안녕하세요. 컴퓨터수학 2분반을 수강하고 있는 소프트웨어학부 컴퓨터과학과 1학년 이유진입니다. 중간고사 발표 시작하겠습니다.

2강에서는 명제, 즉 이진 논리에 대해 배웠습니다. 이진 논리에 대해 배운 이유는 우리가 궁극적으로 알려고 하는 것이 프로그램의 가장 기본적인 단위인 함수이기 때문입니다. 함수를 만드는 것은 x값에 무엇을 넣어도 y값을 구할 수 있는 하나의 루틴을 만드는 일반화 과정입니다. 컴퓨터는 대량의 단순작업을 빠르게 반복하는 장치이기 때문에 이 함수의 효율적인 활용이 필요합니다. 논리 변수와 연산자를 이용한 이진 논리를 중첩하여 함수를 만들기 때문에 우리는 이에 대해 배웠습니다.

여기서 가장 중요한 개념은 명제에 필수적인 의미 부여입니다. 논리연산자를 사용하여 프로그래밍을 할 때 두 명제변수가 올바르게 사용되지 않는다면 프로그램이 돌아가지 않거나 의미 없는 결과값을 내놓을 수 있습니다. 컴퓨터 프로그램은 주어진 연산을 수행하기만 하기 때문에 올바른 명제변수를 선택하는 것이 사용자의 몫입니다. 예를 들어, XOR의 경우 덧셈기로 이용되는 필수적인 논리 회로이고 두 명제변수의 관계의 의미성이 중요하기 때문에 주의해야 하는데, 명제변수 p와 q가 양립할 수 없는 관계이면 안됩니다. 원하는 결과를 도출해내기 위해서는 연산자에 대한 이해가 필수적입니다.

3강에서는 이진 함수와 여러 증명의 종류에 대해 배웠습니다. 컴퓨터는 true와 false, 0과 1로 구성된 이진법과 이에 따른 논리 연산을 진행합니다. 이를 배우는 이유는 다음과 같습니다. 컴퓨터는 이진 논리에 따라 일반화된 함수의 중첩을 통해 하나의 패턴을 만듭니다. 이 패턴을 이진 논리의 증명을 통해 하나의 이론으로 정리합니다. 그리고 이 이론을 이용하여 많은 간단하고 반복적인 일을 빠르고 효율적으로 처리합니다. 따라서 우리는 이 이진 함수와 증명 과정에 대한 이해가 필요합니다.

여기서 가장 중요한 것은 어떤 증명 방법을 사용했는지를 구분할 줄 아는 것입니다. 증명 방법은 다양하고, 우리는 특정 논리를 증명해야 할 때 어떤 방법을 사용하여 증명할지 선택해야합니다. 논리의 증명 과정에서 어떠한 증명 방식이 사용됐는지를 의식하여 알게 된다면 알고리즘을 분석하는 것뿐만 아니라 직접 작성하는 경우에도 큰 도움이 될 것입니다.

4강에서는 집합과 집합의 연산, 집합의 개념에서 확장한 함수와 수열에 대해 배웠습니다. 이런 기초적인 집합의 연산과 수열의 계산을 배워야 하는 이유는, 이 집합과 함수 그리고 수열이 알고리즘을 실행하기 위해서 필요한 수학적인 연산을 구성하기 때문입니다. 집합은 알고리즘에서 데이터를 그룹화하는 방법이고, 집합의 연산은 집합으로 묶인 데이터를 가공하는 방법입니다. 또한 함수는 이런 집합의 개념을 바탕으로 나타낼 수 있습니다. 수열의 합 역시 프로그램 코드를 작성하는 데에 활용할 수 있으며 후에 알고리즘의 key factor과 복잡도를 계산하는 데에도 이용할 수 있으므로 컴퓨터수학에서 배워야 합니다.

여기서 가장 중요한 것은 floor과 ceiling의 개념입니다. Floor은 작거나 같은 정수 중 가장 큰 것이고 ceiling은 크거나 같은 정수 중 가장 작은 것을 의미합니다. 이 개념이 중요한 이유는 우리는 프로그래밍을 함에 있어서 주로 정수를 다루기 때문입니다. 컴퓨터과학은 연속적으로 이루어진 자연계를 모델링하는 것이 아닌, 불연속적인 디지털 세계를 다루기 때문에 실수를 정수로 변환하는 작업이 필요합니다. Floor과 ceiling 개념은 이러한 실수를 정수화 하는 방법으로, 때에 따라 필요한 것으로 선택하여 사용해야 합니다.

5강에서는 여러 알고리즘의 종류와 알고리즘의 복잡도에 대해 배웠습니다. 알고리즘은 소프트웨어의 작동에 가장 큰 영향을 미치는 요소로, 컴퓨터가 문제해결을 하는 데에 가장 명확한 명령 체계입니다. 알고리즘의 종류는 매우 다양하고, 각각의 running time과 효율성은 다릅니다. 더 빠르고 정확하며 효율적인 알고리즘을 사용하는 것이 중요하기 때문에 이 알고리즘들의 작동 방법과 running time에 대한 이해가 필수적입니다. 따라서 여러 알고리즘을 구현하는 대략적인 수도코드와 함께 시간복잡도에 대해 배운 것입니다.

여기서 가장 중요한 개념은 알고리즘의 복잡도, 그 중에서도 Big-O입니다. Big-O는 함수의 upper bound, 즉 시간 복잡도의 상한선을 나타내는 것으로, 함수의 성능을 비교하는 척도가 됩니다. 이 알고리즘이 시행되는 데에 아무리 오래 걸려도 이만큼의 시간이 걸린다는 것을 알려주는 방법이며, 최악의 사건까지 고려가 되기 때문에 최선의 상황을 나타내는 Big-Omega보다 의미 있는 척도입니다. 또한 Big-Theta도 좋은 척도로 작용할 수 있지만 Big-Theta를 나타낼 수 있는 알고리즘이 많지 않기 때문에 Big-O의 중요성이 더 부각됩니다. 프로그래밍을 할 때에는 특정 알고리즘이 얼마나 많은 시간과 메모리를 차지하는지를 알아야 효율적이고 올바른 프로그램을 만들 수 있기 때문에 알고리즘의 복잡도를 측정하는 일은 중요하고 필수적입니다.

6강에서는 5강에 이어 알고리즘 산술 연산의 복잡도에 대해 배우고, 재귀 알고리즘의 개념에 대해 배웠습니다. 재귀 알고리즘을 사용하여 정렬하는 방법의 예시로는 merge sort가 있는데, 이 merge sort는 더 이상 쪼개지지 않을 때까지 데이터를 나눈 후에 합치며 정렬하는 방식을 사용합니다. 이 정렬 방식을 사용하여 데이터를 정렬했을 때와 bubble sort를 사용해 정렬했을 때와 비교하면 전자가 혁신적으로 속도가 빠릅니다. 하드웨어의 성능 차이를 극복할 정도의 효과를 내는 효율적인 알고리즘은 그 자체로 하나의 기술임을 보여주는 예입니다.

여기서 가장 중요한 개념은 재귀 함수, 재귀 알고리즘입니다. 재귀의 개념은 자기 자신을 호출하는 것으로, 재귀 개념을 이용하여 divide와 conquer을 반복하는 merge sort가 작은 시간복잡도를 가질 수 있는 근거가 됩니다. 재귀 개념은 정렬 뿐만 아니라 알고리즘과 컴퓨터 프로그래밍에서 다양하고 많이 사용되는 것으로, 적절한 활용과 올바른 이해가 필요합니다.

중간고사 발표 마치겠습니다. 감사합니다.