Chapter. 08

동적 프로그래밍

핵심 유형 문제풀이

FAST CAMPUS
ONLINE
유형별 문제풀이

강사. 나동빈

Chapter. 08

동적 프로그래밍(핵심 유형 문제풀이)

FAST CAMPUS
ONLINE
나동빈 강사.

Copyright FASTCAMPUS Corp. All Rights Reserved

I 혼자 힘으로 풀어 보기

LCS: https://www.acmicpc.net/problem/9251

문제 제목: LCS

문제 난이도: 하(Easy)

문제 유형: 동적 프로그래밍, LCS

추천 풀이 시간: 30분

Life Changing Education
Fast campus
Copyright FASTCAMPUS Corp. All Rights Reserved

LCS: https://www.acmicpc.net/problem/9251

I 문제 풀이 핵심 아이디어

- 두 수열이 주어졌을 때, 두 수열 모두의 부분 수열이 되는 수열 중 가장 긴 것을 찾아야 합니다.
- 가장 긴 공통 부분 수열(LCS) 문제로 알려진 대표적인 동적 프로그래밍 문제입니다.
- 두 수열의 길이가 N 미만일 때, 시간 복잡도 $O(N^2)$으로 문제를 해결할 수 있습니다.

Life Changing Education

Copyright FASTCAMPUS Corp. All Rights Reserved

LCS: https://www.acmicpc.net/problem/9251

I 문제 풀이 핵심 아이디어

- 두 수열을 각각 X, Y라고 합시다.
- $D[i][j] = X[0 \dots i]$와 $Y[0 \dots j]$의 공통 부분 수열의 최대 길이
- 두 문자열의 길이를 조금씩 늘려 가며 확인하여, 공통 부분 수열의 최대 길이를 계산합니다.
- $$D[i][j] = \begin{cases} D[i-1][j-1]+1 & if \quad X[i] = Y[j] \\ \max(D[i][j-1], D[i-1][j]) & if \quad X[i] \neq Y[j] \end{cases}$$

Life Changing Education
Fast campus
Copyright FASTCAMPUS Corp. All Rights Reserved

LCS: https://www.acmicpc.net/problem/9251

I 문제 풀이 핵심 아이디어

- X = “ACAYKP”이고, Y = “CAPCAK”일 때

[초기화]

	ø	C	A	P	C	A	K
ø	0	0	0	0	0	0	0
A	0						
C	0						
A	0						
Y	0						
K	0						
P	0						

Life Changing Education
Fast campus
Copyright FASTCAMPUS Corp. All Rights Reserved

LCS: https://www.acmicpc.net/problem/9251

I 문제 풀이 핵심 아이디어

- X = “ACAYKP”이고, Y = “CAPCAK”일 때

[i = 1]

	ø	C	A	P	C	A	K
ø	0	0	0	0	0	0	0
A	0	0	1	1	1	1	1
C	0						
A	0						
Y	0						
K	0						
P	0						

Life Changing Education
Fast campus
Copyright FASTCAMPUS Corp. All Rights Reserved

LCS: https://www.acmicpc.net/problem/9251

I 문제 풀이 핵심 아이디어

- X = “ACAYKP”이고, Y = “CAPCAK”일 때

[i = 2]

AC와 CAPC의 최장 공통 부분 수열의 길이

	ø	C	A	P	C	A	K
ø	0	0	0	0	0	0	0
A	0	0	1	1	1	1	1
C	0	1	1	1	2	2	2
A	0						
Y	0						
K	0						
P	0						

Life Changing Education
Fast campus
Copyright FASTCAMPUS Corp. All Rights Reserved

LCS: https://www.acmicpc.net/problem/9251

I 문제 풀이 핵심 아이디어

- X = “ACAYKP”이고, Y = “CAPCAK”일 때

[i = 3]

	ø	C	A	P	C	A	K
ø	0	0	0	0	0	0	0
A	0	0	1	1	1	1	1
C	0	1	1	1	2	2	2
A	0	1	2	2	2	3	3
Y	0						
K	0						
P	0						

Life Changing Education
Fast campus
Copyright FASTCAMPUS Corp. All Rights Reserved

LCS: https://www.acmicpc.net/problem/9251

I 문제 풀이 핵심 아이디어

- X = “ACAYKP”이고, Y = “CAPCAK”일 때

[i = 4]

	ø	C	A	P	C	A	K
ø	0	0	0	0	0	0	0
A	0	0	1	1	1	1	1
C	0	1	1	1	2	2	2
A	0	1	2	2	2	3	3
Y	0	1	2	2	2	3	3
K	0						
P	0						

Life Changing Education
Fast campus
Copyright FASTCAMPUS Corp. All Rights Reserved

LCS: https://www.acmicpc.net/problem/9251

I 문제 풀이 핵심 아이디어

- X = “ACAYKP”이고, Y = “CAPCAK”일 때

[i = 5]

	ø	C	A	P	C	A	K
ø	0	0	0	0	0	0	0
A	0	0	1	1	1	1	1
C	0	1	1	1	2	2	2
A	0	1	2	2	2	3	3
Y	0	1	2	2	2	3	3
K	0	1	2	2	2	3	4
P	0						

Life Changing Education
Fast campus
Copyright FASTCAMPUS Corp. All Rights Reserved

LCS: https://www.acmicpc.net/problem/9251

I 문제 풀이 핵심 아이디어

- X = “ACAYKP”이고, Y = “CAPCAK”일 때

[i = 6]

ACAYKP와 CAPCAK의 최장 공통 부분 수열의 길이

	ø	C	A	P	C	A	K
ø	0	0	0	0	0	0	0
A	0	0	1	1	1	1	1
C	0	1	1	1	2	2	2
A	0	1	2	2	2	3	3
Y	0	1	2	2	2	3	3
K	0	1	2	2	2	3	4
P	0	1	2	3	3	3	4

Copyright FASTCAMPUS Corp. All Rights Reserved

LCS: https://www.acmicpc.net/problem/9251

소스코드

```
x = input()
y = input()

dp = [[0] * (len(y) + 1) for _ in range(len(x) + 1)]

for i in range(1, len(x) + 1):
    for j in range(1, len(y) + 1):
        if x[i - 1] == y[j - 1]:
            dp[i][j] = dp[i - 1][j - 1] + 1
        else:
            dp[i][j] = max(dp[i][j - 1], dp[i - 1][j])

print(dp[len(x)][len(y)])
```


Life Changing Education
Fast campus
Copyright FASTCAMPUS Corp. All Rights Reserved

I 혼자 힘으로 풀어 보기

기타리스트: https://www.acmicpc.net/problem/1495

문제 제목: 기타리스트

문제 난이도: 중(Medium)

문제 유형: 동적 프로그래밍

추천 풀이 시간: 40분

Copyright FASTCAMPUS Corp. All Rights Reserved

기타리스트: https://www.acmicpc.net/problem/1495

I 문제 풀이 핵심 아이디어

- 차례대로 곡을 연주한다는 점에서, 동적 프로그래밍으로 해결할 수 있는 문제입니다.
- 곡의 개수가 N, 볼륨의 최댓값은 M입니다.
- 동적 프로그래밍을 이용하여 시간 복잡도 O(NM)으로 문제를 해결할 수 있습니다.

Life Changing Education
Fast campus
Copyright FASTCAMPUS Corp. All Rights Reserved

기타리스트: https://www.acmicpc.net/problem/1495

I 문제 풀이 핵심 아이디어

- **핵심 아이디어**: 모든 볼륨에 대하여 연주 가능 여부를 계산하기
- $\mathrm{D[i][j+1]} = i$번째 노래일 때 j 크기의 볼륨으로 연주 가능한지 여부
- 노래를 순서대로 확인하며, 매 번 모든 크기의 볼륨에 대하여 검사합니다.
- $D[i][j - V[i]] = True \ \ if \ D[i \ - 1][j] = True$
- $D[i][j + V[i]] = True \ \ if \ D[i \ - 1][j] = True$

Life Changing Education
Fast campus
Copyright FASTCAMPUS Corp. All Rights Reserved

기타리스트: https://www.acmicpc.net/problem/1495

I 문제 풀이 핵심 아이디어

- N = 3, S = 5, M = 10일 때의 예시를 확인해 봅시다.

곡 번호	볼륨
1	5
2	3
3	7

0	1	2	3	4	5	6	7	8	9	10
F	F	F	F	F	**T**	F	F	F	F	F

Life Changing Education
Fast campus
Copyright FASTCAMPUS Corp. All Rights Reserved

기타리스트: https://www.acmicpc.net/problem/1495

I 문제 풀이 핵심 아이디어

- N = 3, S = 5, M = 10일 때의 예시를 확인해 봅시다.

곡 번호	볼륨
1	5
2	3
3	7

0	1	2	3	4	5	6	7	8	9	10
F	F	F	F	F	**T**	F	F	F	F	F
T	F	F	F	F	F	F	F	F	F	**T**

Life Changing Education
Fast campus
Copyright FASTCAMPUS Corp. All Rights Reserved

기타리스트: https://www.acmicpc.net/problem/1495

I 문제 풀이 핵심 아이디어

- N = 3, S = 5, M = 10일 때의 예시를 확인해 봅시다.

곡 번호	볼륨
1	5
2	3
3	7

0	1	2	3	4	5	6	7	8	9	10
F	F	F	F	F	**T**	F	F	F	F	F
T	F	F	F	F	F	F	F	F	F	**T**
F	F	F	**T**	F	F	F	**T**	F	F	F

Life Changing Education
Fast campus
Copyright FASTCAMPUS Corp. All Rights Reserved

기타리스트: https://www.acmicpc.net/problem/1495

I 문제 풀이 핵심 아이디어

- N = 3, S = 5, M = 10일 때의 예시를 확인해 봅시다.

곡 번호	볼륨
1	5
2	3
3	7

0	1	2	3	4	5	6	7	8	9	10
F	F	F	F	F	**T**	F	F	F	F	F
T	F	F	F	F	F	F	F	F	F	**T**
F	F	F	**T**	F	F	F	**T**	F	F	F
T	F	F	F	F	F	F	F	F	F	**T**

Life Changing Education
Fast campus
Copyright FASTCAMPUS Corp. All Rights Reserved

기타리스트: https://www.acmicpc.net/problem/1495

소스코드

```python
n, s, m = map(int, input().split())
array = list(map(int, input().split()))

dp = [[0] * (m + 1) for _ in range(n + 1)]
dp[0][s] = 1

for i in range(1, n + 1):
    for j in range(m + 1):
        if dp[i - 1][j] == 0:
            continue
        if j - array[i - 1] >= 0:
            dp[i][j - array[i - 1]] = 1
        if j + array[i - 1] <= m:
            dp[i][j + array[i - 1]] = 1

result = -1
for i in range(m, -1, -1):
    if dp[n][i] == 1:
        result = i
        break

print(result)
```


Life Changing Education
Fast campus
Copyright FASTCAMPUS Corp. All Rights Reserved

I 혼자 힘으로 풀어 보기

가장 높은 탑 쌓기: https://www.acmicpc.net/problem/2655

문제 제목: 가장 높은 탑 쌓기

문제 난이도: 상(Hard)

문제 유형: 동적 프로그래밍, LIS

추천 풀이 시간: 50분

Life Changing Education
Fast campus
Copyright FASTCAMPUS Corp. All Rights Reserved

가장 높은 탑 쌓기: https://www.acmicpc.net/problem/2655

I 문제 풀이 핵심 아이디어

- 가장 긴 증가하는 부분 수열(LIS) 문제의 심화 변형 문제입니다.
- 벽돌의 수가 N개일 때, 시간 복잡도 $O(N^2)$으로 해결할 수 있습니다.
- 벽돌의 번호를 출력해야 한다는 점에서, 계산된 테이블을 역추적할 수 있어야 합니다.

Life Changing Education
Fast campus
Copyright FASTCAMPUS Corp. All Rights Reserved

가장 높은 탑 쌓기: https://www.acmicpc.net/problem/2655

I 문제 풀이 핵심 아이디어

- 가장 먼저 벽돌을 무게 기준으로 정렬합니다.
- D[i] = 인덱스가 i인 벽돌을 가장 아래에 두었을 때의 최대 높이
- 각 벽돌에 대해서 확인하며 D[i]를 갱신합니다.
- 모든 $0 \leq j < i$ 에 대하여, $D[i] = \max(D[i], D[j] + height[i])\ if\ area[i] > area[j]$

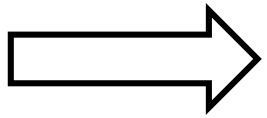
Life Changing Education
Fast campus
Copyright FASTCAMPUS Corp. All Rights Reserved

가장 높은 탑 쌓기: https://www.acmicpc.net/problem/2655

문제 풀이 핵심 아이디어

- $N = 5$일 때, 예시 벽돌들에 대하여 다음과 같이 계산할 수 있습니다.

번호	너비	높이	무게
0	0	0	0
1	25	3	4
2	4	4	6
3	9	2	3
4	16	2	5
5	1	5	2



무게 기준 정렬

번호	너비	높이	무게
0	0	0	0
5	1	5	2
3	9	2	3
1	25	3	4
4	16	2	5
2	4	4	6

Life Changing Education
Fast campus
Copyright FASTCAMPUS Corp. All Rights Reserved

가장 높은 탑 쌓기: https://www.acmicpc.net/problem/2655

문제 풀이 핵심 아이디어

- $N = 5$일 때, 예시 벽돌들에 대하여 다음과 같이 계산할 수 있습니다.

번호	너비	높이	무게
0	0	0	0
5	1	5	2
3	9	2	3
1	25	3	4
4	16	2	5
2	4	4	6

0	1	2	3	4	5
0	0	0	0	0	0

Life Changing Education
Fast campus
Copyright FASTCAMPUS Corp. All Rights Reserved

가장 높은 탑 쌓기: https://www.acmicpc.net/problem/2655

문제 풀이 핵심 아이디어

- $N = 5$일 때, 예시 벽돌들에 대하여 다음과 같이 계산할 수 있습니다.

번호	너비	높이	무게
0	0	0	0
5	1	5	2
3	9	2	3
1	25	3	4
4	16	2	5
2	4	4	6

0	1	2	3	4	5
0	0	0	0	0	0
0	5	0	0	0	0

Life Changing Education
Fast campus
Copyright FASTCAMPUS Corp. All Rights Reserved

가장 높은 탑 쌓기: https://www.acmicpc.net/problem/2655

문제 풀이 핵심 아이디어

- $N = 5$일 때, 예시 벽돌들에 대하여 다음과 같이 계산할 수 있습니다.

번호	너비	높이	무게
0	0	0	0
5	1	5	2
3	9	2	3
1	25	3	4
4	16	2	5
2	4	4	6

0	1	2	3	4	5
0	0	0	0	0	0
0	5	0	0	0	0
0	5	7	0	0	0

Life Changing Education
Fast campus
Copyright FASTCAMPUS Corp. All Rights Reserved

가장 높은 탑 쌓기: https://www.acmicpc.net/problem/2655

I 문제 풀이 핵심 아이디어

- $N = 5$일 때, 예시 벽돌들에 대하여 다음과 같이 계산할 수 있습니다.

번호	너비	높이	무게
0	0	0	0
5	1	5	2
3	9	2	3
1	25	3	4
4	16	2	5
2	4	4	6

0	1	2	3	4	5
0	0	0	0	0	0
0	5	0	0	0	0
0	5	7	0	0	0
0	5	7	10	0	0

Life Changing Education
Fast campus
Copyright FASTCAMPUS Corp. All Rights Reserved

가장 높은 탑 쌓기: https://www.acmicpc.net/problem/2655

문제 풀이 핵심 아이디어

- $N = 5$일 때, 예시 벽돌들에 대하여 다음과 같이 계산할 수 있습니다.

번호	너비	높이	무게
0	0	0	0
5	1	5	2
3	9	2	3
1	25	3	4
4	16	2	5
2	4	4	6

0	1	2	3	4	5
0	0	0	0	0	0
0	5	0	0	0	0
0	5	7	0	0	0
0	5	7	10	0	0
0	5	7	10	9	0

Life Changing Education
Fast campus
Copyright FASTCAMPUS Corp. All Rights Reserved

가장 높은 탑 쌓기: https://www.acmicpc.net/problem/2655

I 문제 풀이 핵심 아이디어

- $N = 5$일 때, 예시 벽돌들에 대하여 다음과 같이 계산할 수 있습니다.

번호	너비	높이	무게
0	0	0	0
5	1	5	2
3	9	2	3
1	25	3	4
4	16	2	5
2	4	4	6

0	1	2	3	4	5
0	0	0	0	0	0
0	5	0	0	0	0
0	5	7	0	0	0
0	5	7	10	0	0
0	5	7	10	9	0
0	5	7	10	9	9

Life Changing Education
Fast campus
Copyright FASTCAMPUS Corp. All Rights Reserved

가장 높은 탑 쌓기: https://www.acmicpc.net/problem/2655

I 문제 풀이 핵심 아이디어

- $N = 5$일 때, 예시 벽돌들에 대하여 다음과 같이 계산할 수 있습니다.

번호	너비	높이	무게
0	0	0	0
5	1	5	2
3	9	2	3
1	25	3	4
4	16	2	5
2	4	4	6

0	1	2	3	4	5
0	0	0	0	0	0
0	5	0	0	0	0
0	5	7	0	0	0
0	5	7	10	0	0
0	5	7	10	9	0
0	5	7	10	9	9

⇨ 이후에 $Max\ Value$ 위치부터 테이블을 역추적합니다. Max Value = 10

Life Changing Education
Fast campus
Copyright FASTCAMPUS Corp. All Rights Reserved

가장 높은 탑 쌓기: https://www.acmicpc.net/problem/2655

소스코드

```python
n = int(input())
array = []

array.append((0, 0, 0, 0))
for i in range(1, n + 1):
    area, height, weight = map(int, input().split())
    array.append((i, area, height, weight))

# 무게를 기준으로 정렬합니다.
array.sort(key=lambda data: data[3])

dp = [0] * (n + 1)

for i in range(1, n + 1):
    for j in range(0, i):
        if array[i][1] > array[j][1]:
            dp[i] = max(dp[i], dp[j] + array[i][2])
```

```python
max_value = max(dp)
index = n
result = []

while index != 0:
    if max_value == dp[index]:
        result.append(array[index][0])
        max_value -= array[index][2]
    index -= 1

result.reverse()
print(len(result))
[print(i) for i in result]
```


Life Changing Education
Fast campus
Copyright FASTCAMPUS Corp. All Rights Reserved