

다익스트라의 ‘프로그래밍의 수련(修練)’ : 여덟 번째, 유클리드 알고리즘의 재고찰(再考察)

(Dijkstra’s “A Discipline of Programming”:
The Eighth Lecture, Euclid’s Algorithm Revisited)

김도형

성신여자대학교 컴퓨터정보학부

Do-Hyung Kim

School of CSE, Sungshin Women’s University

E-mail: dkim@sungshin.ac.kr; URL: <http://cs.sungshin.ac.kr/~dkim>

요 약

이번 튜토리얼에서는 두 양의 정수의 최대공약수를 구하는 유클리드 알고리즘을 다시 살펴본다. 이 알고리즘은 이미 첫 번째 튜토리얼인 ‘수행의 추상화’에서 ‘판지 기계(cardboard machine)’를 사용하여 고찰한 바 있다. 그러나 이번에는 두 번째부터 일곱 번째 튜토리얼 동안 설명하여 온 다익스트라의 프로그래밍 방법론을 사용하여 이 알고리즘을 다시 살펴보는 것이다.

1) 독자들이 지겨워 할 위험성을 무릅쓰고, 나는 이제 또 다른 하나의 장(章)을 유클리드 알고리즘(Euclid’s algorithm)에 할당하려고 한다. 막간을 이용하여 독자들 중 일부는 이미 그 알고리즘을 다음

```
x, y := X, Y;
do x ≠ y → if x > y → x := x - y
            | y > x → y := y - x
fi
od;
```

print(x)

과 같은 형태로 작성하였으리라고 기대한다. 여기에서 반복적 구조의 가드는 대안적 구조가 취소(abort)로 이어지지 않도록 보장한다. 또 다른 독자들은 이 알고리즘이 다음

```
x, y := X, Y;
do x > y → x := x - y
  | y > x → y := y - x
od;
print(x)
```

1) 이번 튜토리얼은 다익스트라의 원저에서 제7장에 해당하는 부분이다.

과 같이 보다 간단하게 작성될 수 있음을 발견

했을 것이다.

이제 판지(cardboard) 놀이는 잊도록 하고, 두 양수(陽數) X 와 Y 의 최대공약수(最大公約數, greatest common divisor)²⁾를 구하기 위한 유클리드 알고리즘을 새로이 만들어 보도록 하자. 이와 같은 문제에 마주쳤을 때, 원론적으로는 항상 두 가지 길이 우리 앞에 펼쳐져 있는 법이다.

한 가지 길은 요구되는 해답의 정의(定義)를 가능한 한 엄밀하게 따라가 보는 것이다. 생각건대 우리는 X 의 약수(約數; divisor)들의 표를 만들 수 있을 것이다; 이 표에 있는 항목들의 개수는 기껏 유한할 것이고, 그 중에는 1이 가장 작은 것이고 X 가 가장 큰 항목일 것이다. (만약 $X=1$ 이라면 최소와 최대인 항목들이 일치할 것이다.) 그런 뒤 우리는 Y 의 약수들의 표도 비슷하게 만들 수 있을 것이다. 이 두 표로부터 우리는 두 표에 모두 나타나는 숫자들의 표를 만들 수 있을 것이다; 그렇게 되면 이것은 X 와 Y 의 공통(common) 약수들의 표가 되고, 1이라는 항목은 포함하고 있을 것이므로 비어 있는 표는 확실히 아니다. 따라서 우리는 이 세 번째 표로부터 크기가 가장 큰 항목을 고를 수 있고(이 표는 유한하기도 하므로!), 그것이 최대(greatest) 공약수가 될 것이다.

앞에서 개략적으로 보인 것처럼, 때로는 정의를 꼼꼼하게 따라가는 것이 우리가 할 수 있는 최상의 일이다. 그러나 만약 계산할 함수의 성질들을 안다면(혹은 찾아낼 수 있다면) 시도해 볼 다른 대안(代案)이 있다. 우리가 (함수에 관

해) 충분히 많은 성질들을 알아서 그것들을 함께 결합하면 그 함수를 결정지을 수도 있으므로, 우리는 이 성질들을 활용하여 해답의 구성을 도모할 수도 있다.

예컨대, 우리는 최대공약수의 경우 $-x$ 의 약수들은 x 자신의 약수들과 같으므로, $GCD(x, y)$ 가 음수 인자(因子)들에 대해서도 정의되며 우리가 인자들의 부호를 바꾼다 해도 ($GCD(x, y)$ 는) 변하지 않는다는 것을 관찰한다. 이 함수는 인자들 중 하나가 0일 때도 역시 정의된다; 그 인자는 무한히 많은 약수들의 표를 가지나(따라서 우리는 그 표를 만들려고 시도하지 말아야 한다) 나머지 한 인자($\neq 0$)가 유한한 약수들의 표를 가지므로, 공통 약수들의 표는 여전히 비어 있지 않고 또한 유한하다. 그래서 우리는 $GCD(x, y)$ 가 $(x, y) \neq (0, 0)$ 인 (x, y) 모두에 대하여 정의된다는 결론에 이른다. 거기에 더해, '공통'이라는 개념의 대칭성(對稱性)³⁾ 때문에 두 숫자의 최대공약수는 두 인자들의 대칭 함수(symmetric function)이다. 조금만 더 생각해 보면, 두 인자의 최대공약수는 인자들 중 하나를 (두 인자의) 합(合)이나 차(差)로 대치한다 하더라도 변하지 않는다는 사실을 우리는 확인할 수 있다. (여기까지) 우리가 안 것을 정리하면, 다음과 같이 적을 수 있다:

$(x, y) \neq (0, 0)$ 인 (x, y) 에 대해,

(a) $GCD(x, y) = GCD(y, x)$.

(b) $GCD(x, y) = GCD(-x, y)$.

(c) $GCD(x, y) = GCD(x + y, y)$
 $= GCD(x - y, y)$, 등등.

(d) $GCD(x, y) = abs(x)$, 만약 $x = y$ 이면.

논의를 위해, 위의 네 가지 성질들이 우리가 GCD -함수에 대해 알고 있는 것의 전부라고 가

2) 다익스트라가 제0장에서 판지 기계를 이용하여 최대공약수 문제를 다룰 때, 제일 처음 문제를 제시할 때를 제외하고는 최대공약수를 나타내기 위해 'GCD'를 사용하였다. 그런데 여기에서 약어(略語)를 사용하지 않고 단어들을 다 적은 것은 다음 단락에 나오는 '해답의 정의'를 따라간다는 방법을 자연스럽게 이끌어내기 위해서이다.

3) 사족이지만, ' A 와 B 의 공약수'는 ' B 와 A 의 공약수'이다.

정하자. 이것들로 충분할까? 여러분이 보다시피, 앞에서부터 세 가지 관계들은 x 와 y 의 최대공약수를 다른 쌍의 최대공약수로 표현하고 있으나, 마지막 것은 x 로 직접 표현하고 있다. 그리고 이러한 점은 우선

$$P = (\text{GCD}(X, Y) = \text{GCD}(x, y))$$

의 참을 확립하는 알고리즘을 강력하게 시사(示唆)⁴⁾하고 있으며(이 조건은 배정문 " $x, y := X, Y$ "에 의해 손쉽게 달성될 수 있다), 여기에서부터 우리는 P 관계가 변하지 않도록 값의 쌍 (x, y) 를 (a)와 (b) 또는 (c)에 따라 '주무른다(massage)'⁵⁾ 만약 우리가 $x = y$ 를 만족하는 상태에 이르도록 이 조작 과정을 해낼 수 있다면, 그 x 의 절대치(絶對值)를 취함으로써 해답을 찾은 것이다.

우리의 궁극적 목표는 P 의 불변(不變) 하에서 $x = y$ 의 참을 확립하는 것이므로, 단조(單調) 감소 함수(monotonically decreasing function)로서 $t = \text{abs}(x - y)$ 를 시도해 볼 수 있겠다.⁶⁾

- 4) 우리의 전제는 본문에 나와 있는 네 가지 성질들이 우리가 GCD-함수에 대해 알고 있는 성질들의 전부라는 것이다. GCD-함수에 관한 성질들 중 처음의 세 개는 본문에 언급되어 있듯이 'X와 Y의 최대공약수'를 다른 숫자의 쌍으로 설명하고 있다. 따라서 우리의 'X와 Y의 최대공약수'를 구하기 위한 방법은, 원래의 숫자 쌍 (X, Y) 를 동일한 최대공약수를 가지는 다른 숫자 쌍 (x, y) 로 바뀌어나가는 것이 주된 연산이 될 것이다.
- 5) 다익스트라는 이 책 전체에 걸쳐서 불변 조건을 유지하면서 입력을 변경시키는 동작을 '주무른다(massage)'고 표현하고 있다. 공감각(共感覺)적인 표현이랄까... 그러나 이 역해(譯解)에서는 편의상 '조작한다'고 표현하는 경우가 많다.
- 6) 최종 목표 상태인 ' $x = y$ ', 즉 ' $x - y = 0$ ' 또는 ' $y - x = 0$ '(달리 표현하면 ' $\text{abs}(x - y) = 0$ ') 상태를 향하여, 알고리즘이 실행됨에 따라 조금씩 가까이 가는 것을 염두에 둔

분석을 간단하게 만들기 위해서—언제나 훌륭한 목표이다!—우리는 x 와 y 가 음이 아닌 값으로 출발했을 때, 음의 값을 도입함으로써 얻을 것이 아무 것도 없다는 점에 주목한다; $x := E$ 라는 배정으로 인해 $x < 0$ 이라는 상태가 성립된다면, $x := -E$ 라는 배정은 결코 t 의 보다 큰 최종 값을 만들지 않을 것이다.⁷⁾ 따라서 우리는

$$P1 = (\text{GCD}(X, Y) = \text{GCD}(x, y))$$

이고

$$P2 = (x \geq 0 \text{ and } y \geq 0)$$

일 때, 불변으로 유지되어야 하는 관계 P 를 다음

$$P = (P1 \text{ and } P2)$$

것이다. 0을 향하여 단조 감소!

- 7) 간단하게 말하자면, 음이 아닌 x 와 y 의 최대공약수를 구하기 위한 조작 과정에서 x (또는 y)를 음의 값으로 만드는 일은 결국 함수 $t = \text{abs}(x - y)$ 의 값을 더 크게 만들므로 의미가 없는 '주무름'이라는 것이다. 자명하다. 여기 서술이 좀 특이하게 되어 있는데, 뜻을 보충하면 다음과 같다: x 에 E 를 배정함으로써 x 가 음의 값이 되었다면, y 가 음이 아니므로 다시 x 에 $-E$ 를 배정하는 경우 t 함수의 값은 작아지면 작아졌지 커지지 않는다. t 는 단조 감소 함수여야 하므로, t 함수가 작아지는 방향은 옳다. 그러나 x 와 y 는 처음에 음이 아닌 값으로 시작되었다. 따라서 어딘가에서 부호를 바꾸는 '주무름'이 없었다면 E 가 음의 값이 되었을 리가 없고, 마찬가지로 x 가 음의 값이 되었을 리가 없다. 이제 다시 E 의 부호를 바꾸어서 x 에 배정하는 일은 중간에 불필요한 일을 하여 목표 상태에서 멀어진 일을 벌충하는 셈이 된다. 결국 중간 단계 어딘가에서 이루어진, 부호를 바꾸는 '주무름'이 하지 않았어야 할 조작인 셈이다.

와 같이 보다 강화한다. 이것은 우리가 성질 (b)에 의해 허용되는 조작인 연산 $x := -x$ 와 $y := -y$ 를 사용할 이유가 없음을 의미한다. 우리에게 남은 것은 다음과 같다:

- (a)로부터: $x, y := y, x$
- (c)로부터: $x := x + y \quad y := y + x$
 $x := x - y \quad y := y - x$
 $x := y - x \quad y := x - y$

이것들을 하나씩 검토해 볼 텐데, $x, y := y, x$ 부터 시작해 보자:

$$\begin{aligned} \text{wp}("x, y := y, x", \text{abs}(x - y) \leq t) &= \\ &= (\text{abs}(y - x) \leq t) \end{aligned}$$

이므로

$$tmin(x, y) = \text{abs}(y - x)$$

이고 따라서

$$\begin{aligned} \text{wdec}("x, y := y, x", \text{abs}(x - y)) &= \\ = (\text{abs}(y - x) \leq \text{abs}(x - y) - 1) &= F \end{aligned}$$

가 된다.

그리고 이렇게 해서—형식적(formal) 유도가 없이는 믿으려고 하지 않는 사람들을 위해서—우리는 우리가 지금까지 발전시켜 온 산술(算術, calculus)⁸⁾에 의해, 조작 연산 $x, y := y, x$ 는 선택된 t 를 실제적으로 감소시키는 데 실패하기 때문에 쓸모가 없다는 것을 증명하였다(또는, 이 표현을 더 좋아한다면, 발견하였다).

8) 제5장과 제6장에서 집중적으로 다루어진, 적절히 종료하는 구조가 되기 위해서는 루프를 반복할 때마다 단조 감소하는 양의 정수 함수가 존재해야 한다는 것을 말한다.

다음으로 시도해 볼 것은 $x := x + y$ 이고, 다시 앞의 장들에 있는 산술을 적용하면:

$$\begin{aligned} \text{wp}("x := x + y", \text{abs}(x - y) \leq t) &= \\ (\text{abs}(x) \leq t) & \\ tmin(x, y) = \text{abs}(x) = x & \\ (\text{우리는 } F \text{를 만족하는 상태로 한정짓는다})^9 & \\ \text{wdec}("x := x + y", \text{abs}(x - y)) &= \\ = (tmin(x, y) \leq t(x, y) - 1) &= \\ = (x \leq \text{abs}(x - y) - 1) &= \\ = (x + 1 \leq \text{abs}(x - y)) &= \\ = (x + 1 \leq x - y \text{ or } x + 1 \leq y - x) & \end{aligned}$$

P 는 첫 항의 부정을 함축하며¹⁰⁾ 여기에 더해 $P \Rightarrow \text{wp}("x := x + y", F)$ 이므로, 가드를 위한 식

$$(P \text{ and } B) \Rightarrow (\text{wp}(SL, F) \text{ and } \text{wdec}(SL, t))$$

는 마지막 항이 만족되면 성립하고, 우리는 첫 번째와—또한 대칭성에 의해—두 번째 가드 명령을 찾은 셈이다:

$$x+1 \leq y-x \rightarrow x := x+y$$

와

$$y+1 \leq x-y \rightarrow y := y+x$$

이 그것들이다. 비슷한 방식으로 우리는

$$1 \leq y \text{ and } 3*y \leq 2*x-1 \rightarrow x := x-y$$

와

9) P 를 만족하므로 x 는 음이 아니고, 따라서 $\text{abs}(x) = x$ 이다.

10) 첫 항 $x + 1 \leq x - y$ 는 $1 \leq -y$ 이고, 이것은 곧 $-1 \geq y = y < 0 = \text{non } (y \geq 0)$ 이다.

$$1 \leq x \text{ and } 3 * x \leq 2 * y - 1 \rightarrow y := y - x$$

와

$$x + 1 \leq y - x \rightarrow x := y - x$$

와

$$y + 1 \leq x - y \rightarrow y := x - y$$

를 찾을 수 있다(형식적 유도는 부지런한 독자들 위한 연습으로 남기겠다).

우리가 구한 것들을 조사해 보면, 앞 장을 마치면서 언급한 방식으로 문제를 해결하는 데 실패하였다는 슬픈 결론을 내리지 않을 수 없다. 즉 $P \text{ and non } BB$ 가 $x = y$ 를 함축하지 않는 것이다.¹¹⁾ (예를 들어, $(x, y) = (5, 7)$ 에 대해서는 모든 가드들이 거짓이다.) 물론 이것이 가르쳐 주는 것은, 앞의 여섯 단계들이 $abs(x - y)$ 가 단조 감소하면서 초기 상태에서부터 최종 상태에 이르는 경로를 항상 제공하는 것은 아니라는 점이다. 따라서 우리는 '다른 가능성'을 모색해야만 한다.

먼저 우리는 $P2$ 를 다음

$$P2 = (x > 0 \text{ and } Y > 0)$$

과 같이 약간 강하게 만들어도 그것을 만족하는 x 와 y 의 초기값에 대해서는 아무런 해가 없으며, 또한 (x, y) 에 대해 0인 값을 만들어낼 아무런 이유도 없다는 점을 깨닫는다. 왜냐하면 이 0이라는 값은 $x = y$ 인 상태에서 빼기를 통해서만 생길 수 있으므로, 그렇다면 이미 최종 상태에 도달한 것이기 때문이다. 그러나 이것은 부차적인 수정일 뿐이고 주된 변경은 새로운 t 함수에 있어야만 하는데, 나는 불변 조건 F 때문에 하한(下限)이 정해지는 t 를 선택할 것을 제안한다.¹²⁾ 명백한 하나의 예는

$$t = x + y$$

이다.

병행 배정문에 대해서는

$$\text{wdec}("x, y := y, x", x + y) = F$$

임을 알 수 있으므로 기각(棄却)한다.

배정문 $x := x + y$ 에 대해서는

11) 알다시피 **non BB**란 반복적 구조의 가드들 중 참인 것이 하나도 없어서 반복이 종료되는 상태이다. 현재 다루고 있는 유클리드 알고리즘에 의한 최대공약수를 구하는 문제의 경우, 적용 가능하다고 여겨지는 모든 연산들을 구하고, 각 연산이 적용되는 것이 옳은 상태 공간에서의 점들을 규정하는 가드들을 구해서 가드 명령들을 만들었다. 이러한 연산들 중 어느 것도 적용할 수 없는 상태인 **non BB**는 문제가 해결된 상태, 즉 우리가 바라는 최종 상태에 도달하였거나 또는 그 상태를 함축하는 상태여야 한다. 그러나 앞의 여섯 개의 가드 명령들의 가드가 모두 거짓인 경우에서조차 우리가 바라는 문제를 해결한 상태인 $x = y$ 에 도달하지 못 한다는 것이다.

12) 우리가 앞에서 문제를 해결하는 데 실패할 때 사용한 t 함수는 $abs(x - y)$ 였다. 이 함수의 하한은 0이나, 그 하한은 불변 조건 P 와 직접적인 관계가 있는 것이 아니라 t 가 절대치 함수라는 특징이 결정된 것이다. (여기서 우리가 하한을 따지는 이유는 t 가 음이 아닌 유한 정수 함수여야 한다는 조건 때문이다.) 그러나 함수 t 는 상태의 함수로서 (상태의) 불변 조건 P 와 밀접한 관계가 있다. 반복적 구조가 실행되는 동안 깨뜨리지 않고 유지해야 할 불변 조건 P 와, 역시 그 동안 단조 감소하면서 음이 아닌 값을 유지하는 t 함수는 현재의 문제가 해결되는 최종 상태로 가까워지기 위해 서로 공조(共助)하는 관계인 것이다. 따라서 실패한 t 함수 대신에 새로이 P 와 관계가 있는 음이 아닌 유한 정수 함수를 모색하는 것이다.

$$\text{wdec}("x := x + j", x + j) = j < 0$$

인데, 이 식이 참일 가능성은 불변 조건 F 에 의해 배제되므로 $j := j + x$ 와 함께) 역시 이 연산도 기각된다.

그러나 다음 배정문 $x := x - j$ 에 대해서는

$$\text{wdec}("x := x - j", x + j) = j > 0$$

인데, P (이러기 위해 강화시킨 것이다)에 의해 함축되는 조건이다. 우리는 희망에 넘친 채

$$\text{wp}("x := x - j", P) = (\text{GCD}(X, Y) = \text{GCD}(x - j, j) \text{ and } x - j > 0 \text{ and } j > 0)$$

을 조사하는데, 가장 바깥쪽에 있는 항들은 P 에 의해 함축되므로 생략할 수 있고 가운데 것만 남는다; 따라서 우리는

$$x > j \rightarrow x := x - j$$

와

$$j > x \rightarrow j := j - x$$

를 찾은 것이며, 이제 조사를 멈출 수도 있다. 왜냐하면 이 두 개의 가드들이 거짓이 될 때는 우리가 원하는 관계 $x = j$ 가 성립하기 때문이다. 우리가 더 진행을 한다면 세 번째와 네 번째의 대안들:

$$x > j - x \text{ and } j > x \rightarrow x := j - x$$

와

$$j > x - j \text{ and } x > j \rightarrow j := x - j$$

를 발견할 것이나, 이것들을 포함시켜서 무엇을 얻을 수 있을지는 불분명하다.¹³⁾

연 습 문 제

1. 동일한 P 에 대해 $t = \max(x, j)$ 라는 선택을 조사해 보라.
2. 동일한 P 에 대해 $t = x + 2 * j$ 라는 선택을 조사해 보라.
3. $X > 0$ 이고 $Y > 0$ 인 X 와 Y 에 대해, 네 개의 변수들을 사용하는 다음 프로그램

```

x, y, u, v := X, Y, Y, X;
do x > y → x, v := x - y, v + u
  | y > x → y, u := y - x, u + v
od;
od;
print(x)
    
```

가 X 와 Y 의 최대공약수를 찍고 이어서 최소공배수(最小公倍數, smallest common multiple)를 찍는다는 것을 증명하라. (연습문제의 끝)

마지막으로, $X > 0$ 이고 $Y > 0$ 이라는 가정을 만족하지 않는 쌍 (X, Y) 를 가지고 이 자그마한 알고리즘을 시작하면, 그리 즐겁지 않은 일이 일어날 것이다. 만약 $(X, Y) = (0, 0)$ 이면 이 알고리즘은 잘못된 결과인 0을 만들어낼 것이고, 인자들 중 하나가 음수이면 수행이 끝나지 않을 것이다. 이 문제점은 (이 알고리즘을) 다음

```

if X > 0 and Y > 0 →
  x, y := X, Y;
    
```

13) 물론 이 가드 명령들을 포함시켜도 문제를 해결하는 데는 상관없다. 다만 프로그램에 군더더기가 붙어서 우아하지 못 해진다는 단점이 있다.

```
do x > y → x := x - y | y > x → y := y - x od;
    print(x)
fi
```

과 같이 작성하여 막을 수 있다. 대안적 구조 내에서 대안을 하나만 제공함으로써, 우리는 이 자그마한 프로그램이 작동하리라고 기대되는 조건을 명확히 서술한 것이다. 이 형태의 알고리즘은 그 정의구역 밖에서 실행을 시작하면 즉각적인 취소로 이어진다는 보다 바람직한 성질을 가진, 잘 보호되고 다소 독립적인 것이 된다.

참고 문헌

[1] Abelson, H. and G. J. Sussman, *Structure and Interpretation of Computer Programs*, 2nd Ed., MIT Press, 1996.

[2] Bergin, T. J. and R. G. Gibson (Eds.), *History of Programming Languages*, Addison Wesley, New York, 1996.

[3] Dahl, O.-J. E. W. Dijkstra, and C. A. R. Hoare, *Structured Programming*, Academic Press, New York, 1972.

[4] Dijkstra, E. W., "The Humble Programmer," *Communications of the ACM*, Vol. 15, No. 10, pp. 859-866, 1972.

[5] Dijkstra, E. W., "Guarded Commands, Nondeterminacy, and Formal Derivation of Programs," *Communications of the ACM*, Vol. 18, No. 8, pp. 453-457, 1975.

[6] Dijkstra, E. W., *A Discipline of Programming*, Prentice Hall, Englewood Cliffs, NJ, 1976.

[7] Ghezzi, C. and M. Jazayeri, *Programming Language Concepts*, 2nd Ed., Wiley, New York, 1987.

[8] Hoare, C. A. R., "An Axiomatic Basis of Computer Programming," *Communications of the ACM*, Vol. 12, No. 10, pp. 576-580,

1969.

[9] Hoare, C. A. R., and N. Wirth, "An Axiomatic Definition of the Programming Language Pascal," *Acta Informatica*, Vol. 2, pp. 335-355, 1973.

[10] Hoare, C. A. R., "The Emperor's Old Clothes," *Communications of the ACM*, Vol. 24, No. 2, pp. 75-83, 1981.

[11] Kafura, D., *Object-Oriented Software Design and Construction with C++*, Prentice Hall, Upper Saddle River, NJ, 1998.

[12] Sammet, J. E., "Programming Languages: History and Future," *Communications of the ACM*, Vol. 15, No. 7, pp. 601-610, 1972.

[13] Shasha, D. and C. Lazere, *Out of Their Minds: The Lives and Discoveries of 15 Great Computer Scientists*, Copernicus, 1998.

[14] Wexelblat, R. L. (Ed.), *History of Programming Languages*, Academic Press, New York, 1981.

김도형



1981 ~ 1985 서울대학교공과대학 컴퓨터공학과(학사)
 1985 ~ 1987 한국과학기술원 전산학과(석사)
 1987 ~ 1992 한국과학기술원 전산학과(박사)
 1992 한국과학기술원 정보전자연구소 연수연구원
 1992 ~ 현재 성신여자대학교 컴퓨터정보학부 부교수
 1997 ~ 1998 스톤니 브룩 소재 뉴욕주립대학교 컴퓨터과학과 객원교수

관심분야: 프로그래밍 언어
