컴퓨터 과학과 수학, 그것들의 관계

DONALD E. KNUTH

소위 컴퓨터과학이라고 불리는 이 학문은 최근에 이미 세계 도처의 대부분의 대학에 널리 퍼져있다. 그리고 지금 이 글은 어떻게 이 과목이 수확과 상호작용 하는지에 대한 개인적 관점의 서술이다. 두 영역사이의 유사성과 차이성을 토론해 봄에 의해서, 그리고 각각이 서로를 돕는 방법들 몇몇을 검사해 봄에 의해서.

전형적인 nontrivial problem은 이들의 상호작용을 설명하기 위해 사용된다.

1. What is Computer Science? 컴퓨터 과학은 상대적으로 젊은 학문이기 때문에, 이것이 어떤 학문인지에 대해서 설명하는 것으로부터 시작하는게 좋을 것 같다. 적어도, 내 아내는 나에게 말했다. 누군가가 그녀에게 내가 무슨 일을 하는지 물어볼 때마다 이것을 설명해야 했었다고, 그리고 나는 오늘날 대부분의 사람들이 나보다 이 영역에 대해 다소 다른 인지들을 가지고 있을 거라고 생각하고 있다. 실제로, 모든 컴퓨터 과학도들은 아마 컴퓨터 과학에 대한 각자의 정의를 가지고 있을 것이다. 이것은 수학자들이 수학의 정의에 대해 모두 다른 정의를 가지고 있는 것 만큼이나 놀라운 일이 아니다. 다행히도, 컴퓨터 과학은 최근 수년간 꾀나 흥했고(Identity crisis를 가지는 것에), 그래서 컴퓨터 과학자들은 스타일 적으로 좋아왔다. 컴퓨터 과학을 설명하는 내가 가장 좋아하는 방식은 알고리즘을 공부하는 것이라고 말하는 것이다. 알고리즘은 간결히 정의된 규칙들의 수열 혹은 나열이며, 이는 어떻게 주어진 인풋 정보들로부터 유한개의 과정을 거쳐 어떻게 구체적인 아웃풋 정보를 생산해 내느냐에 대한 것이기도 하다. 알고리즘에 대한 특정한 표현은 프로그램이며, 이것은 우리가 정보의 특정 표현을 대신하여 데이터 라는 단어를 사용하는 것과 같다.

컴퓨터의 출현에 의해서 만들어진 것들 중에서, 최고로 중요한 발견은 아마도 알고리즘일 것이다. 학문적인 관점으로써, 알고리즘은 무진장 풍부한 킹왕짱 흥미로운 놈이다. 그리고 알고리즘적 관점은 지식을 일반적으로 조직하는 아주 아주 유용한 방법이다. G.E. Forsythe는 "무엇이 자동화 될 수 있을까?"라는 질문이 동시대 문명에 일어나고 있는 가장 철학적이고 실용적인 질문들 중 하나라고 여겨왔다.

이러한 주목들로 부터, 우리가 아마 결론지을 수 있다, 컴퓨터 과학이 컴퓨터의 출현 이전부터 오래 존재했었어야 했었었다고, 이러한 맥락으로, 그것은 그러했다. 이 과목은 역사에 깊게 뿌리박혀 있다. 예를들어 나는 최근에 발견했다. 고대 문서들을 공부하는게 흥미롭다는 것을 발견했다. 3500년 전 바빌로니안에서 무엇이 컴퓨터 공학으로 확장되었는지 그러나 컴퓨터들은 정말로 필요하다 우리가 알고리즘에 일반적인 특성들에 대해 공부하기 전부터 인간의 삶은 단순하지도 않을 뿐더러 단순한 절차로서 빠르게 계산, 예측하기도 어렵다. 그러므로 잠재력이 풍부한 알고리즘의 연구가 일반적인 목적의 계산기가 이용가능해지고 나서야 비로소 충분히 인정되었다.

나는 계산기(그리고 알고리즘이 단순히 산술적이지 않다는 것을 언급해야 한다. 그들은 어떤 종류의 정보도 다룰 수 있으며, 정보가 한번 정확한 방법으로 표현되면, 우리는 심벌의 시퀀스가 컴퓨터 내부에서 마치 숫자인 것처럼 표현된다고 말하곤 한다. 마치 이름같이; 그러나 그것은 심벌의 시퀀스로써 컴퓨터 내부에 표현된 숫자라고 말하는 것이 더 옳은 표현일 것이다.

프랑스에서는 컴퓨터 공학을 "Infomatique"라고 한다; 독일어로는 "Informatik"; 그러나, 이러한 이름들은 알고리즘 자체가 아니라 알고리즘이 조작하는 데이터나 정보 같은 물질들에 주목하고 있다. 반면 노르웨이의 오슬로 대학교 학생들은 컴퓨터 과학에 대해 더 적절한 작명을 했다. 그것을 "Databehandling"이다. 이것을 영어로 바꿔보면 "Data processing"이로 이는 도서관 응용프로그램을 암시하는 것 처럼 보인다. 몇 몇 사람들은 "Computing science"라는 용어를 "Computer science" 대신에 사용하는 것을 제안해왔다.

물론, 무엇이 더 적절한 이름이냐에 대한 연구는 사실 쓸모없다(Pointless). 실제로는 그 이름보다 그 이름이 암시하고 있는 실제 의미와 개념이 중요하기 때문이다. 그것들도 아마 중요하겠지만 컴퓨터 과학의 이러한 다른 이름들이 명백히 더 합법적이고 인정할만한 영역을 만들기 위해서 컴퓨터 기계들 그 자체의 역할의 범위를 줄인다. "Computing machine"에 대한 많은 사람들 중 의견 중 최선은 그것이 필요 악이라는 것이다(다른 방법들이 실패한 경우에 사용되는 어려운 툴이라는 생각) 왜 우리가 어떻게 컴퓨터를 사용해야 하는지에 대한 가르침을 더 많이 강조해야 하는가? 만약 그들이 거의 컴퓨터를 전자현미경 같은 크게 가치있지 않은 도구로 여긴다면..

컴퓨터를 저런 것 보다는 훨씬 더 알고있는 컴퓨터 과학자들은 본능적으로 기계를 소극적으로 다른다 그들의 새로운 학습을 방어하는 츨면에서 그러나, 기계에 대해 자의식이 강할 필요는 없다. 이것은 Newell, Perlis 그리고 Simon에 의해 능숙하게 지적되었다.(그들은 컴퓨터 과학을 단순히 컴퓨터를 공부하는 학문이라고 정의했다. 마치 식물학이 단순히 식물에 대한 학문이며, 천문학이 별에 대한 학문이고, 뭐 그런 것처럼..) 이런 컴퓨터에 의한 현상들은 즉시 복잡해지고 다양화되어갔다. 전기처럼 많은 묘사와 설명이 요구되어졌고 이러한 현상은 공학과 과학 양 학문 모두에 속하게 되었다.

내가 컴퓨터 과학이 알고리즘의 학문이라고 이야기 했을때, 나는 오직 컴퓨터들에 둘러싸인 현상중 하나라고 선정했다. 그래서 컴퓨터과학은 실제로 더 많은 것을 포함하고 있다. 나는 알고리즘들을 강조해왔다. 왜냐하면 그것들이 정말로 이 과목의 핵심이고, 다른 가지들의 근간이 되고 그 가지들을 통합할 수 있는 공통분모라고 생각하기 때문이다. 이것은 미래에 기술이 정착되고 일어날 지 모른다. 25년간 컴퓨팅 머신은 천천히 변해갈 것이고 가까운 미래에 stable한 기술은 없을 것이라고 본다. 꾀나 대조적으로, 그러나 나는 알고리즘의 학문이 도전적이고 중요한 학문으로 남을 것이라고 생각한다. 비록 컴퓨터의 다른 현상들이 미래에 완전히 개척되어 변해버릴 지라도.

> 이러한 컴퓨터 과학의 특성에 관한 논의들에 더 많은 흥미가 있는 독자들은 [17]과 [29]를 보기를 권한다. 추가적으로 위에 reference..

2. Is Computer Science Part of Mathematics? 실제로 많은 컴퓨터 과학자들이 실제로 하고 있는 부분이 수학과 관련이 거의 없습니다. 하지만 만약 알고리즘 쪽으로 이목을 집중해 보면, 이건 수학의 가지 잖아요? 실제로 알고리즘들은 주로 수학자들에 의해 연구되었었고, 누구에 의해서든 간에, 컴퓨터 과학이 정말로 수학의 한 부분이라는 핵심적인 측면은 논쟁의 여지가 있습니다.

그러나, 나는 이러한 논쟁들이 수학이 컴퓨터 과학의 한 부분이라는 제안으로 보상되야 된다고 생각합니다. 따라서, 집합의 상등성의 정의에 따라, 두 과목은 같게 증명되거나 최소, "Schöder-Bernstein theorem"에 따라 같은 능력을 가질것입니다.

제 개인적인 직관으로는 두 집합 포함관계 모두 유효하지 않은 것처럼 보입니다. 이러한 과목들 사이에서 정확한 경계선을 긋는 것은 항상 너무 어려워요.(예를 들어 , 물리화학, 화학물리학 같은 것들을 생각해봐요 ㅋㅋㅋ). 그래도 최소한 수학과 컴퓨터 과학에서의 다른 관점들을 뚜렷히 구분하는 것은 가능해 보입니다.

제가 앞으로 이야기할 진짜 이야기가 아마도 제가 가진 구별을 설명하는 최선의 방법일 것 같습니다. 몇 년 전에 저는 두개의 n X n 최대공약수 D를 가지는 정수 행렬 A, B와 관련된 하나의 수학 이론을 배웠었어요. 이말은 D가 A와 B의 최대공약수라는 이야기이지요. (예를 들어, A = A'D 그리고 B = B'D) 정수 행렬인 A',B'에 대해서요. 그리고 모든 A와 B의 공약수는 D의 약수겠지요. 그래서 저는 어떻게 두 메트릭스의 최대공약수를 계산하는지 궁금했었어요. 며칠 후에 제가 갔었던 컨퍼런스에서 수학자 H. B. Mann을 만났는데요. 그리고 저는 그 친구가 이 문제를 풀 수 있을 거라고 생각했었어요. 저는 그 친구에게 물었고, 그친구는 실제로 정답을 알더군요 !! ㅎㅎㅎ, 그러나 그건 확실히 수학자의 답이었어요, 결코 컴퓨터 과학자의 답은 아니었죠. 그가 말하길, $\mathfrak{R}$을 n x n 정수 행렬의 링이라고 해 , 이 링에서 두 Principal left leals은

principal하므로, *D*는 이렇게 표현되지.

$$\Re A + \Re B = \Re D$$

그러면 *D* 는 *A*와 *B*의 최대공약수이고, 이 식은 확실히 가능한 가장 단순한 것일 것입니다. 여기서 우리는 단지 8개의 심볼만 있으면 다 표현할 수 있어요. 그리고 이것은 수학적 대수학으로 엄격히 증명된 정리들에 의존합니다. 그러나 컴퓨터 과학자의 관점으로서, 이것은 가치가 없습니다. 이것이 무한 집합 $\Re A \quad and \quad \Re B$의 생성을 끌어드렸기 때문에, 그들의 합을 취하고 무한히 많은 행렬 *D* ($\Re D$)를 통해서.. 컴퓨터 공학자로써 나는

$$\begin{pmatrix} 1 & 2 & 3 & 4 \end{pmatrix} + \begin{pmatrix} 4 & 3 & 2 & 1 \end{pmatrix}$$

의 합을 무한한 오퍼레이션을 통해서 계산할 수는 없습니다. 뒤에 이 물음에 대한 컴퓨터 과학자의 답은 제 학생 "Michael Fredman"에 의해 제공되어 있습니다 [15, p.380]을 보시면 됩니다.

제 수학자 친구중 한명이 제게 컴퓨터 과학이 1000개의 심도 있는 정리를 가지면, 기꺼이 이 분야를 가치있는 학문의 한 영역으로 인정하겠다고 하더군요. 이러한 비판은 명백히 다른 이론들 뿐만 아니라 알고리즘을 포함하게 변화되어야 합니다. 그러나 심지어 오늘날의 컴퓨터 과학이 그러한 시험으로 정량화 될 수 없음은 너무나 자명합니다. 만약 "심도 있는"의 의미가 한 슈퍼똑똑이 한명이 수 달을 필요로 해서 발견한 이론이나 알고리즘을 이야기하는 것이라면, 컴퓨터 과학은 그것을 하기에 여전히 너무 어립니다. ㅎㅎ 저는 이 어림을 핸디캡으로 클래임 할 수 있습니다. 우리는 또한 알고리즘을 설명하고 그것들을 이해하고 , 그것들이 옳은지 증명하고 발명하며, 그것들의 행동을 분석하는 데에 최선의 방법이 무엇인지 모릅니다. 비록 이러한 모든 것들 이전에 상당한 과정이 있었더라두요.. 1000개의 심도 있는 결과들이 있습니다. 그러나 아마 현재까지 50개 정도만 발견되었을 거에요.

수학과 컴퓨터 과학이 서로서로에게 미치는 영향, 그리고 상대적 역할들을 설명하기 위해서, 저는 미래를 보았습니다. 그 미래는 컴퓨터 과학이 그 자체로 꾀나 성숙해 있을 때 입니다. 최근 트랜드들은 컴퓨터과학과 수학이 모두 존중받는 학문으로서 존재하는 것을 상상할 수 있게 합니다. 닮은 점을 제공하지만, 개인별 교육의 관점에서 다른 역할이 있습니다. 구글의 "Forsythe"를 다시 한번 인용하면, 과학 혹은 기술교육에서의 최고로 가치있는 습득은 인생 내도록 사용가능한 "GENERAL-PURPOSE" 정신적 도구 입니다. 저는 자연어와 수학을 첫째 즉 이런 도구들 중에서 가장 중요한 것으로 꼽고 싶고, 그 다음으로는 컴퓨터 과학을 꼽고 싶습니다.

수학처럼, 컴퓨터 과학은 일반 교육(국,영,수,과 같은..)으로 여겨질 것입니다. 수학과 다른 과학과목들 처럼, 컴퓨터 과학은 계속해서 두개의 영역으로 쪼게어 질 것입니다. 이론과 적용의 영역으로요. 또한 수학처럼, 컴퓨터 과학은 다소 다른 과학과 다를 것입니다. 그것이 입증할 수 있는 사람에 의해 만들어진 법을 다루는 과목이라는 점에서요. 결코 확실한지 입증할 수 없는 자연법과는 달라요. 따라서 두 과목은 많은 방식에서 공통점이 많을 것입니다. 차이점이라면 주관적 문제와 접근방식의 차이겠지요. 수학이 다소 정리와 무한한 프로세스들, 정적인 관계들을 다룬다면, 컴퓨터 과학은 다소 알고리즘과 유한한 컨스트럭션들, 동적인 관계들을 다루는 차이가 있을 것입니다.

많은 컴퓨터 과학자들은 수학을 해왔습니다. 그러나 훨씬 더 많은 수학자들이 컴퓨터 과학을 해왔습니다. 저는 특정 알고리즘들에 관한 굉장히 많은 수학적 이론들의 인스턴스 들에 의해서 감명 받아왔습니다. 이러한 이론들은 보통 부담되거나 일반적인 알고리즘적 방적식(보통 오늘날 컴퓨터 과학자들이 사용하는) 보다는 훨씬 덜 자연스러운 방식으로 보통 정형화 되어 있습니다. 예를들어 35페이지에 있는 아브라함 왈드의 논문의 대부분의 내용을 2페이지 정도로 표현될 수 있습니다. 만약 그것들을 알고리즘 적인 용어들로 표현한다면요. 또한 ㅁ수많은 다른 예들을 볼 수 있을 것입니다. 그러나 그것들은 또다른 논문을 위한 주제입니다.(여기서 다뤄지지는 않습니다.)

3. Educational side-effects. 컴퓨터 과학을 제대로 공부한 사람이은 어떻게 알고리즘을 다루는지 알고 있습니다. 어떻게 그것들을 만들고, 조작하고, 이해하고 분석하는지에 대해서요. 이러한 지식은 좋은 컴퓨터 프로그램을 작성하는것 훨씬 그 이상을 제공합니다. 이것은 일반적인 목적에서의 정신적 도구이고 이는 다른 과목을 이해하는데 강력한 도움을 줄 것입니다. 그것이 화학이든 언어학이든, 음악이든 뭐든지요.. 아래를 보면 이 주장에 대한 이유를 알 수 있을 거에요. 이런 말 들어보신적 있나요? 어떤 지식을 다른 사람에게 가르칠 수 있지 않다면 그 지식을 온전히 이해한 게 아니다 라는 말이요. 실제로 어떤 사람이 어떤 것을 컴퓨터에게 가르쳐 주기 전까지 그 사람은 진짜로 그것을 이해한게 아닙니다. 예를 들어 알고리즘을 표현하는 것이지요. "자동화된 컴퓨터는 정말로 수학적 생각의 산물로써의 정확한 생각을 요구합니다."[7]. 알고리즘으로 공인화 하는 시도는 우리가 어떤것을 단순히 전통적인 방식으로 이해하는 것 이상의 훨씬 깊고 고차원적인 이해를 요구합니다.

언어학자들은 그들이 언어들을 이해했다고 생각했었습니다. 적어도 그들이 컴퓨터에게 그들의 언어를 설명하려고 하기 전까지는요. 그들은 곧 얼마나 그들이 배워야 할게 더 많다는 것을 깨닳았죠. 많은 이들이 그들이 단순히 작성한 프로그램의 아웃풋을 보는 것보다 모델들을 셋업하면서 배우는게 훨씬 더 많다는 것을 배웠더랬죠.

삼년동안 저는 추상대수학의 대학교 2학년 과정을 가르쳤었습니다. 칼텍에서의 수학 전공자로써, 그리고 항상 가장 어려웠던 주제는 행렬에 대한"Jordan canonical form"에 대한 연구였습니다. 그에 대해 새로운 접근을 시도하던 3번째 되던 해에 주제를 알고리즘적으로 보는 것에 의해서 갑자기 이 주제가 꾀나 클리어해졌고, 같은 것들이 generator와 relations에 의해 정의된 finite group을 논하는 곳에서도 일어났습니다. 그리고 또 다른 코스에서도요(reduction theory of binary quadratic forms). 알고리즘 적으로 이 주제를 봄에 의해서, 수학 정리들의 목적과 의미가 명백해졌습니다.

이후에, 컴퓨터 계산에 관한 책을 쓰면서[15], 저는 시각적으로 모든 자연수에 관한 이론이 자연스러운 방법으로 컴퓨터의 산술 연산을 빠르게 하는 문제와 연결된다는 것을 발견했습니다. 그러므로 저는 자연수 이론에 대한 전통적인 과정들이 아마 저의 관점들을 바꾸었고 기존의 이론들의 실용적인 적용의 아름다운 동기가 됬다고 봅니다.

이러한 예들과 더 많은 이들이 더 내가 알고리즘적 접근의 교육적 가치를 확신하게 했습니다. 이것은 모든 종류의 개념의 이해를 돕습니다. 나는 컴퓨터 과학을 충분히 공부한 학생은 다른 과목을 공부할때에도 그 과목을 극복하는 데에 암시적으로 도움을 받을 것이라고 생각합니다. 그리고 그러므로 컴퓨터 공학이 학부생들이 공부하기에 좋은 교양 필수 과목으로 공부하는 것이 좋다고 말하는 좋은 이유가 될 것입니다. 지금 우리가 수학을 교양 필수 과목으로서 중요시 여기는 것처럼 말입니다. 반면에, 현재 컴퓨터 공학의 학부 과정은 이 목표를 충족시키기에 너무 미흡합니다. 적어도 저는 학부생들이 컴퓨터 공학의 시작 지점에서 제가 원했던 것보다 훨씬 좁은 범위의 교육을 받아 왔다는 것을 찾았습니다. 컴퓨터 과학자들은 물론 그러한 현재의 부족함을 채우기 위해 일하고 있지만, 저는 이것이 알고리즘 대신에 컴퓨터 언어에 대한 지나친 강조에 기인한다고 생각합니다.

4. Some interations. 컴퓨터 과학은 많은 방식으로 수학에 영향을 주고 있습니다. 그리고 저는 좋은 것들의 예 몇가지를 여기에 풀어 놓으려고 합니다. 먼저, 컴퓨터들은 물몬 계산을 위해 산용될 수 있습니다. 그리고 그들은 자주 계산이 지나치게 복잡한 경우의 수학 연구에 적용되어 왔습니다. 그들은 제안된 데이터를 산출하기도 하고 추측을 부숴버리기도 합니다. 예를 들어, 가우스가 말하길, 100만 이하의 소수들의 테이블을 보면서 그가 가진 소수에 관한 정리를 하나 냈습니다. 저의 박사논문에서, 저는 무한히 많고 작은 경우의 컴퓨터 연산들을 면밀히 관착하여, 그 추측을 해결할 수 있었습니다. 또 다른 예로서 "Marshall hall's recent progress"를 들 수 있겠네요.

둘째로 수치 분석의 영역에서 수학과 컴퓨터과학의 명백한 연관성이 있습니다. 논리, 정수론, 저는 이 영역을 다 기억하지는 못하지만, 아무튼 무지 유명합니다. 그러나 반드시 "D. H. LEHMER"의 업적은 언급해야

할 것 같습니다. 그는 몇몇 주목할 만한 방식으로 고전수학의 계산들을 결합해 왔습니다. 예를 들어, 그는 285 이상의 모든 연속한 6개이상의 숫자의 합은 43보타 크거나 같은 소수의 배수를 포함한다 가 있습니다.

또 다른 컴퓨터 과학의 영향은 수학의 영역을 끊임없이 넓혀왔습니다. 존재하는 증명을 알고리즘으로 대체하고, 이렇게 새로 생긴 수학적 오브젝트는 종종 추상론에서의 향상을 이끕니다. 예를 들어, E.C. DADE 와 H. ZASSENHAUS에 주목하여, (그의 1963년 논문을 보면..) 종(부류,종류)의 개념은 이미 orders를 넘어 나머지 정리 쪽에서 그 중요성이 입증 되어 왔습니다. 그래서 계산적 관점으로 소개된 수학적 생각 그 단독으로 그 자신 고유의 이론값을 밝혀냈습니다. 더욱이 위에서 언급한 것처럼, 연속된 알고리즘적 접근은 종종 교육적 가치를 가집니다.

알고리즘적 접근이 수학적 이론들에 영향을 끼치는 또다른 방식은 1 대 1 대응의 생성입니다. 꾀나 종종 수학적 객체들의 특정 타입들과 상등한 간접적인 증명들이 있어왔습니다. 그러면 직접적인 일대일 대응은 실제로 TRUE 이상을 보여줍니다.

> 이산수학, 특히 조합론은 컴퓨터과학의 상승세에 추가적인 후원을 받아왔습니다. 더욱이 모든 이산수학의 다른 영역들도 요즈음 훨씬 더 넓은 적용범위를 갖게 되었습니다.
> 출처를 보고 싶다거나 더 많은 예를 보고 싶다면 [1], [2], [4], [5], [20], [24], [27]을 보길 추천드립니다.
>
> 그러나, 제 견해로는..실제로 컴퓨터과학이 수학에 끼친 최고로 중요한 영향은 위의 논의들과는 다소 다릅니다. 저에게 최고로 중요하게 여겨지는 것은 알고리즘 그 자체에 대한 연구가 흥미로운 수학적 문제들의 "Fertile vein"을 열었다는 점입니다. 그것은 새로운 생각의 결핍 때문에 잘 풀리지 않아왔던 많은 수학적 문제에 새로운 숨을 불어넣어 줄 것입니다. "Charles Babbage" (컴퓨터 과학의 아버지 중 한명)는 1864년에 이미 이렇게 예측 했었습니다. "분석 엔진이 존재하는 한, 컴퓨터 과학의 미래는 밝을 수 밖에 없다".
> ...

저는 많은 흥미를 끄는 수학 문제들을 알고리즘을 분석하면서 발견했었습니다. 얼마나 컴퓨터에서 빠르게 동작하는지 보기 위해서였죠.. 이런 문제들의 전형적인 예는 아래에 있습니다. 예를들어 "Reingold"의 최근 서베이가 있겠네요. 최초의 수학 이론들중 하나는 컴퓨터 과학의 언어론에 영감을 줬습니다. 그것이 지금도 많은 아름다운 결과들은 내고 있지요. [11]과 [12]를 보세요 이러한 새 이론들에서 느끼는 즐거움이 제가 컴퓨터 과학자가 된 이유입니다.

반대로 수학자들 또한 컴퓨터 과학에 중대한 영향을 끼쳤습니다. 거의 모든 수학 지식의 가지들이 어디선가에서는 컴퓨터 과학을 바치고 있다.최근에 이진 트리라고 불리는 이산 객체들을 다루는 문제를 다룬 적이 있었는데 이것은 그것들을 컴퓨터적으로 자주 구현되는 것이었다. 그리고 그 문제의 솔루션은 실제로 complex gamma function과 리만의 zeta function의 곱에 관련이 있었다.[6] 따라서 고전 수학의 결과들은 종종 다소 놀라운 장소에서 아주 유용하게 쓰임이 드러났다.

수학의 컴퓨터과학으로의 응용에 관한 나의 경험 중에서 나에게 가장 놀라웠던 것은 수학의 많은 것들이 특정 이산형 이었다는 사실이었다. (그 예는 아래에 다루겠다.) 이러한 수학들은 거의 통체로 나의 영역이 아니었고, 비록 내가 수학으로 합리적이고 좋은 대학교육과 학부 이후 과정을 밟았더라도, 이러한 테크닉들과 관련하여, 학창시절 거의 내가 만난 것들 대부분이 이 MONTHLY 로부터 일할때 일어났었다. 나는 자연스럽게 전통 커리큘럼이 이러한 많은 이산 수학적 조작들을 포함하기 위해 개정 되어야 하는지 혹은 컴퓨터 과학이 그것들의 빈번한 응용에 예외적이었던 건지 궁금해왔다.