

3주차 과제 풀이

- Dynamic Programming -

안태진(taejin7824@gmail.com)

GitHub(<https://github.com/Taejin1221>)

소프트웨어학과 18학번

상명대학교 CodeCure 소프트웨어부장

Contents

- 피보나치 수
- $2 \times n$ 타일링
- $2 \times n$ 타일링 2
- 동전 2
- 가장 긴 증가하는 부분 수열
- 1로 만들기
- 계단 오르기

피보나치 수

- 재귀적 구조
 - $fib(n) = fib(n - 1) + fib(n - 2)$
- Solution
 - 1차원 DP table을 만든 뒤에 저장된 fibonacci 값을 사용!
 - $dp[n] = dp[n - 1] + dp[n - 2]$

2xn 타일링

- Solution

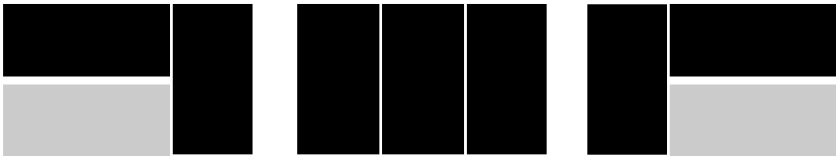
- $n = 1$



- $n = 2$




- $n = 3$



- n 번째 타일은

- $n-1$ 길이인 타일을 만들고  를 붙이거나

- $n-2$ 길이인 타일을 만들고  를 붙이면 됨



- 따라서
$$\text{Tiling}(n) = \begin{cases} 1 & (n = 1) \\ 2 & (n = 2) \\ \text{Tiling}(n - 1) + \text{Tiling}(n - 2) & (n \geq 3) \end{cases}$$

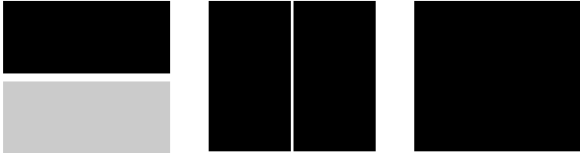
2xn 타일링 2

• Solution

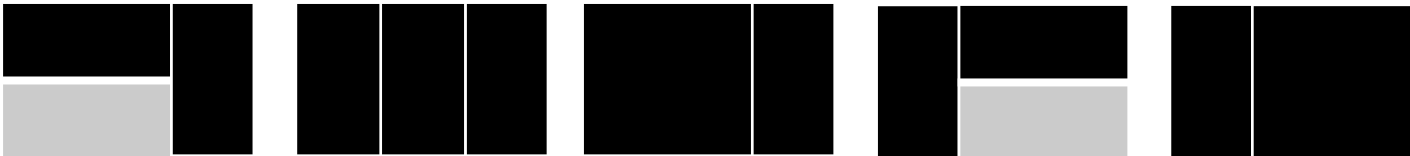
• $n = 1$



• $n = 2$




• $n = 3$



• n 번째 타일은

• $n-1$ 의 길이인 타일을 만들고  를 붙이거나 • $n-2$ 의 길이인 타일을 만들고  을 붙이면 됨

• $n-2$ 의 길이인 타일을 만들고  를 붙이거나

• 따라서
$$\text{Tiling}(n) = \begin{cases} 1 & (n = 1) \\ 3 & (n = 2) \\ \text{Tiling}(n-1) + \text{Tiling}(n-2) + \text{Tiling}(n-2) & (n \geq 3) \end{cases}$$

동전 2

- Solution

- 15원을 최소 개수의 동전으로 만들기 위해 다음 중 가장 작은 것을 고르면 됨
 - 1원 + (14원을 만드는 동전의 최소 개수)
 - 5원 + (10원을 만드는 동전의 최소 개수)
 - 12원 + (3원을 만드는 동전의 최소 개수)
- $getMinCoin(n) = \min([getMinCoin(n - coin) \text{ for } coin \text{ in } coin_list]) + 1$
 - $getMinCoin(n)$ 은 모든 동전($coin$)들에 대해 $getMinCoin(n - coin)$ 중 가장 작은 값 + 1이다.
 - $coin_list$: 동전 가치들의 list(array)
 - $coin$: 한 동전의 가치(원)

가장 긴 증가하는 부분 수열

- Solution
 - arr을 수열이라 하고,
LIS(i)를 arr[i]를 시작으로하는 가장 긴 증가하는 부분 수열이라 할 때
 - 답은 자기 보다 큰 값을 가지는 배열의 값을 시작으로 하는 LIS중 가장 큰 것 + 1
 - LIS(i):
 - result = 1
 - for idx <- i부터 끝까지:
 - if (arr[idx] > arr[i])
 - result = max(result, LCS(idx) + 1)
 - return result

1로 만들기

- Solution
 - 예제
 - 10을 1로 만드는 방법 =>
 - 모든 방법을 다 해봐야하지만, 중복 발생
 - 따라서 $dp[i]$ 를 i 를 1로 만드는 최소한의 연산 수로 두어 중복 제거 후 해결

$$10 \left\{ \begin{array}{l} 5 \left\{ \begin{array}{l} 4 \left\{ \begin{array}{l} 2 \left\{ \begin{array}{l} 1 \\ 1 \end{array} \right\} \\ 3 \left\{ \begin{array}{l} 1 \\ 2 \left\{ \begin{array}{l} 1 \\ 1 \end{array} \right\} \end{array} \right\} \end{array} \right\} \\ 9 \left\{ \begin{array}{l} 3 \left\{ \begin{array}{l} 1 \\ 2 \left\{ \begin{array}{l} 1 \\ 1 \end{array} \right\} \end{array} \right\} \\ 8 \end{array} \right\} \end{array} \right.$$

계단 오르기

- Solution (1/2)
 - $Stair(n)$ 을 n 번째 계단을 오를 때 얻을 수 있는 최대한의 점수라 한다면
 - $Stair(n) = \max(Stair(n-1), Stair(n-2)) + score[n]$
 - $score[i]$: i 번째 계단의 점수
 - 하지만 위의 방식대로 한다면 모든 계단을 들르는 경우가 최대
하지만 조건은 3칸을 연속으로 오르면 안됨

계단 오르기

- Solution (2/2)
 - 따라서 $Stair(n, 1)$ 을 n 번째 계단을 $n-1$ 번째 계단에서 한 계단을 오를 때 얻는 점수, $Stair(n, 2)$ 를 n 번째 계단을 $n-2$ 번째 계단에서 두 계단을 오를 때 얻는 점수라 하면
 - $Stair(n, 1) = Stair(n - 1, 2) + score[n]$
 - $n-1$ 번째 계단을 한 계단 올랐고, 다시 n 번째 계단을 한 계단으로 오르면서 계단 연속이기 때문에 조건에 부합 X,
 - 따라서 $n-2$ 번째 계단을 두 계단 올랐을 경우만 더해줌
 - $Stair(n, 2) = \max(Stair(n - 2, 1), Stair(n - 2, 2)) + score[n]$

감사합니다!