DP 다이나믹 프로그래밍 개념

다이나믹 프로그래밍(Dynamic Programming)

메모리를 적절히 사용하여 수행 시간 효율성을 비약적으로 향상시키는 방법.

이미 계산된 결과(작은 문제)는 별도의 메모리 영역에 저장하여 다시 계산하지 않도록 함.

동적 계획법 이라고도 함.

다이나믹 프로그래밍은 아래의 두가지 방식으로 구성됨.

- 탑다운
- 보텀업

일반적으로 프로그래밍에서의 다이나믹은 '프로그램이 실행되는 도중에'라는 의미를 갖고 있으나,

다이나믹 프로그래밍에서의 다이나믹은 별 의미 없이 사용된 단어임.

프로그래밍에서의 다이나믹과 같은 의미가 아니라는 것 정도만 기억하면 됨.

다이나믹 프로그래밍의 조건

1. 최적 부분 구조 (Optimal Substructure)

   : 큰 문제를 작은 문제로 나눌 수 있으며 작은 문제의 답을 모아서 큰 문제를 해결할 수 있음

2. 중복되는 부분 문제 (Overlapping Subproblem)

   : 동일한 작은 문제를 반복적으로 해결해야함

예시. 피보나치 수열

1. 최적 부분 구조

   : f(4) = f(3) + f(2)

2. 중복되는 부분 문제

   : f(4) = f(3) + f(2) = f(1) + f(2) + f(2)

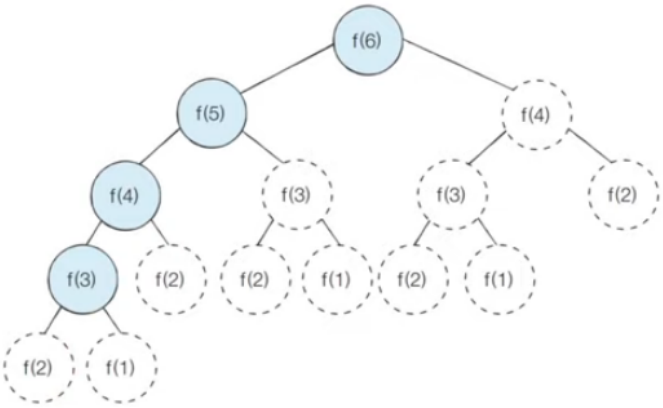
메모제이션 (Memoization)

한 번 계산한 결과를 메모리 공간에 메모하는 기법

다이나믹 프로그래밍을 구현하는 방법 중 하나

- 같은 문제를 다시 호출하면 메모했던 결과를 그대로 가져옴
- 값을 기록해 놓는다는 점에서 캐싱(Caching)이라고도 함

메모제이션 동작 분석



탑다운 VS 보텀업

탑다운(하향식) : 큰 문제를 해결하기 위해 작은 문제를 재귀적으로 호출하여 해결. 한번 계산 된 결과를 기록하기 위해 메모제이션 방식 사용.

보텀업(상향식) : 작은 문제부터 차근차근 해결해 나가면서 먼저 계산된 값을 활용해서 문제를 해결해 나아감. 반복문을 이용함.

다이나믹 프로그래밍의 전형적인 형태는 보텀업 방식.

결과 저장용 리스트는 **'DP 테이블'**이라고 부름

다이나믹 프로그래밍 VS 분할 정복

다이나믹 프로그래밍과 분할 정복은 모두 **최적 부분 구조**를 가질 때 사용할 수 있음.

다이나믹 프로그래밍과 분할 정복의 차이점은 **부분 문제의 중복**.

- 다이나믹 프로그래밍 문제에서는 각 부분 문제들이 서로 영향을 미치며 부분 문제가 중복 됨.
- 분할 정복 문제에서는 동일한 부분 문제가 반복적으로 계산 되지 않음.

분할 정복

예시. 퀵 정렬

한 번 기준 원소가 자리를 변경해서 자리를 잡으면 그 기준 원소의 위치는 바뀌지 않음.

분할 이후에 해당 원소를 다시 처리하는 부분 문제는 호출하지 않음.

DP 문제에서의 핵심

- 주어진 문제가 **다이나믹 프로그래밍 유형임을 파악**하는 것이 중요
- 가장 먼저 그리디, 구현, 완전 탐색 등의 아이디어로 문제를 해결할 수 있는지 검토.
  - 다른 알고리즘으로 풀이 방법이 떠오르지 않으면 다이나믹 프로그래밍을 고려
- 일단 재귀 함수로 비효율적인 완전 탐색 프로그램을 작성한 뒤에 (탑다운) 작은 문제에서 구한 답이 큰 문제에서 그대로 사용될 수 있으면, 코드를 개선하는 방법을 사용
- 일반적인 코딩 테스트 수준에서는 기본 유형의 다이나믹 프로그래밍 문제가 출제되는 경우가 많음.