21.08.23 다이나믹 프로그래밍

- 다이나믹 프로그래밍: 메모리를 적절히 사용하여 수행 시간 효율성을 비약적으로 향상시키는 방법.

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

- 완전탐색으로 구현하는 경우라도, 이미 계산된 결과를 따로 저장하는 다이나믹 프로그래밍을 이용한다면, 시간 복잡도를 획기적으로 줄일 수 있음.

- 다이나믹 프로그래밍의 구현: 두 가지 방식(Top-Down(하향식) / Bottom-Up(상향식))

- 동적 계획법이라고 불리기도 함.

- 일반적인 프로그래밍 분야에서의 동적(Dynamic)과의 차이점

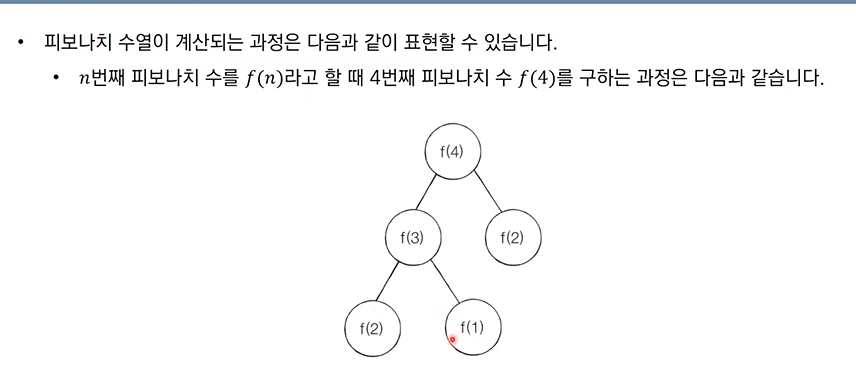
- 자료구조에서의 동적 할당(Dynamic Allocation)은 ‘프로그램이 실행되는 도중에 실행에 필요한 메모리를 할당하는 기법’을 뜻함.

- 다이나믹 프로그래밍(동적 계획법)에서의 ‘동적(Dynamic)’은 **별다른 의미없이 사용된 단어**.

**다이나믹 프로그래밍의 조건**

* 다이나믹 프로그래밍은 문제가 다음의 조건을 만족할 때 사용 가능
  + **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 해결해야 함.

EX) 피보나치 수열

* 형식: 1,1,2,3,5,8,13,21,34,55,89, …
* 점화식(인접한 항들 사이의 관계식)으로 표현:
* 프로그래밍에서는 이러한 수열을 배열이나 리스트를 이용해 표현.
* 
* 소스 코드(단순 재귀):

#피보나치 수열

def fibo(x):

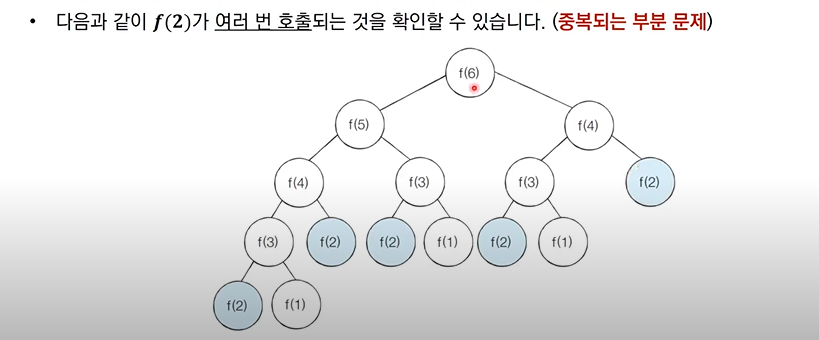
    if x==1 or x==2:

        return 1

    return fibo(x-1)+fibo(x-2)

print(fibo(4))

* 문제점:
  + 1. 지수시간 복잡도를 가지게 됨.
  + 2. 중복되는 부분 문제

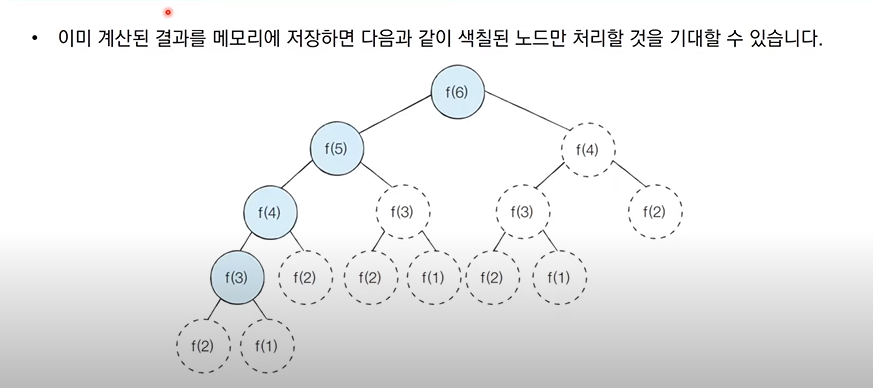


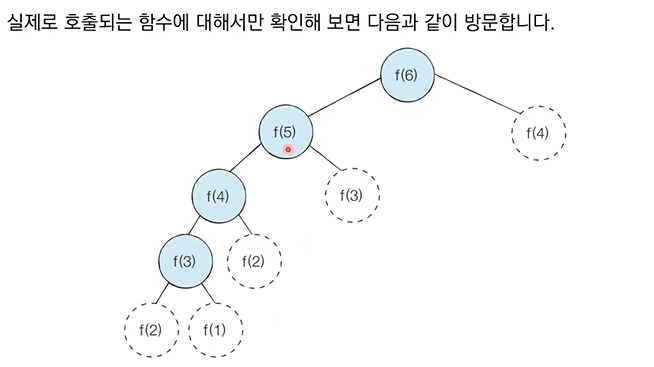
* 해결방안
  + 1. 다이나믹 프로그래밍의 사용 조건(최적 부분 구조, 중복되는 부분 문제)을 만족하는 지 확인.
    - 만족함.

다이나믹 프로그래밍 방법

* 하향식 방법
* 메모이제이션(Memoization)
  + 한 번 계산한 결과를 메모리 공간에 메모하는 기법
    - 같은 문제를 다시 호출하면 메모했던 결과를 그대로 가져옴
    - 값을 기록해 놓는다는 점에서 캐싱(Caching)이라고도 함.
* 탑다운 VS 보텀업
  + 탑다운(메모이제이션): 하향식, 재귀적으로 구현, 계산된 결과를 저장하기 위해 메모이제이션 방식 이용.
  + 보텀업: 상향식, 반복문으로 구현, 전형적인 다이나믹 프로그래밍 형태
    - 결과 저장용 리스트(배열)은 DP 테이블이라고 부름.
  + 엄밀히 말하면 메모이제이션은 이전에 계산된 결과를 일시적으로 기록해 놓는 넓은 개념.
    - 메모이제이션은 다이나믹 프로그래밍에 국한된 개념은 아님.
    - 한 번 계산된 결과를 담아 놓기만 하고 다이나믹 프로그래밍을 위해 활용하지 않을 수도 있음.

EX 풀이) 메모이제이션 동작 분석

* 
* 실제 호출되는 함수만 확인 :



* 메모이제이션을 이용하는 경우 피보나치 수열 함수의 시간복잡도는 O(N)이 됨.
  + 기존 방식이 지수시간(O())을 사용한 것에 비하면 큰 폭으로 성능이 향상됨을 볼 수 있음

다이나믹 프로그래밍 VS 분할 정복

1. 다이나믹 프로그래밍과 분할 정복은 모두 **최적 부분 구조**를 가질 때 사용할 수 있음.
   1. 큰 문제를 작은 문제로 나눌 수 있으며 작은 문제의 답을 모아 큰 문제를 해결할 수 있는 상황.
2. 다이나믹 프로그래밍 과 분할 정복의 차이는 **부분 문제의 중복**에 있다.
   1. **다이나믹 프로그래밍 문제**에서는 각 부분 문제들이 서로 영향을 미치면 **부분 문제가 중복됨**.
   2. 그러나 **분할 정복 문제**에서는 **동일한 부분 문제가 반복적으로 계산되지 않음**.(부분 문제 중복 X)

다이나믹 프로그래밍 문제에 접근 하는 방법

1. 주어진 문제가 다이나믹 프로그래밍 유형인지를 파악하는 것이 중요.
2. 가장 먼저 그리디, 구현, 완전 탐색 등의 아이디어로 문제를 해결할 수 있는지 검토 할 수 있음.

2-1. 문제 해결 방법이 떠오르지 않는 다면 다이나믹 프로그래밍을 고려

1. 일단 재귀 함수로 비효율 적인 완전 탐색 프로그램을 작성한 뒤에 (탑다운) 작은 문제에서 구한 답이 큰 문제에서 그대로 사용될 수 있으면, 코드를 개선하는 방법을 사용.
2. 일반적인 코딩 테스트 수준에서는 기본 유형의 다이나믹 프로그래밍 문제가 출제되는 경우가 많음.