알고리즘 특강 동적계획법

메모이제이션을 활용한 문제 풀이 기법입니다.
굉장히 생소한 개념이며, 초보자분들은 많이 어려워 하지만, 차근차근 하면 됩니다!

동적계획법

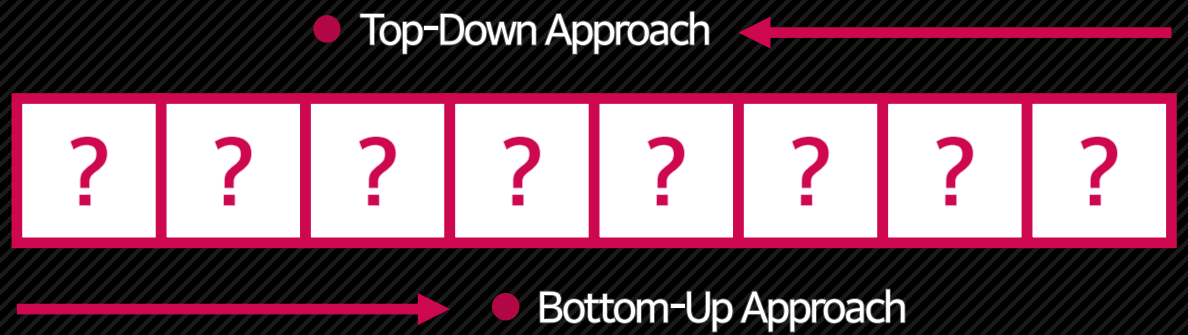
Dynamic Programming

- 특정 범위까지의 값을 구하기 위해 **이전 범위의 값을 활용**하여 효율적으로 값을 얻는 기법.
- 이전 범위의 값을 저장하여, **(Memoization)** 똑같은 문제가 발생했을 때 이를 참조하여 해결함.

언제 동적계획법을 쓸까?

- 큰 문제를 작은 문제로 쪼갰을 때, 작은 문제의 답을 통해 큰 문제의 답을 도출할 수 있는가?
- 큰 문제를 해결하기 위해 작은 문제의 답을 여러 번 구해야 하는가?

탑다운/보텀업



- 이론상 시간복잡도는 두 기법 모두 동일하지만, Top-Down은 재귀 오버헤드 때문에 실제 시간은 더 김.
- 그러나 충분히 무시할 수 있는 시간차이기 때문에, 두 방법 모두 연습하는 것이 좋음!

체계적으로 접근해보자.

- 우리가 구하려는 값이 뭘까?
- 특정 상황에서 선택 가능한 경우는 무엇이 있을까?
- 위에서 정리한 것을 기반으로, 현재 값은 어떻게 도출할 수 있을까?

냅색 문제

Gold 5 - 평범한 배낭 (#12865)

요약

- N개의 물건을 갖고 있는데, 각각의 물건엔 고유의 무게 W와 가치 V를 가지고 있다.
- 최대 K 만큼의 무게 까지만 배낭에 넣을 수 있다고 할 때, 얻을 수 있는 최대의 가치를 출력하는 프로그램을 작성하시오.

제약조건

- 물건의 수 N의 범위는 $1 <= N <= 100$ 이다.
- 배낭의 최대 수용 가능 무게 K의 범위는 $1 <= K <= 100,000$ 이다.
- 각각의 물건의 무게의 범위는 $1 <= W_i <= 100,000$ 이다
- 각각의 물건의 가치의 범위는 $1 <= V_i <= 100,000$ 이다

하나씩 접근해보자…

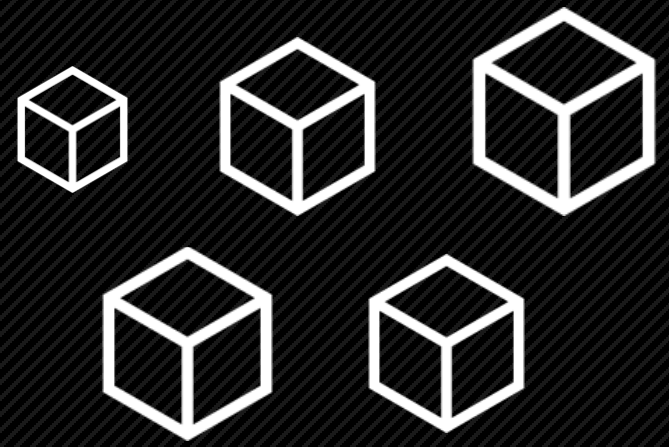
Gold 5 - 평범한 배낭 (#12865)

- 우리가 구하려는 값이 뭘까?
- 특정 상황에서 선택 가능한 경우는 무엇이 있을까?
- 위에서 정리한 것을 기반으로, 현재 값은 어떻게 도출할 수 있을까?

Gold 5 - 평범한 배낭 (#12865)

- 우리가 구하려는 값이 뭘까?

배낭의 남은 무게



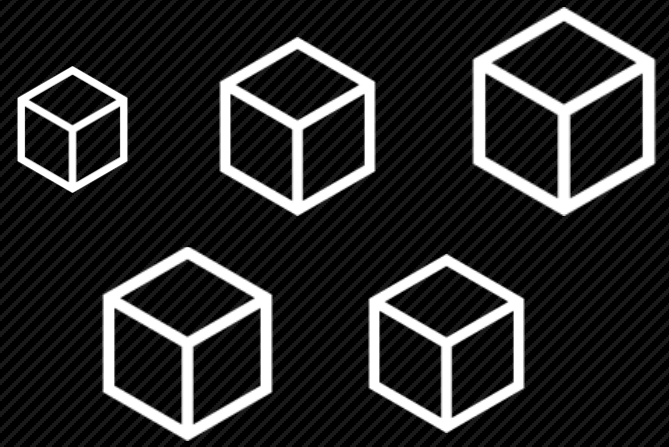
각각의 물건의 무게/가치

시뮬레이션

Gold 5 - 평범한 배낭 (#12865)

- 우리가 구하려는 값이 뭘까?

배낭의 남은 무게



각각의 물건의 무게/가치

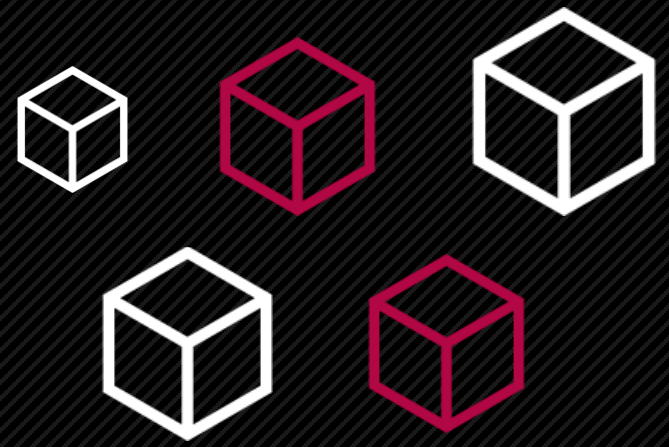
남은 무게를 고려하여 넣을 수 있는 물건들 중 가치의 합이 최대가 되는 경우

문제는…

Gold 5 - 평범한 배낭 (#12865)

- 우리가 구하려는 값이 뭘까?

배낭의 남은 무게



각각의 물건의 무게/가치

두개의 무게가 같다면… 뭘 넣었는지 어떻게 구별할까?

Solution #1

Gold 5 - 평범한 배낭 (#12865)

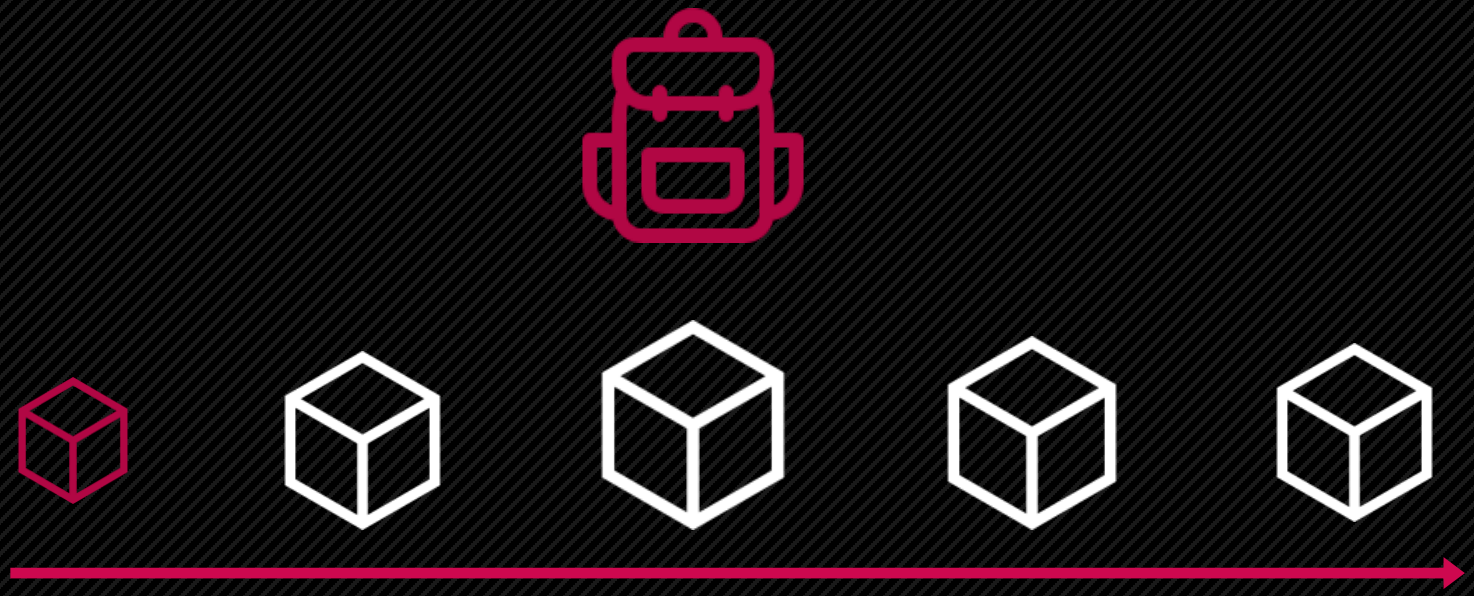
Solution #1

Gold 5 - 평범한 배낭 (#12865)

Solution #1

Gold 5 - 평범한 배낭 (#12865)

- 특정 상황에서 선택 가능한 경우는 무엇이 있을까?

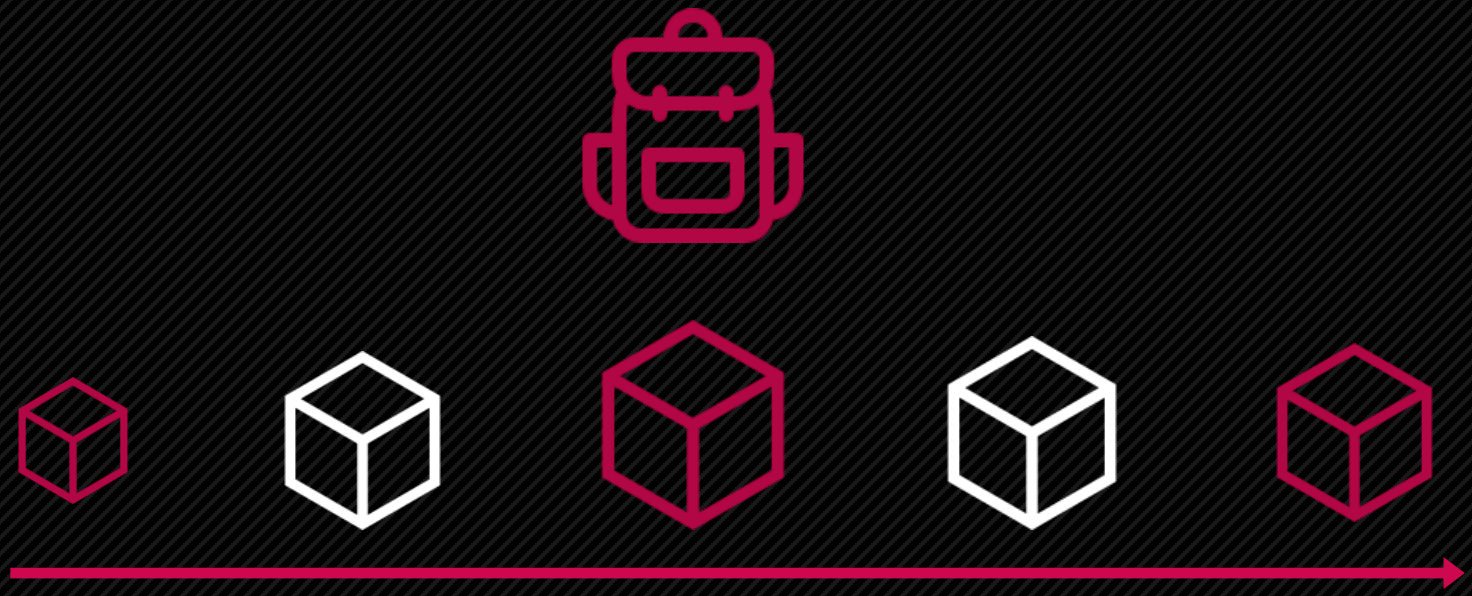


- 가방에 넣는다. (단, 가방의 잔여 공간이 물건보다 커야 함.)
- 넣지 않는다.

Solution #1

Gold 5 - 평범한 배낭 (#12865)

- 위에서 정리한 것을 기반으로, 현재 값은 어떻게 도출할 수 있을까?



- 가방에 넣는다. (단, 가방의 잔여 공간이 물건보다 커야 함.)
- 넣지 않는다.

$$DP[K][0] = MAX(DP[K - W_0][1] + V_0, DP[K][1])$$

플로이드-워셜

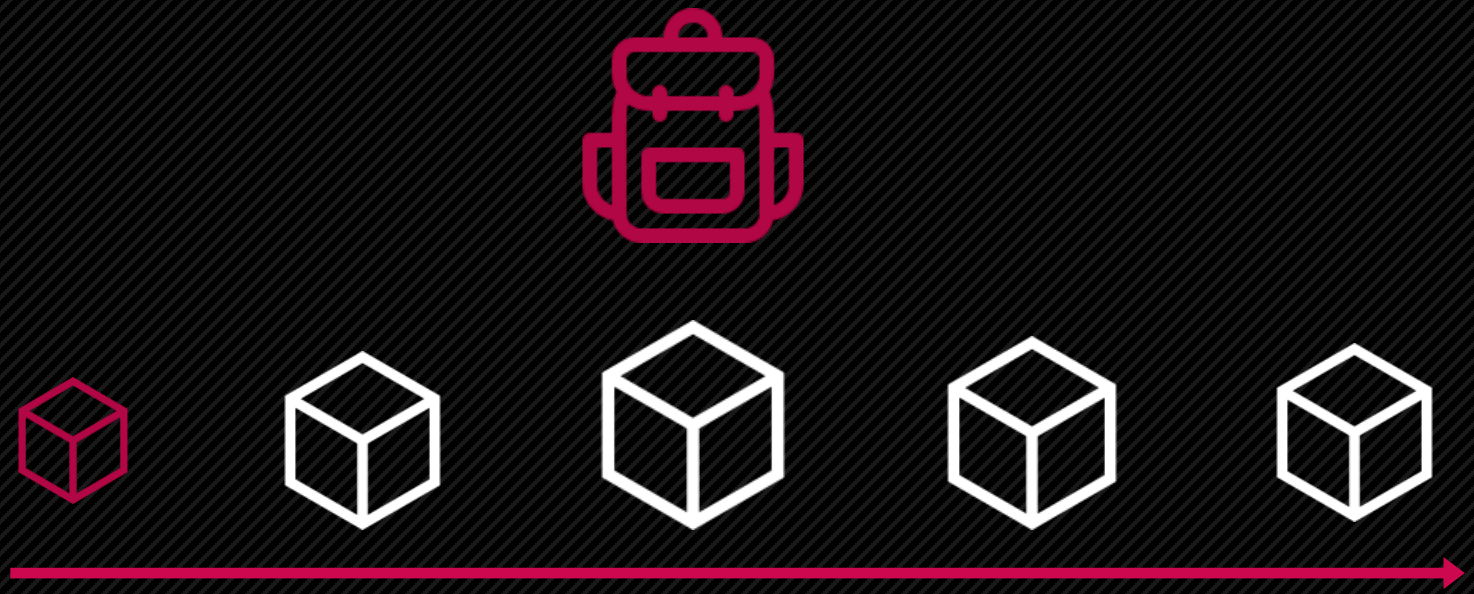
```
for k from 1 to |V|
    for i from 1 to |V|
        for j from 1 to |V|
            if dist[i][j] > dist[i][k] + dist[k][j] then
                set dist[i][j] <- dist[i][k] + dist[k][j]
            end if
```

값을 하나씩 넣어가면서 테이블을 채워 나갔지...

Solution #2

Gold 5 - 평범한 배낭 (#12865)

- 특정 상황에서 선택 가능한 경우는 무엇이 있을까?

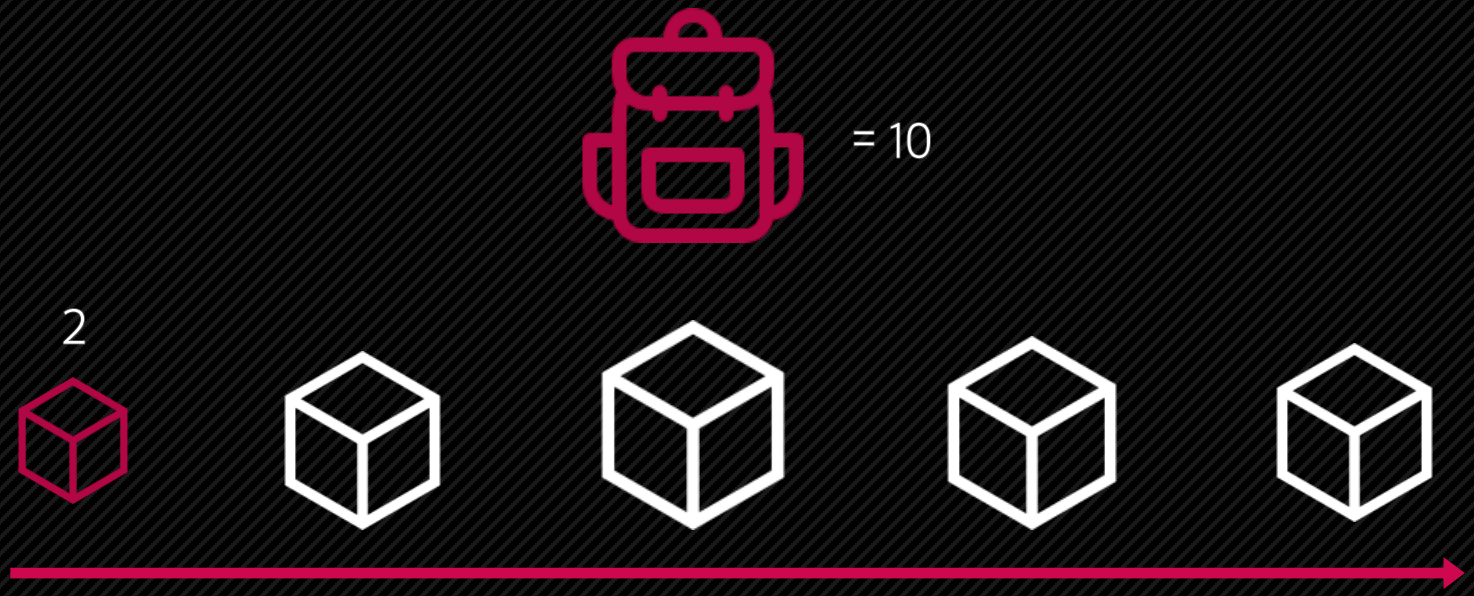


- 가방에 넣는다. (단, 가방의 잔여 공간이 물건보다 커야 함.)
- 넣지 않는다.

Solution #2

Gold 5 - 평범한 배낭 (#12865)

- 특정 상황에서 선택 가능한 경우는 무엇이 있을까?

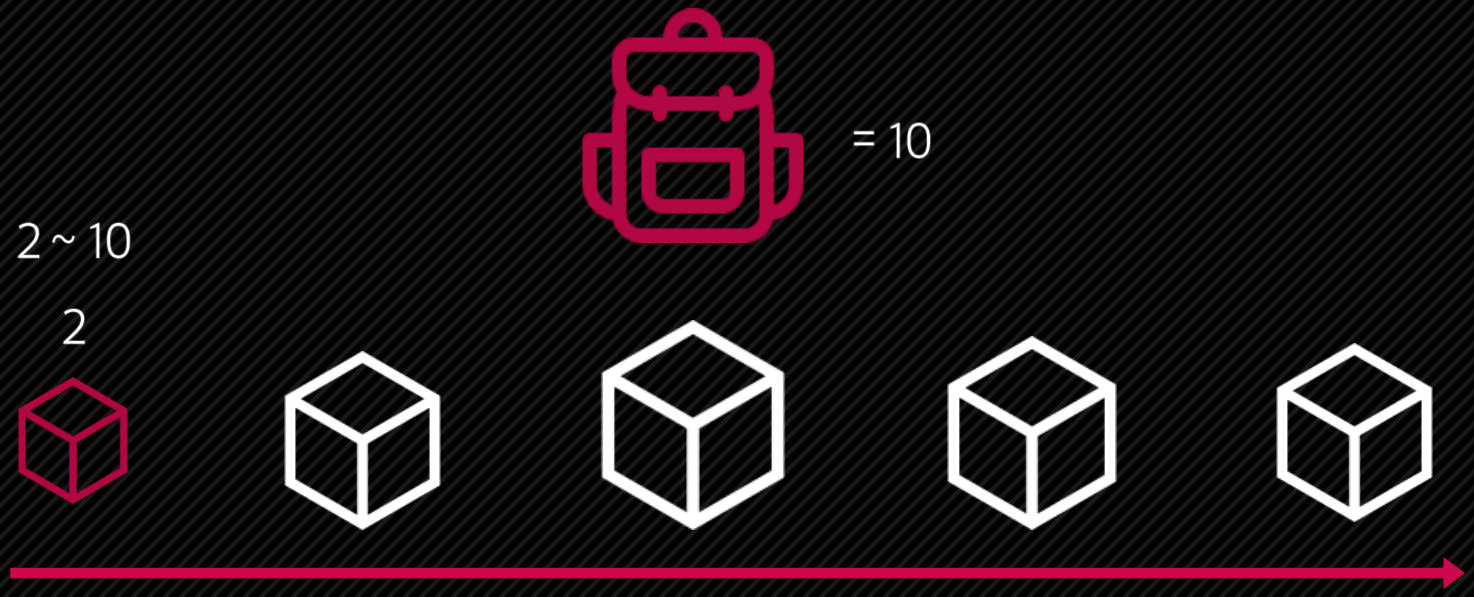


- 가방에 넣는다. (단, 가방의 잔여 공간이 물건보다 커야 함.)
- 넣지 않는다.

Solution #2

Gold 5 - 평범한 배낭 (#12865)

- 특정 상황에서 선택 가능한 경우는 무엇이 있을까?

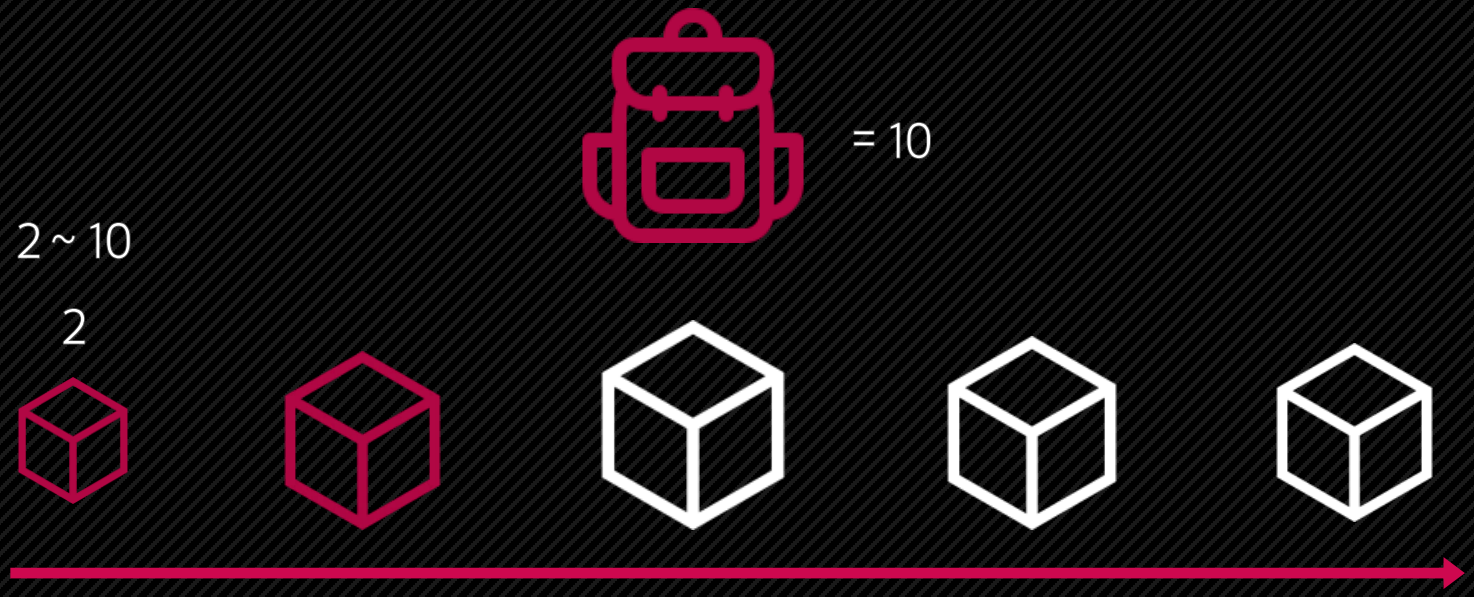


- 가방에 넣는다. (단, 가방의 잔여 공간이 물건보다 커야 함.)
- 넣지 않는다.

Solution #2

Gold 5 - 평범한 배낭 (#12865)

- 특정 상황에서 선택 가능한 경우는 무엇이 있을까?



- 가방에 넣는다. (단, 가방의 잔여 공간이 물건보다 커야 함.)
- 넣지 않는다.

Solution #2

Gold 5 - 평범한 배낭 (#12865)

- 위에서 정리한 것을 기반으로, 현재 값은 어떻게 도출할 수 있을까?

```
function GetDP()
    for (item = 1, 2 ... n)
        for (weight = n ... w[item])
            dp[weight] = max(dp[weight], dp[weight - w[item] + v[item]])
```

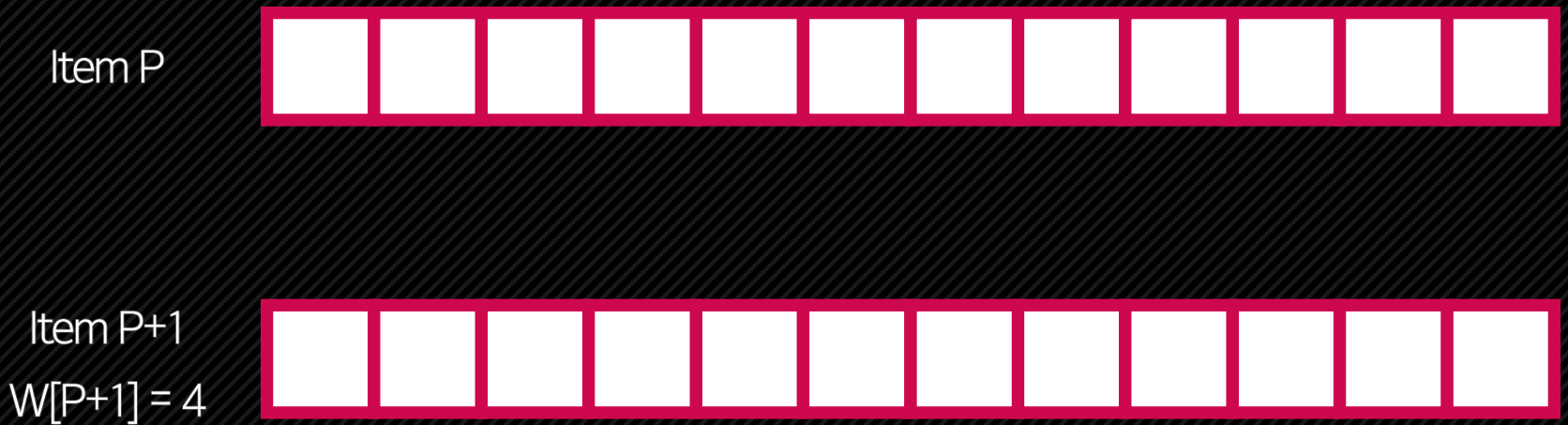
이건 왜 안될까요?

Gold 5 - 평범한 배낭 (#12865)

```
function GetDP()
    for (item = 1, 2 ... n)
        for (weight = w[item] ... n)
            dp[weight] = max(dp[weight], dp[weight - w[item] + v[item]])
```

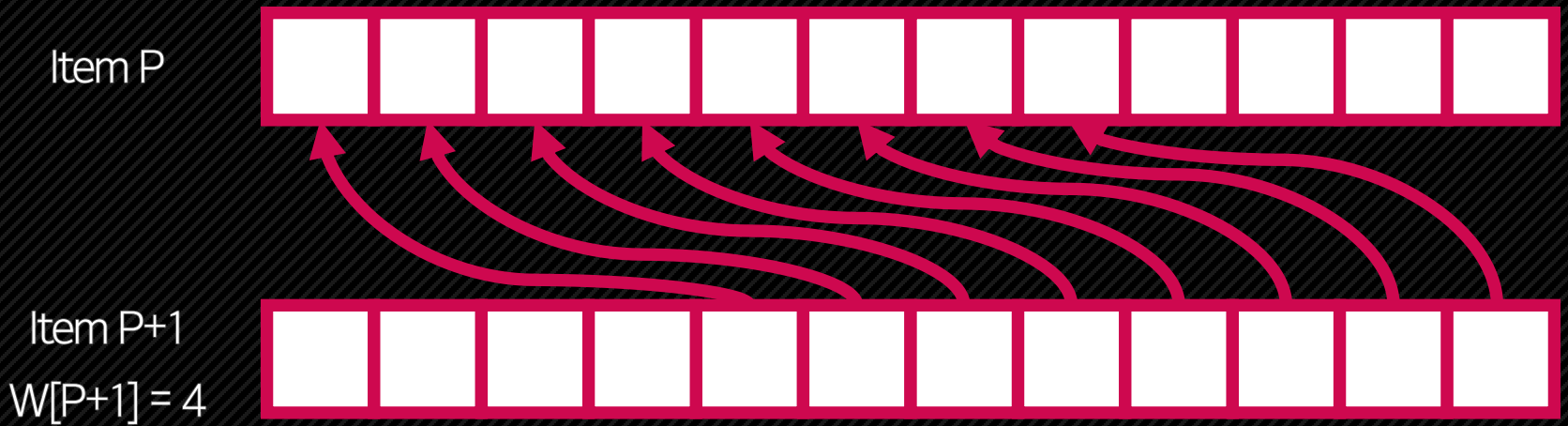
이건 왜 안될까요?

Gold 5 - 평범한 배낭 (#12865)



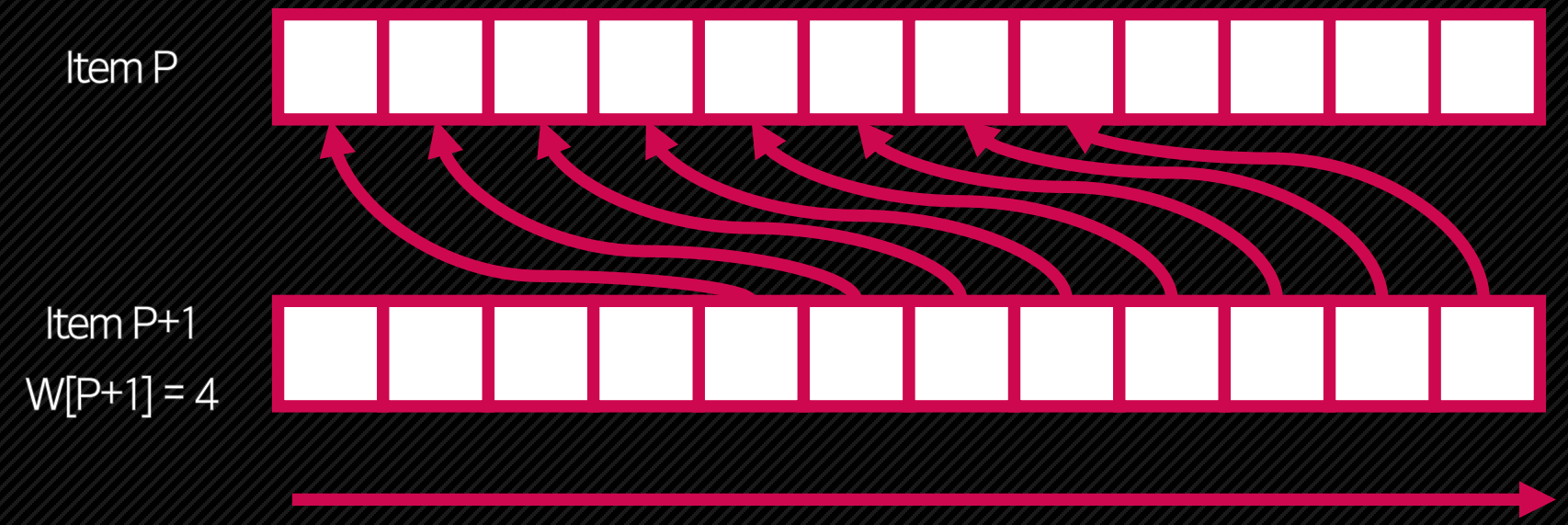
이건 왜 안될까요?

Gold 5 - 평범한 배낭 (#12865)



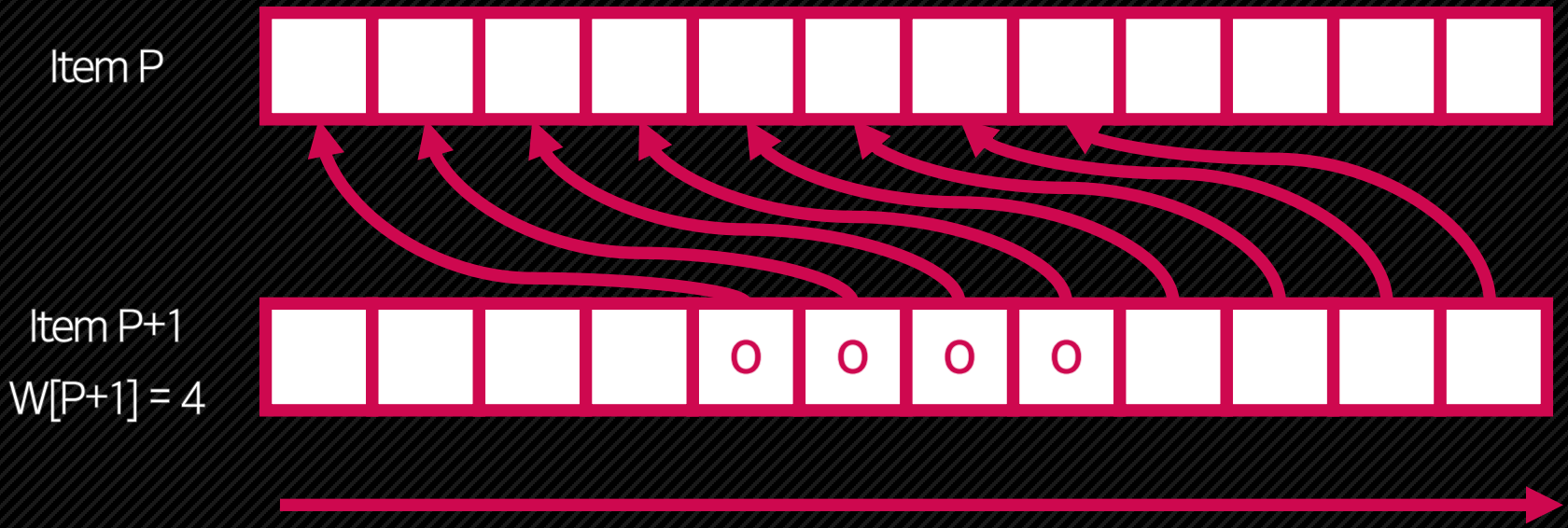
이건 왜 안될까요?

Gold 5 - 평범한 배낭 (#12865)



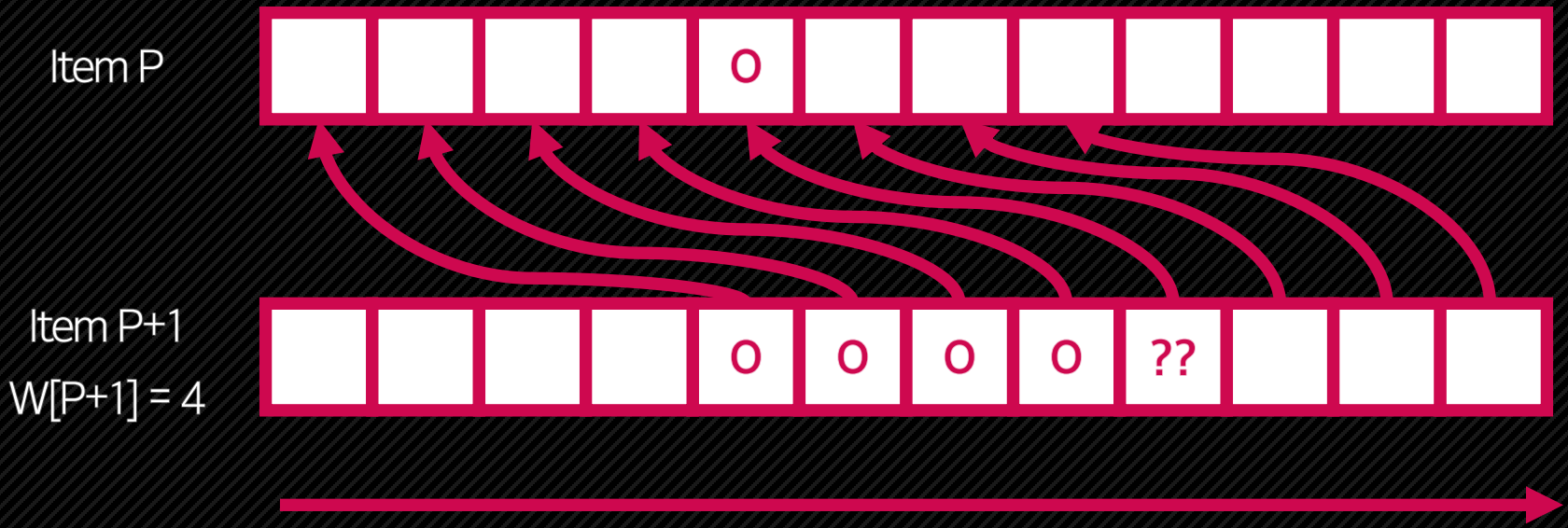
이건 왜 안될까요?

Gold 5 - 평범한 배낭 (#12865)



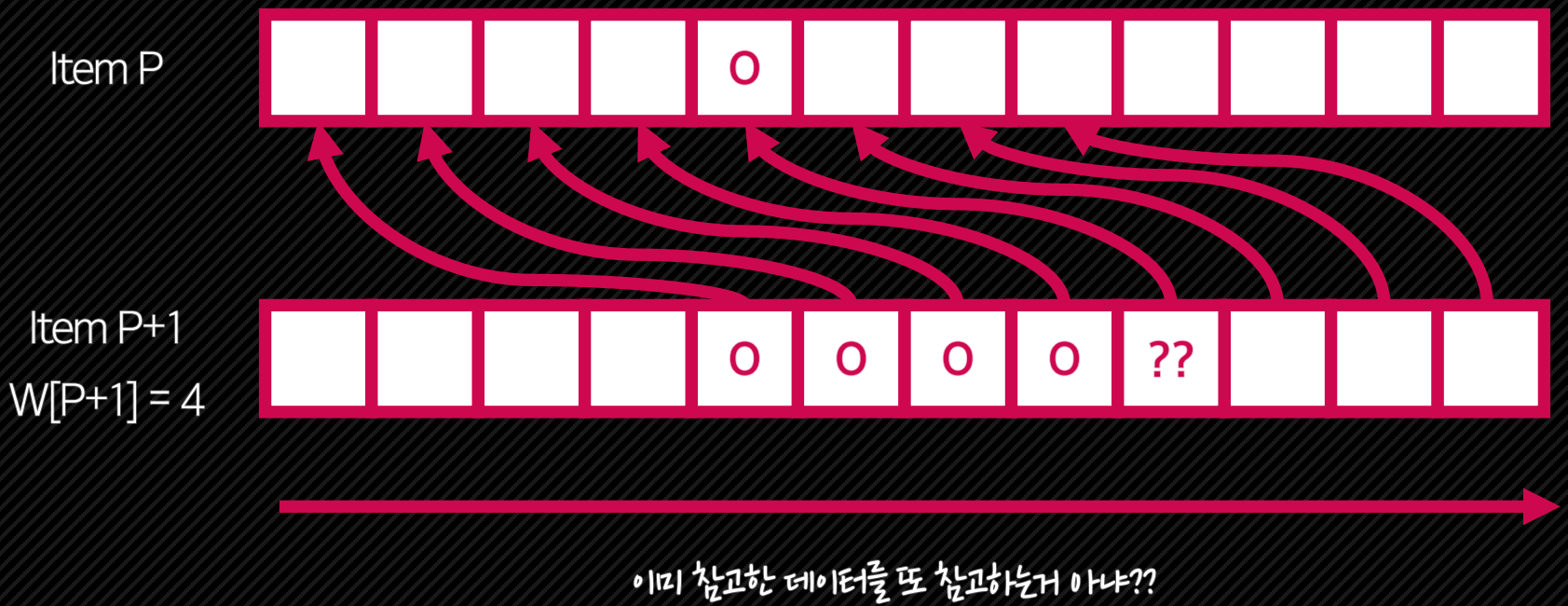
이건 왜 안될까요?

Gold 5 - 평범한 배낭 (#12865)



이건 왜 안될까요?

Gold 5 - 평범한 배낭 (#12865)



되던 안되던 일단 도전.

Gold 3 - 앱 (#7579)

요약

- 각각의 앱을 강제 종료하면 Ci의 시간이 소요되며, Mi의 메모리를 확보할 수 있다.
- 최소한의 시간을 사용하여 특정 용량의 메모리를 확보하려고 할 때, 필요한 최소한의 시간을 구하는 프로그램을 작성하시오.

제약조건

- 앱의 수 N의 범위는 1 <= N <= 100 이다.
- 필요한 메모리 M의 범위는 1 <= M <= 10,000,000 이다.
- Ci의 범위는 1 <= Ci <= 100 이다.
- Mi의 범위는 1 <= Mi <= 10,000,000 이다.

Gold 5 - 이모티콘 (#14226)

요약

- 현재 화면에 이모티콘이 1개 있는데, 다음과 같은 3개의 연산을 해서 S개로 만들려고 한다.
- 화면에 있는 이모티콘 복사/복사한 이모티콘 붙여넣기/하나 삭제
- 각각의 연산은 1초가 소요된다고 할 때, S개를 만들기 위해 필요한 최소한의 시간을 구하는 프로그램을 작성하시오.

제약조건

- S의 범위는 1 <= S <= 1,000 이다.

2차원 DP 추론

Gold 5 - 이모티콘 (#14226)

- 특정 상황에서 선택 가능한 경우는 무엇이 있을까?

- 화면에 있는 이모티콘 복사
- 복사한 이모티콘 붙여넣기
- 이모티콘 하나 삭제

→ 복사/붙여넣기 할 때 클립보드에 있는 길이?

문자열에서 DP?

Gold 5 - LCS (#9251)

요약

- LCS (Longest Common Subsequence)은 두 수열이 주어졌을 때, 모두의 부분 수열 중 가장 긴 것을 의미한다.
- 두 문자열이 주어졌을 때, 이것의 LCS의 길이를 구하는 프로그램을 작성하시오.

제약조건

- 문자열의 길이는 1 <= len <= 1,000 이다.

문자열에서 DP?

Gold 5 - LCS (#9251)

ACAYKP

CAPCAK

문자열에서 DP?

Gold 5 - LCS (#9251)

ACAYKP

CAPCAK

문자열에서 DP?

Gold 5 - LCS (#9251)

- 우리가 구하려는 값이 뭘까?

ACAYKP

CAPCAK

문자열에서 DP?

Gold 5 - LCS (#9251)

- 우리가 구하려는 값이 뭘까?

문자열에서 DP?

Gold 5 - LCS (#9251)

- 우리가 구하려는 값이 뭘까?

짧게 보면…

Gold 5 - LCS (#9251)

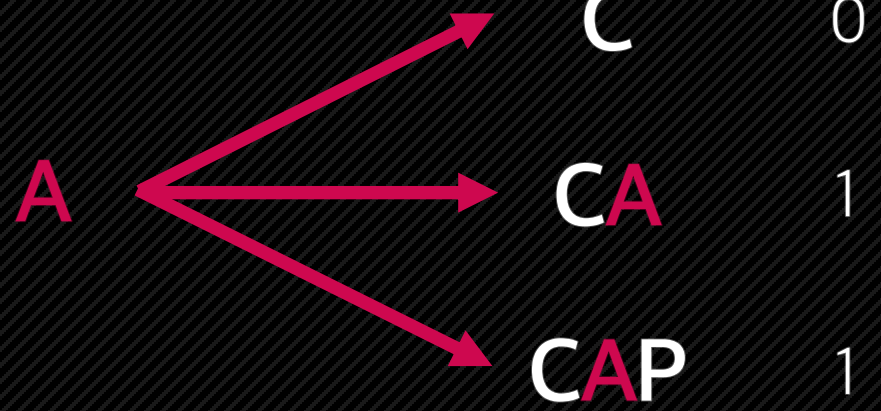
↓

A

CAP

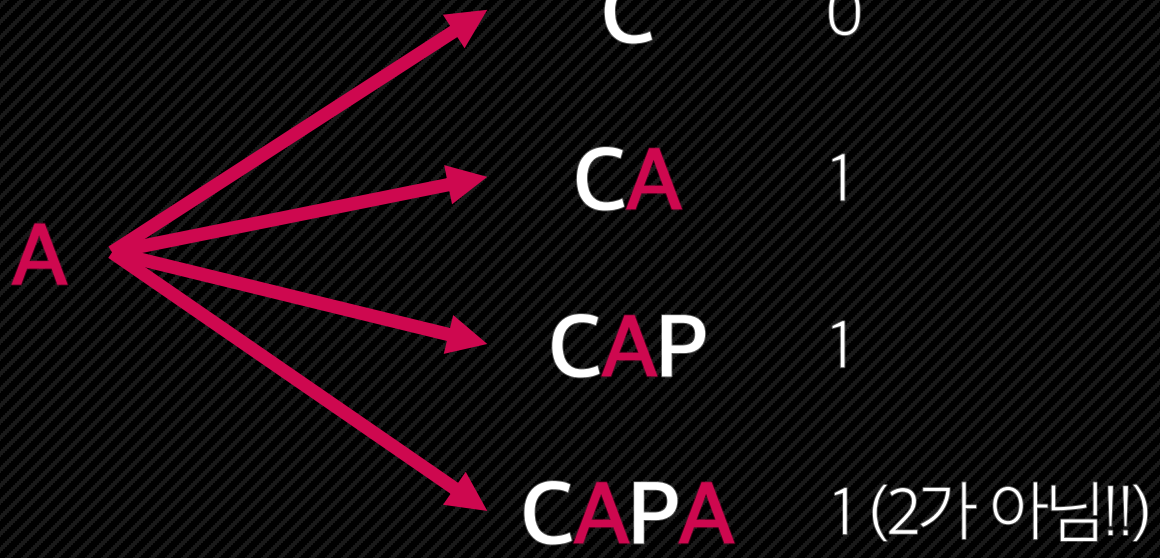
짧게 보면…

Gold 5 - LCS (#9251)



짧게 보면…

Gold 5 - LCS (#9251)



그렇다면…

Gold 5 - LCS (#9251)

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

그렇다면…

Gold 5 - LCS (#9251)

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

그렇다면…

Gold 5 - LCS (#9251)

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

그렇다면…

Gold 5 - LCS (#9251)

		A	C	A	Y	K	P
	C	0	1	1	1	1	1
	A	1	?	0	0	0	0
	P	0	0	0	0	0	0
	C	0	0	0	0	0	0
	A	0	0	0	0	0	0
	K	0	0	0	0	0	0

그렇다면…

Gold 5 - LCS (#9251)

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

그렇다면…

Gold 5 - LCS (#9251)

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

If S1[i] != S2[j]

DP[i][j] = MAX(DP[i - 1][j], DP[i][j - 1])

그렇다면…

Gold 5 - LCS (#9251)

	A	C	A	Y	K	P
C	0	1	1	1	1	1
A	1	1	?	0	0	0
P	0	0	0	0	0	0
C	0	0	0	0	0	0
A	0	0	0	0	0	0
K	0	0	0	0	0	0

If S1[i] != S2[j]

DP[i][j] = MAX(DP[i - 1][j], DP[i][j - 1])

그렇다면…

Gold 5 - LCS (#9251)

		A	C	A	Y	K	P
	0	0	0	0	0	0	0
C	1	0	1	1	1	1	1
A	2	1	1	?	0	0	0
P	3	0	0	0	0	0	0
C	4	0	0	0	0	0	0
A	5	0	0	0	0	0	0
K	6	0	0	0	0	0	0

If S1[i] != S2[j]

DP[i][j] = MAX(DP[i - 1][j], DP[i][j - 1])

그렇다면…

Gold 5 - LCS (#9251)

		A	C	A	Y	K	P
	0	0	0	0	0	0	0
C	1	0	1	1	1	1	1
A	2	1	1	2	0	0	0
P	3	0	0	0	0	0	0
C	4	0	0	0	0	0	0
A	5	0	0	0	0	0	0
K	6	0	0	0	0	0	0

If S1[i] != S2[j]

DP[i][j] = MAX(DP[i - 1][j], DP[i][j - 1])

If S1[i] == S2[j]

DP[i][j] = DP[I - 1][j - 1] + 1

그렇다면…

Gold 5 - LCS (#9251)

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

If S1[i] != S2[j]

DP[i][j] = MAX(DP[i - 1][j], DP[i][j - 1])

If S1[i] == S2[j]

DP[i][j] = DP[I - 1][j - 1] + 1

좀 많이 어렵습니다…

Gold 2 - 팰린드롬? (#10942)

요약

- 수열을 앞에서 읽던 뒤에서 읽던 똑같다면 이것을 팰린드롬이라고 부른다.
- 전체 길이가 N인 수열에서 S부터 E까지의 인덱스를 포함하는 부분수열이 팰린드롬인지 확인하려고 한다.
- M개의 질의가 주어졌을 때, 해당 부분수열이 팰린드롬인지 아닌지 확인하는 프로그램을 작성하시오.

제약조건

- N의 범위는 $1 <= N <= 2{,}000$ 이다.
- M의 범위는 $1 <= M <= 1{,}000{,}000$ 이다.
- 수열에서 각각의 값의 범위는 $1 <= N_i <= 100{,}000$ 이다.

사실 별로 안 어렵습니다.

Gold 2 - 팰린드롬? (#10942)

1 2 1 3 1 2 1

팰린드롬?

사실 별로 안 어렵습니다.

Gold 2 - 팰린드롬? (#10942)

당연한 것들

1 2 1 3 1 2 1

- 1글자: 당연히 팰린드롬.

사실 별로 안 어렵습니다.

Gold 2 - 팰린드롬? (#10942)

당연한 것들

1 2 1 3 1 2 1

- 1글자: 당연히 팰린드롬.
- 2글자: 두개의 숫자가 모두 같아야 팰린드롬.

사실 별로 안 어렵습니다.

Gold 2 - 팰린드롬? (#10942)

당연한 것들

1 2 1 3 1 2 1

- 1글자: 당연히 팰린드롬.
- 2글자: 두개의 숫자가 모두 같아야 팰린드롬.
- 3글자: 가운데를 대칭으로 숫자가 같아야 팰린드롬.

사실 별로 안 어렵습니다.

Gold 2 - 팰린드롬? (#10942)

N = K

1 ? ? ? ? ? 1

안쪽 ???는 팰린드롬일까?

- 팰린드롬이라면?
- 팰린드롬이 아니라면?

사실 별로 안 어렵습니다.

Gold 2 - 팰린드롬? (#10942)

N = K

1 ? ? ? ? ? 1

안쪽 ???는 팰린드롬일까?

- 팰린드롬이라면? → 바깥쪽 수가 같으면 OK!
- 팰린드롬이 아니라면? → 팰린드롬이 될 수 없음…

또 너야?

Gold 2 - 팰린드롬? (#10942)

```
for k from 1 to |V|
    for i from 1 to |V|
        for j from 1 to |V|
            if dist[i][j] > dist[i][k] + dist[k][j] then
                set dist[i][j] <- dist[i][k] + dist[k][j]
            end if
```

값을 하나씩 넣어가면서 테이블을 채워 나갔지...

아이고 이게 뭐야?

Gold 3 - LCS 3 (#1958)

요약

- LCS (Longest Common Subsequence)은 두 수열이 주어졌을 때, 모두의 부분 수열 중 가장 긴 것을 의미한다.
- 세 문자열이 주어졌을 때, 이것의 LCS의 길이를 구하는 프로그램을 작성하시오.

제약조건

- 문자열의 길이는 1 <= len <= 100 이다.

이건 안됩니다.

Gold 3 - LCS 3 (#1958)

S_1, S_2, S_3에 대해

$LCS(S_1, S_2, S_3) = LCS(LCS(S_1, S_2), S_3)$는 안될까?

예시도 되는데…

힌트!

Gold 3 - LCS 3 (#1958)

If S1[i] != S2[j]

DP[i][j] = MAX(DP[i - 1][j], DP[i][j - 1])

If S1[i] == S2[j]

DP[i][j] = DP[I - 1][j - 1] + 1

인자가 3개로 늘어났으니 I, J, K에 대한 관계식으로 바꿔보자!

Gold 4 - 내리막길 (#1520)

요약

- 왼쪽 맨 위에서 출발해 오른쪽 맨 아래로 가려고 한다.
- 그래프의 각각의 점에는 높이가 적혀 있는데, 항상 높이가 더 낮은 지점으로만 이동하려고 한다.
- 그래프가 주어졌을 때, 위 조건에 맞는 경로의 수를 출력하는 프로그램을 작성하시오.

제약조건

- 가로의 크기 N의 범위는 $1 <= N <= 500$ 이다.
- 세로의 크기 M의 범위는 $1 <= M <= 500$ 이다.
- 각각의 칸에 적힌 값의 범위는 $1 <= G_{ij} <= 10,000$ 이다.

Gold 4 - 내리막길 (#1520)

50	45	37	32	30
35	50	40	20	25
30	30	25	17	28
27	24	22	15	10

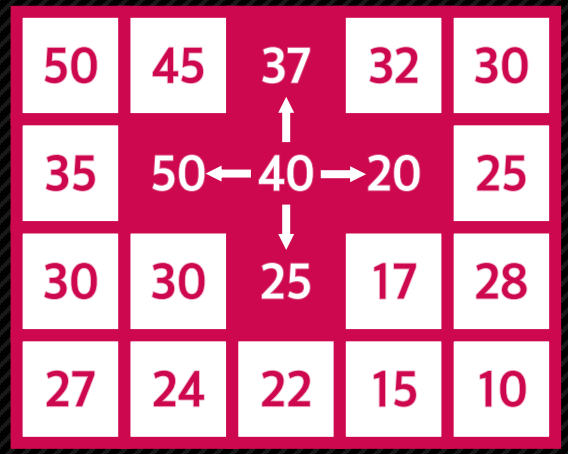
- 우리가 구하려는 값이 뭘까?
- 특정 상황에서 선택 가능한 경우는 무엇이 있을까?
- 위에서 정리한 것을 기반으로, 현재 값은 어떻게 도출할 수 있을까?

Gold 4 - 내리막길 (#1520)

50	45	37	32	30
35	50	40	20	25
30	30	25	17	28
27	24	22	15	10

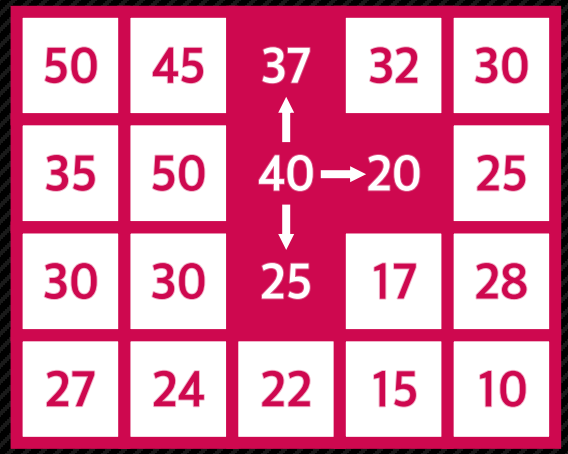
- 우리가 구하려는 값이 뭘까? → 이동할 수 있는 모든 경우의 '수'
- 특정 상황에서 선택 가능한 경우는 무엇이 있을까?
- 위에서 정리한 것을 기반으로, 현재 값은 어떻게 도출할 수 있을까?

Gold 4 - 내리막길 (#1520)



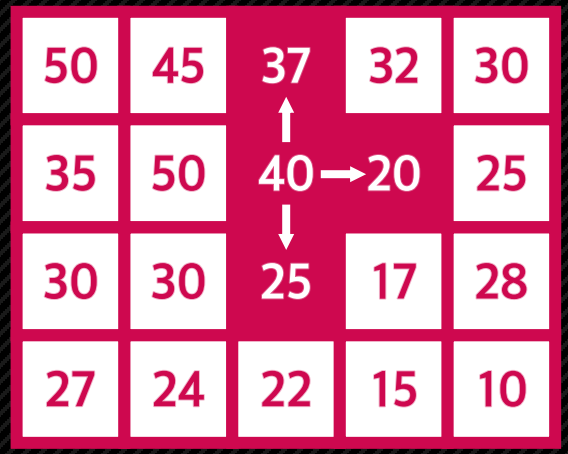
- 우리가 구하려는 값이 뭘까? → 이동할 수 있는 모든 경우의 '수'
- 특정 상황에서 선택 가능한 경우는 무엇이 있을까? → 상/하/좌/우 (단, 높이가 낮아야 함.)
- 위에서 정리한 것을 기반으로, 현재 값은 어떻게 도출할 수 있을까?

Gold 4 - 내리막길 (#1520)



- 우리가 구하려는 값이 뭘까? → 이동할 수 있는 모든 경우의 '수'
- 특정 상황에서 선택 가능한 경우는 무엇이 있을까? → 상/하/좌/우 (단, 높이가 낮아야 함.)
- 위에서 정리한 것을 기반으로, 현재 값은 어떻게 도출할 수 있을까?

Gold 4 - 내리막길 (#1520)



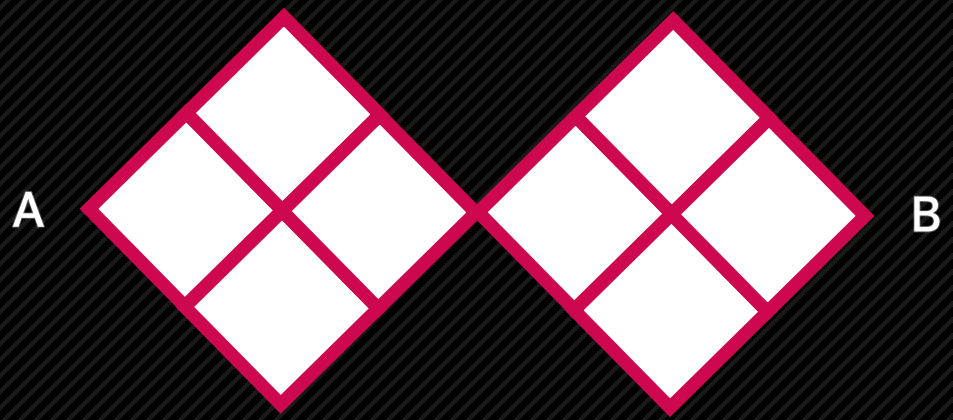
- 우리가 구하려는 값이 뭘까? → 이동할 수 있는 모든 경우의 '수'
- 특정 상황에서 선택 가능한 경우는 무엇이 있을까? → 상/하/좌/우 (단, 높이가 낮아야 함.)
- 위에서 정리한 것을 기반으로, 현재 값은 어떻게 도출할 수 있을까? → DP[I][J] = DP[I-1][J] + DP[I+1][J] + DP[I][J-1] + DP[I][J+1]

고딩때로 돌아갑시다.

5. 그림과 같이 마름모 모양으로 연결된 도로망이 있다.

이 도로망을 따라 A 지점에서 출발하여 B 지점까지 최단거리로 가는 경우의 수는? [3점]

(2013학년도 대학수학능력시험 9월 모의고사 수리영역 가형 5번)

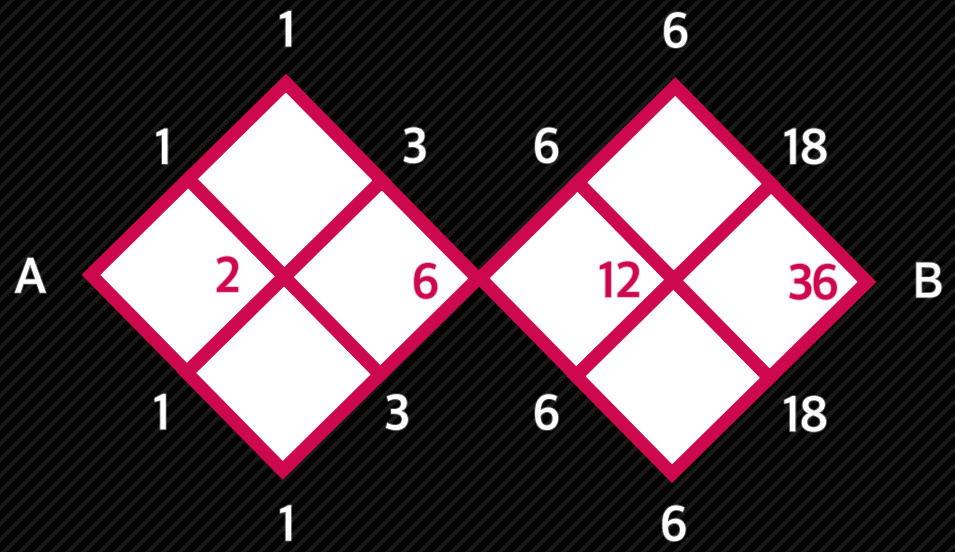


어떻게 풀었더라…

5. 그림과 같이 마름모 모양으로 연결된 도로망이 있다.

이 도로망을 따라 A 지점에서 출발하여 B 지점까지 최단거리로 가는 경우의 수는? [3점]

(2013학년도 대학수학능력시험 9월 모의고사 수리영역 가형 5번)

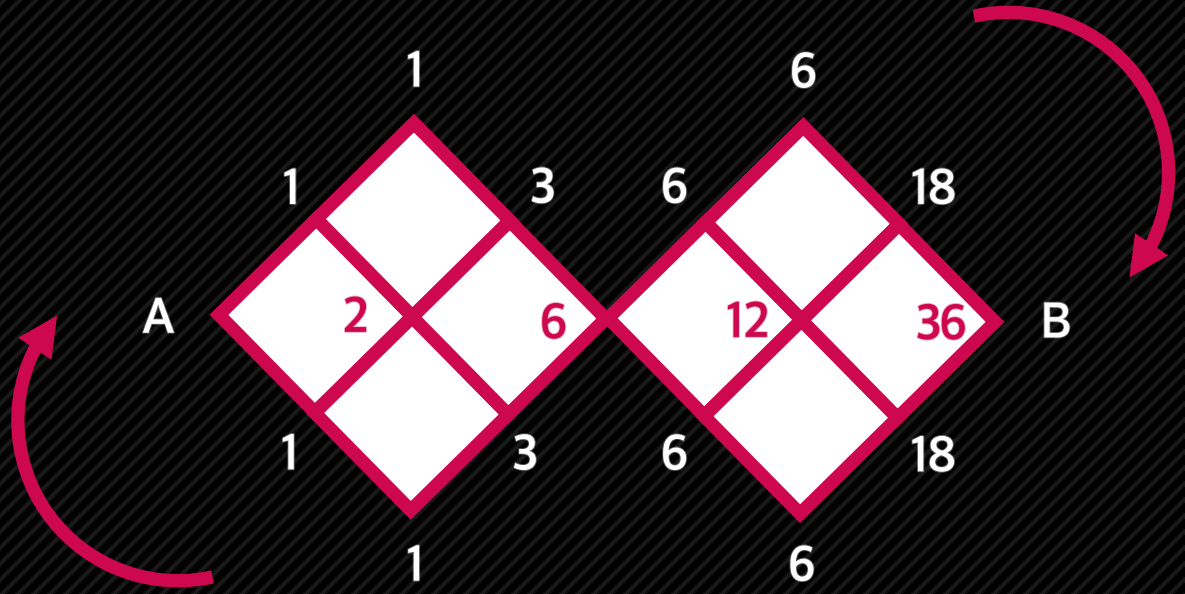


어찌 보면 너도!

5. 그림과 같이 마름모 모양으로 연결된 도로망이 있다.

이 도로망을 따라 A 지점에서 출발하여 B 지점까지 최단거리로 가는 경우의 수는? [3점]

(2013학년도 대학수학능력시험 9월 모의고사 수리영역 가형 5번)

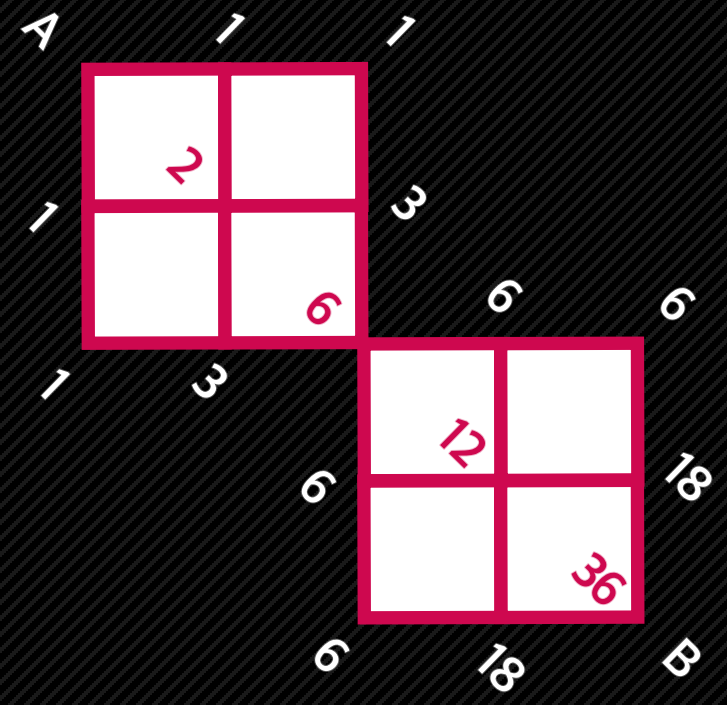


어찌 보면 너도!

5. 그림과 같이 마름모 모양으로 연결된 도로망이 있다.

이 도로망을 따라 A 지점에서 출발하여 B 지점까지 최단거리로 가는 경우의 수는? [3점]

(2013학년도 대학수학능력시험 9월 모의고사 수리영역 가형 5번)

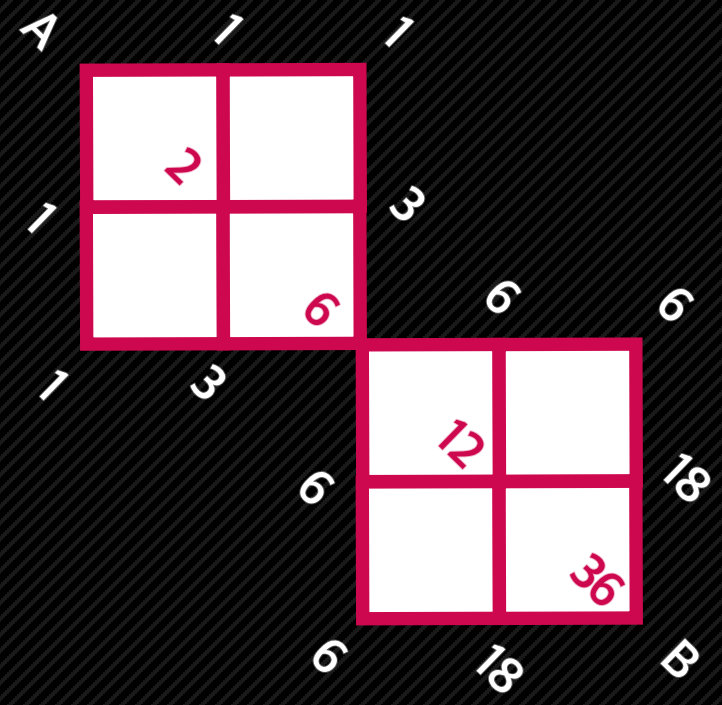


어찌 보면 너도!

5. 그림과 같이 마름모 모양으로 연결된 도로망이 있다.

이 도로망을 따라 A 지점에서 출발하여 B 지점까지 최단거리로 가는 경우의 수는? [3점]

(2013학년도 대학수학능력시험 9월 모의고사 수리영역 가형 5번)



$$DP[I][J] = DP[I-1][J] + DP[I][J-1]$$

은근히 자주 나오니 연습…

Gold 5 - 관악산 등반 (#14699)

요약

- 1부터 N까지 쉼터가 있고, 사이를 잇는 M개의 길이 있다.
- 최대한 많은 쉼터를 방문하고자 하나, 내려가는 길은 가지 않는다.
- 출발하는 쉼터가 주어졌을 때, 방문 가능한 쉼터의 수의 최댓값을 출력하는 프로그램을 작성하시오.

제약조건

- N의 범위는 2 <= N <= 5,000 이다.
- M의 범위는 1 <= M <= 100,000 이다.

그래프… 딱 하나만 더?

Gold 3 - 욕심쟁이 판다 (#1937)

요약

- 판다는 N * N 그래프에서 특정 지점에 자라있는 모든 대나무를 먹은 다음, 상하좌우 중 한군데로 이동한다.
- 단, 판다는 이동한 곳에 있는 대나무의 수가 이전보다 적을 경우 불만을 갖고 단식 투쟁을 한다.
- 판다를 최대한 생존할 수 있는 날을 구하는 프로그램을 작성하시오.

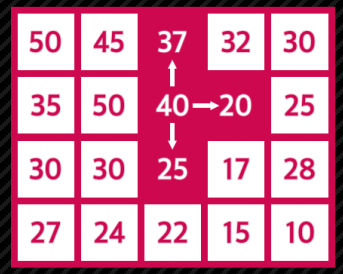
제약조건

- N의 범위는 $1 <= N <= 500$ 이다.
- 각각의 칸의 대나무의 양은 $1 <= G_{ij} <= 1{,}000{,}000$ 이다.

이것도 힌트만...

Gold 3 - 욕심쟁이 판다 (#1937)

Gold 4 - 내리막길 (#1520)



- 우리가 구하려는 값이 뭘까? → 이동할 수 있는 모든 경우의 '수'
- 특정 상황에서 선택 가능한 경우는 무엇이 있을까? → 상/하/좌/우 (단, 높이가 낮아야 함.)
- 위에서 정리한 것을 기반으로, 현재 값은 어떻게 도출할 수 있을까? → DP[i][J] = DP[i-1][J] + DP[i+1][J] + DP[i][J-1] + DP[i][J+1]

- 해당 문제와 유사하나, 시작점이 정의되지 않았으므로 dp값이 -1인 모든 칸에 대해 값을 구해주자!

많이 어렵습니다.

Gold 2 - 가장 긴 증가하는 부분 수열 2 (#12015)

요약

- 수열 A가 주어졌을 때, 가장 긴 증가하는 부분 수열을 구하는 프로그램을 작성하시오.

제약조건

- 수열 A의 길이의 범위는 1 <= len(A) <= 1,000,000 이다.

무슨 차이죠?

Silver 2 - 가장 긴 증가하는 부분 수열 (#11053)

- 수열 A의 길이의 범위는 1 <= len(A) <= 1,000 이다.

Gold 2 - 가장 긴 증가하는 부분 수열 2 (#12015)

- 수열 A의 길이의 범위는 1 <= len(A) <= 1,000,000 이다.

기존의 풀이를 떠올려보자…

Silver 2 - 가장 긴 증가하는 부분수열 (#11053)

$$For\ all\ n+k\ s.t.\ Data_{n+k} > Data_n$$
$$DP_n = MAX(DP_{n+k}) + 1$$

기존의 풀이를 떠올려보자…

Silver 2 - 가장 긴 증가하는 부분수열 (#11053)

$O(N^2)$

For all n+k s.t. $Data_{n+k} > Data_n$

$DP_n = MAX(DP_{n+k}) + 1$

이런 건 어떨까?

Gold 2 - 가장 긴 증가하는 부분 수열 2 (#12015)

맨 뒤에 있는 숫자는 커야 할까? 작아야 할까?

이런 건 어떨까?

Gold 2 - 가장 긴 증가하는 부분 수열 2 (#12015)

맨 뒤에 있는 숫자는 커야 할까? 작아야 할까?

맨 뒤를 최소화하면, 뒤에 더 많은 숫자가 붙을거야!

맨 뒤만 작아야 할까?

Gold 2 - 가장 긴 증가하는 부분 수열 2 (#12015)

[10, 20, 30, 40, 50, 60]

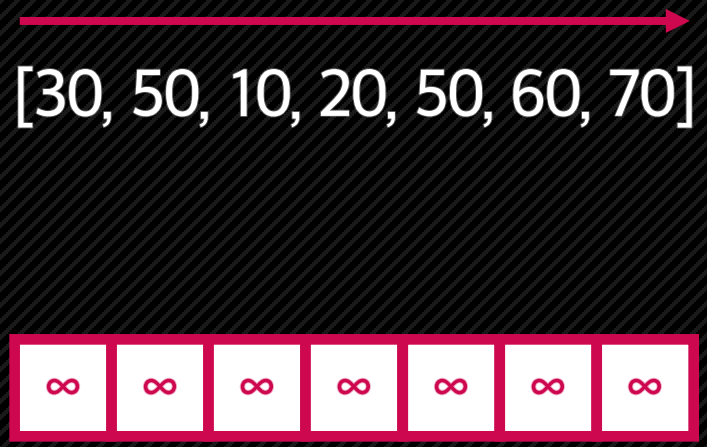
[10, 30, 40]

[10, 20, 30]

결국 전체적으로 가능한 최소값을 유지해야 함!

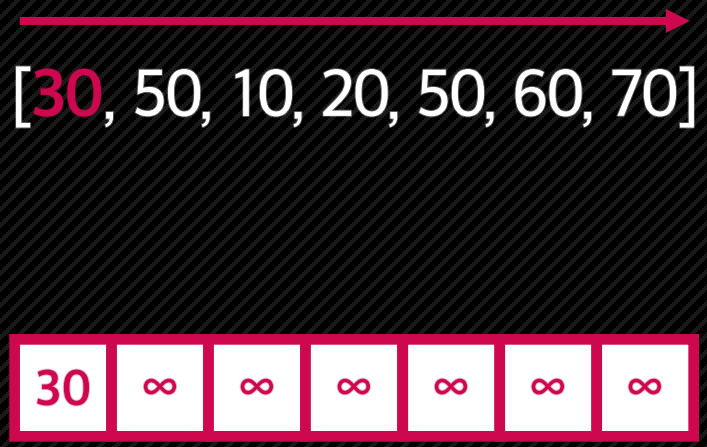
일단 아이디어는 알겠고…

Gold 2 - 가장 긴 증가하는 부분 수열 2 (#12015)



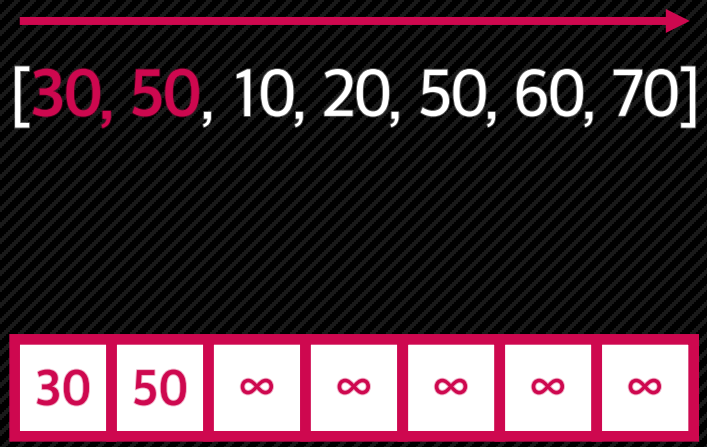
일단 아이디어는 알겠고…

Gold 2 - 가장 긴 증가하는 부분 수열 2 (#12015)



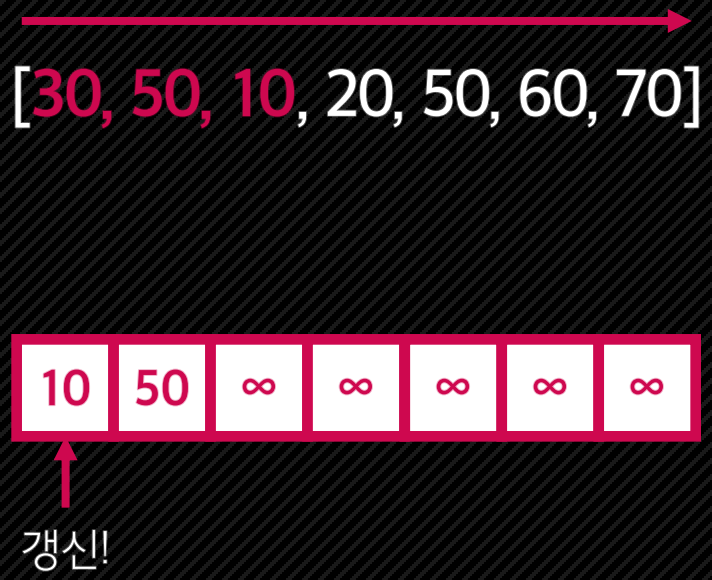
일단 아이디어는 알겠고…

Gold 2 - 가장 긴 증가하는 부분 수열 2 (#12015)



일단 아이디어는 알겠고…

Gold 2 - 가장 긴 증가하는 부분 수열 2 (#12015)

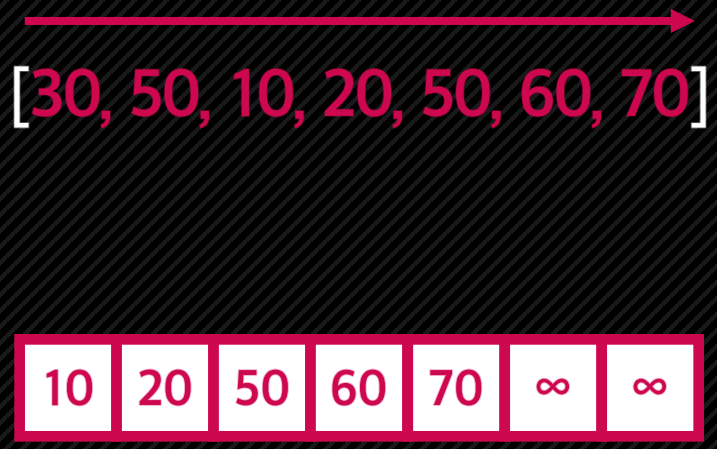


일단 아이디어는 알겠고…

Gold 2 - 가장 긴 증가하는 부분 수열 2 (#12015)

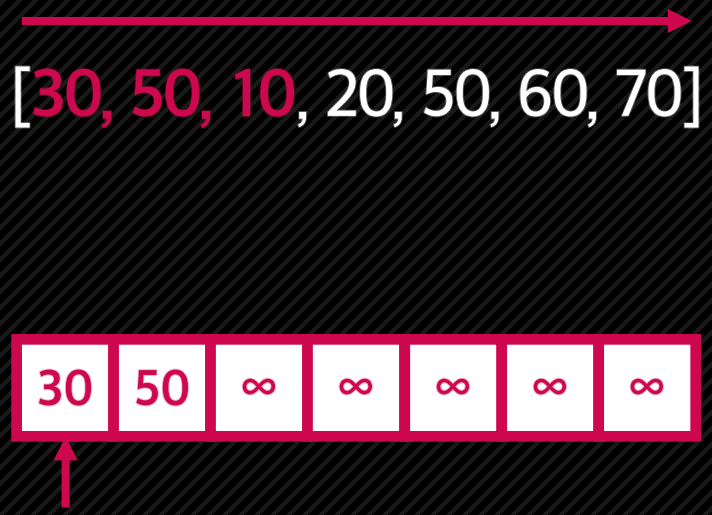
일단 아이디어는 알겠고…

Gold 2 - 가장 긴 증가하는 부분 수열 2 (#12015)

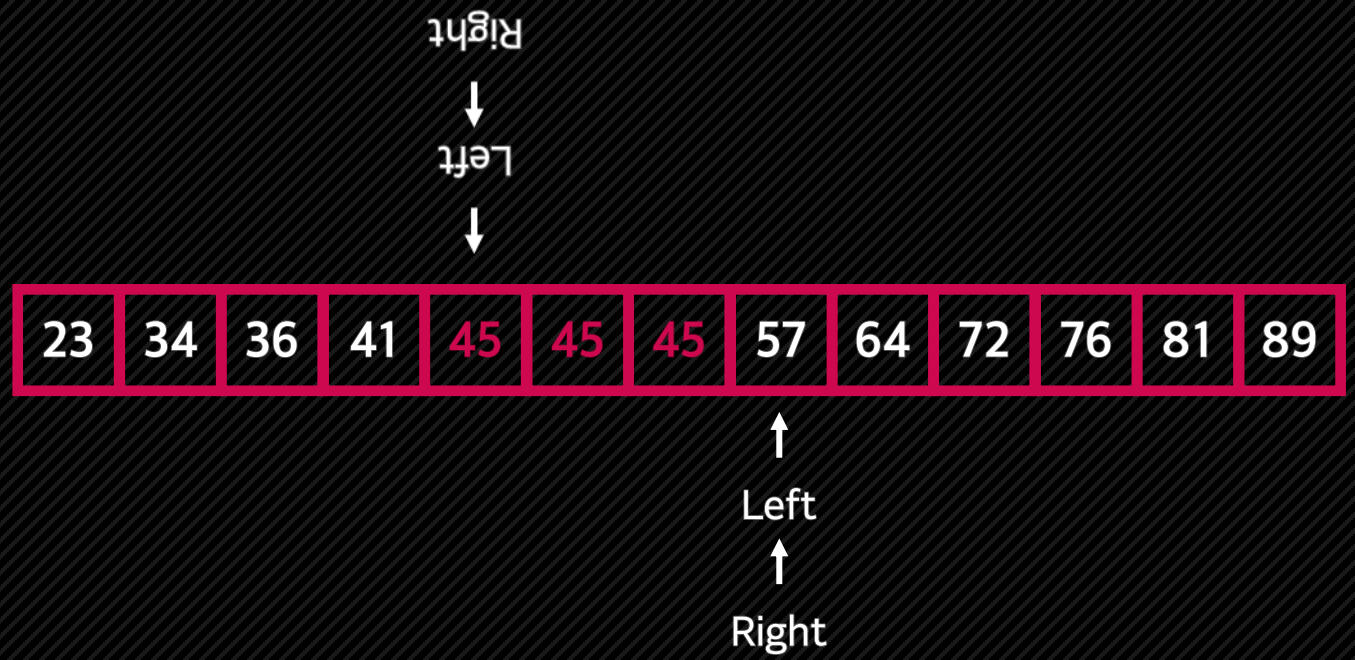


근데 어떻게 찾지??

Gold 2 - 가장 긴 증가하는 부분 수열 2 (#12015)



기억나세요?



- Lower Bound: 원하는 값 N 이상이 처음 나오는 위치
- Upper Bound: 원하는 값 N을 처음으로 초과하는 위치

추가 추천 문제

Gold 5 - 1학년 (#5557)

- 생긴건 1차원으로 생겼지만, 고민해보면 DP 테이블은 좀 다르게 생겼을 겁니다.

Gold 4 - 색상환 (#2482)

- 원형 구조에서의 DP 문제입니다. 한 지점의 값을 고정해서 접근하면 원을 1차원으로 볼 수 있을거에요!

Gold 3 - Dance Dance Revolution (#2342)

- 왼발과 오른발의 정보를 어떻게 저장할지 고민해보세요. 2차원에 꼭 얽매일 필요가 있을까요?

Gold 2 - 양팔저울 (#2629)

- 냅색을 떠올려보세요. 상당히 꼬아 놓은 문제이지만 해결할 수 있을 겁니다.

“Any Question?”