25-1 이니로
알고리즘 멘토링

멘토 - 김수성

돌 게임 4 / 9658

백준 9658 / https://www.acmicpc.net/problem/9658

문제

돌 게임은 두 명이서 즐기는 재밌는 게임이다.

탁자 위에 돌 N개가 있다. 상근이와 창영이는 턴을 번갈아가면서 돌을 가져가며, 돌은 1개, 3개 또는 4개 가져갈 수 있다. 마지막 돌을 가져가는 사람이 게임을 지게 된다.

두 사람이 완벽하게 게임을 했을 때, 이기는 사람을 구하는 프로그램을 작성하시오. 게임은 상근이가 먼저 시작한다.

입력

첫째 줄에 N이 주어진다. ($1 \le N \le 1000$)

출력

상근이가 게임을 이기면 SK를, 창영이가 게임을 이기면 CY을 출력한다.

돌 게임 4 / 9658

처음에 상근이가 3개를 가져 감→ 3개가 남음
두번째에 창영이는 무조건 1개를 가져가야 함 -> 2개가 남음
세번째에 상근이가 1개를 가져 감 -> 1개가 남음
마지막으로 창영이가 1개를 가져가서 상근이가 이기게 됨

출력

상근이가 게임을 이기면 SK를, 창영이가 게임을 이기면 CY을 출력한다.

예제 입력 1 복사

```
6
```

예제 출력 1 복사

```
SK
```

돌 게임 4 / 9658

DP[i][j] = 돌이 i개 남았고, j의 차례일 때 이기는 사람

j = 0이면 상근이의 차례, j = 1이면 창영이의 차례
또한 DP의 값이 0이면 상근이가 이기고,
DP의 값이 1이면 창영이가 이김

돌 게임 4 / 9658

j = 0이면 상근이의 차례, j = 1이면 창영이의 차례
또한 DP의 값이 0이면 상근이가 이기고,
DP의 값이 1이면 창영이가 이김

일단 Base Case를 확인해보자

돌 게임 4 / 9658

j = 0이면 상근이의 차례, j = 1이면 창영이의 차례
또한 DP의 값이 0이면 상근이가 이기고,
DP의 값이 1이면 창영이가 이김

DP[0][0] = 0
돌이 없는데 상근이의 차례면 창영이가 마지막 돌을 가져갔으므로
상근이가 이김

DP[0][1] = 1
돌이 없는데 창영이의 차례면 상근이가 마지막 돌을 가져갔으므로
창영이가 이김

돌 게임 4 / 9658

현재 상근이의 차례이면 돌을 가져갔을 때
본인이 이기는 경우의 수가 있으면 이길 수 있고 없으면 이길 수 없음
또한 돌을 가져가면 차례가 바뀌기 때문에 DP의 인덱스도 바뀜

DP[i][0] = MIN(DP[i - 1][1], DP[i - 3][1], DP[i - 4][1])

돌 게임 4 / 9658

창영이의 차례에도 마찬가지로
본인이 이기는 경우의 수가 있으면 이길 수 있고 없으면 이길 수 없음
또한 돌을 가져가면 차례가 바뀌기 때문에 DP의 인덱스도 바뀜

DP[i][1] = MAX(DP[i - 1][0], DP[i - 3][0], DP[i - 4][0])

돌 게임 4 / 9658

점화식은 다음과 같음

DP[0][0] = 0 DP[0][1] = 1
DP[i][0] = MIN(DP[I – 1][1], DP[I – 3][1], DP[I – 4][1])
DP[i][1] = MAX(DP[I – 1][0], DP[I – 3][0], DP[I – 4][0])

i < 0 일 때 예외 처리를 해줘야 함

돌 게임 4 / 9658

C++

```cpp
#include <iostream>
using namespace std;
using ll = long long;

const ll MAX = 1010;
ll n, dp[MAX][2];
ll diff[3] = {-1, -3, -4};

ll solve(ll cur, ll turn){
    ll& ret = dp[cur][turn];
    if(ret != -1) return ret; ret = turn ^ 1;

    for(int i = 0;i < 3;i++){
        ll nxt = cur + diff[i];
        if(nxt < 0) break; // 돌은 음수가 될 수 없음
        if(turn) ret = max(ret, solve(nxt, turn ^ 1));
        else ret = min(ret, solve(nxt, turn ^ 1));
    }

    return ret;
}

int main(){
    ios::sync_with_stdio(0); // fastio
    cin.tie(0), cout.tie(0); // fastio

    cin >> n;
    for(int i = 0;i < MAX;i++){
        for(int j = 0;j < 2;j++) dp[i][j] = -1;
    }

    dp[0][0] = 0; dp[0][1] = 1;
    cout << (solve(n, 0) ? "CY" : "SK");

    return 0;
}
```

돌 게임 4 / 9658

Python

```python
import sys
input = sys.stdin.readline
sys.setrecursionlimit(10**6)

dp = [[-1] * 2 for _ in range(1010)]
n = int(input().rstrip())

dp[0][0] = 0
dp[0][1] = 1

def solve(cur, turn):
    if dp[cur][turn] != -1:
        return dp[cur][turn]
    dp[cur][turn] = turn ^ 1

    for i in [-1, -3, -4]:
        nxt = cur + i
        # 돌은 음수가 될 수 없음
        if nxt < 0:
            break

        if turn == 1:
            dp[cur][turn] = max(dp[cur][turn], solve(nxt, turn ^ 1))
        else:
            dp[cur][turn] = min(dp[cur][turn], solve(nxt, turn ^ 1))

    return dp[cur][turn]

print("CY" if solve(n, 0) == 1 else "SK")
```

질문?

가장 긴 증가하는 부분 수열 / 11053

백준 11053 / https://www.acmicpc.net/problem/11053

문제

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

예를 들어, 수열 A = {10, 20, 10, 30, 20, 50} 인 경우에 가장 긴 증가하는 부분 수열은 A = {**10**, **20**, 10, **30**, 20, **50**} 이고, 길이는 4이다.

입력

첫째 줄에 수열 A의 크기 N (1 ≤ N ≤ 1,000)이 주어진다.

둘째 줄에는 수열 A를 이루고 있는 A_i가 주어진다. (1 ≤ A_i ≤ 1,000)

출력

첫째 줄에 수열 A의 가장 긴 증가하는 부분 수열의 길이를 출력한다.

가장 긴 증가하는 부분 수열 / 11053

가장 긴 증가하는 부분 수열의 길이를 출력하는 문제
아래에서는 (10 20 10 30 20 50) 으로 정답이 4

출력

첫째 줄에 수열 A의 가장 긴 증가하는 부분 수열의 길이를 출력한다.

예제 입력 1 복사

```
6
10 20 10 30 20 50
```

예제 출력 1 복사

```
4
```

가장 긴 증가하는 부분 수열 / 11053

인덱스 i를 마지막으로 포함하는 수열을 생각해보자

그러면 i 이전의 인덱스의 값이 A[i]보다 작은 어떤 j를
마지막으로 포함하는 수열에 A[i]를 추가해서 만들 수 있음

EX) 10 20 10 30 20 50
-> 10 50
-> 20 50
-> 30 50

가장 긴 증가하는 부분 수열 / 11053

인덱스 i를 마지막으로 포함하는 수열을 생각해보자

그러면 i 이전의 인덱스의 값이 A[i]보다 작은 어떤 j를
마지막으로 포함하는 수열에 A[i]를 추가해서 만들 수 있음

DP[i] = 수열의 마지막이 A[i]인 수열 중 최대 길이

j < i && A[j] < A[i] 인 j에 대해 A[i]를 붙여서 수열을 만들 수 있음
-> DP[i] = DP[j] + 1

가장 긴 증가하는 부분 수열 / 11053

DP[i] = 수열의 마지막이 A[i]인 수열 중 최대 길이

j < i && A[j] < A[i] 인 j에 대해 A[i]를 붙여서 수열을 만들 수 있음
-> DP[i] = DP[j] + 1

Base Case는 DP[0] = 0으로 설정 하면 됨

가장 긴 증가하는 부분 수열 / 11053

C++

```cpp
const ll MAX = 1010;
ll n, a[MAX], dp[MAX];

ll solve(ll cur){
    ll& ret = dp[cur];
    if(ret != -1) return ret; ret = 0;

    for(int i = 0;i < cur;i++){
        // 이전의 값이 크거나 같으면 증가하는 부분 수열이 아님
        if(a[i] >= a[cur]) continue;

        // 이전의 수열 길이에 +1
        ret = max(ret, solve(i) + 1);
    }

    return ret;
}

int main(){
    ios::sync_with_stdio(0); // fastio
    cin.tie(0), cout.tie(0); // fastio

    cin >> n;
    for(int i = 1;i <= n;i++) cin >> a[i];
    for(int i = 0;i < MAX;i++) dp[i] = -1;

    ll result = 0; dp[0] = 0;
    //1 ~ n의 DP값 중 최댓값이 정답
    for(int i = 1;i <= n;i++) result = max(result, solve(i));
    cout << result;

    return 0;
}
```

가장 긴 증가하는 부분 수열 / 11053

Python

```python
MAX = 1010
n = int(input())
a = [0] + list(map(int, input().split()))
dp = [-1] * MAX

def solve(cur):
    if dp[cur] != -1:
        return dp[cur]
    dp[cur] = 0

    for i in range(cur):
        # 이전의 값이 크거나 같으면 증가하는 부분 수열이 아님
        if a[i] >= a[cur]:
            continue
        # 이전의 수열 길이에 +1
        dp[cur] = max(dp[cur], solve(i) + 1)

    return dp[cur]

result = 0
dp[0] = 0

# 1 ~ n의 DP값 중 최댓값이 정답
for i in range(1, n + 1):
    result = max(result, solve(i))

print(result)
```

질문?

쉬운 계단 수 / 10844

백준 10844 / https://www.acmicpc.net/problem/10844

문제

45656이란 수를 보자.

이 수는 인접한 모든 자리의 차이가 1이다. 이런 수를 계단 수라고 한다.

N이 주어질 때, 길이가 N인 계단 수가 총 몇 개 있는지 구해보자. 0으로 시작하는 수는 계단수가 아니다.

입력

첫째 줄에 N이 주어진다. N은 1보다 크거나 같고, 100보다 작거나 같은 자연수이다.

출력

첫째 줄에 정답을 1,000,000,000으로 나눈 나머지를 출력한다.

쉬운 계단 수 / 10844

길이가 1 인 계단 수 1 2 3 4 5 6 7 8 9
길이가 2 인 계단 수 10 21 32 43 54 65 76 87 98
12 23 34 45 56 67 78

예제 입력 1 복사

```
1
```

예제 출력 1 복사

```
9
```

예제 입력 2 복사

```
2
```

예제 출력 2 복사

```
17
```

쉬운 계단 수 / 10844

길이가 1 인 계단 수 1 2 3 4 5 6 7 8 9

길이가 2 인 계단 수 10 21 32 43 54 65 76 87 98
12 23 34 45 56 67 78

계단 수의 마지막 수만 고려해보자
계단 수의 현재 수는 이전의 수와 차이가 1이여야 함

쉬운 계단 수 / 10844

계단 수의 마지막 수만 고려해보자
계단 수의 현재 수는 이전의 수와 차이가 1이여야 함

DP[i][j] = 길이가 i이고, 마지막 수가 j일 때 계단 수의 개수
DP[i][j]는 DP[i - 1][j - 1], DP[i - 1][j + 1]에서 전이 함

쉬운 계단 수 / 10844

DP[i][j] = 길이가 i이고, 마지막 수가 j일 때 계단 수의 개수
DP[i][j]는 DP[i – 1][j – 1], DP[i – 1][j + 1]에서 전이 함

Base Case

i가 1일 때 계단 수는 0으로 시작 할 수 없음
DP[1][j] = 0 (j = 0)
DP[1][j] = 1 (j > 0 && j < 9)

쉬운 계단 수 / 10844

DP[i][j] = 길이가 i이고, 마지막 수가 j일 때 계단 수의 개수
DP[i][j]는 DP[i – 1][j – 1], DP[i – 1][j + 1]에서 전이 함

DP[i][j]는 DP[i – 1][j – 1], DP[i – 1][j + 1]의 계단 수에서
j를 붙여 만들 수 있음
-> DP[i][j] = DP[i – 1][j – 1] + DP[i – 1][j + 1]

j < 0인 경우와 j > 9인 경우는 0으로 처리 해줘야 함
값이 커질 수 있으므로 1e9로 나누기 처리도 해줘야 함

쉬운 계단 수 / 10844

C++

```cpp
int main(){
    ios::sync_with_stdio(0); // fastio
    cin.tie(0), cout.tie(0); // fastio

    ll n; cin >> n;
    for(int i = 0;i < MAX;i++){
        for(int j = 0;j <= 9;j++) dp[i][j] = -1;
    }

    // 0 ~ 9의 합이 정답
    for(int i = 0;i <= 9;i++){
        result += solve(n, i);
        result %= MOD;
    }

    cout << result;
    return 0;
}
```

```cpp
const ll MAX = 101;
const ll MOD = 1e9;
ll n, dp[MAX][10], result;

ll solve(ll cur, ll num){
    // 수가 음수거나 한 자리 수가 아니면 0 반환
    if(num > 9 || num < 0) return 0;
    ll& ret = dp[cur][num];
    if(ret != -1) return ret; ret = 0;

    // 0으로 시작하는 계단 수는 계단 수가 아님
    if(cur == 1) return ret = num ? 1 : 0;

    // 이전의 수와 현재 수가 한개 차이가 나야 함
    ret += solve(cur - 1, num + 1) % MOD;
    ret += solve(cur - 1, num - 1) % MOD;

    return ret %= MOD;
}
```

쉬운 계단 수 / 10844

Python

```python
MOD = 10**9
dp = [[-1] * 10 for _ in range(1010)]

def solve(cur, num):
    # 수가 음수거나 한 자리 수가 아니면 0 반환
    if num < 0 or num > 9:
        return 0

    if dp[cur][num] != -1:
        return dp[cur][num]

    # 0으로 시작하는 계단 수는 계단 수가 아님
    if cur == 1:
        return 1 if num != 0 else 0

    # 이전의 수와 현재 수가 한개 차이가 나야 함
    dp[cur][num] = solve(cur - 1, num - 1) + solve(cur - 1, num + 1)
    dp[cur][num] %= MOD

    return dp[cur][num]

n = int(input().rstrip())

result = 0
# 0 ~ 9의 합이 정답
for i in range(10):
    result += solve(n, i)
    result %= MOD

print(result)
```

질문?

6주차 – 배낭 문제 (Knapsack)

배낭 문제

배낭 문제

N개의 물건과 물건을 담을 수 있는 배낭이 존재
각 물건은 무게와 가치가 존재 함

배낭에는 넣을 수 있는 한도가 정해져 있음
이 때 배낭에 넣을 수 있는 물건의 가치를 최대화 하는 문제

보통 물건을 쪼갤 수 없고
무조건 물건을 넣거나 넣지 않는 선택만 할 수 있음
0 - 1 Knapsack Problem

배낭 문제

배낭 문제

보통 물건을 쪼갤 수 없고
무조건 물건을 넣거나 넣지 않는 선택만 할 수 있음
0 - 1 Knapsack Problem

물건을 쪼갤 수 있으면 무게 대비 가치가 큰 물건을
그리디하게 쪼개서 먼저 넣으면 답이 나옴

물건을 쪼갤 수 없다면?

배낭 문제

배낭 문제

보통 물건을 쪼갤 수 없고
무조건 물건을 넣거나 넣지 않는 선택만 할 수 있음
0 - 1 Knapsack Problem

물건을 쪼갤 수 있으면 무게 대비 가치가 큰 물건을
그리디하게 쪼개서 먼저 넣으면 답이 나옴

물건을 쪼갤 수 없다면?

배낭 문제

배낭 문제

물건을 쪼갤 수 없고,
무조건 물건을 넣거나 넣지 않는 선택만 할 수 있음
0 - 1 Knapsack Problem

물건을 쪼갤 수 있으면 무게 대비 가치가 큰 물건을
그리디하게 쪼개서 먼저 넣으면 답이 나옴

물건을 쪼갤 수 없다면?

평범한 배낭 / 12865

백준 12865 / https://www.acmicpc.net/problem/12865

문제

이 문제는 아주 평범한 배낭에 관한 문제이다.

한 달 후면 국가의 부름을 받게 되는 준서는 여행을 가려고 한다. 세상과의 단절을 슬퍼하며 최대한 즐기기 위한 여행이기 때문에, 가지고 다닐 배낭 또한 최대한 가치 있게 싸려고 한다.

준서가 여행에 필요하다고 생각하는 N개의 물건이 있다. 각 물건은 무게 W와 가치 V를 가지는데, 해당 물건을 배낭에 넣어서 가면 준서가 V만큼 즐길 수 있다. 아직 행군을 해본 적이 없는 준서는 최대 K만큼의 무게만을 넣을 수 있는 배낭만 들고 다닐 수 있다. 준서가 최대한 즐거운 여행을 하기 위해 배낭에 넣을 수 있는 물건들의 가치의 최댓값을 알려주자.

입력

첫 줄에 물품의 수 N($1 \le N \le 100$)과 준서가 버틸 수 있는 무게 K($1 \le K \le 100,000$)가 주어진다. 두 번째 줄부터 N개의 줄에 거쳐 각 물건의 무게 W($1 \le W \le 100,000$)와 해당 물건의 가치 V($0 \le V \le 1,000$)가 주어진다.

입력으로 주어지는 모든 수는 정수이다.

출력

한 줄에 배낭에 넣을 수 있는 물건들의 가치합의 최댓값을 출력한다.

평범한 배낭 / 12865

N = 4, M = 7
두 번째, 세 번째 물건을 배낭에 넣으면
7의 무게로 14의 가치를 얻을 수 있음

예제 입력 1 복사

```
4 7
6 13
4 8
3 6
5 12
```

예제 출력 1 복사

```
14
```

평범한 배낭 / 12865

DP[i][j] = i번째 물건까지 선택을 완료했고,
무게의 합이 j일 때 최대 가치의 합

각 물건은 배낭에 담거나 담지 않을 수 있음

평범한 배낭 / 12865

각 물건은 배낭에 담거나 담지 않을 수 있음

배낭에 물건을 담지 않았을 때
무게의 합과 가치의 합이 그대로임
-> DP[i][j]

배낭에 물건을 담았을 때
무게의 합과 가치의 합이 늘어남
-> DP[i][j - W[i]] + C[i]

평범한 배낭 / 12865

배낭에 물건을 담지 않았을 때
무게의 합과 가치의 합이 그대로임
-> DP[i][j]

배낭에 물건을 담았을 때
무게의 합과 가치의 합이 늘어남
-> DP[i][j – W[i]] + C[i]

DP[i + 1][j] = MAX(DP[i][j], DP[i][j – W[i]] + C[i])

평범한 배낭 / 12865

DP[i + 1][j] = MAX(DP[i][j], DP[i][j – W[i]] + C[i])
물론 j가 음수가 될 수는 없으므로 j – W[i]에서 예외 처리 해줘야 함

평범한 배낭 / 12865

C++

```cpp
const ll MAX = 101;
ll n, m, dp[MAX][101010];
ll w[MAX], c[MAX], result;

ll solve(ll cur, ll num){
    ll& ret = dp[cur][num];
    if(ret != -1) return ret; ret = 0;
    if(!cur) return ret;

    // 물건을 고르지 않음
    ret = max(ret, solve(cur - 1, num));

    // 물건을 고름
    if(num >= w[cur]) ret = max(ret, solve(cur - 1, num - w[cur]) + c[cur]);

    // 정확히 무게의 합이 num이 아닌 경우도 세야 함
    if(num) ret = max(ret, solve(cur, num - 1));

    return ret;
}

int main(){
    ios::sync_with_stdio(0); // fastio
    cin.tie(0), cout.tie(0); // fastio

    cin >> n >> m;
    for(int i = 1;i <= n;i++) cin >> w[i] >> c[i];
    for(int i = 0;i <= n;i++){
        for(int j = 0;j < 101010;j++) dp[i][j] = -1;
    }

    cout << solve(n, m);
    return 0;
}
```

평범한 배낭 / 12865

Python

```python
n, m = map(int, input().split())
w = [0] * (n + 1)
c = [0] * (n + 1)
for i in range(1, n + 1):
    w[i], c[i] = map(int, input().split())

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

def solve(cur, num):
    if dp[cur][num] != -1:
        return dp[cur][num]

    dp[cur][num] = 0
    if cur == 0:
        return dp[cur][num]

    # 물건을 고르지 않음
    dp[cur][num] = max(dp[cur][num], solve(cur - 1, num))

    # 물건을 고름
    if num >= w[cur]:
        dp[cur][num] = max(dp[cur][num], solve(cur - 1, num - w[cur]) + c[cur])

    # 정확히 무게의 합이 num이 아닌 경우도 세야 함
    if num >= 1:
        dp[cur][num] = max(dp[cur][num], dp[cur][num - 1])

    return dp[cur][num]

print(solve(n, m))
```

질문?

양팔 저울 / 2629

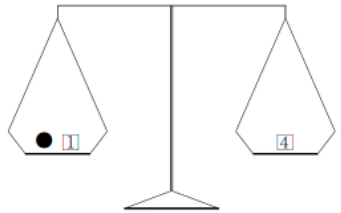
백준 2629 / https://www.acmicpc.net/problem/2629

문제

양팔 저울과 몇 개의 추가 주어졌을 때, 이를 이용하여 입력으로 주어진 구슬의 무게를 확인할 수 있는지를 결정하려고 한다.

무게가 각각 1g과 4g인 두 개의 추가 있을 경우, 주어진 구슬과 1g 추 하나를 양팔 저울의 양쪽에 각각 올려놓아 수평을 이루면 구슬의 무게는 1g이다. 또 다른 구슬이 4g인지를 확인하려면 1g 추 대신 4g 추를 올려놓으면 된다.

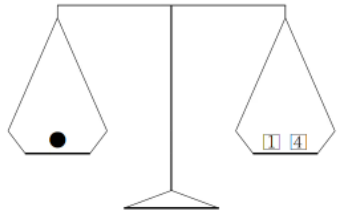
구슬이 3g인 경우 아래 <그림 1>과 같이 구슬과 추를 올려놓으면 양팔 저울이 수평을 이루게 된다. 따라서 각각 1g과 4g인 추가 하나씩 있을 경우 주어진 구슬이 3g인지도 확인해 볼 수 있다.



<그림 1> 구슬이 3g인지 확인하는 방법 (1은 1g인 추, 4는 4g인 추, ●은 무게를 확인할 구슬)

<그림 2>와 같은 방법을 사용하면 구슬이 5g인지도 확인할 수 있다. 구슬이 2g이면 주어진 추를 가지고는 확인할 수 없다.

추들의 무게와 확인할 구슬들의 무게가 입력되었을 때, 주어진 추만을 사용하여 구슬의 무게를 확인 할 수 있는지를 결정하는 프로그램을 작성하시오.



<그림 2> 구슬이 5g인지 확인하는 방법

양팔 저울 / 2629

N개의 추가 주어지고 M번 동안 구슬의 무게가 주어짐
M번 동안 N개의 추로 구슬의 무게를 구할 수 있으면 Y
구할 수 없으면 N을 출력 하면 됨

입력

첫째 줄에는 추의 개수가 자연수로 주어진다. 추의 개수는 30 이하이다. 둘째 줄에는 추의 무게들이 자연수로 가벼운 것부터 차례로 주어진다. 같은 무게의 추가 여러 개 있을 수도 있다. 추의 무게는 500g이하이며, 입력되는 무게들 사이에는 빈칸이 하나씩 있다. 세 번째 줄에는 무게를 확인하고자 하는 구슬들의 개수가 주어진다. 확인할 구슬의 개수는 7이하이다. 네 번째 줄에는 확인하고자 하는 구슬들의 무게가 자연수로 주어지며, 입력되는 무게들 사이에는 빈 칸이 하나씩 있다. 확인하고자 하는 구슬의 무게는 40,000보다 작거나 같은 자연수이다.

출력

주어진 각 구슬의 무게에 대하여 확인이 가능하면 Y, 아니면 N 을 차례로 출력한다. 출력은 한 개의 줄로 이루어지며, 각 구슬에 대한 답 사이에는 빈칸을 하나씩 둔다.

양팔 저울 / 2629

첫 번째 예제
3 = 4 - 1
2 -> 구할 수 없음

예제 입력 1 복사

```
2
1 4
2
3 2
```

예제 출력 1 복사

```
Y N
```

예제 입력 2 복사

```
4
2 3 3 3
3
1 4 10
```

예제 출력 2 복사

```
Y Y N
```

양팔 저울 / 2629

두 번째 예제

1 = 3 - 2　　4 = 3 + 3 - 2

10 -> 구할 수 없음

예제 입력 1 복사

```
2
1 4
2
3 2
```

예제 출력 1 복사

```
Y N
```

예제 입력 2 복사

```
4
2 3 3 3
3
1 4 10
```

예제 출력 2 복사

```
Y Y N
```

양팔 저울 / 2629

각 추는 사용하거나 사용하지 않을 수 있음

추를 구슬의 반대편에 배치하면 구슬 쪽의 무게가 줄어들음
추를 구슬과 같은 곳에 배치하면 구슬 쪽의 무게가 늘어남

구슬의 반대편과 구슬과 같은 곳의 무게가 0이 될 수 있으면 Y

예제 입력 2 복사

```
4
2 3 3 3
3
1 4 10
```

예제 출력 2 복사

```
Y Y N
```

양팔 저울 / 2629

DP[i][j] = i번째 추까지 선택을 완료 했을 때
구슬이 있는 곳의 무게가 반대편의 무게보다 j만큼
크게 할 수 있으면 1, 아니면 0

추를 사용하지 않음

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

추를 구슬과 같은 곳에 놓음

DP[i + 1][j] = DP[i][j + W[i]]

추를 구슬 반대 편에 놓음

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

양팔 저울 / 2629

DP[i][j] = i번째 추까지 선택을 완료 했을 때
구슬이 있는 곳의 무게가 반대편의 무게보다 j만큼
크게 할 수 있으면 1, 아니면 0

구슬 반대편의 무게가 더 클 수도 있음
-> j < 0인 경우도 있음

일반적으로 인덱스가 음수인 경우는 불가능하므로
임의의 상수를 더해서 인덱스를 취급 해주면 됨

양팔 저울 / 2629

C++

```cpp
int main(){
    ios::sync_with_stdio(0); // fastio
    cin.tie(0), cout.tie(0); // fastio

    cin >> n;
    for(int i = 1;i <= n;i++) cin >> a[i];
    cin >> m;
    for(int i = 1;i <= m;i++) cin >> b[i];

    for(int i = 0;i <= n;i++){
        for(int j = 0;j < 101010;j++) dp[i][j] = -1;
    }

    for(int i = 1;i <= m;i++){
        cout << (solve(n, b[i] + MID) ? "Y" : "N") << " ";
    }

    return 0;
}
```

```cpp
const ll MAX = 33;
const ll MID = 50101;
ll n, m, a[MAX], b[MAX];
ll dp[MAX][101010];

ll solve(ll cur, ll num){
    if(num < 0 || num > 101010) return 0;
    ll& ret = dp[cur][num];
    if(ret != -1) return ret;

    // 추를 하나도 놓지 않았으면 무게는 0 이여야 함
    if(cur == 0) return ret = (num == MID ? 1 : 0);

    // 추를 사용하지 않음
    ret = max(ret, solve(cur - 1, num));

    // 추를 구슬과 같은 곳에 놓음
    ret = max(ret, solve(cur - 1, num + a[cur]));

    // 추를 구슬 반대 편에 놓음
    ret = max(ret, solve(cur - 1, num - a[cur]));

    return ret;
}
```

양팔 저울 / 2629

Python

```python
import sys
sys.setrecursionlimit(10**6)
input = sys.stdin.readline

n = int(input().strip())
a = list(map(int, input().split()))
m = int(input().strip())
b = list(map(int, input().split()))

MAX = 33
MID = 50101

dp = [[-1] * 101010 for _ in range(MAX)]
```

```python
def solve(cur, num):
    if num < 0 or num >= 101010:
        return 0

    if dp[cur][num] != -1:
        return dp[cur][num]

    # 추를 하나도 놓지 않았으면 무게는 0 이여야 함
    if cur == -1:
        return 1 if num == MID else 0

    ret = 0

    # 추를 사용하지 않음
    ret = max(ret, solve(cur - 1, num))

    # 추를 구슬과 같은 곳에 놓음
    ret = max(ret, solve(cur - 1, num + a[cur]))

    # 추를 구슬 반대 편에 놓음
    ret = max(ret, solve(cur - 1, num - a[cur]))

    dp[cur][num] = ret
    return ret

for i in b:
    print("Y" if solve(n - 1, i + MID) else "N", end = " ")
```

질문?

1, 2, 3 더하기 4 / 15989

백준 15989 / https://www.acmicpc.net/problem/15989

문제

정수 4를 1, 2, 3의 합으로 나타내는 방법은 총 4가지가 있다. 합을 나타낼 때는 수를 1개 이상 사용해야 한다. 합을 이루고 있는 수의 순서만 다른 것은 같은 것으로 친다.

- 1+1+1+1
- 2+1+1 (1+1+2, 1+2+1)
- 2+2
- 1+3 (3+1)

정수 n이 주어졌을 때, n을 1, 2, 3의 합으로 나타내는 방법의 수를 구하는 프로그램을 작성하시오.

입력

첫째 줄에 테스트 케이스의 개수 T가 주어진다. 각 테스트 케이스는 한 줄로 이루어져 있고, 정수 n이 주어진다. n은 양수이며 10,000보다 작거나 같다.

출력

각 테스트 케이스마다, n을 1, 2, 3의 합으로 나타내는 방법의 수를 출력한다.

1, 2, 3 더하기 4 / 15989

저번주에 풀었던 문제와 비슷함
저번주에 풀었던 문제는 순서를 고려함

이 문제는 순서를 고려하지 않음

- 1+1+1+1
- 2+1+1 (1+1+2, 1+2+1)
- 2+2
- 1+3 (3+1)

예제 입력 1 복사

```
3
4
7
10
```

예제 출력 1 복사

```
4
8
14
```

1, 2, 3 더하기 4 / 15989

이 문제는 순서를 고려하지 않음
2 + 1 + 1, 1 + 1 + 2, 1 + 2 + 1은 같은 것으로 취급

- 1+1+1+1
- 2+1+1 (1+1+2, 1+2+1)
- 2+2
- 1+3 (3+1)

순서를 고려하지 않는 법
-> 순서에 제한을 둬 보자

제한을 두는 법
-> DP 상태에 추가

1, 2, 3 더하기 4 / 15989

- 1+1+1+1
- 2+1+1 (1+1+2, 1+2+1)
- 2+2
- 1+3 (3+1)

제한을 두는 법
-> DP 상태에 추가

순서를 고려할 때 문제를 푸는 법
DP[i] = DP[i – 1] + DP[i – 2] + DP[i – 3]

순서를 고려하지 않으니 제한을 둬보자
-> 숫자는 항상 오름차순으로 나와야 함

1, 2, 3 더하기 4 / 15989

순서를 고려하지 않으니 제한을 둬보자
-> 숫자는 항상 오름차순으로 나와야 함

EX) 1 + 1 + 2
1 + 1 + 2 + 2 + 3
1 + 3 + 3 + 3

1 + 2 + 1 처럼 오름차순이 아니면
올바르지 않은 것으로 간주함

- 1+1+1+1
- 2+1+1 (1+1+2, 1+2+1)
- 2+2
- 1+3 (3+1)

1, 2, 3 더하기 4 / 15989

EX) 1 + 1 + 2
1 + 1 + 2 + 2 + 3
1 + 3 + 3 + 3

- 1+1+1+1
- 2+1+1 (1+1+2, 1+2+1)
- 2+2
- 1+3 (3+1)

숫자를 오름차순으로 배치해보자

숫자 2가 나오면 더 이상 뒤에 + 1 을 붙일 수 없음
숫자 3이 나오면 더 이상 뒤에 + 1, +2 를 붙일 수 없음

1, 2, 3 더하기 4 / 15989

숫자 2가 나오면 더 이상 뒤에 + 1 을 붙일 수 없음
숫자 3이 나오면 더 이상 뒤에 + 1, +2 를 붙일 수 없음

숫자를 몇 까지 사용했는지 알아야 함
-> DP 상태에 추가함

- 1+1+1+1
- 2+1+1 (1+1+2, 1+2+1)
- 2+2
- 1+3 (3+1)

1, 2, 3 더하기 4 / 15989

- 1+1+1+1
- 2+1+1 (1+1+2, 1+2+1)
- 2+2
- 1+3 (3+1)

숫자를 몇 까지 사용했는지 알아야 함
-> DP 상태에 추가함

원래 점화식
DP[i] = i를 1, 2, 3의 합으로 표현한 경우의 수

바뀐 점화식
DP[i][j] = 마지막 숫자가 j일 때 i를 1, 2, 3의 합으로 표현한
오름차순 경우의 수

1, 2, 3 더하기 4 / 15989

바뀐 점화식
DP[i][j] = 마지막 숫자가 j일 때 i를 1, 2, 3의 합으로 표현한
오름차순 경우의 수

DP[i][1] = 마지막 숫자가 1
DP[i][1] = DP[i – 1][1]

DP[i][2] = 마지막 숫자가 2
DP[i][2] = DP[i – 2][1] + DP[i – 2][2]

- 1+1+1+1
- 2+1+1 (1+1+2, 1+2+1)
- 2+2
- 1+3 (3+1)

1, 2, 3 더하기 4 / 15989

바뀐 점화식
DP[i][j] = 마지막 숫자가 j일 때 i를 1, 2, 3의 합으로 표현한
오름차순 경우의 수

DP[i][3] = 마지막 숫자가 3
DP[i][3] = DP[i – 3][1] + DP[i – 3][2] + DP[i – 3][3]

1, 2, 3 더하기 4 / 15989

DP[i][j] = 마지막 숫자가 j일 때 i를 1, 2, 3의 합으로 표현한
오름차순 경우의 수

DP[i][1] = DP[i – 1][1]
DP[i][2] = DP[i – 2][1] + DP[i – 2][2]
DP[i][3] = DP[i – 3][1] + DP[i – 3][2] + DP[i – 3][3]

Base Case
DP[0][1] = 1
DP[0][2] = DP[0][3] = 0

1, 2, 3 더하기 4 / 15989

Base Case

DP[0][1] = 1

DP[0][2] = DP[0][3] = 0

DP[3][3]을 구할 때
DP[3][3] = DP[0][1] + DP[0][2] + DP[0][3] 으로 식이 나옴
DP[0]의 값이 모두 1이면 3이 나와서 틀림

1, 2, 3 더하기 4 / 15989

C++

```cpp
const ll MAX = 10101;
ll dp[MAX][4];

ll solve(ll cur, ll mx){
    if(cur < 0) return 0;
    ll& ret = dp[cur][mx];
    if(ret != -1) return ret; ret = 0;
    if(!cur) return ret = (mx == 1);

    for(int i = 1;i <= mx;i++) ret += solve(cur - mx, i);
    return ret;
}

void run(){
    ll n; cin >> n;
    ll result = 0;
    for(int i = 1;i <= 3;i++) result += solve(n, i);
    cout << result << "\n";
}

int main(){
    ios::sync_with_stdio(0); // fastio
    cin.tie(0), cout.tie(0); // fastio

    for(int i = 0;i < MAX;i++){
        for(int j = 0;j < 4;j++) dp[i][j] = -1;
    }

    ll t; cin >> t;
    while(t--) run();

    return 0;
}
```

1, 2, 3 더하기 4 / 15989

Python

```python
import sys
sys.setrecursionlimit(10**6)
input = sys.stdin.readline

MAX = 10101
dp = [[-1] * 4 for _ in range(MAX)]
```

```python
def run():
    n = int(input().strip())
    result = 0
    for i in range(1, 4):
        result += solve(n, i)
    print(result)

t = int(input().strip())
for _ in range(t):
    run()
```

```python
def solve(cur, mx):
    if cur < 0:
        return 0

    ret = dp[cur][mx]
    if ret != -1:
        return ret

    ret = 0
    if cur == 0:
        dp[cur][mx] = 1 if mx == 1 else 0
        return dp[cur][mx]

    for i in range(1, mx + 1):
        ret += solve(cur - mx, i)

    dp[cur][mx] = ret
    return ret
```

질문?

기본 과제

평범한 배낭 – https://www.acmicpc.net/problem/12865

양팔저울 - https://www.acmicpc.net/problem/2629

1, 2, 3 더하기 4 – https://www.acmicpc.net/problem/15989

동전 2 - https://www.acmicpc.net/problem/2294

가장 긴 바이토닉 부분 수열 - https://www.acmicpc.net/problem/11054

내리막 길 – https://www.acmicpc.net/problem/1520

고생하셨습니다