한국컴퓨터정보학회지
제22권 제2호, 2014. 12.

프로그래밍 경시대회 문제를 이용한 알고리즘 지도방법 제안
- 과학고등학교 사례를 중심으로 -

• 장원영 (충북교육정보원)
• 김성식 (한국교원대학교 컴퓨터교육과)

I. 서론

교육부는 2014년 9월에 문·이과 통합형 교육과정의 총론을 고시하였다. 지금까지 학교 현장에 남아있던 문·이과 칸막이를 없애고 모든 학생들이 인문, 사회, 과학기술에 대한 기초 소양을 함양하여 인문학적 상상력과 과학기술 창조력을 갖춘 창의 융합형 인재로 성장할 수 있도록 우리 교육을 근본적으로 개혁하자는 취지가 담겨 있다. 그리고 그 실천 방안의 중심에는 인문사회적 소양 교육과 과학기술 소양 교육의 강화가 있다. 더불어, 과학기술 소양의 하위 요소로 SW 교육이 확대, 개편될 예정이다. 이와 관련하여 문이과 통합형 교육과정에서 학교급별 주요 교과의 변화는 표 1과 같다.

표 1. 문이과 통합형 교육과정의 주요 교과 변화

학교급	주요 개정 내용
초	· 실과 교과의 대단원으로 SW 기초소양 중심교육으로 개편하고, 17시간 이상 필수 이수
중	· 정보 교과 34시간 이상 필수 이수
고	· 통합사회, 통합과학의 신설 · 심화선택인 정보 교과를 일반선택으로 전환 (고등학교에서 선택 기회 높아짐)

이번에 고시된 총론을 기초로 교과 교육과정의 각론이 개발되어 2015년 9월에 고시될 예정이다. 그리고 모든 정보 교과(단원)의 내용은 SW (원리) 교육 중심으로 변화될 예정이다. 여기서, 총론과 최근 언론에서 이야기하는 SW교육, 또는 코딩교육이라 함은 SW 원리와 코딩을 통해 정보과학적 사고력(Computational Thinking)을 함양하는 교육을 의미한다.

상용 프로그램의 활용 교육에 치우쳤던 7차 교육과정 이후 정보 교과의 내용과 이수 시간에 있어서 이번 문·이과 통합형 교육과정처럼 위상이 높았던 적은 없을 것이다. 따라서 이제는 정보 교과의 각론에 포함될 내용 요소, 교수학습 방법, 평가 등에 대해 심도 있게 논의하고, 정보 교과의 가치를 재정립할 시기라고 할 수 있다. 또한, SW 교육에 있어서 초·중·고의 교육과정과 대학 교육의 연계 방안을 고려해야 할 시기다. 초등학교와 중학교에서 정보 교과가 필수로 지정된 만큼 이러한 초석은 이미 마련된 셈이다.

2009 개정 교육과정에서 정보 교과의 내용은 2007 개정 교육과정과 유사한 정보과학적 원리를 이용한 일상생활의 문제해결에 초점이 맞추어져 있다. 그러나 초등학교와 중학교에서 정보 교과가 필수로 지정된 만큼, 고등학교에서는 일상생활의 문제해결을 뛰어 넘는 정보과학의 원리에 집중해야 한다. 그래야 초, 중, 고 교육과 대학 교육이 서로 위계성을 갖고 연계될 수 있기 때문이다. 학생들이 미적분을 어려워한

다고 해서 고등학교의 교육과정에서 해당 내용을 삭제할 수 없는 것처럼, 고등학교의 정보 교과 또한 내용상 초등학교와 중학교에서 학습한 내용을 토대로 심화된 내용이 포함되어야 할 것이다. 이러한 관점에서 본고에서는 프로그래밍 경시대회 문제를 이용한 알고리즘의 지도 방법을 제안하고자 한다. 프로그래밍 경시대회 문제를 이용하는 이유는 학습자에게 알고리즘과 프로그래밍에 대한 흥미와 동기를 부여하고, 주어진 문제에 맥락화하여 교육하는데 의미가 있다. 이는 과학고등학교의 사례를 기초로 하지만, 문제의 난이도를 조정하여 일반계 고등학교에 적용할 수 있으며, 대학교의 기초 프로그래밍 과정과 알고리즘 등의 강좌에서도 적용할 수 있을 것이다.

II. 이론적 배경

경시대회 문제를 통한 프로그래밍 교육 활성화 방안에 대해서 다양한 논의가 이루어지고 있다. 임형석, 김희철[1]은 경시대회가 대학교육에 기여하는 바를 분석하였는데, '경쟁을 통한 도전의식 함양', '창의적 문제 해결 능력 신장', '협동적 팀원 역할을 할 수 있는 능력 양성' 등이 그것이다. 또한, 조환규[2]는 프로그래밍 경시대회가 컴퓨터 과학의 제반 요소를 가르치고 평가하는데 매우 중요한 한 방법 중 하나이며, 비록 전부는 아닐지라도 이를 적절히 활용하면 큰 교육적 효과를 거둘 수 있다고 평가하고 있다. 최근에는 프로그래밍 경시대회 문제를 교육 현장에서 활용하기 위한 '알고리즘 자동 평가 시스템'에 대한 여러 연구들이 진행되고 있다.

1. 프로그래밍 경시대회 문제의 교육적 가능성

알고리즘 교육에 있어서 프로그래밍 경시대회 문제는 다음의 교육적 가능성을 갖고 있다.

1.1 목표 지향성

일반적인 알고리즘 학습은 현재 학습 중인 알고리즘의 이해를 목표로 한다. 따라서 해당 알고리즘을 이해하고 나면 다양한 문제에 적용해 보는 것이 아니라 새로운 알고리즘에 대한 학습으로 진행한다. 이러한 상황에서 학습자는 자칫 수동적이고 맹목적으로 수업에 참여하게 될 수 있다. 알고리즘은 주어진 문제를 효율적으로 해결하는 데 의미가 있다. 따라서 학습한 알고리즘을 다양한 문제에 적용해 보는 것은 매우 중요한 과정이다. 이러한 과정을 통해 해당 알고리즘의 이해에 그치지 않고 주어진 문제를 정해진 시간 내에 해결하기 위한 알고리즘의 설계와 구현 능력을 함양하게 된다.

1.2 경쟁심과 흥미 유발

알고리즘 자동 평가 시스템을 이용하면 학습자들이 채점한 결과를 토대로 실시간으로 랭킹이 산출되므로 동료들 간에 경쟁적 분위기가 형성되어 알고리즘과 프로그래밍 학습을 촉진한다. 다만, 과도한 경쟁 상황은 오히려 부진 학생의 학습 의욕을 저하시키므로 주의해야 할 것이다.

1.3 알고리즘의 맥락화

학습자에게 제한 시간 내에 주어진 문제를 효율적으로 해결해야 하는 문제 상황을 제시함으로써 이 문제에 적용 가능한 다양한 알고리즘을 고안하게 된다. 이러한 과정에서 알고리즘에 대한 단순한 지식 습득에만 그치는 것이 아니라 창의적이고 심층적으로 깊이 있게 사고하고 자신이 이해한 알고리즘 지식을 활용하는 경험을 갖게 된다.

2. 알고리즘 자동평가 시스템

위에서 열거한 프로그래밍 경시대회 문제의 교육적 가능성을 극대화하기 위해서는 알고리즘 자동평가 시스템이 요구된다. 알고리즘 자동평가 시스템은 학습자가 제출한 프로그램의 정확성과 시간효율성을 자동으로 평가해 주는 시스템이다. 시스템은 학습자가 프로그래밍 언어를 이용하여 주어진 문제에 대한 풀이를 작성하여 제출하면 제출된 소스 코드는 서버에서 파일로 저장되고 이를 컴파일 하여 실행파일로 만든다[3]. 실행프로그램은 문항마다 미리 준비된 입력, 출력 파일의 수에 따라 몇 차례 실행을 반복한다. 매 실행시마다 미리 준비된 데이터가 입력되어 프로그램이 실행되고 그 결과 출력되는 값과 준비된 정답데이터의 값을 비교한다. 만약 두 데이터의 값이 같고, 시간제한, 메모리 제한을 위반하지 않았다면 해당 입력데이터에 대해서는 프로그램이 제대로 동작한 것으로 받아들여진다. 만약 프로그램이 실행되는 동안에 어떠한 오류가 발견되었다면 시스템은 Wrong Answer, Time Limit Exceed, Memory Limit Exceed, Runtime Error, Compile Error 중에서 해당되는 오류 메시지로 피드백 정보

를 제공한다[4]. 알고리즘 자동평가 시스템은 일반적으로 표 2의 4가지 구성요소로 이루어져 있다[5].

표 2. 알고리즘 자동평가 시스템 구성요소

구성요소	관련 설명
① 문제	학습자가 해결해야 할 알고리즘 문제
② 채점데이터 세트	해당 문제에 대한 알고리즘의 정확성과 시간효율성을 측정하기 위한 일련의 채점데이터 세트(입력데이터와 정답데이터의 쌍으로 구성)
③ 자동평가 프로그램	학습자가 제출한 소스코드를 컴파일한 후 생성된 실행파일에 채점데이터를 입력하고 출력값을 정답데이터와 비교하며, 실행파일의 소요시간을 측정하는 프로그램
④ 사용자 서비스 환경	문제의 제시, 자동 채점 결과, 랭킹 산출 등의 서비스를 제공하는 사용자 환경

현재 국내에서 현직 교사들이 운영되고 있는 알고리즘 자동평가 시스템은 표 3과 같다[6]. koistudy와 codingfun은 교사가 자체 개발하여 운영 중이며, codeup과 judgeon은 온라인 저지 오픈소스 hustoj를 활용하였다. codingfun은 클라이언트-서버 기반으로 운영되는 점이 다른 온라인 저지 방식과 다르다.

표 3. 국내 현직 교사가 운영하는 알고리즘 자동평가 시스템

이름	운영자	주소	특징
koistudy	영재고 교사	koistudy.net	무료, 공교육
codeup	특성화고 교사	codeup.kr	무료, 공교육
judgeon	과학고 교사	judgeon.net	무료, 공교육
codingfun	과학고 교사	codingfun.net	무료, 공교육

Ⅲ. 프로그래밍 경시 문제를 이용한 알고리즘 지도방법

한국정보올림피아드의 기출문제, 또는 그와 유사한 문제들은 알고리즘 자동평가 시스템을 이용하여 채점이 수행되므로 표 4와 같은 형식으로 정형화되어 있으며, 각 구성요소들은 알고리즘의 기본조건들이 만족되도록 설계되어 있다[5].

표 4. 문제 구성요소

구성요소	알고리즘의 기본조건		
① 배경 설명			명확성
② 핵심요구사항	정확성 (유효성)		
③ 제한시간	시간 효율성	유한성	
④ 입력데이터의 범위		입력 출력	
⑤ 입출력 형식과 예시			

표 5는 한국정보올림피아드 형식의 '연속부분 최대합' 문제다. 이 문제를 통해 알고리즘의 지도방법에 대해 다양한 관점에서 논의해 보고자 한다. 알고리즘은 Introduction to Algorithms [8]의 의사코드 표기법을 사용하였다.

표 5. 연속부분 최대합 문제

[① 배경 설명]
일차원 배열에 n개의 정수가 담겨 있다. 배열에서 임의의 연속된 구간을 잡아 그 합이 최대가 되도록 하려고 한다. 예를 들어 다음과 같은 일차원 배열이 주어질 때,

1	-3	4	-2	8	-9	3	3	2	-1

다음과 같이 연속된 구간을 선택하면 그 합이 10이 되고

1	-3	**4**	**-2**	**8**	-9	3	3	2	-1

다음과 같이 연속된 구간을 선택하면 그 합이 8이 된다.

1	-3	4	-2	8	-9	**3**	**3**	**2**	-1

[② 핵심요구사항]
n과 n개의 수를 입력받아 합이 최대가 되도록 하는 연속구간을 찾아 그 합을 출력하는 효율적인 프로그램을 작성하시오. 실행 파일의 이름은 문제 코드와 동일하다.

[③ 제한시간]
실행 시간은 0.5초를 넘을 수 없다.

[④ 입력 데이터의 범위, ⑤ 입출력 형식과 예시]
입력 형식
입력 파일의 이름은 input.txt이다. 첫째 줄에 1,000,000 이하의 자연수 n이 주어진다. 둘째 줄에는 일차원 배열에 담겨 있는 n개의 정수가 차례로 주어진다. 일차원 배열에 담긴 정수는 -100이상, 100이하이다.

출력 형식
출력 파일의 이름은 output.txt이다. 첫째 줄부터 연속된 구간의 합 중 최댓값을 출력한다.

입력과 출력의 예

입력 (input.txt)

```
10
1 -3 4 -2 8 -9 3 3 2 -1
```

출력 (output.txt)

```
10
```

이 문제는 n개의 수열에서 합이 최대가 되도록 하는 연속 구간을 찾아 그 합을 구하는 문제다. 수열의 원소가 모두 양수라면 수열의 원소 전체의 합이 해가 될 것이다. 그러나 그 원소가 음수라면 문제가 복잡해진다. 음수를 만났을 때 그 다음에 있는 양수가 합을 보상하고도 남을 것이라고 기대하며 그 음수를 포함시켜야 할까?[7] 이 문제를 Brute Force Method, 분할정복(하향식), 동적계획법(상향식)으로 해결하는 알고리즘을 제시하고, 알고리즘 자동평가 시스템을 사용하여 자동 채점하는 과정과 더불어 실제 소요 시간을 비교해 봄으로서 알고리즘의 성능을 확인하는 학습방법에 대해 논의해 보고자 한다. 지금 제시하는 알고리즘은 특정 자료구조를 요구하지 않으며, 검색이나 정렬 등의 별도의 알고리즘이 필요하지 않다.

1. 알고리즘 디자인의 설계

이 문제는 최대 1,000,000 이하의 자연수 n이 주어지는 상황에서 프로그램의 실행 시간은 0.5초를 초과할 수 없다. 경시대회의 경우 제한시간을 초과하는 케이스에 대해서는 오답처리가 되므로 시간 효율적인 알고리즘을 고안하는 것이 무엇보다 중요하다.

1.1 Brute Force Method의 가치

Brute Force Method는 알고리즘의 시간적 효율을 고려하지 않고, 모든 경우를 일일이 조사하여 해를 구하는 방식이다. 이러한 Brute Force Method는 문제의 해결과 더불어 다른 알고리즘의 효율성을 비교하는데 의미가 있다.

① $O(n^3)$ 알고리즘

이 알고리즘은 1<=i<=j<=n을 만족하는 i, j의 모든 쌍에 대해 A[i..j]의 합을 계산하고 그 중에서 최댓값을 구하는 것이다.

```
MAX-SUM()
1. A[1..n]을 새로운 배열이라 한다.
2. for i=1 to n
3.       for j=i to n
4.             sum=0
5.             for k=i to j
6.                   sum=sum+A[k]
7.       maxsofar=MAX(maxsofar,sum)
8. return maxsofar
```

② $O(n^2)$ 알고리즘 1

이 알고리즘은 A[i..j]의 합을 구하기 위해 j-i+1회의 덧셈을 하는 ①과는 달리 A[i..j]의 합이 바로 전에 계산했던 합(A[i..j-1]의 합)에 A[j]를 더한 결과임을 착안하여 간단하게 부분 합을 구한다.

```
MAX-SUM()
1. A[1..n]을 새로운 배열이라 한다.
2. for i=1 to n
3.       sum=0
4.       for j=i to n
5.             sum=sum+A[k]
6.             maxsofar=MAX(maxsofar,sum)
7. return maxsofar
```

③ $O(n^2)$ 알고리즘 2

이 알고리즘은 미리 생성해 놓은 예비 자료를 이용하여 부분 합을 계산한다. cumA의 i번째 요소는 A[1..i] 요소들의 누적 합계이므로, cumA[j]-cumA[i-1]을 계산하면 A[i..j]의 부분 합을 상수시간에 구할 수 있다.

```
MAX-SUM()
1. A[1..n]을 새로운 배열이라 한다.
2. cumA[0..n]을 새로운 배열이라 한다.
3. cumA[0]=0
4. for i=1 to n
5.       cumA[i]=cumA[i-1]+A[i]
6. for i=1 to n
7.       for j=i to n
8.             sum=cumA[j]-cumA[i-1]
9.             maxsofar=MAX(maxsofar,sum)
10. return maxsofar
```

1.2 효율적 문제해결 접근법 : 하향식과 상향식

지금까지 살펴본 알고리즘 보다 더 효율적인 알고리즘을 찾아야 한다면, 문제를 어떤 방식으로 접근해 나가야 할까? 효율적 문제해결 접근법에는 하향식(Top-Down) 접근과 상향식(Bottom-Up) 접근이 있다. 이 두 가지 방법을 이용한 각각의 알고리즘에 대해 살펴보자.

① O(nlgn) 알고리즘 : 분할 정복 (하향식)

이 알고리즘은 분할 정복 기법을 적용한다. 분할 정복은 크기가 n인 문제를 풀 때, 크기가 절반인 두 개의 하위 문제를 재귀적으로 푼 후, 각각의 부분 해를 결합해서 전체 문제의 해를 구하는 독특한 방식이다. 배열 A의 정중앙을 기준으로 두 부분으로 나누고 앞쪽의 배열에서 최대 부분합을, 뒤쪽의 배열에서 최대 부분합을 계산하고, 마지막으로 분할 경계면에 걸쳐진 최대 부분합을 계산한 후 방금 구한 3개의 값 중 가장 큰 값을 재귀적으로 구함으로써 전체 해를 구할 수 있다.

```
1. A[1..n]을 새로운 배열이라 한다.
2. answer=MAX-SUM(1,n) 호출
MAX-SUM(left, right)
3. lmax=-∞, rmax=-∞
4. if left>right
5.       return 0
6. if left==right
7.       return A[left]
8. mid=(left+right)/2
9. sum=0
10. for i=mid to left
11.       sum=sum+A[i];
12.       lmax=MAX(sum,lmax)
13. sum=0
14. for i=mid+1 to right
15.       sum=sum+A[i];
16.       rmax=MAX(sum,rmax)
17. return MAX(lmax+rmax, MAX-SUM(left,mid),
    AX-SUM(mid+1,right))
```

② O(n) 알고리즘 : 동적 프로그래밍 (상향식)

이 알고리즘은 연속부분 최대합 문제를 가장 효율적으로 해결한다. 배열의 맨 앞에서 시작하여 오른쪽 끝까지 읽어가면서 그때까지의 최대 부분합을 저장하는 방식이다. 이때 i번째 요소를 마지막으로 포함하는 연속부분 최대합은 i-1번째 요소를 마지막으로 포함하는 연속부분 최대합을 포함하거나 i번째 요소만을 포함할 것이다. 이러한 아이디어를 이용하면 전체 해를 선형시간 내에 구할 수 있다.

```
MAX-SUM()
1. A[1..n]을 새로운 배열이라 한다.
2. sumA[1..n]을 새로운 배열이라 한다.
3. sumA[1]=A[1]
4. maxsofar=sumA[1]
5. for i=2 to n
6.       sumA[i]=max(sumA[i-1]+A[i],A[i])
7.       maxsofar=MAX(sumA[i],maxsofar)
8. return maxsofar
```

1.3 알고리즘 성능의 실제적 비교

문제를 해결하는 알고리즘을 작성하고, 이에 대한 실제 소요시간을 비교해 봄으로써 알고리즘의 성능을 직접 확인할 수 있다. 표 6은 Intel(R) Core2 Quad CPU 2.4GHz에서 연속부분 최대합 문제에 대한 각 알고리즘의 소요 시간을 측정한 것이다. 이것은 O 표기법의 시간복잡도 분석보다 구체적이고 명확하다. 실행하는 컴퓨터의 성능에 따라, 그리고 실행 시점의 CPU 상태에 따라 매번 소요시간의 차이가 발생하지만, 소요시간 측정은 알고리즘의 상호 비교에 의미가 있으므로 충분히 교육적 가치가 있다고 할 수 있다. 더불어 학습자들은 시간복잡도의 O 표기법에서 상수항과 계수가 왜 의미가 없는지를 직접 확인할 수 있다.

표 7. 알고리즘의 소요시간 비교

(단위: 초)

	$O(n^3)$	$O(n^2)$ 1	$O(n^2)$ 2	O(nlgn)	O(n)
10	0.210	0.042	0.041	0.022	0.023
100	0.056	0.024	0.023	0.023	0.023
1,000	0.111	0.021	0.021	0.023	0.024
10,000	82.001	0.660	0.068	0.031	0.025
100,000	측정불가	4.390	4.352	0.158	0.051
1,000,000	측정불가	457.999	465.241	2.401	0.295

학습자는 알고리즘 자동 평가시스템의 자동 채점을 통해서 자신의 알고리즘에 대한 제한시간 요건의 충족여부를 바로 확인 가능하다. 즉, $O(n^3)$와 $O(n^2)$ 알고리즘은 최소한 10,000을 초과하는 입력 값의 개수 N에 대해서 제한시간 0.5초 이내에 해를 구할 수 없음을 확인할 수 있다. 그림 1은 $O(n^2)$ 알고리즘을 이용하여 이 문제를 자동 채점한 결과다. 입력 데이터 23번 케이스와 26번 이후 케이스에 대해서 'Time Limit Exceeded' 메시지가 출력된 것을 확인할 수 있다. 알고리즘 자동 평가시스템은 학습자가 제출한 프로그램이 문제의 제한 시간을 초과하도록 결과를 출력하지 못하는 경우 해당 프로세스를 강제 종료하면서 관련 메시지를 출력하도록 설계되어 있다.

```
** Simple KOI 채점 프로그램 0.98.2.0 **

programmed by 장원영 2014.03.27
http://codingfun.net

연결 초기화.
접속 중...
KOI 데이터 DB 접속 성공 !

장원영님이 맞나요? <맞으면 Enter, 틀리면 아무키 입력>

코드 : 00b00
제목 : 연속부분 최대합

위의 문제를 채점하시겠습니까? <맞으면 Enter, 틀리면 아무키 입력>

00b00.exe 채점 시작...

001 : <0.0089 sec> -- Good
002 : <0.0091 sec> -- Good
003 : <0.0088 sec> -- Good
004 : <0.0091 sec> -- Good
005 : <0.0085 sec> -- Good
006 : <0.0090 sec> -- Good
007 : <0.0096 sec> -- Good
008 : <0.0084 sec> -- Good
009 : <0.0091 sec> -- Good
010 : <0.0095 sec> -- Good
011 : <0.0289 sec> -- Good
012 : <0.0107 sec> -- Good
013 : <0.0124 sec> -- Good
014 : <0.0151 sec> -- Good
015 : <0.0220 sec> -- Good
016 : <0.0445 sec> -- Good
017 : <0.0524 sec> -- Good
018 : <0.0291 sec> -- Good
019 : <0.0091 sec> -- Good
020 : <0.0547 sec> -- Good
021 : <0.3579 sec> -- Good
022 : <0.1223 sec> -- Good
023 : <0.5025 sec> -- Time Limit Exceeded / Runtime Error
024 : <0.1791 sec> -- Good
025 : <0.1537 sec> -- Good
026 : <0.5022 sec> -- Time Limit Exceeded / Runtime Error
027 : <0.5026 sec> -- Time Limit Exceeded / Runtime Error
028 : <0.5028 sec> -- Time Limit Exceeded / Runtime Error
029 : <0.5019 sec> -- Time Limit Exceeded / Runtime Error
030 : <0.5022 sec> -- Time Limit Exceeded / Runtime Error
031 : <0.5027 sec> -- Time Limit Exceeded / Runtime Error
032 : <0.5060 sec> -- Time Limit Exceeded / Runtime Error
033 : <0.5023 sec> -- Time Limit Exceeded / Runtime Error
034 : <0.5020 sec> -- Time Limit Exceeded / Runtime Error
035 : <0.5017 sec> -- Time Limit Exceeded / Runtime Error
036 : <0.5119 sec> -- Time Limit Exceeded / Runtime Error
037 : <0.5026 sec> -- Time Limit Exceeded / Runtime Error
038 : <0.5020 sec> -- Time Limit Exceeded / Runtime Error
039 : <0.5020 sec> -- Time Limit Exceeded / Runtime Error
040 : <0.5023 sec> -- Time Limit Exceeded / Runtime Error

채점 끝 !

KOI 데이터 DB 접속 해제 !
```

그림 1. $O(n^2)$ 알고리즘의 자동 채점 결과

알고리즘 자동 평가 시스템에서 적용한 채점 데이터 케이스별 n값은 표 7과 같다.

표 7. 자동 채점을 위한 데이터 케이스별 n값

번호	n값	번호	n값	번호	n값	번호	n값
#1	6	#11	1393	#21	28221	#31	150000
#2	8	#12	1861	#22	15993	#32	200000
#3	2	#13	2617	#23	41846	#33	300000
#4	2	#14	3481	#24	18932	#34	400000
#5	4	#15	5238	#25	18015	#35	500000
#6	20	#16	8882	#26	89123	#36	600000
#7	50	#17	9759	#27	75605	#37	700000
#8	100	#18	33	#28	70839	#38	800000
#9	300	#19	366	#29	98518	#39	900000
#10	600	#20	10000	#30	100000	#40	1000000

1.4 이분 검색(Binary Search)의 새로운 접근

다음은 한국정보올림피아드 2012년 초등부 2번 '예산' 문제다. 이 문제를 통해 이분 검색을 이용하여, 해를 찾는 과정에 대해서 논의해 본다.

표 8. 예산 문제

[① 배경 설명]
국가의 역할 중 하나는 여러 지방의 예산요청을 심사하여 국가의 예산을 분배하는 것이다. 국가예산의 총액은 미리 정해져 있어서 모든 예산요청을 배정해 주기는 어려울 수도 있다. 그래서 정해진 총액 이하에서 가능한 한 최대의 총 예산을 다음과 같은 방법으로 배정한다.

(1) 모든 요청이 배정될 수 있는 경우에는 요청한 금액을 그대로 배정한다.

(2) 모든 요청이 배정될 수 없는 경우에는 특정한 정수 상한액을 계산하여 그 이상인 예산요청에는 모두 상한액을 배정한다. 상한액 이하의 예산요청에 대해서는 요청한 금액을 그대로 배정한다.

예를 들어, 전체 국가예산이 485이고 4개 지방의 예산요청이 각각 120, 110, 140, 150이라고 하자. 이 경우, 상한액을 127로 잡으면, 위의 요청들에 대해서 각 각 120, 110, 127, 127을 배정하고 그 합이 484로 가능한 최대가 된다.

[② 핵심요구사항]
여러 지방의 예산요청과 국가예산의 총액이 주어졌을 때, 위의 조건을 모두 만족하도록 예산을 배정하는 프로그램을 작성하시오.

[③ 제한시간]
프로그램의 실행시간은 1초를 넘을 수 없다.

[④ 입력 데이터의 범위, ⑤ 입출력 형식과 예시]
입력 형식
입력 파일의 이름은 input.txt이다. 첫째 줄에는 지방의 수를 의미하는 정수 N이 주어진다. 3 이상 10,000 이하이다. 다음 줄에는 각 지방의 예산요청을 표현하는 N개의 정수가 빈칸을 사이에 두고 주어진다. 이 값들은 모두 1 이상 100,000 이하이다. 그 다음 줄에는 총예산을 나타내는 정수 M이 주어진다. M은 N이상 1,000,000,000 이하이다.

출력 형식
출력 파일의 이름은 output.txt이다. 첫째 줄에는 배정된 예산들 중 최대값인 정수를 출력한다.

입력과 출력의 예
입력(input.txt)

```
4
120 110 140 150
485
```

출력(output.txt)

```
127
```

이분 검색은 여러 데이터들 중에서 특정 키 값을 빠르게 찾는 문제에 적용된다. 그런데, 발상을 전환하면 최적 해를 찾는 문제에서 이분 검색을 활용할 수 있다. 그러나 대부분의 알고리즘 교재에서는 주어진 수열에서 key 값을 찾는 상황에서 이분 검색을 설명하므로 학습자는 이분 검색의 효용성에 대해서 매우 추상적인 개념으로 학습하게 된다. 즉, 알고리즘은 주어진 문제를 효율적으로 해결하는데 의미가 있으나 문제에 대한 맥락화 없이 학습자에게 전달되어 학습자의 유의미 학습을 오히려 방해한다. 예를 들어, 수학을 배우는 학생이 미적분의 활용도를 모르는 상태에서 미적분 문제를 풀기 위해 애쓰는 과정과 유사하다. 표 8의 예산 문제를 이용하여 이분 검색 알고리즘을 맥락화하여 적용하는 과정을 논의해 보자.

이 문제는 여러 지방의 예산요청과 국가예산의 총액이 주어졌을 때, 주어진 조건에 만족하도록 가능한 최대의 예산을 배정하는 것이다. 만약 국가예산의 총액이 지방의 모든 예산요청액을 충족할 수 있다면 예산 요청액의 최댓값을 출력하면 된다. 그러나 모든 예산 요청액을 충족할 수 없다면, 총 예산 범위 내에서 주어진 조건을 만족하는 최대 배정액을 구해야 한다. 이때 최대 배정액은 [1 ~ 국가 총 예산] 사이에 존재한다는 사실은 명백하다. 즉, 후보해의 도메인 [1 ~ 국가 총 예산]에서 최적해를 찾기 위해 이분 검색을 이용할 수 있다는 것이다. 이분 검색을 통해 주어진 조건을 만족하는 최대 배정액을 찾을 때까지 도메인의 범위를 1/2씩 계속적으로 줄여나가면 된다.

```
1. P[1..N]에는 각 지방의 예산 요청액이 저장됨
2. TOT에는 국가 총 예산이 저장됨
IS-POSSIBLE(b)
3. // 주어진 배정액 b가 주어진 조건에 만족하는지 확인한다.
4. for i=1 to n
5.       if P[i]<=b
6.             TOT=TOT-P[i]
7.       else
8.             TOT=TOT-b
9. if TOT<0
10.      // 배정 불가능
```

```
11.       return 0
12. else
13.       // 배정 가능
14.       return 1
BINARY-SEARCH()
15. max=-∞
16. for i=1 to n
17.       sum=sum+P[i]
18.       if P[i]>max
19.             max=P[i]
20. if sum<=tot
21.       return max
22. left=1
23. right=tot
24. while left<=right
25.       mid=(left+right)/2
26.       // 중앙값 mid는 후보해다.
27.       // 만약 이 후보해가 조건을 만족한다면
28.       // 더 큰 배정액을 찾아야 한다.
29.       // 그렇지 않으면
30.       // 더 작은 배정을 찾아야 한다.
31.       if IS-POSSIBLE(mid)
32.             left=mid+1
33.       else
34.             right=mid-1
35. return right
```

만약 Brute Force Method를 이용하면 후보해의 도메인 [1 ~ 국가 총 예산]을 순서대로 하나씩 확인하면서 최대 배정액을 찾을 것이다. 다음은 이 방법을 적용한 알고리즘이다.

```
LINEAR-SEARCH()
1. max=-∞
2. for i=1 to n
3.       sum=sum+P[i]
4.       if P[i]>max
5.             max=P[i]
6. if sum<=tot
7.       return max
8. i=1
9. while i<=TOT
10.       if IS-POSSIBLE(i)
11.             i=i+1
12.       else
13.             break
14. i=i-1
15. return i
```

첫 번째 알고리즘은 O(lgn)이고, 두 번째 알고리즘은 O(n)이다.

2. 기초 프로그래밍 학습

경시대회 형식의 문제를 C언어 등의 프로그래밍 언어 기초 학습에서 적용할 수도 있다. 즉, 문법을 학습하고, 주어진 예제를 코딩해 보는 단순한 실습에서 벗어나 학습자는 학습 내용의 문법적 요소를 사용해야 해결 가능한 문제들을 풀어봄으로써 자연스럽게 프로그래밍 언어를 학습할 수 있다. 예를 들어 표 9는 프로그래밍 언어의 연산자를 학습한 후 적용할 수 있는 문제다.

표 9. 기초 프로그래밍 문제

두 정수와 그 사이에 사칙연산자가 주어질 때 해당 연산의 결과를 출력하는 프로그램을 작성하시오. 실행 파일의 이름은 문제 코드와 동일하며, 프로그램의 실행시간은 1초를 넘을 수 없다.

입력 형식
입력 파일의 이름은 input.txt이다. 첫째 줄에 정수 2개와 사칙연산자 (+, -, *, /)가 각각 공백을 사이에 두고 주어진다. 단, 정수는 500000 이하이며 나눗셈(/)의 경우 피젯수가 젯수에 의해 나누어 떨어진다.

출력 형식
출력 파일의 이름은 output.txt이다. 첫째 줄에 주어진 입력에 대한 연산 결과를 출력한다.

입력과 출력의 예 1
입력 (input.txt)

```
3 + 10
```

출력 (output.txt)

```
13
```

입력과 출력의 예 2

입력 (input.txt)

```
7 - 12
```

출력 (output.txt)

```
-5
```

입력과 출력의 예 3

입력 (input.txt)

```
12 * 6
```

출력 (output.txt)

```
72
```

입력과 출력의 예 4

입력 (input.txt)

```
45 / 5
```

출력 (output.txt)

```
9
```

이 문제를 통해 학습자는 프로그래밍 언어의 사칙 연산자, 조건문, 그리고, 대입 연산자를 자연스럽게 학습할 수 있게 된다.

IV. 결론

생활 속 모든 것이 소프트웨어로 움직이는 시대가 도래하고 있다. 스마트, 모바일 시대를 지나 다가올 사물 인터넷 시대에는 사람과 사람, 사람과 사물, 사물과 사물들이 서로 연결되어 통신한다. 그런데 이러한 연결과 그 연결을 통해 발생되는 모든 서비스는 SW에 의해서 형상화된다. 즉, SW 없이는 아무것도 할 수 없는 시대가 온 것이다. 교육에서도 마찬가지다. 다양한 교과들을 융합하는 과정에서 학생들은 통섭적 사고를 기르게 된다. 이러한 통섭적 사고는 현실 세계에서 발생하는 다양한 문제를 상황 맥락적으로, 그리고 창의적으로 해결할 때 의미가 있다. SW로 이루어진 세계에서 SW 원리에 대한 지식이나 이해 없이 창의적으로 문제를 해결한다는 것은 어불성설이다. 디지털 창조경제 시대를 맞이하는 문이과 통합형 교육과정에서 SW교육을 강조하는 이유가 바로 여기에 있다.

문이과 통합형 교육과정의 총론에서 SW 교육이 한층 강화될 예정이다. 초, 중학교에서 정보 교육이 필수로 지정된 만큼 이제는 정보교육의 가치를 재정립하기 위해 다양한 노력이 필요할 때다. 이번이 마지막이라는 긴장감으로 초, 중, 고 교육과정, 그리고 대학 교육의 전체적인 큰 틀에서 SW 교육의 연계성과 위계성을 고려하여, 학습 내용, 교수학습 방법, 평가 등의 모든 면을 다시 원점에서부터 검토해야 할 것이다. 초, 중, 고등학교에서 매번 Scratch만 배울 수 없다. 점차적으로 심화된 내용으로 발전해야 한다. 그 과정에서 정보과학의 빅 아이디어(Big Idea)인 '알고리즘'에 대한 체계적인 교육방법에 대한 논의가 필수적이라 할 것이다. 더불어 프로그래밍 경시대회가 정보과학 분야의 영재만을 위한 수월성 교육이라는 편협한 관점에서 벗어나 일반 학생들을 위한 알고리즘 교육 방법의 다양성 차원에서 접근해야 할 것이다.

참고문헌

[1] 임형석, 김희철, "경시대회를 통한 프로그래밍 교육 활성화 방안," 정보과학회지, 제25권, 제7호, 35-37쪽, 2007년 07월

[2] 조환규, "ACM 국제 대학생 프로그래밍 대회(ICPC)와 그 교육적 효과," 정보과학회지, 제25권, 제7호, 26-34쪽, 2007년 07월

[3] 정종광, "과학고 학생을 위한 Online Judge 기반 프로그래밍 평가 시스템의 설계 및 구현," 석사학위 논문, 한국교원대학교, 2010년.

[4] 송지혜, "자기주도학습을 위한 자동채점기반의 프로그래밍 교육 시스템," 박사학위논문, 숭실대학교, 2011년

[5] 장원영, 김성식, "알고리즘 자동평가 시스템의 개발 및 적용 : 프로그래밍 학습 효과 분석," 한국컴퓨터교육학회 논문지, 제17권, 제4호, 26-34쪽, 2014년 07월.

[6] 전현석, 정종광. 김성식, "C언어 기초 학습을 위한 문제 설계 및 운영," 한국컴퓨터교육학회 학술발표대회논문집, 제18권, 제1호, 291-294쪽, 2014년.

[7] 윤성준, 조상민, "*생각하는 프로그래밍*", 인사트, 150-160쪽, 2003년.

[8] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, *"Introduction to Algorithms"*, The MIT Press, pp.18, 2009.

저 자 소 개

장 원 영

1998: 충북대학교 회계학과 경영학사 (경영정보학 부전공)
2002: 충북대학교 컴퓨터교육과 이학사
2003: 충북대학교 교육대학원 상업교육 교육학석사
2004~2013: 충북과학고등학교 교사
2014: 국무총리 표창 (정보문화유공)
현 재: 충북교육정보원 파견교사
한국교원대학교 컴퓨터교육 박사과정
관심분야: 알고리즘 교육, 한국정보올림피아드, Java 언어 교육, Computational Thinking, EPL

김 성 식

1977: 고려대학교 경영학과 경영학사
1988: 오리곤 주립대학교 전산학과 이학석사
1992: 고려대학교 컴퓨터과학과 이학박사
현 재: 한국교원대학교 컴퓨터교육과 교수
관심분야: 컴퓨터교육, 알고리즘, 정보윤리교육