6주차 - 문제 변환하기 (Conversion)

이번 주에 다룰 문제 변환하기는 주어진 문제를 우리가 보다 익숙하게 잘 다룰 수 있는 문제로 변환하여 이미 알고 있는 해법을 적용하는 것입니다. 예를 들어 공간에 관련된 문제나 계층구조에 관련된 문제는 상당히 직관적으로 그래프나 트리로 변환하여 풀 수 있습니다. 우리가 전에 다뤘던 것처럼 트리를 비교하는 문제도 문자열로 변환한 뒤 비교하는 문제로 변환할 수 있습니다.

Infix, Prefix and Postfix Expressions

우리가 일반적으로 알고 있는 수식은 infix expression으로 표현됩니다.

a + b - 3 * c

연산자(operator)가 가운데(in) 와서 이렇게 불립니다. 하지만 프로그래밍 언어 중 일부는 prefix나 postfix expression을 사용하기도 합니다. 이 경우의 장점은 모호함이 존재하는 infix와 달리 명확한 해석이 가능하다는 것입니다. 예를 들어 수학적 지식을 무시한다면 위의 예시는 (a+b) - (3*c)가 될 수도 있고, a + (b-3) * c가 될 수도 있습니다. 이를 prefix로 나타내면, + a - b * 3 c 는 a + (b - 3*c)에 대응되고, - + a b * 3 c 는 (a + b) - (3 * c)에 대응됩니다. 물론 수학적으로 계산하면 연산결과는 같습니다.

여기서 여러분이 하셔야 할 것은 주어진 infix expression을 prefix/postfix expression으로 바꾸는 코드를 작성하는 것입니다. 문제를 단순화하기 위해 연산자는 binary operator인 +, -, * 만 존재한다고 가정하고, 일반적인 수식처럼 *를 +/-보다 먼저 계산하도록 합니다. ( )가 포함되는 경우는 조금 더 복잡해지는데, 기본 아이디어는 같으므로 이 부분은 해보셔도 좋고 넘어가도 좋습니다.

이 문제는 익숙한 이름에서도 알 수 있듯이 주어진 수식을 적절히 트리로 변환하고 방문하며 만나는 노드의 값들을 문자열에 이어 붙이는 것으로 쉽게 해결할 수 있습니다. 단순히 문자열을 다루면 되는 문제이므로 트리의 노드를 방문하는 것 대신 자료구조를 잘 활용하여 노드 구현없이 간단히 처리하는 것도 가능합니다. 잘 생각하면 크게 어렵지 않으므로 가능하면 인터넷 등을 참고하지 말고 스스로 문제를 해결해 보시기 바랍니다.

다음은 테스트에 사용할 수 있는 간단한 예제입니다.

a + b * c - 1 + d * 5

+ a * b c - + 1 * d 5

a b c * + 1 - d 5 * +

택배 물류센터 설립하기

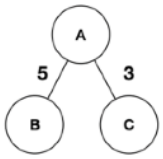
우리가 흔히 이용하는 택배는 한 위치와 다른 위치를 직접 연결하는 방식이 아닌 중간에 거점이 되는 물류센터를 설립하고, 이 물류센터로 화물을 모은 후, 각각의 위치로 다시 배송하는 형태입니다. 여기서 우리가 해결하고 싶은 문제는 주어진 위치에 대해 최적의 물류센터 위치를 찾는 것입니다. 최적의 물류센터 위치는 모든 위치에서 접근이 가능하고, 이 모든 위치를 연결하는 전체비용이 최소가 되는 곳입니다.

cost.txt 파일에는 각 위치별 이동에 드는 비용이 저장되어 있습니다. 예를 들어 A에서 B로 갈 때 드는 비용이 5라면 A-B,5가 됩니다. cost.txt 파일에서 주어진 비용을 고려하여 최적의 물류센터 위치를 찾고 이 때 모든 지점을 연결하는 비용의 총합을 계산하는 코드를 작성하세요. 이미 우리가 다뤘던 알고리즘을 적절히 활용하면 어렵지 않게 해결이 가능하므로, 어떤 것이 유용할 지 생각해보세요.

비용절감 양산체제 구축

한 기업의 공장에서 생산하는 제품은 고객의 요구에 따라 맞춤 주문제작이 가능합니다. 이 기업은 비용 절감을 위해 자신들의 제품을 A~G까지 7종류의 모듈을 적절하게 연결하여 구성할 수 있도록 개발했습니다. 모듈 연결방식은 상위 모듈에 몇 개의 하위 모듈이 연결되는 계층구조로, 트리로 나타낼 수 있습니다.

문제는 각 모듈을 연결하는데에도 비용이 든다는 것입니다. 추가적인 연구개발 결과, 모듈의 특정 연결 형태를 양산할 수 있도록 공장에 라인을 부설하면 이 형태의 모듈연결에 드는 비용을 50%로 줄일 수 있다는 것을 확인하였습니다.



예를 들어, 왼쪽과 같은 A, B, C 모듈을 연결하는데 드는 비용은 8인데, 이와 같은 형태의 모듈연결 형태를 양산하기로 결정하면 이 비용을 4로 줄일 수 있습니다. 공간 및 비용의 제약으로 공장에 추가적인 라인을 부설하는 것은 최대 5개로 제한되며, 각 라인에서는 단 1종류의 고정적인 모듈연결 형태만 제작할 수 있습니다. 형태가 동일한 경우, 모듈연결에 드는 비용은 항상 같습니다.

orders.txt에는 이 기업에서 고객에게 받았던 주문 1,000건이 트리 형태로 저장되어 있습니다. 각 줄에는 1건의 주문이 들어가며, 위의 예제와 같은 트리는 1:A-2:B,5|1:A-3:C,3처럼 edge를 | 로 구분하고, 각각의 노드에는 번호를 붙이며, 그 종류는 콜론을 이용하여 표시하는 형태로 저장되어 있습니다.

현재까지 받았던 주문내용을 분석하여, 5개의 라인에서 생산해야 하는 모듈의 연결형태를 찾고, 이를 이용해 기존의 주문 1,000건을 처리하였을 때 절약 가능한 비용을 계산하는 코드를 작성하세요.

이 문제는 frequent tree pattern mining 또는 frequent subtree mining 문제로, 가장 빈번하게 나타나는 subtree를 찾는 문제입니다. 주어진 트리들에서 각각의 노드를 루트 노드로 하는 subtree를 모두 고려하여 이 중 가장 많이 나타난 것을 찾으면 됩니다. 지금까지 공부했던 내용을 잘 활용하면 어렵지 않게 해결할 수 있습니다.