼는 난이도는 서로 다를 수 있기 때문에 전체 80명의 학생들이 총 10문제의 퀴즈 시험에 소요한 시간이 얼마나 차이가 나는지를 비교 분석하였다. 첫 번째 주는 (알고리즘 1)의 추출 문제로 퀴즈 시험을 실시하였고, 두 번째 주는 (알고리즘 2)의 추출 문제로 퀴즈 시험을 실시하였다. 출제 문제수는 10문제이고 퀴즈 응시 시간은 10분으로 제한하였다. 문제를 다 푼 학생은 바로 제출하고 강의실을 나갔으며 10분을 초과한 경우에는 즉시 시험을 멈추고 문제지를 제출하였다. 시험 시작과 동시에 스톱워치를 시작하고, 학생이 자리에서 일어서는 순간의 시간을 기록하였다. 그리고 시험 응시 결과 학생들에게 출제된 평가 문제의 분포도 및 시험 응시에 소요된 시간을 비교 분석하였다.

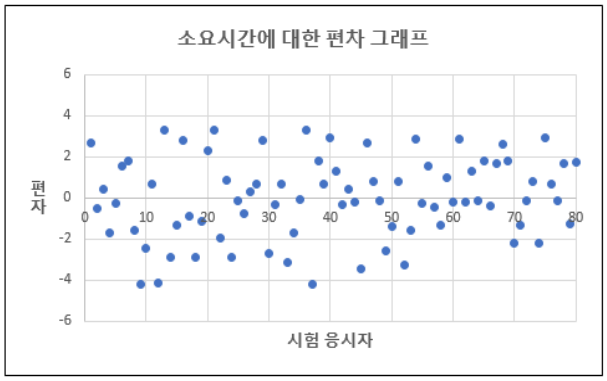
(알고리즘 1) 문제의 난이도는 고려하지 않고 유형 A 5개와 유형B 5개를 랜덤하게 추출하였다. <표 8>은 응시 학생 80명중 10명에게 출제된 문제를 분석한 결과이다.

<표 8> 알고리즘 1의 출제 현황 예시

문제유형	유형 A			유형 B			소요 시간 (종료일시 - 시작일시)
난이도	상	중	하	상	중	하	
학생 1	3	2	0	4	0	1	11분 32초
학생 2	1	1	3	0	3	2	8분 17초
학생 3	2	2	1	0	2	3	9분 10초
학생 4	1	1	3	1	3	1	7분
학생 5	4	0	1	1	2	2	8분 40초
학생 6	2	3	0	3	2	0	10분 25초
학생 7	0	4	1	4	1	0	10분 48초
학생 8	2	2	1	0	2	3	7분 10초
학생 9	1	1	3	1	0	4	4분 48초
학생 10	1	0	4	1	1	3	6분 20초

<표 8>에서 문제 유형별 개수는 일치하지만 서로 다른 난이도의 문제가 랜덤하게 출제되었음을 확인할 수 있다. 결과에서 보이듯이 학생1, 학생6, 학생7의 경우에는 유형B에서 난이도 '상'인 문제가 많이 출제된 경우로 전체 응시 시간 10분을 다 사용한 반면, 대체적으로 2가지 유형에서 '하' 문제가 많이 출제된 학생9, 학생10의 경우에는 소요

시간이 짧음을 확인할 수 있다. [그림 6]는 전체 80명 학생들의 시험 응시 시간이 평균 응시 시간으로부터 얼마나 떨어져 있는지에 대한 편차를 구하여 그래프로 나타낸 것이다. 그림에서 보이듯이 80명 학생의 응시 시간이 평균으로부터 많이 분산되어 분포되어 있다는 것을 확인할 수 있다. 따라서 난이도를 고려하지 않고 시험 유형만을 고려하여 랜덤하게 출제되는 문제는 프로그래밍 언어의 특성상 시험 응시에 소요되는 시간에 대한 형평성 측면에서 효율적인 문제 출제라고 보기 어렵다.



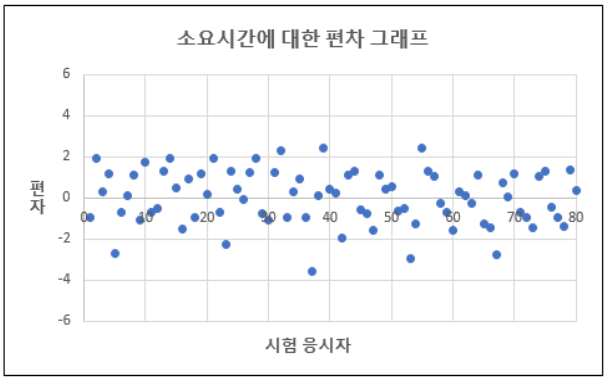
[그림 6] 알고리즘 1 의 퀴즈 소요 시간 그래프

(알고리즘 2) 본 논문에서 제시하는 알고리즘을 이용하여 예상 평균점수 50점에 맞춘 난이도를 적용하여 문제를 추출하였다. <표 9>는 전체 응시 학생 80명중 10명에게 출제된 문제를 분석한 결과이다. 결과에서 보이듯이 모두 같은 유형과 같은 난이도를 가진 문제가 출제되었고, 소요시간도 많은 차이를 보이지 않고 있음을 확인할 수 있다.

<표 9> 알고리즘 2의 출제현황 예시

예상평균점수 50점의 출제 문제수							소요 시간 (종료일시 - 시작일시)
난이도	상		중		하		
문제유형	A	B	A	B	A	B	
학생 1	2	1	2	2	1	2	8분 12초
학생 2	2	1	2	2	1	2	11분
학생 3	2	1	2	2	1	2	9분 34초
학생 4	2	1	2	2	1	2	10분 23초
학생 5	2	1	2	2	1	2	6분 40초
학생 6	2	1	2	2	1	2	8분 36초
학생 7	2	1	2	2	1	2	9분 20초
학생 8	2	1	2	2	1	2	10분 16초
학생 9	2	1	2	2	1	2	8분 50초
학생 10	2	1	2	2	1	2	10분 8초

[그림 7]의 전체 80명 학생의 시험 소요시간에 대한 편차 그래프에서도 (알고리즘 1)에 비해서 전체 학생들이 평균에 가깝게 분포되어 있음을 확인할 수 있다. 이는 제한된 시간 안에 시험에 응시하는 학습자들에게 서로 다른 문제가 출제되는 특징에 비추어 볼 때, 학생들이 실제 느끼는 난이도 측면에서 보다 효율적인 출제 알고리즘임을 확인할 수 있다.



[그림 7] 알고리즘 2 의 퀴즈 소요 시간 그래프

5. 결론 및 향후 연구과제

플립러닝 방식의 프로그래밍 언어 학습 시스템에서 사이버 강의에 대한 평가를 온라인에서 객관식 문제로 진행하는 경우에 출제되는 평가문제는 일반적으로 문제 은행에서 랜덤하게 추출하여 학습자 개개인에게 서로 다른 문제가 주어진다. 이러한 평가 결과가 성적에 반영되기 위해서는 출제되는 평가 문제의 난이도 측면에서의 형평성이 무엇보다 중요하다고 할 수 있다. 이에 본 논문에서는 객관식 문제의 유형을 프로그래밍 언어 과목의 특성을 고려하여 2가지 유형으로 구분하였다. 서로 다른 유형에서 같은 난이도를 가진 문제라 하더라도 학습자들이 느끼는 난이도는 다를 수 있다. 따라서 문제를 출제함에 있어서 문제의 난이도와 유형을 함께 고려하는 객관식 문제 출제 알고리즘을 제시하였다. 제시된 알고리즘은 프로그래밍 언어 과목의 특성을 고려할 때 문제의 난이도만을 고려하거나 문제의 유형만을 고려한 출제 방식에 비해 보다 효율적임을 실험을 통해 확인할 수 있었다.

향후 연구 과제로는 각 유형별 문제의 난이도를 효율적으로 관리하는 부분에 대한 연구가 필요하

다. 본 논문에서는 최초 문제의 난이도가 교수자의 판단 기준에 의해 주어진다. 이러한 난이도가 보다 객관적인 신뢰성을 유지하기 위해서는 학습자들의 평가 결과를 기반으로 각 문제의 난이도를 재조정할 필요가 있다. 동적으로 난이도를 재조정하는 [12]의 연구에서는 개인 또는 집단 학습자들의 평가 후 평균 점수를 기반으로 문제의 쉬운 정도와 어려운 정도를 달리 측정한다. 그리고 여기에 기초하여 각 문제의 정답률과 오답률을 이용한 난이도 비율을 계산하고 이를 이용하여 문제의 난이도를 동적으로 관리한다. 하지만 본 논문의 실험에서 확인했듯이 프로그래밍 언어 과목의 문제 특성상 평가 결과에 따른 평균 점수뿐만 아니라 각 문제를 해결하는데 소요된 시간도 해당 문제의 난이도에 영향을 미칠 수 있다. 따라서 보다 효율적인 객관식 문제 은행의 난이도 관리를 위해서는 학습자들의 평가 결과를 기반으로 한 정답률과 오답률뿐만 아니라 각 문제를 해결하는데 소요된 시간까지 함께 고려한 동적 난이도 재조정에 대한 연구가 필요하다.

참 고 문 헌

[1] 윤회원 · 최성욱 (2017). 효율적인 프로그램 개념 학습을 위한 스크래치 수업 설계. **융복합지식학회 논문지**, 5(2), 57-62.

[2] 안창호 · 이기쁨 · 문석재 (2018). 초보학습자를 위한 엔트리 블록/텍스트 코딩 기반의 프로그래밍 학습방법. **융복합지식학회 논문지**, 6(1), 127-134.

[3] 우여명 외 4 (2014). 웹 기반 C 프로그래밍 언어 학습 시스템의 설계 및 구현. **정보과학회 컴퓨팅의 실제논문지**, 20(12), 640-645.

[4] 최숙영 (2017). 프로그래밍 수업의 플립드러닝 학습모형 설계 및 적용. **한국컴퓨터교육학회 논문지**, 20(4), 27-36.

[5] 이지연 · 김영환 · 김영배 (2014). 학습자 중심 플립드러닝 수업의 적용 사례. **교육공학연구**, 30(2), 163-191.

[6] 이동엽 (2013). 플립드러닝 교수학습 설계 모형 탐구. **디지털융복합연구**, 11(12), 83-92.

[7] 최돈은 외 3 (2000). 동적인 문제 출제 시스템의 설계 및 구현. **한국정보과학회 학술발표논문집**, 27(1B), 690-692.

[8] 임희숙 (1999). 웹기반 지능형 문제은행 시스템의 설계및구현. **석사학위논문, 전남대학교.**

[9] 이민경 · 강수용 (2006). 웹기반 수준별 학습을 고려한 문제은행 시스템의 설계 및 구현. **한국정보과학회 학술발표논문집**, 33(2A), 103-107.

[10] 이현주 외 4 (2003). 학습자의 수준평가를 이용한 웹 기반 자동 문제 출제 시스템. **정보처리학회 논문지A**, 10(5), 579-588.

[11] 김경아, 최은만 (2002). 웹기반교육에서의 자동 문제 출제 시스템. **한국정보처리학회 논문지A**, 9(3), 301-310.

[12] 김은정 · 이상관 · 김성곤 (2008). 이러닝 문제은행기반 출제시스템을 위한 동적 난이도 조정 정책. **한국정보통신학회 논문지**, 12(12), 2232-2238.

김 은 정

1996 국립 경상대학교 전자계산학과 (공학석사)
2001 국립 경상대학교 전자계산학과 (공학박사)
1989 ~ 1993 (주)LG전자 멀티미디어연구소 연구원
2000 ~ 2003 부산외국어대학교 전자컴퓨터공학부 비정년 전임강사
2003 ~ 2005 부산외국어대학교 교양학부 초빙교수
2008 ~ 2010 부산가톨릭대학교 컴퓨터공학과 비정년 전임강사
2010 ~ 현재 부산대학교 교양교육원 시간강사
2019 ~ 현재 경성대학교 교양학부 초빙교수
관심분야: 사이버강의, 플립러닝, 프로그래밍교육
E-Mail: kimeunjung@pusan.ac.kr