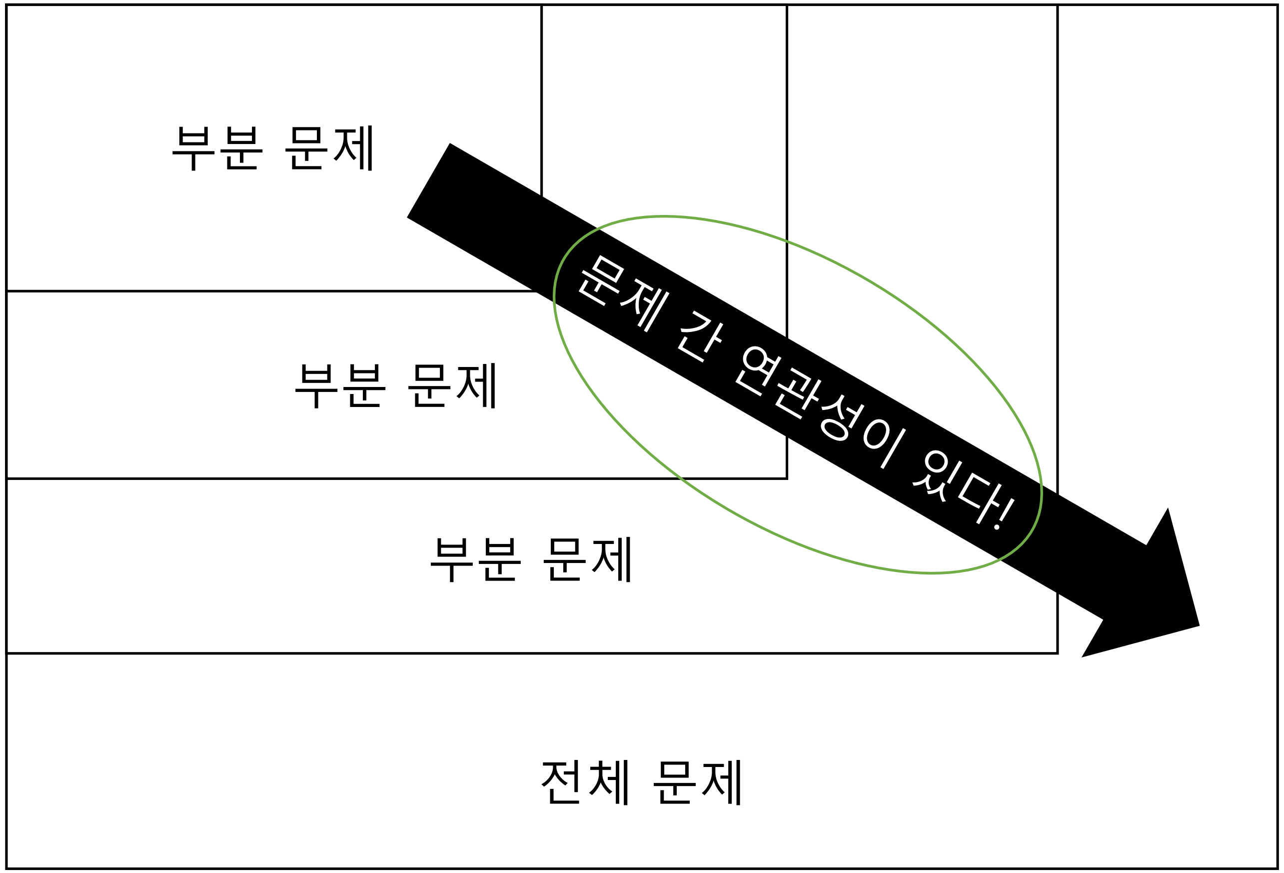
대경 HuStar아카데미
알고리즘 실습

다이나믹 프로그래밍

PIAI 인공지능연구원

POSTECH
POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Dynamic Programming?



작은 부분 문제를 풀고, 그것들을 이용해 큰 전체 문제를 해결!

- Bottom Up 방식
- 분할한 문제 간의 연관성 O

c.f. Divide & Conquer
- Top Down 방식
- 분할한 문제 간의 연관성 X

Dynamic Programming?

1. **부분 문제를 명확하게 정의합니다.**
   - 원래 문제에서 보통 데이터의 크기만 줄이는 경우로 생각
   - 예외적인 케이스도 존재 ➔ 기존 부분 문제 + 제한 조건을 추가

2. **원래 답의 위치를 파악합니다.**
   - 부분 문제가 곧 답의 힌트!
   - 부분 문제의 해답을 모아둘 **배열** 등을 생성 가능
   - 부분 문제가 틀림을 알아내는 데에도 주요한 포인트

사람이 해결해야 함!

3. **재귀 식(점화 식)을 부분 문제의 정의와 수학적 논리에 따라 잘 세웁니다.**
   - 부분 문제를 해결하기 위해선, 더 작은 부분 문제들의 답을 이용
   - Backward Analysis: 이 문제의 답이 어디서 올 수 있었는가를 분석
   - 식에 나오게 되는 변수들을 통해 반복 문을 어떻게 해야 할지 분석
   - 식이 제대로 세워지지 않는다면, 1번으로 돌아감

4. **기저 조건(초기값)을 세웁니다.**
   - 문제 조건으로 주어진 초기 값
   - 3번 식에서는 값을 구할 수 없는 예외적인 경우

Dynamic Programming?

자연수 N이 주어지면, 1~N까지의 합을 더하는 프로그램

1. **부분 문제를 명확하게 정의**
   T[i]: 1~i 까지의 합

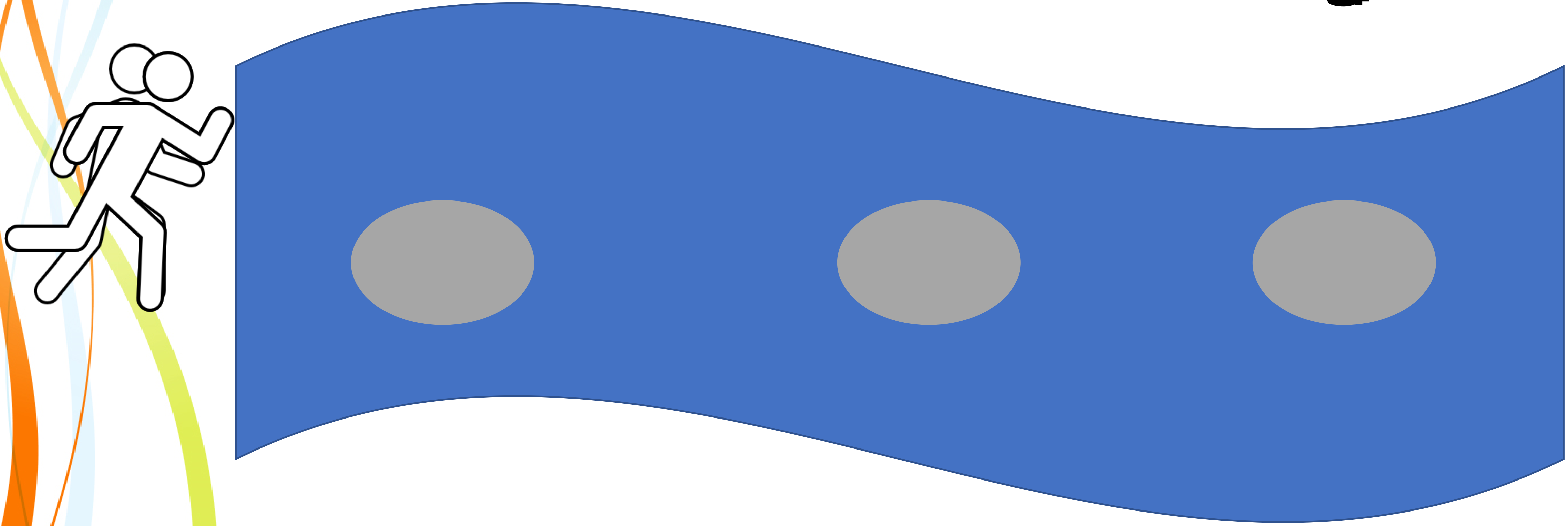
2. **원래 답의 위치**
   T[N]일 것
   ➔ print(T[N])
   ➔ 답이 들어갈 T 배열을 생성

3. **재귀 식(점화 식)**
   T[i] = i + T[i-1]
   ➔ for문이 i로 돈다. (N까지)

4. **기저 조건(초기값)**
   i가 1일 때 T[i] = 1
   ➔ for문 내에 조건이 들어간다.
   ➔ 조건이 아닌 경우는 3번

```
N = int(input())
```

01. 돌다리 건너가기



01. 돌다리 건너가기

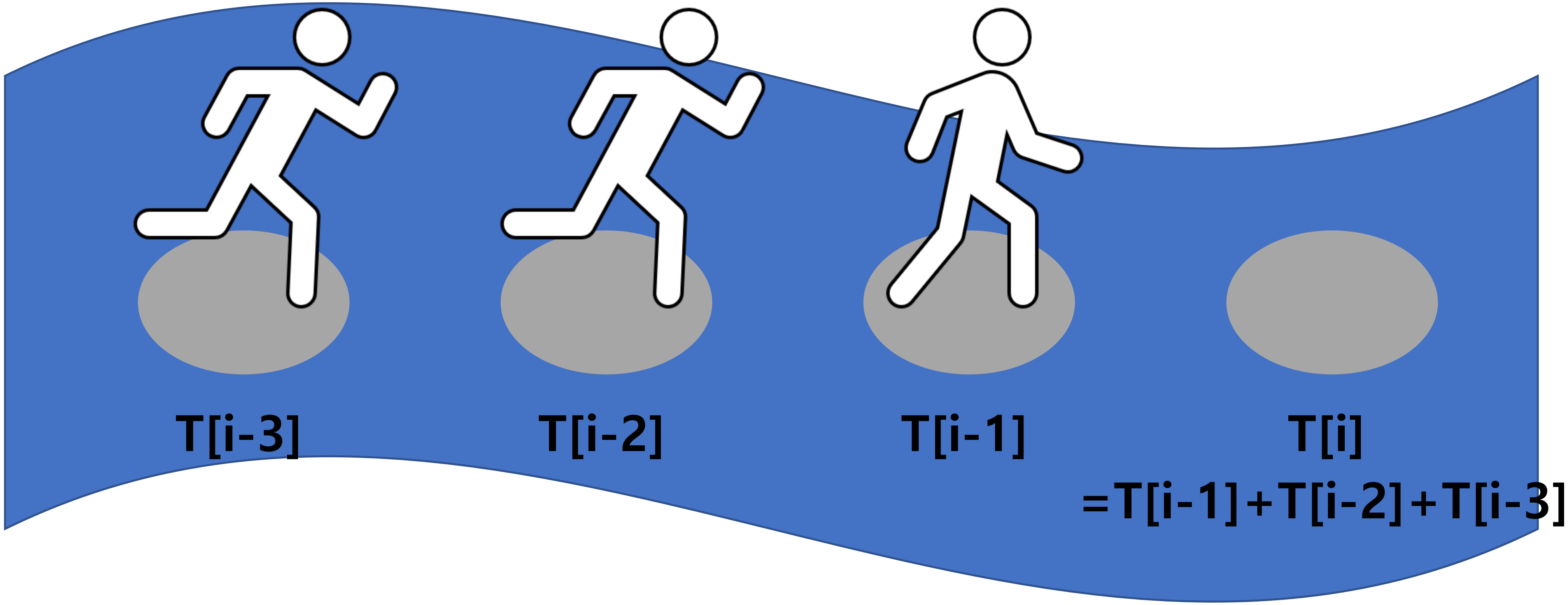
n번째 돌다리를 가는 경우의 수를 1904101441로 나눈 나머지를 계산합니다.

Dynamic Programming의 핵심은 대부분 펜과 노트에서 완성됩니다.

※ $(\boldsymbol{a}+\boldsymbol{b}) \bmod \boldsymbol{m} = \left((\boldsymbol{a} \bmod \boldsymbol{m}) + (\boldsymbol{b} \bmod \boldsymbol{m})\right) \bmod \boldsymbol{m}$

분할 정복의 "n^k mod m 효율적으로 계산하기" 문제를 떠올려봅시다!

01. 돌다리 건너가기



01. 돌다리 건너가기

1. 부분 문제를 명확하게 정의
   - T[i]: i번 돌다리에 도달하는 경우의 수 % 1904101441
2. 원래 답의 위치
   - T[n]
3. 재귀 식(점화 식)
   - T[i] = (T[i-1] + T[i-2] + T[i-3]) % 1904101441
4. 기저 조건(초기값)
   - i=1일 때, T[i] = 1
   - i=2일 때, T[i] = 2
   - i=3일 때, T[i] = 4

T[i] = T[i-1] + T[i-2] + T[i-3]

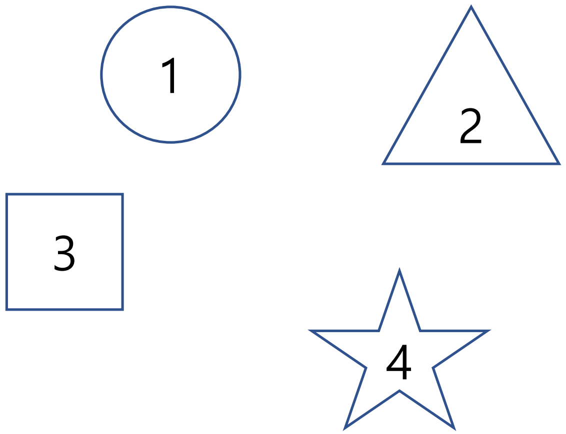
i	1	2	3	4	5	6	7	8	9	10
T[i]	1	2	4	7	13	24	44	81	149	274

01. 돌다리 건너가기

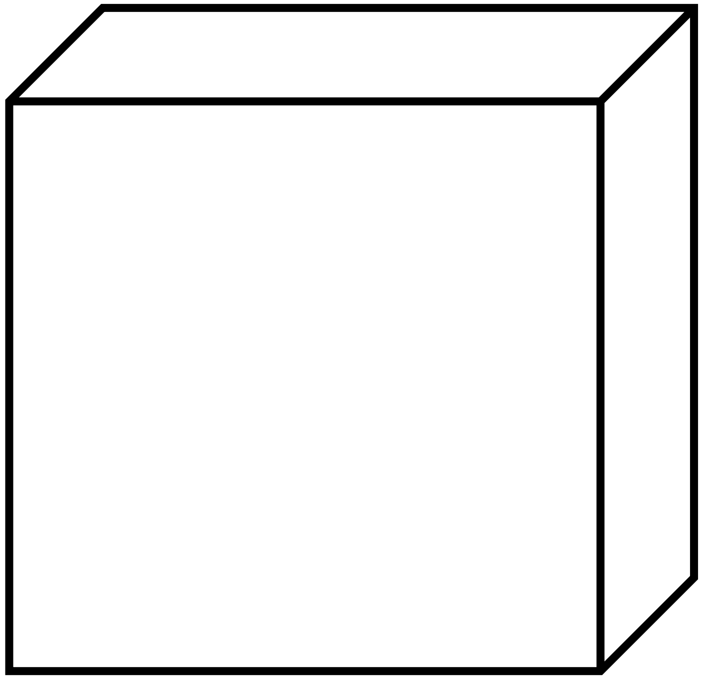
```
T = int(input())
for _ in range(T):
    N = int(input())
    #T[i]: i번째 돌다리에 도달하는 경우의 수
    T = [0]*(N+1)
    for i in range(1,N+1):
        if i == 1: #if ~ elif: 초기값
            T[1] = 1
        elif i == 2:
            T[2] = 2
        elif i == 3:
            T[3] = 4
        else: #점화식
            T[i] = (T[i-1]+T[i-2]+T[i-3])%1904101441
    print(T[N])
```

02. 배송비 절약

	1	2	3	4
무게	6	1	5	3
가치	10	30	100	200



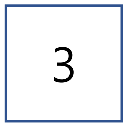
무게: 0 (MAX 10)
가치: 0

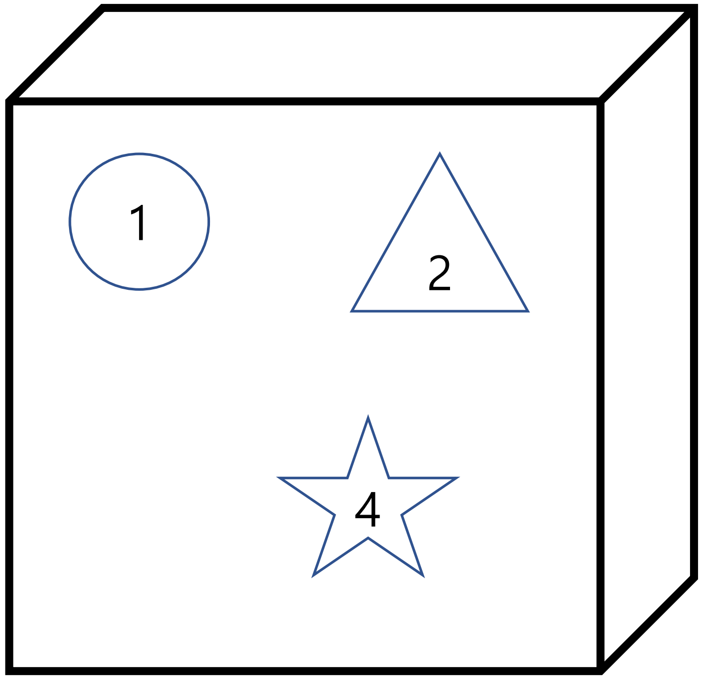


02. 배송비 절약

	1	2	3	4
무게	6	1	5	3
가치	10	30	100	200

무게: 6+1+3=10
가치: 10+30+200=240

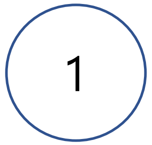


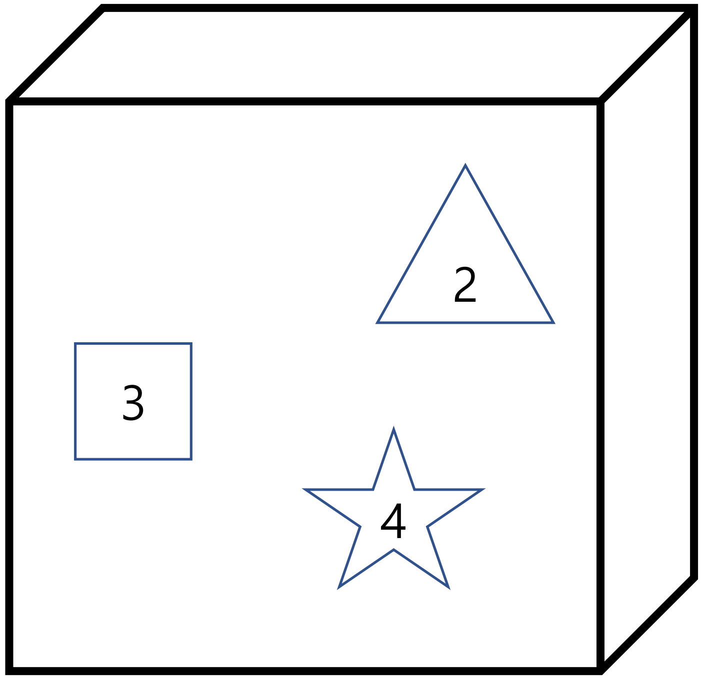


02. 배송비 절약

	1	2	3	4
무게	6	1	5	3
가치	10	30	100	200

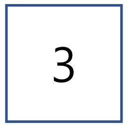
무게: 1+5+3=9 (MAX 10)
가치: 30+100+200=330





02. 배송비 절약

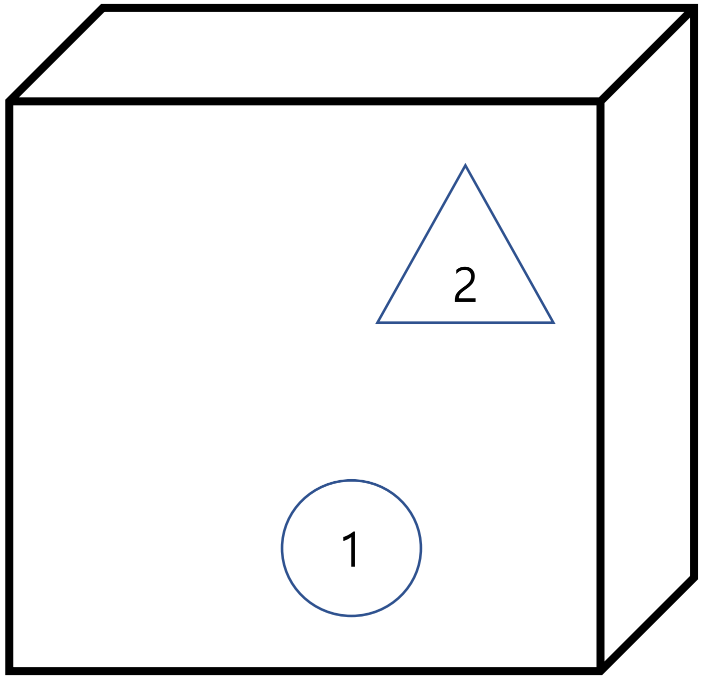
	1	2	3
무게	1	1	10
가치	100	100	300
가치/무게	100	100	30



Greedy를 사용할 경우

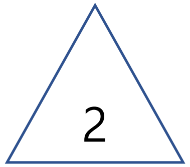
<무거운 용액> 문제와 같이 가치/무게 순으로 선택한다면 옆의 그림과 같이 남은 8칸의 무게에 3번을 잘라서 넣을 수 없어서 손해

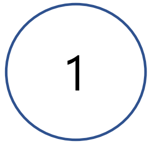
무게: 1+1=2 (MAX 10)
가치: 100+100=200



02. 배송비 절약

	1	2	3
무게	1	1	10
가치	100	100	300
가치/무게	100	100	30

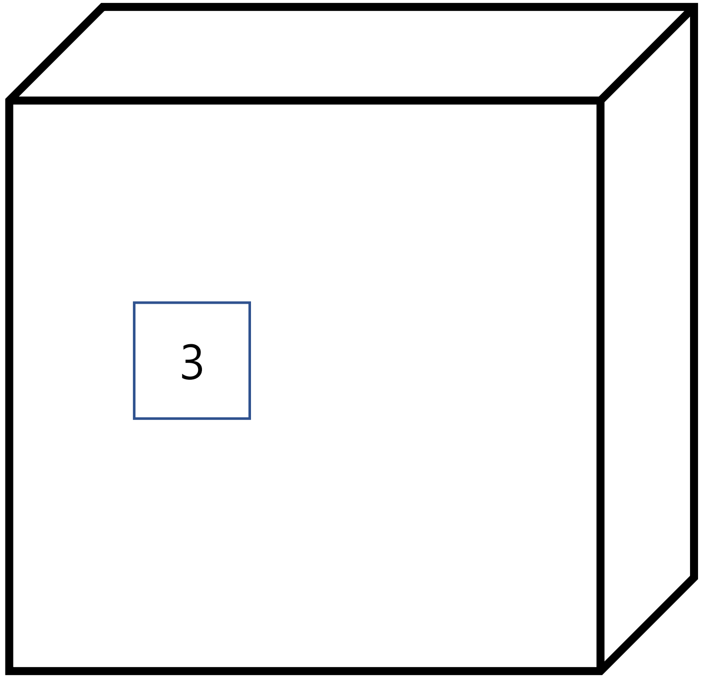




실제 정답

3번 물건이 가치/무게는 더 낮지만 최종적인 가치는 Greedy를 사용한 결과보다 높음

무게: 10 (MAX 10)
가치: 300



02. 배송비 절약

아이디어

D[i][j] = i번째까지의 물건을 가지고 무게 j를 채울 때 가치 합의 최대값

i \ j	0	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	10	10	10	10	10
2	0										
3	0										
4	0										??

02. 배송비 절약

아이디어

D[i][j] = i번째까지의 물건을 가지고 무게를 j 이하로 채울 때 가치 합의 최대값

첫번째 물건을 가지고 무게 0부터 C까지 만들 때 가치의 최대값을 저장.

i-1번째까지의 물건을 가지고 j 이하의 무게를 만드는 최대 가치에 i번째 물건이 추가된 경우.

$$D[i][j] = \begin{cases} \mathrm{D[i-1][j]} & \text{if } w_i > j \\ \max(\mathrm{D[i-1][j]}, \mathrm{D[i-1][j-}w_i\mathrm{]}+v_i) & \text{if } w_i \le \mathrm{j} \end{cases}$$

02. 배송비 절약

$$D[i][j] = \begin{cases} \mathrm{D[i-1][j]} & \text{if } \ w_i > j \\ \max(\mathrm{D[i-1][j]}, \mathrm{D[i-1][j-}w_i\mathrm{]}+v_i) & \text{if } \ w_i \le \mathrm{j} \end{cases}$$

i \ j	0	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	10	10	10	10	10
2	0	30									
3	0										
4	0										??

	1	2	3	4
무게	6	1	5	3
가치	10	30	100	200

02. 배송비 절약

$$D[i][j] = \begin{cases} \mathrm{D[i{-}1][j]} & \text{if } \ w_i > j \\ \max(\mathrm{D[i{-}1][j]}, \mathrm{D[i{-}1][j{-}}w_i\mathrm{]{+}}v_i) & \text{if } \ w_i \le \mathrm{j} \end{cases}$$

i \ j	0	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	10	10	10	10	10
2	0	30	30								
3	0										
4	0										??

	1	2	3	4
무게	6	1	5	3
가치	10	30	100	200

02. 배송비 절약

$$D[i][j] = \begin{cases} \mathrm{D[i-1][j]} & \text{if } \ w_i > j \\ \max(\mathrm{D[i-1][j]}, \mathrm{D[i-1][j-}w_i\mathrm{]+}v_i) & \text{if } \ w_i \le \mathrm{j} \end{cases}$$

i \ j	0	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	10	10	10	10	10
2	0	30	30	30	30	30	30				
3	0										
4	0										??

	1	2	3	4
무게	6	1	5	3
가치	10	30	100	200

02. 배송비 절약

$$D[i][j] = \begin{cases} \mathrm{D[i{-}1][j]} & \text{if } \ w_i > j \\ \max(\mathrm{D[i{-}1][j]}, \mathrm{D[i{-}1][j{-}}w_i\mathrm{]}{+}v_i) & \text{if } \ w_i \le \mathrm{j} \end{cases}$$

i \ j	0	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	10	10	10	10	10
2	0	30	30	30	30	30	30	40			
3	0										
4	0										??

	1	2	3	4
무게	6	1	5	3
가치	10	30	100	200

02. 배송비 절약

$$D[i][j] = \begin{cases} \mathrm{D[i{-}1][j]} & \text{if } \ w_i > j \\ \max(\mathrm{D[i{-}1][j]}, \mathrm{D[i{-}1][j{-}}w_i\mathrm{]}{+}v_i) & \text{if } \ w_i \leq \mathrm{j} \end{cases}$$

i \ j	0	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	10	10	10	10	10
2	0	30	30	30	30	30	30	40	40	40	40
3	0	30	30	30	30	100					
4	0										??

	1	2	3	4
무게	6	1	5	3
가치	10	30	100	200

02. 배송비 절약

$$D[i][j] = \begin{cases} \mathrm{D[i-1][j]} & \text{if } \ w_i > j \\ \max(\mathrm{D[i-1][j]}, \mathrm{D[i-1][j-}w_i\mathrm{]+}v_i) & \text{if } \ w_i \leq \mathrm{j} \end{cases}$$

i \ j	0	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	10	10	10	10	10
2	0	30	30	30	30	30	30	40	40	40	40
3	0	30	30	30	30	100	130				
4	0										??

	1	2	3	4
무게	6	1	5	3
가치	10	30	100	200

02. 배송비 절약

$$D[i][j] = \begin{cases} \mathrm{D[i-1][j]} & \text{if } \ w_i > j \\ \max(\mathrm{D[i-1][j]}, \mathrm{D[i-1][j-}w_i\mathrm{]}+v_i) & \text{if } \ w_i \le \mathrm{j} \end{cases}$$

i \ j	0	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	10	10	10	10	10
2	0	30	30	30	30	30	30	40	40	40	40
3	0	30	30	30	30	100	130	130	130	130	130
4	0	30	30	200	230	230	230	230	300	330	330

	1	2	3	4
무게	6	1	5	3
가치	10	30	100	200

02. 배송비 절약

```python
t = int(input())
for _ in range(t):
    n,C = map(int,input().split())
    w = list(map(int,input().split()))
    v = list(map(int,input().split()))
    #T[i][j]: 0~i번까지의 물건까지 사용하여 무게 j를 채울때 가치 합의 최대값
    T = [[0]*(C+1) for _ in range(n)]
    for i in range(n):
        for j in range(1,C+1):
            if i == 0: #초기값
                if w[i] > j: #0번 물건을 가방에 넣을 수 없는 경우
                    T[i][j] = 0
                else: #있는 경우
                    T[i][j] = v[i]
            else: #점화식
                if w[i] > j: #i번 물건을 가방에 넣을 수 없는 경우
                    T[i][j] = T[i-1][j]
                else: #있는 경우
                    T[i][j] = max(T[i-1][j], T[i-1][j-w[i]]+v[i])
    #해답의 위치
    print(T[n-1][C])
```

03. 세계 암기대회

지우가 오른쪽 아래로 가면서 잃을 수 있는 가장 적은 점수를 계산해주세요!

1	0	1	0	1
0	1	1	1	0
0	1	0	1	1

03. 세계 암기대회

각 칸으로 가면서 잃을 수 있는 최저 점수를 저장한다면?

1	0	1	0	1
0	1	1	1	0
0	1	0	1	1

→

1	1	2	2	3
1	2	2	?	?
1	2	?	?	??

03. 세계 암기대회

$mp(i,j)$: (i,j)까지 가면서 얻을 수 있는 최고 점수
$p(i,j)$: (i,j)에 있는 점수 (0 또는 1)

$$mp(i,j) = \begin{cases} p(i,j) & \text{if } i=0, j=0 \\ mp(i,j-1)+p(i,j) & \text{elif } i=0, j>0 \\ mp(i-1,j)+p(i,j) & \text{elif } i>0, j=0 \\ \min(mp(i-1,j), mp(i,j-1), \mathrm{mp}(\mathrm{i}-1,\mathrm{j}-1)) + p(i,j) & \text{elif } i>0, j>0 \end{cases}$$

03. 세계 암기대회

```python
t = int(input())
for _ in range(t):
    n,m = map(int,input().split())
    data = []
    for i in range(n):
        data.append(list(map(int,input().split())))
    #T[i][j]: (0,0)에서 (i,j)에 도달했을 때 얻을 수 있는 최대 점수
    T = [[0]*m for i in range(n)]
    for i in range(n):
        for j in range(m):
            if i == 0 and j == 0:
                T[i][j] = data[i][j] #시작 칸인 경우 : 자기자신
            elif i == 0: #제일 위쪽 줄인 경우 왼쪽에서밖에 올 수 없다
                T[i][j] = T[i][j-1] + data[i][j]
            elif j == 0: #제일 왼쪽 줄인 경우 위쪽에서밖에 올 수 없다
                T[i][j] = T[i-1][j] + data[i][j]
            else: #그 외의 점화식
                T[i][j] = min(T[i][j-1], T[i-1][j], T[i-1][j-1])+ data[i][j]
    print(T[n-1][m-1]) #답의 위치
```

Project 01. 최대 합 부분 연속 수열 2

리스트가 주어지면, 적당한 구간을 잡습니다.

그 구간들 전부 중에서, 구간 내 원소들의 합이 가장 큰 것을 찾는 프로그램을 작성합니다.

1	-2	3	-4	5	-3	8	-9	22

합: 5+(-3)+8+(-9)=23

Project 01. 최대 합 부분 연속 수열 2

1. 리스트의 크기가 1인 경우에는 원소가 하나 뿐이므로 그 원소의 값이 최대 합
2. 리스트의 크기가 2 이상인 경우, 리스트를 왼쪽 반과 오른쪽 반 두 개로 나누어 각각의 최대 합을 구한다
3. 리스트의 왼쪽과 오른쪽을 걸친 부분 연속 수열의 최대 합을 구한다.

1, 2는 재귀 함수를 통해 구현 **분할정복**

3번은 리스트를 가운데에서 왼쪽/가운데에서 오른쪽으로 하나씩 탐색하며 구한다.

다이나믹 프로그래밍

1	[illegible]	3	-4	5	-3	[illegible]	1	2	-5	5	-9	22

1	-2	3	-4	5	-3	8	1	2	-5	5	-9	22

1	-2	3	-4	5	-3	8	1	2	-5	5	-9	22

1	-2	3	-4	5	-3	8	1	2	-5	5	-9	22

Project 01. 최대 합 부분 연속 수열 2

리스트 L

1	-2	3	-4	5	-3	8	-9	22

1. 부분 문제를 명확하게 정의합니다.

DP[i] = i 번째 원소를 마지막으로 포함하는 최대 합 연속 구간

i = 0 부터 list 크기 -1 까지

Project 01. 최대 합 부분 연속 수열 2

리스트 L

1	-2	3	-4	5	-3	8	-9	22

DP[4] = 5

1. 부분 문제를 명확하게 정의합니다.

DP[i] = i 번째 원소를 마지막으로 포함하는 최대 합 연속 구간

i = 0 부터 list 크기 -1 까지

Project 01. 최대 합 부분 연속 수열 2

리스트 L

1	-2	3	-4	5	-3	8	-9	22

DP[6] = 10

1. 부분 문제를 명확하게 정의합니다.

DP[i] = i 번째 원소를 마지막으로 포함하는 최대 합 연속 구간

i = 0 부터 list 크기 -1 까지

Project 01. 최대 합 부분 연속 수열 2

리스트 L

1	-2	3	-4	5	-3	8	-9	22

1. 부분 문제를 명확하게 정의합니다.

DP[i] = i 번째 원소를 마지막으로 포함하는 최대 합 연속 구간

i = 0 부터 list 크기 -1 까지

2. 원래 답의 위치: $max_i DP[i]$

PIAI 인공지능연구원

POSTECH
POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Project 01. 최대 합 부분 연속 수열 2

리스트 L

1	-2	3	-4	5	-3	8	-9	22

1. 부분 문제를 명확하게 정의합니다.

DP[i] = i 번째 원소를 마지막으로 포함하는 최대 합 연속 구간

i = 0 부터 list 크기 -1 까지

2. 원래 답의 위치: $max_i DP[i]$

3. 재귀 식 (점화 식): $DP[i] = max\,(\,DP[i-1] + L[i], L[i]\,)$

i-1 번째를 구간에 포함시킬 것인지, 제외할 것인지

Project 01. 최대 합 부분 연속 수열 2

리스트 L

1	-2	3	-4	5	-3	8	-9	22

1. 부분 문제를 명확하게 정의합니다.

DP[i] = i 번째 원소를 마지막으로 포함하는 최대 합 연속 구간

i = 0 부터 list 크기 -1 까지

2. 원래 답의 위치: $max_i DP[i]$

3. 재귀 식 (점화 식): $DP[i] = max\,(\,DP[i-1] + L[i], L[i]\,)$

i-1 번째를 구간에 포함시킬 것인지, 제외할 것인지

4. 기저 조건 (초기값): $DP[0] \;=\; L[0]$