심화 알고리즘

01 동적계획법
02 백트래킹
03 최단 경로 알고리즘
04 최소 신장 트리

신 제 용

동적계획법 (Dynamic Programming)

01 동적계획법 이론

02 동적계획법 예시 문제 풀이

신 제 용

ZEROBASE SCHOOL

01 동적 계획법 이론

동적 계획법! DP! 지금 학습합니다.

학습 키워드 – 동적계획법, Dynamic programming, DP, Tabulation, Tabularization, Memoization

Chapter 01
동적 계획법 이론

ZEROBASE SCHOOL

동적 계획법 (Dynamic programming; DP)

- 큰 문제를 부분 문제로 나눈 후 답을 찾아가는 과정에서,
  계산된 결과를 기록하고 재활용하며 문제의 답을 구하는 방식

- **중간 계산 결과를 기록**하기 위한 메모리가 필요

- 한 번 계산한 부분을 다시 계산하지 않아 속도가 빠름

vs. 분할 정복/탐욕 알고리즘

- 분할 정복과의 차이
  - 분할 정복은 부분 문제가 중복되지 않음
  - DP는 **부분 문제가 중복되어 재활용**에 사용

- 그리디 알고리즘과의 차이
  - 그리디 알고리즘은 순간의 최선을 구하는 방식 → 특정 조건에서만 최적해
  - DP는 **모든 방법을 확인 후 최적해** 구하는 방식 → 항상 최적해 보장

동적 계획법 분류

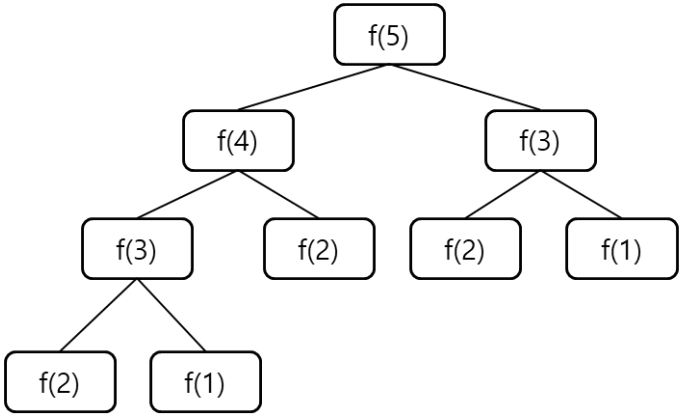
- 타뷸레이션 (Tabulation, tabularization)
  - Bottom-Up 접근
  - 작은 하위 문제부터 풀면서 올라감
  - 모두 계산하면서 차례대로 진행

- 메모이제이션 (Memoization)
  - Top-Down 접근
  - 큰 문제에서 하위 문제를 확인해가며 진행
  - 계산이 필요한 순간 계산하며 진행

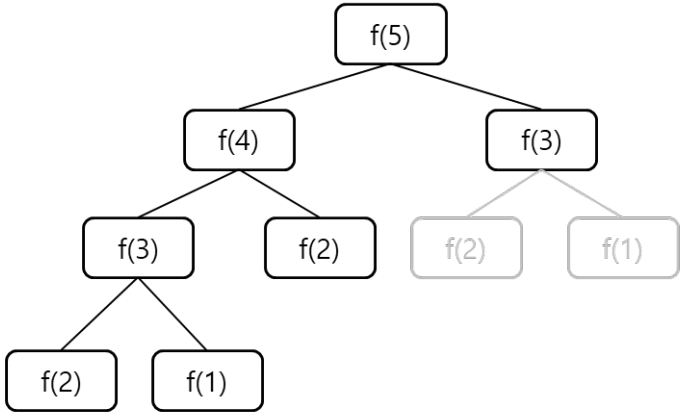
동적 계획법 예시

- 피보나치 수열의 계산

피보나치 기본 재귀 풀이



피보나치 DP 적용 풀이



f(1)	f(2)	f(3)	f(4)	f(5)	...
1	1	2	3	5	...

동적 계획법 예시

- 피보나치 수열의 계산 (Top-Down)

```
fibo = dict()
fibo[0] = 0
fibo[1] = 1

def fibonacci(n):
    if n in fibo:
        return fibo[n]

    res = fibonacci(n-1) + fibonacci(n-2)
    fibo[n] = res
    return res
```

동적 계획법 예시

- 피보나치 수열의 계산 (Bottom-Up)

```python
def fibonacci(n):
    fibo = [None] * (n + 1)
    fibo[0], fibo[1] = 0, 1

    for i in range(2, n + 1):
        fibo[i] = fibo[i - 1] + fibo[i - 2]

    return fibo[n]
```

02 동적 계획법 예시 문제 풀이

다음 챕터에서는 동적 계획법 예시 문제를 학습합니다.

Chapter 01
동적 계획법 이론

ZEROBASE SCHOOL

02 동적 계획법 예시 문제 풀이

초보 개발자의 주적 DP! 예시 문제로 확실하게 이해해 봅시다.

학습 키워드 – 동적 계획법, DP, 구현

Chapter 02

동적 계획법 예시 문제 풀이

ZEROBASE SCHOOL

Problem1

문제 설명

0-1 Knapsack 문제를 해결하세요!

당신은 `capacity` 만큼의 무게를 담을 수 있는 배낭에 물건을 담고자 한다.

각 물건의 무게는 `weight[i]`, 각 물건의 가치는 `value[i]`로 주어져 있을 때,

배낭에 담을 수 있는 가능한 물건들의 가치의 합 중 최대의 값을 구하시오.

매개변수 예시

```
capacity = 20
```

```
weight = [4, 5, 2, 3, 6, 8, 5, 5]
```

```
value = [5, 2, 6, 7, 1, 3, 4, 6]
```

출력 예시

```
28
```

Chapter 02

동적 계획법 예시 문제 풀이

ZEROBASE SCHOOL

Problem2

문제 설명

누리는 개발 공부를 하다 보니 삽으로 땅을 파는 데에 전문가가 되었다. 누리가 파내는 땅은 블록 형태로 구분되어있고, 각 블록은 제거하는 데에 서로 다른 에너지가 소비된다. 땅에 있는 첫 번째 블록은 깊이가 `0`부터 시작하며, 한칸씩 내려갈 때 마다 깊이가 `1`씩 증가한다. 누리가 블록을 제거할 수 있는 조건은 아래와 같다.

- 깊이 `0`에 위치한 블록은 자유롭게 제거할 수 있다.
- 깊이 `d`에 위치한 `i`번째 블록을 제거하려면, 깊이 `d-1`에 위치한 `i-1`, `i`, `i+1`번째 블록 중 하나가 제거되어 있어야 한다.

누리는 깊이 `depth`의 `n`번째 블록에 위치한 화석을 발굴하려고 한다. 각 깊이별 블록을 제거하는 데에 필요한 에너지는 `blocks`에 저장되어 있다. 화석이 위치한 블록을 제거하는 데에 필요한 최소의 에너지를 구하시오. (단, `n`은 `0`부터 시작하며, 모든 깊이에는 동일한 숫자의 블록이 있다.)

매개변수 형식

`depth = 3`

`n = 3`

`blocks = [[5, 6, 2, 6], [1, 6, 4, 9], [5, 6, 9, 4], [55, 14, 21, 14]]`

반환값 형식

`24`

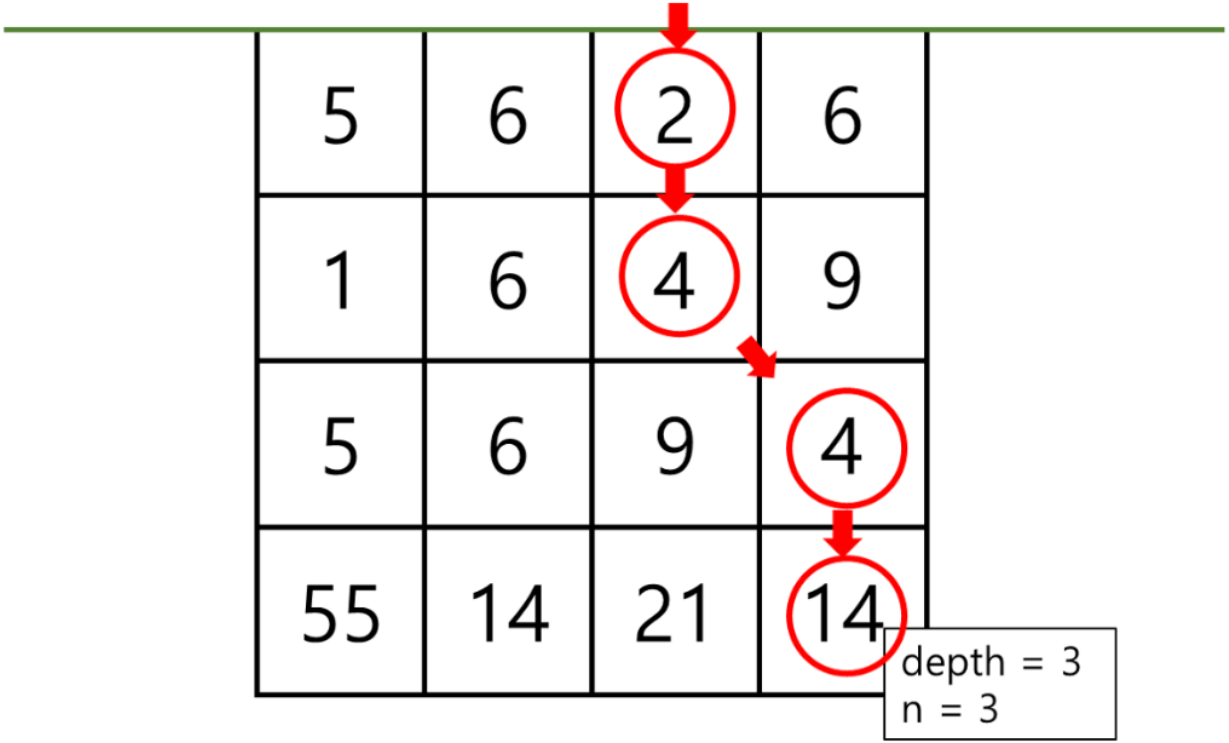
Chapter 02

동적 계획법 예시 문제 풀이

ZEROBASE SCHOOL

예시입출력 설명

아래와 같이 블록을 제거할 경우 최소의 에너지로 화석을 얻을 수 있다.



Chapter 02

동적 계획법 예시 문제 풀이

ZEROBASE SCHOOL

Problem3

문제 설명

당신은 제로국으로 급파된 암살자로, 제로국의 주요 인사들을 암살하는 임무를 맡았다.

제로국에는 총 `N`개의 성이 있고 이 성들은 일정한 가격으로 원형으로 배치되어 있다.

당신은 조사원들을 통해서 다음과 같은 사실을 알아내었다.

- 각 성마다 주요 인사는 한 명씩 배치되어 있다.
- 각 성의 주요 인사를 암살했을 때의 보상은 `rewards[i]`로 주어진다.
- 하나의 성의 주요 인사를 암살할 경우, 인접한 성은 경계태세가 되어 침입할 수 없다.

위 조건에서 달성할 수 있는 최대의 보상을 구하시오.

단, 원형 배치의 특성상 첫 번째 성은 마지막 성과 인접해 있다.

매개변수 형식

`N = 6`

`rewards = [5, 10, 5, 7, 5, 9]`

반환값 형식

`26`

예시입출력 설명

인접한 성 중에는 하나만 침입이 가능하므로, 모든 주요 인사를 암살할 수 없다.

`1`, `3`, `5` 번째 성에 침입할 경우 최대의 보상을 얻을 수 있다.

Problem4

문제 설명

정수로 이루어진 배열 `nums`가 주어지고, `0`번 인덱스에서 시작한다고 하자.

최대 현재 인덱스의 숫자 `nums[i]`만큼 우측으로 이동이 가능하다고 할 때, 최종적으로 마지막 위치까지 도달할 수 있는지 여부를 논리형 값으로 출력하시오.

매개변수 형식

```
nums = [3, 4, 1, 1, 0, 3]
```

반환값 형식

```
True
```

입출력 예시 설명

아래와 같이 이동하면 마지막 위치인 `5` 인덱스까지 도달할 수 있다.

`0` -> `1` -> `5`

Chapter 02

동적 계획법 예시 문제 풀이

ZEROBASE SCHOOL