<알고리즘>

9장 요약문

# NP 이론

Branch and Bound 알고리즘에서 배웠던 knapsack algorithm을 기억해보자. 우리는 knapsack 문제를 두개로 나눠서 배웠다. 하나는 fractional knapsack 문제였고, 다른 하나는 1-0 knapsack 문제였다. 가방에 들어갈 item을 쪼개어 넣을 수 있었던 fractional knapsack 문제는 비교적은 적은 복잡도와 시간을 소요하여 문제의 해답을 구할 수 있다고 했다. 반면에, 1-0 knapsack 문제처럼 item을 쪼갤 수 없어 넣거나 넣지 않을 경우만 있을 때에는 문제가 상당히 복잡해질 뿐만 아니라, bound와 같은 추가적인 개념이 필요하다고 했다.

지난시간에는 1-0 knapsack 문제처럼 해결하기 어려운 문제가 NP문제라고 간단하게 듣고 넘어갔다. 이번 시간에는 NP가 무엇인지, P는 무엇인지, 그동안 존재정도만 파악했던 polynomial이 무엇인지에 대해서 알아보자.

NP이론이라는 것은 복잡한 정도에 따른 문제들의 분류를 말한다. polynomial이라는 것은 지수함수가 아닌 함수를 뜻한다. nlgn처럼 n^2이 되지 않는 수를 말한다. 우리가 앞선 알고리즘 시간에서 배웠던 것처럼 polynomial 시간인 알고리즘은 대체로 문제의 크기가 큰 경우에도 별 무리없이 해답을 구하는 것이 가능하다. 이러한 알고리즘을 efficient 알고리즘이라고 한다. 하지만 efficient 알고리즘은 그 증명이 어렵기 때문에 아직 증명을 하지 못한 문제들도 상당수 존재한다. 우리는 이렇게 세상에 있는 모든 문제를 이미 polynomial로 증명된 문제, polynomial이 아니라고 증명된 문제, 아직 증명조차 안 된 문제로 나눌 수 있다. 우리가 집중해야 할 문제는 아직 증명조차 안 된 문제이다.

NP와 P의 개념을 알아보고, NP-EASY와 NP-HARD의 개념을 들으면서 상당히 애매하고 철학적인 의미의 단어들로 설명이 되어있다고 생각했다. 실제로 NP의 설명에서 나오는 non-deterministic 과 decision이 서로 어떻게 다른 지에 대해서 생각해보아야 했기 때문이다.

하지만 우리가 이번 한 학기동안 살펴본 모든 알고리즘들은 단순히 알고리즘 자체로 본 것이 아니라, 하나의 문제로써 그 해결방법과 성능을 위한 시간복잡도, 그것이 최적해를 구할 수 있느냐는 correction analysis까지 종합하여 살펴본 것이다. 따라서 마지막 단원에서 그 모든 문제들에 대해서 각각이 어떤 범주에 속하며, 왜 그렇게 지수함수 시간복잡도에서 벗어나기 위해 수 많은 사람들이 그들의 시간을 갈아 넣었는지 알아볼 수 있을 것이다.