"지그재그" 문제 풀이

작성자: 김준원

부분문제 1

가장 단순한 방법으로 문제를 푼다. $f(x, y, z)$를 구하기 위해, 수열 A의 가능한 모든 부분수열을 직접 살펴보면서 조건 1., 2., 3.을 만족하는지 확인한다. 길이 N인 수열 A의 부분수열은 총 2^N가지 존재한다(각 원소에 대해서 2가지 선택지 - '고른다'와 '고르지 않는다'가 도착하므로). 각각에 대해 반복문을 한 번 이용해 $O(N)$으로 세 조건을 모두 판별할 수 있다. 세 조건을 모두 만족하는 부분수열 중 가장 긴 부분수열의 길이가 $f(x, y, z)$가 될 것이다. 함수 f의 $O(N^3)$가지 값을 모두 구해야 하므로, 총 시간 복잡도는 $O(N^4 2^N)$이다.

어떤 수열이 조건 1., 2.를 만족하는지는 쉽게 판별할 수 있으므로, 두 조건을 만족하는 부분수열만을 순회해서 조건 3.을 검사하는 방법도 있을 것이다.

부분문제 2

부분문제 1에서는 총 2^N가지의 부분수열을 일일히 순회했다. 여기서 병목은 조건 3.이었는데, '어떤 수열의 가장 긴 지그재그 부분수열'을 더 빠르게 구할 수 있는 방법이 존재한다면 시간 복잡도를 크게 개선할 수 있을 것이다.

길이 K인 수열 $S_1, \ldots, S_K$의 가장 긴 지그재그 부분수열은 다음과 같은 간단한 그리디 알고리즘을 이용해 구할 수 있다.

1. 가장 긴 지그재그 부분수열을 담을 배열 arr를 준비한다. 최초에 arr은 빈 배열이다.
2. $i = 1, 2, \ldots, K$를 순서대로 순회하며,
   (a) arr의 끝에 S_i를 추가한 뒤에도 arr이 지그재그 수열이라면, arr의 끝에 S_i를 추가한다.
   (b) 그렇지 않다면, arr의 마지막 원소를 S_i로 대체한다.

이 알고리즘의 정당성은 직관적으로도 눈에 보이지만, 아래 부분문제 4에 대한 문단에서 구체적으로 논의할 것이다.

이로써 각 x, y, z에 대해 $f(x, y, z)$를 $O(N)$의 시간 복잡도에 구할 수 있다. 먼저 조건 1., 2.를 필터링한 다음 위 알고리즘을 이용하면, 지그재그 수열이어야 한다는 조건 (조건 3.)까지 만족하는 가장 긴 부분수열을 구할 수 있다. 함수 f의 $O(N^3)$가지의 값을 모두 구한 뒤 문제의 답을 구할 수 있으므로, 총 시간 복잡도는 $O(N^4)$이다.

부분문제 3

부분문제 2의 풀이를 개선하여 부분문제 3을 풀 수 있다. 부분문제 2의 풀이에서는 매번 비슷하게 계산되는 $O(N^3)$가지의 값을 구하고 있다는 것을 관찰할 수 있다. 슬라이딩 윈도우 기법을 사용하여 중복되는 계산들을 최적화할 수 있으며, $f(x, y, y+1), f(x, y, y+2), \ldots, f(x, y, N)$까지의 값들을 한 번의 반복으로 구할 수 있다. 시간 복잡도가 $O(N^3)$이 되어 부분문제 3을 해결할 수 있다.

부분문제 4

부분문제 4부터의 부분문제들을 해결하려면 부분문제 3까지와 다른 관점에서 문제를 바라보아야 한다. 각 f의 값들을 일일히 계산한 뒤 합쳐서 g의 값을 만들지 말고, g의 값을 직접 계산해보자. 여기에는 더 많은 관찰이 필요하다.

부분문제 2에서 제시한 가장 긴 지그재그 부분수열을 계산하는 알고리즘을 관찰하면, 수열 $S_1, \ldots, S_K$의 가장 긴 지그재그 부분수열의 길이를 계산하는 간단한 공식을 알 수 있다. (a)가 실행될 때마다 arr의 길이가 하나씩 증가한다. 다른 말로, 총 반복 횟수 K에서 (b)가 실행되는 횟수를 빼면 알고리즘이 끝난 뒤 arr가 된다. 즉, $A_y, \ldots, A_z$에서 x 이하인 수들을 모아서 만든 수열을 C라 하면, $f(x, y, z)$의 값은

1. C의 길이
2. $C_i < C_{i+1} < C_{i+2}$를 만족하는 i의 개수
3. $C_i > C_{i+1} > C_{i+2}$를 만족하는 i의 개수

일 때 $1. - 2. - 3.$가 된다. $f(x, y, z)$의 합으로 정의되는 $g(x)$도 1.를 모두 합한 값에서 2., 3.를 모두 합한 값을 빼서 계산할 수 있다. 이 때, 고정된 x에 대해 가능한 y, z의 값을 모두 고려해 계산한다는 점에 유의하라. 즉, 위에서 정의한 부분수열 C는, 모두 A에서 x 이하인 수들을 모은 부분수열을 잘라 만든 수열이다. A에서 x 이하인 수를 모은 배열을 $B_1, \cdots, B_K$로 두고, 각 원소 B_j의 배열 A에서의 위치를 P_j로 두자.

$g(x)$와 같은 값을 계산할 때에 유용하게 쓰이는 계산법은, 수열의 각 원소에 대해 해당 원소가 세어지는 횟수를 합하는 것이다. 각 j에 대해 B_j를 몇 번 세게 되는지 계산하자. 먼저, B_j를 1.에서 세는 경우는 $y \le P_j \le z$인 경우이며, 총 $P_j \times (N - P_j + 1)$번이다. 2.에서 세는지 여부는 $B_j < B_{j+1} < B_{j+2}$인지에 달려 있다. 이 조건이 성립한다면, B_j를 $P_j \times (N - P_{j+2} + 1)$번 세고, 그렇지 않다면 2.에서는 B_j를 세지 않는다. 3.에서도 $B_j > B_{j+1} > B_{j+2}$가 성립한다면 $P_j \times (N - P_{j+2} + 1)$번 세고, 그렇지 않으면 세지 않는다.

위를 각각 계산한 다음 $1. - 2. - 3.$를 하면 $g(x)$에서 B_j의 '기여도'가 된다. $B_1, B_2, \ldots, B_K$에 대해 이 값들을 모두 합하면 $g(x)$의 값을 얻을 수 있다. 이를 별도의 최적화를 거치지 않고 구현하면 $O(N^3)$의 시간 복잡도로 부분문제 3까지 해결할 수 있고, 집합 자료구조 (예를 들어 C++의 std::set)을 이용해 구현하면 $O(N^2 \log N)$의 시간 복잡도로 부분문제 4까지 해결할 수 있다.

부분문제 5

부분문제 4와 같이 각 원소의 '기여도'를 세는 접근법은 이 풀이를 선형으로 개선할 때에도 그대로 써먹을 수 있다.

x의 값이 $1, 2, \cdots, N$일 때 $g(x)$의 값을 순서대로 계산하는 과정에서, x의 값이 1 증가해서 $x + 1$이 되는 순간을 일종의 쿼리로 생각할 수 있다. x가 증가하여 $x + 1$이 되는 것은 수열 B의 어딘가에 $x + 1$이 하나 추가된 것과 같다. 따라서 $g(x + 1)$의 값을 계산할 때에는, $g(x)$의 값에서 새로 추가된 $x + 1$의 '기여도'를 합해주면 된다. 이는 부분문제 4에서 소개한 방법으로 계산할 수 있다. 수열의 가운데에 $x+1$이 추가됨으로써 (추가된 위치 주변의) 다른 원소의 '기여도'가 변화하는데, 이 또한 계산에 포함시켜야 한다는 점 잊지 말자.

위 아이디어를 이용해 부분문제 4를 푸는 시간 복잡도에서 N을 하나 제거할 수 있으며, $O(N \log N)$의 시간 복잡도로 전체 문제를 해결할 수 있다.