SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

삼성 청년 SW 아카데미

APS 응용

SAMSUNG SW ACADEMY FOR YOUTH

목차

SAMSUNG SW ACADEMY FOR YOUTH

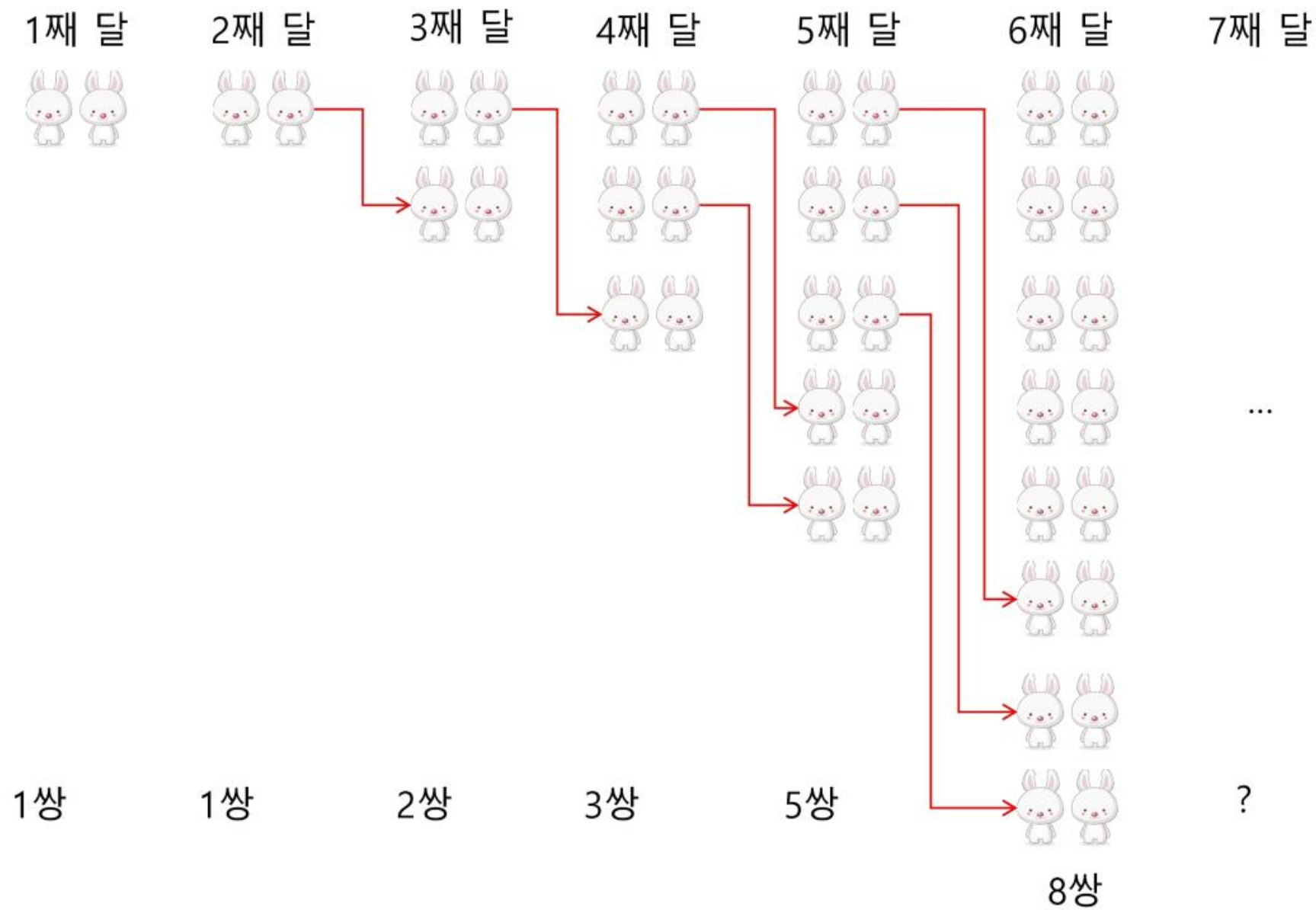
함께가요 미래로!
Enabling People

재귀 호출과 메모이제이션

문제 제시 : 토끼 수 구하기

- **다음과 같은 조건이 있다. n번째 달의 토끼 수는?**
  - 첫 달에는 새로 태어난 토끼 한 쌍만이 존재한다.
  - 두 달 이상이 된 토끼는 번식 가능하다.
  - 번식 가능한 토끼 한 쌍은 매달 새끼 한 쌍을 낳는다.
  - 토끼는 죽지 않는다.

문제 제시 : 토끼 수 구하기



문제 제시 : 토끼 수 구하기

토끼 수 구하기

- *n* 번째 달에 *a* 쌍의 토끼가 있었고
- 다음 *n+1* 번째 달에는 새로 태어난 토끼를 포함해 *b* 쌍이 있었다고 하자.
- 그러면 그 다음 *n+2* 번째 달에는 *a+b* 쌍의 토끼가 있게 된다.
- 이는 *n* 번째 달에 살아있던 토끼는 충분한 나이가 되어 새끼를 낳을 수 있지만, 바로 전달인 *n+1* 번째에 막 태어난 토끼는 아직 새끼를 낳을 수 없기 때문이다.

문제 제시 : 토끼 수 구하기

- **토끼 수 구하기**
  - f(n)을 n번째 달에 토끼 수라고 하면
  - f(n+2) = f(n) + f(n+1) 이 성립하고
  - 이는 유명한 레오나르도 피보나치가 연구한 피보나치 수열이다.

피보나치 수열

- 0(0항)과 1(1항)로 시작하고 이전의 두 수 합을 다음 항으로 하는 수열을 피보나치 수열 이라 한다
  - 0, 1, 1, 2, 3, 5, 8, 13, ...
- 피보나치 수열의 i번 째 값을 계산하는 함수 F를 정의 하면 다음과 같다.
  - F0 = 0, F1 = 1
  - $F_i = F_{i-1} + F_{i-2}$ for i ≥2
- 위의 정의로부터 피보나치 수열의 i번째 항을 반환하는 함수를 재귀함수로 구현할 수 있다.

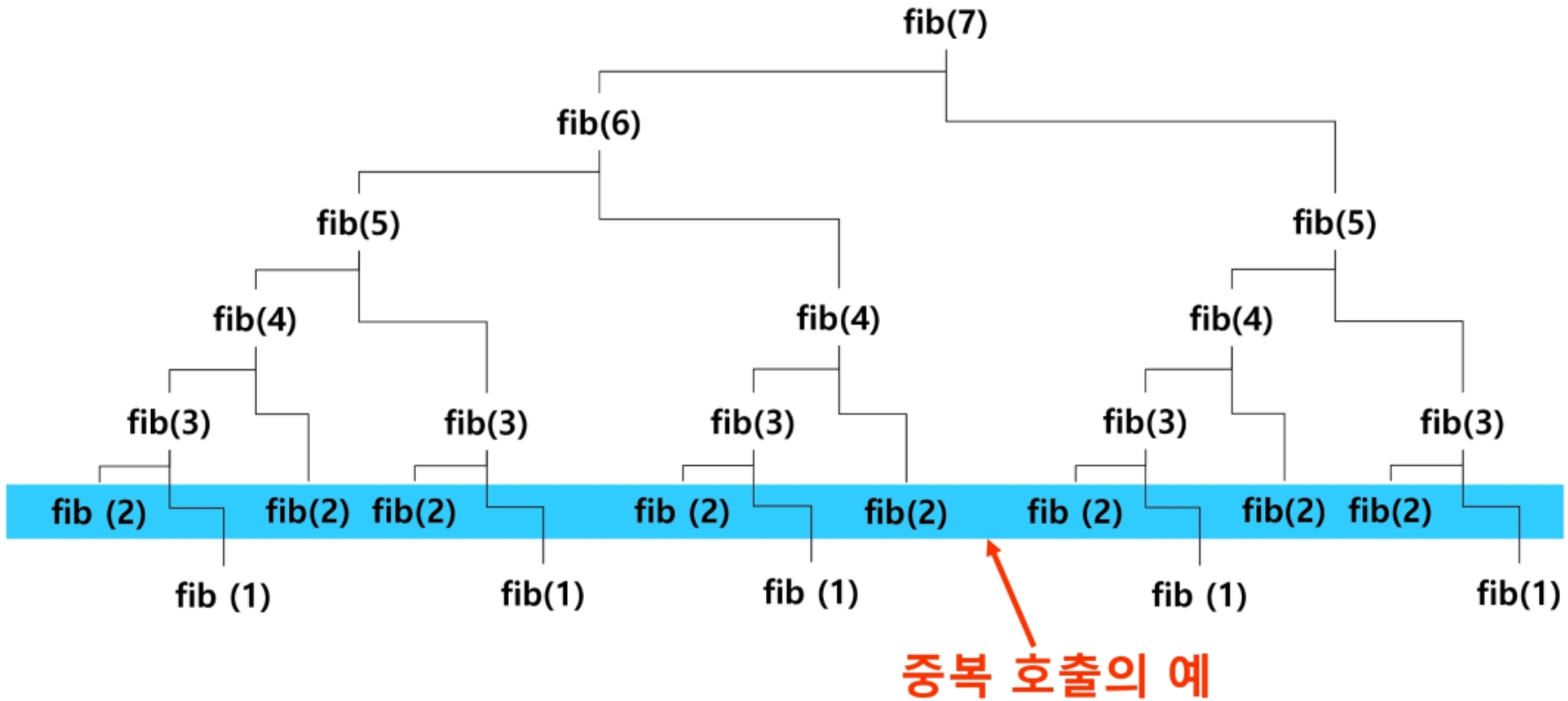
피보나치 수열

- 피보나치 수를 구하는 재귀함수

```
fibo(n)
      IF n < 2 : RETURN n
      ELSE     : RETURN fibo(n - 1) + fibo(n - 2)
```

피보나치 수열

- 앞의 예에서 피보나치 수를 구하는 함수를 재귀함수로 구현한 알고리즘은 문제점이 있다.
- "엄청난 중복 호출이 존재한다"는 것이다.
- 피보나치 수열의 Call Tree



피보나치 수열

- **피보나치 수열을 재귀 구현 했을 때의 질문**
  - 얼마나 중복되었나?
  - 중복을 피할 수 있는 방법은 무엇인가?

메모이제이션

- **메모이제이션(memoization)은 컴퓨터 프로그램을 실행할 때 이전에 계산한 값을 메모리에 저장해서 매번 다시 계산하지 않도록 하여 전체적인 실행속도를 빠르게 하는 기술이다. 동적 계획법의 핵심이 되는 기술이다.**

- **'memoization'은 글자 그대로 해석하면 '메모리에 넣기(to put in memory)' 라는 의미이며 '기억되어야 할 것'이라는 뜻의 라틴어 memorandum에서 파생되었다.**
  - 흔히 '기억하기', '암기하기'라는 뜻의 memorization과 혼동하지만, 정확한 단어는 memoization 이다.
  - 동사형은 memoize이다.

메모이제이션

- 피보나치 수를 구하는 알고리즘에서 fibo1(n)의 값을 계산하자마자 저장하면 (memoize), 실행시간을 O(n)으로 줄일 수 있다.

- Memoization 방법을 적용한 알고리즘은 다음과 같다.

```
memo를 위한 배열을 할당하고, 모두 0으로 초기화 한다
memo[0]을 0으로 memo[1]는 1로 초기화 한다


fibo1(n)
        IF n >= 2 AND memo[n] = 0
                memo[n] ← fibo1(n-1) + fibo1(n-2)
        RETURN memo[n]
```

메모이제이션

- **메모이제이션**
  - 추가적인 메모리 공간이 필요하다.
  - 재귀 함수 호출로 인한 시스템 호출 스택을 사용하게 되고 실행 속도 저하 또는 오버플로우가 발생할 수 있다.

- **해결책은?**

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

동적 계획법

동적 계획법

- 동적 계획법(Dynamic Programming)은 그리디 알고리즘과 같이 **최적화 문제**를 해결하는 알고리즘이다.

- 동적 계획법은 먼저 작은 부분 문제들의 해들을 구하고 이들을 이용하여 보다 큰 크기의 부분 문제들을 해결하여, 최종적으로 원래 주어진 문제를 해결하는 알고리즘 설계 기법이다.

동적 계획법의 적용 요건

- **동적 계획법을 적용하려는 문제는 필히 다음과 같은 요건을 가지고 있어야 한다.**
  - 중복 부분문제 구조(Overlapping subproblems)
  - 최적 부분문제 구조(Optimal substructure)

동적 계획법의 적용 요건

중복 부분문제 구조(Overlapping subproblems)

- DP는 큰 문제를 이루는 작은 문제들을 먼저 해결하고 작은 문제들의 최적 해(Optimal Solution)를 이용하여 순환적으로 큰 문제를 해결한다.
  - 순환적인 관계(recurrence relation)를 명시적으로 표현하기 위해서 동적 계획법에서는 일반적으로 수학적 도구인 점화식을 사용한다.
- DP는 문제의 순환적인 성질 때문에 이전에 계산되어졌던 작은 문제의 해가 다른 어딘가에서 필요하게 되는데(Overlapping subproblems) 이를 위해 DP에서는 이미 해결된 작은 문제들의 해들을 어떤 저장 공간(table)에 저장하게 된다.
- 그리고 이렇게 저장된 해들이 다시 필요할 때 마다 해를 얻기 위해 다시 문제를 재계산하지 않고 table의 참조를 통해서 중복된 계산을 피하게 된다.

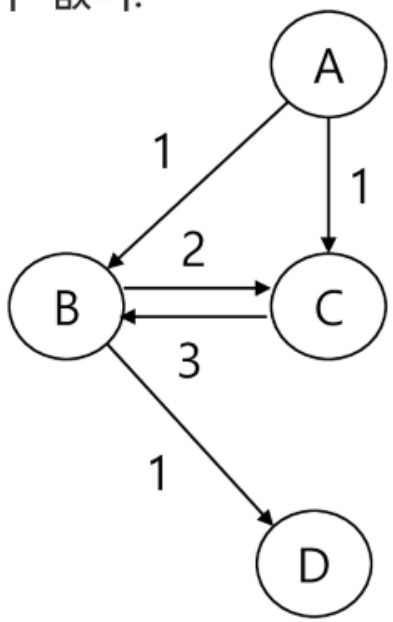
동적 계획법의 적용 요건

최적 부분문제 구조(Optimal substructure)

- 동적 계획법이 최적화에 대한 어느 문제에나 적용될 수 있는 것은 아니다. 주어진 문제가 최적화의 원칙(Principle of Optimality)을 만족해야만 동적 계획법을 효율적으로 적용할 수 있다.
- 최적화의 원칙이란 어떤 문제에 대한 해가 최적일 때 그 해를 구성하는 작은 문제들의 해 역시 최적이어야 한다는 것이다. 동적 계획법의 방법자체가 큰 문제의 최적 해를 작은 문제의 최적해 들을 이용하여 구하기 때문에 만약 큰 문제의 최적해가 작은 문제들의 최적해들로 구성되지 않는다면 이 문제는 동적 계획법을 적용할 수 없다.

동적 계획법의 적용 요건

- **최적의 원칙이 적용되지 않는 예 : 최장경로(Longest Path) 문제**
  - A에서 D로의 최장 경로는 [A, C, B, D]가 된다.
  - 그러나, 이 경로의 부분 경로인 A에서 C 로의 최장 경로는 [A, C]가 아니라 [A, B, C]이다.
  - 최적의 원칙이 적용되지 않는다.
  - 따라서 최장경로문제는 DP로 해결할 수 없다.



분할 정복과 동적계획법의 비교

분할 정복

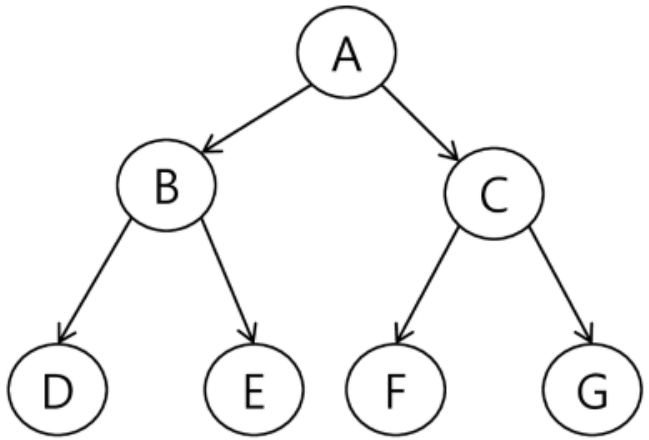
- 연관 없는 부분 문제로 분할 한다.
- 부분문제를 재귀적으로 해결한다.
- 부분문제의 해를 결합(combine)한다.
- 예 : 병합 정렬, 퀵 정렬

DP

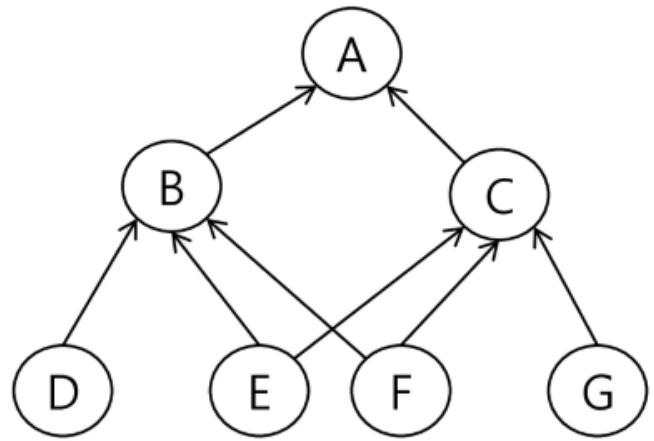
- 부분 문제들이 연관이 없으면 적용할 수 없다. 즉 부분 문제들은 더 작은 부분 문제들을 공유한다.
- 모든 부분 문제를 한번만 계산하고 결과를 저장하고 재사용한다.

분할 정복과 동적계획법의 비교

- **DP에는 부분 문제들 사이에 의존적 관계가 존재한다.**
  - 예를 들면, E, F, G의 해가 C를 해결하는데 사용되어지는 관계가 있다.
- **이러한 관계는 문제에 따라 다르고, 대부분의 경우 뚜렷이 보이지 않아서 함축적인 순서(implicit order)라고 한다.**
- **분할 정복은 하향식 방법으로 DP는 상향식 방법으로 접근한다.**



분할정복



DP

3단계 DP 적용 접근 방법

- **최적해 구조의 특성을 파악하라**
  - 문제를 부분 문제로 나눈다.
- **최적해의 값을 재귀적으로 정의하라**
  - 부분 문제의 최적해 값에 기반하여 문제의 최적해 값을 정의한다.
- **상향식 방법으로 최적해의 값을 계산하라**
  - 가장 작은 부분 문제부터 해를 구한 뒤 테이블에 저장한다.
  - 테이블에 저장되어 있는 부분 문제의 해를 이용하여 점차적으로 상위 부분 문제의 최적해를 구한다. (상향식 방법)

3단계 DP 적용 접근 방법

피보나치 수 DP 적용

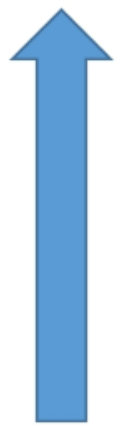
- 피보나치 수는 부분 문제의 답으로부터 본 문제의 답을 얻을 수 있으므로 최적 부분 구조로 이루어져 있다

① 문제를 부분 문제로 분할한다.

② 점화식으로 정의한다.

③ 가장 작은 부분 문제부터 해를 구한다. 그 결과는 테이블에 저장하고, 테이블에 저장된 부분 문제의 해를 이용하여 상위 문제의 해를 구한다.

테이블 인덱스	저장되어 있는 값
[n]	fibo(n)
...	...
[4]	3
[3]	2
[2]	1
[1]	1
[0]	0



3단계 DP 적용 접근 방법

- 피보나치 수 DP 적용 알고리즘

```
fibo_dp(n)
        f[0] ← 0
        f[1] ← 1
        FOR i in 2 → n
                f[i] ← f[i - 1] + f[i - 2]
        RETURN f[n]
```

3단계 DP 적용 접근 방법

피보나치 수 구하기 – DP 알고리즘 분석

- DP 알고리즘이 수행속도가 더 빠르다.
- 이유
  - 재귀 알고리즘과는 달리 중복 계산이 없다.
  - 또한 반복문을 사용하기 때문에 함수 호출이 발생하지 않는다.

계산하는 항(fibo_dp[n])의 총 개수

- n + 1 개수의 항 계산
- 즉, fibo_dp[0] 부터 fibo_dp[n] 까지 단 한번씩만 계산한다.

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

Appendix

문제 제시 : 타일 채우기

- 아래 그림과 같은 2가지 종류의 타일이 있다.

- 위의 타일들을 여러 개 사용해서 가로 2칸, 세로 2칸 크기의 판을 채우는 방법은 다음 3가지이다.

- 이 때, 타일들을 회전해서 채워 넣는 것도 가능하다.
- 위의 2가지 타일들을 여러 개 사용해서 가로 2칸, 세로 N칸 크기의 판을 채우는 방법은 모두 몇 가지가 되는지 구하는 프로그램을 작성하시오. ($1 \le N \le 1,000$)
- 가지 수를 10007로 나눈 나머지를 출력한다.

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

다음 방송에서 만나요!

삼성 청년 SW 아카데미