

“괄호의 값 비교” 문제 풀이

작성자: 김준겸

부분문제 1

길이가 6 이하인 괄호열의 개수는 8개로, 나열하면 (), (()), ()(), ()(), ((())), ()(()), ()()() 이다. 따라서 이 모든 괄호열에 대해 손으로 괄호값을 계산한 후 이를 프로그램에 직접 작성하여 값을 비교할 수 있다. 참고로, 이 중 ()()() 을 제외한 7개의 괄호열의 경우 문제 설명에서 괄호값을 계산해주었다.

부분문제 2

각 괄호열의 길이가 50 이하이므로, 괄호값이 최대가 되는 경우는 (25개가 온 뒤) 25개가 오는 경우이다. 이때 괄호값은 2^{24} 로, 32비트 정수 범위 안이다. 따라서 어떠한 방법으로든 괄호값을 직접 계산하여 문제를 해결할 수 있다.

일반적인 방법은 앞에서부터 괄호를 보며 중첩된 횟수를 바탕으로 괄호값을 계산하는 것이다. 괄호열의 괄호값은 모든 부분 괄호열 ()에 괄호가 씌워진 횟수만큼 2가 거듭해 곱해진 값을 모두 더함으로서 계산된다. 주어진 괄호열이 올바른 괄호열임이 보장되므로, 부분 괄호열 ()가 있을 때 이 괄호열은 최종적으로 앞에 있는 여는 괄호의 개수에서 닫는 괄호의 개수를 뺀 것 만큼 거듭제곱한 값을 가진다. 이를 매번 계산하면 괄호열 길이 N 에 대해 $O(N^3)$ 이고, 2의 거듭제곱을 미리 계산해두거나 비트 연산을 사용하면 $O(N^2)$ 이다. 이에 더해 부분 합을 사용하면 $O(N)$ 에 해결할 수 있다. 스택을 사용하거나, 중첩된 횟수를 관리하여 앞에서부터 괄호열을 훑으며 여는 괄호를 만나면 중첩된 횟수를 하나 증가시키고, 닫는 괄호를 만나면 중첩된 횟수를 하나 감소시키는 방법(스위핑 기법)으로도 $O(N)$ 에 해결할 수 있다.

제한이 상당히 작으므로, 재귀함수나 언어 내장 문자열 처리 함수를 이용해서 문제에서 제공한 정의에 따라 직접 파싱을 시도할 수도 있다. 보통 이 경우 시간복잡도는 $O(N^2)$ 이다.

부분문제 3

괄호열의 길이가 길므로 이 부분문제부터는 괄호열을 직접 계산하려고 시도하면 오버플로우가 발생할 수 있다. 단순 괄호열의 성질을 이용하면, 괄호값을 십진수의 형태로 명시적으로 계산하지 않아도 두 괄호열을 비교할 수 있다. 길이가 $2N$ 인 단순 괄호열의 괄호값은 2^{N-1} 이고, 이어붙인 모든 단순 괄호열의 괄호값이 다르므로 각 괄호열의 괄호값은 서로 다른 2의 거듭제곱의 합 꼴로 나타난다.

$2^0 + 2^1 + \dots + 2^{n-1} < 2^n$ 이므로, 두 괄호값에서 나타나는 가장 큰 2의 거듭제곱 꼴이 다르다면 큰 쪽이 반드시 크다. 만약 가장 큰 2의 거듭제곱 꼴이 같다면, 둘 다 소거하고 남은 거듭제곱 꼴끼리 비교할 수 있다. 모든 거듭제곱 꼴이 같다면 두 괄호열이 같다. 시간복잡도는 괄호열 길이의 합을 N 이라고 할 때 $O(N)$ 이다.

위 풀이를 심화된 관점에서 보면, 각 괄호값을 이진수로 나타낼 수 있다. 서로 다른 2의 거듭제곱의 합은 2^i 가 존재할 경우 뒤에서 i 번째 자리를 1, 존재하지 않을 경우 0으로 두면 이진수로 나타나므로 두 이진수를 비교하는 것과 같아진다.

별해로, 큰 수 자료형을 매우 적절히 사용하면 오버플로우를 내지 않고 위 서브태스크를 계산할 수 있다. 모든 단순 괄호열의 길이가 다르므로 일어나는 덧셈의 횟수는 많아야 괄호열의 길이 N 에 대해 $O(\sqrt{N})$ 이다. 큰 수 자료형에서 덧셈 연산을 수행할 때의 시간복잡도는 수의 크기 X 에 대해 $O(\lg X)$ 이고, 거듭제곱의 경우 단순하게 계산할 경우 필요한 계산량은 매우 많지만 계산해야 하는 값이 모두 2의 거듭제곱 꼴이라는 점을 이용해서 연산에 필요한 비용을 크게 낮출 수 있다. 위와 같은 최적화를 실제로 구현하거나, 언어에서

기본적으로 제공하는 기능을 잘 사용하면 부분문제 3을 부분문제 2의 $O(N)$ 풀이와 동일한 해법을 이용해 해결할 수 있다.

부분문제 4

부분문제 2에서 본 것과 같이, 팔호값은 2의 거듭제곱의 합 꼴로 나타난다. 단 부분문제 3과 같이 서로 다른 거듭제곱의 합 꼴이 아니므로, 부분문제 3과 같이 단순하게 해결할 수는 없다.

먼저 팔호값을 부분문제 3과 같이 2의 거듭제곱의 합 꼴로 나타내되, 부분문제 2에서 사용한 $O(N)$ 계산 방식을 사용하여야 한다. 문제는 $2^x + 2^x = 2^{x+1}$ 과 같이, 팔호값을 구성하는 각 2의 거듭제곱들이 달라도 실제 값은 같을 수 있다는 점이다. 따라서 두 팔호값을 비교하기 위해서는 기존 거듭제곱의 합 꼴을 비교에 더 용이한 꼴로 바꿀 필요가 있다.

만약 한 팔호열의 팔호값을 2의 거듭제곱 꼴로 나타내었을 때 2^x 가 합에 존재할 경우 유일하다고 가정하자. 이 경우 부분문제 3의 경우와 같으므로 같은 방법을 사용하여 두 팔호열의 값을 비교할 수 있다. 이 꼴은 2^i 가 합에 2개 이상 존재할 때, 2^i 가 0개 또는 1개만 남을 때까지 2개씩 계속 합쳐 2^{i+1} 로 만드는 것으로 수행할 수 있다. 만약 i 를 0부터 충분히 큰 수, 이 문제에서는 팔호열의 길이 $2N$ 에 대해 $N - 1$ 까지로 하여 위 작업을 반복한다면 $O(N)$ 에 전체 과정을 완료할 수 있다. 따라서 문제를 해결할 수 있다.

역시 위 풀이를 심화된 관점에서 보면, 이진수의 덧셈을 받아올림 하는 것으로 간주할 수 있다. $2^i + 2^i = 2^{i+1}$ 을 $10\dots00_{(2)} + 10\dots00_{(2)} = 100\dots00_{(2)}$ 로 나타낼 수 있는 점에 주목하자. 어떤 수를 이진수로 표현하는 방식은 유일하므로, 각 팔호열을 이진수로 나타낸 다음 비교한다고 생각할 수 있다. 다수의 십진수를 더 한 다음 받아올림할 때처럼, 특정 자리수를 모두 더한 뒤 2 이상이라면 2로 나눈 나머지를 쓰고 나눈 몫을 받아올림하면 계산 가능하다.