12724

접근법

> - 가장 작은 수를 만들어야 함으로 두 배열 a,b 배열에 대하여 a 배열에서 가장 작은 수, b 배열에서 가장 큰 수를 곱하는 식으로 하는 것이 가장 효율적임을 자명한다.
> - 두 배열을 만들고 그 배열들을 하나는 오름차순, 하나는 내림차순으로 하고 곱해주면 됨

In [ ]:

```
N = int(input())

for i in range(N):
    result = 0
    input()
    str1 = input().split()
    str2 = input().split()
    int1=[]
    int2=[]
    for str_to_int1, str_to_int2 in zip(str1, str2):
        int1.append(int(str_to_int1))
        int2.append(int(str_to_int2))
    int1.sort()
    int2.sort(reverse=True)

    for k,j in zip(int1, int2):
        result += k*j

    print("Case #{}: {}".format(i+1,result))
```

15990

접근법

> - 최대 n이 100,000이다. 이때 가장 작은 숫자들의 조합으로 만들어질 경우는 1과 2가 반복적으로 나와서 100,000까지 증가하는 경우이다. 이 경우, 100,000/3 * 2로하면 약 66,667인데 이게 하나의 조합일 뿐 조합이 엄청 많아진다. 즉, 시간 초과가 뜰거라고 기대하고 연습삼아 완전 탐색 코드를 먼저 짜봤다.

In [ ]:

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

```
N = int(input())

one_to_three = [1,2,3]

def result(number, target, cur_score):
    res = 0
    for i in one_to_three:
        if i == number:
            continue
        if (cur_score + i > target):
            return 0
        if (cur_score + i == target):
            return res+1
        res += result(i, target, cur_score+i)
    return res

for _ in range(N):
    res = 0
    target = int(input())
    cur_score = 0
    res = result(0, target, cur_score)
    print(res%1000000009)
```

런타임 에러가 나는데 이유가 뭘까?! 뭘까용??? 뭘까나??? 까나리 액젓!?
백준에서는 런타임 에러에 대해서 자세히 알려주지 않기 때문에 특정 코드를 **ideone**에 들어가서 확인하면 에러 찾을 수 있다고 한다!

접근법 2

> - 너무나도 당연하게도 특정 값을 만드는 방법이 정해져 있는데 중복이 될 때마다 새로 다시 다 구하는 중복이 존재 한다, 예를 들어 1을 사용하였고 2를 만드는 방법은 1개임을 알 수 있는데 다음 번에 새로 들어올때 또 이걸 다 확인해보는 멍청한 짓을 한다 컴터는 빠가야로!
> - 위의 문제를 고치기 위해서 DP를 사용 하였다. 이때 특정 수 i를 만드는 조합을 그냥 넣는게 아니라 i를 만들 때 이전에 사용된 값을 제외하고 만들어야 하기 때문에 그것 까지 고려하여, 총 3 X 100,000 배열을 만들고 그전에 온 값이 1이라면 첫번째 row에 해당하는 i에 대하여 이미 값이 존재하는지 존재하지 않는지를 확인하였다.

In [ ]:

```
N = int(input())

cache = [[-1 for _ in range(100000)] for _ in range(3)]
cache[0][0] = 0
cache[0][1] = 1
cache[0][2] = 2
cache[1][0] = 1
cache[1][1] = 0
```

```
cache[1][2] = 2
cache[2][0] = 1
cache[2][1] = 1
cache[2][2] = 2


def f(picked, cur):
    if cache[picked-1][cur-1] != -1:
        return cache[picked-1][cur-1]
    ret = 0
    for i in range(1,4):
        temp = 0
        if i == picked:
            continue
        temp += f(i, cur-i)
        cache[i-1][cur-i-1] = temp
        ret += temp
    return ret


for _ in range(N):
    ret = 0
    target = int(input())
    cur_score = 0
    for pick in range(1,4):
        ret += f(pick, target-pick)
    print(ret%1000000009)
```

 이것도 런타임 에러가 난다! 왜일까용?!?!?!
 maximum recursion depth exceeded in comparison 이 에러가 뜬다. 재귀를 너무 많이 했다는 건데 고쳐봐야징

관련 정보를 찾다가 input보다 빠른 python 함수를 알았다!!

> input()은 꽤나 느린 입력 함수이니 stdin.readline()을 쓰세요. 사용법은 똑같습니다.

In [1]:

```
import sys
t = sys.stdin.readline() # input이랑 똑같다고 한다 확인을 해보자! 근데 split이랑은 없는 것 같다! 안쓸란다 나는
```

접근법 3 (성공)

> - 재귀를 하려니깐 너무 깊어진다고 안된다고 해서 처음부터 그냥 값들을 저장하기 위해 한번 반복하고 그 뒤에 알맞는 값들을 뽑아오는 걸 사용하였다

- 이를 위해 값들을 미리 모두 받고 그 중 가장 큰 값보다 1 작은 만큼의 정보들은 모두 만들어 놓았다!
- 이 방법을 사용할 시, 첫 값이 1, 2, 3일 경우 에러가 나서 따로 처리해줘야 한다 (이거 찾는다고 한 세월 다 보냈다.)

In [ ]:

```
N = int(input())

targets = []
for _ in range(N):
    targets.append(int(input())-1)

max_target = max(targets)

cache = [[-1 for _ in range(100000)] for _ in range(3)]
cache[0][0] = 0
cache[0][1] = 1
cache[0][2] = 2
cache[1][0] = 1
cache[1][1] = 0
cache[1][2] = 2
cache[2][0] = 1
cache[2][1] = 1
cache[2][2] = 2

yirun = [1,1,3] # 접근법 3에 3번째 bulle스의 내용이다

ii = 2
while ii < max_target:
        ii += 1
        cache[0][ii] = (cache[1][ii-2] + cache[2][ii-3]) % 1000000009
        cache[1][ii] = (cache[0][ii-1] + cache[2][ii-3]) % 1000000009
        cache[2][ii] = (cache[0][ii-1] + cache[1][ii-2]) % 1000000009

for target in targets:
    if target<=2:
        ret = yirun[target]
        print(ret)
        continue
    ret = 0
    ret += cache[0][target-1] + cache[1][target-2] + cache[2][target-3]

    print(ret%1000000009)
```

1546

접근법

> - 필요합니까? 그냥 계산 때리세요

In [ ]:

```
N = int(input())

scores = input().split()

grades = []

for score in scores:
    grades.append(int(score))

max_score=max(grades)
avg = 0
for i, score in enumerate(scores):
    grades[i] = float(grade/max_score*100)
    avg+= grades[i]

print(avg/N)
```