동적 계획법(Dynamic Programming)

다이나믹 프로그래밍이란?

- 큰 문제를 작은 문제로 쪼개서 푸는 기법
- 모든 작은 문제들은 단 한 번만 풀어야 한다
  - 한 번 푼 것을 다시 풀지 않아 비효율적인 알고리즘을 개선시킨다
- 메모이제이션(Memoization)을 사용한다
  - 메모이제이션 → 이미 정답을 구한 작은 문제의 결과를 따로 배열에 저장하고, 나중에 큰 문제를 풀어나갈 때 똑같은 작은 문제가 나타나면 미리 저장해 둔 값을 이용한다
  - 값을 기록해 놓는다는 점에서 캐싱(Caching)이라고 한다
  - DP에만 국한된 개념은 아니고, 한 번 계산된 결과를 담아 놓고 다른 방식으로도 사용될 수 있다

DP 문제가 성립하는 조건

1. 최적 부분 구조(Optimal Substructure)

- 상위 문제를 하위 문제로 나눌 수 있으며, 하위 문제의 답을 모아서 상위 문제를 해결할 수 있다

2. 중복되는 부분 문제(Overlapping Subproblem)

- 동일한 작은 문제를 반복적으로 해결해야 한다

다이나믹 프로그래밍 접근 방법

Bottom-up(정석)

작은 문제부터 구해 나아가 큰 문제를 해결하는 방법

주로 for문을 이용하여 답을 도출한다

반복문을 사용하기 때문에 Top-down 방식에 비하여 효율성이 좋다고 알려져 있지만, 문제 및 설계에 따라 달라진다고 한다

```
public class Bottomup {

  static int[] dp;
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int n = sc.nextInt();
    dp  = new int[n+1];

    System.out.println(fibo(n));
  }

    // Bottom-up
  static int fibo(int x) {
    dp[1] =1;
    dp[2] =1;
    for(int i=3; i<x+1; i++) {
      dp[i] = dp[i-1] + dp[i-2];
    }
    return dp[x];
  }
}
```

Top-down(사파)

재귀 함수를 이용해서 큰 문제를 풀 때 작은 문제를 필요로 하면 그때서야 작은 문제를 해결하는 방법

주로 재귀 함수를 이용한다

재귀 함수를 이용하기에 사파로 여기고 Bottom-up이 진정한 동적계획법이라는 사람도 있다고 함

```
public class Topdown {
  static int[] dp;
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int n = sc.nextInt();
    dp  = new int[n+1];
```

```
    System.out.println(fibo(n));

  }

    // Top-down
  static int fibo(int x) {
    if( x ==1 || x==2) return 1;
    if(dp[x] != 0) return dp[x];
    dp[x] = fibo(x-1) + fibo(x-2);
    return dp[x];
  }
}
```

백준 24416

동적계획법과 재귀함수의 차이를 보여주는 문제

24416번: 알고리즘 수업 - 피보나치 수 1

BAEKJOON ONLINE JUDGE

https://www.acmicpc.net/problem/24416

[백준] 24416 / 알고리즘 수업 - 피보나치 수 1

문제 오늘도 서준이는 동적 프로그래밍 수업 조교를 하고 있다. 아빠가 수업한 내용을 학생들이 잘 이해했는지 문제를 통해서 확인해보자. 오늘은 n의 피보나치 수를 재귀호출과 동적 프로그래밍으로 구하는 알고

velog

https://velog.io/@maxxyoung/백준-24416-알고리즘-수업-피보나치-수-1

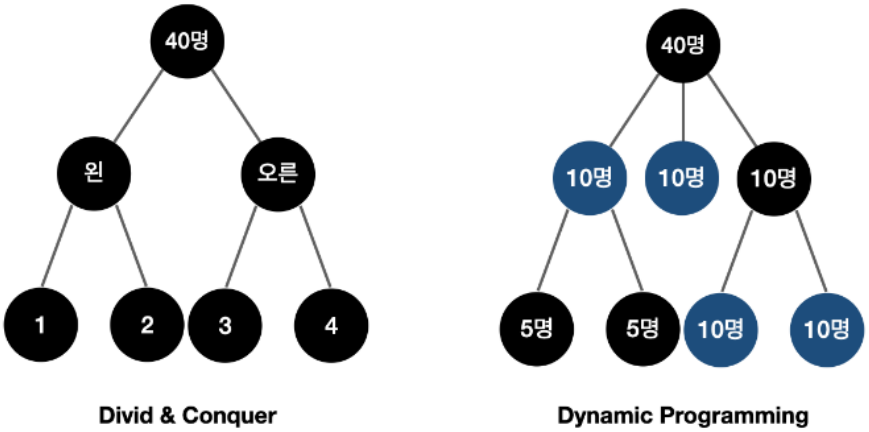
동적 계획법과 분할 정복의 가장 큰 차이점

동적계획법 (DP) VS 분할정복 (Divide and Conquer)

최적 부분 구조는 분할 정복(Divide and conquer) 방식으로 풀 수 있다. DP와 분할 정복은 해당 문제가 최적 부분 구조의 조건을 가질 때 사용할 수 있다. 상위 문제를 작게 하위 문제로 나누어 해결하는 방식으로 처리하면 된다.

그러나! 차이점은 하위 문제의 중복이다.

하위 문제가 독립적이지 않고 중복이 되는 경우에는 DP의 방식이 분할정복보다는 더 나은 시간복잡도를 가진다. 왜냐하면 분할정복에서는 동일한 하위 문제는 각각 독립적으로 구성되어 있기 때문에 반복적으로 계산이 되지 않기 때문이다



D&C, DP 트리

문제 해결을 위해 문제를 잘게 쪼개는 방식은 비슷하다

하지만 동적 계획법의 경우, 그렇게 잘게 쪼개진 부분 문제가 중복되어 상위 문제 해결 시 매우 작은 시간복잡도로 재활용된다!

분할 정복의 부분 문제는 중복되지 않아 Memoization을 사용하지 않는다!

숙제 문제

기본적인 동적 계획법 문제입니다

연관된 문제가 굉장히 많습니다(돌 게임 8까지 있음)

9655번: 돌 게임

탁자 위에 돌 N개가 있다. 상근이와 창영이는 턴을 번갈아가면서 돌을 가져가며, 돌은 1개 또는 3개 가져갈 수 있다. 마지막 돌을 가져가는 사람이 게임을 이기게 된다.

https://www.acmicpc.net/problem/9655

문제집: 돌 게임 (시리즈)

https://www.acmicpc.net/workbook/view/81

강사님이 주셨던 문제 중 두 번째 문제와 연관있는 문제

이번에는 합이 가장 큰 증가 부분 수열을 구하는 문제입니다

11055번: 가장 큰 증가 부분 수열

수열 A가 주어졌을 때, 그 수열의 증가 부분 수열 중에서 합이 가장 큰 것을 구하는 프로그램을 작성하시오.

https://www.acmicpc.net/problem/11055