컴퓨터공학과 20211507 김동건

Dynamic Programming을 사용하여 더 효율적으로 풀 수 있는 문제 2개 제시 및 BruteForce 대비 어느정보 빨라지는 지 설명

첫 번째 문제로는 파스칼의 삼각형이 있다. 파스칼의 삼각형이란 위에서부터 1행이라고 할 때, 각 행 i에 i개의 원소가 있고, 그 값은 바로 위 행에서 자신 위에 있는 두 값의 합이다. 다음 사진은 n = 6일때, 파스칼 삼각형의 예시이다.  
폰트, 타이포그래피, 화이트, 디자인이(가) 표시된 사진

자동 생성된 설명

i번째 행의 j번째 파스칼 수를 f(i, j)라고 하자. f(i, j)를 구하기 위해서는 f(i -1, j -1) + f(i - 1, j)이 필요하고, f(i -1, j -1)와 f(i - 1, j)를 구하기 위해서 또 그 윗 줄의 값이 필요하고 결국 한 자리를 구하기 위해 O(2^i)의 계산이 필요하다. 따라서 이러한 브루트포스 방식은 전체 수의 개수인 n^2에 2^n이 곱해진 O(n^2 \* 2^n)의 시간복잡도가 든다. 그런데, f(i - 1, j -1)에서도 f(i - 2, j -1)이 필요하고, f(i - 1, j)에서도 f(i - 2, j -1)이 필요하다. 즉, 같은 계산을 여러번 하게 된다. 따라서 우리는 메모이제이션을 적용하여 이를 단 한 번만 계산하게 할 수 있다. 따라서 각 수를 구할 때 연산은 O(1)이고 각각 단 한 번만 계산되기 때문에 전체 시간복잡도 O(n^2)에 문제를 해결 가능하다. 위에서 말한 것과 같이 메모이제이션을 통해, 브루트포스 방식의 시간복잡도인 O(n^2 \* 2^n)보다 훨씬 빠른 O(n^2)에 해결이 가능하게 되었다.

두 번째 문제로는 양의 정수들로 이루어진 수열이 주어졌을 때, '가장 큰 증가하는 부분 수열'을 찾는 문제가 있다. 예를 들어, A = [1, 100, 2, 50, 60, 3, 5, 6, 7, 8]이 주어졌을 때, 이 수열의 가장 큰 증가하는 부분 수열은 [**1**, 100, **2**, **50**, **60**, 3, 5, 6, 7, 8]이고, 합은 113이다. 가장 먼저 기본적인 브루트포스 방식을 사용해보자. 모든 부분 수열에 대해 증가수열인지 확인하여 합을 계산하면 된다. 이때, 모든 부분 수열의 개수는 2^n이고, 각 부분 수열이 증가수열인지 확인하는 데에 O(N)의 시간이 든다. 따라서 전체 시간복잡도 O(2^n \* n)에 문제를 해결할 수 있다.

여기서 dynamic programming을 사용해보자. dp[i]를 'i번째 원소가 부분수열의 마지막 원소인 부분 수열들 중 합이 최대인 부분 수열의 합'이라고 정의하자.   
 이제 dp 테이블을 앞에서부터 순차적으로 채워나간다고 생각해 보자. 즉, dp[i]를 구할 때, i보다 작은 범위의 dp 테이블들은 이미 다 채워져있다고 생각하면 된다. 그러면 dp[i]는 다음과 같은 식을 통해 구할 수 있게 된다.   
친필, 폰트, 텍스트, 라인이(가) 표시된 사진

자동 생성된 설명

(계산의 편의를 위해 주어진 수열을 1-based로 받고, A[0]을 0으로 설정해두자. 이렇게 한다면 i가 처음이자 마지막 원소인 경우에 대한 예외처리를 따로 해주지 않아 구현이 편해진다.)  
dp[i]를 구할 때, 앞에 있는 dp값들을 모두 본다. 즉, 각 dp[i]를 구할 때, O(i)가 걸린다. 따라서 전체 시간복잡도는 O(n^2)이다. 브루트포스 방식으로는 O(2^n \* n)인 것에 비해 굉장히 빨라지는 것을 확인할 수 있다.