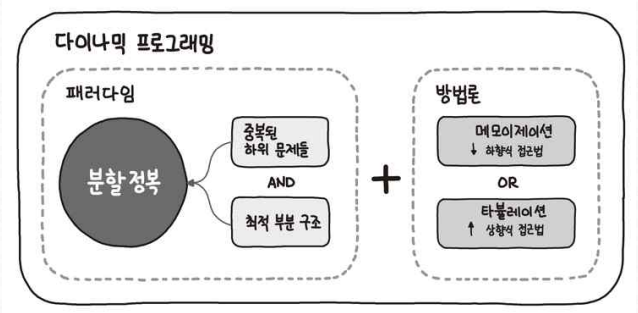
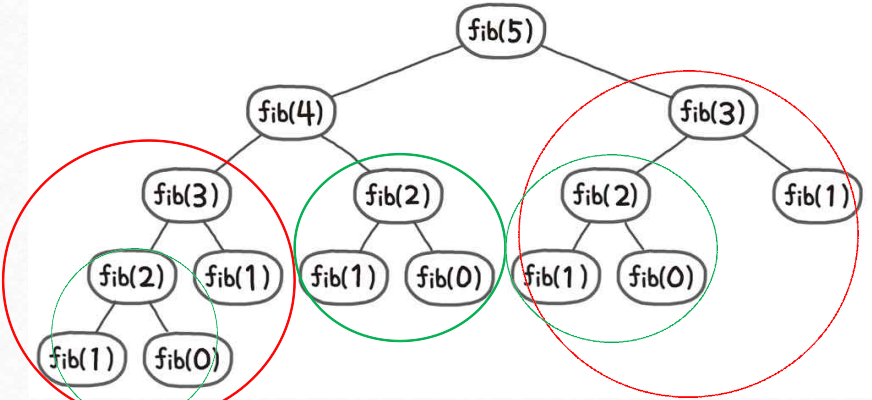
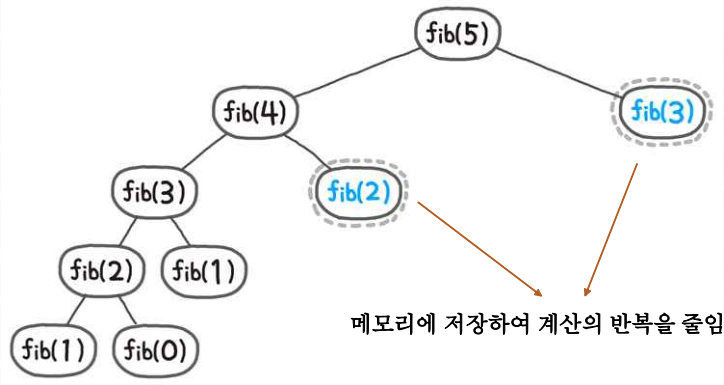
동적계획법 (DP)


다이나믹 프로그래밍
패러다임
분할정복
중복된
하위 문제들
AND
최적 부분 구조
+
방법론
메모이제이션
↓ 하향식 접근법
OR
타뷸레이션
↑ 상향식 접근법


fib(5)
fib(4)
fib(3)
fib(3)
fib(2)
fib(2)
fib(1)
fib(2)
fib(1)
fib(1)
fib(0)
fib(1)
fib(0)
fib(1)
fib(0)


fib(5)
fib(4)
fib(3)
fib(3)
fib(2)
fib(2)
fib(1)
fib(1)
fib(0)
메모리에 저장하여 계산의 반복을 줄임

Top-down 방식

```c
#include <stdio.h>

int dp[100] = { 0, };

int fibo(int n) {
    if (n <= 1) {
        return n;
    } else {
        if (dp[n] > 0) {
            return dp[n];
        }
        dp[n] = fibo(n-1) + fibo(n-2);
        return dp[n];
    }
}

int main() {

    int num = 0;

    printf("항 입력 : ");
    scanf_s("%d", &num);

    printf("%d", fibo(num));
    return 0;
}
```

Bottom-up 방식

```c
#include <stdio.h>

int dp[100] = { 0, };

int fibo(int n) {
    dp[0] = 0;
    dp[1] = 1;
    int i;
    for (i = 2; i <= n; i++ ) {
        dp[i] = dp[i - 1] + dp[i - 2];
    }
    return dp[n];
}

int main() {

    int num = 0;

    printf("항 입력 : ");
    scanf_s("%d", &num);

    printf("%d", fibo(num));
    return 0;
}
```