알고리즘에 대한 직관적 이해

고려대학교 ALPS 회장
2018320207 공인호

알고리즘의 사전적 정의

컴퓨터는 우리의 생각만큼 똑똑하지 않습니다. 컴퓨터에는 매우 한정적인 연산만이 정의되어 있습니다. 예를 들어, 컴퓨터에는 두 수의 최대공약수를 구해주는 연산이 정의되어 있지 않기 때문에, 우리는 "18과 12의 최대 공약수를 찾아줘"라고 명령할 수 없습니다. 대신 우리는 컴퓨터가 할 수 있는 연산들을 적절히 나열해 18과 12의 최대 공약수를 찾아줄 수 있도록 명령할 수 있습니다.

예를 들어, "1부터 시작해서 12까지 숫자에 1씩 더하면서, 그 숫자가 18과 12를 모두 나누어 떨어트릴 수 있는지 확인하고, 가능한 숫자들 중 가장 큰 숫자를 알려줘"라고 명령할 수 있죠. 컴퓨터는 덧셈이 가능하기 때문에 1부터 시작해 숫자를 1씩 증가시키는 것이 가능하며, 나눗셈과 나머지 연산이 가능하기 때문에 18과 12를 앞선 숫자로 나누었을 때 나머지를 구할 수 있습니다. 마지막으로 컴퓨터는 비교 연산이 가능하기 때문에 18과 12에 대해 계산한 나머지가 0인지 아닌지 비교할 수 있고, 또 그중 가장 큰 숫자 또한 찾아낼 수 있습니다.

이처럼 컴퓨터에 정의되어 있는 연산/명령을 적절히 나열함으로써 우리가 원하는 연산이나 작업을 컴퓨터에게 시킬수 있으며, 이러한 명령들의 나열을 "알고리즘"이라고 부릅니다. 그리고 이러한 "알고리즘"을 실제 컴퓨터가 인식할 수 있도록 프로그래밍 언어로 작성하는 과정을 "코딩"이라고 부르며, 그 결과물을 "프로그램"이라고 부릅니다.

알고리즘의 직관적인 이해

알고리즘에 대한 개념은 사람에게도 똑같이 적용 가능합니다. 여러분 앞에 흰 종이와 펜 한 자루가 주어져있고, 아래 문제를 해결해야 한다고 생각해보세요.

어떤 정수 n이 주어졌을 때, 2020^n 을 계산하시오

예를 들어, n이 6으로 주어졌다고 생각해 봅시다. 여러분은 어떻게 2020^6을 계산할 건가요? 우선 종이에 2020을 적고, 2020에 2020을 곱한 4080400를 적고, 이런 작업을 4번 더 반복할 것입니다. 2020^n도 마찬가지로 2020을 종이에 적어놓고 2020을 곱하는 과정을 $n-1$ 번 반복함으로써 계산이 가능할 것입니다.

방금 우리가 했던 행동들을 하나 하나 순서대로 나열한다면, 그것이 바로 2020^n을 구하기 위한 "알고리즘"이 되는 것입니다.

왜 알고리즘을 배워야 하나요?

위에 문제를 해결하기 위해 여러분이 해야 하는 곱셈의 횟수는 몇 번일까요? 총 $n-1$ 번입니다. 만약 여러분 앞에 주어진 n이 2147483648(= 2^{32})이라면, 약 21억 번의 곱셈을 해야 하는 것입니다.

근데 위 알고리즘을 이용해 $2020^{2147483648}$을 구할 때, 매번 2020을 곱하는 대신 2020^2을 곱하면 어떨까요? $2020^{2147483648}$ = $(4080400)^{1073741824}$이기 때문에, 절반의 곱셈으로 원하는 값을 계산할 수 있습니다. 즉, 같은 값을 계산하는데 10억 번의 곱셈을 하지 않아도 되는 것입니다.

더 나아가, 우리는 2147483648 = 2^{32}이라는 것에 초점을 맞춰봅시다. 우리는 2020^2을 알고 있기 때문에 2020^4를 $2020^2 * 2020^2$로 계산할 수 있습니다. 마찬가지로 2020^8, 2020^{16}, 2020^{32}, 2020^{64}, ... 을 계산할 수 있고, 결국 총 32번의 곱셈만으로 $2020^{2147483648}$을 계산하는 것이 가능합니다.

세 알고리즘은 모두 올바른 알고리즘입니다. 주어진 입력에 대해서 알고리즘에 명시된 명령들을 순서대로 따라 했을 때, 원하는 결과값이 나오기 때문입니다. 하지만 그 효율성은 다릅니다. 같은 입력에 대해서 첫 번째 알고리즘은 약 21억번의 계산을 필요로 할 때, 두 번째 알고리즘은 10억 번의 계산을 필요로 하고, 마지막 알고리즘은 단 32 번의 계산을 필요로 합니다.

만약 이 알고리즘을 그대로 프로그램으로 제작한다면, 2147483648이라는 입력에 대해서 세 번째 프로그램은 첫 번째 프로그램보다 약 7000만 배 빠르게 값을 계산하는 것입니다. 그리고 이 격차는 입력으로 주어지는 숫자가 커질수록 더욱 극명하게 나타날 것입니다.

Problem Solving이 무엇인가요?

주어진 문제에 대해서 제한된 시간과 메모리 안에 결과값을 계산해내는 프로그램을 작성하는 것을 말합니다. 이러한 Problem Solving을 통해 우리는 알고리즘 설계 능력과 문제 해결 능력을 키울 수 있습니다.

많은 기업에서 IT 직군을 선발할 때, 이러한 알고리즘 설계 능력과 문제 해결 능력을 평가하기 위해 코딩 테스트를 시행하는 추세입니다.

마지막 인사

앞으로 여러분들은 프로그램을 만들기 위해 수많은 알고리즘을 작성해야 합니다. 또 다른 사람이 작성한 코드를 보며 그 사람이 어떤 알고리즘을 가지고 문제에 접근했는지 파악해야 합니다. 다양한 알고리즘을 접해보고 이를 이용해 문제를 풀어보는 것은 앞서 말한 두 행위를 하는 데 있어 큰 도움을 주는 것뿐만 아니라, 알고리즘의 효율성을 계산하고 이를 개선시킬 수 있는 사고력 또한 증진시켜 주며, 이는 개발자로써 자신의 가치를 키울 수 있는 방법이라고 생각합니다.

감사합니다.