알고리즘 DP 발표

최우진, 김윤재

DP란 무엇인가?

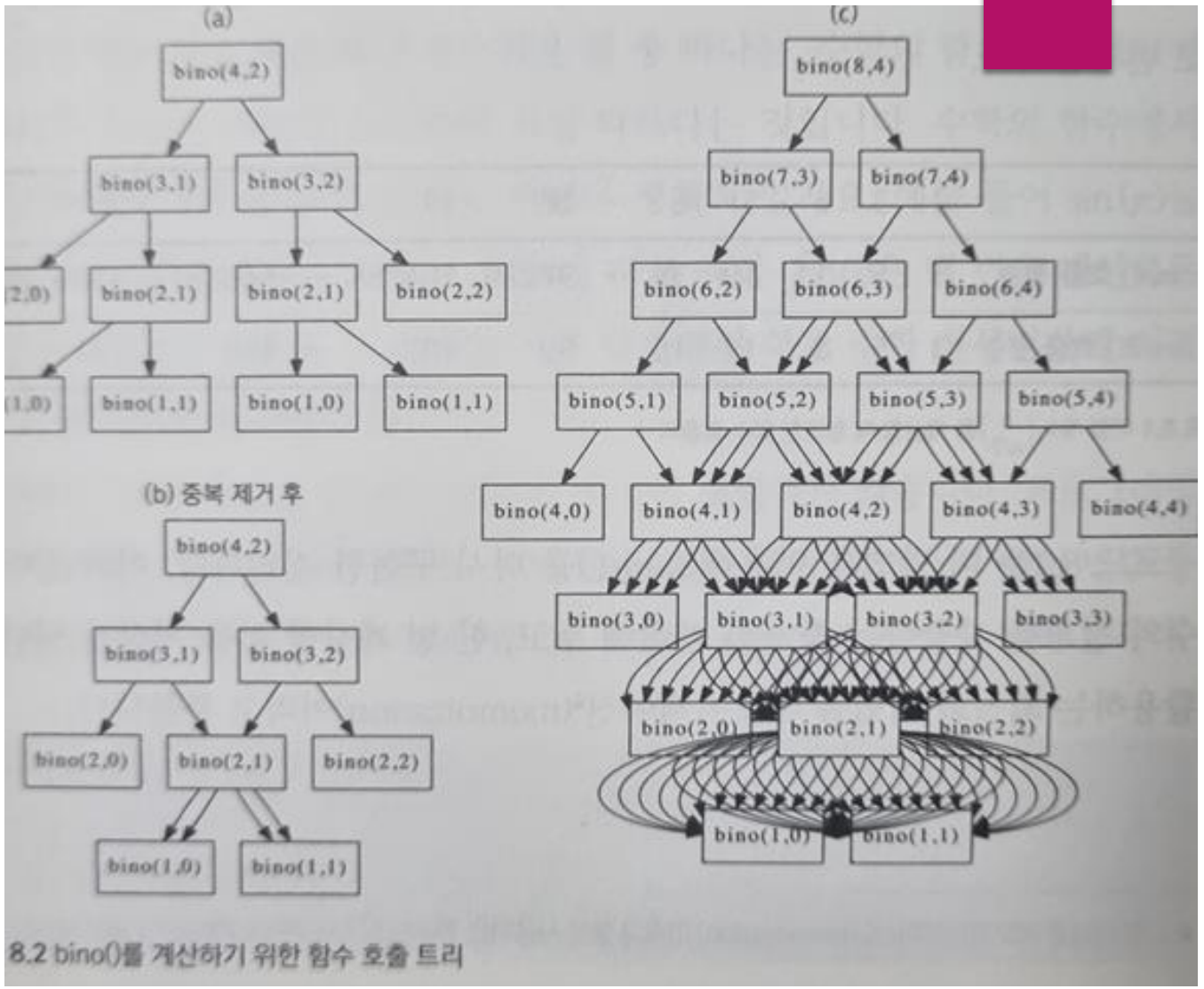
- 복잡한 문제를 간단한 여러 가지의 문제로 나누어 해결하는 방법을 말한다.
- **부분 문제 반복과 최적 부분 구조**를 가지는 알고리즘을 일반적인 방법에 비해 더욱 적은 시간 내에 풀 때 사용한다.

- 예를 들어 피보나치 수열 FIBO(99)같은 경우에는,
  FIBO(99) = FIBO(98) + FIBO(97)이고,
  FIBO(98) = FIBO(97) + FIBO(96) 이다.
  이미 FIBO(97)은 2번 반복이 되었고 FIBO(10)같은 경우에는 셀 수 없이 많이 반복된다. 이럴 때 FIBO(10)의 값을 미리 저장해놓고, FIBO(10)이 필요할 때, 호출한다면 CPU와 시간을 절약할 수 있을 것이다. 이 때의 FIBO(10)이 반복되는 부분 문제이며, 이런 구조가 최적 부분 구조이다.

DP 호출 트리

- 기존에 사용했던 트리 값이 있다면, 해당 값을 불러와서 처리가 가능하다. DP의 가장 큰 장점이다.

- 예를 들어, 피보나치 수열 같은 경우에는 FIBO(N) = FIBO(N-1) + FIBO(N-2)의 구조를 가지므로, DP로 값을 저장해 두었을때 더 빠르게 연산이 가능하다.



8.2 bino()를 계산하기 위한 함수 호출 트리

메모이제이션

- 컴퓨터 프로그램이 **동일한 계산을 반복**해야 할 때, 이전에 **계산한 값을 메모리에 저장**함으로써 동일한 계산의 **반복 수행을 제거**하여 프로그램 **실행 속도를 빠르게 하는 기술**이다.

DP 접근 방법

- **Top-Down**
  - **위에서 아래로 접근하여 큰 문제에서 부분 문제로 쪼개가며, 재귀 호출을 이용하여 푸는 방법**
- **Bottom-Up**
  - **아래에서 위로 접근하여 부분 문제에서부터 문제를 풀어가며 점차 큰 문제를 푸는 방법**

[SILVER I] 쉬운 계단 수(10844번)

```c
#include <stdio.h>

int N;

//그때그때의 실행결과를 담을 배열
long dp[10][101] = {0,};
long cap(int num, int idx);
int main() {
    scanf_s("%d", &N);
    long long hap = 0;
    //계단 수는 대칭적이다. 그러므로 CAP(1)=CAP(8), CAP(2)=CAP(7) 과 같이 정리된다.
    //그러나 수가 0으로 시작할 수는 없으므로, CAP(9)는 따로 더하고, 나머지는 CAP(8) * 2 이렇게 해서 시간과 메모리를 절약한다.
    hap += cap(9, 0) % 1000000000;
    for (int i = 8; i >= 5; i--)
        hap += cap(i, 0) * 2 % 1000000000;
    printf("%lld", hap % 1000000000);
}
long cap(int num, int idx) {
    //설정한 길이의 끝에 도달했을 경우, 1을 반환한다.
    //N = 1 (1,2,3,4,5,6,7,8,9) 9개가 나올 수 있음.
    if (idx == N - 1)
        return 1;
```

```
long cap(int num, int idx) {
    //설정한 길이의 끝에 도달했을 경우, 1을 반환한다.
    //N = 1 (1,2,3,4,5,6,7,8,9) 9개가 나올 수 있음.
    if (idx == N - 1)
        return 1;

    //dp=0일때 밑에를 실행, dp가 있으면 그 값을 리턴.
    if (dp[num][idx] == 0) {

        long temp;
        /*자연수가 0이다 = 계단수를 하나밖에 고를 수 없다.
            자연수가 5면 계단수를 4,6을 고를수있는데
            자연수가 0이면, -1을 고를수 없으므로 1밖에 고를 수 없다.
            그래서 분기를 나눠 처리한 것.
            똑같이, 자연수가 9이면 10을 고를 수 없으므로 분기를 나눴다.
        */
        if (num == 0) {
            temp = cap(1, idx + 1) % 1000000000;
            dp[num][idx] = temp;
            return temp;
        }

        else if (num == 9) {
            temp = cap(8, idx + 1) % 1000000000;
            dp[num][idx] = temp;
            return temp;
        }

        else {
            // 계단 수는 밑으로 1칸, 위로 1칸 진출할 수 있다.
            temp = cap(num + 1, idx + 1) % 1000000000 + cap(num - 1, idx + 1) % 1000000000;
            temp %= 1000000000;
            //계단 수는 대칭을 이루므로, DP[MUM][0]의 값과,DP[NUM][9]의 값은 같다.
            dp[num][idx] = temp;
            dp[9 - num][idx] = temp;
            return temp;
        }
    }
    else {
        return dp[num][idx];
    }
}
```

- Thanks for Watching